UNIVERSITÀ DEGLI STUDI
DI TRENTO

DEPARTMENT OF INFORMATION ENGINEERING AND COMPUTER SCIENCE
ICT International Doctoral School

Automating the Generation of Goal-Oriented Dialogue Managers for Healthcare

Milene Santos Teixeira

Advisor

Dr. Mauro Dragoni

Fondazione Bruno Kessler

Co-Advisor

Dr. Claudio Eccher

Fondazione Bruno Kessler

December 2022

Abstract

Conversational agents can benefit healthcare across different application domains. However, the automated generation of reliable agents is still challenging and lags behind traditional conversational domains. This research exploited the interplay of information management and automated planning to efficiently model the expected behavior of goal-oriented health dialogues. The proposed approach supports the dynamic generation of predictable policies that are used for the management of the health dialogue as well as the identification of the dialogue state. This work advances the state of the art in health dialogue management by automating the generation (and update) of efficient dialogue managers with a reduced cost since they do not require handcrafting of the dialogue policy or large conversational datasets.

Keywords

[dialogue management; automated planning; ontology; healthcare; information management]

Acknowledgments

First and foremost, I would like to thank God. Although I have been a little distant from Him in recent years, I am still a believer and I praise God for my accomplishments.

For all their support during the whole Ph.D. path, I thank my advisor Dr. Mauro Dragoni and co-advisor Dr. Claudio Eccher. I thank them for the many times that they guided me, encouraged me, and made me believe that I was able to pursue this Ph.D. I also thank them for their constant search for opportunities that could expand my network and improve the outcome of my research.

I would like to show my greatest appreciation to the admirable researchers that, in different ways, helped me to shape my work: Dr. Vinicius Maran, Dr. Celia da Costa Pereira, Dr. Christian Muise, and Dr. Tiago Guerreiro. The outcome of this work would not have been the same without your collaboration. *If I have seen further, it is by standing on the shoulders of Giants.*

I also thank both my parents (Jose and Liza), who from an early age, have motivated me to study hard and have shown me that my achievements are consequences of my hard work.

Last but not least, I thank all those friends who empathized with my struggles and somehow motivated me along this journey.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Conversational agents (or dialogue systems) can benefit healthcare across different application domains, such as counseling [80], disease monitoring [51], and rehabilitation support [88]. To the physician, a dialogue system can support diagnosis [123] or image annotation [107], for example. To patients, these systems may contribute by providing reliable information that can reduce the number of medical visits that imply time and financial expenses. On the other hand, a health dialogue system may be capable of clarifying information after medical visits, considering that usually medical visits are time constrained and the patient may feel intimidated to ask questions or to ask for the information to be repeated.

Despite their benefits, current health dialogue systems are limited and lag behind other domains, raising concerns that still need to be addressed [65]. A reason for that is that health dialogues present more complex challenges when compared to the classic conversation paradigm. A dialogue system that supports patient screening, for example, must be able to choose the most appropriate action on each turn to detect as soon as possible risky situations. Besides, a health dialogue system must be based on medical expert knowledge and it cannot rely on automatic responses that may be valid, but not the most adequate, leading a patient to risky situations.

Given such challenges, several approaches implement semantic-aware strategies that rely on ontologies to build these systems [60, 77]. Ontologies, which are defined as "a formal, explicit specification of a shared conceptualization" [53], are capable of supporting a system with domain-specific knowledge, delivering more reliable outcomes. However, automating the generation of health dialogue managers that are explainable and keep control of the dialogue policy is still an open challenge [98].

Automated planning, that has been seen as a promising alternative for dialogue management [32, 94], offers this control without limiting the flexibility of the dialogue. AI planning can treat each utterance as an action, dealing with the action selection problem with the focus of achieving a (dialogue) goal. However, modeling the behavior of the dialogue as a planning problem is a complex task [81, 46]. In addition to domain knowledge, it requires expertise in both dialogue modeling and AI planning, which is not frequent among dialogue authors, limiting the adoption of such a powerful approach. Hence, to build efficient policies, AI planning can benefit from the integration with further mechanisms to formalize the knowledge of a specific domain. These requirements meet the definition of ontology.

This research exploited the interplay of information management and AI planning to efficiently model the expected behavior of a goal-oriented health dialogue. Supported by an ontology that formalizes the goal-oriented conversation scenario, the approach proposed in this research, which is named *OntoPlanDM*, covers the dynamic generation of the strategy that is used for the management of the health dialogue, i.e. the dialogue policy, as well as the identification of the dialogue state. A conversational agent deploying *OntoPlanDM* as its dialogue manager component is expected to handle multi-turn dialogues that are explainable and predictable. The list of contributions of this research is listed next.

