Department of
Information Engineering
and Computer Science
DISI

UNIVERSITY
OF TRENTO - Italy

DISI - Via Sommarive, 14 - 38123 POVO, Trento - Italy
http://disi.unitn.it

Automated Reasoning on TBoxes with Qualified Number Restrictions via SMT

Michele Vescovi michele.vescovi@disi.unitn.it
Roberto Sebastiani roberto.sebastiani@disi.unitn.it
DISI, Università di Trento
Via Sommarive 14, I-38123, Povo, Trento, Italy

Volker Haarslev haarslev@cse.concordia.ca
Concordia University, CSE
1455 de Maisonneuve Blvd.W., Montreal (QC), H3G 1M8, Canada

February 2011

Technical Report # DISI-11-001

Abstract

The problem of reasoning with qualified number restrictions in Description Logics (DLs) has been investigated since the very first research steps in automated reasoning for DLs. Moreover, developing techniques for optimized reasoning with qualified number restrictions and has gained further importance since qualified number restrictions have been added to the forthcoming standard OWL 2 for Semantic Web applications.

On the one hand, however, actual DL reasoning 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 investigate 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.

In this paper we propose and investigate a novel approach for concept satisfiability in acyclic $\mathcal{ALCQ}$ ontologies. The idea is to encode an $\mathcal{ALCQ}$ ontology into a formula in Satisfiability Modulo the Theory of Costs (SMT($\mathcal{C}$)), which is a specific and computationally much cheaper subcase of Linear Arithmetic under the Integers ($\mathcal{LA}(\mathbb{Z})$), and to exploit the power of modern SMT solvers to compute every concept-satisfiability query on a given ontology. We have implemented and tested our approach (called $\mathcal{ALCQ}2SMT_{\mathcal{C}}$) and some optimizations on a wide set of synthesized benchmark formulas, comparing the approach with the main state-of-the-art tools available. Our empirical evaluation confirms the potential of the approach.

1 Introduction

Description logics (DLs) form one of the major foundations of the semantic web and its web ontology language (OWL). In fact, OWL 2, a recent W3C recommendation, is a syntactic variant of a very expressive DL that supports reasoning with so-called *qualified number restrictions* (QNRs). A sound and complete calculus for reasoning with the DL $\mathcal{ALCQ}$ that adds QNRs to the basic DL $\mathcal{ALC}$ was first proposed in [19]. For example, this calculus decides the satisfiability of an $\mathcal{ALCQ}$ concept $(\geq 5\, s.C \sqcap \geq 5\, s.D \sqcap \leq 2\, s.E)$ by trying to find a model with fillers for the role s such that at least 5 fillers are instances of C, at least 5 fillers are instances of D, and at most 2 fillers are instances of E. It satisfies the at-least restrictions by creating 10 fillers for S, 5 of which are instances of C and 5 are instances of D. A concept choose rule non-deterministically assigns E or $\neg E$ to these fillers. In case the at-most restriction $(\leq 2\, s.E)$ is violated a merge rule non-deterministically merges pairs of fillers for S that are instances of E [19]. Searching for a model in such an arithmetically uninformed way can become very inefficient especially when bigger numbers occur in QNRs or several QNRs interact. To the best of our knowledge this calculus still serves as reference in most OWL reasoners (e.g., Pellet [34], FaCT++ [35], HermiT [26]) for implementing reasoning about QNRs. The only exception is Racer [15] where conceptual QNR reasoning is based on an algebraic approach [17] that integrates integer linear programming with DL tableau methods.

In recent works, [31, 32] explored the idea of performing automated reasoning tasks in DLs by encoding problems into Boolean formulas and by exploiting the power of modern SAT techniques. In particular, the experiments in [31] showed that, in practice and despite the theoretical worst-case complexity limits, this approach could handle most or all the $\mathcal{ALC}$ satisfiability problems which were at the reach of the other approaches, with performances which were comparable with, and often better than, those of state-of-the-art tools. Moreover, recently, a new algebraic approach was presented for $\mathcal{ALCQ}$ [8] and extended to deal with $\mathcal{SHQ}$ [11] and $\mathcal{SHOQ}$ [9]. These approaches represent knowledge about interacting QNRs as systems of linear inequations where variables represent cardinalities of sets of domain elements (e.g., role fillers) divided into mutually disjoint decompositions. On a set of synthetic QNR benchmarks these algebraic approaches demonstrated really a superior performance for most test cases [11, 10].

The work presented in this document was inspired by these two novel approaches, combined with the progress in satisfiability modulo theory (SMT) solving techniques. The main idea of this work is thus to encode an $\mathcal{ALCQ}$ ontology into a formula in Satisfiability Modulo the Theory of Costs (SMT($\mathcal{C}$)) [4], which is a specific and computationally much cheaper subcase of Linear Arithmetic under the Integers ($\mathcal{LA}(\mathbb{Z})$), and to exploit the power of modern SMT solvers to compute every concept-satisfiability query on a given ontology. We have implemented and tested our approach (called $\mathcal{ALCQ}2SMT_{\mathcal{C}}$) and some optimizations on a wide set of synthesized benchmark problems with QNRs, comparing our approach with main state-of-the-art OWL reasoners. Our empirical evaluation demonstrates the potential of our approach and, compared with the tested OWL reasoners, demonstrates a significantly better performance in the case of benchmarks having multiple/balanced sources of complexity.

1.1 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 [27, 28, 19, 21, 20, 14, 8, 11].

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 (e.g., [19, 20, 1]) 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 individuals until all the at-most restrictions are satisfied. Many optimization like, e.g., dependency-directed backtracking [20], has been proposed in order to improve this method (we refer the reader to the literature [1]). However, searching for a model in such an arithmetically uninformed or blind way is usually very inefficient, especially in cases with bigger numbers occurring in qualified number restrictions.

For this reason various alternative algebraic methods to the traditional tableau algorithms has been tried [28, 14, 17, 11] in order to enrich the tableau-based reasoning engine with calculus which benefits from arithmetic methods. Haarslev et all. perform many attempts in this research direction [14, 17, 8, 11], a wider literature on these attempts and other arithmetic-based approaches can be found in the lately cited works.

In particular, in [8, 11, 9] has been recently developed a hybrid approach for $\mathcal{SHOQ}$ combining the standard tableau methods with an inequation solver. Such hybrid calculus is based on a standard tableau algorithm for $\mathcal{SH}$ modified and 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 [28]. 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.

Importantly the hybrid algorithm performs the arithmetic component of reasoning before creating any role filler, thus there is no need for a mechanism of merging role fillers. Moreover, it 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 of [12]. 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 2.3). In brief, [12] 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 [30, 31] the $\mathcal{ALC}$ part of the input $\mathcal{SHOQ}$ problem is reduced to a Boolean abstraction including atoms in the other logical constructors which are not expressible in $\mathcal{ALC}$. Then a specific tableau-like solver is responsible to verify if an assignment to the Boolean abstraction satisfies the logical axioms not expressible in $\mathcal{ALC}$, hence the axioms which cannot be rewritten into a SAT problem. Substantially, this approach is an extension of the DPLL-based approach for modal logic proposed in[13],and it mostly differentiate from it and from the modern tableau-based approaches for the expansion at the $\mathcal{ALC}$ level of the Boolean component of reasoning and for the tighter interactions between the theory solver and the assignment enumerator. We are not aware of any further investigation or advance in this approach, that in [12] is only at a preliminary level.

2 Background

2.1 The Description Logic $\mathcal{ALCQ}$

The logic $\mathcal{ALCQ}$ extends the well known logic $\mathcal{ALC}$ adding the *qualified number restriction* constructors. In more details, the *concept descriptions* in $\mathcal{ALCQ}$ (namely $C, D, \ldots$) are inductively defined through the constructors listed in the upper half of Table 1, starting from the non-empty and pair-wise disjoint sets of *concept names* N_C (we use the letters $A, B, \ldots$ for concept names) and *role names* N_R (we use the letters $r, s, \ldots$ for role names). It allows for negations, conjunctions/disjunctions, existential/universal restrictions and, indeed, qualified number restrictions. An $\mathcal{ALCQ}$ *TBox* (or *ontology*) is a finite set of general concept inclusion (GCI) axioms as defined in the lower half of Table 1.

The semantic of $\mathcal{ALCQ}$ 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) $A \in N_C$ to a set $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$ and maps each role name (atomic role) r to a binary relation $r^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$. In the right-most column of Table 1 the inductive extensions of $\cdot^{\mathcal{I}}$ to arbitrary concept descriptions are defined, where n and m are positive integer values and $FIL(r, x)$ is the set of the r-fillers of the individual $x \in \Delta^{\mathcal{I}}$ for the role $r \in N_R$ and is defined as

	Syntax	Semantic
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	$\exists r.C$	$\{x \in \Delta^{\mathcal{I}} \mid \text{there exists } y \in \Delta^{\mathcal{I}} \text{ such that } (x,y) \in r^{\mathcal{I}} \text{ and } y \in C^{\mathcal{I}}\}$
universal restriction	$\forall r.C$	$\{x \in \Delta^{\mathcal{I}} \mid \text{for all } y \in \Delta^{\mathcal{I}} \text{ such that } (x,y) \in r^{\mathcal{I}} \text{ then } y \in C^{\mathcal{I}}\}$
$\geq$ number restriction	$\geq nr.C$	$\{x \in \Delta^{\mathcal{I}} \mid \lvert FIL(r,x) \cap C^{\mathcal{I}} \rvert \geq n\}$
$\leq$ number restriction	$\leq mr.C$	$\{x \in \Delta^{\mathcal{I}} \mid \lvert FIL(r,x) \cap C^{\mathcal{I}} \rvert \leq m\}$
GCI	$C \sqsubseteq D$	$C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$

Table 1: Syntax and semantics of $\mathcal{ALCQ}$ ($n \geq 1$ and $m \geq 0$)s.

$FIL(r,x) = \{y \in \Delta^{\mathcal{I}} | (x,y) \in r^{\mathcal{I}}\}$.

An interpretation $\mathcal{I}$ is a *model* of a given TBox $\mathcal{T}$ if and only if the conditions in the Semantic column of Table 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 $x \in C^{\mathcal{I}}$.

2.1.1 An $\mathcal{ALCQ}$ Normal Form

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. 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 we represent with $\dot{\neg}C$ 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 $C, C_1, C_2, D, D_1, D_2 \in \mathsf{BC}_{\mathcal{T}}$, $r \in N_R$ 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:

$$C \sqsubseteq D \qquad \sqcap_i C_i \sqsubseteq D \qquad C \sqsubseteq \sqcap_i D_i \tag{1}$$
$$\Re r.C \sqsubseteq D \qquad C \sqsubseteq \Re r.D \qquad \Re \in \{\forall, \geq n, \leq m\}, \text{with } n, m \geq 1 \tag{2}$$

where C, C_i, D, D_i are basic concepts. Finally, notice that the first normal form (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 $r \in N_R$. [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]

2.2 Basics on Conflict-Driven Clause-Learning SAT Solving

For the best comprehension of the content of the next sections, 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 [33, 37, 7, 23].

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

[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 [31]. 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.

```
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.                 }
13.                 else break;
14.             }
15.             decide_next_branch(φ, μ);
16.         }
```

Figure 1: Schema of a CDCL DPLL SAT solver.

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.

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.

Most state-of-the-art SAT procedures are evolutions of the Davis-Putnam-Longeman-Loveland (DPLL) procedure [6, 5] and they are based on the CDCL paradigm [33, 36]. A high-level schema of a modern CDCL DPLL engine, adapted from the one presented in [37], is shown in Figure 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, 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`$(\texttt{blevel}, \varphi, \mu)$ 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* [24]. 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. 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 [33, 24, 36]. 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 [37]), 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.

2.3 Satisfiability Modulo Theory

Satisfiability Modulo (the) Theory $\mathcal{T}$, *SMT*($\mathcal{T}$), is the problem of deciding the satisfiability of (typically) ground formulas under a (combination of) background theory $\mathcal{T}$. (Notice that $\mathcal{T}$ can also be a combination of simpler theories: $\mathcal{T} \stackrel{def}{=} \bigcup_i \mathcal{T}_i$.) 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 called a $\mathcal{T}$*-conflict set*, and $\neg\eta$ a $\mathcal{T}$*-conflict clause*). Ff 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 symbols φ, ψ, ϕ denote $\mathcal{T}$-formulas, and μ, η denote sets of $\mathcal{T}$-literals. If $\mathcal{T}2\mathcal{B}(\mu) \models \mathcal{T}2\mathcal{B}(\varphi)$, then we say that μ *propositionally satisfies* φ written $\mu \models_p \varphi$.

In a lazy SMT($\mathcal{T}$) solver the truth assignments for φ are enumerated and checked for $\mathcal{T}$-satisfiability, returning either sat if one $\mathcal{T}$-satisfiable truth assignment is found, unsat otherwise. In practical implementations, φ is given as input to a modified version of DPLL, and when an assignment μ is found s.t. $\mu \models_p \varphi$ μ is fed to the $\mathcal{T}$*-solver*; if μ is $\mathcal{T}$-consistent, then φ is $\mathcal{T}$-consistent; otherwise, $\mathcal{T}$*-solver* returns the conflict set η causing the inconsistency. Then the $\mathcal{T}$-conflict clause $\neg\eta$ is fed to the backjumping and learning mechanism of DPLL ($\mathcal{T}$*-backjumping* and $\mathcal{T}$*-learning*).

Important optimizations are *early pruning* and $\mathcal{T}$*-propagation*: the $\mathcal{T}$-solver is invoked also on an intermediate assignment μ: if it is $\mathcal{T}$-unsatisfiable, then the procedure can backtrack; if not, and if the $\mathcal{T}$*-solver* performs a $\mathcal{T}$-deduction $\{l_1, ..., l_n\} \models_{\mathcal{T}} l$, 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 interested reader is pointed to, e.g., [29, 2], for details and further references.

2.3.1 The Theory of Costs

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 $\mathcal{F} \stackrel{def}{=} \{cost_i \mid i = 1, \ldots, M\}$ 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

$$(cost_i \leq c), \tag{3}$$

with c is an integer value. [3] We focus on problems in which every $cost_i$ is a Boolean cost function (over the integers) in the form:

$$cost_i = \sum_{j=1}^{N_i} \textit{if-then-else}(\psi_{i,j}, c^{\top}_{i,j}, c^{\bot}_{i,j}) \tag{4}$$

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

$$cost_i = \sum_{j=1}^{N_i} \textit{if-then-else}(A_{i,j}, c_{i,j}, 0) \tag{5}$$

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) can be convert straightforwardly and in linear time into another problem with cost functions in form (5), not affecting the solutions of the problem [4].

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 (3), i.e. finding a satisfying assignment for φ having a cost within the admissible range. Every function (5) 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 [4]. However, for efficiency reasons the problem has been addressed in SMT by introducing an ad-hoc *Theory of Costs* $\mathcal{C}$ [4] (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 (5);
- 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 "$(cost_i \leq c)$" (3), 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.
- 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 (5), i.e. the predicate is true if $A_{i,j}$ in (5) 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. [4]For every function of type (5) 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 (5) of the incur costs determined by the assignment of the IC-literals respect the constraints (3) 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};

[3] 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 $(cost_i \leq c-1) \vee \neg(cost_i \leq c)$.

[4] 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'.

- 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.

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

3.1 Possible solutions

In [31, 32] it has been shown that it is possible to efficiently perform reasoning in Description Logic via the encoding into SAT problems. SAT-based technologies have proved to be very mature and largely successful in a wide range of domains. In particular, state-of-the-art SAT solvers, nowadays, are tools able to solve problems of hundreds of millions of variables and clauses. 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. In particular, following the experience acquired from [31, 32] 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 expressivity of the many embedded theories. The idea is to use SMT in the development of new techniques for optimized reasoning with numerical constraints in Description Logics, especially for what concerns those logics which provide language constructors that are somewhat similar to those of the theories that SMT includes.

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 Satisfiability Modulo Theory. With this objective in mind, seen the previous approaches to the problem in literature and given the many kinds of numerical reasoning available in SMT (including solving inequations, counting and others), we identified two main possible alternative encoding solutions:

- To follow the hybrid approach exposed in [8, 9] and to use the theories of SMT in order to perform numerical reasoning on the cardinality of sets of individuals. The approach of [8, 9] is based on the idea of partitioning the space of the possible individuals modeling the given problem in disjoint sets. Every partition represents a unique combination of features the individuals match, i.e. a different intersection of the interpretations of the concepts involved in the numerical restrictions. Then it verifies the satisfiability of qualified number restrictions through a set of linear integer inequations on the values of the partitions' cardinalities. The approach is called hybrid because it combines the typical tableau approach together with a inequations solver handling the arithmetical part of the problem. This 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 representing the whole set of individuals with the same features, instead of creating many different equivalent individuals with the need of mechanism of merging.

  An encoding of this approach into $SMT(\mathcal{LA}(\mathbb{Z}))$ (i.e. Satisfiability Modulo the Theory of Linear Arithmetic over the integers) is quite intuitive. Here we don't expose the details, but the idea is to exploit the Boolean component of SMT in order to represent the satisfiability of proxy individuals and the implications among proxy individuals of different partitions, while to rely on the $\mathcal{LA}(\mathbb{Z})$-solver to check the numerical consistency of the partitions cardinalities wrt. the existing restrictions.

- To devise a tableau-like approach, in which many individuals are created in order to individually satisfy the concepts and qualified number restrictions in the given problem, and to use the theories of SMT in order to count and bound those individuals. In this case the idea should be to exploit the Boolean component of SMT in order to represent the satisfiability of every individual, the relations among individuals and their membership to concept interpretations. The counting of individuals performed by the SMT theories, instead, forces the existence or non-existence of individuals and their membership to some concept interpretations, so that if many equivalent individuals exist only

a number of individuals that is consistent wrt. the existing lower- or upper-bounds, respectively, is "enabled".

It is possible to encode this approach into $SMT(\mathcal{LA}(\mathbb{Z}))$ or, even better, in SMT modulo the Theory of Costs [4] (SMT($\mathcal{C}$)), that we think more naturally fits the required expressivity of numerical restrictions. The Theory of Costs ($\mathcal{C}$), in fact, is a subset of linear arithmetic over the integer ($\mathcal{LA}(\mathbb{Z})$), in which it is possible to define multiple cost variables/functions and define both increases and lower/upper-bounds on such costs. Being $\mathcal{C}$ a subset of $\mathcal{LA}(\mathbb{Z})$, further than being a more direct and easy formalism, should have a lighter and more specialized theory solver.

