PhD Dissertation

International Doctorate School in Information and Communication Technologies

DISI - University of Trento

Exploiting SAT and SMT Techniques for Automated Reasoning and Ontology Manipulation in Description Logics

Michele Vescovi

Advisor:

Prof. Roberto Sebastiani

Università degli Studi di Trento

February 25th, 2011

Abstract

The quest of efficient and scalable reasoning procedures arising from the area of Description Logics and its notable applications in the prominent domains of Semantic Web and bio-medical ontologies, on one hand, and the wide variety of mature and efficient technologies offered by the SAT research area, on the other hand, motivated our research. In this thesis we explore the idea of exploiting the power and efficiency of state-of-the-art SAT-based and SMT-based techniques for automated reasoning and ontology manipulation in Description Logics, proposing a valid alternative to the traditional tableau based algorithms.

We propose and develop novel and complete approaches able to solve Description Logic problems as SAT and SMT (Satisfiability Modulo Theories) ones. With this aim we define sound and complete encodings, and we develop new procedures and optimizations techniques based on a variety of existing SAT-based formalisms and technologies.

In this work, in particular, we focus on three reasoning problems which tackle increasingly more expressive logics or increasingly harder reasoning services, among which we face also non-standard services supporting the debugging of ontologies, like modularization and axiom pinpointing. We implemented our approaches in tools which integrate with the available SAT/SMT-solvers; finally, we show the effectiveness of our novel approaches through very extensive empirical evaluations on benchmarks and ontologies from real applications, in which we compare our performance against the other state-of-the-art available reasoners.

Notice that any advance in the integrated Boolean reasoning techniques and tools will be freely exploited from our novel proposed approaches (in contrast with possible advances in tableau-based algorithms, which presuppose new implementations), extending also to Description Logics and to the emerging fieeld of ontologies the benefits of the observable great and fast advance in the efficiency of these techniques.

Keywords
automated reasoning, knowledge representation, description logics, ontologies, SAT, SMT.

Contents

"Proving that I am right would be recognizing that I could be wrong"
[P. A. Beaumarchais, *The Marriage of Figaro , act I, scene I*, 1778]

"Prouver que j'ai raison serait accorder que je puis avoir tort"
[P. A. Beaumarchais, *Le Mariage de Figaro, acte I, scne I*, 1778]

"Dimostrare che ho ragione significherebbe ammettere che potrei avere torto"
[P.A. Beaumarchais, *Le Nozze di Figaro, atto I, scena I*, 1778]

Chapter 1

Introduction

In this thesis dissertation we explore the idea of exploiting the power and efficiency of state-of-the-art SAT-based techniques for the automated reasoning and ontology manipulation in Description Logics (DLs).

In the last two decades, the problem of automated reasoning in Description Logics has been throughly investigated and DL systems have been employed with success in various application domains. However, this problem has gained further importance since the advent of Semantic Web and the explosion of new applications in the field of ontologies, for which DLs play an important role as the foundation of the web ontology languages.

Actual DL reasoning techniques, however, often lack of efficiency in handling some particular features or prominent reasoning services. We think that the impressive advance in the practical efficiency of SAT/SMT techniques that we have continuously witnessed in the last twenty years, and the wide variety of specific reasoning technologies offered by those two areas can be profitably transferred and exploited for met the quest of efficient procedure arising from the fields of automated reasoning and ontology manipulation in Description Logics.

1.1 Trends in Description Logic and SAT-based Techniques

In our work we combine two different areas of automated reasoning: Description Logics and ontologies are the application domains, while the Boolean-reasoning techniques offered by the SAT/SMT research areas represent the baseline of our work. We briefly introduce the trends, the prominent problems, the applications and the available state-of-the-art-techniques techniques motivating our research.

Reasoning in Description Logic: Applications, Problems and Trends

Description Logics (DLs) (Baader, Calvanese, McGuinness, Nardi, & Patel-Schneider, 2003) are a family of logic-based knowledge representation formalisms aimed at representing the knowledge of an application domain in a structured way, by defining the relevant concepts of the domain and, then, by using these concepts to specify properties

of objects and individuals occurring in the domain. A main characteristic of Description Logics is the emphasis on reasoning; these languages are indeed equipped with formal, logic-based semantics and they provide a set of reasoning services by mean of whom implicitly represented knowledge can be inferred from the explicitly defined one.

Further than in knowledge representation and management systems, Description Logics are employed with success in various application domains, ranging from *natural language processing* to *databases* and, moreover, recently, to *bio-medical ontologies* and in the *Semantic Web*. In this latest field, nowadays, the Description Logic-based language OWL (Bechhofer et al., 2004) has been adopted as the standard ontology language, and currently the second standard OWL 2 (Motik, Patel-Schneider, & Parsia, 2009) has been defined.

In Description Logic, many different constructors have been proposed as a formalism to describe the concepts of the domain, the objects and the relations among them. Every distinct combination of these language constructors might yield to a new Description Logic. For years, researchers have studied the effect of combining these constructors on the complexity and the properties of the different Description Logics obtained. The result of this research is a really wide range of logics which goes from very simple logics with a low expressive power, to very powerful logics where, in contrast, the reasoning is really complex (sometimes undecidable for almost the reasoning services).

A DL knowledge base is generally formed by two components. The first one is the *terminological component* (TBox), informally, the *schema* of the knowledge base; the second one is the *assertional component* (ABox), informally, an instantiation of the terminological schema. The complexity of reasoning on such a knowledge base depends not only on the set of language constructors provided by the implemented Description Logic, but also on the use of one or both the components in the reasoning and from the structure of the axioms/assertions by which these components are defined. If it is allowed to have set of axioms defining cyclic terminologies, for example, logics become harder to reason on and smarter algorithms are required.

For this latest reason, notationally, Description Logics are distinguished by the language constructors they provide and determining their expressive power. Historically, the most studied Description Logic is $\mathcal{ALC}$, that is a notational variant of the Modal Logic K_m (Schild, 1991). The $\mathcal{SH}$ family of languages is another notable class of Description Logics ($\mathcal{S}$ is an abbreviation for $\mathcal{ALC}$ with transitive roles and $\mathcal{H}$ represents the use of role hierarchies). In fact it is the base on which many harder languages have been defined to reach the real-world quest of expressive logics. For example, all the OWL languages belongs to the $\mathcal{SH}$ family.

Recently another family of tractable Description Logics called $\mathcal{EL}$ (which includes also the $\mathcal{EL}$ extensions $\mathcal{EL}^+$ and $\mathcal{EL}^{++}$), has caught the attention of the researchers (Baader, Brandt, & Lutz, 2005; Baader, Peñaloza, & Suntisrivaraporn, 2007). In fact, even though these logics are really less expressive with respect to other logics, they are expressive enough to describe several important bio-medical ontologies such as SNOMED-CT (Spackman, 2000; Baader & Suntisrivaraporn, 2008), the Gene Ontology (The G. O. Consor-

tium, 2000), the National Cancer Institute (NCI) ontology (Sioutos, de Coronado, Haber, Hartel, Shaiu, & Wright, 2007), and the majority of Galen (Rector & Horrocks, 1997). Reasoning on these ontologies represents not only a notable application of Description Logic-based systems, but also a challenge due to the required efficiency and the huge dimensions of this kind of problems.

A relatively-large number of tools are the byproduct of this research. The most notable Description Logic reasoners at the state-of-the-art are FaCT++, Pellet, HermiT and Racer. They all implement the analytic tableau method, except for HermiT which is based on the hypertableaux calculus (Motik, Shearer, & Horrocks, 2009), and they are able to manage really complex logics (including OWL —DL-based— documents).

Among the others, some notable examples of standard reasoning services in Description Logic are: *knowledge-base consistency*, *concept satisfiability*, *concept subsumption* (i.e. to determine if every instance of a specified concept is instance of another specified concept), that is a subcase of the *classification* of all the concepts, *instance checking* (i.e. the problem of deciding if an object is instance of a specified concept) and the *retrieval* of all the objects instance of the specified concept. Anyway, many of these inferences can be reduced from and to each other, i.e. they can be solved as special cases of other inferences. Moreover, many other non-standard or supplemental reasoning services have been defined. Among the others we mention *axiom pinpointing* (e.g. Baader et al., 2007), which allows for debugging ontologies, and *modularization* (e.g. Baader & Suntisrivaraporn, 2008), which has the aim of extracting (as small as possible) subsets of the knowledge base preserving a specified statement or some properties of interest.

For its nature, however, reasoning in Description Logic is a very hard problem (NP-complete, PSpace-complete, ExpTime-complete and even more) (Baader et al., 2003). For this reason, nowadays, the research in Description Logic is proceeding in the space delimited by two main directions: (i) the study of even harder but decidable logics to establish the theoretical boundaries of the field; (ii) the search of easy logics, expressive enough to cover the needs of some practical applications, that often manage simple problems of huge dimensions. In the middle of these two orthogonal research directions, the quest of efficient procedures coming from the many practical applications in the field of ontologies is currently guiding the research on some main streams, among which there are: (iii) the identification of tractable or "less hard" fragments (like Horn fragments) of harder and frequently used logics; (iv) the design of efficient optimization techniques and algorithms which can help in handling efficiently practical problems for worst-case-really-hard logics or services; (v) the definition and analysis of novel non-standard or supplemental reasoning services which utility arises from the fields of ontology manipulation and Semantic Web (e.g. *modularization*, *logical difference* computation, identification of *laconic* or *precise modules* or *justifications*, and so on and so forth) and the design of efficient procedures for handling them.

Thus, due to its theoretical complexity and to the increasing applications of Description Logics in many research fields and practical domains, the development of efficient reasoning algorithms and procedures in Description Logic has become a key research issue.

Advances in SAT, SAT-based Approaches and Beyond

Propositional Satisfiability (SAT) (Biere, Marijn, van Maaren, & Walsh, 2009) is the problem of determining whether a Boolean formula admits at least one satisfying truth assignment to its variables. It is a core problem in mathematical logic and computer theory because it is, for its nature and historical reasons, the most representative NP-complete problem. Moreover, in the last two decades we have witnessed an impressive advance in the efficiency of SAT techniques, which has brought large previously-intractable problems at the reach of state-of-the-art solvers (see, e.g., Zhang & Malik, 2002). Nowadays freely-available SAT solvers (the most efficient and exploited algorithms are based on DPLL in its many variants) can manage problems of hundreds of millions of clauses and variables. As a consequence, many hard real-world problems have been successfully solved by encoding them into SAT.

The research of efficient SAT techniques is still a crucial point, which is of interest for many areas and other domains in computer science like Automated Reasoning, Model Checking, Formal Methods. In fact many SAT-based techniques have witnessed extremely effective in other domains and especially in the area of formal verification. Starting from Planning (Kautz, McAllester, & Selman, 1996) and Bounded LTL Model Checking (BMC) (Biere, Cimatti, Clarke, & Zhu, 1999), many other problems have been encoded into SAT or solved with the aid of SAT tools (e.g., Armando, Castellini, Giunchiglia, Giunchiglia, & Tacchella, 2005; Sebastiani, 2007b) and these approaches are currently state-of-the-art in the respective communities. For example, Model Checking techniques exploiting the power of state-of-the-art SAT solvers have proved to be a valid alternative, and very often superior, to the traditional approaches based on Binary Decision Diagrams (BDDs) (e.g. McMillan, 2003).

Further, the SAT formalism is used to solve problems with a wider extent. Besides returning an assignment (a model) when a formula is satisfiable, also the *all solutions* problem (called *All-SAT*) (see, e.g., Jin, Han, & Somenzi, 2005) for satisfiable formulas, the *unsatisfiable core* problem (called *unsat-core*) (see, e.g., Zhang & Malik, 2003) and the problem of find *interpolants* (McMillan, 2003) for unsatisfiable formulas, are meaningful problems which are attracting a lot of attention from the research community, due to the various applications especially as part of many SAT-based Model Checking algorithms (e.g McMillan, 2003).

The progress in Boolean Satisfiability (SAT) solving techniques, together with the concrete needs from real applications, have inspired significant research on richer and more expressive Boolean formalism like Quantified Boolean Formula (QBF) (see, e.g., Plaisted, Biere, & Zhu, 2003; Giunchiglia, Narizzano, & Tacchella, 2006) and Satisfiability Modulo Theories (SMT) (see Sebastiani, 2007b for an overview). The formalism of plain Propositional Logic (SAT), in fact, is often not suitable or expressive enough for representing many real-world problems. Such problems are more naturally expressible using QBF or SMT. In particular SMT can be seen as an extension of SAT in which the input formula is expressed in (a subset of) first-order logic (typically without quantifiers) with respect to a background theory (for example: *linear arithmetic* both over the *reals* and the *integers*,

its subclass *difference logic*, the theories of *bit vectors*, of *arrays* and of *lists*, and others). Among the others, notable examples of SMT-based approaches are in the field of Model Checking, in the verification of RTL designs and of systems with an infinite number of states like real-time and hybrid control systems, and software (many examples can be found in (Sebastiani, 2007b; Barrett, Sebastiani, Seshia, & Tinelli, 2009)).

The dominating approach for SMT, which underlies most state-of-the-art tools, is based on the integration of a SAT solver and one or more domain-specific solvers for the background theories. The SAT solver enumerates truth assignments which satisfy the Boolean abstraction of the input formula (where distinct theory-specific subformulas are represented/abstracted by distinct Boolean atoms), whilst the domain-specific solvers check the consistency in the respective background theory of the set of literals corresponding to the assignments enumerated. This approach is called *lazy*, in contraposition to the *eager* approach, consisting in encoding an SMT formula into an equivalently satisfiable Boolean formula, and on solving the result with a SAT solver (see, e.g., (Sebastiani, 2007b) for a survey).

Although SMT is still a novel research area, it is also a very active one: new solvers and techniques are continuously proposed, and often with improvements of orders of magnitude in performance with respect to the previous approaches. In particular, since it is based on many of the SAT techniques previously introduced by the research community, some SMT-based approaches and other SMT-based tools for problems like interpolants or the fast recognition of small unsatisfiable cores, have been proposed in the last few years (Cimatti, Griggio, & Sebastiani, 2008, 2007), but they are still at a very preliminary stage, and far from being as mature as SAT-based ones.

1.2 Motivations, Objectives and Methodology

The research trends in Description Logic, its manifold practical applications, the quest of efficient and scalable procedures, on one hand, the wide variety of mature and efficient techniques offered by the SAT research area, on the other hand, motivated our research.

Besides the traditional applications (Baader et al., 2003), Description Logics are assuming notable relevance due to the extension of their applications even in "hotter" domains, especially in the field of the Semantic Web and in many other research fields connected to the use of ontologies. Because of the orthogonal needs — on one hand, for expressive Description Logics supporting complex knowledge representation systems or ontologies and, on the other hand, for easy and tractable Description Logics supporting simple structured ontologies of huge dimensions — the problem of finding efficient reasoning procedures in Description Logic has become crucial. SAT-based technologies, in the meanwhile, proved to be mature and largely successful in many other automated reasoning fields, first of all in the case of the very hard (and often also huge) problems arising from the practical applications of formal verification. Further, the more SAT is a simple and intuitive problem, the more it is adaptable and suitable for the encoding of different formalisms. SMT, instead, can be a very powerful formalism combining the

adaptability of SAT with the expressivity of the many embedded theories. Thus we believe that the state-of-the-art Boolean reasoning techniques that we can borrow from the successful research area of SAT and from the rising area of SMT, can be in practice powerful and suitable "tools" able to meet the current quest of efficiency from the prominent real applications of huge ontologies and hard Description Logics.

Our central objective is to develop new techniques for reasoning in Description Logics and on real ontologies, through the definition of novel encodings and procedures able to solve Description Logic problems as SAT (or even SMT) ones. We aim at exploiting the capabilities of the state-of-the-art Boolean reasoning techniques proposing a convenient alternative to the traditional tableau based algorithms. Notice that any advance in the exploited Boolean reasoning techniques/tools will be freely inherited from our novel proposed approaches (in contrast with possible advances in tableau-based algorithms, which presuppose new implementations), extending also to Description Logics (and thus, by consequence, to the emerging fields of ontologies) the benefits of the observable great and fast advance in the efficiency of these techniques. With this main objective in mind we also aim at trying the applicability of the larger possible number of SAT-based technologies and formalisms (choosing the more suitable one) and at approaching a wide range of problems among the different Description Logic languages and reasoning services.

Based on these ideas and motivations, our work explored the efficiency of encoding different reasoning services in various Description Logics to SAT and SMT, by way of the following steps: (i) the definition of a formal (and as efficient as possible) encoding proving its soundness and completeness, (ii) the implementation of a tool which realizes the encoding and (iii) the evaluation of the integration of our tool together with the SAT/SMT tools on benchmarks or ontologies relative to real applications, comparing the performance with the other state-of-the-art approaches and available tools. In particular we started from a well-known problem and, then, we moved to increasingly harder ones by facing more expressive logics and by approaching non-standard reasoning services. Moreover, we realized the above exposed steps in an incremental way. First, we devised, implemented and evaluated the basic encoding in order to preliminarily check the potentials of each novel approach. Second, we pushed the performance and the range of problems at the reach of our approach, by introducing optimizations or by exploiting supplemental SAT-based techniques. We always analyzed the effects of these enhancements through very extensive empirical test sessions.

1.3 Research Directions and Faced Problems

Our central objective is to improve the state-of-the-art of the reasoning procedures into the prominent application area of Description Logics and ontology manipulation, by proposing new techniques built on the mature and largely successful SAT-based technologies.

In the Description Logic research area a relatively large number of reasoners is available, which already show very interesting performances on expressive Description Logics. They handle a large variety of logics and all offer the most common and studied inference

services. Thus, the proposal of novel techniques and new reasoners must take into account of the directions and features in which the currently available techniques are still lacking or in which the existing general-purpose and highly-optimized systems can be better replaced by specific tools. In particular we individuated three currently "hot" topics in which the benefits gained by SAT-based techniques can be profitably extended to the automated reasoning in Description Logics:

- the efficient manipulation of huge real ontologies, thanks to the attitude of SAT in handling huge size problems;
- the development of new techniques for optimized reasoning with numerical constraints, in which the SMT technology can be helpful;
- the design of efficient procedures for the emerging non-standard reasoning services by mean of the many existent supplemental SAT techniques (like all-SAT and others).

In our research, with these intents, we invested in three different directions, iteratively reproducing the steps of the methodology exposed in the previous section:

1. Start from relatively-easy Description Logics and standard/well-studied reasoning services by mean of a direct encoding into SAT, in order to first evaluate the feasibility and potentials of our idea and outline the applicability boundaries of our approach.
2. With the experience acquired from the previous direction, explore the opportunity of threat strongest ontologies and reasoning in the harder Description Logics, possibly by way of more expressive Boolean reasoning formalisms like SMT. In particular, we investigate the integration of SMT especially for what concerns those logics which provide language constructors that are somewhat similar to those of the theories that SMT includes.
3. Face even more complex and "non-standard" reasoning services with the aid, when necessary, of other SAT tools/techniques (like, e.g., *unsat-core* or *All-SAT*), and evaluate the response and scalability of our approach in the concrete applications of huge real ontologies.

In order to explore all the three above exposed research directions we chose to approach, in particular, three gradually harder significant representative problems:

Concept satisfiability in $\mathcal{ALC}$ (with empty TBox) via SAT. We start our investigation focusing on plain (i.e. with empty TBox) concept satisfiability in the core Description Logic $\mathcal{ALC}$, by mean of a direct encoding into plain Propositional Logic satisfiability. While $\mathcal{ALC}$ is, historically, the most studied Description Logic, concept satisfiability is the core reasoning service in DL knowledge bases (subsumption, instance checking and other services are likewise important but, often, they can be reduced each other with satisfiability, or solved with simple modifications).

Concept satisfiability wrt. empty TBoxes in $\mathcal{ALC}$ is a PSPACE-complete problem. We started exploring our idea from this problem because we want to test our new approach on a central and standard problem in Description Logic, so that well established benchmark problems are available, and so that we can compare our results against the widest possible set of different approaches. In fact, due to its equivalency with the core Modal Logic K_m, a rich variety of reasoning tools for simple concept satisfiability in $\mathcal{ALC}$ is available, including also (but not only) tools based on tableau, BDDs and QBF techniques.

Concept satisfiability in $\mathcal{ALCQ}$ (with acyclic TBox) via SMT. Among the many Description Logics, $\mathcal{ALC}$ is a relatively-easy formalism. Moving to more expressive logics, we chose to explore the use of SMT in order to handle qualified number restrictions. Thus we faced the problem of concept satisfiability in $\mathcal{ALCQ}$ (Faddoul, Farsinia, Haarslev, & Möller, 2008) wrt. acyclic TBoxes. Even if, from a purely theoretical perspective, it is still a PSPACE-complete reasoning task, [1] wrt. the previously approached problem we include two other main sources of complexity: the presence of TBoxes and the presence of numerical constraints.

Reasoning with qualified number restriction is a prominent research issue in Description Logic (see, e.g., Faddoul & Haarslev, 2010), in fact the current techniques often lacks of efficiency, especially when the number of the restrictions is higher or when the values involved in the restrictions their selves are big. For this reason ontology designers most likely avoid the use of these constructors, even if they are very natural (sometimes essential) in many domains. Thus the quest of efficient procedures for handling qualified number restrictions is not only an important issue for the automated reasoning in Description Logic, but it has also particularly important consequences for the development of the ontology-design area.

We encoded $\mathcal{ALCQ}$-concept satisfiability in SMT modulo the Theory of Costs (Cimatti, Franzén, Griggio, Sebastiani, & Stenico, 2010), that we think naturally fits the expressivity of numerical restrictions. The Theory of Costs, in fact, is a subset of linear arithmetic over the integer, in which it is possible to define multiple cost variables/functions and define both increases and lower/upper-bounds on such costs.

Axiom Pinpointing in $\mathcal{EL}^+$ (with general TBox) via Horn-SAT, C.A., all-SMT. We conclude this thesis by approaching the problem of efficient and scalable non-standard reasoning on the huge real bio-medical ontologies. In this third case we solve the problem of axiom pinpointing in the logic $\mathcal{EL}^+$ wrt. general TBoxes (Baader et al., 2007).

A very prominent research area in Description Logic concerns the representation and manipulation of bio-medical ontologies. Even if they are often very simple in

[1] Concept satisfiability in $\mathcal{ALCQ}$ wrt. general TBoxes, instead, is EXPTIME-complete. However we chose to handle only acyclic TBoxes not to avoid to switch to the upper class of complexity but to concentrate only on the encoding, postponing the issue of introducing techniques like blocking to handle cyclicity.

the structure, the reasoning on these ontologies is a challenging task due to their huge dimensions. The $\mathcal{EL}$ family of lightweight (i.e. less expressive but tractable) Description Logics (including $\mathcal{EL}^+$) has been defined as a response to the quests coming from the numerous concrete applications of these ontologies (Baader et al., 2005).

Among the different possible reasoning services, we chose the *axiom pinpointing* problem (Baader et al., 2007) because it is really useful in concrete applications. In more details, the axiom pinpointing problem consists in finding one or many minimal subsets of a DL knowledge base that have consequence on some inferred properties; so it can be used, e.g., in order to find one or many sets of axioms (called MinAs) which cause an unwanted subsumption relation. In other words, axiom pinpointing allows for debugging ontologies. In $\mathcal{EL}^+$ finding one MinA is a polynomial problem, while finding all the MinAs for a given subsumption relation is worst-case output exponential.

The identification of all the minimal subsets of axioms causing an undesired inference requires an iterative process. Therefore, we investigated the use of the *All-SAT*/*All-SMT* (Jin et al., 2005; Lahiri, Nieuwenhuis, & Oliveras, 2006) techniques applied in the framework of our proposed novel approach based on Boolean reasoning techniques. In a nutshell, the idea is to build off-line a polynomial-size Horn propositional formula encoding the full classification of the input ontology. Then, we can find one MinA by applying *SAT under assumption* and by exploiting *Conflict Analysis* looking for a minimal set of assumptions falsifying the encoded formula and the negation of the queried subsumption. Finally, we use *All-SMT*in order to uniquely enumerate all the single MinAs computed in such a way. In this research stream, in order to increase the performance of our approach, we also deal with the supplemental reasoning problem of modularization (see, e.g., Baader & Suntisrivaraporn, 2008).

1.4 Dissertation Outline

Here we provide a content outline of the present dissertation, and we also give account of either published or recently submitted publications.

The dissertation is divided in two main parts. The first part, consisting of Chapters 2, 3 and 4, briefly describes the state-of-the-art and gives the necessary background for the rest of the dissertation. The second part, consisting of Chapters 5, 6 and 7, presents the original contributions of this thesis by describing the theoretical, technical and practical results achieved in three different prominent problems of automated reasoning in Description Logics and ontologies.

In the following we give more details on the parts/chapters of the present dissertation:

Part I: Preliminaries. In this part we give the necessary theoretical background for the reading of this thesis. In particular, our work combines two different areas of automated reasoning: Description Logics and ontologies are the application domains,

while the Boolean-reasoning techniques offered by the SAT/SMT research areas represent the baseline of our work. For this reason we don't go into too much details in discussing the state-of-the-art and describing the background for these two areas; instead, we just provide the indispensable notions for both of them and, especially, for the specific cases of study on which we give our innovative contributions. Moreover, since we approach three problems which has been approached by considerably different methods, we chose to separately discuss in the relative chapters of novel contributions the related works and previous approaches for every handled problem.

- **Chapter 2: Brief on the State of the art.** In this chapter we look from two different perspective at the state of the art in the two areas of automated reasoning we are combine in this work. On the one hand we analyze the trends and the prominent problems in automated reasoning in Description Logics, in which we give our new contributions. On the other hand we analyze the technologies and procedures at the state-of-the-art offered in the area of Boolean-reasoning, which are the techniques we exploit to perform efficient reasoning on the tackled problems.
- **Chapter 3: Description Logics.** We introduce Description Logics. We present the notation, the semantic and the main reasoning services provided, both the standard/traditional and the non-standard/emerging ones. Then we present the specific logics which are the cases of study of our dissertation.
- **Chapter 4: SAT-based Techniques.** We provide the main notions concerning SAT and SMT, with the focus on the techniques and the reasoning procedures offered by these two areas. In particular we present Conflict-Driven Clause-Learning SAT solving, SAT solving under assumptions, SMT modulo the Theory of Costs and the All-SAT/All-SMT technique.

Part II: Original Contributions. Here we present the original contributions of this thesis. In this part we divide in three distinct chapters the contributions in the three specific and increasingly harder problems we have chosen as representative of the three main research directions given in Section 1.3. For the sake of uniformity, we structured every chapter following the applied methodology, which is exposed in Section 1.2. In general, for each chapter: (i) we discuss the specific reasoning problem, the related works and the previous approaches; (ii) we explain the motivations which have led us to choose the treated problem, the SAT-based techniques we have select to handle it, and the goals of our approach; (iii) we define a basic encoding of the problem into the chosen SAT-based formalism, proving the soundness and completeness of the encoding (we also optionally report the results of some preliminary evaluation); (iv) we develop and describe the optimizations and enhancements introduced to further push the performances of our approach; (vi) we always conclude with an extensive empirical evaluation, in which we present the implemented tool and the competitor systems, we describe the benchmark problems and we discuss the results, highlighting the innovative contributions of our approach.

– **Chapter 5: Encoding $\mathcal{ALC}/K_m$-satisfiability into SAT.** We tackle the problem of concept satisfiability in $\mathcal{ALC}$ wrt. empty TBoxes by encoding it into SAT. In order to have a wider set of competitors approaches and well-established benchmark problems we present and solve the equivalent problem of modal K_m-satisfiability. In this chapter we present K_m2SAT, defining a basic encoding and several preprocessing and on-the-fly optimizations; we finally compare with a very large set of state-of-the-art approaches for modal K_m-satisfiability.

  [Sebastiani & Vescovi, 2006] Roberto Sebastiani and Michele Vescovi. Encoding the Satisfiability of Modal and Description Logics into SAT: the Case Study of $K(m)/\mathcal{ALC}$. In: *Proceedings of the 9th International Conference on Theory and Applications of Satisfiability Testing*. (SAT'06) Seattle, USA. 11-15 August 2006. (vol. 4121 of LNCS, pp. 130-135, Springer).

  [Sebastiani & Vescovi, 2009a] Roberto Sebastiani and Michele Vescovi. Automated Reasoning in Modal and Description Logics via SAT Encoding: the Case Study of $K(m)/\mathcal{ALC}$-Satisfiability. In: *Journal of Artificial Intelligence Research*. (JAIR) June 2009. (vol. 35, num. 1, pp. 343-389, AAAI Press).

– **Chapter 6: Handling Number Restrictions as SMT Problems.** We face the problem of concept satisfiability in $\mathcal{ALCQ}$ wrt. acyclic TBoxes, introducing in that way also reasoning on qualified number restrictions and TBox reasoning. In this chapter we present our approach $\mathcal{ALCQ}2SMT_{\mathcal{C}}$ which encodes $\mathcal{ALCQ}$ concept satisfiability into SMT modulo the Theory of Costs (SMT($\mathcal{C}$)). We define the basic encoding and a very effective partitioning optimization aiming at reducing the search space. We finally compare our method against the main available state-of-the-art reasoner.

  [Haarslev, Sebastiani, & Vescovi, 2011] Volker Haarslev, Roberto Sebastiani and Michele Vescovi. Automated Reasoning in $\mathcal{ALCQ}$ via SMT. Submitted to: *23nd International Conference on Automated Deduction*. (CADE-23) Wrocaw, Poland. 31 July – 5 August 2011.

– **Chapter 7: Exhaustively Debugging $\mathcal{EL}^+$ TBoxes via Horn-SAT and All-SMT.** We solve the problem of axiom pinpointing in the lightweight Description Logic $\mathcal{EL}^+$ via encoding into Horn-SAT and exploiting Conflict Analysis. By extending our approach into the All-SMT framework, we show how we manage to exhaustively debug huge $\mathcal{EL}^+$ ontologies. We describe our new method called $\mathcal{EL}^+$2SAT and we define the encoding, the debugging procedure and several optimizations. In particular, we also deal with modularization, for which we propose an extremely efficient and precise fully SAT-based method. We conclude by extensively testing our approach on some real-world bio-medical ontologies (among which SNOMED-CT) and by comparing with the other $\mathcal{EL}^+$-specific tool at the state-of-the-art.

  [Sebastiani & Vescovi, 2009b] Roberto Sebastiani and Michele Vescovi. Axiom Pinpointing in Lightweight Description Logics via Horn-SAT Encoding and

Conflict Analysis. In: *Proceedings of the 22nd International Conference on Automated Deduction*. (CADE-22) Montreal, Canada. 2-7 August 2009. (vol. 5663 of LNCS, pp. 84-89, Springer).

[Sebastiani & Vescovi, 2011] Roberto Sebastiani and Michele Vescovi. Efficiently Debugging $\mathcal{EL}^+$ Ontologies via SAT and SMT techniques. Under submission to: *Journal of Artificial Intelligence Research*. (JAIR).

Part I

Preliminaries

Chapter 2

Brief on the State of the Art

2.1 Reasoning in Description Logic and Handling Ontologies

Description Logics (Baader et al., 2003) are a family of logic-based knowledge representation formalism aimed to represent the knowledge of an application domain in a structured way.

A DL knowledge base is generally formed by two components. The first one is the *terminological component* (TBox). It represents the *intensional knowledge* of the domain, i.e, informally, the *schema* of the DL knowledge base; it defines the relevant *concepts* of the domain by means of a set of axioms which introduces the names of the concepts, their definitions, and the (binary) relations among them (*roles*). The second one is the *assertional component* (ABox). It represents the *extensional knowledge* stored in the DL knowledge base, that is, informally, an instantiation of the terminological schema. In this component concepts and roles can be instantiated (trough assertions) by individual names (objects).

The complexity of reasoning on such a knowledge base depends not only from the language constructors provided by the implemented Description Logic, but also from the use of one or both the components in the reasoning and from the structure of the axiom/assertions by which these components are defined. For example, it can be allowed or not for set of axioms defining cyclic terminologies, choice that yields harder logics and requires smarter algorithms.

Inferences. The purpose of a DL knowledge representation system goes beyond storing concept definitions and assertions regarding objects. Indeed, the role of this kind of system becomes relevant when the well-founded formal semantic and structure of the language is exploited to perform specific kinds of reasoning. The different kinds of reasoning performed by a DL system are defined as logical inferences (Baader et al., 2003). Currently, at the state-of-the-art, all the effective inference methods are based on analytic tableau. Without going into too much details, some important inference problems are:

- *Concept satisfiability* (does the concept represent a non-empty part of the do-

main?).

- *Subsumption* (is a given concept a "more general case" of another concept?) is a key step to create a classification of all the concepts described in the DL knowledge base.
- *Instance checking* (is a given object an instance of a specific concept?).
- *Knowledge-base consistency* (is the knowledge base consistent –i.e. meaningful at all–?).
- *Retrieval* (which are all the objects in the domain instance of a given concept?).
- *Realization* (in a set of concepts, which is the concept that better represents –i.e. the most specific concept– a given object?).
- *Equivalence* (are two concepts equivalent –i.e., do they represent the same part of the domain– with respect to the defined terminology?) and *disjointness* (are two concepts disjoint –i.e., do they represent a common part of the domain or not– with respect to the defined terminology?).
- *Axiom pinpointing* is the problem of finding a minimal (or minimum) subset of a given knowledge base that have a given consequence (for example the set of axioms that lead a concept to be subsumed by another concept) (Baader & Peñaloza, 2007).

Anyway, many of these inferences can be reduced each other, i.e. they can be solved as special cases of other inferences.

Expressive power. Description Logics are distinguished by the language constructors they provide. The set of language constructors used determines the expressive power of the logic.

Historically, the language $\mathcal{AL}$ (*attributive language*) has been chosen as the basic description language, since it seems to provide the minimal constructors useful for practical interests. $\mathcal{AL}$ allows for: atomic negation of concepts, concepts intersection, universal restriction and limited existential quantification on concepts. Other Description Logics are obtained extending $\mathcal{AL}$ with some extra constructors. These other logics are denoted by a string in which every letter represents one constructor which extends the $\mathcal{AL}$ capabilities. Other studied constructors are: complex concept negation ($\mathcal{C}$), concepts union ($\mathcal{U}$), full existential quantification ($\mathcal{E}$), cardinality restrictions on roles quantification ($\mathcal{N}$), qualified cardinality restrictions ($\mathcal{Q}$), nominals - i.e. enumerated classes of object value restrictions - ($\mathcal{O}$), role hierarchies ($\mathcal{H}$), role disjointness and limited complex role inclusion axioms ($\mathcal{R}$), inverse roles ($\mathcal{I}$), functional roles ($\mathcal{F}$), use of data-type roles, data values or data types ($^{(\mathcal{D})}$). Not all the languages obtained from the combination of these capabilities are distinct. For example, the language $\mathcal{ALCUE}$ is equivalent to the well known logic $\mathcal{ALC}$ since the combination of union and full existential quantification can be expressed using negation, and vice versa.

Moreover, theoretically and in many real application, the definition of transitive roles (usually, denoted with the use of the symbol $^+$) has been identified as an important requirement. This capability provides a useful additional expressive power to the languages but, as a drawback, it increases the complexity of reasoning requiring special *blocking* techniques to ensure the termination properties of the algorithms (Baader, 1991; Horrocks & Sattler, 1999; Baader et al., 2003).

Notable Description Logics. The most studied Description Logic is $\mathcal{ALC}$, that is the logic $\mathcal{AL}$ extended with the union of concepts, complex negation of concepts and unrestricted existential quantification. $\mathcal{ALC}$ is a notational variant of the Modal Logic $K(m)$ (Schild, 1991).

The $\mathcal{SH}$ family of languages is another notable class of Description Logics. In fact it is the base on which many harder languages has been defined to reach the real world quest of expressive logics. For example, all the OWL languages belongs to the $\mathcal{SH}$ family (the logic underlying OWL DL is $\mathcal{SHOIN}^{(\mathcal{D})}$ whilst OWL 2 is based on $\mathcal{SHROIQ}^{(\mathcal{D})}$). Note that $\mathcal{S}$ is an abbreviation for $\mathcal{ALC}$ with transitive roles and that $\mathcal{H}$ represents the use of role hierarchies.

Nevertheless, many simpler logics have been intensively studied by the DL community. In particular the logics $\mathcal{FL}_0$ and $\mathcal{FL}^-$ (Baader et al., 2003) are important for historical reasons and as a first attempt of find easy tractable (in the sense of the complexity of reasoning) Description Logics. In particular, $\mathcal{FL}^-$ is the sublanguage of $\mathcal{AL}$ obtained disallowing atomic negation, whereas $\mathcal{FL}_0$ is the sublanguage of $\mathcal{FL}^-$ obtained disallowing also the limited existential quantification.

Recently another easy Description Logic, called $\mathcal{EL}$, has catch the attention of the researchers (Baader et al., 2005, 2007). This easy logic allows for concepts conjunction and for existential restrictions instead of allowing for value restrictions as its counterpart $\mathcal{FL}_0$. This gives to $\mathcal{EL}$ better algorithmic properties with respect to $\mathcal{FL}_0$. For example, whereas subsumption in $\mathcal{FL}_0$ is PSPACE-complete, in $\mathcal{EL}$ it is polynomial, also allowing for general concept inclusion axioms (the same holds for the $\mathcal{EL}$ extensions $\mathcal{EL}^+$, which provides general role inclusions by mean of whom it is possible to represent transitive roles, and $\mathcal{EL}^{++}$, which adds concept/role assertions and concrete domains).

Bio-medical ontologies. As stated above, recently a considerable effort have been spent by the research community studying the easy Description Logics $\mathcal{EL}$, $\mathcal{EL}^+$ and $\mathcal{EL}^{++}$. Even though these logics are really less expressive with respect to other logics like those of the $\mathcal{AL}$ family, they are enough expressive to describe several important bio-medical ontologies such as SNOMED-CT (Spackman, 2000), NCI (Sioutos et al., 2007), the Gene Ontology (The G. O. Consortium, 2000), and the majority of Galen (Rector & Horrocks, 1997). Thus, reasoning on these ontologies represents not only a notable application of DL-based systems but also a challenge due to the required efficiency and the huge dimensions of this kind of problems.

We present in more details the mentioned ontologies (more information can be found in the work of Suntisrivaraporn, 2009, from which we have extracted the following information):

SNOMED-CT: The Systematized Nomenclature of Medicine, Clinical Terms (SNOMED-CT) (Schulz, Suntisrivaraporn, & Baader, 2007), is a large standardized clinical terminology adopted by health care sectors in several countries. It is a comprehensive clinical and medical ontology that covers concepts from many domains, among which: anatomy, diseases, pharmaceutical products, medical procedures and others. Previously known as SNOMED-RT (Reference Terminology) (Spackman, Campbell, & Cote, 1997; Spackman, 2000) it has reached the comprehensive form when merged with the Clinical Terms Version 3 ontology. In 2007, the International Health Terminology Standards Development Organization (IHTSDO) has been founded with the aim of to internationalize and to promote SNOMED-CT as the standard reference clinical terminology among the affiliate countries.

GALEN: The GALEN ontology (Rector & Horrocks, 1997) is the product of the homonymous European project that was launched in the 1992 with the objective of facilitate the interaction of medical information systems by mean of a common reference model for medical terminologies. It has firstly been translated into the DL format by Horrocks as benchmark problem for its reasoner FACT (Horrocks, 1998). A fine-tuned Description Logic for NOT-GALEN is $\mathcal{ELHIF}^+$, that is $\mathcal{EL}^+$ enriched with role inverses and functionalities. However, the large majority of GALEN can be represented in $\mathcal{EL}^+$, in particular two $\mathcal{EL}^+$-based variants of GALEN: NOT-GALEN and FULL-GALEN, have been widely used in benchmarking DL-reasoners.

NCI: The NCI thesaurus (Sioutos et al., 2007) is a large ontology about classification of cancers, developed by the US National Cancer Institute (NCI). Though containing several domain and range restrictions, the structure of this ontology is very simple.

GENEONTOLOGY: The Gene Ontology (The G. O. Consortium, 2000) is a controlled vocabulary that describes gene and gene product attributes. Like many others (including the previous NCI) it is included in the repository of the Open Biomedical Ontologies (OBO), which is a large library of ontologies from the bio-medical domains. The Gene Ontology is very simple in structure (apart from the definition of one transitive role it purely relies on concept definitions) and it is expressible in tractable extensions of the logic $\mathcal{EL}$.

OWL. The OWL Web Ontology Language (Horrocks, Patel-Schneider, & van Harmelen, 2003,Bechhofer et al., 2004) is a new formal language for representing ontologies in the Semantic Web; moreover, nowadays, it has become the standard language in the field. OWL has been developed by the World Wide Web Consortium and is largely

based on Description Logic. In particular OWL has three increasingly-expressive sublanguages: OWL Lite, OWL DL and OWL Full.

OWL Lite: supports classification hierarchy and simple constraints, for example cardinality constraints with cardinality only of 0 or 1. It is based on the logic $\mathcal{SHIF}^{(\mathcal{D})}$ (which has ExpTime complexity). However, it has been rarely used and thus now it results obsolete.

OWL DL: it supports the maximum possible expressiveness while retaining computational completeness and decidability. It includes all the OWL language constructors, but they can be used only under certain restrictions which make OWL DL related to the Description Logic $\mathcal{SHOIN}^{(\mathcal{D})}$ (which has NExpTime complexity).

OWL Full: maximize the expressiveness and the syntactic freedom (extending RDF) having fewer constraints on use, but without computational guarantees. For these reasons it operates outside the boundaries of Description Logic, whereas all reasoning task in OWL Lite or OWL DL can be reduced to DL knowledge base satisfiability in the respective logic.

OWL 2. Since October 2009 the OWL 2 Web Ontology Language (Motik et al., 2009), is the second W3C recommendation as ontology language for the Semantic Web, representing a substantial revision of the first standard OWL. Some of the new features brought into OWL 2 are only syntactic sugar, while others gain new expressivity to the language. In terms of Description Logic, OWL 2 has been defined as an extension of the logic previously underlying OWL ($\mathcal{SHOIN}^{(\mathcal{D})}$) with a number of expressive means that were suggested by ontology developers in order to make it more useful in practice. Among the added constructors there are: complex role inclusions (which allow for representing many kind of properties useful in real-world terminologies); disjoint, reflexive, symmetric, transitive, and irreflexive roles; and, importantly, qualified number restrictions. The resulting logic, called $\mathcal{SROIQ}^{(\mathcal{D})}$, is still a decidibile language (Horrocks, Kutz, & Sattler, 2006) but is computationally harder than $\mathcal{SHOIN}^{(\mathcal{D})}$ (Kazakov, 2008).

Importantly, OWL 2 adds three new specific profiles (i.e., syntactic subsets that can be used in conforming ontology) to improve scalability in typical applications. The three new profiles are: OWL 2 EL, OWL 2 QL and OWL 2 RL, which are all more restrictive than OWL DL (Motik et al., 2009):

OWL 2 EL: is based on the logic $\mathcal{EL}^{++}$, it enables polynomial time algorithms for all the standard reasoning tasks; it is particularly suitable for applications where very large ontologies are needed, and where expressive power can be traded for performance guarantees.

OWL 2 QL: enables conjunctive queries to be answered in logarithmic space using standard relational database technology; it is particularly suitable for applica-

tions on relatively lightweight ontologies with a large numbers of individuals, where data are necessarily/conveniently accessed directly via relational queries.

OWL 2 RL: enables, instead, the implementation of polynomial time reasoning algorithms using rule-extended database technologies, operating directly on RDF triples.

Reasoners. The most notable description logic reasoner at the state-of-the-art are FaCT, DLP (Horrocks & Patel-Schneider, 1999), FaCT++ (Tsarkov & Horrocks, 2006) and Racer (now called RacerPro) (Haarslev & Moeller, 2001; Haarslev & Möller, 2003), More recent tools are the OWL reasoners Pellet (Sirin, Parsia, Grau, Kalyanpur, & Katz, 2007) and HermiT (Motik et al., 2009). They all implement the analytic tableau method [1] and they are able to manage really complex logics (e.g. Racer implements the logic $\mathcal{SHOIQ}^{(\mathcal{D})}$); in particular the last four mentioned tools can all handle OWL Lite as well as OWL DL documents (many of them, moreover, are currently upgrading in order to gradually introducing the new OWL 2 features, when not already handled).

2.2 SAT and SAT-based techniques

Propositional Satisfiability (SAT) is the problem of determining whether a Boolean formula admits at least one satisfying truth assignment to its variables. It is a core problem in mathematical logic and computer theory because it is, for its nature and historical reasons, the most representative NP-complete problem (Garey & Johnson, 1979). The research of efficient SAT techniques is still a crucial point, which is of interest for many areas in computer science, like Automated Reasoning, Model Checking, Formal Methods, and many others.

In a broad sense, a SAT solver is any procedure that is able to decide such a problem. There are many state-of-the-art propositional decision procedures at disposal, such as, e.g., Davis-Putnam-Logemann-Loveland (DPLL) (Davis & Putnam, 1960; Davis, Longemann, & Loveland, 1962), OBDD (Bryant, 1992) procedures, or even partial decision procedures based on the stochastic local search methodology.

However, nowadays, the most efficient and exploited algorithm is DPLL in its many variants (see, e.g., Zhang & Malik, 2002). DPLL tries to find a satisfying assignment recursively by assigning, at each step, a value to a proposition. The input formula must be previously reduced in conjunctive normal form (CNF)[2] (even if *non-CNF* variants of DPLL and *circuit-solvers* are currently really active research topics with increasing performance). At each step, if there exists a clause made up by only one literal, then DPLL assigns it to true; otherwise, it chooses a literal l and it applies *branching*. There

[1] HermiT (Motik et al., 2009), in particular, implements a novel calculus (which is an evolution of the tableaux-based method) known as "hypertableaux" (Motik, Shearer, & Horrocks, 2007; Motik et al., 2009; Baumgartner, Furbach, & Pelzer, 2010).

[2] A Boolean formula is in CNF if and only if it is in the form $\bigwedge_i \bigvee_{j_i} l_{j_i}$ where l_{j_i} are literals. Every disjunction $\bigvee_{j_i} l_{j_i}$ is called *clause*.

are several techniques to improve the efficiency of DPLL such as, e.g., backjumping, learning, random restart (see Zhang & Malik, 2002, for an overview). These techniques yield an impressive advance in the efficiency of the DPLL procedure.

Unsat-core. The SAT formalism is used to solve problems with a wider extent. Besides returning an assignment (a model) when a formula is satisfiable, the solver can also be used to produce a proof of unsatisfiability (Zhang & Malik, 2003) in the case the formula is not satisfiable. This proof helps in finding an unsatisfiable subsets of the original problem clauses that is called an *unsatisfiable (unsat) core* for the formula.

Modern SAT-solvers based on DPLL provide the unsat core as a byproduct of the proof of unsatisfiability. However it is very hard to obtain a minimal core or, even more, to compute a minimal set of clauses that cover all the unsat cores, which are useful information for a lot of SAT-based applications. These kind of proofs and byproducts have been used in various SAT-based Model Checking algorithms (e.g. they are the key step of the refining model in abstraction/refinement based verification). For these reasons, dedicated techniques and tools have been devised for this scope.

All-SAT. The fast enumeration of all the satisfying assignments of a propositional formula is another problem which has many applications in the design of hardware and software. Many solutions, optimization and special tools has been proposed for this problem called *All-SAT* (e.g., Grumberg, Schuster, & Yadgar, 2004; Jin et al., 2005). An approach to this problem that has recently emerged augments a clause-recording propositional satisfiability solver with the ability of to add blocking clauses. One generates a blocking clause from a satisfying assignment by taking its complement. The resulting clause prevents the solver from visiting the same solution again. Every time a blocking clause is added the search is resumed until the instance becomes unsatisfiable.

However, since an approach which naively enumerate each satisfying assignment using the standard SAT-solvers (customized for the satisfiability problem) with blocked solutions could be very inefficient and require a huge amount of memory, various optimization techniques are applied to get smaller and "smarter" blocking clauses, combined with the development of customized tools.

Interpolants. An *interpolant* for an unsatisfiable formula $A \wedge B$ is a formula ϕ such that: (i) A implies ϕ; (ii) $\phi \wedge B$ is unsatisfiable; and (iii) ϕ contains only variables that are common to A and B. If $A \wedge B$ is an unsatisfiable propositional formula, an interpolant for it always exists, and, as an unsatisfiable core, is a byproduct of the proof of unsatisfiability, and can be obtained in linear time from it. Many techniques using interpolants have been devised in the last few years, thus nowadays interpolants are a hot topic, since interpolation-based algorithms have become popular in the field of Model Checking see, e.g., McMillan, 2003.

Many SAT-based techniques have witnessed extremely effective in other domains and especially in the area of formal verification. Starting from Planning (Kautz et al., 1996) and Bounded LTL Model Checking (BMC) (Biere et al., 1999), many other problems have been encoded into SAT or solved with the aid of SAT tools (Audemard, Cimatti, Kornilowicz, & Sebastiani, 2002; Armando et al., 2005; Sebastiani, 2007b) and these approaches are currently state-of-the-art in the respective communities. For example, Model Checking techniques exploiting the power of state-of-the-art SAT solvers have proved to be a valid alternative, and very often superior, to the traditional approaches based on Binary Decision Diagrams (BDDs) (e.g., McMillan, 2003).

2.3 Beyond SAT: SMT and QBF

The advance in Boolean Satisfiability (SAT) solving techniques, together with the concrete needs from real applications, have inspired significant research on richer and more expressive Boolean formalism like Quantified Boolean Formula (QBF) (see, e.g., Plaisted et al., 2003; Giunchiglia et al., 2006) and Satisfiability Modulo Theories (SMT) (see Sebastiani, 2007b; Barrett et al., 2009 for an overview). The formalism of plain Propositional Logic (SAT), in fact, is often not suitable or expressive enough for representing many real-world problems.

QBF. QBF satisfiability (Giunchiglia, Marin, & Narizzano, 2009) is a generalization of the Boolean Satisfiability problem. A Quantified Boolean Formula is a propositional formula with a quantifier prefix, in which both existential and universal quantifiers can be applied to each variable. QBF-satisfiability is probably one of the most representative PSPACE-complete problem (Garey & Johnson, 1979). The most important difference between QBF and SAT (that is, indeed, a NP-complete problem) lies in the fact that the quantification order of the variables in which the formula is evaluated (i.e. the restriction in the order of the decision variables) matters.

QBF solvers are naturally very similar to those implemented for SAT. Currently most state-of-the-art solvers extend DPLL based SAT solving techniques to QBF solving (and also apply conflict-driven clause learning, backjumping and other techniques Zhang & Malik, 2002; Giunchiglia, Narizzano, & Tacchella, 2002, 2003; Giunchiglia et al., 2006) or extend other well known SAT approaches like, e.g., BDDs (Pan & Vardi, 2004).

Recently the QBF problem has also attracted a lot of attention in the formal verification community, because many interesting problems, such as model checking of LTL formulas, are PSPACE-complete and can be naturally modeled as QBF problems (Dershowitz, Hanna, & Katz, 2005; Jussila, Biere, Sinz, Kröning, & Wintersteiger, 2007; Jussila & Biere, 2007). Implementations of QBF solvers are steadily improving and a QBF formulation of a problem may be exponentially more succinct with respect to a SAT one, thus there is a potential for a huge speed-up using QBF solvers.

SMT. SMT can be seen as an extension of SAT in which the input formula is expressed in (a subset of) first-order logic (typically without quantifiers) with respect to a background theory (for example: *linear arithmetic* both over the *reals* and the *integers*, its subclass *difference logic*, the theories of *bit vectors*, of *arrays* and of *lists*, and others).

The dominating approach for SMT, which underlies most state-of-the-art tools, is based on the integration of a SAT solver and one or more domain-specific solvers for the background theories. The SAT solver enumerates truth assignments which satisfy the Boolean abstraction of the input formula (where distinct theory-specific subformulas are represented/abstracted by distinct Boolean atoms), whilst the domain-specific solvers check the consistency in the respective background theory of the set of literals corresponding to the assignments enumerated. This approach is called *lazy*, in contraposition to the *eager* approach, consisting on encoding an SMT formula into an equivalently satisfiable Boolean formula, and on solving the result with a SAT solver (see, e.g., Barrett et al., 2009 for a survey).

In particular, as happened for SAT, some SMT-based approach and other SMT-based tools for problems like interpolants or the fast recognition of small unsatisfiable cores, have been proposed in the last few years (Cimatti et al., 2008, 2007), but they are still at a very preliminary stage, and far from being as mature as SAT-based ones. Among these, notable examples of SMT-based approaches are in the field of Model Checking, in the verification of RTL designs and of systems with an infinite number of states (like real-time and hybrid control systems, and software) (many examples can be found in the survey of Sebastiani, 2007b).

SMT and QBF are very active areas: new solvers and techniques are continuously proposed, and often with improvements of orders of magnitude in performance with respect to the previous approaches.

Chapter 3

Description Logics

In this chapter we give the essential background concerning Description Logics (DLs), which are the application domain of our research. We first give a short overview of the ideas underlying Description Logics and they usefulness in knowledge representation. Second, we introduce the logical constructors, the syntax and the semantics of DLs, with a particular attention to the notation used in this work. Then, after having discussed some typical or prominent inference problems in Description Logic, we close this chapter presenting in more details the logics which are cases of study of this work.

3.1 Knowledge Representation and Description Logic

Knowledge Representation (KR) is an important subject in artificial intelligence and cognitive science. The fundamental goal of KR and automated reasoning is to represent knowledge in a manner that facilitates inferencing (i.e. drawing conclusions) from expressed knowledge. Generally speaking, KR is the approach to store explicit knowledge about a particular domain so that computers are able to process and use it, and above all to infer implicit knowledge from the one explicitly represented. *Description Logics (DLs)* (Baader et al., 2003), in particular, belong to a successful family of logic-based KR formalisms, allowing to represent and reason with conceptual knowledge about a domain of interest.

Description Logics aim at representing the knowledge of an application domain in a structured way, by defining the relevant concepts of the domain and, then, by using these concepts to specify properties of objects and individuals occurring in the domain. The basic syntactic building blocks are, thus, atomic concepts (unary predicates), atomic roles (binary predicates) representing relations between concepts, and individuals (constants). As the name Description Logics indicates, one of the characteristics of these languages is that, unlike some of their predecessors, they are equipped with formal, logic-based semantics. Importantly, the declarative semantic of Description Logic is defined formally and independently from any specific reasoning algorithms.

Another main characteristic of Description Logics is the emphasis on reasoning; the formal, logic-based semantics with which DL languages are equipped allow for providing a

set of reasoning services by mean of whom implicitly represented knowledge can be inferred from the explicitly defined one. Reasoning is the central objective in DLs. Among the others, some notable examples of reasoning services are the classification of concepts, and the problem of decide if an object is instance of a specified concept. Because Description Logics are a KR formalism, and since in KR one usually assumes that a KR system should always answer the queries of a user in reasonable time, DL researchers are interested only in decidable languages and in decision procedures for which termination is guaranteed, both for positive and for negative answers. For its nature, unfortunately, reasoning in Description Logic is a very hard problem (NP-complete, PSPACE-complete, EXPTIME-complete and even more) (Baader et al., 2003), thus even if it can be guaranteed to have an answer in finite time, this not imply that the answer is given in "a reasonable" time. However, that being non-polynomial in the worst case does not prevent a DL reasoning service from being useful in practice, provided that sophisticated optimization techniques are used when implementing a system based on such a DL.

In Description Logic, many different constructors have been proposed as a formalism to describe the concepts of the domain, the objects and the relations among them. Every distinct combination of these language constructors might yield to a new Description Logic. In particular, the expressive power of the logic obtained and the decidability and complexity of the inference problems depend from the effect of combining these constructors. On the one hand, very expressive DLs are likely to have inference problems of high complexity (or they may even be undecidable). On the other hand, very weak DLs (with efficient reasoning procedures) may not be sufficiently expressive to represent the important concepts of the given application domain. Hence, investigating this trade-off between the expressivity of DLs and the complexity of their reasoning problems has been one of the most important issues in DL research. The result of this research is a really wide range of logics which goes from very simple logics with a low expressive power, to very powerful logics where, in contrast, the reasoning is really complex (Baader et al., 2003).

Further than in knowledge representation, Description Logics have been employed with success in various application domains of computer science, ranging from *natural language processing*, *distributed computing* to *databases* and, moreover, recently, to *bio-medical ontologies* and in the *Semantic Web*. In this latest field, nowadays, the Description Logic-based languages of the OWL family (Bechhofer et al., 2004; Motik et al., 2009) have been adopted as the standard ontology languages. For this reason, the problem of automated reasoning in Description Logics has been thoroughly investigated

3.2 Constructors, Notation and Semantics

In Description Logic, a very wide set of different constructors have been proposed as a formalism to describe the concepts of the domain, the objects and the relations among them.

A DL ontology (or knowledge base) is generally formed by two components. The first one is the terminological component (TBox), informally, the schema of the ontology;

the second one is the assertional component (ABox), informally, an instantiation of the terminological schema. The complexity of reasoning on a particular ontology depends not only on the set of language constructors provided by the implemented Description Logic, but also on the presence of one or both (or none) the TBox/ABox components and, moreover, from the structure of the axioms/assertions by mean of which these components are defined. For instance, it can be allowed or not to have a set of axioms defining cyclic terminologies, (fact that requires smarted reasoning algorithms and often increases the theoretical complexity).

For this latest reason, notationally, Description Logics are distinguished by the language constructors they provide and which determine their expressive power. In particular, the *concept constructors* of the language determine how concepts can be defined in the language, while the *ontological constructors* determine the relations among concepts, roles and individuals and further role properties (like, e.g., transitivity). Thus, on the one hand concept constructors define the structure of concepts (as if they was defined wrt. an empty TBox) while ontological constructors determine the TBox/ABox structure and properties.

Constructors and Notation.
The main *concept constructors* with which Description Logics are equipped are listed in the upper-most block of Table 3.1. Formally, *concept descriptions* in Description Logic are inductively defined through a set of logical constructors starting from the non-empty and pair-wise disjoint sets of *concept names* ${N_C}^{\mathcal{T}}$, *role names* ${N_R}^{\mathcal{T}}$, and *individual names* $\mathsf{Ind}^{\mathcal{T}}$. In this work we use the first uppercase letters of the alphabet A, B, C, C_i, D, D_i, $E, \ldots$ to denote concept names, while we use the uppercase letters X, X_i, $Y, \ldots$ or the notation $\hat{C}$, $\hat{D}, \ldots$ to denote generic or complex concepts. Moreover, we represent role names by mean of the lowercase letters r, r_i, s, $t, \ldots$, while we denote individuals with the lowercase letters a, b, c, x, $y, \ldots$. Finally, the *signature* of a concept [resp. role, axiom, axiom set], denoted with $\mathsf{signature}()$, is the set of all the concept, role and individual names occurring in the description of the concept [resp. role, axiom, axiom set]. In particular the signature of a TBox $\mathcal{T}$, $\mathsf{signature}(\mathcal{T})$, is the set of all the concept, role and individual names occurring in $\mathcal{T}$, that is $\mathsf{signature}(\mathcal{T}) = {N_C}^{\mathcal{T}} \cup {N_R}^{\mathcal{T}} \cup \mathsf{Ind}^{\mathcal{T}}$.

A TBox $\mathcal{T}$ in Description Logic is a finite set of axioms defined starting from a set of concept and role descriptions and by mean of a set of axiom constructors. In the second and third blocks of Table 3.1 we have listed the main *axiom constructors* concerning respectively the concepts and roles of a TBox (hereafter, with a small abuse of notation, we sometimes represent role inclusions with the symbol $\sqsubseteq_r$, in order to better distinguish them from concept inclusions, when necessary). Finally, an ABox in Description Logic is a finite set of *concept* and *role assertions* as defined in the lower-most block of Table 3.1. Others concept/axiom constructors have been introduced in Description Logics (e.g., concrete domains, domain/range restrictions,...), but in Table 3.1 we reported the main ones. An *ontology* (or knowledge base) $\mathcal{O}$ is composed of a TBox and a (possibly empty) ABox as previously defined. Let $\mathcal{A}$ be a generic set of axioms/assertions (e.g., an

	Syntax	Semantics
top	$\top$	$\Delta^{\mathcal{I}}$
bottom	$\bot$	$\emptyset$
nominal	$\{a\}$	$\{a^{\mathcal{I}}\}$
negation	$\neg X$	$\Delta^{\mathcal{I}} \setminus X^{\mathcal{I}}$
conjunction	$X \sqcap Y$	$X^{\mathcal{I}} \cap Y^{\mathcal{I}}$
disjunction	$X \sqcup Y$	$X^{\mathcal{I}} \cup Y^{\mathcal{I}}$
existential restriction	$\exists r.Y$	$\{x \in \Delta^{\mathcal{I}} \mid$ there exists $y \in \Delta^{\mathcal{I}}$ s.t. $(x,y) \in r^{\mathcal{I}}$ and $y \in Y^{\mathcal{I}}\}$
universal restriction	$\forall r.Y$	$\{x \in \Delta^{\mathcal{I}} \mid$ for all $y \in \Delta^{\mathcal{I}}$ s.t. $(x,y) \in r^{\mathcal{I}}$ then $y \in Y^{\mathcal{I}}\}$
$\geq$ number restriction	$\geq n\ r.Y$	$\{x \in \Delta^{\mathcal{I}} \mid |FIL(r,x) \cap Y^{\mathcal{I}}| \geq n\ \}$, n integer value s.t. $n \geq 1$
$\leq$ number restriction	$\leq m\ r.Y$	$\{x \in \Delta^{\mathcal{I}} \mid |FIL(r,x) \cap Y^{\mathcal{I}}| \leq m\}$, m integer value s.t. $m \geq 0$
concept inclusion	$X \sqsubseteq Y$	$X^{\mathcal{I}} \subseteq Y^{\mathcal{I}}$
concept definition	$A \equiv Y$	$A^{\mathcal{I}} = Y^{\mathcal{I}}$
role hierarchy	$r \sqsubseteq s$	$r^{\mathcal{I}} \subseteq s^{\mathcal{I}}$
role inclusion	$r_1 \circ \cdots \circ r_n \sqsubseteq s$	$r_1^{\mathcal{I}} \circ \cdots \circ r_k^{\mathcal{I}} \subseteq r^{\mathcal{I}}$
transitivity	$\mathsf{transitive}(r)$	$r^{\mathcal{I}}$ is transitive
reflexivity	$\mathsf{reflexive}(r)$	$r^{\mathcal{I}}$ is reflexive
functionality	$\mathsf{functional}(r)$	$\forall x \in \Delta^{\mathcal{I}} : |FIL(r,x)| \leq 1$
inverse role	$\mathsf{inv}(r,s)$, r^-	$s^{\mathcal{I}}$ is the inverse of $r^{\mathcal{I}}$, i.e. $s^{\mathcal{I}} = (r^-)^{\mathcal{I}} \stackrel{\text{def}}{=} \{(y,x) \mid (x,y) \in r^{\mathcal{I}}\}$
concept assertion	$C(a)$	$a^{\mathcal{I}} \in C^{\mathcal{I}}$
role assertion	$r(a,b)$	$(a^{\mathcal{I}}, b^{\mathcal{I}}) \in r^{\mathcal{I}}$

Table 3.1: Syntax and semantics of the main concept and axiom constructors in DL. (Here, $FIL(r,x)$ is the set of the r-fillers, also called r-successors, of the individual $x \in \Delta^{\mathcal{I}}$ for the role $r \in {N_R}^{\mathcal{I}}$, and is defined as $FIL(r,x) = \{y \in \Delta^{\mathcal{I}} \mid (x,y) \in r^{\mathcal{I}}\}$.)

ontology $\mathcal{O}$ or a TBox $\mathcal{T}$), then we denote by $|\mathcal{A}|$ (the *size* of $\mathcal{A}$) the number of axioms in $\mathcal{A}$.

Other considerations are in order. *Concept inclusions*, as defined in Table 3.1, concern concepts of any kind. If, given a TBox $\mathcal{T}$, there are no cyclic dependency among the concept descriptions/names defined by the axioms of $\mathcal{T}$, then $\mathcal{T}$ is said to be *acyclic*. Moreover, as a rule a TBox $\mathcal{T}$ is defined observing the uniqueness condition, because of which it is assumed that for each concept name A it exists at most one concept definition $A \equiv X$ or concept inclusion $A \sqsubseteq X$, for some concept description X. If it is not the case we say that $\mathcal{T}$ is a *general TBox*, and every axiom of the form $X \sqsubseteq Y$ is a *general concept inclusion (GCI)* (where we use the expression $X \equiv Y$ as an abbreviation of the two GCIs $X \sqsubseteq Y$ and $Y \sqsubseteq X$). With a small abuse of notation we may refer to a *role inclusion (RI)* meaning, indistinctly, a "role hierarchy" or a "role inclusion" axiom of Table 3.1; when necessary, we distinguish the second from the first kind by saying *complex role inclusion*.

Let $\mathcal{L}$ be a specific Description Logic (language) given by the combination of a subset

of the concept/ontological constructors of Table 3.1. Given a TBox $\mathcal{T}$ defined in $\mathcal{L}$, we denote with $\mathsf{PC}_\mathcal{T}$ the set of the *primitive concepts* for $\mathcal{T}$, i.e. the smallest set of concepts containing: (i) the top and the bottom concept $\top$ and $\bot$, if provided by $\mathcal{L}$, and (ii) all concept names in ${N_C}^\mathcal{T}$; if negation is a concept constructor in $\mathcal{L}$ then we denote with $\mathsf{BC}_\mathcal{T}$ the set of the *basic concepts* for $\mathcal{T}$, i.e. the smallest set of concepts such that $\mathsf{PC}_\mathcal{T} \subseteq \mathsf{BC}_\mathcal{T}$ and containing also: (iii) all the concepts of $\mathcal{T}$ in the form $\neg C$ where $C \in {N_C}^\mathcal{T}$.

Semantics.
The semantic of a TBox/ABox in $\mathcal{L}$ is defined in terms of *interpretations*. An interpretation $\mathcal{I}$ is a couple $\mathcal{I} = (\Delta^\mathcal{I}, \cdot^\mathcal{I})$, where $\Delta^\mathcal{I}$ is the domain, i.e. a non-empty set of individuals, and $\cdot^\mathcal{I}$ is the interpretation function which maps each concept name (atomic concept) $A \in {N_C}^\mathcal{T}$ to a set $A^\mathcal{I} \subseteq \Delta^\mathcal{I}$, maps each role name (atomic role) $r \in {N_R}^\mathcal{T}$ to a binary relation $r^\mathcal{I} \subseteq \Delta^\mathcal{I} \times \Delta^\mathcal{I}$, and maps every individual $a \in \mathsf{Ind}^\mathcal{T}$ in the respective individual $a^\mathcal{I}$ of the domain $\Delta^\mathcal{I}$. In the right-most column of Table 3.1 the inductive extensions of $\cdot^\mathcal{I}$ to arbitrary concept descriptions are defined.

An interpretation $\mathcal{I}$ is a *model* of a given TBox $\mathcal{T}$ if and only if the conditions in the Semantics (right-most) column of Table 3.1 are respected for every axiom in $\mathcal{T}$; if and only if this is the case the TBox $\mathcal{T}$ is said to be *consistent*. A concept C is said to be *satisfiable* wrt. $\mathcal{T}$ if and only if there exists a model $\mathcal{I}$ of $\mathcal{T}$ with $C^\mathcal{I} \neq \emptyset$, i.e. there exists an individual $x \in \Delta^\mathcal{I}$ as an instance of C, i.e. such that $x \in C^\mathcal{I}$. A TBox $\mathcal{T}'$ is a *conservative extension* of the TBox $\mathcal{T}$ if every model of $\mathcal{T}'$ is also a model of $\mathcal{T}$, and every model of $\mathcal{T}$ can be extended to a model of $\mathcal{T}'$ by appropriately defining the interpretations of the additional concept and role names.

Examples of Relevant Description Logics.
As previously stated Description Logics are distinguished by the language constructors they provide. The set of language constructors used determines the expressive power of the logic. Historically, the language $\mathcal{AL}$ has been considered as the basic equipment for a description language, providing: atomic negation of concepts, conjunctions, universal restrictions and limited existential quantification on concepts. Starting from $\mathcal{AL}$ many differently expressive Description Logics (mostly richer, but also some less expressive ones) have been proposed and deeply studied. However, not all the languages obtained from the combination of the above expose capabilities are distinct. For example, the language $\mathcal{ALUE}$ is equivalent to the well known logic $\mathcal{ALC}$ since the combination of disjunctions and existential restrictions can be expressed using conjunctions, universal restrictions and negation, and vice versa.

In Table 3.2 we present some notable Description Logics and the main language constructors they provide. In particular: $\mathcal{ALC}$, $\mathcal{ALCQ}$ and $\mathcal{EL}^+$ are cases of study of this work. While $\mathcal{ALC}$ is the most studied Description Logic, $\mathcal{ALCQ}$ is important because it adds qualified number restrictions to $\mathcal{ALC}$. $\mathcal{FL}_0$ is a deeply-investigated subset of $\mathcal{ALC}$ which is tractable for empty TBoxes. $\mathcal{EL}^+$, instead, is a lightweight Description Logic with many prominent applications in representing bio-medical ontologies, among which

		$\mathcal{FL}_0$	$\mathcal{EL}^+$	$\mathcal{EL}^{++}$	$\mathcal{ELHIF}^+$	$\mathcal{ALC}$	$\mathcal{ALCQ}$	$\mathcal{SH}$	$\mathcal{SHIF}^{(\mathcal{D})}$	$\mathcal{SHOIN}^{(\mathcal{D})}$	$\mathcal{SROIQ}^{(\mathcal{D})}$
			Snomed	OWL 2 EL	Galen				OWL Lite	OWL DL	OWL 2
top		☺	☺	☺	☺	☺	☺	☺	☺	☺	☺
bottom				☺	☺	☺	☺	☺	☺	☺	☺
nominals	$\mathcal{O}$			☺	☺					☺	☺
negation	$\mathcal{C}$					☺	☺	☺	☺	☺	☺
conjunction		☺	☺	☺	☺	☺	☺	☺	☺	☺	☺
disjunction	$\mathcal{U}$					☺	☺	☺	☺	☺	☺
existential restr.	$\mathcal{E}$		☺	☺	☺	☺	☺	☺	☺	☺	☺
universal restr.		☺				☺	☺	☺	☺	☺	☺
number restr.	$\mathcal{N}$						☺			☺	☺
qualif. num. restr.	$\mathcal{Q}$						☺				☺
concrete domain				☺	☺				☺	☺	☺
data values/types	$^{(\mathcal{D})}$								☺	☺	☺
role hierarchy	$\mathcal{H}$		☺	☺	☺			☺	☺	☺	☺
complex role inc.	$\mathcal{R}$		☺	☺	☺						☺
transitivity	$^+$		☺	☺	☺			☺	☺	☺	☺
functionality	$\mathcal{F}$				☺				☺	☺	☺
inverse role	$\mathcal{I}$				☺				☺	☺	☺
domain restr.				☺	☺				☺	☺	☺
range restr.				☺	☺				☺	☺	☺

Table 3.2: Relevant Description Logics and their logical constructors.

Snomed-CT (see Section 2.1); $\mathcal{ELHIF}^+$ extends $\mathcal{EL}^+$ with all the constructors required in order to represent, instead, the totality of the ontology Galen (see Section 2.1). $\mathcal{EL}^{++}$ is another extension of $\mathcal{EL}^+$, which is of particular interest because it is the logic underlying the tractable fragment (profile) OWL 2 EL of the OWL 2 standard. The $\mathcal{SH}$ logic is the basic language of another notable class of Description Logics. In more details $\mathcal{SH}$ enriches $\mathcal{ALC}$ with transitive roles and role hierarchies; it is the base on which many harder languages has been defined to reach the real-world quest of expressive logics. For example, all the OWL languages belong to the $\mathcal{SH}$ family. In particular the logic underlying OWL DL is $\mathcal{SHOIN}^{(\mathcal{D})}$ whilst OWL 2 is based on $\mathcal{SHROIQ}^{(\mathcal{D})}$. Finally $\mathcal{SHIF}^{(\mathcal{D})}$ is the logic underlying the OWL Lite fragment of OWL.

3.3 Reasoning Services

A main characteristic of Description Logics is the emphasis on reasoning. The different DL systems offer different set of reasoning services by mean of whom implicitly represented knowledge can be inferred from the explicitly defined one. Plenty of different reasoning services (also known as logical inference problems) have been defined in Description Logics. However some of them are considered fundamental for every DL system and are commonly supported by all the state-of-the-art reasoners. These services are usually referred as *standard* reasoning services. In contraposition, many other new important

inference problems have emerged from practical applications of Description Logics and ontologies, often requiring a more complex forms or higher capabilities of reasoning. Problems in this second class of inferences are usually referred as *non-standard* or *supplemental* reasoning services.

In the following we look in more details at the standard and non-standard reasoning services, in particular at those we tackle in this thesis (for more details we refer to the book of Baader et al., 2003).

3.3.1 Standard Reasoning Services

As briefly exposed in Section 2.1, some of the main standard inference problems are: *concept satisfiability* (or alternatively *concept unsatisfiability*), *concept subsumption*, *classification* and *concept hierarchy computation*, *knowledge-base consistency checking*, *instance checking*, *instance retrieval* and *realization*.

We give the formal definition of these reasoning services (notice that many of these problems can solved being reduced each other):

Knowledge-base Consistency. A given TBox (ontology/knowledge-base) $\mathcal{T}$ is *consistent* if and only if it there exists a model $\mathcal{I}$ for $\mathcal{T}$, otherwise $\mathcal{T}$ is said to be *inconsistent*. We recall from Section 3.2 that an interpretation $\mathcal{I}$ is a *model* for a given TBox $\mathcal{T}$ if and only if $\mathcal{I}$ satisfies the semantics of every axiom of $\mathcal{T}$.

Concept Satisfiability. A concept X is said to be *satisfiable* wrt. the TBox $\mathcal{T}$ if and only if there exists a model $\mathcal{I}$ of $\mathcal{T}$ with $X^{\mathcal{I}} \neq \emptyset$, i.e. there exists an individual $x \in \Delta^{\mathcal{I}}$ as an instance of X, i.e. such that $x \in X^{\mathcal{I}}$. Otherwise the concept is said to be *unsatisfiable*. Similarly, it is possible to solve the *disjointness* problem. Given the concepts X and Y, X *is disjoint from* Y wrt. $\mathcal{T}$, if and only if $X^{\mathcal{I}} \cap Y^{\mathcal{I}} = \emptyset$ for every model $\mathcal{I}$ of $\mathcal{T}$ (that is if $X \sqcap Y$ is unsatisfiable).

Sometimes, it is necessary to decide if a concept X is satisfiable wrt. to an empty TBox; in such a case every arbitrary interpretation that makes X non-empty leads X to be satisfiable.

Subsumption*, *Classification* and *Hierarchy computation. Given the concepts X and Y, Y *subsumes* X wrt. the TBox $\mathcal{T}$, written $X \sqsubseteq_{\mathcal{T}} Y$ (or simply $X \sqsubseteq Y$ when it is clear to which TBox we refer to), if and only if $X^{\mathcal{I}} \subseteq Y^{\mathcal{I}}$ for every model $\mathcal{I}$ of $\mathcal{T}$. The computation of all subsumption relations between concept names occurring in $\mathcal{T}$ is called *classification* of $\mathcal{T}$. Moreover, given the classification of a TBox $\mathcal{T}$, it is possible to compute the *concept hierarchy* (or *subsumption hierarchy*) of $\mathcal{T}$, which is simply the computation of the partial ordering induced by all the subsumption relations among all the concept names of $\mathcal{T}$.

Another well known standard reasoning problem strictly connected to subsumption is *equivalence*. Briefly, X *is equivalent to* Y wrt. the TBox $\mathcal{T}$, written $X \equiv_{\mathcal{T}} Y$, if and only if $X^{\mathcal{I}} = Y^{\mathcal{I}}$ for every model $\mathcal{I}$ of $\mathcal{T}$ (i.e., if both $X \sqsubseteq_{\mathcal{T}} Y$ and $Y \sqsubseteq_{\mathcal{T}} X$).

Instance Checking. Instance checking is the problem of to decide if a given individual (or, respectively, a couple of individuals) is an instance of a given concept (or, respectively, they are an instance of a given role). Formally, given a TBox $\mathcal{T}$, the concept X and the role r of $\mathcal{T}$, then a given individual a is an *instance* of X if and only if $a^{\mathcal{I}} \in X^{\mathcal{I}}$ for every model $\mathcal{I}$ of $\mathcal{T}$. Similarly, given the individuals a, b then the pair (a, b) is an *instance* of r if and only if $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in r^{\mathcal{I}}$ for every model $\mathcal{I}$ of $\mathcal{T}$.

Instance Retrieval* and *Realization. Informally, *instance retrieval* is the problem of to report all the individuals of the input ontology which are instances of a particular concept name C. The dual inference problem to retrieval is realization. Given an individual a and a set of concepts of the a TBox $\mathcal{T}$, *Realization* consists in find the most specific concepts $\{C_i\}_i$ from the set such that a is an instance of every C_i, where, "the most specific concepts" are those which are minimal wrt. the concept hierarchy of $\mathcal{T}$.

3.3.2 Supplemental Reasoning Services

Especially from the fields of ontology manipulation and Semantic Web (but not only) many novel non-standard/supplemental reasoning services have arisen which are of particular utility for ontology developers, maintainers and users. Some example are *logical difference* computation (Konev, Walther, & Wolter, 2008c) and *modularization* (Konev, Lutz, Walther, & Wolter, 2008b; Suntisrivaraporn, Qi, Ji, & Haase, 2008), *axiom pinpointing* (Baader et al., 2007; Peñaloza & Sertkaya, 2010b) also known as the problem of *find a/all justification(s)* (and, furthermore, the identification of *laconic* or *precise modules* or *justifications*; Horridge, Parsia, & Sattler, 2008), *abduction* and *contraction* (Colucci, Noia, Sciascio, Donini, & Mongiello, 2005; Bienvenu, 2008) and many others more. The development of efficient procedures to handle such services is, nowadays, a key research topic in the automated reasoning on Description Logics and ontologies.

In the rest of this section we present in more details the two supplemental inference problems we approached in our work, which are *axiom pinpointing* and *modularization*.

Axiom Pinpointing/Finding Justifications

A very prominent reasoning service emerging from the field of the representation and manipulation of ontologies is the problem of *finding justifications* (e.g., Kalyanpur, Parsia, Horridge, & Sirin, 2007), in the context of Description Logics called also *axiom pinpointing* (e.g., Baader et al., 2007). The quest of efficient procedures for this inference problem arises from many concrete applications where there is the necessity of debug potentially really complex or huge ontologies.

In more details, the axiom pinpointing problem consists in finding one or many minimal subsets of a DL knowledge-base/ontology that have consequence on some inferred properties. So, it can be used, e.g., in order to find one or many sets of axioms (called *MinAs* or *justifications*) which cause an unwanted consequence like, e.g., a subsumption relation or the unsatisfiability of some concepts; in such a way, it allows for debugging

ontologies. In this work we refer to this problem calling it *axiom pinpointing*, because specifically tackle the problem of debugging undesired subsumption relations in Description Logics TBoxes (i.e. the target is to find the axioms responsible of the inference of such a subsumption relation). For the same reasons we chose to use the terminology *MinAs* for representing justifications, as done by Baader et al. (2007).

Formally, consider the subsumption relation $C \sqsubseteq_{\mathcal{T}} D$, with C, D primitive concepts of $\mathcal{T}$. If it holds also $C \sqsubseteq_{\mathcal{S}} D$ for some set $\mathcal{S} \subseteq \mathcal{T}$ of axioms, then $\mathcal{S}$ is called an *axiom set*, shortly *nMinA*, for $C \sqsubseteq D$ wrt. $\mathcal{T}$. If $\mathcal{S}$ is also minimal, i.e. if $C \not\sqsubseteq_{\mathcal{S}'} D$ for every $\mathcal{S}'$ s.t. $\mathcal{S}' \subset \mathcal{S}$, then $\mathcal{S}$ is called a *minimal axiom set*, shortly *MinA*, for $C \sqsubseteq D$ wrt. $\mathcal{T}$. Therefore, in these terms, *axiom pinpointing* is the problem of finding single or all the MinAs for the given subsumption relation. Hereafter we refer to these problems as single/exhaustive axiom pinpointing or, respectively, as the one-MinA/all-MinAs problems.

Notice that the number of possible MinAs for an existing subsumption is can be exponential in the number of the axioms of $\mathcal{T}$ (Baader & Peñaloza, 2007). However, interestingly, axiom pinpointing is a tractable problem in logics like $\mathcal{EL}^+$ with many prominent applications in the field of bio-medical ontologies. In $\mathcal{EL}^+$, in particular, finding one MinA is a polynomial task, while finding all the MinAs is worst-case output-exponential (Baader et al., 2007).

Modularization

Real-world ontologies are often very large in size, proportionally to the expressivity of their underlying logic; for instance, bio-medical ontologies (see Section 2.1) are often huge in size, e.g., SNOMED-CT has more than 300,000 axioms. Thus, despite the availability of tractable/efficient algorithms, the handling of such ontologies is out of the reach for those algorithms, because they can result computationally too expensive when applied to the whole input huge ontologies.

In order to cope with this problem, supplemental reasoning techniques has been proposed in the literature. *Modularization* resulted a key technique in reaching this objective; the modularization technique consists in computing specific subsets, called *modules*, of a given input ontology (TBox) $\mathcal{T}$. A *module* $\mathcal{M}$ of $\mathcal{T}$ is any subset of the axioms of $\mathcal{T}$ which preserves a specific property/statement of interest (for example a subsumption relation) or all the possible statements that can be formulated under a given subset of ontology's symbols of interest (for instance all the subsumption relations existing among some selected concept names). We call *signature* a set of symbols of interest from the input ontology's. We say that a set of axioms $\mathcal{M}_\Sigma$ is a module for the signature Σ in $\mathcal{T}$ (briefly it is a Σ-module) if $\mathcal{M}_\Sigma \subseteq \mathcal{T}$ and $\mathcal{M}_\Sigma$ preserves every statement of interest concerning the symbols in Σ. In particular, when the module $\mathcal{M}_\Sigma$ preserves all the possible subsumption relations concerning the symbols of the signature Σ we say that $\mathcal{M}_\Sigma$ is a *subsumption module* for Σ in $\mathcal{T}$.

Different forms of modularization have been proposed in literature (Noy & Musen, 2003; Seidenberg & Rector, 2006; Grau, Horrocks, Kazakov, & Sattler, 2007; Konev, Lutz,

Walther, & Wolter, 2008a; Konev et al., 2008b; Suntisrivaraporn et al., 2008), including semantic or syntactic and logic-based or not. Since modularization is in general highly complex when based on the respect of some semantic properties (with some exceptions), most of the modularization techniques relies on syntactic methods. In particular, Baader and Suntisrivaraporn (2008), Suntisrivaraporn (2009) succeeded in significantly enhance the performance of axiom pinpointing for large-scale lightweight ontologies by applying modularization.

3.4 The Core Description Logic $\mathcal{ALC}$

Historically, the language $\mathcal{AL}$ (*attributive language*) has been chosen as the basic description language, since it seems to provide the minimal constructors useful for practical interests. $\mathcal{AL}$ allows for: atomic negation of concepts, concepts intersection, universal restriction and limited existential quantification on concepts

The most studied Description Logic, however, has been $\mathcal{ALC}$, that extends $\mathcal{AL}$ with concept disjunction, complex negation of concepts and unrestricted existential quantification, which are essential constructors for many applications (Baader et al., 2003). In fact, $\mathcal{ALC}$ offers both the means for performing complete propositional reasoning (providing negation, and concept union/intersection) and for reasoning in terms of and between individuals, through existential and universal quantification. In the upper- and lower-most blocks of Table 3.3 we recall the constructors, the syntax and the semantics of $\mathcal{ALC}$ (the central block instead refers only to $\mathcal{ALCQ}$).

Reasoning in $\mathcal{ALC}$, unfortunately, is not tractable. Satisfiability wrt. empty or acyclic TBoxes in $\mathcal{ALC}$ is, in fact, a PSPACE-complete problem. The complexity of satisfiability then increase up to EXPTIME-complete for general TBoxes.

Further than $\mathcal{ALC}$, other two logics strictly related to $\mathcal{ALC}$ are cases of study of the present work. The first is the Description Logic $\mathcal{ALCQ}$ (see, e.g., Hollunder & Baader, 1991; Faddoul et al., 2008) which extends $\mathcal{ALC}$ with qualified number restrictions and that we discuss in some more details in Section 3.4.1. The second is the Modal Logic K_m. It is worth reminding that the research in Modal and Description Logics has followed two parallel routes until the core Description Logic $\mathcal{ALC}$ and the core modal logic K_m have been proved to be one a notational variant of the other, by the seminal work of Schild (1991). [1] Thus, reasoning in K_m is equivalent to reasoning in $\mathcal{ALC}$. For historical reasons, in this work we specifically focus on the modal logic K_m, due to the wider set of previous approaches and well-established benchmark problems directly available for such a logic. Therefore, in Section 3.4.2 we provide the very necessary background on modal logic and, specifically, on the logic K_m.

[1] Since then, analogous results have been produced for a bunch of other logics, so that, nowadays the two research lines have mostly merged into one research flow.

	Syntax	Semantics
bottom	$\bot$	$\emptyset$
top	$\top$	$\Delta^{\mathcal{I}}$
negation	$\neg C$	$\Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$
conjunction	$C \sqcap D$	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$
disjunction	$C \sqcup D$	$C^{\mathcal{I}} \cup D^{\mathcal{I}}$
existential restriction	$\exists r.C$	$\{x \in \Delta^{\mathcal{I}} \mid$ there exists $y \in \Delta^{\mathcal{I}}$ s.t. $(x,y) \in r^{\mathcal{I}}$ and $y \in C^{\mathcal{I}}\}$
universal restriction	$\forall r.C$	$\{x \in \Delta^{\mathcal{I}} \mid$ for all $y \in \Delta^{\mathcal{I}}$ s.t. $(x,y) \in r^{\mathcal{I}}$ then $y \in C^{\mathcal{I}}\}$
at-least qualified number restriction	$\geq n\, r.C$	$\{x \in \Delta^{\mathcal{I}} \mid \lvert FIL(r,x) \cap C^{\mathcal{I}}\rvert \geq n\ \}$
at-most qualified number restriction	$\leq m\, r.C$	$\{x \in \Delta^{\mathcal{I}} \mid \lvert FIL(r,x) \cap C^{\mathcal{I}}\rvert \leq m\}$
concept inclusion	$C \sqsubseteq D$	$C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$

Table 3.3: Syntax and semantics of $\mathcal{ALCQ}$ (n, m are integer values with $n \geq 1$ and $m \geq 0$).

3.4.1 Case Study: The Description Logics $\mathcal{ALCQ}$

The Description Logic $\mathcal{ALCQ}$ (Hollunder & Baader, 1991) allows for reasoning also in terms of cardinalities of set of individuals. $\mathcal{ALCQ}$, in fact, extends the core and well-known logic $\mathcal{ALC}$ by adding the *qualified number restriction* constructors, also known as qualified cardinality restrictions (see the central block of Table 3.3).

Qualified number (or cardinality) restrictions ($\mathcal{Q}$) are the generalization of cardinality restrictions on roles quantification ($\mathcal{N}$). The latter ones are used to forces cardinality constraints on the number of possible individuals which are successors of a specified role r. The former ones extends the expressivity of (unqualified) cardinality restrictions by allowing also for defining lower- and upper-bounds to the number of r-successors which are instance of a specific concept. Roughly speaking, for instance, the qualified number restriction $(\leq n\ r.\top)$ is equivalent to the (unqualified) cardinality restriction $(\leq n\ r)$, which establish that the instance of such a concept can have a maximum number n of r-successors. Due to this observation, we only consider (general) qualified number restrictions. An $\mathcal{ALCQ}$ *TBox* (or *ontology*) is a finite set of concept inclusion axioms between $\mathcal{ALCQ}$ concept expressions, as defined in Table 3.3.

Concerning Table 3.3 we recall that $FIL(r,x)$ is the set of the r-fillers of the individual $x \in \Delta^{\mathcal{I}}$ for the role $r \in {N_R}^{\mathcal{T}}$, and it is defined as $FIL(r,x) = \{y \in \Delta^{\mathcal{I}} | (x,y) \in r^{\mathcal{I}}\}$. An interpretation of an $\mathcal{ALCQ}$ TBox, therefore, must also satisfy all the cardinality constraints on the role and concept interpretations imposed by the defined number restrictions.

Reasoning in $\mathcal{ALCQ}$ is worst-case as complex as reasoning in $\mathcal{ALC}$. In fact, concept satisfiability wrt. empty/acyclic TBoxes in $\mathcal{ALCQ}$ is indeed a PSPACE-complete problem, while it is an EXPTIME-complete problem wrt. general TBoxes. However, from a practi-

cal point of view, reasoning on the number of individuals and with cardinality constraints is, algorithmically, significantly harder than reasoning on $\mathcal{ALC}$ (Horrocks, Sattler, & Tobies, 2000a) and requires more sophisticated techniques (see, e.g., Haarslev, Timmann, & Möller, 2001; Haarslev & Möller, 2001). In particular, developing techniques for optimized/efficient reasoning with qualified number restriction is a prominent research issue in Description Logic (see, e.g., Farsiniamarj & Haarslev, 2010; Faddoul & Haarslev, 2010), in fact the current techniques often lacks of efficiency, especially when the number of the restrictions is higher or when the values involved in the restrictions their selves are big. This problem has gained further importance since qualified number restrictions have been added to the OWL 2 recommendation for Semantic Web applications.

3.4.2 Case Study: The Modal Logic K_m

We recall some basic definitions and properties of K_m. Given a non-empty set of primitive propositions $\mathcal{A} = \{A_1, A_2, \ldots\}$, a set of m modal operators $\mathcal{B} = \{\Box_1, \ldots, \Box_m\}$, and the constants "*True*" and "*False*" (that we denote respectively with "$\top$" and "$\bot$") the language of K_m is the least set of formulas containing $\mathcal{A}$, closed under the set of propositional connectives $\{\neg, \wedge, \vee, \rightarrow, \leftrightarrow\}$ and the set of modal operators in $\mathcal{B} \cup \{\Diamond_1, \ldots, \Diamond_m\}$. Notationally, we use the Greek letters $\alpha, \beta, \varphi, \psi, \nu, \pi$ to denote formulas in the language of K_m (K_m-formulas hereafter). Notice that we can consider $\{\neg, \wedge\}$ together with $\mathcal{B}$ as the group of the primitive connectives/operators, defining the remaining in the standard way, that is: "$\Diamond_r\varphi$" for "$\neg\Box_r\neg\varphi$", "$\varphi_1 \vee \varphi_2$" for "$\neg(\neg\varphi_1 \wedge \neg\varphi_2)$", "$\varphi_1 \rightarrow \varphi_2$" for "$\neg(\varphi_1 \wedge \neg\varphi_2)$", "$\varphi_1 \leftrightarrow \varphi_2$" for "$\neg(\varphi_1 \wedge \neg\varphi_2) \wedge \neg(\varphi_2 \wedge \neg\varphi_1)$". (Hereafter formulas like $\neg\neg\psi$ are implicitly assumed to be simplified into ψ, so that, if ψ is $\neg\phi$, then by "$\neg\psi$" we mean "ϕ".) Notationally, we often write "$(\bigwedge_i l_i) \rightarrow \bigvee_j l_j$" for the clause "$\bigvee_j \neg l_i \vee \bigvee_j l_j$", and "$(\bigwedge_i l_i) \rightarrow (\bigwedge_j l_j)$" for the conjunction of clauses "$\bigwedge_j(\bigvee_i \neg l_i \vee l_j)$". Further, we often write $\Box_r$ or $\Diamond_r$ meaning one specific/generic modal operator, where it is assumed that $r = 1, \ldots, m$; and we denote by $\Box_r^i$ the nested application of the $\Box_r$ operator i times: $\Box_r^0\psi := \psi$ and $\Box_r^{i+1}\psi := \Box_r\Box_r^i\psi$. We call *depth* of φ, written *depth(φ)*, the maximum number of nested modal operators in φ. We call a *propositional atom* every primitive proposition in $\mathcal{A}$, and a *propositional literal* every propositional atom (*positive literal*) or its negation (*negative literal*). We call a *modal atom* every formula which is either in the form $\Box_r\varphi$ or in the form $\Diamond_r\varphi$.

In order to make our presentation more uniform, and to avoid considering the polarity of subformulas, we adopt the traditional representation of K_m-formulas (introduced, as far as we know, by Fitting, 1983 and widely used in literature, e.g. Fitting, 1983; Massacci, 2000; Donini & Massacci, 2000) from the following table:

α	α_1	α_2	β	β_1	β_2	π^r	π_0^r	ν^r	ν_0^r
$(\varphi_1 \wedge \varphi_2)$	φ_1	φ_2	$(\varphi_1 \vee \varphi_2)$	φ_1	φ_2	$\Diamond_r\varphi_1$	φ_1	$\Box_r\varphi_1$	φ_1
$\neg(\varphi_1 \vee \varphi_2)$	$\neg\varphi_1$	$\neg\varphi_2$	$\neg(\varphi_1 \wedge \varphi_2)$	$\neg\varphi_1$	$\neg\varphi_2$	$\neg\Box_r\varphi_1$	$\neg\varphi_1$	$\neg\Diamond_r\varphi_1$	$\neg\varphi_1$
$\neg(\varphi_1 \rightarrow \varphi_2)$	φ_1	$\neg\varphi_2$	$(\varphi_1 \rightarrow \varphi_2)$	$\neg\varphi_1$	φ_2				

in which non-literal K_m-formulas are grouped into four categories: α's (conjunctive), β's

(disjunctive), π's (existential), ν's (universal). Importantly, all such formulas occur in the main formula with positive polarity only. This allows for disregarding the issue of polarity of subformulas.

The semantic of modal logics is given by means of Kripke structures. A *Kripke structure* for K_m is a tuple $\mathcal{M} = \langle \mathcal{U}, \mathcal{L}, \mathcal{R}_1, \ldots, \mathcal{R}_m \rangle$, where $\mathcal{U}$ is a set of states, $\mathcal{L}$ is a function $\mathcal{L} : \mathcal{A} \times \mathcal{U} \longmapsto \{True, False\}$, and each $\mathcal{R}_r$ is a binary relation on the states of $\mathcal{U}$. With an abuse of notation we write "$u \in \mathcal{M}$" instead of "$u \in \mathcal{U}$". We call a *situation* any pair $\mathcal{M}, u$, $\mathcal{M}$ being a Kripke structure and $u \in \mathcal{M}$. The binary relation $\models$ between a modal formula φ and a situation $\mathcal{M}, u$ is defined as follows:

$$
\begin{array}{lcl}
\mathcal{M}, u \models \top; & & \\
\mathcal{M}, u \not\models \bot; & & \\
\mathcal{M}, u \models A_i,\ A_i \in \mathcal{A} & \iff & \mathcal{L}(A_i, u) = True; \\
\mathcal{M}, u \models \neg A_i,\ A_i \in \mathcal{A} & \iff & \mathcal{L}(A_i, u) = False; \\
\mathcal{M}, u \models \alpha & \iff & \mathcal{M}, u \models \alpha_1 \textit{ and } \mathcal{M}, u \models \alpha_2; \\
\mathcal{M}, u \models \beta & \iff & \mathcal{M}, u \models \beta_1 \textit{ or } \mathcal{M}, u \models \beta_2; \\
\mathcal{M}, u \models \pi^r & \iff & \mathcal{M}, w \models \pi_0^r \text{ for some } w \in \mathcal{U} \text{ s.t. } \mathcal{R}_r(u, w) \text{ holds in } \mathcal{M}; \\
\mathcal{M}, u \models \nu^r & \iff & \mathcal{M}, w \models \nu_0^r \text{ for every } w \in \mathcal{U} \text{ s.t. } \mathcal{R}_r(u, w) \text{ holds in } \mathcal{M}.
\end{array}
$$

"$\mathcal{M}, u \models \varphi$" should be read as "$\mathcal{M}, u$ *satisfy* φ in K_m" (alternatively, "$\mathcal{M}, u$ K_m-satisfies φ"). We say that a K_m-formula φ is satisfiable in K_m (K_m-satisfiable henceforth) if and only if there exist $\mathcal{M}$ and $u \in \mathcal{M}$ s.t. $\mathcal{M}, u \models \varphi$. (When this causes no ambiguity, we sometimes drop the prefix "K_m-".) We say that w is a *successor* of u through $\mathcal{R}_r$ iff $\mathcal{R}_r(u, w)$ holds in $\mathcal{M}$.

The problem of determining the K_m-satisfiability of a K_m-formula φ is decidable and PSPACE-complete (Ladner, 1977; Halpern & Moses, 1992), even restricting the language to a single Boolean atom (i.e., $\mathcal{A} = \{A_1\}$; Halpern, 1995); if we impose a bound on the modal depth of the K_m-formulas, the problem reduces to NP-complete (Halpern, 1995). For a more detailed description on K_m— including, e.g., axiomatic characterization, decidability and complexity results — we refer the reader to the works of Halpern and Moses (1992), and Halpern (1995).

A K_m-formula is said to be in *Negative Normal Form (NNF)* if it is written in terms of the symbols $\Box_r$, $\Diamond_r$, $\wedge$, $\vee$ and propositional literals A_i, $\neg A_i$ (i.e., if all negations occur only before propositional atoms in $\mathcal{A}$). Every K_m-formula φ can be converted into an equivalent one $NNF(\varphi)$ by recursively applying the rewriting rules: $\neg\Box_r\varphi \Longrightarrow \Diamond_r\neg\varphi$, $\neg\Diamond_r\varphi \Longrightarrow \Box_r\neg\varphi$, $\neg(\varphi_1 \wedge \varphi_2) \Longrightarrow (\neg\varphi_1 \vee \neg\varphi_2)$, $\neg(\varphi_1 \vee \varphi_2) \Longrightarrow (\neg\varphi_1 \wedge \neg\varphi_2)$, $\neg\neg\varphi \Longrightarrow \varphi$.

A K_m-formula is said to be in *Box Normal Form (BNF)* (Pan, Sattler, & Vardi, 2002; Pan & Vardi, 2003) if it is written in terms of the symbols $\Box_r$, $\neg\Box_r$, $\wedge$, $\vee$, and propositional literals A_i, $\neg A_i$ (i.e., if no diamonds are there, and all negations occur only before boxes or before propositional atoms in $\mathcal{A}$). Every K_m-formula φ can be converted into an equivalent one $BNF(\varphi)$ by recursively applying the rewriting rules: $\Diamond_r\varphi \Longrightarrow \neg\Box_r\neg\varphi$, $\neg(\varphi_1 \wedge \varphi_2) \Longrightarrow (\neg\varphi_1 \vee \neg\varphi_2)$, $\neg(\varphi_1 \vee \varphi_2) \Longrightarrow (\neg\varphi_1 \wedge \neg\varphi_2)$, $\neg\neg\varphi \Longrightarrow \varphi$.

3.5 Lightweight Description Logics

Since the first intractability results for Description Logics, a lot of effort has been spent in the attempt of defining tractable logics (in the sense of the polynomiality of the reasoning).

Many simple logics have been studied by the DL community with this aim but, in particular, the logics $\mathcal{FL}_0$ and $\mathcal{FL}^-$ (Baader et al., 2003) are important for historical reasons, as first attempts of find easy and tractable Description Logics. In particular, $\mathcal{FL}^-$ is the sublanguage of $\mathcal{ALC}$ obtained disallowing atomic negation and concept disjunctions whereas $\mathcal{FL}_0$ is the sublanguage of $\mathcal{FL}^-$ obtained disallowing also the limited existential quantification. In particular, universal quantification (also known as value restriction) was considered an indispensable feature to let a language be considered a Description Logic. Nevertheless, $\mathcal{FL}_0$ resulted to be tractable (i.e. polynomial-time decidable) only for empty TBoxes; instead, the complexity of classification in $\mathcal{FL}_0$ grows up to be PSpace-compete and ExpTime-compete when considering cyclic or general TBoxes, respectively. For these (and many other) reasons, the DL community has given up in searching for tractable logics for about two decades, focusing instead in the investigation of increasingly more expressive languages and in the development of efficient and optimized reasoning procedures, having good performance in practice, though worst-case exponential in theory or even worse.

Recently, however, the choice of universal restriction as a mandatory constructor for DLs has been reconsidered, and another easy Description Logic, called $\mathcal{EL}$ (Baader, 2003), strongly emerged due to prominent applications in which universal restrictions are not necessary; currently, the idea is that a Description Logic must provide at least one quantification constructor, either existential or universal. The Description Logic $\mathcal{EL}$ allows for concept conjunctions and for existential restrictions instead of for value restrictions as its counterpart $\mathcal{FL}_0$. This gives to $\mathcal{EL}$ better algorithmic properties with respect to $\mathcal{FL}_0$, being the classification of (even) general $\mathcal{EL}$ TBoxes polynomial-time. These languages have been called *lightweight* Description Logics, being less expressive but tractable in practice. Many extensions of this logic have been considered in the literature (see, e.g., Baader et al., 2005; Baader, Brandt, & Lutz, 2008); while in the next section we discuss in more details the family of Description Logics derived from $\mathcal{EL}$, in Table 3.5 we compare the main logical constructors provided by the above mentioned lightweight/tractable languages. [2]

3.5.1 Case Study: $\mathcal{EL}^+$ and the $\mathcal{EL}$ Family of Description Logics

In contrast to the trend of the last two decades (Baader et al., 2003), in which the research in Description Logic has focused on investigating increasingly expressive logics, the recent quest for tractable logic-based languages arising from the field of bio-medical ontologies has attracted a lot of attention on the *lightweight* Description Logic $\mathcal{EL}$ and its family (Baader et al., 2005; Baader, Lutz, & Suntisrivaraporn, 2006b; Motik & Horrocks, 2008; Magka, Kazakov, & Horrocks, 2010). $\mathcal{EL}$ allows for conjunctions, existential

[2] Notice that we have not tick "general concept inclusions" for $\mathcal{FL}_0$ and for $\mathcal{FL}_-$, not because they are not allowed in these languages, but because they require non-polynomial reasoning algorithms.

	$\mathcal{FL}_0$	$\mathcal{FL}^-$	$\mathcal{HL}$	$\mathcal{EL}$	$\mathcal{ELH}$	$\mathcal{EL}^+$	$\mathcal{ELOH}$	$\mathcal{EL}^{++}$
top	☺	☺	☺	☺	☺	☺	☺	☺
bottom							☺	☺
nominals							☺	☺
conjunction	☺	☺	☺	☺	☺	☺	☺	☺
existential restriction		☺		☺	☺	☺	☺	☺
universal restriction	☺	☺						
concrete domains								☺
general concept inclusion			☺	☺	☺	☺	☺	☺
role hierarchy				☺	☺	☺	☺	☺
complex role inclusion					☺	☺	☺	☺
transitivity					☺	☺	☺	☺
concept assertion								☺
role assertion								☺

Table 3.4: Logical constructors in tractable/lightweight Description Logics.

restrictions and, very importantly, supports TBoxes made of general concept inclusions (GCIs).

In particular, the extension $\mathcal{EL}^+$ of $\mathcal{EL}$ is of particular relevance due to its algorithmic properties and due to its capability of expressing several important and widely used real-world bio-medical ontologies, of which we have reported a brief survey in Section 2.1. Moreover, the Description Logic community has spent a considerable effort in the attempt of extending $\mathcal{EL}$, defining a maximal subset of logical constructors expressive enough to cover as much as possible the needs of the practical applications in the above mentioned ontologies, but whose inference problems remain tractable. Beside the logic $\mathcal{EL}^+$ (Baader et al., 2006b), on which we focus in this work, many other extension of $\mathcal{EL}$ or tractable fragments of even harder logics have been recently studied (Baader et al., 2005, 2008; Kazakov, 2009; Magka et al., 2010).

In the right most block of Table 3.5 we listed the logical constructors allowed by the different Description Logics of the $\mathcal{EL}$-family. [3] We remark that in our classification we considered the original definition of these logics; thus $\mathcal{EL}^{++}$ represent the maximal tractable extension of $\mathcal{EL}$, with constructors which have lately added also to $\mathcal{EL}^+$.

The Description Logic $\mathcal{EL}^+$.
The peculiarity of $\mathcal{EL}^+$ wrt. its sub-logic $\mathcal{EL}$ is that it allows for *complex role inclusion axioms* with composition. This is of particular relevance because it can be used to express not only *role hierarchy* (e.g., $r \sqsubseteq s$) but also important role properties such as *transitivity* (e.g., $r \circ r \sqsubseteq r$) and *right* or *left-identity* (e.g., respectively $r \circ s \sqsubseteq r$ or $s \circ r \sqsubseteq r$).

For the sake of the reader convenience, we recall in Table 3.5 the concept/ontological

[3] Notice that, strictly speaking, $\mathcal{HL}$ is not properly a Description Logic; in fact, it allows only for propositional constructors (the name $\mathcal{HL}$ stays for Horn Logic). However we have included it in our classification, as subset of $\mathcal{EL}$, because in practice there are many $\mathcal{HL}$ ontologies which are expressed by mean of DL-based languages/formalisms.

	Syntax	Semantics
top	$\top$	$\Delta^{\mathcal{I}}$
conjunction	$X \sqcap Y$	$X^{\mathcal{I}} \cap Y^{\mathcal{I}}$
existential restriction	$\exists r.X$	$\{x \in \Delta^{\mathcal{I}} \mid \exists y \in \Delta^{\mathcal{I}} : (x,y) \in r^{\mathcal{I}} \wedge y \in X^{\mathcal{I}}\}$
general concept inclusion	$X \sqsubseteq Y$	$X^{\mathcal{I}} \subseteq Y^{\mathcal{I}}$
role inclusion	$r_1 \circ \cdots \circ r_n \sqsubseteq s$	$r_1^{\mathcal{I}} \circ \cdots \circ r_k^{\mathcal{I}} \subseteq s^{\mathcal{I}}$

Table 3.5: Syntax and semantics of the $\mathcal{EL}^+$ lightweight Description Logic.

constructors for $\mathcal{EL}^+$. Concept descriptions in $\mathcal{EL}^+$ are inductively defined through the constructors listed in the upper part of Table 3.5, starting from the set of the primitive concepts and roles. Then, an $\mathcal{EL}^+$ TBox (or ontology) is a finite set of general concept inclusion (GCI) and role inclusion (RI) axioms, as defined in the lower part of Table 3.5.

For instance, in the following Example 3.5.1 we introduce a sample fragment of an $\mathcal{EL}^+$-based medical ontology that we will use also later along this work.

Example 3.5.1. We consider the following small $\mathcal{EL}^+$ ontology in the medical domain adapted from the one proposed by Suntisrivaraporn (2009), which we also call $\mathcal{O}_{\mathsf{med}}$. [4]

$$\mathsf{Appendix} \sqsubseteq \mathsf{BodyPart} \sqcap \exists \mathsf{partOf}.\mathsf{Intestine} \qquad a_1$$
$$\mathsf{Endocardium} \sqsubseteq \mathsf{Tissue} \sqcap \exists \mathsf{partOf}.\mathsf{HeartValve} \qquad a_2$$
$$\mathsf{Pericardium} \sqsubseteq \mathsf{Tissue} \sqcap \exists \mathsf{containedIn}.\mathsf{Heart} \qquad a_3$$
$$\mathsf{Appendicitis} \sqsubseteq \mathsf{Inflammation} \sqcap \exists \mathsf{hasLocation}.\mathsf{Appendix} \qquad a_4$$
$$\mathsf{Endocarditis} \sqsubseteq \mathsf{Inflammation} \sqcap \exists \mathsf{hasLocation}.\mathsf{Endocardium} \qquad a_5$$
$$\mathsf{Pericarditis} \sqsubseteq \mathsf{Inflammation} \sqcap \exists \mathsf{hasLocation}.\mathsf{Pericardium} \qquad a_6$$
$$\mathsf{Inflammation} \sqsubseteq \mathsf{Disease} \sqcap \exists \mathsf{actsOn}.\mathsf{Tissue} \qquad a_7$$
$$\mathsf{Disease} \sqcap \exists \mathsf{hasLocation}.\mathsf{Heart} \sqsubseteq \mathsf{HeartDisease} \qquad a_8$$
$$\mathsf{HeartDisease} \sqsubseteq \exists \mathsf{hasState}.\mathsf{NeedsTreatement} \qquad a_9$$
$$\mathsf{partOf} \circ \mathsf{partOf} \sqsubseteq_r \mathsf{partOf} \qquad a_{10}$$
$$\mathsf{hasLocation} \circ \mathsf{containedIn} \sqsubseteq_r \mathsf{hasLocation} \qquad a_{11}$$

The ontology expresses the relations between some kinds of inflammatory disease and body parts or their states. Notice that the ontology is made of nine GCIs and two RIs. In particular the RI axiom a_{10} express the transitivity of partOf while a_{11} is a right-identity which express the fact that everything that has location in some body part which is contained in a more general body part is located also in that second body part. As a matter of example the signature of the whole ontology $\mathcal{O}_{\mathsf{med}}$ is $\mathsf{signature}(\mathcal{O}_{\mathsf{med}}) = \{\mathsf{Appendix}$,$\mathsf{BodyPart}$, $\mathsf{Intestine}$, $\mathsf{Endocardium}$, Tissue, $\mathsf{HeartValve}$, $\mathsf{Pericardium}$, Heart, $\mathsf{Appendicitis}$, $\mathsf{Inflammation}$,

[4] We stress that all the sample ontologies included in this work are used only to clarify some points of the exposition. We neither care they are completely inclusive of their specific domains nor we care about the correctness of these (sub)ontologies, both from the medical and the ontology-design perspective.

Endocarditis, Pericarditis, Disease, HeartDisease, NeedsTreatement} ∪ {partOf, containedIn, actsOn, hasLocation, hasState}. ◇

Importantly, *concept subsumption* and *classification* in $\mathcal{EL}^+$ can be performed in *polynomial time* (Baader et al., 2005, 2007). Many others standard reasoning services (e.g. concept satisfiability, consistence checking, instance checking) are polynomially reducible each others (Baader et al., 2003). In this case, concept satisfiability and consistence checking are not interesting, since in $\mathcal{EL}^+$ there is no constructor that can cause logical inconsistencies. Furthermore, the complexity of many others non-standard reasoning services in $\mathcal{EL}^+$ and in its sublogics have been investigated in the literature (e.g., abduction, Bienvenu, 2008). Interestingly, also the *axiom pinpointing* problem has been shown to be tractable in $\mathcal{EL}^+$ (Baader et al., 2007). In particular, in $\mathcal{EL}^+$*finding one MinA* is a *polynomial* problem, while finding *all the MinAs* for a given subsumption relation is *worst-case output exponential*.

Chapter 4

SAT-based Techniques

In this chapter we provide the basic background notions concerning the state-of-the-art SAT-based techniques, which are the baseline of the novel approaches we investigate in this thesis. Here we don't go into the specific theoretical and operational details of the modern SAT techniques; instead, we describe the main technologies offered by the SAT and SMT areas we have exploited in this work, (like, e.g., Conflict Analysis, and others).

Most state-of-the-art SAT procedures are evolutions of the well-known DPLL procedure, therefore we start discussing the basic notions on SAT and on the modern evolution of the DPLL-based procedure. Then we move to lazy Satisfiability Modulo Theory (SMT), presenting also some relevant theories for this work. Finally, we close the background on SAT and SMT by describing the problem of finding all the solutions by mean of the All-SAT and All-SMT techniques.

4.1 Basics on CDCL SAT Solving

The problem of SAT solving is the foundation of our approach. In fact, either directly or indirectly (in the case of SMT), we always rely on SAT solvers to reason on the DL problems we face in this work. Thus, for the best comprehension of the content of this dissertation, we recall some notions on SAT and *Conflict-Driven Clause-Learning (CDCL)* SAT solving. For a much deeper description, we refer the reader to the literature (e.g., Silva & Sakallah, 1996; Zhang & Malik, 2002; Eén & Sörensson, 2004; Lynce & Silva, 2004).

4.1.1 Basics on SAT and Notation.

We assume the standard syntactic and semantic notions of propositional logic. Given a non-empty set of primitive propositions $\mathcal{P} = \{p_1, p_2, \ldots\}$, the language of propositional logic is the least set of formulas containing $\mathcal{P}$ and the primitive constants $\top$ and $\bot$ ("true" and "false") and closed under the set of standard propositional connectives $\{\neg, \wedge, \vee, \rightarrow, \leftrightarrow\}$. We call a *propositional atom* (also called propositional *variable*) every primitive proposition in $\mathcal{P}$, and a *propositional literal* every propositional atom (*positive literal*) or

its negation (*negative literal*). We implicitly remove double negations: e.g., if l is the negative literal $\neg p_i$, by $\neg l$ we mean p_i rather than $\neg\neg p_i$. We represent a truth assignment μ as a conjunction of literals $\bigwedge_i l_i$ (or analogously as a set of literals $\{l_i\}_i$) with the intended meaning that a positive [resp. negative] literal p_i means that p_i is assigned to true [resp. false].

A propositional formula is in *conjunctive normal form, CNF,* if it is written as a conjunction of disjunctions of literals: $\bigwedge_i \bigvee_j l_{ij}$. Each disjunction of literals $\bigvee_j l_{ij}$ is called a *clause*. Notationally, we often write clauses as implications: "$(\bigwedge_i l_i) \rightarrow (\bigvee_j l_j)$" for "$\bigvee_i \neg l_i \vee \bigvee_j l_j$"; also, if η is a conjunction of literals $\bigwedge_i l_i$, we write $\neg\eta$ for the clause $\bigvee_i \neg l_i$, and vice versa.

A *unit clause* is a clause with only one literal. A *Horn clause* is a clause containing at most one positive literal, and a *Horn formula* is a conjunction of Horn clauses. Notice that Horn clauses are either unary positive clauses, or they contain at least one negative literal. A *definite Horn clause* is a non-unary Horn clause containing exactly one positive literal and at least one negative one, and a *definite Horn formula* is a conjunction of definite Horn clauses. (Intuitively, definite Horn formulas represents sets of implications between propositional variables $(\bigwedge_{i=1}^{n} p_i) \rightarrow p_j$ s.t. $n > 0$.)

Notice that a definite Horn formula ϕ is always satisfiable, since it is satisfied by both the assignments $\mu_\top$ and $\mu_\bot$ which assign all variables to true and false respectively. Notice also that, for every subset $\{p_i\}_i$ of propositional variables in ϕ, $\phi \wedge \bigwedge_i p_i$ and $\phi \wedge \bigwedge_i \neg p_i$ are satisfied by $\mu_\top$ and $\mu_\bot$ respectively. Thus, in order to falsify a definite Horn formula ϕ, it is necessary to conjoin to it at least one positive and one negative literal.

The problem of detecting the satisfiability of a propositional CNF formula, also referred as the *SAT problem*, is NP-complete. A *SAT solver* is a tool able to solve the SAT problem. The problem of detecting the satisfiability of a propositional Horn formula, also referred as the *Horn-SAT problem*, is polynomial.

4.1.2 CDCL SAT Solving.

Most state-of-the-art SAT procedures are evolutions of the Davis-Putnam-Longeman-Loveland (DPLL) procedure (Davis & Putnam, 1960; Davis et al., 1962) and they are based on the CDCL paradigm (Silva & Sakallah, 1996; Zhang, Madigan, Moskewicz, & Malik, 2001). A high-level schema of a modern CDCL DPLL engine, adapted from the one presented by Zhang and Malik (2002), is shown in Figure 4.1. The propositional formula φ is in CNF; the assignment μ is initially empty, and it is updated in a stack-based manner.

In the main loop, `decide_next_branch`(φ, μ) (line `15.`) chooses an unassigned literal l from φ according to some heuristic criterion, and adds it to μ. (This operation is called *decision*, l is called *decision literal* and the number of decision literals in μ after this operation is called the *decision level* of l.) In the inner loop, `bcp`(φ, μ) iteratively deduces literals l from the current assignment and updates φ and μ accordingly; this step is repeated until either μ satisfies φ, or μ falsifies φ, or no more literals can be deduced,

```
1.      SatValue DPLL (formula φ, assignment μ)
2.          while (1) {
3.              while (1) {
4.                  status = bcp(φ, μ);
5.                  if (status == sat)
6.                      return sat;
7.                  else if (status == conflict) {
8.                      blevel = analyze_conflict(φ, μ);
9.                      if (blevel == 0)
10.                         return unsat;
11.                     else backtrack(blevel,φ, μ);
12.                 } else break;
13.             }
14.             decide_next_branch(φ, μ);
15.         }
```

Figure 4.1: Schema of a modern Conflict-Driven Clause-Learning DPLL-based SAT solver.

returning sat, conflict and unknown respectively. In the first case, DPLL returns sat. In the second case, analyze_conflict(φ, μ) detects the subset η of μ which caused the conflict (*conflict set*) and the decision level blevel to backtrack. (This process is called *conflict analysis*, and is described in more details below.) If blevel is 0, then a conflict exists even without branching, so that DPLL returns unsat. Otherwise, backtrack(blevel, φ, μ) adds the *blocking clause* $\neg\eta$ to φ (*learning*) and backtracks up to blevel (*backjumping*), popping out of μ all literals whose decision level is greater than blevel, and updating φ accordingly. In the third case, DPLL exits the inner loop, looking for the next decision.

bcp is based on *Boolean Constraint Propagation (BCP)*, that is, the iterative application of *unit propagation*: if a unit clause l occurs in φ, then l is added to μ, all negative occurrences of l are declared false and all clauses with positive occurrences of l are declared satisfied. Current CDCL SAT solvers include extremely fast implementations of bcp based on the *two-watched-literals scheme* (Moskewicz, Madigan, Zhao, Zhang, & Malik, 2001). This scheme maintains the property that only two different unassigned literals on each clause are watched by a pointer. When a watched literal is assigned to false, the pointer moves looking for another unassigned literal to watch; if none is found, then a new unit clause is detected. Satisfied clauses are not removed; rather, they are lazily detected and ignored when performing propagations. This scheme requires, for every literal, only the storage of its current assignment status (true, false, unassigned) and the list of the pointers to the clauses it watches, that, in this way, are immediately accessible from the literal. Importantly, notice that a complete run of bcp requires an amount of steps which is linear in the number of clauses containing the negation of some of the propagated literals.

analyze_conflict works as follows (Silva & Sakallah, 1996; Moskewicz et al., 2001; Zhang et al., 2001). Each literal is tagged with its decision level, that is, the literal

corresponding to the nth decision and the literals derived by unit-propagation after that decision are labeled with n; each non-decision literal l in μ is also tagged by a link to the clause ψ_l causing its unit-propagation (called the *antecedent clause* of l). When a clause ψ is falsified by the current assignment —in which case we say that a *conflict* occurs and ψ is the *conflicting clause*— a *conflict clause* ψ' is computed from ψ s.t. ψ' contains only one literal l_u which has been assigned at the last decision level. ψ' is computed starting from $\psi' = \psi$ by iteratively resolving ψ' with the antecedent clause ψ_l of some literal l in ψ' (typically the last-assigned literal in ψ', see Zhang & Malik, 2002), until some stop criterion is met. E.g., with the *1st-UIP Scheme* the last-assigned literal in ψ' is the one always picked, and the process stops as soon as ψ' contains only one literal l_u assigned at the last decision level; with the *Decision Scheme*, ψ' must contain only decision literals, including the last-assigned one.

Notice that, if φ is a Horn formula, then one run of `bcp` is sufficient to decide its satisfiability: if $\texttt{bcp}(\varphi, \{\})$ returns `conflict`, then φ is unsatisfiable; otherwise φ is satisfiable because, since all unit clauses have been removed from φ, all remaining clauses contain at least one negative literal, so that assigning all unassigned literals to false would satisfy the formula φ.

4.1.3 CDCL SAT Solving Under Assumptions.

The schema in Figure 4.1 can be adapted to check also the satisfiability of a CNF propositional formula φ *under a set of assumptions* $\mathcal{L} \stackrel{def}{=} \{l_1, ..., l_k\}$. (From a purely-logical viewpoint, this corresponds to check the satisfiability of $\bigwedge_{l_i \in \mathcal{L}} l_i \wedge \varphi$.) This works as follows: $l_1, ..., l_k$ are initially assigned to true, they are tagged as decision literals and added to μ, then the decision level is reset to 0 and `DPLL` enters the external loop. If $\bigwedge_{l_i \in \mathcal{L}} l_i \wedge \varphi$ is consistent, then `DPLL` returns `sat`; otherwise, `DPLL` eventually backtracks up to level 0 and then stops, returning `conflict`. Importantly, if `analyze_conflict` uses the Decision Scheme mentioned above, then the final conflict clause will be in the form $\bigvee_{l_j \in \mathcal{L}'} \neg l_j$ s.t. $\mathcal{L}'$ is the (possibly much smaller) subset of $\mathcal{L}$ which actually caused the inconsistency revealed by the SAT solver (i.e., s.t. $\bigwedge_{l_j \in \mathcal{L}'} l_j \wedge \varphi$ is inconsistent). In fact, at the very last branch, `analyze_conflict` will iteratively resolve the conflicting clause with the antecedent clauses of the unit-propagated literals until only decision literals are left: since this conflict has caused a backtrack up to level 0, these literals are necessarily all part of $\mathcal{L}$.

This technique is very useful in some situations. First, sometimes one needs checking the satisfiability of a (possibly very big) formula φ under many different sets of assumptions $\mathcal{L}_1, ..., \mathcal{L}_N$. If this is the case, instead of running `DPLL` on $\bigwedge_{l_i \in \mathcal{L}_j} l_i \wedge \varphi$ for every $\mathcal{L}_j$ —which means parsing the formulas and initializing `DPLL` from scratch each time— it is sufficient to parse φ and initialize `DPLL` only once, and run the search under the different sets of assumptions $\mathcal{L}_1, ..., \mathcal{L}_N$. This is particularly important when parsing and initialization times are relevant wrt. solving times. In particular, if φ is a Horn formula, solving φ under assumptions requires only one run of `bcp`, whose computational cost depends linearly only on the clauses where the unit-propagated literals occur.

Second, this technique can be used in association with the use of *selector variables*: all the clauses ψ_i of φ can be substituted by the corresponding clauses $s_i \rightarrow \psi_i$, all s_is being fresh propositional variables, which are initially assumed to be true (i.e., $\mathcal{L} = \{s_i \mid \psi_i \in \varphi\}$). If φ is unsatisfiable, then the final conflict clause will be of the form $\bigvee_{s_k \in \mathcal{L}'} \neg s_k$, s.t. $\{\psi_k \mid s_k \in \mathcal{L}'\}$ is the actual subset of clauses which caused the inconsistency of φ. This technique is used to compute unsatisfiable cores of CNF propositional formulas (Lynce & Silva, 2004).