1.1 Contributions

This thesis provides the five following contributions:

1. A survey of the state-of-the-art on plan-based approaches for dialogue management. This survey identifies (i) the types of planning employed in recent approaches, (ii) the planning characteristics that justify its adoption in dialogue systems, and (iii) the challenges posed on the development of plan-based dialogue managers.

2. An ontology that formalizes the goal-oriented conversational paradigm. This ontology aims to: (i) semantically model and describe the concepts of the dialogue and (ii) aid the understanding and management of the entire dialogue workflow.

3. An agent-based framework for the selection of the most useful action on each state of the health dialogue. This framework contributes to the generation of a multi-turn dialogue that uses as few questions as possible to retrieve the information, seeking a fast, but also a cautious solution.

4. *OntoPlanDM*, an approach integrating automated planning and information management for the automated generation of health dialogue managers. This goal-oriented approach is aimed to be integrated into a suitable dialogue system architecture that provides the natural language components.

5. *Puffbot*, a prototype of a conversational agent deploying *OntoPlanDM*. *Puffbot* was designed to support the monitoring of the asthma disease. It was tested and evaluated by both health experts and end-users and this system is expected to be used in further research projects conducted at the Fondazione Bruno Kessler.

1.2 Publications

This section provides a list of the author's publications that are direct results of the research conducted in her Ph.D.

1. Milene Santos Teixeira and Mauro Dragoni. A Review of Plan-Based Approaches for Dialogue Management. In *Cognitive Computation*, pages 1-20. Springer, 2022.

2. Milene Santos Teixeira, Vinícius Maran, and Mauro Dragoni. The interplay of a conversational ontology and AI planning for health dialogue management. In *Proceedings of the 36th Annual ACM Symposium on Applied Computing*, pages 611-619. 2021.

3. Milene Santos Teixeira, Vinícius Maran, and Mauro Dragoni. Towards Semantic-Awareness for Information Management and Planning in Health Dialogues. In *34th International Symposium on Computer-Based Medical Systems (CBMS)*, pages 372-377. IEEE, 2021.

4. Milene Santos Teixeira, Célia da Costa Pereira, and Mauro Dragoni. Information Usefulness as a Strategy for Action Selection in Health Dialogues. In *2020 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT)*, pages 323-330. IEEE, 2020.

5. Milene Santos Teixeira, Célia da Costa Pereira, and Mauro Dragoni. A Goal-Based Framework for Supporting Medical Assistance: The Case of Chronic Diseases. In *International Conference on Principles and Practice of Multi-Agent Systems*, pages 281-298. Springer, 2020.

6. Milene Santos Teixeira, Mauro Dragoni, and Claudio Eccher. A planning strategy for dialogue management in healthcare. In *SWH@ ISWC*. 2019.

Paper #1 corresponds to the survey identified as contribution #1. Papers #2 and #3 reported a significant, but partial description of the approach proposed in this thesis (contribution #4). While paper #2 presents an evaluation from the experts' point of view, paper #3 has an evaluation of the planning technical aspects. Contributions #2 and #5 (an early version) were also addressed on them. Paper #4, instead, introduced contribution #3, and paper #5 extended it with the concepts of *cautiousness* and *fast-solving* solutions. Finally, paper #6 corresponds to an initial investigation conducted at the very beginning of the Ph.D.

1.3 Structure of the Thesis

Chapter 2 presents the fundamentals constituting the background knowledge necessary to comprehend this work as well as the motivation for this research. An asthma scenario is presented to illustrate the dialogue systems approached. *Chapter 3* describes the recent literature on health dialogue management, presenting the main challenges and efforts from the community to address this topic. Ontology-based dialogue systems are also discussed in this chapter, while plan-based approaches are part of an extensive discussion in *Chapter 4*. *Chapter 5* introduces the ontology proposed during this research, which was used in the defined approach. *Chapter 6*, instead, introduces the usefulness framework that was designed to support planning on action selection by focusing on the challenges of the health domain. In *Chapter 7*, we put these elements together, defining the approach for the generation of health dialogue managers. *Chapter 8* describes the studies conducted to evaluate this work; first our early evaluations are described and, next, three user studies are detailed. Finally, *Chapter 9* closes this thesis, summarizing the research conducted, presenting the limitations of this research, and suggesting directions for future research.