Between these two alternatives we privilege the encoding into $SMT(\mathcal{C})$, due to the following reasons:

- The Theory of Costs $\mathcal{C}$ is a really simple theory who needs a lightweight and simple solver (based on sums and checks wrt. given bounds) while Linear Arithmetic $\mathcal{LA}$ is much more complicated and needs a, likely, much more time-consuming theory solver.
- The consideration above, also, meets the idea of 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 lighten as much as possible the single but numerous queries (performed on SMT), for which a fast response is more important.
- The approach based on partitioning leads a-priori to an exponential number of individuals and variables for every different set of qualified number restrictions that must be encoded. In fact the number of possible distinct intersection of n distinct concepts is 2^n. Encoding such an approach in $SMT(\mathcal{LA}(\mathbb{Z}))$ would affect exponentially both in the number of the Boolean variables and in the number of the integer ones. On the contrary, in the $SMT(\mathcal{C})$ encoding, the number of integer variables introduced for every group of restrictions is linear in the number of restrictions, the number of individuals is linear in the values included in the number restrictions and the total number of Boolean variables and cost contributions polynomial in these values.
- The linear dependence of the second approach from the values included in the restrictions, when replicated at every encoding level, can be more negatively impacting than the exponentiality of the first approach wrt. the number of restrictions, but we think that:
  - in real world, ontologies more commonly have a high number of qualified number restrictions than big values occurring in them (that, further, can be rationalized in some cases):
  - 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 in the theory solvers;
  - The second approach allows for heuristics or enhanced encodings in which the big numerical intervals generated by the values of the number restrictions can be handled introducing groups of individual, in place of single individuals, resulting in an even smaller number of variables.

Notice that, somehow, also our approach can be defined "hybrid". In fact, even if it simulates the tableau approach, it combines two orthogonal components of reasoning by mean of the SMT solver.

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

We encode the problem of concept satisfiability in description logic $\mathcal{ALCQ}$ wrt. acyclic TBoxes into an SMT($\mathcal{C}$) (Satisfiability Modulo the Theory of Costs) problem. Given an acyclic $\mathcal{ALCQ}$ TBox $\mathcal{T}$ we denote with $\mathcal{ALCQ}2SMT_{\mathcal{C}}(\mathcal{T})$ the encoding of $\mathcal{T}$ into SMT($\mathcal{C}$). We also assume that every axiom/concept description in $\mathcal{T}$ is in the normal form exposed in Section 2.1.1, thus, in particular, in NNF. In a nutshell, the encoding simulates the construction of an interpretation $\mathcal{I}$ observing the semantic of the encoded TBox $\mathcal{T}$, by:

- introducing possible individuals for the domain $\Delta^{\mathcal{I}}$ of the interpretation $\mathcal{I}$;
- representing with Boolean variables whether an individual is in $\Delta^{\mathcal{I}}$ and whether it belongs to the interpretation of one specific concept;

- using $\mathcal{C}$-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.

If a satisfying truth assignment μ for $\mathcal{ALCQ}2SMT_{\mathcal{C}}(\mathcal{T})$ exists, then from μ it is possible to obtain a model for $\mathcal{T}$, thus, in particular, the number of individuals in such a model respects all the numerical constraints given by the qualified number restrictions in $\mathcal{T}$.

We represent uniquely *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. A label σ can be either the label 1 or a label in the form $\sigma'.r.n$, where σ' is another label, $r \in N_R$ and $n \geq 1$ an integer value. 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 the individual σ is an instance of the concept C in the hypothetical interpretation $\mathcal{I}$, i.e., briefly, $\sigma \in C^{\mathcal{I}}$.

Definition 1. We define $A_{\langle\ ,\ \rangle}$ an injective function which maps one instantiated concept $\langle \sigma, C \rangle$ such that C is not a negation (i.e. it is in the form $\neg C'$), 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 and 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 2. 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 (since we are considering concepts in normal form), 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 qualified number restrictions 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 1 and 2 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 in the encoding. Given those definitions, the encoding simulates the construction of an interpretation by introducing new individuals, by establishing relations between individuals and by regulating the membership of individuals to the interpretations of concepts in $\mathcal{T}$, counting the occurrences. The encoding works essentially by mean of the following principles:

- Uses 1 as the root individual.
- The semantic of the concept inclusions stated by the axioms of $\mathcal{T}$ is maintained through Boolean implications between the variables representing instantiated concepts.
- Given an individual σ every at-least qualified number restriction $\langle \sigma, \geq nr.C \rangle$ is handled by introducing exactly n new possible individuals $\sigma.r.i$ in relation with σ through r and specifically satisfying the concept C. This ensures that it there exists the required minimum number of individuals satisfying the number restriction. The existence of the individuals is forced by binding each of them to an incur cost for the respective cost variable of value 1, and then fixing a lower-bound for cost variable.
- When many at-least restrictions coexist wrt. the same individual σ, many different new individuals are introduced in order to trivially satisfy all such at-least restrictions. However, if also at-most restrictions $\langle \sigma, \leq mr.C \rangle$ exist, the number of individuals $\sigma.r.i$ that can really exist in the interpretation of C is not unlimited but bounded by the values m. Every at-most restriction is handled by fixing an upper-bound for the respective cost variables $\mathsf{indiv}^{C}_{\sigma.r}$ which "counts" the individuals in relation with σ through r and satisfying C. Thus, in order to allow for satisfying both at-least and at-most

restrictions, the encoding must allow for sharing the individuals independently introduced for distinct at-least restrictions, so that one single individual can satisfy more than one at-least restriction and can concur to more than one cost variable. This reduces the total number of distinct individuals that must be enabled and counted.

Now we give a formal description of such an encoding.

Definition 3. Let $\mathcal{T}$ being an acyclic $\mathcal{ALCQ}$ TBox in normal form [5] 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;
- $\mathcal{I}^{\mathcal{T}}_{-}, \mathcal{I}^{\mathcal{T}}_{+}$ are two sets containing all the possible instantiated concepts represented by the encoding, which combines the individuals of $\Sigma^{\mathcal{T}}$ with the basic concept of $\mathcal{T}$, and which represents respectively implicant and implied instantiated concepts.
- $A_{\langle\ ,\ \rangle}$ is the function uniquely mapping possible instantiated concepts to Boolean variables defined in Definition 1;
- indiv is the function mapping possible instantiated concepts to individuals cost variables defined in Definition 2;
- $\varphi^{\mathcal{T}}$ is a CNF Boolean combination of propositional- and $\mathcal{C}$-literals encoding $\mathcal{T}$ into SMT($\mathcal{C}$). [6]

Since $\varphi^{\mathcal{T}}$ is in CNF, it is a conjunction of a set of clauses made up of disjunctions of propositional- and $\mathcal{C}$-literals. Thus 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 $\hat{C} \sqsubseteq \hat{D} \in \mathcal{T}$.
3. If $\sigma \in \Sigma^{\mathcal{T}}$, for every axiom $\sqcap_i C_i \sqsubseteq \sqcup_j D_j \in \mathcal{T}$ such that $\{\langle \sigma, \mathcal{C}_i \rangle \mid C_i \in \mathsf{nc}(\hat{C})\} \subseteq \mathcal{I}^{\mathcal{T}}_{-} \cup \mathcal{I}^{\mathcal{T}}_{+}$, then
$$\{\ \langle \sigma, D_j \rangle \mid D_j \in \mathsf{nc}(\hat{D})\ \} \ \subseteq\ \mathcal{I}^{\mathcal{T}}_{+}$$
and
$$(\bigwedge_i L_{\langle \sigma,\ C_i \rangle}) \rightarrow (\bigvee_j L_{\langle \sigma,\ D_j \rangle}) \ \in\ \varphi^{\mathcal{T}}. \tag{6}$$
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. $\langle \sigma, \leq n-1r.C \rangle \in \mathcal{I}^{\mathcal{T}}_{-}$, or $\langle \sigma, \forall r.\neg C \rangle \in \mathcal{I}^{\mathcal{T}}_{-}$ assuming $n = 1$], 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
$$\{\ \mathsf{IC}(\mathsf{indiv}^C_{\sigma.r}, 1, k^C_i) \rightarrow L_{\langle \sigma.r.k^C_i,\ C \rangle} \mid i = 1, \ldots, n\ \} \ \subset\ \varphi^{\mathcal{T}}, \tag{7}$$
$$\{\ \mathsf{IC}(\mathsf{indiv}^C_{\sigma.r}, 1, k^C_i) \rightarrow A_{\langle \sigma.r.k^C_i,\ \top \rangle} \mid i = 1, \ldots, n\ \} \ \subset\ \varphi^{\mathcal{T}}, \tag{8}$$
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. $\langle \sigma, \leq n'-1r.D \rangle \in \mathcal{I}^{\mathcal{T}}_{-}$ ...], with $C \neq D$ and $i = 1, \ldots, n$, $j = 1, \ldots, n'$. We assume only consecutive values for the individuals $\sigma.r.j$, 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. $\langle \sigma, \leq n'-1r.D \rangle \in \mathcal{I}^{\mathcal{T}}_{-}$ ...].

[5] See Section 2.1.1.

[6] 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.

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

$$((A_{\langle \sigma,\ \geq nr.C \rangle} \wedge A_{\langle \sigma,\ \top \rangle}) \rightarrow \neg \mathsf{BC}(\mathsf{indiv}_{\sigma.r}^{C}, n-1)) \ \in \ \varphi^{\mathcal{T}}, \tag{9}$$

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

$$((\neg \mathsf{BC}(\mathsf{indiv}_{\sigma.r}^{C}, n-1) \wedge A_{\langle \sigma,\ \top \rangle}) \rightarrow A_{\langle \sigma,\ \geq nr.C \rangle}) \ \in \ \varphi^{\mathcal{T}}. \tag{10}$$

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

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

and

$$\begin{aligned} &\{ \ \mathsf{IC}(\mathsf{indiv}_{\sigma.r}^{D}, 1, k_i^C) \rightarrow L_{\langle \sigma.r.k_i^C,\ D \rangle} \mid i = 1, \ldots, n \ \} \ \cup \\ &\{ \ \mathsf{IC}(\mathsf{indiv}_{\sigma.r}^{C}, 1, k_i^D) \rightarrow L_{\langle \sigma.r.k_i^D,\ C \rangle} \mid i = 1, \ldots, n' \ \} \ \subset \ \varphi^{\mathcal{T}}, \end{aligned} \tag{11}$$

$$\begin{aligned} &\{ \ \mathsf{IC}(\mathsf{indiv}_{\sigma.r}^{D}, 1, k_i^C) \rightarrow A_{\langle \sigma.r.k_i^C,\ \top \rangle} \mid i = 1, \ldots, n \ \} \ \cup \\ &\{ \ \mathsf{IC}(\mathsf{indiv}_{\sigma.r}^{C}, 1, k_i^D) \rightarrow A_{\langle \sigma.r.k_i^D,\ \top \rangle} \mid i = 1, \ldots, n' \ \} \ \subset \ \varphi^{\mathcal{T}}. \end{aligned} \tag{12}$$

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

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

and

$$\{ \ (L_{\langle \sigma.r.j,\ C \rangle} \wedge A_{\langle \sigma.r.j,\ \top \rangle}) \rightarrow \mathsf{IC}(\mathsf{indiv}_{\sigma.r}^{C}, 1, j) \mid \sigma.r.j \in \Sigma^{\mathcal{T}} \ \} \ \subset \ \varphi^{\mathcal{T}}. \tag{13}$$

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

$$((A_{\langle \sigma,\ \leq mr.C \rangle} \wedge A_{\langle \sigma,\ \top \rangle}) \rightarrow \mathsf{BC}(\mathsf{indiv}_{\sigma.r}^{C}, m)) \ \in \ \varphi^{\mathcal{T}}, \tag{14}$$

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

$$((\mathsf{BC}(\mathsf{indiv}_{\sigma.r}^{C}, m) \wedge A_{\langle \sigma,\ \top \rangle}) \rightarrow A_{\langle \sigma,\ \leq mr.C \rangle}) \ \in \ \varphi^{\mathcal{T}}. \tag{15}$$

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

$$\{ \ ((A_{\langle \sigma,\ \forall r.C \rangle} \wedge A_{\langle \sigma.r.j,\ \top \rangle}) \rightarrow L_{\langle \sigma.r.j,\ C \rangle}) \mid \sigma.r.j \in \Sigma^{\mathcal{T}} \ \} \ \subset \ \varphi^{\mathcal{T}}, \tag{16}$$

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

$$((\mathsf{BC}(\mathsf{indiv}_{\sigma.r}^{\neg C}, 0) \wedge A_{\langle \sigma,\ \top \rangle}) \rightarrow A_{\langle \sigma,\ \forall r.C \rangle}) \ \in \ \varphi^{\mathcal{T}}. \tag{17}$$

$\diamond$

Importantly, $\mathcal{ALCQ}2SMT_{\mathcal{C}}$ as defined in Definition 3, allow to solve the TBox consistency and concept satisfiability problems via encoding into SMT($\mathcal{C}$), as stated by the following results. In order to not break the flow of the exposition, the proof have been moved to Appendix A,

Theorem 1. *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 3) is satisfiable.*

Theorem 2. *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 3, a normal concept $\hat{C}$, such that $\hat{C} \in \mathcal{T}$, 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.*

We remark some facts on the above exposed encoding of Definition 3:

- 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 due to the Theory of Costs. In fact in the Theory of Costs the final value of a cost variable is determined by the sum of its enabled (i.e., assigned to true) incur costs. Thus, if no incur costs are defined for a given cost variable, then the final cost for such a variable is due to be zero. Henceforth, while the clauses at point 5. are introduced in order to allow for satisfying implied at-least restrictions, they are also necessary in order to allow for "unsatisfying" the occurring implicant at-most restrictions. In other words only introducing some individuals and incur costs a left-hand side at-most restriction is guaranteed to be potentially falsified, avoiding some axioms to be applied (while applying the same axioms could lead to unsatisfiability). In contrast, the clauses at point 8. work in the opposite way: the are introduced to allow detecting the unsatisfiability of right-hand side at-most restrictions and, vice versa, to potentially force the application of the axioms of $\mathcal{T}$ 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 (where $\forall r.C$ is equivalent to $\leq 0 r.\neg C$).

- Point 4. is necessary to force the encoding of axioms having on the left-hand side restrictions wrt. the role r, when other restrictions wrt. r are involved. Such kind of axioms can easily create cycles in TBoxes, thus we remark that our encoding ensures termination only for acyclic TBoxes. Under this hypothesis it is not necessary to encode such axioms in other circumstances, since axioms are already encoded and checked for mutual inconsistency wrt. the root individual 1.

- In the clauses of type (7), (8), (11), (12) and (7), (13), every IC-literal has cost value 1 and has index equal to the index of the individual the incur cost refers to. This ensures that all the indexes of distinct IC-literals (incur costs) for the same cost variable $\mathsf{indiv}^{C}_{\sigma.r}$ differ each other, and easily allows for associating every instantiated concept always to the same (and only) incur cost.

- Notice that if, for the same σ, r and C, many $\langle \sigma, \geq nr.C \rangle \in \mathcal{I}^{\mathcal{T}}_{+}$ or $\langle \sigma, \leq nr.C \rangle \in \mathcal{I}^{\mathcal{T}}_{-}$ fall in the conditions of point 5. for different values of n, then, being n^* the highest value of n, only exactly n^* new individuals and n^* instances of clauses of type (7) and (8) are required to be in $\varphi^{\mathcal{T}}$. In contrast, one distinct clause of type (9) must be included in $\varphi^{\mathcal{T}}$ 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 (14) in the case of different values of m for the same σ, r and C).

- Point 7. is meaningful when $C \neq D$, in fact if $C = D$ then clauses (11) and (12) exactly correspond to clauses (7) and (8).

- By way of the Theory of Costs clauses (9) and (14), 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 (9) forces some IC-literals to be assigned to true (explaining the fact that the implications (7) and (11) work only in one direction), a clause of type (14) bounds the number of IC-literals that can be enabled (motivating the implications (13) and their opposite direction).

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.

3.3 An Encoding Algorithm

Here we sketch an algorithm for building the encoding defined in the previous section.

We follow Definition 3 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 3 by extending the set $\Sigma^{\mathcal{T}}$ with new individuals and by adding new clauses to the SMT($\mathcal{C}$) formula $\varphi^{z}\mathcal{T}$. The sets of the Boolean literals encoding instantiated concepts 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 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 different individual (in particular any "child" individual $\sigma.r.i$). Henceforth, the algorithm handles individuals in a BFS manner, and in more details 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 $\varphi^{\mathcal{T}}$ is extended encoding all the TBox axioms in 1 with clauses of type (6), as consequence of the points 2. and 3. of the definition of $\mathcal{ALCQ}2SMT_{\mathcal{C}}(\mathcal{T})$.

Iteration For every individual σ of $\Sigma^{\mathcal{T}}$ extracted from Q, five expansions phases are applied following Definition 3:

(a) This phase realizes the points 3. and 4. of the definition. In the first phase pure propositional clauses are added to $\mathcal{ALCQ}2SMT_{\mathcal{C}}(\mathcal{T})$, exhaustively encoding all the implications of type (6) for σ due to 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 axioms are fully matched in σ (as stated by the conditions of point 3. for simple axioms and by the conditions of point 4. for axioms involving left-hand side qualified number restrictions).

Then, (qualified number/universal) restrictions are partitioned wrt. the role r they refer to, and for every partition of restrictions the next four phases are applied:

(b) This phase realizes the points 5. and 6. [resp. 9.] of the definition. 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 grouped and sorted in decreasing order wrt. their numerical value n_i. This has been done, new individuals and clauses of type (7) and (8) (point 5.) are introduced once for every different concept C only for the restriction with the highest value of n_i. Instead, one different clause of type (9) [resp. (15) or (17)] is added for every different C and every different kind of restriction or value of n_i.

(c) If, wrt. σ and r, more than one at-least restriction coexist with some at-most restrictions, then a further encoding phase is necessary in order to allowing for sharing the newly introduced individuals $\sigma.r.i$, as provided by point 7. of Definition 3. Thus, all the clauses of type (11) and (12) are introduced for every at-least restriction $\langle\sigma, \geq n_j r.C_j\rangle$ and every individual $\sigma.r.i$.

(d) This phase handles at-most restrictions and realizes the points 8. and 9. [resp. 6.] of the definition. 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 (13) (point 8.) are introduced only once for every different concept C, while one clause of type (14) [resp. (10)] is added for every different C and every different kind of restriction or value of m_i.

(e) At last, universal restrictions are handled realizing the point 10. of the definition. Given r, for each restriction $\langle\sigma, \forall r.C\rangle$ and every individual $\sigma.r.i$, the algorithm introduces one clause of type (16).

We avoid to show a pseudocode for the exposed algorithm, because it would be very long without adding any useful information wrt. Definition 2 and the textual description above.

Proposition 3. *Given a normal-form, acyclic $\mathcal{ALCQ}$ TBox $\mathcal{T}$, 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 $\mathcal{T}$ (when the premises of the axioms are already included in the encoding) or (b)-(e) they encode restrictions introducing new individuals and concept names in those individuals. Termination is guaranteed due to the following:

- every axiom of $\mathcal{T}$ is expanded at most once for every σ encountered during the algorithm;
- 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 only to concept names;

- the TBox $\mathcal{T}$ is assumed to be acyclic, thus a concept name can not recur cyclically during the encoding.