4.2 Satisfiability Modulo Theory (SMT)

Satisfiability Modulo Theories (SMT) or, alternatively, *Satisfiability Modulo (the) Theory* $\mathcal{T}$, *SMT*($\mathcal{T}$), is the problem of deciding the satisfiability of a (typically quantifier-free) first-order formula with respect to some decidable first-order background theory $\mathcal{T}$. (Notice that $\mathcal{T}$ can also be a combination of simpler theories: $\mathcal{T} \stackrel{\text{def}}{=} \bigcup_i \mathcal{T}_i$.) We call an *SMT*($\mathcal{T}$) *solver* any tool able to decide SMT($\mathcal{T}$). We call a *theory solver for* $\mathcal{T}$, $\mathcal{T}$*-solver*, any tool able to decide the satisfiability in $\mathcal{T}$ of sets/conjunctions of ground atomic formulas and their negations ($\mathcal{T}$*-literals*). If the input set of $\mathcal{T}$-literals μ is $\mathcal{T}$-unsatisfiable, then $\mathcal{T}$*-solver* returns unsat and the subset η of $\mathcal{T}$-literals in μ which was found $\mathcal{T}$-unsatisfiable (η is hereafter called a $\mathcal{T}$*-conflict set*, and $\neg\eta$ a $\mathcal{T}$*-conflict clause*). If μ is $\mathcal{T}$-satisfiable, then $\mathcal{T}$*-solver* returns sat; it may also be able to return some unassigned $\mathcal{T}$-literal l s.t. $\{l_1, ..., l_n\} \models_{\mathcal{T}} l$, where $\{l_1, ..., l_n\} \subseteq \mu$. We call this process $\mathcal{T}$-deduction and $(\bigvee_{i=1}^{n} \neg l_i \vee l)$ a $\mathcal{T}$*-deduction clause*.

We adopt the following terminology and notation. The bijective function $\mathcal{T}2\mathcal{B}$ ("$\mathcal{T}$-to-Boolean"), called *Boolean abstraction*, maps propositional variables into themselves, ground $\mathcal{T}$-atoms into fresh propositional variables, and is homomorphic w.r.t. Boolean operators and set inclusion. The function $\mathcal{B}2\mathcal{T}$ ("Boolean-to-$\mathcal{T}$"), called *Boolean refinement*, is the inverse of $\mathcal{T}2\mathcal{B}$ (i.e. $\mathcal{B}2\mathcal{T} \stackrel{\text{def}}{=} \mathcal{T}2\mathcal{B}^{-1}$). The symbols φ, ψ, ϕ denote $\mathcal{T}$-formulas, and μ, η denote sets of $\mathcal{T}$-literals. φ^p, ψ^p, instead, denote propositional formulas, and μ^p, η^p denote sets of propositional literals (i.e., truth assignments). In particular we use this latter symbols as synonyms for the Boolean abstraction of φ, ψ, and of μ, η respectively, and vice versa (e.g., φ^p denotes $\mathcal{T}2\mathcal{B}(\varphi)$, and μ denotes $\mathcal{B}2\mathcal{T}(\mu^p)$). If $\mathcal{T}2\mathcal{B}(\mu) \models \mathcal{T}2\mathcal{B}(\varphi)$, then we say that μ *propositionally satisfies* φ written $\mu \models_p \varphi$. With a little abuse of terminology, we will omit specifying "the Boolean abstraction of" when referring to propositional reasoning steps.

Examples of the more prominent and interesting theories provided by the state-of-the-art SMT solvers are those of *Equality and Uninterpreted Functions* ($\mathcal{EUF}$), *Linear Arithmetic* ($\mathcal{LA}$), both *over the reals/rationals* ($\mathcal{LA}(\mathbb{Q})$) and *over the integers* ($\mathcal{LA}(\mathbb{Z})$), its subclasses of *Difference Logic* ($\mathcal{DL}$), *Unit-Two-Variable-Per-Inequality* ($\mathcal{UTVPI}$) and the novel *Theory of Costs* ($\mathcal{C}$) (Cimatti et al., 2010), the theories of *Bit-Vectors* ($\mathcal{BV}$), of *Arrays* ($\mathcal{AR}$) and of *Lists* ($\mathcal{LI}$). These problems are typically not handled adequately by

```
1.    SatValue T-DPLL (T-formula φ)
2.       φ^p = T2B(φ);
3.       while (DPLL(φ^p, μ_i^p) == sat) {
4.          if (T-solver(B2T (μ_i^p)) == sat)
5.             return sat;
6.          φ^p = φ^p ∧ ¬μ_i^p;
7.       }
8.       return unsat;
```

Figure 4.2: A simplified offline integration schema for lazy SMT($\mathcal{T}$) procedures.

standard automated theorem provers. In order to devise efficient $\mathcal{T}$-solvers for the various theories, SMT borrows ideas and techniques from many disciplines. For instance, the most efficient $\mathcal{T}$-solvers for theories involving arithmetic (e.g., $\mathcal{LA}(\mathbb{Q})$, $\mathcal{LA}(\mathbb{Z})$, $\mathcal{DL}$) are based on numerical algorithms borrowed from linear programming, integer programming and shortest-path, which have been adapted to work in a logic context in synergy with a SAT solver. However, the choice of suitable procedures for the $\mathcal{T}$-solvers and their adaptation for being integrated with SAT solvers (and vice versa) is not always straightforward. In fact, the features which make a SAT solver or a $\mathcal{T}$-solver suitable for an efficient integration are often different from those which make them efficient as standalone solvers. For example, some features of a $\mathcal{T}$-solver which allow for maximizing the synergy with the SAT solver are often more important than the efficiency of the $\mathcal{T}$-solver itself (Sebastiani, 2007b).

4.2.1 Lazy SMT

If we look at SMT from a SAT perspective, then the problem is equivalent to searching for a truth assignment μ^p satisfying the propositional formula $\mathcal{T}2\mathcal{B}(\varphi)$ and being consistent wrt. the theory $\mathcal{T}$. To this extent, most SMT solvers are based on the *lazy SMT* paradigm. In a lazy SMT($\mathcal{T}$) solver a DPLL-based SAT solver is used to enumerate the propositional truth assignments $\mu^p_1, \mu^p_2, ...$ satisfying $\varphi^p = \mathcal{T}2\mathcal{B}(\varphi)$: every time a satisfying assignment μ^p_i is generated, μ^p_i is fed to the $\mathcal{T}$-solver in order to be checked for $\mathcal{T}$-consistency. In a naive schema if μ^p_i is $\mathcal{T}$-consistent, then φ is $\mathcal{T}$-consistent and the SMT($\mathcal{T}$) solver returns `sat`; otherwise μ^p_i is used as blocking clause in order to avoid generating μ^p_i again, and the enumeration of the truth assignments continue. We call $\mathcal{T}$-`DPLL` a procedure for SMT($\mathcal{T}$) implementing the lazy approach. A pseudocode for $\mathcal{T}$-`DPLL` procedure following the naive schema above exposed, is given in Figure 4.2; it integrates a DPLL procedure enumerating assignments and a $\mathcal{T}$-solver checking $\mathcal{T}$-consistency.

Modern SMT solvers, however, exploit the evolution of modern state-of-the-art DPLL-based SAT solvers, which are built on the Conflict-Driven Clause-Learning schema ex-

```
1.     SatValue T-DPLL (T-formula φ)
2.        if (T-preprocess(φ) == conflict);
3.           return unsat;
4.        φ^p = T2B (φ);
5.        while (1) {
6.           T-decide_next_branch(φ^p, μ_i^p);
7.           while (1) {
8.              status = T-deduce(φ^p, μ_i^p);
9.              if (status == sat) {
10.                μ_i = B2T (μ_i^p);
11.                return sat;
12.             } else if (status == conflict) {
13.                blevel = T-analyze_conflict(φ^p, μ_i^p);
14.                if (blevel == 0)
15.                   return unsat;
16.                else T-backtrack(blevel,φ^p, μ_i^p);
17.             } else break;
18.          }
19.       }
```

Figure 4.3: An online schema of $\mathcal{T}$-DPLL based on modern CDCL DPLL-based engine.

posed in Section 4.1.2. In the advance scheme, if μ_i^p is $\mathcal{T}$-consistent, then φ is $\mathcal{T}$-consistent and the SMT($\mathcal{T}$) solver returns sat; otherwise, $\mathcal{T}$-*solver* returns the conflict set η causing the inconsistency. The $\mathcal{T}$-conflict clause $\neg\eta$ is used by the CDCL SAT solver as a conflicting clause for triggering the backjumping and learning mechanism (called $\mathcal{T}$-*backjumping* and $\mathcal{T}$-*learning*). Thus, the information from the theory are used to improve the choice of next branches, to find more efficient conflicts and to guide the backtracking process. The enumeration proceeds until a $\mathcal{T}$-consistent assignment μ_i^p is found, otherwise the SMT($\mathcal{T}$) solver returns unsat. Important optimizations are *early pruning* and $\mathcal{T}$-*propagation*: the $\mathcal{T}$-solver is invoked also on intermediate assignments $\mu_i^{p\prime}$: if it is $\mathcal{T}$-unsatisfiable, then the procedure can backtrack; if not, and if the $\mathcal{T}$-*solver* performs a $\mathcal{T}$-deduction $\{l_1, ..., l_n\} \models_{\mathcal{T}} l$ (where the literal l represents the assignment of an unassigned atom in $\mu_i^{p\prime}$), then l can be unit-propagated, and the $\mathcal{T}$-deduction clause $(\bigvee_{i=1}^{n} \neg l_i \vee l)$ can be used in backjumping and learning.

The above schema is a coarse abstraction of the procedures underlying all the state-of-the-art lazy SMT tools. The state-of-the-art lazy SMT solvers directly integrate online the components of a modified CDCL SAT solver (see Figure 4.1) with the handling of $\mathcal{T}$ like in the procedure of Figure 4.3 (Sebastiani, 2007b). The interested reader is pointed to, e.g., the works of Sebastiani (2007b), Barrett et al. (2009), for details, surveys on SMT and SMT-solving techniques and further references.

4.2.2 The Theory of Linear Arithmetic over the Integers ($\mathcal{LA}(\mathbb{Z})$)

The theory of *Linear Arithmetic over the integers* ($\mathcal{LA}(\mathbb{Z})$) is the quantifier-free First Order theory with equality whose atoms are written in the form $(a_1 \cdot x_1 + \cdots + a_n \cdot x_n \bowtie a_0)$, such that $\bowtie \in \{<, \leq, =, \neq, \geq, >\}$, and the a_is are (interpreted) constant symbols, each labeling one value in $\mathbb{Z}$. The atomic expressions are interpreted according to the standard semantics of linear arithmetic on $\mathbb{Z}$. $\mathcal{LA}(\mathbb{Z})$ is stably-infinite and non-convex. The $\mathcal{LA}(\mathbb{Z})$-satisfiability of sets of quantifier-free literals is decidable and NP-complete (Papadimitriou, 1981). Many algorithms have been conceived, involving techniques like Euler's reduction, Gomory-cuts application, Fourier-Motzkin algorithm, branch-and-bound (We refer the reader to, e.g., Sebastiani, 2007b for the bibliography concerning more formal definitions and notions on $\mathcal{LA}(\mathbb{Z})$ and also $\mathcal{LA}(\mathbb{Q})$.) Many incremental and backtrackable algorithms for $\mathcal{LA}(\mathbb{Z})$-solvers have been conceived, which can perform conflict-set generation and theory propagation (see, e.g., Berezin, Ganesh, & Dill, 2003; Dutertre & de Moura, 2006). In general, it is very important to remark that, in order to avoid incorrect results due to numerical errors and to overflows, all $\mathcal{T}$-solvers for $\mathcal{LA}(\mathbb{Z})$ (or $\mathcal{LA}(\mathbb{Q})$ and their subtheories) which are based on numerical algorithms must be implemented on top of infinite-precision-arithmetic software packages. In the next section we will discuss with more details the Theory of Costs $\mathcal{C}$ which is a sub-theory of $\mathcal{LA}(\mathbb{Z})$.

Example 4.2.1. Here below we report two simple SMT($\mathcal{LA}(\mathbb{Z})$) formulas φ and ψ, which are, respectively, satisfiable and unsatisfiable:

$$\begin{aligned}\varphi \stackrel{\text{def}}{=} & a \wedge (b \vee c) \\ & \wedge (a \rightarrow (x + 2y > 15)) \\ & \wedge (b \rightarrow (x + y = 10)) \\ & \wedge (c \rightarrow (x \geq 5)),\end{aligned} \qquad \begin{aligned}\psi \stackrel{\text{def}}{=} & a \wedge (b \wedge c) \\ & \wedge (a \rightarrow (x + 2y > 15)) \\ & \wedge (b \rightarrow (x + y = 10)) \\ & \wedge (c \rightarrow (x \geq 5)).\end{aligned}$$

In fact φ is satisfied, e.g., by the assignment $\mu = \{a, b, \neg c, (x+2y>15), (x+y=10), \neg(x \geq 5)\}$, where μ is $\mathcal{LA}(\mathbb{Z})$-consistent, because the inequations: $(x + 2y > 15)$ and $(x + y = 10)$ are satisfied in $\mathcal{LA}(\mathbb{Z})$ by every x, y such that $y > 5$ and $x = 10 - y$. On the contrary, in the case of ψ, also c and $(x \geq 5)$ must be assigned to true in order to propositionally satisfy $\mathcal{T}2\mathcal{B}\psi$; this create a conflict wrt. the $\mathcal{LA}(\mathbb{Z})$ theory because there exist no integer values x, y satisfying all the inequations in $\{(x + 2y > 15), (x + y = 10), (x \geq 5)\}$ (in fact, for $x \geq 5$ and $x = 10 - y$ the maximum value of $x + 2y$ is 15 for $x = 5$ and, thus, $y = 5$). $\diamond$

4.2.3 Case Study: The Theory of Costs ($\mathcal{C}$)

The language of the *Theory of Costs* allows for express multiple *cost functions* and, for each of these, allows for define cost increases and both lower- and upper-bounds depending on arbitrary Boolean conditions. In particular, in this work we consider a theory of costs over the integer in which every cost function is a *Boolean cost function*.

Let $\mathcal{T}$ be a first-order theory. We consider a pair $\langle \varphi, \mathcal{F} \rangle$, where $\mathcal{F} \stackrel{\text{def}}{=} \{cost_i \mid i = 1, \ldots, M\}$ is a set of M distinct integer cost functions and where φ is a Boolean combination of ground $\mathcal{T}$-atoms and atoms in the form

$$(cost_i \leq c), \tag{4.1}$$

with c is an integer value. [1] We focus on problems in which every $cost_i$ is a Boolean cost function (over the integers) in the form:

$$cost_i = \sum_{j=1}^{N_i} \textit{if-then-else}(\psi_{i,j}, c_{i,j}^{\top}, c_{i,j}^{\bot}) \tag{4.2}$$

sucht that, for every i and every j, $\psi_{i,j}$ is a formula in $\mathcal{T}$, $c_{i,j}^{\top}, c_{i,j}^{\bot}$ are integer constant values and *if-then-else* is a function such that *if-then-else*$(\psi_{i,j}, c_{i,j}^{\top}, c_{i,j}^{\bot})$ returns $c_{i,j}^{\top}$ if $\psi_{i,j}$ holds, $c_{i,j}^{\bot}$ otherwise. Wlog. we can restrict our attention to problems $\langle \varphi, \mathcal{F} \rangle$ in which, for every i:

$$cost_i = \sum_{j=1}^{N_i} \textit{if-then-else}(A_{i,j}, c_{i,j}, 0) \tag{4.3}$$

and such that, for every j, $A_{i,j}$ is a Boolean literal and $c_{i,j} > 0$. In fact, any problem with cost functions in form (4.2) can be convert straightforwardly and in linear time into another problem with cost functions in form (4.3), not affecting the solutions of the problem (Cimatti et al., 2010).

The problem consists in to decide the satisfiability of the formula φ under the background theory $\mathcal{T}$ and satisfying all the cost constraints of the form (4.1), i.e. finding a satisfying assignment for φ having a cost within the admissible range. Every function (4.3) can be easily encoded into subformulas in the theory of linear arithmetic over the integers ($\mathcal{LA}(\mathbb{Z})$), and the whole problem $\langle \varphi, \mathcal{F} \rangle$ into a ground $\mathcal{T} \cup \mathcal{LA}(\mathbb{Z})$-formula, with $\mathcal{T}, \mathcal{LA}(\mathbb{Z})$ completely-disjoint theories, so that to be handled by an SMT solver (Cimatti et al., 2010). However, for efficiency reasons the problem has been addressed in SMT by introducing an ad-hoc *Theory of Costs* $\mathcal{C}$ (Cimatti et al., 2010) (which, wrt. the use of linear arithmetic, also results in a much more clear and compact formalism) consisting in:

- a collection $\mathcal{V}$ of M fresh integer variables $\mathcal{V} = \{v_1^{cost}, \ldots, v_M^{cost}\}$, that we call *cost variables*, respectively denoting the final output of the functions $cost_1, \ldots, cost_M$ of type (4.3);
- a fresh binary predicate BC (*bounded cost*) defined over the set of the cost variables and the set of the integers, such that $\mathsf{BC}(v_i^{cost}, c)$ represents the constraint "$(cost_i \leq c)$" (4.1), i.e. the predicate is true if the cost function $cost_i$ (whose final cost is represented by v_i^{cost}) is upper-bounded by the integer value c, and false vice versa.

[1] Notice that every atom in the form $(cost_i \bowtie c)$ with $\bowtie \in \{=, \neq, <, \leq, >, \geq\}$ can be expressed as a Boolean combination of $j \geq 1$ atoms in the form $(cost_i \bowtie c_j)$, for some c_j integer values derived from c. For instance $(cost_i \neq c)$ can be expressed as $(cost_i \leq c - 1) \vee \neg(cost_i \leq c)$.

– a fresh ternary predicate IC (*incur cost*) defined over the sets of the cost variables, of the integers and of the naturals, such that every $\mathsf{IC}(v_i^{cost}, c_{i,j}, j)$ represents the jth element of sum (4.3), i.e. the predicate is true if $A_{i,j}$ in (4.3) is true so that the amount $c_{i,j}$ is added to the cost function $cost_i$ (corresponding to an increment of $c_{i,j}$ of the cost variable v_i^{cost}), and false vice versa. [2]For every function of type (4.3) exactly N_i distinct atoms $\mathsf{IC}(v_i^{cost}, c_{i,j}, j)$ must be introduced.

We call $\mathcal{C}$*-atoms* all the BC and IC atoms, and $\mathcal{C}$*-literals* all $\mathcal{C}$-atoms and their negations. We call a $\mathcal{C}\cup\mathcal{T}$*-formula* any Boolean combination of ground $\mathcal{T}$- and $\mathcal{C}$-atoms. We call $\mathcal{C}$*-solver* a decision procedure (theory solver) for the Theory of Costs $\mathcal{C}$ above exposed. Given a $\mathcal{C}\cup\mathcal{T}$*-formula* φ the $\mathcal{C}$-solver takes as input a truth assignment $\mu_{\mathcal{C}}$ to the $\mathcal{C}$-literals of φ and checks whether $\mu_{\mathcal{C}}$ is $\mathcal{C}$-satisfiable, i.e. if the assignment $\mu_{\mathcal{C}}$ is consistent wrt. to the Theory of Costs. Informally speaking, the assignment $\mu_{\mathcal{C}}$ is consistent wrt. to the Theory of Costs if, for every cost variable v_i^{cost}, the sum (4.3) of the incur costs determined by the assignment of the IC-literals respect the constraints (4.1) determined by the assignment of the respective BC-literals. In this work we are interested only in the case in which $\mathcal{T}$ is pure propositional logic; we simply call $\mathcal{C}$*-formula* every $\mathcal{C}\cup\mathcal{T}$-formula in which $\mathcal{T}$ is pure propositional logic and we call SMT($\mathcal{C}$) solver the solver including the $\mathcal{C}$-solver for the Theory of Costs.

With this formalism, notice that:

– to force a $\mathcal{C}$-atom $\mathsf{BC}(v_i^{cost}, c)$ to be true mean to state an upper-bound of c for the cost function represented by v_i^{cost};

– similarly, it is possible to state a lower-bound (with value $c+1$) for v_i^{cost} by forcing to true the $\mathcal{C}$-literal $\neg\mathsf{BC}(v_i^{cost}, c)$;

– the jth incur cost for v_i^{cost} represented by the $\mathcal{C}$-atom $\mathsf{IC}(v_i^{cost}, c_{i,j}, j)$ contributes to the final cost of v_i^{cost} with an amount of $c_{i,j}$ only if such an atom is assigned to true;

– if in an SMT($\mathcal{C}$) formula every stated incur cost IC for the variable v_i^{cost} has value $c_{i,j} = 1$, then fixing an upper-bound [resp. lower-bound] of value c for v_i^{cost} through a BC literal, forces at-most [resp. at-least] c IC-atoms for v_i^{cost} to be assigned to true, in order to satisfy the formula.

Example 4.2.2. Consider, for instance, the following SMT($\mathcal{C}$) formula:

$$\begin{gathered}\mathsf{spend} \wedge \mathsf{save} \wedge (\mathsf{buy\_new} \vee \mathsf{recharge}) \\ \wedge (\mathsf{spend} \rightarrow \neg\mathsf{BC}(cost, 0)) \wedge (\mathsf{save} \rightarrow \mathsf{BC}(cost, 5)) \\ \wedge (\mathsf{buy\_new} \rightarrow \mathsf{IC}(cost, 4, 1)) \wedge (\mathsf{IC}(cost, 4, 1) \rightarrow \mathsf{tax}_2) \wedge (\mathsf{tax}_2 \rightarrow \mathsf{IC}(cost, 2, 2)) \\ \wedge (\mathsf{recharge} \rightarrow \mathsf{IC}(cost, 2, 1)) \wedge (\mathsf{IC}(cost, 2, 1) \rightarrow \mathsf{tax}_1) \wedge (\mathsf{tax}_1 \rightarrow \mathsf{IC}(cost, 1, 1))\end{gathered}$$

that is satisfied by the truth assignment $\mu = \{\mathsf{spend}, \mathsf{save}, \mathsf{recharge}, \mathsf{tax}_1, \mathsf{IC}(cost, 2, 1), \mathsf{IC}(cost, 1, 1), \mathsf{BC}(cost, 5)\} \cup \{\neg\mathsf{buy\_new}, \neg\mathsf{tax}_2, \neg\mathsf{IC}(cost, 4, 1), \neg\mathsf{IC}(cost, 2, 2),$

[2]The index j in $\mathsf{IC}(v_i^{cost}, c_{i,j}, j)$ is necessary to avoid using exactly the same predicate instantiation (atom) for two constants $c_{i,j}$ and $c_{i,j'}$ with the same value but different indexes j and j'.

$\neg\mathsf{BC}(cost, 0)\}$. Notice that μ is $\mathcal{C}$-consistent because the final value of the cost variable *cost* is 3 and, in particular, $0 < cost \leq 5$. Notice that $\mathsf{IC}(cost, 2, 1)$ and $\mathsf{IC}(cost, 2, 2)$ are two distinct $\mathcal{C}$-literals (having the same incur cost but different indexes) representing two different sources of cost. Notice, finally, that the formula could be satisfied also by assigning tax_2 and $\mathsf{IC}(cost, 2, 2)$ to true, in which case the final value of *cost* would be 5; in the practice, the implementations of SMT($\mathcal{C}$) and in particular of the $\mathcal{C}$-solver (Cimatti et al., 2010) can be instructed to address the cost minimization problem, so that the problem is solved by finding one assigment such that it satisfies the input formula and minimizes one (elected) of the Boolean cost functions. ◇

4.2.4 All-SAT and All-SMT

Conventional SAT/SMT solvers are targeted to computing just one solution. However, it is possible to modify the `DPLL` and $\mathcal{T}$`-DPLL` procedures of Figures 4.1 and 4.3 respectively (see Sections 4.1.2 and 4.2.1), so that to enumerate *all models* of a satisfiable propositional/$\mathcal{T}$ formula φ. For this purpose several approaches have been proposed in literature, in order to turn a `DPLL`/$\mathcal{T}$`-DPLL` engine into an *All-SAT*/*All-SMT* tool, i.e., an engine that can enumerate all models for the given input formula (e.g. Grumberg et al., 2004; Jin et al., 2005; Lahiri et al., 2006).

In principle, to solve All-SAT, it is enough to force the SAT solver to continue the search after getting each satisfying assignment. In the first All-SAT approaches once a satisfying assignment is found then a blocking clause is generated by taking its complement. Blocking clauses are added to the input formula being examined to prevent the SAT solver from finding the same solution again. This basic approach could be very inefficient due to the worst-case exponential growth of the clause set (one additional blocking clause for each model found) and due to the fact that it is often necessary to restart the search from scratch. Therefore, various optimization techniques have been applied in order to re-use as much as possible the current search state and to get smaller blocking clauses (Jin et al., 2005; Lahiri et al., 2006).

In more details, given the formula φ, whenever a (partial) model μ for φ is found, the key idea behind All-SAT/All-SMT is to treat $\neg\mu$ as a blocking clause and then to continue the search. At the end of the search, the list $\mu_1, ..., \mu_n$ of models found is a list of all models of φ. One problem of this naive procedure is that adding to φ a "fake" blocking clause (namely $\neg\eta_k$) each time a new satisfying truth assignment η_k is found may cause an exponential blowup of φ. However, Lahiri et al. (2006) have shown that this problem can be overcome by exploiting the best known conflict analysis techniques and conflict-driven backjumping, without the need of keeping the blocking clauses or the learned lemmas. Each time a model η_k is found, it is possible to consider $\neg\eta_k$ as a conflicting clause to feed to `analyze_conflict` and to perform conflict-driven backjumping as if the blocking clause $\neg\eta_k$ belonged to the clause set; importantly, *it is not necessary to add permanently the conflicting clause* $\neg\eta_k$ *to* φ *as a blocking clause*. Keeping the lemmas learned in backjump steps (or the blocking clauses) is only optional: it is sufficient to keep the conflict clause

resulting from conflict analysis only as long as they are active. (A clause is considered to be currently *active* if it occurs in the implication graph, that is, if it is the antecedent clause of some literal in the current assignment. See Zhang et al., 2001.)

Lahiri et al. (2006) proved that this technique terminates and allows for enumerating all models. (The proof is based on a notion of lexicographic ordering on search states, stating that a search state is more advanced than another if it contains more information at lower decision levels.) The only potential drawback of this technique is that some models may be found more than once. However, according to the empirical evaluation of Lahiri et al. (2006), this events appears to be rare and it has very low impact on performances, which are much better than those of the naive version.

Notice at last that this approach works identically for SAT and SMT, once checked that the satisfying enumerated truth assignments are also $\mathcal{T}$-consistent. In particular, the All-SMT technique can be applied also to enumerate all the counter-models of φ with respect to a theory $\mathcal{T}$; more specifically, All-SMT can be used to enumerate all the assignments for which a given purely propositional formula φ is unsatisfiable, if a second `DPLL` instance run again over φ is used as $\mathcal{T}$-solver. We refer the reader to the work of Lahiri et al. (2006) for a more detailed explanation of All-SMT.

Part II

Original Contributions

Chapter 5

Encoding $\mathcal{ALC}/K(m)$-satisfiability into SAT

In the last two decades, the problem of automated reasoning in modal and description logics has been thoroughly investigated. In particular, many approaches have been proposed for efficiently handling the satisfiability of the core normal modal logic K_m, and of its notational variant, the description logic $\mathcal{ALC}$. Although simple in structure, $K_m/\mathcal{ALC}$ is computationally very hard to reason on, its satisfiability being PSPACE-complete.

In this first step of our work we start exploring the idea of performing automated reasoning tasks in modal and description logics by encoding them into SAT, so that to be handled by state-of-the-art SAT tools. As with most previous approaches, we begin our investigation from the satisfiability in K_m, which offer a very wide set of approaches and systems with which to compare. We propose an efficient encoding, and we test it on an extensive set of benchmarks, comparing against the main state-of-the-art tools available. [1]

5.1 Previous Approaches and Related Works

In the last twenty years, modal and description logics have provided an essential framework for many applications in numerous areas of computer science, including artificial intelligence, formal verification, database theory, distributed computing and, more recently, semantic web and ontologies. For this reason, the problem of automated reasoning in modal and description logics has been thoroughly investigated (e.g., Fitting, 1983; Ladner, 1977; Baader & Hollunder, 1991; Halpern & Moses, 1992; Baader, Franconi, Hollunder, Nebel, & Profitlich, 1994; Massacci, 2000). In particular, the research in modal and description logics has followed two parallel routes until the seminal work of Schild (1991), which proved that the core modal logic K_m and the core description logic $\mathcal{ALC}$ are one a notational variant of the other. Since then, analogous results have been produced for a bunch of other logics, so that, nowadays the two research lines have mostly merged

[1] This chapter is mostly based on the journal paper: Sebastiani & Vescovi, 2009a.

into one research flow.

Many approaches have been proposed for efficiently reasoning in modal and description logics, starting from the problem of checking the satisfiability in the core normal modal logic K_m and in its notational variant, the description logic $\mathcal{ALC}$ (hereafter simply "K_m" and "$\mathcal{ALC}$"). We classify them as follows.

- The "classic" *tableau-based* approach (Fitting, 1983; Baader & Hollunder, 1991; Massacci, 2000) is based on the construction of propositional tableau branches, which are recursively expanded on demand by generating successor nodes in a candidate Kripke model. KRIS (Baader & Hollunder, 1991; Baader et al., 1994), CRACK (Franconi, 1998), LWB (Balsiger, Heuerding, & Schwendimann, 1998) were among the main representative tools of this approach.

- The *DPLL-based* approach (Giunchiglia & Sebastiani, 1996, 2000) differs from the previous one mostly in the fact that a Davis-Putnam-Logemann-Loveland (DPLL) procedure, which treats the modal subformulas as propositions, is used instead of the classic propositional tableaux procedure at each nesting level of the modal operators. KSAT (Giunchiglia & Sebastiani, 1996), ESAT (Giunchiglia, Giunchiglia, & Tacchella, 2002) and *SAT (Tacchella, 1999), are the representative tools of this approach.

These two approaches merged into the "modern" tableaux-based approach, which has been extended to work with more expressive description logics and to provide more sophisticate reasoning functions. Among the tools employing this approach, we recall FACT/FACT++ and DLP (Horrocks & Patel-Schneider, 1999), and RACER (now also called RacerPro) (Haarslev & Moeller, 2001; Haarslev & Möller, 2003). [2] More recent tools which have joined the tableaux-based category are the OWL reasoners PELLET (Sirin et al., 2007) and HERMIT (Motik et al., 2009), this later implementing a novel calculus known as "hypertableaux" (Motik et al., 2007, 2009; Baumgartner et al., 2010).

Continuing with the classification other approaches to this problem are:

- In the *translational* approach (Hustadt & Schmidt, 1999; Areces, Gennari, Heguiabehere, & de Rijke, 2000) the modal formula is encoded into first-order logic (FOL), and the encoded formula can be decided efficiently by a FOL theorem prover (Areces et al., 2000). MSPASS (Hustadt, Schmidt, & Weidenbach, 1999) is the most representative tool of this approach.

- The *CSP-based approach* (Brand, Gennari, & de Rijke, 2003) differs from the tableaux-based and DPLL-based ones mostly in the fact that a CSP (Constraint Satisfaction Problem) engine is used instead of tableaux/DPLL. KCSP is the only representative tool of this approach.

[2] Notice that there is not an universal agreement on the terminology "tableaux-based" and "DPLL-based". E.g., tools like FACT/FACT++, DLP, and RACER are most often called "tableau-based", although they use a DPLL-like algorithm instead of propositional tableaux for handling the propositional component of reasoning (Horrocks, 1998; Patel-Schneider, 1998; Horrocks & Patel-Schneider, 1999; Haarslev & Moeller, 2001).

– In the *Inverse-method* approach (Voronkov, 1999, 2001), a search procedure is based on the "inverted" version of a sequent calculus (which can be seen as a modalized version of propositional resolution). KK (Voronkov, 1999) is the only representative tool of this approach.

– In the *Automata-theoretic* approach, (a symbolic representation based on BDDs – Binary Decision Diagrams – of) a tree automaton accepting all the tree models of the input formula is implicitly built and checked for emptiness (Pan et al., 2002; Pan & Vardi, 2003). KBDD (Pan & Vardi, 2003) is the only representative tool of this approach.

– Pan and Vardi (2003) presented also an encoding of K-satisfiability into QBF-satisfiability (which is PSPACE-complete too), combined with the use of a state-of-the-art QBF (Quantified Boolean Formula) solver. We call this approach *QBF-encoding* approach.

To the best of our knowledge, the last three approaches so far are restricted to the satisfiability in K_m only. The translational approach, instead, has been applied to numerous modal and description logics (e.g. traditional modal logics like T_m and $S4_m$, and dynamic modal logics) and to the relational calculus, whilst the CSP-based approach has been recently extended to KT and $S4$ by Stevenson, Britz, and Hörne (2008). [3]

A significant amount of benchmarks formulas have been produced for testing the effectiveness of the different techniques (Halpern & Moses, 1992; Giunchiglia, Roveri, & Sebastiani, 1996; Heuerding & Schwendimann, 1996; Horrocks, Patel-Schneider, & Sebastiani, 2000; Massacci, 1999; Patel-Schneider & Sebastiani, 2001, 2003).

5.2 Motivations and Goals

In this chapter we start exploring the idea of performing automated reasoning tasks in modal and description logics by encoding them into SAT, so that to be handled by state-of-the-art SAT tools. As with most previous approaches, we begin our investigation from the satisfiability in K_m, which (as discussed in the previous section) offer a very wide variety of approaches, systems and benchmark problems with which to compare.

In theory, the task may look hopeless because of worst-case complexity issues: in fact, with few exceptions, the satisfiability problem in most modal and description logics is not in NP, typically being PSPACE-complete or even harder —PSPACE-complete for K_m (Ladner, 1977; Halpern & Moses, 1992)— so that the encoding is in worst-case non polynomial(assuming NP $\neq$ PSPACE).

In practice, however, a few considerations allow for not discarding that this approach may be competitive with the state-of-the-art approaches. First, the non-polynomial bounds above are *worst-case* bounds, and formulas may have different behaviors from

[3] As far as we know, the extension of $KCSP$ of Stevenson et al. (2008) is still in form of prototype and is not yet available.

that of the pathological formulas which can be found in textbooks. (E.g., notice that the exponentiality is based on the hypothesis of unboundedness of some parameter like the modal depth; Halpern & Moses, 1992; Halpern, 1995.) Second, some tricks in the encoding may allow for reducing the size of the encoded formula significantly. Third, as the amount of RAM memory in current computers is in the order of the GBytes and current SAT solvers can successfully handle huge formulas, the encoding of many modal formulas (at least of those which are not too hard to solve also for the competitors) may be at the reach of a SAT solver. Finally, even for PSPACE-complete logics like K_m, also other state-of-the-art approaches are not guaranteed to use polynomial memory.

Content. The rest of this chapter is structured as follows. In Section 5.3 we describe the basic encoding from K_m to SAT, while we give the equivalent encoding from $\mathcal{ALC}$ in Section 5.4 (for a better readability the proof of soundness and completeness of our approach has been moved in Section 5.8 as appendix). In Section 5.5 we describe and discuss the main encoding optimizations introduced, either as preprocessing or as on-the-fly optimizations, providing many examples. In Section 5.6 we present the empirical evaluation and discuss the results; in particular, we analyze the effectiveness of the various optimizations introduced on a huge set of diversified benchmark problems, and we compare the performance of our new approach against the other approaches at the state of the art. (The results from the trials of the compared tool and some extra plots concerning the evaluation are reported in the appendix Section 5.9.) In Section 5.7 we describe the main contributions given by this first part of our research and the guidelines we have educed form our following investigation.

The basic encoding and some preliminary steps in the approach exposed in this chapter have been performed during the master thesis (Vescovi, 2006), which results have been published by (Sebastiani & Vescovi, 2006). The optimization techniques and the results of the extended empirical evaluation have been presented by (Sebastiani & Vescovi, 2009a).

5.3 The Basic Encoding: $K_m 2SAT$

We borrow some notation from the *Single Step Tableau (SST)* framework (Massacci, 2000; Donini & Massacci, 2000). We represent uniquely states in $\mathcal{M}$ as labels σ, represented as non empty sequences of integers $1.n_1^{r_1}.n_2^{r_2}. \ldots .n_k^{r_k}$, s.t. the label 1 represents the root state, and $\sigma.n^r$ represents the n-th $\mathcal{R}_r$-successor of σ (where $r \in \{1, \ldots, m\}$). With a little abuse of notation, hereafter we may say "a state σ" meaning "a state labeled by σ". We call a *labeled formula* a pair $\langle \sigma, \psi \rangle$, such that σ is a state label and ψ is a K_m-formula, and we call *labeled subformulas of* a labeled formula $\langle \sigma, \psi \rangle$ all the labeled formulas $\langle \sigma, \phi \rangle$ such that ϕ is a subformula of ψ. We conventionally write $\Box, \Diamond$ for $\Box_r, \Diamond_r$, respectively, when only one modal operator exists; accordingly, we write $\sigma.n$ for $\sigma.n^r$ in the same case.

Definition 1 ($K_m 2SAT$ encoding)**.** Let $A_{\langle\ ,\ \rangle}$ be an *injective* function which maps a labeled formula $\langle \sigma, \psi \rangle$, s.t. ψ is not in the form $\neg\phi$, into a Boolean variable $A_{\langle \sigma,\ \psi \rangle}$. We conventionally assume that $A_{\langle \sigma,\ \top \rangle}$ is $\top$ and $A_{\langle \sigma,\ \bot \rangle}$ is $\bot$. Let $L_{\langle \sigma,\ \psi \rangle}$ denote $\neg A_{\langle \sigma,\ \phi \rangle}$ if

ψ is in the form $\neg\phi$, $A_{\langle\sigma,\ \psi\rangle}$ otherwise. Given a K_m-formula φ, $K_m2SAT\varphi$ is the SAT encoding for φ and is recursively defined as the following Boolean CNF formula: [4]

$$K_m2SAT(\varphi) \stackrel{\text{def}}{=} A_{\langle 1,\ \varphi\rangle} \wedge Def(1,\ \varphi) \tag{5.1}$$

$$Def(\sigma,\ \top) \stackrel{\text{def}}{=} \top \tag{5.2}$$

$$Def(\sigma,\ \bot) \stackrel{\text{def}}{=} \top \tag{5.3}$$

$$Def(\sigma,\ A_i) \stackrel{\text{def}}{=} \top \tag{5.4}$$

$$Def(\sigma,\ \neg A_i) \stackrel{\text{def}}{=} \top \tag{5.5}$$

$$Def(\sigma,\ \alpha) \stackrel{\text{def}}{=} (L_{\langle\sigma,\ \alpha\rangle} \rightarrow (L_{\langle\sigma,\ \alpha_1\rangle} \wedge L_{\langle\sigma,\ \alpha_2\rangle})) \wedge Def(\sigma,\ \alpha_1) \wedge Def(\sigma,\ \alpha_2) \tag{5.6}$$

$$Def(\sigma,\ \beta) \stackrel{\text{def}}{=} (L_{\langle\sigma,\ \beta\rangle} \rightarrow (L_{\langle\sigma,\ \beta_1\rangle} \vee L_{\langle\sigma,\ \beta_2\rangle})) \wedge Def(\sigma,\ \beta_1) \wedge Def(\sigma,\ \beta_2) \tag{5.7}$$

$$Def(\sigma,\ \pi^{r,j}) \stackrel{\text{def}}{=} (L_{\langle\sigma,\ \pi^{r,j}\rangle} \rightarrow L_{\langle\sigma.j^r,\ \pi_0^{r,j}\rangle}) \wedge Def(\sigma.j^r,\ \pi_0^{r,j}) \tag{5.8}$$

$$Def(\sigma,\ \nu^r) \stackrel{\text{def}}{=} \bigwedge_{\substack{\text{for every}\\ \langle\sigma,\ \pi^{r,i}\rangle}} \big(((L_{\langle\sigma,\ \nu^r\rangle} \wedge L_{\langle\sigma,\ \pi^{r,i}\rangle}) \rightarrow L_{\langle\sigma.i^r,\ \nu_0^r\rangle}) \ \wedge\ Def(\sigma.i^r,\ \nu_0^r)\big). \tag{5.9}$$

Here by "$\pi^{r,j}$" we mean that $\pi^{r,j}$ is the j-th distinct π^r formula labeled by σ. $\diamond$

Notice that (5.6) and (5.7) generalize to the case of n-ary $\wedge$ and $\vee$ in the obvious way: if ϕ is $\bigotimes_{i=1}^n \phi_i$ s.t. $\bigotimes \in \{\wedge, \vee\}$, then $Def(\sigma,\ \phi) \stackrel{\text{def}}{=} (L_{\langle\sigma,\ \phi\rangle} \rightarrow \bigotimes_{i=1}^n L_{\langle\sigma,\ \phi_i\rangle}) \wedge \bigwedge_{i=1}^n Def(\sigma,\ \phi_i)$. Although conceptually trivial, this fact has an important practical consequence: in order to encode $\bigotimes_{i=1}^n \phi_i$ one needs adding only one Boolean variable rather than up to $n-1$, see Section 5.5.2. Notice also that in rule (5.9) the literals of the type $L_{\langle\sigma,\ \pi^{r,i}\rangle}$ are strictly necessary; in fact, the SAT problem must consider and encode all the possibly occurring states, but it can be the case, e.g., that a $\pi^{r,i}$ formula occurring in a disjunction is assigned to false for a particular state label σ (which, in SAT, corresponds to assign $L_{\langle\sigma,\ \pi^{r,i}\rangle}$ to false). In this situation all the labeled formulas regarding the state label $\sigma.i^r$ are useless, in particular those generated by the expansion of the ν formulas interacting with $\pi^{r,i}$. [5]

We assume that the K_m-formulas are represented as DAGs (Direct Acyclic Graphs), so that to avoid the expansion of the same $Def(\sigma,\ \psi)$ more than once. Then the various $Def(\sigma,\ \psi)$ are expanded in a breadth-first manner wrt. the tree of labels, that is, all the possible expansions for the same (newly introduced) σ are completed before starting the expansions for a different state label σ', and different state label are expanded in the order they are introduced (thus all the expansions for a given state are always handled before those of any deeper state). Moreover, following what done by Massacci (2000), we assume that, for each σ, the $Def(\sigma,\ \psi)$'s are expanded in the order: $\alpha/\beta, \pi, \nu$. Thus, each $Def(\sigma,\ \nu^r)$ is expanded after the expansion of all $Def(\sigma,\ \pi^{r,i})$'s, so that $Def(\sigma,\ \nu^r)$ will generate one clause $((L_{\langle\sigma,\ \nu^r\rangle} \wedge L_{\langle\sigma,\ \pi^{r,i}\rangle}) \rightarrow L_{\langle\sigma.i^r,\ \nu_0^r\rangle})$ and one novel definition $Def(\sigma.i^r,\ \nu_0^r)$

[4] We say that the formula is in CNF because we represent clauses as implications, according to the notation described at the beginning of Section 4.1.1.

[5] Indeed, (5.9) is a finite conjunction. In fact the number of π-subformulas is obviously finite and K_m benefits of the *finite-tree-model* property (see, e.g., Pan et al., 2002; Pan & Vardi, 2003).

for each $Def(\sigma,\ \pi^{r,i})$ expanded. [6]

Intuitively, it is easy to see that $K_m2SAT(\varphi)$ mimics the construction of an SST tableau expansion (Massacci, 2000; Donini & Massacci, 2000). We have the following fact.

Theorem 1. *A K_m-formula φ is K_m-satisfiable if and only if the corresponding Boolean formula $K_m2SAT(\varphi)$ is satisfiable.*

The complete proof of Theorem 1 can be found in Appendix 5.8.

Notice that, due to (5.9), the number of variables and clauses in $K_m2SAT(\varphi)$ may grow exponentially with $depth(\varphi)$. This is in accordance to what was stated by Halpern and Moses (1992).

Example 5.3.1 (NNF)**.** Let φ_{nnf} be $(\Diamond A_1 \vee \Diamond(A_2 \vee A_3)) \wedge \Box\neg A_1 \wedge \Box\neg A_2 \wedge \Box\neg A_3$. [7] It is easy to see that φ_{nnf} is K_1-unsatisfiable: the $\Diamond$-atoms impose that at least one atom A_i is true in at least one successor of the root state, whilst the $\Box$-atoms impose that all atoms A_i are false in all successor states of the root state. $K_m2SAT(\varphi_{nnf})$ is: [8]

1.		$A_{\langle 1,\ \varphi_{nnf}\rangle}$	*(1)*
2.	$\wedge$	$(\ A_{\langle 1,\ \varphi_{nnf}\rangle} \rightarrow (A_{\langle 1,\ \Diamond A_1 \vee \Diamond(A_2 \vee A_3)\rangle} \wedge A_{\langle 1,\ \Box\neg A_1\rangle} \wedge A_{\langle 1,\ \Box\neg A_2\rangle} \wedge A_{\langle 1,\ \Box\neg A_3\rangle})\ )$	*(6)*
3.	$\wedge$	$(\ A_{\langle 1,\ \Diamond A_1 \vee \Diamond(A_2 \vee A_3)\rangle} \rightarrow (A_{\langle 1,\ \Diamond A_1\rangle} \vee A_{\langle 1,\ \Diamond(A_2 \vee A_3)\rangle})\ )$	*(7)*
4.	$\wedge$	$(\ A_{\langle 1,\ \Diamond A_1\rangle} \rightarrow A_{\langle 1.1,\ A_1\rangle}\ )$	*(8)*
5.	$\wedge$	$(\ A_{\langle 1,\ \Diamond(A_2 \vee A_3)\rangle} \rightarrow A_{\langle 1.2,\ A_2 \vee A_3\rangle}\ )$	*(8)*
6.	$\wedge$	$(\ (A_{\langle 1,\ \Box\neg A_1\rangle} \wedge A_{\langle 1,\ \Diamond A_1\rangle}) \rightarrow \neg A_{\langle 1.1,\ A_1\rangle}\ )$	*(9)*
7.	$\wedge$	$(\ (A_{\langle 1,\ \Box\neg A_2\rangle} \wedge A_{\langle 1,\ \Diamond A_1\rangle}) \rightarrow \neg A_{\langle 1.1,\ A_2\rangle}\ )$	*(9)*
8.	$\wedge$	$(\ (A_{\langle 1,\ \Box\neg A_3\rangle} \wedge A_{\langle 1,\ \Diamond A_1\rangle}) \rightarrow \neg A_{\langle 1.1,\ A_3\rangle}\ )$	*(9)*
9.	$\wedge$	$(\ (A_{\langle 1,\ \Box\neg A_1\rangle} \wedge A_{\langle 1,\ \Diamond(A_2 \vee A_3)\rangle}) \rightarrow \neg A_{\langle 1.2,\ A_1\rangle}\ )$	*(9)*
10.	$\wedge$	$(\ (A_{\langle 1,\ \Box\neg A_2\rangle} \wedge A_{\langle 1,\ \Diamond(A_2 \vee A_3)\rangle}) \rightarrow \neg A_{\langle 1.2,\ A_2\rangle}\ )$	*(9)*
11.	$\wedge$	$(\ (A_{\langle 1,\ \Box\neg A_3\rangle} \wedge A_{\langle 1,\ \Diamond(A_2 \vee A_3)\rangle}) \rightarrow \neg A_{\langle 1.2,\ A_3\rangle}\ )$	*(9)*
12.	$\wedge$	$(\ A_{\langle 1.2,\ A_2 \vee A_3\rangle} \rightarrow (A_{\langle 1.2,\ A_2\rangle} \vee A_{\langle 1.2,\ A_3\rangle})\ )$	*(7)*

After a run of Boolean constraint propagation (BCP), 3. reduces to the implicate disjunction. If the first element $A_{\langle 1,\ \Diamond A_1\rangle}$ is assigned to true, then by BCP we have a conflict on 4. and 6. If it is set to false, then the second element $A_{\langle 1,\ \Diamond(A_2 \vee A_3)\rangle}$ is assigned to true, and by BCP we have a conflict on 12. Thus $K_m2SAT(\varphi_{nnf})$ is unsatisfiable. $\Diamond$

5.4 The Equivalent $\mathcal{ALC}$ Encoding

In this part of our work we have chosen to approach the K_m-satisfiability problem because (historically) it offers a wider set of alternative approaches and benchmark problems.

[6] In practice, even if the definition of K_m2SAT is recursive, the Def expansions are performed grouped by states. More precisely, all the $Def(\sigma.n^r,\ \psi)$ expansions, for any formula ψ, every modality index r and every defined n, are done together (in the $\alpha/\beta, \pi, \nu$ order above exposed) and necessarily after that all the $Def(\sigma,\ \varphi)$ expansions have been completed.

[7] For K_1-formulas we omit the box and diamond indexes, i.e., we write $\Box$, $\Diamond$ for $\Box_1$, $\Diamond_1$.

[8] In all examples we report at the very end of each line, i.e. after each clause, the number of the K_m2SAT encoding rule applied to generate that clause. We also drop the application of the rules (5.2), (5.3), (5.4) and (5.5).

However, since this thesis concentrates on reasoning in Description Logics and ontologies, for the sake of completeness we define an equivalent encoding for the Description Logic $\mathcal{ALC}$. The fast reader can conveniently skip the current section.

The satisfiability of a K_m formula has been proved by Schild (1991) to be equivalent to the problem of *$\mathcal{ALC}$-concept satisfiability wrt. empty TBoxes*. Similarly to what done in the previous section we represent uniquely individuals in the domain of a concept interpretation as labels σ. In order to distinguish labels referring to individuals from labels referring to modal states we represented an individual's label as non empty sequences of integers $1.r_1.n_1.r_2.n_2\ ...\ .r_k.n_k$, s.t. the label 1 represents the root individual, and $\sigma.r.n$ represents the n-th r-successor of σ (i.e. the n-th successor of σ through the role r). We call an *instantiated concept* a pair $\langle \sigma, \hat{C} \rangle$, such that σ is an individual (or its label) and $\hat{C}$ is an $\mathcal{ALC}$-concept expression. Without going into too much other details, we define $\mathcal{ALC}2SAT$ equivalently to K_m2SAT as follows:

Definition 2 ($\mathcal{ALC}2SAT$ encoding)**.** Let $A_{\langle\ ,\ \rangle}$ be an *injective* function which maps an instantiated concept $\langle \sigma, C \rangle$, s.t. C is not in the form $\neg C$, into a Boolean variable $A_{\langle \sigma,\ C \rangle}$. We conventionally assume that $A_{\langle \sigma,\ \top \rangle}$ is $\top$ and $A_{\langle \sigma,\ \bot \rangle}$ is $\bot$. Let $L_{\langle \sigma,\ C \rangle}$ denote $\neg A_{\langle \sigma,\ C \rangle}$ if C is in the form $\neg C$, $A_{\langle \sigma,\ C \rangle}$ otherwise. Given an $\mathcal{ALC}$-concept $\hat{C}$, $\mathcal{ALC}2SAT(\hat{C})$ is the SAT encoding for $\hat{C}$ and is recursively defined as the following Boolean CNF formula:

$$
\begin{aligned}
\mathcal{ALC}2SAT(\hat{C}) &\stackrel{\text{def}}{=} A_{\langle 1,\ \hat{C} \rangle} \wedge Def(1,\ \hat{C}) \\
Def(\sigma,\ \top) &\stackrel{\text{def}}{=} \top \\
Def(\sigma,\ \bot) &\stackrel{\text{def}}{=} \top \\
Def(\sigma,\ C_i) &\stackrel{\text{def}}{=} \top \\
Def(\sigma,\ \neg C_i) &\stackrel{\text{def}}{=} \top \\
Def(\sigma,\ \hat{C}_1 \sqcap \hat{C}_2) &\stackrel{\text{def}}{=} (L_{\langle \sigma,\ \hat{C}_1 \sqcap \hat{C}_2 \rangle} \rightarrow (L_{\langle \sigma,\ \hat{C}_1 \rangle} \wedge L_{\langle \sigma,\ \hat{C}_2 \rangle})) \wedge Def(\sigma,\ \hat{C}_1) \wedge Def(\sigma,\ \hat{C}_2) \\
Def(\sigma,\ \hat{C}_1 \sqcup \hat{C}_2) &\stackrel{\text{def}}{=} (L_{\langle \sigma,\ \hat{C}_1 \sqcup \hat{C}_2 \rangle} \rightarrow (L_{\langle \sigma,\ \hat{C}_1 \rangle} \vee L_{\langle \sigma,\ \hat{C}_2 \rangle})) \wedge Def(\sigma,\ \hat{C}_1) \wedge Def(\sigma,\ \hat{C}_2) \\
Def(\sigma,\ \exists r.\hat{C}_j) &\stackrel{\text{def}}{=} (L_{\langle \sigma,\ \exists r.\hat{C}_j \rangle} \rightarrow L_{\langle \sigma.r.j,\ \hat{C}_j \rangle}) \wedge Def(\sigma.r.j,\ \hat{C}_j) \\
Def(\sigma,\ \forall r.\hat{C}') &\stackrel{\text{def}}{=} \bigwedge_{\substack{\text{for every} \\ \langle \sigma,\ \exists r.\hat{C}_i \rangle}} \Big(((L_{\langle \sigma,\ \forall r.\hat{C}' \rangle} \wedge L_{\langle \sigma,\ \exists r.\hat{C}_i \rangle}) \rightarrow L_{\langle \sigma.r.i,\ \hat{C}' \rangle}) \ \wedge\ Def(\sigma.r.i,\ \hat{C}') \Big).
\end{aligned}
$$

where C_i are primitive concept names, while $\hat{C}, \hat{C}'$ and $\hat{C}_i$ are generic concept expressions. Here by "$\exists r.\hat{C}_j$" we mean that $\exists r.\hat{C}_j$ is the j-th distinct concept in the form $\exists r.\hat{C}$ (for some $\hat{C}$) labeled by σ. $\diamond$

Then the same considerations of the previous section hold also for this $\mathcal{ALC}$ encoding. In particular, it holds the following direct consequence of Theorem 1.

Corollary 2. *An $\mathcal{ALC}$-concept $\hat{C}$ is satisfiable wrt. an empty TBox if and only if the corresponding Boolean formula $\mathcal{ALC}2SAT(\hat{C})$ is satisfiable.*

5.5 Optimizations

The basic encoding of Section 5.3 is rather naive, and can be much improved to many extents, in order to reduce the size of the output propositional formula, or to make it easier to solve by DPLL, or both. We distinguish two main kinds of optimizations:
We analyze these techniques in detail.

5.5.1 Pre-conversion into BNF

Many systems use to pre-convert the input K_m-formulas into NNF (e.g., Baader et al., 1994; Massacci, 2000). In our approach, instead, we pre-convert them into BNF (like, e.g., Giunchiglia & Sebastiani, 1996; Pan et al., 2002). For our approach, the advantage of the latter representation is that, when one $\Box_r\psi$ occurs both positively and negatively (like, e.g., in $(\Box_r\psi \vee ...) \wedge (\neg\Box_r\psi \vee ...) \wedge ...$), then both occurrences of $\Box_r\psi$ are labeled by the same Boolean atom $A_{\langle\sigma,\ \Box_r\psi\rangle}$, and hence they are always assigned the same truth value by DPLL. With NNF, instead, the negative occurrence $\neg\Box_r\psi$ is rewritten into $\Diamond_r(nnf(\neg\psi))$, so that two distinct Boolean atoms $A_{\langle\sigma,\ \Box_r(nnf(\psi))\rangle}$ and $A_{\langle\sigma,\ \Diamond_r(nnf(\neg\psi))\rangle}$ are generated; DPLL can assign them the same truth value, creating a hidden conflict which may require some extra Boolean search to reveal. [9]

Example 5.5.1 (BNF)**.** We consider the BNF variant of the φ_{nnf} formula of Example 5.3.1, $\varphi_{bnf} = (\neg\Box\neg A_1 \vee \neg\Box(\neg A_2 \wedge \neg A_3)) \wedge \Box\neg A_1 \wedge \Box\neg A_2 \wedge \Box\neg A_3$. As before, it is easy to see that φ_{bnf} is K_1-unsatisfiable. $K_m2SAT(\varphi_{bnf})$ is: [10]

1. $A_{\langle 1,\ \varphi_{bnf}\rangle}$ *(1)*
2. $\wedge\ (\ A_{\langle 1,\ \varphi_{bnf}\rangle} \rightarrow (A_{\langle 1,\ (\neg\Box\neg A_1 \vee \neg\Box(\neg A_2 \wedge \neg A_3))\rangle} \wedge A_{\langle 1,\ \Box\neg A_1\rangle} \wedge A_{\langle 1,\ \Box\neg A_2\rangle} \wedge A_{\langle 1,\ \Box\neg A_3\rangle}))$ *(6)*
3. $\wedge\ (\ A_{\langle 1,\ (\neg\Box\neg A_1 \vee \neg\Box(\neg A_2 \wedge \neg A_3))\rangle} \rightarrow (\neg A_{\langle 1,\ \Box\neg A_1\rangle} \vee \neg A_{\langle 1,\ \Box(\neg A_2 \wedge \neg A_3)\rangle})\ )$ *(7)*
4. $\wedge\ (\ \neg A_{\langle 1,\ \Box\neg A_1\rangle} \rightarrow A_{\langle 1.1,\ A_1\rangle}\ )$ *(8)*
5. $\wedge\ (\ \neg A_{\langle 1,\ \Box(\neg A_2 \wedge \neg A_3)\rangle} \rightarrow \neg A_{\langle 1.2,\ (\neg A_2 \wedge \neg A_3)\rangle}\ )$ *(8)*
6. $\wedge\ (\ (A_{\langle 1,\ \Box\neg A_1\rangle} \wedge \neg A_{\langle 1,\ \Box\neg A_1\rangle}) \rightarrow \neg A_{\langle 1.1,\ A_1\rangle}\ )$ *(9)*
7. $\wedge\ (\ (A_{\langle 1,\ \Box\neg A_2\rangle} \wedge \neg A_{\langle 1,\ \Box\neg A_1\rangle}) \rightarrow \neg A_{\langle 1.1,\ A_2\rangle}\ )$ *(9)*
8. $\wedge\ (\ (A_{\langle 1,\ \Box\neg A_3\rangle} \wedge \neg A_{\langle 1,\ \Box\neg A_1\rangle}) \rightarrow \neg A_{\langle 1.1,\ A_3\rangle}\ )$ *(9)*
9. $\wedge\ (\ (A_{\langle 1,\ \Box\neg A_1\rangle} \wedge \neg A_{\langle 1,\ \Box(\neg A_2 \wedge \neg A_3)\rangle}) \rightarrow \neg A_{\langle 1.2,\ A_1\rangle}\ )$ *(9)*
10. $\wedge\ (\ (A_{\langle 1,\ \Box\neg A_2\rangle} \wedge \neg A_{\langle 1,\ \Box(\neg A_2 \wedge \neg A_3)\rangle}) \rightarrow \neg A_{\langle 1.2,\ A_2\rangle}\ )$ *(9)*
11. $\wedge\ (\ (A_{\langle 1,\ \Box\neg A_3\rangle} \wedge \neg A_{\langle 1,\ \Box(\neg A_2 \wedge \neg A_3)\rangle}) \rightarrow \neg A_{\langle 1.2,\ A_3\rangle}\ )$ *(9)*
12. $\wedge\ (\ \neg A_{\langle 1.2,\ (\neg A_2 \wedge \neg A_3)\rangle} \rightarrow (A_{\langle 1.2,\ A_2\rangle} \vee A_{\langle 1.2,\ A_3\rangle})\ )$ *(7)*

Unlike with the NNF formula φ_{nnf} in Example 5.3.1, $K_m2SAT(\varphi_{bnf})$ is found unsatisfiable directly by BCP. In fact, the unit-propagation of $A_{\langle 1,\ \Box\neg A_1\rangle}$ from 2. causes $\neg A_{\langle 1,\ \Box\neg A_1\rangle}$ in 3. to be false, so that one of the two (unsatisfiable) branches induced by the disjunction is cut a priori. With φ_{nnf}, K_m2SAT does not recognize $\Box\neg A_1$ and $\Diamond A_1$ to be one the

[9] Notice that this consideration holds for every representation involving both boxes and diamonds; we refer to NNF simply because it is the most popular of these representations.

[10] Notice that the valid clause 6. can be dropped. See the explanation in Section 5.5.5.

negation of the other, so that two distinct atoms $A_{\langle 1,\ \Box\neg A_1\rangle}$ and $A_{\langle 1,\ \Diamond A_1\rangle}$ are generated. Hence $A_{\langle 1,\ \Box\neg A_1\rangle}$ and $A_{\langle 1,\ \Diamond A_1\rangle}$ cannot be recognized by DPLL to be one the negation of the other, s.t. DPLL may need exploring one Boolean branch more. $\Diamond$

In the following we will assume the formulas are in BNF (although most of the optimizations which follow work also for other representations).

5.5.2 Normalization of Modal Atoms

One potential source of inefficiency for DPLL-based procedures is the occurrence in the input formula of semantically-equivalent though syntactically-different modal atoms ψ' and ψ'' (e.g., $\Box_1(A_1 \vee A_2)$ and $\Box_1(A_2 \vee A_1)$), which are not recognized as such by $K_m 2SAT$. This causes the introduction of duplicated Boolean atoms $A_{\langle \sigma,\ \psi'\rangle}$ and $A_{\langle \sigma,\ \psi''\rangle}$ and — much worse— of duplicated subformulas $Def(\sigma,\ \psi')$ and $Def(\sigma,\ \psi'')$. This fact can have very negative consequences, in particular when ψ' and ψ'' occur with negative polarity, because this causes the creation of distinct versions of the same successor states, and the duplication of whole parts of the output formula.

Example 5.5.2. Consider the K_m-formula $(\phi_1 \vee \neg\Box_1(A_2 \vee A_1)) \wedge (\phi_2 \vee \neg\Box_1(A_1 \vee A_2)) \wedge \phi_3$, s.t. ϕ_1, ϕ_2, ϕ_3 are possibly-big K_m-formulas. Then $K_m 2SAT$ creates two distinct atoms $A_{\langle 1,\ \neg\Box_1(A_2 \vee A_1)\rangle}$ and $A_{\langle 1,\ \neg\Box_1(A_1 \vee A_2)\rangle}$ and two distinct formulas $Def(1,\ \neg\Box_1(A_2 \vee A_1))$ and $Def(1,\ \neg\Box_1(A_1 \vee A_2))$. The latter will cause the creation of two distinct states 1.1 and 1.2. Thus, the recursive expansion of all $\Box_1$-formulas occurring positively in ϕ_1, ϕ_2, ϕ_3 will be duplicated for these two states. $\Diamond$

In order to cope with this problem, as done by Giunchiglia and Sebastiani (1996), we apply some normalization steps to modal atoms with the intent of rewriting as many as possible syntactically-different but semantically-equivalent modal atoms into syntactically-identical ones. This can be achieved by a recursive application of some simple validity-preserving rewriting rules.

Sorting: modal atoms are internally sorted according to some criterion, so that atoms which are identical modulo reordering are rewritten into the same atom (e.g., $\Box_i(\varphi_2 \vee \varphi_1)$ and $\Box_i(\varphi_1 \vee \varphi_2)$ are both rewritten into $\Box_i(\varphi_1 \vee \varphi_2)$).

Flattening: the associativity of $\wedge$ and $\vee$ is exploited and combinations of $\wedge$'s or $\vee$'s are "flattened" into n-ary $\wedge$'s or $\vee$'s respectively (e.g., $\Box_i(\varphi_1 \vee (\varphi_2 \vee \varphi_3))$ and $\Box_i((\varphi_1 \vee \varphi_2) \vee \varphi_3)$ are both rewritten into $\Box_i(\varphi_1 \vee \varphi_2 \vee \varphi_3)$).

Flattening has also the advantage of reducing the number of novel atoms introduced in the encoding, as a consequence of the fact noticed in Section 5.3. One possible drawback of this technique is that it can reduce the sharing of subformulas (e.g., with $\Box_i((\varphi_1 \vee \varphi_2) \vee \varphi_3)$ and $\Box_i((\varphi_1 \vee \varphi_2) \vee \varphi_4)$, the common part is no more shared). However, we have empirically experienced that this drawback is negligible wrt. the advantages of flattening.

5.5.3 Box Lifting

As second preprocessing the K_m-formula can also be rewritten by recursively applying the K_m-validity-preserving "box lifting rules":

$$(\Box_r\varphi_1 \wedge \Box_r\varphi_2) \implies \Box_r(\varphi_1 \wedge \varphi_2), \quad (\neg\Box_r\varphi_1 \vee \neg\Box_r\varphi_2) \implies \neg\Box_r(\varphi_1 \wedge \varphi_2). \tag{5.10}$$

This has the potential benefit of reducing the number of π^r formulas, and hence the number of labels $\sigma.i^r$ to take into account in the expansion of the $Def(\sigma,\ \nu^r)$'s (5.9). We call *lifting* this preprocessing.

Example 5.5.3 (Box lifting)**.** If we apply the rules (5.10) to the formula of Example 5.5.1, then we have $\varphi_{bnflift} = \neg\Box(\neg A_1 \wedge \neg A_2 \wedge \neg A_3) \wedge \Box(\neg A_1 \wedge \neg A_2 \wedge \neg A_3)$. Consequently, $K_m2SAT(\varphi_{bnflift})$ is:

$$\begin{array}{lll}
1. & & A_{\langle 1,\ \varphi_{bnflift}\rangle} & (1)\\
2. & \wedge & (\ A_{\langle 1,\ \varphi_{bnflift}\rangle} \rightarrow (\neg A_{\langle 1,\ \Box(\neg A_1\wedge\neg A_2\wedge\neg A_3)\rangle} \wedge A_{\langle 1,\ \Box(\neg A_1\wedge\neg A_2\wedge\neg A_3)\rangle})\ ) & (6)\\
3. & \wedge & (\ \neg A_{\langle 1,\ \Box(\neg A_1\wedge\neg A_2\wedge\neg A_3)\rangle} \rightarrow \neg A_{\langle 1.1,\ (\neg A_1\wedge\neg A_2\wedge\neg A_3)\rangle}\ ) & (8)\\
4. & \wedge & ((\ A_{\langle 1,\ \Box(\neg A_1\wedge\neg A_2\wedge\neg A_3)\rangle} \wedge \neg A_{\langle 1,\ \Box(\neg A_1\wedge\neg A_2\wedge\neg A_3)\rangle}) \rightarrow A_{\langle 1.1,\ (\neg A_1\wedge\neg A_2\wedge\neg A_3)\rangle}\ ) & (9)\\
5. & \wedge & (\ \neg A_{\langle 1.1,\ (\neg A_1\wedge\neg A_2\wedge\neg A_3)\rangle} \rightarrow (A_{\langle 1.1,\ A_1\rangle} \vee A_{\langle 1.1,\ A_2\rangle} \vee A_{\langle 1.1,\ A_3\rangle})\ ) & (7)\\
6. & \wedge & (\ A_{\langle 1.1,\ (\neg A_1\wedge\neg A_2\wedge\neg A_3)\rangle} \rightarrow (\neg A_{\langle 1.1,\ A_1\rangle} \wedge \neg A_{\langle 1.1,\ A_2\rangle} \wedge \neg A_{\langle 1.1,\ A_3\rangle})\ ). & (6)
\end{array}$$

$K_m2SAT(\varphi_{bnflift})$ is found unsatisfiable directly by BCP on clauses 1. and 2.. Only one successor state (1.1) is considered. Notice that 3., 4., 5. and 6. are redundant, because 1. and 2. alone are unsatisfiable. [11] ◇

5.5.4 Controlled Box Lifting

One potential drawback of applying the lifting rules is that, by collapsing the formula $(\Box_r\varphi_1 \wedge \Box_r\varphi_2)$ into $\Box_r(\varphi_1 \wedge \varphi_2)$ and $(\neg\Box_r\varphi_1 \vee \neg\Box_r\varphi_2)$ into $\neg\Box_r(\varphi_1 \wedge \varphi_2)$, the possibility of sharing box subformulas in the DAG representation of the input K_m-formula is reduced.

In order to cope with this problem we provide an alternative policy for applying box lifting, that is, to apply the rules (5.10) only when neither box subformula occurring in the implicant in (5.10) has multiple occurrences. We call this policy *controlled box lifting.*

Example 5.5.4 (Controlled Box Lifting)**.** We apply *Controlled Box Lifting* to the formula of Example 5.5.1, then we have $\varphi_{bnfclift} = (\neg\Box\neg A_1 \vee \neg\Box(\neg A_2 \wedge \neg A_3)) \wedge \Box\neg A_1 \wedge \Box(\neg A_2 \wedge \neg A_3)$ since the rules (5.10) are applied among all the box subformulas except for $\Box\neg A_1$,

[11] In our actual implementation, trivial cases like $\varphi_{bnflift}$ are found to be unsatisfiable directly during the construction of the DAG representations, so their encoding is never generated.

which is shared. It follows that $K_m2SAT(\varphi_{bnfclift})$ is:

$$
\begin{array}{lll}
1. & & A_{\langle 1,\ \varphi_{bnfclift}\rangle} & (1)\\
2. & \wedge & (\ A_{\langle 1,\ \varphi_{bnfclift}\rangle} \rightarrow (A_{\langle 1,\ (\neg\Box\neg A_1 \vee \neg\Box(\neg A_2 \wedge \neg A_3))\rangle} \wedge A_{\langle 1,\ \Box\neg A_1\rangle} \wedge A_{\langle 1,\ \Box(\neg A_2 \wedge \neg A_3)\rangle}\ ) & (6)\\
3. & \wedge & (\ A_{\langle 1,\ (\neg\Box\neg A_1 \vee \neg\Box(\neg A_2 \wedge \neg A_3))\rangle} \rightarrow (\neg A_{\langle 1,\ \Box\neg A_1\rangle} \vee \neg A_{\langle 1,\ \Box(\neg A_2 \wedge \neg A_3)\rangle})\ ) & (7)\\
4. & \wedge & (\ \neg A_{\langle 1,\ \Box\neg A_1\rangle} \rightarrow A_{\langle 1.1,\ A_1\rangle}\ ) & (8)\\
5. & \wedge & (\ \neg A_{\langle 1,\ \Box(\neg A_2 \wedge \neg A_3)\rangle} \rightarrow \neg A_{\langle 1.2,\ (\neg A_2 \wedge \neg A_3)\rangle}\ ) & (8)\\
6. & \wedge & (\ (A_{\langle 1,\ \Box\neg A_1\rangle} \wedge \neg A_{\langle 1,\ \Box\neg A_1\rangle}) \rightarrow \neg A_{\langle 1.1,\ A_1\rangle}\ ) & (9)\\
7. & \wedge & (\ (A_{\langle 1,\ \Box(\neg A_2 \wedge \neg A_3)\rangle} \wedge \neg A_{\langle 1,\ \Box\neg A_1\rangle}) \rightarrow A_{\langle 1.1,\ (\neg A_2 \wedge \neg A_3)\rangle}\ ) & (9)\\
8. & \wedge & (\ (A_{\langle 1,\ \Box\neg A_1\rangle} \wedge \neg A_{\langle 1,\ \Box(\neg A_2 \wedge \neg A_3)\rangle}) \rightarrow \neg A_{\langle 1.2,\ A_1\rangle}\ ) & (9)\\
9. & \wedge & (\ (A_{\langle 1,\ \Box(\neg A_2 \wedge \neg A_3)\rangle} \wedge \neg A_{\langle 1,\ \Box(\neg A_2 \wedge \neg A_3)\rangle}) \rightarrow A_{\langle 1.2,\ (\neg A_2 \wedge \neg A_3)\rangle}\ ) & (9)\\
10. & \wedge & (\ A_{\langle 1.1,\ (\neg A_2 \wedge \neg A_3)\rangle} \rightarrow (\neg A_{\langle 1.1,\ A_2\rangle} \wedge \neg A_{\langle 1.1,\ A_3\rangle})\ ) & (6)\\
11. & \wedge & (\ \neg A_{\langle 1.2,\ (\neg A_2 \wedge \neg A_3)\rangle} \rightarrow (A_{\langle 1.2,\ A_2\rangle} \vee A_{\langle 1.2,\ A_3\rangle})\ ) & (7)\\
12. & \wedge & (\ A_{\langle 1.2,\ (\neg A_2 \wedge \neg A_3)\rangle} \rightarrow (\neg A_{\langle 1.2,\ A_2\rangle} \wedge \neg A_{\langle 1.2,\ A_3\rangle})\ ) & (6)
\end{array}
$$

$K_m2SAT(\varphi_{bnfclift})$ is found unsatisfiable directly by BCP on clauses 1., 2. and 3.. Notice that the unit propagation of $A_{\langle 1,\ \Box\neg A_1\rangle}$ and $A_{\langle 1,\ \Box(\neg A_2 \wedge \neg A_3)\rangle}$ from 2. causes the implicate disjunction in 3. to be false. ◇

5.5.5 On-the-fly Boolean Simplification and Truth Propagation

A first straightforward on-the-fly optimization is that of applying recursively the standard rewriting rules for the Boolean simplification of the formula like, e.g.,

$$
\begin{array}{llll}
\langle\sigma,\varphi\rangle \wedge \langle\sigma,\varphi\rangle & \Longrightarrow \langle\sigma,\varphi\rangle, & \langle\sigma,\varphi\rangle \vee \langle\sigma,\varphi\rangle & \Longrightarrow \langle\sigma,\varphi\rangle,\\
\langle\sigma,\varphi_1\rangle \wedge \langle\sigma,(\varphi_1 \vee \varphi_2)\rangle & \Longrightarrow \langle\sigma,\varphi_1\rangle, & \langle\sigma,\varphi_1\rangle \vee \langle\sigma,(\varphi_1 \wedge \varphi_2)\rangle & \Longrightarrow \langle\sigma,\varphi_1\rangle,\\
\langle\sigma,\varphi\rangle \wedge \neg\langle\sigma,\varphi\rangle & \Longrightarrow \langle\sigma,\bot\rangle, & \langle\sigma,\varphi\rangle \vee \neg\langle\sigma,\varphi\rangle & \Longrightarrow \langle\sigma,\top\rangle,\\
\ldots, & & &
\end{array}
$$

and for the propagation of truth/falsehood through Boolean operators like, e.g.,

$$
\begin{array}{llll}
\neg\langle\sigma,\bot\rangle & \Longrightarrow \langle\sigma,\top\rangle, & \neg\langle\sigma,\top\rangle & \Longrightarrow \langle\sigma,\bot\rangle,\\
\langle\sigma,\varphi\rangle \wedge \langle\sigma,\top\rangle & \Longrightarrow \langle\sigma,\varphi\rangle, & \langle\sigma,\varphi\rangle \wedge \langle\sigma,\bot\rangle & \Longrightarrow \langle\sigma,\bot\rangle,\\
\langle\sigma,\varphi\rangle \vee \langle\sigma,\top\rangle & \Longrightarrow \langle\sigma,\top\rangle, & \langle\sigma,\varphi\rangle \vee \langle\sigma,\bot\rangle & \Longrightarrow \langle\sigma,\varphi\rangle,\\
\ldots. & & &
\end{array}
$$

Example 5.5.5. If we consider the K_m-formula $\varphi_{bnflift} = \neg\Box(\neg A_1 \wedge \neg A_2 \wedge \neg A_3) \wedge \Box(\neg A_1 \wedge \neg A_2 \wedge \neg A_3)$ of Example 5.5.3 and we apply the Boolean simplification rule $\langle\sigma,\varphi\rangle \wedge \neg\langle\sigma,\varphi\rangle \Longrightarrow \langle\sigma,\bot\rangle$, then $\langle\sigma,\varphi_{bnflift}\rangle$ is simplified into $\langle\sigma,\bot\rangle$. ◇

One important subcase of on-the-fly Boolean simplification avoids the useless encoding of incompatible π^r and ν^r formulas. In BNF, in fact, the same subformula $\Box_r\psi$ may occur in the same state σ both positively and negatively (like $\pi^r = \neg\Box_r\psi$ and $\nu^r = \Box_r\psi$). If so, K_m2SAT labels both those occurrences of $\Box_r\psi$ with the same Boolean atom $A_{\langle\sigma,\ \Box_r\psi\rangle}$, and produces recursively two distinct subsets of clauses in the encoding, by applying (5.8) to

$\neg\Box_r\psi$ and (5.9) to $\Box_r\psi$ respectively. However, the latter step (5.9) generates a *valid* clause $(A_{\langle\sigma,\ \Box_r\psi\rangle} \wedge \neg A_{\langle\sigma,\ \Box_r\psi\rangle}) \rightarrow A_{\langle\sigma.i,\ \psi\rangle}$, so that we can avoid generating it. Consequently, if $A_{\langle\sigma.i^r,\ \psi\rangle}$ no more occurs in the formula, then $Def(\sigma.i^r,\ \psi)$ should not be generated, as there is no more need of defining $\langle\sigma.i^r, \psi\rangle$. [12]

Example 5.5.6. If we apply this observation in the construction of the formulas of Examples 5.5.1 and 5.5.4, we have the following facts:

- In the formula $K_m2SAT(\varphi_{bnf})$ of Example 5.5.1, clause 6. is valid and thus it is dropped.
- In the formula $K_m2SAT(\varphi_{bnfclift})$ of Example 5.5.4, both valid clauses 6. and 9. are dropped, so that 12. is not generated. ◇

Hereafter we assume that on-the-fly Boolean simplification is applied also in combination with the techniques described in the next sections.

5.5.6 On-the-fly Truth Propagation Through Modal Operators

Truth and falsehood —which can derive by the application of the techniques in Section 5.5.5, Section 5.5.7 and Section 5.5.8— may be propagated on-the-fly also though modal operators. First, for every σ, both positive and negative instances of $\langle\sigma, \Box_r\top\rangle$ can be safely simplified by applying the rewriting rule $\langle\sigma, \Box_r\top\rangle \Longrightarrow \langle\sigma, \top\rangle$.

Second, we notice the following fact. When we have a positive occurrence of $\langle\sigma, \neg\Box_r\bot\rangle$ for some σ (we suppose wlog. that we have only that π^r-formula for σ), [13] by definition of (5.8) and (5.9) we have

$$Def(\sigma,\ \neg\Box_r\bot) \ = \ (L_{\langle\sigma,\ \neg\Box_r\bot\rangle} \rightarrow A_{\langle\sigma.j^r,\ \top\rangle}) \wedge Def(\sigma.j^r,\ \top), \tag{5.11}$$

$$Def(\sigma,\ \Box_r\psi) \ = \ ((L_{\langle\sigma,\ \Box_r\psi\rangle} \wedge L_{\langle\sigma,\ \neg\Box_r\bot\rangle}) \rightarrow L_{\langle\sigma.j^r,\ \psi\rangle}) \wedge Def(\sigma.j^r,\ \psi) \tag{5.12}$$

for some new label $\sigma.j^r$ and for every $\Box_r\psi$ occurring positively in σ. $Def(\sigma,\ \neg\Box_r\bot)$ reduces to $\top$ because both $A_{\langle\sigma.j^r,\ \top\rangle}$ and $Def(\sigma.j^r,\ \top)$ reduce to $\top$. If at least another distinct π-formula $\neg\Box_r\varphi$ occurs positively in σ, however, there is no need for the $\sigma.j^r$ label in (5.11) and (5.12) to be a *new* label, and we can re-use instead the label $\sigma.i^r$ introduced in the expansion of $Def(\sigma,\ \neg\Box_r\varphi)$, as follows:

$$Def(\sigma,\ \neg\Box_r\varphi) \ = \ (L_{\langle\sigma,\ \neg\Box_r\varphi\rangle} \rightarrow L_{\langle\sigma.i^r,\ \neg\varphi\rangle}) \wedge Def(\sigma.i^r,\ \neg\varphi). \tag{5.13}$$

Thus (5.11) is dropped and, for every $\langle\sigma, \Box_r\psi\rangle$ occurring positively, we write:

$$Def(\sigma,\ \Box_r\psi) \ = \ ((L_{\langle\sigma,\ \Box_r\psi\rangle} \wedge L_{\langle\sigma,\ \neg\Box_r\bot\rangle}) \rightarrow L_{\langle\sigma.i^r,\ \psi\rangle}) \wedge Def(\sigma.i^r,\ \psi) \tag{5.14}$$

instead of (5.12). (Notice the label $\sigma.i^r$ introduced in (5.13) rather than the label $\sigma.j^r$ of (5.11).)

[12] Here the "if" is due to the fact that it may be the case that $A_{\langle\sigma.i^r,\ \psi\rangle}$ is generated anyway from the expansion of some other subformula, like, e.g., $\Box_r(\psi \vee \phi)$. If this is the case, $Def(\sigma.i^r,\ \psi)$ must be generated anyway.

[13] E.g., $\neg\Box_r\bot$ may result from applying the steps of Section 5.5.1 and of Section 5.5.5 to $\neg\Box_r(\Box_r A_1 \wedge \Diamond_r\neg A_1)$.

This is motivated by the fact that $Def(\sigma,\ \neg\Box_r\bot)$ forces the existence of at least one successor of σ but imposes no constraints on which formulas should hold there, so that we can use some other already-defined successor state, if any. This fact has the important benefit of eliminating useless successor states from the encoding.

Example 5.5.7. Let φ be the BNF K-formula:

$$(\neg A_1 \vee \neg\Box A_2) \wedge (A_1 \vee \neg\Box\bot) \wedge (\neg A_1 \vee A_3) \wedge (\neg A_1 \vee \neg A_3) \wedge (A_1 \vee \Box\neg A_4) \wedge \Box A_4.$$

φ is K-inconsistent, because the only possible assignment is $\{\neg A_1, \neg\Box\bot, \Box\neg A_4, \Box A_4\}$, which is K-inconsistent. $K_m2SAT(\varphi)$ is encoded as follows:

1.		$A_{\langle 1,\ \varphi\rangle}$	*(1)*
2.	$\wedge$	$(A_{\langle 1,\ \varphi\rangle} \rightarrow (A_{\langle 1,\ (\neg A_1\vee\neg\Box A_2)\rangle} \wedge A_{\langle 1,\ (A_1\vee\neg\Box\bot)\rangle} \wedge A_{\langle 1,\ (\neg A_1\vee A_3)\rangle} \wedge A_{\langle 1,\ (A_1\vee\Box\neg A_4)\rangle} \wedge A_{\langle 1,\ \Box A_4\rangle}))$	*(6)*
3.	$\wedge$	$(A_{\langle 1,\ (\neg A_1\vee\neg\Box A_2)\rangle} \rightarrow (\neg A_{\langle 1,\ A_1\rangle} \vee \neg A_{\langle 1,\ \Box A_2\rangle}))$	*(7)*
4.	$\wedge$	$(A_{\langle 1,\ (A_1\vee\neg\Box\bot)\rangle} \rightarrow (A_{\langle 1,\ A_1\rangle} \vee \neg A_{\langle 1,\ \Box\bot\rangle}))$	*(7)*
5.	$\wedge$	$(A_{\langle 1,\ (\neg A_1\vee A_3)\rangle} \rightarrow (\neg A_{\langle 1,\ A_1\rangle} \vee A_{\langle 1,\ A_3\rangle}))$	*(7)*
6.	$\wedge$	$(A_{\langle 1,\ (\neg A_1\vee\neg A_3)\rangle} \rightarrow (\neg A_{\langle 1,\ A_1\rangle} \vee \neg A_{\langle 1,\ A_3\rangle}))$	*(7)*
7.	$\wedge$	$(A_{\langle 1,\ (A_1\vee\Box\neg A_4)\rangle} \rightarrow (A_{\langle 1,\ A_1\rangle} \vee A_{\langle 1,\ \Box\neg A_4\rangle}))$	*(7)*
8.	$\wedge$	$(\neg A_{\langle 1,\ \Box A_2\rangle} \rightarrow \neg A_{\langle 1.1,\ A_2\rangle})$	*(8)*
9.	$\wedge$	$((A_{\langle 1,\ \Box\neg A_4\rangle} \wedge \neg A_{\langle 1,\ \Box A_2\rangle}) \rightarrow \neg A_{\langle 1.1,\ A_4\rangle})$	*(9)*
10.	$\wedge$	$((A_{\langle 1,\ \Box A_4\rangle} \wedge \neg A_{\langle 1,\ \Box A_2\rangle}) \rightarrow A_{\langle 1.1,\ A_4\rangle})$	*(9)*
11.	$\wedge$	$(\neg A_{\langle 1,\ \Box\bot\rangle} \rightarrow \neg A_{\langle 1.1,\ \bot\rangle})$	*(8)*
12.	$\wedge$	$((A_{\langle 1,\ \Box\neg A_4\rangle} \wedge \neg A_{\langle 1,\ \Box\bot\rangle}) \rightarrow \neg A_{\langle 1.1,\ A_4\rangle})$	*(9)*
13.	$\wedge$	$((A_{\langle 1,\ \Box A_4\rangle} \wedge \neg A_{\langle 1,\ \Box\bot\rangle}) \rightarrow A_{\langle 1.1,\ A_4\rangle})$	*(9)*

Clause 11. is then simplified into $\top$. (In a practical implementation it is not even generated.) Notice that in clauses 11., 12. and 13. it is used the label 1.1 of clauses 8., 9. and 10. rather than a new label 1.2. Thus, only one successor label is generated.

When DPLL is run on $K_m2SAT(\varphi)$, by BCP 1. and 2. are immediately satisfied and the implicants are removed from 3., 4., 5., 6.. Thanks to 5. and 6., $A_{\langle 1,\ A_1\rangle}$ can be assigned only to false, which causes 3. to be satisfied and forces the assignment of the literals $\neg A_{\langle 1,\ \Box\bot\rangle}$, $A_{\langle 1,\ \Box\neg A_4\rangle}$ by BCP on 3. and 7. and hence of $\neg A_{\langle 1.1,\ \bot\rangle}$, $\neg A_{\langle 1.1,\ A_4\rangle}$ and $A_{\langle 1.1,\ A_4\rangle}$ by BCP on 12. and 13., causing a contradiction. $\diamond$

It is worth noticing that (5.14) is strictly necessary for the correctness of the encoding even when another π-formula occurs in σ. (E.g., in Example 5.5.7, without 12. and 13. the formula $K_m2SAT(\varphi)$ would become satisfiable because $A_{\langle 1,\ \Box A_2\rangle}$ could be safely be assigned to true by DPLL, which would satisfy 8., 9. and 10..)

Hereafter we assume that this technique is applied also in combination with the techniques described in Section 5.5.5 and in the next sections.

5.5.7 On-the-fly Pure-Literal Reduction

Another technique, evolved from that proposed by Pan and Vardi (2003), applies Pure-Literal Reduction (PLR) on-the-fly during the construction of $K_m2SAT(\varphi)$. When for a

label σ all the clauses containing atoms in the form $A_{\langle \sigma,\ \psi\rangle}$ have been generated, if some of them occurs only positively [resp. negatively], then it can be safely assigned to true [resp. to false], and hence the clauses containing $A_{\langle \sigma,\ \psi\rangle}$ can be dropped. [14] As a consequence, some other atom $A_{\langle \sigma,\ \psi'\rangle}$ can become pure, so that the process is repeated until a fixpoint is reached.

Example 5.5.8. Consider the formula φ_{bnf} of Example 5.5.1. During the construction of $K_m2SAT(\varphi_{bnf})$, after 1.-8. are generated, no more clause containing atoms in the form $A_{\langle 1.1,\ \psi\rangle}$ is to be generated. Then we notice that $A_{\langle 1.1,\ A_2\rangle}$ and $A_{\langle 1.1,\ A_3\rangle}$ occur only negatively, so that they can be safely assigned to false. Therefore, 7. and 8. can be safely dropped. Same discourse applies lately to $A_{\langle 1.2,\ A_1\rangle}$ and 9.. The resulting formula is found inconsistent by BCP. (In fact, notice from Example 5.5.1 that the atoms $A_{\langle 1.1,\ A_2\rangle}$, $A_{\langle 1.1,\ A_3\rangle}$, and $A_{\langle 1.2,\ A_1\rangle}$ play no role in the unsatisfiability of $K_m2SAT(\varphi_{bnf})$.) $\diamond$

We remark the differences between PLR and the Pure-Literal Reduction technique proposed by Pan and Vardi (2003). In KBDD (Pan et al., 2002; Pan & Vardi, 2003), the Pure-Literal Reduction is a preprocessing step which is applied to the input modal formula, either at global level (i.e. looking for pure-polarity primitive propositions for the whole formula) or, more effectively, at different modal depths (i.e. looking for pure-polarity primitive propositions for the subformulas at the same nesting level of modal operators).

Our technique is much more fine-grained, as PLR is applied on-the-fly with a single-state granularity, obtaining a much stronger reduction effect.

Example 5.5.9. Consider again the BNF K_m-formula φ_{bnf} discussed in Examples 5.5.1 and 5.5.8: $\varphi_{bnf} = (\neg\Box\neg A_1 \vee \neg\Box(\neg A_2 \wedge \neg A_3)) \wedge \Box\neg A_1 \wedge \Box\neg A_2 \wedge \Box\neg A_3$. It is immediate to see that all primitive propositions A_1, A_2, A_3 occur at every modal depth with both polarities, so that the technique of Pan and Vardi (2003) produces no effect on this formula. $\diamond$

5.5.8 On-the-fly Boolean Constraint Propagation

One major problem of the basic encoding of Definition 1 (Section 5.3) is that it is "purely-syntactic", that is, it does not consider the possible truth values of the subformulas, and the effect of their propagation through the Boolean and modal connectives. In particular, K_m2SAT applies (5.8) [resp. (5.9)] to *every* π-subformula [resp. ν-subformula], regardless the fact that the truth values which can be deterministically assigned to the labeled subformulas of $\langle 1, \varphi\rangle$ may allow for dropping some labeled π-/ν-subformulas, and thus prevent the need of encoding them.

One solution to this problem is that of applying Boolean Constraint Propagation (BCP) on-the-fly during the construction of $K_m2SAT(\varphi)$, starting from the fact that $A_{\langle 1,\ \varphi\rangle}$ must be true. If a contradiction is found, then $K_m2SAT(\varphi)$ is unsatisfiable, so

[14] In our actual implementation this reduction is performed directly within an intermediate data structure, so that these clauses are never generated.

that the formula is not expanded any further, and the encoder returns the formula "$\bot$". [15] When BCP allows for dropping one implication in (5.6)-(5.9) without assigning some of its implicate literals, namely $L_{\langle \sigma,\ \psi_i \rangle}$, then $\langle \sigma, \psi_i \rangle$ needs not to be defined, so that $Def(\sigma,\ \psi_i)$ must not be expanded. [16] Importantly, dropping $Def(\sigma,\ \pi^{r,j})$ for some π-formula $\langle \sigma, \pi^{r,j} \rangle$ prevents generating the label $\sigma.j^r$ (5.8) and all its successor labels $\sigma.j^r.\sigma'$ (corresponding to the subtree of states rooted in $\sigma.j^r$), so that all the corresponding labeled subformulas are not encoded.

Example 5.5.10. Consider Example 5.5.1, and suppose we apply on-the-fly BCP. During the construction of 1., 2. and 3. in $K_m2SAT(\varphi_{bnf})$, the atoms $A_{\langle 1,\ \varphi_{bnf} \rangle}$, $A_{\langle 1,\ (\neg\Box\neg A_1 \vee \neg\Box(\neg A_2 \wedge \neg A_3)) \rangle}$, $A_{\langle 1,\ \Box\neg A_1 \rangle}$, $A_{\langle 1,\ \Box\neg A_2 \rangle}$ and $A_{\langle 1,\ \Box\neg A_3 \rangle}$ are deterministically assigned to true by BCP. This causes the removal from 3. of the first-implied disjunct $\neg A_{\langle 1,\ \Box\neg A_1 \rangle}$, so that there is no need to generate $Def(1,\ \neg\Box\neg A_1)$, and hence label 1.1. is not defined and 4. is not generated. While building 5., $A_{\langle 1.2,\ (\neg A_2 \wedge \neg A_3) \rangle}$, is unit-propagated. As label 1.1. is not defined, 6., 7. and 8. are not generated. Then during the construction of 5., 9., 10., 11. and 12., by applying BCP a contradiction is found, so that $K_m2SAT(\varphi)$ is $\bot$.

An analogous situation happens with $\varphi_{bnflift}$ in Example 5.5.3: while building 1. and 2. a contradiction is found by BCP, s.t. K_m2SAT returns $\bot$ without expanding the formula any further. Same discourse holds for $\varphi_{bnfclift}$ in Example 5.5.4: while building 1., 2. and 3. a contradiction is found by BCP, s.t. K_m2SAT returns $\bot$ without expanding the formula any further. $\diamond$

5.5.9 Soundness and Completeness of the Proposed Optimizations

In this section we briefly discuss all the optimizations presented in the previous sections (from Section 5.5.1 to Section 5.5.8) showing that they can be safely applied keeping our encoding sound and complete. Summarizing, we have introduced:

Preprocessings transformations. Applied on the input modal formula before the encoding:

- *Pre-conversion into BNF*
- *Normalization of Modal Atoms*
- *Box Lifting*
- *Controlled Box Lifting*

On-the-fly simplifications. Applied to the SAT formula during the encoding:

- *On-the-fly Boolean Simplification and Truth Propagation*

[15] For the sake of compatibility with standard SAT solvers, our actual implementation returns the formula $A_1 \wedge \neg A_1$.

[16] Here we make the same consideration as in Footnote 12: if $L_{\langle \sigma.j,\ \psi \rangle}$ is generated also from the expansion of some other subformula, (e.g., $\Box_r(\psi \vee \phi)$), then (another instance of) $Def(\sigma.i,\ \psi)$ must be generated anyway.

- *On-the-fly Truth Propagation through Modal Operators*
- *On-the-fly Pure-Literal Reduction*
- *On-the-fly Boolean Constraint Propagation*

First, notice that all the *Preprocessing* optimizations consist simply in equivalence or validity-preserving transformations on the input modal formula (often performing normalization operations) so that the satisfiability of the formula is unchanged, and it can not affect the soundness and completeness of our approach.

Second, *On-the-fly* simplifications apply on the encoded SAT formula, simplifying it directly during the building process. Thus, they don't affect the soundness and completeness of our encoding as far as such applied simplifications preserve the satisfiability of the encoded SAT formula. In this sense, soundness and completeness are trivially preserved by *On-the-fly Boolean Simplification and Truth Propagation*, *On-the-fly Pure-Literal Reduction* and *On-the-fly Boolean Constraint Propagation*, which apply well-known safe transformations in the SAT encoding. Notice, in particular, the following facts:

- The propositional *Pure-Literal Rule (PLR)* (Davis et al., 1962) is a well-known sound and complete propositional reduction stating that: when one proposition occurs only positively [resp. negatively] in the formula, it can be safely assigned to *true* [resp. *false*]. Thus *On-the-fly Pure-Literal Reduction* can be safely applied on our under-encoding SAT formula.
- *Boolean Constraint Propagation (BCP)* is the iterative application of unit-propagation that is the well-known (sound and complete) reduction at the base of every modern SAT solver. Thus *On-the-fly Boolean Constraint Propagation* preserves the soundness and completeness of our encoding, since the effect of applying On-the-fly BCP is the same of applying traditional BCP on the whole SAT formula generated by our basic encoding.

Moreover, *On-the-fly Truth Propagation through Modal Operators* is a trivial and intuitive application of the semantic of modal operators combined with truth values.

However, alternatively, the soundness and completeness of all these On-the-fly optimizations can be trivially proved under a modal perspective by mean of reductions to Single Step Tableaux (Massacci, 2000) similarly to what done by Vescovi (2006) in proving the soundness and completeness of the basic $K_m 2SAT$ encoding (we remark in particular that a variant of the modal *Pure-Literal Reduction* has been proved to be sound and completed by Pan and Vardi (2003), and such proof can be easily extended to meet our more specific case).

5.5.10 A Paradigmatic Example: Halpern & Moses Branching Formulas.

Among all optimizations described in this Section 5.5, on-the-fly BCP is by far the most effective. In order to better understand this fact, we consider as a paradigmatic example

the branching formulas φ_h^K by Halpern and Moses (1992, 1995) (also called "`k_branch_n`" in the set of benchmark formulas proposed by Heuerding and Schwendimann, 1996) and their unsatisfiable version (called "`k_branch_p`" in the above-mentioned benchmark suite).

Given a single modality $\Box$, an integer parameter h, and the primitive propositions $D_0, \ldots, D_{h+1}, P_1, \ldots, P_h$, the formulas φ_h^K are defined as follows: [17]

$$\varphi_h^K \stackrel{\text{def}}{=} D_0 \ \wedge \ \neg D_1 \ \wedge \ \bigwedge_{i=0}^{h} \Box^i(\mathit{depth} \wedge \mathit{determined} \wedge \mathit{branching}), \tag{5.15}$$

$$\mathit{depth} \stackrel{\text{def}}{=} \bigwedge_{i=1}^{h+1}(D_i \rightarrow D_{i-1}), \tag{5.16}$$

$$\mathit{determined} \stackrel{\text{def}}{=} \bigwedge_{i=1}^{h}\left(D_i \ \rightarrow \ \left(\begin{array}{ll} (\ P_i \rightarrow \Box(D_i \rightarrow \ P_i)) & \wedge \\ (\neg P_i \rightarrow \Box(D_i \rightarrow \neg P_i)) & \end{array} \right) \right), \tag{5.17}$$

$$\mathit{branching} \stackrel{\text{def}}{=} \bigwedge_{i=0}^{h-1}\left((D_i \wedge \neg D_{i+1}) \ \rightarrow \ \left(\begin{array}{ll} \Diamond(D_{i+1} \wedge \neg D_{i+2} \wedge \ P_{i+1}) & \wedge \\ \Diamond(D_{i+1} \wedge \neg D_{i+2} \wedge \neg P_{i+1}) & \end{array} \right) \right). \tag{5.18}$$

A conjunction of the formulas *depth*, *determined* and *branching* is repeated at every nesting level of modal operators (i.e. at every depth): *depth* captures the relation between the D_i's at every level; *determined* states that, if P_i is true [false] in a state at depth $\geq i$, then it is true [false] in all the successor states of depth $\geq i$; *branching* states that, for every node at depth i, it is possible to find two successor states at depth $i+1$ such that P_{i+1} is true in one and false in the other. For each value of the parameter h, φ_h^K is K-satisfiable, and every Kripke model M that satisfies it has at least $2^{h+1}-1$ states. In fact, φ_h^K is build in such a way to force the construction of a binary-tree Kripke model of depth $h+1$, each of whose leaves encodes a distinct truth assignment to the primitive propositions $P_1, \ldots, P_h$, whilst each D_i is true in all and only the states occurring at a depth $\geq i$ in the tree (and thus denotes the level of nesting).

The unsatisfiable counterpart formulas proposed by Heuerding and Schwendimann (1996) (whose negations are the valid formulas called `k_branch_p` in the previously-mentioned benchmark suite, which are exposed in more details in Section 5.6.1) are obtained by conjoining to (5.15) the formula:

$$\Box^h P_{\lfloor \frac{h}{3} \rfloor + 1} \tag{5.19}$$

(where $\lfloor x \rfloor$ is the integer part of x) which forces the atom $P_{\lfloor \frac{h}{3} \rfloor + 1}$ to be true in all depth-h states of the candidate Kripke model, which is incompatible with the fact that the remaining specifications say that it has to be false in half depth-h states. [18]

[17] For the sake of better readability, here we adopt the description given by Halpern and Moses (1992) without converting the formulas into BNF. This fact does not affect the discussion.

[18] Heuerding and Schwendimann do not explain the choice of the index "$\lfloor \frac{h}{3} \rfloor + 1$". We understand that also other choices would have done the job.

These formulas are very pathological for many approaches (Giunchiglia & Sebastiani, 2000; Giunchiglia, Giunchiglia, Sebastiani, & Tacchella, 2000; Horrocks et al., 2000). In particular, before introducing on-the-fly BCP, they used to be the pet hate of our K_m2SAT approach, as they caused the generation of huge Boolean formulas. In fact, due to *branching* (5.18), φ_h^K contains $2h$ $\Diamond$-formulas (i.e., π-formulas) at every depth. Therefore, the K_m2SAT encoder of Section 5.3 has to consider $1 + 2h + (2h)^2 + ... + (2h)^{h+1} = ((2h)^{h+2} - 1)/(2h - 1)$ distinct labels, which is about h^{h+1} times the number of those labeling the states which are actually needed. (None of the optimizations of Sections 5.5.1-5.5.7 is of any help with these formulas, because neither BNF encoding nor atom normalization causes any sharing of subformulas, the formulas are already in lifted form, and no literal occurs pure. [19])

This pathological behavior can be mostly overcome by applying on-the-fly-BCP, because some truth values can be deterministically assigned to some subformulas of φ_h^K by on-the-fly-BCP, which prevent encoding some or even most $\Box/\Diamond$-subformulas.

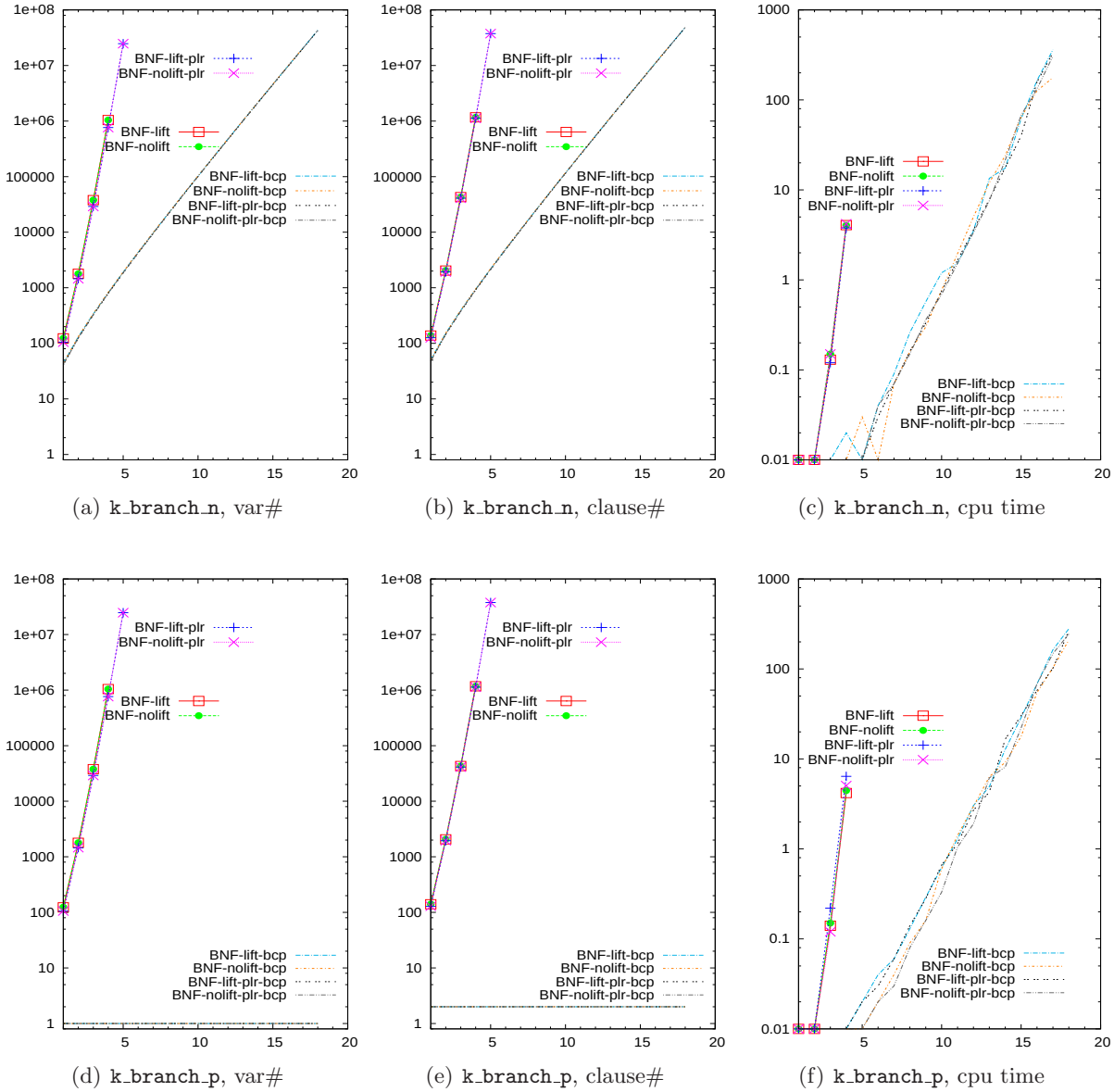
In fact, consider the *branching* and *determined* formulas occurring in φ_h^K at a generic depth $d \in \{0...h\}$, which determine the states at level d in the tree. As in these states $D_0, ..., D_d$ are forced to be true and $D_{d+1}, ..., D_{h+1}$ are forced to be false, then all but the d-th conjunct in *branching* (all conjuncts if $d = h$) are forced to be true and thus they could be dropped. Therefore, only 2 $\Diamond$-formulas per non-leaf level could be considered instead, causing the generation of $2^{h+1} - 1$ labels overall. Similarly, in all states at level d the last $h - d$ conjuncts in *determined* are forced to be true and could be dropped, reducing significantly the number of $\Box$-formulas to be considered.

It is easy to see that this is exactly what happens by applying on-the-fly-BCP. In fact, suppose that the construction of $K_m2SAT(\varphi_h^K)$ has reached depth d (that is, the point where for every state σ at level d, the $Def(\sigma,\ \alpha)$'s and $Def(\sigma,\ \beta)$'s are expanded but no $Def(\sigma,\ \pi)$ and $Def(\sigma,\ \nu)$ is expanded yet). Then, BCP deterministically assigns true to the literals $L_{\langle\sigma,\ D_0\rangle}, ..., L_{\langle\sigma,\ D_d\rangle}$ and false to $L_{\langle\sigma,\ D_{d+1}\rangle}, ..., L_{\langle\sigma,\ D_{h+1}\rangle}$, which removes all but one conjuncts in *branching*, so that only two $Def(\sigma,\ \pi)$'s out of $2h$ ones are actually expanded; similarly, the last $h - d$ conjuncts in *determined* are removed, so that the corresponding $Def(\sigma,\ \nu)$'s are not expanded.

As far as the unsatisfiable version $K_m2SAT(\varphi_h^K \wedge \Box^h P_{\lfloor \frac{h}{3} \rfloor + 1})$ is concerned, when the expansion reaches depth h, thanks to (5.19), $L_{\langle\sigma,\ P_{\lfloor \frac{h}{3} \rfloor + 1}\rangle}$ is generated and deterministically assigned to true by BCP for every depth-h label σ; thanks to *determined* and *branching*, BCP assigns all literals $L_{\langle\sigma,\ P_1\rangle}, ..., L_{\langle\sigma,\ P_h\rangle}$ deterministically, so that $L_{\langle\sigma,\ P_{\lfloor \frac{h}{3} \rfloor + 1}\rangle}$ is assigned to false for 50% of the depth-h labels σ. This causes a contradiction, so that the encoder stops the expansion and returns $\bot$.

Figure 5.1 shows the growth in size and the CPU time required to encode and solve $K_m2SAT(\varphi_h^K)$ (1st row) and $K_m2SAT(\varphi_h^K \wedge \Box^h P_{\lfloor \frac{h}{3} \rfloor + 1})$ (2nd row) wrt. the parameter h, for eight combinations of the following options of the encoder: with and without box-lifting, with and without on-the-fly PLR, with and without on-the-fly BCP. (Notice the

[19] More precisely, only one literal, $\neg D_{h+1}$, occurs pure in *branching*, but assigning it plays no role in simplifying the formula.



(a) `k_branch_n`, var# (b) `k_branch_n`, clause# (c) `k_branch_n`, cpu time

(d) `k_branch_p`, var# (e) `k_branch_p`, clause# (f) `k_branch_p`, cpu time

Figure 5.1: Empirical analysis of $K_m 2SAT$ on Halpern & Moses formulas wrt. the depth parameter h, for different options of the encoder. 1st row: `k_branch_n`, corresponding to $K_m 2SAT(\varphi_h^K)$, formulas (satisfiable); 2nd row: `k_branch_p`, corresponding to $K_m 2SAT(\varphi_h^K \wedge \Box^h P_{\lfloor \frac{h}{3} \rfloor + 1})$, formulas (unsatisfiable). Left: number of Boolean variables; center: number of clauses; right: total CPU time requested to encoding+solving (where the solving step has been performed through RSAT). See Section 5.6 for more technical details.

log scale of the y axis.) In Figure 5.1(d) the plots of the four versions "`-xxx-bcp`" (with on-the-fly BCP) coincide with the line of value 1 (i.e, one variable) and in Figure 5.1(e) they coincide with an horizontal line of value 2 (i.e, two clauses), corresponding to the fact that the 1-variable/2-clause formula $A_1 \wedge \neg A_1$ is returned (see Footnote 15).

We notice a few facts. First, for both formulas, the eight plots always collapse into two groups of overlapping plots, representing the four variants with and without on-the-fly BCP respectively. This shows that box-lifting and on-the-fly PLR are almost irrelevant in the encoding of these formulas, causing just little variations in the time required by the encoder (Figures 5.1(c) and 5.1(f)); notice that enabling on-the-fly PLR alone permits to encode (but not to solve) only one problem more wrt. the versions without both on-the-fly PLR and BCP. Second, the four versions with on-the-fly-BCP always outperform of several orders magnitude these without this option, in terms of both size of encoded formulas and of CPU time required to encode and solve them. In particular, in the case of the unsatisfiable variant (Figure 5.1, second row) the encoder returns the $\bot$ formula, so that no actual work is required to the SAT solver (the plot of Figure 5.1(f) refers only to encoding time).

5.6 Empirical Evaluation

In order to verify empirically the effectiveness of this approach, we have performed a very-extensive empirical test session on about 14,000 $K_m/\mathcal{ALC}$ formulas. We have implemented the encoder $K_m 2SAT$ in C++, with some flags corresponding to the optimizations exposed in the previous section: (i) `NNF`/`BNF`, performing a pre-conversion into NNF/BNF before the encoding; (ii) `lift`/`ctrl.lift`/`nolift`, performing respectively Box Lifting, Controlled Box Lifting or no Box Lifting before the encoding; (iii) `plr` if on-the-fly Pure Literal Reduction is performed and (iv) `bcp` if on-the-fly Boolean Constraint Propagation is performed. The techniques introduced in Section 5.5.2, Section 5.5.5 and Section 5.5.6 are hardwired in the encoder. Moreover, as pre-conversion into `BNF` almost always produces smaller formulas than `NNF`, we have set the `BNF` flag as a default.

In combination with $K_m 2SAT$ we have tried several SAT solvers on our encoded formulas (including ZCHAFF `2004.11.15`, SIEGE `v4`, BERKMIN `5.6.1`, MINISAT `v1.13`, SAT-ELITE `v1.0`, SAT-ELITE GTI `2005 submission` [20], MINISAT 2.0 `061208` and RSAT `1.03`). After a preliminary evaluation and further intensive experiments we have selected RSAT `1.03` (Pipatsrisawat & Darwiche, 2006), because it produced the best overall performances on our benchmark suites (although the performance gaps wrt. other SAT tools, e.g. MINISAT 2.0, were not dramatic).

We have downloaded the available versions of the state-of-the-art tools for K_m-

[20] In the preliminary evaluation of the available SAT solvers we have also tried SAT-ELITE as a preprocessor to reduce the size of the SAT formula generated by $K_m 2SAT$ without the `bcp` option before to solve it. However, even if the preprocessing can signinificantly reduce the size of the formula, it has turned out that this preprocessing is too time-expensive and that the overall time spent for preprocessing and then solving the reduced problem is higher than that solving directly the original encoded SAT formula.

satisfiability. After an empirical evaluation [21] we have selected RACER `1-7-24` (Haarslev & Moeller, 2001) and *SAT `1.3` (Tacchella, 1999) as the best representatives of the tableaux/DPLL-based tools, MSPASS `v 1.0.0t.1.3` (Hustadt & Schmidt, 1999; Hustadt et al., 1999) [22] as the best representative of the FOL-encoding approach, KBDD (unique version) (Pan et al., 2002; Pan & Vardi, 2003) [23] as the representative of the automata-theoretic approach. No representative of the CSP-based and of the inverse method approaches could be used. [24] Notice that all these tools but RACER are experimental tools, as far as K_m2SAT which is a prototype, and many of them (e.g. *SAT and KBDD) are no longer maintained.

Finally, as representative of the QBF-encoding approach, we have selected the K-QBF translator (Pan & Vardi, 2003) combined with the SKIZZO version `0.8.2` QBF solver (Benedetti, 2005), which turned out to be by far [25] the best QBF solver on our benchmarks among the freely-available QBF solvers from the QBF2006 competition (Narizzano, Pulina, & Tacchella, 2006). (In our evaluation we have considered the tools : 2CLSQ, SQBF, PREQUANTOR—i.e. PREQUEL +QUANTOR— QUANTOR `2.11`, and SEMPROP `010604`. In Figure 5.11(b) of the appendix Section 5.9, we plot the results of this comparison.)[26]

All tests presented in this section have been performed on a two-processor Intel Xeon 3.0GHz computer, with 1 MByte Cache each processor, 4 GByte RAM, with Red Hat Linux 3.0 Enterprise Server, where four processes can run in parallel. When reporting the results for one K_m2SAT +RSAT version, the CPU times reported are the sums of both

[21] As we did for the selection of the SAT solver, in order to select the tools to be used in the empirical evaluation, we have performed a preliminary evaluation on the smaller benchmark suites (i.e. the LWB and, sometimes, the TANCS 2000 ones; see later). Importantly, from this preliminary evaluation RACER turned out to be definitely more efficient than FACT++, being able to solve more problems in less time. Also, we repeated this preliminary evaluation with a newer version of FACT++ (`v1.2.3`, March 5th, 2009) and the same version of RACER used in this first part of our work. In this evaluation RACER solves ten more problems than FACT++ on the LWB benchmark, and over than one hundred of problems more than FACT++ on the whole TANCS 2000 suite. Also on $\Box_m$-CNF random problems RACER outperforms FACT++. (We include in the appendix Section 5.9 the plots of this comparison between RACER and FACT++. See Figures 5.10 and 5.11(a).)

[22] We have run MSPASS with the options `-EMLTranslation=2 -EMLFuncNary=1 -Sorts=0 -CNFOptSkolem=0 -CNFStrSkolem=0 -Select=2 -Split=-1 -DocProof=0 -PProblem=0 -PKept=0 -PGiven=0`, which are suggested for K_m-formulas in the MSPASS README file. We have also tried other options, but the former gave the best performances.

[23] KBDD has been recompiled to be run with an increased internal memory bound of 1 GB.

[24] At the moment KЖ is not freely available, and we failed in the attempt of obtaining it from the authors. KCSP is a prolog piece of software, which is difficult to compare in performances wrt. other optimized tools on a common platform; moreover, KCSP is no more maintained since 2005, and it is not competitive wrt. state-of-the-art tools (Brand, 2008). Other tools like LEANK, □KE, LWB, KRIS are not competitive with the ones listed above (Horrocks et al., 2000). KSAT (Giunchiglia & Sebastiani, 1996, 2000; Giunchiglia et al., 2000) has been reimplemented into *SAT.

[25] Unlike with the choice of SAT solver, the performance gaps from the best choice and the others were very significant: e.g., in the LWB benchmark (see later), SKIZZO was able to solve nearly 90 problems more than its best QBF competitor.

[26] At the moment of this empirical evaluation many reasoners like PELLET or HERMIT were not yet available, as far as more recent QBF and SAT solvers aside from those above mentioned or newer versions of some above listed tools. The results of this evaluation, in fact, have been previously included in the work of Sebastiani and Vescovi (2009a).

the encoding and RSAT solving times. When reporting the results for K-QBF +sKIZZO, the CPU times reported are only due to sKIZZO because the time spent by the K-QBF converter is negligible.

We anticipate that, for all formulas of all benchmark suites, all tools under test —i.e. all the variants of K_m2SAT +RSAT and all the state-of-the-art K_m-satisfiability solvers— agreed on the satisfiability/unsatisfiability result when terminating within the timeout.

Remark 1. Due to the big number of empirical tests performed and to the huge amount of data plotted, and due to limitations in size, and in order to to make the plots clearly distinguishable in the figures, we have limited the number of plots included in the following part, considering only the most meaningful ones and those regarding the most challenging benchmark problems faced. For the sake of the reader's convenience, however, we may include in our considerations also thes results measured on the easier problems here not plotted when discussing the empirical evaluation.

5.6.1 Test Description

We have performed our empirical evaluation on three different well-known benchmarks suites of $K_m/\mathcal{ALC}$ problems: the LWB (Heuerding & Schwendimann, 1996), the random $\Box_m$-CNF (Horrocks et al., 2000; Patel-Schneider & Sebastiani, 2003) and the TANCS 2000 (Massacci & Donini, 2000) benchmark suites. We are not aware of any other publicly-available benchmark suite on $K_m/\mathcal{ALC}$-satisfiability from the literature. These three groups of benchmark formulas allow us to test the effectiveness of our approach on a large number of problems of various sizes, depths, hardness and characteristics, for a total amount of about 14,000 formulas.

In particular, these benchmark formulas allow us to fairly evaluate the different tools both on the *modal* component and on the *Boolean* component of reasoning which are intrinsic in the K_m-satisfiability problem, as we discuss later in Section 5.6.4.

In the following we describe these three benchmark suites.

The LWB Benchmark Suite

As a first group of benchmark formulas we used the LWB benchmark suite used in a comparison at Tableaux'98 (Heuerding & Schwendimann, 1996). It consists of 9 classes of parametrized formulas (each in two versions, provable "_p" or not-provable "_n" [27]), for a total amount of 378 formulas. The parameter allows for creating formulas of increasing size and difficulty.

The benchmark methodology is to test formulas from each class, in increasing difficulty, until one formula cannot be solved within a given timeout, 1000 seconds in our tests. [28] The result from this class is the parameter's value of the largest (and hardest) formula that can be solved within the time limit. The parameter ranges only from 1 to 21 so that,

[27] Since all tools check K_m-(un)satisfiability, all formulas are negated, so that the negations of the provable formulas are checked to be unsatisfiable, whilst the negation of the other formulas are checked to be satisfiable.

[28] We also set a 1 GB file-size limit for the encoding produced by K_m2SAT.

if a system can solve all 21 instances of a class, the result is given as 21. For a discussion on this benchmark suite, we refer the reader to the work of Heuerding and Schwendimann (1996) and of Horrocks et al. (2000).

The Random $\Box_m$-CNF Benchmark Suite

As a second group of benchmark formulas, we have selected the random $\Box_m$-CNF testbed described by Horrocks et al. (2000), and Patel-Schneider and Sebastiani (2003). This is a generalization of the well-known random k-SAT test methods, and is the final result of a long discussion in the communities of modal and description logics on how to to obtain significant and flawless random benchmarks for modal/description logics (Giunchiglia & Sebastiani, 1996; Hustadt & Schmidt, 1999; Giunchiglia et al., 2000; Horrocks et al., 2000; Patel-Schneider & Sebastiani, 2003).

In the $\Box_m$-CNF test methodology, a $\Box_m$-CNF formula is randomly generated according to the following parameters:

- the (maximum) modal depth d;
- the number of top-level clauses L;
- the number of literal per clause clauses k;
- the number of distinct propositional variables N;
- the number of distinct box symbols m;
- the percentage p of purely-propositional literals in clauses occurring at depth $< d$, s.t. each clause of length k contains on average $p \cdot k$ randomly-picked Boolean literals and $k - p \cdot k$ randomly-generated modal literals $\Box_r \psi$, $\neg\Box_r \psi$. [29]

(We refer the reader to the works of Horrocks et al., 2000, and Patel-Schneider & Sebastiani, 2003 for a more detailed description.)

A typical problem set is characterized by fixed values of d, k, N, m, and p: L is varied in such a way as to empirically cover the "100% satisfiable / 100% unsatisfiable" transition. In other words, many problems with the same values of d, k, N, m, and p but an increasing number of clauses L are generated, starting from really small, typically satisfiable problems (i.e. with a probability of generating a satisfiable problem near to one) to huge problems, where the increasing interactions among the numerous clauses typically leads to unsatisfiable problems (i.e. it makes the probability of generating satisfiable problems converging to zero). Then, for each tuple of the five values in a problem set, a certain number of $\Box_m$-CNF formulas are randomly generated, and the resulting formulas are given in the input to the procedure under test, with a maximum time bound. The fraction of formulas which were solved within a given timeout, and the median/percentile

[29] More precisely, the number of Boolean literals in a clause is $\lfloor p \cdot k \rfloor$ (resp. $\lceil p \cdot k \rceil$) with probability $\lceil p \cdot k \rceil - p \cdot k$ (resp. $1 - (\lceil p \cdot k \rceil - p \cdot k)$). Notice that typically the smaller is p, the harder is the problem (Horrocks et al., 2000; Patel-Schneider & Sebastiani, 2003).

values of CPU times are plotted against the ratio L/N. Also, the fraction of satisfiable/unsatisfiable formulas is plotted for a better understanding.

Following the methodology proposed by Horrocks et al. (2000), and by Patel-Schneider and Sebastiani (2003), we have fixed $m = 1$, $k = 3$ and 100 samples per point in all tests, and we have selected two groups: an "easier" one, with $d = 1$, $p = 0.5$, $N = 6, 7, 8, 9$, $L/N = 10..60$, and a "harder" one, with $d = 2$, $p = 0.6, 0.5$, $N = 3, 4$, $L/N = 30..150$ with $p = 0.6$ and $L/N = 50..140$ with $p = 0.5$, varying the L/N ratio in steps of 5, for a total amount of 13,200 formulas.

In each test, we imposed a timeout of 500 seconds per sample [30] and we calculated the number of samples which were solved within the timeout, and the 50%th and 90%th percentiles of CPU time. [31] In order to correlate the performances with the (un)satisfiability of the sample formulas, in the background of each plot we also plot the satisfiable/unsatisfiable ratio.

The TANCS 2000 Benchmark Suite

Finally, as a third group of benchmark formulas, we used the MODAL PSPACE division benchmark suite used in the comparison at TANCS 2000 (Massacci & Donini, 2000). It contains both satisfiable and unsatisfiable formulas, with scalable hardness. In this benchmark suite, which we call TANCS 2000, the formulas are constructed by translating QBF formulas into K using three translation schemas, namely the Schmidt-Schauss-Smolka translation (240 problems with many different depths, from 19 to 112), the Ladner translation (240 problems, again with depths in the same range 19 – 112), and the Halpern translation (56 problems of depth among: 20, 28, 40, 56, 80 or 112) (Massacci & Donini, 2000). As done by Massacci and Donini, we call these classes *easy*, *medium* and *hard* respectively.