Chapter 2

Background and Motivation

This chapter aims to introduce the general aspects of the research conducted and presented in this Ph.D. thesis. First, we describe a sample motivational scenario that is related to the automatic remote management of an asthma action plan. Conversational agents represent relevant AI-based tools to support such scenarios. This scenario was chosen since it presents a relatively limited complexity when compared to other health domains. Hence, it delivers an easier way for the reader to comprehend the main rationales behind the work presented in this thesis. For instance, the scenario clearly states the types of dialogue addressed, what information should be retrieved, which possible classifications can be achieved, and what are the corresponding recommendations for each situation. Therefore, it is exploited as the starting point for the development of the proposed approach and it will be mentioned in different parts of this thesis. From this scenario, the proposed approach is expected to be exploited to cover more complex problems. Next, this chapter introduces some fundamental background on goal-oriented dialogue systems as well as the types of dialogue covered in this work. Finally, the motivation for proposing an approach for the automated generation of health dialogue managers is given and a general idea of the proposed approach is presented.

2.1 The Asthma Scenario

The research described in this thesis has been conducted with the aim of providing an approach that can be reused by different health dialogue systems. However, to ground the problem and work on a solution towards a real-world application, the asthma scenario has been exploited as a running example throughout this thesis. This scenario was chosen for two main reasons. First, such a scenario needs the management of conversations that are not limited to single sessions but that require a multi-turn conversation due to the necessity of acquiring information about the different types of symptoms held by the patient. Second, the size of this scenario is limited enough to perform "debugging" operations for both reasoning and planning activities in order to verify the effectiveness of the proposed solution. Although we are aware of the increasing complexity within other healthcare-wise domains, which can involve many external factors (e.g., management of information coming from sensors or web services), it is possible to describe the asthma scenario with a finite set of entities that, at the same time, make it possible to explore the conversational paradigm thoroughly.

Thanks to a collaboration with health experts (pulmonologists) from the Trentino Healthcare Department [1], an action plan addressing asthma undesired events has been designed. The action plan, which can be seen in Figure 2.1, describes four classes:

- *Green zone*: the patient is in a normal condition (i.e. no crisis);
- *Yellow zone*: the patient has a light asthma attack;
- *Amber zone*: the patient has a heavy asthma attack;
- *Red zone*: represents an emergency.

[1] https://trentinosalutedigitale.com/en/

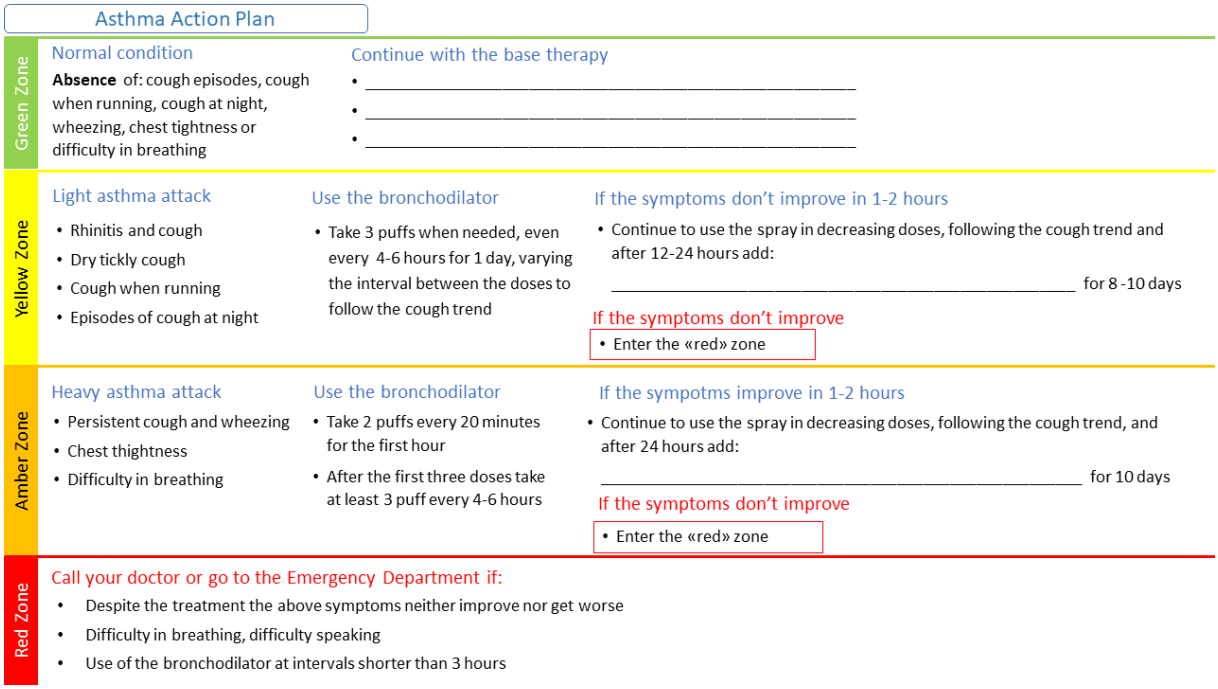


Figure 2.1: Asthma Action Plan.