□

Since $\mathcal{ALCQ}$ has the finite tree model property [22]. 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 the number of expansion of the first kind performed by our encoding algorithm is linear in the size of $\mathcal{T}$ for every different σ, 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 numerical values of the encoded qualified number restrictions. Overall the size of $\varphi^{\mathcal{T}}$ is bounded by the product of the sum of the values in the qualified number restrictions of $\mathcal{T}$ with the size of a model for $\mathcal{T}$ (that is worst-case exponential in the size of $\mathcal{T}$). It is easy to see that the complexity of our encoding algorithm is polynomial in the size of the output formula $\varphi^{\mathcal{T}}$ since every encoding phase can be trivially performed in polynomial time.

4 Smart Individuals Partitioning

4.1 The Need of Partitioning

A main drawback of the basic $\mathcal{ALCQ}2SMT_{\mathcal{C}}$ encoding is that for every encoded σ, it introduces exactly as much new individuals as is the sum of the values included in all the at-least restrictions instantiated in σ. Thus, even in presence of a small number of restrictions but which include large numerical values, the total number of individuals introduced by the encoding can be huge. Furthermore, the effect of this drawback is significantly increased because it is replicated at many levels in the encoding (i.e. these individuals might be further expanded leading to introduction of other numerous individuals), and, more importantly, because of the sharing of the individuals (when necessary). This latter fact leads to the introduction of an even larger number of clauses and propositional variables, which significantly increases the hardness of reasoning on the resulting SMT($\mathcal{C}$) formula.

However, it is quite intuitive that it is not strictly necessary to introduce in the $\mathcal{ALCQ}2SMT_{\mathcal{C}}$ encoding exactly as many individuals as many they are in the interpretation satisfying the formula or, even worst, in all the values included in the qualified number restrictions. In general, it is possible to divide the individuals in groups which have identical properties (belongs to the same concepts' interpretations), i.e. to compute a partitioning of the individuals, and then use only one single "proxy individual" (one single label in the encoding) as representative for all the individuals of one specific partition. So called proxy individual has been used, e.g., in [8, 9], each representing one of the exponentially-many intersections of the concepts (interpretations) having a restriction in common wrt. the same role. Independently from the cardinality of the intersection, the proxy individual is "the witness" of the consistency/inconsistency of the intersection.

As stated above, partitions must be made of individuals with identical properties. Looking at Definition 3 we point out the following facts:

- Except for the root individual 1, new individuals $\sigma.r.i$ are introduced (like in a tree model) 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. σ and wrt. the role r they refer to.

- If, for the given σ and r, only one single at-least restriction $\langle \sigma, \geq nr.C \rangle$ exists, then all the individuals $\sigma.r.k_i^C$, with $i = 1, \ldots, n$, have identical properties and can form one single partition, being represented by only one proxy individual. The same consideration holds for many at-least restrictions if no at-most restriction exists. Every group of individuals referring to a distinct at-least restriction can represent a distinct partition, cause the sharing of individuals is not necessary.

- If, for the same σ and r, at least one at-most restriction and many different at-least restrictions $\langle \sigma, \geq n_j r.C_j \rangle$ coexist exactly N different individuals $\sigma.r.i$, with $i = 1, \ldots, N$ and $N = \sum_j n_j$, are introduced and shared among the interpretations of all the concepts C_j. Thus, in this scenario, a partitioning of the N individuals introduced should be able to represent all the possible intersections between the different concept interpretations.

However, in the latter case, it is not important to consider all the possible cardinalities of these intersections. Instead, it is sufficient to distinguish between an empty intersection and some other "limit" cases, where the cardinality of the intersections in these limit cases depends on the specific values of the qualified number restrictions involved. Summarizing, partitions of individuals (and representative proxy individuals) can be used in our encoding in place of many more single individuals, where one different partitioning of the individuals is computed for every group of qualified number restrictions referring to the same σ and the same role r, by taking into account the specific values included in the qualified number restrictions of the group.

Example 4.1. 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 $\Delta^{\mathcal{I}}$. Notice that, for the sample restrictions, many other possible combinations of the 1010 introduced individuals and many possible satisfying/unsatisfying interpretations would be possible in theory, but these "limit" combinations are enough to represent the significant cases concerning satisfiability.

4.2 Proxy Individuals and Smart Partitioning

In order to handle partitions of individuals we extend the $\mathcal{ALCQ}2SMT_{\mathcal{C}}$ formalism by 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 $i \leq j$, thus the label represent a set of individuals whose cardinality is $j - i + 1$. A normal label is a special case of a cumulative label, with $i = j$ and, thus, 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 all the c distinct individuals: $\sigma.r.i{+}1, \ldots, \sigma.r.i{+}c$, having the same characteristics, by mean of only one cumulative label $\sigma.r.(i{+}1 \rightarrow i{+}c)$ representing all of them. 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 individuals. In particular, we can consider every individual like a proxy individual where a normal individual is proxy only of itself.

This has been said, the idea is to compute *smart* partitioning of the individuals that must be encoded in $\mathcal{ALCQ}2SMT_{\mathcal{C}}$ and, in such a way, to use a smaller set of proxy individuals (one for every group of individuals in the partitioning) in place of the larger set of all the original single possible individuals. 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 the 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.

Now we formally define a possible smart partitioning for the individuals introduced in the $\mathcal{ALCQ}2SMT_{\mathcal{C}}$ encoding.

Definition 4. Let $\mathcal{T}$ being an acyclic $\mathcal{ALCQ}$ TBox in normal form and $\mathcal{ALCQ}2SMT_{\mathcal{C}}(\mathcal{T})$ be the SMT($\mathcal{C}$) encoding for $\mathcal{T}$ defined in Definition 3. Given the individual $\sigma \in \Sigma^{\mathcal{T}}$ and the role r of $\mathcal{T}$ we define the arrays: [7]

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

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

[7] Equal n_i or m_j values can repeat as many time as they occur.

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:

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

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

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

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 $p_{|\mathcal{P}_{\sigma.r}|} = \max\{N^{\geq}_{\sigma.r}, N^{\leq}_{\sigma.r}\}$.

Concerning Definition 4 notice the following facts. Given σ and r, $N^{\geq}_{\sigma.r}$ represents the number of individuals of the form $\sigma.r.i$ 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 the last 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 $\mathcal{P}^{\geq}_{\sigma.r}, \mathcal{P}^{\leq}_{\sigma.r}, \mathcal{P}_{\sigma.r}$ 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 4 by the fact that $\emptyset, X \in 2^X$, for any set X. The two partitioning shown in Example 4.1 are computed in accordance with Definition 4.

Definition 4 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.
- The union of $\mathcal{P}^{\geq}_{\sigma.r}$ with $\mathcal{P}^{\leq}_{\sigma.r}$ 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 cardinality of the non-empty intersections are possible with the 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 (for some qualified number restrictions) to another limit case (for some other restrictions).

4.3 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, if we assume that they are all part of the same partition, then it is possible to substitute n clauses referring to the distinct individuals $\sigma.r.k^C_1, \ldots, \sigma.r.k^C_n$, but identical to each other in structure, with one single clause referring to the proxy individual $\sigma.r.(k^C_i \rightarrow k^C_n)$. Moreover, if each of the original clauses produce an 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 will produce an incur cost of n (e.g., including the literal $\mathsf{IC}(\mathsf{indiv}^C_{\sigma.r}, n, k^C_1)$)

Concretely, we can enhance Definition 3 as follows by taking advantage of the partitioning technique defined in Definition 4. First of all the set $\Sigma^{\mathcal{T}}$, and the instantiated concepts included in $\mathcal{I}^{\mathcal{T}}_{-}$ and $\mathcal{I}^{\mathcal{T}}_{-}$ must be assumed, generically, to be made 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. Without

[8] The instantiated concepts $\langle \sigma, \forall r.C_i \rangle \in \mathcal{I}^{\mathcal{T}}_{-}$ must be considered like $\langle \sigma, \leq 0 r.\neg C_i \rangle \in \mathcal{I}^{\mathcal{T}}_{-}$.

going into too much details, and repeat the whole Definition 3 we point out only the necessary differences, assuming that, for any σ and r, the partitioning $\mathcal{P}_{\sigma.r}$ is available.

Second, the n clauses of the types (7) and (8) at point 5. are replaced by the following:

$$\{\ \mathsf{IC}(\mathsf{indiv}^C_{\sigma.r},\ cost_j,\ idx_j) \rightarrow L_{\langle \sigma_{proxy_j},\ C\rangle} \mid p_j \in \mathcal{P}_{\sigma.r},\ 0 < p_j \leq n\ \} \subset \varphi^{\mathcal{T}}, \tag{18}$$

$$\{\ \mathsf{IC}(\mathsf{indiv}^C_{\sigma.r},\ cost_j,\ idx_j) \rightarrow A_{\langle \sigma_{proxy_j},\ \top\rangle} \mid p_j \in \mathcal{P}_{\sigma.r},\ 0 < p_j \leq n\ \} \subset \varphi^{\mathcal{T}}, \tag{19}$$

$$cost_j = p_j - p_{(j-1)},\quad idx_j = k^C_1 + p_{(j-1)},\quad \sigma_{proxy_j} = \sigma.r.k^C_1 + p_{(j-1)} \rightarrow k^C_1 + p_j - 1.$$

Thus the value of every incur cost is given by the number of individuals included in each partition, i.e. included in between the indexes p_{j-1} and p_j. Notice that since $\mathcal{P}_{\sigma.r}$ includes all the possible sums among all the other possible restrictions' values and n, then $n, k^C_1 - 1, k^C_1 + n - 1 \in \mathcal{P}_{\sigma.r}$ (i.e., if partitioned, n is exactly partitioned). 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 $k^C_1 \geq 1$ represents the index of the first individual introduced for C. Clauses (11), (12) at point 7. are modified accordingly.

Third, the clauses of type (13) defined at point 8. must take into account proxy individuals and the relative incur cost, potentially bigger than 1. Hence those clauses are replaced by the following ones:

$$\{\ (L_{\langle \sigma.r.(i \rightarrow j),\ C\rangle} \wedge A_{\langle \sigma.r.(i \rightarrow j),\ \top\rangle}) \rightarrow \mathsf{IC}(\mathsf{indiv}^C_{\sigma.r},\ j{-}i{+}1,\ i) \mid \sigma.r.(i{\rightarrow}j) \in \Sigma^{\mathcal{T}}\ \}. \tag{20}$$

Clauses (16) 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 3 trivially come by consequence.

We make the following observations on what here above stated:

- If, for the given σ and r, the conditions of point 7. of Definition 3 do not hold (e.g. no at-most restriction exists), then, with an even more efficient partitioning, we can require only the following two clauses:

$$\mathsf{IC}(\mathsf{indiv}^C_{\sigma.r}, n, k^C_1) \rightarrow L_{\langle \sigma.r.(k^C_1 \rightarrow k^C_1 + n - 1),\ C\rangle},$$
$$\mathsf{IC}(\mathsf{indiv}^C_{\sigma.r}, n, k^C_1) \rightarrow A_{\langle \sigma.r.(k^C_1 \rightarrow k^C_1 + n - 1),\ \top\rangle}$$

  to be part of $\varphi^{\mathcal{T}}$, for every $\langle \sigma, {\geq}nr.C\rangle$.

- Otherwise, if the conditions of point 7. hold, then $\varphi^{\mathcal{T}}$ contains all the clauses:

$$\{\ \mathsf{IC}(\mathsf{indiv}^C_{\sigma.r},\ p_j - p_{j-1},\ p_{j-1} + 1) \rightarrow L_{\langle \sigma.r.(p_{j-1}+1 \rightarrow p_j),\ C\rangle} \mid p_j \in \mathcal{P}_{\sigma.r},\ j > 1\ \} \cup$$
$$\{\ \mathsf{IC}(\mathsf{indiv}^C_{\sigma.r},\ p_j - p_{j-1},\ p_{j-1} + 1) \rightarrow A_{\langle \sigma.r.(p_{j-1}+1 \rightarrow p_j),\ \top\rangle} \mid p_j \in \mathcal{P}_{\sigma.r},\ j > 1\ \}$$

  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 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.

4.4 Partitioning Algorithm

G iven the individual σ, the role r and the respective arrays $\mathcal{N}^{\geq}_{\sigma.r}$ and $\mathcal{N}^{\leq}_{\sigma.r}$, in Figure 2 we expose the pseudocode of the algorithm computing the smart partitioning of Definition 4. In particular, instead of computing $\mathcal{P}_{\sigma.r}$, in the pseudocode of Figure 2 we compute the array $\mathcal{D}_{\sigma.r}$ of the partitions sizes represented by $\mathcal{P}_{\sigma.r}$, which are the values in which we are interested (the j-th element of $\mathcal{D}_{\sigma.r}$ represents $p_j - p_{j-1}$).

Proposition 4. *Given the individual σ and the role r, the algorithm of Figure 2, 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 4, 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 2: Exponential-time algorithm computing smart partitioning.

Proof. We analyze the complexity of the algorithm of Figure 2. Let n_i be the number of the list N 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 is, in the worst case, the number of the different possible combinations of k previously handled numbers $n_1, \ldots, n_{i-1}$, with k from 0 to $i-1$. 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 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 more independent from the values of the qualified number restrictions in the TBox. With smart partitioning the magnitude of the values doesn't effect the size/hardness of the

encoding, which, instead, is effected by the interactions among the values (e.g., the frequency of the values or the differences among them matter).

5 Empirical Evaluation

In order to verify empirically the effectiveness of our novel approach, we have performed a preliminary empirical test session on about 600 synthesized and parametrized $\mathcal{ALCQ}$ problems, on which we solved concept satisfiability wrt. a non-empty TBox.

We have implemented the encoder called $\mathcal{ALCQ}$2SMT in C++, in which, the *smart partitioning* technique of Section 4 can be optionally enabled. In the following, when exposing the results of our evaluation, 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`) [3], that actually is the first SMT-solver including the Theory of Costs [4].

We have downloaded the available versions of state-of-the-art tools FaCT++ (version `v1.4.0`) [35], Pellet (version `2.1.1`) [34], and Racer (version `1-9-0`) [15, 16] in order to compare their performance wrt. those of our tool on every presented test case.

All the tests presented in this section have been performed on a biprocessor dual-core `Intel Xeon 2.66 GHz` 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.

5.1 Test Descriptions

In this section we present the sets of test cases we chose for our evaluation.

As discussed in [11] 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 restriction in Description Logic 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 recent standard OWL 2 [25].

Thus, in this preliminary analysis, we chose to follow the same benchmarking approach of [11] and rely on synthesized test cases to empirically evaluate the performance of our novel approach under different perspectives. Therefore, we have adapted to $\mathcal{ALCQ}$ the $\mathcal{SHQ}$ problems from [11]. 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 expression.

In the following we describe with more details the six groups of different test cases defined. Wrt. to [11], we add a further group of problems with 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 affect on one of the above mentioned complexity sources, for instance by determining the number of qualified number restrictions, and so on and so forth. 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 on the test cases, we further parametrized the test cases of [11], adding in many cases the parameter n which varies those number when they are not directly related to i. When listing the chosen values for n we will underline the value originally used in[11].

This has been said, in our evaluation we test the satisfiability of the concept C wrt. the following groups of TBoxes:

Increasing Values of Numbers Occurring in Restrictions.
First we analyze the effect of having increasingly high values occurring in the qualified number restrictions. We define 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, and where the values included 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 successor 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 it is unsatisfiable.

We call $\texttt{increasing\_lin\_sat}_i$ and $\texttt{increasing\_lin\_unsat}_i$ the satisfiable and, respectively, unsatisfiable version of this problem, where i represents the index (and hardness) of the problem and 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, respectively, unsatisfiable version 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 that were adapted to DL-reasoning as dependency-directed backtracking [20]. 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 shown [20] to significantly improved the performance of DL systems in dealing qualified number restrictions.

This test suite test 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 TBoxes:

$$C \sqsubseteq \geq n\, r.D_1 \;\sqcap\; \cdots \;\sqcap\; \geq n\, r.D_i \;\sqcap\; \leq ni-1\, r.\top,$$
$$D_j \sqcap D_k \sqsubseteq \bot, \quad 1 \leq j < k \leq i.$$

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 and at most $ni-1$ successor are allowed, we can conclude that 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 of restrictions and therefore in a larger number of variables. We call $\texttt{backtracking}_\texttt{i}(n)$ these problems, where i ranges in the interval $i = 1, 2, 3, \ldots, 20$ and where n ($n = 1, 2, \underline{3}, 10$) regulates the combined effect of the values occurring in number restrictions.

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 we compare the performance of reasoning on problems which ranges form satisfiable to unsatisfiable ones, 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. [9]

We call these problems $\texttt{sat\_unsat}_i(n)$, for which we chose the values $i = 1, 2, 4, 6, \ldots, 24$ and $n = 1, \underline{10}$.

Increasing Number of Qualified Number Restrictions.
The number of qualified number restriction occurring in the problems is one of the factors which mostly influences 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 a very extreme case, where the qualified number restrictions include very high values ($n = 50$).

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 4.4). the effectiveness of smart partitioning increases when included into the 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 $i = 1, 2 \ldots, n$. [10] Notice that C is still satisfiable.

[9] In [11] a second variant of this problem is proposed. In this alternative version the concept name D replaces $\top$ and is conjoint in all the four at-least restrictions. This variant has been introduced in order to study an unexpected behavior of their hybrid approach (due to the integrated arithmetic reasoner) in the limit cases $i \leq 3n$. 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.

[10] 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, for a fixed total number of restrictions we evaluate the performance of the various systems wrt. such a ratio. 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 them the same values of [11], 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.

5.2 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 against those of the other tools above listed.

The results of these experiments are graphically summarized in Figures 3, 4, 5, 6, 7, and 8. In order to to make the plots clearly readable in the figures, we have chosen to maximize 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 different tools to solve the problem varying i. Plots referring the the same group of benchmarks are grouped in the same figure.

From the the exposed results we notice a few facts:

- $\mathcal{ALCQ}$2SMT+MathSat with enabled smart partitioning (shortly $\mathcal{ALCQ}$2SMT+MathSat `S.P.`) results one of the best performer in all the test cases but in the artful `backtracking` problems (Figure 5) and in the `var_restr_num` problems (Figure 7) 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 7), while, either together with Racer or together with FaCT++, it is the best performing tool in:
  - all the `increasing` test sets (Figures 3 and 4) (both satisfiable/unsatisfiable and linearly/exponentially increasing);
  - all the `restr_ratio` and all the `sat_unsat` benchmarks (Figures 6 and 8 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 ones).
- 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. [11] FaCT++, instead, performs very well in general, except for unsatisfiable

[11] 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.