All formulas from each class are tested within a timeout of 1000 seconds. [32] For each class, we report the number of solved formulas (X axis) and the total (cumulative) CPU time spent for solving these formulas (Y axes). For each class the results are plotted sorting the solved problems from the easiest one to the hardest one.

5.6.2 An Empirical Comparison of the Different Variants of K_m2SAT

We have first evaluated the various variants of the encoding in combination with RSAT. In order to avoid considering too many combinations of the flags, we have considered the `BNF` format, and we have grouped `plr` and `bcp` into one parameter `plr-bcp`, restricting thus our investigation to 6 combinations: `BNF`, `lift/ctrl.lift/nolift`, and `plr-bcp` on/off.

[30] With also a 512 MB file-size limit for the encoding produced by K_m2SAT.

[31] Due to the lack of space and for the sake of clarity we won't include in this current section the 90%th percentiles plots. These plots, however, can be found in the appendix Section 5.9. Further, for the same reasons, we'll skip to report the plots regarding some of the easiest class of the benchmark suite (e.g. those with $d = 1$ and lower values of N).

[32] We also set a 1 GB file-size limit for the encoding produced by K_m2SAT.

(We recall that the techniques introduced in Section 5.5.2, Section 5.5.5 and Section 5.5.6 are hardwired in the encoder.) Here we expose and analyze the results wrt. the three different suites of benchmark problems.

Results on the LWB Benchmark Suite

The results on the LWB benchmark suite are summarized in Table 5.1 and Figure 5.2.

Table 5.1(a) reports in the left block the indexes of the hardest formulas encoded within the file-size limit and, in the right block, those of the hardest formulas solved within the timeout by RSAT; Table 5.1(b) reports the numbers of variables and clauses of $K_m 2SAT(\varphi)$, referring to the hardest formulas solved within the timeout by RSAT (i.e., those reported in the right block of Table 5.1(a)). For instance, the `BNF-ctrl.lift-plr-bcp` encoding of `k_dum_n(21)` contains $11 \cdot 10^6$ variables and $14 \cdot 10^6$ clauses; it is the hardest `k_dum_n` problem solved by RSAT with `BNF-ctrl.lift-plr-bcp` and it is the first which is not solved with `BNF-ctrl.lift`.

Looking at the numbers of cases solved in Table 5.1(a), we notice that the introduction of the on-the-fly Pure Literal Reduction and Boolean Constraint Propagation optimizations is really effective and produces a consistent performance enhancement (the effect of these optimizations is eye-catching in the branching formulas `k_branch_*` – see Section 5.5.10 – and in the `k_path_*` formulas). We also notice that `lift` sometimes introduces some slight further improvement.

The view of Tables 5.1(a) and 5.1(b) hides the actual CPU times required to encode and solve the problems. Small gaps in the numbers of Table 5.1(a) may correspond to big gaps in CPU time. In order to analyze also this aspect, in Figure 5.2 we plotted the total cumulative amount of CPU time spent by all the variants of $K_m 2SAT$ +RSAT to solve all the problems of the LWB benchmark, sorted by hardness. For this plot, we also considered three more options —`BNF`, `lift`/`ctrl.lift`/`nolift`, with `plr` on and `bcp` off— so that to evaluate also the effect of `plr` and `bcp` separately. We notice that the plots are clearly clustered into three groups of increasing performance: `BNF-*`, `BNF-*-plr`, and `BNF-*-plr-bcp`., "`*`" representing the three options `lift`/`ctrl.lift`/`nolift`. This highlights the fact that on this suite on-the-fly Pure Literal Reduction significantly improves the performances, that on-the-fly Boolean Constraint Propagation introduces drastic improvements, and that the variations due to Box Lifting are minor wrt. the other two optimizations.

Overall, the configuration `BNF-lift-plr-bcp` turns out to be the best performer on this suite, with a tiny advantage wrt. `BNF-ctrl.lift-plr-bcp`.

Results on the Random $\Box_m$-CNF Benchmark Suite

The results on the random $\Box_m$-CNF benchmark suite are reported in Figures 5.3 and 5.4.

In Figure 5.3 we report the 50%-percentile CPU times required to encode and solve the formulas by the different $K_m 2SAT$ +RSAT variants for the hardest benchmarks problems.

	K_m2SAT, encoded						K_m2SAT + RSAT, solved					
				plr-bcp						plr-bcp		
lifting	no	yes	ctrl	no	yes	ctrl	no	yes	ctrl	no	yes	ctrl
k_branch_n	4	4	4	**18**	**18**	**18**	4	4	4	**17**	**17**	**17**
k_branch_p	4	4	4	**18**	**18**	**18**	4	4	4	**18**	**18**	**18**
k_d4_n	8	8	8	8	**9**	8	**8**	**8**	**8**	**8**	**8**	**8**
k_d4_p	**14**	**14**	**14**	**14**	**14**	**14**	**14**	**14**	**14**	**14**	**14**	**14**
k_dum_n	20	20	20	**21**	**21**	**21**	20	20	20	**21**	**21**	**21**
k_dum_p	19	19	19	**21**	**21**	**21**	18	18	18	**21**	**21**	**21**
k_grz_n	**21**	**21**	**21**	**21**	**21**	**21**	**21**	**21**	**21**	**21**	**21**	**21**
k_grz_p	**21**	**21**	**21**	**21**	**21**	**21**	**21**	**21**	**21**	**21**	**21**	**21**
k_lin_n	**21**	**21**	**21**	**21**	**21**	**21**	**21**	**21**	**21**	**21**	**21**	**21**
k_lin_p	**21**	**21**	**21**	**21**	**21**	**21**	**21**	**21**	**21**	**21**	**21**	**21**
k_path_n	7	7	7	14	**15**	14	7	7	7	13	**14**	13
k_path_p	8	8	8	15	**16**	15	8	8	8	15	**16**	15
k_ph_n	**21**	**21**	**21**	**21**	**21**	**21**	**21**	**21**	**21**	**21**	**21**	**21**
k_ph_p	**21**	**21**	**21**	**21**	**21**	**21**	10	**11**	10	10	10	**11**
k_poly_n	**21**	**21**	**21**	**21**	**21**	**21**	**21**	**21**	**21**	**21**	**21**	**21**
k_poly_p	**21**	**21**	**21**	**21**	**21**	**21**	**21**	**21**	**21**	**21**	**21**	**21**
k_t4p_n	**6**	**6**	**6**	**6**	**6**	**6**	5	**6**	5	**6**	**6**	**6**
k_t4p_p	**11**	**11**	**11**	**11**	**11**	**11**	10	10	10	**11**	**11**	**11**

(a) Indexes of the hardest problems encoded (left) and of the hardest problems solved (right).

	number of variables (10^3)						number of clauses (10^3)					
				plr-bcp						plr-bcp		
lifting	no	yes	ctrl	no	yes	ctrl	no	yes	ctrl	no	yes	ctrl
k_branch_n	1000	1000	1000	20000	20000	20000	1000	1000	1000	23000	23000	23000
k_branch_p	1000	1000	1000	0	0	0	1000	1000	1000	0	0	0
k_d4_n	12000	6000	12000	10000	26000	10000	17000	9000	17000	16000	43000	16000
k_d4_p	19000	18000	19000	0	0	0	28000	25000	28000	0	0	0
k_dum_n	19000	19000	19000	11000	11000	11000	23000	23000	23000	14000	14000	14000
k_dum_p	11000	11000	11000	20000	19000	20000	14000	13000	14000	26000	25000	26000
k_grz_n	10	10	10	5	5	5	10	10	10	6	6	6
k_grz_p	8	8	8	0.2	0.1	0.2	8	8	8	0.3	0.2	0.2
k_lin_n	30	30	20	20	10	20	50	50	20	30	30	30
k_lin_p	0	0	0	0	0	0	0	0	0	0	0	0
k_path_n	11000	12000	11000	10000	7000	10000	13000	14000	13000	14000	9000	13000
k_path_p	11000	12000	11000	26000	16000	26000	13000	14000	13000	35000	20000	35000
k_ph_n	50	300	50	50	300	50	50	300	50	50	600	50
k_ph_p	3	13	3	3	8	4	3	14	3	3	14	5
k_poly_n	200	20	20	200	20	20	200	20	20	200	20	20
k_poly_p	200	20	20	200	20	20	200	20	20	200	20	20
k_t4p_n	4000	21000	4000	17000	14000	17000	4000	22000	4000	20000	17000	20000
k_t4p_p	12000	10000	12000	20000	18000	20000	12000	11000	12000	24000	21000	24000

(b) # of variables and # of clauses of the hardest problems solved.
Note: Here "0" means that the formula is simplified into $\bot$ by K_m2SAT.

Table 5.1: Comparison of the variants of K_m2SAT +RSAT on the LWB benchmarks.

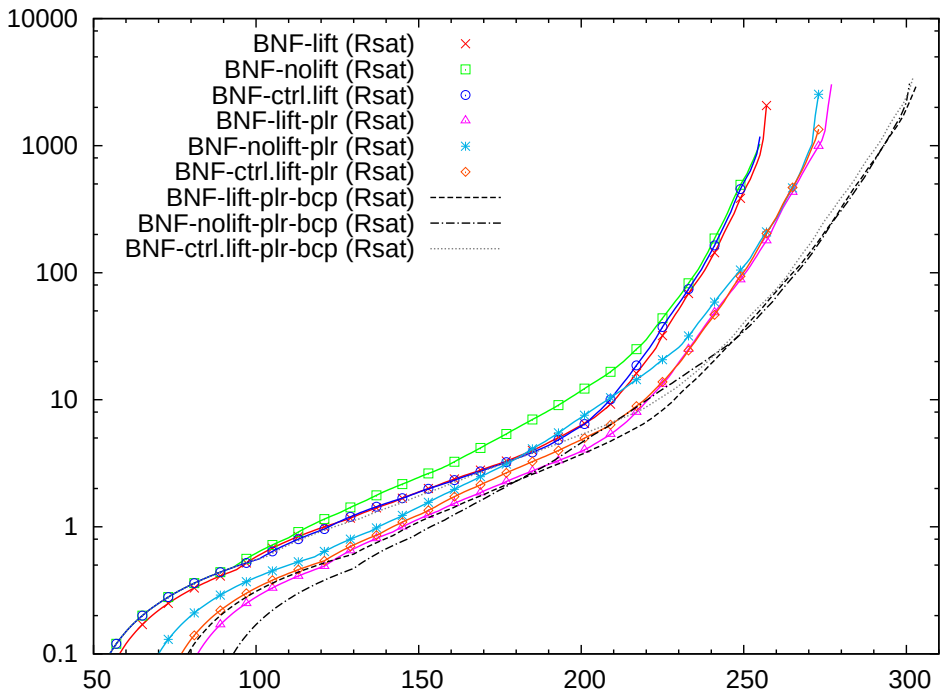


Figure 5.2: Comparison of different variants of K_m2SAT +RSAT on the LWB problems. X axis: number of solved problems; Y axis: total CPU time spent (sorting problems from the easiest to the hardest).

We don't report the percentage of solved problems since it is always 100%, i.e. K_m2SAT +RSAT terminates within the timeout for every problem in the benchmark suite.

The tests with depth $d = 1$ (see the results on the hardest problems of the class in the first row of Figure 5.3) are simply too easy for K_m2SAT +RSAT (but not for its competitors, see Section 5.6.3) which solved every sample formula in less than 1 second. Although the tests exposed in the second and third row of Figure 5.3 are more challenging, they are all solved within the timeout as well. We have noticed also that the results are rather regular, since there are no big gaps between 50%- and 90%-percentile values.

In general, we do not have relevant performance gaps between the various configurations on this benchmark suite; it seems that in the majority of cases `ctrl.lift` slightly beats `nolift` and `nolift` slightly beats `lift`. These gaps are even more relevant if we consider the size of the formulas generated (Figure 5.4). We believe that this may be due to the fact that random $\Box_m$-CNF formulas may contain lots of shared subformulas $\Box_r\psi$, so that lifting may cause a reduction of such sharing (see Section 5.3). Further, `plr-bcp` does not seem to introduce relevant improvements here. We believe that this is due to the fact that these random formulas produce pure and unit literals with very low or even zero probability.

Overall, the configuration `BNF-nolift` turns out to be the best performer on this suite, with a slight advantage wrt. `BNF-ctrl.lift-plr-bcp`.

Finally, from some plots of Figure 5.3 we notice that for K_m2SAT +RSAT the problems

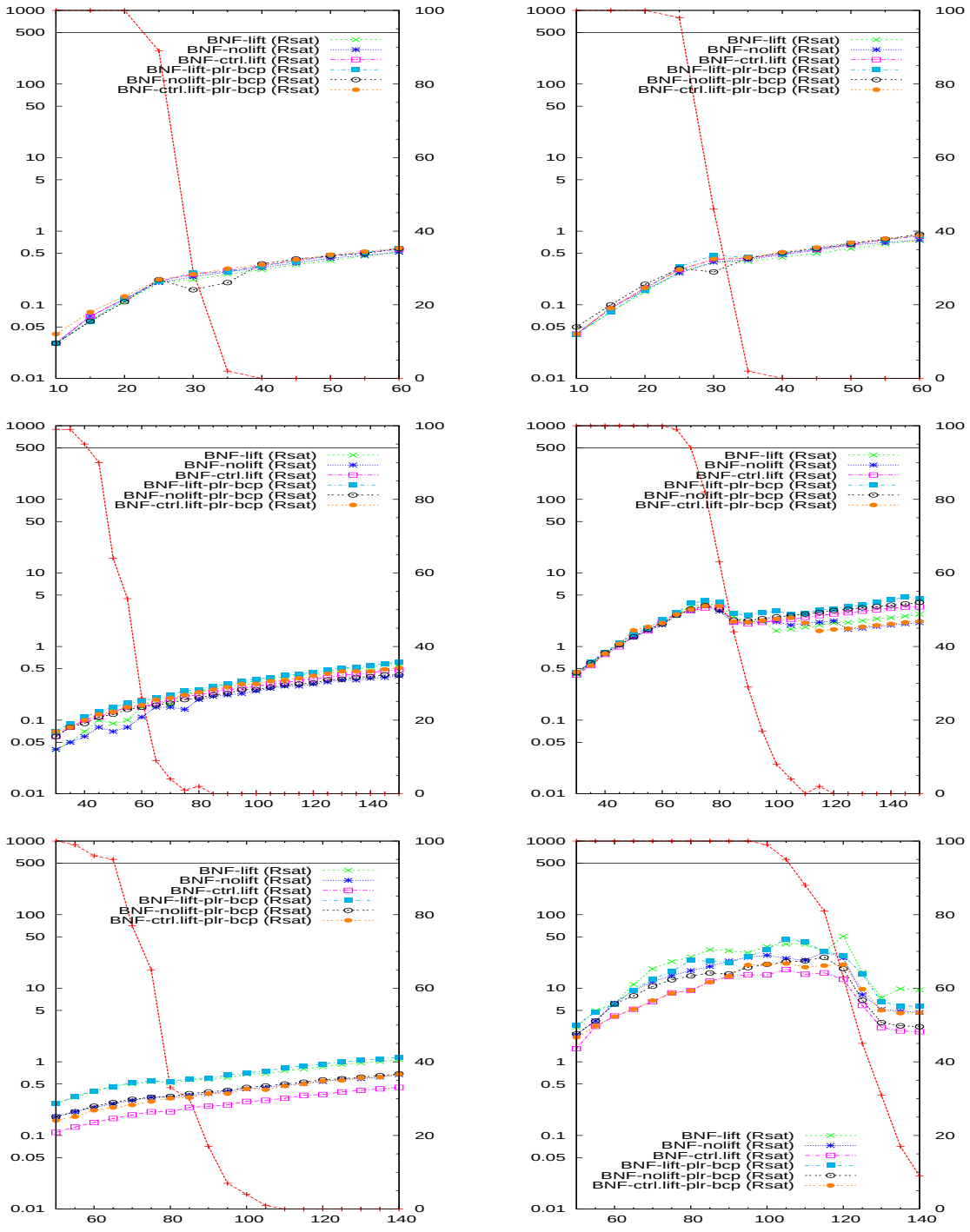


Figure 5.3: Comparison among different variants of K_m2SAT +RSAT on random problems. X axis: $\#clauses/N$. Y axis: median (50th percentile) CPU time, 100 samples/point. 1st row: $d = 1$, $p = 0.5$, $N = 8, 9$; 2nd row: $d = 2$, $p = 0.6$, $N = 3, 4$; 3rd row: $d = 2$, $p = 0.5$, $N = 3, 4$. Background: % of satisfiable instances.

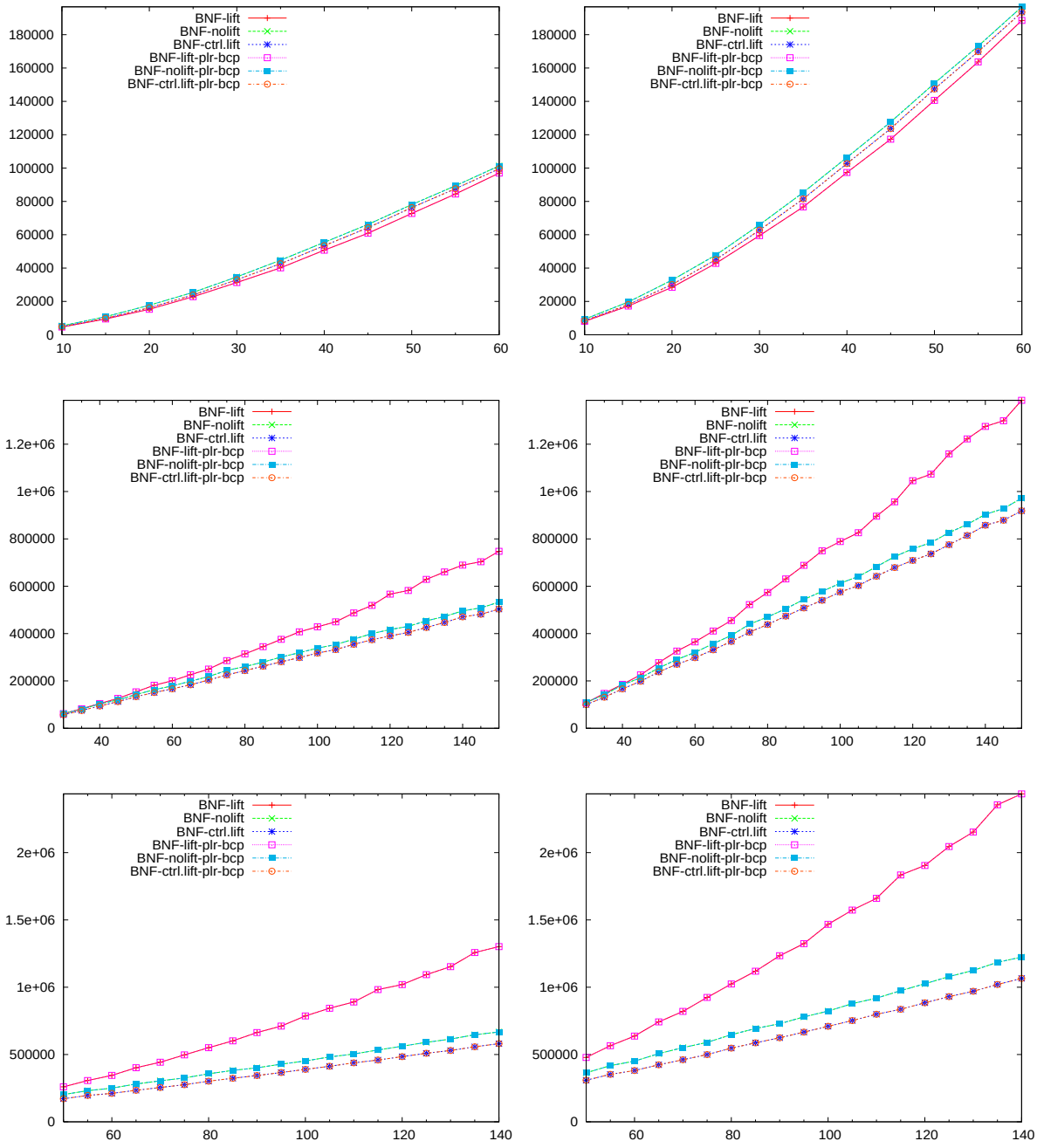


Figure 5.4: Comparison among different variants of $K_m 2SAT$ on random problems. X axis: $\#clauses/N$. Y axis: 1st column: $\#variables$ in the SAT encoding (90th percentiles), 100 samples/point; 2nd column: $\#clauses$ in the SAT encoding (90th percentiles), 100 samples/point. 1st row: $d = 1$, $p = 0.5$, $N = 9$; 2nd row: $d = 2$, $p = 0.6$, $N = 4$; 3rd row: $d = 2$, $p = 0.5$, $N = 4$.

tend to be harder within the satisfiability/unsatisfiability transition area. (This fact holds especially for RACER and *SAT, see Section 5.6.3.) This seems to confirm the fact that the easy-hard-easy pattern of random k-SAT extends also to $\Box_m$-CNF, as already observed in literature (Giunchiglia & Sebastiani, 1996, 2000; Giunchiglia et al., 2000; Horrocks et al., 2000; Patel-Schneider & Sebastiani, 2003).

Results on the TANCS 2000 Benchmark Suite

The comparison among the K_m2SAT variants on the TANCS 2000 benchmark is presented in Figures 5.5 and 5.6, where different `BNF` variants of K_m2SAT are compared both enabling or disabling `lift/ctrl.lif` and `plr-bcp`.

In Figure 5.5, from top-left to bottom, we present the cumulative CPU times spent by K_m2SAT +RSAT on the easy, medium and hard categories respectively. In Figure 5.6 we present the plots of the number of variables and clauses of the formulas solved. We notice that there are only slight differences among the different variants of K_m2SAT; `BNF` with `lift` is the best option which allows for solving more problems in the hard class and requiring less CPU time.

We remark that, despite the expected exponential growth of the encoded formulas wrt. the modal depth, in this benchmark K_m2SAT +RSAT allows for encoding and solving problems of modal depth greater than 100 for the easy class and greater than 50 for the medium and hard classes, producing and solving SAT-encoded formulas with more than 10^7 variables and $1.4 \cdot 10^7$ clauses.

5.6.3 An Empirical Comparison wrt. the Other Approaches

We proceed with the comparison of our approach wrt. the current state-of-the-art evaluating K_m2SAT +RSAT against the other K_m-satisfiability solvers listed above. In more details, we chose to compare the performance of the other solvers against both the best "local" K_m2SAT +RSAT variant for the single benchmark suite and the best "global" K_m2SAT +RSAT variant among all the benchmark suites, which we have identified in `BNF-ctrl.lift-plr-bcp`.

Comparison on the LWB Benchmark Suite

The results on the LWB benchmark suite are summarized numerically and graphically in Table 5.2. From Table 5.2(a) we notice a few facts: RACER and *SAT are the best performers (confirming the analysis done by Horrocks et al., 2000) with a significant gap wrt. the others; then, K-QBF +sKIZZO solves a few more problems than K_m2SAT +RSAT; then follows KBDD which outperforms MSPASS, which is the worst performer. In detail, K_m2SAT +RSAT is (one of) the worst performer(s) on `k_d4_*` and `k_t4_*`, the fourth best performer on `k_path_n`, the third best performer on `k_path_p` and `k_branch_p`, and it is (one of) the best performer(s) on `k_branch_n`, `k_dum_*`, `k_grz_*`, `k_lin_*`, `k_ph_*` and `k_poly_*`; it is the absolute best performer on `k_branch_n` and `k_ph_p`.

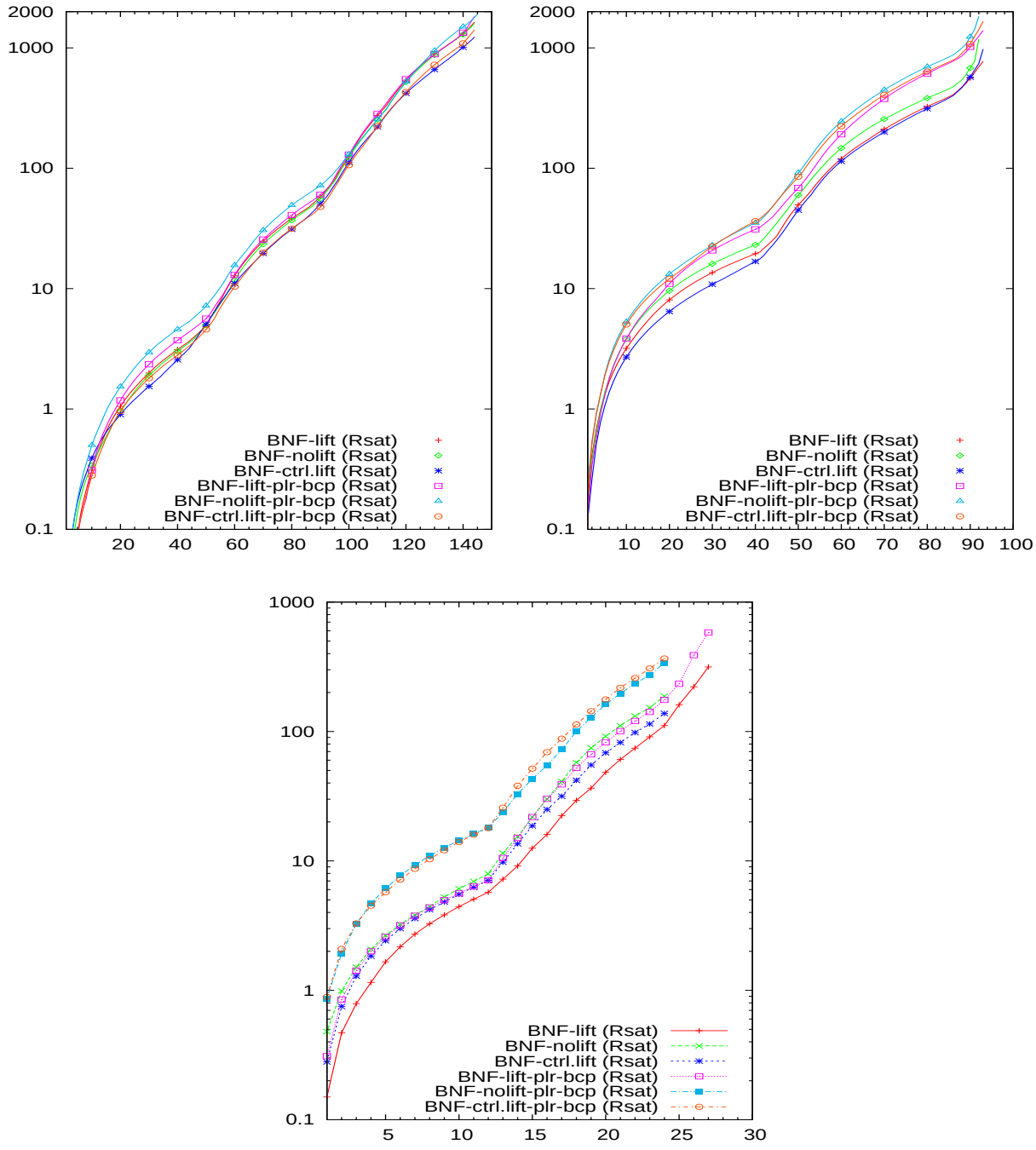


Figure 5.5: Comparison among different variants of $K_m 2SAT$ +RSAT on TANCS 2000 problems. X axis: number of solved problems. Y axis: total cumulative CPU time spent. 1st (top-left) to 3th (bottom) plot: easy, medium, hard problems. (Problems are sorted from the easiest to the hardest).

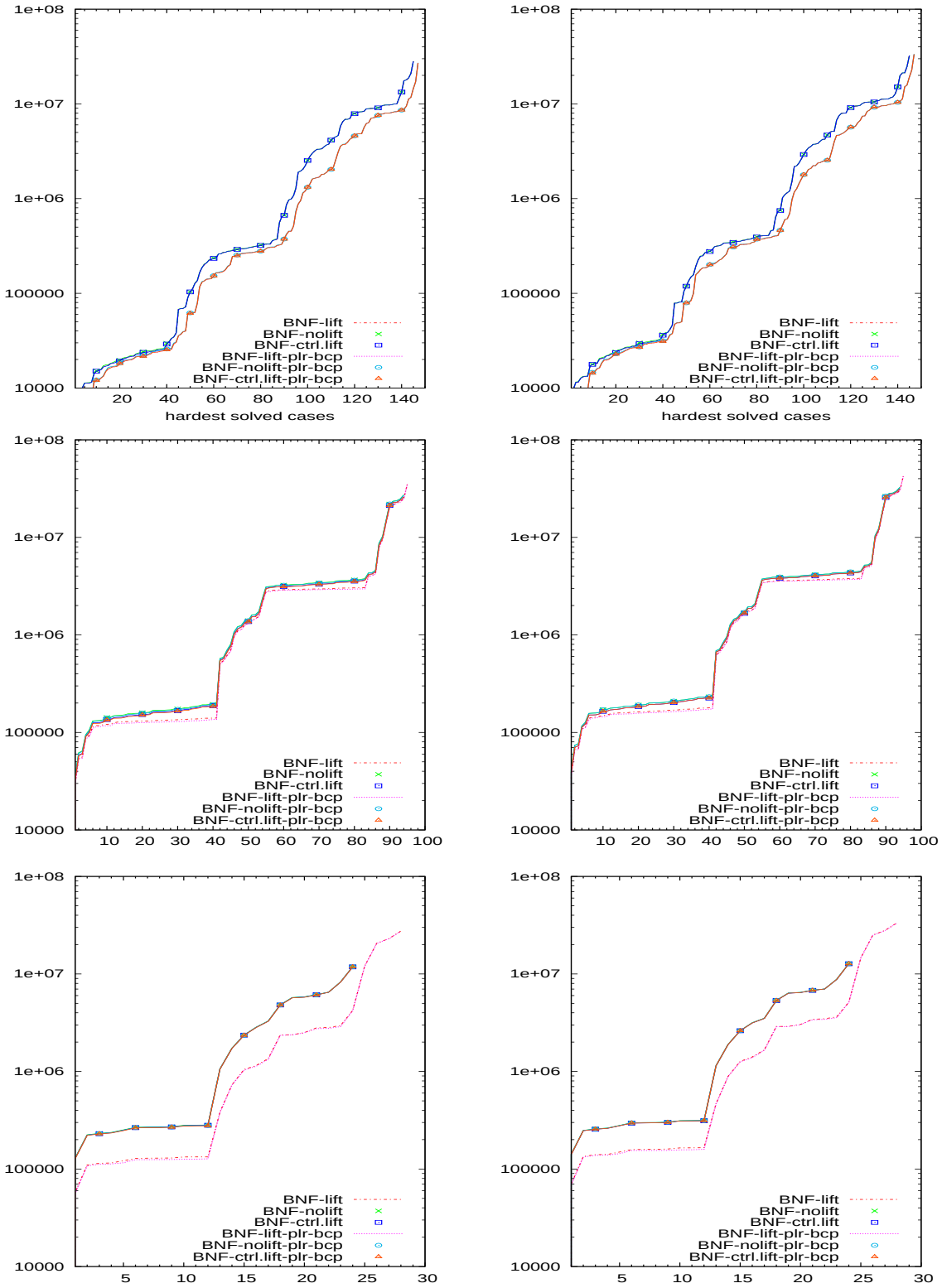
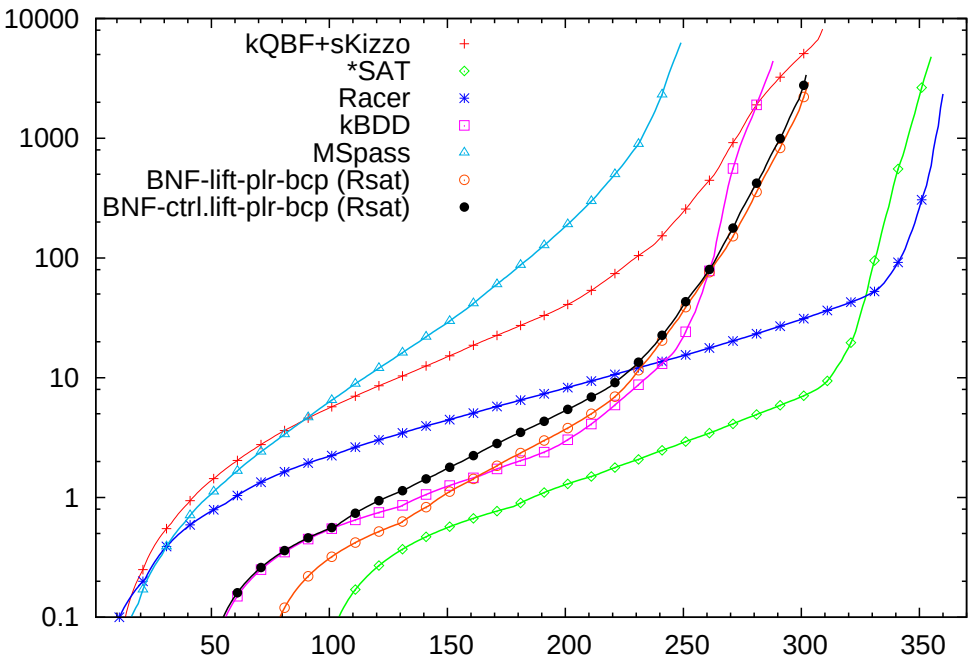


Figure 5.6: Comparison among different variants of K_m2SAT on TANCS 2000 problems. X axis: number of the harder solved problem. Y axis: 1st, 2nd column: *#variables*, *#clauses* in the SAT encoding of the problem. 1st to 3rd row: easy, medium, hard problems.

test	other tools					K_m2SAT + RSAT	
	K-QBF + sKizzo	KBDD	MSPASS	RACER	*SAT	BNF-plr-bcp -ctrl.lift	BNF-plr-bcp -lift
k_branch_n	4	8	10	15	14	**17**	**17**
k_branch_p	16	8	10	**21**	**21**	18	18
k_d4_n	8	**21**	**21**	**21**	**21**	8	8
k_d4_p	**21**	**21**	**21**	**21**	**21**	14	14
k_dum_n	**21**	**21**	**21**	**21**	**21**	**21**	**21**
k_dum_p	**21**	**21**	**21**	**21**	**21**	**21**	**21**
k_grz_n	19	**21**	**21**	**21**	**21**	**21**	**21**
k_grz_p	**21**	**21**	**21**	**21**	**21**	**21**	**21**
k_lin_n	20	**21**	**21**	**21**	**21**	**21**	**21**
k_lin_p	**21**	**21**	3	**21**	**21**	**21**	**21**
k_path_n	9	**21**	4	**21**	**21**	13	14
k_path_p	13	17	5	**21**	**21**	15	16
k_ph_n	**21**	4	12	**21**	13	**21**	**21**
k_ph_p	10	4	8	9	9	**11**	10
k_poly_n	**21**	8	7	**21**	**21**	**21**	**21**
k_poly_p	**21**	8	7	**21**	**21**	**21**	**21**
k_t4p_n	**21**	**21**	12	**21**	**21**	6	6
k_t4p_p	**21**	**21**	**21**	**21**	**21**	11	11

(a) Indexes of the hardest problems solved.



(b) X axis: # of problems solved; Y axis: total (cumulative) CPU time spent.

Table 5.2: Comparison of K_m2SAT +RSAT against the other tools on the LWB benchmarks.

In Table 5.2(b) we give a graphical representation of this comparison, plotting the number of solved problems by each approach against the total cumulative amount of CPU time spent. We notice that, even if K_m2SAT +Rsat solves a few problems less than K-QBF +sKizzo, K_m2SAT +Rsat is mostly faster than K-QBF +sKizzo.

Comparison on the Random $\Box_m$-CNF Benchmark Suite

In the random $\Box_m$-CNF benchmark suite the results are dominated by K_m2SAT +Rsat. For the hardest categories among the three groups of problems used in the evaluation, we report in Figure 5.7 the number of problems solved by each tool within the timeout, and in Figure 5.8 the median CPU time (i.e. the 50%th percentile).

Looking at Figure 5.7 we notice that K_m2SAT +Rsat (in both versions) is the only tool which always terminates within the timeout, whilst *SAT and Racer sometimes do not terminate in the hardest problems, K-QBF +sKizzo very often does not terminate, and Mspass and KBDD do not terminate for most values.

In Figure 5.8 we notice that K_m2SAT +Rsat is most often the best performer (in particular with the hardest problems), followed by *SAT and Racer. (This is even much more evident if we consider the 90%th percentile of CPU time, whose plots are included in Figure 5.13 of Section 5.9.) In all these tests, K-QBF +sKizzo, Mspass and KBDD are drastically outperformed by the others.

Comparison on the TANCS 2000 Benchmark Suite

The results of the TANCS 2000 benchmark are summarized in Figure 5.9, from the easy category (top-left) to the hard category (bottom).

From the plots we notice that the relative performances of the tools under test vary significantly with the category: Racer and *SAT are among the best performers in all categories; K-QBF +sKizzo behaves well on the easy and medium categories but solves very few problems on the hard one; KBDD behaves very well on the easy category, but solves very few problems in the medium and hard ones. Mspass is among the worst performers in all categories: in particular, it does not solve any problem in the hard category; finally, K_m2SAT +Rsat is the worst performer on the easy category, it outperforms all competitors but *SAT and Racer on the medium category, and competes head-to-head with both Racer and *SAT for the first position on the hard category: the "local-best" configuration `BNF-lift` beats both competitors; the "global-best" configuration `BNF-ctrl.lift-prl-bcp` solves as many problems as Racer, with one-order-magnitude CPU-time performance gap, and two problems less than *SAT.

Notice that the classification of the benchmark problems in "easy", "medium" and "hard" given by Massacci and Donini (2000) is based on the translation schema used to produce every modal problem and refers to its "reasoning component", but it is not necessarily related to other components (like, e.g., the modal depth) which affect the size of our encoding and, hence, the efficiency of our approach. This may explain the fact, e.g., that the "easy" problems are not so easy for our approach, and viceversa.

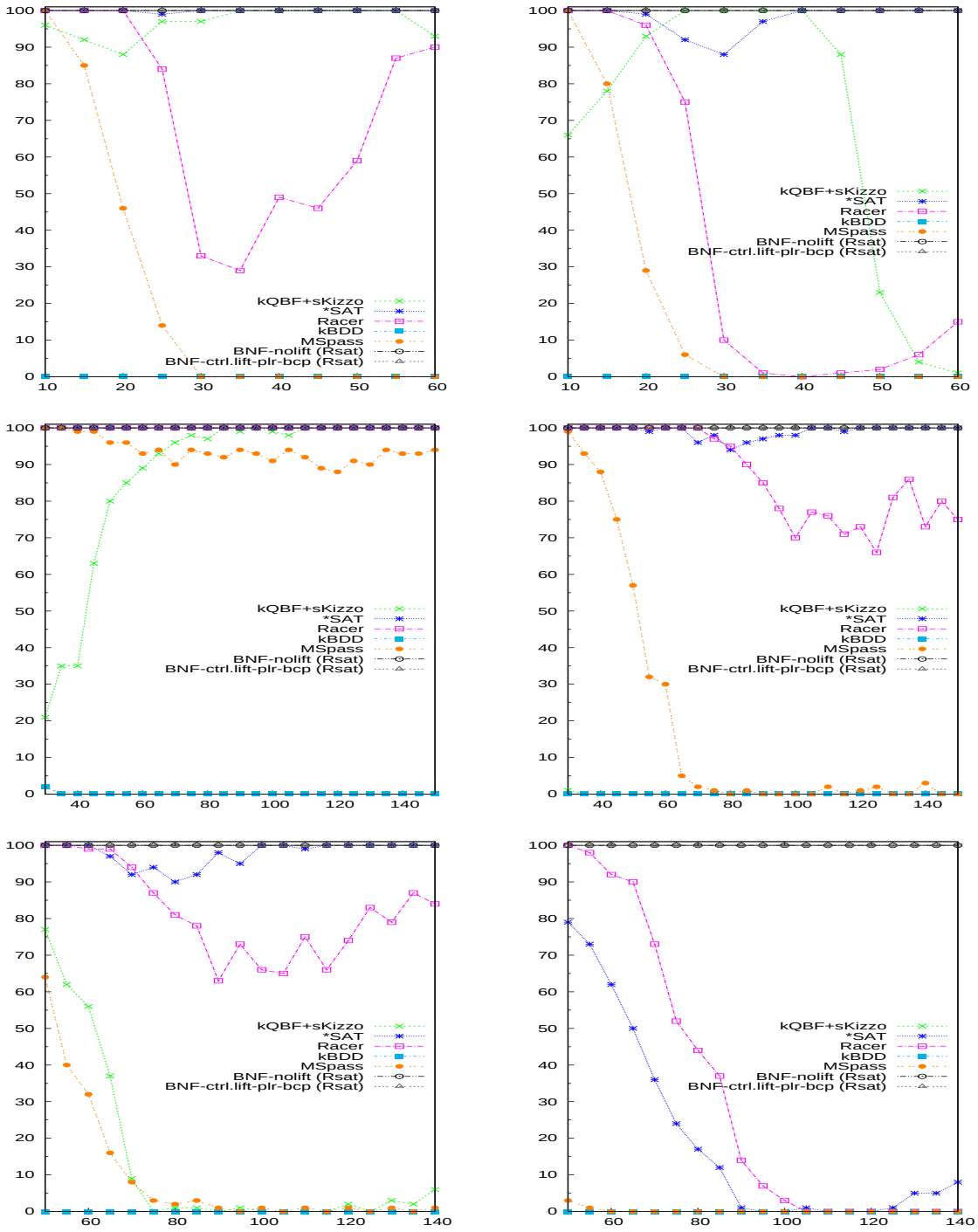


Figure 5.7: Comparison against other approaches on random problems. X axis: $\#clauses/N$. Y axis: % of problems solved within the timeout, 100 samples/point. 1st row: $d = 1$, $p = 0.5$, $N = 8, 9$; 2nd row: $d = 2$, $p = 0.6$, $N = 3, 4$; 3rd row: $d = 2$, $p = 0.5$, $N = 3, 4$.

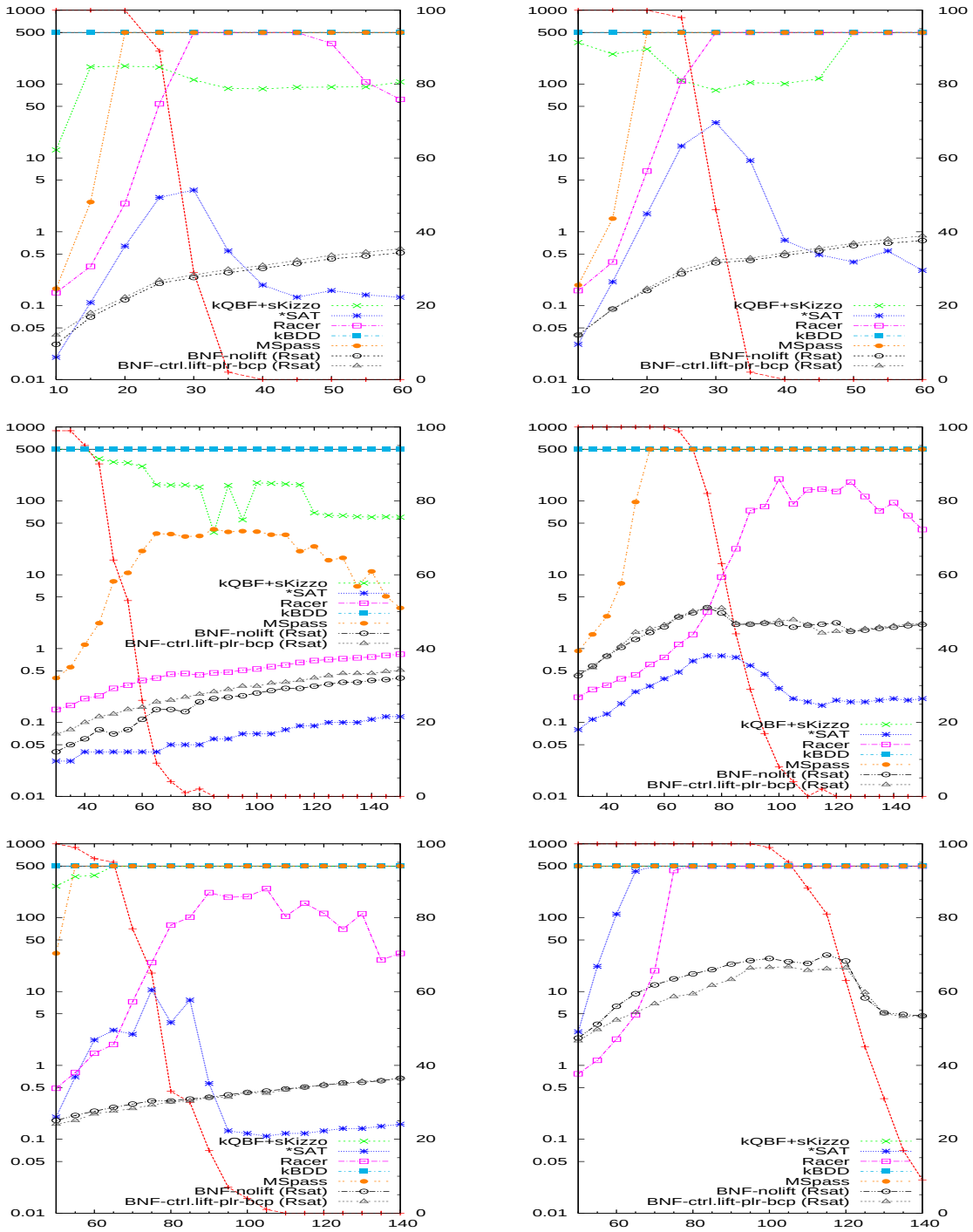


Figure 5.8: Comparison against other approaches on random problems. X axis: $\#clauses/N$. Y axis: median (50th percentile) CPU time, 100 samples/point. 1st row: $d = 1$, $p = 0.5$, $N = 8, 9$; 2nd row: $d = 2$, $p = 0.6$, $N = 3, 4$; 3rd row: $d = 2$, $p = 0.5$, $N = 3, 4$. Background: % of satisfiable instances.

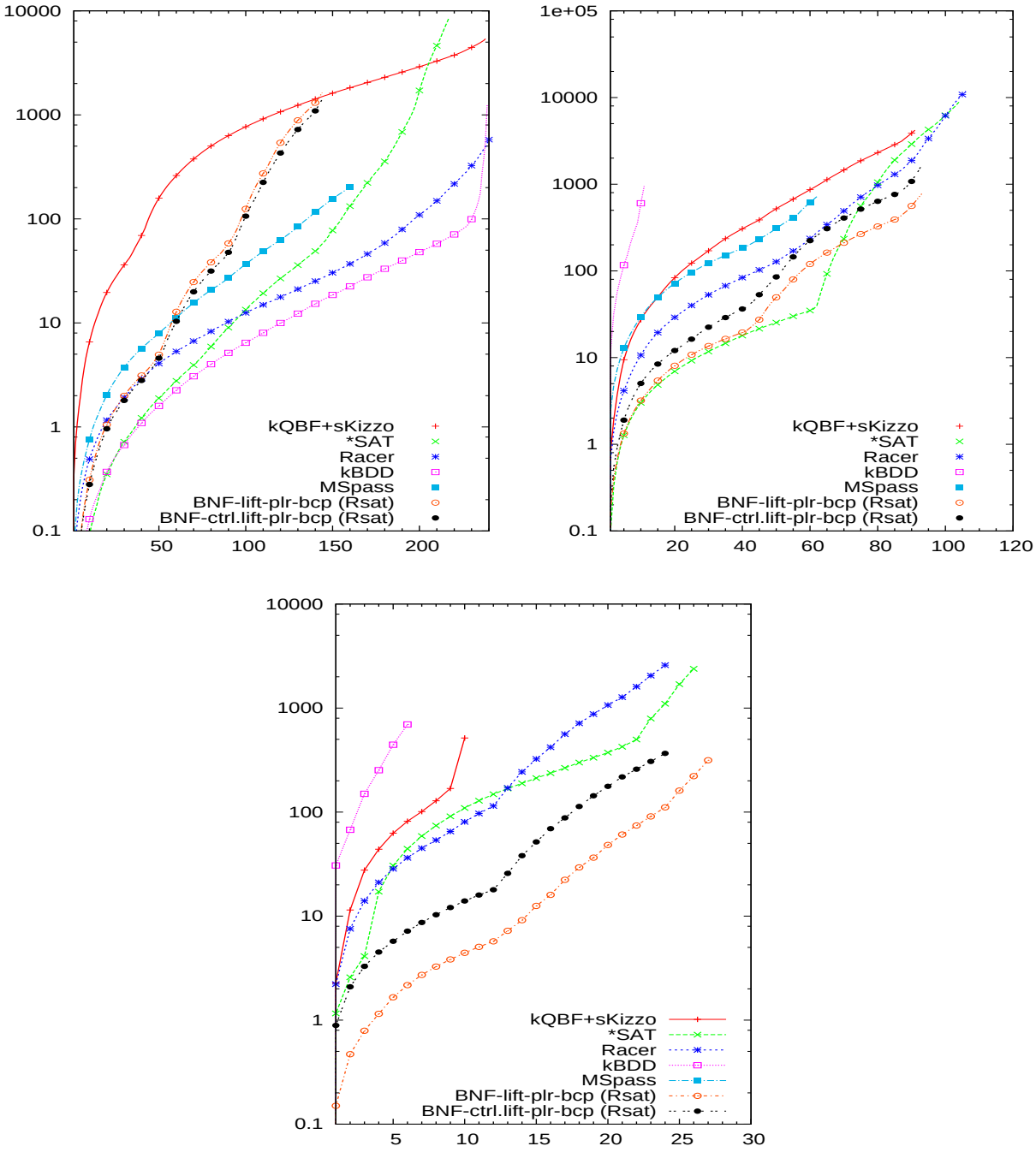


Figure 5.9: Comparison against other approaches on TANCS 2000 problems. X axis: number of solved problems. Y axis: total cumulative CPU time spent. 1st (top-left) to 3th (bottom) plot: easy, medium, hard problems. (Problems are sorted from the easiest to the hardest).

5.6.4 Discussion

As highlighted by Giunchiglia et al. (2000), and Horrocks et al. (2000), the satisfiability problem in modal logics like K_m is characterized by the alternation of two orthogonal components of reasoning: a *Boolean* component, performing Boolean reasoning within each state, and a *modal* component, generating the successor states of each state. The latter must cope with the fact that the candidate models may be up to exponentially big wrt. $depth(\varphi)$, whilst the former must cope with the fact that there may be up to exponentially many candidate (sub)models to explore. In the K_m2SAT +DPLL approach the encoder has to handle the whole modal component (rules (5.8) and (5.9)), whilst the handling of the whole Boolean component is delegated to an external SAT solver.

From the results displayed in Section 5.6.3 we notice that the relative performances of the K_m2SAT +DPLL approach wrt. other state-of-the-art tools range from cases where K_m2SAT +RSAT is much less efficient than other state-of-the-art approaches (e.g., the `k_d4` and `k_t4p` formulas) up to formulas where it is much more efficient (e.g., the `k_ph_p` or the $\Box_m$-CNF formulas with $d = 1$). In the middle stands a large majority of formulas in which the approach competes well against the other state-of-the art tools; in particular, K_m2SAT +RSAT competes very well or even outperforms the other approaches based on translations into different formalisms (the translational approach, the automata-theoretic approach and the QBF-encoding approach).

A simple explanation of the former fact could be that the K_m2SAT +DPLL approach loses on problems with high modal depth, or where the modal component of reasoning dominates (like, e.g., the `k_d4` and `k_t4p` formulas), and wins on problems where the Boolean component of reasoning dominates (like, e.g., the `k_ph_n` or the $\Box_m$-CNF formulas with $d = 1$), and it is competitive for formulas in which both components are relevant.

We notice, however, that K_m2SAT +RSAT wins in the hard category of TANCS 2000 benchmarks, with modal depths greater than 50, and on `k_branch_n`, where the search is dominated by the modal component. [33] In fact, we recall that reducing the Boolean component of reasoning may produce a reduction also of the modal reasoning effort, because it may reduce the number of successor states to analyze (e.g. Sebastiani, 2007a, 2007b). Thus, e.g., techniques like on-the-fly BCP, although exploiting only purely-Boolean properties, may produce not only a drastic pruning of the Boolean search, but also a drastic reduction in the size of the model investigated, because they cut a priori the amount of successor states to expand.

5.7 Contributions and Lesson Learned

In this first approach to automated reasoning in Description Logics we have explored the idea of encoding $K_m/\mathcal{ALC}$-satisfiability into SAT, so that to be handled by state-of-the-

[33] The `k_branch_n` formulas are very hard from the perspective of modal reasoning, because they require finding one model $\mathcal{M}$ with $2^{d+1} - 1$ states (Halpern & Moses, 1992), but no Boolean reasoning within each state is really required (Giunchiglia et al., 2000; Horrocks et al., 2000): e.g., *SAT solves `k_branch_n(d)` with $2^{d+1} - 1$ calls to its embedded DPLL engine, one for each state of $\mathcal{M}$, each call solved by BCP only.

art SAT tools. We have showed that, despite the intrinsic risk of blowup in the size of the encoded formulas, the performances of this approach are comparable with those of current state-of-the-art tools on a rather extensive variety of empirical tests. In particular, the performance of our approach are surprisingly interesting for practical problems.

In this first part of our work we show that, at least for K_m-satisfiability, by exploiting some smart optimizations in the encoding phase, the SAT-encoding approach becomes competitive in practice with previous approaches. To this extent, the contributions of this part of our work are manyfold.

- As a byproduct, we have obtained an empirical evaluation of current tools for K_m-satisfiability available, which is very extensive in terms of both amount and variety of benchmarks and of number and representativeness of the tools evaluated. We are not aware of any other such evaluation in the recent literature.
- We propose a basic encoding of K_m formulas into purely-propositional ones, and prove that the encoding is satisfiability-preserving.
- We describe some optimizations of the encoding, both in form of preprocessing and of on-the-fly simplification. These techniques allow for significant (and in some cases dramatic) reductions in the size of the resulting Boolean formulas, and in performances of the SAT solver thereafter.
- Through our empirical comparison against the main state-of-the-art tools available, we show that, despite the NP-vs.-PSPACE issue, this approach can handle most or all the problems which are at the reach of the other approaches, with performances which are comparable with, and sometimes even better than, those of the current and highly-optimized state-of-the-art tools.

In our perspective, this last point is the most surprising contribution of this first approach, which motivated us to continue in this research stream and to step to harder problems. We also stress the fact that with our approach the encoder can be interfaced with every SAT solver in a plug-and-play manner, so that to benefit for free of every improvement in the technology of SAT solvers which has been or will be made available.

With the experience acquired in this approach to the throughly investigated logic $\mathcal{ALC}$ we aim at explore the opportunity of threat strongest ontologies reasoning on more expressive Description Logics and for more complex form of reasoning (e.g., including TBoxes).

5.8 Appendix: The Proof of Correctness & Completeness

Some Further Notation

Let ψ be a K_m-formula. We denote by $\overline{\psi}$ the representation of $\neg\psi$ in the current formalism: in NNF, $\overline{\Diamond_r\psi} \stackrel{\text{def}}{=} \Box_r\overline{\psi}$, $\overline{\Box_r\psi} \stackrel{\text{def}}{=} \Diamond_r\overline{\psi}$, $\overline{\psi_1 \wedge \psi_2} \stackrel{\text{def}}{=} \overline{\psi_1} \vee \overline{\psi_2}$, $\overline{\psi_1 \vee \psi_2} \stackrel{\text{def}}{=} \overline{\psi_1} \wedge \overline{\psi_2}$, $\overline{A_i} \stackrel{\text{def}}{=} \neg A_i$, $\overline{\neg A_i} \stackrel{\text{def}}{=} A_i$; in BNF, $\overline{\neg\Box_r\psi} \stackrel{\text{def}}{=} \Box_r\psi$, $\overline{\Box_r\psi} \stackrel{\text{def}}{=} \neg\Box_r\psi$, $\overline{\psi_1 \wedge \psi_2} \stackrel{\text{def}}{=} \overline{\psi_1} \vee \overline{\psi_2}$, $\overline{\psi_1 \vee \psi_2} \stackrel{\text{def}}{=} \overline{\psi_1} \wedge \overline{\psi_2}$, $\overline{A_i} \stackrel{\text{def}}{=} \neg A_i$, $\overline{\neg A_i} \stackrel{\text{def}}{=} A_i$.

We represent a truth assignment μ as a set of literals, with the intended meaning that a positive literal A_i (resp. negative literal $\neg A_i$) in μ means that A_i is assigned to true (resp. false). We say that μ *assigns* a literal l if either $l \in \mu$ or $\neg l \in \mu$. We say that a literal l occurs in a Boolean formula ϕ iff the atom of l occurs in ϕ.

Let $\mathcal{M}$ denote a Kripke model, and let σ be the label of a generic state u_σ in $\mathcal{M}$. We label (and denote) by "1" the root state of $\mathcal{M}$. By "$\langle\sigma : \psi\rangle \in \mathcal{M}$" we mean that $u_\sigma \in \mathcal{M}$ and $\mathcal{M}, u_\sigma \models \psi$. Thus, for every σ s.t. $u_\sigma \in \mathcal{M}$, either $\langle\sigma : \psi\rangle \in \mathcal{M}$ or $\langle\sigma : \overline{\psi}\rangle \in \mathcal{M}$.

For convenience, instead of (5.9) sometimes we use the equivalent definition:

$$Def(\sigma,\ \nu^r) \stackrel{\text{def}}{=} (L_{\langle\sigma,\ \nu^r\rangle} \rightarrow \bigwedge_{\substack{\text{for every}\\ \langle\sigma,\pi^{r,i}\rangle}} (L_{\langle\sigma,\ \pi^{r,i}\rangle} \rightarrow L_{\langle\sigma.i,\ \nu_0^r\rangle})) \wedge \bigwedge_{\substack{\text{for every}\\ \langle\sigma,\pi^{r,i}\rangle}} Def(\sigma.i,\ \nu_0^r) \quad (5.20)$$

Notice that each $Def(\sigma,\ \psi)$ in (5.6), (5.7), (5.8), (5.20) is written in the general form

$$(L_{\langle\sigma,\ \psi\rangle} \rightarrow \Phi_{\langle\sigma,\psi\rangle}) \wedge \bigwedge_{\langle\sigma',\psi'\rangle} Def(\sigma',\ \psi'). \quad (5.21)$$

We call *definition implication* for $Def(\sigma,\ \psi)$ the expressions "$(L_{\langle\sigma,\ \psi\rangle} \rightarrow \Phi_{\langle\sigma,\psi\rangle})$".

Soundness and Completeness of K_m2SAT

Let φ be a K_m-formula. We prove the following theorem, which states the soundness and completeness of K_m2SAT as defined in Section 5.3

Theorem 3. *A K_m-formula φ is K_m-satisfiable if and only if the corresponding $K_m2SAT(\varphi)$ is satisfiable.*

Proof. It is a direct consequence of the following Lemmas 4 and 5. □

Lemma 4. *Given a K_m-formula φ, if $K_m2SAT(\varphi)$ is satisfiable, then there exists a Kripke model $\mathcal{M}$ s.t. $\mathcal{M}, 1 \models \varphi$.*

Proof. Let μ be a total truth assignment satisfying $K_m2SAT(\varphi)$ as defined by Definition 1. We build from μ a Kripke model $\mathcal{M} = \langle\mathcal{U}, \mathcal{L}, \mathcal{R}_1, \ldots, \mathcal{R}_m\rangle$ as follows:

$$\mathcal{U} \stackrel{\text{def}}{=} \{\sigma\ :\ A_{\langle\sigma,\ \psi\rangle}\ occurs\ in\ K_m2SAT(\varphi)\ for\ some\ \psi\} \quad (5.22)$$

$$\mathcal{L}(\sigma, A_i) \stackrel{\text{def}}{=} \begin{cases} True & if\ L_{\langle\sigma,\ A_i\rangle} \in \mu \\ False & if\ \neg L_{\langle\sigma,\ A_i\rangle} \in \mu \end{cases} \quad (5.23)$$

$$\mathcal{R}_r \stackrel{\text{def}}{=} \{\langle\sigma, \sigma.i\rangle\ :\ L_{\langle\sigma,\ \pi^{r,i}\rangle} \in \mu\}. \quad (5.24)$$

We show by induction on the structure of φ that, for every $\langle \sigma, \psi \rangle$ s.t. $L_{\langle \sigma,\ \psi \rangle}$ occurs on $K_m 2SAT(\varphi)$,

$$\langle \sigma : \psi \rangle \in \mathcal{M} \ \text{ if } \ L_{\langle \sigma,\ \psi \rangle} \in \mu. \tag{5.25}$$

Base

$\psi = A_i$ or $\psi = \neg A_i$. Then (5.25) follows trivially from (5.23).

Step

$\psi = \alpha$. Let $L_{\langle \sigma,\ \alpha \rangle} \in \mu$.

As μ satisfies (5.6), $L_{\langle \sigma,\ \alpha_i \rangle} \in \mu$ for every $i \in \{1, 2\}$.

By inductive hypothesis, $\langle \sigma : \alpha_i \rangle \in \mathcal{M}$ for every $i \in \{1, 2\}$.

Then, by definition, $\langle \sigma : \alpha \rangle \in \mathcal{M}$.

Thus, $\langle \sigma : \alpha \rangle \in \mathcal{M}$ if $L_{\langle \sigma,\ \alpha \rangle} \in \mu$.

$\psi = \beta$. Let $L_{\langle \sigma,\ \beta \rangle} \in \mu$.

As μ satisfies (5.7), $L_{\langle \sigma,\ \beta_i \rangle} \in \mu$ for some $i \in \{1, 2\}$.

By inductive hypothesis, $\langle \sigma : \beta_i \rangle \in \mathcal{M}$ for some $i \in \{1, 2\}$.

Then, by definition, $\langle \sigma : \beta \rangle \in \mathcal{M}$.

Thus, $\langle \sigma : \beta \rangle \in \mathcal{M}$ if $L_{\langle \sigma,\ \beta \rangle} \in \mu$.

$\psi = \pi^{r,j}$. Let $L_{\langle \sigma,\ \pi^{r,j} \rangle} \in \mu$.

As μ satisfies (5.8), $L_{\langle \sigma.j,\ \pi_0^{r,j} \rangle} \in \mu$.

By inductive hypothesis, $\langle \sigma.j : \pi_0^{r,j} \rangle \in \mathcal{M}$.

Then, by definition and by (5.24), $\langle \sigma : \pi^{r,j} \rangle \in \mathcal{M}$.

Thus, $\langle \sigma : \pi^{r,j} \rangle \in \mathcal{M}$ if $L_{\langle \sigma,\ \pi^{r,j} \rangle} \in \mu$.

$\psi = \nu^r$. Let $L_{\langle \sigma,\ \nu^r \rangle} \in \mu$.

As μ satisfies (5.9), for every $\langle \sigma, \pi^{r,i} \rangle$ s.t. $L_{\langle \sigma,\ \pi^{r,i} \rangle} \in \mu$, we have that $L_{\langle \sigma.i,\ \nu_0^r \rangle} \in \mu$.

By inductive hypothesis, we have that $\langle \sigma : \pi^{r,i} \rangle \in \mathcal{M}$ and $\langle \sigma.i : \nu_0^r \rangle \in \mathcal{M}$.

Then, by definition and by (5.24), $\langle \sigma : \nu^r \rangle \in \mathcal{M}$.

Thus, $\langle \sigma : \nu^r \rangle \in \mathcal{M}$ if $L_{\langle \sigma,\ \nu^r \rangle} \in \mu$.

If $\mu \models K_m 2SAT(\varphi)$, then $A_{\langle 1,\ \varphi \rangle} \in \mu$. Thus, by (5.25), $\langle 1 : \varphi \rangle \in \mathcal{M}$, i.e., $\mathcal{M}, 1 \models \varphi$. $\square$

Lemma 5. *Given a K_m-formula φ, if there exists a Kripke model $\mathcal{M}$ s.t. $\mathcal{M}, 1 \models \varphi$, then $K_m2SAT(\varphi)$ is satisfiable.*

Proof. Let $\mathcal{M}$ be a Kripke model s.t. $\mathcal{M}, 1 \models \varphi$. We build from $\mathcal{M}$ a truth assignment μ for $K_m2SAT(\varphi)$ (defined by Definition 1) recursively as follows: [34]

$$\mu \stackrel{\text{def}}{=} \mu_{\mathcal{M}} \cup \mu_{\overline{\mathcal{M}}} \tag{5.26}$$

$$\begin{aligned} \mu_{\mathcal{M}} &\stackrel{\text{def}}{=} \{L_{\langle\sigma,\ \psi\rangle} \in K_m2SAT(\varphi)\ :\ \langle\sigma,\psi\rangle \in \mathcal{M}\} \\ &\cup\ \{\neg L_{\langle\sigma,\ \psi\rangle} \in K_m2SAT(\varphi)\ :\ \langle\sigma,\overline{\psi}\rangle \in \mathcal{M}\} \end{aligned} \tag{5.27}$$

$$\mu_{\overline{\mathcal{M}}} \stackrel{\text{def}}{=} \mu_{\pi\nu} \cup \mu_{\alpha\beta} \cup \mu_{\mathcal{A}} \tag{5.28}$$

$$\begin{aligned} \mu_{\pi\nu} &\stackrel{\text{def}}{=} \{\neg L_{\langle\sigma,\ \pi^{r,i}\rangle} \in K_m2SAT(\varphi)\ :\ \sigma \notin \mathcal{M}\} \\ &\cup\ \{L_{\langle\sigma,\ \nu^r\rangle} \in K_m2SAT(\varphi)\ :\ \sigma \notin \mathcal{M}\} \end{aligned} \tag{5.29}$$

$$\begin{aligned} \mu_{\alpha\beta} &\stackrel{\text{def}}{=} \{\neg L_{\langle\sigma,\ \alpha\rangle} \in K_m2SAT(\varphi)\ :\ \sigma \notin \mathcal{M} \text{ and } \neg L_{\langle\sigma,\ \alpha_i\rangle} \in \mu_{\overline{\mathcal{M}}} \text{ for some } i \in \{1,2\}\} \\ &\cup\ \{\neg L_{\langle\sigma,\ \beta\rangle} \in K_m2SAT(\varphi)\ :\ \sigma \notin \mathcal{M} \text{ and } \neg L_{\langle\sigma,\ \beta_i\rangle} \in \mu_{\overline{\mathcal{M}}} \text{ for every } i \in \{1,2\}\}. \end{aligned} \tag{5.30}$$

where $\mu_{\mathcal{A}}$ is a consistent truth assignment for the literals $L_{\langle\sigma,\ A_i\rangle}$ s.t. $A_i \in \mathcal{A}$ and $\sigma \notin \mathcal{M}$.

By construction, for every $L_{\langle\sigma,\ \psi\rangle}$ in $K_m2SAT(\varphi)$, μ assigns $L_{\langle\sigma,\ \psi\rangle}$ to true iff it assigns $L_{\langle\sigma,\ \overline{\psi}\rangle}$ to false and vice versa, so that μ is a consistent truth assignment.

First, we show that $\mu_{\mathcal{M}}$ satisfies the definition implications of all $Def(\sigma,\ \psi)$'s and $Def(\sigma,\ \overline{\psi})$' s.t. $\sigma \in \mathcal{M}$. Let $\sigma \in \mathcal{M}$. We distinguish four cases.

$\psi = \alpha$. Thus $\overline{\psi} = \beta$ s.t. $\beta_1 = \overline{\alpha_1}$ and $\beta_2 = \overline{\alpha_2}$.

- If $\langle\sigma : \alpha\rangle \in \mathcal{M}$ (and hence $\langle\sigma : \beta\rangle \notin \mathcal{M}$), then for both i's $\langle\sigma : \alpha_i\rangle \in \mathcal{M}$ and $\langle\sigma : \beta_i\rangle \notin \mathcal{M}$. Thus, by (5.27), $\{L_{\langle\sigma,\ \alpha_1\rangle}, L_{\langle\sigma,\ \alpha_2\rangle}, \neg L_{\langle\sigma,\ \beta\rangle}\} \subseteq \mu_{\mathcal{M}}$, so that $\mu_{\mathcal{M}}$ satisfies the definition implications of both $Def(\sigma,\ \alpha)$ and $Def(\sigma,\ \beta)$.
- If $\langle\sigma : \alpha\rangle \notin \mathcal{M}$ (and hence $\langle\sigma, \beta\rangle \in \mathcal{M}$), then for some i $\langle\sigma : \alpha_i\rangle \notin \mathcal{M}$ and $\langle\sigma : \beta_i\rangle \in \mathcal{M}$. Thus, by (5.27), $\{\neg L_{\langle\sigma,\ \alpha\rangle}, L_{\langle\sigma,\ \beta_i\rangle}\} \subseteq \mu_{\mathcal{M}}$, so that $\mu_{\mathcal{M}}$ satisfies the definition implications of both $Def(\sigma,\ \alpha)$ and $Def(\sigma,\ \beta)$.

$\psi = \beta$. Like in the previous case, inverting ψ and $\overline{\psi}$.

$\psi = \pi^{r,j}$. Thus $\overline{\psi} = \nu^r$ s.t. $\nu^r_0 = \overline{\pi^{r,j}_0}$.

- If $\langle\sigma : \pi^{r,j}\rangle \in \mathcal{M}$ (and hence $\langle\sigma : \nu^r\rangle \notin \mathcal{M}$), then $\langle\sigma.j : \pi^{r,j}_0\rangle \in \mathcal{M}$. Thus, by (5.27), $\{L_{\langle\sigma.j,\ \pi^{r,j}_0\rangle}, \neg L_{\langle\sigma,\ \nu^r\rangle}\} \subseteq \mu_{\mathcal{M}}$, so that $\mu_{\mathcal{M}}$ satisfies the definition implications of both $Def(\sigma,\ \pi^{r,j})$ and $Def(\sigma,\ \nu^r)$.

[34] We assume that $\mu_{\mathcal{M}}$, $\mu_{\pi\nu}$ and $\mu_{\alpha\beta}$ are generated *in order*, so that $\mu_{\alpha\beta}$ is generated recursively starting from $\mu_{\pi\nu}$. Intuitively, $\mu_{\mathcal{M}}$ assigns the literals $L_{\langle\sigma,\ \psi\rangle}$ s.t. $\sigma \in \mathcal{M}$ in such a way to mimic $\mathcal{M}$; $\mu_{\overline{\mathcal{M}}}$ assigns the other literals in such a way to mimic the fact that no state outside those in $\mathcal{M}$ is generated (i.e., all $L_{\langle\sigma,\ \pi\rangle}$'s are assigned false and the $L_{\langle\sigma,\ \nu\rangle}$'s, $L_{\langle\sigma,\ \alpha\rangle}$'s, $L_{\langle\sigma,\ \beta\rangle}$'s are assigned consequently).

– If $\langle \sigma : \pi^{r,j} \rangle \notin \mathcal{M}$ (and hence $\langle \sigma : \nu^r \rangle \in \mathcal{M}$), then by (5.27) $\neg L_{\langle \sigma,\ \pi^{r,j} \rangle} \in \mu_{\mathcal{M}}$, so that $\mu_{\mathcal{M}}$ satisfies the definition implications of $Def(\sigma,\ \pi^{r,j})$.
As far as $Def(\sigma,\ \nu^r)$ is concerned, we partition the clauses in (5.9):

$$((L_{\langle \sigma,\ \nu^r \rangle} \wedge L_{\langle \sigma,\ \pi^{r,i} \rangle}) \rightarrow L_{\langle \sigma.i,\ \nu_0^r \rangle}) \tag{5.31}$$

into two subsets. The first is the set of clauses (5.31) for which $\langle \sigma : \pi^{r,i} \rangle \in \mathcal{M}$. As $\langle \sigma : \nu^r \rangle \in \mathcal{M}$, we have that $\langle \sigma.i : \nu_0^r \rangle \in \mathcal{M}$. Thus, by (5.27), $L_{\langle \sigma.i,\ \nu_0^r \rangle} \in \mu_{\mathcal{M}}$, so that $\mu_{\mathcal{M}}$ satisfies (5.31). The second is the set of clauses (5.31) for which $\langle \sigma : \pi^{r,i} \rangle \notin \mathcal{M}$. By (5.27) we have that $\neg L_{\langle \sigma,\ \pi^{r,i} \rangle} \in \mu_{\mathcal{M}}$, so that $\mu_{\mathcal{M}}$ satisfies (5.31). Thus, $\mu_{\mathcal{M}}$ satisfies the definition implications also of $Def(\sigma,\ \nu^r)$.

$\psi = \nu^r$. Like in the previous case, inverting ψ and $\overline{\psi}$.

Notice that, if $\sigma \notin \mathcal{M}$, then $\sigma.i \notin \mathcal{M}$ for every i. Thus, for every $Def(\sigma,\ \psi)$ s.t. $\sigma \notin \mathcal{M}$, all atoms in the implication definition of $Def(\sigma,\ \psi)$ are not assigned by $\mu_{\mathcal{M}}$.

Second, we show by induction on the recursive structure of $\mu_{\overline{\mathcal{M}}}$ that $\mu_{\overline{\mathcal{M}}}$ satisfies the definition implications of all $Def(\sigma,\ \psi)$'s and $Def(\sigma,\ \overline{\psi})$'s s.t. $\sigma \notin \mathcal{M}$. Let $\sigma \notin \mathcal{M}$.

As a base step, by (5.29), $\mu_{\pi\nu}$ satisfies the definition implications of all $Def(\sigma,\ \pi^{r,i})$'s and $Def(\sigma,\ \nu^r)$'s because it assigns false to all $L_{\langle \sigma,\ \pi^{r,i} \rangle}$'s. Indeed, $\mu_{\mathcal{A}}$ assigns every literal of the type $L_{\langle \sigma,\ A_i \rangle}$ s.t $A_i \in \mathcal{A}$ and $\sigma \notin \mathcal{M}$ (notice that all the $Def(\sigma,\ A_i)$'s definitions are trivially satisfied and don't contain any definition implications).

As inductive step, we show on the inductive structure of $\mu_{\alpha\beta}$ that $\mu_{\alpha\beta}$ satisfies the definition implications of all $Def(\sigma,\ \alpha)$'s and $Def(\sigma,\ \beta)$'s

Let $\psi \stackrel{\text{def}}{=} \alpha$ and $\overline{\psi} = \beta$ s.t. $\beta_i = \overline{\alpha_i}$ (or vice versa). Then we have that:

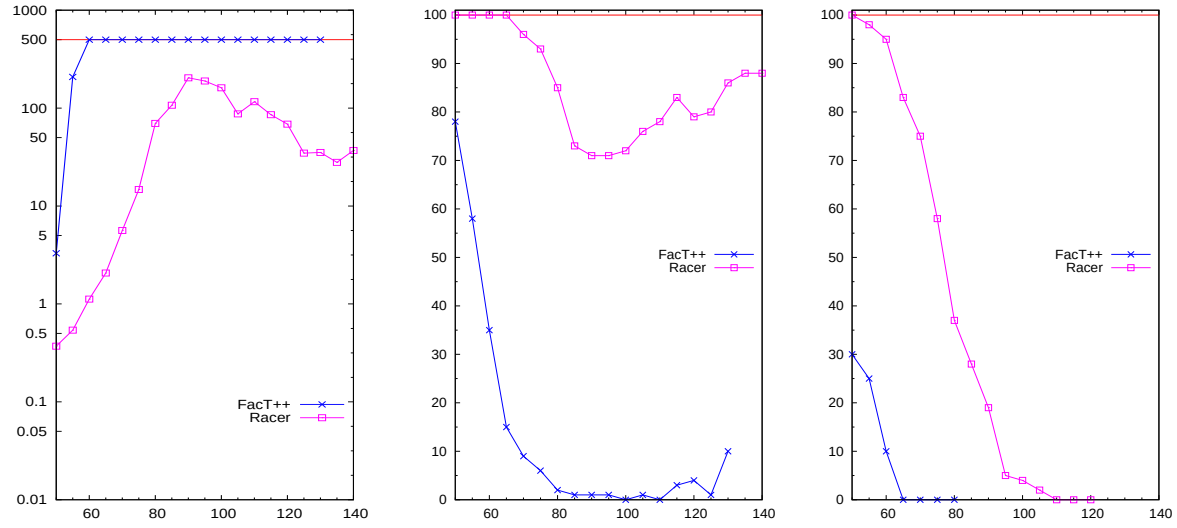
– if both $L_{\langle \sigma,\ \alpha_i \rangle}$'s (respectively at least one $L_{\langle \sigma,\ \beta_i \rangle}$) are assigned true by $\mu_{\overline{\mathcal{M}}}$, then the definition implications of $Def(\sigma,\ \alpha)$ (respectively $Def(\sigma,\ \beta)$) is already trivially satisfied;

– if at least one $L_{\langle \sigma,\ \alpha_i \rangle}$ (respectively both $L_{\langle \sigma,\ \beta_i \rangle}$'s) is assigned false by $\mu_{\overline{\mathcal{M}}}$, then by (5.30) $L_{\langle \sigma,\ \alpha \rangle}$ (respectively $L_{\langle \sigma,\ \beta \rangle}$) is assigned false by $\mu_{\alpha\beta}$, which satisfies the definition implication of $Def(\sigma,\ \alpha)$ (respectively $Def(\sigma,\ \beta)$).

Thus $\mu_{\overline{\mathcal{M}}}$ satisfies the definition implications of all the $Def(\sigma,\ \psi)$'s and $Def(\sigma,\ \overline{\psi})$'s s.t. $\sigma \notin \mathcal{M}$.

On the whole, $\mu \models Def(\sigma,\ \psi)$ for every $Def(\sigma,\ \psi)$. By construction, $\mu_{\mathcal{M}} \models A_{\langle 1,\ \varphi \rangle}$ since $\langle 1 : \varphi \rangle \in \mathcal{M}$. Therefore $\mu \models K_m 2SAT(\varphi)$. □

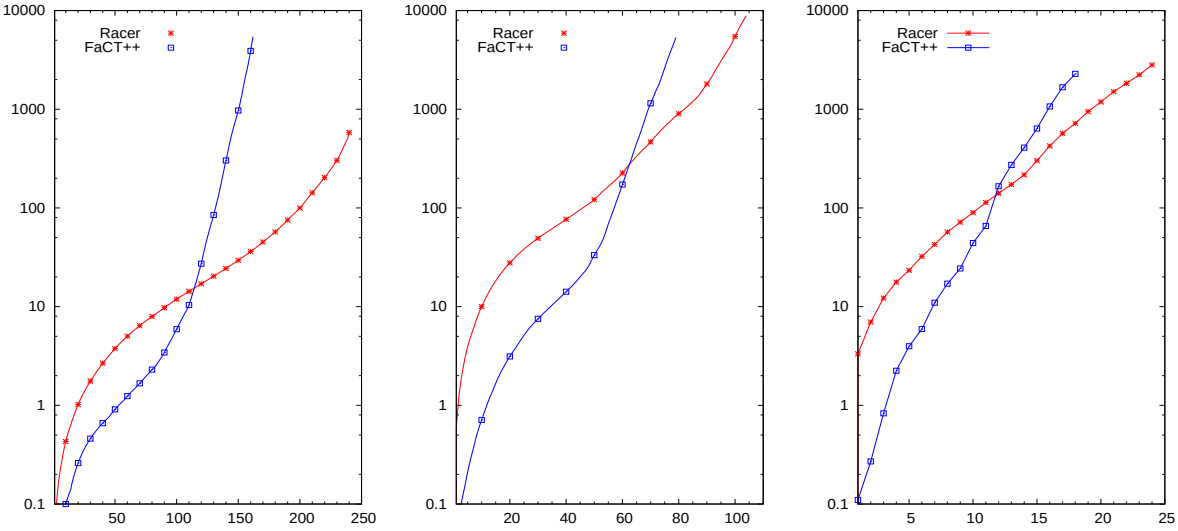
5.9 Appendix: Evaluation Trials and Auxiliary Plots



(a) rand. $d = 2, p = 0.5, N = 3$, median CPU time

(b) rand. $d = 2, p = 0.5, N = 3$, % term. prob.

(c) rand. $d = 2, p = 0.5, N = 4$, % term. prob.

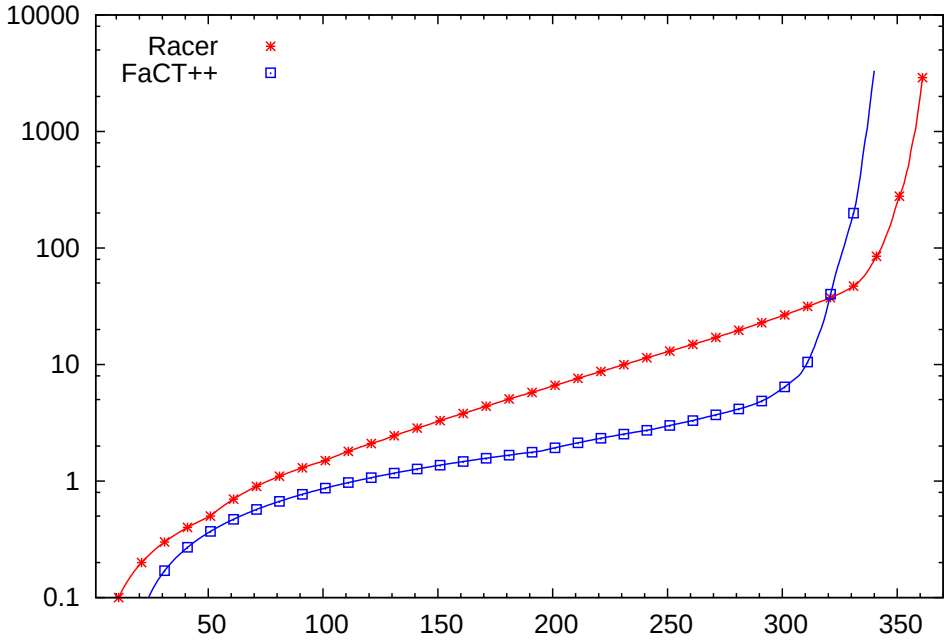


(d) TANCS easy problems

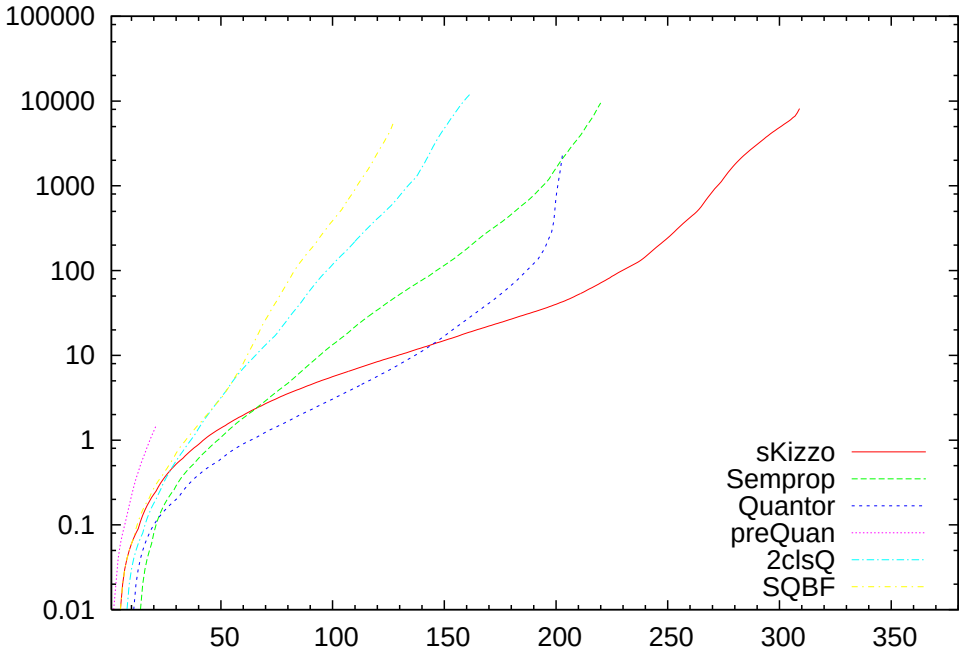
(e) TANCS medium problems

(f) TANCS hard problems

Figure 5.10: Preliminary comparison between RACER and FACT++ on the different benchmarks. 1st row: on TANCS 2000 problems; X axis: number of solved problems, Y axis: total cumulative CPU time spent. 1nd row: random problems; X axis: $\#clauses/N$, Y axis: median CPU time or % of problems solved within the timeout (100 samples/point). (FACT++ is always fixed to 500 sec. CPU times for every other random plot).



(a) RACER and FACT++ comparison.



(b) QBF solvers comparison.

Figure 5.11: Tool trials on the LWB benchmarks. X axis: # of problems solved; Y axis: total (cumulative) CPU time spent.

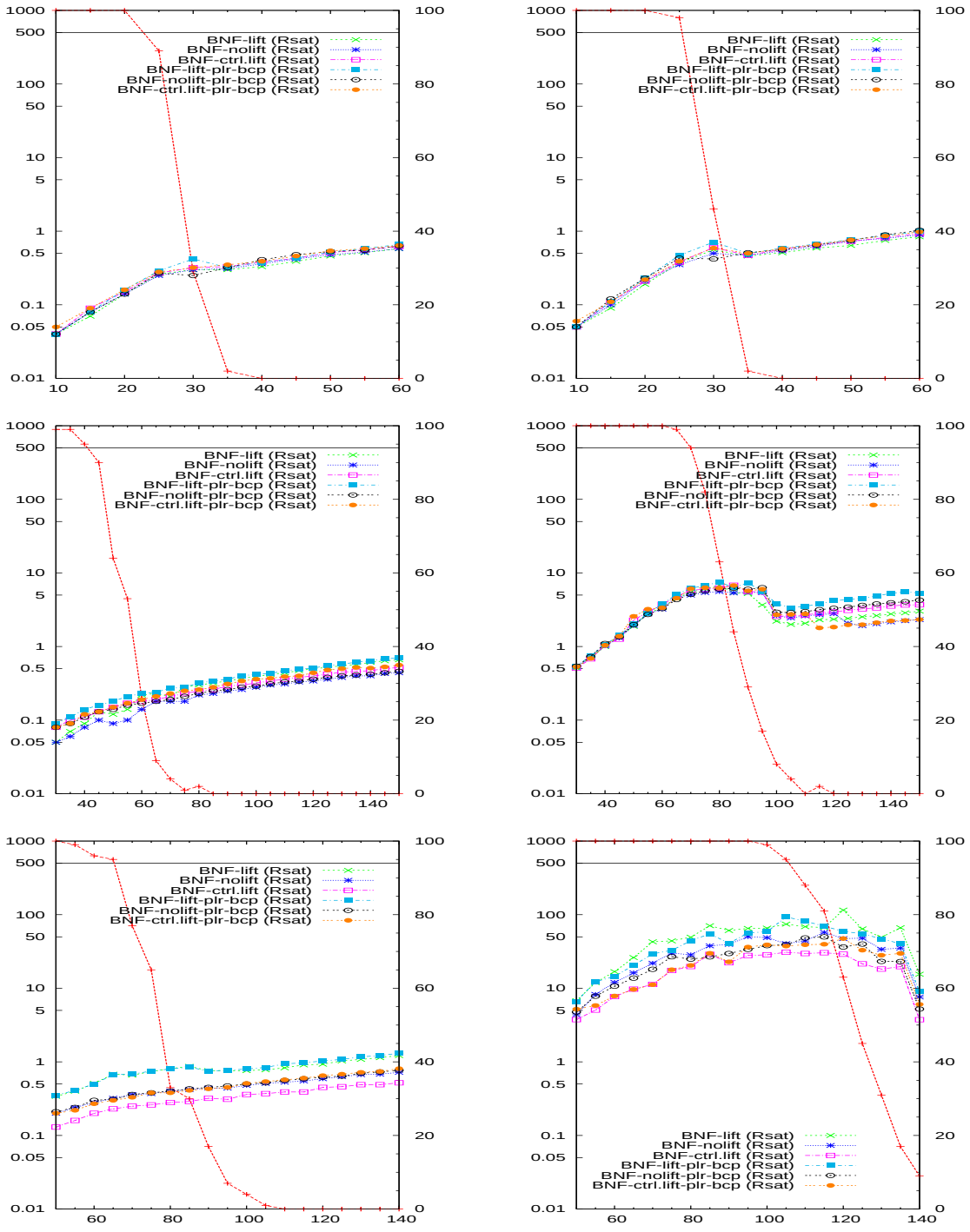


Figure 5.12: Comparison among different variants of $K_m 2SAT$ +RSAT on random problems. X axis: $\#clauses/N$. Y axis: 90th percentile of CPU time, 100 samples/point. 1st row: $d = 1$, $p = 0.5$, $N = 8, 9$; 2nd row: $d = 2$, $p = 0.6$, $N = 3, 4$; 3rd row: $d = 2$, $p = 0.5$, $N = 3, 4$. Background: % of satisfiable instances.

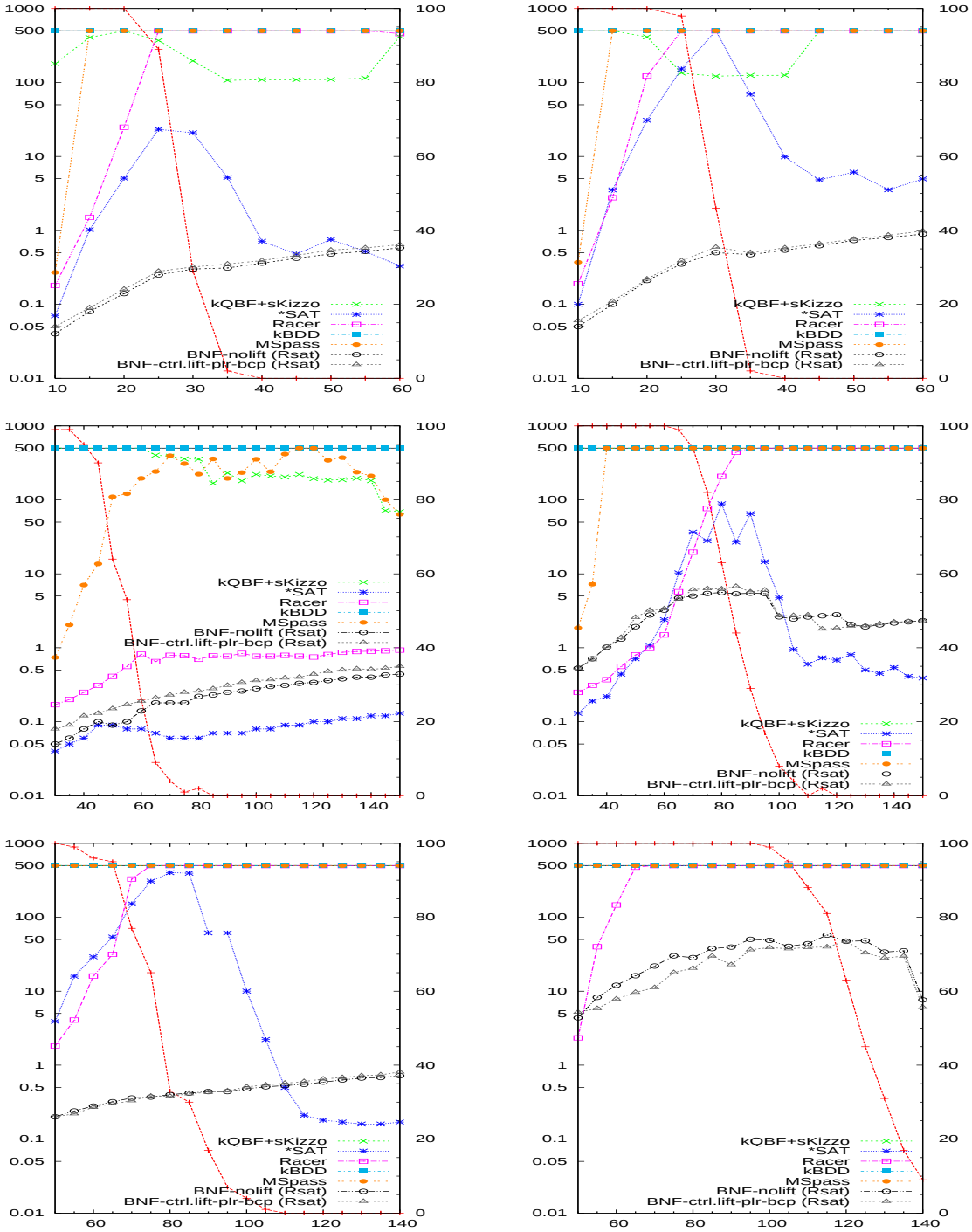


Figure 5.13: Comparison against other approaches on random problems. X axis: $\#clauses/N$. Y axis: 90th percentile of CPU time, 100 samples/point. 1st row: $d = 1$, $p = 0.5$, $N = 8, 9$; 2nd row: $d = 2$, $p = 0.6$, $N = 3, 4$; 3rd row: $d = 2$, $p = 0.5$, $N = 3, 4$. Background: % of satisfiable instances.

Chapter 6

Handling Number Restrictions as SMT Problems

On the one hand the recent explosion of Semantic Web applications often require the use of numbers for expressing cardinality restrictions. The more ontologies are attracting applications and new domains to be expressed, the more there is the quest of efficiently handling number restrictions. In particular, developing techniques for optimized reasoning with qualified number restrictions has become an important goal because qualified number restrictions have been added to the OWL 2 standard. However actual techniques often lack of efficiency in handling those features, especially when the number of restrictions or the values involved are high. On the other hand the manifold problem of reasoning including numerical constraints is a well-established and thoroughly investigated problem in the SMT community, in which a lot of effort is continuously spent in order to enhance the efficiency of reasoning techniques for such kind of problems.

Thus, in this chapter we move further in our investigation tackling TBox reasoning and numeric logical constructors. Here we propose and improve a novel approach for concept satisfiability in acyclic $\mathcal{ALCQ}$ ontologies. The idea is to encode an $\mathcal{ALCQ}$ ontology into a SMT formula modulo a subset of Linear Arithmetic under the Integers, and to exploit the power of modern SMT solvers to compute every concept satisfiability query on the given input ontology.

6.1 Other Approaches and Related Works

The problem of reasoning with qualified number restrictions in Description Logic has been throughly investigated since the very first research steps in the automated reasoning in Modal and Description Logics till today (Ohlbach & Koehler, 1997, 1999; Hollunder & Baader, 1991; Horrocks, Sattler, & Tobies, 2000b; Horrocks et al., 2000a; Haarslev & Möller, 2001; Faddoul et al., 2008; Farsiniamarj & Haarslev, 2010).

The quest of efficient procedures to reason on very expressive Description Logics arising especially from the field of Semantic Web, indeed, has given new vigor and prominence to this stream of research. In particular the research community is spending

a lot of effort in finding alternative solutions to the traditional tableau-based method for handling qualified number restrictions.

Most DL tableau algorithms (see, e.g. Hollunder & Baader, 1991; Horrocks et al., 2000a; Baader et al., 2003) check the satisfiability of concept including qualified number restrictions by creating the necessary number of individuals (called fillers) satisfying all the at-least restrictions and, then, they try to reduce the number of such individuals by non-deterministically merging pairs of fillers until the upper bounds specified in the at-most restrictions are satisfied. Many optimization like, e.g., dependency-directed backtracking (Horrocks et al., 2000a), has been proposed in order to improve this method (for more ones we refer the reader to the literature, Baader et al., 2003). However, searching for a model in such an arithmetically uninformed and blind way can become very inefficient, especially when bigger numbers occurr in qualified number restrictions or several number restrictions interact. To the best of our knowledge this calculus still serves as reference in most tableau-based OWL reasoners (e.g., PELLET Sirin et al., 2007, or FACT++ Tsarkov & Horrocks, 2006) for implementing reasoning about qulaified number restrictions. The only exception is RACER (Haarslev & Moeller, 2001) where conceptual qualified number restriction reasoning is based on an algebraic approach (Haarslev et al., 2001) that integrates integer linear programming with DL-tableau methods.

Various alternative algebraic methods to the traditional tableau algorithms have been proposed (Ohlbach & Koehler, 1999; Haarslev & Möller, 2001; Haarslev et al., 2001; Farsiniamarj & Haarslev, 2010), following the idea of enrich the tableau-based reasoning engine with calculus which benefit from arithmetic methods. Haarslev et all., in particular, have performed many attempts in this research direction (Haarslev & Möller, 2001; Haarslev et al., 2001; Farsiniamarj & Haarslev, 2010; Faddoul & Haarslev, 2010); a wider literature on these attempts and other arithmetic-based approaches can be found in the lately cited works. Among these, Faddoul et al. (2008, and others) recently developed a hybrid approach combining the standard tableau methods with an inequation solver. Such hybrid calculus (initially proposed for $\mathcal{ALCQ}$, then extended up to the logic $\mathcal{SHOQ}$) is based on a standard tableau algorithm for $\mathcal{SH}$ extended to deal with qualified number restrictions and includes an inequation solver based on integer linear programming. The algorithm encodes number restrictions into a set of inequations using the so-called atomic decomposition technique (Ohlbach & Koehler, 1999). In a nutshell the idea is to partition the possible role fillers in all the exponentially many conjunctions of the concepts involved in the qualified number restrictions, and to encode the cardinality constraints for the partitions in a system of integer inequations. The set of inequations is processed by the inequation solver which finds, if possible, a minimal non-negative integer solution (i.e. a distribution of role fillers constrained by number restrictions) satisfying the inequations. The algorithm ensures that such a distribution of role fillers also satisfies the logical restrictions.

Since this hybrid algorithm collects all the information about arithmetic expressions before creating any role filler, it will not satisfy any at-least restriction by violating an

at-most restriction and there is no need for merging mechanisms of role fillers. Moreover, the hybrid approach has the benefit of being not affected by the values of numbers occurring in number restrictions and of allowing for creating only one, so-called, proxy individual (thus, only one branch in the tableau) representing a distinct set of role fillers with the same logical properties. On the contrary, the main drawback of this approach is that atomic decomposition always results in an exponential number of integer variables (and possible proxy-individuals) wrt. the number of coexisting number restrictions.

We also mention the SMT-based approach outlined by Gasse and Haarslev (2009). The idea is to develop an SMT-like DL reasoner for the expressive logic $\mathcal{SHOQ}$ which follows the typical architecture of an SMT-solver (see Section 4.2). In brief, Gasse and Haarslev (2009) proposes to separate each problem in two components of reasoning: a propositional component which is handled from the embedded SAT-solver and a "background theory" component handled by a specific $\mathcal{T}$-solver. Through the encoding proposed by Sebastiani and Vescovi (2006) (see Section 5.4) the $\mathcal{ALC}$ part of the input $\mathcal{SHOQ}$ problem is reduced to a Boolean abstraction including atoms in the logical constructors which are not expressible in $\mathcal{ALC}$. Then the $\mathcal{T}$-solver component is responsible to verify if an assignment to such Boolean abstraction satisfies the logical axioms that are not expressible in $\mathcal{ALC}$, hence the axioms which have not been rewritten into a SAT problem. In practice the $\mathcal{T}$-solver is the implementation of a sub-component of a traditional tableau-based DL reasoner. Substantially, this approach is an extension of the DPLL-based approach for modal logic proposed by Giunchiglia and Sebastiani (1996, 2000) and implemented in KSAT and *SAT (see Section 5.1); The main difference from this latter approach and from the modern tableau-based approaches lays om the expansion at the $\mathcal{ALC}$ level of the Boolean component of reasoning and for the tighter interactions between the $\mathcal{T}$-solver and the assignment enumerator. We are not aware of any further investigation or advance in the work of Gasse and Haarslev (2009) that is only at a preliminary level.

6.2 Motivations and Goals

On the one hand, reasoning with qualified number restriction is a very prominent research issue in Description Logic, because the current techniques often lacks of efficiency especially when the number of restrictions is high or when big numbers occur in restrictions. Because in the presence of inefficient reasoning techniques with qualified number restrictions, ontology designers will most likely avoid the use of these constructs, even if they are very natural in many domains. Thus, the more new domains are represented through DL-based ontologies the more devising efficient procedures for handling qualified number restrictions can have particularly important consequences for the development of the ontology-design area, besides that for the field of automated reasoning in Description Logics. In particular, the explosion of Semantic Web applications and the inclusion of number restrictions in the recent OWL 2 standard increased the quest of languages allowing for expressing cardinality restrictions, yielding the development of techniques for

optimized reasoning with qualified number restrictions an essential research goal.

On the other hand, the progress in Boolean Satisfiability (SAT) solving techniques, together with the concrete needs from real applications, have inspired significant research on richer and more expressive Boolean formalism like Satisfiability Modulo Theories (SMT). SMT can be seen as an extension of SAT in which the input formula is expressed in (a subset of) first-order logic (typically without quantifiers) wrt. one or more background theories (for example: linear arithmetic –over the reals or the integers–, its subclass difference logic, the theories of bit vectors, of arrays and of lists, and others). The dominating approach for SMT, which underlies most state-of-the-art tools, is based on the integration of a SAT solver and one or more domain-specific solvers for the background theories. The SAT solver enumerates truth assignments which satisfy the Boolean abstraction of the input formula (where distinct theory-specific subformulas are represented/abstracted by distinct Boolean atoms), whilst each domain-specific solver checks the consistency in the respective background theory of the subset of assigned literals corresponding to its own theory. This approach is called lazy (see Section 4.2 for the background or, e.g., Sebastiani, 2007b for a survey). Although SMT is still a novel research area, it is also a very active one: new solvers and techniques are continuously proposed, and often with improvements of orders of magnitude in performance wrt. the previous approaches.

In Chapter 5 we have shown that it is possible to efficiently perform reasoning in Description Logic via encoding into SAT problems. In particular, we have learned from the results exposed in Section 5.6 that state-of-the-art SAT solvers are tools able to efficiently solve problems of hundreds of millions of variables and clauses. Following the experience acquired in Chapter 5 we think that also the rising SMT technologies can be, in practice, very powerful and suitable tools to reason on Description Logic problems, combining the adaptability and scalability of SAT with the expressiveness of the many embedded theories. The promising performance of some algebraic/hybrid systems presented in Section 6.1 and the surprising results of our previous approach to $\mathcal{ALC}$ (see Section 5.7 for a summary) motivated us in following this intuition.

The idea is to exploit SMT in the development of new techniques for optimized reasoning with numerical constraints in Description Logics, especially for what concerns those language constructors that are somewhat similar to those provided by the theories included in SMT. So we open this research stream, starting from the encoding of concept satisfiability in the logic $\mathcal{ALCQ}$, (which extends $\mathcal{ALC}$ with qualified number restrictions) into SMT. The objective is to investigate the potentials and to perform a first step in the direction of defining a framework for efficiently handling very expressive logics by mean of the integration of the very advance SAT/SMT-based techniques at the state of the art. Notice that our approach differs from the one of Gasse and Haarslev (2009) because we are not proposing an SMT-like architecture for an $\mathcal{ALCQ}$ reasoner but to directly encode $\mathcal{ALCQ}$ problems into SMT so that to be handled by the available SMT-solvers. This difference is very important for two main aspects, which corresponds to two of the founding motivations of our research line. First, in our approach we try to move as much reasoning as possible in a first, expensive but "done-once-for-all" encoding phase (performed by

a specific tool), in order to ease as much as possible the single but numerous following queries performed directly on SMT, 9and for which a fast response is more important). Second, we can exploit any advance in the SMT technologies for free, without the need of reimplementing our approach.

In this part of our work we face, specifically, the problem of concept satisfiability in $\mathcal{ALCQ}$ (see Section 3.4.1) wrt. acyclic TBoxes. Concept satisfiability wrt. acyclic TBoxes is still a PSPACE-complete reasoning task in $\mathcal{ALCQ}$ where, instead, concept satisfiability wrt. general TBoxes is EXPTIME-complete. We chose to handle only acyclic TBoxes not to avoid to switch to the upper class of complexity but to concentrate only on the encoding, postponing the issue of introducing techniques like blocking to handle cyclicity. Even if, from a purely theoretical perspective, the worst-case complexity of the tackled problem is the same of the previously approached K_m-satisfiability, it is worth noticing that in this second part of our research we include two further main sources of complexity: TBoxes and numerical constraints. Notice that handling TBoxes significantly change the form of reasoning and can also inhibit the applicability of some of the optimization we introduced for the $K_m/\mathcal{ALC}$ case (see Section 5.5) first of all On-the-fly Boolean Constraint Propagation.

Content. The rest of this chapter is structured as follows. In Section 6.3 we discuss two alternative implementations of our approach and give the motivations for the chosen one. In Section 6.4 we present a normal form for $\mathcal{ALCQ}$ TBoxes used in the following of this chapter, and we describe the transformations rules necessary to obtain normalize a noraml-form TBox from an input TBox which is not in normal form. In Section 6.5 we define the our novel encoding from $\mathcal{ALCQ}$ into SMT modulo the Theory of Costs, we expose the theoretical results (whose proofs are reported as appendix in Section 6.9) and we propose an encoding algorithm. In Section 6.6 we discuss the need in our approach of optimization techniques aiming at reducing the number of encoded individuals, then we formally describe and develop a partitioning algorithm of the individuals to cope with this issue. In Section 6.7 we present our tool $\mathcal{ALCQ}$2SMT we describe the empirical evaluation set-up, and the chosen benchmark problems, discussing the results.

In particular, in this latter section we analyze the effectiveness of partitioning in our approach and we compare the performance of the integration of $\mathcal{ALCQ}$2SMT with an SMT-solver against the performances of the other state-of-the-art reasoners. (Some less significant results from the empirical evaluation are reported in the appendix Section 6.11.) In Section 6.8 we describe the contributions of this part of our research.

The encoding, the partitioning technique ans some preliminary results from this research stream have been recently submitted for publication in one international conference (Haarslev et al., 2011).

6.3 Alternative Solutions

Given the many kinds of numerical reasoning available in SMT (including linear arithmetic, inequations, counting and others), we identify two main possible alternative en-

coding solutions inspired from previous approaches exposed in Section 6.1:

1. One possible approach is to follow the hybrid approach of Faddoul et al. (2008), Faddoul and Haarslev (2010), and to use the arithmetic theories of SMT in order to perform numerical reasoning on the cardinality of groups of individuals (through atomic decompositions (Ohlbach & Koehler, 1999)). An encoding of this approach into SMT ($\mathcal{LA}(\mathbb{Z})$) (see Section 4.2.2) is quite intuitive. We don't go into details, but the idea is to exploit the Boolean component of SMT in order to check the logical satisfiability of proxy individuals representing each possible group/partition of role fillers, and to rely on the $\mathcal{LA}(\mathbb{Z})$-solver to check the numerical consistency of the partitions cardinality wrt. the existing qualified number restrictions. This approach has the main benefit of being not affected by the values occurring in number restrictions.

2. A second possible approach is to mimic the traditional tableau-based algorithms. Tableau-based algorithms satisfy all the possible at-least restrictions by introducing many individuals as role fillers and then allow the merging of individuals when their number exceeds some at-most restrictions. In this second implementation, the idea is to exploit the Boolean component of SMT in order to represent the satisfiability of single individuals and their membership to concept interpretations, and then to use some theory in SMT in order to count and bound the number of these individuals. In place of merging, we allow for sharing all the introduced individuals among different concept interpretations, so that the numerical theory forces only a consistent number of individuals to exist in order to satisfying the lower/upper-bounds.

   It is possible to encode this approach into SMT modulo the Theory of Costs (SMT($\mathcal{C}$)), that we think more naturally fits the expressiveness required by qualified number restrictions (see Section 4.2.3). The Theory of Costs, in fact, is a subset of $\mathcal{LA}(\mathbb{Z})$, in which it is possible to define multiple cost variables/functions and to define both costs' increments and costs' lower/upper-bounds. Being $\mathcal{C}$ a subset of $\mathcal{LA}(\mathbb{Z})$, further than being an easier and more compact formalism, it has a lighter, more specific and efficient theory solver, because it is based only on sums and checks (Cimatti et al., 2010).

Between these two alternative solutions we have privileged the encoding into SMT($\mathcal{C}$), due to the following reasons:

– Theory of Costs $\mathcal{C}$ is a really simple theory who needs a lightweight and simple solver while Linear Arithmetic $\mathcal{LA}$ is much more complicated and thus, likely, needs a much more time-consuming theory solver.

– Due to atomic decomposition, the hybrid approach leads a-priori to an exponential number of partitions and proxy-individuals wrt. the number of qualified number restrictions that must be encoded (no matter what is the nature of the concept expressions). Encoding such an approach in SMT ($\mathcal{LA}(\mathbb{Z})$) would affect exponentially

both the number of the Boolean variables (representing proxy individuals) and, more importantly, the number of the integer variables (representing cardinality). On the contrary, in the SMT($\mathcal{C}$) encoding the number of integer variables necessary for every group of restrictions is linear in the number of the restrictions, while the number of individuals is linear wrt. the values occurring in number restrictions.

– Whether the linear dependence of the second approach from the values included in the number restrictions is more negatively impacting than the exponentiality dependence of the first approach from the number of restrictions or not, depends from the nature of the specific encoded $\mathcal{ALCQ}$ problem. However, we think that:

  - real-world ontologies reasonably should have a high number of qualified number restrictions in which just a few can have big values occurring in them (otherwise those values can, somehow, be rationalized);

  - given the power of the underlying SAT-solvers in SMT, a huge number of Boolean variables can be more affordable than a huge number of integer variables that must be handled by the theory solver;

  - the idea of introducing as many role fillers (individuals) as the values occurring in the at-least restrictions is only a first naive way of implement the second alternative encoding; we think that this approach can be improved by heuristics or enhanced encodings in which individuals are handled in groups (depending from the specific values included in the restrictions), borrowing from the hybrid approach the idea of represent a group of individuals with only one proxy individual.

6.4 A Normal Form for $\mathcal{ALCQ}$

We briefly recall some notation from Section 3.2. Given a TBox $\mathcal{T}$, we denote with $\mathsf{BC}_\mathcal{T}$ the set of the *basic concepts* for $\mathcal{T}$, i.e. the smallest set of concepts containing: (i) the top and the bottom concepts $\top$ and $\bot$; (ii) all the concepts of $\mathcal{T}$ in the form C and $\neg C$ where C is a concept name in ${N_C}^\mathcal{T}$. In this chapter, we use simple letters like $C, D, \ldots, N, M, \ldots$ to denote the basic concepts in $\mathsf{BC}_\mathcal{T}$ (thus, C can be used to represent a concept $\neg C'$ with $C' \in \mathsf{BC}_\mathcal{T}$), whilst we use the notation $\hat{C}, \hat{D}, \ldots$ for complex concepts $\hat{C}, \hat{D} \notin \mathsf{BC}_\mathcal{T}$.

Wlog. we assume all the $\mathcal{ALCQ}$ concept definitions to be in *negative normal form* (NNF), thus negation only applies to concept names. Starting from a generic concept definition C it is possible to obtain an equivalent concept definition in NNF, applying the

following linear transformations, where $\dot{\neg}C$ represents the NNF transformation of $\neg C$:

$$\neg(C \sqcap D) \Longrightarrow \dot{\neg}C \sqcup \dot{\neg}D \qquad \neg(C \sqcup D) \Longrightarrow \dot{\neg}C \sqcap \dot{\neg}D$$
$$\neg\exists r.C \Longrightarrow \forall r.\dot{\neg}C \qquad \neg\forall r.C \Longrightarrow \exists r.\dot{\neg}C$$
$$\neg{\geq}nr.C \Longrightarrow {\leq}n-1r.C \qquad \neg{\leq}mr.C \Longrightarrow {\geq}m+1r.C$$
$$\neg\neg C \Longrightarrow C \qquad \neg\bot \Longrightarrow \top \qquad \neg\top \Longrightarrow \bot$$

Then we restrict our attention to those $\mathcal{ALCQ}$ TBoxes in which all axioms are in the following normal form:

$$C \sqsubseteq D$$
$$C_1 \sqcap C_2 \sqsubseteq D \qquad C \sqsubseteq D_1 \sqcap D_2$$
$$C_1 \sqcup C_2 \sqsubseteq D \qquad C \sqsubseteq D_1 \sqcup D_2$$
$$\Re r.C \sqsubseteq D \qquad C \sqsubseteq \Re r.D$$

with $C, C_1, C_2, D, D_1, D_2 \in \mathsf{BC}_{\mathcal{T}}$, $r \in {N_R}^{\mathcal{T}}$ and where $\Re \in \{\exists, \forall, \geq n, \leq m\}$, i.e. it can be any of the possible restriction operators allowed in $\mathcal{ALCQ}$.

Any given TBox $\mathcal{T}$ can be turned into a normalized one $\mathcal{T}'$ that is a *conservative extension* of $\mathcal{T}$ by introducing new concept names. A TBox $\mathcal{T}'$ is a *conservative extension* of the TBox $\mathcal{T}$ if every model of $\mathcal{T}'$ is also a model of $\mathcal{T}$, and every model of $\mathcal{T}$ can be extended to a model of $\mathcal{T}'$ by appropriately defining the interpretations of the additional concept and role names. The transformation of a TBox $\mathcal{T}$ into a normalized one $\mathcal{T}'$ can be done in linear time (and, thus, $\mathcal{T}'$ has no more than linear size w.r.t. the size of $\mathcal{T}$) applying exhaustively the following transformation rules:

$$\hat{C} \sqsubseteq \hat{D} \Longrightarrow \{\hat{C} \sqsubseteq N,\ N \sqsubseteq M,\ M \sqsubseteq \hat{D}\}$$

$$C \sqcap \hat{C} \sqsubseteq D \Longrightarrow \{C \sqcap N \sqsubseteq D,\ N \sqsubseteq \hat{C}\} \qquad C \sqsubseteq D \sqcap \hat{D} \Longrightarrow \{C \sqsubseteq D \sqcap N,\ \hat{D} \sqsubseteq N\}$$
$$C \sqcup \hat{C} \sqsubseteq D \Longrightarrow \{C \sqcup N \sqsubseteq D,\ N \sqsubseteq \hat{C}\} \qquad C \sqsubseteq D \sqcup \hat{D} \Longrightarrow \{C \sqsubseteq D \sqcup N,\ \hat{D} \sqsubseteq N\}$$
$$\Re r.\hat{C} \sqsubseteq D \Longrightarrow \{\Re r.N \sqsubseteq D,\ N \sqsubseteq \hat{C}\} \qquad C \sqsubseteq \Re r.\hat{D} \Longrightarrow \{C \sqsubseteq \Re r.N,\ \hat{D} \sqsubseteq N\}$$

where $\Re \in \{\exists, \forall, \geq n, \leq m\}$, $C, D \in \mathsf{BC}_{\mathcal{T}}$, $\hat{C}, \hat{D} \notin \mathsf{BC}_{\mathcal{T}}$, and $N, M \notin \mathsf{BC}_{\mathcal{T}}$ are fresh concept names newly introduced in order to define complex concept descriptions. Notice that, during normalization, when a complex concept description appears both at the left- and at the right-hand side of some concept inclusions it can be better defined by mean of the same new concept name instead of by introducing two different fresh names for it.

Even if, for convenience and wlog., we sometimes restrict to binary conjunction/disjunction relations, in practice we can relax such a constraint and allow for having n-ary conjunctions and disjunctions of basic concepts that we represent respectively with $\sqcap_i C_i$ and $\sqcup_i C_i$. Moreover, in order to safely reduce the number of possible cases and to increase the number of equivalent concepts having the same description, we further

refine the considered normal form applying the following equivalence transformations to the axioms and concepts of $\mathcal{T}'$:

$$C \sqsubseteq D_1 \sqcap \ldots \sqcap D_n \Longrightarrow \{C \sqsubseteq D_1, \ldots, C \sqsubseteq D_n\} \qquad \exists r.C \Longrightarrow \geq 1 r.C$$
$$C_1 \sqcup \ldots \sqcup C_n \sqsubseteq D \Longrightarrow \{C_1 \sqsubseteq D, \ldots, C_n \sqsubseteq D\} \qquad \leq 0 r.C \Longrightarrow \forall r.\dot{\neg} C$$

thus, we avoid left-hand side disjunctions and right-hand side conjunctions and we handle existential and zero at-most restrictions as special cases of, respectively, qualified number and universal restrictions. This has been said the resulting considered normal form is the following:

$$C \sqsubseteq D \qquad \sqcap_i C_i \sqsubseteq D \qquad C \sqsubseteq \sqcap_i D_i \tag{6.1}$$
$$\Re r.C \sqsubseteq D \qquad C \sqsubseteq \Re r.D \qquad \Re \in \{\forall, \geq n, \leq m\}, \text{with } n, m \geq 1 \tag{6.2}$$

where C, C_i, D, D_i are basic concepts. Finally, notice that the first of the normal forms (6.1) is a special case of the successive two normal forms with $i = 1$.

We call *normal concept* of a normal TBox $\mathcal{T}'$ every non-conjunctive and non-disjunctive concept description occurring in the concept inclusions $\mathcal{T}'$; we call $\mathsf{NC}_{\mathcal{T}'}$ the set of all the normal concepts of $\mathcal{T}'$. Practically, an element of the set $\mathsf{NC}_{\mathcal{T}'}$ is either a concept C with $C \in \mathsf{BC}_{\mathcal{T}'}$ or a concept in the form $\Re r.C$, with $\Re \in \{\exists, \forall, \geq n, \leq m\}$, $C \in \mathsf{BC}_{\mathcal{T}'}$ and $r \in {N_R}^{\mathcal{T}}$. [1] Given a non-normal concept $\hat{C}$ (that is a conjunction or a disjunction of normal concepts) we identify with $\mathsf{nc}(\hat{C})$ the set of normal concepts of which $\hat{C}$ is composed, where $\mathsf{nc}(\hat{C}) = \{\hat{C}\}$ if $\hat{C}$ is normal. [2]

6.5 Concept Satisfiability in $\mathcal{ALCQ}$ via SMT($\mathcal{C}$) solving

We encode the problem of concept satisfiability in $\mathcal{ALCQ}$ into SMT modulo the Theory of Costs ($\mathcal{C}$). Given an acyclic $\mathcal{ALCQ}$ TBox $\mathcal{T}$ we denote with $\mathcal{ALCQ}2SMT_{\mathcal{C}}(\mathcal{T})$ the SMT($\mathcal{C}$) encoding for $\mathcal{T}$. We also assume that every axiom description in $\mathcal{T}$ is in the normal form exposed in Section 6.4; thus, in particular, every concept expression in $\mathcal{T}$ is assumed to be in NNF. In a nutshell, the encoding simulates the construction of an hypothetical interpretation for $\mathcal{T}$, so that if a satisfying truth assignment μ for $\mathcal{ALCQ}2SMT_{\mathcal{C}}(\mathcal{T})$ exists, tit is define from μ a model $\mathcal{I}$ for $\mathcal{T}$, respecting all the numerical constraints given number restrictions. The encoding essentially works as follows:

- we introduce possible individuals for the interpretation domain $\Delta^{\mathcal{I}}$;
- we represent with Boolean variables whether an introduced individual concretely exists in $\Delta^{\mathcal{I}}$ and whether it belongs to the interpretation of one specific concept;

[1] Note that if $\mathcal{T}'$ is a normal TBox, conservative extension of the non-normal TBox $\mathcal{T}$, $\mathsf{BC}_{\mathcal{T}} \subseteq \mathsf{BC}_{\mathcal{T}'}$.

[2] We anticipate that, at the effect of the encoding we propose in this work, the normalization of the given TBox is not strictly necessary since it is possible to recursively label non-normal concepts and their sub-concepts with fresh variables and then encode with new clauses the relations between the main concept and their subconcepts, like done in (Sebastiani & Vescovi, 2009a). However, we introduced the exposed normal form in order to reduce the possible cases that must be considered, simplifying the exposition, the encoding and the formal proofs.

– we use $\mathcal{C}$-atoms in order to count the number of individuals and to express the bounds imposed in $\mathcal{T}$ by mean of the qualified number restrictions.

In the following we formally define the encoding, we expose the theoretical results and give an encoding algorithm.

6.5.1 Encoding $\mathcal{ALCQ}$ into SMT($\mathcal{C}$)

As previously done in Section 5.4, we uniquely represent *individuals* in $\Delta^{\mathcal{I}}$ by means of *labels* σ, represented as non-empty sequences of positive integer values and role names in ${N_R}^{\mathcal{T}}$. A label σ can be either the label 1 or a label in the form $\sigma'.r.n$, where σ' is another label, $r \in {N_R}^{\mathcal{T}}$ and $n \geq 1$. With a small abuse of notation, hereafter we may say "the individual σ" meaning "the individual labeled by σ". Moreover, we call *instantiated concept* a pair $\langle \sigma, C \rangle$, such that σ is an individual and C is an $\mathcal{ALCQ}$ normal concept, representing the fact that σ is an instance of C in the $\mathcal{I}$, i.e., briefly, $\sigma \in C^{\mathcal{I}}$.

Definition 3 ($A_{\langle\ ,\ \rangle}$, concept variable/literal)**.** We define $A_{\langle\ ,\ \rangle}$ an injective function which maps one instantiated concept $\langle \sigma, C \rangle$ such that C is not in the form $\neg C'$, for any C', into a Boolean variable $A_{\langle \sigma,\ C \rangle}$ that we call *concept variable*. Let the literal $L_{\langle \sigma,\ C \rangle}$, that we call *concept literal*, denote $\neg A_{\langle \sigma,\ C' \rangle}$ if C is in the form $\neg C'$, $A_{\langle \sigma,\ C \rangle}$ otherwise.

The truth value of the concept literal $L_{\langle \sigma,\ C \rangle}$ states whether the instantiation relation between σ and C [resp. $\neg C$] holds, i.e. if $\langle \sigma, C \rangle$ [resp. $\langle \sigma, \neg C \rangle$] is an existing instantiated concept. We conventionally assume that $A_{\langle \sigma,\ \bot \rangle}$ is $\bot$. Notice also that $\langle \sigma, \top \rangle$ means $\sigma \in \Delta^{\mathcal{I}}$, i.e. that if $A_{\langle \sigma,\ \top \rangle}$ is assigned to true then the individual σ exists in the domain of the interpretation. We informally say that an individual σ (meaning $\langle \sigma, \top \rangle$) or an instantiated concept $\langle \sigma, C \rangle$ is "enabled" meaning that the respective literal is assigned to true.