Each risk group (i.e. zone) can be identified according to the *symptoms* that are reported by the patient. For example, *chest tightness* and *difficulty in breathing* might indicate that the patient is in an *Amber zone*. The decision on whether one symptom is enough to classify the patient or all should be confirmed is part of domain knowledge and should be informed by the experts.

Each classification is associated with different *recommendations*, which can be seen in the middle column of the Figure. The *Amber zone*, for example, tells the patient to "*use the bronchodilator*" and additional details are given next: "*Take 2 puffs every 20 minutes...*".

In the last column, we can notice that the plan presents two possible directions to be followed between 1 and 2 hours from the first recommendation. While a new *recommendation* is given to the patient in case the symptoms improved, the second option takes the patient to the red zone (i.e. emergency) if the symptoms did not improve. When the latter hap-

pens, the *recommendation* linked to the red zone should be followed, i.e. "*Call your doctor or go to the Emergency Department*".

In this research, we exploit how to deliver such health information by using a conversational paradigm. Therefore, the described scenario is exploited to build a multi-turn goal-oriented dialogue system to support patients affected by asthma. The resulting dialogue system should first hold a conversation session where it asks the patient about recent asthma symptoms. The goal of the dialogue is to reach a real-time classification of the patient's status and, then, provide a corresponding recommendation. To address the full action plan, the dialogue system should conduct a new interaction 1-2 hours after the first recommendation, following the guidelines of the action plan.

2.2 Goal-Oriented Dialogue Systems

Conversational agents or dialogue systems have been built with different purposes and they fall into two categories according to the purpose of the dialogue [59]. *Chatbots* are dialogue systems whose only goal is to informally chit-chat with a human on any topic for which training data is available. Chatbots usually cover multi-turn interactions that imitate a human-human conversation rather than participating and contributing to some purpose. *Task-oriented* or *goal-oriented dialogue systems*, on the other hand, conduct single or multi-turn dialogue interactions with the user in order to accomplish a task in the given domain (e.g. booking a hotel). To retrieve the necessary information to accomplish the given task, these systems usually conduct a *slot-filling* dialogue. As the dialogue explored in this work aims to achieve goals within the healthcare domain, this work follows the architecture of goal-oriented dialogue systems.

Most goal-oriented dialogue systems follow a standard "NL pipeline"

design, which is composed of three main components [129], namely:

- *Natural Language Understanding* (NLU) component, which receives the user input and recognizes an *intent*;
- *Dialogue Manager* (DM), which receives the *intent* and tracks the current state of the dialogue to decide the next *dialogue action*;
- *Natural Language Generation* (NLG) component, which converts the *dialogue action* to a human-understandable response that is given by the system to the end-user.

The focus of this research is on the DM component. The DM is responsible for the flow of the conversation in a dialogue system [75], i.e., it deals with the problem of selecting an *action* to execute in a given *state*. States are used to understand and track the current status of the conversation, i.e., to identify what has already been discussed and what should come next. In a DM, actions correspond to the exchanged utterances since computationally, a dialogue can be seen as a series of communicative acts [5] (dialogue actions) that are used to accomplish particular purposes (goals). Both the NLU and NLG components are out of the scope of this work and we refer the reader to [124] and [47] for surveys on those topics, respectively. Nonetheless, the approach developed in this research is aimed to be integrated into a suitable dialogue system architecture that provides the natural language components.

2.2.1 Types of Dialogue

To address scenarios like the one described in Section 2.1, this work covers two types of goal-oriented dialogues from the categorization proposed by Walton and Krabbe (1995) [121]:

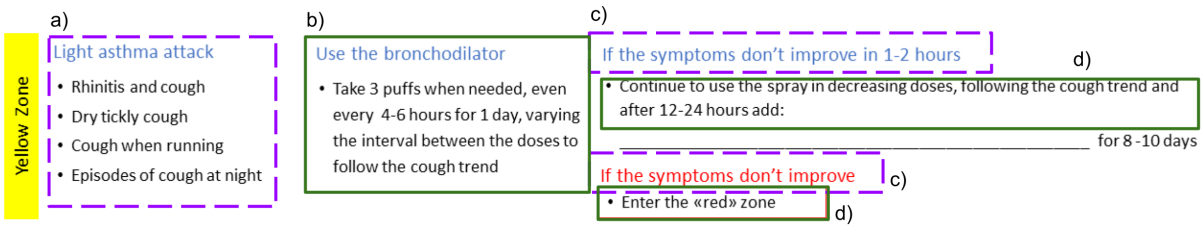


Figure 2.2: Sub-dialogues Identified in the Asthma Action Plan.