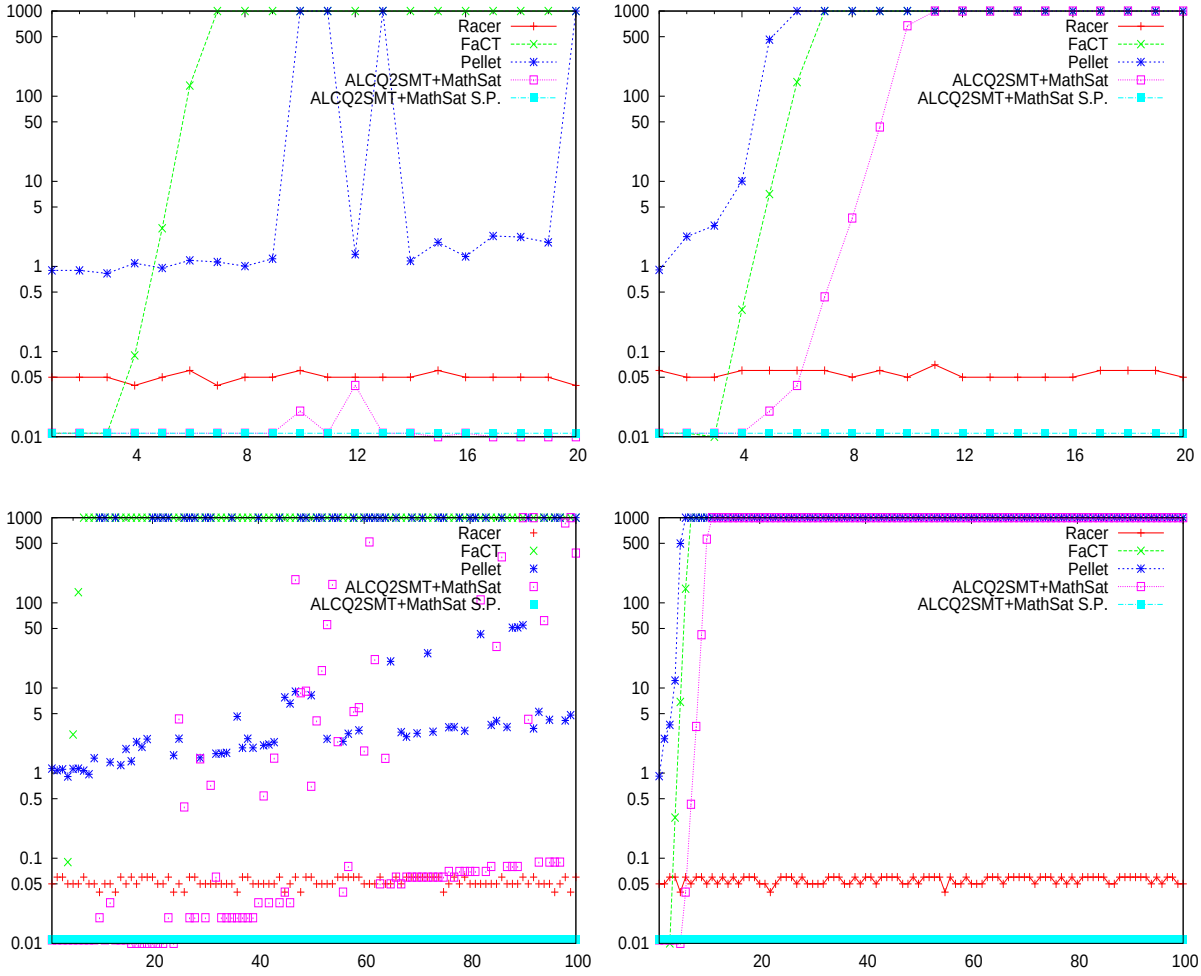

Figure 3: 1st column: $\texttt{increasing\_lin\_sat}_i$; 2nd column: $\texttt{increasing\_lin\_unsat}_i$. 1st row: 20 problems; 2nd row: 100 problems. X axis: test case index; Y axis: CPU time (sec).

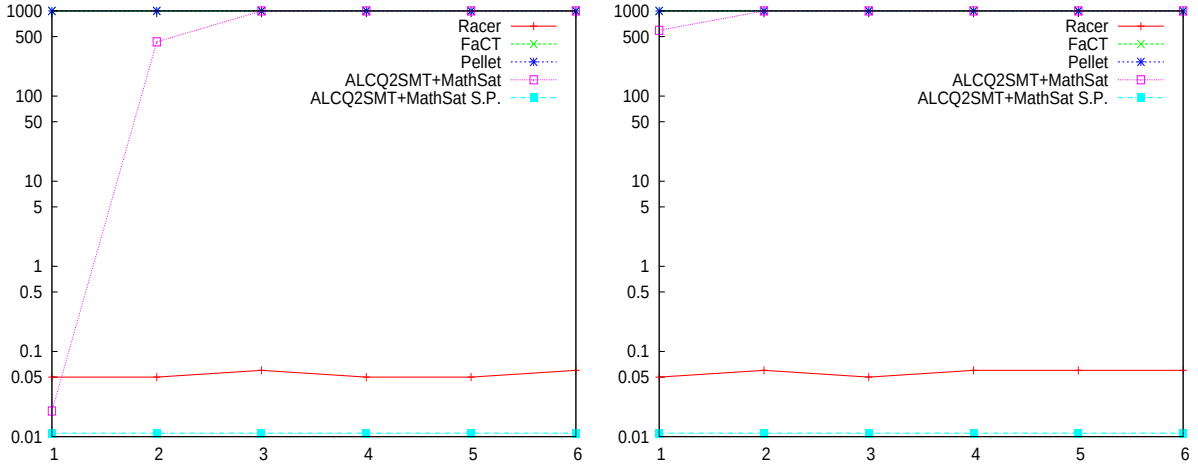

Figure 4: Left: $\texttt{increasing\_exp\_sat}_i$; right: $\texttt{increasing\_exp\_unsat}_i$. X axis: test case index; Y axis: CPU time (sec).

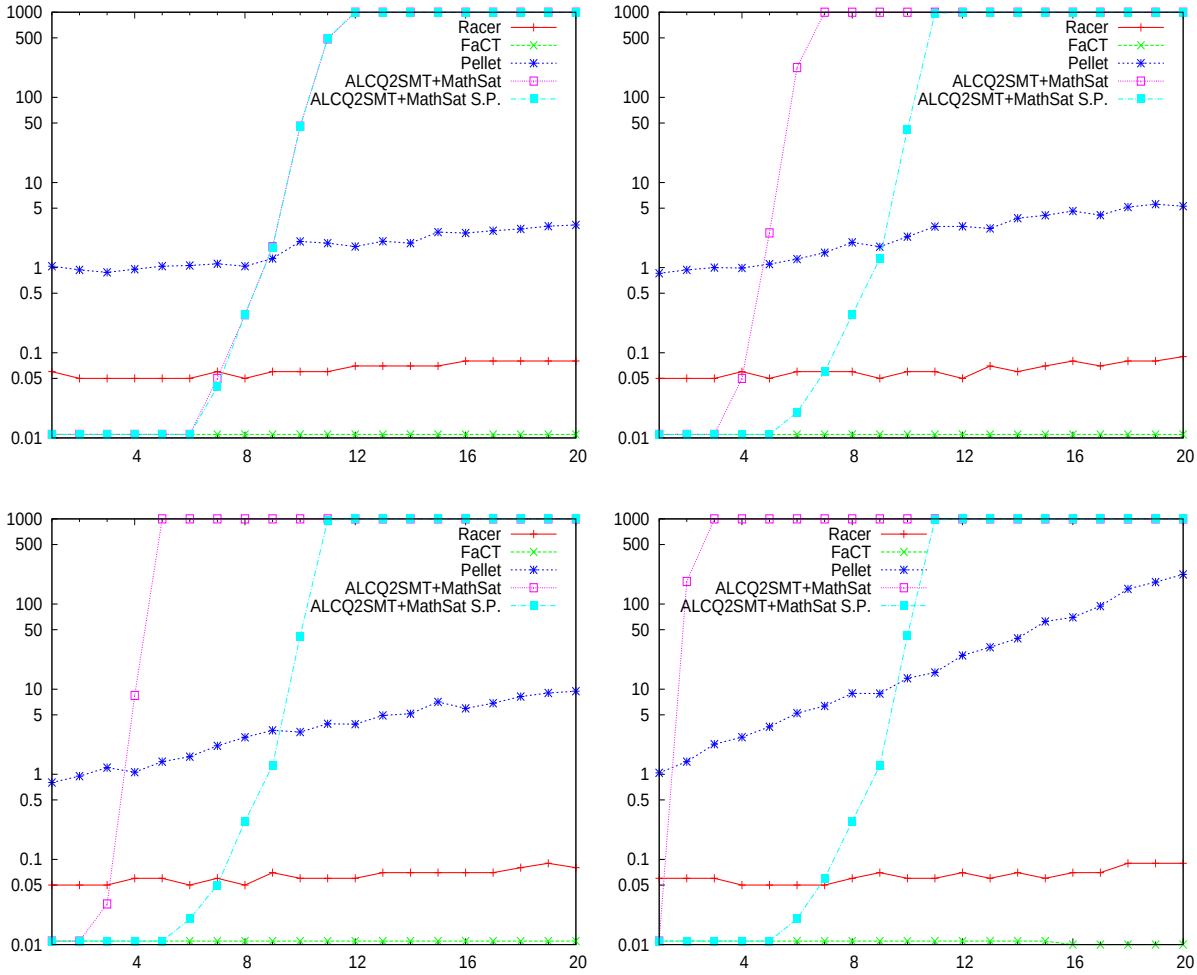

Figure 5: $\texttt{backtracking}_i(n)$. Top,left: $n = 1$; top,right: $n = 2$; bot,left: $n = 3$; bot,right: $n = 10$. X axis: test case index; Y axis: CPU time (sec).

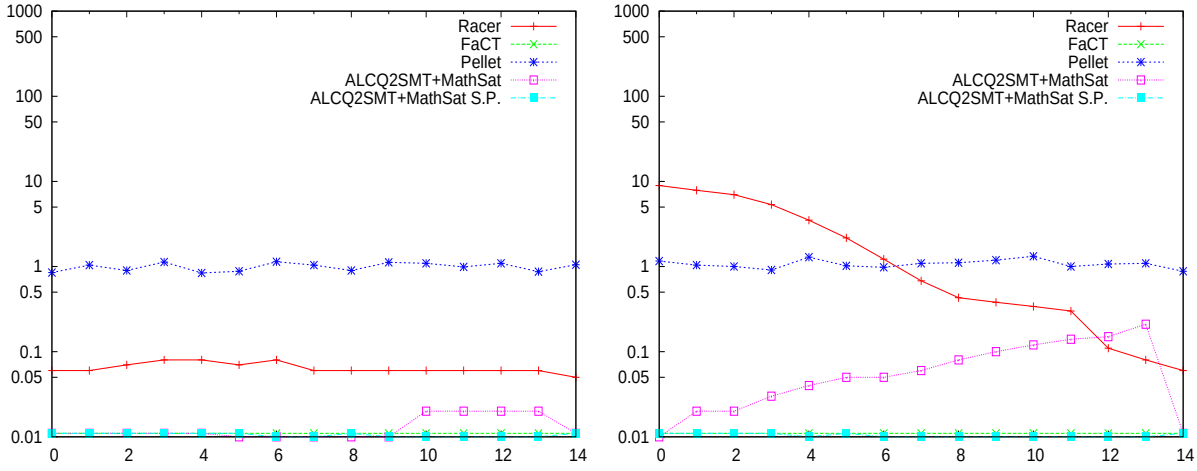

Figure 6: $\texttt{restr\_ratio}_i(n)$. Left: $n = 1$; right: $n = 5$. X axis: test case index; Y axis: CPU time (sec).

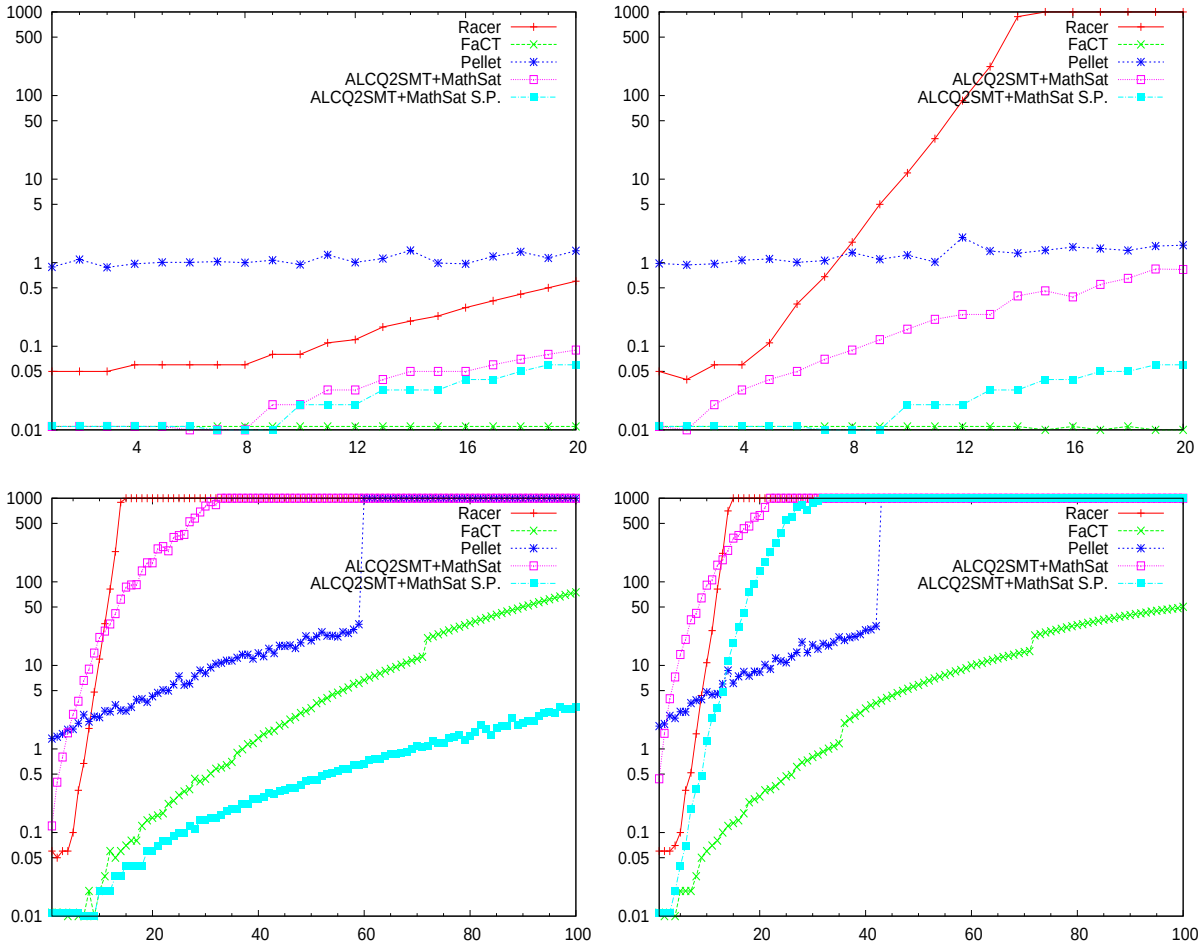

Figure 7: Top,left: $\texttt{restr\_num}_i(1)$; top,right: $\texttt{restr\_num}_i(5)$; bot,left: $\texttt{restr\_num}_i(50)$; bot,right: $\texttt{var\_restr\_num}_i(100)$. 1st row: 20 problems; 2nd row: 100 problems. X axis: test case index; Y axis: CPU time (sec).

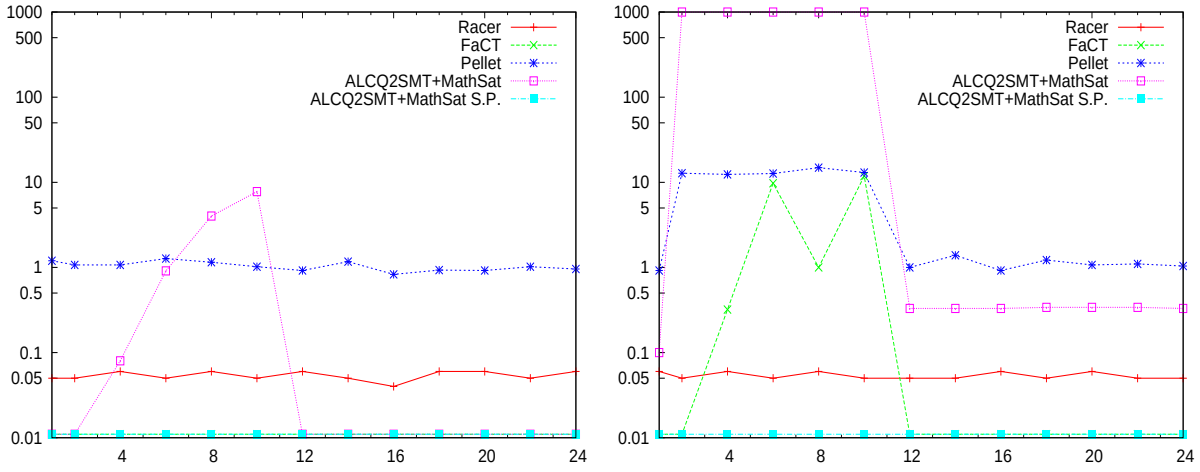

Figure 8: $\texttt{sat\_unsat}_i(n)$. Left: $n = 1$; right: $n = 10$. X axis: test case index; Y axis: CPU time (sec).

problems and problems including high values in 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 3) $\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 the ability of the SAT/SMT techniques in handling large-size problems shows effective. 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 problem 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 4 show even more clearly the evidenced effectiveness of smart partitioning.

- The $\texttt{backtracking}_i(n)$ benchmark problems (Figure 5) 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 for SMT is similar to that of the well-known Halpern & Moses branching formulas for modal logic $K_m/\mathcal{ALC}$ [18] for SAT when encoded following the approach of [31]. 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 $i = 11$ that is, thus, the intrinsic limit for MathSat.

- In the $\texttt{restr\_ratio}_i(n)$ test cases our tools is the best performer together with FaCT++ (Figure 6). 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 in them. Therefore, for our approach, the only source of complexity in the case of these problems is their 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 the sharing of the individuals and to the encoding of the at-most operator it self. The problem with index $i = m = 14$ which presents no at-most restriction is trivially satisfiable because no merge/sharing operations are neither encoded nor performed during the solving phase. 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.

- The $\texttt{restr\_num}_i(n)$ and $\texttt{var\_restr\_num}_i(n)$ benchmarks are likely the most challenging problems, overall, especially when combined with high values of the parameter n. From Figure 7, it is easy to see that in the $\texttt{restr\_num}_i(n)$ case the harder is the reasoning (due to the increase of the index i and of the parameter n) the more our variant $\mathcal{ALCQ}$2SMT+MATHSAT `S.P.` outperforms the other tools. This is almost due to the benefits given by smart partitioning, as can be deduced comparing the first tree plots of Figure 7 with the bottom-right one representing the $\texttt{var\_restr\_num}_i(100)$ test case (which in part inhibits the benefits of partitioning). Even if $\mathcal{ALCQ}$2SMT+MATHSAT `S.P.` is able to solve some problems more than basic $\mathcal{ALCQ}$2SMT+MATHSAT (respectively 31 against 21), also its CPU times quickly increase with i due to the variability of the values in number restrictions. 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 within the timeout only the first 21 and 42 problems, respectively, of the second benchmark. 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 7 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 3, 4 and 6 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 8) 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 and by mean of the influence of the Theory of Costs which forces upper bounds on the activated and countable individuals. As consequence MATHSAT must try and check for consistency wrt. the Theory of Costs many unsatisfiable assignments before a problem is discovered to be unsatisfiable. 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.