Definition 4 (indiv, individuals cost variable)**.** We define indiv a function which maps one instantiated concept $\langle \sigma, \Re r.C \rangle$, such that $\Re \in \{\geq n, \leq m\}$ and C is a basic concept [3], into a cost variable $\mathsf{indiv}^C_{\sigma.r}$ in the Theory of Costs, that we call *individuals cost variable*.

Notice that the function indiv is not injective, since the same cost variable $\mathsf{indiv}^C_{\sigma.r}$ is "shared" among all the instantiated concepts which refer both to the same individual σ and to some qualified number restriction involving the same role r and the same basic concept C. However, notice also that $\langle \sigma, \Re r.C \rangle$ and $\langle \sigma, \Re r.\neg C \rangle$ are mapped to different cost variables. Given the individuals cost variable $\mathsf{indiv}^C_{\sigma.r}$, the final value of the variable represents the number of individuals which are in relation with the individual σ via the role r and which are in the interpretation of C, in other words the final value of $\mathsf{indiv}^C_{\sigma.r}$ exactly represents the cardinality of $FIL(r, \sigma) \cap C^{\mathcal{I}}$.

Definitions 3 and 4 are at the base of the $\mathcal{ALCQ}2SMT_{\mathcal{C}}(\mathcal{T})$ encoding. They allow for mapping couples made up of individuals and concepts to Boolean and cost variables, respectively. The encoding works essentially by mean of the following principles:

[3] We remark that we are considering only concepts in normal form.

- The individual 1 is the root individual.
- The axioms of $\mathcal{T}$ representing the inclusions between two concepts are encoded through Boolean implications between the respective concept variables.
- Given an individual σ every at-least qualified number restriction $\langle \sigma, \geq nr.C \rangle$ is handled by introducing exactly n new r-successors $\sigma.r.i$ for σ, that are supposed to be in $C^{\mathcal{I}}$. The existence of individuals is forced by binding each of them to an incur cost of value 1 for the cost variable $\mathsf{indiv}^{C}_{\sigma.r}$, and then fixing a lower-bound for $\mathsf{indiv}^{C}_{\sigma.r}$.
- When, both at-least and at-most restrictions coexist wrt. a given σ, the number of the many individuals introduced in order to trivially satisfy all the at-least restrictions must be bounded. Thus, first every at-most restriction is handled by fixing an upper-bound for the respective cost variables and, second, the encoding allows for sharing individuals separately introduced by distinct at-least restrictions, so that one single individual can belong to many concept interpretations and concur in satisfying many at-least restrictions.

Now we give a formal description of our novel encoding.

Definition 5 ($\mathcal{ALCQ}2SMT_{\mathcal{C}}$ encoding)**.** Let $\mathcal{T}$ being an acyclic $\mathcal{ALCQ}$ TBox in normal form [4] and, wlog., assume that every axiom of $\mathcal{T}$ is in the form $\hat{C} \sqsubseteq \hat{D}$, with $\hat{C} = \sqcap_i C_i$, $\hat{D} = \sqcup_j D_j$, where $i, j \geq 1$ and $i = 1$ [resp. $j = 1$] in the case in which $\hat{C}$ [resp. $\hat{D}$] is a basic concept.

The SMT($\mathcal{C}$) encoding $\mathcal{ALCQ}2SMT_{\mathcal{C}}(\mathcal{T})$ for $\mathcal{T}$ is defined as the sextuple $\langle \Sigma^{\mathcal{T}}, \mathcal{I}^{\mathcal{T}}_{-}, \mathcal{I}^{\mathcal{T}}_{+}, A_{\langle\ ,\ \rangle}, \mathsf{indiv}, \varphi^{\mathcal{T}} \rangle$, where:

- $\Sigma^{\mathcal{T}}$ is the set of all the possible individuals introduced;
- $\mathcal{I}^{\mathcal{T}}_{-}, \mathcal{I}^{\mathcal{T}}_{+}$ represent, respectively, the set of the implicant and implied instantiated concepts introduced in the encoding, which are built of individuals in $\Sigma^{\mathcal{T}}$ and basic concept of $\mathcal{T}$;
- $A_{\langle\ ,\ \rangle}$ and indiv are the functions defined in Definition 3 and Definition 4, respectively;
- $\varphi^{\mathcal{T}}$ is a CNF Boolean combination of propositional- and $\mathcal{C}$-literals encoding $\mathcal{T}$ into SMT($\mathcal{C}$) [5]; we represent $\varphi^{\mathcal{T}}$ as such a set of clauses.

The sets $\Sigma^{\mathcal{T}}, \mathcal{I}^{\mathcal{T}}_{-}, \mathcal{I}^{\mathcal{T}}_{+}$ and $\varphi^{\mathcal{T}}$ are incrementally defined as the minimum sets such that:

1. $1 \in \Sigma^{\mathcal{T}}$, $\langle 1, \top \rangle \in \mathcal{I}^{\mathcal{T}}_{-}$, $\langle 1, \top \rangle \in \mathcal{I}^{\mathcal{T}}_{+}$ and $(A_{\langle 1,\ \top \rangle}) \in \varphi^{\mathcal{T}}$.
2. $\{\langle 1, C_i \rangle \mid C_i \in \mathsf{nc}(\hat{C})\} \subseteq \mathcal{I}^{\mathcal{T}}_{-}$, for every axiom $\hat{C} \sqsubseteq \hat{D} \in \mathcal{T}$.

[4] See Section 6.4.

[5] All the clauses of $\mathcal{ALCQ}2SMT_{\mathcal{C}}(\mathcal{T})$ are intended to be in CNF even if we reported them in form of implications.

3. If $\sigma \in \Sigma^{\mathcal{T}}$, for every axiom $\sqcap_i C_i \sqsubseteq \sqcup_j D_j \in \mathcal{T}$ such that $\{\langle \sigma, C_i\rangle \mid C_i \in \mathsf{nc}(\hat{C})\} \subseteq \mathcal{I}^{\mathcal{T}}_- \cup \mathcal{I}^{\mathcal{T}}_+$, then

$$\{\ \langle \sigma, D_j\rangle \mid D_j \in \mathsf{nc}(\hat{D})\ \} \ \subseteq\ \mathcal{I}^{\mathcal{T}}_+$$

and

$$(\bigwedge_i L_{\langle \sigma,\ C_i\rangle}) \rightarrow (\bigvee_j L_{\langle \sigma,\ D_j\rangle}) \ \in\ \varphi^{\mathcal{T}}. \tag{6.3}$$

4. If $\sigma \in \Sigma^{\mathcal{T}}$ and $\langle \sigma, \Re'.r.C'\rangle \in \mathcal{I}^{\mathcal{T}}_+$, with $\Re' \in \{\geq n', \leq m', \forall\}$, then

$$\{\ \langle \sigma, \Re r.C\rangle \mid \Re r.C \sqsubseteq \hat{D} \in \mathcal{T}\ \} \ \subseteq\ \mathcal{I}^{\mathcal{T}}_-,$$

for every $\Re \in \{\geq n, \leq m, \forall\}$.

5. If $\sigma \in \Sigma^{\mathcal{T}}$ and $\langle \sigma, \geq nr.C\rangle \in \mathcal{I}^{\mathcal{T}}_+$ [resp. $\langle \sigma, \leq n-1r.C\rangle \in \mathcal{I}^{\mathcal{T}}_-$] [resp. $\langle \sigma, \forall r.\neg C\rangle \in \mathcal{I}^{\mathcal{T}}_-$, $n \stackrel{\text{def}}{=} 1$], then

$$\{\ \sigma.r.k^C_i \mid i = 1, \ldots, n\ \} \ \subset\ \Sigma^{\mathcal{T}},$$

$$\{\ \langle \sigma.r.k^C_i, C\rangle \mid i = 1, \ldots, n\ \} \ \cup\ \{\ \langle \sigma.r.k^C_i, \top\rangle \mid i = 1, \ldots, n\ \} \ \subset\ \mathcal{I}^{\mathcal{T}}_-$$

and

$$\{\ \mathsf{IC}(\mathsf{indiv}^C_{\sigma.r}, 1, k^C_i) \rightarrow L_{\langle \sigma.r.k^C_i,\ C\rangle} \mid i = 1, \ldots, n\ \} \ \subset\ \varphi^{\mathcal{T}}, \tag{6.4}$$

$$\{\ \mathsf{IC}(\mathsf{indiv}^C_{\sigma.r}, 1, k^C_i) \rightarrow A_{\langle \sigma.r.k^C_i,\ \top\rangle} \mid i = 1, \ldots, n\ \} \ \subset\ \varphi^{\mathcal{T}}, \tag{6.5}$$

where $k^C_1 \geq 1$, $k^C_{i+1} = k^C_i + 1$ and $k^C_i \neq k^D_j$ for every $\langle \sigma, \geq n'r.D\rangle \in \mathcal{I}^{\mathcal{T}}_+$ [resp. $\langle \sigma, \leq n'-1r.D\rangle \in \mathcal{I}^{\mathcal{T}}_-$] [resp. $\langle \sigma, \forall r.\neg C\rangle \in \mathcal{I}^{\mathcal{T}}_-$, $n' \stackrel{\text{def}}{=} 1$], with $C \neq D$ and $i = 1, \ldots, n$, $j = 1, \ldots, n'$. We assume only consecutive values for the individuals $\sigma.r.j$. [6]

6. If $\sigma \in \Sigma^{\mathcal{T}}$ and $\langle \sigma, \geq nr.C\rangle \in \mathcal{I}^{\mathcal{T}}_+$, then

$$((A_{\langle \sigma,\ \geq nr.C\rangle} \wedge A_{\langle \sigma,\ \top\rangle}) \rightarrow \neg\mathsf{BC}(\mathsf{indiv}^C_{\sigma.r}, n-1)) \ \in\ \varphi^{\mathcal{T}}, \tag{6.6}$$

while, if $\sigma \in \Sigma^{\mathcal{T}}$ and $\langle \sigma, \geq nr.C\rangle \in \mathcal{I}^{\mathcal{T}}_-$, then

$$((\neg\mathsf{BC}(\mathsf{indiv}^C_{\sigma.r}, n-1) \wedge A_{\langle \sigma,\ \top\rangle}) \rightarrow A_{\langle \sigma,\ \geq nr.C\rangle}) \ \in\ \varphi^{\mathcal{T}}. \tag{6.7}$$

7. If $\sigma \in \Sigma^{\mathcal{T}}$, $\langle \sigma, \leq mr.E\rangle \in \mathcal{I}^{\mathcal{T}}_+$ [resp. $\langle \sigma, \geq mr.E\rangle \in \mathcal{I}^{\mathcal{T}}_-$], $\langle \sigma, \geq nr.C\rangle \in \mathcal{I}^{\mathcal{T}}_+$, [resp. $\langle \sigma, \leq n-1r.C\rangle \in \mathcal{I}^{\mathcal{T}}_-$] and $\langle \sigma, \geq n'r.D\rangle \in \mathcal{I}^{\mathcal{T}}_+$ [resp. $\langle \sigma, \leq n'-1r.D\rangle \in \mathcal{I}^{\mathcal{T}}_-$ or $\langle \sigma, \forall r.\neg C\rangle \in \mathcal{I}^{\mathcal{T}}_-$ assuming $n' = 1$], then

$$\{\ \langle \sigma.r.k^C_i, D\rangle \mid i = 1, \ldots, n\ \} \ \cup\ \{\ \langle \sigma.r.k^D_i, C\rangle \mid i = 1, \ldots, n'\ \} \ \subset\ \mathcal{I}^{\mathcal{T}}_-$$

[6] Thus either $k^C_1 = 1$ or $k^C_1 = k^D_{n'} + 1$, for some $\langle \sigma, \geq n'r.D\rangle \in \mathcal{I}^{\mathcal{T}}_+$ [resp. $\langle \sigma, \leq n' - 1r.D\rangle \in \mathcal{I}^{\mathcal{T}}_-$] [resp. $\langle \sigma, \forall r.\neg C\rangle \in \mathcal{I}^{\mathcal{T}}_-$, $n' \stackrel{\text{def}}{=} 1$].

and

$$\{\ \mathsf{IC}(\mathsf{indiv}^D_{\sigma.r}, 1, k^C_i) \rightarrow L_{\langle \sigma.r.k^C_i,\ D\rangle} \mid i = 1, \ldots, n\ \}\ \cup$$
$$\{\ \mathsf{IC}(\mathsf{indiv}^C_{\sigma.r}, 1, k^D_i) \rightarrow L_{\langle \sigma.r.k^D_i,\ C\rangle} \mid i = 1, \ldots, n'\ \}\ \subset\ \varphi^{\mathcal{T}}, \tag{6.8}$$
$$\{\ \mathsf{IC}(\mathsf{indiv}^D_{\sigma.r}, 1, k^C_i) \rightarrow A_{\langle \sigma.r.k^C_i,\ \top\rangle} \mid i = 1, \ldots, n\ \}\ \cup$$
$$\{\ \mathsf{IC}(\mathsf{indiv}^C_{\sigma.r}, 1, k^D_i) \rightarrow A_{\langle \sigma.r.k^D_i,\ \top\rangle} \mid i = 1, \ldots, n'\ \}\ \subset\ \varphi^{\mathcal{T}}. \tag{6.9}$$

8. If $\sigma \in \Sigma^{\mathcal{T}}$ and $\langle \sigma, \leq m r.C\rangle \in \mathcal{I}^{\mathcal{T}}_+$ [resp. $\langle \sigma, \geq n r.C\rangle \in \mathcal{I}^{\mathcal{T}}_-$], then

$$\{\ \langle \sigma.r.j, C\rangle \mid \sigma.r.j \in \Sigma^{\mathcal{T}}\ \}\ \subset\ \mathcal{I}^{\mathcal{T}}_-$$

and

$$\{\ (L_{\langle \sigma.r.j,\ C\rangle} \wedge A_{\langle \sigma.r.j,\ \top\rangle}) \rightarrow \mathsf{IC}(\mathsf{indiv}^C_{\sigma.r}, 1, j) \mid \sigma.r.j \in \Sigma^{\mathcal{T}}\ \}\ \subset\ \varphi^{\mathcal{T}}. \tag{6.10}$$

9. If $\sigma \in \Sigma^{\mathcal{T}}$ and $\langle \sigma, \leq m r.C\rangle \in \mathcal{I}^{\mathcal{T}}_+$, then

$$((A_{\langle \sigma,\ \leq m r.C\rangle} \wedge A_{\langle \sigma,\ \top\rangle}) \rightarrow \mathsf{BC}(\mathsf{indiv}^C_{\sigma.r}, m))\ \in\ \varphi^{\mathcal{T}}, \tag{6.11}$$

while, if $\sigma \in \Sigma^{\mathcal{T}}$ and $\langle \sigma, \leq m r.C\rangle \in \mathcal{I}^{\mathcal{T}}_-$, then

$$((\mathsf{BC}(\mathsf{indiv}^C_{\sigma.r}, m) \wedge A_{\langle \sigma,\ \top\rangle}) \rightarrow A_{\langle \sigma,\ \leq m r.C\rangle})\ \in\ \varphi^{\mathcal{T}}. \tag{6.12}$$

10. if $\sigma \in \Sigma^{\mathcal{T}}$ and $\langle \sigma, \forall r.C\rangle \in \mathcal{I}^{\mathcal{T}}_+$, then

$$\{\ \langle \sigma.r.j, C\rangle \mid \sigma.r.j \in \Sigma^{\mathcal{T}}\ \}\ \subset\ \mathcal{I}^{\mathcal{T}}_-$$

and

$$\{\ ((A_{\langle \sigma,\ \forall r.C\rangle} \wedge A_{\langle \sigma.r.j,\ \top\rangle}) \rightarrow L_{\langle \sigma.r.j,\ C\rangle}) \mid \sigma.r.j \in \Sigma^{\mathcal{T}}\ \}\ \subset\ \varphi^{\mathcal{T}}, \tag{6.13}$$

while, if $\sigma \in \Sigma^{\mathcal{T}}$ and $\langle \sigma, \forall r.C\rangle \in \mathcal{I}^{\mathcal{T}}_-$, then

$$((\mathsf{BC}(\mathsf{indiv}^{\neg C}_{\sigma.r}, 0) \wedge A_{\langle \sigma,\ \top\rangle}) \rightarrow A_{\langle \sigma,\ \forall r.C\rangle})\ \in\ \varphi^{\mathcal{T}}. \tag{6.14}$$

$\diamond$

Importantly, $\mathcal{ALCQ}2SMT_{\mathcal{C}}$ as defined in Definition 5, allow to solve the TBox consistency and concept satisfiability problems via encoding into SMT($\mathcal{C}$), as stated by the following results.

Theorem 6. *An $\mathcal{ALCQ}$ acyclic TBox $\mathcal{T}$ in normal form is consistent if and only if the SMT($\mathcal{C}$)-formula $\varphi^{\mathcal{T}}$ of $\mathcal{ALCQ}2SMT_{\mathcal{C}}(\mathcal{T})$ (Definition 5) is satisfiable.*

Theorem 7. *Given an $\mathcal{ALCQ}$ acyclic TBox $\mathcal{T}$ in normal form and the encoding $\mathcal{ALCQ}2SMT_{\mathcal{C}}(\mathcal{T}) = \langle \Sigma^{\mathcal{T}}, \mathcal{I}^{\mathcal{T}}_-, \mathcal{I}^{\mathcal{T}}_+, A_{\langle\ ,\ \rangle}, \mathsf{indiv}, \varphi^{\mathcal{T}}\rangle$ of Definition 5, then the normal concept $\hat{C}$, such that $\hat{C} \sqsubseteq \hat{D} \in \mathcal{T}$, is satisfiable wrt. $\mathcal{T}$ if and only if the SMT($\mathcal{C}$)-formula $\varphi^{\mathcal{T}} \wedge L_{\langle 1,\ \hat{C}\rangle}$ is satisfiable.*

In order to not break the flow of the exposition, the proof have been moved to Appendix 6.9. For the same reason we include as Appendix 6.10 one example of the $\mathcal{ALCQ}2SMT_{\mathcal{C}}$ encoding.

We remark some facts on the above exposed encoding of Definition 5:

- Notice that, at the effect of the encoding, at-most restrictions occurring at the left-hand side of an axiom behave in the same way right-hand side at-least restrictions behave, and vice versa (see points 5. and 8.). This is necessary for the Theory of Costs. In fact in the Theory of Costs the final value of a cost variable is determined by the sum of the enabled (i.e., assigned to true) incur costs. Thus, if no incur cost is defined for a given cost variable, then the final value of the variable would be zero. In fact, point 5. is necessary both to allow for satisfying implied at-least restrictions and also to guarantee that a left-hand side at-most restriction could be potentially falsified (avoiding some axioms to be applied). In contrast, the clauses at point 8. work in the opposite way: the are introduced to allow for detecting the unsatisfiability of right-hand side at-most restrictions and, vice versa, to potentially force the application of axioms having an at-least restriction on the left-hand side. Last, notice that left-hand side universal restrictions behave in the same way of at-most left-hand side restrictions.

- Point 4. is necessary to force the encoding of axioms having left-hand side restrictions. Such kind of axioms can easily create cycles in TBoxes, thus we remark that our encoding ensures termination only for acyclic TBoxes.

- In the clauses of type (6.4), (6.5), (6.8), (6.9) and (6.4), (6.10), every IC-literal has cost value 1 and has the same index of the bound individual.. This ensures that IC-literals referring to distinct individuals/cost variables are represented by distinct $\mathcal{C}$-atoms.

- Point 7. is meaningful when $C \neq D$. In fact, if $C = D$ then clauses (6.8) and (6.9) exactly correspond to clauses (6.4) and (6.5).

- Clauses (6.6) and (6.11), are those concretely ensuring the numerical satisfiability of both at-least and at-most restrictions. Whilst, in order to be satisfied, a clause of type (6.6) forces some IC-literals to be assigned to true (explaining the fact that the implications (6.4) and (6.8) work only in one direction), a clause of type (6.11) bounds the number of IC-literals that can be enabled (motivating the implications (6.10) and their opposite direction).

Notice that if, for the same σ, r and C, many $\langle \sigma, \geq nr.C \rangle \in \mathcal{I}_+^{\mathcal{I}}$ or $\langle \sigma, \leq nr.C \rangle \in \mathcal{I}_-^{\mathcal{I}}$ fall in the conditions of point 5. for different values of n, being n^* the highest of these values, then only n^* new individuals and n^* instances of clauses of type (6.4) and (6.5) are in $\varphi^{\mathcal{I}}$. In contrast, one distinct clause of type (6.6) is in $\varphi^{\mathcal{I}}$ for every different value of n; in fact to every different concept instantiation, e.g. $\langle \sigma, \geq nr.C \rangle$, corresponds a different Boolean

variable, e.g. $A_{\langle\sigma,\ \geq nr.C\rangle}$. (The same observation holds for the clauses of type (6.11) in the case of different values of m wrt. the same σ, r and C.)

Importantly, wlog., hereafter we generically refer to at-least and at-most restrictions or, respectively, to generic instantiated concepts $\langle\sigma, \geq nr.C\rangle$ or $\langle\sigma, \leq mr.C\rangle$ occurring during the encoding, meaning the right-side (implied) ones, but implicitly including also the cases of left-side at-most (or universal) and, respectively, left-side at-least restrictions. Here below we show an example concerning the effect in the encoding of left-hand side qualified number restrictions, as remarked in the previous points.

Example 6.5.1. Consider the two TBoxes composed of the following four axioms:

$$C \sqsubseteq (\geq 1\ t.A \sqcap \forall t.B \sqcap \forall t.D),$$
$$A \sqsubseteq \geq 1\ r.X, \qquad B \sqsubseteq \geq 1\ r.\neg X, \qquad D \sqsubseteq \forall s.\neg Y,$$

and, alternately, of one further axiom between the following axioms:

$$\geq 2r.\top \sqsubseteq \geq 1\ s.Y \tag{6.15}$$
$$\leq 2r.\top \sqsubseteq \geq 1\ s.Y \tag{6.16}$$

We call $\mathcal{T}^{\mathsf{unsat}}$ the TBox including (6.15) and not including (6.16), vice versa we call $\mathcal{T}^{\mathsf{sat}}$ the TBox obtained considering (6.16) instead of (6.15). In this example we consider the problem of determining the satisfiability of the concept C in $\mathcal{T}^{\mathsf{unsat}}$ and $\mathcal{T}^{\mathsf{sat}}$.

Here below we report the relevant parts wrt. the satisfiability of C of both the encodings $\varphi^{\mathcal{T}^{\mathsf{unsat}}}$ and $\varphi^{\mathcal{T}^{\mathsf{sat}}}$. First, we consider the encoding of the first axiom defining C, and the encoding of the included restrictions concerning the role t; then we show the encoding of the axioms defining A, B and D when instantiated in the individual $1.t.1$:

$$\begin{aligned}
&A_{\langle 1,\ C\rangle} \rightarrow A_{\langle 1,\ \geq 1t.A\rangle} && \wedge (A_{\langle 1,\ \geq 1t.A\rangle} \wedge A_{\langle 1,\ \top\rangle}) \rightarrow \neg\mathsf{BC}(\mathsf{indiv}^{A}_{1.t}, 0)\\
&\wedge A_{\langle 1,\ C\rangle} \rightarrow A_{\langle 1,\ \forall t.B\rangle} && \wedge \mathsf{IC}(\mathsf{indiv}^{A}_{1.t}, 1, 1) \rightarrow A_{\langle 1.t.1,\ A\rangle}\\
&\wedge A_{\langle 1,\ C\rangle} \rightarrow A_{\langle 1,\ \forall t.D\rangle} && \wedge \mathsf{IC}(\mathsf{indiv}^{A}_{1.t}, 1, 1) \rightarrow A_{\langle 1.t.1,\ \top\rangle}
\end{aligned}$$

$$\begin{aligned}
&\wedge (A_{\langle 1,\ \forall t.B\rangle} \wedge A_{\langle 1.t.1,\ \top\rangle}) \rightarrow A_{\langle 1.t.1,\ B\rangle} && \wedge A_{\langle 1.t.1,\ A\rangle} \rightarrow A_{\langle 1.t.1,\ \geq 1r.X\rangle}\\
&\wedge (A_{\langle 1,\ \forall t.D\rangle} \wedge A_{\langle 1.t.1,\ \top\rangle}) \rightarrow A_{\langle 1.t.1,\ D\rangle} && \wedge A_{\langle 1.t.1,\ B\rangle} \rightarrow A_{\langle 1.t.1,\ \geq 1r.\neg X\rangle}\\
& && \wedge A_{\langle 1.t.1,\ D\rangle} \rightarrow A_{\langle 1.t.1,\ \forall s.\neg Y\rangle}
\end{aligned}$$

Second, we consider the encoding of the at-least number restrictions wrt. the role r instantiated in the individual $1.t.1$:

$$\begin{aligned}
&\wedge (A_{\langle 1.t.1,\ \geq 1r.X\rangle} \wedge A_{\langle 1.t.1.r.1,\ \top\rangle}) \rightarrow \neg\mathsf{BC}(\mathsf{indiv}^{X}_{1.t.1.r}, 0) && \wedge (A_{\langle 1.t.1,\ \geq 1r.\neg X\rangle} \wedge A_{\langle 1.t.1.r.2,\ \top\rangle}) \rightarrow \neg\mathsf{BC}(\mathsf{indiv}^{\neg X}_{1.t.1.r}, 0)\\
&\wedge \mathsf{IC}(\mathsf{indiv}^{X}_{1.t.1.r}, 1, 1) \rightarrow A_{\langle 1.t.1.r.1,\ X\rangle} && \wedge \mathsf{IC}(\mathsf{indiv}^{\neg X}_{1.t.1.r}, 1, 2) \rightarrow \neg A_{\langle 1.t.1.r.2,\ X\rangle}\\
&\wedge \mathsf{IC}(\mathsf{indiv}^{X}_{1.t.1.r}, 1, 1) \rightarrow A_{\langle 1.t.1.r.1,\ \top\rangle} && \wedge \mathsf{IC}(\mathsf{indiv}^{\neg X}_{1.t.1.r}, 1, 2) \rightarrow A_{\langle 1.t.1.r.2,\ \top\rangle}
\end{aligned}$$

In the case of the TBox $\mathcal{T}^{\mathsf{unsat}}$, i.e. when including axiom (6.15), also the following clauses are encoded into $\varphi^{\mathcal{T}^{\mathsf{unsat}}}$:

$$\begin{array}{ll} \wedge\,(\neg\mathsf{BC}(\mathsf{indiv}^{\top}_{1.t.1.r},1)\wedge A_{\langle 1.t.1,\ \top\rangle})\rightarrow A_{\langle 1.t.1,\ \geq 2r.\top\rangle} & \wedge\, A_{\langle 1.t.1.r.1,\ \top\rangle}\rightarrow \mathsf{IC}(\mathsf{indiv}^{\top}_{1.t.1.r},1,1) \\ \wedge\, A_{\langle 1.t.1,\ \geq 2r.\top\rangle}\rightarrow A_{\langle 1.t.1,\ \geq 1s.Y\rangle} & \wedge\, A_{\langle 1.t.1.r.2,\ \top\rangle}\rightarrow \mathsf{IC}(\mathsf{indiv}^{\top}_{1.t.1.r},1,2) \end{array} \tag{6.17}$$

Notice that the coexistence of the restrictions $\langle 1.t.1, \geq 1\ r.X\rangle, \langle 1.t.1, \geq 1\ r.\neg X\rangle \in \mathcal{I}_{+}^{\mathcal{T}^{\mathsf{unsat}}}$ and $\langle 1.t.1, \geq 2\ r.\top\rangle \in \mathcal{I}_{-}^{\mathcal{T}^{\mathsf{unsat}}}$ causes the introduction in the complete formula $\varphi^{\mathcal{T}^{\mathsf{unsat}}}$ of the clauses encoding the sharing of the individuals $1.t.1.r.1$ and $1.t.1.r.2$ between the (conflicting) concepts X and $\neg X$. However, here we don't show these clauses because they are not relevant for the satisfiability/unsatisfiability of C. If, instead, $\mathcal{T}^{\mathsf{sat}}$ is considered, i.e. when the TBox includes the axiom (6.16) instead of axiom (6.15), the following clauses are encoded into $\varphi^{\mathcal{T}^{\mathsf{sat}}}$:

$$\begin{array}{ll} \wedge\,(\mathsf{BC}(\mathsf{indiv}^{\top}_{1.t.1.r},2)\wedge A_{\langle 1.t.1,\ \top\rangle})\rightarrow A_{\langle 1.t.1,\ \leq 2r.\top\rangle} & \wedge\,\mathsf{IC}(\mathsf{indiv}^{\top}_{1.t.1.r},1,3)\rightarrow A_{\langle 1.t.1.r.3,\ \top\rangle} \\ \wedge\, A_{\langle 1.t.1,\ \leq 2r.\top\rangle}\rightarrow A_{\langle 1.t.1,\ \geq 1s.Y\rangle} & \wedge\,\mathsf{IC}(\mathsf{indiv}^{\top}_{1.t.1.r},1,4)\rightarrow A_{\langle 1.t.1.r.4,\ \top\rangle} \\ & \wedge\,\mathsf{IC}(\mathsf{indiv}^{\top}_{1.t.1.r},1,5)\rightarrow A_{\langle 1.t.1.r.5,\ \top\rangle} \end{array} \tag{6.18}$$

Finally, the following clauses are included both in $\varphi^{\mathcal{T}^{\mathsf{unsat}}}$ and $\varphi^{\mathcal{T}^{\mathsf{sat}}}$, because of the encoding of the restrictions instantiated in $1.t.1$ and concerning the role s:

$$\begin{array}{ll} \wedge\,(A_{\langle 1.t.1,\ \geq 1s.Y\rangle}\wedge A_{\langle 1.t.1,\ \top\rangle})\rightarrow \neg\mathsf{BC}(\mathsf{indiv}^{Y}_{1.t.1.s},0) & \wedge\,\mathsf{IC}(\mathsf{indiv}^{Y}_{1.t.1.s},1,1)\rightarrow A_{\langle 1.t.1.s.1,\ Y\rangle} \\ \wedge\,(A_{\langle 1.t.1,\ \forall s.\neg Y\rangle}\wedge A_{\langle 1.t.1.s.1,\ \top\rangle})\rightarrow \neg A_{\langle 1.t.1.s.1,\ Y\rangle} & \wedge\,\mathsf{IC}(\mathsf{indiv}^{Y}_{1.t.1.s},1,1)\rightarrow A_{\langle 1.t.1.s.1,\ \top\rangle} \end{array} \tag{6.19}$$

Thus, while $\varphi^{\mathcal{T}^{\mathsf{unsat}}}$ includes the clauses (6.17) and (6.19), $\varphi^{\mathcal{T}^{\mathsf{unsat}}}$ includes the clauses (6.18) and (6.19). Notice that the SMT($\mathcal{C}$) formula $\varphi^{\mathcal{T}^{\mathsf{unsat}}}\wedge A_{\langle 1,\ C\rangle}$ is unsatisfiable because both the distinct individuals $1.t.1.r.1$ and $1.t.1.r.2$ must exist forcing, via the Theory of Costs in the clauses (6.17), both $A_{\langle 1.t.1,\ \geq 2r.\top\rangle}$ and $A_{\langle 1.t.1,\ \geq 1s.Y\rangle}$ to be true; such assignment leads to a conflict in the clauses of the group (6.19). On the contrary $\varphi^{\mathcal{T}^{\mathsf{sat}}}\wedge A_{\langle 1,\ C\rangle}$ is satisfiable. In fact, clauses (6.18) allow for the existence of at least a third individual (among $1.t.1.r.3, 1.t.1.r.4$ and $1.t.1.r.5$) such that, if existing, causes the literal $\mathsf{BC}(\mathsf{indiv}^{\top}_{1.t.1.r},2)$ in the first clause of the group (6.18) to be assigned to false, so that also the literals $A_{\langle 1.t.1,\ \leq 2r.\top\rangle}$ and $A_{\langle 1.t.1,\ \geq 1s.\top\rangle}$ can be assigned to false, and so that the clauses of the group (6.19) do not conflict in $\varphi^{\mathcal{T}^{\mathsf{sat}}}\wedge A_{\langle 1,\ C\rangle}$. ◇

6.5.2 An Encoding Algorithm

Here we sketch an algorithm for building the encoding previously defined. In doing so, we follow Definition 5, which already outlines the structure of a possible algorithm building $\mathcal{ALCQ}2SMT_{\mathcal{C}}(\mathcal{T})$.

The algorithm is based on expansion rules which mimic Definition 5 by extending the set $\Sigma^{\mathcal{T}}$ with new individuals and by adding new clauses to the SMT($\mathcal{C}$) formula $\varphi^{\mathcal{T}}$. The sets of the Boolean literals that have been introduced in $\varphi^{\mathcal{T}}$ at the left-hand side and at the right-hand side of the implications automatically represent, respectively, the respective instantiated concepts in the sets $\mathcal{I}^{\mathcal{T}}_{-}$ and $\mathcal{I}^{\mathcal{T}}_{+}$. Each time a new individual is introduced in $\Sigma^{\mathcal{T}}$ it is enqueued into a *queue of individuals* Q. Expansion rules are then applied individual-by-individual wrt. the last individual σ dequeued from Q, so that all the expansion rules concerning σ are applied consecutively and before all the rules concerning any other individual (in particular any successor $\sigma.r.i$). Henceforth, the algorithm handles individuals in a BFS manner. In more details it works as follows:

Initialization $\Sigma^{\mathcal{T}}$ and the queue Q are initialized with the root individual 1, while $\varphi^{\mathcal{T}}$ is initially set to the unit clause "$A_{\langle 1,\ \top\rangle}$". Then $\varphi^{\mathcal{T}}$, as consequence of the points 2. and 3. of the definition of $\mathcal{ALCQ}2SMT_{\mathcal{C}}(\mathcal{T})$, is extended encoding all the TBox axioms in 1 with clauses of type (6.3).

Iteration For every individual σ extracted from Q, the following expansions phases are applied in the given order:

(a) Realizes the points 3. and 4. of Definition 5 handling axioms. Pure propositional clauses are added to $\mathcal{ALCQ}2SMT_{\mathcal{C}}(\mathcal{T})$ exhaustively encoding all the implications of type (6.3) representing the axioms of $\mathcal{T}$. Every axiom is ensured to be encoded at most once, only if it is not yet "σ-expanded" and if the premises of the axiom are fully matched in σ.

Then, the restrictions instantiated in σ are grouped wrt. the role r they refer to, and the next four phases are applied for every group of restrictions:

(b) Realizes the points 5. and 6. [resp. 9.] of Definition 5 handling at-least restrictions. At-least restrictions are handled before all the other restrictions since they are the responsible of the introduction of new individuals. Given r, the different at-least restrictions $\langle \sigma, \geq n_i r.C\rangle$ wrt. the same concept C are sorted in decreasing order wrt. the occurring value n_i. This has been done, new individuals and clauses of type (6.4) and (6.5) (point 5.) are introduced once for every different concept C in the case of the highest n_i. Instead, one different clause of type (6.6) [resp. (6.12) or (6.14)] is added for every distinct restriction.

(c) Handles multiple at-least/at-most restrictions. If, wrt. σ and r, many at-least restrictions coexist with some at-most ones, then a further encoding phase is necessary in order to encode the sharing of all the individuals $\sigma.r.i$, as provided by point 7. of Definition 5. Thus, in this circumstance, all the clauses of type (6.8) and (6.9) are introduced for every at-least restriction $\langle \sigma, \geq n_j r.C_j\rangle$ and every $\sigma.r.i \in \Sigma^{\mathcal{T}}$.

(d) Handles at-most restrictions, realizing the points 8. and 9. [resp. 6.] of Definition 5. Given r, the different at-most restrictions $\langle \sigma, \leq m_i r.C\rangle$ wrt. the same

concept C are grouped. The clauses of type (6.10) (point 8.) are introduced only once for every different concept C, while one clause of type (6.11) [resp. (6.7)] is added for every different C and every different kind of restriction or value of m_i.

(e) Handles universal restrictions, realizing the point 10. of Definition 5. For each restriction $\langle \sigma, \forall r.C \rangle$ and every $\sigma.r.i \in \Sigma^{\mathcal{T}}$, the algorithm introduces one clause of type (6.13).

We avoid to show a pseudocode for the exposed algorithm, because it would be very long without adding any useful information wrt. Definition 4 and the above exposed textual description.

Proposition 8. *Given an acyclic $\mathcal{ALCQ}$ TBox $\mathcal{T}$ in normal-form, the above exposed encoding algorithm terminates.*

Proof. Notice that the expansion rules building $\Sigma^{\mathcal{T}}$ and $\varphi^{\mathcal{T}}$ from $\mathcal{T}$ are of two kinds: either (a) they propositionally expand an axiom of $\mathcal{T}$ or (b)-(e) they encode restrictions introducing new individuals, concept names in those individuals and incur/bounded costs for the relative cost variable. Termination is guaranteed due to the following facts:

- every axiom of $\mathcal{T}$ is expanded at most once for every σ enqueued in Q;
- a bounded number of individuals is introduced every time a qualified number restriction is handled;
- the expansion rules reduce the complexity of the concepts they handle, until they reduce to concept names;
- $\mathcal{T}$ is assumed to be acyclic, thus a concept name can not recur cyclically.

□

Since $\mathcal{ALCQ}$ has the finite tree model property (Lutz, Areces, Horrocks, & Sattler, 2005) a worst-case exponential (in the size of $\mathcal{T}$) finite model for $\mathcal{T}$ is ensured to exist. In particular, no blocking techniques are necessary to find such a module since $\mathcal{T}$ is acyclic. While, for every σ, the number of expansion of the first kind performed by our encoding algorithm is linear in the size of $\mathcal{T}$, the number of expansions of the second kind depends on the size of $\mathcal{T}$ and on the number of new individuals introduced, i.e. from the values occurring in qualified number restrictions. Thus, overall, the size of $\varphi^{\mathcal{T}}$ is bounded by the product of the sum of the values occurring in the qualified number restrictions of $\mathcal{T}$ and the size of a model for $\mathcal{T}$. Moreover, it is easy to see that our encoding algorithm is output-polynomial (in $\varphi^{\mathcal{T}}$), since every encoding phase can be performed in polynomial time.

6.6 Optimization: Smart Individuals Partitioning

A main drawback of the basic $\mathcal{ALCQ}2SMT_{\mathcal{C}}$ encoding is that it introduces exactly as many new individuals as is the sum of the values included in all the at-least restrictions instantiated in every σ. Thus, the total number of individuals introduced by our basic encoding can be potentially huge. This number, moreover, can exponentially increased when nested number restrictions must be encoded. significantly impacting on the size and on the hardness of the resulting SMT($\mathcal{C}$) formula. However, it is quite intuitive that it is not strictly necessary to introduce all such individuals. In this section we propose a numerical techniques to cope with this problem, and to reduce the size and hardness of our encodnig.

6.6.1 The Need of Partitioning

In general, it is possible to handle groups of individuals having identical properties (i.e. which belong to the same concepts interpretations) instead of using single ones, and then to use only one "proxy" individual as representative for all the individuals of the group. Thus, similarly to the hybrid approach of (Faddoul et al., 2008; Faddoul & Haarslev, 2010), the idea is to to compute a partitioning of the individuals, and then to replace in the encoding many signle individuals (and the relative variables/clauses) with only one proxy individuals. Proxy individual has been used by (Faddoul et al., 2008; Faddoul & Haarslev, 2010), for representing one of the exponentially-many mutually disjoint decompositions of all the concepts interacting through qualified number restrictions. Indepenendtly from the cardinality of the partition it represents, a proxy individual is "the witness" of the consistency/inconsistency of the properties of all the individuals of the group.

As stated above, partitions must be made of individuals with identical properties. Looking at Definition 5 we point out the following facts:

- Except for the root individual 1, individuals $\sigma.r.i$ are introduced (like in a tree) in order to encode at-least restrictions $\langle \sigma, \geq nr.C \rangle$. Importantly, since $\mathcal{ALCQ}$ doesn't allow for role hierarchies, first of all individuals are naturally partitioned in groups wrt. r and wrt. the individual σ of which they are r-successors.

- If, given σ and r, only at-least restrictions $\langle \sigma, \geq nr.C \rangle$ exist, for every such restriction all the individuals $\sigma.r.k_i^C$, with $i = 1, \ldots, n$, can be part of one single partition. In fact, in this circumstance the sharing of individuals is not necessary.

- Otherwise, i.e. when both at-most restrictions and the at-least restrictions $\langle \sigma, \geq n_j r.C_j \rangle$ coexist for the given σ and r, the $N = \sum_j n_j$ individuals introduced can still be partitioned, but the partitioning must allow for representing pssobile intersections between the interpretations $C_j^{\mathcal{I}}$.

However, in the latter case, not all all the possible cardinalities of the intersections must be considered. Instead, it is sufficient to distinguish between the empty intersections and some "limit" cases, depending on the specific values occurring in the qualified number

restrictions. To sum up, given σ and r, we can compute a partitioning of all the individuals $\sigma.r.i$ referring to σ and r, by taking into account the values included in the qualified number restrictions which condern σ and r.

Example 6.6.1. For instance, suppose that it is necessary to encode the restrictions: $\langle \sigma, \geq 10 r.C \rangle$ and $\langle \sigma, \geq 1000 r.D \rangle$. The basic $\mathcal{ALCQ}2SMT_{\mathcal{C}}$ encoding would introduce 1010 distinct individuals. Applying the above explained idea, instead, we could divide these 1010 individuals in, e.g., three partitions of respectively 10, 990 and again 10 individuals. This partitioning allows for representing both (but not only) the configuration in which 10 individuals belong to $C^{\mathcal{I}}$ and other 1000 (i.e., 990 plus 10) distinct individuals belong to $D^{\mathcal{I}}$ and also the configuration in which the 10 individuals of $C^{\mathcal{I}}$ are in common with $D^{\mathcal{I}}$, not enabling the other 10 individuals. If, for example, also $\langle \sigma, \leq 1005 r.\top \rangle$ must be encoded, then the last 10 individuals could be further divided into two distinct partitions. This partitioning allows for sharing 0, 5 or 10 of these last 10 individuals between $C^{\mathcal{I}}$ and $D^{\mathcal{I}}$, covering (in general) the cases in which exactly 0, 5, 10, 15, 20, 990, 995, 1000, 1005 or 1010 of these individuals exist in $\Delta^{\mathcal{I}}$ (being part or not of $C^{\mathcal{I}}$ and/or $D^{\mathcal{I}}$). Even if not exhaustive these combinations of the 1010 introduced individuals are enought to represent the significant cases concerning satisfiability.

6.6.2 Proxy Individuals and Smart Partitioning

In order to handle partitions of individuals we extend the $\mathcal{ALCQ}2SMT_{\mathcal{C}}$ formalism introducing *cumulative labels* and *proxy individuals*. Given a normal/cumulative label σ' and a role r, a *cumulative label* $\sigma'.r.(i \rightarrow j)$ represents a group of consecutive individuals by mean of the range of integer values $i \rightarrow j$, with $i \leq j$, so that the label represent a set of individuals whose cardinality is $j - i + 1$. A normal label, thus, is simply a special case of a cumulative label, with $i = j$ and cardinality 1. For a normal label we can both write $\sigma'.r.(i \rightarrow i)$ and $\sigma'.r.i$. For instance, in the encoding we can represent c distinct individuals: $\sigma.r.i{+}1, \ldots, \sigma.r.i{+}c$, having the same characteristics, by mean of only one cumulative label $\sigma.r.(i{+}1 \rightarrow i{+}c)$. With a small abuse of notation, in the following we call *proxy individual* any $\sigma.r.(i \rightarrow j)$, meaning both: (i) the cumulative label representing the set of individuals $\sigma.r.i, \sigma.r.i{+}1, \ldots, \sigma.r.j$ and (ii) that $\sigma.r.(i \rightarrow j)$ can be one/any of these individuals acting as proxy for all the other individuals of the set. Hereafter we generally speak of individuals meaning, indifferently, either normal or proxy ones. In particular, every single individual can be seen as proxy of itself.

This has been said, the idea is to compute a *smart* partitioning of the individuals encoded in $\mathcal{ALCQ}2SMT_{\mathcal{C}}$. With *smart* we mean a "safe but as small as possible" partitioning, i.e. a partitioning which reduces as much as possible the number of partitions but which safely preserves the semantic of the problem, so that the cardinalities of the computed partitions allow for representing every relevant case wrt. satisfiability. In Definition 6 we formally define our smart partitioning:

Definition 6. Let $\mathcal{T}$ being an acyclic $\mathcal{ALCQ}$ TBox in normal form and $\mathcal{ALCQ}2SMT_{\mathcal{C}}(\mathcal{T})$ being the SMT($\mathcal{C}$) encoding for $\mathcal{T}$ defined in Definition 5. Given the individual $\sigma \in \Sigma^{\mathcal{T}}$ and the role r of $\mathcal{T}$ we define the arrays: [7]

$$\mathcal{N}^{\geq}_{\sigma.r} \stackrel{\text{def}}{=} \{\ n_i \ \mid \langle \sigma, \geq n_i r.C_i \rangle \in \mathcal{I}^{\mathcal{T}}_{+} \text{ or } \langle \sigma, \leq n_i - 1 r.C_i \rangle \in \mathcal{I}^{\mathcal{T}}_{-} \ \}_i \ ^{8} \quad \text{and}$$
$$\mathcal{N}^{\leq}_{\sigma.r} \stackrel{\text{def}}{=} \{\ m_j \mid \langle \sigma, \leq m_j r.D_j \rangle \in \mathcal{I}^{\mathcal{T}}_{+} \text{ or } \langle \sigma, \geq m_j r.D_j \rangle \in \mathcal{I}^{\mathcal{T}}_{-} \ \}_j$$

representing the collections of all the numerical values included in the qualified number restrictions occurring in σ wrt. r. From $\mathcal{N}^{\geq}_{\sigma.r}$ and $\mathcal{N}^{\leq}_{\sigma.r}$, respectively, we define the integer values:

$$N^{\geq}_{\sigma.r} \stackrel{\text{def}}{=} \Sigma_{n_i \in \mathcal{N}^{\geq}_{\sigma.r}}\ n_i \quad \text{and} \quad N^{\leq}_{\sigma.r} \stackrel{\text{def}}{=} \Sigma_{m_j \in \mathcal{N}^{\leq}_{\sigma.r}}\ m_j.$$

Being 2^X the power set for the set/array X, we define the set $\mathcal{P}_{\sigma.r} \stackrel{\text{def}}{=} \mathcal{P}^{\geq}_{\sigma.r} \cup \mathcal{P}^{\leq}_{\sigma.r}$ as the *smart partitioning* for the $N^{\geq}_{\sigma.r}$ individuals of $\Sigma^{\mathcal{T}}$ of the form $\sigma.r.k$, where:

$$\mathcal{P}^{\geq}_{\sigma.r} \stackrel{\text{def}}{=} \{\ n_S \ \mid S \in 2^{\mathcal{N}^{\geq}_{\sigma.r}},\ n_S \ = 0 + \Sigma_{n_k \in S}\ n_k \ \} \quad \text{and}$$
$$\mathcal{P}^{\leq}_{\sigma.r} \stackrel{\text{def}}{=} \{\ m_S \mid S \in 2^{\mathcal{N}^{\leq}_{\sigma.r}},\ m_S = 0 + \Sigma_{m_k \in S}\ m_k \ \}.$$

Finally, we define $p_i \in \mathcal{P}_{\sigma.r}$ the i-th sorted element of $\mathcal{P}_{\sigma.r}$, so that $p_i < p_{i+1}$, and, in particular, $p_1 = 0$ and $p_{|\mathcal{P}_{\sigma.r}|} = \max\{N^{\geq}_{\sigma.r}, N^{\leq}_{\sigma.r}\}$.

Given σ and r, $N^{\geq}_{\sigma.r}$ of Definition 6 represents the number of individuals of the form $\sigma.r.i$ formely introduced in $\mathcal{ALCQ}2SMT_{\mathcal{C}}$ and which we want to partition in groups. Assuming to include in each computed partition consecutive individuals among $\sigma.r.1, \ldots, \sigma.r.N^{\geq}_{\sigma.r}$, the *smart partitioning* $\mathcal{P}_{\sigma.r}$ represents the set of the indexes of referring to the individual of every partition, so that every partition can be represented by the proxy individual $\sigma.r.(p_{j-1} + 1 \rightarrow p_j)$, with $j > 1$. Notice also that $\mathcal{P}^{\geq}_{\sigma.r}, \mathcal{P}^{\leq}_{\sigma.r}$ and $\mathcal{P}_{\sigma.r}$ are sets, thus, equal values are uniquely represented in them. In particular, the values of p_1 and $p_{|\mathcal{P}_{\sigma.r}|}$ are guaranteed to be the ones mentioned in Definition 6 by the fact that $\emptyset, X \in 2^X$, for any set X. The two partitioning shown in Example 6.6.1 are computed in accordance with Definition 6.

Definition 6 defines a safe partitioning, in fact:

- It takes into account all the values of the qualified number restrictions instantiated for σ and wrt. r.
- It considers all the possible sums of the values n_i [resp. m_j] for all the at-least [resp. at-most] restrictions, which allows for representing all the possible lower-bounds [resp. upper-bounds] in case of disjoint (i.e. with empty intersection) concept interpretations.

[7] Equal n_i or m_j values can repeat as many time as they occur.
[8] The instantiated concepts $\langle \sigma, \forall r.C_i \rangle \in \mathcal{I}^{\mathcal{T}}_{-}$ must be considered like $\langle \sigma, \leq 0 r.\neg C_i \rangle \in \mathcal{I}^{\mathcal{T}}_{-}$.

– The union of $\mathcal{P}^{\geq}_{\sigma.r}$ and $\mathcal{P}^{\leq}_{\sigma.r}$ represents the combination of lower- and upper-bounds, respectively.

– By sorting all the possible sums and by considering the distance between these values as the size of a partition (from $p_{j-1}+1$ to p_j), it also allows for representing all the possible (non-empty) intersections of concept interpretations.

– Not all the cardinalities of the non-empty intersections are possible with this partitioning, but "limit" cases are guaranteed to be represented. In particular, including or not a partition of individuals in one interpretation corresponds to pass from one limit case to another limit case.

6.6.3 Exploit Smart Partitioning in $\mathcal{ALCQ}2SMT_{\mathcal{C}}$

Using partitions and proxy individuals doesn't effect the $\mathcal{ALCQ}2SMT_{\mathcal{C}}$ encoding thanks to the fact that the Theory of Costs allows for arbitrary incur costs. So, for example, it is possible to substitute n clauses referring to the distinct individuals $\sigma.r.k^C_1, \ldots, \sigma.r.k^C_n$, with one single clause referring to the proxy individual $\sigma.r.(k^C_i \rightarrow k^C_n)$, if we assum all such individuals to be part of the same partition. Moreover, if each of the original clauses produce ans incur cost of value 1 (e.g., with the literals $\mathsf{IC}(\mathsf{indiv}^C_{\sigma.r}, 1, k^C_i)$) the unique cumulative clause produces an incur cost of n (e.g., including the literal $\mathsf{IC}(\mathsf{indiv}^C_{\sigma.r}, n, k^C_1)$)

Concretely, we enhance Definition 5 by taking advantage of the partitioning technique defined in Definition 6. In the following we point out only the necessary modification to Definition 5, assuming that, for every σ and r, the partitioning $\mathcal{P}_{\sigma.r}$ is available.

– First of all the set $\Sigma^{\mathcal{T}}$, and the instantiated concepts included in $\mathcal{I}^{\mathcal{T}}_{-}$ and $\mathcal{I}^{\mathcal{T}}_{-}$ are assumed, generically, to be made up of proxy individuals. Then, by consequence, also the functions $A_{\langle\ ,\ \rangle}$ and indiv are assumed to map proxy individuals to, respectively, Boolean and cost variables.

– Second, the n clauses of the types (6.4) and (6.5) at point 5. are replaced by the following:

$$\{\ \mathsf{IC}(\mathsf{indiv}^C_{\sigma.r},\ cost_j,\ idx_j) \rightarrow L_{\langle \sigma_{proxy_j},\ C\rangle} \mid p_j \in \mathcal{P}_{\sigma.r},\ 0 < p_j \leq n\ \} \subset \varphi^{\mathcal{T}}, (6.20)$$

$$\{\ \mathsf{IC}(\mathsf{indiv}^C_{\sigma.r},\ cost_j,\ idx_j) \rightarrow A_{\langle \sigma_{proxy_j},\ \top\rangle} \mid p_j \in \mathcal{P}_{\sigma.r},\ 0 < p_j \leq n\ \} \subset \varphi^{\mathcal{T}}, (6.21)$$

$$cost_j = p_j - p_{(j-1)},\quad idx_j = k^C_1 + p_{(j-1)},\quad \sigma_{proxy_j} = \sigma.r.k^C_1 + p_{(j-1)} \rightarrow k^C_1 + p_j - 1.$$

Notice that since $\mathcal{P}_{\sigma.r}$ includes all the possible sums among all the other possible number restrictions' values and n, then $n, k^C_1 - 1, k^C_1 + n - 1 \in \mathcal{P}_{\sigma.r}$ (i.e., n is exactly partitioned in $\mathcal{P}_{\sigma.r}$). The values of idx_j and σ_{proxy_j}, instead, can be explained remembering that each p_j represents the last index of a partition and that the first index p_1 is 0, while $k^C_1 \geq 1$.

Clauses (6.8), (6.9) at point 7. are modified accordingly.

– Third, the clauses of type (6.10) defined at point 8. must take into account proxy individuals and the relative incur cost. Hence they are replaced by the clauses:

$$\{\ (L_{\langle \sigma.r.(i \to j),\ C\rangle} \wedge A_{\langle \sigma.r.(i \to j),\ \top\rangle}) \to \mathsf{IC}(\mathsf{indiv}^C_{\sigma.r},\ j{-}i{+}1,\ i) \mid \sigma.r.(i \to j) \in \Sigma^{\mathcal{T}}\ \}. \quad (6.22)$$

Clauses (6.13) at point 10. are substituted by clauses handling proxy individuals, in the same way.

– Finally, the differences in the definitions of $\mathcal{I}^{\mathcal{T}}_{-}, \mathcal{I}^{\mathcal{T}}_{+}$ and $\Sigma^{\mathcal{T}}$ for all these points of Definition 5 trivially come by consequence.

We make the following observations:

- If, for σ and r, the conditions of point 7. of Definition 5 do not hold (e.g. no at-most restriction exists), then an even more efficient partitioning requires only the following two clauses for every $\langle \sigma, \geq nr.C\rangle$:

$$\mathsf{IC}(\mathsf{indiv}^C_{\sigma.r}, n, k^C_1) \to L_{\langle \sigma.r.(k^C_1 \to k^C_1+n-1),\ C\rangle},$$
$$\mathsf{IC}(\mathsf{indiv}^C_{\sigma.r}, n, k^C_1) \to A_{\langle \sigma.r.(k^C_1 \to k^C_1+n-1),\ \top\rangle}.$$

- Otherwise, if the conditions of point 7. hold, then $\varphi^{\mathcal{T}}$ contains all the clauses:

$$\{\ \mathsf{IC}(\mathsf{indiv}^C_{\sigma.r},\ p_j - p_{j-1},\ p_{j-1}+1) \to L_{\langle \sigma.r.(p_{j-1}+1 \to\ p_j),\ C\rangle} \mid p_j \in \mathcal{P}_{\sigma.r},\ j > 1\ \} \cup$$
$$\{\ \mathsf{IC}(\mathsf{indiv}^C_{\sigma.r},\ p_j - p_{j-1},\ p_{j-1}+1) \to A_{\langle \sigma.r.(p_{j-1}+1 \to\ p_j),\ \top\rangle} \mid p_j \in \mathcal{P}_{\sigma.r},\ j > 1\ \}$$

for every $\langle \sigma, \geq nr.C\rangle$, as consequence of point 5. and of the sharing of (proxy) individuals performed at point 7..

From these two observations we can conclude that it is convenient to compute a smart partitioning of the $N^{\geq}_{\sigma.r}$ new individuals introduced for σ and r only when the sharing of individuals is performed (point 7.), otherwise one single proxy individual for each at-least restriction can be directly used.

6.6.4 Partitioning Algorithm

In Figure 6.1 we expose the pseudocode of the algorithm computing the smart partitioning of Definition 6, given the individual σ, the role r and the respective arrays $\mathcal{N}^{\geq}_{\sigma.r}$ and $\mathcal{N}^{\leq}_{\sigma.r}$. In particular, in Figure 6.1 instead of computing $\mathcal{P}_{\sigma.r}$ we compute the array $\mathcal{D}_{\sigma.r}$ of the partitions sizes represented by $\mathcal{P}_{\sigma.r}$ (e.g., the j-th element of $\mathcal{D}_{\sigma.r}$ represents $p_j - p_{j-1}$), which are the values in which we are interested.

Proposition 9. *Given the individual σ and the role r, the algorithm of Figure 6.1, which takes as input the arrays $\mathcal{N}^{\geq}_{\sigma.r}$ and $\mathcal{N}^{\leq}_{\sigma.r}$ and computes the smart partitioning $\mathcal{P}_{\sigma.r}$ of Definition 6, has worst-case complexity $O(2^{\max\{|\mathcal{N}^{\geq}_{\sigma.r}|, |\mathcal{N}^{\leq}_{\sigma.r}|\}})$.*

```
IntList compute-Combinations (IntVector N)
// P and Q are both initially empty and are, respectively, a list and a queue of integers
 1.    insert 0 in P as first element;
 2.    for each number n_i in N
 3.       move P to the first element;
 4.       while not end-of P
 5.          let m be the current element of P;
 6.          enqueue n_i + m into Q;
 7.          move P to the next element;
 8.          while (Q is not empty) and
                   (end-of P or s ≤ m, with s,m current elements of Q,P)
 9.             dequeue s from Q;
10.             if end-of P or s < m then
11.                insert s in P before the current element;
12.    return P;

IntVector compute-Partitioning (IntVectors N^≥_σ.r, N^≤_σ.r)
// D_σ.r is a vector of integer values
13.    P^≥_σ.r = compute-Combinations(N^≥_σ.r);
14.    P^≤_σ.r = compute-Combinations(N^≤_σ.r);
15.    P_σ.r = merge(P^≥_σ.r, P^≤_σ.r);
16.    let s be the size of P_σ.r; i = 2; j = 1;
17.    while i ≤ s
18.       d = P_σ.r[i] − P_σ.r[i − 1]; i = i + 1;
19.       if d > 0 then
20.          D_σ.r[j] = d; j = j + 1;
21.    set to j the size of D_σ.r;
22.    return D_σ.r;
```

Figure 6.1: Exponential-time algorithm computing smart partitioning.

Proof. We analyze the complexity of the algorithm of Figure 6.1. Let $n_i \in N$ be the number handled at the i-th iteration of the outer-most cycle (starting at instruction `2.`) of the procedure `compute-Combinations`. At each iteration, from `3.` to `7.`, a number of operations linear in the current size of the list $\mathcal{P}$ is performed: that, in the worst case, is equal to the number of the different possible combinations of k previously handled numbers $n_1, \ldots, n_{i-1}$, with $0 \leq k \leq i-1$. Thus the i-th iteration of the procedure executes a number of operations linear in:

$$\sum_{k=0}^{i-1} \binom{i-1}{k} \;=\; \sum_{k=0}^{i-1} \binom{i-1}{k} 1^k \cdot 1^{i-1-k} \;=\; (1+1)^{i-1} \;=\; 2^{i-1}.$$

Since each combination computed is inserted in the queue Q once, at the instruction `7.`,

and the number of all the operations `8-11.` is linear in the size of $\mathcal{P}$ and Q, then the cost of `compute-Combinations` (counting all the iterations from 1 to $|N|$) is of worst-case complexity $O(2^{|N|})$, consistently with the size of the power set for N. Accordingly, the cost of `compute-Partitioning` is $O(2^{\max\{|\mathcal{N}^{\geq}_{\sigma.r}|,|\mathcal{N}^{\leq}_{\sigma.r}|\}})$. □

However, notice that instructions `10-11.` avoid saving repeated combinations already present in $\mathcal{P}$ (in fact $\mathcal{N}^{\geq}_{\sigma.r}$ and $\mathcal{N}^{\leq}_{\sigma.r}$ are arrays, while $\mathcal{P}^{\geq}_{\sigma.r}$ and $\mathcal{P}^{\leq}_{\sigma.r}$ are sets). This can lead to a sensible cost reduction in the average case, when many values repeat frequently in the handled qualified number restrictions. We believe that, despite the worst-case cost of the smart partitioning algorithm, the reduction in the number of clauses and, especially, the reduction in the number of individuals encoded (which can impact exponentially, when repeated at any nesting level), would significantly enhance the whole performance of our approach. In fact, not only we can gain a significant reduction in the size of the encoding $\mathcal{ALCQ}2SMT_{\mathcal{C}}(\mathcal{T})$ but, especially, partitioning can strongly reduce the hardness of the SMT($\mathcal{C}$) reasoning on the encoded problem. Furthermore, partitioning yields our approach to be independent from the values of the qualified number restrictions in the TBox. With smart partitioning the magnitude/offset of the values doesn't effect the size/hardness of the encoding; the effectiveness of smart partitioning, instead, is effected by the interactions among the values (e.g., the frequency of the values or of the differences among them matter).

6.7 Empirical Evaluation

In order to empirically verify the effectiveness of our novel approach, we have performed a preliminary empirical test session on about 700 synthesized and parametrized $\mathcal{ALCQ}$ problems, on which we solved concept satisfiability wrt. non-empty TBoxes.

We have implemented the encoder called $\mathcal{ALCQ}$2SMT in C++, in which, the *smart partitioning* technique of Section 6.6 can be optionally enabled. In the following we distinguish with the abbreviation `S.P.` the results referring to $\mathcal{ALCQ}$2SMT with enabled smart partitioning. In combination with $\mathcal{ALCQ}$2SMT, we have applied on the resulting SMT($\mathcal{C}$) formulas MathSat (version `3.4.1`) (Bruttomesso, Cimatti, Franzén, Griggio, & Sebastiani, 2008), that actually is the first SMT-solver including the Theory of Costs (Cimatti et al., 2010).

We have downloaded the available versions of the state-of-the-art tools FaCT++ (version `v1.4.0`) (Tsarkov & Horrocks, 2006), Pellet (version `2.1.1`) (Sirin et al., 2007), and Racer (RacerPro version `1-9-0`) (Haarslev & Moeller, 2001; Haarslev & Möller, 2003) in order to compare their performance wrt. those of our approach. We have not included in the comparison HermiT (Motik et al., 2009), that is a hypertableau reasoner and thus its handling of qualified number restrictions is not comparable with the standard tableaux-based reasoners (it is much worse), and the hybrid approach of (Farsiniamarj & Haarslev, 2010), which is still a prototype and not yet publicly available.

All the tests presented in this section have been performed on a biprocessor dual-core `Intel Xeon 2.66 GHz, 64 bit` machine, with `16 GB` of RAM, running `Debian Linux 2.6.18-6-amd64`, where four processes can run in parallel. We set a 1000 seconds timeout for every tool and every concept-satisfiability query. We also fixed a bound of 1 GB of disk space for the SMT($\mathcal{C}$) encoding in output from $\mathcal{ALCQ}$2SMT (however, in the test cases here reported the bound has never been reached).

When reporting the results for one $\mathcal{ALCQ}$2SMT+MathSat configuration (either including or not smart partitioning), the CPU times reported are the sums of both the $\mathcal{ALCQ}$2SMT encoding and MathSat solving times (both including the loading and parsing of the input problem). We anticipate that, for all test problems, all tools under examination (i.e. all the variants of $\mathcal{ALCQ}$2SMT+MathSat and all the state-of-the-art DL reasoners) agreed on the satisfiability/unsatisfiability results when terminating within the timeout (with the exception of Pellet in the test cases of Section 6.7.4).

6.7.1 Test Descriptions

In this section we present the sets of test cases we chose for our evaluation.

As discussed by Farsiniamarj and Haarslev (2010), one major problem with benchmarking in this case is the lack of real-world ontologies including meaningful and significant uses of qualified number restrictions. The current well-known benchmarks are not well suited to address typical real-world needs; there exist not many comprehensive real-world ontologies suitable as benchmarks for hard Description Logics and they mostly do not contain non-trivial numerical constraints. In fact, the current techniques for reasoning with qualified number restrictions in Description Logics often lacks of efficiency, especially when the number of the restrictions is higher or when the values involved in the restrictions their selves are big. For this reason ontology designers most likely avoid the use of these constructors, even if they are very natural (sometimes essential) in many domains. Moreover, the design of benchmarks ontologies, in the last years, concentrated on those constructors that can be described with OWL, while qualified number restrictions are expressive and hard to handle constructors added only to the second and more recent W3C reccomendation OWL 2 (Motik et al., 2009).

Thus, we chose to follow the same benchmarking approach of (Farsiniamarj & Haarslev, 2010) which relies on synthesized test cases. Therefore, we have adapted to $\mathcal{ALCQ}$ the $\mathcal{SHQ}$ problems proposed by (Farsiniamarj & Haarslev, 2010). These problems focus on concept expressions only containing qualified number restrictions and define different sets of problems stressing on different source of complexity of the reasoning in $\mathcal{ALCQ}$, which are:

1. the size of values occurring in number restrictions (namely, n and m in the restrictions of the form $\geq nr.C$ and $\leq mr.C$);
2. the number of qualified number restrictions;
3. the ratio between the number of at-least restrictions and the number of at-most restrictions;

4. the satisfiability versus the unsatisfiability of the input concept expressions.

In the following we describe with more details the six groups of different test cases definedwe have borrowed from the work of Farsiniamarj and Haarslev (2010). In our eveluation we add a further group of problems which tests the effect of having a large variety of different values occurring in number restrictions; while this characteristic shouldn't affect the other reasoners, it represents a significant factor for the effectiveness of our partitioning technique. Every different test problem is characterized by an index i, which influences on one of the above mentioned complexity sources (for instance by determining the number of qualified number restrictions in the concept expression, and so on and so forth); thus, in general (but not for all the test cases) the high is the index i the hard is the problem. Since values occurring in qualified number restrictions are one of the sources of complexity which can strongly influence the performance of reasoning, we further parametrized the original test cases of Farsiniamarj and Haarslev (2010) by adding the parameter n which varies those values when they are not directly related to i. [9] This has been said, in our evaluation we test the satisfiability of the concept C wrt. the groups of TBoxes exposed in the following.

Increasing Values of Numbers Occurring in Restrictions.
First we analyze the effect of having increasingly high values occurring in the qualified number restrictions. We check the satisfiability of C in the TBox:

$$C \sqsubseteq \geq 2i\ r.(A \sqcup B)\ \sqcap\ \leq i\ r.A\ \sqcap\ \leq i\ r.B\ \sqcap\ ((\leq i-1\ r.\neg A) \sqcup (\leq j\ r.\neg B)),$$

with $j = i$ for *satisfiable* problems and $j = i-1$ for *unsatisfiable* ones. In these problems the values occurring in number restrictions increment gradually with i.

In order to be satisfied, the concept C requires to have at least $2i$ r-successors in $(A \sqcup B)$ for every individuals in its own interpretation. The two at-most restrictions $\leq i\ r.A$ and $\leq i\ r.B$ bound to i the number of successors that can be in A and, respectively, in B. Thus i successors are in $(\neg A \sqcap B)$ and the other i must belong to $(A \sqcap \neg B)$. Therefore, it can be concluded that if $j = i$ then C is satisfied by choosing i individuals in B, otherwise C is unsatisfiable.

We call $\texttt{increasing\_lin\_sat}_i$ and $\texttt{increasing\_lin\_unsat}_i$ the satisfiable and unsatisfiable version, respectively, where i ranges in the interval $i = 1, 2, 3, \ldots, 100$. Moreover, we call $\texttt{increasing\_exp\_sat}_i$ and $\texttt{increasing\_exp\_unsat}_i$ the satisfiable and unsatisfiable version, respectively, of an exponential variant of this benchmark, in which i (and accordingly j) is replaced by 10^i.

Backtracking.
One of the major well-known optimization techniques addressing the complexity of reasoning with number restrictions is dependency-directed backtracking or backjumping. Backjumping or conflict-directed backjumping are well-known improved backtracking methods

[9] When listing the chosen values for n we will underline the value originally used in(Farsiniamarj & Haarslev, 2010).

that were adapted to DL-reasoning as dependency-directed backtracking (Horrocks et al., 2000a). In tableau methods, these techniques detect the sources of an encountered clash and try to bypass during backtracking branching points that are not related to the sources of the clash. By means of this method, an algorithm can prune branches that will end up with the same sort of clash. In particular, this technique has shown to significantly improve the performance of DL systems in dealing qualified number restrictions (Horrocks et al., 2000a).

This benchmark tests the performance of the compared systems on some cases in which the effect of backtracking could be particularly important. In order to observe the impact of backtracking, we tested the *unsatisfiable* concept C in the following TBox:

$$C \sqsubseteq \geq n\ r.D_1\ \sqcap\ \cdots\ \sqcap\ \geq n\ r.D_i\ \sqcap\ \leq ni-1\ r.\top,$$
$$D_j \sqcap D_k \sqsubseteq \bot,\ \ 1 \leq j < k \leq i.$$

Due to at-least restrictions an individual in C must have n r-successors in every D_i. Since these $n \cdot i$ successors are instances of mutually disjoint concepts D_i where at most $(ni-1)$ successors are allowed; thus C cannot be satisfied. Plain tableau algorithms, without dependency-directed backtracking, could incur in an exponential number of branching ending in a clash for a failed merging of distinct successors.

In this test suite, every increase of i results in more number restrictions and therefore in a larger number of disjoint concepts. We call these problems $\texttt{backtracking}_\texttt{i}(n)$, where i ranges in the interval $i = 1, 2, 3, \ldots, 20$ and where n regulates the combined effect of the values occurring in number restrictions ($n = 1, 2, \underline{3}, 10$). In particular the case $n = 1$ shows the pure effect of backtracking, which might be further increased by incrementing n.

Satisfiable vs. Unsatisfiable Concepts.
In this experiment the performance of reasoning on problems which ranges form satisfiable to unsatisfiable ones are compared, depending on the values included in the number restrictions. The test cases are concepts containing four qualified at-least restrictions and one unqualified at-most restriction according to the following pattern:

$$C \sqsubseteq \geq 3n\ r.(A \sqcap B)\ \sqcap\ \geq 3n\ r.(\neg A \sqcap B)\ \sqcap\ \geq 3n\ r.(A \sqcap \neg B)\ \sqcap\ \geq 3n\ r.(\neg A \sqcap \neg B)$$
$$\sqcap\ \leq in\ r.\top.$$

Since the four at-least restrictions require mutual disjoint groups of fillers, C requires at-least $12n$ distinct r-successor to be satisfied. Thus C is satisfiable for problems with $i \geq 12$, unsatisfiable otherwise. [10] We call these problems $\texttt{sat\_unsat}_i(n)$, for which we chose the values $i = 1, 2, 4, 6, \ldots, 24$ and $n = 1, \underline{10}$.

[10] Farsiniamarj and Haarslev (2010) proposed a second variant of this problem. In the alternative variant the concept name D replaces $\top$ and is conjoint in all the four at-least restrictions. This second version has been introduced in order to study an unexpected behavior of their hybrid approach in the limit cases $i \leq 3n$ (due to the integrated arithmetic reasoner). However, at the effect of the system we are comparing here, this second variant do not present significant differences wrt. the first one above proposed.

Increasing Number of Qualified Number Restrictions.
The number of qualified number restriction occurring in the problems is one of the factors which mostly influence the complexity of reasoning. Therefore, in this experiment the concept C is built starting from one at-least restriction and then it is extended gradually, at the growing of the index i. In order to keep the ratio between the number of at-least and at-most restrictions fixed, at every step one new at-least and one now at-most restriction are added:

$$\begin{aligned} C \sqsubseteq \geq 4n\ r.\top\ \sqcap\ \geq 2n\ r.D_1\ \sqcap\ \geq 2n\ r.D_2\ \sqcap\ \cdots\ \sqcap\ \geq 2n\ r.D_i \\ \sqcap\ \leq n\ r.(\neg D_1 \sqcup \neg D_2)\ \sqcap\ \leq n\ r.(\neg D_2 \sqcup \neg D_3)\ \sqcap\ \cdots \\ \ldots\ \sqcap\ \leq n\ r.(\neg D_i \sqcup \neg D_{i+1}). \end{aligned}$$

Notice that every such a problem contains exactly $2i + 1$ number restrictions. We call these problems $\texttt{restr\_num}_i(n)$; C is satisfiable for every $i, n \geq 1$. Being a central experiment in our benchmarking, we let i range in $i = 1, 2, 3, \ldots, 100$ and we chose $n = 1, \underline{5}, 50$, so that we test the performance of all the tools also in very extreme cases, where very high values ($n = 50$) occur in the qualified number restrictions.

Increasing Number of Qualified Number Restrictions with Variable Values.
We think that our smart partitioning technique could be very effective in improving the performance of the $\mathcal{ALCQ}$2SMT + MathSat approach. As discussed in the complexity analysis of Section 6.6.4, the effectiveness of smart partitioning increases when the values included into number restrictions repeats frequently, leading both to a smaller number of partitions and to a faster execution of the partitioning algorithm. On the contrary, the performance of our smart partitioning algorithm should deteriorate when the input problem presents a combination of a great number of restrictions (which directly affects the complexity of the partitioning algorithm) and in each restriction occurrs a different value. In particular, the more different are the values the more the complexity of the algorithm approaches to the worst-case complexity, and the more the output encoding results in a greater number of partitions (and, thus, in a larger and harder SMT problem). So we propose the following variant of the $\texttt{restr\_num}_i(n)$ benchmark, that we call $\texttt{var\_restr\_num}_i(n)$:

$$\begin{aligned} C \sqsubseteq \geq 4n\ r.\top\ \sqcap\ \geq 2n\ r.D_1\ \sqcap\ \geq 2(n-1)\ r.D_2\ \sqcap\ \cdots\ \sqcap\ \geq 2(n-i+1)\ r.D_i \\ \sqcap\ \leq n\ r.(\neg D_1 \sqcup \neg D_2)\ \sqcap\ \leq n-1\ r.(\neg D_2 \sqcup \neg D_3)\ \sqcap\ \cdots \\ \ldots\ \sqcap\ \leq (n-i+1)\ r.(\neg D_i \sqcup \neg D_{i+1}). \end{aligned}$$

This group of problems introduces variable values in the qualified number restrictions (notice that all the restrictions includes mutually different values) and an increasing number of restrictions following the index i. In this case it must be $n \geq i$. We chose $n = 100$ and $i = 1, 2 \ldots, n$. [11] Notice that C is still a satisfiable concept.

[11] For instance, if $n = 100$ and $i = 3$ then $\texttt{var\_restr\_num}_3(100) = C \sqsubseteq \geq 400r.\top \sqcap \geq 200r.D1 \sqcap \geq 198r.D2 \sqcap \geq 196r.D3 \sqcap \leq 100r.(\neg D1 \sqcup \neg D2) \sqcap \leq 99r.(\neg D2 \sqcup \neg D3) \sqcap \leq 98r.(\neg D3 \sqcup \neg D4)$, and so on and so forth.

Number of At-least vs. Number of At-most Restrictions.
In addition to the pure number of qualified number restrictions, the ratio between the number of at-least and the number of at-most restrictions could affect the complexity of reasoning. Therefore, in this experiment, the performance of the various systems are evaluated wrt. such a ratio given a fixed total number of restrictions. The structure of the concept expression is similar to the previous ones ($\texttt{restr\_num}_i(n)$) and the concept expressions C are easily satisfiable:

$$\begin{aligned} C \sqsubseteq \geq 4n\ r.\top\ \sqcap\ \geq 2n\ r.D_1\ \sqcap\ \geq 2n\ r.D_2\ \sqcap\ \cdots\ \sqcap\ \geq 2n\ r.D_i \\ \sqcap\ \leq n\ r.(\neg D_1 \sqcup \neg D_2)\ \sqcap\ \leq n\ r.(\neg D_2 \sqcup \neg D_3)\ \sqcap\ \cdots \\ \ldots\ \sqcap\ \leq n\ r.(\neg D_{m-i} \sqcup \neg D_{m-i+1}). \end{aligned}$$

We call these problems $\texttt{restr\_ratio}_i(n)$, and we chose for i the same values proposed by Farsiniamarj and Haarslev (2010), i.e. $i = 0, 1, \ldots, m$, with $m = 14$ and where $n = 1, \underline{5}$. Notice that the number of qualified number restrictions is fixed and is set to $m + 1$, that in our case is 15. Thus, the first problem with index $i = 0$ has a ratio of "at-least"-"at-most" restrictions of 1-14, the second with index $i = 1$ has a ratio of 2-13, and so on and so forth till $i = m = 14$ where the ratio is 15-0.

Notice that, in all the test cases, the concept expressions involving C are always complex concept expressions. Thus after normalization every concept expression reduces to a non-empty TBox with a certain number of axioms.

6.7.2 Comparison wrt. State-of-the-art Tools

We first compare our novel approach wrt. the other state-of-the-art reasoners, evaluating on all the benchmarks described in the previous section the performance of $\mathcal{ALCQ}$2SMT+MathSat and those of the other selected tools.

The results of our experiments are graphically summarized in Figures 6.2, 6.3, 6.4, 6.5, 6.6, and 6.7. In order to make the plots clearly readable in the figures, we have maximized the surface of every plot by moving to the figure's caption all the information and parameters concerning the represented test cases. For each distinct test set and parameters configuration we compared the total CPU times required by each tool to solve the i-th problem. Plots referring the same group of benchmarks are grouped in the same figure.

From the exposed results we notice a few facts:

- $\mathcal{ALCQ}$2SMT+MathSat `S.P.` results one of the best performer in all the test cases but in the artful `backtracking` problems (Figure 6.4) and in the `var_restr_num` problems (Figure 6.6), that have been specifically designed to counteract smart partitioning.
- $\mathcal{ALCQ}$2SMT+MathSat `S.P.` is the absolute best performer in the very hard test set $\texttt{restr\_num}_i(50)$ (Figure 6.6), while, together with either Racer or FaCT++, it is the best performing tool in:

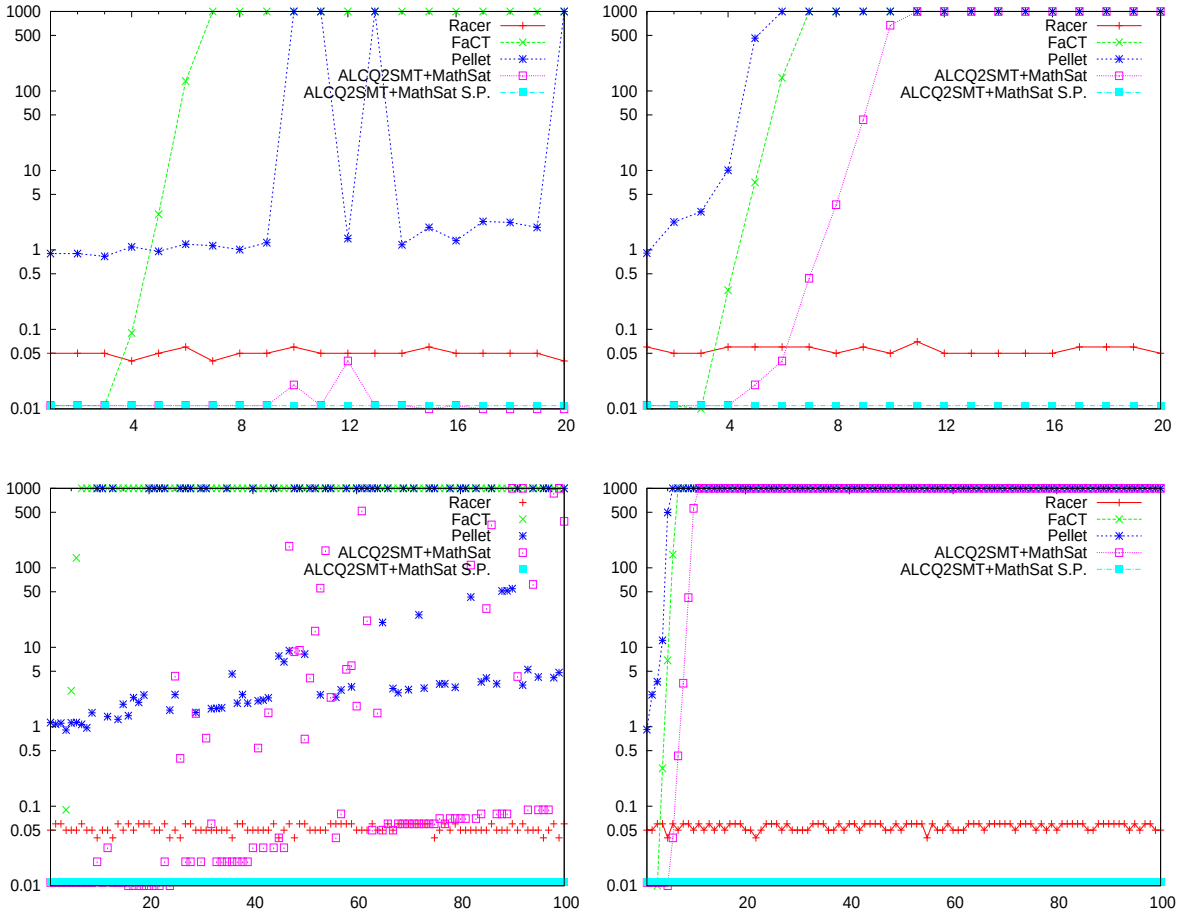


Figure 6.2: 1st column: $\texttt{increasing\_lin\_sat}_i$; 2nd column: $\texttt{increasing\_lin\_unsat}_i$. 1st row: 20 problems; 2nd row: 100 problems. X axis: test case index; Y axis: CPU time (sec).

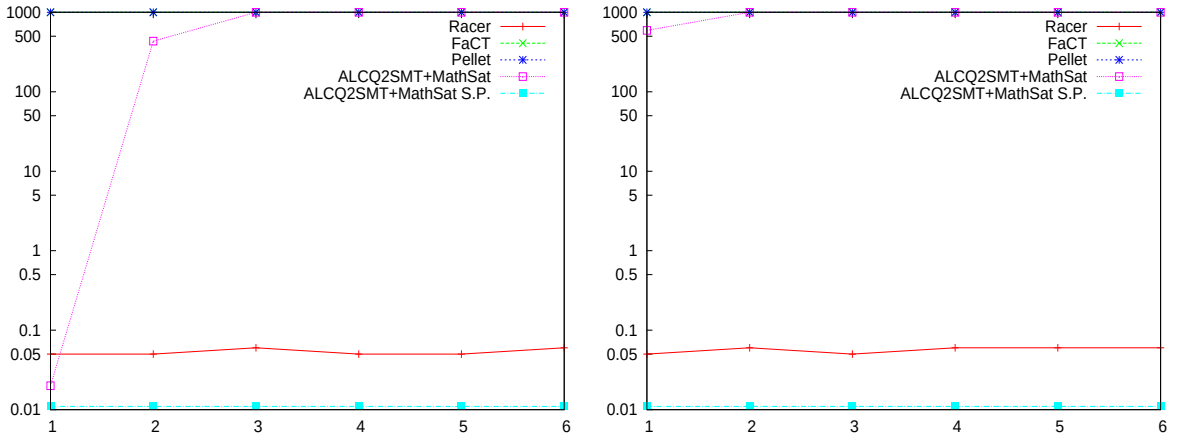


Figure 6.3: Left: $\texttt{increasing\_exp\_sat}_i$; Right: $\texttt{increasing\_exp\_unsat}_i$. X axis: test case index; Y axis: CPU time (sec).

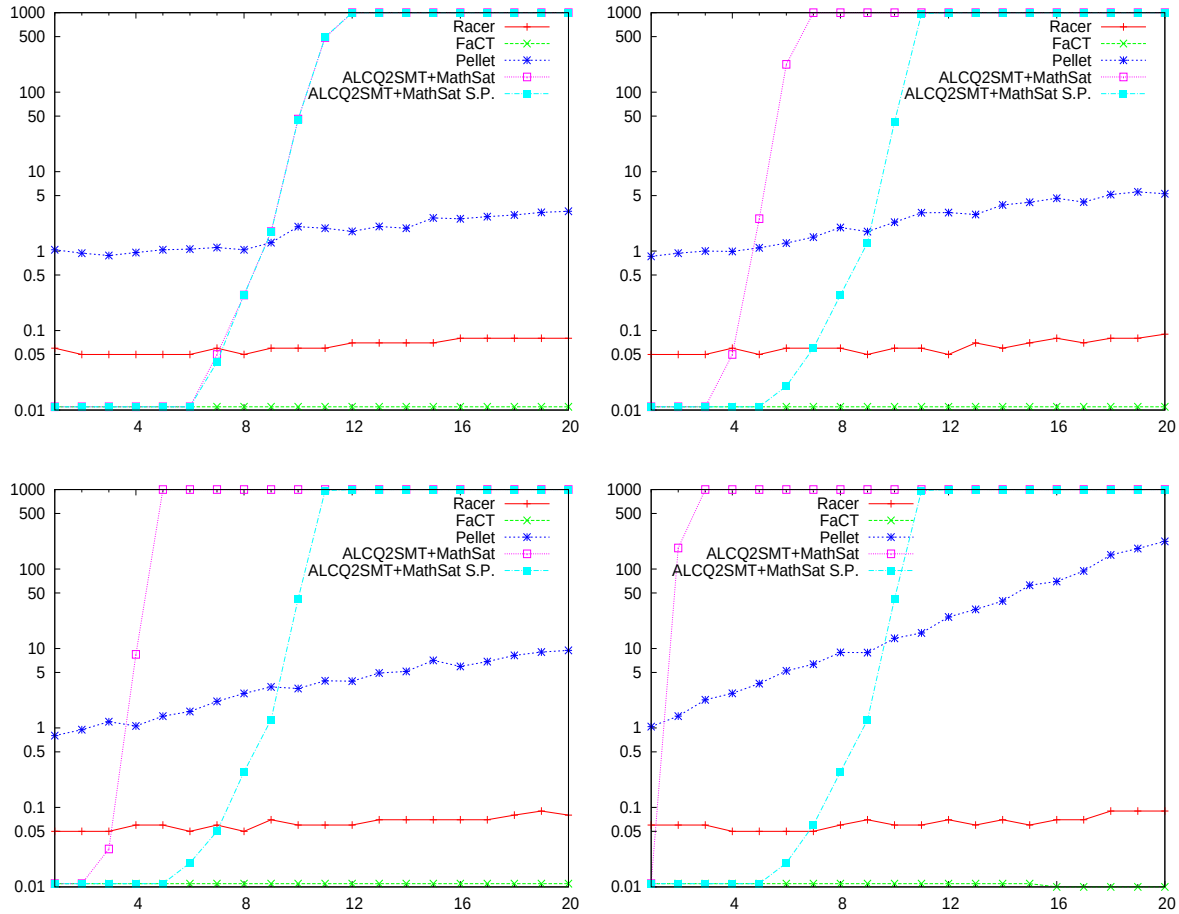


Figure 6.4: $\texttt{backtracking}_i(n)$. Top-left: $n = 1$; Top-right: $n = 2$; Bottom-left: $n = 3$; Bottom-right: $n = 10$. X axis: test case index; Y axis: CPU time (sec).

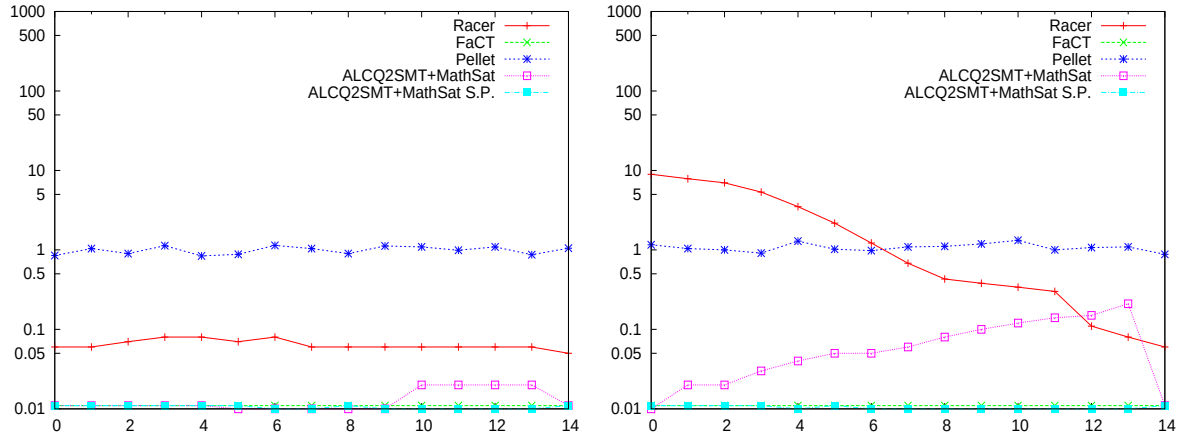


Figure 6.5: $\texttt{restr\_ratio}_i(n)$. Left: $n = 1$; Right: $n = 5$. X axis: test case index; Y axis: CPU time (sec).

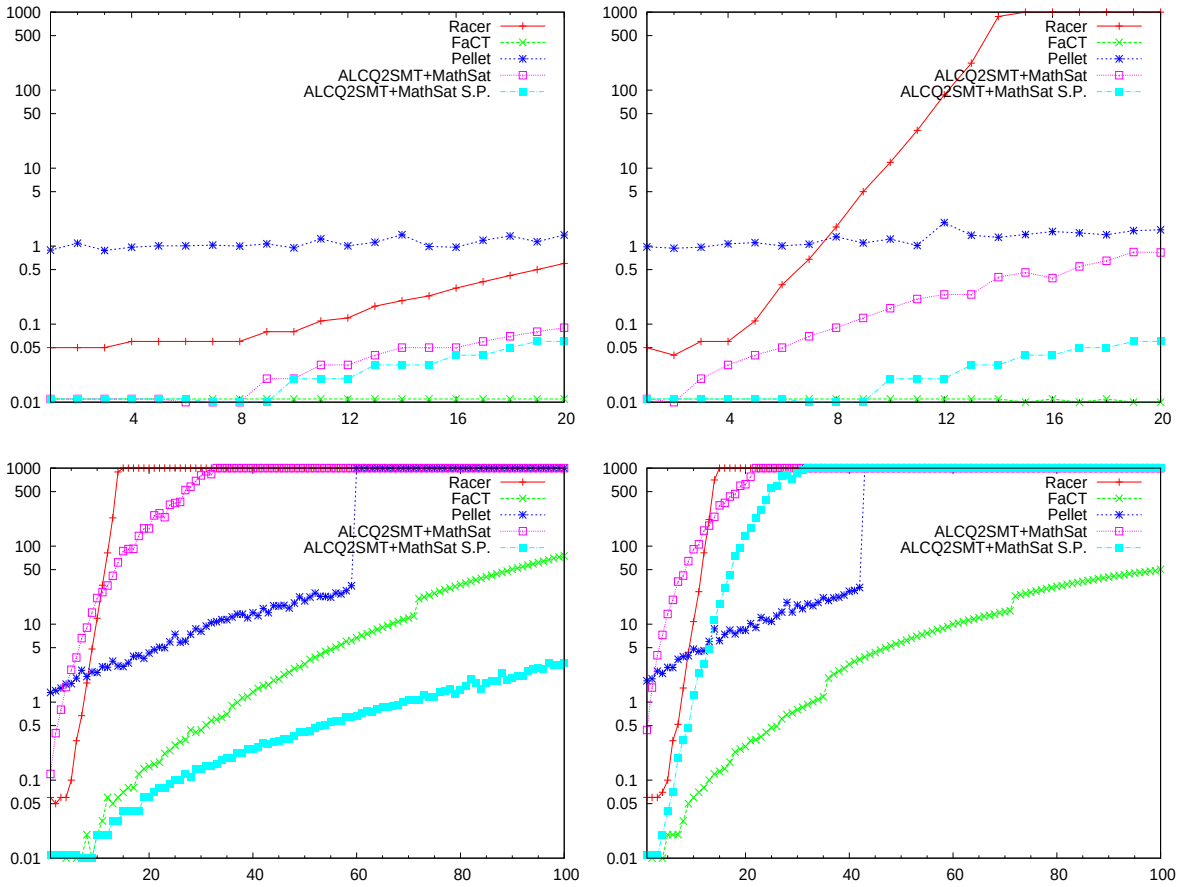


Figure 6.6: Top-left: $\texttt{restr\_num}_i(1)$; Top-right: $\texttt{restr\_num}_i(5)$; Bottom-left: $\texttt{restr\_num}_i(50)$; Bottom-right: $\texttt{var\_restr\_num}_i(100)$. 1st row: 20 problems; 2nd row: 100 problems. X axis: test case index; Y axis: CPU time (sec).

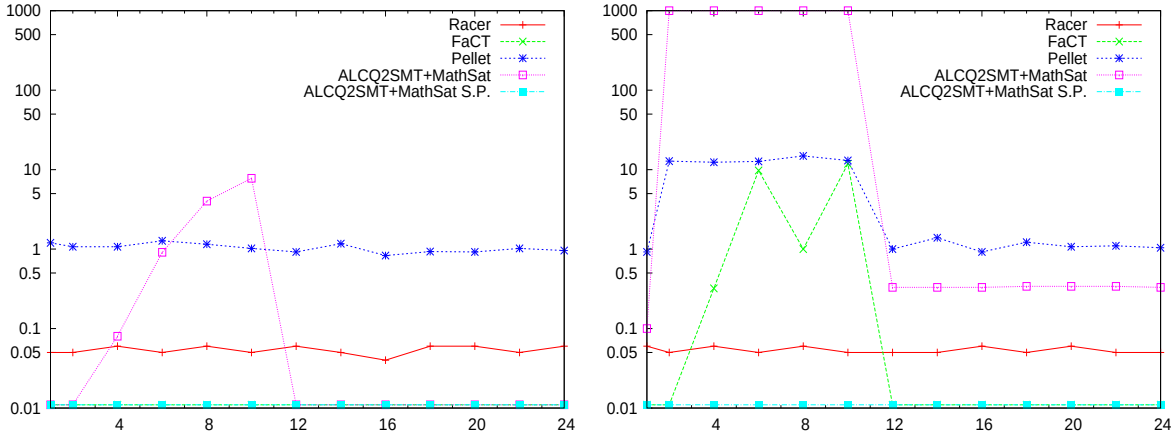


Figure 6.7: $\texttt{sat\_unsat}_i(n)$. Left: $n = 1$; Right: $n = 10$. X axis: test case index; Y axis: CPU time (sec).

  - all the `increasing` test sets (Figures 6.2 and 6.3) (both satisfiable/unsatisfiable and linearly/exponentially increasing);
  - all the `restr_ratio` and all the `sat_unsat` benchmarks (Figures 6.5 and 6.7 respectively), independently from the values of the parameter n.

  Notice that, even if graphically RACER seems to have slightly worst performance wrt. other tools, it takes less than 0.1 sec. to solve most of the benchmark problems. This very small gap is only due to the standard overhead of RACER, that is a very complex and strongly optimized system (it includes a wide set of optimizations, also some specific for qualified number restrictions).

- Overall PELLET seems to be the less efficient system, even if it is not the worst performing tool in each specific test case. This is almost due to the fact that it has a basic overhead of about 1 second on every input problem. [12] FACT++, instead, performs very well in general, except for unsatisfiable problems and problems including high values in the number restrictions. To the best of our knowledge both FACT++ and PELLET have no specific optimization technique for dealing with qualified number restrictions.

- Smart partitioning strongly enhances the performance of the basic $\mathcal{ALCQ}$2SMT+MATHSAT configuration often by reducing the cumulative CPU times of orders of magnitude, or even better "grounding them to zero". However, in many experiments, $\mathcal{ALCQ}$2SMT+MATHSAT scores not worse than some other tools. In particular, $\mathcal{ALCQ}$2SMT+MATHSAT performs better than RACER in all the `restr_num` and `restr_ratio` test cases, and better than FACT++ and PELLET (on average) in all the possible `increasing` benchmark problems.

In more details:

- In the `increasing_lin` test sets (Figure 6.2) $\mathcal{ALCQ}$2SMT+MATHSAT is one of the best performers. It performs comparably with RACER and better than the other reasoners even without smart partitioning. In particular, the basic variant of the approach it is able to solve up to 100 satisfiable problems and 10 unsatisfiable ones. Despite the hardness of the problem it emerges the ability of the SAT/SMT techniques in handling large-size problems. Unsatisfiable benchmarks are much more complex to reason on, in fact (if no specific optimization techniques for number restrictions are applied) they require that all the possible attempts to merge/share individuals fail before to detect unsatisfiability. Thanks to smart partitioning, instead, these problems results straightforward for $\mathcal{ALCQ}$2SMT+MATHSAT, being the encoded problems tirvial absolutely indipendent from the values occurring in the qualified number restrictions. The exponentially increasing test cases `increasing_exp` confirm this analysis; the plots of Figure 6.3 show even more clearly the evidenced effectiveness of smart partitioning.

[12] We think that this high overhead is probably due to the fact that PELLET looks for unsatisfiable concepts instead of checking the specific satisfiability of the queried concept.

- The $\texttt{backtracking}_i(n)$ benchmark problems (Figure 6.4) are the most challenging for $\mathcal{ALCQ}$2SMT+MathSat approach. Also the $\mathcal{ALCQ}$2SMT+MathSat `S.P.` variant cannot solve any `backtracking` problem with index $i \geq 12$, whichever value for the parameter n we chose. In this experiment $\mathcal{ALCQ}$2SMT+MathSat is the worst performer because, even if not huge in size, the encoded `backtracking` problems result very hard to be solved in MathSat. In fact the artful structure of these problems acts on the Boolean component of reasoning and leads to an exponential number of branching decisions and subsequent backtrackings, due to the attempts of merging/sharing disjoint individuals. If we considering the Boolean abstraction of the SMT($\mathcal{C}$) problem generated by $\mathcal{ALCQ}$2SMT, the effect of the encoded `backtracking` problems on the SMT encoding is similar to that of the well-known Halpern & Moses branching formulas (Halpern & Moses, 1992) for modal logic $K_m/\mathcal{ALC}$ on the SAT encoding of Chapter 5 (Sebastiani & Vescovi, 2009a) (see, in particular, Section 5.5.10). In this latter case the exponentiality is caused by a combination of nested existential/universal restrictions and opposite-polarity propositional variables, while in the case of the $\mathcal{ALCQ}$ `backtracking` problems it is caused by the combination of at-least and at-most numerical restrictions involving disjoint concepts. Notice, at last, $\mathcal{ALCQ}$2SMT+MathSat and $\mathcal{ALCQ}$2SMT+MathSat `S.P.` coincide in the base case $n = 1$, but the performance of $\mathcal{ALCQ}$2SMT+MathSat gradually degrade following the increase of the parameter n. With smart partitioning, instead, the hardness of the resulting problem is indipendent from n, but MathSat never succeeds for indexes greater than $i = 11$.

- In the $\texttt{restr\_ratio}_i(n)$ test cases our tools are the best performers together with FaCT++ (Figure 6.5). The total CPU time taken by $\mathcal{ALCQ}$2SMT+MathSat with no partitioning gradually increases following the increase in the number of the encoded individuals. In fact, $\texttt{restr\_ratio}_i(n)$ problems are easily satisfiable, 'cause at-least and at-most restrictions do not mutually conflict. Therefore, for our approach, the only source of complexity is the size. This has been said, the high is the index i of the problem the high is the number of at-least restrictions included in C and, thus, the high is the number of clauses in the SMT($\mathcal{C}$) formula produced by $\mathcal{ALCQ}$2SMT. Notice that if (at least) one at-most restriction is in the concept expression then the number of variables and clauses significantly increase due to the sharing of the individuals and to the encoding of the at-most operator it self. On the contrary, the problem with index $i = m = 14$, which presents no at-most restriction, is trivially satisfiable. While Pellet is very stable for these problems and takes (on average) 1 second for each of them, Racer is surprisingly the worst performer. The higher is the number of at-most restrictions the more the performance of Racer deteriorate.

- Overall, the $\texttt{restr\_num}_i(n)$ and $\texttt{var\_restr\_num}_i(n)$ benchmarks are likely the most challenging problems, especially when combined with high values of the parame-

ter n. From Figure 6.6, it is easy to see that in $\texttt{restr\_num}_i(n)$ the harder is the reasoning (due to the increase of the index i and of the parameter n) the more $\mathcal{ALCQ}$2SMT+MathSat `S.P.` outperforms the other tools. This is almost smart partitioning's merit (e.g., compare the first three plots of Figure 6.6 with the bottom-right one representing $\texttt{var\_restr\_num}_i(100)$ in which smart partitioning is partially inhibited).

In the case of $\texttt{var\_restr\_num}_i(100)$, wrt. $\texttt{restr\_num}_i(50)$, basic $\mathcal{ALCQ}$2SMT+MathSat, as far as Pellet, seem to suffer the transition of n from 50 to 100. While $\mathcal{ALCQ}$2SMT+MathSat and Pellet solve 32 and 59 problems, respectively, of the first mentioned benchmark, they succeed in solving only the first 21 and 42 problems, respectively, of the second one. In $\texttt{var\_restr\_num}_i(100)$, even if $\mathcal{ALCQ}$2SMT+MathSat `S.P.` solves some more problems than the basic variant (respectively 31 against 21), CPU times quickly increase with i due to the lower effectiveness of smart partitioning (due to the variability of the values in number restrictions). Finally, notice that: (i) in the $\texttt{var\_restr\_num}_i(100)$ test set FaCT++ is the only tool able to solve all the problems, (ii) in all the benchmarks of Figure 6.6 Racer is the worst performing system (in fact, for every test case with $n > 1$ Racer solves only 14 problems and its trends seems to be indipendent from n). Considering also the results of Figures 6.2, 6.3 and 6.5 we can guess that Racer is more sensible to the number of qualified number restrictions than to the values occurring in the restrictions themselves.

- The $\texttt{sat\_unsat}_i(n)$ problems (Figure 6.7) confirm the well-known fact that reasoning on unsatisfiable concepts is more difficult than on satisfiable ones. $\mathcal{ALCQ}$2SMT+MathSat in fact, as far as Pellet and FaCT++, presents significantly worse performance in the first unsatisfiable cases than in the second satisfiable ones. This behavior is much more visible for $n = 10$, where $\mathcal{ALCQ}$2SMT+MathSat does not succeed in solve all but one the unsatisfiable problems. The encoding performed by $\mathcal{ALCQ}$2SMT, in effect, inherits some drawbacks of the standard tableau-based approaches. As a matter of fact our encoding indirectly simulates the merging of individuals in the tableau-based algorithm, by mean of the sharing of individuals. Nevertheless, smart partitioning strongly reduces the number of individuals necessary to represent each problem, so that they all result extremely easy for MathSat, independently from n.

6.7.3 Analysis of $\mathcal{ALCQ}$2SMT

We proceed in this section by analyzing the specific behavior of $\mathcal{ALCQ}$2SMT. In particular, we look in more details at the performance of the encoding phase and at the nature of the encoded problems.

In the previous section we have discussed the general performance of $\mathcal{ALCQ}$2SMT+MathSat, without distinguishing between the time spend by

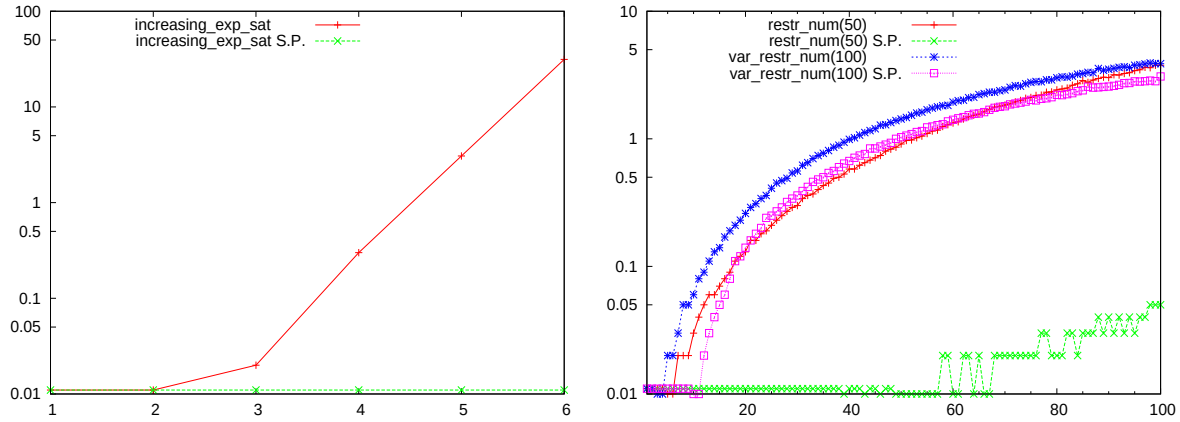


Figure 6.8: CPU times of $\mathcal{ALCQ}$2SMT. Left: $\texttt{increasing\_exp\_sat}_i$, $i = 1, \ldots, 6$; Right: $\texttt{restr\_num}_i(50), \texttt{var\_restr\_num}_i(100)$, $i = 1, \ldots, 100$. X axis: test case index; Y axis: CPU time (sec).

$\mathcal{ALCQ}$2SMT in the encoding phase and the time spent by MathSat in the solving one. So, we first analyze the practical impact of the encoding phase by considering the performance of $\mathcal{ALCQ}$2SMT alone.

In the very majority of the tested problems the time spent by $\mathcal{ALCQ}$2SMT in the encoding phase have resulted negligible (less or equal to 10^{-2} sec.). In Figure 6.8 we plot the only significant test cases in which $\mathcal{ALCQ}$2SMT taken more than one hundredth of a second. Notice, from the left-side plot of Figure 6.8, that $\mathcal{ALCQ}$2SMT takes linear CPU time wrt. the values occurring in the qualified number restrictions in encoding the `increasing_exp_sat` problems. In fact, the $\mathcal{ALCQ}$2SMT CPU time grows exponentially with i exactly as the values grows with a rate of 10^i. On the contrary, $\mathcal{ALCQ}$2SMT `S.P.` results absolutely indipendent from such values (i.e. the encoding time is unchanged for every problem's index) if smart partitioning is applied. The precisely same results have been noticed in handling the `increasing_exp_unsat` group of problems; in fact the satisfiability/unsatisfiability of the problem only affects the solving phase.

From the right-side plot of Figure 6.8, instead, we observe the different effectiveness of smart partitioning on the $\texttt{restr\_num}_i(50)$ and $\texttt{var\_restr\_num}_i(100)$ benchmarks, which are similar in structure but which numerically are considerably different. While smart partitioning succeeds in cutting down the encoding times for $\texttt{restr\_num}_i(50)$, the gain produced by the partitioning technique in the $\texttt{var\_restr\_num}_i(100)$ cases is not significant. Nevertheless, this is not a bad news. In fact, in this last case the possibly onerous cost of the partitioning algorithm does not increase the whole encoding time. As expected, the benefits produced by smart partitioning technique in reducing the number of individuals compensates the computational cost of the partitioning procedure it self, also in the cases in which the technique is not particularly effective. Notice, at last, that the time spent by $\mathcal{ALCQ}$2SMT never exceedes 4 seconds even if the number of restrictions in such problems can be huge.

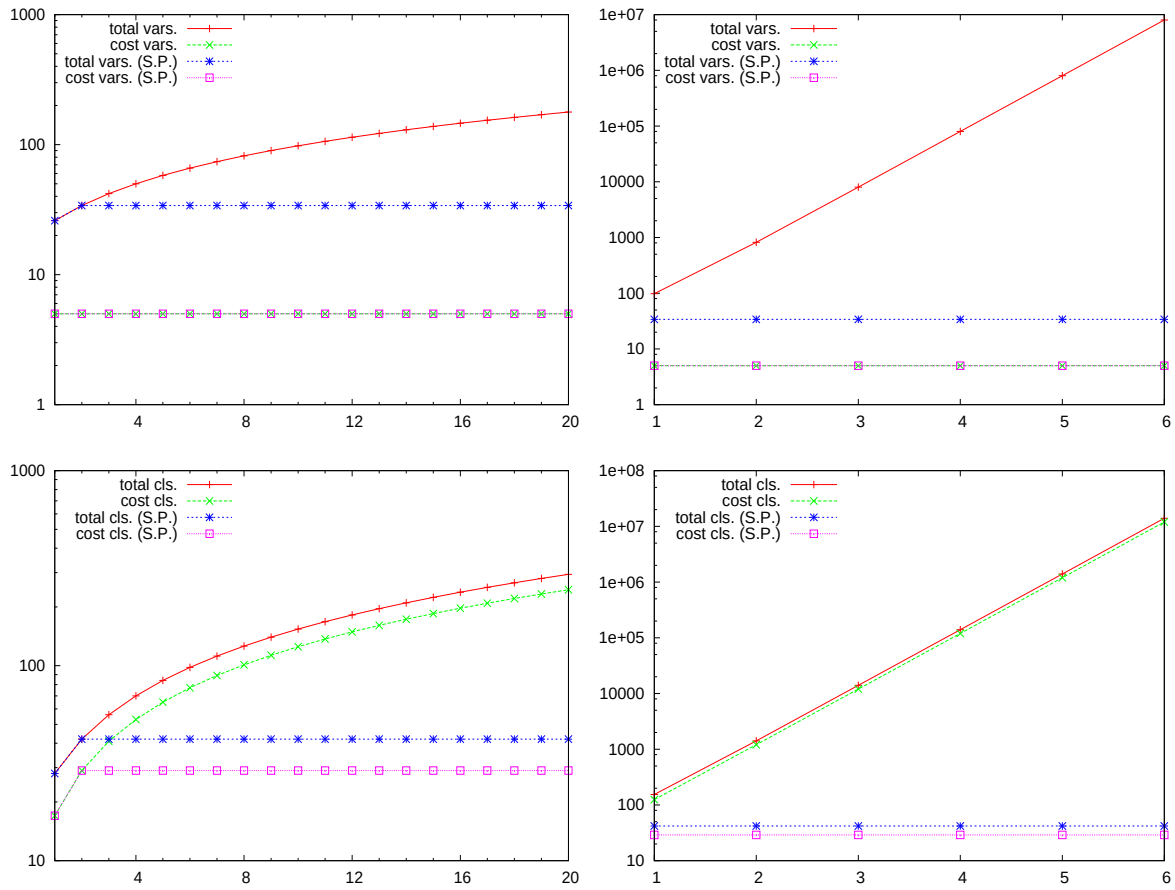


Figure 6.9: 1st column: `increasing_lin_sat`$_i$, $i = 1, \ldots, 20$; 2nd column: `increasing_exp_sat`$_i$, $i = 1, \ldots, 6$. 1st row: variables; 2nd row: clauses. X axis: test case index; Y axis: #variables/clauses.

In Figures 6.9, 6.10, 6.11, 6.12, 6.13, and 6.14, instead we compare in plots the number of variables and clauses produced in output by $\mathcal{ALCQ}$2SMT. In particular, we compare these values for the two variants of $\mathcal{ALCQ}$2SMT, the basic one and the one including smart partitioning (`S.P.`). In the plots we identify with "`total`" the total number of variables or, respectively, the total number of clauses, forming each encoded problem. Instead, we identify with "`cost`" the number of cost variables or, respectively, the number of clauses containing $\mathcal{C}$-literals. Therefore, the difference between the `total` and `cost` curves represents, respectively, the number of Boolean variables and the number of purely propositional clauses generated by $\mathcal{ALCQ}$2SMT. Due to the big number of different benchmarks, we have limited the number of plots included in this section by considering only the most meaningful cases and some representative ones for every different kind of benchmark. In fact, from the point of view of the number of variables and clauses, the `increasing_lin_sat` (Figure 6.9) and the `increasing_lin_unsat`

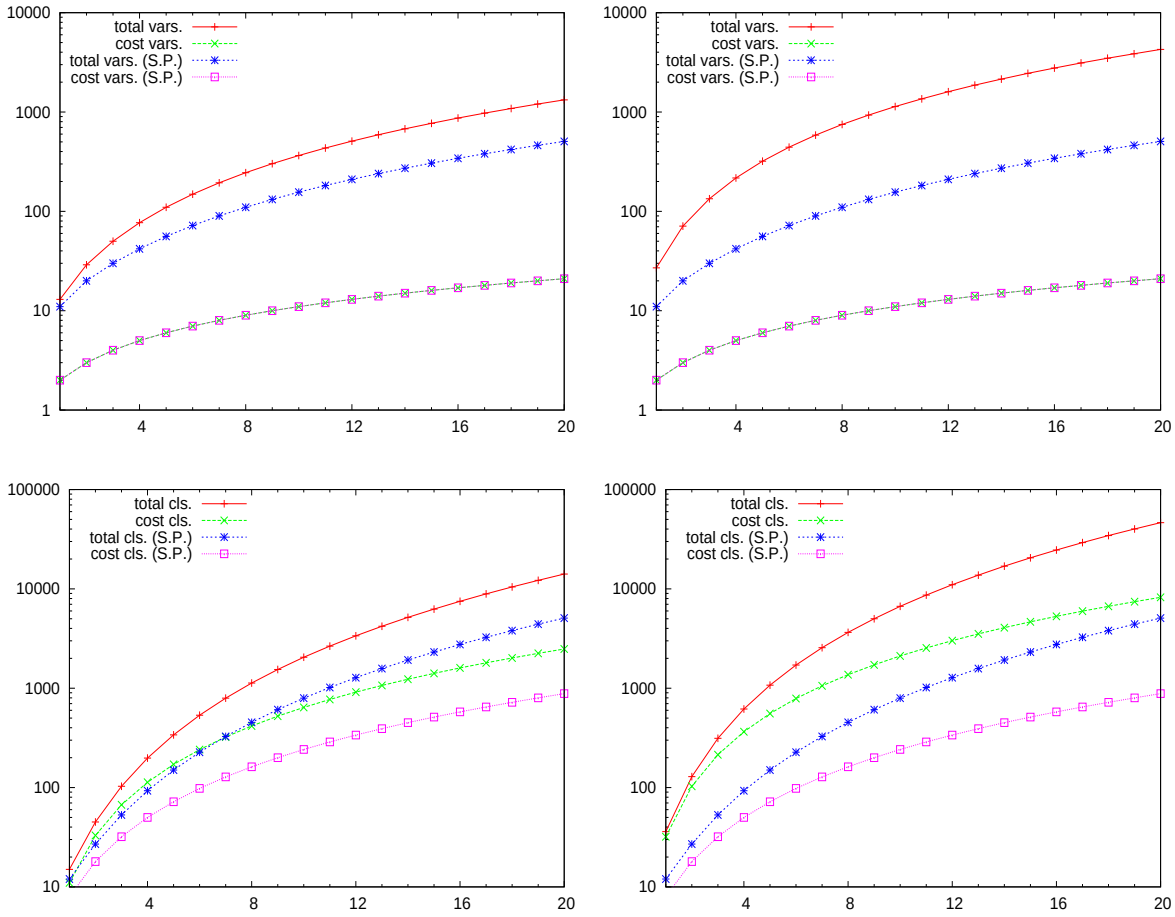


Figure 6.10: $\texttt{backtracking}_i(n)$. 1st column: $n = 3$; 2nd column: $n = 10$. 1st row: variables; 2nd row: clauses. X axis: test case index; Y axis: #variables/clauses.

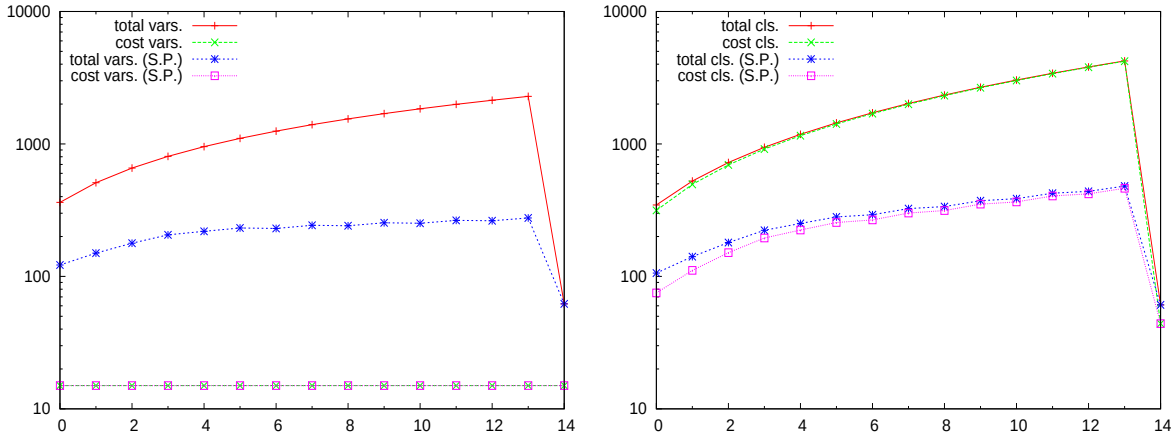


Figure 6.11: $\texttt{restr\_ratio}_i(5)$. Left: variables; Right: clauses. X axis: test case index; Y axis: #variables/clauses.

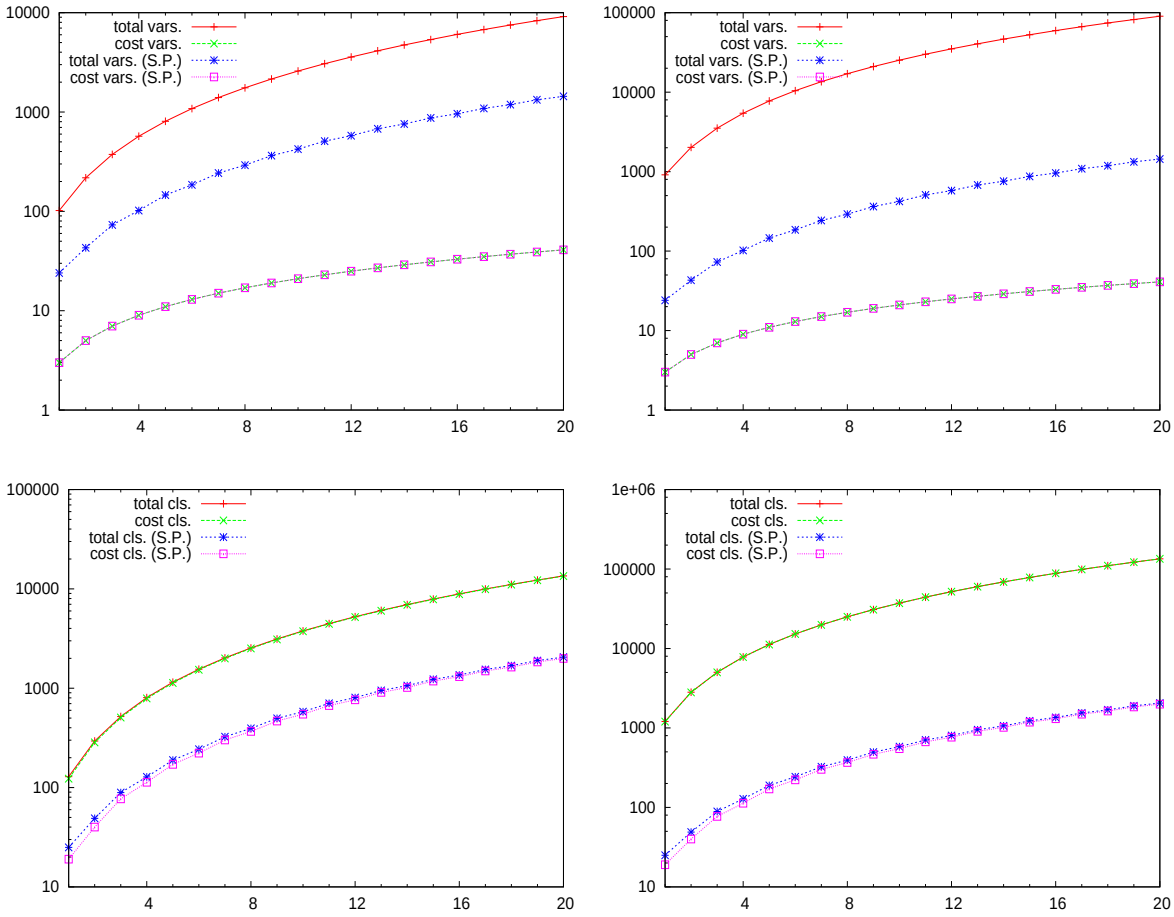


Figure 6.12: $\texttt{restr\_num}_i(n)$. 1st column: $n = 5$; 2nd column: $n = 50$. 1st row: variables; 2nd row: clauses. X axis: test case index; Y axis: #variables/clauses.

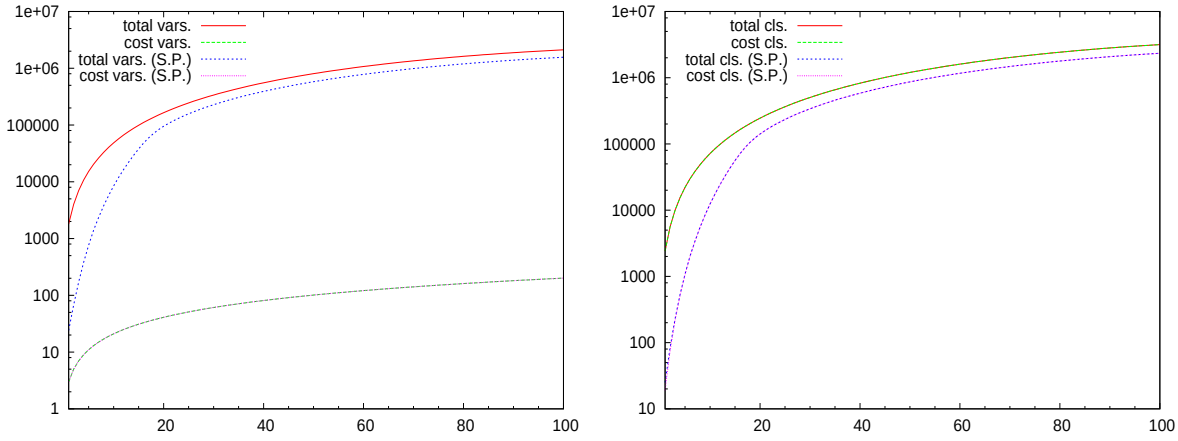


Figure 6.13: $\texttt{var\_restr\_num}_i(100)$. Left: variables; Right: clauses. X axis: test case index; Y axis: #variables/clauses.