- *information-seeking*: in the initial situation the agent has none or little knowledge about the human agent's current situation and the dialogue consists of an interrogation with the purpose of acquiring knowledge to reveal a position (classification);

- *deliberation*: the previous dialogue revealed a situation that presents the need for acting and this new dialogue decides upon the actions to take, with the intention to influence the current situation.

In the asthma scenario, we identified the possibility of implementing 4 sub-dialogues: 2 of type *information-seeking* (dashed lines in Figure 2.2) and 2 of type *deliberation* (continuous lines in Figure 2.2). For readability purposes, Figure 2.2 highlights the sub-dialogues by taking only the *yellow zone* as an example. However, the description below is applied to the whole action plan.

Sub-dialogue #1 An *information-seeking* sub-dialogue (Figure 2.2-a) is conducted to acquire information on which symptoms the patient has. The goal of this multi-turn sub-dialogue is to classify the current situation, that is, to understand what type of asthma crisis the patient is presenting.

Sub-dialogue #2 Following the classification made in the previous sub-dialogue, a short *deliberation* sub-dialogue (Figure 2.2-b) is conducted with the goal of providing a recommendation to the patient. This recommen-

dation aims at changing the current situation, i.e. improving the patient's condition.

Sub-dialogue #3 The action plan defines that, for some zones, the situation should be re-evaluated after 1-2 hours. The way that the dialogue agent can learn this information is by asking the patient about improvements. Therefore, the third sub-dialogue consists of a short *information-seeking* sub-dialogue (Figure 2.2-c) that, similar to *Sub-dialogue #1*, has the goal of classifying the current situation. However, the possible classifications in this sub-dialogue are only two: (i) either the patient is still in the same condition or (ii) in case of no improvement, the most risky level should be classified, namely the *red zone*.

Sub-dialogue #4 The last sub-dialogue, a *deliberation* sub-dialogue (Figure 2.2-d), follows the classification of *Sub-dialogue #3*. Like in *Sub-dialogue #2*, this is a short sub-dialogue that has the goal of providing a recommendation to the patient.

2.3 Motivation

Dialogue systems can support healthcare from different perspectives. These systems can support not only patients but also physicians by delivering reliable information in an interactive and more natural way. However, health dialogues present more complex challenges with respect to the classic conversation paradigm. Some of these challenges can be observed in the asthma scenario described in Section 2.1. First, to avoid putting the patient in risky situations, an asthma monitoring dialogue system is recommended to provide information that is based on expert knowledge, like the action plan exhibited. A challenge also can be observed in the four

risk groups described, which present different urgency levels. To efficiently address this scenario, an asthma monitoring dialogue system should focus first on confirming or disregarding riskier groups. On one hand, this system must avoid automatic responses that may be valid, but not the most appropriate. On the other hand, the system must acquire enough information to reduce as much as possible its uncertainty of the situation. As a consequence of the latter, such systems tend to result in large dialogue policies, which are an additional challenge for dialogue managers. Finally, the health domain requires the adoption of verifiable agents, demanding approaches for dialogue management that are explainable and allow control of the dialogue policy.

Because of these and further challenges, current health dialogue systems are limited and lag behind traditional conversational domains. While the automated generation of traditional conversational domains has benefited from data-driven techniques, privacy constraints limit retrieving information from real-world health dialogues, limiting the adoption of these techniques. In addition, data-driven approaches face the challenge of not being fully capable to provide explanations for their decisions. Meanwhile, the generation of verifiable agents still relies on handcrafted dialogue policies, which result in quite limited dialogue systems. A technique capable of automating the generation of verifiable policies, that has been reported as a promising strategy for dialogue management is automated planning. However, modeling the behavior of the dialogue as a planning problem is a complex task. In addition to domain knowledge, it requires expertise in both dialogue modeling and AI planning, which is not frequent among dialogue authors, limiting the adoption of such a powerful approach.