5.3 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 $\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 9 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 9, that in encoding the `increasing_exp_sat` problems $\mathcal{ALCQ}$2SMT takes linear CPU time wrt. the values occurring in the qualified number restrictions of every problem. 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 results absolutely indipendent from such values (i.e. the encoding time is unchanged for every problem's index) when 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 of the problem, while the $\mathcal{ALCQ}2SMT_{\mathcal{C}}$ encoding is almost identical (except for the upper-bound value fixed for the cost variables relative to the only at-most restriction included).

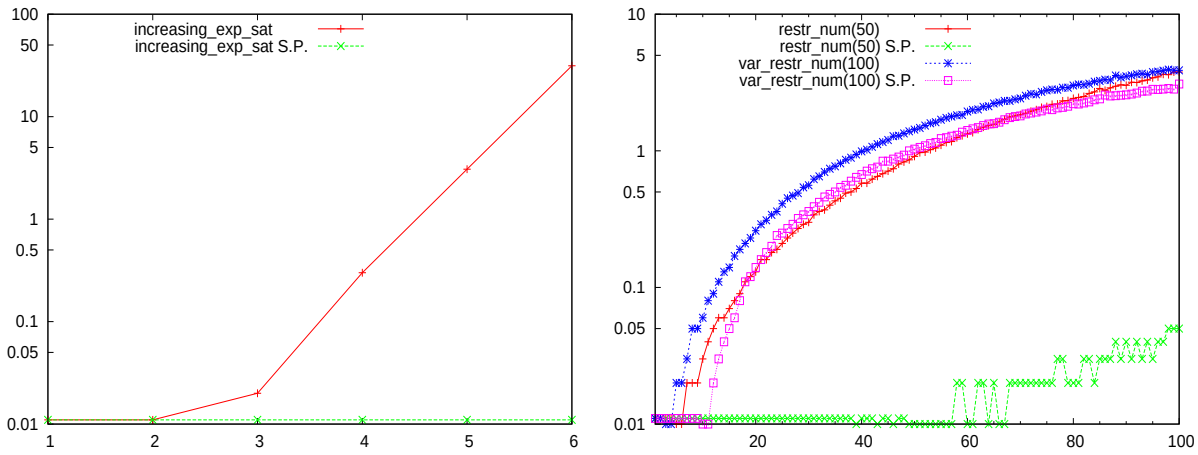

Figure 9: CPU times of $\mathcal{ALCQ}$2SMT. Left: $\texttt{increasing\_exp\_sat}_i$, $i = 1, \ldots, 6$; right: $\texttt{restr\_num}_i(50), \texttt{var\_restr\_num}_i(100)$, $i = 1, \ldots, 100$. X axis: test case index; Y axis: CPU time (sec).

From the right-side plot of Figure 9, instead, we observe the different effectiveness of smart partitioning on the similar in structure but numerically different benchmarks $\texttt{restr\_num}_i(50)$ and $\texttt{var\_restr\_num}_i(100)$. 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, even if in this last case the encoding time is almost unchanged by applying partitioning (because the variability in the values reduces the effectiveness of smart partitioning), 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 when the technique is not particularly effective. Notice, at last, that the time spent by $\mathcal{ALCQ}$2SMT never exceedes 4 seconds even if the number of restrictions in such problems can be huge, (the hardest problems with index i, further than having high numerical values –$n = 50, 100$– present more than 200 qualified number restrictions).

In Figures 10, 11, 12, 13, 14, and 15, instead we compare in plots the number of variables and clauses produced in output by $\mathcal{ALCQ}$2SMT in different test cases. 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, i.e. we have included the plots concerning particularly challenging benchmarks and some representative ones for every different kind of concept expressions. In fact, from the point of view of the number of variables and clauses, the `increasing_lin_sat` (Figure 10) and the `increasing_lin_unsat` problems presents exactly the same characteristics (the same arguments, of course, is valid for the exponentially increasing problems). In this case 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 11), 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 12 and 13 the configuration $n = 1$ for the `restr_ratio` and the `restr_num` benchmarks respectively. In fact, comparing with the higher values of n, these test cases show the same trend for the two variants of $\mathcal{ALCQ}$2SMT but with much less visible differences. For the sake of the reader's convenience, however, all the other plots are available in the Appendix B.

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

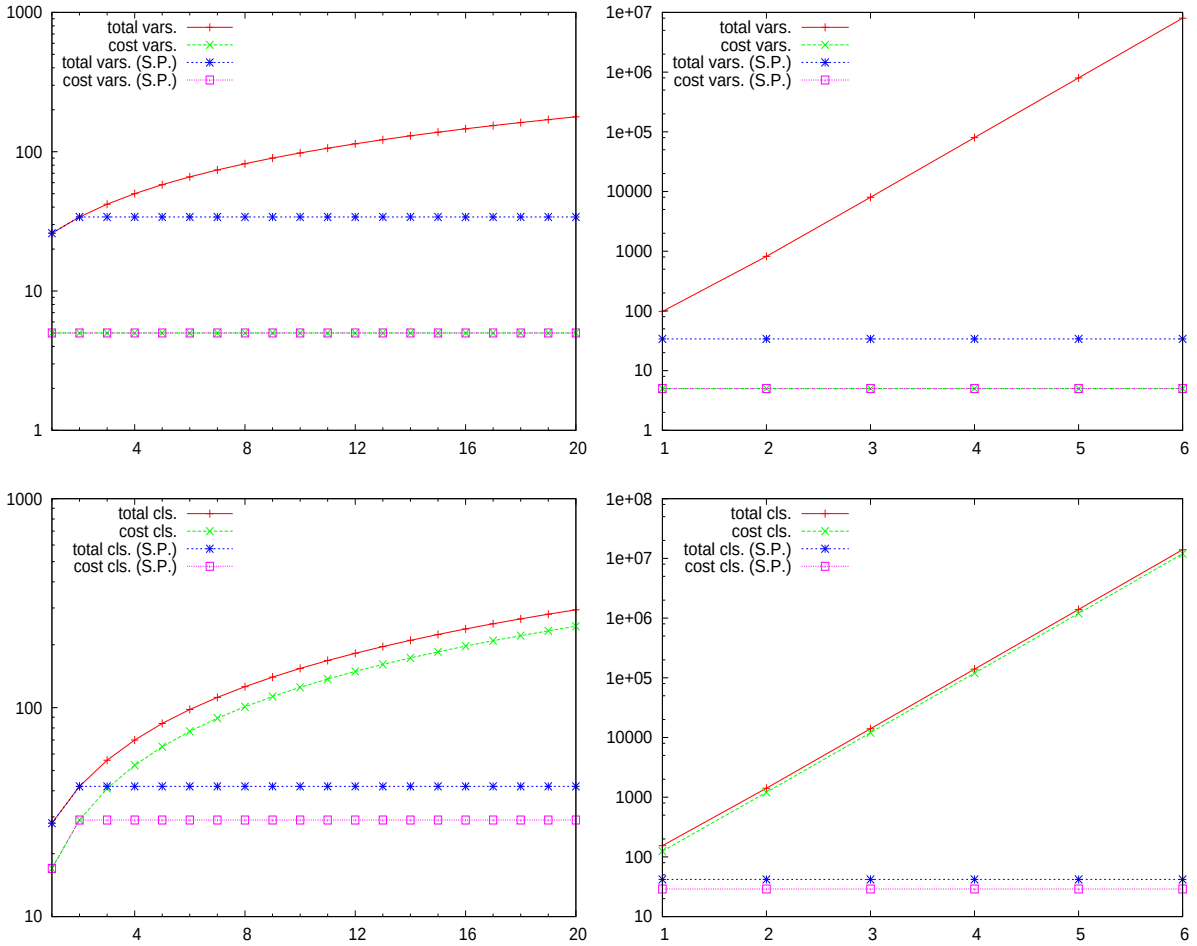

Figure 10: 1st column: `increasing_lin_sat`$_i$, $i = 1, \ldots, 20$; 2nd column: `increasing_exp_sat`$_i$, $i = 1, \ldots, 6$. 1st row: variables; 2nd row: clauses. X axis: test case index; Y axis: #variables/clauses.

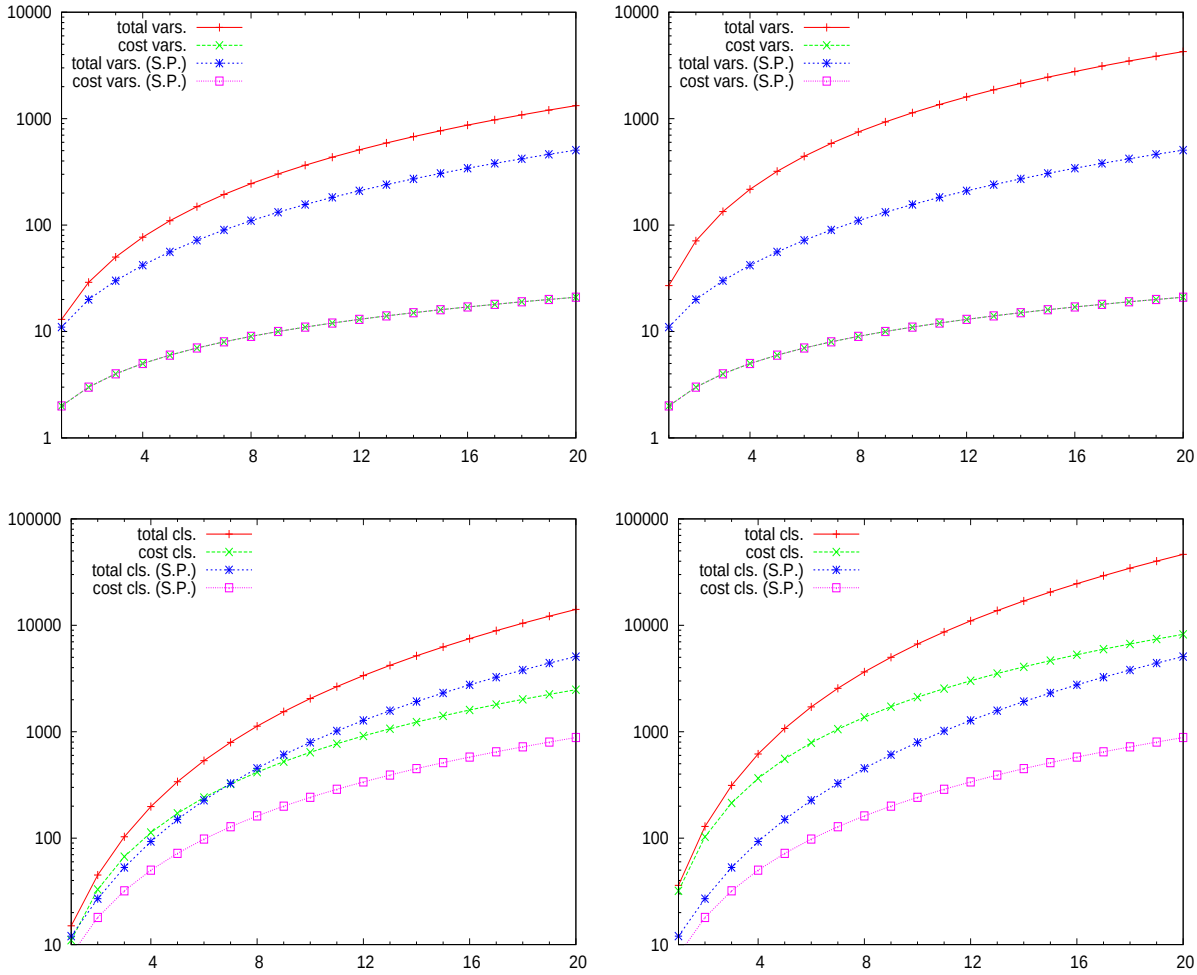

Figure 11: `backtracking`$_i(n)$. 1st column: $n = 3$; 2nd column: $n = 10$. 1st row: variables; 2nd row: clauses. X axis: test case index; Y axis: #variables/clauses.

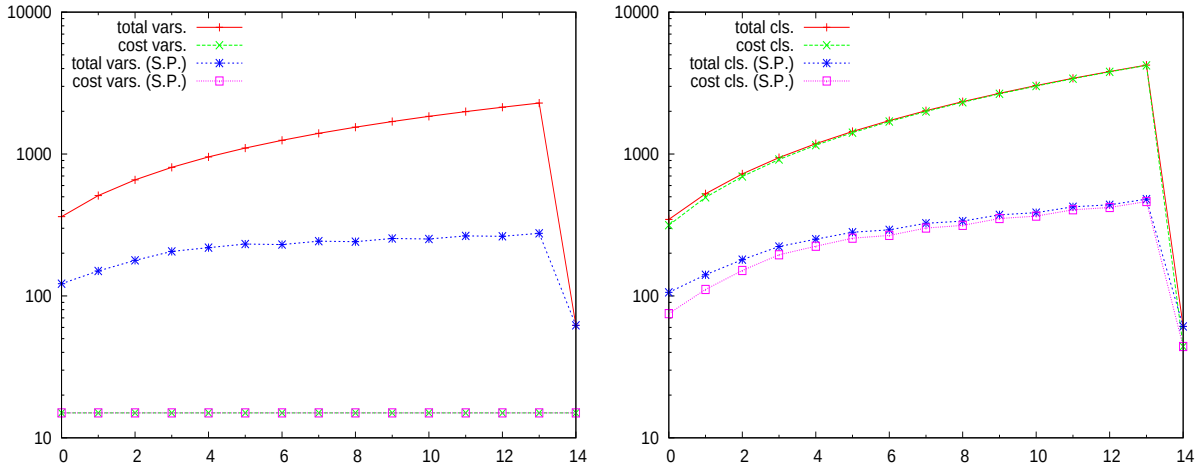

Figure 12: `restr_ratio`$_i(5)$. Left: variables; right: clauses. X axis: test case index; Y axis: #variables/clauses.

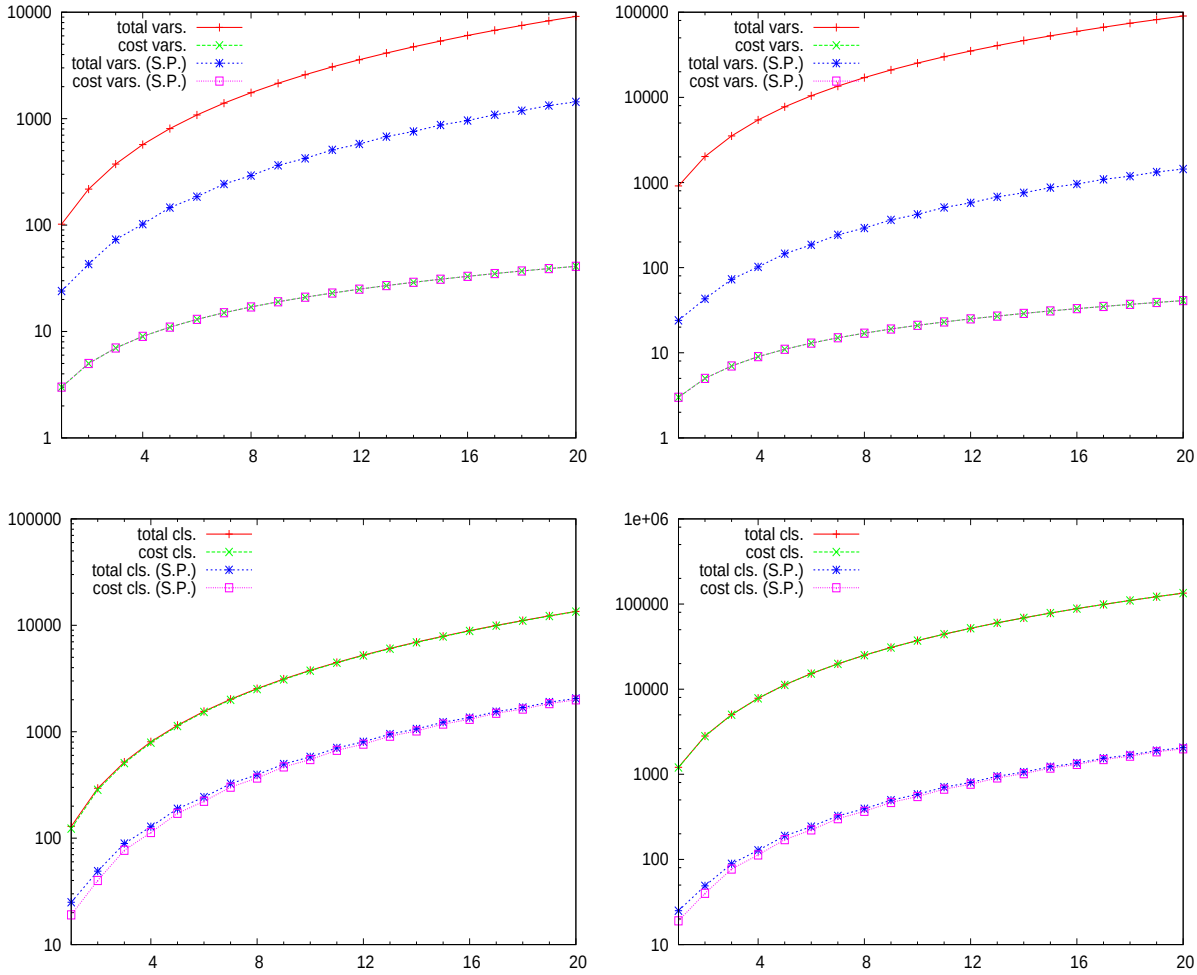

Figure 13: $\texttt{restr\_num}_i(n)$. 1st column: $n = 5$; 2nd column: $n = 50$. 1st row: variables; 2nd row: clauses. X axis: test case index; Y axis: #variables/clauses.