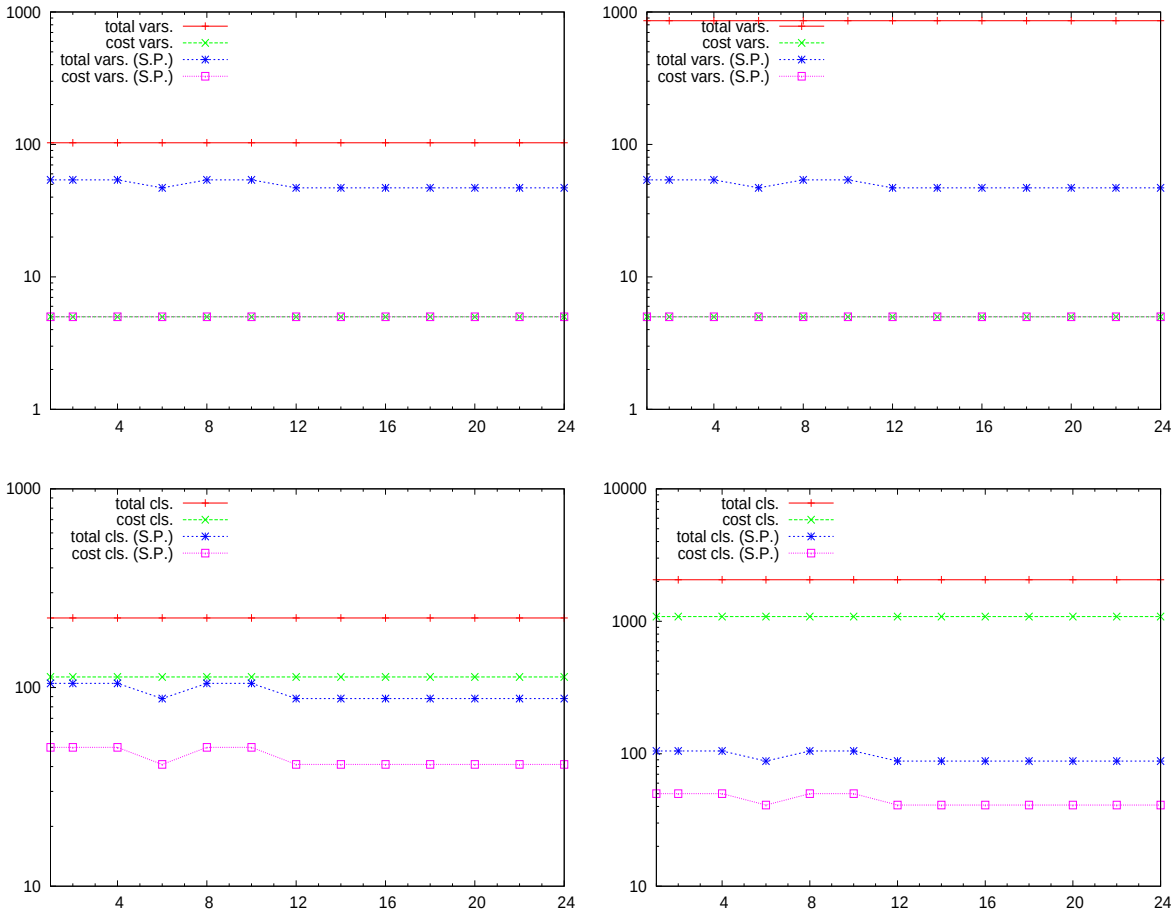


Figure 6.14: `sat_unsat`$_i(n)$. 1st column: $n = 1$; 2nd column: $n = 10$. 1st row: variables; 2nd row: clauses. X axis: test case index; Y axis: #variables/clauses.

problems presents exactly the same characteristics (the same argument is valid for the exponentially increasing problems). In some cases (`increasing_lin_sat/unsat` and `restr_ratio` with $n = 50$) we plot only the problem indexes i up to 20 instead of up to 100, cause they are more clearly readable and equivalently represent the trends of the plotted quantities. For the `backtracking` benchmarks (Figure 6.10), instead, we chose two representative groups of problems with $n = 3$ and $n = 10$. The case $n = 1$ shows no differences between the use or not of smart partitioning, while in the case $n = 2$ the differences are less clearly osservable than in the reported cases. For similar reasons we don't show in Figures 6.11 and 6.12 the configuration $n = 1$ for the `restr_ratio` and the `restr_num` benchmarks respectively. For the sake of the reader's convenience, however, all the other plots are available in the appendix Section 6.11.

From the exposed plots we highlight some facts:

- In all the different test cases $\mathcal{ALCQ}$2SMT and $\mathcal{ALCQ}$2SMT `S.P.` produce exactly

the same number of cost variables. In fact, smart partitioning impacts in reducing the number of individuals but does not change the logic of the encoded SMT($\mathcal{C}$) problem; A different cost variable, in fact, is uniquely introduced for every distinct combination of individual, concept name and role name occurring in the qualified number restrictions. However, if the encoded TBox provides nested qualified number restrictions then the reduction in the number of individuals would lead also to a sensible reduction of the cost variables.

- Many test sets present a very low and fixed number of cost variables, due to the structure of concept expressions which often include a fixed number of qualified number restrictions. For instance, in the `increasing` and `sat_unsat` benchmarks (Figures 6.9 and 6.14), the number of qualified number restrictions and of encoded cost variables is fixed to 5, while it is equal to 15 in the `restr_ratio` benchmarks (Figure 6.11), having chosen $m = 14$. The number of Boolean variables, indeed, is predominant in all the test cases, also in the `backtracking`, `restr_num` and `var_restr_num` cases (Figure 6.10, 6.12 and 6.13), where the number of cost variables linearly increases with the index i.

- As can be easily predicted from the definition of our encoding, the number of total clauses is tightly related to the number of Boolean variables introduced. For this reason, smart partitioning positively affects both in reducing the number of individual/Boolean variables and in reducing the total size of the encoded problems.

- The major part of the clauses encoded by $\mathcal{ALCQ}$2SMT contains $\mathcal{C}$-literals. This property is even more evident for the `restr_num` and `var_restr_num` benchmarks (Figures 6.12 and 6.13). On the contrary, the only exception to this observation is in the `backtracking` test cases (Figure 6.10), where the encoding of the mutual disjunctions conditions between concepts produces a high number of purely Boolean implications.

- Generally, without smart partitioning, the numbers of variables and clauses linearly follow the value of the index i and/or the value of the parameter n. The `increasing` problems are an evedint case of the relation between the size of the encoded problem and the index i. Similarly, notice that in the `sat_unsat` test cases (Figure 6.14), where the size of the problem is indipendent from i, that an increase of one order of magnitude in the value of n, from 1 to 10, determines an increase of one order of magnitude also in the number of clauses and variables encoded by basic $\mathcal{ALCQ}$2SMT.

- Curiously, in the `sat_unsat` benchmark (Figure 6.14) one peculiarity of our partitioning technique is slightly perceptible. Notice, in fact, that enabling smart partitioning the number of variables and clauses are not absolutely unchanged, but present some minimum for the indexes equal to 6 or greater than 12. This tricky behavior depends from smart partitioning, where the combinations of values occurring in at-least restrictions are merged with the combinations of values occurring in

at-most restrictions. In the `sat_unsat` problems, the first are multiple of 3 for a maximum of 12, and the second follow exactly i, with $i = 1, 2, 4, 6, \ldots, 24$. Thus, when i is 6 or is greater than 12 the merging of these values generates one less partition, explaining the slight difference with the other values of i.

- The size of encoded and solved problems can be very large. E.g., $\mathcal{ALCQ}$2SMT+MathSat solves all the problems of the $\texttt{rest\_num}_i(50)$ benchmark, which present up to 10^6 variables and clauses (in Figure 6.12 they reach 10^5 variables and clauses for $i = 20$). Moreover, both with and without smart partitioning, $\mathcal{ALCQ}$2SMT+MathSat has shown able to solve problems with more than 10^5 variables and clauses in the very hard $\texttt{var\_rest\_num}_i(100)$ test set (see, e.g., the problem of index $i = 20$ in the plots of Figure 6.13). Nevertheless, as previously discussed, the size of the problem is not the only source of complexity. For instance, without smart partitioning, the unsatisfiable problems of `sat_unsat` results extremely hard for $\mathcal{ALCQ}$2SMT+MathSat, even if they are stable in the order of "only" 1000 variables and clauses (Figure 6.14) for every i.

6.7.4 Discussion

As similarly discussed in Section 5.6.4 wrt. $\mathcal{ALC}$, the concept satisfiability problem in logics like $\mathcal{ALCQ}$ is characterized by the alternation of many orthogonal components of reasoning. In terms of the semantic of the input problem, i.e. in terms of finding an interpretation for the given TBox/concept, we individuate the following components of reasoning: (i) a propositional component, performing reasoning within each individual, in order to satisfy the concepts involved, the conjunctions, the disjunctions and/or the negations; (ii) a modal component, generating the successor individuals of each individual, in order to satisfy existential/at-least restrictions; (iii) an arithmetical component performing reasoning on the whole space of the possible successor individuals, in order to satisfy the numerical constraints imposed by both at-least and at-most (or universal) restrictions. The first component (i) must cope with the fact that there may be exponentially many candidate models to explore. The second component (ii) must face with the fact that the candidate models may be exponentially big wrt. the nesting depth of restrictions in the input TBox, and (without optimization) wrt. the values occurring in the number restrictions. The last component (iii) must cope to the numerical consistency of all the possible models. This component of reasoning is strongly correlated with the former ones causing a further source of exponentiality when the bounds on the number of individuals cause that exponentially many more models (given by all the possible partitioning of individuals) must be explored.

In the $\mathcal{ALCQ}$2SMT+MathSat approach the encoder has to handle the whole component (ii), whilst the handling of the propositional (i) and arithmetical components (iii) are delegated to the SMT solver (if we except for their interactions with (ii), which result in the encoding of the sharing of individuals). Notice that, with our encoding, the interactions among the three components are regulated in two main ways. The

Boolean component of SMT assigns the Boolean abstraction of the given SMT($\mathcal{C}$) formula, and assigns also the $\mathcal{C}$-literals, determining the existence of individuals and the sharing/merging of them. The $\mathcal{C}$-solver checks the consistency of the assignment wrt. the Theory of Costs $\mathcal{C}$, verifying that all the numerical constraints are satisfied. In the unfavorable case it forces and guides (through theory propagation and theory backjumping, see Section 4.2) the generation of a new assignment.

From the results reported in this section we notice that the performances of our approach strongly depends from the application, or not, of smart partitioning. Even if there are problems in which basic $\mathcal{ALCQ}$2SMT+MathSat is competitive or even outperforms the other tools, the benefits given by smart partitioning are outstanding. The effectiveness of smart partitioning lays in the drastic reduction it produces in the size of the output problems. In particular, the more is the logical complexity of the encoded problem (e.g. unsatisfiable problems) the more prominent are the benefits of smart partitioning, cause the sensible reductions in size affect exponentially during the SMT($\mathcal{C}$)-solving phase.

The relative performances of $\mathcal{ALCQ}$2SMT+MathSat `S.P.` wrt. other state-of-the-art reasoners range from a very few artificial cases where it is much less efficient than other state-of-the-art systems (e.g., the `backtracking` and `var_restr_num` benchmarks) up to formulas where it is much more efficient of all the other tools (e.g., in the `increasing` and `restr_num` test cases). In many cases our novel approach competes well against the other state-of-the art tools, reporting comparable performance.

An explanation of the former observation is that the $\mathcal{ALCQ}$2SMT+MathSat approach suffers, in particular, in two cases. First, in the problems in which there is a strong interaction between either the (i) or the (ii) component of reasoning and the (iii) one, so that the possible exponentiality in the propositional component or in the number of successors causes a huge number of inconsistent calls to the $\mathcal{C}$-solver, which can not be profitably exploited to guide the propositional component via theory backjumping because the encoding is decoupled from the search. Second, wrt. the other approaches, $\mathcal{ALCQ}$2SMT+MathSat relatively lose efficiency in those cases in which a consistent increase in the size of the encoded problem is not balanced by a significant increase in the hardness of the respective input problem, so that our approach is affected by the size of the encoding (due to the encoding of the (ii) component of reasoning) while the other state-of-the-art tools can exploit specific optimizations or reasoning techniques. Smart partitioning specifically acts on reducing the weight of the encoding of the (ii) component.

On the contrary, when enhanced with smart partitioning, our approach dominates in the problems where the high values occurring in qualified number restrictions or the high number of qualified number restrictions undermine the other approaches. Moreover, when smart partitioning reduces even very hard problems to a reasonable-size SMT($\mathcal{C}$) formula, our approach is extremely efficient and outperforms all the other systems. This is due to the power of SAT/SMT techniques which relies on extremely efficient and

well-engineered tools (able to solve large size problems) and on specific and optimized theory solvers. Summarizing our approach have shown to be very effective also in huge or really complex problems in the cases in which the three component of reasoning are well-balanced, or in which either the (i) or the (iii) components prevail, without a too thick interaction with the (ii) one.

A strength of our approach is that, thanks to smart partitioning, it works independently from the values occurring in qualified number restrictions. As we have predicted and shown, the more values repeats in qualified number restrictions the more smart partitioning results effective. Even if the effectiveness of smart partitioning may reduce depending from the properties of the values included in the restrictions, (in particular from their combinations and their variability), the important fact is that the resulting encoding does not depend from the order of magnitude of such values. For instance, if the same variability in the values occurs with either a ratio or an offset of 1 or of 1 million, the result of smart partitioning (i.e. the number of distinct partitions produced) is the same. However, we think that in real-world ontologies extreme cases as those of the `var_restr_num` benchmark rarely occur. Furthermore, even if theoretically it is potentially very expensive, in practice smart partitioning have shown computationally efficient. We think that the time spent in executing the smart partitioning procedure is compensated from the gain it gives in avoiding some encoding steps.

Finally, notice that, in terms of CPU times, the encoding phase performed by $\mathcal{ALCQ}$2SMT mostly results negligible, while the major source of computational complexity lay in the solving phase. This shows that encoding can be convenient when a scalable and efficient solving phase is guaranteed. From this point of view, notice also that the solving time of MathSat (that is a complex and well established SMT solver, differently from $\mathcal{ALCQ}$2SMT that is a prototype) is also often negligible, without any overhead.

Scalability

We close our experimental evaluation by separately discussing the scalability issue. We have chosen to face this issue here, separately, for two main reasons. First, the scalability issue can be seen under many different perspectives. In particular, it involves all the different component of reasoning we previously analyzed and all the different sources of complexity we disjointly examined in this experimental evaluation. Thus, it is helpful having previously discussed all such points. Second, the only way of correctly analyze the scalability of our approach in comparison with the other state-of-the-art-tools should be try the performances of the various systems in increasingly larger and harder real-world ontologies. Unfortunately, as discussed in Section 6.7.1 and by Farsiniamarj and Haarslev (2010), currently this is not possible due to the lack of significant and meaningful real-world ontologies making use of qualified number restrictions. Thus we can only try to combine the "ingredients" that we have previously individually analyzed.

One way of evaluating scalability could be analyze the effect of increasingly more nested restrictions. With this aim we need to introduce a further set of benchmark problems which completely differs from all the benchmarks proposed by the approach of Farsiniamarj and Haarslev (2010), and that we have adapted in this work. Notice that having nested occurrences of qualified number restrictions is a further source of complexity, that much more significantly impacts in our approach than in traditional the tableau-based algorithms, because we handle the component (ii) of reasoning via encoding. This has been said, we add the following class of benchmark problems.

Nesting Depth of Qualified Number Restrictions.
We evaluate the effect of having nested occurrences of qualified number restrictions by solving the satisfiability of C in the following TBoxes indexed by i:

$$\begin{aligned} C &\sqsubseteq \geq 2n\ r.A_1 \sqcap \geq 2n\ r.B_1 \sqcap \leq 3n\ r.\top, \\ A_1 \sqcap B_1 &\sqsubseteq \geq 2n\ r.A_2 \sqcap \geq 2n\ r.B_2 \sqcap \leq 3n\ r.\top, \quad \ldots, \\ \ldots, \quad A_{i-1} \sqcap B_{i-1} &\sqsubseteq \geq 2n\ r.A_i \sqcap \geq 2n\ r.B_i \sqcap \leq 3n\ r.\top. \end{aligned}$$

In these TBoxes the number or nested qualified number restrictions is equal to the value of the index i. The combined effect of the two at-least and of the one at-most restrictions defined in the j-th axiom of the TBox is that of forcing the existence of at least n distinct r-successors in $(A_j \sqcap B_j)$. Consequently, this forces the application of the next $(j+1)$-th axiom, which introduces a deeper nested restriction, and so on and so forth till the i-th (last) axiom. C is satisfiable in every such TBox. We call these problems $\texttt{nested\_restr\_sat}_i(n)$, while we call $\texttt{nested\_restr\_unsat}_i(n)$ the respective unsatisfiable variants, obtained by adding to each TBox the axiom "$A_i \sqcap B_i \sqsubseteq \bot$". This latter axiom, in fact, conflicts with the ith (last) axiom of the TBox at the deepest nesting level i. We run these test cases with $i = 1, \ldots, 20$ and $n = 5, 50$, in the same system configuration exposed in the first part of Section 6.7.

In Figure 6.15 we expose all the plots concerning the experimental results for the $\texttt{nested\_restr\_sat}_i(5)$ benchmark. From top-left to bottom-right we respectively plot: the performance comparison among $\mathcal{ALCQ}$2SMT+MathSat and the other considered reasoners, the encoding time taken by $\mathcal{ALCQ}$2SMT alone, the numbers of variables and the number of clauses resulting in the encoded problems. In the three latter plots we include only a few problems, in fact the basic and the `S.P.` variants of $\mathcal{ALCQ}$2SMT both exceed the limit of `1 GB` file size for all the test cases with $i \geq 5$ and $i \geq 7$, respectively. In fact, nested restrictions exponentially affects the size of our encoding.

From Figure 6.15 we notice a couple of facts. First, as predicted, from the last three plots we can see how smart partitioning drastically reduces also the number of cost variables, if number restrictions acts at more than one nesting depth. In the examined case, the two variants of $\mathcal{ALCQ}$2SMT exponentially differs each other both in the numbers of clauses and in the number of Boolean/cost variables (and, consequently, in the CPU times required during the encoding phase). In a few test cases the gap between the two

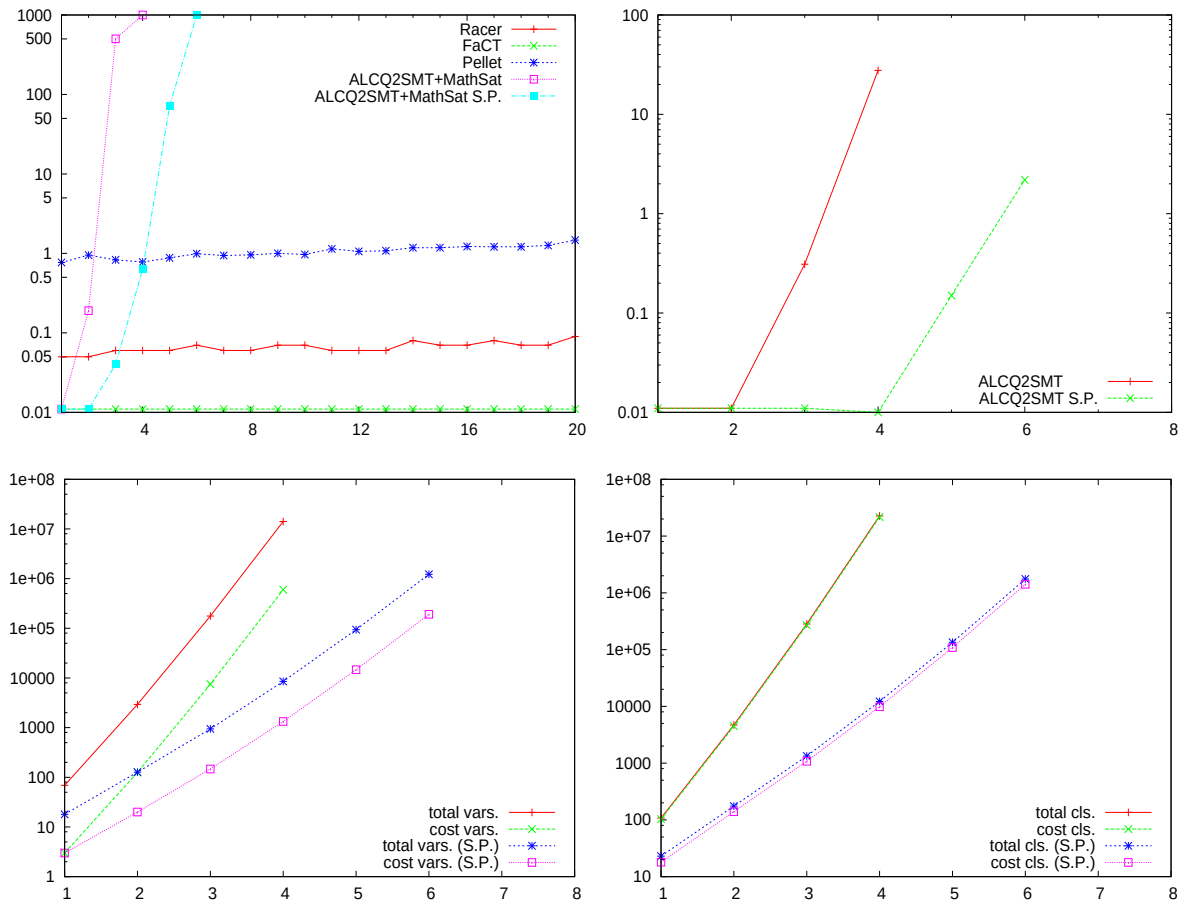


Figure 6.15: `nested_restr_sat`$_i(5)$. Top-left: comparison against other tools; Top-right: $\mathcal{ALCQ}$2SMT times; Bottom-left: # enc. variables; Bottom-right: # enc. clauses. X axis: test case index; Y axis: 1st row: CPU time (sec); 2nd row: #variables/clauses.

variants increases up to three orders of magnitude. Second, from the first plot, we notice that the performance of two variants of $\mathcal{ALCQ}$2SMT+MathSat are way far from the performances of the other systems, solving only the first 3 and, respectively, 5 test cases within the 1000 sec. timeout.

However, in order to confirm that the scalability issue is very controversial, in Figure 6.16, from left to right, we report the tools comparison in other two slightly different test cases: `nested_restr_sat`$_i(50)$ and `nested_restr_unsat`$_i(5)$, respectively. From the first plot it can be noticed how the performances of Pellet and FaCT++ gradually and significantly deteriorate having increased n (while our `S.P.` variants have identical performances). From the second plot it can be noticed that $\mathcal{ALCQ}$2SMT+MathSat `S.P.` dominates the other tools, except for Racer, when we pass from `nested_restr_sat`$_i(5)$ to the unsatisfiable cases `nested_restr_unsat`$_i(5)$. Notice that the only difference between the two benchmarks consists in one simple axiom, acting exclusively at the deepest

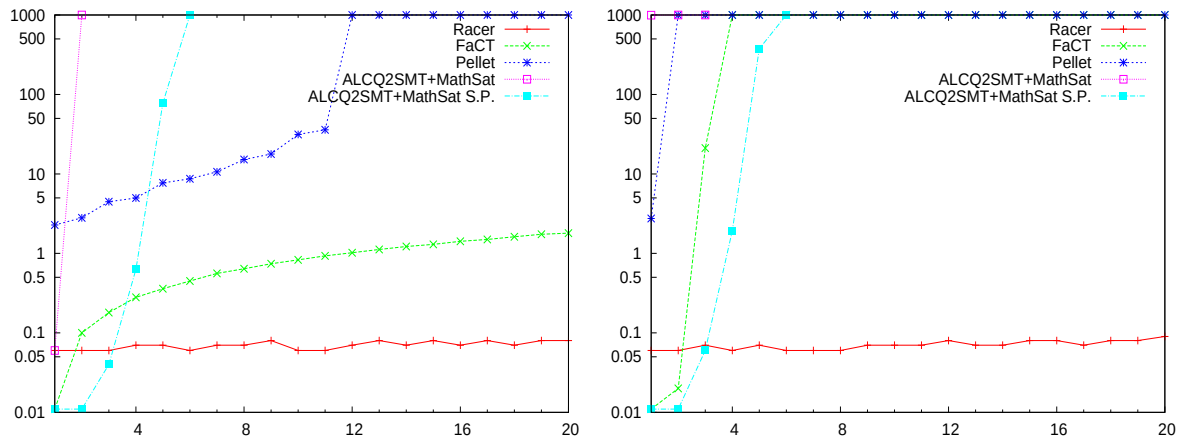


Figure 6.16: Left: $\texttt{nested\_restr\_sat}_i(50)$; Right: $\texttt{nested\_restr\_unsat}_i(5)$. X axis: test case index; Y axis: CPU time (sec).

level of nesting. Nevertheless, this is enough to transform the original problem into a non-trivial one and inhibit specific optimization techniques. In this latter benchmark PELLET solves only 1 problem (in this benchmark PELLET has given unsound results incorrectly returning "sat" in some of the problems with $i > 1$) against the 3 of FACT++ and the 5 of $\mathcal{ALCQ}$2SMT+MATHSAT `S.P.`.

This analysis shows how the scalability of the different tools can not be precisely evaluated only on the basis of the nesting depth of qualified number restrictions, but should take into account all the possible combinations and interactions of all the sources of complexity (including the number of qualified number restrictions, the values occurring in them, and so on and so forth). E.g. RACER could have scalability problems if a relatively low nesting depth combines with a high number of restrictions. Wrt. real-world problems, in which the different sources of complexity should be somehow balanced and not degenerating, we think that the overall performance of $\mathcal{ALCQ}$2SMT+MATHSAT `S.P.` in the distinct problems are promising for a good scalability on real ontologies. $\mathcal{ALCQ}$2SMT+MATHSAT `S.P.`, in fact, have proved of being able to handle a big number of qualified number restrictions, of being able to solve both satisfiable and unsatisfiable problems up to five "full" nesting levels of number restrictions and, mainly, to be indipendent from the order of magnitude of the values occurring in number restrictions. We remark that this is only a preliminary analysis since we do not dispose of real practical problems.

6.8 Contributions

In this part of work we propose a new approach to solve concept satisfiability in the Description Logic $\mathcal{ALCQ}$ wrt. acyclic TBoxes. We encode such a problem into SMT modulo the Theory of Costs, then we further develop an optimization called *smart partitioning*, which reduces the size and the hardness of the encoded problems.

We implemented our novel approach called $\mathcal{ALCQ}2SMT_{\mathcal{C}}$, into a tool $\mathcal{ALCQ}$2SMT that we run in combination of the SMT solver MathSat. In an extensive empirical test session performed on synthesized concept satisfiability problems (stressing different sources of complexity in reasoning), we evaluate the performance of our approach against the other state-of-the-art reasoners FaCT++, Pellet and Racer.

The best version of our approach $\mathcal{ALCQ}$2SMT+MathSat `S.P.` (including smart partitioning), have shown to perform extremely well on benchmarks presenting multiple/balanced sources of complexity, that we think well-fit the requirements of possible real-world ontologies. In particular, we have noticed that the size of the encoding is not the main complexity issue for our approach, which has shown to be very effective also on large or really complex problems. For instance, MathSat scales up to solving encoded problems with more than 10^5 clauses and Boolean variables, and more than 10^4 cost variables, resulting from very hard $\mathcal{ALCQ}$ problems having nested qualified number restrictions.

The smart partitioning technique is one of the more important contribution of this part of our work. Smart partitioning have turned out to be extremely effective, being able to drastically (and exponentially) reduce the size of the output SMT($\mathcal{C}$) problems, up to three orders of magnitude wrt. the basic variant of $\mathcal{ALCQ}$2SMT in the more challenging test cases (exponentially impacting also in the number of cost variables in case of nested number restrictions). We remark that this technique makes our approach independent from the magnitude/offset of the values occurring in qualified number restrictions and, indeed, it can handle very efficiently really large numbers of coexisting qualified number restrictions, when the values occurring in the restrictions repeat frequently (which is a circumstance that, realistically, could often happen in real-world problems). Furthermore, smart partitioning has experimentally proved to not impact in the cost of the encoding phase, despite its worst-case complexity. We think that this technique should be further investigated in order to conceive possible refinements, and also to be used in order to improve the actual tableaux-based state-of-the-art approaches.

From a different perspective, notice at last that the total time of $\mathcal{ALCQ}$2SMT (in particular) and MathSat often results negligible without any measurable overheads. Thus, our approach can be profitably fit into integrated approaches like the one proposed by Gasse and Haarslev (2009). Last but not least, we improved the benchmarking approach defined by Farsiniamarj and Haarslev (2010), widening the parametric set-up and defining two new groups of benchmark problems stressing on different complexity sources.

In future works we aim at directly compare with the hybrid approach of Faddoul and Haarslev (2010) (that currently is only a prototype not yet publicly available), and hopefully to include in our experimental evaluation real-world $\mathcal{ALCQ}$ ontologies, whether provided by the ontology-design community. We also plan to further investigate other possible encodings. In particular, it could be interesting to implement also the SMT($\mathcal{LA}(\mathbb{Z})$) encoding mimic the hybrid approach mentioned in Section 6.3, in order to directly compare the benefits of both the possible encodings (see Section 6.3). This could lead to devise

a combined encoding which, on the bases of the specific properties of the input ontology concerning qualified number restrictions (e.g., the number of restrictions and the nature of the values occurring in them), can chose the most suitable encoding. This approach could be pushed further by alternating the applied encoding individual-by-individual, on the basis of the local number restrictions instances (leading to a SMT($\mathcal{C} \cup \mathcal{LA}(\mathbb{Z})$) encoding). We also plan to extend our approach to general TBoxes, and to investigate its extensions to other logical constructors like role hierarchies or ABox reasoning.

6.9 Appendix: Soundness and Completeness of $\mathcal{ALCQ}2SMT_{\mathcal{C}}$

Theorem 6. *An $\mathcal{ALCQ}$ acyclic TBox $\mathcal{T}$ in normal form is consistent if and only if the SMT($\mathcal{C}$)-formula $\varphi^{\mathcal{T}}$ of $\mathcal{ALCQ}2SMT_{\mathcal{C}}(\mathcal{T})$ (Definition 5) is satisfiable.*

Proof. It is a direct consequence of the following Lemmas 10 and 12. □

Lemma 10 (Soundness)**.** *Given an $\mathcal{ALCQ}$ acyclic TBox $\mathcal{T}$ in normal form and the encoding $\mathcal{ALCQ}2SMT_{\mathcal{C}}(\mathcal{T}) = \langle \Sigma^{\mathcal{T}}, \mathcal{I}^{\mathcal{T}}_{-}, \mathcal{I}^{\mathcal{T}}_{+}, A_{\langle\ ,\ \rangle}, \mathsf{indiv}, \varphi^{\mathcal{T}} \rangle$ of Defintion 5, if the SMT($\mathcal{C}$)-formula $\varphi^{\mathcal{T}}$ is satisfiable then $\mathcal{T}$ is consistent.*

Proof. Let μ by a total truth assignment satisfying $\varphi^{\mathcal{T}}$, where we represent with $L_{\langle \sigma,\ C\rangle} \in \mu$ the fact that the literal $L_{\langle \sigma,\ C\rangle}$ is assigned true in μ. Notice that, since $\varphi^{\mathcal{T}}$ is an SMT($\mathcal{C}$)-formula μ assigns truth value also to the $\mathcal{C}$-literals (BC- and IC-literals) of $\varphi^{\mathcal{T}}$. We must prove that it also exists a model for $\mathcal{T}$.

From μ, we define $\mathcal{I}_\mu$ being the following interpretation:

$$\Delta^{\mathcal{I}_\mu} \stackrel{\text{def}}{=} \{\ \sigma \mid A_{\langle \sigma,\ \top\rangle} \text{ occurs true in } \mu\ \}, \tag{6.23}$$

$$\hat{C}^{\mathcal{I}_\mu} \stackrel{\text{def}}{=} \{\ \sigma \mid \sigma \in \Delta^{\mathcal{I}_\mu} \text{ and } L_{\langle \sigma,\ \hat{C}\rangle} \text{ occurs true in } \mu\ \}, \tag{6.24}$$

$$r^{\mathcal{I}_\mu} \stackrel{\text{def}}{=} \{\ (\sigma, \sigma.r.i) \mid \sigma, \sigma.r.i \in \Delta^{\mathcal{I}_\mu}\ \}, \tag{6.25}$$

for every normal concept $\hat{C}$ and every role r in $\mathcal{T}$. In particular, by construction, it follows for every σ:

$$\sigma \in \hat{C}^{\mathcal{I}_\mu} \quad \text{if and only if} \quad L_{\langle \sigma,\ \hat{C}\rangle} \in \mu \ \text{ and } \ A_{\langle \sigma,\ \top\rangle} \in \mu. \tag{6.26}$$

For non-normal concepts we define $\mathcal{I}_\mu$ such that:

$$(\sqcap_i\ C_i)^{\mathcal{I}_\mu} \stackrel{\text{def}}{=} \{\ \sigma \mid \sigma \in \Delta^{\mathcal{I}_\mu} \text{ and } \mu \text{ satisfies } \wedge_i\ L_{\langle \sigma,\ C_i\rangle}\ \}, \tag{6.27}$$

$$(\sqcup_j D_j)^{\mathcal{I}_\mu} \stackrel{\text{def}}{=} \{\ \sigma \mid \sigma \in \Delta^{\mathcal{I}_\mu} \text{ and } \mu \text{ satisfies } \vee_j\ L_{\langle \sigma,\ D_j\rangle}\ \}. \tag{6.28}$$

Notice that in normal form only concept description in NNF are considered. Thus in $\varphi^{\mathcal{T}}$ the literal $L_{\langle \sigma,\ \hat{C}\rangle}$ always corresponds to the positive Boolean variables $A_{\langle \sigma,\ \hat{C}\rangle}$, but for a basic concepts which can corresponds either to $A_{\langle \sigma,\ C\rangle}$ or to $\neg A_{\langle \sigma,\ C\rangle}$ for some concept name C.

We prove by induction on the structure of $\mathcal{T}$ that $\mathcal{I}_\mu$ is semantically consistent and that it is a model for $\mathcal{T}$. With this purpose, for every axiom $\hat{C} \sqsubseteq \hat{D} \in \mathcal{T}$ in normal form and every individual σ we must prove that $\mathcal{I}_\mu$ satisfies the following conditions:

(a) if σ respect the semantic of $\hat{C}$ then $\sigma \in \hat{C}^{\mathcal{I}_\mu}$;

(b) if $\sigma \in \hat{C}^{\mathcal{I}_\mu}$ then $\sigma \in \hat{D}^{\mathcal{I}_\mu}$ (i.e. $\hat{C}^{\mathcal{I}_\mu} \subseteq \hat{C}^{\mathcal{I}_\mu}$, respecting the semantic of the axiom $\hat{C} \sqsubseteq \hat{D}$);

(c) if $\sigma \in \hat{D}^{\mathcal{I}_\mu}$ then σ respect the semantic of $\hat{D}$.

When we talk about the semantic of concepts and axioms we always refer to Table 3.3. Notice that, if $\mathcal{T}$ is empty $\varphi^{\mathcal{T}}$ is the unit clause $A_{\langle 1,\ \top\rangle}$ and, thus, there is only one possible truth assignment $\mu = \{A_{\langle 1,\ \top\rangle}\}$. The interpretation $\mathcal{I}_\mu$, which is made of the non empty domain $\Delta^{\mathcal{I}_\mu} = \{1\}$, trivially satisfies $\mathcal{T}$.

(a) Let's first prove the condition (a). We prove it by induction on the structure of the concept $\hat{C}$

Base. The base cases when $\hat{C}$ is a basic concept: $\top, \bot$ or the concept name C, are trivially satisfied by, respectively, (6.23) and (6.24) of the definition of $\mathcal{I}_\mu$ (remember that $A_{\langle\sigma,\ \bot\rangle}$ is assumed to be $\bot$ for every σ).

Inductive Step. Now we prove the claim for every possible kind of non-basic concept $\hat{C}$ allowed from the axiom normal form of Section 6.4. By hypothesis the generic individual σ respects the semantic of $\hat{C}$ reported in the right-most column of Table 3.3.

$\neg C$: By hypothesis we have $\sigma \in \Delta^{\mathcal{I}_\mu} \setminus C^{\mathcal{I}_\mu}$. By construction of $\mathcal{I}_\mu$ (6.24), for every concept name C, $(\neg C)^{\mathcal{I}_\mu} = \{\sigma \mid \sigma \in \Delta^{\mathcal{I}_\mu}$ and $\neg A_{\langle\sigma,\ C\rangle}$ occurs true in $\mu\}$, that is $(\neg C)^{\mathcal{I}_\mu} = \{\sigma \mid \sigma \in \Delta^{\mathcal{I}_\mu}$ and $A_{\langle\sigma,\ C\rangle}$ occurs false in $\mu\}$. Since, instead, $C^{\mathcal{I}_\mu} = \{\sigma \mid \sigma \in \Delta^{\mathcal{I}_\mu}$ and $A_{\langle\sigma,\ C\rangle}$ occurs true in $\mu\}$, then $C^{\mathcal{I}_\mu} \cap (\neg C)^{\mathcal{I}_\mu} = \emptyset$, $C^{\mathcal{I}_\mu} \cup (\neg C)^{\mathcal{I}_\mu} = \Delta^{\mathcal{I}_\mu}$ and, thus, $\Delta^{\mathcal{I}_\mu} \setminus C^{\mathcal{I}_\mu} = (\neg C)^{\mathcal{I}_\mu}$. It follows $\sigma \in (\neg C)^{\mathcal{I}_\mu}$.

$C_1 \sqcap C_2$: By hypothesis we have $\sigma \in C_1^{\mathcal{I}_\mu} \cap C_2^{\mathcal{I}_\mu}$. So, since both $\sigma \in C_1^{\mathcal{I}_\mu}$ and $\sigma \in C_2^{\mathcal{I}_\mu}$ then, by (6.26), we have that the literals $L_{\langle\sigma,\ C_1\rangle}$ and $L_{\langle\sigma,\ C_2\rangle}$ are in $\varphi^{\mathcal{T}}$ and they are both, as well as $A_{\langle\sigma,\ \top\rangle}$, assigned to true in μ. It follows $\sigma \in (C_1 \sqcap C_2)^{\mathcal{I}_\mu}$ by definition of $\mathcal{I}_\mu$ (6.27).

We prove the other following three cases under the hypothesis that: $\langle\sigma, \Re r.C\rangle \in \mathcal{I}_-^{\mathcal{T}}$, with $\Re \in \{\geq n, \leq m, \forall\}$, assuming that point 4. of Definition 5 applies. This because in our encoding we only consider acyclic TBoxes.

$\geq nr.C$: By hypothesis it there exist a set of individuals $F_{\sigma,r,C} = \{\sigma.r.j \mid \sigma.r.j \in \Delta^{\mathcal{I}_\mu}, \sigma.r.j \in C^{\mathcal{I}_\mu}$ and $(\sigma, \sigma.r.j) \in r^{\mathcal{I}_\mu}\}$, which has at least cardinality n. Suppose, wlog., that $F_{\sigma,r,C} = \{\sigma.r.1, \ldots, \sigma.r.n\}$. From the hypothesis and by definition of $\mathcal{I}_\mu$ (6.26) it follows $L_{\langle\sigma.r.j,\ C\rangle}, A_{\langle\sigma.r.j,\ \top\rangle} \in \mu$ for every $j = 1, \ldots, n$. From $\langle\sigma, \geq nr.C\rangle \in \mathcal{I}_-^{\mathcal{T}}$ it follows (point 6.) that $\varphi^{\mathcal{T}}$ contains the clause $((\neg \mathsf{BC}(\mathsf{indiv}_{\sigma.r}^C, n-1) \wedge A_{\langle\sigma,\ \top\rangle}) \rightarrow A_{\langle\sigma,\ \geq nr.C\rangle})$ of type (6.7) and (point 8.) at least the n distinct implications $((L_{\langle\sigma.r.j,\ C\rangle} \wedge A_{\langle\sigma.r.j,\ \top\rangle}) \rightarrow \mathsf{IC}(\mathsf{indiv}_{\sigma.r}^C, 1, j))$ of type (6.10) for all the distinct $\sigma.r.j$. Thus the variable $\mathsf{indiv}_{\sigma.r}^C$ has at least cost n so that the $A_{\langle\sigma,\ \geq nr.C\rangle}$ must be assigned to true. It follows by definition of $\mathcal{I}_\mu$ that $\sigma \in (\geq nr.C)^{\mathcal{I}_\mu}$.

$\leq mr.C$: By hypothesis, since σ respect the semantic of $\leq mr.C$, the set of individuals $\{\sigma.r.j \mid \sigma.r.j \in \Delta^{\mathcal{I}_\mu}, \sigma.r.j \in C^{\mathcal{I}_\mu}$ and $(\sigma, \sigma.r.j) \in r^{\mathcal{I}_\mu}\}$, has a cardinality not greater than m. Thus no more than m literals in the forms $L_{\langle\sigma.r.j,\ C\rangle}$ can be assigned to true in μ. Since we assume $\langle\sigma, \leq mr.C\rangle \in \mathcal{I}_-^{\mathcal{T}}$, the formula $\varphi^{\mathcal{T}}$ (point 9.) contains

the clause $((\mathsf{BC}(\mathsf{indiv}^C_{\sigma.r}, m) \wedge A_{\langle\sigma,\ \top\rangle}) \rightarrow A_{\langle\sigma,\ \leq mr.C\rangle})$ of type (6.12) and (point 5.) the $m+1$ distinct implications $(\mathsf{IC}(\mathsf{indiv}^C_{\sigma.r}, 1, k^C_i) \rightarrow L_{\langle\sigma.r.k^C_i,\ C\rangle})$ of type (6.4), for all the distinct $\sigma.r.k^C_i$, with $i = 1, \ldots, m+1$. Thus, in $\varphi^{\mathcal{T}}$, more than m clause of type (6.4) exist. Suppose by contradiction that the value of the cost variable $\mathsf{indiv}^C_{\sigma.r}$ is greater than m. Thus more than m distinct $\mathcal{C}$-literals $\mathsf{IC}(\mathsf{indiv}^C_{\sigma.r}, 1, j)$ must be assigned to true in μ. But if these IC-literals are those occuring in clauses of type (6.4), like those above mentioned of index k^C_i, $i = 1, \ldots, m+1$, we get a contradiction, 'cause more than m distinct literals $L_{\langle\sigma.r.j,\ C\rangle}$ should be assigned to true in μ in order to satisfy those clauses. Notice that IC-literals may occur in clauses of type (6.10), introduced for right-hand side at-most restrictions (or left-hand side at-least ones). However these clauses are introduced only for the individuals $\sigma.r.j$ already in $\Sigma^{\mathcal{T}}$; thus, if other individuals $\sigma.r.j$ different from the $\sigma.r.k^C_i$ ones are in $\Sigma^{\mathcal{T}}$, then there are the conditions of point 7 of Definition 5, forcing the sharing of individuals and the introduction in $\varphi^{\mathcal{T}}$ of all the implications (6.8) and (6.9) for every $\sigma.r.j$. Those clauses forces the assignment to true of every literal $L_{\langle\sigma.r.j,\ C\rangle}$ such that $\mathsf{IC}(\mathsf{indiv}^C_{\sigma.r}, 1, j)$ is assigned to true, contradicting the fact that at most m of those individuals can be assigned to true. Hence, $\mathsf{indiv}^C_{\sigma.r}$ must have a value not greater than m. It follows from the clause $((\mathsf{BC}(\mathsf{indiv}^C_{\sigma.r}, m) \wedge A_{\langle\sigma,\ \top\rangle}) \rightarrow A_{\langle\sigma,\ \leq mr.C\rangle})$ (6.12) that $A_{\langle\sigma,\ \leq mr.C\rangle}$ must be assigned to true, and thus, by definition of $\mathcal{I}_\mu$, that $\sigma \in (\leq mr.C)^{\mathcal{I}_\mu}$.

$\forall r.C$: By hypothesis, since σ respect the semantic of $\forall r.C$, the set of individuals $F_{\sigma,r} = \{\sigma.r.j \mid \sigma.r.j \in \Delta^{\mathcal{I}_\mu}, (\sigma, \sigma.r.j) \in r^{\mathcal{I}_\mu}\}$ is such that $F_{\sigma,r} \subseteq C^{\mathcal{I}_\mu}$, i.e. for every $\sigma.r.j \in \Delta^{\mathcal{I}_\mu}$ it holds $\sigma.r.j \in C^{\mathcal{I}_\mu}$ and, thus, $\sigma.r.j \notin (\neg C)^{\mathcal{I}_\mu}$. Thus there can not exist literals $L_{\langle\sigma.r.j,\ \neg C\rangle}$ assigned to true in μ. Since we assume $\langle\sigma, \forall r.C\rangle \in \mathcal{I}^{\mathcal{T}}_-$, the formula $\varphi^{\mathcal{T}}$ (point 10.) contains the clause $((\mathsf{BC}(\mathsf{indiv}^{\neg C}_{\sigma.r}, 0) \wedge A_{\langle\sigma,\ \top\rangle}) \rightarrow A_{\langle\sigma,\ \forall r.C\rangle})$ of type (6.14) and (point 5.) the single implication $(\mathsf{IC}(\mathsf{indiv}^{\neg C}_{\sigma.r}, 1, k^{\neg C}_1) \rightarrow L_{\langle\sigma.r.k^{\neg C}_i,\ \neg C\rangle})$ of type (6.4). Since a left-hand side $\forall r.C$ behaves like a left-hand side $\leq mr.C$, we can use the same arguments of the previous point in the proof in order to prove that $\mathsf{indiv}^{\neg C}_{\sigma.r}$ must have value 0; In fact, otherwise, we could get a contradiction with the fact (by hypothesis) that there cannot exist true literals $L_{\langle\sigma.r.j,\ \neg C\rangle}$ in $< mu$. Thus, it follows from the clause $((\mathsf{BC}(\mathsf{indiv}^{\neg C}_{\sigma.r}, 0) \wedge A_{\langle\sigma,\ \top\rangle}) \rightarrow A_{\langle\sigma,\ \forall r.C\rangle})$ (6.14) that $A_{\langle\sigma,\ \forall r.C\rangle}$ must be assigned to true, and thus, by definition of $\mathcal{I}_\mu$, that $\sigma \in (\forall r.C)^{\mathcal{I}_\mu}$.

(b) The condition (b) trivially follows from point 6.4. of Definition 5. Let us consider the following cases of axioms: (i) $\hat{C} \sqsubseteq \hat{D}$, (ii) $\sqcap_i C_j \sqsubseteq D$ and (iii) $C \sqsubseteq \sqcup_j D_j$, with $\hat{C}, \hat{D}$ normal concepts and C, C_i, D, D_j basic concepts. Any axiom of $\mathcal{T}$ in normal form is a sub-case of one among (i), (ii) and (iii). We already proved in (a) that the definition of I_μ is consistent wrt. the semantic of every left-hand side concept.

(i) By hypothesis we have $\sigma \in \hat{C}^{\mathcal{I}_\mu}$ and, thus, $L_{\langle\sigma,\ \hat{C}\rangle} \in \mu$ by definition of $\mathcal{I}_\mu$ (6.26). Since $L_{\langle\sigma,\ \hat{C}\rangle}$ is in $\varphi^{\mathcal{T}}$, then (point 6.4.) $\varphi^{\mathcal{T}}$ contains the clause $(L_{\langle\sigma,\ \hat{C}\rangle} \rightarrow L_{\langle\sigma,\ \hat{D}\rangle})$ of type (6.3). It follows $L_{\langle\sigma,\ \hat{D}\rangle} \in \mu$ because $\varphi^{\mathcal{T}}$ is satisfiable and, thus, $\sigma \in \hat{D}^{\mathcal{I}_\mu}$, by definition of $\mathcal{I}_\mu$ (6.24).

(ii) Similarly, by hypothesis we have $\sigma(\sqcap_i C_i)^{\mathcal{I}_\mu}$ and, thus, that $\bigwedge_i L_{\langle \sigma,\ C_i\rangle}$ is satisfied by μ, by definition of $\mathcal{I}_\mu$ (6.27). Since every $L_{\langle \sigma,\ C_i\rangle}$ is in $\varphi^{\mathcal{T}}$, then (point 6.4.) $\varphi^{\mathcal{T}}$ contains the clause $((\bigwedge_i L_{\langle \sigma,\ C_i\rangle}) \rightarrow L_{\langle \sigma,\ D\rangle})$ of type (6.3). It follows $L_{\langle \sigma,\ D\rangle} \in \mu$ because $\varphi^{\mathcal{T}}$ is satisfiable and, thus, $\sigma \in D^{\mathcal{I}_\mu}$ by (6.24).

(iii) In the last case, if $\sigma \in C^{\mathcal{I}_\mu}$ by hypothesis, then $L_{\langle \sigma,\ C\rangle} \in \mu$ by definition of $\mathcal{I}_\mu$ (6.26). Since $L_{\langle \sigma,\ C\rangle}$ is in $\varphi^{\mathcal{T}}$, then (point 6.4.) $\varphi^{\mathcal{T}}$ contains the clause $(L_{\langle \sigma,\ C\rangle} \rightarrow (\bigvee_j L_{\langle \sigma,\ D_j\rangle}))$ of type (6.3). Since $\varphi^{\mathcal{T}}$ is satisfiable it follows that $\bigvee_j L_{\langle \sigma,\ D_j\rangle}$ must be satisfied by μ and, thus, that $\sigma \in (\sqcup_j D_j)^{\mathcal{I}_\mu}$ by definition of $\mathcal{I}_\mu$ (6.28).

(c) Finally, let's prove by induction on the structure of the concept $\hat{D}$ that if $\sigma \in \hat{D}^{\mathcal{I}_\mu}$ then σ respect the semantic of $\hat{D}$

Base. When $\hat{D}$ is a basic concept: $\top, \bot$ or the concept name D, the claim is trivially satisfied by the definition of $\mathcal{I}_\mu$, similarly to (a).

Inductive Step. We prove the claim for every possible kind of non-basic concept $\hat{D}$ considered in the normal form of Section 6.4. By hypothesis the $\sigma \in \hat{D}^{\mathcal{I}_\mu}$.

$\neg D$: Let $\sigma \in (\neg D)^{\mathcal{I}_\mu}$. By definition (6.28) $\sigma \in \Delta^{\mathcal{I}_\mu}$ and $L_{\langle \sigma,\ \neg D\rangle}$ is true in μ, i.e. $\neg L_{\langle \sigma,\ D\rangle} \in \mu$ It follows $\sigma \notin D^{\mathcal{I}_\mu}$ and, thus, $\sigma \in \Delta^{\mathcal{I}_\mu} \setminus D^{\mathcal{I}_\mu}$.

$D_1 \sqcup D_2$: Let $\sigma \in (D_1 \sqcup D_2)^{\mathcal{I}_\mu}$. By (6.28), $A_{\langle \sigma,\ D_1\rangle} \vee A_{\langle \sigma,\ D_2\rangle}$ is satisfied by μ and $\sigma \in \Delta^{\mathcal{I}_\mu}$, thus $A_{\langle \sigma,\ \top\rangle} \in \mu$. It follows that at least one of the literals $L_{\langle \sigma,\ D_1\rangle}$ and $L_{\langle \sigma,\ D_2\rangle}$ is true in μ. Hence, since we already have $\sigma \in \Delta^{\mathcal{I}_\mu}$, by (6.26) either $\sigma \in D_1^{\mathcal{I}_\mu}$ or $\sigma \in D_2^{\mathcal{I}_\mu}$, which let us to conclude that $\sigma \in D_1^{\mathcal{I}_\mu} \cup D_2^{\mathcal{I}_\mu}$, i.e. $\sigma \in (D_1 \sqcup D_2)^{\mathcal{I}_\mu}$.

$\geq nr.D$: Let $\sigma \in (\geq nr.D)^{\mathcal{I}_\mu}$ by hypothesis, then we must prove that there exist at least n distinct individuals $\sigma.r.j \in \Delta^{\mathcal{I}_\mu}$ such that $(\sigma, \sigma.r.j) \in r^{\mathcal{I}_\mu}$ and $\sigma.r.j \in D^{\mathcal{I}_\mu}$. By definition of $\mathcal{I}_\mu$ (6.26) we have $L_{\langle \sigma,\ \geq nr.D\rangle}, A_{\langle \sigma,\ \top\rangle} \in \mu$. Further, since $\geq nr.D$ occurs at the right-hand side of the axiom so that $\langle \sigma, \geq nr.D\rangle \in \mathcal{I}_+^{\mathcal{I}}$, $\varphi^{\mathcal{T}}$ contains the clauses (6.4), (6.5) [resp. (6.8), (6.9)] and (6.6) due to the application wrt. σ and $\geq nr.D$ of the points 5. [resp. 7.] and 6., respectively, of Definition 5. Due to the clause (6.6): $(A_{\langle \sigma,\ \geq nr.D\rangle} \wedge A_{\langle \sigma,\ \top\rangle}) \rightarrow \neg \mathsf{BC}(\mathsf{indiv}^D_{\sigma.r}, n-1)$ the SMT($\mathcal{C}$) formula $\varphi^{\mathcal{T}}$ can be satisfiable only if the $\mathcal{C}$-literal $\mathsf{BC}(\mathsf{indiv}^D_{\sigma.r}, n-1)$ is assigned to false, that is only if at least n different incur-cost literals $\mathsf{IC}(\mathsf{indiv}^D_{\sigma.r}, 1, \ldots)$ are assigned true in μ (because all the IC-literals in $\varphi^{\mathcal{T}}$ have cost 1). Notice that these IC-literals certainly belong to clauses of type (6.4) [resp. (6.8)]. In fact, they can belong also to clauses of type (6.10), but those clauses either refer to individuals $\sigma.r.k_i^D$ introduced because of $\geq nr.D$ or to different individuals $\sigma.r.k_j^E$. In this second case, the coexistence of more than one at-least restriction and one at-most forces the sharing of the individuals, causes the sharing of individuals due tu the point 7. of Definition 5. By consequence every IC-literal of the type $\mathsf{IC}(\mathsf{indiv}^D_{\sigma.r}, 1, \ldots)$ occur also as left-hand side literal in the

implications of type (6.8). By these implications, it follows that at least n of the correspondent literals $L_{\langle\sigma.r.j,\ D\rangle}$ must be assigned true in μ. Further, it follows from the implications (6.5) [resp. (6.9)] that also at least n literals of the form $A_{\langle\sigma.r.j,\ \top\rangle}$ are assigned to true in μ. Hence, by (6.26), we have at least n distinct individuals $\sigma.r.j$ such that $\sigma.r.j \in \Delta^{\mathcal{I}_\mu}$, $\sigma.r.j \in D^{\mathcal{I}_\mu}$ and such that $(\sigma, \sigma.r.j) \in r^{\mathcal{I}_\mu}$ by construction of $r^{\mathcal{I}_\mu}$ (6.25).

$\leq mr.D$: Let $\sigma \in (\leq mr.D)^{\mathcal{I}_\mu}$ by hypothesis, then we must prove that there exist at most m distinct individuals $\sigma.r.j \in \Delta^{\mathcal{I}_\mu}$ such that $(\sigma, \sigma.r.j) \in r^{\mathcal{I}_\mu}$ and $\sigma.r.j \in D^{\mathcal{I}_\mu}$. By definition of $\mathcal{I}_\mu$ (6.26) we have $L_{\langle\sigma,\ \leq nr.D\rangle}, A_{\langle\sigma,\ \top\rangle} \in \mu$. Further, since $\leq nr.D$ occurs at the right-hand side of the axiom so that $\langle\sigma, \leq nr.D\rangle \in \mathcal{I}^{\mathcal{T}}_{+}$, $\varphi^{\mathcal{T}}$ contains all the clauses of type (6.10) and the clause (6.11) due to the application wrt. σ and $\leq mr.D$ of the points 8. and 6., respectively, of Definition 5. Due to the clause (6.11) $(A_{\langle\sigma,\ \leq mr.D\rangle} \wedge A_{\langle\sigma,\ \top\rangle}) \rightarrow \mathsf{BC}(\mathsf{indiv}^D_{\sigma.r}, m)$, the SMT($\mathcal{C}$) formula $\varphi^{\mathcal{T}}$ is satisfied if and only if at most m different incur-cost literals $\mathrm{IC}\mathsf{indiv}^D_{\sigma.r}1\ldots$ are assigned to true in μ. Let us suppose by contradiction that even if $\sigma \in (\leq mr.D)^{\mathcal{I}_\mu}$ there exist more than m distinct individuals $\sigma.r.j \in D^{\mathcal{I}_\mu}$. Then, by (6.26), it follows that there are more than m different literals $L_{\langle\sigma.r.j,\ D\rangle}$ and more than m different literals $A_{\langle\sigma.r.j,\ \top\rangle}$ assigned to true in μ. Since $\varphi^{\mathcal{T}}$ includes one clause (6.10) of the type: $(L_{\langle\sigma.r.j,\ D\rangle} \wedge A_{\langle\sigma.r.j,\ \top\rangle}) \rightarrow \mathsf{IC}(\mathsf{indiv}^D_{\sigma.r}, 1, \ldots)$, for every $\sigma.r.j \in \Sigma^{\mathcal{T}}$, then it follows that more than m distinct IC-literals wrt. to $\mathsf{indiv}^D_{\sigma.r}$ must be assigned to true. But this conflicts in the Theory of Costs with the fact that $\varphi^{\mathcal{T}}$ is satisfiable and $\mathsf{indiv}^D_{\sigma.r}$ is bounded by m as previously stated (since $\mathsf{BC}(\mathsf{indiv}^D_{\sigma.r}, m)$ is true in μ). Thus we get a contradiction, proving the claim.

$\forall r.D$: Let $\sigma \in (\forall r.D)^{\mathcal{I}_\mu}$. We must prove that, for every individual $\sigma.r.j \in \Delta^{\mathcal{I}_\mu}$ such that $(\sigma, \sigma.r.j) \in r^{\mathcal{I}_\mu}$, $\sigma.r.j \in D^{\mathcal{I}_\mu}$. Since $L_{\langle\sigma,\ \forall r.D\rangle}$ is in $\varphi^{\mathcal{T}}$, $\varphi^{\mathcal{T}}$ must include also the clauses (6.13): $(A_{\langle\sigma,\ \forall r.D\rangle} \wedge A_{\langle\sigma.r.j,\ \top\rangle}) \rightarrow A_{\langle\sigma.r.j,\ D\rangle}$, for every $\sigma.r.j \in \Sigma^{\mathcal{T}}$, from point 10. of Definition 5. By definition of $\Delta^{\mathcal{I}_\mu}$ (6.23), $\sigma.r.j \in \Delta^{\mathcal{I}_\mu}$ if and only if $A_{\langle\sigma.r.j,\ \top\rangle}$ is assigned to true in μ. Thus, since, by construction (6.25), $(\sigma, \sigma.r.j) \in r^{\mathcal{I}_\mu}$ if and only if $\sigma, \sigma.r.j \in \Delta^{\mathcal{I}_\mu}$, we have $A_{\langle\sigma.r.j,\ \top\rangle} \in \mu$ for every $(\sigma, \sigma.r.j) \in r^{\mathcal{I}_\mu}$. Since $L_{\langle\sigma,\ \forall r.D\rangle} \in \mu$ by hypothesis, and $A_{\langle\sigma.r.j,\ \top\rangle} \in \mu$ for every $(\sigma, \sigma.r.j) \in r^{\mathcal{I}_\mu}$, then also $A_{\langle\sigma.r.j,\ D\rangle}$ must be assigned to true μ in order to satisfy the clauses (6.13) of $\varphi^{\mathcal{T}}$ (that is satisfiable). It follows (6.26) that $\sigma.r.j \in D^{\mathcal{I}_\mu}$ for every $(\sigma, \sigma.r.j) \in r^{\mathcal{I}_\mu}$.

□

Lemma 11. *Given an $\mathcal{ALCQ}$ acyclic TBox $\mathcal{T}$ in normal form and the encoding $\mathcal{ALCQ}2SMT_{\mathcal{C}}(\mathcal{T}) = \langle \Sigma^{\mathcal{T}}, \mathcal{I}^{\mathcal{T}}_{-}, \mathcal{I}^{\mathcal{T}}_{+}, A_{\langle\ ,\ \rangle}, \mathsf{indiv}, \varphi^{\mathcal{T}} \rangle$ of Defintion 5, if there exists a model $\mathcal{I}$ for $\mathcal{T}$ then it also exists a model $\mathcal{I}_\Sigma$ for $\mathcal{T}$, such that $\Delta^{\mathcal{I}_\Sigma} \subseteq \Sigma_{\mathcal{T}}$ and that $r^{\mathcal{I}_\Sigma} \subseteq \{(\sigma, \sigma.r.i) \mid \sigma, \sigma.r.i \in \Sigma^{\mathcal{T}}\}$, for every role $r \in \mathcal{T}$.*

Proof. We remark that here we don't discuss about the properties of $\mathcal{ALCQ}2SMT_{\mathcal{C}}$, we only show that $\Sigma^{\mathcal{T}}$ is a super set of $\Delta^{\mathcal{I}_\Sigma}$ for some model $\mathcal{I}_\Sigma$ for $\mathcal{T}$. Suppose that $\mathcal{T}$ is consistent. It is known that $\mathcal{ALCQ}$ has the finite (and) tree model property (Lutz et al., 2005). Thus suppose that $\mathcal{I}$ is a finite tree model for $\mathcal{T}$. Wlog., among the many possible finite tree model for $\mathcal{T}$, we can safely chose a model $\mathcal{I}$ for $\mathcal{T}$ such that:

(a) qualified number restrictions wrt. different roles are satisfied by distinct individuals;

(b) if a given individual x, does not belong to the interpretation of any at-most qualified number restrictions, then every different at-least qualified number restriction that x must satisfy is satisfied by mean of relations with always different (each other) individuals;

(c) every at-least number restriction $\geq nr.C$ in $\mathcal{T}$ is satisfied through the minimum possible number of relations between individuals in $\mathcal{I}$, that could be n, when possible.

This has been said we map the individuals of $\mathcal{I}$ to the individuals of $\Sigma^{\mathcal{T}}$ and we call $\mathcal{I}_\Sigma$ the model resulting from this mapping. The mapping is defined recursively as follows:

Base. The root individual of the tree model $\mathcal{I}$ is mapped to the individual $1 \in \Sigma^{\mathcal{T}}$.

Step. Given an individual $x \in \Delta^{\mathcal{I}}$ mapped to the individual $\sigma \in \Delta^{\mathcal{I}_\Sigma}$ and, thus, $\sigma \in \Sigma^{\mathcal{T}}$ by inductive hypothesis, we must provide a mapping for every "child" individual y_i of x in the tree model $\mathcal{I}$ (i.e. every individual $y_i \in \Delta^{\mathcal{I}}$ such that $(x, y_i) \in r^{\mathcal{I}}$, for some role r) to one individual of $\Sigma^{\mathcal{T}}$.
Let us consider the generic role r. Thanks to (a) for every r we can define a different mapping. If x must not satisfy any at-least number restriction in r then there are no individuals in relation with x through r in $\mathcal{I}$, due to the hypothesis (c). Thus we can distinguish the following remaining two cases:

- The individual x must satisfy only at-least number restrictions and no at-most number restrictions wrt. r. Formally, consider the case $x \in (\geq n_j r.C_j)^{\mathcal{I}}$ for some integer values n_j, some concepts C_j with $j \geq 1$, and $x \notin (\leq m r.D)^{\mathcal{I}}$ for any integer m and any concept D. Then, for every j, by the hypothesis (a), (b) and (c) there are exactly n_j distinct individuals y^j_i, with $i = 1, \ldots, n_j$, such that $y^j_i \in \Delta^{\mathcal{I}}$, $y^j_i \in (C_j)^{\mathcal{I}}$ and $(x, y^j_i) \in r^{\mathcal{I}}$. Notice that, due to the hypothesis (b), it holds $y^j_i \neq y^{j'}_{i'}$ for every $j \neq j'$ or $i \neq i'$.

  By inductive hypothesis we have $\sigma \in (\geq n_j r.C_j)^{\mathcal{I}_\Sigma}$ for every j, and, by definition of $\mathcal{ALCQ}2SMT_{\mathcal{C}}(\mathcal{T})$ (point 5.) there exist exactly n_j distinct individuals $\sigma.r.k^{C_j}_i \in \Sigma^{\mathcal{T}}$,

with $i = 1, \ldots, n_j$. For every j and for $i = 1, \ldots, n_j$ we map the individual y_i^j of $\Delta^{\mathcal{I}}$ to the respective individual $\sigma.r.k_i^{C_j}$ of $\Sigma^{\mathcal{T}}$.

- Otherwise $x \in (\geq n_j r.C_j)^{\mathcal{I}}$ and $x \in (\leq m_k r.D_k)^{\mathcal{I}}$, for some integer values n_j, m_k, and some concepts C_j, D_k, with $j, k \geq 1$. As stated above $\mathcal{I}$ is model for $\mathcal{T}$ and it complies with the hypothesis (a) and (c). So, for every j, there are exactly n_j distinct individuals y_i^j, with $i = 1, \ldots, n_j$, such that $y_i^j \in \Delta^{\mathcal{I}}$, $y_i^j \in C_j^{\mathcal{I}}$ and $(x, y_i^j) \in r^{\mathcal{I}}$ and, for every k, there are at-most m_k distinct individuals $y_{i'}^k$, with $i' = 1, \ldots, m_k$ such that $y_{i'}^k \in \Delta^{\mathcal{I}}$, $y_{i'}^k \in D_k^{\mathcal{I}}$ and $(x, y_{i'}^k) \in r^{\mathcal{I}}$. Due to at-most restrictions, for which the hypothesis (b) do not hold, notice that individuals can be shared, i.e. it is possible to have $y_i^j = y_{i'}^{j'}$ (or $y_i^k = y_{i'}^{k'}$) for some $j \neq j'$ (or $k \neq k'$) and some values of i, i'.

  For every j and k, by inductive hypothesis we have $\sigma \in (\geq n_j r.C_j)^{\mathcal{I}_\Sigma}$ and $\sigma \in (\leq m_k r.D_k)^{\mathcal{I}_\Sigma}$. By definition of $\mathcal{ALCQ}2SMT_{\mathcal{C}}(\mathcal{T})$ (point 5.) there are exactly n_j distinct individuals $\sigma.r.k_i^{C_j} \in \Sigma^{\mathcal{T}}$, with $i = 1, \ldots, n_j$, for every $\geq n_j r.C_j$, (so there are enough individuals in order to satisfy all the at-least restrictions). Notice that, in the hypothesis that both $\sigma \in (\geq n_j r.C_j)^{\mathcal{I}}$ and $\sigma \in (\leq m_k r.D_k)^{\mathcal{I}}$, $\varphi^{\mathcal{T}}$ is then extended with the clauses (6.8) and (6.9) (point 7.) for every $i = 1, \ldots, \sum_j n_j$ (allowing for sharing individuals). Thus all the individuals of $\Sigma^{\mathcal{T}}$ in the form $\sigma.r.i$, with $i = 1, \ldots, \sum_j n_j$ above mentioned are equivalently expressive to each other, and any mapping between the individuals y_h^j and these individuals of $\Sigma^{\mathcal{T}}$ is suitable, provided that it must be a function, i.e. that if $y_h^j = y_{h'}^{j'}$, for some $j \neq j'$ and some values h, h', then y_h^j and $y_{h'}^{j'}$ are mapped to the same individual of $\Sigma^{\mathcal{T}}$.

Notice that the mapping from $\mathcal{I}$ to $\mathcal{I}_\Sigma$ we shown respect the property: $r^{\mathcal{I}_\Sigma} \subseteq \{(\sigma, \sigma.r.i) \mid \sigma, \sigma.r.i \in \Sigma^{\mathcal{T}}\}$. □

Lemma 12 (Completeness)**.** *Given an $\mathcal{ALCQ}$ acyclic TBox $\mathcal{T}$ in normal form and the encoding $\mathcal{ALCQ}2SMT_{\mathcal{C}}(\mathcal{T}) = \langle \Sigma^{\mathcal{T}}, \mathcal{I}_-^{\mathcal{T}}, \mathcal{I}_+^{\mathcal{T}}, A_{\langle\ ,\ \rangle}, \mathsf{indiv}, \varphi^{\mathcal{T}} \rangle$ of Defintion 5, if $\mathcal{T}$ in is consistent then the SMT($\mathcal{C}$)-formula $\varphi^{\mathcal{T}}$ is satisfiable.*

Proof. Given that $\mathcal{T}$ is consistent, it exists a model $\mathcal{I}$ for $\mathcal{T}$ such that $\Delta^{\mathcal{I}} \subseteq \Sigma^{\mathcal{T}}$ and that $r^{\mathcal{I}_\Sigma} \subseteq \{(\sigma, \sigma.r.i) \mid \sigma, \sigma.r.i \in \Sigma^{\mathcal{T}}\}$, for every role $r \in \mathcal{T}$, as stated in Lemma 11. We built from $\mathcal{I}$ a total truth assignment μ satisfying $\varphi^{\mathcal{T}}$, as follows:

$$\mu \stackrel{\text{def}}{=} \mu_{\mathcal{I}} \cup \mu_{\overline{\mathcal{I}}} \tag{6.29}$$

$$\mu_{\mathcal{I}} \stackrel{\text{def}}{=} \mu_{\Delta} \cup \mu_X \cup \mu_{\geq} \cup \mu_{\leq} \cup \mu_{\forall} \cup \mu_{\mathsf{IC}} \tag{6.30}$$

$$\mu_{\Delta} \stackrel{\text{def}}{=} \{\ A_{\langle \sigma,\ \top\rangle} \mid A_{\langle \sigma,\ \top\rangle} \text{ literal of } \varphi^{\mathcal{T}},\ \sigma \in \Delta^{\mathcal{I}}\} \tag{6.31}$$

$$\begin{aligned}\mu_X \stackrel{\text{def}}{=}& \{\ L_{\langle \sigma,\ X\rangle} \mid L_{\langle \sigma,\ X\rangle} \text{ literal of } \varphi^{\mathcal{T}},\ \sigma \in \Delta^{\mathcal{I}} \text{ and } \sigma \in X^{\mathcal{I}}\}\\ \cup\ & \{\neg L_{\langle \sigma,\ X\rangle} \mid L_{\langle \sigma,\ X\rangle} \text{ literal of } \varphi^{\mathcal{T}},\ \sigma \in \Delta^{\mathcal{I}} \text{ and } \sigma \notin X^{\mathcal{I}}\}\end{aligned} \tag{6.32}$$

$$\begin{aligned}\mu_{\geq} \stackrel{\text{def}}{=}& \{\neg\mathsf{BC}(\mathsf{indiv}^C_{\sigma.r}, n-1) \mid \neg\mathsf{BC}(\mathsf{indiv}^C_{\sigma.r}, n-1), L_{\langle \sigma,\ \geq nr.C\rangle} \in c,\ \text{with } c \in \varphi^{\mathcal{T}},\\ & \qquad \sigma \in \Delta^{\mathcal{I}} \text{ and } \sigma \in (\geq nr.C)^{\mathcal{I}}\}\\ \cup\ & \{\ \mathsf{BC}(\mathsf{indiv}^C_{\sigma.r}, n-1) \mid \neg\mathsf{BC}(\mathsf{indiv}^C_{\sigma.r}, n-1), L_{\langle \sigma,\ \geq nr.C\rangle} \in c,\ \text{with } c \in \varphi^{\mathcal{T}},\\ & \qquad \sigma \in \Delta^{\mathcal{I}} \text{ and } \sigma \notin (\geq nr.C)^{\mathcal{I}}\}\end{aligned} \tag{6.33}$$

$$\begin{aligned}\mu_{\leq} \stackrel{\text{def}}{=}& \{\ \mathsf{BC}(\mathsf{indiv}^C_{\sigma.r}, m) \mid \mathsf{BC}(\mathsf{indiv}^C_{\sigma.r}, m), L_{\langle \sigma,\ \leq mr.C\rangle} \in c,\ \text{with } c \in \varphi^{\mathcal{T}},\\ & \qquad \sigma \in \Delta^{\mathcal{I}} \text{ and } \sigma \in (\leq mr.C)^{\mathcal{I}}\}\\ \cup\ & \{\neg\mathsf{BC}(\mathsf{indiv}^C_{\sigma.r}, m) \mid \mathsf{BC}(\mathsf{indiv}^C_{\sigma.r}, m), L_{\langle \sigma,\ \leq mr.C\rangle} \in c,\ \text{with } c \in \varphi^{\mathcal{T}},\\ & \qquad \sigma \in \Delta^{\mathcal{I}} \text{ and } \sigma \notin (\leq mr.C)^{\mathcal{I}}\}\end{aligned} \tag{6.34}$$

$$\begin{aligned}\mu_{\forall} \stackrel{\text{def}}{=}& \{\ \mathsf{BC}(\mathsf{indiv}^C_{\sigma.r}, 0) \mid \mathsf{BC}(\mathsf{indiv}^{\neg C}_{\sigma.r}, 0), L_{\langle \sigma,\ \forall r.C\rangle} \in c,\ \text{with } c \in \varphi^{\mathcal{T}},\\ & \qquad \sigma \in \Delta^{\mathcal{I}} \text{ and } \sigma \in (\forall r.C)^{\mathcal{I}}\}\\ \cup\ & \{\neg\mathsf{BC}(\mathsf{indiv}^C_{\sigma.r}, 0) \mid \mathsf{BC}(\mathsf{indiv}^{\neg C}_{\sigma.r}, 0), L_{\langle \sigma,\ \forall r.C\rangle} \in c,\ \text{with } c \in \varphi^{\mathcal{T}},\\ & \qquad \sigma \in \Delta^{\mathcal{I}} \text{ and } \sigma \notin (\forall r.C)^{\mathcal{I}}\}\end{aligned} \tag{6.35}$$

$$\begin{aligned}\mu_{\mathsf{IC}} \stackrel{\text{def}}{=}& \{\ \mathsf{IC}(\mathsf{indiv}^C_{\sigma.r}, 1, i) \mid \mathsf{IC}(\mathsf{indiv}^C_{\sigma.r}, 1, i), L_{\langle \sigma.r.i,\ C\rangle} \in c,\ \text{with } c \in \varphi^{\mathcal{T}},\\ & \qquad \sigma \in \Delta^{\mathcal{I}},\ \text{ and } \sigma.r.i \in \Delta^{\mathcal{I}} \text{ and } \sigma.r.i \in C^{\mathcal{I}}\}\\ \cup\ & \{\neg\mathsf{IC}(\mathsf{indiv}^C_{\sigma.r}, 1, i) \mid \mathsf{IC}(\mathsf{indiv}^C_{\sigma.r}, 1, i), L_{\langle \sigma.r.i,\ C\rangle} \in c,\ \text{with } c \in \varphi^{\mathcal{T}},\\ & \qquad \sigma \in \Delta^{\mathcal{I}},\ \text{ but } \sigma.r.i \notin \Delta^{\mathcal{I}} \text{ or } \sigma.r.i \notin C^{\mathcal{I}}\}\end{aligned} \tag{6.36}$$

$$\mu_{\overline{\mathcal{I}}} \stackrel{\text{def}}{=} \mu_{\overline{\Delta}} \cup \mu_{\overline{X}} \cup \mu_{\overline{\mathsf{BC}}} \cup \mu_{\overline{\mathsf{IC}}} \tag{6.37}$$

$$\mu_{\overline{\Delta}} \stackrel{\text{def}}{=} \{\neg A_{\langle \sigma,\ \top\rangle} \mid A_{\langle \sigma,\ \top\rangle} \text{ literal of } \varphi^{\mathcal{T}},\ \sigma \notin \Delta^{\mathcal{I}}\} \tag{6.38}$$

$$\mu_{\overline{\mathsf{BC}}} \stackrel{\text{def}}{=} \{\ \mathsf{BC}(\mathsf{indiv}^C_{\sigma.r}, n) \mid \mathsf{BC}(\mathsf{indiv}^C_{\sigma.r}, \ldots) \text{ literal of } \varphi^{\mathcal{T}},\ \sigma \notin \Delta^{\mathcal{I}},\ \text{for any } n\} \tag{6.39}$$

$$\mu_{\overline{\mathsf{IC}}} \stackrel{\text{def}}{=} \{\neg\mathsf{IC}(\mathsf{indiv}^C_{\sigma.r}, 1, i) \mid \mathsf{IC}(\mathsf{indiv}^C_{\sigma.r}, 1, \ldots) \text{ literal of } \varphi^{\mathcal{T}},\ \sigma \notin \Delta^{\mathcal{I}},\ \text{for any } i\} \tag{6.40}$$

where $\mu_{\overline{X}}$ is a consistent truth assignment satisfying all the clauses of type (6.3) of $\varphi^{\mathcal{T}}$ in the case $\sigma \notin \Delta^{\mathcal{I}}$.

We remark that every clause of $\varphi^{\mathcal{T}}$ is defined wrt. to a specific individual σ. By construction $\mu_{\mathcal{I}}$ and $\mu_{\overline{\mathcal{I}}}$ assign the two complementary partitions of the Boolean- and $\mathcal{C}$-literals of $\varphi^{\mathcal{T}}$, those referring to some $\sigma \in \Delta^{\mathcal{I}}$ and, respectively, those referring to some $\sigma \notin \Delta^{\mathcal{I}}$. In particular, also μ_{IC} and $\mu_{\overline{\mathsf{IC}}}$ assigns the two different partitions of IC-literals. In fact μ_{IC} assigns those literals involving cost variables $\mathsf{indiv}^C_{\sigma.r}$ referring to some $\sigma \in \Delta^{\mathcal{I}}$ (but also possibly related to non enabled individuals $\sigma.r.i \in \Sigma^{\mathcal{T}}$, but $\sigma.r.i \notin \Delta^{\mathcal{I}}$), while $\mu_{\overline{\mathsf{IC}}}$ assigns the IC-literals involving a cost variable $\mathsf{indiv}^C_{\sigma.r}$ for some $\sigma \notin \Delta^{\mathcal{I}}$. This is necessary because $\mathcal{ALCQ}2SMT_{\mathcal{C}}(\mathcal{T})$ encodes a super-set $\Sigma^{\mathcal{T}}$ of possible individuals, with the aim of include a consistent set of individuals defining a model for $\mathcal{T}$.

It is easy to see that μ is a total and consistent truth assignment for $\varphi^{\mathcal{T}}$.

First we show that μ, and in particular $\mu_{\mathcal{I}} \cup \mu_{\overline{\Delta}}$, **propositionally satisfies all the clauses of $\varphi^{\mathcal{T}}$ such that** $\sigma \in \Delta^{\mathcal{I}}$, for every type of clause from (6.3) to (6.14).

(6.3): Clauses of type (6.3) represents the propositional encoding of the concept inclusions of $\mathcal{T}$. We can distinguish three cases:

- An axiom $\hat{C} \sqsubseteq \hat{D}$, for two generic normal concepts $\hat{C}$ and $\hat{D}$, is encoded into the clause $L_{\langle \sigma,\ \hat{C}\rangle} \rightarrow L_{\langle \sigma,\ \hat{D}\rangle}$. Since $\sigma \in \Delta^{\mathcal{I}}$ and $\mathcal{I}$ is a model for $\mathcal{T}$, then it holds $\hat{C}^{\mathcal{I}} \subseteq \hat{D}^{\mathcal{I}}$. Thus, if $\sigma \in \hat{C}^{\mathcal{I}}$ then $\sigma \in \hat{D}^{\mathcal{I}}$, from which it follows, by (6.32), that either $L_{\langle \sigma,\ X\rangle}, L_{\langle \sigma,\ Y\rangle} \in \mu_X$, or $\neg L_{\langle \sigma,\ X\rangle} \in \mu_X$. In both cases the clause is satisfied.
- An axiom $C_1 \sqcap \mathcal{C}_2 \sqsubseteq D$, with C_1, C_2 and D basic concepts, is encoded into the clause $(L_{\langle \sigma,\ C_1\rangle} \wedge L_{\langle \sigma,\ C_2\rangle}) \rightarrow L_{\langle \sigma,\ D\rangle}$. Since $\sigma \in \Delta^{\mathcal{I}}$ and $\mathcal{I}$ is a model for $\mathcal{T}$ it holds $(C_1 \sqcap C_2)^{\mathcal{I}} \subseteq D^{\mathcal{I}}$. Thus if $\sigma \in C_1^{\mathcal{I}} \cap C_2^{\mathcal{I}}$ then $\sigma \in D^{\mathcal{I}}$, from which it follows, by (6.32), that either $L_{\langle \sigma,\ C_1\rangle}, L_{\langle \sigma,\ C_2\rangle}, L_{\langle \sigma,\ D\rangle} \in \mu_X$ or at least one between $\neg L_{\langle \sigma,\ C_1\rangle}$ and $\neg L_{\langle \sigma,\ C_2\rangle}$ is in μ_X. In both cases the clause is satisfied.
- An axiom $C \sqsubseteq D_1 \sqcup D_2$ with D_1, D_2 and C basic concepts, is encoded into the clause $L_{\langle \sigma,\ C\rangle} \rightarrow (L_{\langle \sigma,\ D_1\rangle} \vee L_{\langle \sigma,\ D_2\rangle})$. Since $\sigma \in \Delta^{\mathcal{I}}$ and $\mathcal{I}$ is a model for $\mathcal{T}$ it holds $C^{\mathcal{I}} \subseteq (D_1 \sqcup D_2)^{\mathcal{I}}$. Thus if $\sigma \in C^{\mathcal{I}}$ then $\sigma \in D_1^{\mathcal{I}} \cup D_2^{\mathcal{I}}$, from which it follows, by 6.32, that either $L_{\langle \sigma,\ C\rangle}$ and at least one between $L_{\langle \sigma,\ D_1\rangle}$ and $L_{\langle \sigma,\ D_2\rangle}$ is in μ_X or $\neg L_{\langle \sigma,\ C\rangle} \in \mu_X$. In both cases the clause is satisfied.

(6.4), (6.5), (6.8), (6.9): Wlog. let us consider the case in which the index of the IC-literal is i and, thus, it is associated to the individual $\sigma.r.i$, for some role r, some basic concept C, the integer values i and $\sigma \in \Delta^{\mathcal{I}}$. Thus we must show that the clauses: $\mathsf{IC}(\mathsf{indiv}^C_{\sigma.r}, 1, i) \rightarrow L_{\langle \sigma.r.i,\ C\rangle}$ of type (6.4))/(6.8), and $\mathsf{IC}(\mathsf{indiv}^C_{\sigma.r}, 1, i) \rightarrow A_{\langle \sigma.r.i,\ \top\rangle}$ of type (6.5)/(6.9), are satisfied. We can distinguish two cases:

- if both $\sigma.r.i \in \Delta^{\mathcal{I}}$ and $\sigma.r.i \in C^{\mathcal{I}}$, then we have $A_{\langle \sigma.r.i,\ \top\rangle} \in \mu_\Delta$ from (6.31), $L_{\langle \sigma.r.i,\ C\rangle} \in \mu_X$ from (6.32) and $\mathsf{IC}(\mathsf{indiv}^C_{\sigma.r}, 1, i) \in \mu_{\mathsf{IC}}$ from (6.36), so that μ satisfies both the clauses;
- if, on the contrary, either $\sigma.r.i \notin \Delta^{\mathcal{I}}$ or $\sigma.r.i \notin C^{\mathcal{I}}$, then we have $\neg\mathsf{IC}(\mathsf{indiv}^C_{\sigma.r}, 1, i) \in \mu_{\mathsf{IC}}$ from 6.36, which trivially satisfies both the clauses.

(6.6), (6.7): Let us consider the clause $(A_{\langle \sigma,\ \geq nr.C\rangle} \wedge A_{\langle \sigma,\ \top\rangle}) \rightarrow \neg\mathsf{BC}(\mathsf{indiv}^C_{\sigma.r}, n-1)$ of type (6.6) and the clause $(\neg\mathsf{BC}(\mathsf{indiv}^C_{\sigma.r}, n-1) \wedge A_{\langle \sigma,\ \top\rangle}) \rightarrow A_{\langle \sigma,\ \geq nr.C\rangle}$ of type (6.7). Since $\sigma \in \Delta^{\mathcal{I}}$ by hypothesis then $A_{\langle \sigma,\ \top\rangle} \in \mu_\Delta$ by (6.31). If also $\sigma \in (\geq nr.C)^{\mathcal{I}}$, then $A_{\langle \sigma,\ \geq nr.C\rangle} \in \mu_X$ by (6.32) and $\neg\mathsf{BC}(\mathsf{indiv}^C_{\sigma.r}, n-1) \in \mu_{\geq}$ by (6.33) satisfying both the clauses. Otherwise, if $\sigma \notin (\geq nr.C)^{\mathcal{I}}$, then the clause (6.6) is trivially satisfied since $\neg A_{\langle \sigma,\ \geq nr.C\rangle} \in \mu_X$ by (6.32), while $\mathsf{BC}(\mathsf{indiv}^C_{\sigma.r}, n-1) \in \mu_{\geq}$ by (6.33) which satisfies the clause (6.7).

(6.10): Wlog. let us consider the case in witch the index of the IC-literal is i and, thus, it is associated to the individual $\sigma.r.i$, for some role r, some basic concept C, the integer value i and $\sigma \in \Delta^{\mathcal{I}}$. Thus we must show that the clause: $(L_{\langle \sigma.r.i,\ C\rangle} \wedge A_{\langle \sigma.r.i,\ \top\rangle}) \rightarrow \mathsf{IC}(\mathsf{indiv}^{C}_{\sigma.r}, 1, i)$ of type (6.10) is satisfied. We can distinguish three cases:

- if both $\sigma.r.i \in \Delta^{\mathcal{I}}$ and $\sigma.r.i \in C^{\mathcal{I}}$, then we have $A_{\langle \sigma.r.i,\ \top\rangle} \in \mu_{\Delta}$ from (6.31), $L_{\langle \sigma.r.i,\ C\rangle} \in \mu_X$ from (6.32) and $\mathsf{IC}(\mathsf{indiv}^{C}_{\sigma.r}, 1, j) \in \mu_{\mathsf{IC}}$ from (6.36), so that μ satisfies the clause;
- if, on the contrary, $\sigma.r.i \notin C^{\mathcal{I}}$, then we have $\neg L_{\langle \sigma.r.i,\ C\rangle} \in \mu_X$ from (6.32), which trivially satisfies the clause;
- otherwise, if $\sigma.r.i \notin \Delta^{\mathcal{I}}$, then we have $\neg A_{\langle \sigma.r.i,\ \top\rangle} \in \mu_{\overline{\Delta}}$ from (6.38), which trivially satisfies the clause.

(6.11), (6.12): Let us consider the clause $(A_{\langle \sigma,\ \leq mr.C\rangle} \wedge A_{\langle \sigma,\ \top\rangle}) \rightarrow \mathsf{BC}(\mathsf{indiv}^{C}_{\sigma.r}, m)$ of type (6.11). and the clause $(\mathsf{BC}(\mathsf{indiv}^{C}_{\sigma.r}, m) \wedge A_{\langle \sigma,\ \top\rangle}) \rightarrow A_{\langle \sigma,\ \leq mr.C\rangle}$ of type (6.12). Since $\sigma \in \Delta^{\mathcal{I}}$ by hypothesis, then $A_{\langle \sigma,\ \top\rangle} \in \mu_{\Delta}$ by (6.31). If also $\sigma \in (\leq mr.C)^{\mathcal{I}}$, then $A_{\langle \sigma,\ \leq mr.C\rangle} \in \mu_X$ by (6.32) and $\mathsf{BC}(\mathsf{indiv}^{C}_{\sigma.r}, m) \in \mu_{\leq}$ by (6.34) satisfying both the clauses. Otherwise, if $\sigma \notin (\leq mr.C)^{\mathcal{I}}$, then the clause (6.11) is trivially satisfied since $\neg A_{\langle \sigma,\ \leq mr.C\rangle} \in \mu_X$ by (6.32), while $\neg\mathsf{BC}(\mathsf{indiv}^{C}_{\sigma.r}, m) \in \mu_{\leq}$ by (6.34) which satisfies the clause (6.12).

(6.13): Wlog. let us consider the generic clause of type (6.13): $(A_{\langle \sigma,\ \forall r.C\rangle} \wedge A_{\langle \sigma.r.i,\ \top\rangle}) \rightarrow L_{\langle \sigma.r.i,\ C\rangle}$, for some role r, some basic concept C, the integer value i, $\sigma \in \Delta^{\mathcal{I}}$ and $\sigma.r.i \in \Sigma^{\mathcal{T}}$. Further, let us consider the case in which $\sigma \in (\forall r.C)^{\mathcal{I}}$; otherwise, the clause is trivially satisfied from $\neg A_{\langle \sigma,\ \forall r.C\rangle} \in \mu_X$, due to (6.32). If $\sigma \in (\forall r.C)^{\mathcal{I}}$ then we have $A_{\langle \sigma,\ \forall r.C\rangle} \in \mu_X$ from (6.32) and we can distinguish two more cases:

- if $\sigma.r.i \notin \Delta^{\mathcal{I}}$ the clause is trivially satisfied from $\neg A_{\langle \sigma.r.i,\ \top\rangle} \in \mu_{\overline{\Delta}}$, due to (6.38);
- if, on the contrary, $\sigma.r.i \in \Delta^{\mathcal{I}}$, since $\mathcal{I}$ is a model for $\mathcal{T}$, then $\sigma \in (\forall r.C)^{\mathcal{I}}$ implies $\sigma.r.i \in C^{\mathcal{I}}$ for every $(\sigma, \sigma.r.i) \in r^{\mathcal{I}}$, from which it follows $L_{\langle \sigma.r.i,\ C\rangle} \in \mu_X$, due to (6.32). Further, from $\sigma.r.i \in \Delta^{\mathcal{I}}$ we have $A_{\langle \sigma.r.i,\ \top\rangle} \in \mu_{\Delta}$ (6.31) satisfying the clause.

(6.14): Let us consider the clause $(\mathsf{BC}(\mathsf{indiv}^{\neg C}_{\sigma.r}, 0) \wedge A_{\langle \sigma,\ \top\rangle}) \rightarrow A_{\langle \sigma,\ \forall r.C\rangle}$ of type (6.14). Since $\sigma \in \Delta^{\mathcal{I}}$ by hypothesis, then $A_{\langle \sigma,\ \top\rangle} \in \mu_{\Delta}$ by (6.31). Then, by definition of μ_X (6.32) and $\mu_{\forall}$ (6.35) respectively, either $\sigma \in (\forall r.C)^{\mathcal{I}}$ and both $A_{\langle \sigma,\ \forall r.C\rangle}$, $\mathsf{BC}(\mathsf{indiv}^{\neg C}_{\sigma.r}, 0)$ are true in μ, or $\sigma \notin (\forall r.C)^{\mathcal{I}}$ and they are both false in μ. In both cases μ satisfies the clause (6.12).

Second we show that μ, and in particular $\mu_{\overline{\mathcal{I}}}$, propositionally satisfies all the clauses of $\varphi^{\mathcal{T}}$ such that $\sigma \notin \Delta^{\mathcal{I}}$. We prove this fact for every type of clause from (6.3) to (6.14).

(6.4), (6.5), (6.8), (6.9): All the clauses of these types are trivially satisfied by $\mu_{\overline{\mathsf{IC}}}$, since, by (6.40), we have $\neg\mathsf{IC}(\mathsf{indiv}^{C}_{\sigma.r}, 1, \ldots) \in \mu_{\overline{\mathsf{IC}}}$ for every literal $\mathsf{IC}(\mathsf{indiv}^{C}_{\sigma.r}, 1, \ldots)$ such that $\sigma \notin \Delta^{\mathcal{I}}$.

(6.6), (6.7), (6.11), (6.12), (6.14): All the clauses of these types are trivially satisfied by $\mu_{\overline{\Delta}}$, in fact, since $\sigma \notin \Delta^{\mathcal{I}}$, then $\neg A_{\langle \sigma,\ \top \rangle} \in \mu_{\overline{\Delta}}$ by (6.38).

(6.10), (6.13): The same argument of the previous point can be spent for these clauses. In fact, since $\mathcal{I}$ is a model for $\mathcal{T}$, if $\sigma \notin \Delta^{\mathcal{I}}$ then also $\sigma.r.i \notin \Delta^{\mathcal{I}}$ for every $\sigma.r.i \in \Sigma^{\mathcal{T}}$. Thus we have $\neg A_{\langle \sigma.r.i,\ \top \rangle} \in \mu_{\overline{\Delta}}$ by (6.38), which trivially satisfies the clauses.

(6.3): Finally, we consider the case of all the clauses of type (6.3) in which $\sigma \notin \Delta^{\mathcal{I}}$. The clauses of type (6.3) are the propositional correspondence of the concept inclusions of $\mathcal{T}$ and in particular, by definition of $\varphi^{\mathcal{T}}$, a clause of type (6.3) can exist in $\varphi^{\mathcal{T}}$ wrt. the individual σ only if the same clause exists in $\varphi^{\mathcal{T}}$ wrt. the individual 1 (because every axiom is encoded in 1). But, since $1 \in \Delta^{\mathcal{I}}$ and we have already proved that every clause of $\varphi^{\mathcal{T}}$ wrt. some σ such that $\sigma \in \Delta^{\mathcal{I}}$ is satisfiable, then there exists a satisfying truth assignment for all the literals occurring in all the clauses of type (6.3) wrt. the individual 1. If such an assignment exists, then there exists also a consistent truth assignment, that we called $\mu_{\overline{X}}$, for all the literals occuring in clauses of type (6.3) wrt. any other individual $\sigma \notin \Delta^{\mathcal{I}}$, such that it satisfies all these clauses. In fact, notice that the clauses of type (6.3) wrt. any individual σ are a subset of those of the same kind wrt. 1. Notice also (as shown above) that all the other clauses different from type (6.3) are already satisfied by sub-assignments of $\mu_{\overline{\mathcal{I}}}$ which do not include any of the literals assigned by $\mu_{\overline{X}}$. Thus $\mu_{\overline{X}}$ exists and satisfies all the clauses of type (6.3) in the cases of $\sigma \notin \Delta^{\mathcal{I}}$.

Finally we show that μ satisfies $\varphi^{\mathcal{T}}$ with respect to the Theory of Costs $\mathcal{C}$.
So we must prove that μ satisfies all the constraints introduced by the $\mathcal{C}$-literals, that is if a bound (BC-literal) wrt. the cost variable $\mathsf{indiv}^C_{\sigma.r}$ is assigned to true [resp. false] then the sum of all the incur costs for $\mathsf{indiv}^C_{\sigma.r}$ does not [resp. does] exceed the bound. Since all the incur costs defined in $\varphi^{\mathcal{T}}$ have value 1, it means that the number of IC-literals assigned to true is not [resp. is] greater than the fixed bound. We prove this fact distinguishing some cases:

- First, we consider all the clauses containing $\mathcal{C}$-literals referring to some cost variable $\mathsf{indiv}^C_{\sigma.r}$, with $\sigma \notin \Delta^{\mathcal{I}}$. Notice that $\mu_{\overline{\mathsf{BC}}}$ (6.39) assigns to true every bound $\mathsf{BC}(\mathsf{indiv}^C_{\sigma.r}, \ldots)$ such that $\sigma \notin \Delta^{\mathcal{I}}$ while, instead, $\mu_{\overline{\mathsf{IC}}}$ (6.40) assigns to false every incur cost $\mathsf{IC}(\mathsf{indiv}^C_{\sigma.r}, 1, \ldots)$ for the same σ. This assigment is consistent wrt. the Theory of Costs. In fact, by assigning to true all the BC-literals, only upper-bounds are fixed, and these upper-bounds are all trivially satisfied because (with no enabled incur costs) every cost variable $\mathsf{indiv}^C_{\sigma.r}$ valuates to zero.

- Second, we consider the case $\sigma \in \Delta^{\mathcal{I}}$. Notice that the sub-assignments $\mu_{\geq}$, $\mu_{\leq}$ and $\mu_{\forall}$, all assign values to the BC-literals when $\sigma \in \Delta^{\mathcal{I}}$. First of all, they assign a value to all such BC-literals, in fact they cover all the possible cases of clauses in which BC-literals can appear. Second, even if they possibly assign the same BC-literal twice, (for the same σ, r and C, when $n - 1 = m$ or $n - 1 = 0$) they are mutually consistent. In fact they are guarateed by the semantic of $\mathcal{I}$, which is a model for $< T$.

Thus, in the case $n-1=m$ if $\sigma \in (\geq nr.C)^{\mathcal{I}}$, then there are at least n individuals $\sigma.r.i \in C^{\mathcal{I}}$, and thus $\sigma \notin (\leq mr.C)^{\mathcal{I}}$. And, vice versa, if $\sigma \in (\leq mr.C)^{\mathcal{I}}$ and $m = n-1$ then $\sigma \notin (\geq nr.C)^{\mathcal{I}}$. Similarly $\mu_{\geq}$ and $\mu_{\forall}$ can assign the same BC-literals, while $\mu_{\forall}$ and $\mu_{\leq}$ can never intersect. But, also in this case, if $n = 1$ and $\sigma \in (\geq 1r.\neg C)^{\mathcal{I}}$, then it exists at least one individual $\sigma.r.i \in (\neg C)^{\mathcal{I}}$, and thus $\sigma \notin (\forall r.C)^{\mathcal{I}}$, and vice versa. Further, the semantic of $< I$ guarantees that it could never happen, e.g., $\sigma \notin (\leq mr.C)^{\mathcal{I}}$ and $\sigma \notin (\geq nr.C)^{\mathcal{I}}$, with $n-1=m$, for $\sigma \in \Delta^{\mathcal{I}}$. Notice at last that, by definition of $\mathcal{ALCQ}2SMT_{\mathcal{C}}$, if BC-literals are introduced for some σ with the respective literals $L_{\langle \sigma,\ \Re r.C \rangle}$ for some restriction $\Re$, then also the many respective IC-literals and possible individuals $\sigma.r.i$ are introduced wrt. the same σ.

With these premises let as prove the other following exhaustive sub-cases, in the cases in which the mentioned literals occur in $\varphi^{\mathcal{T}}$:

- Let consider either the case $\sigma \in (\geq nr.C)^{\mathcal{I}}$ or $\sigma \notin (\leq mr.C)^{\mathcal{I}}$ for a generic value of n and $m = n-1$. It follows, either by (6.33) for $\mu_{\geq}$ or by (6.34) for $\mu_{\leq}$, $\neg\mathsf{BC}(\mathsf{indiv}^{C}_{\sigma.r}, n-1) \in \mu$, thus there must be at least n distinct enabled incur costs of value 1 wrt. $\mathsf{indiv}^{C}_{\sigma.r}$, in order to be consistent wrt. the Theory of Costs. Given $\sigma \in (\geq nr.C)^{\mathcal{I}}$ (or, respectively, $\sigma \notin (\leq mr.C)^{\mathcal{I}}$), since $\mathcal{I}$ is a model for $\mathcal{T}$, there must be at least n distinct individuals $\sigma.r.i$ such that $\sigma.r.i \in \Delta^{\mathcal{I}}$ and $\sigma.r.i \in C^{\mathcal{I}}$. Hence, by (6.36), μ_{IC} consistently assigns to true at least n distinct literals in the form $\mathsf{IC}(\mathsf{indiv}^{C}_{\sigma.r}, 1, i)$, as required.
- In the opposite case, if $\sigma \notin (\geq nr.C)^{\mathcal{I}}$ or $\sigma \in (\leq mr.C)^{\mathcal{I}}$ for a generic value of n and $m = n-1$, we have $\mathsf{BC}(\mathsf{indiv}^{C}_{\sigma.r}, m) \in \mu$ either by (6.34) for $\mu_{\leq}$ or by (6.33) for $\mu_{\geq}$. Given $\sigma \in (\leq mr.C)^{\mathcal{I}}$ (or, respectively, $\sigma \notin (\geq nr.C)^{\mathcal{I}}$), there can not exist more than m distinct individuals $\sigma.r.i$ such that $\sigma.r.i \in \Delta^{\mathcal{I}}$ and $\sigma.r.i \in C^{\mathcal{I}}$. Hence, by (6.36), μ_{IC} assigns to true at most m distinct literals in the form $\mathsf{IC}(\mathsf{indiv}^{C}_{\sigma.r}, 1, i)$, satisfying the fixed bound in the Theory of Costs.
- If it holds $\sigma \in (\forall r.C)^{\mathcal{I}}$, then $\mathsf{BC}(\mathsf{indiv}^{\neg C}_{\sigma.r}, 0) \in \mu_{\forall}$ by (6.35), so there can not be any incur cost wrt. $\mathsf{indiv}^{\neg C}_{\sigma.r}$ assigned to true. Since $\mathcal{I}$ is a model for $\mathcal{T}$, $\sigma \in (\forall r.C)^{\mathcal{I}}$ implies that for every individual $\sigma.r.i \in \Delta^{\mathcal{I}}$ it holds $\sigma.r.i \in C^{\mathcal{I}}$ (in fact, by Lemma 11, the individuals in relation with σ through $r^{\mathcal{I}}$ are all and only those in the form $\sigma.r.i \in \Delta^{\mathcal{I}}$). Consequently, for every $\sigma.r.i \in \Delta^{\mathcal{I}}$ it holds $\sigma.r.i \notin (\neg C)^{\mathcal{I}}$, thus no literals $\mathsf{IC}(\mathsf{indiv}^{\neg C}_{\sigma.r}, 1, i)$ can be assigned to true neither by μ_{IC} (6.36) nor by $\mu_{\overline{\mathsf{IC}}}$ (6.40), consistently with the Theory of Costs.
- If, on the contrary, $\sigma \notin (\forall r.C)^{\mathcal{I}}$, then $\neg\mathsf{BC}(\mathsf{indiv}^{\neg C}_{\sigma.r}, 0) \in \mu_{\forall}$ by (6.35). Since $\mathcal{I}$ is a model for $\mathcal{T}$, $\sigma \in (\forall r.C)^{\mathcal{I}}$ implies that it exists at least one individual $\sigma.r.i \in \Delta^{\mathcal{I}}$ such that $\sigma.r.i \in (\neg C)^{\mathcal{I}}$ Consequently, at least one literal $\mathsf{IC}(\mathsf{indiv}^{\neg C}_{\sigma.r}, 1, i)$ is assigned to true by μ_{IC} (6.36), consistently with the Theory of Costs.

□

Theorem 7. *Given an $\mathcal{ALCQ}$ acyclic TBox $\mathcal{T}$ in normal form and the encoding $\mathcal{ALCQ}2SMT_{\mathcal{C}}(\mathcal{T}) = \langle \Sigma^{\mathcal{T}}, \mathcal{I}^{\mathcal{T}}_{-}, \mathcal{I}^{\mathcal{T}}_{+}, A_{\langle\ ,\ \rangle}, \mathsf{indiv}, \varphi^{\mathcal{T}} \rangle$ of Definition 5, then the normal concept $\hat{C}$, such that $\hat{C} \sqsubseteq \hat{D} \in \mathcal{T}$, is satisfiable wrt. $\mathcal{T}$ if and only if the SMT($\mathcal{C}$)-formula $\varphi^{\mathcal{T}} \wedge L_{\langle 1,\ \hat{C}\rangle}$ is satisfiable.*

Proof. First let us prove that our approach is sound, that is if $\varphi^{\mathcal{T}} \wedge L_{\langle 1,\ \hat{C}\rangle}$ is satisfiable then $\hat{C}$ is satisfiable wrt. $\mathcal{T}$. In other words, we prove that if $\varphi^{\mathcal{T}} \wedge L_{\langle 1,\ \hat{C}\rangle}$ is satisfiable then it there exists an interpretation $\mathcal{I}$, such that $\mathcal{I}$ is a model for $\mathcal{T}$ and $\hat{C}^{\mathcal{I}} \neq \emptyset$. Notice that $\varphi^{\mathcal{T}} \wedge L_{\langle 1,\ \hat{C}\rangle}$ is satisfiable if and only if $\varphi^{\mathcal{T}}$ is satisfiable. Let us call μ the truth assignment satisfying $\varphi^{\mathcal{T}}$ and such that $L_{\langle 1,\ \hat{C}\rangle} \in \mu$. This has been said, we chose the interpretation $\mathcal{I}_{\mu}$ by Lemma 10 as a model for $\mathcal{T}$. Since we have $L_{\langle 1,\ \hat{C}\rangle} \in \mu$, it is a direct consequence of the Lemma 10 that $1 \in \hat{C}^{\mathcal{I}_{\mu}}$, so that $\hat{C}^{\mathcal{I}_{\mu}} \neq \emptyset$, i.e. $\hat{C}$ is satisfiable wrt. $\mathcal{T}$.

Then we prove that our approach is complete. We must prove that if $\hat{C}$ is satisfiable wrt. $\mathcal{T}$ then $\mathcal{ALCQ}2SMT_{\mathcal{C}}(\mathcal{T}) \wedge L_{\langle 1,\ C\rangle}$ is satisfiable. We assume that $\mathcal{T}$ is consistent (otherwise it follows trivially by Theorem 6 that $\varphi^{\mathcal{T}}$ is unsatisfiable), and that the interpretation $\mathcal{I}$ is a model for $\mathcal{T}$ such that $\hat{C}^{\mathcal{I}} \neq \emptyset$ (i.e. $\hat{C}$ is satisfiable wrt. $\mathcal{T}$). Further, we can assume $\Delta^{\mathcal{I}} \subseteq \Sigma^{\mathcal{T}}$ (Lemma 11) from which it follows, by Lemma 12, that there exists a truth assignment μ satisfying $\varphi^{\mathcal{T}}$ build up as in (6.29). In particular, since $\hat{C} \sqsubseteq \hat{D} \in \mathcal{T}$ and $\hat{C}$ is consistent and has been encoded in 1 with $1 \in \Delta^{\mathcal{I}}$, we have $1 \in \hat{C}^{\mathcal{I}}$. From this latter fact it follows $L_{\langle 1,\ \hat{C}\rangle} \in \mu_X \subseteq \mu$ due to Lemma 12 (6.32). □

6.10 Appendix: An Encoding Example

Consider the acyclic $\mathcal{ALCQ}$ TBox $\mathcal{T}^*$ composed of the following axioms (for briefness, in the rest of this example we refer to the right-side short version of the axioms of $\mathcal{T}^*$ and we skip to transform them in normal form):

$\mathsf{HappyFather} \sqsubseteq \geq 2\ \mathsf{hasSon.Professor}$	$F \sqsubseteq \geq 2\ s.P$
$\mathsf{HappyFather} \sqsubseteq \geq 2\ \mathsf{hasSon.Medic}$	$F \sqsubseteq \geq 2\ s.M$
$\mathsf{HappyFather} \sqsubseteq \geq 2\ \mathsf{hasSon.Rich}$	$F \sqsubseteq \geq 2\ s.R$
$\mathsf{HappyFather} \sqsubseteq \leq 3\ \mathsf{hasSon.}\top$	$F \sqsubseteq \leq 3\ s.\top$
$\mathsf{Professor} \sqsubseteq \exists \mathsf{hasIncome.LowSalary}$	$P \sqsubseteq \geq 1\ r.L$
$\mathsf{Professor} \sqsubseteq \leq 2\ \mathsf{hasIncome.}\top$	$P \sqsubseteq \leq 2\ r.\top$
$\mathsf{Rich} \sqsubseteq \exists\ \mathsf{hasIncome.}\top \sqcap (\forall\ \mathsf{hasIncome.HighSalary} \sqcup \geq 3\ \mathsf{hasIncome.}\top)$	$R \sqsubseteq \geq 1\ r.\top \sqcap (\forall r.H \sqcup \geq 3\ r.\top)$
$\mathsf{LowSalary} \sqsubseteq \neg \mathsf{HighSalary}$	$L \sqsubseteq \neg H$

The formula $\varphi^{\mathcal{T}^*}$ of $\mathcal{ALCQ}2SMT_{\mathcal{C}}$ ($\mathcal{T}^*$) is generated as follows:

1. Encoding of the TBox axioms (6.3) in the root individual 1:

$$
\begin{aligned}
& A_{\langle 1,\ \top\rangle} \\
\wedge\ & A_{\langle 1,\ F\rangle} \rightarrow A_{\langle 1,\ \geq 2s.P\rangle} && \wedge\ A_{\langle 1,\ P\rangle} \rightarrow A_{\langle 1,\ \geq 1r.L\rangle} \\
\wedge\ & A_{\langle 1,\ F\rangle} \rightarrow A_{\langle 1,\ \geq 2s.M\rangle} && \wedge\ A_{\langle 1,\ P\rangle} \rightarrow A_{\langle 1,\ \leq 2r.\top\rangle} \\
\wedge\ & A_{\langle 1,\ F\rangle} \rightarrow A_{\langle 1,\ \geq 2s.R\rangle} && \wedge\ A_{\langle 1,\ R\rangle} \rightarrow A_{\langle 1,\ \geq 1r.\top\rangle} \\
\wedge\ & A_{\langle 1,\ F\rangle} \rightarrow A_{\langle 1,\ \leq 3s.\top\rangle} && \wedge\ A_{\langle 1,\ R\rangle} \rightarrow (A_{\langle 1,\ \forall r.H\rangle} \vee A_{\langle 1,\ \geq 3r.\top\rangle}) \\
\wedge\ & A_{\langle 1,\ L\rangle} \rightarrow \neg A_{\langle 1,\ H\rangle}
\end{aligned}
$$

2. Encoding of the at-least number restrictions wrt. the role r and the individual 1 (i.e. $\langle 1, \geq n\ r.C\rangle$), through the clauses (6.6) and (6.4), (6.5):

$$
\begin{aligned}
\wedge\ & (A_{\langle 1,\ \geq 1r.L\rangle} \wedge A_{\langle 1,\ \top\rangle}) \rightarrow \neg \mathsf{BC}(\mathsf{indiv}^{L}_{1.r}, 0) \\
\wedge\ & \mathsf{IC}(\mathsf{indiv}^{L}_{1.r}, 1, 1) \rightarrow A_{\langle 1.r.1,\ L\rangle} && \wedge\ \mathsf{IC}(\mathsf{indiv}^{L}_{1.r}, 1, 1) \rightarrow A_{\langle 1.r.1,\ \top\rangle} \\
\wedge\ & (A_{\langle 1,\ \geq 3r.\top\rangle} \wedge A_{\langle 1,\ \top\rangle}) \rightarrow \neg \mathsf{BC}(\mathsf{indiv}^{\top}_{1.r}, 2) \\
\wedge\ & (A_{\langle 1,\ \geq 1r.\top\rangle} \wedge A_{\langle 1,\ \top\rangle}) \rightarrow \neg \mathsf{BC}(\mathsf{indiv}^{\top}_{1.r}, 0) \\
\wedge\ & \mathsf{IC}(\mathsf{indiv}^{\top}_{1.r}, 1, 2) \rightarrow A_{\langle 1.r.2,\ \top\rangle} && \wedge\ \top \\
\wedge\ & \mathsf{IC}(\mathsf{indiv}^{\top}_{1.r}, 1, 3) \rightarrow A_{\langle 1.r.3,\ \top\rangle} && \wedge\ \top \\
\wedge\ & \mathsf{IC}(\mathsf{indiv}^{\top}_{1.r}, 1, 4) \rightarrow A_{\langle 1.r.4,\ \top\rangle} && \wedge\ \top
\end{aligned}
$$

   Notice that when an at-least restriction applies to the concept $\top$ (see, e.g., the encoding of the $\langle 1, \geq 3\ r.\top\rangle$) the clauses of type (6.5) are identical to the clause of type (6.4), thus the first can be avoided (here we replace them with $\top$). Notice also that, for what concerns the introduction of new individuals and the relative IC-clauses, only the at-least restriction with the greater value of n must be encoded, when many at-least restrictions refer to the same concept and role (e.g., for the instantiated concepts $\langle 1, \geq 3\ r.\top\rangle$ and $\langle 1, \geq 1\ r.\top\rangle$ only three individuals, instead of four, must be introduced).

3. Encoding of the at-most restrictions wrt. the role r and in the root label 1 (i.e. $\langle 1, \leq m\ r.C\rangle$). The expansion (6.11) introduces in $\varphi^{\mathcal{T}^*}$ the clause:

$$\wedge\ (A_{\langle 1,\ \leq 2r.\top\rangle} \wedge A_{\langle 1,\ \top\rangle}) \rightarrow \mathsf{BC}(\mathsf{indiv}^{\top}_{1.r}, 2)$$

while the interaction with the previously encoded at-least restrictions for r and 1 is handled by sharing the individuals, as done through the expansions (6.8) and (6.9):

$$\wedge\ \mathsf{IC}(\mathsf{indiv}^{L}_{1.r}, 1, 2) \rightarrow A_{\langle 1.r.2,\ L\rangle} \qquad \wedge\ \mathsf{IC}(\mathsf{indiv}^{L}_{1.r}, 1, 2) \rightarrow A_{\langle 1.r.2,\ \top\rangle}$$
$$\wedge\ \mathsf{IC}(\mathsf{indiv}^{L}_{1.r}, 1, 3) \rightarrow A_{\langle 1.r.3,\ L\rangle} \qquad \wedge\ \mathsf{IC}(\mathsf{indiv}^{L}_{1.r}, 1, 3) \rightarrow A_{\langle 1.r.3,\ \top\rangle}$$
$$\wedge\ \mathsf{IC}(\mathsf{indiv}^{L}_{1.r}, 1, 4) \rightarrow A_{\langle 1.r.4,\ L\rangle} \qquad \wedge\ \mathsf{IC}(\mathsf{indiv}^{L}_{1.r}, 1, 4) \rightarrow A_{\langle 1.r.4,\ \top\rangle}$$
$$\wedge\ \mathsf{IC}(\mathsf{indiv}^{\top}_{1.r}, 1, 1) \rightarrow A_{\langle 1.r.1,\ \top\rangle} \qquad \wedge\ \top$$

Finally, for every previously generated individual one clause of type (6.10) is introduced:

$$\wedge\ A_{\langle 1.r.1,\ \top\rangle} \rightarrow \mathsf{IC}(\mathsf{indiv}^{\top}_{1.r}, 1, 1) \qquad \wedge\ A_{\langle 1.r.3,\ \top\rangle} \rightarrow \mathsf{IC}(\mathsf{indiv}^{\top}_{1.r}, 1, 3)$$
$$\wedge\ A_{\langle 1.r.2,\ \top\rangle} \rightarrow \mathsf{IC}(\mathsf{indiv}^{\top}_{1.r}, 1, 2) \qquad \wedge\ A_{\langle 1.r.4,\ \top\rangle} \rightarrow \mathsf{IC}(\mathsf{indiv}^{\top}_{1.r}, 1, 4)$$

Notice that these latter clauses of type (6.10) are simpler wrt. their expected form, because their two implying literals are identical each other (they both refer to $\top$), and thus only one literal is necessary.