This work exploits the automated generation of dialogue managers that handle goal-oriented dialogues for the health domain. To build efficient policies, we exploit the integration of AI planning with ontologies, a fur-

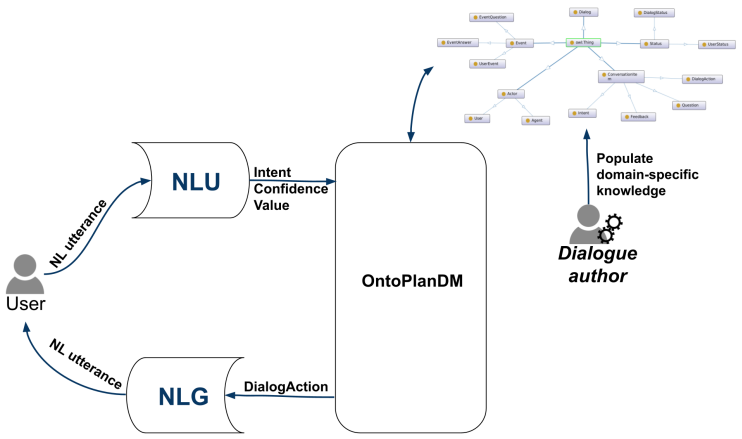


Figure 2.3: Relation of *OntoPlanDM* with the Natural Language Components.

ther mechanism to formalize the knowledge of a specific domain. Further techniques related to information management are addressed with the aim of delivering a reusable approach that is capable of addressing the needs of a health dialogue.

The proposed approach, which is named *OntoPlanDM*, relies on the traditional architecture of task-oriented dialogue systems, discussed in Section 2.2. Figure 2.3 positions *OntoPlanDM* within the standard "NL pipeline" design of these systems. As previously mentioned, this research focuses on the dialogue manager component, leaving the natural language layers (i.e. the layers that interact with the end-user) to external modules to be integrated during the development of the dialogue system. As can be seen in Figure 2.3, *OntoPlanDM* exploits an ontology that models the conversational domain, which is introduced in Chapter 5. This ontology is to be fed with domain-specific knowledge by the dialogue author and this knowledge can be shared among the three components (NLU, DM, and NLG). However, while the ontology is a key feature of the dialogue manager, its deployment into the natural language components (NLU and NLG) is a

design choice.

OntoPlanDM consists of an *offline* stage, which preprocesses the knowledge and prepares the dialogue system for execution; and an *online* stage in which the dialogue is conducted in practice. The following three modules, that are detailed in Chapter 7, compose it:

- *model acquisition*: this offline module specifies how domain knowledge can be acquired for the generation of a dialogue system in a new domain;
- *planning*: the planning module is part of both offline and online reasoning. Its role is the automated generation of the plan that will be adopted as the dialogue policy, i.e. the resource that determines the state-action pairs to be used during a dialogue session;
- *dialogue execution*: this online module is in charge of running the dialogue and interacting with the end-user.

This work intnds to cover complex scenarios where it is necessary to address the following aspects: (i) to manage long conversations, possibly having a high number of interactions; (ii) to keep track of users' status in order to send proper requests or feedback based on the whole context; (iii) to exploit background knowledge in order to have at any time all information about the domain in which the conversational agent has been deployed; and (iv) to plan dialogues able to dynamically evolve based on the information that has been already acquired and on the long-term goals of the dialogue. While this approach can be reused for different applications within healthcare, its focus is on health dialogue systems for domains that need to retrieve information in a slot-filling manner with the aim of classifying this information to further execute an intervention.

Chapter 3

State of the Art

This Section first discusses some of the particularities of the management of health dialogues, identifying the state-of-the-art techniques to address these dialogues. Then, some of the relevant literature on ontology-based dialogue management is discussed, followed by the discussion on approaches that integrate plan-based and ontology-based dialogue systems. Since an extensive study on plan-based approaches was conducted during this research, a separated chapter is dedicated to plan-based dialogue management (Chapter 4).

3.1 Health Dialogue Management

While dialogue systems have the potential to benefit health across different domain applications, the management of health dialogues is a complex process [97, 54, 100] and it is still a challenge to generate a dialogue system for a real-world health problem [65]. Health dialogues can benefit from many of the aspects that are also common to other domains, such as a *mixed-initiated* behavior that allows shared control over the dialogue [118, 80, 52] or implicit and explicit confirmation questions [129].