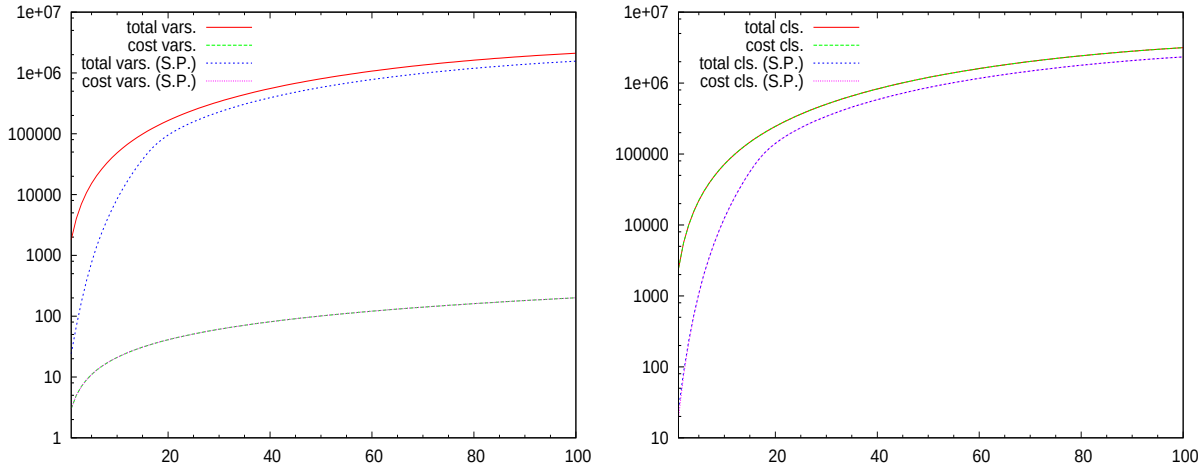

Figure 14: $\texttt{var\_restr\_num}_i(100)$. Left: variables; right: clauses. X axis: test case index; Y axis: #variables/clauses.

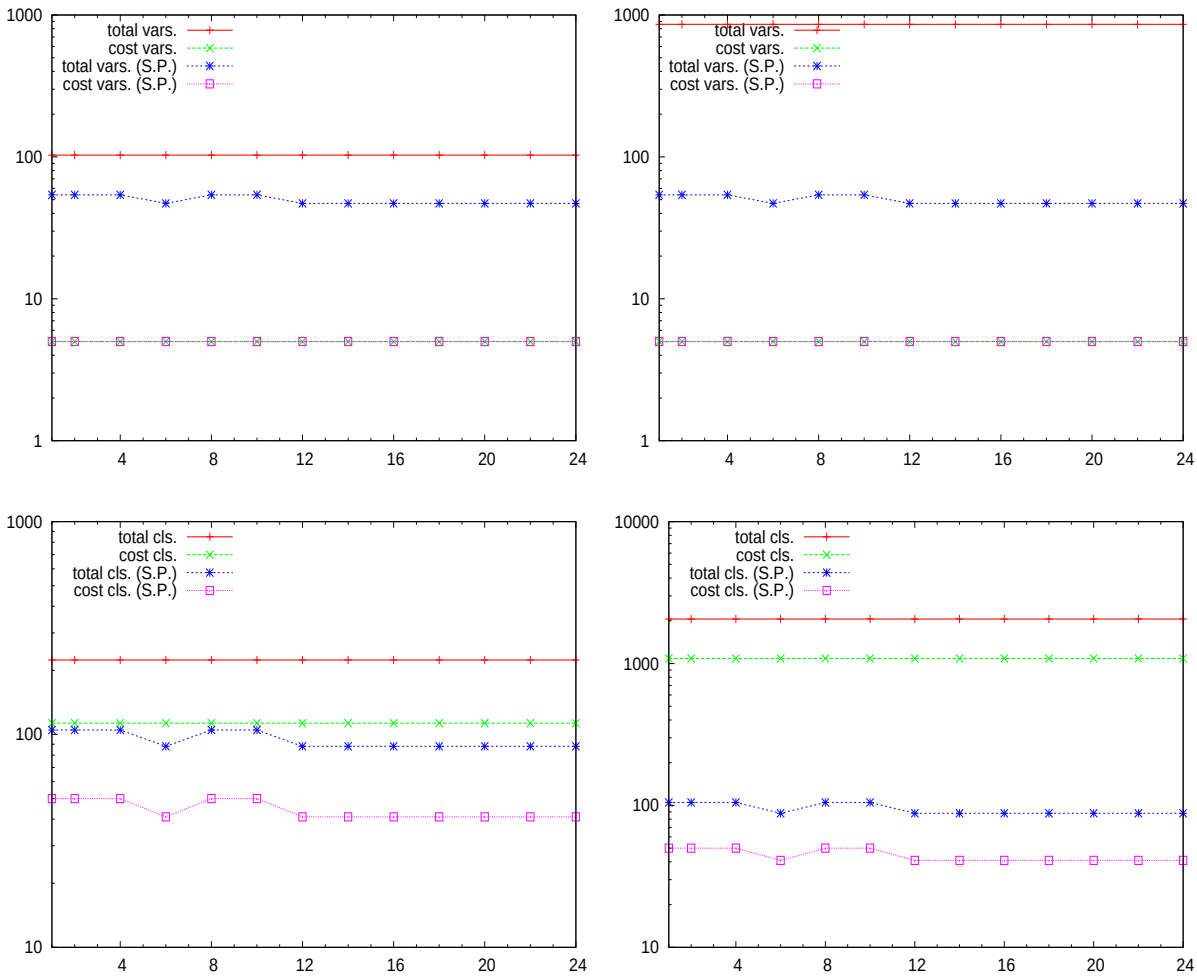

Figure 15: `sat_unsat`$_i(n)$. 1st column: $n = 1$; 2nd column: $n = 10$. 1st row: variables; 2nd row: clauses. X axis: test case index; Y axis: #variables/clauses.

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. In particular, the number of cost variables encoded only depends from the structure of the problem, a new cost variable is uniquely introduced for every distinct combination of individual, concept name and role name occurring in the qualified number restrictions of the problem. Notice that, if the encoded TBox provide nested qualified number restrictions then the reduction in the number of individuals would lead also to a sensible reduction of the cost variables, increasingly with the number of nested restrictions.

- Cost variables never depends on the value of the extra parameter n, differently from the number of clauses and of Boolean variables that scrictly depend from the number of introduced individuals.

- 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 10 and 15), 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 12), having chosen $m = 14$. However, the number of Boolean variables is predominant in all the test cases, also in the `backtracking`, `restr_num` and `var_restr_num` cases (Figure 11, 13 and 14), 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 in the encoding and in reducing the total size of the encoded problem.

- The major part of the clauses encoded by $\mathcal{ALCQ}$2SMT contains $\mathcal{C}$-literals. This property is even more evident for the the `restr_num` and `var_restr_num` benchmarks (Figures 13 and 14). On the contrary, the only exception to this observation is in the `backtracking` test cases (Figure 11), where the encoding of the mutual disjunctions conditions between concepts produces also 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. For instance, the relation with the index i (and, thus, with the values included in qualified number restrictions) is particularly clear in the `increasing` problems. Similarly, we can notice increases in the number of variables and clauses proportional to the increase in the value of n, e.g., in the `sat_unsat` test cases (Figure 15) where, instead, the size of the problem is indipendent from i. In such benchmark an increase of one order of magnitude in n, from 1 to 10, determines an increase of one order of magnitude also in the number of clauses and variables.

- Curiously, in the `sat_unsat` benchmark (Figure 15) 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 when for the indexes equal to 6 or greater than 12. This tricky behavior depends from our partitioning technique where the combinations of values occurring in at-least restrictions are merged with the combinations of values occurring in at-most restrictions. In these problems, the first are multiple of 3 for a maximum of 12, the seconds follow exactly i, where $i = 1, 2, 4, 6, \ldots, 24$. Thus, when i is 6 or is greater than 12 the merging of these values generates one partition less, which explains 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 $\texttt{rest\_num}_i(5)$ benchmark, which present up to 10^4 variables and clauses (Figure 13). 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 14). 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, for every i, they are stable in the order of "only" 1000 variables and clauses (Figure 15).

5.4 Discussion

As similarly discussed in [31] 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, conjunctions, disjunctions and/or 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 between the three components are regulated in two main ways. The Boolean component of SMT assigns the Boolean abstraction of the input SMT($\mathcal{C}$) problem, assigning also the $\mathcal{C}$-literals which determine 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 2.3). 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}$2SMT+MathSat is competitive or even outperforms the other tools, the benefits given by smart partitioning are outstanding. The effectiveness of smart partitioning lays in the drastic reduction it produces in the size of the output problems, while it does not change their logic. In particular, the more is the logical complexity of the encoded problem (e.g. unsatisfiable problems) the more prominent are the benefits of smart partitioning, cause the sensible reductions in size affect exponentially during the SMT($\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 case the approach competes well against the other state-of-the art tools, reporting comparable performance.

A simple explanation of the former observation could be 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. 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 input problem, so that our approach is affected by the size of the encoding (due to the (ii) component of the reasoning) while the other state-of-the-art tools can exploit specific optimization or reasoning techniques. Smart partitioning specifically acts on reducing the weight 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($\mathcal{C}$) problem, 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 the 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 depends 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, instead, of 1 million the result of smart partitioning (i.e. the number of distinct partitions that must be encoded) is the same. However, we think that in real-world ontologies extreme cases as those of the `var_restr_num` benchmark rarely occurs. Furthermore, even if theoretically and potentially expensive, in practice smart partitioning have shown computationally efficient. We think that the time spent in executing the smart partitioning routine is compensated from the gain it gives in the encoding steps that can be avoided.

Finally, notice that, in terms of performance, 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. Thus we think that the integration of such techniques and tools in a wider context may lead to successful results.

5.4.1 Scalability

We close our experimental evaluation by discussing the scalability issue.

We have chosen to face this issue here and 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 5.1 and in [11], 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 analyzed separately.

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 [11] and that we adapted in this work. Notice that, having nested occurrences of qualified number restrictions, is in our approach a further source of complexity much more significant than in traditional tableau-based algorithm, 'cause 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 jth 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 axioms, each of them introducing a deeper nested restrictions till the ith (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 conflict with the ith (last) axiom of the TBox at the deepest nesting level i. In the same above exposed system configuration, we run these test cases with $i = 1, \ldots, 20$ and $n = 5, 50$.

In Figure 16 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 16 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 cost variables, if number restrictions acts at more than one nesting depth. In the examined case, the 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 encoding CPU times required). In a few test cases, in fact, the gap between the two variants increases

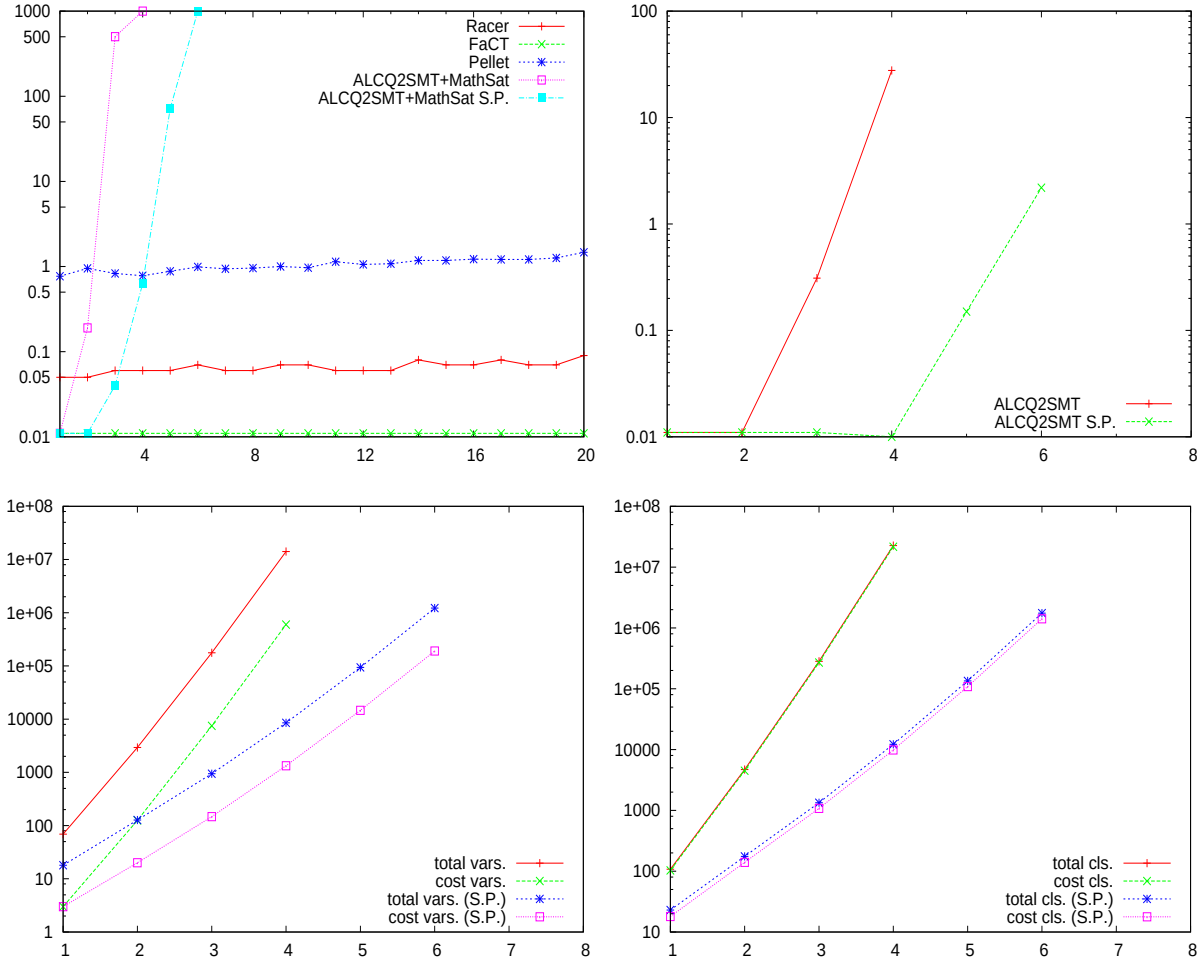

Figure 16: `nested_restr_sat`$_i(5)$. Top,left: comparison against other tools; top,right: $\mathcal{ALCQ}$2SMT times; bot,left: # enc. variables; bot,right: # enc. clauses. X axis: test case index; Y axis: 1st row: CPU time (sec), 2nd row: #variables/clauses.

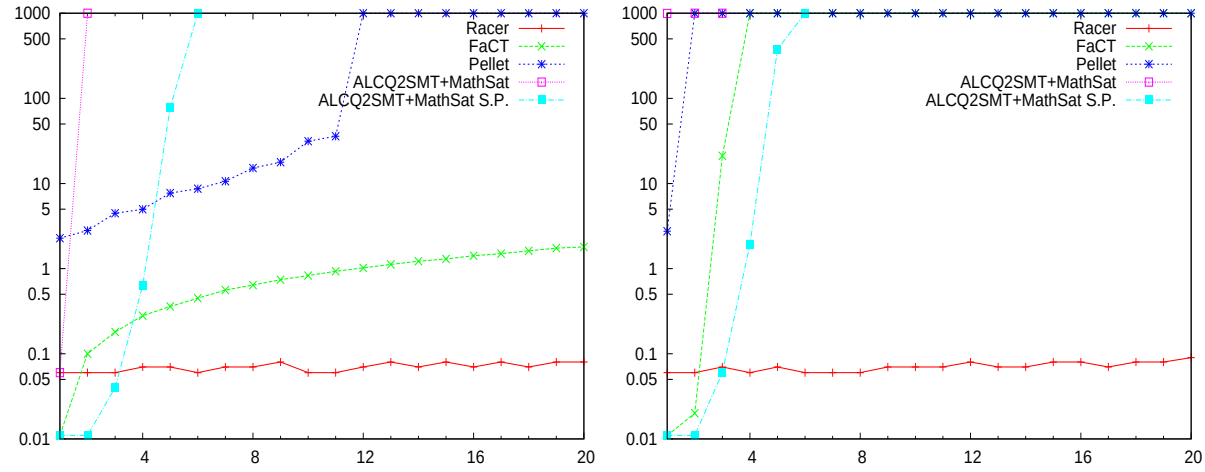

Figure 17: Left: `nested_restr_sat`$_i(50)$; right: `nested_restr_unsat`$_i(5)$. X axis: test case index; Y axis: CPU time (sec).

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 17 we report the tools comparison in other two slightly different test cases: $\texttt{nested\_restr\_sat}_i(50)$ and $\texttt{nested\_restr\_unsat}_i(5)$ respectively, from left to right. From the first plot it can be noticed how the performances of Pellet and FaCT++ gradually and significantly deteriorate having increased n, while 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 $\texttt{nested\_restr\_sat}_i(5)$ to the unsatisfiable cases $\texttt{nested\_restr\_unsat}_i(5)$. Notice that the only difference between the two benchmarks consists in one simple axiom, acting exclusively at the deepest 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 (and, furthermore, shown to be unsound in the successive one) 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. possible real-world ontologies, in which the different sources of complexity should be somehow balanced and not degenerating, we think that, overall, the performance shown by $\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.` 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 is only a preliminary analysis since we do not dispose of real practical problems.

6 Conclusions

In this 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 propose a further optimization, called smart partitioning, aiming at significantly reducing the size and the complexity 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 reasoning complexity, 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}$2SMT+MathSat `S.P.` (including smart partitioning), have shown to perform extremely well on benchmarks presenting multiple/balanced sources of complexity (that we think well-fit the requirements of real-world problems). In particular, we have noticed that the size of the encoding is not the main complexity issue for our approach, which has shown to be very effective also on large or really complex problems (e.g., MathSat scales up to encoded problems with more than 10^5 Boolean variables and clauses, and 10^4 cost variables, in very hard problems having nested qualified number restrictions).Finally, smart partitioning turns out to be extremely effective, being able to drastically (and exponentially) reduce the size of the output SMT($\mathcal{C}$) problems, up to three orders of magnitude in the more test cases wrt. basic $\mathcal{ALCQ}$2SMT (it exponentially impacts also in the number of cost variables in case of nested number restrictions). We remark that partitioning makes our approach independent from the magnitude/offset of the values occurring in qualified number restrictions.

A Proof

Theorem 1. *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 3) is satisfiable.*

Proof. It is a direct consequence of the following Lemmas 5 and 7. □

Lemma 5 (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 3, 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:

$$\Delta^{\mathcal{I}_\mu} \stackrel{def}{=} \{\ \sigma \mid A_{\langle \sigma,\ \top\rangle} \text{ occurs true in } \mu\ \}, \tag{21}$$

$$\hat{C}^{\mathcal{I}_\mu} \stackrel{def}{=} \{\ \sigma \mid \sigma \in \Delta^{\mathcal{I}_\mu} \text{ and } L_{\langle \sigma,\ \hat{C}\rangle} \text{ occurs true in } \mu\ \}, \tag{22}$$

$$r^{\mathcal{I}_\mu} \stackrel{def}{=} \{\ (\sigma, \sigma.r.i) \mid \sigma, \sigma.r.i \in \Delta^{\mathcal{I}_\mu}\ \}, \tag{23}$$

for every normal concept $\hat{C}$ and every role r in $\mathcal{T}$. In particular, by construction, it follows for every σ:

$$\sigma \in \hat{C}^{\mathcal{I}_\mu} \quad \text{if and only if} \quad L_{\langle \sigma,\ \hat{C}\rangle} \in \mu \ \text{ and } \ A_{\langle \sigma,\ \top\rangle} \in \mu. \tag{24}$$

For non-normal concepts we define $\mathcal{I}_\mu$ such that:

$$(\sqcap_i\ C_i)^{\mathcal{I}_\mu} \stackrel{def}{=} \{\ \sigma \mid \sigma \in \Delta^{\mathcal{I}_\mu} \text{ and } \mu \text{ satisfies } \wedge_i\ L_{\langle \sigma,\ C_i\rangle}\ \}, \tag{25}$$

$$(\sqcup_j D_j)^{\mathcal{I}_\mu} \stackrel{def}{=} \{\ \sigma \mid \sigma \in \Delta^{\mathcal{I}_\mu} \text{ and } \mu \text{ satisfies } \vee_j\ L_{\langle \sigma,\ D_j\rangle}\ \}. \tag{26}$$

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 1. 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, (21) and (22) 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 2.1.1. By hypothesis the generic individual σ respects the semantic of $\hat{C}$ reported in the right-most column of Table 1.