4. Encoding of the universal restrictions wrt. the role r and in the root label 1 ($\langle 1, \forall r.C\rangle$), by mean of clauses of type (6.13):

$$\wedge\ (A_{\langle 1,\ \forall r.H\rangle} \wedge A_{\langle 1.r.1,\ \top\rangle}) \rightarrow A_{\langle 1.r.1,\ H\rangle} \qquad \wedge\ (A_{\langle 1,\ \forall r.H\rangle} \wedge A_{\langle 1.r.3,\ \top\rangle}) \rightarrow A_{\langle 1.r.3,\ H\rangle}$$
$$\wedge\ (A_{\langle 1,\ \forall r.H\rangle} \wedge A_{\langle 1.r.2,\ \top\rangle}) \rightarrow A_{\langle 1.r.2,\ H\rangle} \qquad \wedge\ (A_{\langle 1,\ \forall r.H\rangle} \wedge A_{\langle 1.r.4,\ \top\rangle}) \rightarrow A_{\langle 1.r.4,\ H\rangle}$$

5. Expansion of the TBox axioms (6.3) in the individuals $1.r.1, \ldots, 1.r.4$:

$$\wedge\ A_{\langle 1.r.1,\ L\rangle} \rightarrow \neg A_{\langle 1.r.1,\ H\rangle} \qquad \wedge\ A_{\langle 1.r.3,\ L\rangle} \rightarrow \neg A_{\langle 1.r.3,\ H\rangle}$$
$$\wedge\ A_{\langle 1.r.2,\ L\rangle} \rightarrow \neg A_{\langle 1.r.2,\ H\rangle} \qquad \wedge\ A_{\langle 1.r.4,\ L\rangle} \rightarrow \neg A_{\langle 1.r.4,\ H\rangle}$$

6. Encoding of the at-least restrictions wrt. the role s instantiated in 1 by mean of the clauses (6.6) and (6.4), (6.5):

$$\wedge\ (A_{\langle 1,\ \geq 2s.P\rangle} \wedge A_{\langle 1,\ \top\rangle}) \rightarrow \neg\mathsf{BC}(\mathsf{indiv}^{P}_{1.s}, 1)$$
$$\wedge\ \mathsf{IC}(\mathsf{indiv}^{P}_{1.s}, 1, 1) \rightarrow A_{\langle 1.s.1,\ P\rangle} \qquad \wedge\ \mathsf{IC}(\mathsf{indiv}^{P}_{1.s}, 1, 1) \rightarrow A_{\langle 1.s.1,\ \top\rangle}$$
$$\wedge\ \mathsf{IC}(\mathsf{indiv}^{P}_{1.s}, 1, 2) \rightarrow A_{\langle 1.s.2,\ P\rangle} \qquad \wedge\ \mathsf{IC}(\mathsf{indiv}^{P}_{1.s}, 1, 2) \rightarrow A_{\langle 1.s.2,\ \top\rangle}$$
$$\wedge\ (A_{\langle 1,\ \geq 2s.M\rangle} \wedge A_{\langle 1,\ \top\rangle}) \rightarrow \neg\mathsf{BC}(\mathsf{indiv}^{M}_{1.s}, 1)$$
$$\wedge\ \mathsf{IC}(\mathsf{indiv}^{M}_{1.s}, 1, 3) \rightarrow A_{\langle 1.s.3,\ M\rangle} \qquad \wedge\ \mathsf{IC}(\mathsf{indiv}^{M}_{1.s}, 1, 3) \rightarrow A_{\langle 1.s.3,\ \top\rangle}$$
$$\wedge\ \mathsf{IC}(\mathsf{indiv}^{M}_{1.s}, 1, 4) \rightarrow A_{\langle 1.s.4,\ M\rangle} \qquad \wedge\ \mathsf{IC}(\mathsf{indiv}^{M}_{1.s}, 1, 4) \rightarrow A_{\langle 1.s.4,\ \top\rangle}$$
$$\wedge\ (A_{\langle 1,\ \geq 2s.R\rangle} \wedge A_{\langle 1,\ \top\rangle}) \rightarrow \neg\mathsf{BC}(\mathsf{indiv}^{R}_{1.s}, 1)$$
$$\wedge\ \mathsf{IC}(\mathsf{indiv}^{R}_{1.s}, 1, 5) \rightarrow A_{\langle 1.s.5,\ R\rangle} \qquad \wedge\ \mathsf{IC}(\mathsf{indiv}^{R}_{1.s}, 1, 5) \rightarrow A_{\langle 1.s.5,\ \top\rangle}$$
$$\wedge\ \mathsf{IC}(\mathsf{indiv}^{R}_{1.s}, 1, 6) \rightarrow A_{\langle 1.s.6,\ R\rangle} \qquad \wedge\ \mathsf{IC}(\mathsf{indiv}^{R}_{1.s}, 1, 6) \rightarrow A_{\langle 1.s.6,\ \top\rangle}$$

7. Encoding of the at-most restrictions wrt. the role s in the root label 1; by mean of the clause (6.11):

$$\wedge\ (A_{\langle 1,\ \leq 3s.\top\rangle} \wedge A_{\langle 1,\ \top\rangle}) \rightarrow \mathsf{BC}(\mathsf{indiv}^{\top}_{1.s}, 3)$$

the clauses (6.8) and (6.9):

$$\begin{aligned}
&\wedge\, \mathsf{IC}(\mathsf{indiv}^P_{1.s}, 1, 3) \rightarrow A_{\langle 1.s.3,\ P\rangle} && \wedge\, \mathsf{IC}(\mathsf{indiv}^P_{1.s}, 1, 3) \rightarrow A_{\langle 1.s.3,\ \top\rangle}\\
&\wedge\, \mathsf{IC}(\mathsf{indiv}^P_{1.s}, 1, 4) \rightarrow A_{\langle 1.s.4,\ P\rangle} && \wedge\, \mathsf{IC}(\mathsf{indiv}^P_{1.s}, 1, 4) \rightarrow A_{\langle 1.s.4,\ \top\rangle}\\
&\wedge\, \mathsf{IC}(\mathsf{indiv}^P_{1.s}, 1, 5) \rightarrow A_{\langle 1.s.5,\ P\rangle} && \wedge\, \mathsf{IC}(\mathsf{indiv}^P_{1.s}, 1, 5) \rightarrow A_{\langle 1.s.5,\ \top\rangle}\\
&\wedge\, \mathsf{IC}(\mathsf{indiv}^P_{1.s}, 1, 6) \rightarrow A_{\langle 1.s.6,\ P\rangle} && \wedge\, \mathsf{IC}(\mathsf{indiv}^P_{1.s}, 1, 6) \rightarrow A_{\langle 1.s.6,\ \top\rangle}\\
&\wedge\, \mathsf{IC}(\mathsf{indiv}^M_{1.s}, 1, 1) \rightarrow A_{\langle 1.s.1,\ M\rangle} && \wedge\, \mathsf{IC}(\mathsf{indiv}^M_{1.s}, 1, 1) \rightarrow A_{\langle 1.s.1,\ \top\rangle}\\
&\wedge\, \mathsf{IC}(\mathsf{indiv}^M_{1.s}, 1, 2) \rightarrow A_{\langle 1.s.2,\ M\rangle} && \wedge\, \mathsf{IC}(\mathsf{indiv}^M_{1.s}, 1, 2) \rightarrow A_{\langle 1.s.2,\ \top\rangle}\\
&\wedge\, \mathsf{IC}(\mathsf{indiv}^M_{1.s}, 1, 5) \rightarrow A_{\langle 1.s.5,\ M\rangle} && \wedge\, \mathsf{IC}(\mathsf{indiv}^M_{1.s}, 1, 5) \rightarrow A_{\langle 1.s.5,\ \top\rangle}\\
&\wedge\, \mathsf{IC}(\mathsf{indiv}^M_{1.s}, 1, 6) \rightarrow A_{\langle 1.s.6,\ M\rangle} && \wedge\, \mathsf{IC}(\mathsf{indiv}^M_{1.s}, 1, 6) \rightarrow A_{\langle 1.s.6,\ \top\rangle}\\
&\wedge\, \mathsf{IC}(\mathsf{indiv}^R_{1.s}, 1, 1) \rightarrow A_{\langle 1.s.1,\ R\rangle} && \wedge\, \mathsf{IC}(\mathsf{indiv}^R_{1.s}, 1, 1) \rightarrow A_{\langle 1.s.1,\ \top\rangle}\\
&\wedge\, \mathsf{IC}(\mathsf{indiv}^R_{1.s}, 1, 2) \rightarrow A_{\langle 1.s.2,\ R\rangle} && \wedge\, \mathsf{IC}(\mathsf{indiv}^R_{1.s}, 1, 2) \rightarrow A_{\langle 1.s.2,\ \top\rangle}\\
&\wedge\, \mathsf{IC}(\mathsf{indiv}^R_{1.s}, 1, 3) \rightarrow A_{\langle 1.s.3,\ R\rangle} && \wedge\, \mathsf{IC}(\mathsf{indiv}^R_{1.s}, 1, 3) \rightarrow A_{\langle 1.s.3,\ \top\rangle}\\
&\wedge\, \mathsf{IC}(\mathsf{indiv}^R_{1.s}, 1, 4) \rightarrow A_{\langle 1.s.4,\ R\rangle} && \wedge\, \mathsf{IC}(\mathsf{indiv}^R_{1.s}, 1, 4) \rightarrow A_{\langle 1.s.4,\ \top\rangle}
\end{aligned}$$

which allow to share individuals, and of the clauses (6.10) wrt. all the introduced successor individuals:

$$\begin{aligned}
&\wedge\, A_{\langle 1.s.1,\ \top\rangle} \rightarrow \mathsf{IC}(\mathsf{indiv}^\top_{1.s}, 1, 1) && \wedge\, A_{\langle 1.s.4,\ \top\rangle} \rightarrow \mathsf{IC}(\mathsf{indiv}^\top_{1.s}, 1, 4)\\
&\wedge\, A_{\langle 1.s.2,\ \top\rangle} \rightarrow \mathsf{IC}(\mathsf{indiv}^\top_{1.s}, 1, 2) && \wedge\, A_{\langle 1.s.5,\ \top\rangle} \rightarrow \mathsf{IC}(\mathsf{indiv}^\top_{1.s}, 1, 5)\\
&\wedge\, A_{\langle 1.s.3,\ \top\rangle} \rightarrow \mathsf{IC}(\mathsf{indiv}^\top_{1.s}, 1, 3) && \wedge\, A_{\langle 1.s.6,\ \top\rangle} \rightarrow \mathsf{IC}(\mathsf{indiv}^\top_{1.s}, 1, 6)
\end{aligned}$$

8. Expansion of the TBox axioms (6.3) in every $1.s.i$ individual, with $i = 1, \ldots 6$: [13]

$$\begin{aligned}
&\wedge\, A_{\langle 1.s.i,\ P\rangle} \rightarrow A_{\langle 1.s.i,\ \geq 1 r.L\rangle}\\
&\wedge\, A_{\langle 1.s.i,\ P\rangle} \rightarrow A_{\langle 1.s.i,\ \leq 2 r.\top\rangle}\\
&\wedge\, A_{\langle 1.s.i,\ R\rangle} \rightarrow A_{\langle 1.s.i,\ \geq 1 r.\top\rangle}\\
&\wedge\, A_{\langle 1.s.i,\ R\rangle} \rightarrow (A_{\langle 1.s.i,\ \forall r.H\rangle} \vee A_{\langle 1.s.i,\ \geq 3 r.\top\rangle})
\end{aligned}$$

9. For $i = 1, \ldots 6$, encoding of the clauses (6.6) and (6.4), (6.5) for the at-least restrictions concerning the role r instantiated in $1.s.i$:

$$\begin{aligned}
&\wedge\, (A_{\langle 1.s.i,\ \geq 1 r.L\rangle} \wedge A_{\langle 1.s.i,\ \top\rangle}) \rightarrow \neg\mathsf{BC}(\mathsf{indiv}^L_{1.s.i.r}, 0)\\
&\wedge\, \mathsf{IC}(\mathsf{indiv}^L_{1.s.i.r}, 1, 1) \rightarrow A_{\langle 1.s.i.r.1,\ L\rangle} && \wedge\, \mathsf{IC}(\mathsf{indiv}^L_{1.s.i.r}, 1, 1) \rightarrow A_{\langle 1.s.i.r.1,\ \top\rangle}\\
&\wedge\, (A_{\langle 1.s.i,\ \geq 3 r.\top\rangle} \wedge A_{\langle 1.s.i,\ \top\rangle}) \rightarrow \neg\mathsf{BC}(\mathsf{indiv}^\top_{1.s.i.r}, 2)\\
&\wedge\, (A_{\langle 1.s.i,\ \geq 1 r.\top\rangle} \wedge A_{\langle 1.s.i,\ \top\rangle}) \rightarrow \neg\mathsf{BC}(\mathsf{indiv}^\top_{1.s.i.r}, 0)\\
&\wedge\, \mathsf{IC}(\mathsf{indiv}^\top_{1.s.i.r}, 1, 2) \rightarrow A_{\langle 1.s.i.r.2,\ \top\rangle} && \wedge\, \top\\
&\wedge\, \mathsf{IC}(\mathsf{indiv}^\top_{1.s.i.r}, 1, 3) \rightarrow A_{\langle 1.s.i.r.3,\ \top\rangle} && \wedge\, \top\\
&\wedge\, \mathsf{IC}(\mathsf{indiv}^\top_{1.s.i.r}, 1, 4) \rightarrow A_{\langle 1.s.i.r.4,\ \top\rangle} && \wedge\, \top
\end{aligned}$$

10. For $i = 1, \ldots 6$, encoding of the at-most restrictions wrt. the role r and instantiated in every individual $1.s.i$; first an upper bound is fixed via the clause (6.11):

$$\wedge\, (A_{\langle 1.s.i,\ \leq 2 r.\top\rangle} \wedge A_{\langle 1.s.i,\ \top\rangle}) \rightarrow \mathsf{BC}(\mathsf{indiv}^\top_{1.s.i.r}, 2)$$

[13] We remark that practically, despite our exposition in this example, the expansion of the encoding for every individual is fully performed before than any expansion concerning other individuals.

then, through the clauses (6.8) and (6.9), it is encoded the sharing of the individuals previously introduced by the different at-least restrictions:

$$\wedge\ \mathsf{IC}(\mathsf{indiv}^{L}_{1.s.i.r}, 1, 2) \rightarrow A_{\langle 1.s.i.r.2,\ L\rangle} \qquad \wedge\ \mathsf{IC}(\mathsf{indiv}^{L}_{1.s.i.r}, 1, 2) \rightarrow A_{\langle 1.s.i.r.2,\ \top\rangle}$$
$$\wedge\ \mathsf{IC}(\mathsf{indiv}^{L}_{1.s.i.r}, 1, 3) \rightarrow A_{\langle 1.s.i.r.3,\ L\rangle} \qquad \wedge\ \mathsf{IC}(\mathsf{indiv}^{L}_{1.s.i.r}, 1, 3) \rightarrow A_{\langle 1.s.i.r.3,\ \top\rangle}$$
$$\wedge\ \mathsf{IC}(\mathsf{indiv}^{L}_{1.s.i.r}, 1, 4) \rightarrow A_{\langle 1.s.i.r.4,\ L\rangle} \qquad \wedge\ \mathsf{IC}(\mathsf{indiv}^{L}_{1.s.i.r}, 1, 4) \rightarrow A_{\langle 1.s.i.r.4,\ \top\rangle}$$
$$\wedge\ \mathsf{IC}(\mathsf{indiv}^{\top}_{1.s.i.r}, 1, 1) \rightarrow A_{\langle 1.s.i.r.1,\ \top\rangle} \qquad \wedge\ \top$$

and, at last, the clauses of type (6.10) are introduced for all the previously generated individuals:

$$\wedge\ A_{\langle 1.s.i.r.1,\ \top\rangle} \rightarrow \mathsf{IC}(\mathsf{indiv}^{\top}_{1.s.i.r}, 1, 1) \qquad \wedge\ A_{\langle 1.s.i.r.3,\ \top\rangle} \rightarrow \mathsf{IC}(\mathsf{indiv}^{\top}_{1.s.i.r}, 1, 3)$$
$$\wedge\ A_{\langle 1.s.i.r.2,\ \top\rangle} \rightarrow \mathsf{IC}(\mathsf{indiv}^{\top}_{1.s.i.r}, 1, 2) \qquad \wedge\ A_{\langle 1.s.i.r.4,\ \top\rangle} \rightarrow \mathsf{IC}(\mathsf{indiv}^{\top}_{1.s.i.r}, 1, 4)$$

11. For $i = 1, \ldots 6$ encoding of the universal restrictions wrt. the role r instantiated in every individual $1.s.i$:

$$\wedge\ (A_{\langle 1.s.i,\ \forall r.H\rangle} \wedge A_{\langle 1.s.i.r.1,\ \top\rangle}) \rightarrow A_{\langle 1.s.i.r.1,\ H\rangle} \qquad \wedge\ (A_{\langle 1.s.i,\ \forall r.H\rangle} \wedge A_{\langle 1.s.i.r.3,\ \top\rangle}) \rightarrow A_{\langle 1.s.i.r.3,\ H\rangle}$$
$$\wedge\ (A_{\langle 1.s.i,\ \forall r.H\rangle} \wedge A_{\langle 1.s.i.r.2,\ \top\rangle}) \rightarrow A_{\langle 1.s.i.r.2,\ H\rangle} \qquad \wedge\ (A_{\langle 1.s.i,\ \forall r.H\rangle} \wedge A_{\langle 1.s.i.r.4,\ \top\rangle}) \rightarrow A_{\langle 1.s.i.r.4,\ H\rangle}$$

12. For $i = 1, \ldots 6$, expansion of the TBox axioms (6.3) in the individuals $1.s.i.r.1$, ..., $1.s.i.r.4$:

$$\wedge\ A_{\langle 1.s.i.r.1,\ L\rangle} \rightarrow \neg A_{\langle 1.s.i.r.1,\ H\rangle} \qquad \wedge\ A_{\langle 1.s.i.r.3,\ L\rangle} \rightarrow \neg A_{\langle 1.s.i.r.3,\ H\rangle}$$
$$\wedge\ A_{\langle 1.s.i.r.2,\ L\rangle} \rightarrow \neg A_{\langle 1.s.i.r.2,\ H\rangle} \qquad \wedge\ A_{\langle 1.s.i.r.4,\ L\rangle} \rightarrow \neg A_{\langle 1.s.i.r.4,\ H\rangle}$$

6.11 Appendix: Additional Plots on $\mathcal{ALCQ}2SMT_{\mathcal{C}}$

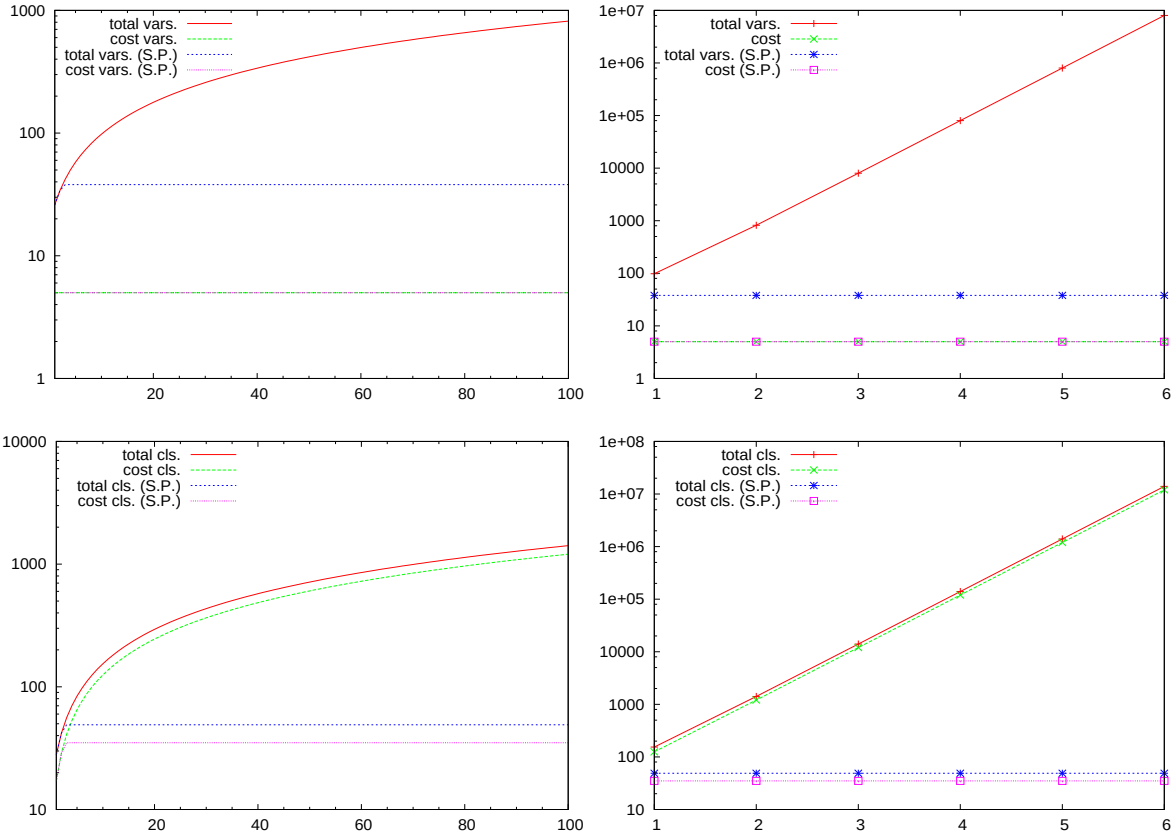


Figure 6.17: 1st column: $\texttt{increasing\_lin\_unsat}_i$, $i = 1, \ldots, 20$; 2nd column: $\texttt{increasing\_exp\_unsat}_i$, $i = 1, \ldots, 6$. 1st row: variables; 2nd row: clauses. X axis: test case index; Y axis: #variables/clauses.

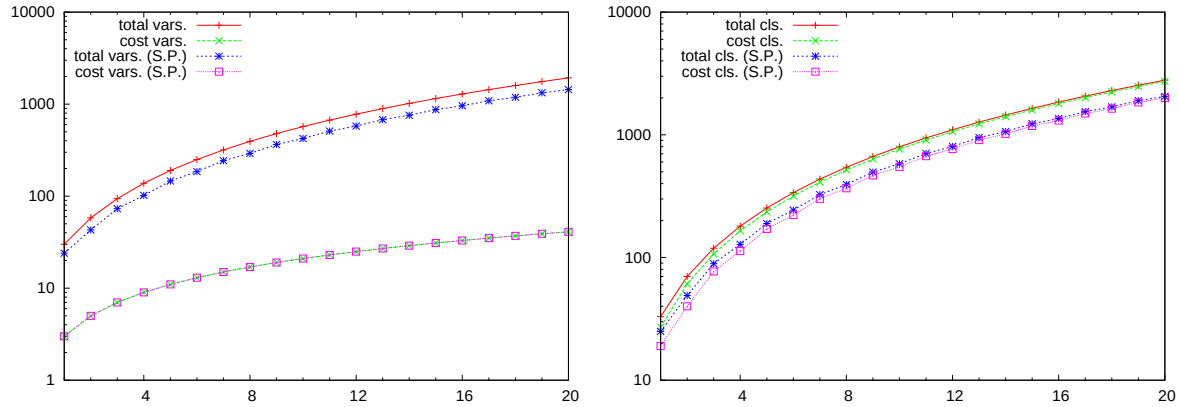


Figure 6.18: $\texttt{restr\_num}_i(1)$. Left: variables; right: clauses. X axis: test case index; Y axis: #variables/clauses.

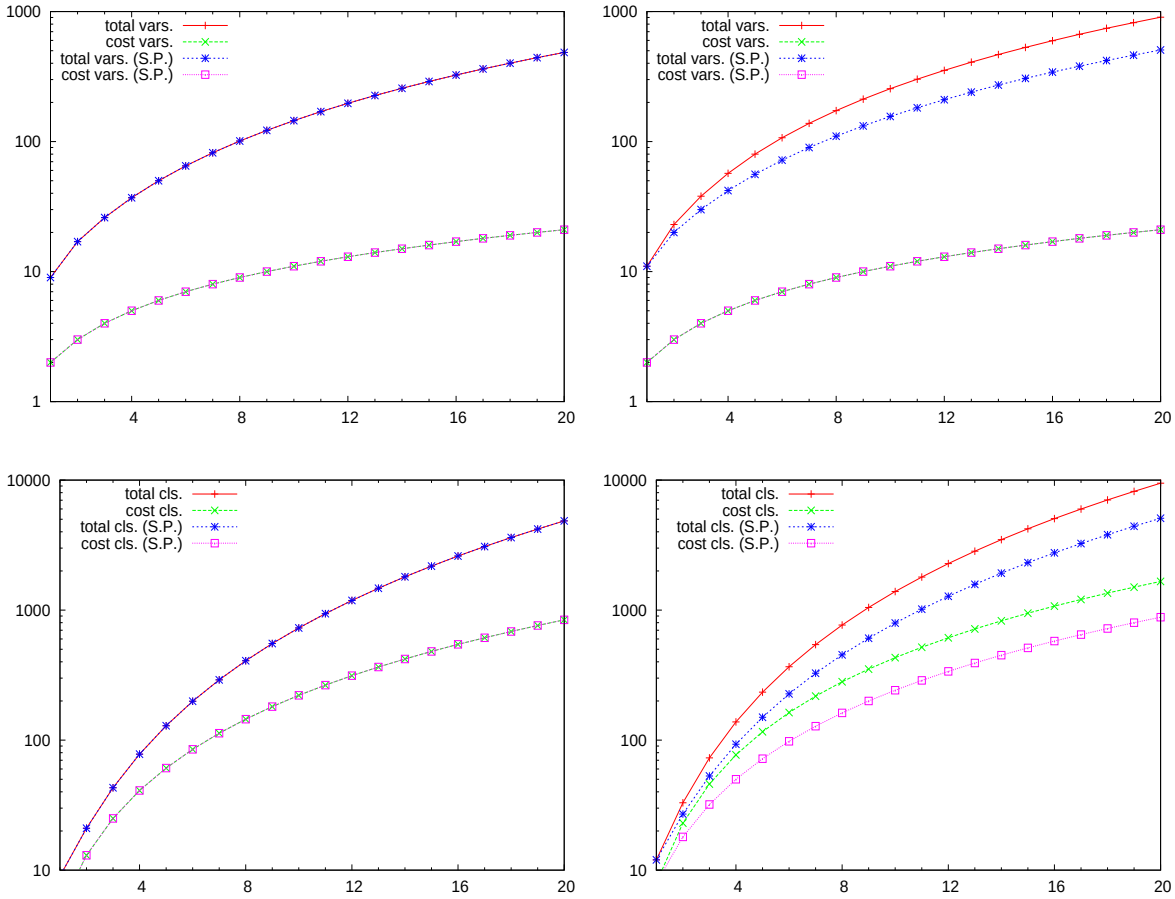


Figure 6.19: $\texttt{backtracking}_i(n)$. 1st column: $n = 1$; 2nd column: $n = 2$. 1st row: variables; 2nd row: clauses. X axis: test case index; Y axis: #variables/clauses.

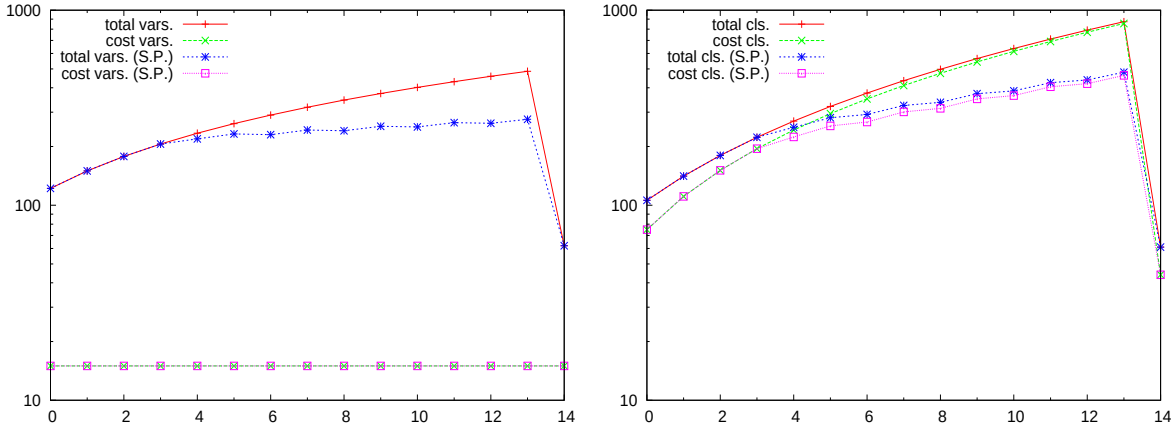


Figure 6.20: $\texttt{restr\_ratio}_i(1)$. Left: variables; right: clauses. X axis: test case index; Y axis: #variables/clauses.

Chapter 7

Exhaustively Debugging $\mathcal{EL}^+$ TBoxes via Horn-SAT and All-SMT

The recent quest for tractable logic-based languages arising from the field of bio-medical ontologies has raised a lot of attention on *lightweight* (i.e. less expressive but tractable) description logics, like $\mathcal{EL}$ and its family. To this extent, automated reasoning techniques in these logics have been developed for computing not only concept subsumptions, but also to pinpoint the set of axioms causing each subsumption. This task, called *axiom pinpointing*, allows the user for debugging ontologies by identifying the minimal subsets of axioms in the ontology causing undesired inferences.

In this last part of the thesis we build on previous work from the literature and push the envelope of a novel approach for axiom pinpointing in the logic $\mathcal{EL}^+$ and its sub-logics. In a nutshell, the idea is to build off-line a polynomial-size Horn propositional formula encoding the full classification of the input ontology, and to exploit the power of modern SAT and SMT techniques (like Boolean Constraint Propagation, Conflict Analysis, All-SMT) to compute subsumption and to efficiently perform axiom pinpointing. We first verify the potential of the approach and then we improve it thanks to some ad-hoc optimizations, ranging from an SMT-like theory propagation to a SAT-based technique for modularization. Thanks to the power of these technologies, we show in an extensive empirical evaluation how our tool $\mathcal{EL}^+$2SAT is extremely efficient and can deal with huge medical ontologies like SNOMED-CT in negligibel time. [1]

7.1 Related Works

In contrast to the trend of the last two decades (Baader et al., 2003), in which the research in Description Logic has focused on investigating increasingly expressive logics, the recent quest for tractable logic-based languages arising from the field of bio-medical ontologies has attracted a lot of attention on *lightweight* (i.e. less expressive but tractable) Description Logics, like $\mathcal{EL}$ and its family (Baader et al., 2005, 2006b, 2007; Konev et al.,

[1] The first part of this chapter is partially based on the conference paper: Sebastiani & Vescovi, 2009b.

2008c; Motik & Horrocks, 2008; Baader et al., 2008; Bienvenu, 2008; Lutz, Toman, & Wolter, 2009; Magka et al., 2010; Peñaloza & Sertkaya, 2010b). $\mathcal{EL}$ allows for conjunctions, existential restrictions and supports TBoxes with general concept inclusions. In particular, the logic $\mathcal{EL}^+$ (Baader et al., 2005, 2006b, 2007), which extends $\mathcal{EL}$ by adding also complex role inclusion axioms, is of particular relevance due to its algorithmic properties and due to its capability of expressing several important and widely-used bio-medical ontologies, such as SNOMED-CT (Spackman et al., 1997; Spackman, 2000; Suntisrivaraporn et al., 2007), NCI (Sioutos et al., 2007), GENEONTOLOGY (The G. O. Consortium, 2000) and the majority of GALEN (Rector & Horrocks, 1997) (see Section 2.1 for more details). In fact in $\mathcal{EL}^+$ not only standard logic problems such as *concept subsumption* (e.g., "is `Amputation-of-Finger` a subconcept of `Amputation-of-Arm` in the ontology SNOMED-CT?", Baader & Suntisrivaraporn, 2008), but also more sophisticated logic problems such as *axiom pinpointing* (e.g., "Find a minimal set of axioms in SNOMED-CT which are responsible of the fact that `Amputation-of-Finger` is a subconcept of `Amputation-of-Arm`?", Baader & Suntisrivaraporn, 2008) are tractable (Baader et al., 2007; Peñaloza & Sertkaya, 2010b). Importantly, the problem of axiom pinpointing in $\mathcal{EL}^+$ is of great interest for debugging complex bio-medical ontologies (see, e.g., Baader, Lutz, & Suntisrivaraporn, 2006a; Baader & Suntisrivaraporn, 2008; Suntisrivaraporn, 2009), precisely identifying and presenting the minimal subsets of the ontology axioms causing undesired existing concept subsumption relations.

To this extent, the problems of concept subsumption and axiom pinpointing in $\mathcal{EL}^+$ have been thoroughly investigated, and the development od efficient algorithms and tools able to handle and debug large real-world ontologies is a "hot" research issue. Thus, novel algorithm for these functionalities have been implemented and tested with success on large real-world ontologies, including SNOMED-CT (see, e.g., Baader et al., 2006b, 2007; Baader & Suntisrivaraporn, 2008; Sebastiani & Vescovi, 2009b). Further, the axiom pinpointing problem is increasingly catching attention especially in the *Semantic Web* research community (see, e.g., Kalyanpur et al., 2007; Suntisrivaraporn et al., 2008; Horridge et al., 2008; Peñaloza & Sertkaya, 2010a). In this domain the same problem is normally called *Find a/all Justification(s)* and is mostly solved via (so-called) *black-box* approaches, which try to extract justifications through blind search algorithms working regardless the specific properties of the handled logic/ontology (Schlobach & Cornet, 2003; Kalyanpur et al., 2007; Suntisrivaraporn et al., 2008; Horridge et al., 2008). The reasons of this choice lay in the nature of the ontologies treated by Semantic-Web applications, which spread on a wide range of languages and which are usually restricted in size but potentially extremely complex. In spite of that the need of design dedicated *white-box* (i.e. exploiting the specific knowledge on the structure of the handled problems) efficient procedures for $\mathcal{EL}^+$ is explained by the nature of the real-world $\mathcal{EL}^+$-ontologies: relatively simple in their structure but possibly huge in size. Thus, on the one hand exploiting the simple semantic of the input ontology should not be computationally expensive, on the other hand the huge dimension of the problem forces to exploit as much as possible the semantic of the input ontology and the properties of the underlying logic, developing

$\mathcal{EL}^+$-specific tools.

Moreover, the Description Logic community has spent a considerable effort in the attempt of extending $\mathcal{EL}$, defining a maximal subset of logical constructors expressive enough to cover the needs of the practical applications above mentioned, but whose inference problems remain tractable. Beside the logic $\mathcal{EL}^+$ (Baader et al., 2006b), on which we focus in this work, many other extension of $\mathcal{EL}$ or tractable fragments of even harder logics have been recently studied (Baader et al., 2005, 2008; Kazakov, 2009; Magka et al., 2010).

7.2 Motivations, Goals and Proposed Solution

In Chapters 5 and 6 we have proved how plain SAT solving and plain SMT, respectively, can be efficiently used to perform automated reasoning in Description Logics and ontologies. We have shown that exploiting the power of these techniques can be helpful in getting rid of some of the main weakness of the current DL reasoners, like the effect on reasoning of numerical constructors and, especially, the possibly huge size of the input problems. Moreover we have discussed how, nowadays, the large availability of cheep and large amounts of disk space and memory capabilities, can make profitable to move part of the reasoning in an encoding preprocessing (which, also, we experienced to be not particularly time expensive) into possibly larger problems but with a faster response on the single reasoning queries.

Starting from this considerations, in this last part of our research we want to investigate the effectiveness and applicability of more specific SAT/SMT-based technologies by approaching even more complex emerging non-standard reasoning services. In particular we aim at tackling one of those problems in which the existing general-purpose and highly-optimized systems can be better replaced by specific tools. As shown in the work of Suntisrivaraporn (2009), the previously described "hot" problem of debugging huge $\mathcal{EL}^+$ ontologies seems to perfectly fits our research purposes, and allows for evaluating the scalability of our approach on the concrete application of the huge real bio-medial ontologies.

The problem of axiom pinpointing in $\mathcal{EL}^+$ is characterized by two main characteristics: the simplicity of the logical constructors underlying the input ontologies and the potential huge dimensions of the problem. For this reason SAT seems to be a perfect tool able to solve simple problems of extremely huge dimensions. In particular, it is clearly evident the similarity between the axiom pinpointing problem in DL, consisting in identifying subsets of axioms causing an undesired inference, and Conflict Analysis in SAT which, indeed, aims at identifying a set of clauses/assumptions causing the unsatisfiability of the input formula (see Sections 4.1.2 and 4.1.3). Moreover, the identification of all such (minimal) subsets of axioms requires an iterative process. Therefore, we investigated the use of the All-SAT/All-SMT techniques (see Section 4.2.4 or Jin et al., 2005; Lahiri et al., 2006, for more details) applied in the framework of our proposed novel approach based on Boolean reasoning techniques.

In a nutshell, our idea is to build online a *polynomial-size* Horn propositional formula encoding the full classification of the input ontology. Then, we can find one MinA by applying SAT under assumption and by exploiting Conflict Analysis. Finally, we use the all-SMT approach in order to uniquely enumerate all the single MinAs computed in such a way. In this research stream, in order to increase the performance of axiom pinpointing, we also deal with the supplemental reasoning problem of modularization (as done by Baader & Suntisrivaraporn, 2008; Suntisrivaraporn, 2009). For its analogy with the problem of avoiding the state explosions in model checking we introduce a modularization techniques inspired by the well-known cone-of-influence reduction (Clarke, Grumberg, & Peled, 1999).

For the sake of the reader convenience we sketch the main elements of the approach we propose hereafter. In this chapter we build on previous work from the literature of $\mathcal{EL}^+$ reasoning (Baader et al., 2006b, 2007; Baader & Suntisrivaraporn, 2008) and of SAT and SMT (Biere et al., 2009; Barrett et al., 2009) and describe a simple and novel approach for (concept subsumption and) axiom pinpointing in $\mathcal{EL}^+$ and, hence, in its sub-logics $\mathcal{EL}$ and $\mathcal{ELH}$. We generate *polynomial-size* Horn propositional formulas representing part or all the deduction steps performed by the classification algorithms proposed by Baader et al. (2006b, 2007), and we manipulate them by exploiting the functionalities of modern SAT/SMT solvers, like: *Boolean Constraint Propagation (BCP)* (Moskewicz et al., 2001), *two-watched literals* (Moskewicz et al., 2001), *conflict analysis under assumptions* (Moskewicz et al., 2001; Eén & Sörensson, 2004), *All-SMT* (Lahiri et al., 2006), and *theory propagation* (e.g., Sebastiani, 2007b). In particular, we show that from an ontology $\mathcal{T}$ it is possible to generate in polynomial time Horn propositional formulas $\phi^{all}_{\mathcal{T}}$, and some restricted versions: $\phi_{\mathcal{T}}$, $\phi^{one}_{\mathcal{T}}$ and $\phi^{all}_{\mathcal{T}(po)}$, such that for every pair of primitive concepts C_i, D_i:

(i) concept subsumption is performed by one run of BCP on $\phi_{\mathcal{T}}$ or $\phi^{one}_{\mathcal{T}}$;

(ii) *one* non-minimal set of axioms (nMinA) responsible for the derivation of $C_i \sqsubseteq_{\mathcal{T}} D_i$ is computed by one run of BCP and conflict analysis on $\phi^{one}_{\mathcal{T}}$ or $\phi^{all}_{\mathcal{T}}/\phi^{all}_{\mathcal{T}(po)}$;

(iii) *one minimal* such set (MinA) is computed by iterating process (ii) on $\phi^{all}_{\mathcal{T}}/\phi^{all}_{\mathcal{T}(po)}$ for an amount of times up-to-linear in the size of the first nMinA found;

(iv) the same task of (iii) can also be computed by iteratively applying process (ii) on an up-to-linear sequence of increasingly-smaller formulas $\phi^{one}_{\mathcal{T}}$,$\phi^{one}_{\mathcal{S}_1}$,...,$\phi^{one}_{\mathcal{S}_k}$;

(v) *all* MinAs can be enumerated by means of All-SMT techniques on $\phi^{all}_{\mathcal{T}}/\phi^{all}_{\mathcal{T}(po)}$, using step (iii) as a subroutine.

It is worth noticing that (i) and (ii) are instantaneous even with huge formulas, and that (v) requires building a *polynomial-size* formula $\phi^{all}_{\mathcal{T}}/\phi^{all}_{\mathcal{T}(po)}$, in contrast to the exponential-size formula required by the Baader et al.'s (2007) all-MinAs process.

We have implemented a tool called $\mathcal{EL}^+$2SAT and performed an empirical evaluation on the available ontologies, whose results confirmed the potential of our novel approach (see also Sebastiani & Vescovi, 2009b). On these bases we refine our general approach

to the all-MinAs problem developing and implementing in the framework of our All-SMT procedure: a novel, fully SAT-based (so-called) *Cone-of-influence Modularization* technique relying on the two-watched literals schem Moskewicz et al., 2001) in order to reduce the search space during the all-MinAs enumeration, and a SMT-like *theory propagation* technique (e.g., Sebastiani, 2007b) which helps to early-discover MinAs during the enumeration. Then we further enhance our approach by exploiting $\mathcal{EL}^+$2SAT in a "three-phases" combined approach which yields to outstanding practical results. The idea is:

(vi) to perform the cone-of-influence module extraction in a fully SAT-based manner from $\phi^{all}_{\mathcal{T}}/\phi^{all}_{\mathcal{T}(po)}$, preserving axiom pinpointing in the query $C_i \sqsubseteq D_i$;

(vii) to use step (vi) in order to extract a module $\mathcal{M}$ specific for $C_i \sqsubseteq D_i$ and then to perform step (v) on the freshly-encoded (and orders-of-magnitude-smaller) formula $\phi^{all}_{\mathcal{M}(po)}$ in order to compute *all* the MinAs for $C_i \sqsubseteq D_i$.

Content.
This has been said, the rest of this chapter is structured as follows. In Section 7.3 we present the main previous approach to classification, modularization and axiom pinpointing in the logic $\mathcal{EL}^+$, on which we built our novel approach. In Section 7.4 we present our SAT-based procedures for concept subsumption and axiom pinpointing (both for extracting one or for enumerating all the MinAs), and we discuss our technique in comparison with the one presented in Section 7.3. In Section 7.5 we have a preliminary empirical evaluation of this novel approach implemented in the $\mathcal{EL}^+$2SAT tool. These first sections can be seen as the preliminary presentation of our method.

In the second part we significantly push the envelope of our approach. In Section 7.6 we introduce the above mentioned optimization techniques (i.e. modularization, theory propagation and combined approach) which strongly speed up the search of the MinAs in our approach In Section 7.7 we extensively evaluate our approach and the single implemented optimization techniques by mean of exhaustive debugging queries on the real-world benchmark ontologies mentioned in Section 7.1. In our empirical evaluation we also compare with the other $\mathcal{EL}^+$-specific state-of-the-art tool CEL (Baader et al., 2006a; Suntisrivaraporn, 2009). (For the sake of readability we move the less prominent part of the empirical results in appendix of the chapter, Section 7.10.) Finally, in Section 7.8 we summarize the main innovative contributions and we outline directions for future research on this topic. Section 7.9, in appendix, contains the proofs of all the theoretical results presented along the chapter.

The preliminary part of this reasearch, including the basic encoding and technique for concept subsumption and axiom pinpointing and the preliminary evaluation, have been published by (Sebastiani & Vescovi, 2009b). A complete work presenting the overall method, including modularization, the combined approach, all the other optimizations and the extensive empirical evaluation is currently under submission to a journal (Sebastiani & Vescovi, 2011).

7.3 Classification and Axiom Pinpointing in $\mathcal{EL}^+$ so far

In this section we overview the main notions concerning concept subsumption, classification, and axiom pinpointing in $\mathcal{EL}^+$ (Section 3.5.1), by exposing the main previous $\mathcal{EL}^+$-specific approach from the literature. We build our method on this previous work, thus the content of this section is fundamental for the best comprehension of the rest of this chapter and of our novel approach. In our approach we inherit part of the notions exposed in the following, while later in this chapter we will compare our method with this previous approach by pointing out the main differences and innovations.

7.3.1 A Normal Form for $\mathcal{EL}^+$

In $\mathcal{EL}^+$ it is convenient to establish and work with a *normal form* of the input problem, which helps to make explanations, proofs, reasoning rules and algorithms simpler and more general. Usually the following normal form for the $\mathcal{EL}^+$ TBoxes is considered (Baader et al., 2005, 2006b; Baader & Peñaloza, 2007; Baader et al., 2007):

$$(C_1 \sqcap ... \sqcap C_k) \sqsubseteq D \qquad k \geq 1 \tag{7.1}$$

$$C \sqsubseteq \exists r.D \tag{7.2}$$

$$\exists r.C \sqsubseteq D \tag{7.3}$$

$$r_1 \circ \cdots \circ r_n \sqsubseteq s \qquad n \geq 1 \tag{7.4}$$

such that $C_1, \ldots, C_k, D \in \mathsf{PC}_{\mathcal{T}}$ and $r_1, \ldots, r_n, s \in {N_R}^{\mathcal{T}}$. A TBox $\mathcal{T}$ can be turned into a normalized TBox $\mathcal{T}'$ that is a conservative extension of $\mathcal{T}$ (Baader et al., 2005), by introducing new concept names. In a nutshell, normalization consists in rewriting axioms in the form $C \sqsubseteq C_1 \sqcap \ldots \sqcap C_n$ into $C \sqsubseteq C_1$, $\ldots$, $C \sqsubseteq C_n$, and in substituting, when needed, instances of complex concepts of the forms $\exists r.C$ and $C_1 \sqcap ... \sqcap C_k$ with fresh concept names (namely, C' and C''), adding the axioms $C' \sqsubseteq \exists r.C$ [resp. $\exists r.C \sqsubseteq C'$] and $C'' \sqsubseteq C_1, \ldots, C'' \sqsubseteq C_k$ [resp. $(C_1 \sqcap \ldots \sqcap C_k) \sqsubseteq C''$] for every complex concept substituted in the right [resp. left] part of an axiom. The normal TBox $\mathcal{T}'$ resulting from this process is composed of two kinds of axioms:

- some *top-level* axioms representing the original axioms of $\mathcal{T}$ (with complex sub-concepts substituted by the fresh concept names);
- some *definition* axioms representing the labeling of complex sub-concepts with the newly introduced concept names.

If a complex concept appears on the same side (left or right) of different axioms only one definition axiom is necessary.

Normalization can be performed in linear time wrt. the size of $\mathcal{T}$, and the size of $\mathcal{T}'$ is linear wrt. that of $\mathcal{T}$ (Baader et al., 2005). We call *normal concept* of a normal TBox $\mathcal{T}'$ every non-conjunctive concept description occurring in the concept inclusions of $\mathcal{T}'$; we call $\mathsf{NC}'_{\mathcal{T}}$ the set of all the normal concepts of $\mathcal{T}'$. (I.e., the set $\mathsf{NC}'_{\mathcal{T}}$ consists in all the concepts of the form C or $\exists r.C$, with $C \in \mathsf{PC}_{\mathcal{T}'}$ and $r \in {N_R}^{\mathcal{T}'}$.)

Example 7.3.1. The following set of axioms, that we call $\mathcal{O}_{\mathsf{milk}}$, is adapted from a fragment of the ontology NOT-GALEN, and represents some facts and relationships concerning the concept Milk:

$$\begin{aligned}
\mathsf{BodyFluid} &\sqsubseteq \mathsf{Fluid} && m_1\\
\mathsf{Liquid} &\sqsubseteq \mathsf{Fluid} && m_2\\
\mathsf{BodyFluid} &\equiv \mathsf{BodySubstance} \sqcap \exists \mathsf{hasPhysicalState.liquidState} && m_3\\
\mathsf{BodySubstance} &\sqsubseteq \mathsf{Substance} && m_4\\
\mathsf{Milk} &\sqsubseteq \mathsf{BodySubstance} && m_5\\
\mathsf{Milk} &\sqsubseteq \exists \mathsf{hasPhysicalState.liquidState} && m_6\\
\mathsf{Milk} &\sqsubseteq \exists \mathsf{isActedOnSpecificallyBy}.(\mathsf{Secretion} \sqcap \exists \mathsf{isFunctionOf.Breast}) && m_7\\
\mathsf{SecretedSubstance} &\equiv \mathsf{Substance} \sqcap \exists \mathsf{isActedOnBy.Secretion} && m_8\\
\mathsf{Liquid} &\equiv \mathsf{Substance} \sqcap \exists \mathsf{hasPhysicalState.liquidState} && m_9\\
\mathsf{liquidState} &\equiv \mathsf{PhysicalState} \sqcap \exists \mathsf{hasState.liquid} && m_{10}\\
\mathsf{isActedOnSpecificallyBy} &\sqsubseteq_r \mathsf{isActedOnBy} && m_{11}
\end{aligned}$$

We consider the normalization of axioms m_3 and m_7. Since m_3 is an equivalence, it is split into a couple of inclusion axioms:

$$\begin{aligned}
\mathsf{BodyFluid} &\sqsubseteq \mathsf{BodySubstance} \sqcap \exists \mathsf{hasPhysicalState.liquidState} && m_{3a}\\
\mathsf{BodySubstance} \sqcap \exists \mathsf{hasPhysicalState.liquidState} &\sqsubseteq \mathsf{BodyFluid} && m_{3b}.
\end{aligned}$$

Then we split m_{3a} into two distinct axioms and introduce a primitive concept N labeling $\exists$hasPhysicalState.liquidState. The resulting normalization is:

$$\begin{aligned}
\mathsf{BodyFluid} &\sqsubseteq \mathsf{BodySubstance}\\
\mathsf{BodyFluid} &\sqsubseteq \mathsf{N}\\
\mathsf{N} &\sqsubseteq \exists \mathsf{hasPhysicalState.liquidState}\\
\mathsf{BodySubstance} \sqcap \mathsf{N} &\sqsubseteq \mathsf{BodyFluid}\\
\exists \mathsf{hasPhysicalState.liquidState} &\sqsubseteq \mathsf{N}.
\end{aligned}$$

The first, second and fourth axioms are top-level ones, whilst the third and fifth are definitions of N. The normalization of m_7, instead, requires the introduction of another fresh concept name M labeling (Secretion $\sqcap$ $\exists$isFunctionOf.Breast). Then, the definition of M is split in two axioms:

$$\begin{aligned}
\mathsf{Milk} &\sqsubseteq \exists \mathsf{isActedOnSpecificallyBy.M}\\
\mathsf{M} &\sqsubseteq \mathsf{Secretion}\\
\mathsf{M} &\sqsubseteq \exists \mathsf{isFunctionOf.Breast}.
\end{aligned}$$

$\diamond$

Subsumption assertions ($\ldots \in \mathcal{A}$)	**TBox's axioms** ($\ldots \in \mathcal{T}$)	... added to $\mathcal{A}$
$X \sqsubseteq C_1$, $X \sqsubseteq C_2$, $\ldots$, $X \sqsubseteq C_k$ $\quad k \geq 1$	$C_1 \sqcap \cdots \sqcap C_k \sqsubseteq D$	$X \sqsubseteq D$
$X \sqsubseteq C$	$C \sqsubseteq \exists r.D$	$X \sqsubseteq \exists r.D$
$X \sqsubseteq \exists r.E$, $E \sqsubseteq C$	$\exists r.C \sqsubseteq D$	$X \sqsubseteq D$
$X \sqsubseteq \exists r.D$	$r \sqsubseteq s$	$X \sqsubseteq \exists s.D$
$X \sqsubseteq \exists r_1.E_1$, $\ldots$, $E_{n-1} \sqsubseteq \exists r_n.D$ $\quad n \geq 2$	$r_1 \circ \cdots \circ r_n \sqsubseteq s$	$X \sqsubseteq \exists s.D$

Table 7.1: Completion rules of the concept subsumption algorithm for $\mathcal{EL}^+$. A rule reads as follows: if the assertions/axioms in the left column belong to $\mathcal{A}$, the GCI/RI of the central column belongs to $\mathcal{T}$, and the assertion of the right column is not already in $\mathcal{A}$, then the assertion of the right column is added to $\mathcal{A}$.

7.3.2 Concept Subsumption in $\mathcal{EL}^+$.

Given a normalized TBox $\mathcal{T}$ over the set of primitive concepts $\mathsf{PC}_\mathcal{T}$ and the set of primitive roles ${N_R}^\mathcal{T}$, the subsumption algorithm for $\mathcal{EL}^+$ proposed by Baader et al. (2007) generates and extends a set $\mathcal{A}$ of assertions through the completion rules defined in Table 7.1. By "*assertion*" we mean every known or deduced subsumption relation between normal concepts of the TBox $\mathcal{T}$. The algorithm starts with the initial set $\mathcal{A} = \{a_i \in \mathcal{T} \mid a_i \text{ is a GCI}\} \cup \{C \sqsubseteq C,\ C \sqsubseteq \top \mid C \in \mathsf{PC}_\mathcal{T}\}$ and extends $\mathcal{A}$ using the rules of Table 7.1, until no more assertions can be added. (Notice that a rule is applied only if it extends $\mathcal{A}$.) We call *propositional completion rules* the first two completion rules in Table 7.1, and *non-propositional completion rules* the other three rules.

Example 7.3.2. We report all the subsumption relations that can be inferred in $\mathcal{O}_{\mathsf{med}}$ (Example 3.5.1) from its original axioms. Once $\mathcal{O}_{\mathsf{med}}$ is turned into normal form its full classification $\mathcal{A}_{\mathcal{O}_{\mathsf{med}}}$ is the following:

$\mathsf{Appendix} \sqsubseteq \mathsf{BodyPart}$	a'_1	$\mathsf{Appendix} \sqsubseteq \exists \mathsf{partOf}.\mathsf{Intestine}$	a''_1
$\mathsf{Endocardium} \sqsubseteq \mathsf{Tissue}$	a'_2	$\mathsf{Endocardium} \sqsubseteq \exists \mathsf{partOf}.\mathsf{HeartValve}$	a''_2
$\mathsf{Pericardium} \sqsubseteq \mathsf{Tissue}$	a'_3	$\mathsf{Pericardium} \sqsubseteq \exists \mathsf{containedIn}.\mathsf{Heart}$	a''_3
$\mathsf{Appendicitis} \sqsubseteq \mathsf{Inflammation}$	a'_4	$\mathsf{Appendicitis} \sqsubseteq \exists \mathsf{hasLoc}.\mathsf{Appendix}$	a''_4
$\mathsf{Endocarditis} \sqsubseteq \mathsf{Inflammation}$	a'_5	$\mathsf{Endocarditis} \sqsubseteq \exists \mathsf{hasLoc}.\mathsf{Endocardium}$	a''_5
$\mathsf{Pericarditis} \sqsubseteq \mathsf{Inflammation}$	a'_6	$\mathsf{Pericarditis} \sqsubseteq \exists \mathsf{hasLoc}.\mathsf{Pericardium}$	a''_6
$\mathsf{Inflammation} \sqsubseteq \mathsf{Disease}$	a'_7	$\mathsf{Inflammation} \sqsubseteq \exists \mathsf{actsOn}.\mathsf{Tissue}$	a''_7
$\mathsf{Disease} \sqcup \mathsf{New} \sqsubseteq \mathsf{HeartDisease}$	a'_8	$\exists \mathsf{hasLoc}.\mathsf{Heart} \sqsubseteq \mathsf{New}$	a_0
$\mathsf{HeartDisease} \sqsubseteq \exists \mathsf{hasState}.\mathsf{NeedsTreat.}$	a_9		
$\mathsf{Appendicitis} \sqsubseteq \mathsf{Disease}$	b_1 $r_1(a'_4, a'_7)$	$\mathsf{Appendicitis} \sqsubseteq \exists \mathsf{actsOn}.\mathsf{Tissue}$	b_2 $r_2(a'_4, a''_7)$
$\mathsf{Endocarditis} \sqsubseteq \mathsf{Disease}$	b_3 $r_1(a'_5, a'_7)$	$\mathsf{Endocarditis} \sqsubseteq \exists \mathsf{actsOn}.\mathsf{Tissue}$	b_4 $r_2(a'_5, a''_7)$
$\mathsf{Pericarditis} \sqsubseteq \mathsf{Disease}$	b_5 $r_1(a'_6, a'_7)$	$\mathsf{Pericarditis} \sqsubseteq \exists \mathsf{actsOn}.\mathsf{Tissue}$	b_6 $r_2(a'_6, a''_7)$
$\mathsf{Pericarditis} \sqsubseteq \exists \mathsf{hasLoc}.\mathsf{Heart}$	b_7 $r_5(a''_6, a''_3, a_{11})$	$\mathsf{Pericarditis} \sqsubseteq \mathsf{New}$	b_8 $r_3(b_7, a_0)$
$\mathsf{Pericarditis} \sqsubseteq \mathsf{HeartDisease}$	b_9 $r_1(b_5, b_8, a'_8)$	$\mathsf{Pericarditis} \sqsubseteq \exists \mathsf{hasState}.\mathsf{NeedsTreat.}$	b_{10} $r_2(b_9, a_9)$

In particular, the first seventeen GCIs ($a'_1, a''_1, \ldots, a'_8, a_0, a_9$ in the first nine rows) plus the two RIs $a_{10} \stackrel{\text{def}}{=} \mathsf{partOf} \circ \mathsf{partOf} \sqsubseteq \mathsf{partOf}$ and $a_{11} \stackrel{\text{def}}{=} \mathsf{hasLocation} \circ \mathsf{containedIn} \sqsubseteq \mathsf{hasLocation}$ compose the normalization of $\mathcal{O}_{\mathsf{med}}$. We labeled a'_i (and a''_i) the normal-form top-level axiom(s) resulting from the normalization of the original axiom a_i (see Example 3.5.1),

while we labeled a_0 the new definition axiom introduced in order to normalize a_8. Next we labeled b_j every other subsumption relation (assertion) inferred through the classification algorithm above exposed, with $j = 1, 2, \ldots 10$ following the inference order of the algorithm. In particular, for each new assertion inferred we show the index of the completion rule of Table 7.1 applied (r_1 for the first rule, r_2 for the second, and so on and so forth) and the label of its necessary premises in between parenthesis. For instance, the new assertion b_9 is inferred applying a ternary instance of the first completion rule of Table 7.1, where the premises of the rule are the other assertions b_5, b_8 and the axiom a'_8. Finally, notice that three premises are necessary in order to infer b_8 via the third completion rule r_3, but the second assertion is the trivial inclusion $\mathsf{Heart} \sqsubseteq \mathsf{Heart}$. ◇

Baader et al. (2005) proved the soundness and the completeness of the algorithm together with the fact that the algorithm terminates after polynomially-many rule applications, each of which can be performed in polynomial time. Intuitively, since the number of concept and role names is linear in the size of the input TBox, the algorithm cannot add to $\mathcal{A}$ more than the cardinality of $\mathsf{PC}_\mathcal{T} \times \mathsf{PC}_\mathcal{T} \times {N_R}^\mathcal{T}$ assertions. Thus, since no rule removes assertions from $\mathcal{A}$, the algorithm stops after at most a polynomial number of rule applications. Moreover, every rule application can be performed in polynomial time.

Once a complete classification of the normalized TBox is computed and stored in some ad-hoc data structure, if $C, D \in \mathsf{PC}_\mathcal{T}$, then $C \sqsubseteq_\mathcal{T} D$ iff the pair C, D can be retrieved from the latter structure. The problem of computing $X \sqsubseteq_\mathcal{T} Y$ s.t. $X, Y \notin \mathsf{PC}_\mathcal{T}$ can be reduced to that of computing $C \sqsubseteq_{\mathcal{T} \cup \{C \sqsubseteq X, Y \sqsubseteq D\}} D$, s.t. C and D are two new concept names.

Axiom Pinpointing in $\mathcal{EL}^+$.

We recall one important definition (Baader et al., 2007).

Definition 7 (nMinA, MinA)**.** Consider the subsumption relation $C_i \sqsubseteq_\mathcal{T} D_i$, with $C_i, D_i \in \mathsf{PC}_\mathcal{T}$. If $C_i \sqsubseteq_\mathcal{S} D_i$ for some set $\mathcal{S} \subseteq \mathcal{T}$ of axioms, then $\mathcal{S}$ is called an *axiom set (nMinA)* for $C_i \sqsubseteq D_i$ wrt. $\mathcal{T}$. If $C_i \not\sqsubseteq_{\mathcal{S}'} D_i$ for every $\mathcal{S}'$ s.t. $\mathcal{S}' \subset \mathcal{S}$, then $\mathcal{S}$ is called a *minimal axiom set (MinA)* for $C_i \sqsubseteq D_i$ wrt. $\mathcal{T}$.

Example 7.3.3. In the ontology $\mathcal{O}_{\mathsf{milk}}$ of Example 7.3.1, a MinA for $\mathsf{Milk} \sqsubseteq \mathsf{SecretedSubstance}$ wrt. $\mathcal{O}_{\mathsf{milk}}$ is given by the original axioms $\{m_4, m_5, m_7, m_8, m_{11}\}$. [2] In particular, m_4 and m_5 are necessary to infer $\mathsf{Milk} \sqsubseteq \mathsf{Substance}$, while from (the normalization of) m_7 and m_{11}, it follows that $\mathsf{Milk} \sqsubseteq \exists \mathsf{isActedOnBy.Secretion}$. Finally $\mathsf{Milk} \sqsubseteq \mathsf{SecretedSubstance}$ can be inferred from the two previous premises and the definition of $\mathsf{SecretedSubstance}$ in m_8. ◇

Baader et al. (2007) proposed a technique for computing all MinAs for $C_i \sqsubseteq_\mathcal{T} D_i$ wrt. $\mathcal{T}$: during the classification of $\mathcal{T}$, a *pinpointing formula* (namely $\Phi^{C_i \sqsubseteq_\mathcal{T} D_i}$) is built, which is a monotone propositional formula [3] on the set of variables $\mathcal{P}_\mathcal{T} \stackrel{\text{def}}{=} \{s_{[ax_j]} \mid ax_j \in \mathcal{T}\}$ s.t.,

[2] This is the only MinA for this subsumption in $\mathcal{O}_{\mathsf{milk}}$.

[3] A *monotone* propositional formula is a propositional formula whose only connectives are $\wedge$ and $\vee$.

for every $\mathcal{O} \subseteq \mathcal{T}$, $\mathcal{O}$ is a MinA for $C_i \sqsubseteq_\mathcal{T} D_i$ iff $\{s_{[ax_i]} \mid ax_i \in \mathcal{O}\}$ is a minimal valuation of $\Phi^{C_i \sqsubseteq_\mathcal{T} D_i}$. In a nutshell, the process of building $\Phi^{C_i \sqsubseteq_\mathcal{T} D_i}$ works as follows. Every axiom $ax_j \in \mathcal{T}$ is encoded with a propositional variable $s_{[ax_j]}$ and every deduced assertion $a_j \in \mathcal{A}$ is encoded into a monotone propositional formula. Consider each application of a completion rule r during the classification of $\mathcal{T}$, and let ϕ_r be the conjunction of the labels (i.e. variables and monotone formulas) of the axioms and the assertions in the preconditions of r (respectively). If a new assertion is deduced as consequence of r, then it is labeled with the formula ϕ_r. Otherwise, if the assertion in the consequence of r is already in $\mathcal{A}$ and it is labeled with ψ, then its label is updated with $\psi^* = \psi \vee \phi_r$, unless $\phi_r \models \psi$.

The Baader et al.'s (2007) all-MinAs algorithm consists thus in (i) building $\Phi^{C_i \sqsubseteq_\mathcal{T} D_i}$ and (ii) computing all minimal valuations of $\Phi^{C_i \sqsubseteq_\mathcal{T} D_i}$. This algorithm, however, has serious limitations in terms of complexity: first, the algorithm for generating $\Phi^{C_i \sqsubseteq_\mathcal{T} D_i}$ requires intermediate logical checks, each of them involving the solution of an NP-complete problem; second, the size of $\Phi^{C_i \sqsubseteq_\mathcal{T} D_i}$ can be exponential wrt. that of $\mathcal{T}$ (Baader et al., 2007). More generally, Baader et al. (2007) proved also that there is no output-polynomial algorithm for computing all MinAs (unless P=NP). (To the best of our knowledge, there is no publicly-available implementation of the all-MinAs algorithm above.) Consequently, Baader et al. (2007) concentrated the effort on finding polynomial algorithms for finding *one* MinA at a time, proposing a linear-search minimization algorithm which allowed for finding MinAs for FULL-GALEN efficiently. This technique was further improved by Baader and Suntisrivaraporn (2008) by means of a binary-search minimization algorithm, and by a novel algorithm exploiting the notion of *reachability-modules*, which allowed to efficiently find MinAs for the much bigger SNOMED-CT ontology. We refer the readers to the literature (Baader et al., 2007; Baader & Suntisrivaraporn, 2008) for a detailed description.

Furthermore, in a recent work, Suntisrivaraporn, 2009 solved the all-MinAs problem with a different approach based on the techniques of the Hitting Set Tree (HST), where the universal set is the whole ontology and the set of the all MinAs is collection of the minimal subsets to be found. In particular the hitting set tree is expanded along the algorithm computing, at the end, all the MinAs for the given subsumption. In this approach the optimized algorithm and the linear minimization algorithm exposed above are used as subroutines to initialize the algorithm and to minimize the resulting sets respectively. However, also this techniques has the major drawback of performance in large-scale ontologies.

Therefore, the above technique has been implemented in combination with the *reachability-based module-extraction technique* (Suntisrivaraporn, 2009), which drastically reduces the search space of the HST algorithm. We briefly describe such technique in Section 7.3.3, referring the readers to the literature (Suntisrivaraporn, 2009) for a much more detailed explanation.

7.3.3 Axiom Pinpointing with Reachability-based Modularization in $\mathcal{EL}^+$

Real-world (medical) $\mathcal{EL}^+$ ontologies are often huge in size (e.g., SNOMED-CT'09 has more than 300,000 axioms). Thus, despite the low complexity of the pinpointing problem in $\mathcal{EL}^+$, the handling of such ontologies is out of the reach of the pinpointing algorithms described in previous sections.

Luckily, for a given subsumption assertion $a_i \stackrel{\text{def}}{=} C_i \sqsubseteq D_i$ which can be derived from $\mathcal{T}$, it is typically the case that only a minority of the axioms in $\mathcal{T}$ may have any role in any derivation of a_i. Thus, a key idea is to identify a priori a strict subset $\mathcal{M}_{a_i}$ of $\mathcal{T}$ which is sufficient to perform every possible derivation of a_i. (Hence $\mathcal{M}_{a_i}$ contains every MinA for a_i.) Concretely, the idea is to exploit *modularization* (see Section 3.3.2) in order to extract a *subsumption* (or *axiom pinpointing*) *preserving module* $\mathcal{M}_{a_i}$.

In particular, in order to improve the efficiency of axiom-pinpointing algorithms in $\mathcal{EL}^+$ described in previous sections, Baader and Suntisrivaraporn (2008) proposed the syntactic *reachability-based modularization* technique (see also Suntisrivaraporn, 2009) that has been lately extended to harder logics by Suntisrivaraporn et al. (2008).

We recall from the work of Suntisrivaraporn (2009) the basic facts about reachability-based modularization.

Definition 8 (Σ-reachable symbols, axioms, reachability-based module)**.** Given an $\mathcal{EL}^+$ TBox $\mathcal{T}$ and a signature $\Sigma \subseteq \mathsf{signature}(\mathcal{T})$, the set of the Σ-*reachable symbols* of $\mathcal{T}$ is recursively defined as follows:

(i) every symbol $x \in \Sigma$ is Σ-reachable;

(ii) for every axiom $\hat{C} \sqsubseteq \hat{D}$ of $\mathcal{T}$, if all the symbols in $\mathsf{signature}(\hat{C})$ are Σ-reachable, then all symbols $y \in \mathsf{signature}(\hat{D})$ are Σ-reachable.

Given the set of the Σ-reachable symbols, an axiomn $\hat{C} \sqsubseteq \hat{D} \in \mathcal{T}$ is a Σ-*reachable axiom* of $\mathcal{T}$ if x is Σ-reachable for every symbol $x \in \mathsf{signature}(\hat{C})$. The Σ-*reachability-based module* for Σ in $\mathcal{T}$, denoted by $\mathcal{M}_\Sigma^{\mathsf{reach}}$, is the set of all the Σ-reachable axioms of $\mathcal{T}$. $\diamond$

(With a little abuse of notation, if Σ consists only of a single concept name C, then we denote its reachability-based module by $\mathcal{M}_C^{\mathsf{reach}}$ rather then by $\mathcal{M}_{\{C\}}^{\mathsf{reach}}$.)

Example 7.3.4. Consider again the ontology $\mathcal{O}_{\mathsf{med}}$ in Example 3.5.1. The reachability-based module for the signature $\Sigma_{\mathsf{Pericarditis}}$ is $\mathcal{M}^{\mathsf{reach}}_{\mathsf{Pericarditis}} = \{a_3, a_6, a_7, a_8, a_9, a_{11}\}$. In fact, starting from the symbol Pericarditis, axiom a_6 is included in the module and the symbols Pericarditis, Inflammation, hasLocation and Pericardium are marked as $\Sigma_{\mathsf{Pericarditis}}$-reachable. From Pericardium and Inflammation, axioms a_3 and a_7 are included in the module and hence Tissue, containedIn, Heart Disease, actsOn are added to the $\Sigma_{\mathsf{Pericaridits}}$-reachable symbols. The three left-side symbols of a_8 (i.e. Disease, hasLocation, Heart) are now $\Sigma_{\mathsf{Pericaridits}}$-reachable, so that a_8 is also added to the module. Hence HearthDisease becomes $\Sigma_{\mathsf{Pericaridits}}$-reachable, so that a_9 is added to the module, making HasState, NeedsTreatment $\Sigma_{\mathsf{Pericaridits}}$-reachable. Moreover, since both containedIn and hasLocation are $\Sigma_{\mathsf{Pericaridits}}$-reachable, then also a_{11} is added to the module. No other axiom can then be added to the module. $\diamond$

Example 7.3.5. Second, consider the ontology $\mathcal{O}_{\mathsf{milk}}$ (Example 7.3.1) the reachability module $\mathcal{M}^{\mathsf{reach}}_{\mathsf{Milk}}$ for the single concept Milk in $\mathcal{O}_{\mathsf{milk}}$ is exactly the whole ontology $\mathcal{O}_{\mathsf{milk}}$. This fact can be quickly and trivially checked. In fact all the axioms and all the symbols contained in $\mathcal{O}_{\mathsf{milk}}$ are Milk-reachable, starting from the axioms m_5, m_6 and m_7 in which the concept Milk is the only symbol at the left side of the inclusion, and then iteratively reaching every other axiom. ◇

Intuitively, an axiom ax of $\mathcal{T}$ is included in the reachiability-based module $\mathcal{M}^{\mathsf{reach}}_{\Sigma}$ for Σ if and only if the symbols of Σ syntactically refer to the symbols in ax, either directly or indirectly via other axioms of $\mathcal{T}$. All the axioms which are thus "syntactically connected" to Σ are included in the reachability-based module for Σ.

Notice that the reachability-based modularization is a purely *syntactic* technique, because the semantic of the axioms and of the operators involved are not considered in the construction of the modules. Moreover, notice that this modularization techiniques is fully indipendent from the completion rules used in the classification of $\mathcal{T}$. In the following we will refer to this technique either calling it reachability-based modularization or (more generically) syntactic modularization.

Property 1. Let Σ be a signature on the TBox $\mathcal{T}$, and let $\hat{C}, \hat{D}$ be arbitrary $\mathcal{EL}^+$ concept descriptions such that $\mathsf{signature}(\hat{C}) \subseteq \Sigma$. Then $\hat{C} \sqsubseteq_{\mathcal{T}} \hat{D}$ if and only if $\hat{C} \sqsubseteq_{\mathcal{M}^{\mathsf{reach}}_{\Sigma}} \hat{D}$. ◇

Thus, for every subsumption relation $C \sqsubseteq_{\mathcal{T}} D$, the process of axiom pinpointing plus reachability-based modularization for a_i consists in:

(i) computing the reachability-based module $\mathcal{M}^{\mathsf{reach}}_{C}$,

(ii) applying the axiom pinpointing algorithm to $\mathcal{M}^{\mathsf{reach}}_{C}$ instead than to $\mathcal{T}$.

Suntisrivaraporn (2009) computes reachability-based modules through a queue-based algorithm which iteratively adds axioms to the initially empty module, starting from the given input signature. The algorithm is shown to be quadratic wrt. $|\mathcal{T}|$. However, if $|\mathcal{M}_{a_i}| << |\mathcal{T}|$ (as it is often the case), then the modularizatin process can drastically improve the efficiency of the pinpointing process.

7.4 Axiom Pinpointing via Horn SAT and Conflict Analysis

In this section we start presenting our novel contributions. In order not to break the flow of the discourse, the proofs of the new results have been moved to Appendix 7.9.

7.4.1 Classification and Concept Subsumption via Horn SAT solving

We temporarily assume that $\mathcal{T}$ is the result of a normalization process, as described in Section 7.3. (We will consider some issues related to the normalization at the end of Section 7.4.2.) We consider first the problem of concept subsumption. We build a Horn propositional formula $\phi_{\mathcal{T}}$ representing the classification of the input ontology $\mathcal{T}$.

Definition 9 ($\mathcal{EL}^+2sat(a_i)$, $\phi_\mathcal{T}$)**.** Let $\mathcal{T}$ be an $\mathcal{EL}^+$ TBox in normal form and let $\mathcal{A}$ be the classification of $\mathcal{T}$. We introduce the set of propositional variables $\{p_{[X]} \mid X \in \mathsf{NC}_\mathcal{T}\}$, that we call *concept variables*, s.t. each concept variable $p_{[X]}$ is uniquely-associated to the respective concept X. For each assertion $a_i \in \mathcal{A}$, we define the *propositional encoding of* a_i, written $\mathcal{EL}^+2sat(a_i)$, as follows:

$$\mathcal{EL}^+2sat(a_i) \stackrel{\text{def}}{=} \begin{array}{rll} p_{[C_1]} \wedge ... \wedge p_{[C_k]} \rightarrow p_{[D]} & k \geq 1 & \text{if } a_i \text{ is of type (7.1);} \quad (7.5) \\ p_{[C]} \rightarrow p_{[\exists r.D]} & & \text{if } a_i \text{ is of type (7.2);} \quad (7.6) \\ p_{[\exists r.C]} \rightarrow p_{[D]} & & \text{if } a_i \text{ is of type (7.3).} \quad (7.7) \end{array}$$

Then we define the following CNF Horn propositional formula:

$$\phi_\mathcal{T} \stackrel{\text{def}}{=} \bigwedge_{a_i \in \mathcal{A}} \mathcal{EL}^+2sat(a_i). \quad (7.8)$$

◇

Notice that we do not encode trivial axioms of the form $C \sqsubseteq C$ and $C \sqsubseteq \top$ because they generate valid clauses $p_{[C]} \rightarrow p_{[C]}$ and $p_{[C]} \rightarrow \top$.

Since the clauses (7.5)-(7.7) are definite Horn clauses, $\phi_\mathcal{T}$ is a definite Horn formula. Thus, $\phi_\mathcal{T}$ is satisfiable, and it is necessary to conjoin it with at least one positive and one negative literal in order to make it unsatisfiable.

Theorem 13. *Given an $\mathcal{EL}^+$ TBox $\mathcal{T}$ in normal form, for every pair of concept names C, D in $\mathsf{PC}_\mathcal{T}$, $C \sqsubseteq_\mathcal{T} D$ if and only if the Horn propositional formula $\phi_\mathcal{T} \wedge p_{[C]} \wedge \neg p_{[D]}$ is unsatisfiable.*

In practice, in order to build $\phi_\mathcal{T}$, we initially set it to an empty set of clauses; then we run the classification algorithm of Section 7.3: for every (non-trivial) original axiom of $\mathcal{T}$ or every deduced assertion a_i of the form (7.1)-(7.3) which is added to $\mathcal{A}$, we add to $\phi_\mathcal{T}$ the clause $\mathcal{EL}^+2sat(a_i)$.

Remark 2. Notice that, since the classification algorithm of Section 7.3 terminates after a polynomial number of rule applications and $|\mathcal{A}| \leq |\mathsf{PC}_\mathcal{T}|^2 \cdot |{N_R}^\mathcal{T}|$ (see Section 7.3.2), $\phi_\mathcal{T}$ is worst-case polynomial in size wrt. $|\mathcal{T}|$ and can be generated in polynomial time. ◇

Once $\phi_\mathcal{T}$ has been generated, in order to perform concept subsumption, we exploit the technique of CDCL SAT solving under assumptions described in Section 4.1: once $\phi_\mathcal{T}$ is parsed and `DPLL` is initialized, each subsumption query $C_i \sqsubseteq_\mathcal{T} D_i$ corresponds to solving $\phi_\mathcal{T}$ under the assumption list $\mathcal{L}_i \stackrel{\text{def}}{=} \{p_{[C_i]}, \neg p_{[D_i]}\}$. This corresponds to one single run of `bcp`, whose cost depends linearly only on the clauses where the unit-propagated literals occur. In practice, if $C_i \sqsubseteq_\mathcal{T} D_i$ then $\phi_\mathcal{T}$ contains the clause $p_{[C_i]} \rightarrow p_{[D_i]}$, so that `bcp` stops as soon as $p_{[C_i]}$ and $\neg p_{[D_i]}$ are unit-propagated.

Example 7.4.1. Consider the classification $\mathcal{A}_{\mathcal{O}_{\mathsf{med}}}$ in Example 7.3.2. [4] Then we have:

$$
\begin{array}{lrcl}
\phi_{\mathcal{O}_{\mathsf{med}}} \stackrel{\text{def}}{=} & p_{[\mathsf{Appendix}]} \rightarrow p_{[\mathsf{BodyPart}]} & a'_1 & \wedge \quad p_{[\mathsf{Appendix}]} \rightarrow p_{[\exists \mathsf{partOf.Intestine}]} \quad a''_1 \\
\wedge & p_{[\mathsf{Endocardium}]} \rightarrow p_{[\mathsf{Tissue}]} & a'_2 & \wedge \quad p_{[\mathsf{Endocardium}]} \rightarrow p_{[\exists \mathsf{partOf.HeartValve}]} \quad a''_2 \\
\wedge & p_{[\mathsf{Pericardium}]} \rightarrow p_{[\mathsf{Tissue}]} & a'_3 & \wedge \quad p_{[\mathsf{Pericardium}]} \rightarrow p_{[\exists \mathsf{containedIn.Heart}]} \quad a''_3 \\
\wedge & p_{[\mathsf{Appendicitis}]} \rightarrow p_{[\mathsf{Inflammation}]} & a'_4 & \wedge \quad p_{[\mathsf{Appendicitis}]} \rightarrow p_{[\exists \mathsf{hasLocation.Appendix}]} \quad a''_4 \\
\wedge & p_{[\mathsf{Endocarditis}]} \rightarrow p_{[\mathsf{Inflammation}]} & a'_5 & \wedge \quad p_{[\mathsf{Endocarditis}]} \rightarrow p_{[\exists \mathsf{hasLocation.Endocardium}]} \quad a''_5 \\
\wedge & p_{[\mathsf{Pericarditis}]} \rightarrow p_{[\mathsf{Inflammation}]} & a'_6 & \wedge \quad p_{[\mathsf{Pericarditis}]} \rightarrow p_{[\exists \mathsf{hasLocation.Pericardium}]} \quad a''_6 \\
\wedge & p_{[\mathsf{Inflammation}]} \rightarrow p_{[\mathsf{Disease}]} & a'_7 & \wedge \quad p_{[\mathsf{Inflammation}]} \rightarrow p_{[\exists \mathsf{actsOn.Tissue}]} \quad a''_7 \\
\wedge & p_{[\mathsf{Disease}]} \wedge p_{[\mathsf{New}]} \rightarrow p_{[\mathsf{HeartDisease}]} & a'_8 & \wedge \quad p_{[\exists \mathsf{hasLocation.Heart}]} \rightarrow p_{[\mathsf{New}]} \quad a_0 \\
\wedge & p_{[\mathsf{HeartDisease}]} \rightarrow p_{[\exists \mathsf{hasState.NeedsTreatement}]} & a_9 & \\
\wedge & p_{[\mathsf{Appendicitis}]} \rightarrow p_{[\mathsf{Disease}]} & b_1 & \wedge \quad p_{[\mathsf{Appendicitis}]} \rightarrow p_{[\exists \mathsf{actsOn.Tissue}]} \quad b_2 \\
\wedge & p_{[\mathsf{Endocarditis}]} \rightarrow p_{[\mathsf{Disease}]} & b_3 & \wedge \quad p_{[\mathsf{Endocarditis}]} \rightarrow p_{[\exists \mathsf{actsOn.Tissue}]} \quad b_4 \\
\wedge & p_{[\mathsf{Pericarditis}]} \rightarrow p_{[\mathsf{Disease}]} & b_5 & \wedge \quad p_{[\mathsf{Pericarditis}]} \rightarrow p_{[\exists \mathsf{actsOn.Tissue}]} \quad b_6 \\
\wedge & p_{[\mathsf{Pericarditis}]} \rightarrow p_{[\exists \mathsf{hasLocation.Heart}]} & b_7 & \wedge \quad p_{[\mathsf{Pericarditis}]} \rightarrow p_{[\mathsf{New}]} \quad b_8 \\
\wedge & p_{[\mathsf{Pericarditis}]} \rightarrow p_{[\mathsf{HeartDisease}]} & b_9 & \wedge \quad p_{[\mathsf{Pericarditis}]} \rightarrow p_{[\exists \mathsf{hasState.NeedsTreatement}]} \quad b_{10}
\end{array}
$$

The clauses in the first nine rows represent the encoding of the (normalized) axioms of $\mathcal{O}_{\mathsf{med}}$, while the clauses in the last five rows represent the encoding of the other subsumption relations deduced from the axioms of $\mathcal{O}_{\mathsf{med}}$. We use the same labeling of Example 7.3.2.

For instance, performing concept subsumption on the query $\mathsf{Pericarditis} \sqsubseteq \mathsf{HeartDisease}$ corresponds to solve $\phi_{\mathcal{O}_{\mathsf{med}}}$ under the assumption list $\{p_{[\mathsf{Pericarditis}]}, \neg p_{[\mathsf{HeartDisease}]}\}$, which is clearly unsatisfiable because clause b_9 of $\phi_{\mathcal{O}_{\mathsf{med}}}$ is falsified by the two assumed literals.

Instead, if the query is $\mathsf{Appendicitis} \sqsubseteq \mathsf{HeartDisease}$, this corresponds to solve $\phi_{\mathcal{O}_{\mathsf{med}}}$ under the assumptions $\{p_{[\mathsf{Appendicitis}]}, \neg p_{[\mathsf{HeartDisease}]}\}$. This leads to the unit-propagation in $\phi_{\mathcal{O}_{\mathsf{med}}}$ of $p_{[\mathsf{Inflammation}]}$ and $p_{[\exists \mathsf{hasLocation.Appendix}]}$ from a'_4 and a''_4 respectively, then to the unit-propagation of $p_{[\mathsf{Disease}]}$ and $p_{[\exists \mathsf{actsOn.Tissue}]}$ from a'_7 and a''_7 (or equivalently from b_1 and b_2) respectively. After that, no more atom can be unit-propagated and no clause is falsified. Thus, since $\phi_{\mathcal{O}_{\mathsf{med}}}$ is a Horn formula, we can conclude that it is satisfiable, and that $\mathsf{Appendicitis} \sqsubseteq \mathsf{HeartDisease}$ is not a subsumption relation deducible from $\mathcal{O}_{\mathsf{med}}$. $\diamond$

7.4.2 Computing single and all MinAs via Conflict Analysis

We consider the general problem of generating MinAs. We build a more-general Horn propositional formula $\phi^{all}_{\mathcal{T}}$ representing the complete classification DAG of the input normalized ontology $\mathcal{T}$. [5] The size of $\phi^{all}_{\mathcal{T}}$ is polynomial wrt. that of $\mathcal{T}$.

Building the formula $\phi^{all}_{\mathcal{T}}$.

In order to make the explanation much simpler, we assume wlog. that in all the axioms of $\mathcal{T}$ all $\sqcap$'s and $\circ$'s are binary, i.e., that $1 \leq k \leq 2$ in (7.2) and $1 \leq n \leq 2$ in (7.4). This

[4] In this section and in the following ones, with a small abuse of notation, we use the names $\mathcal{O}_{\mathsf{med}}$ and $\mathcal{O}_{\mathsf{milk}}$ to identify both the original ontology and its normalized version.

[5] Here "complete" means "including also the rule applications generating already-generated assertions".

is not restrictive, since, e.g., each GCI axiom of the form $C_1 \sqcap ... \sqcap C_k \sqsubseteq D$ in $\mathcal{T}$ can be rewritten into the set $\{C_1 \sqcap C_2 \sqsubseteq C_{1:2}, C_{1:2} \sqcap C_3 \sqsubseteq C_{1:3}, ..., C_{1:k-1} \sqcap C_k \sqsubseteq D\}$, and each RI axiom of the form $r_1 \circ \cdots \circ r_n \sqsubseteq s$ can be rewritten into the set $\{r_1 \circ r_2 \sqsubseteq r_{1:2}, r_{1:2} \circ r_3 \sqsubseteq r_{1:3}, ..., r_{1:n-1} \circ r_n \sqsubseteq s\}$, each $C_{1:i}$ and $r_{1:j}$ being a fresh concept name and a fresh role name respectively.

Definition 10 ($\phi^{all}_{\mathcal{T}}$, $\phi_{\mathcal{T}(so)}$, $\phi^{all}_{\mathcal{T}(po)}$)**.** Let $\mathcal{T}$ be an $\mathcal{EL}^+$ TBox in normal form and let $\mathcal{A}$ be the classification of $\mathcal{T}$. We consider the concept variables $\{p_{[X]} \mid X \in \mathsf{NC}_{\mathcal{T}}\}$. We introduce the set of propositional variables $\{s_{[a_i]} \mid a_i \in \mathcal{A}\}$, that we call *assertion [resp. axiom] selector variables*. Then $\phi^{all}_{\mathcal{T}} \stackrel{\text{def}}{=} \phi_{\mathcal{T}(so)} \wedge \phi^{all}_{\mathcal{T}(po)}$, where

- $\phi_{\mathcal{T}(so)}$ is the conjunction of all the clauses:

$$\{s_{[a_i]} \rightarrow \mathcal{EL}^+ 2sat(a_i) \mid a_i \in \mathcal{A}\} \tag{7.9}$$

  that we call *assertion clauses*,

- and $\phi^{all}_{\mathcal{T}(po)}$ is the conjunction of all the clauses:

$$\{(s_{[a_i]} \wedge s_{[a_{i'}]} \wedge s_{[a_j]}) \rightarrow s_{[a_k]} \mid a_i, a_{i'}, a_k \in \mathcal{A},\ a_j \in \mathcal{T} \text{ and } r(a_i, a_{i'}, a_j, a_k)\} \tag{7.10}$$

  that we call *rule clauses*. With $r(a_i, a_{i'}, a_j, a_k)$ we mean that r is one of the completion rules of the classification algorithm of Section 7.3 and $a_i, a_{i'}, a_j, a_k$ are valid instances of (respectively) the preconditions (left and central columns of Table 7.1) and of the conclusion (right column of Table 7.1) of r. (Some require only one assertion a_i and one axiom a_j as premises of the rule r; in these cases let $s_{[a_{i'}]}$ be $\top$.) $\diamond$

Notice that (7.9) and (7.10) are definite Horn clauses and, hence, $\phi^{all}_{\mathcal{T}}$ is a definite Horn formula.

Proposition 14. *The size of the formula $\phi^{all}_{\mathcal{T}}$ defined in Definition 10 is worst-case polynomial in the size of the TBox $\mathcal{T}$.*

The result in Theorem 13 extends straightforwardly to $\phi^{all}_{\mathcal{T}}$, as described in the following.

Theorem 15. *Given an $\mathcal{EL}^+$ TBox $\mathcal{T}$ in normal form, for every $\mathcal{S} \subseteq \mathcal{T}$ and for every pair of concept names C, D in $\mathsf{PC}_{\mathcal{T}}$, $C \sqsubseteq_{\mathcal{S}} D$ if and only if the Horn propositional formula $\phi^{all}_{\mathcal{T}(po)} \wedge \bigwedge_{ax_i \in \mathcal{S}} s_{[ax_i]} \wedge \neg s_{[C \sqsubseteq D]}$ is unsatisfiable.*

Theorem 16. *Given an $\mathcal{EL}^+$ TBox $\mathcal{T}$ in normal form, for every $\mathcal{S} \subseteq \mathcal{T}$ and for every pair of concept names C, D in $\mathsf{PC}_{\mathcal{T}}$, $C \sqsubseteq_{\mathcal{S}} D$ if and only if the Horn propositional formula $\phi^{all}_{\mathcal{T}} \wedge \bigwedge_{ax_i \in \mathcal{S}} s_{[ax_i]} \wedge p_{[C]} \wedge \neg p_{[D]}$ is unsatisfiable.*

Corollary 17. *Given an $\mathcal{EL}^+$ TBox $\mathcal{T}$ in normal form, for every pair of concept names C, D in $\mathsf{PC}_{\mathcal{T}}$, $C \sqsubseteq_{\mathcal{T}} D$ if and only if the Horn propositional formula $\phi^{all}_{\mathcal{T}} \wedge \bigwedge_{ax_i \in \mathcal{T}} s_{[ax_i]} \wedge p_{[C]} \wedge \neg p_{[D]}$ [resp. $\phi^{all}_{\mathcal{T}(po)} \wedge \bigwedge_{ax_i \in \mathcal{T}} s_{[ax_i]} \wedge \neg s_{[C \sqsubseteq D]}$] is unsatisfiable.*

Intuitively, $\phi^{all}_{\mathcal{T}(po)}$ mimics the whole classification process, each rule clause representing one rule application. Thus, if a SAT solver is fed the formula $\phi^{all}_{\mathcal{T}(po)} \wedge \bigwedge_{ax_i \in \mathcal{S}} s_{[ax_i]} \wedge \neg s_{[C \sqsubseteq D]}$ of Theorem 15 (or $\phi^{all}_{\mathcal{T}(po)}$ under the assumption list $\{\neg s_{[C \sqsubseteq D]}\} \cup \{s_{[ax_i]} \mid ax_i \in \mathcal{S}\}$), then all the variables $s_{[a_j]}$ s.t. a_j can be deduced from $\mathcal{S}$ are instantly unit-propagated. If (and only if) $C \sqsubseteq_{\mathcal{S}} D$, then also $s_{[C \sqsubseteq D]}$ is unit-propagated, causing a conflict. Similarly, if the formula $\phi^{all}_{\mathcal{T}} \wedge \bigwedge_{ax_i \in \mathcal{S}} s_{[ax_i]} \wedge p_{[C]} \wedge \neg p_{[D]}$ of Theorem 16 is fed to the SAT solver, then if (and only if) $C \sqsubseteq_{\mathcal{S}} D$, then $s_{[C \sqsubseteq D]}$ is unit-propagated, which causes a conflict against the assertion clause $s_{[C \sqsubseteq D]} \rightarrow (p_{[C]} \rightarrow p_{[D]})$ in $\phi^{all}_{\mathcal{T}}$ and the unit clauses $p_{[C]} \wedge \neg p_{[D]}$.

Notice that, in general, there may be more than one way of deducing $C \sqsubseteq D$ from $\mathcal{S}$. This corresponds to the fact that there may be more than one unit-propagation sequence leading to the propagation of $s_{[C \sqsubseteq D]}$. (We will investigate this issue in Section 7.4.2.)

Remark 3. Theorem 16 suggest that, once the formula $\phi^{all}_{\mathcal{T}}$ is generated, it is possible to reason in terms of every subset $\mathcal{S}$ of $\mathcal{T}$ by "selecting" all and only the axioms we are interested in. This requires no new formula generation or computation on $\mathcal{S}$ or $\mathcal{T}$. Rather, it is sufficient to restrict the list of the assumptions for each query on $\phi^{all}_{\mathcal{T}}$ to the set of the selector variables of the axioms of $\mathcal{S}$ and to the selector variable of the query. $\diamond$

Example 7.4.2. Consider the ontology $\mathcal{O}_{\mathsf{med}}$ in Example 3.5.1. Then $\phi^{all}_{\mathcal{O}_{\mathsf{med}}} \stackrel{\text{def}}{=} \phi_{\mathcal{O}_{\mathsf{med}}(so)} \wedge \phi^{all}_{\mathcal{O}_{\mathsf{med}}(po)}$:

$$
\begin{aligned}
\phi_{\mathcal{O}_{\mathsf{med}}(so)} \stackrel{\text{def}}{=}\ & s_{[a_1]} \rightarrow (p_{[\mathsf{Appendix}]} \rightarrow p_{[\mathsf{BodyPart}]}) && \wedge\ s_{[a_1]} \rightarrow (p_{[\mathsf{Appendix}]} \rightarrow p_{[\exists \mathsf{partOf.Intestine}]})\\
\wedge\ & s_{[a_2]} \rightarrow (p_{[\mathsf{Endocardium}]} \rightarrow p_{[\mathsf{Tissue}]}) && \wedge\ s_{[a_2]} \rightarrow (p_{[\mathsf{Endocardium}]} \rightarrow p_{[\exists \mathsf{partOf.HeartValve}]})\\
\wedge\ & s_{[a_3]} \rightarrow (p_{[\mathsf{Pericardium}]} \rightarrow p_{[\mathsf{Tissue}]}) && \wedge\ s_{[a_3]} \rightarrow (p_{[\mathsf{Pericardium}]} \rightarrow p_{[\exists \mathsf{containedIn.Heart}]})\\
\wedge\ & s_{[a_4]} \rightarrow (p_{[\mathsf{Appendicitis}]} \rightarrow p_{[\mathsf{Inflammation}]}) && \wedge\ s_{[a_5]} \rightarrow (p_{[\mathsf{Appendicitis}]} \rightarrow p_{[\exists \mathsf{hasLocation.Appendix}]})\\
\wedge\ & s_{[a_5]} \rightarrow (p_{[\mathsf{Endocarditis}]} \rightarrow p_{[\mathsf{Inflammation}]}) && \wedge\ s_{[a_5]} \rightarrow (p_{[\mathsf{Endocarditis}]} \rightarrow p_{[\exists \mathsf{hasLocation.Endocardium}]})\\
\wedge\ & s_{[a_6]} \rightarrow (p_{[\mathsf{Pericarditis}]} \rightarrow p_{[\mathsf{Inflammation}]}) && \wedge\ s_{[a_6]} \rightarrow (p_{[\mathsf{Pericarditis}]} \rightarrow p_{[\exists \mathsf{hasLocation.Pericardium}]})\\
\wedge\ & s_{[a_7]} \rightarrow (p_{[\mathsf{Inflammation}]} \rightarrow p_{[\mathsf{Disease}]}) && \wedge\ s_{[a_7]} \rightarrow (p_{[\mathsf{Inflammation}]} \rightarrow p_{[\exists \mathsf{actsOn.Tissue}]})\\
\wedge\ & s_{[a_8]} \rightarrow (p_{[\mathsf{Disease}]} \wedge p_{[\mathsf{New}]} \rightarrow p_{[\mathsf{HeartDisease}]}) && \wedge\ s_{[a_0]} \rightarrow (p_{[\exists \mathsf{hasLocation.Heart}]} \rightarrow p_{[\mathsf{New}]})\\
\wedge\ & s_{[a_9]} \rightarrow (p_{[\mathsf{HeartDisease}]} \rightarrow p_{[\exists \mathsf{hasState.NeedsTreatement}]}) &&\\
\wedge\ & s_{[b_1]} \rightarrow (p_{[\mathsf{Appendicitis}]} \rightarrow p_{[\mathsf{Disease}]}) && \wedge\ s_{[b_2]} \rightarrow (p_{[\mathsf{Appendicitis}]} \rightarrow p_{[\exists \mathsf{actsOn.Tissue}]})\\
\wedge\ & s_{[b_3]} \rightarrow (p_{[\mathsf{Endocarditis}]} \rightarrow p_{[\mathsf{Disease}]}) && \wedge\ s_{[b_4]} \rightarrow (p_{[\mathsf{Endocarditis}]} \rightarrow p_{[\exists \mathsf{actsOn.Tissue}]})\\
\wedge\ & s_{[b_5]} \rightarrow (p_{[\mathsf{Pericarditis}]} \rightarrow p_{[\mathsf{Disease}]}) && \wedge\ s_{[b_6]} \rightarrow (p_{[\mathsf{Pericarditis}]} \rightarrow p_{[\exists \mathsf{actsOn.Tissue}]})\\
\wedge\ & s_{[b_7]} \rightarrow (p_{[\mathsf{Pericarditis}]} \rightarrow p_{[\exists \mathsf{hasLocation.Heart}]}) && \wedge\ s_{[b_8]} \rightarrow (p_{[\mathsf{Pericarditis}]} \rightarrow p_{[\mathsf{New}]})\\
\wedge\ & s_{[b_9]} \rightarrow (p_{[\mathsf{Pericarditis}]} \rightarrow p_{[\mathsf{HeartDisease}]}) && \wedge\ s_{[b_{10}]} \rightarrow (p_{[\mathsf{Pericarditis}]} \rightarrow p_{[\exists \mathsf{hasState.NeedsTreatement}]})
\end{aligned}
$$

$$
\begin{aligned}
\phi^{all}_{\mathcal{O}_{\mathsf{med}}(po)} \stackrel{\text{def}}{=}\ & s_{[a_4]} \wedge s_{[a_7]} \rightarrow s_{[b_1]} && \wedge\ s_{[a_4]} \wedge s_{[a_7]} \rightarrow s_{[b_2]}\\
\wedge\ & s_{[a_5]} \wedge s_{[a_7]} \rightarrow s_{[b_3]} && \wedge\ s_{[a_5]} \wedge s_{[a_7]} \rightarrow s_{[b_4]}\\
\wedge\ & s_{[a_6]} \wedge s_{[a_7]} \rightarrow s_{[b_5]} && \wedge\ s_{[a_6]} \wedge s_{[a_7]} \rightarrow s_{[b_6]}\\
\wedge\ & s_{[a_6]} \wedge s_{[a_3]} \wedge s_{[a_{11}]} \rightarrow s_{[b_7]} && \wedge\ s_{[b_7]} \wedge s_{[a_0]} \rightarrow s_{[b_8]}\\
\wedge\ & s_{[b_5]} \wedge s_{[b_8]} \wedge s_{[a_8]} \rightarrow s_{[b_9]} && \wedge\ s_{[b_9]} \wedge s_{[a_9]} \rightarrow s_{[b_{10}]}.
\end{aligned}
$$

Notice that $s_{[a_{10}]}$ and $s_{[a_{11}]}$ refer to the RI axioms a_{10} and a_{11} in Example 3.5.1, so that no corresponding assertion rule occurs in $\phi_{\mathcal{O}_{\mathsf{med}}(so)}$.

Consider the formula $\phi^{all}_{\mathcal{O}_{\mathsf{med}}} \wedge \bigwedge_{i=0,\ldots,11} s_{[a_i]} \wedge p_{[\mathsf{Pericarditis}]} \wedge \neg p_{[\mathsf{HeartDisease}]}$. The propagation of $p_{[\mathsf{Pericarditis}]}$ and $\neg p_{[\mathsf{HeartDisease}]}$ causes the propagation of $\neg s_{[b_9]}$ from the last but one clause of $\phi_{\mathcal{O}_{\mathsf{med}}(so)}$. The propagation of $s_{[a_0]}$, $s_{[a_3]}$, $s_{[a_6]}$, $s_{[a_7]}$ and $s_{[a_{11}]}$ causes that of $s_{[b_5]}, s_{[b_7]}$ and hence of $s_{[b_8]}$ (from the fifth, seventh and eighth clauses of $\phi^{all}_{\mathcal{O}_{\mathsf{med}}(po)}$). Thus, since also $s_{[a_8]}$ is propagated, the nineth clause in $\phi^{all}_{\mathcal{O}_{\mathsf{med}}(po)}$ is falsified. Thus, we conclude that $\mathsf{Pericarditis} \sqsubseteq_{\mathcal{O}_{\mathsf{med}}} \mathsf{HeartDisease}$. It is easy to see, instead, that the formula $\phi^{all}_{\mathcal{O}_{\mathsf{med}}} \wedge \bigwedge_{a_i \in \mathcal{O}_{\mathsf{med}}} s_{[a_i]} \wedge p_{[\mathsf{Appendicitis}]} \wedge \neg p_{[\mathsf{HeartDisease}]}$ is satisfiable, from which we conclude that $\mathsf{Appendicitis} \not\sqsubseteq_{\mathcal{O}_{\mathsf{med}}} \mathsf{HeartDisease}$. ◇

Example 7.4.3. We report some sample clauses from the formula $\phi^{all}_{\mathcal{O}_{\mathsf{milk}}(po)}$ for the ontology $\mathcal{O}_{\mathsf{milk}}$ of Example 7.3.1. (On the right side we show the mapping between the axiom/assertion selector variables included in the sample clauses and the concept inclusions they represent.)

$$
\begin{array}{rcll}
\phi^{all}_{\mathcal{O}_{\mathsf{milk}}(po)} \stackrel{\text{def}}{=} & \ldots \wedge \ldots & m_0 = \exists \mathsf{hasPhysState.liquidState} \sqsubseteq \mathsf{N} & \\
s_{[m_5]} \wedge s_{[m_4]} \rightarrow s_{[n_1]} & \wedge \ldots & m_1 = \mathsf{BodyFluid} \sqsubseteq \mathsf{Fluid} & n_1 = \mathsf{Milk} \sqsubseteq \mathsf{Substance} \\
s_{[m_6]} \wedge s_{[m_0]} \rightarrow s_{[n_2]} & \wedge \ldots & m_2 = \mathsf{Liquid} \sqsubseteq \mathsf{Fluid} & n_2 = \mathsf{Milk} \sqsubseteq \mathsf{N} \\
s_{[m_5]} \wedge s_{[n_2]} \wedge s_{[m_3]} \rightarrow s_{[n_3]} & \wedge \ldots & m_3 = \mathsf{BodySubstance} \sqcap \mathsf{N} \sqsubseteq \mathsf{BodyFluid} & n_3 = \mathsf{Milk} \sqsubseteq \mathsf{BodyFluid} \\
s_{[n_1]} \wedge s_{[n_2]} \wedge s_{[m_9]} \rightarrow s_{[n_4]} & \wedge \ldots & m_4 = \mathsf{BodySubstance} \sqsubseteq \mathsf{Substance} & n_4 = \mathsf{Milk} \sqsubseteq \mathsf{Liquid} \\
s_{[n_3]} \wedge s_{[m_1]} \rightarrow s_{[n_5]} & \wedge \ldots & m_5 = \mathsf{Milk} \sqsubseteq \mathsf{BodySubstance} & n_5 = \mathsf{Milk} \sqsubseteq \mathsf{Fluid} \\
s_{[n_4]} \wedge s_{[m_2]} \rightarrow s_{[n_5]} & \wedge \ldots & m_6 = \mathsf{Milk} \sqsubseteq \exists \mathsf{hasPhysState.liquidState} & \\
\ldots & & m_9 = \mathsf{Substance} \sqcap \mathsf{N} \sqsubseteq \mathsf{Liquid} &
\end{array}
$$

We notice that, assuming all the $s_{[m_i]}$'s, there are two dinstinct chains of unit-propagations leading to propagate $s_{[n_5]}$: one from $\{s_{[m_0]}, s_{[m_1]}, s_{[m_3]}, s_{[m_5]}, s_{[m_6]}\}$, propagating $s_{[n_2]}$, $s_{[n_3]}$ and $s_{[n_5]}$, and another from $\{s_{[m_0]}, s_{[m_2]}, s_{[m_4]}, s_{[m_5]}, s_{[m_6]}, s_{[m_9]}\}$, propagating $s_{[n_1]}$, $s_{[n_2]}$, $s_{[n_4]}$ and $s_{[n_5]}$, corresponding respectively to the deduction of n_2, n_3 and n_5 from the axioms $\{m_0, m_1, m_3, m_5, m_6\}$ and to that of n_1, n_2, n_4 and n_5 from $\{m_0, m_2, m_4, m_5, m_6, m_9\}$. Thus we can conclude that $\{m_0, m_1, m_3, m_5, m_6\}$ and $\{m_0, m_2, m_4, m_5, m_6, m_9\}$ are two nMinAs for $n_5 \stackrel{\text{def}}{=} \mathsf{Milk} \sqsubseteq_{\mathcal{O}_{\mathsf{milk}}} \mathsf{Fluid}$. Since they are also minimal, they are also MinAs for n_5. [6] ◇

In practice, in order to build the formula $\phi^{all}_{\mathcal{T}}$, we run an extended version of the classification algorithm of Section 7.3, whose pseudo-code representation is presented in Figure 7.1, and which is based on the following main steps:

1. initially set $\phi_{\mathcal{T}(so)}$ and $\phi^{all}_{\mathcal{T}(po)}$ to the empty set of clauses. Then for every non-trivial GCI axiom $a_i \in \mathcal{T}$, add to $\phi_{\mathcal{T}(so)}$ the corresponding assertion clause of type (7.9);

[6] Notice that if the ontology contains a chain of trivial concept inclusions of the type $C \sqsubseteq D$, $D \sqsubseteq E$, $E \sqsubseteq F$, etc. all the possible combinations in the application of the first completion rule of Table 7.1 in a different order are encoded, due to the transitivity of concept inclusion. Nevertheless, the axioms selector variables involved in such clauses (i.e. the variables of the nMinA for such deductions) are always the same. Moreover, depending on the structure of the encoded ontology, it cannot be excluded that there no exists also a rule application leading to the deduction of some ontology's axioms.

```
ClauseSet build-φ^all_T (NormalizedOntology T)
// Initialization of Q, A, φ_T(so), φ^all_T(po)
 1.    Q = ∅; A = ∅; φ_T(so) = ∅; φ^all_T(po) = ∅;
 2.    for each primitive concept C in T
 3.       add C ⊑ C and C ⊑ ⊤ to A; introduce s_[C⊑C] = s_[C⊑⊤] = ⊤;
 4.       enqueue {C ⊑ C, C ⊑ ⊤} into Q;
 5.    for each GCI or RI axiom ax in T
 6.       add ax to A; introduce s_[ax];
 7.       if (ax is a non-trivial GCI axiom) then
 8.          add the clause (s_[ax] → EL+2sat(ax)) to φ_T(so);
 9.          enqueue ax into Q; }}
// Updating A, B and Q (B is the set of already-handled assertions)
10.    B = ∅;
11.    while (Q is not empty)
12.       dequeue a_h from Q;
13.       for each rule instance r(a_i, a_i', a_j, a_k) such that {a_i, a_i', a_j} \ B = {a_h}
14.          if a_k ∉ A then
15.             add a_k to A; introduce s_[a_k];
16.             add the clause (s_[a_k] → EL+2sat(a_k)) to φ_T(so);
17.             enqueue a_k into Q;
18.          add the clause ((s_[a_i] ∧ s_[a_i'] ∧ s_[a_j]) → s_[a_k]) to φ^all_T(po);
19.       B = B ∪ {a_h};
20.    return φ^all_T ≝ φ_T(so) ∧ φ^all_T(po);
```

Figure 7.1: Polynomial-time algorithm building the formula $\phi^{all}_{\mathcal{T}}$. Q is a queue of assertions, $\mathcal{A}$ and $\mathcal{B}$ are sets of assertions.

2. for every newly-deduced assertion $a_i \in \mathcal{A}$, add to $\phi_{\mathcal{T}(so)}$ the corresponding assertion clause of type (7.9);

3. for every possible rule instantiation $r(a_i, a_{i'}, a_j, a_k)$ of a completion rule r of Table 7.1 (either extending $\mathcal{A}$ or not), add to $\phi^{all}_{\mathcal{T}(po)}$ the corresponding rule clause of type (7.10).

(Notice that step 3. is novel wrt. the classification algorithm of Section 7.3 when applied to already-generated assertions in $\mathcal{A}$.) We perform step 2. and 3. in a queue-based manner, which ensures that every possible distinct (i.e. with different antecedents) rule application is applied only once. This is achieved with the following strategy: initially all GCI axioms are added to a queue Q and all axioms are included in $\mathcal{A}$. At each iteration one assertion a_h is dequeued, and steps 2. and/or 3. are applied *to all and only* the rule applications whose antecedents are exactly a_h and one or two of the previously-dequeued axioms/assertions $a_1, ..., a_{h-1}$. The novel assertions a_k which are deduced by the rule applications in step 2 are enqueued into Q and added to $\mathcal{A}$. This process ends when Q is empty.

The algorithm exposed above requires a polynomial amount of steps wrt. the size of $\mathcal{T}$. In fact, it avoids to repeat the same rule application (i.e. with exactly the same antecedents and consequence) more than once, and each rule application leads to the introduction of one or two clauses. Therefore the algorithm requires linear time wrt. the size of $\phi^{all}_{\mathcal{T}}$ that, in Proposition 14, has been proved to be at most polynomial in the size of $\mathcal{T}$.

Computing one MinA.

Once $\phi^{all}_{\mathcal{T}}$ is generated, in order to compute one MinA, we can exploit the technique of CDCL SAT solving under assumptions, adopting the Decision Scheme, described in Section 4.1. Given the set of the axiom selector variables $\mathcal{P}_{\mathcal{T}} \stackrel{\text{def}}{=} \{s_{[ax_j]} \mid ax_j \in \mathcal{T}\}$, after $\phi^{all}_{\mathcal{T}}$ is parsed and `DPLL` is initialized, each query $C_i \sqsubseteq_{\mathcal{T}} D_i$ corresponds to solving $\phi^{all}_{\mathcal{T}}$ under the assumption list $\mathcal{L}_i \stackrel{\text{def}}{=} \mathcal{P}_{\mathcal{T}} \cup \{p_{[C_i]}, \neg p_{[D_i]}\}$. This corresponds to a single run of `bcp` and one run of `analyze_conflict`, whose cost depends linearly only on the clauses where the unit-propagated literals occur. If `bcp` does not return `conflict`, then `sat` is returned without even performing conflict analysis. If `bcp` returns `conflict`, as explained in Section 4.1, then `analyze_conflict` produces a conflict clause $\psi^{C_i,D_i}_{\mathcal{T}^*} \stackrel{\text{def}}{=} p_{[D_i]} \vee \neg p_{[C_i]} \vee \bigvee_{a_i \in \mathcal{T}^*} \neg s_{[a_i]}$ s.t. $\mathcal{T}^*$ is an nMinA for $C_i \sqsubseteq_{\mathcal{T}} D_i$. In fact, since $\phi^{all}_{\mathcal{T}}$ is a definite Horn formula, the presence of both $p_{[C_i]}$ and $\neg p_{[D_i]}$ in $\mathcal{L}_i$ is necessary for causing the conflict, so that, due to the Decision Scheme, the conflict set necessarily contains both of them. (Intuitively, `analyze_conflict` implicitly spans upward the classification sub-DAG rooted in $C_i \sqsubseteq_{\mathcal{T}} D_i$ and having $\mathcal{T}^*$ as leaf nodes, which contains all and only the nodes of the assertions which have been used to generate $C_i \sqsubseteq_{\mathcal{T}} D_i$.)

In the general case $\mathcal{T}^*$ is not necessarily minimal. In order to minimize it, we can apply the SAT-based variant of the linear minimization algorithm of Baader et al. (2007) in Figure 7.2. (We assume that $\phi^{all}_{\mathcal{T}}$ has been parsed and `DPLL` has been initialized, and that $\phi^{all}_{\mathcal{T}}$ has been solved under the assumption list $\mathcal{L}_i$ above, producing the conflict clause $\psi^{C_i,D_i}_{\mathcal{T}^*}$ and hence the nMinA $\mathcal{T}^*$; then `lin-extract-MinA`$_{DPLL}(C_i, D_i, \mathcal{T}^*, \phi^{all}_{\mathcal{T}})$ is invoked.) In a nutshell, the algorithm of Figure 7.2 tries to remove one-by-one the axioms a_js in $\mathcal{T}^*$, each time checking whether the reduced set of axioms $\mathcal{S} \setminus \{a_j\}$ is still such that $C_i \sqsubseteq_{S \setminus \{a_j\}} D_i$. As before, each call to `DPLLUnderAssumptions` requires only one run of `bcp`.

The correctness of this algorithm is a straightforward consequence of Theorem 16, and is stated in the next corollary.

Corollary 18. *Given an $\mathcal{EL}^+$ TBox $\mathcal{T}$ in normal form and the Horn propositional formula $\phi^{all}_{\mathcal{T}}$, for every pair of concept names C, D in $\mathsf{PC}_{\mathcal{T}}$, and every axiom set $\mathcal{T}^*$ (nMinA) for $C \sqsubseteq_{\mathcal{T}} D$, the algorithm* `lin-extract-MinA`$_{DPLL}(C, D, \mathcal{T}^*, \phi^{all}_{\mathcal{T}})$ *of Figure 7.2 computes a* **minimal** *axiom set $\mathcal{S} \subseteq \mathcal{T}^*$ such that $C \sqsubseteq_{\mathcal{S}} D$ ($\mathcal{S}$ is a MinA for $C \sqsubseteq D$ wrt. $\mathcal{T}$).*

This schema can be further improved as follows: if `DPLLUnderAssumptions` performs also conflict analysis and returns (the conflict clause corresponding to) an nMinA $\mathcal{S}'$ s.t. $\mathcal{S}' \subset \mathcal{S} \setminus \{a_i\}$, then $\mathcal{S}$ is assigned to $\mathcal{S}'$ and all axioms in $(\mathcal{S} \setminus \{a_j\}) \setminus \mathcal{S}'$ will not be

```
AxiomSet lin-extract-MinA_DPLL(Concept C_i, D_i, AxiomSet T*, formula φ^all_T)
1.    S = T*;
2.    for each axiom ax_j in T*
3.       L = {s_[ax_i] | ax_i ∈ S \ {ax_j}} ∪ {p_[C_i], ¬p_[D_i]};
4.       if (DPLLUnderAssumptions(φ^all_T, L) == unsat)
5.          S = S \ {ax_j};
6.    return S;
```

Figure 7.2: SAT-based variant of the linear MinA-extracting algorithm of Baader et al. (2007).

selected in next loops. As an alternative choice, one can implement instead (a SAT-based version of) the binary-search variant of the minimization algorithm (see, e.g., Baader & Suntisrivaraporn, 2008).

It is important to notice that the formula $\phi^{all}_{\mathcal{T}}$ is never updated: in order to check $C_i \sqsubseteq_{S\setminus\{a_j\}} D_i$, it suffices to drop $s_{[a_j]}$ from the assumption list. The latter fact makes (the encoding of) the axiom a_j useless for `bcp` to falsify the clauses encoding $C_i \sqsubseteq_{\mathcal{T}} D_i$, so that `DPLLUnderAssumptions` returns `unsat` if and only if a different falsifying chain of unit-propagations can be found, corresponding to a different sequence of rule applications again generating $C_i \sqsubseteq_{\mathcal{T}} D_i$. Notice that this fact is made possible by the definition of the encoding, which allows for including all the alternative sequences of rule applications generating the same assertions (i.e. the clauses added at step 3. of the algorithm for some assertion already in $\mathcal{A}$).

We also notice that one straightforward variant to this technique, which is feasible since typically $|\mathcal{T}^*| \ll |\mathcal{T}|$, is to compute another formula $\phi^{all}_{\mathcal{T}^*}$ from scratch and to feed it to the algorithm of Figure 7.2 instead of $\phi^{all}_{\mathcal{T}}$.

One very important remark is in order. During pinpointing the only clause of type (7.9) in $\phi^{all}_{\mathcal{T}}$ which is involved in the conflict analysis process is $s_{[C_i \sqsubseteq D_i]} \rightarrow (p_{[C_i]} \rightarrow p_{[D_i]})$, which reduces to the unit clause $\neg s_{[C_i \sqsubseteq D_i]}$ after the unit-propagation of the assumption literals $p_{[C_i]}, \neg p_{[D_i]}$. Thus, one may want to decouple pinpointing from subsumption and use the formula of Theorem 15 instead. Thus each query $C_i \sqsubseteq_{\mathcal{T}} D_i$ corresponds to solving $\phi^{all}_{\mathcal{T}(po)}$ under the assumption list $\mathcal{L}_i \stackrel{\text{def}}{=} \mathcal{P}_{\mathcal{T}} \cup \{\neg s_{[C_i \sqsubseteq D_i]}\}$, so that the algorithm for pinpointing is changed only in the fact that $\phi^{all}_{\mathcal{T}(po)}$ and $\{\neg s_{[C_i \sqsubseteq D_i]}\}$ are used instead of $\phi^{all}_{\mathcal{T}}$ and $\{p_{[C_i]}, \neg p_{[D_i]}\}$ respectively. Thus, wlog. in the remaining part of this section we will reason using $\phi^{all}_{\mathcal{T}(po)}$ and $\{\neg s_{[C_i \sqsubseteq D_i]}\}$. (The same results, however, can be obtained using $\phi^{all}_{\mathcal{T}}$ and $\{p_{[C_i]}, \neg p_{[D_i]}\}$ instead.)

This has been said, in Figure 7.3 we propose the pseudo-code of the algorithm which, given: the subformula $\phi^{all}_{\mathcal{T}(po)}$, the set of the axiom selector variables $\mathcal{P}_{\mathcal{S}}$ and the query selector variable $s_{[C_i \sqsubseteq D_i]}$, computes (the set of the axiom selector variables $\mathcal{P}_{\mathcal{S}^*}$ representing) one MinA $\mathcal{S}^*$ for $C \sqsubseteq D$ wrt. $\mathcal{S}$. The algorithm includes a fully boolean version of the minimization algorithm of Figure 7.2. The following result is a strightforwad consequence of Theorem 15 and Corollary 18.

```
MinA-SelVars compute-one-MinA(formula φ^all_T(po), assumptions P_S, query {¬s_[C_i⊑D_i]})
1.     if (DPLLUnderAssumptions(φ^all_T(po), P_S ∪ {¬s_[C_i⊑D_i]}) == unsat)
2.        set P_S* from the conflict analysis;
3.        for each axiom selector variable s_[ax_j] in P_S*
4.           L = P_S* \ {s_[ax_j]};
5.           if (DPLLUnderAssumptions(φ^all_T(po), L) == unsat)
6.              P_S* = L; // or set P_S* from the conflict analysis
7.        return P_S* \ {¬s_[C_i⊑D_i]};
8.     else
9.        return ∅;
```

Figure 7.3: SAT-based extraction of one MinA wrt. the given query.

Corollary 19. *Given an $\mathcal{EL}^+$ TBox $\mathcal{T}$ in normal form, the Horn propositional formula $\phi^{all}_{\mathcal{T}(po)}$, and a set of axiom selector variables $\mathcal{P}_\mathcal{S} \subseteq \mathcal{P}_\mathcal{T}$ from $\phi^{all}_{\mathcal{T}(po)}$, for every pair of concept names C, D in $\mathsf{PC}_\mathcal{T}$, the algorithm* `compute-one-MinA`$(\phi^{all}_{\mathcal{T}(po)}, \mathcal{P}_\mathcal{S}, \{\neg s_{[C\sqsubseteq D]}\})$ *of Figure 7.3 computes one set of axiom selector variables $\mathcal{P}_{\mathcal{S}^*}$ such that:*

- *either $C \not\sqsubseteq_\mathcal{S} D$ and $\mathcal{P}_{\mathcal{S}^*} = \emptyset$,*
- *or $C \sqsubseteq_\mathcal{S} D$, $\mathcal{P}_{\mathcal{S}^*} \neq \emptyset$ and $\mathcal{S}^* \subseteq \mathcal{S}$ is a* **minimal** *axiom set such that $C \sqsubseteq_{\mathcal{S}^*} D$ ($\mathcal{S}^*$ is a MinA for $C \sqsubseteq D$ wrt. $\mathcal{S}$).*

We remark that, thanks to the result proved in Theorem 15, the algorithm of Figure 7.3 is able to compute one MinA wrt. to $\mathcal{S}$ (if existing), whichever subset $\mathcal{S}$ of $\mathcal{T}$ is chosen, as stated in Corollary 19.

Computing all MinAs.

We describe a way of generating *all* MinAs for $C_i \sqsubseteq_\mathcal{T} D_i$ from $\phi^{all}_{\mathcal{T}(po)}$ and $\{\neg s_{[C_i\sqsubseteq D_i]}\}$. In a nutshell, the idea is to assume $\{\neg s_{[C_i\sqsubseteq D_i]}\}$ and to enumerate all possible minimal truth assignments on the axiom selector variables in $\mathcal{P}_\mathcal{T} \stackrel{\text{def}}{=} \{s_{[ax_j]} \mid ax_j \in \mathcal{T}\}$ which cause the inconsistency of the formula $\phi^{all}_{\mathcal{T}(po)}$. This can be implemented by means of a variant of the All-SMT technique of Lahiri et al. (2006). A naive version of this technique is described as follows.

We consider a propositional CNF formula φ on the set of axiom selector variables in $\mathcal{P}_\mathcal{T}$ φ is initially set to $\top$. One top-level instance of `DPLL` (namely `DPLL1`) is used to enumerate a complete set of truth assignments $\{\mu_k\}_k$ on the axiom selector variables in $\mathcal{P}_\mathcal{T}$ which satisfy φ. Every time that a novel assignment μ_k is generated, μ_k and $\{\neg s_{[C_i\sqsubseteq D_i]}\}$ are passed to an ad-hoc "$\mathcal{T}$-solver" checking whether it causes the inconsistency of the formula $\phi^{all}_{\mathcal{T}(po)}$ (i.e. it represents an nMinA for $C_i \sqsubseteq_\mathcal{T} D_i$). If this is the case, then the $\mathcal{T}$-solver returns a minimal subset $\mathcal{P}_{\mathcal{T}^*_k} = \{s_{[ax_j]} \mid ax_j \in \mathcal{T}^*_k\}$ of μ_k and $\mathcal{P}_\mathcal{T}$, s.t. $\mathcal{T}^*_k$ is a MinA, which caused such inconsistency. The clause $\psi^*_k \stackrel{\text{def}}{=} \bigvee_{ax_j\in\mathcal{T}^*_k} \neg s_{[ax_j]}$ (i.e. $\neg\mathcal{P}_{\mathcal{T}^*_k}$) is

```
MinA-Set compute-all-MinAs (encoding φ^all_T(po), assumptions P_T, query {¬s_[Ci⊑Di]})
 1.    φ = ⊤;
 2.    while (DPLL1(φ, P_T, μ_k) == sat)
 3.       P_Tk* = T-solver(φ^all_T(po), μ_k, {¬s_[Ci⊑Di]});
 4.       if (P_Tk* == ∅)
 5.          blevel = DPLL1_analyze_conflict(φ, μ_k);
 6.          if (blevel == 0) return;
 7.          φ = φ ∧ ¬μ_k;
 8.          DPLL1_backtrack(blevel, φ, μ_k);
 9.       else
10.          output T_k*;
11.          φ = φ ∧ ¬P_Tk*;
12.          DPLL1_backtrack(0, φ, P_Tk*);
```

Figure 7.4: all-SMT-based algorithm generating "all MinAs" wrt. the given query ($\mathcal{T}$-`solver` is `compute-one-MinA` from Figure 7.3).

then added to φ as a blocking clause and it is used as a conflict clause for driving next backjumping step. Otherwise, the $\mathcal{T}$-solver returns an empty set and `DPLL1` can use $\neg\mu_k$ as a "fake" conflict clause, which is added to φ as a blocking clause and is used as a conflict clause for driving next backjumping step. The whole process terminates when `backtrack` back-jumps to `blevel` zero. The set of all MinAs $\{\mathcal{T}_k^*\}_k$ are returned as output.

Notice that, since φ is initially set to $\top$, the first enumerated truth assignment μ_1 can be any of the exponentially many possible. Setting to positive the preferential polarity for the new assigned variables in `DPLL1` ensures that the first generated assignments contain a larger number of enabled (i.e. set to true) selector variables, so that it is more likely that they include new nMinAs. In this case, the first assignment $\mu_1 \stackrel{\text{def}}{=} \{s_{[ax_i]} \mid ax_i \in \mathcal{P}_\mathcal{T}\}$ (i.e. all the axiom selector variables set to true) and it always lead to the discovery of the first MinA for every existent subsumption relation in $\mathcal{T}$, so that at-least one MinA is always found in polynomial time including the time spent for building $\phi^{all}_{\mathcal{T}(po)}$.

Concretely, the $\mathcal{T}$-solver is the procedure `compute-one-MinA` described in Figure 7.3, which include a second instance of `DPLL` (under assumptions), namely `DPLL2`.In the $\mathcal{T}$-solver, we assume that $\phi^{all}_{\mathcal{T}(po)}$ is parsed and that `DPLL2` is initialized only once, before the whole process starts. The pseudocode of the whole procedure is given in Figure 7.4. `DPLL1` is assumed to run given the propositional formula φ and the set of propositional variables over which it is defined. It eventually returns (in the satisfiable case) a truth assignment μ_k.

Here we show that this naive procedure returns all the MinAs for any given query $C_i \sqsubseteq_\mathcal{T} D_i$.

Proposition 20. *Given the $\phi^{all}_{\mathcal{T}(po)}$ encoding for an $\mathcal{EL}^+$ TBox $\mathcal{T}$ in normal form, the set of the selector variables $\mathcal{P}_\mathcal{T}$ relative to the axioms of $\mathcal{T}$ and the selector variable $s_{[C\sqsubseteq D]}$*

from the existing subsumption $C \sqsubseteq_{\mathcal{T}} D$, the procedure described above and summarized in the pseudocode of Figure 7.4 produces all the MinAs for the subsumption $C \sqsubseteq_{\mathcal{T}} D$.

Proof. The procedure enumerates truth assignments on the variables in $\mathcal{P}_{\mathcal{T}}$ and checks whether they cause together with $\{\neg s_{[C_i \sqsubseteq D_i]}\}$ the inconsistency of the formula $\phi^{all}_{\mathcal{T}(po)}$ by `bcp` only. The search ends when all possible such assignments violate some conflict clause added to φ from the $\mathcal{T}$-solver (either an actual conflict clause or a "fake" one), that is, when we have $\bigwedge_h(\bigvee_{ax_j \in \mathcal{T}^*_h} \neg s_{[ax_j]}) \wedge \bigwedge_k(\neg \mu_k) \models \bot$, [7] where φ is set to $\top$. This means that every total assignment η on the variables in $\mathcal{P}_{\mathcal{T}}$ violates some clause in the latter formula, in particular: if η is s.t. the formula $\eta \wedge \phi^{all}_{\mathcal{T}(po)}$ is satisfiable, then η violates one of the clauses of the form $\neg \mu_k$, otherwise η violates one of the clauses of the form $\bigvee_{ax_j \in \mathcal{T}^*_h} \neg s_{[ax_j]}$. Let $\mathcal{S}$ be a set of axioms, and let $\eta_{\mathcal{S}} \stackrel{\text{def}}{=} \bigwedge_{ax_i \in \mathcal{S}} s_{[ax_i]} \wedge \bigwedge_{ax_i \in (\mathcal{T} \setminus \mathcal{S})} \neg s_{[ax_i]}$. If $C \sqsubseteq_{\mathcal{S}} D$, then $\neg s_{[C \sqsubseteq D]} \wedge \eta_{\mathcal{S}} \wedge \phi^{all}_{\mathcal{T}(po)}$ is unsatisfiable by Theorem 15. Thus, $\eta_{\mathcal{S}}$ violates some clause $\bigvee_{ax_j \in \mathcal{T}^*_h} \neg s_{[ax_j]}$, as stated above, that is, $\mathcal{S} \supseteq T^*_h$ for some MinA T^*_h. Thus, this procedure returns all MinAs for $C \sqsubseteq_{\mathcal{T}} D$. □

Example 7.4.4. Let us, for example, find all the MinAs for the (existing) subsumption $\mathsf{Milk} \sqsubseteq_{\mathcal{O}_{\mathsf{milk}}} \mathsf{Fluid}$ in the ontology $\mathcal{O}_{\mathsf{milk}}$ (Example 7.3.1). Hence, we run the above exposed procedure sumarized in Figure 7.4 on the formula $\phi^{all}_{\mathcal{O}_{\mathsf{milk}}(po)}$ and wrt. the query $s_{[\mathsf{Milk} \sqsubseteq \mathsf{Fluid}]}$. [8] The top-level `DPLL1` enumerates all the truth assignments on the variables $\mathcal{P}_{\mathcal{O}_{\mathsf{milk}}} \stackrel{\text{def}}{=} \{s_{[m_i]} \mid m_i \in \mathcal{O}_{\mathsf{milk}}\}$, satisfying the formula φ initially set to $\top$. The first produced truth assignment is $\mu_1 = \mathcal{P}_{\mathcal{O}_{\mathsf{milk}}} = \{s_{[m_i]} \mid m_i \in \mathcal{O}_{\mathsf{milk}}\}$: running $\mathcal{T}$`-solver` on $\phi^{all}_{\mathcal{O}_{\mathsf{milk}}(po)}$ and the input assumptions $\mu_1 \cup \{\neg s_{[\mathsf{Milk} \sqsubseteq \mathsf{Fluid}]}\}$ leads to the identification of the first MinA, e.g., $\mathcal{O}^*_{\mathsf{milk}1} \stackrel{\text{def}}{=} \{m_0, m_2, m_4, m_5, m_6, m_9\}$. [9] Thus the blocking clause $\psi^*_1 \stackrel{\text{def}}{=} (\neg s_{[m_0]} \vee \neg s_{[m_2]} \vee \neg s_{[m_4]} \vee \neg s_{[m_5]} \vee \neg s_{[m_6]} \vee \neg s_{[m_9]})$ is added to φ, which becomes $\varphi = \psi^*_1$. Next, a second truth assignment μ_2 is generated by `DPLL1`; μ_2 must not be conflicting with φ, so contain the negative occurence of at-least one of the selector variables representing the axioms in $\mathcal{O}^*_{\mathsf{milk}1}$. Suppose wlog. $\mu_2 = \mathcal{P}_{\mathcal{O}_{\mathsf{milk}}} \setminus \{s_{[m_5]}\}$. $\phi^{all}_{\mathcal{O}_{\mathsf{milk}}(po)} \wedge \mu_2 \wedge \{\neg s_{[\mathsf{Milk} \sqsubseteq \mathsf{Fluid}]}\}$ is satisfiable, so $\mathcal{T}$`-solver` returns the empty set and $\neg \mu_2$ is used by `DPLL1` as a fake blocking clause, driving the backjumping. [10] Then `DPLL1` generates a third assignment μ_3 which is not conflicting both with ψ^*_1 and with $\neg \mu_2$. Now suppose $\mu_3 = \mathcal{P}_{\mathcal{O}_{\mathsf{milk}}} \setminus \{s_{[m_4]}\}$. In this case $\mathcal{T}$`-solver` returns the set of selector variables $\mathcal{P}_{\mathcal{O}^*_{\mathsf{milk}2}}$ referring to the new MinA $\mathcal{O}^*_{\mathsf{milk}2} \stackrel{\text{def}}{=} \{m_0, m_1, m_3, m_5, m_6\}$, so that φ becomes $\varphi = \psi^*_1 \wedge \neg \mu_2 \wedge \psi^*_2$, where $\psi^*_2 \stackrel{\text{def}}{=} (\neg s_{[m_0]} \vee \neg s_{[m_1]} \vee \neg s_{[m_3]} \vee \neg s_{[m_5]} \vee \neg s_{[m_6]})$, and so on and so forth until termination,

[7] In general, an SMT solver which is run on a $\mathcal{T}$-unsatisfiable formula φ stops when $\varphi \wedge \bigwedge_k \neg \eta_k \models \bot$, s.t. the η_ks are the $\mathcal{T}$-conflict sets returned by the $\mathcal{T}$-solver and "$\models$" is purely-propositional entailment.