However, health dialogues present some additional challenges that require special attention. For example, these dialogues tend to result in huge

state spaces and, consequently, longer dialogues when compared to traditional domains [127, 98, 97, 15]. A reason for that is the large number of variables that might be required to fully understand a situation and take appropriate decisions. A health dialogue system cannot rely on automatic responses that may be valid, but not the most adequate, leading a patient to risky situations. The different flows that can result from the user's responses are another factor that significantly increases the state-space of the health dialogue [127]. As the value given to a slot affects the flow of the dialogue, the dialogue manager needs to adapt according to the received answers (e.g. slot-value triggers a possibly risky situation). Traditional dialogues do not face this problem, since they usually make use of the slot-values only to execute queries, being it enough to adopt some encoding that informs whether the slot-value is present or absent [24, 52].

Due to the existence of several health-related emergencies, *criticality* is an indispensable factor in a health dialogue [17]. A dialogue in a domain that may present emergencies must be able to handle this factor through the action selection, prioritizing questions that will collect the most useful information that make it possible to detect an emergency as fast as possible or even the need to obtain more details about the situation. Moreover, a health dialogue system must also address *expertise*, basing its reasoning on the knowledge of medical experts [16] in order to emulate the experts' behavior during the conversation (e.g. choice of questions) and reproduce the decisions that they would take for the patient. An expert-based dialogue system is not only accountable, but is also more likely to gain the trust of the patient.

Explainability is another concern for health dialogue agents. Having control over the system behavior and explainable systems' decisions (from the dialogue author's perspective) are interests of the health domain [98]. In addition, considering that misunderstandings may have critical conse-

quences in this domain, being able to provide the user with explanations that justify the system's decisions would allow the correction of information that was possibly misunderstood by the agent.

All these factors increase the complexity to be handled by the dialogue manager component. Automating the generation of health dialogue managers that are explainable and keep control of the dialogue policy is still an open challenge [98]. In fact, given these requirements, current commercial health dialogue systems are highly domain-specific and heavily based on hand-crafted rules and features [65].

The most common types of dialogue managers used in health dialogue systems are *finite-state* and *frame-based* [110, 65] approaches. These dialogue managers can be very efficient in simple dialogues whose goal is to fill a few slots. However, they have as common disadvantages the high human effort that is required to design new dialogues and the fact that they are not capable of processing complex and long multi-turn dialogues that require actions beyond slot-filling (e.g. *acknowledge, propose, assert*).

The generation of a goal-oriented dialogue agent to manage complex health dialogues is not trivial. Aiming to address complex health dialogues, several approaches have been proposed based on *data-driven* techniques [123, 36, 29], which commonly outperform hand-crafted dialogue managers [128]. Notably, dialogue systems that rely on reinforcement-learning techniques can be very efficient at learning policies from training data. However, besides data acquisition and annotation being a costly process, the health domain adds some extra complexity to it, mainly due to privacy constraints that limit retrieving information from real-world dialogues [98]. Another important limitation of these approaches with respect to the health domain is that they remove some of the control on the policy, not being fully capable of providing explanations of their decisions, a behavior that may not be desirable in the health domain [98].

A technique that provides verifiable agents and has been reported as a promising strategy for dialogue management [94, 32, 12, 67], being also investigated to address complex health dialogues [80], is *automated planning*. Since manually building extensive dialogue trees for complex health dialogues is time-consuming (and also may not be feasible), one of the main advantages of planning is the automation of this process. However, modeling a dialogue as a planning problem can be a challenging process that involves expertise in both dialogue modeling and AI planning, which might not be very common among dialogue authors. In fact, aiming to achieve efficient dialogue policies, state-of-the-art approaches for plan-based health dialogue still hand-code complex parts of the dialogue [54, 106, 22, 18, 103]. Chapter 4 discusses automated planning and presents an extensive review of recent plan-based approaches for dialogue management.

As previously discussed, health dialogue systems must rely on knowledge to deliver efficient agents that reduce risks to the end-user. For this reason, several approaches implement semantic-aware strategies that rely on ontologies to build such systems [60, 77]. The next Section discusses the use of ontologies in dialogue systems.

3.2 Ontology-based Dialogue Systems

An *ontology* is a formal and explicit specification of a set of shared concepts [23]. Besides modeling sets of concepts and their relationship within a domain of knowledge, ontologies are also used to describe and reason about the properties of the given domain.

The literature presents a huge number of works that rely on knowledge-based approaches to build conversational agents, either goal-oriented [3, 79] or not [112, 68, 78]. Ontologies can contribute to a dialogue with features such as the expansion of the number of concepts addressed by the dialogue

manager, without requiring the dialogue author to explicitly inform all the different alternatives. Ontologies also provide semantics and make it possible to reason on different concepts to infer new information. For example, by retrieving concepts a and b, it is possible to infer c, without explicitly asking the end-user for this information.