$\neg C$: By hypothesis we have $\sigma \in \Delta^{\mathcal{I}_\mu} \backslash \mathcal{C}^{\mathcal{I}_\mu}$. By construction of $\mathcal{I}_\mu$ (22), 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 (24), 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 $\mathcal{I}_\mu$ (25).

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 3 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 $\mathcal{I}_\mu$ (24) 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 (10) 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 (13) 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 $((\mathsf{BC}(\mathsf{indiv}_{\sigma.r}^C, m) \wedge A_{\langle \sigma,\ \top\rangle}) \rightarrow A_{\langle \sigma,\ \leq mr.C\rangle})$ of type (15) and (point 5.) the $m+1$ distinct implications $(\mathsf{IC}(\mathsf{indiv}_{\sigma.r}^C, 1, k_i^C) \rightarrow L_{\langle \sigma.r.k_i^C,\ C\rangle})$ of type (7), for all the distinct $\sigma.r.k_i^C$, with $i = 1, \ldots, m+1$. Thus, in $\varphi^{\mathcal{T}}$, more than m clause of type (7) exist. Suppose by contradiction that the value of the cost variable $\mathsf{indiv}_{\sigma.r}^C$ is greater than m. Thus more than m distinct $\mathcal{C}$-literals $\mathsf{IC}(\mathsf{indiv}_{\sigma.r}^C, 1, j)$ must be assigned to true in μ. But if these IC-literals are those occuring in clauses of type (7), like those above mentioned of index k_i^C, $i = 1, \ldots, m+1$, 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 (13), 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 $\sigma.r.k_i^C$ ones are in $\Sigma^{\mathcal{T}}$, then there are the conditions of point 7 of Definition 3, forcing the sharing of individuals and the introduction in $\varphi^{\mathcal{T}}$ of all the implications (11) and (12) for every $\sigma.r.j$. Those clauses forces the assignment to true of every literal $L_{\langle \sigma.r.j,\ C\rangle}$ such that $\mathsf{IC}(\mathsf{indiv}_{\sigma.r}^C, 1, j)$ is assigned to true, contradicting the fact that at most m of those individuals can be assigned to true. Hence, $\mathsf{indiv}_{\sigma.r}^C$ must have a value not greater than m. It follows from the clause $((\mathsf{BC}(\mathsf{indiv}_{\sigma.r}^C, m) \wedge A_{\langle \sigma,\ \top\rangle}) \rightarrow A_{\langle \sigma,\ \leq mr.C\rangle})$ (15) 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 $((\mathsf{BC}(\mathsf{indiv}_{\sigma.r}^{\neg C}, 0) \wedge A_{\langle \sigma,\ \top\rangle}) \rightarrow A_{\langle \sigma,\ \forall r.C\rangle})$ of type (17) and (point 5.) the single implication $(\mathsf{IC}(\mathsf{indiv}_{\sigma.r}^{\neg C}, 1, k_1^{\neg C}) \rightarrow L_{\langle \sigma.r.k_i^{\neg C},\ \neg C\rangle})$ of type (7). 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 $\mathsf{indiv}_{\sigma.r}^{\neg C}$ must have value 0; In fact, otherwise, we could get a contradiction with the 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 $((\mathsf{BC}(\mathsf{indiv}_{\sigma.r}^{\neg C}, 0) \wedge A_{\langle \sigma,\ \top\rangle}) \rightarrow A_{\langle \sigma,\ \forall r.C\rangle})$ (17) 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 7. of Definition 3. 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 $\mathcal{I}_\mu$ (24). Since $L_{\langle \sigma,\ \hat{C}\rangle}$ is in $\varphi^{\mathcal{T}}$, then (point 7.) $\varphi^{\mathcal{T}}$ contains the clause $(L_{\langle \sigma,\ \hat{C}\rangle} \rightarrow L_{\langle \sigma,\ \hat{D}\rangle})$ of type (6). 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 $\mathcal{I}_\mu$ (22).

(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 $\mathcal{I}_\mu$ (25). Since every $L_{\langle \sigma,\ C_i\rangle}$ is in $\varphi^{\mathcal{T}}$, then (point 7.) $\varphi^{\mathcal{T}}$ contains the clause $((\bigwedge_i L_{\langle \sigma,\ C_i\rangle}) \rightarrow L_{\langle \sigma,\ D\rangle})$ of type (6). It follows $L_{\langle \sigma,\ D\rangle} \in \mu$ because $\varphi^{\mathcal{T}}$ is satisfiable and, thus, $\sigma \in D^{\mathcal{I}_\mu}$ by (22).

(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 $\mathcal{I}_\mu$ (24). Since $L_{\langle \sigma,\ C\rangle}$ is in $\varphi^{\mathcal{T}}$, then (point 7.) $\varphi^{\mathcal{T}}$ contains the clause $(L_{\langle \sigma,\ C\rangle} \rightarrow (\bigvee_j L_{\langle \sigma,\ D_j\rangle}))$ of type (6). 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 $\mathcal{I}_\mu$ (26).

(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 2.1.1. By hypothesis the $\sigma \in \hat{D}^{\mathcal{I}_\mu}$.

$\neg D$: Let $\sigma \in (\neg D)^{\mathcal{I}_\mu}$. By definition (26) $\sigma \in \Delta^{\mathcal{I}_\mu}$ 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 (26), $A_{\langle \sigma,\ D_1\rangle} \vee A_{\langle \sigma,\ D_2\rangle}$ 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 (24) 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 $\mathcal{I}_\mu$ (24) 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 (7), (8) [resp. (11), (12)] and (9) due to the application wrt. σ and $\geq nr.D$ of the points 5. [resp. 7.] and 6., respectively, of Definition 3. Due to the clause (9): $(A_{\langle \sigma,\ \geq nr.D\rangle} \wedge A_{\langle \sigma,\ \top\rangle}) \rightarrow \neg \mathsf{BC}(\mathsf{indiv}^D_{\sigma.r}, n-1)$ 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 (7) [resp. (11)]. In fact, they can belong also to clauses of type (13), 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 3. 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 (11). 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 (8) [resp. (12)] that also at least n literals of the form $A_{\langle \sigma.r.j,\ \top\rangle}$ are assigned to true in μ. Hence, by (24), 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 $r^{\mathcal{I}_\mu}$ (23).

$\leq mr.D$: 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 $\mathcal{I}_\mu$ (24) 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 $\langle \sigma, \leq nr.D\rangle \in \mathcal{I}_+^{\mathcal{I}}$, $\varphi^{\mathcal{T}}$ contains all the clauses of type (13) and the clause (14) due to the application wrt. σ and $\leq mr.D$ of the points 8. and 6., respectively, of Definition 3. Due to the the clause (14) $(A_{\langle \sigma,\ \leq mr.D\rangle} \wedge A_{\langle \sigma,\ \top\rangle}) \rightarrow \mathsf{BC}(\mathsf{indiv}^D_{\sigma.r}, m)$, the SMT($\mathcal{C}$) formula $\varphi^{\mathcal{T}}$ is satisfied if and only if at most m different incur-cost literals $\mathsf{IC}\mathsf{indiv}^D_{\sigma.r}1\ldots$ 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 (24), 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 (13) 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 (16): $(A_{\langle \sigma,\ \forall r.D\rangle} \wedge A_{\langle \sigma.r.j,\ \top\rangle}) \rightarrow A_{\langle \sigma.r.j,\ D\rangle}$, for every $\sigma.r.j \in \Sigma^{\mathcal{T}}$, from point 10. of Definition 3. By definition of $\Delta^{\mathcal{I}_\mu}$ (21), $\sigma.r.j \in \Delta^{\mathcal{I}_\mu}$ if and only if $A_{\langle \sigma.r.j,\ \top\rangle}$ is assigned to true in μ. Thus, since, by construction (23), $(\sigma, \sigma.r.j) \in r^{\mathcal{I}_\mu}$ 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 (16) of $\varphi^{\mathcal{T}}$ (that is satisfiable). It follows (24) that $\sigma.r.j \in D^{\mathcal{I}_\mu}$ for every $(\sigma, \sigma.r.j) \in r^{\mathcal{I}_\mu}$.

□

Lemma 6. *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 3, 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 [22]. 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 $i = 1, \ldots, n_j$, s.t. $y^j_i \in \Delta^{\mathcal{I}}$, $y^j_i \in (C_j)^{\mathcal{I}}$ and $(x, y^j_i) \in r^{\mathcal{I}}$. Notice that, due to 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 $\sigma.r.k_i^{C_j} \in \Sigma^{\mathcal{T}}$, with $i = 1, \ldots, n_j$. For every j and for $i = 1, \ldots, n_j$ we map the individual y^j_i 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^j_i, with $i = 1, \ldots, n_j$, such that $y^j_i \in \Delta^{\mathcal{I}}$, $y^j_i \in C^{\mathcal{I}}_j$ and $(x, y^j_i) \in r^{\mathcal{I}}$ and, for every k, there are at-most m_k distinct individuals $y^k_{i'}$, with $i' = 1, \ldots, m_k$ such that $y^k_{i'} \in \Delta^{\mathcal{I}}$, $y^k_{i'} \in D^{\mathcal{I}}_k$ and $(x, y^k_{i'}) \in r^{\mathcal{I}}$. 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 $y^j_i = y^{j'}_{i'}$ (or $y^k_i = y^{k'}_{i'}$) 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 (11) and (12) (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^j_h and these individuals of $\Sigma^{\mathcal{T}}$ is suitable, provided that it must be a function, i.e. that if $y^j_h = y^{j'}_{h'}$, for some $j \neq j'$ and some values h, h', then y^j_h and $y^{j'}_{h'}$ 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 7 (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 3, 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 6. We built from $\mathcal{I}$ a total truth assignment μ satisfying $\varphi^{\mathcal{T}}$, as follows:

$$\mu \stackrel{def}{=} \mu_{\mathcal{I}} \cup \mu_{\overline{\mathcal{I}}} \tag{27}$$

$$\begin{aligned}
\mu_{\mathcal{I}} &\stackrel{def}{=} \mu_{\Delta} \cup \mu_X \cup \mu_{\geq} \cup \mu_{\leq} \cup \mu_{\forall} \cup \mu_{\mathsf{IC}} && (28)\\
\mu_{\Delta} &\stackrel{def}{=} \{\ A_{\langle \sigma,\ \top\rangle} \mid A_{\langle \sigma,\ \top\rangle} \text{ literal of } \varphi^{\mathcal{T}},\ \sigma \in \Delta^{\mathcal{I}}\} && (29)\\
\mu_X &\stackrel{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}}\} && (30)\\
\mu_{\geq} &\stackrel{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}}\} && (31)\\
\mu_{\leq} &\stackrel{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}}\} && (32)\\
\mu_{\forall} &\stackrel{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}}\} && (33)\\
\mu_{\mathsf{IC}} &\stackrel{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}}\} && (34)
\end{aligned}$$

$$\begin{aligned}
\mu_{\overline{\mathcal{I}}} &\stackrel{def}{=} \mu_{\overline{\Delta}} \cup \mu_{\overline{X}} \cup \mu_{\overline{\mathsf{BC}}} \cup \mu_{\overline{\mathsf{IC}}} && (35)\\
\mu_{\overline{\Delta}} &\stackrel{def}{=} \{\neg A_{\langle \sigma,\ \top\rangle} \mid A_{\langle \sigma,\ \top\rangle} \text{ literal of } \varphi^{\mathcal{T}},\ \sigma \notin \Delta^{\mathcal{I}}\} && (36)\\
\mu_{\overline{\mathsf{BC}}} &\stackrel{def}{=} \{\ \mathsf{BC}(\mathsf{indiv}^C_{\sigma.r}, n) \mid \mathsf{BC}(\mathsf{indiv}^C_{\sigma.r}, \ldots) \text{ literal of } \varphi^{\mathcal{T}},\ \sigma \notin \Delta^{\mathcal{I}},\ \text{for any } n\} && (37)\\
\mu_{\overline{\mathsf{IC}}} &\stackrel{def}{=} \{\neg\mathsf{IC}(\mathsf{indiv}^C_{\sigma.r}, 1, i) \mid \mathsf{IC}(\mathsf{indiv}^C_{\sigma.r}, 1, \ldots) \text{ literal of } \varphi^{\mathcal{T}},\ \sigma \notin \Delta^{\mathcal{I}},\ \text{for any } i\} && (38)
\end{aligned}$$

where $\mu_{\overline{X}}$ is a consistent truth assignment satisfying all the clauses of type (6) 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) to (17).

(6): Clauses of type (6) 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 (30), 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 $C_1 \sqcap C_2 \sqsubseteq D$, 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 (30), 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 30, 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.

(7), (8), (11), (12): 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 (7))/(11), and $\mathsf{IC}(\mathsf{indiv}^C_{\sigma.r}, 1, i) \rightarrow A_{\langle \sigma.r.i,\ \top\rangle}$ of type (8)/(12), 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 (29), $L_{\langle \sigma.r.i,\ C\rangle} \in \mu_X$ from (30) and $\mathsf{IC}(\mathsf{indiv}^C_{\sigma.r}, 1, i) \in \mu_{\mathsf{IC}}$ from (34), 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 34, which trivially satisfies both the clauses.

(9), (10): 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 (9) 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 (10). Since $\sigma \in \Delta^{\mathcal{I}}$ by hypothesis then $A_{\langle \sigma,\ \top\rangle} \in \mu_\Delta$ by (29). If also $\sigma \in (\geq nr.C)^{\mathcal{I}}$, then $A_{\langle \sigma,\ \geq nr.C\rangle} \in \mu_X$ by (30) and $\neg\mathsf{BC}(\mathsf{indiv}^C_{\sigma.r}, n-1) \in \mu_{\geq}$ by (31) satisfying both the clauses. Otherwise, if $\sigma \notin (\geq nr.C)^{\mathcal{I}}$, then the clause (9) is trivially satisfied since $\neg A_{\langle \sigma,\ \geq nr.C\rangle} \in \mu_X$ by (30), while $\mathsf{BC}(\mathsf{indiv}^C_{\sigma.r}, n-1) \in \mu_{\geq}$ by (31) which satisfies the clause (10).

(13): 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 (13) 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 (29), $L_{\langle \sigma.r.i,\ C\rangle} \in \mu_X$ from (30) and $\mathsf{IC}(\mathsf{indiv}^C_{\sigma.r}, 1, j) \in \mu_{\mathsf{IC}}$ from (34), 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 (30), 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 (36), which trivially satisfies the clause.

(14), (15): 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 (14). 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 (15). Since $\sigma \in \Delta^{\mathcal{I}}$ by hypothesis, then $A_{\langle \sigma,\ \top\rangle} \in \mu_\Delta$ by (29). If also $\sigma \in (\leq mr.C)^{\mathcal{I}}$, then $A_{\langle \sigma,\ \leq mr.C\rangle} \in \mu_X$ by (30) and $\mathsf{BC}(\mathsf{indiv}^C_{\sigma.r}, m) \in \mu_{\leq}$

by (32) satisfying both the clauses. Otherwise, if $\sigma \notin (\leq mr.C)^{\mathcal{I}}$, then the clause (14) is trivially satisfied since $\neg A_{\langle \sigma,\ \leq mr.C\rangle} \in \mu_X$ by (30), while $\neg \mathsf{BC}(\mathsf{indiv}^C_{\sigma.r}, m) \in \mu_{\leq}$ by (32) which satisfies the clause (15).

(16): Wlog. let us consider the generic clause of type (16): $(A_{\langle \sigma,\ \forall r.C\rangle} \wedge A_{\langle \sigma.r.i,\ \top\rangle}) \rightarrow L_{\langle \sigma.r.i,\ C\rangle}$, 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 (30). If $\sigma \in (\forall r.C)^{\mathcal{I}}$ then we have $A_{\langle \sigma,\ \forall r.C\rangle} \in \mu_X$ from (30) 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 (36);
- 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 (30). Further, from $\sigma.r.i \in \Delta^{\mathcal{I}}$ we have $A_{\langle \sigma.r.i,\ \top\rangle} \in \mu_{\Delta}$ (29) satisfying the clause.

(17): 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 (17). Since $\sigma \in \Delta^{\mathcal{I}}$ by hypothesis, then $A_{\langle \sigma,\ \top\rangle} \in \mu_{\Delta}$ by (29). Then, by definition of μ_X (30) and $\mu_{\forall}$ (33) 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 (15).

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) to (17).

(7), (8), (11), (12): All the clauses of these types are trivially satisfied by $\mu_{\overline{\mathsf{IC}}}$, since, by (38), 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 $\sigma \notin \Delta^{\mathcal{I}}$.

(9), (10), (14), (15), (17): 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 (36).

(13), (16): 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 (36), which trivially satisfies the clauses.

(6): Finally, we consider the case of all the clauses of type (6) in which $\sigma \notin \Delta^{\mathcal{I}}$. The clauses of type (6) 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) 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) 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) 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) 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) 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) 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 $\mu_{\overline{\mathsf{BC}}}$ (37) assigns to true every bound $\mathsf{BC}(\mathsf{indiv}^C_{\sigma.r}, \ldots)$ such that $\sigma \notin \Delta^{\mathcal{I}}$ while, instead, $\mu_{\overline{\mathsf{IC}}}$ (38) 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 (31) for $\mu_{\geq}$ or by (32) 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 (34), 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 (32) for $\mu_{\leq}$ or by (31) 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 (34), 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 (33), 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 6, 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} (34) nor by $\mu_{\overline{\mathsf{IC}}}$ (38), 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 (33). 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} (34), consistently with the Theory of Costs.

□

Moreover, we prove the following result.