[8] Here we only simulate the execution of the procedure without showing the encoding $\phi^{all}_{\mathcal{O}_{\mathsf{milk}}(po)}$, whose significant clauses can be found in Example 7.4.3. Our purpose, in fact, is only to show the steps of the procedure in a concrete case.

[9] With m_0 we represent all the new definition axioms introduced during the normalization of the ontology, which label complex sub-concepts with fresh concept names (see Section 7.3.1 and Section 7.4.2).

[10] In fact $m_5 \stackrel{\text{def}}{=} \mathsf{Milk} \sqsubseteq \mathsf{BodySubstance}$ is a key axiom included in any MinA for $\mathsf{Milk} \sqsubseteq_{\mathcal{O}_{\mathsf{milk}}} \mathsf{Fluid}$. To be a $\mathsf{BodySubstance}$ is necessary both in order to be a $\mathsf{BodyFluid}$ (m_3) and in order to be (a $\mathsf{Substance}$ first and then) a Liquid (m_4 and m_9).

when all the possible assignments have been enumerated by `DPLL1` (no other MinA for $\mathsf{Milk} \sqsubseteq_{\mathcal{O}_{\mathsf{milk}}} \mathsf{Fluid}$ than $\mathcal{O}_{\mathsf{milk}_1}^*$ and $\mathcal{O}_{\mathsf{milk}_2}^*$ are contained in $\mathcal{O}_{\mathsf{milk}}$). ◇

One problem of the naive procedure above is that adding to φ a "fake" blocking clause (namely $\neg\eta_k$) each time a new satisfying truth assignment η_k is found may cause an exponential blowup of φ. As shown by Lahiri et al. (2006), this problem can be overcome by exploiting conflict analysis techniques. Each time a model η_k is found, it is possible to consider $\neg\eta_k$ as a conflicting clause to feed to `analyze_conflict` and to perform conflict-driven backjumping as if the blocking clause $\neg\eta_k$ belonged to the clause set; importantly, it is not necessary to add permanently the conflicting clause $\neg\eta_k$ to φ as a blocking clause, and it is sufficient to keep the conflict clause resulting from conflict analysis only as long as it is active.

Lahiri et al. (2006) proved that this technique terminates and allows for enumerating all models. (Notice that the generation of blocking clauses ψ_k^* representing MinAs is not affected, since in this case we add ψ_k^* to φ as blocking clause.) We refer the reader to Section 4.2.4) for more details.

One remark is in order. The reason why we use two different instances of `DPLL` is that we must distinguish unit-propagations of negated axiom selector variables $\neg s_{[ax_i]}$ on learned clauses from those performed on the clauses in $\phi^{all}_{\mathcal{T}(po)}$: on the one hand, we want to allow the former ones because they prevent exploring the same assignments more than once; on the other hand, we want to avoid the latter ones (or to perform them in a controlled way, as explained in Section 7.6.3 for the theory propagation variant) because they may prevent generating some counter-model of interest.

Computing one MinA using a much smaller formula.

Although polynomial, $\phi^{all}_{\mathcal{T}}/\phi^{all}_{\mathcal{T}(po)}$ may be huge for very-big ontologies $\mathcal{T}$ like SNOMED-CT. For these situations, we propose here a variant of the one-MinA procedure (which is an improved SAT-based version of the simplified one-MinA algorithm of Baader et al., 2007) using the much smaller formula $\phi^{one}_{\mathcal{T}}$ [resp. $\phi^{one}_{\mathcal{T}(po)}$].

Definition 11. Let $\mathcal{T}$ be an $\mathcal{EL}^+$ ontology in normal form and let $\mathcal{A}$ be the classification of $\mathcal{T}$. $\phi^{one}_{\mathcal{T}}$ is a CNF Horn propositional formula on:

- the variables $\{p_{[X]} \mid X \in \mathsf{NC}_{\mathcal{T}}\}$, that we call *concept variables*,
- the variables $\{s_{[a_i]} \mid a_i \in \mathcal{A}\}$, that we call *selector variables*,

and defined as $\phi^{one}_{\mathcal{T}} = \phi_{\mathcal{T}(so)} \wedge \phi^{one}_{\mathcal{T}(po)}$, where

- $\phi_{\mathcal{T}(so)}$ is the one defined in Definition 10;
- and $\phi^{one}_{\mathcal{T}(po)}$ is one (of the possibly many) sub-formula of $\phi^{all}_{\mathcal{T}(po)}$ (see Definition 10) defined as the conjunction of the rule clauses:

$$\{(s_{[a_i]} \wedge s_{[a_{i'}]} \wedge s_{[a_j]}) \rightarrow s_{[a_k]} \mid a_i, a'_i, a_k \in \mathcal{A},\ a_j \in \mathcal{T} \text{ and } r(a_i, a_{i'}, a_j, a_k) \in \mathcal{R}^*\} \quad (7.11)$$

where $\mathcal{R}^*$ is one set including all and only the completion-rule applications involved in one of the possibly many consistent (i.e. able to deduce once –and only once– starting from the axioms of $\mathcal{T}$ each of the assertions in $\mathcal{A}$) execution of the classification algorithm for $\mathcal{T}$. ◇

Thus, informally, $\phi^{one}_{\mathcal{T}}$ [resp. $\phi^{one}_{\mathcal{T}(po)}$] is defined like $\phi^{all}_{\mathcal{T}}$ [resp. $\phi^{all}_{\mathcal{T}(po)}$] with the difference that *exactly only one rule clause* of type (7.10) is included in the formula for every assertion $a \in \mathcal{A}$.

Remark 4. Notice that, while $\phi^{all}_{\mathcal{T}}$ [resp. $\phi^{all}_{\mathcal{T}(po)}$] is unique (once established the set of completion rules from which it depends), $\phi^{one}_{\mathcal{T}}$ [resp. $\phi^{one}_{\mathcal{T}(po)}$] is not an uniquely defined formula because many distinct instances of $\phi^{one}_{\mathcal{T}}$ [resp. $\phi^{one}_{\mathcal{T}(po)}$] can be defined for the same set of completion rules. In fact it can be defined different instances of $\phi^{one}_{\mathcal{T}}$ [resp. $\phi^{one}_{\mathcal{T}(po)}$] for different orders in the application of the completion rules leading to a complete classification $\mathcal{A}$ of $\mathcal{T}$. ◇

However, still, this ensures that every existing subsumption relation $C_i \sqsubseteq_{\mathcal{T}} D_i$ is represented by at-least one MinA in $\phi^{one}_{\mathcal{T}}$ [resp. $\phi^{one}_{\mathcal{T}(po)}$]. In fact, intuitively, it is possible to imagine the assertions of $\mathcal{A}$ like nodes of a connected direct graph, where an edge is placed in between every precondition and the relative consequence in a completion rule application (or in a $\phi^{one}_{\mathcal{T}}$ clause). This guarantees that it is possible to reach (i.e. to infer) every assertion a of $\mathcal{A}$ starting from the axioms of $\mathcal{T}$.

In particular $\phi^{one}_{\mathcal{T}}$ is sufficient to compute *one* non-minimal axiom set $\mathcal{T}^*$ by one run of `bcp` and `analyze_conflict`, as seen before. Since $\phi^{one}_{\mathcal{T}}$ does not represent all deductions of $C_i \sqsubseteq_{\mathcal{T}} D_i$, we cannot use the algorithm in Figure 7.2 to minimize it. However, since typically $\mathcal{T}^* \ll \mathcal{T}$, one can cheaply compute $\phi^{one}_{\mathcal{T}^*}$ and run a variant of the algorithm in Figure 7.2 in which at each loop a novel formula $\phi^{one}_{\mathcal{S}\setminus\{a_i\}}$ is computed and fed to `DPLLUnderAssumptions` together with the updated $\mathcal{L}$. One further variant is to compute instead $\phi^{all}_{\mathcal{T}^*(po)}$ and feed it to the algorithm in Figure 7.2.

The following result is formally proved in Appendix 7.9.

Corollary 21. *Given an $\mathcal{EL}^+$ TBox $\mathcal{T}$ in normal form, for every pair of concept names C_i, D_i in $\mathsf{PC}_{\mathcal{T}}$, the following facts hold:*

(i) for every $\mathcal{S} \subseteq \mathcal{T}$, if $\phi^{one}_{\mathcal{T}} \wedge \bigwedge_{a_i \in \mathcal{S}} s_{[a_i]} \wedge p_{[C_i]} \wedge \neg p_{[D_i]}$ is unsatisfiable then $C_i \sqsubseteq_{\mathcal{S}} D_i$;

(ii) if $C_i \sqsubseteq_{\mathcal{T}} D_i$ then there exists at least one (possibly proper) subset $\mathcal{S} \subseteq \mathcal{T}$ such that $\phi^{one}_{\mathcal{T}} \wedge \bigwedge_{a_i \in \mathcal{S}} s_{[a_i]} \wedge p_{[C_i]} \wedge \neg p_{[D_i]}$ is unsatisfiable.

$\phi^{one}_{\mathcal{T}}$ [resp. $\phi^{one}_{\mathcal{T}(po)}$] can be computed like $\phi^{all}_{\mathcal{T}}$ [resp. $\phi^{all}_{\mathcal{T}(po)}$] in the last part of Section 7.4.2, except that step 3. is never executed alone but always and only when also step 2. does. This has been said, it is more convenient to run a modified version of the classification algorithm of Section 7.3 as done in generating $\phi_{\mathcal{T}}$, so that only one deduction for each assertion is computed (including the respective rule clause (7.10)), than to run our algorithm for $\phi^{all}_{\mathcal{T}}$.

Handling normalization.

The normalized TBox $\mathcal{T} \stackrel{\text{def}}{=} \{ax_1, ..., ax_N\}$ can result from normalizing the non-normal one $\hat{\mathcal{T}} \stackrel{\text{def}}{=} \{\hat{ax}_1, ..., \hat{ax}_{\hat{N}}\}$ by means of the process hinted in Section 7.3; $|\mathcal{T}|$ is $O(|\hat{\mathcal{T}}|)$. Each original axiom $\hat{ax}_i$ is converted into a set of normalized axioms $\{ax_{i1}, ..., ax_{in_i}\}$, and each normalized axiom ax_j can be reused (in the case of a definition axiom) in the conversion of several original axioms $\hat{ax}_{j1}, ..., \hat{ax}_{jm_j}$. In order to handle non-normal TBoxes $\hat{\mathcal{T}}$, one variant of the technique of Baader et al. (2007) can be adopted: for every $\hat{ax}_i$, we add to $\phi^{all}_{\mathcal{T}(po)}$ [resp. $\phi^{all}_{\mathcal{T}}$] the set of clauses $\{s_{[\hat{ax}_i]} \rightarrow s_{[ax_{i1}]}, ..., s_{[\hat{ax}_i]} \rightarrow s_{[ax_{in_i}]}\}$, and then we use $\mathcal{P}_{\hat{\mathcal{T}}} \stackrel{\text{def}}{=} \{s_{[\hat{ax}_1]}, ..., s_{[\hat{ax}_{\hat{N}}]}\}$ as the novel set of axiom selector variables for the one-MinA and all-MinAs algorithms described above. Thus `analyze_conflict` finds conflict clauses in terms of variables in $\mathcal{P}_{\hat{\mathcal{T}}}$ rather than in $\mathcal{P}_{\mathcal{T}}$. (In practice, we treat normalization as the application of a novel kind of completion rules.) Since $\mathcal{P}_{\hat{\mathcal{T}}}$ is typically smaller than $\mathcal{P}_{\mathcal{T}}$, this may cause a significant reduction in the search space that the `DPLL1` must explore during the all-MinAs procedure of Figure 7.4. (Notice that when one ax_j is shared by $\hat{ax}_{j1}, ..., \hat{ax}_{jm_j}$, the clause set $\{s_{[\hat{ax}_{j1}]} \rightarrow s_{[a_j]}, ..., s_{[\hat{ax}_{jm_j}]} \rightarrow s_{[a_j]}\}$ is equivalent to $(s_{[\hat{ax}_{j1}]} \vee ... \vee s_{[\hat{ax}_{jm_j}]}) \rightarrow s_{[a_j]}$.)

Alternatively, the more compact solution we adopted allows for using directly and only the selector variables referring to the original axioms $\hat{\mathcal{T}} = \{\hat{ax}_1, ..., \hat{ax}_{\hat{N}}\}$. In such a way no extra clause is added to the encoding and a smaller number of selector variables is used. In fact, every non-normal axiom of $\mathcal{T}$ is normalized into two sets of normal axioms: (i) a set of *top-level axioms* in which complex concept descriptions are substituted by newly introduced concept names, and which keep representing the original concept inclusions, (ii) and a set of *definition(s) (axioms)* which represent the relations between the fresh concept names and the corresponding complex concept descriptions. For example the concept inclusion $\exists r.A \sqcap \exists s.B \sqsubseteq C \sqcap D$ is normalized into the set of top-level normal axioms: $\{X \sqsubseteq C, X \sqsubseteq D\}$ and the set of definition axioms: $\{\exists r.A \sqsubseteq Y, \exists s.B \sqsubseteq Z, Y \sqcap Z \sqsubseteq X\}$, which define Y as $\exists r.A$, Z as $\exists s.B$ and X as $Y \sqcap Z$.

The idea is to:

(i) use the same original axiom selector variable $s_{[\hat{ax}_i]}$ for all the top-level normal axioms coming out from the normalization of $\hat{ax}_i$; [11]

(ii) associate the same unique selector variable $s_{[a_0]}$ to all the description axioms introduced.

An informal explanation of this latest choice is that definition axioms play the role of labeling for complex concepts, so they take part in the deduction of a queried subsumption only in consequence of top-level axioms. Further, queries are always expressed in terms of original concept names, so we are ensured that the top-level selector variables of the original axioms are always (and firstly) involved in the search. The single selector variable $s_{[a_0]}$, instead, is used to represent and enable all the axioms defining the new concept

[11] Notice that more than one top-level axiom can result from the normalization of an equivalence relation or from the normalization of a right-side conjunction.

names coming out from the normalization. Thus, the presence of $s_{[a_0]}$ in a MinA is not of interest, it only indicates that at-least one of the axiom included in the MinA has been normalized. Finally, notice that some definitions can be (partially) shared among many different original axioms, but the above-exposed solution is transparent wrt. these situations. This schema for handling normalization has been already used in Examples 7.4.2 and 7.4.4.

(Hereafter we will call $\mathcal{T}$ the input TBox, assuming that it is in normal form, no matter if it is resulting from a normalization process or not and if we use the selector variables referring to the original axioms or to the normalized ones.)

7.4.3 Discussion

We first compare our all-MinAs technique for $\mathcal{EL}^+$ of Section 7.4.2 with that presented by Baader et al. (2007). By comparing the pinpointing formula $\Phi^{C_i \sqsubseteq_\mathcal{T} D_i}$ of Baader et al. (2007) (see also Section 7.3) with $\phi^{all}_{\mathcal{T}(po)}$, and by analyzing the way they are built and used, we highlight the following differences:

(i) $\Phi^{C_i \sqsubseteq_\mathcal{T} D_i}$ is built only on axiom selector variables in $\mathcal{P}_\mathcal{T} \stackrel{\text{def}}{=} \{s_{[ax_j]} \mid ax_j \in \mathcal{T}\}$, whilst $\phi^{all}_{\mathcal{T}(po)}$ is build on *all* selector variables in $\mathcal{P}_\mathcal{A} \stackrel{\text{def}}{=} \{s_{[a_j]} \mid a_j \in \mathcal{A}\}$ (i.e., of both axioms and inferred assertions);

(ii) the size of $\Phi^{C_i \sqsubseteq_\mathcal{T} D_i}$ and the time to compute it are worst-case *exponential* in $|\mathcal{T}|$ (Baader et al., 2007), whilst the size of $\phi^{all}_{\mathcal{T}(po)}$ and the time to compute it are worst-case *polynomial* in $|\mathcal{T}|$;

(iii) the algorithm for generating $\Phi^{C_i \sqsubseteq_\mathcal{T} D_i}$ of (Baader et al., 2007) requires intermediate logical checks, whilst the algorithm for building $\phi^{all}_{\mathcal{T}(po)}$ does not;

(iv) each MinA is a *model* of $\Phi^{C_i \sqsubseteq_\mathcal{T} D_i}$, whilst it is (the projection to $\mathcal{P}_\mathcal{T}$ of) a *counter-model* of $\phi^{all}_{\mathcal{T}(po)}$.

Moreover, our process can reason directly in terms of (the selector variables of) the input axioms, no matter whether normal or not.

In accordance with the Theorem 5 of Baader et al. (2007), also our approach is not output-polynomial, because in our proposed all-MinAs procedure even the enumeration of a polynomial amount of MinAs may require exploring an exponential amount of models. In our proposed approach, however, the potential exponentiality is completely relegated to the final step of our approach, i.e. to our variant of the all-SMT search, since the construction of the SAT formula is polynomial. Thus we can build $\phi^{all}_{\mathcal{T}(po)}$ once and then, for each $C_i \sqsubseteq_\mathcal{T} D_i$ of interest, run the all-SMT procedure until either it terminates or a given timeout is reached: in the latter case, we can collect the MinAs generated so far. (Notice that the fact that `DPLL1` selects *positive* axiom selector variables first tends to anticipate the enumeration of over-constrained assignments wrt. to that of under-constrained ones, so that it is more likely that counter-models, and thus MinAs, are enumerated during the first part of the search. In particular it is assured that it finds one MinA in polynomial

time.) With the all-MinAs algorithm of Baader et al. (2007), instead, it may take an exponential amount of time to build the pinpointing formula $\Phi^{C_i \sqsubseteq_\mathcal{T} D_i}$ before starting the enumeration of the MinAs.

As far as the generation of each single MinA of Section 7.4.2 is concerned, another interesting feature of our approach relates to the minimization algorithm of Figure 7.2: we notice that, once $\phi^{all}_{\mathcal{T}(po)}$ is generated, in order to evaluate different subsets $\mathcal{S} \setminus \{a_j\}$ of the axiom sets, it suffices to assume different selector variables, without modifying the formula, and perform one run of `bcp`. Similarly, if we want to compute one or all MinAs for different deduced assertion, e.g. $C_1 \sqsubseteq_\mathcal{T} D_1, \ldots, C_j \sqsubseteq_\mathcal{T} D_j, \ldots$, we do not need recomputing $\phi^{all}_{\mathcal{T}(po)}$ each time, we just need assuming (i.e. querying) each time a different axiom selector variable, e.g. respectively: $\neg s_{[C_1 \sqsubseteq_\mathcal{T} D_1]}, \ldots, \neg s_{[C_j \sqsubseteq_\mathcal{T} D_j]}, \ldots$.

7.5 A Preliminary Empirical Evaluation

In order to test the feasibility of our approach, we have implemented a first version of the procedures of Section 7.4 (hereafter referred as $\mathcal{EL}^+$2SAT) and we have performed a preliminary empirical evaluation of $\mathcal{EL}^+$2SAT on the ontologies mentioned in Sections 2.1 and 7.1, in more details: SNOMED-CT, NCI, GENEONTOLOGY, FULL-GALEN and NOT-GALEN. [12] This latter two ontologies are derived from GALEN; in particular: NOT-GALEN is a stripped-down version of GALEN with no role functionality, that has been widely used for benchmarking several standard DL-reasoners, while FULL-GALEN represents the full GALEN medical ontology, excluding role inverses and functionalities. We have implemented $\mathcal{EL}^+$2SAT in C++, including and modifying the code of the SAT solver MINISAT2.0 070721 (Eén & Sörensson, 2004). All tests have been run on a biprocessor dual-core machine `Intel Xeon 3.00GHz` with 4GB RAM on `Linux RedHat 2.6.9-11`. [13]

The results of this first evaluation are presented in Table 7.2. The first block reports the data of each ontology. The second and third blocks report respectively the size of the encoded formula, in terms of variable and clause number, and the CPU time taken to compute them. [14] The fourth block reports the time taken to load the formulas and to initialize `DPLL`. The fifth block reports the average time (on 100000 sample queries) required by checking subsumptions. [15] (Notice that $\phi_\mathcal{T}$ and $\phi^{one}_\mathcal{T}$ must be loaded and `DPLL` must be initialized only once for all queries.) The sixth block reports the same data for the computation of one nMinA, on 5000 sample queries. [16] (Loading times are the same

[12] The first four ontologies are available at `http://lat.inf.tu-dresden.de/~meng/toyont.html`, whilst SNOMED-CT'09 is courtesy of IHTSDO `http://www.ihtsdo.org/`.

[13] $\mathcal{EL}^+$2SAT is available from `http://disi.unitn.it/~rseba/elsat/`.

[14] The classification alone (excluding the time taken in encoding the problem and in computing the additional *rule clauses* for pinpointing) required respectively: 0.60, 2.24, 2.84, 34.06 and 3738.82 seconds for $\phi_\mathcal{T}$, 0.99, 2.63, 4.13, 41.19 and 3893.20 seconds for $\phi^{one}_\mathcal{T}$. In the case of $\phi^{all}_{\mathcal{T}(po)}$ the times are not distinguishable.

[15] The queries have been generated randomly, extracting about 2000 primitive concept names from each ontology and then randomly selecting 100000 queries from all the possible combinations of these concept names.

[16] We chose the first 5000 "unsatisfiable" queries we encounter when analyzing all the possible pairwise combinations of primitive concept names of each ontology.

Ontology	notGalen	GeneOnt.	NCI	fullGalen	Snomed'09
# of prim. concepts	2748	20465	27652	23135	310075
# of orig. axioms	4379	20466	46800	36544	310025
# of norm. axioms	8740	29897	46800	81340	857459
# of role names	413	1	50	949	62
# of role axioms	442	1	0	1014	12
Size (var#|clause#)					
$\phi_{\mathcal{T}}$	5.4e3|1.8e4	2.2e4|4.2e4	3.2e4|4.7e4	4.8e4|7.3e5	5.3e5|8.4e6
$\phi_{\mathcal{T}}^{one}$	2.3e4|2.7e4	5.5e4|5.4e4	7.8e4|4.7e4	7.3e5|1.4e6	8.4e6|1.6e7
$\phi_{\mathcal{T}(po)}^{all}$	1.7e5|2.2e5	2.1e5|2.6e5	2.9e5|3.0e5	5.3e6|1.2e7	2.6e7|8.4e7
Encode time					
$\phi_{\mathcal{T}}$	0.65	2.37	2.98	35.28	3753.04
$\phi_{\mathcal{T}}^{one}$	2.06	4.15	6.19	68.94	4069.84
$\phi_{\mathcal{T}(po)}^{all}$	1.17	1.56	2.37	178.41	198476.59
Load time					
$\phi_{\mathcal{T}}$	0.11	0.37	1.01	1.93	21.16
$\phi_{\mathcal{T}}^{one}$	0.18	0.55	1.17	5.95	59.88
Subsumption (on 10^5*)*					
$\phi_{\mathcal{T}}$	0.00002	0.00002	0.00003	0.00003	0.00004
$\phi_{\mathcal{T}}^{one}$	0.00003	0.00002	0.00003	0.00004	0.00008
nMinA $\phi_{\mathcal{T}}^{one}$ *(on* 5000*)*	0.00012	0.00027	0.00042	0.00369	0.05938
MinA $\phi_{\mathcal{T}}^{one}$ *(on* 100*)*					
– Load time	0.175	0.387	0.694	6.443	63.324
– Extract time	0.066	0.082	0.214	0.303	3.280
– DPLL Search time	0.004	0.004	0.002	0.010	0.093
MinA $\phi_{\mathcal{T}(po)}^{all}$ *(on* 100*)*					
– Load time	1.061	1.385	1.370	39.551	150.697
– DPLL Search time	0.023	0.027	0.036	0.331	0.351
allMinA $\phi_{\mathcal{T}(po)}^{all}$*(on* 30*)*					
– 50% *#MinA/time*	1/1.50	1/1.76	4/1.79	3/53.40	15/274.70
– 90% *#MinA/time*	2/1.59	4/2.11	6/1.86	9/63.61	32/493.61
– 100% *#MinA/time*	2/1.64	8/2.79	9/2.89	15/150.95	40/588.33

Table 7.2: Results of the preliminary evaluation of $\mathcal{EL}^+$2SAT on five $\mathcal{EL}^+$ ontologies. ("XeN" is "$X \cdot 10^N$". CPU times are in seconds.)

as above.) The seventh block reports the average times on 100 samples required to compute one MinA with $\phi_{\mathcal{T}}^{one}$, which computes the sequence of formulas $\phi_{\mathcal{T}}^{one}, \ldots, \phi_{\mathcal{S}\setminus a_i}^{one}, \ldots$.[17] (Here we report the sum of all the loading times, including the loading time of the first formula which dominates all the others; but notice that the process of loading of the first $\phi_{\mathcal{T}}^{one}$ can be shared by different samples. *Extract time* represents the total time spent in reconstructing and encoding all the intermediate sub-ontologies $\mathcal{S} \setminus \{a_i\}$. *DPLL Search time* is the sum of all the different DPLL's times.) The eighth block reports the average

[17] The queries are selected randomly from the 5000 samples introduced above.

times on 100 samples required to compute one MinA with $\phi^{all}_{\mathcal{T}(po)}$. The ninth block reports the results (50th, 90th and 100th percentiles) of running the all-MinAs procedure on 30 samples, each with a timeout of 1000s (loading included), and counting the number of MinAs generated and the time taken until the last MinA is generated. [18]

Notice that, although huge, a Horn formula of up to 10^8 clauses is at the reach of a SAT solver like MiniSat (e.g., Sebastiani & Vescovi, 2006, 2009a handled *non-Horn* formulas of $3.5 \cdot 10^7$ clauses).

Although still very preliminary, these empirical results allow us to notice a few facts:

(i) once the formulas are loaded, concept subsumption and computation of nMinAs are instantaneous, even with very-big formulas $\phi_{\mathcal{T}}$ and $\phi^{one}_{\mathcal{T}}$;

(ii) in the computation of single MinAs, with both $\phi^{one}_{\mathcal{T}}$ and $\phi^{all}_{\mathcal{T}(po)}$, DPLL search times are very low or even negligible: most time is taken by loading the main formula (which can be performed only once for all) and by extracting the information from intermediate results. Notice that $\mathcal{EL}^+$2SAT succeeded in computing some MinAs even with the huge ontology Snomed-CT'09;

(iii) although no sample concluded the full enumeration within the timeout of 1000s, the all-MinAs procedure allowed for enumerating a set of MinAs. Remarkably, all MinAs are all found in the very first part of the search, as expected.

7.6 Pushing the Envelope

7.6.1 Working on Sub-Ontologies

The results proved in Theorems 16 and 15 (Section 7.4.2) imply a very important fact: given the ontology $\mathcal{T}$, once $\phi^{all}_{\mathcal{T}(po)}/\phi^{all}_{\mathcal{T}}$ is encoded, in our approach we can reason on any possible desired sub-ontology $\mathcal{S}$ of $\mathcal{T}$ without the need for any extra modification or new encoding, by simply using different set of assumptions. This fact is also the base of the linear minimization algorithm exposed if Figure 7.2 of Section 7.4.2, which allows to obtain a MinA from the nMinA returned by conflict analysis.

Let us consider the all-MinAs problem in the case of $\phi^{all}_{\mathcal{T}(po)}$. Given a query $C_i \sqsubseteq D_i$ and any subset $\mathcal{S}$ of the axioms of $\mathcal{T}$ (i.e. any sub-ontology of $\mathcal{T}$), searching all the MinAs responsible for $C_i \sqsubseteq_{\mathcal{S}} D_i$ corresponds to solving $\phi^{all}_{\mathcal{T}(po)}$ under all the possible assumption lists generated by $\{\neg s_{[C_i \sqsubseteq D_i]}\} \cup \mathcal{P}_{\mathcal{S}}$, where $\mathcal{P}_{\mathcal{S}} \stackrel{\text{def}}{=} \{s_{[ax_j]} \mid ax_j \in \mathcal{S}\}$. Thus the algorithm computing all the MinAs is unchanged, it only enumerates assignments on the axiom selector variables of the (smaller) assumption list $\mathcal{P}_{\mathcal{S}}$ instead of $\mathcal{P}_{\mathcal{T}}$.

Henceforth, we can consider the more general problem of finding all the MinAs for any query $C_i \sqsubseteq_{\mathcal{S}} D_i$ and for any sub-ontology $\mathcal{S} \subseteq \mathcal{T}$, given the encoding $\phi^{all}_{\mathcal{T}(po)}/\phi^{all}_{\mathcal{T}}$ and the assumption list $\mathcal{P}_{\mathcal{S}}$. Notice that this configuration immediately allows for a more fine

[18] First, we sort the assertions computed for each ontology wrt. the number of occurrences as implicate in *rule clauses* then, following this order, we pick with a probability of 0.25 (to avoid queries which are too similar) the 30 sample assertions to be queried.

grained debugging of ontologies, for example it allows for testing the interactions among only some selected parts of an ontology or for transparently working in terms of *refutable* and *irrefutable* axioms (see, e.g., Baader et al., 2007). In fact, in many applications it is necessary to partition an ontology distinguishing two kinds of axioms: trusted ones whose correctness is established, and refutable ones for which the correctness is still doubted by the designer (or user) of the ontology. For example, if an already well-established ontology is extended, one might view the newly added axioms as refutable, but trust the previously existing part of the ontology.

In the following, for sake of clarity, we still refer to $\phi^{all}_{\mathcal{T}(po)}/\phi^{all}_{\mathcal{T}}$ and to the problem of searching all the MinAs responsible in $\mathcal{T}$ (and so using $\mathcal{P}_{\mathcal{T}}$) for a given subsumption query $C_i \sqsubseteq D_i$, but exactly the same techniques can be applied always on $\phi^{all}_{\mathcal{T}(po)}/\phi^{all}_{\mathcal{T}}$ but considering the subsets $\mathcal{S}/\mathcal{P}_{\mathcal{S}}$ instead of $\mathcal{T}/\mathcal{P}_{\mathcal{T}}$.

7.6.2 Cone-of-influence Modularization

Most of the real-world problems are too large to be entirely enumerated by our approach, so we must try to reduce the search space as much as possible. For this reason we have implemented a SAT-based form of *modularization* similar in aim to that proposed by Baader and Suntisrivaraporn (2008), Suntisrivaraporn (2009) (see Section 7.3.3), which isolates the subset, called *module*, of only (the propositional variables in $\phi^{all}_{\mathcal{T}}/\phi^{all}_{\mathcal{T}(po)}$ labeling) those axioms in $\mathcal{T}$ which might have a role in the inference of a given query $C_i \sqsubseteq_{\mathcal{T}} D_i$. After having isolated such a module of the axioms of $\mathcal{T}$ we restrict our All-SMT process to these selector variables only, as exposed above in Section 7.6.1.

For its analogy with the cone-of-influence reduction in model checking (Kurshan, 1994; Clarke et al., 1999) we call our technique *Cone-of-influence Modularization*. A *Cone-of-influence Module* is defined as follows.

Definition 12. Let $\mathcal{T}$ be an $\mathcal{EL}^+$ TBox in normal form, $C_i \sqsubseteq_{\mathcal{T}} D_i$ be an existing subsumption relation in $\mathcal{T}$, $\phi^{all}_{\mathcal{T}(po)}$ be the encoding of $\mathcal{T}$ defined in Definition 10 and $\mathcal{P}_{\mathcal{T}} \stackrel{\text{def}}{=} \{s_{[ax_i]} | ax_i \in \mathcal{T}\}$ the set of the axiom selector variables (Definition 10) for $\mathcal{T}$. The *Cone of influence* for $C_i \sqsubseteq_{\mathcal{T}} D_i$ wrt. $\phi^{all}_{\mathcal{T}(po)}$, namely $\mathcal{C}^{C_i \sqsubseteq D_i}_{\mathcal{T}}$, is the maximal set of selector variables, such that:

(i) $s_{[C_i \sqsubseteq D_i]} \in \mathcal{C}^{C_i \sqsubseteq D_i}_{\mathcal{T}}$;

(ii) if $s_{[a_i]} \in \mathcal{C}^{C_i \sqsubseteq D_i}_{\mathcal{T}}$, then $s_{[a_j]} \in \mathcal{C}^{C_i \sqsubseteq D_i}_{\mathcal{T}}$ for every $s_{[a_j]}$ of every clause $(\bigwedge_j s_{[a_j]}) \rightarrow s_{[a_i]}$ in $\phi^{all}_{\mathcal{T}(po)}$.

The *Cone-of-influence Subformula* $\phi_{\mathcal{C}^{C_i \sqsubseteq D_i}_{\mathcal{T}}}$ for $C_i \sqsubseteq_{\mathcal{T}} D_i$ wrt. $\phi^{all}_{\mathcal{T}(po)}$ is defined as $\phi_{\mathcal{C}^{C_i \sqsubseteq D_i}_{\mathcal{T}}} \stackrel{\text{def}}{=} \{c \mid c \in \phi^{all}_{\mathcal{T}(po)}, c \text{ is } ((\bigwedge_j s_{[a_j]}) \rightarrow s_{[a_i]}) \text{ and } s_{[a_i]} \in \mathcal{C}^{C_i \sqsubseteq D_i}_{\mathcal{T}}\}$. Further, we define *Cone-of-influence Module's Assumptions* for $C_i \sqsubseteq_{\mathcal{T}} D_i$ wrt. $\phi^{all}_{\mathcal{T}(po)}$, namely $\mathcal{M}^{C_i \sqsubseteq D_i}_{\mathcal{T}}$, the set of axiom selector variables $\mathcal{M}^{C_i \sqsubseteq D_i}_{\mathcal{T}} \stackrel{\text{def}}{=} \mathcal{C}^{C_i \sqsubseteq D_i}_{\mathcal{T}} \cap \mathcal{P}_{\mathcal{T}}$. Finally, the *Cone-of-influence Module* for $C_i \sqsubseteq_{\mathcal{T}} D_i$ wrt. $\phi^{all}_{\mathcal{T}(po)}$, namely $\mathcal{M}^{\mathsf{c.o.i.}}_{C_i \sqsubseteq D_i}$, is the set $\mathcal{M}^{\mathsf{c.o.i.}}_{C_i \sqsubseteq D_i} \subseteq \mathcal{T}$ of axioms such that $\mathcal{M}^{\mathsf{c.o.i.}}_{C_i \sqsubseteq D_i} \stackrel{\text{def}}{=} \{ax_i \mid s_{[ax_i]} \in \mathcal{M}^{C_i \sqsubseteq D_i}_{\mathcal{T}}\}$. ◇

The following result is a straightforward consequence of Definition 12 and Theorem 15, and it is formally proved in Appendix 7.9.

Theorem 22. *Given an $\mathcal{EL}^+$ TBox $\mathcal{T}$ in normal form and the formula $\phi^{all}_{\mathcal{T}(po)}$ as defined in Definition 10, for every pair of concept names C, D in $\mathsf{PC}_{\mathcal{T}}$, the following facts hold:*

(i) $C \sqsubseteq_{\mathcal{T}} D$ if and only if $C \sqsubseteq_{\mathcal{M}^{\mathsf{c.o.i.}}_{C \sqsubseteq D}} D$;

(ii) if $\mathcal{S}$ is a minimal subset $\mathcal{S} \subseteq \mathcal{T}$ such that $C \sqsubseteq_{\mathcal{S}} D$ and $C \not\sqsubseteq_{\mathcal{S}'} D$ for every $\mathcal{S}' \subset \mathcal{S}$ (i.e. $\mathcal{S}$ is a MinA for $C \sqsubseteq_{\mathcal{T}} D$), then $\mathcal{S} \subseteq \mathcal{M}^{\mathsf{c.o.i.}}_{C \sqsubseteq D}$,

where $\mathcal{M}^{\mathsf{c.o.i.}}_{C_i \sqsubseteq D_i} \subseteq \mathcal{T}$ is the cone-of-influence module for $C \sqsubseteq D$ wrt. $\phi^{all}_{\mathcal{T}(po)}$, as defined in Definition 12.

Importantly, point (i) of Theorem 22 is a direct consequence of point (ii) but we distinguished them because:

(i) states that the cone-of-influence module preserves the subsumption relation for which the module is computed;

(i) states that the cone-of-influence module contains all the possible MinAs responsible of such a subsumption relation.

In the practice, Cone-of-influence Modularization can be performed through a simple queue-based algorithm. Let Q be a queue containing positive occurrences of the selector variables of $\phi^{all}_{\mathcal{T}(po)}$. We initially set Q to $\{s_{[C_i \sqsubseteq D_i]}\}$ (so that it contains only the query selector variable). At every iteration a selector variable $s_{[a_i]}$ is dequeued from Q and handled by the algorithm, which enqueues into Q every other selector variable $s_{[a_j]}$ occurring negatively in any of the clauses of $\phi^{all}_{\mathcal{T}(po)}$ in which $s_{[a_i]}$ occurs positively (i.e. all clauses of the form $(\bigwedge_j s_{[a_j]}) \rightarrow s_{[a_i]}$, that is $s_{[a_i]} \vee \bigvee_j \neg s_{[a_j]}$). We run the algorithm until Q is not empty. Notice that each selector variable $s_{[a_i]}$ must be enqueue in Q at most once during the whole modularization procedure. The set of all the selector variables handled during this process until termination is the cone of influence $\mathcal{C}^{C_i \sqsubseteq D_i}_{\mathcal{T}}$ for $C_i \sqsubseteq_{\mathcal{T}} D_i$, while considering only the axioms [resp. axiom selector variables] encountered we obtain the cone-of-influence module $\mathcal{M}^{\mathsf{c.o.i.}}_{C_i \sqsubseteq D_i}$ [reps. cone-of-influence module's assumptions $\mathcal{M}^{C_i \sqsubseteq D_i}_{\mathcal{T}}$]. The pseudocode in Figure 7.5 summarizes this Cone-of-influence Modularization algorithm. Informally, the procedure of Figure 7.5 traverses the cone of influence for the query $C_i \sqsubseteq_{\mathcal{T}} D_i$ starting from the positive selector variable $s_{[C_i \sqsubseteq D_i]}$ and then backwardly going through the set of implications (clauses) taking part to the inference of $s_{[C_i \sqsubseteq D_i]}$.

Importantly, step `5.` of the algorithm in Figure 7.5 can be performed efficiently if we exploit the *two-watched-literals* technique (Moskewicz et al., 2001) (see Section 4.1) implemented in all the modern state-of-the-art SAT solvers. In fact, since all the clauses in $\phi^{all}_{\mathcal{T}(po)}$ are definite Horn clauses (see Section 7.4.2) they are all implications having exactly one and only one positive literal (that is $s_{[a_i]}$ in our exposition). Therefore, at loading/parsing time, we can force the only positive literal of each clause to be one of its two watched literals. This ensures that at step `5.`, through the two-watched-literal

```
Variable-Set compute-allMinAs (Encoding φ^all_T(po), Assumptions P_T, Query s_[C_i⊑D_i])
 1.    Q = ∅; M^{C_i⊑D_i}_T = ∅ [M^{c.o.i.}_{C_i⊑D_i} = ∅];
 2.    enqueue s_[C_i⊑D_i] in Q; mark C_i ⊑ D_i as reached;
 3.    while Q is not empty {
 4.       dequeue s_[a_i] from Q;
 5.       let C_{a_i} be the set of all the clauses of the form s_[a_i] ∨ ⋁_j ¬s_[a_j];
 6.       for each clause c ∈ C_{a_i} {
 7.          for each ¬s_[a_j] occurring in c {
 8.             if a_j is not reached
 9.                enqueue s_[a_j] in Q; mark a_j as reached;
10.             if s_[a_j] ∈ P_T
11.                M^{C_i⊑D_i}_T = M^{C_i⊑D_i}_T ∪ {s_[a_j]} [M^{c.o.i.}_{C_i⊑D_i} = M^{c.o.i.}_{C_i⊑D_i} ∪ {a_j}];
12.          }
13.       }
14.    }
15.    return M^{C_i⊑D_i}_T
```

Figure 7.5: Cone-of-influence Modularization wrt. the given encoding and query.

scheme, we can obtain the set C_{a_i} of all the clauses in which the literal $s_{[a_i]}$ appears positively in linear wrt. the cardinality of C_{a_i} itself.

Proposition 23. *Given the Horn propositional formula $\phi^{all}_{\mathcal{T}(po)}$, the set of assumptions $\mathcal{P}_\mathcal{T}$ and the query $s_{[C_i \sqsubseteq D_i]}$, the algorithm of Figure 7.5 computes the cone-of-influence module $\mathcal{M}^{\mathsf{c.o.i.}}_{C_i \sqsubseteq D_i}$ for the query $C_i \sqsubseteq_\mathcal{T} D_i$ wrt. $\phi^{all}_{\mathcal{T}(po)}$ in linear time wrt. the cone-of-influence subformula $\phi_{\mathcal{C}^{C_i \sqsubseteq D_i}_\mathcal{T}}$ for $C_i \sqsubseteq_\mathcal{T} D_i$ wrt. $\phi^{all}_{\mathcal{T}(po)}$.*

Proof. In the algorithm of Figure 7.5 we assured that each selector variable is enqueued in Q (and thus subsequently handled at step `4.`) at most once, when it is *reached* for the first time. Thus it is easy to see that the algorithm executes a number of iterations linear wrt. the number of selector variables included in Q, that is exactly the size of $\mathcal{C}^{C_i \sqsubseteq D_i}_\mathcal{T}$. In each iteration, once dequeued the selector variable $s_{[a_i]}$ at step `4.`, the algorithm: (i) gets the clauses C_{a_i} (step `5.`), (ii) executes a fixed number of instructions for every literal of every clause $c \in C_{a_i}$ (steps `6.-13.`). Since (i) is performed in linear time wrt. $|C_{a_i}|$, exploiting the two-watched-literals scheme, the complexity of the algorithm is dominated by the operations (ii). Thus, since wlog. we can assume that every clause has at-most four literals, the modularization algorithm of Figure 7.5 terminates in linear time wrt. the number of clauses handled by the algorithm, that is $O(|\phi_{\mathcal{C}^{C_i \sqsubseteq D_i}_\mathcal{T}}|)$ (Definition 12). □

Since the Cone-of-influence Modularization algorithm is not computationally expensive and since the modules computed are typically orders of magnitude smaller than $\mathcal{T}$, this technique drastically improves the performance of our approach.

Remark 5. Importantly, this technique works directly on the propositional input formula $\phi^{all}_{\mathcal{T}}/\phi^{all}_{\mathcal{T}(po)}$, with no need for re-compute something from $\mathcal{T}$ or for any other form of $\mathcal{EL}^+$ reasoning. From the perspective of our approach, this is a point in favor of our cone-of-influence modularization since we can solve every query working directly on the SAT formula regardless the original ontology. Further, in a more general case, through exactly the same procedure we can obtain "for free" the cone-of-influence module' assertions $\mathcal{M}^{C_i \sqsubseteq D_i}_{\mathcal{S}}$ for any desired sub-ontology $\mathcal{S}$ of $\mathcal{T}$. [19] Once $\mathcal{C}^{C_i \sqsubseteq D_i}_{\mathcal{T}}$ has been computed from $\phi^{all}_{\mathcal{T}(po)}$, it is simply $\mathcal{M}^{C_i \sqsubseteq D_i}_{\mathcal{S}} \stackrel{\text{def}}{=} \mathcal{C}^{C_i \sqsubseteq D_i}_{\mathcal{T}} \cap \mathcal{P}_{\mathcal{S}}$. ◇

Example 7.6.1. Consider the case of computing the cone-of-influence module $\mathcal{M}^{\mathsf{c.o.i.}}_{\mathsf{Pericarditis} \sqsubseteq \mathsf{HeartDisease}}$ for the query $\mathsf{Pericarditis} \sqsubseteq_{\mathcal{O}_{\mathsf{med}}} \mathsf{HeartDisease}$ wrt. the encoding $\phi^{all}_{\mathcal{O}_{\mathsf{med}}(po)}$ of the sample ontology $\mathcal{O}_{\mathsf{med}}$ (Example 3.5.1). Here below we rewrite the encoding $\phi^{all}_{\mathcal{O}_{\mathsf{med}}(po)}$ from Example 7.4.2, where the query $\mathsf{Pericarditis} \sqsubseteq_{\mathcal{O}_{\mathsf{med}}} \mathsf{HeartDisease}$ corresponds to the assumption b_9. We underline the clauses and literals involved during the modularization process:

$$
\begin{array}{lcl}
\phi^{all}_{\mathcal{O}_{\mathsf{med}}(po)} \stackrel{\text{def}}{=} & s_{[a_4]} \wedge s_{[a_7]} \rightarrow s_{[b_1]} \quad \wedge \quad s_{[a_4]} \wedge s_{[a_7]} \rightarrow s_{[b_2]} & \texttt{(a)} \\
\wedge & s_{[a_5]} \wedge s_{[a_7]} \rightarrow s_{[b_3]} \quad \wedge \quad s_{[a_5]} \wedge s_{[a_7]} \rightarrow s_{[b_4]} & \texttt{(b)} \\
\wedge & \underline{\underline{s_{[a_6]}}} \wedge \underline{\underline{s_{[a_7]}}} \rightarrow \underline{s_{[b_5]}} \quad \wedge \quad s_{[a_6]} \wedge s_{[a_7]} \rightarrow s_{[b_6]} & \texttt{(c)} \\
\wedge & \underline{\underline{s_{[a_6]}}} \wedge \underline{\underline{s_{[a_3]}}} \wedge \underline{\underline{s_{[a_{11}]}}} \rightarrow \underline{s_{[b_7]}} \quad \wedge \quad \underline{\underline{s_{[b_7]}}} \wedge \underline{\underline{s_{[a_0]}}} \rightarrow \underline{s_{[b_8]}} & \texttt{(d)} \\
\wedge & \underline{s_{[b_5]}} \wedge \underline{\underline{s_{[b_8]}}} \wedge \underline{\underline{s_{[a_8]}}} \rightarrow \underline{s_{[b_9]}} \quad \wedge \quad s_{[b_9]} \wedge s_{[a_9]} \rightarrow s_{[b_{10}]} & \texttt{(e)}
\end{array}
$$

In particular: (i) literals queried through the two-watched-literals technique are wavy underlined, (ii) literals inserted in the queue to be subsequently handled are singly underlined and (iii) literals representing axioms are doubly underlined. This has been said, the set of the axiom/assumption selector variables marked (ii) and (iii) represent the cone of influence $\mathcal{C}^{\mathsf{Pericarditis} \sqsubseteq \mathsf{HeartDisease}}_{\mathcal{O}_{\mathsf{med}}}$, in particular the set of the axioms represented by the selector variables marked (iii) represent the module $\mathcal{M}^{\mathsf{c.o.i.}}_{\mathsf{Pericarditis} \sqsubseteq \mathsf{HeartDisease}}$, so $\mathcal{M}^{\mathsf{c.o.i.}}_{\mathsf{Pericarditis} \sqsubseteq \mathsf{HeartDisease}} = \{a_0, a_3, a_6, a_7, a_8, a_{11}\}$. Concretely the computation of the module starts from the positive literal $s_{[b_9]}$. The first clause in row `(e)` is returned in constant time thanks to the two-watched-literals scheme, so that the literals $s_{[b_8]}, s_{[b_5]}$ are enqueued and a_8 added to the initially empty module. Similarly, from $s_{[b_8]}$ (second clause of row `(d)`), $s_{[b_7]}$ is enqueued and a_0 included into the module. In the last two steps of modularization the other axioms a_6, a_7 and a_3, a_{11} are added to the module consequently to the analysis of the clauses implying $s_{[b_5]}$ (first clause of row `(c)`) and $s_{[b_7]}$ (first clause of row `(d)`), respectively. Notice that, a_9 is not included in the cone-of-influence module for $\mathsf{Pericarditis} \sqsubseteq \mathsf{HeartDisease}$ unlike in the reachability-based module for $\mathsf{Pericarditis}$ (see Example 7.3.4 for the computation of $\mathcal{M}^{\mathsf{reach}}_{\mathsf{Pericarditis}}$). ◇

Even if similar in aim *Cone-of-influence Modularization* and *Reachability-based Modularization* (Baader & Suntisrivaraporn, 2008; Suntisrivaraporn, 2009) (see Section 7.3.3) present many differences.

[19] For example the user can decide to work on a sub-ontology $\mathcal{S}$ of $\mathcal{T}$; then modularization allows for further "refining" the search space for every single different query wrt. $\mathcal{S}$.

First, differently from Reachability-based Modularization, in which modules are extracted from the original axioms of $\mathcal{T}$ proceeding forward on the base of *syntactic* interactions between signatures of the axioms (starting from the given signature), Cone-of-influence Modularization extracts modules in a SAT-based manner proceeding backward according to the propositional interactions between the clauses of the encoding $\phi^{all}_{\mathcal{T}(po)}$ (starting from the query selector variable). Therefore, even if Cone-of-influence Modularization relies directly on the propositional encoding instead of reasoning in terms of description logic, it is a *semantic* modularization technique, unlike Reachability-based Modularization. In fact, the SAT formula $\phi^{all}_{\mathcal{T}(po)}$ includes (and hides) the semantic of $\mathcal{T}$ handled during the construction of the formula.

Second, Reachability-based Modularization does not depend on the specific set of completion rules adopted, whilst the cone-of-influence technique does. Indeed, reachability-based modules are computed before the classification independently from the completion rules, while cone-of-influence modules are computed after the classification on the base of the encoding $\phi^{all}_{\mathcal{T}(po)}$, whose definition depends on the specific set of completion rules adopted. A different cone-of-influence modules would be produced if a different set of completion rules was used.

Third, we state that our approach is more precise than the reachability-based one, i.e. we state that it extracts smaller modules. Further than relying on semantic information from the classification step, our modularization technique exploits both the information given by the query $C_i \sqsubseteq_{\mathcal{T}} D_i$. Notice, in fact, that while Cone-of-influence Modularization computes a module "dependent" both from C_i and D_i (i.e. dependent exactly from the subsumption relation $C_i \sqsubseteq D_i$) Reachability-based Modularization computes a module "dependent" only from C_i. For example, the reachability-based module computed for the query $C_i \sqsubseteq D_i$ is the same computed for the query $C_i \sqsubseteq E_i$, because it includes everything is syntactically reachable from C_i in both cases; more precisely, it also includes in the module everything else is syntactically reachable either from D_i or from E_i. This fact let us suppose that some kind of combination of a forward reachability from C_i and a backward one from D_i could be used to refine the reachability-based technique.

Example 7.6.2. The following example shows in the ontology NOT-GALEN the differences between the reachability-based module $\mathcal{M}^{\mathsf{reach}}_{\mathsf{Lithium}}$ and the cone-of-influence module $\mathcal{M}^{\mathsf{c.o.i.}}_{\mathsf{L.}\sqsubseteq\mathsf{C.S.}}$ for the existing subsumption relation $\mathsf{Lithium} \sqsubseteq_{\text{NOT-GALEN}} \mathsf{ChemicalSubstance}$.

$$\begin{array}{rl}
\mathsf{Lithium} & \sqsubseteq \mathsf{ElementalChemical} \sqcap \mathsf{Metal} \\
\mathsf{ElementalChemical} & \sqsubseteq \mathsf{ChemicalSubstance} \\
\mathsf{Metal} & \sqsubseteq \mathsf{ComplexChemicals} \\
\mathsf{ComplexChemicals} & \sqsubseteq \mathsf{ChemicalSubstance} \\
\hline
\mathsf{ChemicalSubstance} & \sqsubseteq \mathsf{Substance} \\
\mathsf{Substance} & \sqsubseteq \mathsf{GeneralisedSubstance} \\
\mathsf{GeneralisedSubstance} & \sqsubseteq \mathsf{DomainCategory} \\
\mathsf{DomainCategory} & \sqsubseteq \mathsf{TopCategory} \\
\mathsf{Substance} & \sqsubseteq \exists \mathsf{hasCountability.infinitelyDivisible} \\
\mathsf{hasCountability} & \sqsubseteq_r \mathsf{StructuralAppearanceModifier} \\
\mathsf{StructuralAppearanceModifier} & \sqsubseteq_r \mathsf{StructuralModifierAttribute} \\
\mathsf{StructuralModifierAttribute} & \sqsubseteq_r \mathsf{hasFeature}
\end{array}
\qquad
\begin{array}{rl}
\mathsf{hasFeature} & \sqsubseteq_r \mathsf{FeatureStateAttribute} \\
\mathsf{FeatureStateAttribute} & \sqsubseteq_r \mathsf{ModifierAttribute} \\
\mathsf{ModifierAttribute} & \sqsubseteq_r \mathsf{DomainAttribute} \\
\mathsf{DomainAttribute} & \sqsubseteq_r \mathsf{Attribute} \\
\mathsf{infinitelyDivisible} & \sqsubseteq \mathsf{mass} \\
\mathsf{mass} & \sqsubseteq \mathsf{CountabilityStatus} \\
\mathsf{CountabilityStatus} & \sqsubseteq \mathsf{AbstractStatus} \\
\mathsf{AbstractStatus} & \sqsubseteq \mathsf{Status} \\
\mathsf{Status} & \sqsubseteq \mathsf{Aspect} \\
\mathsf{Aspect} & \sqsubseteq \mathsf{ModifierConcept} \\
\mathsf{ModifierConcept} & \sqsubseteq \mathsf{DomainCategory}
\end{array}$$

While $\mathcal{M}^{\mathsf{reach}}_{\mathsf{Lithium}}$ consists of all the 23 listed axioms of NOT-GALEN, $\mathcal{M}^{\mathsf{c.o.i.}}_{\mathsf{L.}\sqsubseteq\mathsf{C.S.}}$ consists of

only the first 4 axioms (listed above the separation line).

The following Proposition 24 proves the statement above, i.e. that Cone-of-influence Modularization produces smaller modules than Reachability-based Modularization. Proof is included in Appendix 7.9.

Proposition 24. *Given an $\mathcal{EL}^+$ TBox $\mathcal{T}$ in normal form and the formula $\phi^{all}_{\mathcal{T}(po)}$ as defined in Definition 10, for every pair of concept names C, D in $\mathsf{PC}_{\mathcal{T}}$ it holds $\mathcal{M}^{\mathsf{c.o.i.}}_{C\sqsubseteq D} \subseteq \mathcal{M}^{\mathsf{reach}}_{C}$, where $\mathcal{M}^{\mathsf{reach}}_{C}$ and $\mathcal{M}^{\mathsf{c.o.i.}}_{C\sqsubseteq D}$ are, respectively, the reachability-based module (Definition 8) and cone-of-influence module (Definition 12) for $C \sqsubseteq D$.*

7.6.3 Theory Propagation

Another important improvement to the basic procedure for the all MinAs enumeration we have introduced in the new version of $\mathcal{EL}^+$2SAT is that of exploiting *early pruning* and *theory propagation*, two well-known techniques from SMT (see, e.g., Sebastiani, 2007b; Barrett et al., 2009).

The idea is that the $\mathcal{T}$-solver can work also on partial truth assignments in order to early discover new conflicting assignments. Given the truth assignment μ_k on $\mathcal{P}_{\mathcal{T}}$ generated by `DPLL1`, during the propagation of μ_k the $\mathcal{T}$-solver can recognize if the partial truth assignment $\mu_k{}' \subset \mu_k$ causes the unit-propagation of one (or more) $\neg s_{[ax_j]}$ s.t. $s_{[ax_j]} \in \mathcal{P}_{\mathcal{T}}$ and $s_{[ax_j]}$ is unassigned. In this case, the antecedent clause of $\neg s_{[ax_j]}$ can be fed to `analyze_conflict` in the `DPLL2` (internal to the $\mathcal{T}$-solver), which returns the clause $s_{[C_i\sqsubseteq D_i]} \vee \neg\mu_k{}''$ s.t. $\mu_k{}'' \subseteq \mu_k{}' \subset \mu_k$ and $\neg s_{[C_i\sqsubseteq D_i]} \wedge \mu_k{}''$ causes the propagation of $\neg s_{[ax_j]}$. [20] Intuitively, this is equivalent to say that, if $\neg s_{[C_i\sqsubseteq D_i]} \wedge \mu_k{}' \wedge s_{[ax_j]}$ is passed to the $\mathcal{T}$-solver, then `DPLL2` would return `conflict` and the $\mathcal{T}$-conflict clause $s_{[C_i\sqsubseteq D_i]} \vee \neg\mu_k{}'' \vee \neg s_{[ax_j]}$. Thus $\mu''_k \wedge s_{[ax_j]}$ represents a non-minimal set of axioms causing the inconsistency of $\phi^{all}_{\mathcal{T}(po)}$, that can be further minimized by the linear algorithm of Figure 7.2, and which is *theory propagated* form the $\mathcal{T}$-solver in order to early discover new MinAs even before being generated from `DPLL1`. [21] Also this technique can transparently work wrt. a set of assertion variables $\mathcal{P}_{\mathcal{S}}$ in place of $\mathcal{P}_{\mathcal{T}}$, where $\mathcal{S} \subseteq \mathcal{T}$ is any sub-ontology of $\mathcal{T}$.

Notice that from the same assignment μ_k theory propagation can return many different $\mu''_k \wedge s_{[ax_j]}$ conflicting assignments, but since they are non-minimal they potentially include either different or the same MinA(s). This fact has the drawback that minimization can be invoked many times without leading to new MinAs. Thus, we need a (preferably fast) check which decides whether the given assignment contains a new MinA or not, avoiding the minimization process in the second case. For this reason we extended $\mathcal{EL}^+$2SAT

[20] As before (Section 7.4.2) we assume that `analyze_conflict` uses the Decision Scheme.

[21] In lazy SMT *early pruning*, instead, simply exploits the idea of intermediate partial checks during the construction of assignments so that if a conflict is early discovered by the partial check it can significantly prune the time necessary to extend such a partial assignment to a total and satisfiable one. In our case, since we are looking for unsatisfiable assignments and the single CPU time taken by one satisfiability-check of the $\mathcal{T}$-solver is negligible, *early pruning* is not meaningful.

including a third DPLL instance, namely `DPLL3`. `DPLL3` runs over the variables $\mathcal{P}_\mathcal{T}$ [resp. $\mathcal{P}_\mathcal{S}$] and initially the empty set of clauses. Every time a new MinA $\mathcal{T}_k^*$ [resp. $\mathcal{S}_k^*$] is found we immediately add the clause $\neg\mathcal{S}_k^*$ [resp. $\neg\mathcal{S}_k^*$] to `DPLL3`. In order to "quickly check" if an assignment $\mu_k{}'$ does not conflict with the previously found MinAs, it is sufficient to check if `DPLL3` under the assumption of $\mu_k{}'$ returns `sat`. Notice that `DPLL1` cannot be used for this purpose because it is used to enumerate the assignments and, thus, we cannot change its internal state.

7.6.4 Refining Cone-of-influence Modularization

After a preliminary evaluation, we discovered an undesired behavior of our modularization algorithm when applied to the encoding of originally non-normal ontologies.

Analyzing the cone-of-influence modules produced by $\mathcal{EL}^+$2SAT, we find out a drawback of the labeling scheme we have adopted to handle non-normal ontologies (see Section 7.4.2), which affects the Cone-of-influence Modularization and causes unexpected growth in the modules computed by the procedure of Figure 7.5. This is due to the inclusion in the module of labels representing one non-normal axiom but more non-normal ones and, especially, due to the role played by the label $s_{[a_0]}$, with which we represent all the definition axioms resulting from the normalization process (Section 7.4.2). The inclusion of such labels may cause the iterative introduction of useless assumptions/axioms in the cone of influence. Practically speaking, this problem results particularly evident in some queries for the GALEN-based benchmark ontologies FULL-GALEN and NOT-GALEN. One of the main sources of this undesired drawback is the normalization of complex equivalence axioms like axiom m_3 in Example 7.3.1 (Section 7.3.1). Notice that from one axiom such as m_3 five normalized axioms are introduced: accordingly to the scheme adopted (see Section 7.4.2) the three top-level axioms (the first two and the fourth of Example 7.3.1) are all labeled with the same selector variable $s_{[m_3]}$, while the other two definition axioms are labeled with the unique selector variable $s_{[m_0]}$. Structures of non-normal-form axioms similar to the one of m_3 are extremely frequent in FULL-GALEN and NOT-GALEN, where such structures are diffusely used in order to express unique definitions of concept names. Once encoded, these structures causes cyclic dependencies and mutual deductions involving the labels assigned to such axioms which, in turn, may cause the inclusion of many other assertions/axioms from other occurrences of the same labels.

In order to limit the effects of this observed drawback, we further refine our Cone-of-influence Modularization procedure exposed in Section 7.6.2. The refinement is based on the following two simple rules. We observed that we can earlier stop the cone-of-influence iterative construction exposed in Figure 7.5, by excluding at step `6.` all the clauses c such that:

(a) $c \in C_{a_i}$, when a_i is an *axiom* selector variable,

(b) or $\neg s_{[C_i \sqsubseteq D_i]} \in c$ (i.e. $s_{[C_i \sqsubseteq D_i]}$ occurs negatively in c), with $C_i \sqsubseteq_\mathcal{T} D_i$ the input query.

These two simple rules allow for reducing the annoying effects on modularization caused by the labeling of the normalization of complex axioms. We remark that this refinement does

not concern the definition and the properties of Cone-of-Influence Modularization which, indeed, is defined on normal form TBoxes; here we only discuss its practical implementation when combined with the labeling for non-normal axioms proposed in Section 7.4.2. Notice that (a) can be applied under the assumption that axioms are non-otherwise deducible concept inclusions. If it is not the case the application of (a) can cause the exclusion of those MinAs in which one particular axiom a_i can be deduced from the combination of other axioms. Thus the choice on whether to apply the refinement (a) or not, should be left to the user.

Alternatively, the highlighted undesired phenomenon can be completely avoided, without the need of any refinement, by directly encoding the normal-form conservative extension of the original ontology and, therefore, by handling (and labeling) every resulting axiom as an original one. Notice that we can chose this way in order to compute exact cone-of-influence modules on the set of the normalized axioms and, then, to go back to exact modules expressed terms of original axioms.

7.6.5 Working on Smaller Ontologies: $\mathcal{EL}^+$2SAT$\times$2

Finally, we try to push further the boundaries of practical axiom pinpointing in $\mathcal{EL}^+$ by exploiting the reasoning capabilities of $\mathcal{EL}^+$2SAT directly on reduced-size input ontologies. In this way we intend to reduce the total overhead during the $\mathcal{EL}^+$2SAT enumeration by working directly on smaller formulas, instead of indirectly working (by mean of the use of a restricted number of assumptions) on a sub-formula of the whole, huge, encoding $\phi^{all}_{\mathcal{T}}$. The idea is to exploit the benefits of a combined use of $\mathcal{EL}^+$2SAT, in which we separate the modularization and the search of MinAs processes in two distinct executions of $\mathcal{EL}^+$2SAT on two different formulas. So, we first exploit modularization in order to extract a query-dependent sub-ontology on which, subsequently, we run our —complete— $\mathcal{EL}^+$2SAT approach, i.e. both the encoding and the all-MinAs search phases of $\mathcal{EL}^+$2SAT.

Concretely, given a subsumption query $C_i \sqsubseteq_{\mathcal{T}} D_i$, the combined approach carries out the following three phases:

1. having loaded the encoding $\phi^{all}_{\mathcal{T}(po)}$ of the input ontology $\mathcal{T}$, it extracts the query-dependent sub-ontology $\mathcal{O} = \mathcal{M}^{\text{c.o.i.}}_{C_i \sqsubseteq D_i}$ through the modularization routine of $\mathcal{EL}^+$2SAT;
2. it classifies $\mathcal{O}$ producing the $\mathcal{EL}^+$2SAT's Horn-SAT encoding $\phi^{all}_{\mathcal{O}(po)}$;
3. it exhaustively searches MinAs through $\mathcal{EL}^+$2SAT on $\phi^{all}_{\mathcal{O}(po)}$.

Concerning this combined approach we remark some facts:

– It is convenient to run two distinct instances of $\mathcal{EL}^+$2SAT: (i) a first permanent instance of $\mathcal{EL}^+$2SAT that, for every query, realizes step 1. by performing Cone-of-influence Modularization on the full encoding $\phi^{all}_{\mathcal{T}(po)}$, and (ii) a second dynamic

instance of $\mathcal{EL}^+$2SAT which carries out steps 2. and 3. on the extracted sub-ontology. Notice, in fact, that this latter instance works, for every query, on a different sub-ontology which needs to be handled from scratch.

– Once the encoding $\phi^{all}_{\mathcal{T}}$ of the input ontology has been loaded by the first $\mathcal{EL}^+$2SAT instance (i), a cone-of-influence module can be computed in negligible time for an unlimited number of queries, with no need for reloading the encoding.

– Wrt. the possibly huge size of the original ontology, modularization reduces each problem to a very small sub-ontology $\mathcal{O}$. Thus, the time taken by the second instance of $\mathcal{EL}^+$2SAT (ii) in encoding a few-hundreds-of-axioms ontology $\mathcal{O}$ into $\phi^{all}_{\mathcal{O}(po)}$, and then loading $\phi^{all}_{\mathcal{O}(po)}$ is negligible (see, e.g, the encoding/loading time spent for NOT-GALEN —which has thousands of axioms— in our preliminary evaluation of Section 7.5).

We call $\mathcal{EL}^+$2SAT×2 this combined approach. From the above remarked points it is immediately clear that, overall, $\mathcal{EL}^+$2SAT×2 allows for a faster enumeration of the MinAs. In fact, while phases 1. and 2. are expected to be negligible even for really huge input ontologies, in phase 3. the second instance of $\mathcal{EL}^+$2SAT runs over the classification of tens or hundreds of axioms instead of on the classification of —tens or hundreds of— thousands of axioms. This can strongly reduce the overhead of every internal call to the $\mathcal{T}$-solver during the MinAs enumeration, leading in total to a drastic enhancement, beyond the previous use of modules via assumptions.

7.7 An Extensive Experimental Evaluation

We have implemented the new optimizations exposed in Section 7.6, improving our tool $\mathcal{EL}^+$2SAT, and we performed an extensive experimental evaluation on the same five ontologies we used during our preliminary evaluation exposed in Section 7.5, which are: SNOMED-CT'09, NCI, GENEONTOLOGY NOT-GALEN and FULL-GALEN (see Section 7.1 and Section 7.5 for more details). For all these ontologies we focused on the all-MinAs problem and, in particular, we compared different versions of $\mathcal{EL}^+$2SAT and the other $\mathcal{EL}^+$-specific tool CEL (Baader et al., 2006a) (`v.1.1.2 Oct'09` [22]), which implements also the Reachability-based Modularization of Suntisrivaraporn (2009).

For every ontology we run two groups of 50 test subsumption/pinpointing queries each, extracted among the subsumption relations which are deducible from $\mathcal{T}$ (thus there exists at least one MinA for every chosen query). A first group of queries, that we call `random`, has been generated picking 50 random queries among all the possible existing and non-trivial ones (in order to understand the behavior of the tool on a normal debugging circumstance). A second group of queries, that we call `selected`, has been chosen trying to select the "hardest" possible subsumption relations, that is those potentially including

[22] As far as we know this is the last available version from `http://lat.inf.tu-dresden.de/systems/cel/` and `http://code.google.com/p/cel/`.

the highest number of different MinAs (in order to understand the behavior of the tool in the most difficult debugging situations). We selected them considering the first 50 subsumptions $C_i \sqsubseteq D_i$ whose selector variable $s_{[C_i \sqsubseteq D_i]}$ appears more frequently positively in the encoding $\phi^{all}_{\mathcal{T}(po)}$.

All tests have been run on the same machines and configurations of Section 7.5, again setting a 1000 seconds timeout for every query. In the following we present and discuss the results of this evaluation, reporting both the CPU times taken by the tools to compute the pinpointed MinAs and the number of MinAs found. Further, we report also some data concerning the size of the modules obtained applying the Cone-of-influence Modularization for $\mathcal{EL}^+$2SAT (see Section 7.6.2) and the Reachability-based Modularization for CEL. It is worth noticing that our modularization technique needs a negligible time ($\leq 10^{-2}$ sec.) in all the test cases we performed except for $\mathcal{EL}^+$2SAT (b) (i.e. without the refinement discussed in Section 7.6.4) in the FULL-GALEN problems, where it takes 0.3 and 0.5 sec. on average, for the `random` and the `selected` tests, respectively. In the evaluation we executed the following five different versions of $\mathcal{EL}^+$2SAT, which gradually introduce the optimization exposed in Section 7.6:

(a) the first early-prototype used in the feasibility evaluation of Section 7.5 and presented by Sebastiani and Vescovi (2009b) (which does not include any of the enhancements exposed in Section 7.6);

(b) a first enhanced version including *Cone-of-influence Modularization* (see Section 7.6.2);

(c) a second enhanced version including *Cone-of-influence Modularization* and *Theory Propagation* (see Section 7.6.3);

(d) the version including both the previous enhancements and also the "fast check" presented in the second part of Section 7.6.3 (which early-discards nMinAs not containing new MinAs);

(e) the last version including all the previously mentioned optimizations and also the modularization refinement presented in Section 7.6.4.

Moreover, we run the $\mathcal{EL}^+$2SAT $\times$ 2 combined approach, [23] that we shortly distinguish with the symbol "$\times$2".

In Tables 7.3 and 7.4 we report in detail the results of each single query in the case of the more challenging ontology SNOMED-CT'09. In Tables 7.6 and 7.7, instead, the detailed results for FULL-GALEN are listed. For a better readability of this section we moved in Appendix 7.10 the same kind of tables for the other three ontologies.

In these tables we compare with $\mathcal{EL}^+$2SAT $\times$ 2 and CEL the detailed results of three representative variants of $\mathcal{EL}^+$2SAT: (a) representing its basic version, (b) which witness the benefits of modularization, and (e) which shows the total effect of all the proposed

[23] We run two instance of $\mathcal{EL}^+$2SAT version (e) for SNOMED-CT, FULL-GALEN and NOT-GALEN, two instance of $\mathcal{EL}^+$2SAT (d) for NCI and GENEONTOLOGY.

optimizations. For each query, both for every $\mathcal{EL}^+$2SAT variants and for CEL, we expose: in the left-most part of the tables the data concerning the modularization techniques, in the central part the CPU times required by the tools, in the right-most part the number of MinAs resulting from the search. In particular, in the left-most part of the tables we expose the size of the modules extracted through the Cone-of-influence Modularization for $\mathcal{EL}^+$2SAT side by side to the size of the modules produced by the Reachability-based Modularization for CEL. Concerning $\mathcal{EL}^+$2SAT we expose, from left to right, the size of the modules produced in the versions (b)-(d) and, respectively, the size of the modules resulting after the refinement included in the last version (e). In the central part, for every compared $\mathcal{EL}^+$2SAT version, we report the total elapsed time from the beginning of the search until the last MinA found is returned, and for CEL we report the termination time of the search. We avoid to consider the encoding time for $\mathcal{EL}^+$2SAT (that can be done off-line once forever) and the loading times of the ontology/encoding (that can be done once for the whole test session) both for CEL and $\mathcal{EL}^+$2SAT. [24]

$\mathcal{EL}^+$2SAT $\times$ 2 deserves a separate mention. Since we must evaluate the overall performance of such combined approach, we must count the CPU time arising from every of the three phases of $\mathcal{EL}^+$2SAT $\times$ 2 (see Section 7.6.5). For what concerns the first stable instance of $\mathcal{EL}^+$2SAT only the modularization time (phase 1.) must be considered, since the whole ontology's encoding can be loaded once and exploited for every modularization query, without reloading. Instead, all the CPU time required during the operations performed by the second instance of $\mathcal{EL}^+$2SAT must be taken into account, because they refer each time to a different sub-ontology. The second $\mathcal{EL}^+$2SAT instance must respectively: (i) encode the extracted sub-ontology, (ii) load the new encoding, (iii) perform the MinAs' search; notice that phase 2. of the combined approach is represented by (i), while the time spent in (ii) and (iii) falls into phase 3. The times reported in the column referring to $\mathcal{EL}^+$2SAT $\times$ 2 consist in the total of these four CPU times, i.e. the sum of the modularization time and of the (i)-(iii) times. In order to show that phase 1 and 2 are negligible (as predicted in Section 7.6.5) we report separately in the column titled "*enc.*" the sum of the Cone-of-influence Modularization and of the sub-ontology's encoding times. Finally, in the right-most part of the tables we expose the number of MinAs pinpointed fer every version/tool.

For what concerns $\mathcal{EL}^+$2SAT, in the time columns we identify with a $*$ each test case whose enumeration has completely terminated within the timeout, while for CEL we mark with a $*$ each problem on which CEL has *regularly and completely* terminated within the timeout. Having an accurate comparison between $\mathcal{EL}^+$2SAT and CEL, in fact, is problematic for two reasons. First, the current version of CEL [25] stops after reporting

[24] These times for $\mathcal{EL}^+$2SAT can be found in the preliminary empirical evaluation of Section 7.5, while for CEL they are on average: 25.1, 4.0, 3.4, 1.4, 0.4 sec. for SNOMED'09, FULLGALEN, NCI, GENEONT., NOTGALEN respectively. Obviously, the same times are valid also for every other further experiment reported later in this work.

[25] A completely reimplemented version of CEL is currently under development (personal communication from the authors).

at most 10 MinAs. [26] Second, all the MinAs found by CEL are reported all at the end of the search, so that it is not possible to interrupt the execution of CEL at the expiration of the 1000 sec. timeout having the partial results. Thus we chose to run CEL without a timeout. [27] Due to these factors, we think that it is not completely meaningful to compare each other the execution times of $\mathcal{EL}^+$2SAT and CEL. However, we reported both of them in order to give an idea of the performance of the two tools. This have been said, in order to have a better view of the results we split the tables in horizontal groups, depending on whether the MinAs search has been completed by $\mathcal{EL}^+$2SAT (in its different versions) and/or by CEL. Thus, the problems which have been completely solved both by $\mathcal{EL}^+$2SAT and by CEL are exposed in the top-most part of the tables, while in the lower part we report the problems which (eventually) neither $\mathcal{EL}^+$2SAT nor CEL completely solved. [28] Notice that when CEL either stops due to the limit of 10 MinAs or runs over the 1000 sec. time limit on one problem we can not consider such a problem "completely solved" by CEL. The other groups of problems are included in the middle parts of the tables.

In Table 7.5, instead, we summarize and compare the results of our empirical evaluation for all the five benchmark ontologies and their two test suites `random` and `selected`, respectively, on all the $\mathcal{EL}^+$2SAT variants and CEL. The results are exposed from the most challenging ontology SNOMED'09 (on top) to the least challenging one NOTGALEN (bottom). First, in Table 7.5(a) we report numerical information. In the left-most block of the table we counted, for every variant, the number of $\mathcal{EL}^+$2SAT computations completely terminating the enumeration before the timeout; for CEL we indicated the number of computations regularly and completely terminating the search (i.e. not stopping at 10 MinAs) within a total time less or equal to the fixed timeout. In the central block we counted the total amount of MinAs pinpointed by all the tools and variants before termination. In the right-most block we present the average size of the modules extracted by the two tools. Next, in Table 7.5(b) we report the summary information about time performances. For every information reported we compute both the 50th and the 90th percentiles; notice, in fact, that problems can present an extremely high variance in the performance measures, thus showing only an average value would not be significant. In the left-most block we report the statistics concerning the time elapsed during the $\mathcal{EL}^+$2SAT executions between the output of two different MinAs, i.e. either the time spent by $\mathcal{EL}^+$2SAT in order to discover the first MinA or the time elapsed between each pair of consecutive MinAs detections. In the central block we report, instead, the statistics concerning the total elapsed times, summarizing the single data reported in the detailed tables above introduced. Finally, in the right-most block, we listed the time required by

[26] We have reported this problem to the authors, but so far they were not able to provide us a version without this problem.

[27] Usually CEL stops before the 1000 sec. timeout. Very often this happens 'cause the limit of 10 MinAs has been reached. However, there are cases in the FULL-GALEN ontology in which CEL runs longer than 1000 sec. even if the problem includes less than 10 MinAs.

[28] In order to allow our empirical evaluation for being repeatable and comparable we report on the left side of each line in the tables the index of the problem within the test suite.

	Module size			MinAs search time (sec.)						#MinAs				
	$\mathcal{EL}^+$2SAT		CEL	$\mathcal{EL}^+$2SAT					CEL	$\mathcal{EL}^+$2SAT				CEL
	(b)	(e)		(a)	(b)	(e)	*enc.*	×2		(a)	(b)	(e)	×2	
1	9	9	31	19.5	3.7*	1.1*	*0.0*	0.0*	21.4*	2	3	3	3	3
2	2	2	17	6.5	0.6*	0.6*	*0.0*	0.0*	20.5*	1	1	1	1	1
5	3	2	12	6.5	0.6*	0.6*	*0.0*	0.0*	20.5*	1	1	1	1	1
6	7	7	9	6.7	0.8*	0.7*	*0.0*	0.0*	20.6*	1	1	1	1	1
7	7	7	9	19.5	0.9*	1.0*	*0.0*	0.0*	20.8*	2	2	2	2	2
10	4	4	15	13.0	0.7*	0.8*	*0.0*	0.0*	20.8*	2	2	2	2	2
11	10	10	27	65.4	7.0*	2.2*	*0.0*	0.0*	22.0*	4	5	5	5	5
12	5	5	21	6.6	0.7*	0.7*	*0.0*	0.0*	20.7*	1	1	1	1	1
13	10	10	47	38.7	21.1*	3.0*	*0.0*	0.0*	22.6*	3	6	6	6	6
17	11	11	22	13.3	1.0*	1.3*	*0.0*	0.0*	21.3*	2	2	2	2	2
18	6	6	10	6.6	0.7*	0.7*	*0.0*	0.0*	20.6*	1	1	1	1	1
19	2	2	3	6.5	0.6*	0.6*	*0.0*	0.0*	20.5*	1	1	1	1	1
21	8	8	42	26.2	1.1*	1.3*	*0.0*	0.0*	21.4*	3	3	3	3	3
23	5	5	11	6.6	0.7*	0.7*	*0.0*	0.0*	20.6*	1	1	1	1	1
24	4	4	6	6.6	0.7*	0.7*	*0.0*	0.0*	20.6*	1	1	1	1	1
28	3	3	42	6.5	0.6*	0.6*	*0.0*	0.0*	20.7*	1	1	1	1	1
29	2	2	54	6.5	0.6*	0.6*	*0.0*	0.0*	20.6*	1	1	1	1	1
31	3	3	6	6.5	0.6*	0.6*	*0.0*	0.0*	20.5*	1	1	1	1	1
32	7	7	9	25.8	0.9*	1.0*	*0.0*	0.0*	20.8*	2	2	2	2	2
33	8	8	14	38.7	1.2*	1.3*	*0.0*	0.0*	21.3*	3	3	3	3	3
36	6	6	8	6.6	0.7*	0.7*	*0.0*	0.0*	20.6*	1	1	1	1	1
39	9	9	10	6.7	0.8*	0.8*	*0.0*	0.0*	20.6*	1	1	1	1	1
45	14	14	44	104.4	136.0*	261.4*	*0.0*	0.0*	22.7*	4	4	4	4	4
14	64	6	91	6.6	0.7	0.7*	*0.0*	0.0*	20.9*	1	1	1	1	1
26	21	21	39	64.7	3.2	2.5	*0.0*	0.0*	21.5*	2	2	2	2	2
38	16	16	25	39.6	521.0	38.8	*0.0*	0.0*	23.0*	4	7	7	7	7
40	17	17	25	6.5	0.6	0.7	*0.0*	0.0*	20.6*	1	1	1	1	1
47	20	20	27	52.1	2.1	2.3	*0.0*	0.0*	22.1*	3	3	3	3	3
8	24	24	53	59.4	552.4	42.6	*0.0*	6.6*	27.2	5	15	16	20	10
20	38	23	55	146.8	483.5	653.2	*0.0*	22.1*	26.1	6	10	16	26	10
46	19	19	21	284.5	409.4	128.1	*0.0*	0.3*	23.9	7	10	10	12	10
3	48	48	66	92.8	125.6	125.8	*0.0*	0.1	33.2*	4	4	8	8	8
4	39	39	46	6.8	1.0	1.3	*0.0*	0.0	21.0*	1	1	1	1	1
9	76	76	88	19.7	1.2	1.3	*0.0*	0.0	21.9*	2	2	2	2	2
15	66	66	75	72.3	243.5	243.7	*0.0*	0.2	27.0*	3	6	6	6	6
16	71	60	74	85.4	6.1	2.1	*0.0*	0.0	22.6*	3	3	3	3	3
22	49	49	62	27.4	126.9	902.1	*0.0*	0.6	30.1*	3	4	8	8	8
25	45	45	50	13.7	2.4	2.2	*0.0*	0.0	21.7*	2	2	2	2	2
30	58	58	63	182.1	501.3	6.4	*0.0*	0.0	22.8*	3	3	3	3	3
34	148	147	161	6.7	0.8	0.8	*0.0*	0.1	21.3*	1	1	1	1	1
35	82	82	87	19.3	1.0	1.0	*0.0*	0.0	21.0*	2	2	2	2	2
37	53	53	58	6.7	0.8	0.8	*0.0*	0.0	21.0*	1	1	1	1	1
41	118	118	134	39.4	665.3	23.8	*0.0*	0.4	25.0*	3	5	4	5	5
43	63	63	78	136.8	10.6	4.4	*0.0*	12.4	23.7*	2	2	2	3	3
27	37	37	89	642.6	75.5	646.6	*0.0*	675.8	29.9	9	10	13	21	10
42	141	141	171	727.8	718.7	27.0	*0.0*	993.1	34.5	13	17	10	133	10
44	57	57	68	116.4	111.2	697.1	*0.0*	152.1	30.0	3	9	13	13	10
48	57	57	68	445.4	7.6	18.9	*0.0*	424.0	27.0	6	5	9	116	10
49	38	38	39	327.2	70.7	210.4	*0.0*	0.3	27.4	7	11	13	14	10
50	60	60	67	137.9	21.7	31.2	*0.0*	373.8	26.7	9	8	8	44	10

Table 7.3: Results of the 50 `random` test queries for SNOMED-CT.

	Module size			MinAs search time (sec.)						#MinAs				
	$\mathcal{EL}^+$2SAT		CEL	$\mathcal{EL}^+$2SAT					CEL	$\mathcal{EL}^+$2SAT				CEL
	(b)	(e)		(a)	(b)	(e)	*enc.*	×2		(a)	(b)	(e)	×2	
8	144	144	156	19.2	1.3	1.3	*0.0*	0.1	20.9*	2	2	2	2	2
11	120	119	130	25.8	1.6	1.6	*0.0*	0.1	21.4*	3	3	3	3	3
20	157	157	164	6.7	0.7	0.9	*0.0*	0.1	21.0*	1	1	1	1	1
21	137	137	178	6.6	0.8	0.9	*0.0*	0.0	21.3*	1	1	1	1	1
22	200	200	214	6.7	0.7	0.8	*0.1*	0.1	21.3*	1	1	1	1	1
23	191	191	198	6.6	0.7	0.8	*0.1*	0.1	21.4*	1	1	1	1	1
28	144	144	151	6.6	0.7	0.9	*0.0*	0.0	21.2*	1	1	1	1	1
29	172	172	177	19.8	1.6	1.9	*0.1*	0.2	21.9*	2	2	2	2	2
30	124	124	134	6.5	0.6	0.8	*0.0*	0.1	20.6*	1	1	1	1	1
31	128	128	139	6.5	0.6	0.8	*0.0*	0.1	20.7*	1	1	1	1	1
33	104	104	112	13.3	1.5	1.2	*0.0*	0.0	21.7*	2	2	2	2	2
34	146	146	158	6.6	0.7	0.8	*0.0*	0.1	21.1*	1	1	1	1	1
35	182	182	196	6.6	0.7	0.8	*0.1*	0.1	21.2*	1	1	1	1	1
39	81	81	92	6.5	0.6	0.6	*0.0*	0.0	20.7*	1	1	1	1	1
41	68	68	76	6.5	0.6	0.6	*0.0*	0.0	20.6*	1	1	1	1	1
43	137	137	142	38.9	42.6	40.1	*0.1*	0.2	24.1*	3	6	6	6	6
44	139	139	151	6.5	0.6	0.8	*0.0*	0.1	20.7*	1	1	1	1	1
45	135	135	147	6.5	0.6	0.8	*0.0*	0.1	20.7*	1	1	1	1	1
46	121	121	126	58.3	129.8	6.8	*0.0*	0.1	22.6*	3	4	4	4	4
47	87	87	94	6.6	0.7	0.7	*0.0*	0.0	20.8*	1	1	1	1	1
48	147	147	152	19.3	1.1	1.1	*0.1*	0.1	21.0*	2	2	2	2	2
1	137	137	142	26.8	298.3	18.8	*0.0*	634.3	29.2	3	7	6	23	10
2	107	107	112	324.1	375.1	54.1	*0.0*	618.2	29.4	4	10	10	28	10
3	92	92	104	176.0	121.8	14.9	*0.0*	303.3	27.3	9	9	10	73	10
4	120	120	130	564.4	51.0	526.7	*0.0*	59.8	29.2	10	11	15	30	10
5	125	125	135	206.9	55.8	113.9	*0.0*	337.6	30.4	7	14	17	32	10
6	181	181	196	839.1	104.4	664.4	*0.0*	282.0	30.9	21	21	17	31	10
7	135	135	140	335.4	306.0	259.9	*0.0*	416.2	30.5	7	9	8	28	10
9	111	111	124	287.7	143.8	210.1	*0.0*	907.6	28.3	6	10	14	19	10
10	117	117	132	406.9	12.0	13.0	*0.0*	0.3	28.7	12	9	11	18	10
12	144	144	149	158.0	316.5	5.7	*0.0*	958.2	31.0	4	8	5	19	10
13	181	181	195	907.0	71.4	60.9	*0.0*	8.3	30.6	12	15	31	44	10
14	122	122	137	54.8	8.0	6.8	*0.0*	75.6	31.2	4	4	4	15	10
15	138	138	143	119.2	8.1	609.3	*0.0*	679.4	31.7	5	5	13	35	10
16	142	142	147	309.5	603.7	287.4	*0.0*	465.8	31.1	6	5	16	47	10
17	126	126	138	574.7	32.6	680.7	*0.0*	83.2	29.3	15	12	28	52	10
18	113	113	122	194.0	45.8	21.2	*0.0*	3.8	28.4	11	11	11	14	10
19	113	113	122	867.0	704.3	101.2	*0.0*	10.6	28.8	9	13	30	38	10
24	147	147	158	129.7	26.8	152.3	*0.0*	0.2	31.3	5	9	10	10	10
25	120	120	128	470.6	6.0	12.8	*0.0*	0.0	30.2	8	5	9	10	10
26	121	121	131	67.2	33.5	12.9	*0.0*	0.5	30.2	5	10	9	14	10
27	112	112	122	83.0	343.1	873.8	*0.0*	496.2	29.6	9	19	60	74	10
32	129	129	150	224.4	133.7	295.8	*0.0*	0.7	31.8	6	6	6	10	10
36	162	162	173	158.0	264.6	7.8	*0.0*	149.5	33.2	5	13	5	20	10
37	114	114	121	189.1	14.0	775.3	*0.0*	271.2	29.8	7	6	12	36	10
38	106	106	112	182.5	48.2	487.4	*0.0*	162.0	28.8	7	10	14	11	10
40	98	98	105	909.6	862.9	823.7	*0.0*	392.8	28.5	12	13	14	23	10
42	71	71	74	333.1	1.8	2.2	*0.0*	22.2	27.1	12	2	3	19	10
49	158	157	231	704.5	5.2	9.0	*0.0*	103.4	33.2	6	4	6	10	10
50	95	95	100	443.3	265.2	478.0	*0.0*	53.6	28.4	4	8	10	14	10

Table 7.4: Results of the 50 `selected` test queries for SNOMED-CT.

	# terminated							# MinAs							Module size avg.		
	$\mathcal{EL}^+$2SAT						CEL	$\mathcal{EL}^+$2SAT						CEL	$\mathcal{EL}^+$2SAT		CEL
test	(a)	(b)	(c)	(d)	(e)	×2		(a)	(b)	(c)	(d)	(e)	×2		(b)	(e)	
SNOMED'09																	
random	0	23	23	23	24	31	41	150	192	223	207	210	503	194	33	31	46
selected	0	0	0	0	0	0	21	262	313	421	439	439	832	325	130	129	141
FULLGALEN																	
random	0	10	10	10	16	23	38	74	76	76	79	79	83	84	3156	51	8926
selected	0	0	0	0	0	11	48	55	55	55	55	55	55	55	3951	146	14801
NCI																	
random	0	40	40	40	41	46	44	159	172	172	200	200	212	150	9	9	39
selected	0	29	29	29	29	36	33	417	416	416	445	445	451	337	20	20	48
GENEONT.																	
random	0	49	49	49	49	50	46	132	164	164	165	165	168	143	5	5	20
selected	0	17	17	17	18	46	8	579	853	846	843	848	965	480	20	20	33
NOTGALEN																	
random	0	17	17	17	36	37	50	67	68	68	68	68	66	68	105	14	95
selected	0	0	0	0	1	50	50	91	91	91	91	91	91	91	159	25	131
TOTAL	0	185	185	185	214	330	379	1982	2398	2532	2592	2600	3426	1927			

(a) Number of terminated problems and pinpointed MinAs.

		Elapsed betw. MinAs (s)						Total pinpointing (s)							Modulariz. (s)		
		$\mathcal{EL}^+$2SAT						$\mathcal{EL}^+$2SAT						CEL	$\mathcal{EL}^+$2SAT		CEL
test	%	(a)	(b)	(c)	(d)	(e)	×2	(a)	(b)	(c)	(d)	(e)	×2		(b)	(e)	
SNOMED'09																	
random	50th	7.4	1.0	1.3	0.9	0.8	0.0	99.3	86.9	87.4	87.3	87.2	0.0	21.4	.000	.000	3.36
	90th	66.	32.	49.	19.	28.	1.6	262.	569.	655.	295.	329.	22.	27.4	.004	.004	3.41
selected	50th	7.2	1.5	1.7	1.6	1.6	0.0	163.	98.	122.	94.	97.	0.4	28.4	.008	.004	3.39
	90th	113.	18.	49.	21.	21.	2.6	655.	402.	817.	690.	694.	496.	31.2	.012	.008	3.41
FULLGALEN																	
random	50th	3.9	0.5	0.5	0.5	0.5	0.0	39.9	22.2	22.1	22.2	34.7	0.0	195.	.369	.001	2.25
	90th	17.6	1.7	1.8	1.8	0.8	0.0	60.7	24.4	23.8	23.9	38.1	0.0	1922.	.376	.002	2.27
selected	50th	3.7	0.7	0.7	0.6	0.4	0.0	39.1	36.6	36.6	36.5	34.7	0.0	321.	.541	.003	2.26
	90th	3.8	0.8	0.8	0.8	0.6	0.0	39.7	37.2	37.0	37.0	36.3	0.0	803.	.578	.004	2.27
NCI																	
random	50th	0.2	0.0	0.0	0.1	0.0	0.0	2.1	2.0	2.0	2.0	1.9	0.0	1.3	.001	.001	0.39
	90th	9.1	3.4	3.4	6.6	6.1	0.2	70.	186.	185.	357.	321.	1.8	4.2	.001	.001	0.39
selected	50th	0.3	0.0	0.0	0.0	0.0	0.0	13.5	2.2	2.2	2.2	2.2	0.0	2.5	.001	.001	0.39
	90th	31.4	1.0	1.0	0.2	0.2	0.0	634.	313.	317.	33.	29.	32.	6.8	.001	.001	0.40
GENEONT.																	
random	50th	0.2	0.0	0.0	0.0	0.0	0.0	2.1	1.8	1.7	1.8	1.7	0.0	0.8	.000	.000	0.26
	90th	8.5	0.1	0.2	0.2	0.2	0.0	21.2	1.9	1.9	2.2	2.0	0.0	1.4	.001	.001	0.26
selected	50th	0.3	0.0	0.0	0.0	0.0	0.0	292.	144.	129.	92.	86.	1.3	1.8	.000	.000	0.26
	90th	58.	26.	25.	13.	13.	0.3	845.	885.	823.	736.	688.	72.	2.6	.001	.001	0.27
NOTGALEN																	
random	50th	0.1	0.0	0.0	0.0	0.0	0.0	1.0	0.6	0.6	0.6	0.9	0.0	0.3	.002	.000	0.08
	90th	0.3	0.0	0.0	0.0	0.0	0.0	1.4	0.6	0.6	0.6	1.1	0.0	1.3	.002	.001	0.09
selected	50th	0.1	0.0	0.0	0.0	0.0	0.0	1.1	0.9	1.0	1.0	0.6	0.0	0.7	.003	.000	0.08
	90th	0.2	0.0	0.0	0.0	0.0	0.0	1.2	1.1	1.0	1.0	1.0	0.0	1.1	.003	.001	0.09

(b) CPU times in pinpointing and modularization.

Table 7.5: Summary results of $\mathcal{EL}^+$2SAT (all versions) and CEL on all the test problems.

the two Cone-Of-Influence Modularization procedures of $\mathcal{EL}^+$2SAT (the normal and the refined one) and by the Reachability-based Modularization of CEL. Since in CEL it is not possible to directly distinguish between the time taken by the modularization procedure and the time required by the following search , we run separate modularization queries in order to measure these times. Both the pinpointing and the modularization times don't include the loading time of the ontology, as explained above. Notice, at last, that the time statistics for CEL includes also problems terminated after the timeout or stopped because of the detection of 10 MinAs. As above explained we can not have more exact statistics on CEL.

7.7.1 Discussion

Here we discuss the result of our extensive experimental evaluation. First, we analyze in general the performance of $\mathcal{EL}^+$2SAT and we compare the results of its different variants. Then we discuss the contribution of the combined approach $\mathcal{EL}^+$2SAT $\times$ 2. Second, we look in more details at the characteristic of the different test sets selected for our evaluation. Finally, we compare both $\mathcal{EL}^+$2SAT and $\mathcal{EL}^+$2SAT $\times$ 2 against CEL.

Analysis of $\mathcal{EL}^+$2SAT
This experimental evaluation allows us to notice a few facts concerning $\mathcal{EL}^+$2SAT alone:

- Every newly introduced enhancements improve the performances of $\mathcal{EL}^+$2SAT, increasing the number of terminating test cases and, more important, the total number of MinAs found within the timeout (see the last line of Table 7.5(a)).
- Each different enhancement to $\mathcal{EL}^+$2SAT impacts on the order of the search performed among all the possible variable assignments. Thus, sometimes "less efficient" versions of $\mathcal{EL}^+$2SAT can incur in a more convenient enumeration order, pinpointing a higher number of MinAs within the timeout (see, e.g., the 42nd `random` or the 36th `selected` problems for SNOMED-CT). However the last version (e) finds the highest total number of MinAs overall the 500 queries of our benchmark problems (see also the results exposed in Appendix 7.10).
- With some exceptions, the more optimized is the version of $\mathcal{EL}^+$2SAT the faster it pinpoints the MinAs. Notice that it is not possible to directly compare on average the times we measured, due to the phenomenon discussed in the previous point and since different variants of $\mathcal{EL}^+$2SAT may report two different numbers of MinAs. However, when pinpointing a large number of MinAs (thus, when the cost of some optimizations, like for instance theory propagation, is amortized) newer versions of $\mathcal{EL}^+$2SAT perform significantly better than the less optimized ones (the statistics reported in Table 7.5(b) for the harder `selected` test sets give a pretty good idea of this trend).
- The Cone-of-influence Modularization technique is negligible in time but dramatically impacts on $\mathcal{EL}^+$2SAT in improving its performances. This fact can be easily

	Module size			MinAs search time (sec.)						#MinAs				
	$\mathcal{EL}^+$2SAT		CEL	$\mathcal{EL}^+$2SAT					CEL	$\mathcal{EL}^+$2SAT				CEL
	(b)	(e)		(a)	(b)	(e)	*enc.*	×2		(a)	(b)	(e)	×2	
8	5	5	143	3.7	0.4*	0.4*	*0.0*	0.0*	1.9*	1	1	1	1	1
15	4	4	135	3.7	0.3*	0.4*	*0.0*	0.0*	1.8*	1	1	1	1	1
27	3	3	13	3.7	0.3*	0.3*	*0.0*	0.0*	1.7*	1	1	1	1	1
29	3	3	12	3.7	0.3*	0.4*	*0.0*	0.0*	1.6*	1	1	1	1	1
31	6	6	11	3.8	0.4*	0.5*	*0.0*	0.0*	1.7*	1	1	1	1	1
32	12	12	26	71.4	2.0*	1.4*	*0.0*	0.0*	2.3*	4	4	4	4	4
35	6	6	138	3.7	0.4*	0.3*	*0.0*	0.0*	2.0*	1	1	1	1	1
39	2	2	4	3.6	0.3*	0.3*	*0.0*	0.0*	1.6*	1	1	1	1	1
42	3	3	14790	3.6	0.3*	0.4*	*0.0*	0.0*	152.3*	1	1	1	1	1
45	7	7	11	3.8	0.4*	0.5*	*0.0*	0.0*	1.7*	1	1	1	1	1
5	3943	6	30	3.8	0.4	0.4*	*0.0*	0.0*	1.7*	1	1	1	1	1
6	3945	12	60	3.8	0.4	0.4*	*0.0*	0.0*	1.8*	1	1	1	1	1
17	3944	6	56	3.6	0.3	0.4*	*0.0*	0.0*	1.7*	1	1	1	1	1
34	3944	5	14869	3.7	0.3	0.2*	*0.0*	0.0*	79.5*	1	1	1	1	1
36	3943	3	30	3.6	0.4	0.4*	*0.0*	0.0*	1.6*	1	1	1	1	1
49	3943	5	14789	3.7	0.4	0.4*	*0.0*	0.0*	195.4*	1	1	1	1	1
16	3947	19	163	3.9	0.6	0.7	*0.0*	0.0*	2.3*	1	1	1	1	1
20	3941	109	14891	3.8	0.5	0.4	*0.0*	0.0*	611.1*	1	1	1	1	1
25	3941	24	14789	3.7	0.5	0.3	*0.0*	0.0*	584.5*	1	1	1	1	1
26	3940	22	14789	3.7	0.4	0.3	*0.0*	0.0*	254.5*	1	1	1	1	1
33	3940	45	15086	3.7	0.4	0.4	*0.0*	0.0*	497.9*	1	1	1	1	1
48	3940	22	111	49.0	4.1	1.8	*0.0*	0.0*	3.5*	2	2	2	2	2
2	3950	17	14789	14.6	1.6	0.8	*0.0*	0.0*	1271.	2	2	2	2	2
7	3942	81	141	11.2	1.5	0.9	*0.0*	0.0	4.1*	2	2	2	2	2
9	3942	81	14789	3.8	0.4	0.4	*0.0*	0.0	407.0*	1	1	1	1	1
10	3941	60	14789	3.8	0.5	0.5	*0.0*	0.0	818.8*	1	1	1	1	1
11	3943	48	119	14.2	1.6	0.9	*0.0*	0.0	2.9*	2	2	2	2	2
13	3941	58	14835	3.6	0.3	0.3	*0.0*	0.0	76.1*	1	1	1	1	1
18	3941	59	14789	3.9	0.6	0.6	*0.0*	0.0	681.1*	1	1	1	1	1
19	3942	70	258	3.8	0.5	0.5	*0.0*	0.0	2.8*	1	1	1	1	1
21	3962	97	14804	4.0	0.6	0.8	*0.0*	0.0	893.4*	1	1	1	1	1
22	3948	39	14789	3.7	0.4	0.3	*0.0*	0.0	477.5*	1	1	1	1	1
23	3943	80	14789	3.8	0.5	0.5	*0.0*	0.0	773.0*	1	1	1	1	1
24	3943	137	14789	3.9	0.5	0.5	*0.0*	0.0	758.3*	1	1	1	1	1
30	3944	119	14790	3.6	0.4	0.4	*0.0*	0.0	9.8*	1	1	1	1	1
37	3942	121	14789	24.7	2.6	1.2	*0.0*	0.0	835.9*	2	2	2	2	2
38	3943	42	85	3.9	0.5	0.5	*0.0*	0.0	2.0*	1	1	1	1	1
46	3945	99	14808	13.9	1.3	0.8	*0.0*	0.0	988.3*	2	2	2	2	2
50	3941	74	14811	10.5	1.2	0.9	*0.0*	0.0	919.0*	2	2	2	2	2
1	3942	57	14790	14.7	2.1	1.5	*0.0*	0.0	3470.	2	2	2	2	2
3	3943	118	14826	4.0	0.7	0.7	*0.0*	0.0	1496.	1	1	1	1	1
4	3946	39	14802	11.1	1.5	0.9	*0.0*	0.0	1865.	2	2	2	2	2
12	3951	191	14952	4.2	0.7	0.6	*0.0*	0.1	1594.	1	1	1	1	1
14	3942	95	14789	21.6	2.3	1.3	*0.0*	0.0	2073.	2	2	2	2	2
28	3947	34	253	113.4	54.8	467.	*0.0*	0.4	20.0	7	9	10	10	10
40	3947	98	14790	43.1	4.3	2.7	*0.0*	997.	8162.	3	3	5	9	10
41	3944	85	14839	14.3	1.8	1.2	*0.0*	0.0	1560.	2	2	2	2	2
43	3942	68	14819	7.7	1.2	1.1	*0.0*	0.0	1922.	2	2	2	2	2
44	3944	103	14789	28.3	2.8	1.6	*0.0*	0.0	2900.	2	2	2	2	2
47	3943	71	14789	4.1	0.7	0.8	*0.0*	0.0	2083.	1	1	1	1	1

Table 7.6: Results of the 50 `random` test queries for FULL-GALEN.

	Module size			MinAs search time (sec.)						#MinAs				
	$\mathcal{EL}^+$2SAT		CEL	$\mathcal{EL}^+$2SAT					CEL	$\mathcal{EL}^+$2SAT				CEL
	(b)	(e)		(a)	(b)	(e)	*enc.*	×2		(a)	(b)	(e)	×2	
7	3956	41	14789	3.6	0.7	0.4	*0.0*	0.0*	321.2*	1	1	1	1	1
8	3954	40	14789	3.7	0.6	0.4	*0.0*	0.0*	207.1*	1	1	1	1	1
9	3954	40	14789	3.7	0.7	0.4	*0.0*	0.0*	210.5*	1	1	1	1	1
14	3944	35	14789	3.6	0.7	0.4	*0.0*	0.0*	175.3*	1	1	1	1	1
15	3948	119	14789	3.6	0.6	0.4	*0.0*	0.0*	288.9*	1	1	1	1	1
17	3951	146	14789	3.6	0.6	0.4	*0.0*	0.0*	208.7*	1	1	1	1	1
33	3948	111	14943	3.8	0.8	0.6	*0.0*	0.0*	937.8*	1	1	1	1	1
34	3946	90	14789	3.8	0.5	0.4	*0.0*	0.0*	929.9*	1	1	1	1	1
44	3955	33	14789	3.6	0.4	0.3	*0.0*	0.0*	393.9*	1	1	1	1	1
48	3950	68	14874	3.7	0.4	0.3	*0.0*	0.0*	158.9*	1	1	1	1	1
49	3950	68	14874	3.7	0.4	0.3	*0.0*	0.0*	158.9*	1	1	1	1	1
1	3958	221	14835	3.1	0.8	0.4	*0.0*	0.1	639.3*	1	1	1	1	1
2	3958	219	14838	3.8	0.8	0.5	*0.0*	0.1	621.9*	1	1	1	1	1
3	3955	217	14836	3.7	0.8	0.5	*0.0*	0.1	545.4*	1	1	1	1	1
4	3957	217	14835	3.8	0.8	0.5	*0.0*	0.1	588.0*	1	1	1	1	1
5	3943	132	14789	11.0	2.0	0.9	*0.0*	0.0	690.6*	2	2	2	2	2
6	3957	160	14789	3.6	0.7	0.4	*0.0*	0.0	322.4*	1	1	1	1	1
10	3963	180	14789	3.7	0.7	0.5	*0.0*	0.1	396.8*	1	1	1	1	1
11	3956	158	14789	3.6	0.7	0.5	*0.0*	0.0	325.4*	1	1	1	1	1
12	3956	158	14789	3.7	0.7	0.5	*0.0*	0.0	427.9*	1	1	1	1	1
13	3950	195	14793	3.7	0.7	0.5	*0.0*	0.1	297.5*	1	1	1	1	1
16	3949	194	14789	3.6	0.7	0.4	*0.0*	0.1	229.8*	1	1	1	1	1
18	3951	196	14789	3.7	0.6	0.4	*0.0*	0.1	204.0*	1	1	1	1	1
19	3951	196	14789	3.6	0.7	0.4	*0.0*	0.1	228.5*	1	1	1	1	1
20	3945	158	14789	3.7	0.8	0.6	*0.0*	0.0	378.9*	1	1	1	1	1
21	3940	114	14789	3.6	0.7	0.4	*0.0*	0.0	240.8*	1	1	1	1	1
22	3957	157	14789	10.9	2.0	1.0	*0.0*	0.0	801.7*	2	2	2	2	2
23	3955	162	14789	3.6	0.7	0.6	*0.0*	0.1	497.9*	1	1	1	1	1
24	3953	163	14789	3.7	0.7	0.6	*0.0*	0.1	500.5*	1	1	1	1	1
25	3942	123	14789	3.6	0.6	0.4	*0.0*	0.0	69.0*	1	1	1	1	1
26	3959	176	14789	3.7	0.6	0.6	*0.0*	0.1	300.2*	1	1	1	1	1
27	3959	176	14789	3.7	0.6	0.5	*0.0*	0.1	303.4*	1	1	1	1	1
28	3959	176	14789	3.7	0.7	0.4	*0.0*	0.1	331.8*	1	1	1	1	1
29	3951	189	14789	3.7	0.6	0.5	*0.0*	0.1	223.2*	1	1	1	1	1
30	3951	181	14789	3.7	0.7	0.5	*0.0*	0.1	223.5*	1	1	1	1	1
31	3951	181	14789	3.6	0.5	0.4	*0.0*	0.1	202.9*	1	1	1	1	1
32	3951	189	14789	3.6	0.7	0.5	*0.0*	0.1	204.2*	1	1	1	1	1
35	3946	153	14789	10.8	1.7	0.9	*0.0*	0.1	705.7*	2	2	2	2	2
36	3954	164	14790	3.7	0.4	0.5	*0.0*	0.1	503.2*	1	1	1	1	1
37	3947	105	14789	3.8	0.6	0.4	*0.0*	0.0	733.4*	1	1	1	1	1
38	3947	148	14789	10.8	1.3	0.7	*0.0*	0.1	803.2*	2	2	2	2	2
39	3962	160	14789	3.7	0.4	0.4	*0.0*	0.0	396.7*	1	1	1	1	1
40	3940	119	14789	3.7	0.4	0.3	*0.0*	0.0	311.0*	1	1	1	1	1
41	3942	134	14789	3.7	0.4	0.3	*0.0*	0.0	215.4*	1	1	1	1	1
42	3941	150	14789	3.6	0.4	0.3	*0.0*	0.0	212.2*	1	1	1	1	1
43	3948	155	14789	3.6	0.4	0.3	*0.0*	0.1	168.1*	1	1	1	1	1
45	3947	149	14789	3.6	0.4	0.3	*0.0*	0.0	252.5*	1	1	1	1	1
50	3962	209	14789	11.0	1.3	0.7	*0.0*	0.1	902.5*	2	2	2	2	2
46	3943	158	14826	3.8	0.5	0.4	*0.0*	0.0	1538.	1	1	1	1	1
47	3945	145	14840	3.8	0.5	0.5	*0.0*	0.0	1427.	1	1	1	1	1

Table 7.7: Results of the 50 `selected` test queries for FULL-GALEN.

observed comparing the (a) version of $\mathcal{EL}^+$2SAT wrt. the successive version (b) which includes the Cone-of-influence Modularization. It is worth noticing that our modularization technique, further than being really simple and fast, is purely SAT-based and it doesn't require any semantic information on the original ontology. To better understand the effect of modularization in reducing the search space, it is possible to compare the modules' sizes reported in this current section with the initial size of the input ontologies reported in the empirical evaluation of Section 7.5. Notice that modularization is essential to allow for terminating the complete enumeration of the possible assignments and leads to a faster detection of the MinAs.

- Theory propagation significantly increases the number of pinpointed MinAs. As can be seen comparing the (b) and (c) variants of $\mathcal{EL}^+$2SAT in Table 7.5(a), theory propagation allows for earlier-discovering almost 140 more MinAs in SNOMED-CT within the timeout. Notice that the increment is more significant in the `selected` group of problems, which is the hardest group of test cases in our experimental evaluation (in terms of the combined effect of modules sizes and number of contained MinAs). The later enhancement to theory propagation introduced in the version (d) further improves this result allowing for a faster enumeration, which leads to find 60 more MinAs overall the five ontologies.

- The Cone-of-Influence Modularization refinement introduced in the version (e) leads to a visible improvement in terms of the number of terminated test cases for FULL-GALEN (and also for the simpler NOT-GALEN), by partially overcoming [29] the drawback of normalization discussed in Section 7.6.4 (see the size of the modules in the right-most block of Table 7.5(a)). This latest version does not significantly increase the total number of found MinAs, because it mostly affects the GALEN-based ontologies, which typically present a very low number of MinAs per query.

- The performances of $\mathcal{EL}^+$2SAT for common queries, represented by the `random` test suite, are clearly satisfactory and largely better than on the harder `selected` queries. In particular, comparing the different 50th and 90th percentiles times we can notice that, even if there are some really challenging queries, on average queries are easily affordable by $\mathcal{EL}^+$2SAT.

Analysis of $\mathcal{EL}^+$2SAT $\times$ 2

Since from the very first sight the benefits of use the $\mathcal{EL}^+$2SAT$\times$2 combined approach are outstanding under every possible perspective. The results of $\mathcal{EL}^+$2SAT $\times$ 2 significantly enhance those of all the previous versions of $\mathcal{EL}^+$2SAT on all the ontologies and all the test cases. In general, we notice the following important facts:

[29] Notice that, especially in the `selected` benchmark, $\mathcal{EL}^+$2SAT $\times$ 2 is able to terminate the enumeration of some FULL-GALEN problems with modules in the order of a hundred of axioms. In fact, when the second instance of $\mathcal{EL}^+$2SAT re-apply cone-of-influence modularization to the freshly encoded sub-ontology, it is able to further reduce the search space that, therefore, still includes a large number of superfluous axioms.

– The combined approach gives better results on all the five benchmark ontologies considerably increasing both the number of terminated problems and the number of pinpointed MinAs. Wrt. $\mathcal{EL}^+$2SAT (e), $\mathcal{EL}^+$2SAT $\times$ 2 terminates the enumeration in 116 more problems and produces in output more than 800 more MinAs overall the 500 test cases.

– $\mathcal{EL}^+$2SAT $\times$ 2 increases, wrt. $\mathcal{EL}^+$2SAT (e) the number of detected MinAs almost in every "not-already-completed" query.

– $\mathcal{EL}^+$2SAT$\times$2 cuts down, close to zero, the CPU time required to find all the existing MinAs in the very majority of the test cases. In particular, whichever the ontology or the query is, $\mathcal{EL}^+$2SAT $\times$ 2 allows for computing "some" MinAs in negligible time.

– The CPU time required in the first two phases of the combined approach, which are Cone-of-influence Modularization and the encoding of the extracted sub-ontology (see Section 7.6.5), absolutely negligible ($\leq 10^{-1}$) for every query and every ontology (SNOMED-CT and FULL-GALEN included).

– What really yields the combined approach able to this dramatic improvement wrt. $\mathcal{EL}^+$2SAT is handling, in the third phase, really small ontologies. The sub-ontologies extracted through modularization are orders of magnitude smaller than the original input ontology. Even if the number of possible assignments that $\mathcal{EL}^+$2SAT $\times$ 2 enumerates is the same enumerated by $\mathcal{EL}^+$2SAT with modularization, [30] every single internal call to the $\mathcal{T}$-solver during the enumeration is performed on a formula that is many orders of magnitude smaller. This, due to overheads, leads globally to an extremely faster enumeration. In contrast, the previous two phases, as above highlighted, does not impact on the total computational time. For example, looking at the single results exposed, we can empirically state that this improvement shifts the bound for termination from modules of about 20 axioms (15 for the huge SNOMED-CT) to modules of almost 26/27 axioms (for all the ontologies).

Analyzing in details what happens for SNOMED-CT'09 and for the other four benchmark ontologies (see Table 7.5 and the detailed data in this section and in Appendix 7.10) we notice that:

– Overall, in SNOMED-CT, the combined approach allows to report hundreds more MinAs both in the `selected` test suite and in the case of the `random` queries. With $\mathcal{EL}^+$2SAT $\times$ 2, no SNOMED-CT query report a lower number of MinAs wrt. CEL. For what concerns termination, the combined approach still does not terminate the enumeration in any of the `selected` problems, since the size of the modules is still too large. However $\mathcal{EL}^+$2SAT $\times$ 2 terminates in 7 more `random` cases. Notice that, in general, $\mathcal{EL}^+$2SAT$\times$2 is able to report up to ten or more MinAs for SNOMED-CT

[30] In both cases all the possible assignments on the set of the selector variables in the cone-of-influence module are enumerated.

in less than one second (see the top-most problems of Tables 7.3 and 7.4 or, e.g., the 24-26th `selected` queries).

– Performance are significantly better in the cases of FULL-GALEN and NCI. For the first ontology $\mathcal{EL}^+$2SAT $\times$ 2 is the first $\mathcal{EL}^+$2SAT-based variant able to terminate on some `selected` problems, and it also terminates in more `random` ones (11 and 7 problems respectively). For the NCI ontology, instead, $\mathcal{EL}^+$2SAT $\times$ 2 terminates the enumeration in 12 new problems.

– The contribution of $\mathcal{EL}^+$2SAT $\times$ 2 is particular evident for GENEONTOLOGY and NOT-GALEN. Thanks to $\mathcal{EL}^+$2SAT $\times$ 2 all the GENEONTOLOGY all but four the benchmark problems are completely enumerated. In NOT-GALEN, instead, $\mathcal{EL}^+$2SAT $\times$ 2 terminates in all the `selected` problems (and, overall, in 50 more problems wrt. $\mathcal{EL}^+$2SAT (e)). All the `selected` problems have, in fact, a module size (applying the refinement of Section 7.6.4) within the empirical bound of 26/27 axioms.

Analysis of the benchmark problems
From the results exposed in this section and in Appendix 7.10, we make some observations concerning our benchmark problems:

- SNOMED-CT'09: from Table 7.5 we can notice that the pinpointing problems for SNOMED-CT are extremely challenging. On average they present large modules and, very often, they include a high number of MinAs. In particular, no `selected` query can be completely enumerated by any $\mathcal{EL}^+$2SAT variant, since rarely the size of cone-of-influence modules fall below a hundred of axioms. Many SNOMED-CT problems include a huge number of MinAs. For instance in the 42nd and 48th `random` problem $\mathcal{EL}^+$2SAT $\times$ 2 is able to pinpoint much more than 100 MinAs each. Nevertheless, when the number of MinAs is relatively small (less than 10 MinAs, as guaranteed by the result of CEL) both $\mathcal{EL}^+$2SAT $\times$ 2 and also $\mathcal{EL}^+$2SAT are able to detect all the existing MinAs in a very short time (that is some seconds in the case of $\mathcal{EL}^+$2SAT and negligible time for $\mathcal{EL}^+$2SAT $\times$ 2). We noticed, at last, $\mathcal{EL}^+$2SAT (b)-(e) succeeds in complete the enumeration of all the possible assignments when the modules are within the bound of approximately 15 axioms (see, e.g., Table 7.3), while this bound is around 20 axioms for the other four smaller ontologies.

- FULL-GALEN: analyzing the single results we notice that the `random` and the `selected` problems of FULL-GALEN has similar properties: they are mostly hard problems characterized by relatively wide (cones-of-influence) modules which, in contrast, contain just a few (usually 1 or 2) MinAs. While the problems in the `selected` category are averagely harder but are very similar each other, `random` problems present some extreme (either really easy or really hard) cases. Extremely easy problems (i.e. with modules smaller than 20 axioms) are just a few, limited to the `random` category. Only a couple of `random` problems, instead, present more than 2 MinAs; in particular the 40th one seems to be the hardest problem of the whole

evaluation: it requires up to 8000 sec. to be only partially completed (with the limit of 10 MinAs) by CEL, much more than any SNOMED-CT problem. In general, $\mathcal{EL}^+$2SAT is able to find all the (few) existing MinAs extremely quickly, without late responses (while it is not always true in CEL, which does not terminate within 1000 sec. for many FULL-GALEN instances), even if $\mathcal{EL}^+$2SAT and $\mathcal{EL}^+$2SAT $\times$ 2 rarely succeed in complete the enumeration of all the assignments given the typically large modules extracted for this ontology.

- NCI: the NCI ontology presents a spread variety of problems. On average NCI's queries are very easy and they are characterized by cone-of-influence modules smaller than 20 axioms (that is the empirical bound within the NCI benchmark problems can be completely enumerated by single $\mathcal{EL}^+$2SAT) but there are also harder problems which do not terminate within the timeout and which, anyhow, rarely exceed the 40 axioms in the cone-of-influence module size. The hardest problems can present a very high number of MinAs. `selected` problems are likely harder than the `random` ones having, on average, larger modules and more MinAs to be found. Looking at the `random` problems we can empirically state that the major part of the existing subsumption relations in NCI (one subsumption query on two) has only one MinA causing it, and when MinAs are just a few they all can be found really quickly.

- GENEONTOLOGY: `random` and `selected` queries in GENEONTOLOGY present very different properties. `random` problems are likely trivial (most frequently they reduce to cone-of-influence modules smaller than 10 axioms) and present a few MinAs. Only one `random` problem (27th) does not terminate within the timeout for the best $\mathcal{EL}^+$2SAT variant, none for $\mathcal{EL}^+$2SAT $\times$ 2. Such a problem looks to be hard as much as the largest part of the `selected` problems. `selected` problems are characterized by larger modules and a high numbers of MinAs. Those problem, in fact, are often beyond the empirical termination bound for $\mathcal{EL}^+$2SAT (that is of about 20 axioms for small ontologies), but they never exceed the module size of 30 axioms, so that they mostly fall into the bound of 26/27 axioms which allows for being completely enumerated by $\mathcal{EL}^+$2SAT $\times$ 2. Since, especially in the `selected` category, the total and single elapsed times in finding MinAs are uncommonly high for GENEONTOLOGY we hypothesize that MinAs are "spread" in the search space bounded by the axioms of the extracted module. In fact, notice from Table 7.5(b) that GENEONTOLOGY presents the highest values in the left and central blocks of the table, comparable with those of SNOMED-CT, but where SNOMED-CT'09 is an ontology orders of magnitude larger in size. In other words, the MinAs found by $\mathcal{EL}^+$2SAT in GENEONTOLOGY are not concentrated in the first part of the enumeration but, instead, are more delayed during the search wrt. what happens for the other benchmark ontologies.

- NOT-GALEN: for NOT-GALEN we can spend the same arguments we spent for the FULL-GALEN benchmark ontology. In fact NOT-GALEN has, in general, the same properties of those in FULL-GALEN with the difference that they are one order of

magnitude smaller. However, the modules extracted are still too large (with the exception of some trivial `random` queries) to allow $\mathcal{EL}^+$2SAT for completing the enumeration, but they are mostly at the reach of $\mathcal{EL}^+$2SAT $\times$ 2.

In general, both looking at the single results and at the summaries exposed in Table 7.5 `selected` queries resulted harder than the `random` ones, as predicted. On average, they contain more MinAs, they require higher CPU times and they report lower numbers of terminated problems. So, we can conclude that we chose a consistent criteria identifying the `selected` groups of queries; `random` queries, instead, well represent what "statistically" can happen by picking-up any query.

Many SNOMED-CT and FULL-GALEN problems are still too large in their final module size to be completely enumerated by $\mathcal{EL}^+$2SAT or $\mathcal{EL}^+$2SAT $\times$ 2, especially in the `selected` test suites. We want to remark how these results show that, once we overcome the ontology-size issue thanks to modularization techniques, the FULL-GALEN problems can be challenging as far as the SNOMED-CT ones, even if they have, on average, different peculiarities wrt. the number of included MinAs.

Comparing $\mathcal{EL}^+$2SAT and $\mathcal{EL}^+$2SAT $\times$ 2 wrt. CEL

Comparing the results of $\mathcal{EL}^+$2SAT $\times$ 2 and of $\mathcal{EL}^+$2SAT version (e) with those of CEL some other considerations are in order:

- $\mathcal{EL}^+$2SAT and $\mathcal{EL}^+$2SAT $\times$ 2 always detect a correct number of MinAs (i.e. greater or equal to the number returned by CEL) when terminating within the timeout.

- When not terminating within the timeout $\mathcal{EL}^+$2SAT (e) pinpoints a lower number of MinAs wrt. CEL in 16 of the 500 queries, among which 14 cases concern queries with more than 10 MinAs and only 2 cases concerns ontologies different than SNOMED-CT. $\mathcal{EL}^+$2SAT $\times$ 2 instead returns a lower number of MinAs wrt. CEL only in the 22nd `random` problem of NOT-GALEN (on which, curiously, all the variants of $\mathcal{EL}^+$2SAT —(a), (b) and (e)— scored better [31]) and in the 40th `random` problem of FULL-GALEN (where, however, CEL takes more than 8000 sec. to reach the 10 MinAs limit, while $\mathcal{EL}^+$2SAT $\times$ 2 stops at 9 MinAs at the 1000 sec. timeout).

- When the Reachability-based Modularization of CEL produces modules which are not small/simple enough, like in FULL-GALEN (Tables 7.6 and 7.7) then the CEL's search algorithm is very time-expensive. Notice, for instance, that CEL runs over the 1000 sec. time limit in 10 `random` and 2 `selected` FULL-GALEN's cases. In FULL-GALEN, in fact, CEL has significantly worst performance also wrt. $\mathcal{EL}^+$2SAT (a) or (b). [32] Disregarding the size issue, indeed, GALEN is the hardest ontology in our test set from a logical-structure perspective. In GALEN many role inclusions and many transitivity axioms occur and axioms are "strongly connected" each other.

[31] Thus it represents a purely accidental case caused by the order followed during the assignment enumeration.

[32] Despite the really small size of the ontology, the same phenomenon is perceptible looking at the NOT-GALEN results.

This fact clearly appears from the very huge module's sizes reported by CEL for the FULL-GALEN ontology (see, e.g., the last column of Table 7.5(a)).

- Concerning modularization we experimentally verified that the modularization routine of $\mathcal{EL}^+$2SAT is more precise than the CEL's one, producing smaller modules, as formally proved in Proposition 24 (Section 7.6.2). Thus, further than being orders of magnitude faster, [33] the Cone-of-influence Modularization of $\mathcal{EL}^+$2SAT dominates wrt. the CEL syntactic modularization. This can be noticed by looking not only at the average module's size but also comparing the sizes for each single query: while for the (b) version of $\mathcal{EL}^+$2SAT we can point out some exception when handling non-normal ontologies (see Section 7.6.4), $\mathcal{EL}^+$2SAT version (e) always extracts smaller modules than CEL.

- While, currently, CEL lacks in the test cases including a high number of MinAs, one point in favor of CEL wrt. $\mathcal{EL}^+$2SAT is that it easily recognizes when no more MinAs are present for a given query, whilst in the $\mathcal{EL}^+$2SAT approach we must always complete the enumeration before termination.

- The execution times reported by $\mathcal{EL}^+$2SAT $\times$ 2 are way better than those of CEL. When both $\mathcal{EL}^+$2SAT $\times$ 2 and CEL find less than 10 MinAs (whichever is the query and the ontology) $\mathcal{EL}^+$2SAT $\times$ 2 absolutely outperforms CEL by almost always detecting all the MinAs in negligible time. For instance, in SNOMED-CT $\mathcal{EL}^+$2SAT$\times$2 can report the last MinA in about 0.1 seconds, against about 20 seconds for CEL. This gap is orders of magnitude greater for FULL-GALEN where $\mathcal{EL}^+$2SAT $\times$ 2 finds in negligible times MinAs for which CEL requires hundreds of seconds. Really often $\mathcal{EL}^+$2SAT $\times$ 2 is able, at the same time, to report a significantly higher number of MinAs wrt. CEL. The comparison between the "$\times$2" and the CEL columns in the central part of Table 7.5(b) gives a clear idea of these results. Thus, except for termination, $\mathcal{EL}^+$2SAT $\times$ 2 outperforms CEL on all the five benchmark ontologies, with the more evident case of FULL-GALEN.

- However, we remark that a complete and precise comparison between $\mathcal{EL}^+$2SAT $\times$ 2 (or, respectively, $\mathcal{EL}^+$2SAT) and CEL is still not possible since CEL currently stops after having reported at most 10 MinAs (e.g. if, as stated by Baader et al., 2007; Suntisrivaraporn, 2009, the CEL underlying algorithm has a worst-case exponential behavior, how does it behave in the SNOMED-CT'09 test cases with more than 100 MinAs?).

One important remark is in order. Even if in some FULL-GALEN and NOT-GALEN problems $\mathcal{EL}^+$2SAT (b) produces larger cone-of-influence modules than the respective

[33] Cone-of-influence Modularization of $\mathcal{EL}^+$2SAT is much more fast than the Reachability-based Modularization of CEL also in the cases in which the module extracted are comparable in size. In particular, while Cone-of-influence Modularization has linear cost wrt. the size of the cone of influence for the given query, Reachability-based Modularization seems to depend from the size of the input ontology (see the right-most column of Table 7.5(b)). This fact is further confirmed that the modularization times reported for CEL do not present differences between `random` (smaller modules) and `selected` (larger modules) problems, unlike for $\mathcal{EL}^+$2SAT.

	FULL-GALEN				NOT-GALEN			
	random		selected		random		selected	
	$\mathcal{E}$2S(b)	CEL	$\mathcal{E}$2S(b)	CEL	$\mathcal{E}$2S(b)	CEL	$\mathcal{E}$2S(b)	CEL
1	143	32285	756	32419	39	301	122	210
2	594	32284	739	32433	3	6	121	207
3	347	32403	725	32427	3	58	124	255
4	86	32317	725	32419	115	208	124	255
5	11	42	503	32284	12	28	124	252
6	24	84	502	32284	2	80	121	227
7	241	307	491	32284	6	19	124	255
8	5	218	493	32284	102	255	124	255
9	219	32284	493	32284	16	45	124	255
10	194	32284	583	32284	23	46	124	255
11	109	198	534	32284	9	9	124	255
12	609	32905	534	32284	24	367	124	255
13	156	32420	581	32288	107	228	124	255
14	278	32284	427	32284	6	7	121	219
15	4	338	453	32284	110	472	121	216
16	39	263	579	32284	5	9	121	227
17	23	84	484	32284	115	208	124	257
18	193	32284	583	32284	8	9	121	219
19	198	656	583	32284	8	22	121	216
20	309	33010	518	32284	28	469	124	252
21	254	32315	382	32284	3	10	121	221
22	95	32284	609	32284	118	217	121	224
23	254	32284	510	32284	109	229	121	207
24	406	32284	525	32284	4	9	121	225
25	46	32284	458	32284	104	610	123	210
26	267	32284	611	32284	69	121	121	225
27	3	13	611	32284	115	208	121	228
28	86	647	611	32284	116	214	121	219
29	3	12	563	32284	5	544	121	222
30	383	32285	553	32284	37	301	128	193
31	6	11	553	32284	20	310	121	224
32	12	32	563	32284	3	7	121	227
33	135	33430	635	32863	22	322	173	544
34	576	32648	555	32284	7	139	173	353
35	6	341	510	32284	5	29	173	358
36	11	42	527	32285	5	22	173	441
37	387	32284	436	32284	5	239	173	403
38	145	192	501	32284	76	279	173	539
39	2	4	506	32284	110	227	119	223
40	405	32285	374	32284	2	6	119	223
41	235	32436	434	32284	115	181	119	219
42	3	32285	483	32284	28	74	115	208
43	187	32395	507	32284	2	6	115	211
44	340	32284	492	32284	115	208	115	205
45	7	11	582	32284	236	933	115	214
46	314	32363	570	32391	3	8	115	208
47	230	32284	540	32439	3	8	115	208
48	51	170	629	32598	115	208	115	208
49	8	32284	629	32598	35	107	139	419
50	198	32353	630	32284	124	255	115	208

Table 7.8: Comparison between Cone-of-Influence ($\mathcal{EL}^+$2SAT) and Reachability-based (CEL) Modularization on the normal-form conservative extensions of the GALEN-based ontologies (for each problem and modularization technique the size of the module extracted is reported).

reachability-based modules extracted by CEL, this fact does not contradict what we proved in Proposition 24 of Section 7.6.2. We already discussed this issue in Section 7.6.4 arguing that it is a drawback that arises from the use of non-normal-form input ontologies. Here we experimentally prove this claim. In Table 7.8 we report the sizes of the modules extracted by $\mathcal{EL}^+$2SAT (b) (shortly $\mathcal{E}$2S(b)) and CEL, respectively, for every FULL-GALEN and NOT-GALEN query performed on the normal-form conservative extensions of these two ontologies. Thus, we have shown also for FULL-GALEN and NOT-GALEN that, as stated by Proposition 24, Cone-of-influence Modularization produces smaller and more precise modules wrt. the Reachability-based one.

7.8 Innovative Results and Further Potentials

In this last part of our work we have proposed a novel SAT-based approach to solve the problem of axiom pinpointing in the lightweight Description Logic $\mathcal{EL}^+$ (end its sublogics), allowing for handling and debugging huge real-world problems from the very prominent field of bio-medical ontologies. We have encoded the classification of the input ontology into a Horn-SAT formula, so that to exploit Boolean Constraint Propagation and Conflict Analysis from the modern state-of-the-art SAT solvers in order to (extremely) quickly find a MinA/justification for every given (undesired) subsumption relation. Then we have extended our approach into the all-SMT framework, so that all the MinAs/justifications for the given queries can be found.

We implemented our novel approach into a tool called $\mathcal{EL}^+$2SAT, which integrates the MINISAT2 SAT solver. After a preliminary evaluation, we have significantly improved our approach by mean of many further optimizations: (i) a SMT-like theory propagation technique, (ii) a simple but very effective fully SAT-based modularization procedure (built on top of the two-watched-literals scheme of modern SAT solvers), and (iii) a combined use of $\mathcal{EL}^+$2SAT, called $\mathcal{EL}^+$2SAT $\times$ 2, which strongly reduces the overheads during the MinAs search, significantly fastening the overall search. We have evaluated the benefits of all the introduced enhancements and of the combined approach in a very extensive empirical test session run on five real-world benchmark ontologies, among which the challenging FULL-GALEN and the huge SNOMED-CT. We have also compared with the other $\mathcal{EL}^+$-specific tool CEL (Baader et al., 2006a).

The innovative contributions of our work are manifold:

- Our tools $\mathcal{EL}^+$2SAT, $\mathcal{EL}^+$2SAT $\times$ 2 reported excellent results being able of exhaustively debug the largest part of the 500 benchmark queries, finding thousands of MinAs.

- With the combined approach $\mathcal{EL}^+$2SAT $\times$ 2 we can obtain tens of MinAs on every subsumption query and every benchmark ontologies in a negligible (often unmeasurable) time.

- We remark that the problem of axiom pinpointing in $\mathcal{EL}^+$ is output exponential, thus the search for all the MinAs could be very expensive if an exponential number of MinAs exist for the given query. However, even in such cases (e.g. on the SNOMED-CT ontology) $\mathcal{EL}^+$2SAT$\times$2 has been able to find hundreds of MinAs within the 1000 sec. timeout we fixed during the empirical evaluation. Moreover, the performance of our approach are particularly outstanding for random queries (representing a common/average debugging situation) or in the queries with a relatively low number of MinAs, which can be all immediately reported by our approach.

- We propose a really efficient approach that is able to find most or all the searched MinAs in negligible time. Our approach significantly outperforms the other stat-of-the-art tool CEL, finding many more MinAs and presenting relevantly shorter response time. Even if a completely fair comparison is not possible yet (because CEL stops after discovering 10 MinAs [34]), when reporting the same number of MinAs our approach presents CPU times which are up to four order of magnitude smaller than those reported by CEL on the tough ontologies SNOMED-CT and FULL-GALEN.

- Our optimization techniques have resulted very effective, being able of gradually enhance the performance of our approach, either leading to the discovery of more MinAs or to fasten the MinAs search/enumeration.

- We have propose a novel modularization technique, called *Cone of Influence Modularization* which have resulted more precise and fast than the previous proposed techniques for $\mathcal{EL}^+$. We have proved that our modularization dominates the Reachability-based modularization of Baader and Suntisrivaraporn (2008), Suntisrivaraporn (2009) (i.e. it produces smaller or equal-size modules). Empirically, the Cone of Influence Modularization implemented in $\mathcal{EL}^+$2SAT, further than being negligible in time on every query and input ontology, extracts modules which are often drastically smaller than those produced by CEL.

- Importantly, we remark that our modularization technique is fully SAT-based and can be directly computed on the encoded SAT problem, decoupled from any extra ontological information. Nevertheless, it is a *semantic* modularization technique instead of a syntactic one like the one performed by (Baader & Suntisrivaraporn, 2008; Suntisrivaraporn, 2009) and included in CEL.

On a wider extent, with our approach we have shown that:

- it could be convenient to move the main part of the reasoning in a first and done-only-once encoding phases, in order to obtain extremely fast responses to the single (and possibly numerous) queries;

- once encoded, it is possible to reason on very hard and huge $\mathcal{EL}^+$ input ontologies like they were simple ones, when relying on effective and efficient modularization techniques;

[34] See Section 7.7 for more details.

- it is possible to perform efficient semantic modularization in $\mathcal{EL}^+$;
- we can profitably apply the state-of-the-art SAT/SMT-based tools and approaches to efficiently handle and perform reasoning task on DL-based ontologies, especially when the size of the input problem limits the other reasoning techniques.

Future Works

Research-wise, we aim at pushing our modularization approach even further by exploiting the structure of our Horn-SAT encoding so that to extract sub-modules from sub-cones of influence [35] in order to increasing the enumerations terminating in our approach (and thus the number of MinAs detected). [36]

Moreover, we plan to investigate alternative sets of completion rules, which may be more suitable for producing smaller $\phi^{all}_{\mathcal{T}(po)}$ formulas, and to extend our techniques to (Horn fragments of) richer logics (e.g., Kazakov, 2009; Magka et al., 2010), or to many other reasoning services (see, e.g., Bienvenu, 2008; Horridge et al., 2008). Finally, we are curios to investigate possible applications of the available *unsat core* techniques from the SAT community to the field of the automated reasoning in Description Logics and ontologies.

[35] Starting from the different clauses inferring the same queried subsumption relation (or some subsumptions in the higher part of the cone of influence) it is possible to compute different sub-modules instead of the whole complete one (which results from the union of all the computed sub-modules). Notice that, the internal structure of $\mathcal{EL}^+$2SAT can easily allow for keeping track of previously found MinAs, guiding the search from sub-module to sub-module. The combined approach $\mathcal{EL}^+$2SAT $\times 2$, instead, can be instructed to work on the encoding of the whole module, using sub-modules as different lists of assumptions during the search.

[36] Notice that, even if potentially overlapping, to enumerate all the possible assignments on a group of cone-of-influence sub-modules could be exponentially less expensive than to complete the enumeration of all the possible assignments on the whole original cone-of-influence module.

7.9 Appendix: Proofs

In this section we define and prove formally all the results stated along this chapter.

Theorem 13. *Given an $\mathcal{EL}^+$ TBox $\mathcal{T}$ in normal form, for every pair of concept names C, D in $\mathsf{PC}_\mathcal{T}$, $C \sqsubseteq_\mathcal{T} D$ if and only if the Horn propositional formula $\phi_\mathcal{T} \wedge p_{[C]} \wedge \neg p_{[D]}$ is unsatisfiable.*

Proof.
(**Only if.**) By definition the clause $p_{[C]} \rightarrow p_{[D]}$, that is $\mathcal{EL}^+2sat(C \sqsubseteq D)$, is in $\phi_\mathcal{T}$ if and only if $C \sqsubseteq D$ belongs to $\mathcal{A}$ that is, since $\mathcal{A}$ is the classification of $\mathcal{T}$, if and only if $C \sqsubseteq_\mathcal{T} D$. Thus, if $C \sqsubseteq_\mathcal{T} D$ then $\phi_\mathcal{T} \wedge p_{[C]} \wedge \neg p_{[D]}$ is unsatisfiable.

(**If.**) Vice versa, if $\phi_\mathcal{T} \wedge p_{[C]} \wedge \neg p_{[D]}$ is unsatisfiable, then it follows that $\phi_\mathcal{T} \models p_{[C]} \rightarrow p_{[D]}$. Thus, there must be a resolution derivation P of $p_{[C]} \rightarrow p_{[D]}$ from some subset of clauses $\psi_1, ..., \psi_n$ in $\phi_\mathcal{T}$. Since $\phi_\mathcal{T}$ is a definite Horn formula, every resolution step can be written in the form

$$\frac{(\bigwedge_i p_{[X_i]}) \rightarrow p_{[X_k]} \qquad (p_{[X_k]} \wedge \bigwedge_j p_{[X_j]}) \rightarrow p_{[X_n]}}{(\bigwedge_i p_{[X_i]} \wedge \bigwedge_j p_{[X_j]}) \rightarrow p_{[X_n]}} \tag{7.12}$$

s.t. all X's are concepts. Each corresponding derivation step

$$\frac{(\bigsqcap_i X_i) \sqsubseteq X_k \qquad (X_k \sqcap \bigsqcap_j X_j) \sqsubseteq X_n}{(\bigsqcap_i X_i \sqcap \bigsqcap_j X_j) \sqsubseteq X_n} \tag{7.13}$$

is valid in $\mathcal{EL}^+$. [37] Moreover, by definition, each ψ_i is $\mathcal{EL}^+2SAT(a_i)$ for some a_i which is either in $\mathcal{T}$ or has been derived from $\mathcal{T}$. Hence, there is a valid derivation of $C \sqsubseteq D$ from $\mathcal{T}$ in $\mathcal{EL}^+$, so that $C \sqsubseteq_\mathcal{T} D$. □

Proposition 14. *The size of the formula $\phi^{all}_\mathcal{T}$ defined in Definition 10 is worst-case polynomial in the size of the TBox $\mathcal{T}$.*

Proof. Under the assumption that conjunctions and role compositions are binary, every completion rule instantiation (Table 7.1) has at most three antecedents: one axiom and two assertions. Thus, the number of different rule instantiations $r(a_i, a_{i'}, a_j, a_k)$ on the axioms and assertions in $\mathcal{A}$ is upper-bounded by $|\mathcal{A}|^2 \cdot |\mathcal{T}|$. Recalling that $|\mathcal{A}| \leq |\mathsf{PC}_\mathcal{T}|^2 \cdot |\mathsf{PR}_\mathcal{T}|$ and hence $|\mathcal{A}|$ is polynomial in $|\mathcal{T}|$, we have the thesis. □

[37] Notice that this statement is purely based on $\mathcal{EL}^+$ semantic, meaning "if an interpretation satisfies the premises, it satisfies also the conclusion", and does not depend on the set of derivation rules used.

Lemma 25. *A literal ℓ can be derived by unit-propagation from $\phi^{all}_{\mathcal{T}} \wedge \bigwedge_{ax_i \in \mathcal{S}} s_{[ax_i]}$ [resp. $\phi^{all}_{\mathcal{T}(po)} \wedge \bigwedge_{ax_i \in \mathcal{S}} s_{[ax_i]}$] if and only if ℓ is a positive occurence of a selector variable $s_{[a_k]}$ for some subsumption relation a_k which is deducible from $\mathcal{S}$.*

Proof.
(**If.**) If the subsumption relation a_k is deducible from $\mathcal{S}$ then there is a chain of rule applications $r_1, \ldots, r_N$ which allows for deriving a_k starting from a set of premises $\mathcal{S}' \subseteq \mathcal{S}$, and such that $r_N = r(a_i, a_{i'}, a_j, a_k)$ for some rule r. We reason by induction on the number N of rules applied in order to deduce a_k from $\mathcal{S}'$.

Base: If $N = 1$ then $a_i, a_{i'}$ and a_j must all be axioms in $\mathcal{S}'$. By construction, the rule clause (7.10) $c_{r_N} = (s_{[a_i]} \wedge s_{[a_{i'}]} \wedge s_{[a_j]}) \rightarrow s_{[a_k]}$ is included in $\phi^{all}_{\mathcal{T}(po)}$, and thus $s_{[a_k]}$ is derived by unit-propagation from $\phi^{all}_{\mathcal{T}(po)} \wedge \bigwedge_{ax_i \in \mathcal{S}} s_{[ax_i]}$ where all the axiom selector variables relative to the axioms of $\mathcal{S}$ are propagated.

Step: We assume that a_k has been drived from the rule applications $r_1, \ldots, r_{N-1}, r_N$. So, by construction, $\phi^{all}_{\mathcal{T}(po)}$ includes the corresponding set of rule clauses (7.10) $c_{r_1}, \ldots, c_{r_N}$, where $c_{r_N} = (s_{[a_i]} \wedge s_{[a_{i'}]} \wedge s_{[a_j]}) \rightarrow s_{[a_k]}$ and $a_i, a_{i'}$ and a_j are either axioms of $\mathcal{S}'$ or subsumption relations consequence of one of the rules among $r_1, \ldots, r_{N-1}$. In both cases all the selector variables $s_{[a_i]}, s_{[a_{i'}]}$ and $s_{[a_j]}$ are positively unit-propagated by inductive hypothesis. Thus $s_{[a_k]}$ is also positively unit-propagation from c_{r_N}.

Notice also that, since $\phi^{all}_{\mathcal{T}(po)}$ is a subformula of $\phi^{all}_{\mathcal{T}}$, every literal that is unit-propagated in the first is also unit-propagated in the second.

(**Only if.**) We reason by induction on the number of unit-propagation steps performed on $\phi^{all}_{\mathcal{T}(po)} \wedge \bigwedge_{ax_i \in \mathcal{S}} s_{[ax_i]}$:

Base: By defition, all the axiom selector variables $s_{[ax_i]}$ such that $ax_i \in \mathcal{S}$ are initially unit-propagated. Since they are axioms of $\mathcal{S}$ they are trivially deducible from $\mathcal{S}$.

Step: Let $\mathcal{L}$ be the set of all literals derived by N applications of unit-propagations steps and assume that all literals in $\mathcal{L}$ are in the form $s_{[a_j]}$. Since only assertion clauses (7.9) contain concept variables $p_{[X]}$ (for some concept X) and each assertion clause contains at least two of them, then: (i) no concept variables can be unit-propagated, and (ii) no assertion clause causes unit-propagations, but only rule clauses (7.10) (all included in the subformula $\phi^{all}_{\mathcal{T}(po)}$) do. Thus only positive selection variables $s_{[a_k]}$ can be propagated, and only from rule clauses (7.10) in the form $(s_{[a_i]} \wedge s_{[a_{i'}]} \wedge s_{[a_j]}) \rightarrow s_{[a_k]}$ such that $\{s_{[a_i]}, s_{[a_{i'}]}, s_{[a_j]}\} \subseteq \mathcal{L}$ and $r(a_i, a_{i'}, a_j, a_k)$ is a rule application. Since, by inductive hipothesis, $a_i, a_{i'}, a_j$ are derived from $\mathcal{S}$, so that a_k is also derived from $\mathcal{S}$ by means of r.

We remark that the literals which can be unit-propagated from $\phi^{all}_{\mathcal{T}(po)} \wedge \bigwedge_{ax_i \in \mathcal{S}} s_{[ax_i]}$ are all and only those which can be unit-propagated from $\phi^{all}_{\mathcal{T}} \wedge \bigwedge_{ax_i \in \mathcal{S}} s_{[ax_i]}$ because, as above stated at point (ii), assertion clauses (7.9) play no role in unit-propagations. □

Theorem 15. *Given an $\mathcal{EL}^+$ TBox $\mathcal{T}$ in normal form, for every $\mathcal{S} \subseteq \mathcal{T}$ and for every pair of concept names C, D in $\mathsf{PC}_\mathcal{T}$, $C \sqsubseteq_\mathcal{S} D$ if and only if the Horn propositional formula $\phi^{all}_{\mathcal{T}(po)} \wedge \bigwedge_{ax_i \in \mathcal{S}} s_{[ax_i]} \wedge \neg s_{[C \sqsubseteq D]}$ is unsatisfiable.*

Proof. Due to the soundness and completeness of the set of classification rules reported in Section 7.3, $C \sqsubseteq_\mathcal{S} D$ if and only if there exists a sequence of rule applications $r_1, \ldots, r_N$ generating $C \sqsubseteq D$ from $\mathcal{S}$. If and only if this is the case, by definition, $\phi^{all}_{\mathcal{T}(po)}$ contains all the rule clauses of type (7.10) corresponding to all the rule applications $r_1, \ldots, r_N$.

(**Only if.**) Thus, on the one hand, if $C \sqsubseteq_\mathcal{S} D$, then $\bigwedge_{ax_i \in \mathcal{S}} s_{[ax_i]}$ and all the clauses of type (7.10) corresponding to all the rule applications $r_1, \ldots, r_N$, from Lemma 25, force $s_{[C \sqsubseteq D]}$ to be true, which falsifies the unit clause $\neg s_{[C \sqsubseteq D]}$. Thus $\phi^{all}_{\mathcal{T}(po)} \wedge \bigwedge_{ax_i \in \mathcal{S}} s_{[ax_i]} \wedge \neg s_{[C \sqsubseteq D]}$ is unsatisfiable.

(**If.**) On the other hand suppose $\phi^{all}_{\mathcal{T}(po)} \wedge \bigwedge_{ax_i \in \mathcal{S}} s_{[ax_i]} \wedge \neg s_{[C \sqsubseteq D]}$ is unsatisfiable. Let $\phi^*_\mathcal{T}$ be the result of assigning in $\phi^{all}_{\mathcal{T}(po)}$ all $s_{[ax_i]}$ with ax_i in $\mathcal{S}$ to $\top$ and unit-propagating the values. (Notice that $\phi^*_\mathcal{T}$ is satisfiable since $\phi^{all}_{\mathcal{T}(po)} \wedge \bigwedge_{ax_i \in \mathcal{S}}$ is satisfiable.) By Lemma 25 and by the definition of $\phi^{all}_{\mathcal{T}(po)}$ and $\phi^*_\mathcal{T}$, all and only the variables $s_{[a_j]}$ s.t. a_j can be derived from $\mathcal{S}$ are unit-propagated in this process. Thus, by contradiction, if $C \sqsubseteq D$ could not be derived from $\mathcal{S}$, then $s_{[C \sqsubseteq D]}$ would not be unit-propagated in this process, so that $\phi^*_\mathcal{T} \wedge \neg s_{[C \sqsubseteq D]}$ would be satisfiable; hence $\phi^{all}_{\mathcal{T}(po)} \wedge \bigwedge_{ax_i \in \mathcal{S}} s_{[ax_i]} \wedge \neg s_{[C \sqsubseteq D]}$ would be satisfiable, violating the hypothesis. □

Theorem 16. *Given an $\mathcal{EL}^+$ TBox $\mathcal{T}$ in normal form, for every $\mathcal{S} \subseteq \mathcal{T}$ and for every pair of concept names C, D in $\mathsf{PC}_\mathcal{T}$, $C \sqsubseteq_\mathcal{S} D$ if and only if the Horn propositional formula $\phi^{all}_\mathcal{T} \wedge \bigwedge_{ax_i \in \mathcal{S}} s_{[ax_i]} \wedge p_{[C]} \wedge \neg p_{[D]}$ is unsatisfiable.*

Proof.
(**Only if.**) Suppose $C \sqsubseteq_\mathcal{S} D$. Then $C \sqsubseteq_\mathcal{T} D$, thus, by Definition 10 and due to the soundness and completeness of the classification rules, $\phi_{\mathcal{T}(so)}$ contains the assertion clause

$$s_{[C \sqsubseteq D]} \rightarrow (p_{[C]} \rightarrow p_{[D]}). \tag{7.14}$$

Let $\phi^*_{\mathcal{T}(so)}$ be $\phi_{\mathcal{T}(so)}$ without clause (7.14). Since $\phi^{all}_\mathcal{T} \stackrel{\text{def}}{=} \phi_{\mathcal{T}(so)} \wedge \phi^{all}_{\mathcal{T}(po)}$, we have that $\phi^{all}_\mathcal{T} \wedge \bigwedge_{ax_i \in \mathcal{S}} s_{[ax_i]} \wedge p_{[C]} \wedge \neg p_{[D]}$ can be rewritten as $\phi^*_{\mathcal{T}(so)} \wedge \neg s_{[C \sqsubseteq D]} \wedge \phi^{all}_{\mathcal{T}(po)} \wedge \bigwedge_{ax_i \in \mathcal{S}} s_{[ax_i]} \wedge p_{[C]} \wedge \neg p_{[D]}$ that is unsatisfiable due to Theorem 15.

(**If.**) Suppose $\phi^{all}_\mathcal{T} \wedge \bigwedge_{ax_i \in \mathcal{S}} s_{[ax_i]} \wedge p_{[C]} \wedge \neg p_{[D]}$ is unsatisfiable. Since $\phi^{all}_\mathcal{T}$ is a definite Horn formula, $\phi^{all}_\mathcal{T} \wedge \bigwedge_{ax_i \in \mathcal{S}}$ is satisfiable. Let $\phi^*_\mathcal{T}$ be the result of assigning in $\phi^{all}_\mathcal{T}$ all $s_{[ax_i]}$ with ax_i in $\mathcal{S}$ to true and of unit-propagating the values. Hence $\phi^*_\mathcal{T} \wedge p_{[C]} \wedge \neg p_{[D]}$ is unsatisfiable and $\phi^*_\mathcal{T}$ is satisfiable.

By Lemma 25 and by the definition of $\phi^{all}_{\mathcal{T}}$ and $\phi^{*}_{\mathcal{T}}$, all and only the variables $s_{[a_j]}$ such that a_j can be derived from $\mathcal{S}$ are unit-propagated in the process of building $\phi^{*}_{\mathcal{T}}$. Thus $\phi^{*}_{\mathcal{T}}$ consists only on clauses in the forms:

(i) $\mathcal{EL}^+2SAT(a_j)$ such that a_j is derived from $\mathcal{S}$ (assertion clauses (7.9) from $\phi_{\mathcal{T}(so)}$),

(ii) $(s_{[a_j]} \rightarrow \mathcal{EL}^+2SAT(a_j))$ such that a_j is not derived from $\mathcal{S}$ (assertion clauses (7.9) from $\phi_{\mathcal{T}(so)}$), and

(iii) $(\bigwedge_j s_{[a_j]}) \rightarrow s_{[a_k]}$ such that a_j, a_k are not derived from $\mathcal{S}$ (rule clauses (7.10) from $\phi^{all}_{\mathcal{T}(po)}$).

Notice that all the clauses (ii) and (iii) contain at least one literal in the form $\neg s_{[a_j]}$ such that a_j is not derived from $\mathcal{S}$, and that clauses (i) contain only concept variables. Notice also that, by Lemma 25 and by Definition 9, all and only the clauses of type (i) are exactly $\phi_{\mathcal{S}}$, i.e. the propositional encoding of all the subsumption relations deducible by $\mathcal{S}$.

Since $\phi^{*}_{\mathcal{T}} \wedge p_{[C]} \wedge \neg p_{[D]}$ is unsatisfiable, it must be that some clauses of type (i) is falsified because whichever concept variable is propagated in $\phi^{*}_{\mathcal{T}}$ there is no way of resolving away the literals $\neg s_{[a_j]}$ from clauses (ii), (iii). Thus also $\phi_{\mathcal{S}} \wedge p_{[C]} \wedge \neg p_{[D]}$ is unsatisfiable, from which it follows that $C \sqsubseteq_{\mathcal{S}} D$, by Theorem 13.

Notice that from the latter fact we can also conclude that the clause $p_{[C]} \rightarrow p_{[D]}$ belongs to $\phi_{\mathcal{S}}$ and therefore, by Definition 10, that (7.14) belogns to $\phi^{all}_{\mathcal{S}}$ (more precisely $\phi_{\mathcal{S}(so)}$). Thus (7.14) belongs also to $\phi^{all}_{\mathcal{T}}$. □

Corollary 17. *Given an $\mathcal{EL}^+$ TBox $\mathcal{T}$ in normal form, for every pair of concept names C, D in $\mathsf{PC}_{\mathcal{T}}$, $C \sqsubseteq_{\mathcal{T}} D$ if and only if the Horn propositional formula $\phi^{all}_{\mathcal{T}} \wedge \bigwedge_{a_i \in \mathcal{T}} s_{[ax_i]} \wedge p_{[C]} \wedge \neg p_{[D]}$ [resp. $\phi^{all}_{\mathcal{T}(po)} \wedge \bigwedge_{ax_i \in \mathcal{T}} s_{[ax_i]} \wedge \neg s_{[C \sqsubseteq D]}$] is unsatisfiable.* □

Corollary 18. *Given an $\mathcal{EL}^+$ TBox $\mathcal{T}$ in normal form and the Horn propositional formula $\phi^{all}_{\mathcal{T}}$, for every pair of concept names C, D in $\mathsf{PC}_{\mathcal{T}}$, and every axiom set $\mathcal{T}^*$ (nMinA) for $C \sqsubseteq_{\mathcal{T}} D$, the algorithm* `lin-extract-MinA`$_{DPLL}(C, D, \mathcal{T}^*, \phi^{all}_{\mathcal{T}})$ *of Figure 7.2 computes a* **minimal** *axiom set $\mathcal{S} \subseteq \mathcal{T}^*$ such that $C \sqsubseteq_{\mathcal{S}} D$ ($\mathcal{S}$ is a MinA for $C \sqsubseteq D$ wrt. $\mathcal{T}$).*

Proof. Let us call $\mathcal{S}$ the axiom set returned by `lin-extract-MinA`$_{DPLL}$ on termination and call $\mathcal{S}_j \supseteq \mathcal{S}$ every intermediate axiom set (nMinA) obtained at line `5.` removing axiom ax_j from the previous intermediate axiom set. In particular $\mathcal{S}_0$ is $\mathcal{T}^*$ and $\mathcal{S}$ coincides with the last of the $\mathcal{S}_j$s.

Notice that Theorem 16 can be read as follow: a subset $\mathcal{S}' \subseteq \mathcal{T}$ is an axiom set (nMinA) for the subsumption relation $C \sqsubseteq_{\mathcal{T}} D$ if and only if `DPLLUnderAssumptions(`$\phi^{all}_{\mathcal{T}}$`,` $\mathcal{S}' \cup \{p_{[C]}, \neg p_{[D]}\}$`)` returns `unsat`. Our claim follows straightforwardly by this remark.

First, all the sets $\mathcal{S}_j$ are nMinAs for $C \sqsubseteq_{\mathcal{T}} D$, since $\mathcal{S}_0 = \mathcal{T}^*$ is an nMinA by hypothesis and every other $\mathcal{S}_j$ is an nMinAs since $\mathcal{S}_j$ is assigned in the case that `DPLLUnderAssumptions` returns `unsat`. Thus $\mathcal{S}$ is, indeed, an nMinA. Second, suppose by contradiction that the set $\mathcal{S}$ returned by the algorithm is not minimal. If this

is the case then there exists an axiom $ax_k \in \mathcal{S}$ s.t. $C \sqsubseteq_{\mathcal{S}\setminus\{ax_k\}} D$ and, therefore, s.t. `DPLLUnderAssumptions(`$\phi^{all}_{\mathcal{T}}$`,` $(\mathcal{S} \setminus \{ax_k\}) \cup \{p_{[C]}, \neg p_{[D]}\}$`)` returns `unsat`, as previously stated. Then since at step `2.` the algorithm tries all the axioms in $\mathcal{T}^*$ (ax_k included), it must exist an intermediate axiom set $\mathcal{S}_i \supseteq \mathcal{S}$ such that at step `3.` the assumptions $\mathcal{L} = \{p_{[C]}, \neg p_{[D]}\} \cup (S_i \setminus \{ax_k\})$ are used. But, at step `4.`, `DPLLUnderAssumptions(`$\phi^{all}_{\mathcal{T}}$`,` $(\mathcal{S} \setminus \{ax_k\}) \cup \{p_{[C]}, \neg p_{[D]}\}$`)` returns `unsat` if and only if `DPLLUnderAssumptions(`$\phi^{all}_{\mathcal{T}}$`,` $(\mathcal{S}_i \setminus \{ax_k\}) \cup \{p_{[C]}, \neg p_{[D]}\}$`)` returns `unsat`, because $\mathcal{S}_i$ is a superset of the assumptions of $\mathcal{S}$. It follows that, before termination, at step `5.` the axiom set $\mathcal{S}_k = \mathcal{S}_i \setminus \{ax_k\}$ must be chosen by the algorithm and, thus, it must be $\mathcal{S}_k \supseteq \mathcal{S}$. So $ax_k \notin \mathcal{S}$, giving a contradiction and proving that $\mathcal{S}$ is minimal. □

Corollary 19. *Given an $\mathcal{EL}^+$ TBox $\mathcal{T}$ in normal form, the Horn propositional formula $\phi^{all}_{\mathcal{T}(po)}$, and a set of axiom selector variables $\mathcal{P}_{\mathcal{S}} \subseteq \mathcal{P}_{\mathcal{T}}$ from $\phi^{all}_{\mathcal{T}(po)}$, for every pair of concept names C, D in $\mathsf{PC}_{\mathcal{T}}$, the algorithm* `compute-one-MinA(`$\phi^{all}_{\mathcal{T}(po)}, \mathcal{P}_{\mathcal{S}}, \{\neg s_{[C\sqsubseteq D]}\}$`)` *of Figure 7.3 computes one set of axiom selector variables $\mathcal{P}_{\mathcal{S}^*}$ such that:*

- *either $C \not\sqsubseteq_{\mathcal{S}} D$ and $\mathcal{P}_{\mathcal{S}^*} = \emptyset$,*
- *or $C \sqsubseteq_{\mathcal{S}} D$, $\mathcal{P}_{\mathcal{S}^*} \neq \emptyset$ and $\mathcal{S}^* \subseteq \mathcal{S}$ is a* **minimal** *axiom set such that $C \sqsubseteq_{\mathcal{S}^*} D$ ($\mathcal{S}^*$ is a MinA for $C \sqsubseteq D$ wrt. $\mathcal{S}$).* □

Corollary 21. *Given an $\mathcal{EL}^+$ TBox $\mathcal{T}$ in normal form, for every pair of concept names C, D in $\mathsf{PC}_{\mathcal{T}}$, the following facts hold:*

(i) for every $\mathcal{S} \subseteq \mathcal{T}$, if $\phi^{one}_{\mathcal{T}} \wedge \bigwedge_{ax_i \in \mathcal{S}} s_{[ax_i]} \wedge p_{[C]} \wedge \neg p_{[D]}$ is unsatisfiable then $C \sqsubseteq_{\mathcal{S}} D$;

(ii) if $C \sqsubseteq_{\mathcal{T}} D$ then there exists at least one (possibly proper) subset $\mathcal{S} \subseteq \mathcal{T}$ such that $\phi^{one}_{\mathcal{T}} \wedge \bigwedge_{ax_i \in \mathcal{S}} s_{[ax_i]} \wedge p_{[C]} \wedge \neg p_{[D]}$ is unsatisfiable.

Proof.

(i) If $\phi^{one}_{\mathcal{T}} \wedge \bigwedge_{ax_i \in \mathcal{S}} s_{[ax_i]} \wedge p_{[C]} \wedge \neg p_{[D]}$ is unsatisfiable, then also if $\phi^{all}_{\mathcal{T}} \wedge \bigwedge_{ax_i \in \mathcal{S}} s_{[ax_i]} \wedge p_{[C]} \wedge \neg p_{[D]}$ is unsatisfiable, so that $C \sqsubseteq_{\mathcal{S}} D$ by Theorem 16.

(ii) Let $\mathcal{R}^*$ be one complete and consistent set of completion-rule applications generating the classification $\mathcal{A}$ on which $\phi^{one}_{\mathcal{T}}$ is defined. If $C \sqsubseteq_{\mathcal{T}} D$, and since $\mathcal{A}$ is the classification of $\mathcal{T}$ obtained through a set of sound and complete completion rules, $\mathcal{R}^*$ must contain a sequence of rule applications $r_1, \ldots, r_k$ generating $C \sqsubseteq D$ from a set $\mathcal{S}$. If this is the case, by construction, $\phi^{one}_{\mathcal{T}}$ contains the assertion clause (7.14) and all the rule clauses corresponding to the rule applications $r_1, \ldots, r_k$. Thus, $\bigwedge_{ax_i \in \mathcal{S}} s_{[ax_i]}$ and all the rule clauses corresponding to the rule applications $r_1, \ldots, r_k$ force $s_{[C\sqsubseteq D]}$ to be true and $p_{[C]} \wedge \neg p_{[D]}$ forces $p_{[C]}$ and $\neg p_{[D]}$ to be true, which falsifies the clause (7.14) in $\phi^{one}_{\mathcal{T}}$. Thus $\phi^{one}_{\mathcal{T}} \wedge \bigwedge_{ax_i \in \mathcal{S}} s_{[ax_i]} \wedge p_{[C]} \wedge \neg p_{[D]}$ is unsatisfiable for some $\mathcal{S} \subseteq \mathcal{T}$.

□

Theorem 22. *Given an $\mathcal{EL}^+$ TBox $\mathcal{T}$ in normal form and the formula $\phi^{all}_{\mathcal{T}(po)}$ as defined in Definition 10, for every pair of concept names C, D in $\mathsf{PC}_{\mathcal{T}}$, the following facts hold:*

(i) $C \sqsubseteq_{\mathcal{T}} D$ if and only if $C \sqsubseteq_{\mathcal{M}^{\mathsf{c.o.i.}}_{C\sqsubseteq D}} D$;

(ii) if $\mathcal{S}$ is a minimal subset $\mathcal{S} \subseteq \mathcal{T}$ such that $C \sqsubseteq_{\mathcal{S}} D$ and $C \not\sqsubseteq_{\mathcal{S}'} D$ for every $\mathcal{S}' \subset \mathcal{S}$ (i.e. $\mathcal{S}$ is a MinA for $C \sqsubseteq_{\mathcal{T}} D$), then $\mathcal{S} \subseteq \mathcal{M}^{\mathsf{c.o.i.}}_{C\sqsubseteq D}$,

where $\mathcal{M}^{\mathsf{c.o.i.}}_{C_i\sqsubseteq D_i} \subseteq \mathcal{T}$ is the cone-of-influence module for $C \sqsubseteq D$ wrt. $\phi^{all}_{\mathcal{T}(po)}$, as defined in Definition 12.

Proof.

(i) By Theorem 15 and by Definition 12 of $\mathcal{M}^{\mathsf{c.o.i.}}_{C\sqsubseteq D}$ wrt. $\phi^{all}_{\mathcal{T}(po)}$, in place of fact (i), we can equivalently prove that the Horn propositional formula $\phi^{all}_{\mathcal{T}(po)} \wedge \bigwedge_{ax_i\in\mathcal{T}} s_{[ax_i]} \wedge \neg s_{[C\sqsubseteq D]}$, namely $\psi^{C\sqsubseteq D}_{\mathcal{T}}$, is unsatisfiable if and only if the Horn propositional formula $\phi^{all}_{\mathcal{T}(po)} \wedge \bigwedge_{ax_i\in\mathcal{M}^{\mathsf{c.o.i.}}_{C\sqsubseteq D}} s_{[ax_i]} \wedge \neg s_{[C\sqsubseteq D]}$, namely $\psi^{C\sqsubseteq D}_{\mathcal{M}^{\mathsf{c.o.i.}}_{C\sqsubseteq D}}$, is unsatisfiable. On the one hand, since $\psi^{C\sqsubseteq D}_{\mathcal{T}} = \psi^{C\sqsubseteq D}_{\mathcal{M}^{\mathsf{c.o.i.}}_{C\sqsubseteq D}} \wedge \bigwedge_{ax_i\in(\mathcal{T}\setminus\mathcal{M}^{\mathsf{c.o.i.}}_{C\sqsubseteq D})} s_{[ax_i]}$, if $\psi^{C\sqsubseteq D}_{\mathcal{M}^{\mathsf{c.o.i.}}_{C\sqsubseteq D}}$ is unsatisfiable then $\psi^{C\sqsubseteq D}_{\mathcal{T}}$ is also trivially unsatisfiable. On the other hand, we consider any minimal subset $\mathcal{S}$ of $\mathcal{T}$ such that $\phi^{all}_{\mathcal{T}(po)} \wedge \bigwedge_{ax_i\in\mathcal{S}} s_{[ax_i]} \wedge \neg s_{[C\sqsubseteq D]}$ is unsatisfiable and we refer to the following proof of fact (ii).

(ii) Point (ii) is again consequence of Theorem 15. Let $\mathcal{S}$ be any minimal subset of $\mathcal{T}$ such that $C \sqsubseteq_{\mathcal{S}} D$ and $C \not\sqsubseteq_{\mathcal{S}'} D$ for every $\mathcal{S}' \subset \mathcal{S}$. By Theorem 15 we have that $\phi^{all}_{\mathcal{T}(po)} \wedge \bigwedge_{ax_i\in\mathcal{S}} s_{[ax_i]} \wedge \neg s_{[C\sqsubseteq D]}$ is unsatisfiable. Since $\phi^{all}_{\mathcal{T}(po)} \wedge \bigwedge_{ax_i\in\mathcal{S}} s_{[ax_i]}$ is trivially satisfiable, it follows that $\phi^{all}_{\mathcal{T}(po)} \wedge \bigwedge_{ax_i\in\mathcal{S}} s_{[ax_i]} \models s_{[C\sqsubseteq D]}$. But since, by Theorem 15, $\phi^{all}_{\mathcal{T}(po)} \wedge \bigwedge_{ax_i\in(\mathcal{S}\setminus\{ax_j\})} s_{[ax_i]} \wedge \neg s_{[C\sqsubseteq D]}$ is instead satisfiable for every $ax_j \in \mathcal{S}$, then every $s_{[ax_j]}$ is included in the subformula of $\phi^{all}_{\mathcal{T}(po)}$ responsible for the deduction of $s_{[C\sqsubseteq D]}$ (i.e. $s_{[ax_j]}$ is included in $\phi_{\mathcal{C}^{C\sqsubseteq D}_{\mathcal{T}}}$), so that $s_{[ax_j]} \in \mathcal{C}^{C\sqsubseteq D}_{\mathcal{T}}$, for every $ax_j \in \mathcal{S}$. Accordingly, by Definition 12, $ax_j \in \mathcal{M}^{\mathsf{c.o.i.}}_{C\sqsubseteq D}$ for every $ax_j \in \mathcal{S}$ and we have the thesis $\mathcal{S} \subseteq \mathcal{M}^{\mathsf{c.o.i.}}_{C\sqsubseteq D}$.

□

Proposition 24. *Given an $\mathcal{EL}^+$ TBox $\mathcal{T}$ in normal form and the formula $\phi^{all}_{\mathcal{T}(po)}$ as defined in Definition 10, for every pair of concept names C, D in $\mathsf{PC}_\mathcal{T}$ it holds $\mathcal{M}^{\mathsf{c.o.i.}}_{C \sqsubseteq D} \subseteq \mathcal{M}^{\mathsf{reach}}_{C}$, where $\mathcal{M}^{\mathsf{reach}}_{C}$ and $\mathcal{M}^{\mathsf{c.o.i.}}_{C \sqsubseteq D}$ are, respectively, the reachability-based module (Definition 8) and cone-of-influence module (Definition 12) for $C \sqsubseteq D$.*

Proof. We remark that the cone-of-influence module $\mathcal{M}^{\mathsf{c.o.i.}}_{C \sqsubseteq D}$ for $C \sqsubseteq_\mathcal{T} D$ is defined wrt. the formula $\phi^{all}_{\mathcal{T}(po)}$, thus it is defined wrt. a sound and complete set of completion rules for the classification problem. Let us consider the set of completion rules of Table 7.1 and, for sake of clarity, the generic concept X so that we prove the stronger fact $\mathcal{M}^{\mathsf{c.o.i.}}_{X \sqsubseteq \hat{D}} \subseteq \mathcal{M}^{\mathsf{reach}}_{X}$ for every pair of concept names $X \in \mathsf{PC}_\mathcal{T}$ and $\hat{D} \in \mathsf{NC}_\mathcal{T}$.

If a completion rule of Table 7.1 can be applied starting from a set of assertions (first column) and one axiom (second column), namely ax, deducing a new subsumption relation $X \sqsubseteq \hat{D}$ (third column), then such a completion rule is encoded into $\phi^{all}_{\mathcal{T}(po)}$ and, accordingly, the axiom ax becomes part of the cone-of-influence module $\mathcal{M}^{\mathsf{c.o.i.}}_{X \sqsubseteq \hat{D}}$ for $X \sqsubseteq \hat{D}$. Wlog. we limit our proof to axiom ax but the same considerations apply to all the other premises of the rule application when they are axioms of $\mathcal{T}$. In particular we prove that $ax \in \mathcal{M}^{\mathsf{c.o.i.}}_{X \sqsubseteq \hat{D}}$ implies $ax \in \mathcal{M}^{\mathsf{reach}}_{X}$.

Notice the following two facts:

(i) : given a subsumption relation a, the cone-of-influence module $\mathcal{M}^{\mathsf{c.o.i.}}_{a}$ for a is fully included in any cone-of-influence module $\mathcal{M}^{\mathsf{c.o.i.}}_{a'}$, with a' any subsumption relation such that $s_{[a]}$ is in the cone of influence $\mathcal{C}^{a'}_{\mathcal{T}}$ for a'.

(ii) : the reachability-based module for a given concept X includes all the reachability-based modules for any other concept that is X-reachable.

This has been said consider the five completion rules of Table 7.1:

1. Suppose that the first completion rule is applied on the previously deduced subsumptions $X \sqsubseteq C_1, \ldots, X \sqsubseteq C_k$ (with $k \geq 1$), and on the axiom $ax \stackrel{\text{def}}{=} C_1 \sqcap \cdots \sqcap C_k \sqsubseteq D$. Thus, $ax \in \mathcal{M}^{\mathsf{c.o.i.}}_{X \sqsubseteq \hat{D}}$. Since Reachability-based Modularization preserves subsumption (Property 1), then the symbols $C_1, \ldots, C_k$ must be X-reachable and, therefore, ax is a X-reachable axiom by Definition 8.

2. The same arguments of the previous point can be spent for the second completion rule with the premise $X \sqsubseteq C$, the symbol C and the axiom $ax \stackrel{\text{def}}{=} C \sqsubseteq \exists r.D$.

3. Suppose that the third completion rule is applied on the previously deduced subsumptions $X \sqsubseteq \exists r.E$ and $E \sqsubseteq C$, and on the axiom $ax = \exists r.C \sqsubseteq D$. Thus, $ax \in \mathcal{M}^{\mathsf{c.o.i.}}_{X \sqsubseteq \hat{D}}$. Accordingly, r and E must be X-reachable and everything that is E-reachable is also X-reachable. Henceforth C is in turn X-reachable. It follows that ax is a X-reachable axiom by Definition 8.

4. The same arguments of point 1 can be spent for the forth completion rule with the premise $X \sqsubseteq \exists r.C$, the symbol r and the axiom $ax \stackrel{\text{def}}{=} r \sqsubseteq s$.

5. Suppose that the fifth completion rule is applied on the previously deduced subsumptions $X \sqsubseteq \exists r_1.E_1, \ldots, E_{n-1} \sqsubseteq \exists r_n.D$ (with $n \geq 2$), and on the axiom $ax \stackrel{\text{def}}{=} r_1 \circ \cdots \circ r_n$. Thus, $ax \in \mathcal{M}^{\mathsf{c.o.i.}}_{X \sqsubseteq \hat{D}}$. Then E_1, r_1 must be X-reachable and everything that is E_1/r_1-reachable is also X-reachable, in particular E_2 and r_2 are X-reachable, and so on and so forth. So, it follows that also the symbols $E_1, \ldots, E_{n-1}, D$ and $r_1, \ldots, r_n$ are X-reachable. Thus, by Definition 8, ax is a X-reachable axiom.

Concluding, we proved for every possible completion rule applied in building $\phi^{all}_{\mathcal{T}(po)}$ that if $ax \in \mathcal{M}^{\mathsf{c.o.i.}}_{X \sqsubseteq \hat{D}}$ then ax is X-reachable, i.e. that every ax included in a cone-of-influence module is included in the respective reachability-based module. Thus, by facts (i) and (ii), we proved $\mathcal{M}^{\mathsf{c.o.i.}}_{X \sqsubseteq \hat{D}} \subseteq \mathcal{M}^{\mathsf{reach}}_{X}$. □

7.10 Appendix: Further Experimental Evaluation Data

	Module size			MinAs search time (sec.)						#MinAs				
	$\mathcal{EL}^+$2SAT		CEL	$\mathcal{EL}^+$2SAT					CEL	$\mathcal{EL}^+$2SAT				CEL
	(b)	(e)		(a)	(b)	(e)	*enc.*	×2		(a)	(b)	(e)	×2	
1	2	2	13	0.2	0.0*	0.1*	*0.0*	0.0*	1.2*	1	1	1	1	1
2	3	3	42	0.2	0.0*	0.0*	*0.0*	0.0*	1.3*	1	1	1	1	1
3	3	3	17	0.2	0.0*	0.0*	*0.0*	0.0*	1.2*	1	1	1	1	1
4	4	4	4	0.2	0.0*	0.1*	*0.0*	0.0*	1.2*	1	1	1	1	1
5	14	14	25	0.6	0.3*	0.2*	*0.0*	0.0*	2.0*	3	4	4	4	4
6	4	4	50	0.2	0.0*	0.1*	*0.0*	0.0*	1.3*	1	1	1	1	1
7	5	5	19	0.2	0.0*	0.1*	*0.0*	0.0*	1.3*	1	1	1	1	1
8	5	5	91	0.2	0.0*	0.1*	*0.0*	0.0*	1.4*	1	1	1	1	1
10	2	2	29	0.2	0.0*	0.1*	*0.0*	0.0*	1.2*	1	1	1	1	1
11	9	9	42	0.9	0.1*	0.1*	*0.0*	0.0*	1.6*	3	3	3	3	3
13	5	5	39	0.2	0.0*	0.1*	*0.0*	0.0*	1.3*	1	1	1	1	1
14	4	4	158	0.2	0.0*	0.1*	*0.0*	0.0*	1.4*	1	1	1	1	1
15	2	2	270	0.2	0.0*	0.0*	*0.0*	0.0*	1.3*	1	1	1	1	1
16	2	2	66	0.2	0.0*	0.1*	*0.0*	0.0*	1.3*	1	1	1	1	1
17	4	4	7	0.2	0.0*	0.1*	*0.0*	0.0*	1.2*	1	1	1	1	1
18	8	8	39	0.6	0.1*	0.1*	*0.0*	0.0*	1.5*	2	2	2	2	2
19	3	3	26	0.2	0.0*	0.0*	*0.0*	0.0*	1.3*	1	1	1	1	1
21	10	10	49	1.3	0.1*	0.2*	*0.0*	0.0*	2.0*	4	4	4	4	4
22	2	2	13	0.2	0.0*	0.0*	*0.0*	0.0*	1.2*	1	1	1	1	1
23	2	2	35	0.2	0.0*	0.1*	*0.0*	0.0*	1.2*	1	1	1	1	1
24	2	2	25	0.2	0.0*	0.1*	*0.0*	0.0*	1.2*	1	1	1	1	1
26	3	3	19	0.2	0.0*	0.0*	*0.0*	0.0*	1.2*	1	1	1	1	1
28	5	5	51	0.2	0.0*	0.1*	*0.0*	0.0*	1.3*	1	1	1	1	1
29	3	3	5	0.2	0.0*	0.1*	*0.0*	0.0*	1.2*	1	1	1	1	1
31	2	2	50	0.2	0.0*	0.1*	*0.0*	0.0*	1.2*	1	1	1	1	1
32	4	4	15	0.2	0.0*	0.1*	*0.0*	0.0*	1.2*	1	1	1	1	1
33	12	12	24	0.4	0.1*	0.1*	*0.0*	0.0*	1.6*	2	2	2	2	2
34	11	11	67	74.2	0.2*	0.9*	*0.0*	0.0*	1.9*	3	3	3	3	3
35	2	2	3	0.2	0.0*	0.1*	*0.0*	0.0*	1.2*	1	1	1	1	1
36	6	6	6	0.2	0.0*	0.1*	*0.0*	0.0*	1.2*	1	1	1	1	1
37	3	3	4	0.2	0.0*	0.0*	*0.0*	0.0*	1.2*	1	1	1	1	1
38	6	6	25	0.6	0.0*	0.1*	*0.0*	0.0*	1.4*	2	2	2	2	2
39	8	8	17	0.2	0.0*	0.1*	*0.0*	0.0*	1.4*	1	1	1	1	1
42	6	6	9	0.4	0.0*	0.1*	*0.0*	0.0*	1.3*	2	2	2	2	2
43	2	2	2	0.2	0.0*	0.0*	*0.0*	0.0*	1.2*	1	1	1	1	1
44	4	4	11	0.2	0.0*	0.1*	*0.0*	0.0*	1.3*	1	1	1	1	1
45	9	9	54	1.1	0.1*	0.1*	*0.0*	0.0*	1.7*	3	3	3	3	3
46	7	7	54	0.4	0.0*	0.1*	*0.0*	0.0*	1.5*	2	2	2	2	2
47	5	5	7	0.2	0.0*	0.1*	*0.0*	0.0*	1.2*	1	1	1	1	1
49	6	6	31	0.4	0.0*	0.1*	*0.0*	0.0*	1.4*	2	2	2	2	2
9	19	19	56	12.6	4.8	0.3*	*0.0*	0.0*	4.9*	6	7	7	7	7
20	24	24	34	7.8	932.9	0.7	*0.0*	0.0*	9.6*	5	8	8	8	8
27	20	20	20	66.6	107.9	425.6	*0.0*	1.2*	4.0*	7	8	8	8	8
30	21	21	22	78.7	459.2	839.3	*0.0*	5.1*	3.9	7	12	12	12	10
41	26	26	67	624.8	400.8	318.8	*0.0*	1.8*	4.9	10	11	11	11	10
50	22	22	54	54.0	865.3	104.0	*0.0*	0.5*	9.6	9	10	10	10	10
48	31	31	57	14.0	1.7	344.0	*0.0*	2.0	251.*	9	8	9	9	9
12	32	32	51	229.9	558.2	20.0	*0.0*	78.1	3.7	14	21	18	22	10
25	38	38	50	67.7	0.3	335.1	*0.0*	782.	4.2	18	7	27	31	10
40	33	33	57	335.5	183.7	840.7	*0.0*	95.1	3.9	17	22	32	36	10

Table 7.9: Results of the 50 `random` test queries for NCI.

	Module size			MinAs search time (sec.)						#MinAs				
	$\mathcal{EL}^+$2SAT		CEL	$\mathcal{EL}^+$2SAT					CEL	$\mathcal{EL}^+$2SAT				CEL
	(b)	(e)		(a)	(b)	(e)	*enc.*	×2		(a)	(b)	(e)	×2	
4	15	15	57	3.0	0.5*	1.1*	*0.0*	0.0*	3.1*	6	6	6	6	6
5	14	14	51	6.1	0.2*	0.2*	*0.0*	0.0*	2.5*	6	6	6	6	6
6	13	13	15	2.4	0.3*	0.2*	*0.0*	0.0*	1.9*	4	4	4	4	4
8	16	16	26	15.1	0.2*	0.2*	*0.0*	0.0*	2.4*	5	5	5	5	5
10	12	12	19	15.7	0.9*	0.2*	*0.0*	0.0*	2.0*	6	6	6	6	6
11	12	12	36	4.4	0.1*	0.2*	*0.0*	0.0*	1.9*	5	5	5	5	5
12	12	12	36	2.1	0.2*	0.2*	*0.0*	0.0*	1.9*	5	5	5	5	5
13	12	12	39	9.0	0.1*	0.1*	*0.0*	0.0*	1.9*	4	4	4	4	4
14	14	14	43	17.5	0.1*	0.2*	*0.0*	0.0*	2.4*	5	5	5	5	5
15	15	15	44	15.1	0.2*	0.2*	*0.0*	0.0*	3.1*	7	7	7	7	7
16	14	14	22	5.0	0.1*	0.2*	*0.0*	0.0*	1.9*	4	4	4	4	4
18	12	12	15	1.4	0.1*	0.1*	*0.0*	0.0*	1.7*	4	4	4	4	4
19	12	12	35	1.8	0.1*	0.1*	*0.0*	0.0*	1.8*	4	4	4	4	4
20	10	10	32	1.4	0.1*	0.1*	*0.0*	0.0*	1.7*	4	4	4	4	4
21	11	11	57	1.4	0.1*	0.1*	*0.0*	0.0*	1.8*	4	4	4	4	4
22	12	12	38	1.7	0.1*	0.1*	*0.0*	0.0*	1.9*	4	4	4	4	4
23	11	11	54	1.7	0.3*	0.2*	*0.0*	0.0*	1.8*	5	5	5	5	5
24	11	11	45	1.5	0.1*	0.2*	*0.0*	0.0*	1.9*	5	5	5	5	5
25	14	14	45	2.3	0.2*	0.3*	*0.0*	0.0*	2.3*	6	6	6	6	6
27	13	13	58	1.7	0.1*	0.2*	*0.0*	0.0*	2.2*	5	5	5	5	5
28	15	15	37	2.0	0.2*	0.2*	*0.0*	0.0*	2.4*	5	5	5	5	5
29	13	13	58	1.7	0.1*	0.2*	*0.0*	0.0*	2.2*	5	5	5	5	5
30	9	9	60	16.2	0.1*	0.1*	*0.0*	0.0*	1.7*	4	4	4	4	4
31	8	8	43	13.2	0.1*	0.1*	*0.0*	0.0*	1.5*	4	4	4	4	4
32	16	16	56	36.1	0.2*	1.3*	*0.0*	0.0*	3.2*	6	6	6	6	6
33	11	11	40	1.2	0.1*	0.1*	*0.0*	0.0*	1.8*	4	4	4	4	4
34	13	13	21	1.1	0.1*	0.1*	*0.0*	0.0*	1.9*	4	4	4	4	4
35	10	10	69	1.1	0.1*	0.1*	*0.0*	0.0*	1.7*	4	4	4	4	4
41	17	17	66	15.2	9.8*	0.2*	*0.0*	0.0*	3.5*	6	6	6	6	6
7	22	22	35	19.2	6.3	3.0	*0.0*	0.0*	10.1*	7	7	7	7	7
9	23	23	44	90.2	116.1	9.1	*0.0*	0.1*	15.3*	7	7	7	7	7
17	22	22	24	11.6	0.3	0.2	*0.0*	0.0*	4.4*	6	6	6	6	6
49	26	26	38	3.2	0.3	5.0	*0.0*	0.0*	15.3*	5	7	7	7	7
26	20	20	68	73.1	3.2	2.6	*0.0*	2.0*	3.6	10	11	11	11	10
42	26	26	65	183.7	308.3	0.4	*0.0*	0.0*	6.8	11	11	11	11	10
43	19	19	68	833.8	1.2	0.6	*0.0*	0.0*	4.4	10	10	10	10	10
1	38	38	75	519.0	249.6	27.3	*0.0*	69.6	2.9	29	32	29	27	10
2	28	28	63	851.3	95.7	136.3	*0.0*	68.7	3.1	23	21	23	23	10
3	31	31	68	211.4	311.6	0.9	*0.0*	0.0	2.6	16	18	18	18	10
36	38	38	91	7.6	20.3	11.7	*0.0*	8.8	5.8	11	11	11	12	10
37	27	27	66	30.0	5.1	1.3	*0.0*	0.0	6.8	9	11	11	11	10
38	29	29	68	803.7	3.0	0.9	*0.0*	0.0	6.1	17	17	18	18	10
39	43	43	55	588.4	618.3	939.3	*0.0*	201.	3.2	15	20	27	26	10
40	32	32	91	141.7	436.6	1.6	*0.0*	0.0	5.5	11	9	11	11	10
44	34	34	46	183.0	47.9	5.2	*0.0*	32.1	3.2	12	10	14	16	10
45	31	31	43	4.7	21.7	12.1	*0.0*	0.1	14.6	8	10	11	11	10
46	40	40	52	696.5	497.2	47.0	*0.0*	585.	2.9	26	19	24	27	10
47	36	36	48	282.7	607.6	129.6	*0.0*	0.8	9.1	9	9	13	13	10
48	40	40	52	942.7	501.9	49.8	*0.0*	581.	5.0	23	19	24	27	10
50	28	28	40	631.9	77.1	24.0	*0.0*	0.1	4.8	12	11	12	12	10

Table 7.10: Results of the 50 `selected` test queries for NCI.

	Module size			MinAs search time (sec.)						#MinAs				
	$\mathcal{EL}^+$2SAT		CEL	$\mathcal{EL}^+$2SAT					CEL	$\mathcal{EL}^+$2SAT				CEL
	(b)	(e)		(a)	(b)	(e)	*enc.*	×2		(a)	(b)	(e)	×2	
1	3	3	8	0.2	0.0*	0.0*	*0.0*	0.0*	0.7*	1	1	1	1	1
2	3	3	18	0.5	0.0*	0.0*	*0.0*	0.0*	0.8*	2	2	2	2	2
3	2	2	6	0.2	0.0*	0.0*	*0.0*	0.0*	0.7*	1	1	1	1	1
4	7	7	35	0.7	0.1*	0.1*	*0.0*	0.0*	1.1*	3	4	4	4	4
5	9	9	37	0.9	0.0*	0.1*	*0.0*	0.0*	1.3*	5	5	5	5	5
6	5	5	23	3.6	0.0*	0.1*	*0.0*	0.0*	0.9*	3	3	3	3	3
8	5	5	11	0.7	0.0*	0.0*	*0.0*	0.0*	0.8*	2	2	2	2	2
9	3	3	19	0.2	0.0*	0.0*	*0.0*	0.0*	0.8*	1	1	1	1	1
10	2	2	7	0.2	0.0*	0.0*	*0.0*	0.0*	0.7*	1	1	1	1	1
11	11	11	36	71.5	0.3*	0.4*	*0.0*	0.0*	1.6*	5	6	6	6	6
12	4	4	7	0.2	0.0*	0.0*	*0.0*	0.0*	0.8*	1	1	1	1	1
13	6	6	19	9.6	0.1*	0.1*	*0.0*	0.0*	0.9*	3	3	3	3	3
14	12	12	24	1.2	0.8*	4.1*	*0.0*	0.0*	1.5*	6	7	7	7	7
15	4	4	9	0.3	0.0*	0.0*	*0.0*	0.0*	0.8*	2	2	2	2	2
16	2	2	8	0.2	0.0*	0.0*	*0.0*	0.0*	0.7*	1	1	1	1	1
17	3	3	16	0.2	0.0*	0.0*	*0.0*	0.0*	0.7*	1	1	1	1	1
18	6	6	28	0.9	0.0*	0.1*	*0.0*	0.0*	0.9*	2	2	2	2	2
19	9	9	55	207.	0.2*	0.1*	*0.0*	0.0*	1.3*	5	5	5	5	5
20	9	9	42	1.1	0.1*	0.2*	*0.0*	0.0*	1.4*	5	5	5	5	5
21	7	7	15	0.2	0.0*	0.0*	*0.0*	0.0*	0.8*	1	1	1	1	1
22	3	3	31	0.2	0.0*	0.0*	*0.0*	0.0*	0.8*	1	1	1	1	1
23	7	7	55	0.5	0.1*	0.1*	*0.0*	0.0*	1.2*	3	4	4	4	4
24	2	2	7	0.2	0.0*	0.0*	*0.0*	0.0*	0.7*	1	1	1	1	1
25	2	2	4	0.2	0.0*	0.0*	*0.0*	0.0*	0.7*	1	1	1	1	1
26	4	4	12	0.2	0.0*	0.0*	*0.0*	0.0*	0.8*	1	1	1	1	1
28	3	3	10	0.2	0.0*	0.0*	*0.0*	0.0*	0.7*	1	1	1	1	1
29	9	9	26	1.1	0.2*	0.1*	*0.0*	0.0*	1.1*	3	4	4	4	4
31	7	7	8	0.2	0.0*	0.0*	*0.0*	0.0*	0.8*	1	1	1	1	1
32	10	10	16	1.1	0.1*	0.1*	*0.0*	0.0*	0.9*	2	2	2	2	2
33	5	5	55	0.7	0.0*	0.0*	*0.0*	0.0*	0.9*	3	3	3	3	3
34	2	2	20	0.2	0.0*	0.0*	*0.0*	0.0*	0.8*	1	1	1	1	1
35	5	5	8	0.9	0.0*	0.0*	*0.0*	0.0*	0.8*	2	2	2	2	2
36	3	3	20	0.2	0.0*	0.0*	*0.0*	0.0*	0.7*	1	1	1	1	1
37	4	4	6	0.2	0.0*	0.0*	*0.0*	0.0*	0.7*	1	1	1	1	1
38	2	2	4	0.2	0.0*	0.0*	*0.0*	0.0*	0.7*	1	1	1	1	1
39	2	2	26	0.2	0.0*	0.0*	*0.0*	0.0*	0.7*	1	1	1	1	1
40	6	6	10	0.9	0.0*	0.1*	*0.0*	0.0*	0.8*	3	3	3	3	3
41	8	8	35	19.5	0.1*	0.3*	*0.0*	0.0*	1.1*	4	4	4	4	4
42	3	3	27	0.2	0.0*	0.0*	*0.0*	0.0*	0.8*	1	1	1	1	1
43	2	2	5	0.2	0.0*	0.0*	*0.0*	0.0*	0.7*	1	1	1	1	1
44	4	4	30	0.2	0.0*	0.0*	*0.0*	0.0*	0.8*	1	1	1	1	1
45	3	3	28	0.2	0.0*	0.0*	*0.0*	0.0*	0.8*	1	1	1	1	1
46	10	10	17	1.3	0.2*	0.4*	*0.0*	0.0*	1.2*	3	6	6	6	6
47	5	5	6	0.2	0.0*	0.0*	*0.0*	0.0*	0.7*	1	1	1	1	1
49	8	8	21	190.	0.1*	0.1*	*0.0*	0.0*	0.9*	3	3	3	3	3
50	4	4	13	0.5	0.0*	0.0*	*0.0*	0.0*	0.8*	2	2	2	2	2
7	15	15	18	14.5	0.4*	1.0*	*0.0*	0.0*	1.8	8	12	12	12	10
30	13	13	14	1.3	3.8*	0.7*	*0.0*	0.0*	1.4	5	11	11	11	10
48	13	13	22	691.	1.5*	0.7*	*0.0*	0.0*	1.6	11	11	11	11	10
27	22	22	27	346.	652.	526.	*0.0*	38.5*	2.1	13	27	28	31	10

Table 7.11: Results of the 50 `random` test queries for GeneOntology.

	Module size			MinAs search time (sec.)						#MinAs				
	$\mathcal{EL}^+$2SAT		CEL	$\mathcal{EL}^+$2SAT					CEL	$\mathcal{EL}^+$2SAT				CEL
	(b)	(e)		(a)	(b)	(e)	*enc.*	×2		(a)	(b)	(e)	×2	
8	16	16	25	141.0	24.3*	2.9*	*0.0*	0.0*	2.1*	4	9	9	9	9
27	15	15	23	123.2	26.8*	12.1*	*0.0*	0.1*	1.5*	5	7	7	7	7
39	15	15	23	289.6	36.2*	15.0*	*0.0*	0.1*	1.5*	4	7	7	7	7
42	13	13	41	677.5	2.5*	0.2*	*0.0*	0.0*	1.8*	9	9	9	9	9
47	15	15	23	23.1	11.9*	29.8*	*0.0*	0.1*	1.5*	3	7	7	7	7
49	12	12	28	2.3	0.2*	2.5*	*0.0*	0.0*	1.2*	6	6	6	6	6
50	11	11	27	50.7	0.8*	0.3*	*0.0*	0.0*	1.1*	6	6	6	6	6
3	20	20	33	161.9	118.3	84.3	*0.0*	0.5*	3.5*	9	9	9	9	9
7	18	18	27	3.6	94.9*	155.5*	*0.0*	1.0*	1.7	12	16	16	16	10
25	15	15	27	997.3	22.1*	33.3*	*0.0*	0.2*	1.5	12	16	16	16	10
28	13	13	20	40.7	4.1*	1.3*	*0.0*	0.0*	1.4	9	10	10	10	10
30	18	18	35	3.0	229.2*	194.8*	*0.0*	0.4*	1.8	11	17	17	17	10
32	13	13	20	197.6	2.4*	1.5*	*0.0*	0.0*	2.3	9	10	10	10	10
37	18	18	35	736.0	167.4*	56.8*	*0.0*	0.7*	1.9	12	17	17	17	10
40	14	14	22	651.2	1.5*	0.3*	*0.0*	0.0*	1.4	11	11	11	11	10
41	12	12	25	592.3	3.4*	0.6*	*0.0*	0.0*	1.4	11	11	11	11	10
44	19	19	33	847.1	398.8*	12.8*	*0.0*	0.2*	1.6	12	26	26	26	10
45	17	17	25	158.9	96.1*	22.6*	*0.0*	0.1*	1.7	14	15	15	15	10
18	19	19	41	19.6	165.6	503.6*	*0.0*	2.5*	2.1	5	16	16	16	10
1	25	25	45	352.4	19.3	125.8	*0.0*	1.8*	2.8	8	11	11	11	10
4	20	20	37	843.4	123.1	410.9	*0.0*	2.1*	1.6	16	20	20	20	10
5	26	26	37	41.0	100.6	44.7	*0.0*	93.1*	1.9	12	15	14	20	10
6	23	23	40	534.0	822.7	693.0	*0.0*	4.6*	1.7	18	21	24	24	10
9	20	20	37	891.2	205.1	799.0	*0.0*	3.5*	1.7	15	19	20	20	10
10	22	22	39	26.2	938.2	371.1	*0.0*	31.1*	1.7	16	22	18	23	10
11	20	20	37	874.7	693.9	327.3	*0.0*	1.3*	1.6	16	20	20	20	10
12	20	20	37	448.3	385.7	686.2	*0.0*	2.3*	2.7	16	20	20	20	10
13	22	22	31	398.7	865.3	9.8	*0.0*	23.0*	1.8	8	28	23	30	10
16	24	24	41	354.8	107.7	58.0	*0.0*	2.4*	1.9	10	10	18	21	10
17	21	21	44	934.1	997.0	898.6	*0.0*	14.6*	2.0	7	15	20	20	10
20	26	26	37	45.3	986.1	284.9	*0.0*	420.3*	1.9	9	18	17	19	10
21	25	25	37	678.2	163.2	972.4	*0.0*	0.8*	2.1	15	16	28	28	10
22	24	24	35	106.4	130.2	404.7	*0.0*	0.0*	1.7	11	12	18	18	10
23	26	26	37	2.6	729.5	25.0	*0.0*	26.5*	1.9	7	16	15	19	10
24	23	23	32	91.4	947.4	462.2	*0.0*	12.4*	1.9	15	17	18	18	10
26	21	21	30	307.2	883.0	741.6	*0.0*	2.7*	1.8	15	15	21	21	10
29	23	23	37	11.5	951.9	679.9	*0.0*	8.9*	2.8	13	25	22	32	10
31	25	25	45	293.8	780.3	470.0	*0.0*	201.8*	1.9	18	35	28	38	10
33	20	20	32	616.2	26.7	7.7	*0.0*	0.1*	3.3	8	14	14	14	10
34	25	25	45	222.6	265.2	0.2	*0.0*	22.9*	1.7	25	25	9	38	10
35	21	21	33	18.2	118.7	127.1	*0.0*	1.3*	1.9	9	16	16	16	10
36	23	23	37	692.6	830.5	93.0	*0.0*	1.8*	1.6	18	29	31	32	10
38	20	20	32	99.2	26.1	6.9	*0.0*	0.1*	2.5	9	14	14	14	10
43	20	20	32	524.6	19.9	10.5	*0.0*	0.1*	2.6	11	14	14	14	10
46	23	23	37	313.4	203.6	365.1	*0.0*	10.4*	1.7	14	32	29	32	10
48	20	20	29	793.8	142.5	426.4	*0.0*	1.9*	1.8	11	20	20	20	10
2	28	28	45	678.2	537.2	203.9	*0.0*	393.1	2.0	15	23	27	30	10
14	30	30	40	39.7	497.9	124.8	*0.0*	71.6	1.8	20	26	19	37	10
15	29	29	38	67.2	555.7	561.3	*0.0*	43.5	1.9	19	30	37	37	10
19	29	29	38	371.4	719.3	6.9	*0.0*	136.0	1.9	11	30	18	34	10

Table 7.12: Results of the 50 `selected` test queries for GeneOntology.

	Module size			MinAs search time (sec.)						#MinAs				
	$\mathcal{EL}^+$2SAT		CEL	$\mathcal{EL}^+$2SAT					CEL	$\mathcal{EL}^+$2SAT				CEL
	(b)	(e)		(a)	(b)	(e)	*enc.*	×2		(a)	(b)	(e)	×2	
2	3	3	6	0.1	0.0*	0.0*	*0.0*	0.0*	0.2*	1	1	1	1	1
6	2	2	58	0.1	0.0*	0.0*	*0.0*	0.0*	0.2*	1	1	1	1	1
7	6	6	17	0.1	0.0*	0.0*	*0.0*	0.0*	0.2*	1	1	1	1	1
11	9	9	9	0.1	0.0*	0.0*	*0.0*	0.0*	0.2*	1	1	1	1	1
14	6	6	7	0.1	0.0*	0.0*	*0.0*	0.0*	0.2*	1	1	1	1	1
16	5	5	9	0.1	0.0*	0.0*	*0.0*	0.0*	0.2*	1	1	1	1	1
18	8	8	9	0.1	0.0*	0.0*	*0.0*	0.0*	0.2*	1	1	1	1	1
19	8	8	22	0.1	0.0*	0.0*	*0.0*	0.0*	0.3*	1	1	1	1	1
21	3	3	10	0.1	0.0*	0.0*	*0.0*	0.0*	0.2*	1	1	1	1	1
24	4	4	9	0.1	0.0*	0.0*	*0.0*	0.0*	0.2*	1	1	1	1	1
32	3	3	7	0.1	0.0*	0.0*	*0.0*	0.0*	0.2*	1	1	1	1	1
35	4	4	23	0.2	0.0*	0.0*	*0.0*	0.0*	0.3*	2	2	2	2	2
36	5	5	22	0.1	0.0*	0.0*	*0.0*	0.0*	0.2*	1	1	1	1	1
40	2	2	6	0.1	0.0*	0.0*	*0.0*	0.0*	0.2*	1	1	1	1	1
43	2	2	6	0.1	0.0*	0.0*	*0.0*	0.0*	0.2*	1	1	1	1	1
46	3	3	8	0.1	0.0*	0.0*	*0.0*	0.0*	0.2*	1	1	1	1	1
47	3	3	8	0.1	0.0*	0.0*	*0.0*	0.0*	0.2*	1	1	1	1	1
1	156	12	153	0.6	0.0	0.0*	*0.0*	0.0*	1.0*	2	2	2	2	2
3	153	3	48	0.1	0.0	0.0*	*0.0*	0.0*	0.2*	1	1	1	1	1
5	154	9	22	0.1	0.0	0.0*	*0.0*	0.0*	0.2*	1	1	1	1	1
9	160	13	36	0.1	0.0	0.0*	*0.0*	0.0*	0.3*	1	1	1	1	1
10	151	13	34	0.1	0.0	0.0*	*0.0*	0.0*	0.3*	1	1	1	1	1
12	155	2	181	0.1	0.0	0.0*	*0.0*	0.0*	0.3*	1	1	1	1	1
17	157	2	113	0.1	0.0	0.0*	*0.0*	0.0*	0.2*	1	1	1	1	1
20	157	10	239	0.1	0.0	0.0*	*0.0*	0.0*	0.9*	1	1	1	1	1
26	153	2	60	0.1	0.0	0.0*	*0.0*	0.0*	0.2*	1	1	1	1	1
29	155	5	297	0.1	0.0	0.0*	*0.0*	0.0*	1.0*	1	1	1	1	1
30	155	10	153	0.4	0.0	0.0*	*0.0*	0.0*	0.7*	2	2	2	2	2
31	157	5	165	0.1	0.0	0.0*	*0.0*	0.0*	0.3*	1	1	1	1	1
33	159	10	171	0.1	0.0	0.0*	*0.0*	0.0*	0.4*	1	1	1	1	1
34	160	3	69	0.1	0.0	0.0*	*0.0*	0.0*	0.2*	1	1	1	1	1
37	155	5	135	0.1	0.0	0.0*	*0.0*	0.0*	0.4*	1	1	1	1	1
38	164	3	134	0.1	0.0	0.0*	*0.0*	0.0*	0.4*	1	1	1	1	1
42	153	15	53	0.1	0.0	0.0*	*0.0*	0.0*	0.3*	1	1	1	1	1
48	157	4	113	0.1	0.0	0.0*	*0.0*	0.0*	0.3*	1	1	1	1	1
49	155	16	69	0.1	0.0	0.0*	*0.0*	0.0*	0.4*	1	1	1	1	1
8	151	25	151	0.1	0.0	0.0	*0.0*	0.0*	0.5*	1	1	1	1	1
4	157	31	113	0.3	0.0	0.0	*0.0*	0.0	0.9*	2	2	2	2	2
13	154	30	109	0.1	0.0	0.0	*0.0*	0.0	0.6*	1	1	1	1	1
15	154	30	246	0.1	0.0	0.0	*0.0*	0.0	1.5*	1	1	1	1	1
22	159	31	121	0.6	0.2	0.5	*0.0*	0.0	1.7*	3	4	4	2	4
23	154	30	130	483.	0.1	0.2	*0.0*	0.0	2.1*	5	5	5	5	5
25	176	41	307	0.5	0.0	0.0	*0.0*	0.0	7.2*	2	2	2	2	2
27	157	31	113	0.3	0.0	0.0	*0.0*	0.0	1.3*	2	2	2	2	2
28	157	32	119	0.3	0.0	0.0	*0.0*	0.0	0.9*	2	2	2	2	2
39	155	32	114	0.4	0.0	0.0	*0.0*	0.0	1.2*	2	2	2	2	2
41	157	31	86	0.6	0.0	0.0	*0.0*	0.0	1.0*	2	2	2	2	2
44	157	31	113	0.4	0.0	0.0	*0.0*	0.0	1.2*	2	2	2	2	2
45	195	75	452	0.1	0.0	0.0	*0.0*	0.0	3.9*	1	1	1	1	1
50	158	34	130	0.3	0.0	0.0	*0.0*	0.0	1.0*	2	2	2	2	2

Table 7.13: Results of the 50 `random` test queries for NOT-GALEN.

	Module size			MinAs search time (sec.)						#MinAs				
	$\mathcal{EL}^+$2SAT		CEL	$\mathcal{EL}^+$2SAT					CEL	$\mathcal{EL}^+$2SAT				CEL
	(b)	(e)		(a)	(b)	(e)	*enc.*	×2		(a)	(b)	(e)	×2	
30	160	13	86	0.1	0.0	0.0*	*0.0*	0.0*	0.3*	1	1	1	1	1
1	158	27	97	0.3	0.0	0.0	*0.0*	0.0*	0.6*	2	2	2	2	2
2	157	26	96	0.3	0.0	0.0	*0.0*	0.0*	0.8*	2	2	2	2	2
3	158	26	130	0.3	0.0	0.0	*0.0*	0.0*	0.7*	2	2	2	2	2
4	158	26	130	0.3	0.0	0.0	*0.0*	0.0*	0.7*	2	2	2	2	2
5	158	26	127	0.3	0.0	0.0	*0.0*	0.0*	0.7*	2	2	2	2	2
6	157	26	116	0.3	0.0	0.0	*0.0*	0.0*	0.7*	2	2	2	2	2
7	158	26	130	0.2	0.0	0.0	*0.0*	0.0*	0.7*	2	2	2	2	2
8	158	26	130	0.2	0.0	0.0	*0.0*	0.0*	1.1*	2	2	2	2	2
9	158	26	130	0.2	0.0	0.0	*0.0*	0.0*	1.0*	2	2	2	2	2
10	158	26	130	0.2	0.0	0.0	*0.0*	0.0*	0.7*	2	2	2	2	2
11	158	26	130	0.2	0.0	0.0	*0.0*	0.0*	1.2*	2	2	2	2	2
12	158	26	130	0.3	0.0	0.0	*0.0*	0.0*	0.7*	2	2	2	2	2
13	158	26	130	0.3	0.0	0.0	*0.0*	0.0*	0.7*	2	2	2	2	2
14	157	26	106	0.3	0.0	0.0	*0.0*	0.0*	0.6*	2	2	2	2	2
15	157	26	105	0.3	0.0	0.0	*0.0*	0.0*	0.6*	2	2	2	2	2
16	157	26	116	0.3	0.0	0.0	*0.0*	0.0*	1.0*	2	2	2	2	2
17	158	26	132	0.3	0.0	0.0	*0.0*	0.0*	0.6*	2	2	2	2	2
18	157	26	106	0.2	0.0	0.0	*0.0*	0.0*	1.0*	2	2	2	2	2
19	157	26	105	0.2	0.0	0.0	*0.0*	0.0*	0.6*	2	2	2	2	2
20	158	26	127	0.3	0.0	0.0	*0.0*	0.0*	0.7*	2	2	2	2	2
21	157	26	110	0.2	0.0	0.0	*0.0*	0.0*	0.6*	2	2	2	2	2
22	157	26	113	0.2	0.0	0.0	*0.0*	0.0*	1.1*	2	2	2	2	2
23	157	26	96	0.2	0.0	0.0	*0.0*	0.0*	0.6*	2	2	2	2	2
24	157	26	112	0.2	0.0	0.0	*0.0*	0.0*	0.6*	2	2	2	2	2
25	158	26	97	0.2	0.0	0.0	*0.0*	0.0*	0.6*	2	2	2	2	2
26	157	26	112	0.2	0.0	0.0	*0.0*	0.0*	0.6*	2	2	2	2	2
27	157	26	115	0.2	0.0	0.0	*0.0*	0.0*	0.7*	2	2	2	2	2
28	157	26	106	0.3	0.0	0.0	*0.0*	0.0*	0.8*	2	2	2	2	2
29	157	26	109	0.2	0.0	0.0	*0.0*	0.0*	0.6*	2	2	2	2	2
31	157	26	113	0.3	0.0	0.0	*0.0*	0.0*	0.7*	2	2	2	2	2
32	157	26	116	0.2	0.0	0.0	*0.0*	0.0*	0.6*	2	2	2	2	2
33	177	25	297	0.1	0.0	0.0	*0.0*	0.0*	1.4*	1	1	1	1	1
34	177	25	180	0.1	0.0	0.0	*0.0*	0.0*	0.6*	1	1	1	1	1
35	177	25	183	0.1	0.0	0.0	*0.0*	0.0*	0.6*	1	1	1	1	1
36	177	25	232	0.1	0.0	0.0	*0.0*	0.0*	1.1*	1	1	1	1	1
37	177	25	215	0.1	0.0	0.0	*0.0*	0.0*	0.8*	1	1	1	1	1
38	177	25	277	0.1	0.0	0.0	*0.0*	0.0*	1.2*	1	1	1	1	1
39	157	28	111	0.3	0.0	0.0	*0.0*	0.0*	0.7*	2	2	2	2	2
40	157	28	111	0.2	0.0	0.0	*0.0*	0.0*	0.6*	2	2	2	2	2
41	157	24	110	0.1	0.0	0.0	*0.0*	0.0*	0.4*	1	1	1	1	1
42	157	25	113	0.2	0.0	0.0	*0.0*	0.0*	0.6*	2	2	2	2	2
43	157	25	116	0.2	0.0	0.0	*0.0*	0.0*	1.0*	2	2	2	2	2
44	157	25	110	0.2	0.0	0.0	*0.0*	0.0*	0.7*	2	2	2	2	2
45	157	25	119	0.2	0.0	0.0	*0.0*	0.0*	0.6*	2	2	2	2	2
46	157	25	113	0.2	0.0	0.0	*0.0*	0.0*	0.6*	2	2	2	2	2
47	157	25	113	0.2	0.0	0.0	*0.0*	0.0*	0.6*	2	2	2	2	2
48	157	25	113	0.3	0.0	0.0	*0.0*	0.0*	0.6*	2	2	2	2	2
49	160	23	239	0.1	0.0	0.0	*0.0*	0.0*	0.9*	1	1	1	1	1
50	157	25	113	0.3	0.0	0.0	*0.0*	0.0*	0.7*	2	2	2	2	2

Table 7.14: Results of the 50 `selected` test queries for NOT-GALEN.

Chapter 8

Conclusions

Discussion of the Achieved Results

In this work we have explored the idea of exploiting SAT-based techniques (including SMT) for the automated reasoning in Description Logics and the manipulation of ontologies, so that to be handled by state-of-the-art SAT and SMT tools. We believe that the results we achieved are only the first of a research line which can have a very broad impact, due to the manifold applications that ontologies and Description Logics have nowadays, especially in the field of Semantic Web and of bio-medical ontologies.

In particular with this work we covered a rich variety of problems by applying a wide range of solutions which are currently offered by the SAT/SMT research field, with the main objective of showing how these SAT-based techniques can be easily and efficiently adapted in order to solve different prominent research problems in the field of Description Logics.

The promising results gained in our first attempts gave us the motivation to apply those techniques, from time to time, to harder problems. In more details:

1. First, we have approached one of the central and most studied problem in the history of Modal and Description Logic, solving the satisfiability of modal K_m formulas (which is equivalent to concept satisfiability in $\mathcal{ALC}$ with empty TBoxes) via encoding into SAT, so that to be handled by state-of-the-art SAT-solvers.

2. Then, we have moved to even harder logics introducing TBoxes and including numerical constraints. We have solved concept satisfiability in acyclic $\mathcal{ALCQ}$ ontologies via the encoding into SMT modulo the Theory of Costs, so that to be handled by state-of-the-art SMT solvers.

3. We also have investigated non-standard reasoning services in the prominent area of lightweight Description Logics, solving the problem of axiom pinpointing in $\mathcal{EL}^+$ with general TBoxes, via the encoding of the classification of the ontology into a Horn-SAT formula. We have devised on top of a modern SAT solver an all-SMT technique able to find all the minimal axiom sets responsible for a subsumption relation. We

have also devised a novel, extremely efficient and precise modularization technique, which exploits the internal structures of modern state-of-the-art SAT solvers.

We have implemented all the three above proposed techniques in three different prototype tools that we compared with the other available state-of-the-art tools in very extensive empirical test sessions.

We have shown that, despite the intrinsic risk of blowup in the size of our encoded formulas (due to the complexity of reasoning in $\mathcal{ALC}$ or $\mathcal{ALCQ}$ and to the huge size of the real bio-medical ontologies in $\mathcal{EL}^+$, respectively) the performances of our techniques often push the state-of-the-art in the case of the problems we investigated, thanks also to the introduction of many novel and extremely effective SAT-based optimizations.

Moreover, some important considerations and remarks are due:

- Referring to the manipulation of ontologies, we showed how it can be convenient (as done in our approach) to move part of the reasoning complexity into a first "encoding" phase, that must be done only once and can be exploited "ad libitum" in order to have faster single answer to each query on the ontology.

- We show how the many techniques proposed by the SAT and SMT research communities can be efficiently reused and adapted in order to solve prominent problems in the area of Description Logics and ontologies. In particular, since these techniques have been designed to meet the requirements of the (usually challenging and huge) SAT/SMT real problems, they resulted to be particularly scalable and efficient.

- We remark that our approach allows for directly benefiting "for free" from current and future enhancements in SAT/SMT-solvers technology, without any extra effort or modification.

Future Works

We proposed some first steps in investigating the idea of apply SAT/SMT techniques for automated reasoning in Description Logics. Having in mind the impact that SAT and SMT techniques had in the model checking and formal verification areas, it is our opinion that this idea can be explored and extended in many possible directions with significant results.

In particular, research-wise, we see several important research lines to explore in order to extend or enhance our results:

- extend our techniques (when feasible) to other and more expressive logics, for example including: ABox reasoning, individuals, general TBoxes, role hierarchies, other logical operators, and so on and so forth;

- apply our $\mathcal{EL}^+$ technique to the tractable, increasingly more studied, (Horn) fragments of harder logics;

- approach other standard/non-standard reasoning services and more sophisticated inference problems;
- extend our axiom pinpointing technique either to new encodings or to a general SAT-based framework (appropriately using selector variables), in order to produce explanations for unsatisfiability or other inference problems;
- investigate alternative encodings or sets of completion rules, which might be more suitable for producing smaller/easier SAT/SMT formulas.

On the other hand we consider the implementation of a user-friendly tool including all the different reasoning services we offered for the different logics, and of a GUI so that to make it usable by the ontologies-domain experts.

Acknowledgments:

Roberto Sebastiani, Volker Haarslev and Alberto Griggio.

Bibliography

Areces, C., Gennari, R., Heguiabehere, J., & de Rijke, M. (2000). Tree-based Heuristics in Modal Theorem Proving. In *Proceedings of ECAI 2000*, pp. 199–203. IOS Press.

Armando, A., Castellini, C., Giunchiglia, E., Giunchiglia, F., & Tacchella, A. (2005). SAT-based decision procedures for Automated Reasoning: A unifying perspective. In *Proceedings of Mechanizing Mathematical Reasoning*, Vol. 2605 of *LNCS*, pp. 46–58. Springer.

Audemard, G., Cimatti, A., Kornilowicz, A., & Sebastiani, R. (2002). Bounded Model Checking for Timed Systems. In *Proceedings of FORTE'02*, Vol. 2529 of *LNCS*, pp. 243–259. Springer.

Baader, F. (1991). Augmenting Concept Languages by Transitive Closure of Roles: An Alternative to Terminological Cycles. In *Proceedings of IJCAI-91*, pp. 446–451.

Baader, F., Brandt, S., & Lutz, C. (2005). Pushing the $\mathcal{EL}$ Envelope. In *Proceedings of IJCAI-05*, pp. 364–369. Morgan-Kaufmann Publishers.

Baader, F., Franconi, E., Hollunder, B., Nebel, B., & Profitlich, H. J. (1994). An Empirical Analysis of Optimization Techniques for Terminological Representation Systems or: Making KRIS get a move on. *Applied Artificial Intelligence. Special Issue on Knowledge Base Management*, *4*, 109–132.

Baader, F., & Hollunder, B. (1991). A Terminological Knowledge Representation System with Complete Inference Algorithms. In *Proceedings of First International Workshop on Processing Declarative Knowledge*, Vol. 572 of *LNCS*, pp. 67–85. Springer–Verlag.

Baader, F., Lutz, C., & Suntisrivaraporn, B. (2006a). CEL—a polynomial-time reasoner for life science ontologies. In *Proceedings of IJCAR 2006*, Vol. 4130 of *LNAI*, pp. 287–291. Springer-Verlag.

Baader, F., Lutz, C., & Suntisrivaraporn, B. (2006b). Efficient Reasoning in $\mathcal{EL}^+$. In *Proceedings of Description Logics 2006*, Vol. 189 of *CEUR-WS*.

Baader, F. (2003). Terminological Cycles in a Description Logic with Existential Restrictions. In *Proceedings of IJCAI-03*, pp. 325–330. Morgan Kaufmann.

Baader, F., Brandt, S., & Lutz, C. (2008). Pushing the $\mathcal{EL}$ Envelope Further. In *Proceedings of the OWLED 2008 DC Workshop on OWL: Experiences and Directions*.

Baader, F., Calvanese, D., McGuinness, D., Nardi, D., & Patel-Schneider, P. F. (Eds.). (2003). *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press.

Baader, F., & Peñaloza, R. (2007). Axiom Pinpointing in General Tableaux. In *Proceedings of Tableaux 2007*, LNAI, pp. 11–27. Springer.

Baader, F., Peñaloza, R., & Suntisrivaraporn, B. (2007). Pinpointing in the Description Logic $\mathcal{EL}^+$. In *Proceedings of KI2007*, Vol. 4667 of *LNCS*, pp. 52–67. Springer.

Baader, F., & Suntisrivaraporn, B. (2008). Debugging SNOMED CT Using Axiom Pinpointing in the Description Logic $\mathcal{EL}^+$. In *Proceedings of KR-MED'08*, Vol. 410 of *CEUR-WS*.

Balsiger, P., Heuerding, A., & Schwendimann, S. (1998). Logics Workbench 1.0. In *Proceedings of Tableaux 1998*, Vol. 1397 of *LNCS*, pp. 35–37. Springer.

Barrett, C. W., Sebastiani, R., Seshia, S. A., & Tinelli, C. (2009). *Satisfiability Modulo Theories*, chap. 26, pp. 825–885. Vol. 185 of Biere et al. (Biere et al., 2009).

Baumgartner, P., Furbach, U., & Pelzer, B. (2010). The Hyper Tableaux Calculus with Equality and an Application to Finite Model Computation. *Journal of Logic and Computation*, *20*(1), 77–109.

Bechhofer, S., van Harmelen, F., Hendler, J., Horrocks, I., McGuinness, D. L., Patel-Schneider, P. F., & Stein, L. A. (2004). OWL Web Ontology Language reference. W3C Recommendation. Available at `http://www.w3.org/TR/owl-ref/`.

Benedetti, M. (2005). sKizzo: A Suite to Evaluate and Certify QBFs. In *Proceedings of CADE-20*, Vol. 3632 of *LNCS*, pp. 369–376. Springer.

Berezin, S., Ganesh, V., & Dill, D. L. (2003). An Online Proof-Producing Decision Procedure for Mixed-Integer Linear Arithmetic. In *Proceedings of TACAS 2003*, Vol. 2619 of *LNCS*, pp. 521–536. Springer.

Bienvenu, M. (2008). Complexity of Abduction in the $\mathcal{EL}$ Family of Lightweight Description Logics. In *Proceedings of KR2008*, pp. 220–230. AAAI Press.

Biere, A., Marijn, H. J. H., van Maaren, H., & Walsh, T. (Eds.). (2009). *Handbook of Satisfiability*, Vol. 185 of *Frontiers in Artificial Intelligence and Applications*. IOS Press.

Biere, A., Cimatti, A., Clarke, E. M., & Zhu, Y. (1999). Symbolic Model Checking without BDDs. In *Proceedings of TACAS 1999*, Vol. 1579 of *LNCS*, pp. 193–207. Springer.

Brand, S. (2008). Personal communication..

Brand, S., Gennari, R., & de Rijke, M. (2003). Constraint Programming for Modelling and Solving Modal Satisfability. In *Proceedings of CP 2003*, Vol. 2833 of *LNCS*, pp. 795–800. Springer.

Bruttomesso, R., Cimatti, A., Franzén, A., Griggio, A., & Sebastiani, R. (2008). The MathSAT 4 SMT Solver. In *Proceedings of CAV'08*, Vol. 5123 of *LNCS*, pp. 299–303. Springer.

Bryant, R. E. (1992). Symbolic Boolean manipolation with ordered binary-decision diagrams. *ACM Computing Surveys*, *24*(3), 293–318.

Cimatti, A., Franzén, A., Griggio, A., Sebastiani, R., & Stenico, C. (2010). Satisfiability Modulo the Theory of Costs: Foundations and Applications. In *Proceedings of TACAS 2010*, Vol. 6015 of *LNCS*, pp. 99–113. Springer.

Cimatti, A., Griggio, A., & Sebastiani, R. (2007). A simple and flexible way of computing small Unsatisfiable Cores in SAT Modulo Theories. In *Proceedings of SAT'07*, Vol. 4501 of *LNCS*, pp. 334–339. Springer.

Cimatti, A., Griggio, A., & Sebastiani, R. (2008). Efficient Interpolant Generation in Satisfiability Modulo Theories. In *Proceedings of TACAS 2008*, Vol. 4963 of *LNCS*, pp. 397–412. Springer.

Clarke, E., Grumberg, O., & Peled, D. A. (1999). *Model Checking*. MIT Press.

Colucci, S., Noia, T. D., Sciascio, E. D., Donini, F. M., & Mongiello, M. (2005). Concept Abduction and Contraction for Semantic-based Discovery of Matches and Negotiation Spaces in an e-Marketplace. *Electronic Commerce Research and Applications*, *4*(4), 345–361.

Davis, M., Longemann, G., & Loveland, D. (1962). A Machine Program for Theorem-proving. *Communications of the ACM*, *5*, 394–397.

Davis, M., & Putnam, H. (1960). A Computing Procedure for Quantification Theory. *Journal of the ACM*, *7*, 201–215.

Dershowitz, N., Hanna, Z., & Katz, J. (2005). Bounded Model Checking with QBF. In *Proceedings of SAT'05*, Vol. 3569 of *LNCS*, pp. 408–414. Springer.

Donini, F., & Massacci, F. (2000). EXPTIME tableaux for $\mathcal{ALC}$. *Artificial Intelligence*, *124*(1), 87–138.

Dutertre, B., & de Moura, L. (2006). A Fast Linear-Arithmetic Solver for DPLL(T). In *Proceedings of CAV'06*, Vol. 4144 of *LNCS*, pp. 81–94. Springer.

Eén, N., & Sörensson, N. (2004). An Extensible SAT-solver. In *Proceedings of SAT'03*, Vol. 2919 of *LNCS*, pp. 502–518. Springer.

Faddoul, J., Farsinia, N., Haarslev, V., & Möller, R. (2008). A Hybrid Tableau Algorithm for $\mathcal{ALCQ}$. In *Proceedings of Description Logics 2008*, Vol. 353 of *CEUR Workshop Proceedings*. CEUR-WS.org.

Faddoul, J., & Haarslev, V. (2010). Algebraic Tableau Reasoning for the Description Logic $\mathcal{SHOQ}$. *Journal of Applied Logic, Special Issue on Hybrid Logics*, *8*(4), 334–355.

Farsiniamarj, N., & Haarslev, V. (2010). Practical Reasoning with Qualified Number Restrictions: A Hybrid ABox Calculus for the Description Logic $\mathcal{SHQ}$. *Journal of AI Communications*, *23*(2-3), 205–240.

Fitting, M. (1983). *Proof Methods for Modal and Intuitionistic Logics*. D. Reidel Publishing.

Franconi, E. (1998). CRACK. In *Proceedings of Description Logics 1998*, Vol. 11 of *CEUR Workshop Proceedings*. CEUR-WS.org.

Garey, M. R., & Johnson, D. S. (1979). *Computers and Intractability*. Freeman and Company.

Gasse, F., & Haarslev, V. (2009). Expressive Description Logics via SAT: The Story so Far. In *Proceedings of the SMT-2009 Workshop*, Vol. 375 of *ACM International Conference Proceeding Series*, pp. 30–34.

Giunchiglia, E., Giunchiglia, F., Sebastiani, R., & Tacchella, A. (2000). SAT vs. Translation based decision procedures for modal logics: a comparative evaluation. *Journal of Applied Non-Classical Logics*, *10*(2), 145–172.

Giunchiglia, E., Narizzano, M., & Tacchella, A. (2002). Learning for Quantified Boolean Logic Satisfiability. In *Proceedings of AAAI/IAAI*, pp. 649–654.

Giunchiglia, E., Narizzano, M., & Tacchella, A. (2003). Backjumping for Quantified Boolean Logic Satisfiability. *Artificial Intelligence*, *145*(1-2), 99–120.

Giunchiglia, E., Narizzano, M., & Tacchella, A. (2006). Clause/Term Resolution and Learning in the evaluation of Quantified Boolean Formulas. *Journal of Artificial Intelligence Research (JAIR)*, *26*, 371–416.

Giunchiglia, E., Giunchiglia, F., & Tacchella, A. (2002). SAT-Based Decision Procedures for Classical Modal Logics. *Journal of Automated Reasoning (JAR)*, *28*(2), 143–171.

Giunchiglia, E., Marin, P., & Narizzano, M. (2009). *Reasoning with Quantified Boolean Formulas*, chap. 24, pp. 761–780. Vol. 185 of Biere et al. (Biere et al., 2009).

Giunchiglia, F., & Sebastiani, R. (1996). Building decision procedures for modal logics from propositional decision procedures - the case study of modal K. In *Proceedings of CADE-13*, Vol. 1104 of *LNAI*, pp. 583–597. Springer.

Giunchiglia, F., & Sebastiani, R. (2000). Building decision procedures for modal logics from propositional decision procedures - the case study of modal K(m). *Information and Computation*, *162*(1/2), 158–178.

Giunchiglia, F., Roveri, M., & Sebastiani, R. (1996). A New Method for Testing Decision Procedures in Modal and Terminological Logics. In *Proceedings of Description Logics 1996*, Vol. WS-96-05 of *AAAI Technical Report*, pp. 119–123. AAAI Press.

Grau, B. C., Horrocks, I., Kazakov, Y., & Sattler, U. (2007). Just the Right Amount: Extracting Modules from Ontologies. In *Proceedings of WWiW-07*, pp. 717–726. ACM.

Grumberg, O., Schuster, A., & Yadgar, A. (2004). Memory Efficient All-Solutions SAT Solver and Its Application for Reachability Analysis. In *Proceedings of FMCAD 2004*, Vol. 3312 of *LNCS*, pp. 275–289. Springer.

Haarslev, V., & Moeller, R. (2001). RACER System Description. In *Proceedings of IJCAR 2001*, Vol. 2083 of *LNAI*, pp. 701–706. Springer.

Haarslev, V., & Möller, R. (2003). Racer: A Core Inference Engine for the Semantic Web. In *Proceedings of EON2003*, pp. 27–36.

Haarslev, V., & Möller, R. (2001). Optimizing Reasoning in Description Logics with Qualified Number Restrictions. In *Proceedings of Description Logics 2001*, Vol. 49 of *CEUR Workshop Proceedings*. CEUR-WS.org.

Haarslev, V., Sebastiani, R., & Vescovi, M. (2011). Automated Reasoning in $\mathcal{ALCQ}$ via SMT. Submitted to *CADE-23*.

Haarslev, V., Timmann, M., & Möller, R. (2001). Combining Tableaux and Algebraic Methods for Reasoning with Qualified Number Restrictions. In *Proceedings of Description Logics 2001*, Vol. 49 of *CEUR Workshop Proceedings*. CEUR-WS.org.

Halpern, J. Y. (1995). The effect of bounding the number of primitive propositions and the depth of nesting on the complexity of modal logic. *Artificial Intelligence*, *75*(3), 361–372.

Halpern, J. Y., & Moses, Y. (1992). A guide to the completeness and complexity for modal logics of knowledge and belief. *Artificial Intelligence*, *54*(3), 319–379.

Heuerding, A., & Schwendimann, S. (1996). A benchmark method for the propositional modal logics K, KT, S4. Tech. rep. IAM-96-015, University of Bern, Switzerland.

Hollunder, B., & Baader, F. (1991). Qualifying Number Restrictions in Concept Languages. In *Proceedings of KR1991*, pp. 335–346.

Horridge, M., Parsia, B., & Sattler, U. (2008). Laconic and Precise Justifications in OWL. In *Proceedings of ISWC'08*, Vol. 5318 of *LNCS*, pp. 323–338.

Horrocks, I. (1998). Using an Expressive Description Logic: FaCT or Fiction?. In *Proceedings of KR1998*, pp. 636–647. Morgan Kaufmann.

Horrocks, I., & Patel-Schneider, P. F. (1999). Optimizing Description Logic Subsumption. *Journal of Logic and Computation*, *9*(3), 267–293.

Horrocks, I., Patel-Schneider, P. F., & Sebastiani, R. (2000). An Analysis of Empirical Testing for Modal Decision Procedures. *Logic Journal of the IGPL*, *8*(3), 293–323.

Horrocks, I., Patel-Schneider, P. F., & van Harmelen, F. (2003). From $\mathcal{SHIQ}$ and RDF to OWL: The making of a Web ontology language. *Journal of Web Semantics*, *1*(1), 7–26.

Horrocks, I., & Sattler, U. (1999). A Description Logic with Transitive and Inverse Roles and Role Hierarchies. *Journal of Logic and Computation*, *9*(3), 385–410.

Horrocks, I., Kutz, O., & Sattler, U. (2006). The Even More Irresistible $\mathcal{SROIQ}$. In *Proceedings of KR2006*, pp. 57–67. AAAI Press.

Horrocks, I., Sattler, U., & Tobies, S. (2000a). Practical Reasoning for Very Expressive Description Logics. *Logic Journal of the IGPL*, *8*(3), 239–264.

Horrocks, I., Sattler, U., & Tobies, S. (2000b). Reasoning with Individuals for the Description Logic $\mathcal{SHIQ}$. In *CADE-17*, Vol. 1831 of *LNCS*, pp. 482–496. Springer.

Hustadt, U., Schmidt, R. A., & Weidenbach, C. (1999). MSPASS: Subsumption Testing with SPASS. In *Proceedings of Description Logics 1999*, Vol. 22 of *CEUR Workshop Proceedings*, pp. 136–137. CEUR-WS.org.

Hustadt, U., & Schmidt, R. (1999). An empirical analysis of modal theorem provers. *Journal of Applied Non-Classical Logics*, *9*(4), 479–522.

Jin, H., Han, H., & Somenzi, F. (2005). Efficient Conflict Analysis for Finding All Satisfying Assignments of a Boolean Circuit. In *Proceedings of TACAS 2005*, Vol. 3440 of *LNCS*, pp. 287–300. Springer.

Jussila, T., & Biere, A. (2007). Compressing BMC encodings with QBF. *Electronic Notes in Theoretical Computer Science*, *174*(3), 45–56.

Jussila, T., Biere, A., Sinz, C., Kröning, D., & Wintersteiger, C. M. (2007). A First Step Towards a Unified Proof Checker for QBF. In *Proceedings of SAT'07*, Vol. 4501 of *LNCS*, pp. 201–214. Springer.

Kalyanpur, A., Parsia, B., Horridge, M., & Sirin, E. (2007). Finding All Justifications of OWL DL Entailments. In *Proceedings of 6th ISWC/ASWC*, Vol. 4825 of *LNCS*, pp. 267–280. Springer.

Kautz, H. A., McAllester, D. A., & Selman, B. (1996). Encoding Plans in Propositional Logic. In *Proceedings of KR1996*, pp. 374–384. AAAI Press.

Kazakov, Y. (2008). $\mathcal{RIQ}$ and $\mathcal{SROIQ}$ are Harder than $\mathcal{SHOIQ}$. In *Proceedings of KR2008*, pp. 274–284. AAAI Press.

Kazakov, Y. (2009). Consequence-Driven Reasoning for Horn SHIQ Ontologies. In *Proceedings of IJCAI-09*, pp. 2040–2045.

Konev, B., Lutz, C., Walther, D., & Wolter, F. (2008a). Logical Difference and Module Extraction with CEX and MEX. In *Proceedings of Description Logics 2008*, Vol. 353 of *CEUR-WS*.

Konev, B., Lutz, C., Walther, D., & Wolter, F. (2008b). Semantic Modularity and Module Extraction in Description Logics. In *Proceedings of ECAI 2008*, Vol. 178 of *Frontiers in Artificial Intelligence and Applications*, pp. 55–59. IOS Press.

Konev, B., Walther, D., & Wolter, F. (2008c). The Logical Difference Problem for Description Logic Terminologies. In *Proceedings of IJCAR 2008*, Vol. 5195 of *LNCS*, pp. 259–274. Springer.

Kurshan, R. P. (1994). *Computer-aided Verification of Coordinating Processes: the Automata-theoretic Approach*. Princeton University Press, Princeton, NJ, USA.

Ladner, R. (1977). The computational complexity of provability in systems of modal propositional logic. *SIAM Journal on Computing*, *6*(3), 467–480.

Lahiri, S. K., Nieuwenhuis, R., & Oliveras, A. (2006). SMT Techniques for Fast Predicate Abstraction. In *Proceedings of CAV'06*, Vol. 4144 of *LNCS*, pp. 424–437. Springer.

Lutz, C., Toman, D., & Wolter, F. (2009). Conjunctive Query Answering in the Description Logic $\mathcal{EL}$ Using a Relational Database System. In *Proceedings of IJCAI-09*, pp. 2070–2075.

Lutz, C., Areces, C., Horrocks, I., & Sattler, U. (2005). Keys, Nominals, and Concrete Domains. *Journal of Artificial Intelligence Research (JAIR)*, *23*(1), 667–726.

Lynce, I., & Silva, J. P. (2004). On Computing Minimum Unsatisfiable Cores.. In *Proceedings of SAT'04*, pp. 305–310.

Magka, D., Kazakov, Y., & Horrocks, I. (2010). Tractable Extensions of the Description Logic $\mathcal{EL}$ with Numerical Datatypes. In *Proceedings of IJCAR 2010*, LNCS, pp. 61–75. Springer.

Massacci, F. (1999). Design and Results of Tableaux-99 Non-Classical (Modal) System Competition. In *Proceedings of Tableaux 1999*, Vol. 1617 of *LNCS*, pp. 14–18. Springer–Verlang.

Massacci, F. (2000). Single Step Tableaux for modal logics: methodology, computations, algorithms. *Journal of Automated Reasoning (JAR)*, *24*(3), 319–364.

Massacci, F., & Donini, F. (2000). Design and results of TANCS-2000, Automated Reasoning with Analytic Tableaux and Related Methods. In *Proceedings of Tableaux 2000*, Vol. 1847 of *LNCS*, pp. 52–56. Springer.

McMillan, K. (2003). Interpolation and SAT-based model checking. In *Proceedings of CAV'03*, Vol. 2725 of *LNCS*, pp. 1–13.

Moskewicz, M. W., Madigan, C. F., Zhao, Y., Zhang, L., & Malik, S. (2001). Chaff: Engineering an Efficient SAT Solver. In *Proceedings of DAC'01*, pp. 530–535. ACM.

Motik, B., Patel-Schneider, P. F., & Parsia, B. (27 October 2009). OWL 2 Web Ontology Language: Structural Specication and Functional-Style Syntax. W3C Recommendation. Available at `http://www.w3.org/TR/owl2-syntax/`.

Motik, B., & Horrocks, I. (2008). Individual Reuse in Description Logic Reasoning. In *Proceedings of IJCAR 2008*, Vol. 5195 of *LNCS*, pp. 242–258. Springer.

Motik, B., Shearer, R., & Horrocks, I. (2007). Optimized Reasoning in Description Logics Using Hypertableaux. In *CADE-21*, Vol. 4603 of *LNCS*, pp. 67–83. Springer.

Motik, B., Shearer, R., & Horrocks, I. (2009). Hypertableau Reasoning for Description Logics. *Journal of Artificial Intelligence Research (JAIR)*, *36*, 165–228.

Narizzano, M., Pulina, L., & Tacchella, A. (2006). The QBFEVAL Web Portal. In *Proceedings of JELIA'06*, Vol. 4160 of *LNCS*, pp. 494–497. Springer. See: `http://www.qbflib.org/index_eval.php`.

Noy, N. F., & Musen, M. A. (2003). The PROMPT suite: interactive tools for ontology merging and mapping. *International Journal of Human-Computing Studies*, *59*, 983–1024.

Ohlbach, H. J., & Koehler, J. (1997). Role Hierarchies and Number Restrictions. In *Proceedings of Description Logics 1997*, Vol. 410 of *URA-CNRS*.

Ohlbach, H. J., & Koehler, J. (1999). Modal Logics, Description Logics and Arithmetic Reasoning. *Artificial Intelligence*, *109*(1-2), 1–31.

Pan, G., Sattler, U., & Vardi, M. Y. (2002). BDD-Based Decision Procedures for K. In *Proceedings of CADE-18*, Vol. 2392 of *LNCS*, pp. 16–30. Springer.

Pan, G., & Vardi, M. Y. (2003). Optimizing a BDD-based modal solver. In *Proceedings of CADE-19*, Vol. 2741 of *LNAI*, pp. 75–89. Springer.

Pan, G., & Vardi, M. Y. (2004). Symbolic decision procedures for QBF. In *Proceedings of CP2004*, Vol. 3258 of *LNCS*, pp. 453–467. Springer.

Papadimitriou, C. H. (1981). On the Complexity of Integer Programming. *Journal of the ACM*, *28*(4), 765–768.

Patel-Schneider, P. F. (1998). DLP system description. In *Proceedings of Tableaux 1998*, pp. 87–89.

Patel-Schneider, P. F., & Sebastiani, R. (2001). A new System and Methodology for Generating Random Modal Formulae. In *Proceedings of IJCAR 2001*, Vol. 2083 of *LNAI*, pp. 464–468. Springer-Verlag.

Patel-Schneider, P. F., & Sebastiani, R. (2003). A New General Method to Generate Random Modal Formulae for Testing Decision Procedures. *Journal of Artificial Intelligence Research (JAIR)*, *18*, 351–389.

Peñaloza, R., & Sertkaya, B. (2010a). Complexity of Axiom Pinpointing in the DL-Lite Family of Description Logics. In *Proceedings of ECAI 2010*, Vol. 215 of *Frontiers in Artificial Intelligence and Applications*, pp. 29–34. IOS Press.

Peñaloza, R., & Sertkaya, B. (2010b). On the Complexity of Axiom Pinpointing in the $\mathcal{EL}$ Family of Description Logics. In *Proceedings of KR2010*. AAAI Press.

Pipatsrisawat, T., & Darwiche, A. (2006). SAT Solver Description: Rsat. System Description. Available at: `http://fmv.jku.at/sat-race-2006/descriptions/9-rsat.pdf`.

Plaisted, D. A., Biere, A., & Zhu, Y. (2003). A satisfiability procedure for Quantified Boolean Formulae. *Discrete Applied Mathematics*, *130*(2), 291–328.

Rector, A., & Horrocks, I. (1997). Experience Building a Large, Re-usable Medical Ontology using a Description Logic with Transitivity and Concept Inclusions. In *Proceedings of the Workshop on Ontological Engineering, AAAI'97*. AAAI Press.

Schild, K. D. (1991). A correspondence theory for terminological logics: preliminary report. In *Proceedings of IJCAI-91*, pp. 466–471.

Schlobach, S., & Cornet, R. (2003). Non-Standard Reasoning Services for the Debugging of Description Logic Terminologies. In *Proceeding of IJCAI-03*, pp. 355–362. Morgan Kaufmann.

Schulz, S., Suntisrivaraporn, B., & Baader, F. (2007). SNOMED CT's problem list: Ontologists' and logicians' therapy suggestions. In *Proceedings of The Medinfo 2007 Congress*, SHTI-series. IOS Press.

Sebastiani, R. (2007a). From KSAT to Delayed Theory Combination: Exploiting DPLL Outside the SAT Domain. In *Proceedings of FroCoS'07*, Vol. 4720 of *LNCS*, pp. 28–46. Springer. Invited talk.

Sebastiani, R. (2007b). Lazy Satisfiability Modulo Theories. *Journal on Satisfiability, Boolean Modeling and Computation (JSAT)*, *3*, 141–224.

Sebastiani, R., & Vescovi, M. (2006). Encoding the Satisfiability of Modal and Description Logics into SAT: The Case Study of $K(m)/\mathcal{ALC}$. In *Proceedings of SAT'06*, Vol. 4121 of *LNCS*, pp. 130–135. Springer.

Sebastiani, R., & Vescovi, M. (2009a). Automated Reasoning in Modal and Description Logics via SAT Encoding: the Case Study of $K(m)/\mathcal{ALC}$-Satisfiability. *Journal of Artificial Intelligence Research (JAIR)*, *35*(1), 275–341.

Sebastiani, R., & Vescovi, M. (2009b). Axiom Pinpointing in Lightweight Description Logics via Horn-SAT Encoding and Conflict Analysis. In *Proceedings of CADE-22*, Vol. 5663 of *LNCS*, pp. 84–99. Springer.

Sebastiani, R., & Vescovi, M. (2011). Efficiently Debugging $\mathcal{EL}^+$ Ontologies via SAT and SMT techniques. Under submission to *Journal of Artificial Intelligence Research (JAIR)*.

Seidenberg, J., & Rector, A. (2006). Web Ontology Segmentation: Analysis, Classification and Use. In *Proceedings of WWW'06*, pp. 13–22. ACM.

Silva, J. P., & Sakallah, K. A. (1996). GRASP – A new Search Algorithm for Satisfiability. In *Proceedings of ICCAD'96*, pp. 220–227. IEEE Computer Society.

Sioutos, N., de Coronado, S., Haber, M. W., Hartel, F. W., Shaiu, W., & Wright, L. W. (2007). NCI Thesaurus: A semantic model integrating cancer-related clinical and molecular information. *Journal of Biomedical Informatics*, *40*(1), 30–43.

Sirin, E., Parsia, B., Grau, B. C., Kalyanpur, A., & Katz, Y. (2007). Pellet: A practical OWL-DL reasoner. *Journal of Web Semantics*, *5*(2), 51–53.

Spackman, K. A. (2000). Managing clinical terminology hierarchies using algorithmic calculation of subsumption: Experience with SNOMED RT. *Journal of American Medical Informatics Association, JAMIA (Fall Symposium Special Issue)*.

Spackman, K. A., Campbell, K. E., & Cote, R. A. (1997). SNOMED RT: A reference terminology for health care. *Journal of American Medical Informatics Association, JAMIA (Fall Symposium Supplement)*.

Stevenson, L., Britz, K., & Hörne, T. (2008). KT and S4 Satisfiability in a Constraint Logic Environment. In *PRICAI-08*, Vol. 5351 of *LNCS*, pp. 370–381. Springer.

Suntisrivaraporn, B. (2009). *Polynomial-Time Reasoning Support for Design and Maintenance of Large-Scale Biomedical Ontologies*. Ph.D. thesis, University of Dresden.

Suntisrivaraporn, B., Baader, F., Schulz, S., & Spackman, K. A. (2007). Replacing sep-triplets in snomed ct using tractable description logic operators. In *Proceedings of AIME'07*, LNCS. Springer-Verlag.

Suntisrivaraporn, B., Qi, G., Ji, Q., & Haase, P. (2008). A Modularization-Based Approach to Finding All Justifications for OWL DL Entailments. In *Proceedings of ASWC'08*, pp. 1–15. Springer-Verlag.

Tacchella, A. (1999). *SAT system description. In *Proceedings of Description Logics 1999*, Vol. 22 of *CEUR Workshop Proceedings*, pp. 142–144. CEUR-WS.org.

The G. O. Consortium (2000). Gene Ontology: Tool for the unification of biology. *Nature Genetics*, *25*, 25–29.

Tsarkov, D., & Horrocks, I. (2006). FaCT++ Description Logic Reasoner: System Description. In *Proceedings of IJCAR 2006*, Vol. 4130 of *LNAI*, pp. 292–297. Springer.

Vescovi, M. (2006). Automated Reasoning in Modal Logic via SAT. Master's thesis, Università degli Studi di Trento (University of Trento), Italy.

Voronkov, A. (1999). KЯ: a theorem prover for K. In *Proceedings of CADE-16*, Vol. 1632 of *LNAI*, pp. 383–387. Springer.

Voronkov, A. (2001). How to optimize proof-search in modal logics: new methods of proving redundancy criteria for sequent calculi. *ACM Transactions on Computational Logic*, *2*(2), 182–215.

Zhang, L., & Malik, S. (2002). Conflict driven learning in a Quantified Boolean satisfiability solver. In *Proceedings of ICCAD'02*, pp. 442–449. ACM.

Zhang, L., & Malik, S. (2003). Extracting Small Unsatisfiable Cores from Unsatisfiable Boolean Formula. In *Proceedings of SAT'03*.

Zhang, L., Madigan, C. F., Moskewicz, M. W., & Malik, S. (2001). Efficient Conflict Driven Learning in Boolean Satisfiability Solver. In *Proceedings of ICCAD'01*, pp. 279–285.

Zhang, L., & Malik, S. (2002). The Quest for Efficient Boolean Satisfiability Solvers. In *Proceedings of CAV'02*, Vol. 2404 of *LNCS*, pp. 17–36. Springer.