Theorem 2. *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 3, 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 5 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 5 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 1 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 6) from which it follows, by Lemma 7, that there exists a truth assignment μ satisfying $\varphi^{\mathcal{T}}$ build up as in (27). 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 7 (30). $\square$

B Appendix: Additional plots on $\mathcal{ALCQ}2SMT_{\mathcal{C}}$

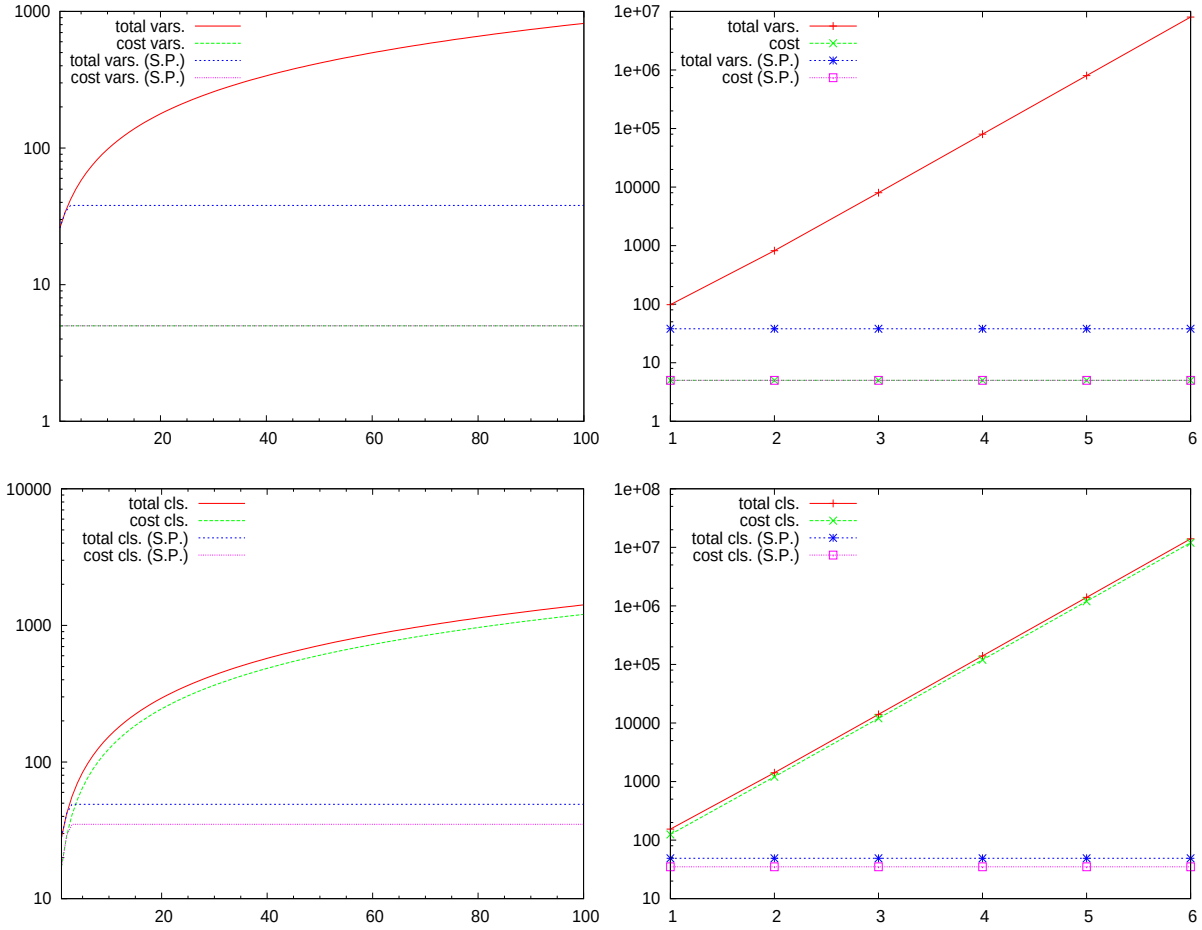

Figure 18: 1st column: $\texttt{increasing\_lin\_unsat}_i$, $i = 1, \ldots, 20$; 2nd column: $\texttt{increasing\_exp\_unsat}_i$, $i = 1, \ldots, 6$. 1st row: variables; 2nd row: clauses. X axis: test case index; Y axis: #variables/clauses.

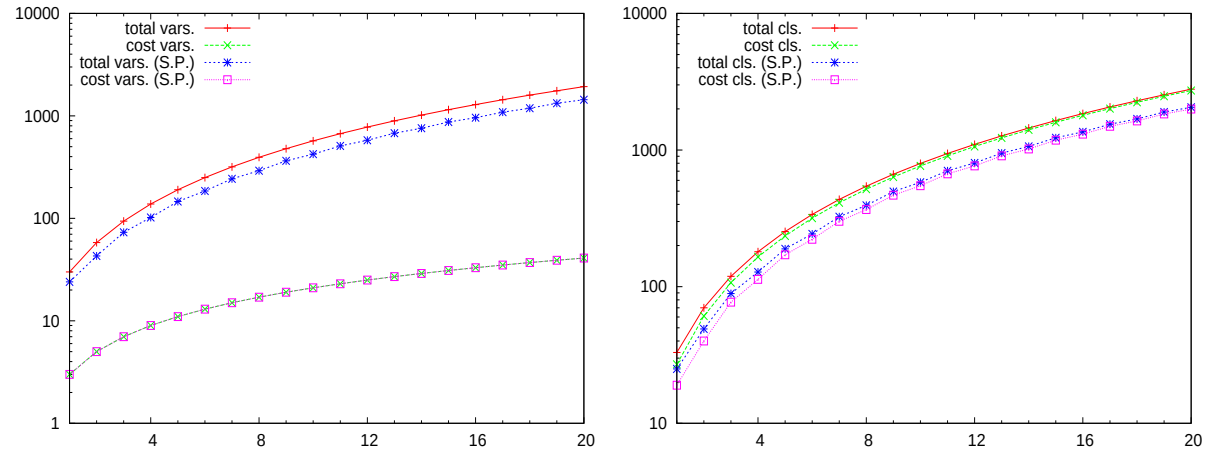

Figure 19: $\texttt{restr\_num}_i(1)$. Left: variables; right: clauses. X axis: test case index; Y axis: #variables/clauses.

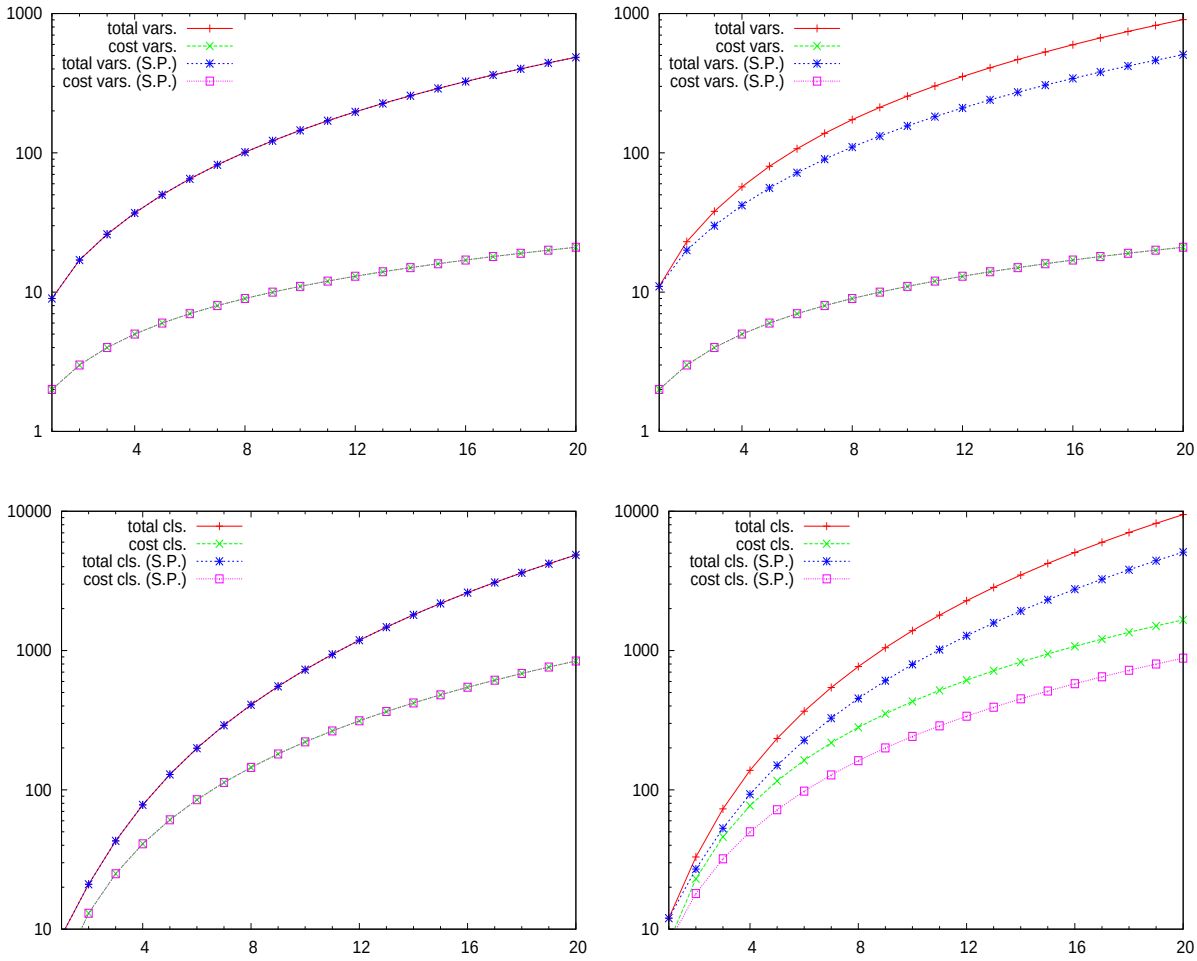

Figure 20: $\texttt{backtracking}_i(n)$. 1st column: $n = 1$; 2nd column: $n = 2$. 1st row: variables; 2nd row: clauses. X axis: test case index; Y axis: #variables/clauses.

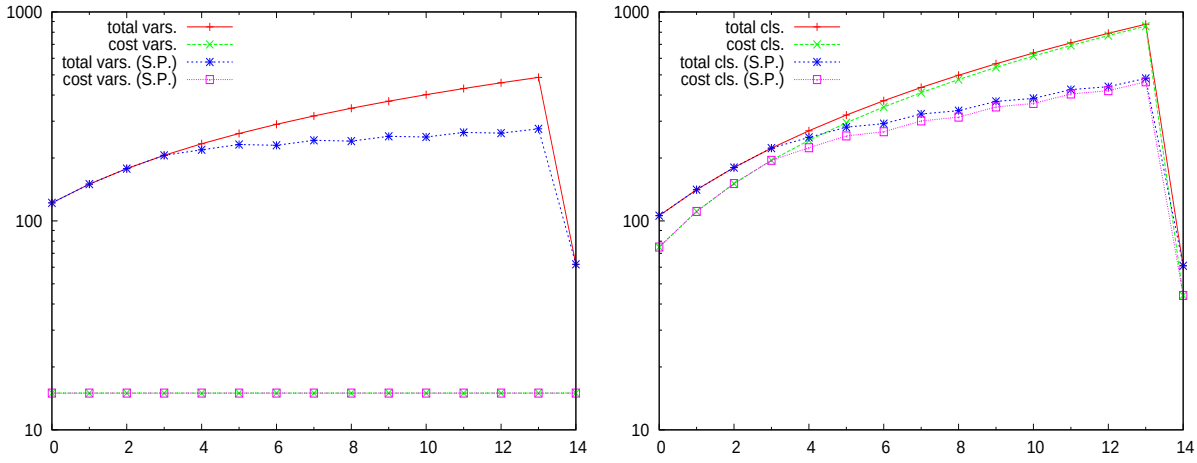

Figure 21: $\texttt{restr\_ratio}_i(1)$. Left: variables; right: clauses. X axis: test case index; Y axis: #variables/clauses.

References

[1] F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, 2003.

[2] C. W. Barrett, R. Sebastiani, S. A. Seshia, and C. Tinelli. Satisfiability Modulo Theories. In *Handbook of Satisfiability*, chapter 26, pages 825–885. IOS Press, 2009.

[3] R. Bruttomesso, A. Cimatti, A. Franzén, A. Griggio, and R. Sebastiani. The MathSAT 4 SMT Solver. In *Proc. of the CAV/08*, volume 5123 of *LNCS*, pages 299–303. Springer, 2008.

[4] A. Cimatti, A. Franzén, A. Griggio, R. Sebastiani, and C. Stenico. Satisfiability Modulo the Theory of Costs: Foundations and Applications. In *Proc. of TACAS 2010*, volume 6015 of *LNCS*, pages 99–113. Springer, 2010.

[5] M. Davis, G. Longemann, and D. Loveland. A Machine Program for Theorem-proving. *Communications of the ACM*, 5:394–397, July 1962.

[6] M. Davis and H. Putnam. A Computing Procedure for Quantification Theory. *Journal of the ACM*, 7:201–215, July 1960.

[7] N. Eén and N. Sörensson. An Extensible SAT-solver. In *Proc. of SAT'03*, volume 2919 of *LNCS*, pages 502–518. Springer, 2004.

[8] J. Faddoul, N. Farsinia, V. Haarslev, and R. Möller. A Hybrid Tableau Algorithm for $\mathcal{ALCQ}$. In *Description Logics 2008 (DL2008)*, volume 353 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2008.

[9] J. Faddoul and V. Haarslev. Algebraic Tableau Reasoning for the Description Logic $\mathcal{SHOQ}$. *Journal of Applied Logic, Special Issue on Hybrid Logics*, 8(4):334–355, December 2010.

[10] J. Faddoul and V. Haarslev. Optimizing algebraic tableau reasoning for $\mathcal{SHOQ}$: First experimental results. In *Proc. of DL'2010, Waterloo, Canada, May 4-7*, pages 161–172, 2010.

[11] N. Farsiniamarj and V. Haarslev. Practical reasoning with qualified number restrictions: A hybrid abox calculus for the description logic $\mathcal{SHQ}$. *J. AI Communications*, 23(2-3):205–240, March 2010.

[12] F. Gasse and V. Haarslev. Expressive Description Logics via SAT: The Story so Far. In *Proc. of the SMT-2009 Workshop*, volume 375 of *ACM Int. Conf. Proc. Series*, pages 30–34, 2009.

[13] F. Giunchiglia and R. Sebastiani. Building decision procedures for modal logics from propositional decision procedures - the case study of modal K. In *Proc. of CADE-13*, volume 1104 of *LNAI*, pages 583–597. Springer, August 1996.

[14] V. Haarslev and R. Möller. Optimizing Reasoning in Description Logics with Qualified Number Restrictions. In *Description Logics*, volume 49 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2001.

[15] V. Haarslev and R. Möller. RACER System Description. In *Proc. of IJCAR'01*, volume 2083 of *LNAI*, pages 701–706. Springer, July 2001.

[16] V. Haarslev and R. Möller. Racer: A core inference engine for the semantic web. In *Proc. of EON2003*, pages 27–36, 2003.

[17] V. Haarslev, M. Timmann, and R. Möller. Combining Tableaux and Algebraic Methods for Reasoning with Qualified Number Restrictions. In *Proc. of DL'2001, Aug. 1-3, Stanford, USA*, volume 49 of *CEUR Workshop Proceedings*, pages 152–161. CEUR-WS.org, 2001.

[18] J. Halpern and Y. Moses. A guide to the completeness and complexity for modal logics of knowledge and belief. *Artificial Intelligence*, 54(3):319–379, 1992.

[19] B. Hollunder and F. Baader. Qualifying Number Restrictions in Concept Languages. In *Proc. of KR'91*, pages 335–346, Boston (USA), 1991.

[20] I. Horrocks, U. Sattler, and S. Tobies. Practical reasoning for very expressive description logics. *Logic Journal of the IGPL*, 8(3):239–264, 2000.

[21] I. Horrocks, U. Sattler, and S. Tobies. Reasoning with Individuals for the Description Logic $\mathcal{SHIQ}$. In *CADE-17*, volume 1831 of *Lecture Notes in Computer Science*, pages 482–496. Springer, 2000.

[22] C. Lutz, C. Areces, I. Horrocks, and U. Sattler. Keys, Nominals, and Concrete Domains. *Journal of Artificial Intelligence Research*, 23(1):667–726, 2005.

[23] I. Lynce and J. P. Silva. On Computing Minimum Unsatisfiable Cores. In *Proceedings of SAT'04*, pages 305–310, 2004.

[24] M. W. Moskewicz, C. F. Madigan, Y. Zhao, L. Zhang, and S. Malik. Chaff: Engineering an Efficient SAT Solver. In *Proc. of DAC'01*, pages 530–535. ACM, 2001.

[25] B. Motik, P. F. Patel-Schneider, and B. Parsia. OWL 2 Web Ontology Language: Structural Specication and Functional-Style Syntax. W3C Recommendation, 27 October 2009. Available at `http://www.w3.org/TR/owl2-syntax/`.

[26] B. Motik, R. Shearer, and I. Horrocks. Hypertableau Reasoning for Description Logics. *Journal of Artificial Intelligence Research*, 36:165–228, 2009.

[27] H. J. Ohlbach and J. Koehler. Role Hierarchies and Number Restrictions. In *Description Logics 1997*, volume 410 of *URA-CNRS*, 1997.

[28] H. J. Ohlbach and J. Koehler. Modal Logics, Description Logics and Arithmetic Reasoning. *Journal of Artificial Intelligence*, 109(1-2):1–31, 1999.

[29] R. Sebastiani. Lazy Satisfiability Modulo Theories. *Journal on Satisfiability, Boolean Modeling and Computation, JSAT*, 3:141–224, 2007.

[30] R. Sebastiani and M. Vescovi. Encoding the Satisfiability of Modal and Description Logics into SAT: The Case Study of $K(m)/\mathcal{ALC}$. In *Proceedings of SAT'06*, volume 4121 of *LNCS*, pages 130–135. Springer, 2006.

[31] R. Sebastiani and M. Vescovi. 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):343–389, June 2009.

[32] R. Sebastiani and M. Vescovi. Axiom Pinpointing in Lightweight Description Logics via Horn-SAT Encoding and Conflict Analysis. In *Proceedings of of CADE-22*, volume 5663 of *Lecture Notes in Computer Science*, pages 84–99. Springer, 2009.

[33] J. P. Silva and K. A. Sakallah. GRASP – A new Search Algorithm for Satisfiability. In *Proceedings of ICCAD'96*, pages 220–227. IEEE Computer Society, 1996.

[34] E. Sirin, B. Parsia, B. C. Grau, A. Kalyanpur, and Y. Katz. Pellet: A practical OWL-DL reasoner. *Journal of Web Semantics*, 5(2):51–53, June 2007.

[35] D. Tsarkov and I. Horrocks. FaCT++ description logic reasoner: System description. In *Proc. of IJCAR'06*, volume 4130 of *LNAI*, pages 292–297. Springer, 2006.

[36] L. Zhang, C. F. Madigan, M. W. Moskewicz, and S. Malik. Efficient Conflict Driven Learning in Boolean Satisfiability Solver. In *Proceedings of ICCAD'01*, pages 279–285, 2001.

[37] L. Zhang and S. Malik. The Quest for Efficient Boolean Satisfiability Solvers. In *Proc. of CAV'02*, volume 2404 of *LNCS*, pages 17–36. Springer, 2002.