In the healthcare domain, ontologies have been employed for the management of dialogues on topics such as patient referrals [79], behavior change [18] or drugs prescription [99]. In most of these works, the contribution of the ontology is to feed the dialogue manager with domain-specific knowledge. That is, although the dialogue manager exploits the ontology to ground the conversation, the structure of the conversation (e.g. turn-taking, type of action to execute next) is defined by external strategies.

In a different way, the ontology introduced as part of this thesis, which is described in Chapter 5, models the conversational domain itself, aiming to provide semantics to the conversational scenario. Other relevant works that also have exploited the concept of ontology for modeling the conversational paradigm are described next (Section 3.2.1).

3.2.1 Conversational Ontologies

This Section identifies three relevant works that, to some extent, also have exploited the concept of ontology for modeling the conversational paradigm. They are summarized next:

- OntoVPA [125] is a powerful dialogue management system designed to be employed in virtual personal assistants (VPA). OntoVPA employs ontologies for both domain and dialogue representation. These ontologies are used to interpret the user input and to fill slots required to accomplish a user request. Meanwhile, response generation relies on ontology rules and reasoning. It is important to note that

VPAs mostly include question-answering scenarios that employ short interactions, with limited complexity (when compared to a healthcare scenario, for example). However, online response computation can become challenging for scenarios with high complexity.

- OwlSpeak [56] is a spoken dialogue manager that uses an ontology that models a task-oriented (spoken) dialogue. It keeps the domain-independent components of the ontology separate from domain-specific parts of the dialogue, being reusable to new conversational domains. OwlSpeak treats the dialogue state as a belief, for which it generates a VoiceXML snippet with the grammars that are expected to be recognized in this state. To select the agents' next move, on the other hand, OwlSpeak relies on a predefined agenda that connects all possible dialogue moves within an interaction.

- PHIDO [4] is an ontology that formalizes the structure for the communication of health information. With the aim of managing dialogues on patient counseling, the ontology was designed by fusing behavioral change and dialogue theories. Although the focus of the authors is to provide information on the human papillomavirus vaccine, PHIDO is generic enough to be reused in other health counseling topics. PHIDO treats dialogue interactions as small tasks to be accomplished. This way, each task has a sequence of speech acts (utterances), whose ordering is predefined as properties of the ontology. As PHIDO treats the dialogue as a discussion, it does not focus on achieving one main goal. Instead, it has some communicative goals and its focus is on delivering informative interactions that clarify users' concerns about the domain (vaccine).

The ontology provided in this work (Chapter 5) differs from the listed ones, mainly in the description of components that support the reason-

ing on the information provided by the user with the aim of inferring the current users' status to adapt the conversation accordingly. In fact, despite the research findings on this theme and the trajectory that shows a neat interaction between statistical inference approaches and ontologies for modeling the entire dialogue [41], there is a lack of a shared ontology. The proposed ontology aims to fill this gap.

Section 3.3 discusses works that integrated to some extent the two techniques exploited in this research, i.e. ontologies and automated planning.

3.3 Integrating Ontologies and Automated Planning

Plan-based dialogue systems are knowledge-based systems. The integration of ontologies into these systems can bring benefits such as (i) providing additional expressiveness for representing the conversational scenario as a planning problem or (ii) the possibility of decoupling planning variables from the information that is gathered during the conversation.

This Section introduces a few approaches that integrated, to some extent, planning and ontologies with different purposes. The focus of the Section is on the recent literature related to dialogue systems, which is presented in Section 3.3.1. However, to better define the approach proposed in this thesis, further approaches that exploited such an integration for systems beyond conversational agents were also analyzed. These approaches are presented in Section 3.3.2.

3.3.1 Approaches for Dialogue Systems

The literature has exploited the knowledge from domain-specific ontologies with different purposes. Common examples are (i) mapping the planned state to the real-world state, (ii) identifying the topic or context of the

dialogue, or (iii) specifying how domain-specific knowledge affects real-world states.

The work of Franzoni et al. [43], for example, proposes an approach to address emotional affordances in human-machine interactions. To achieve this goal, the authors model emotional affordances in an ontology and provide a framework that models aspects related to this topic as a planning problem. During execution, the planned tasks are executed according to the state that is recognized with the support of the knowledge from the ontology. In addition, a relation of time with the emotions supports the scheduling of the planned actions, that should respect the actual time that the emotion is identified. Galescu et al. [45], instead, rely on ontology concepts to understand the dialogue context. However, in this approach, the contribution of the ontology is directed to interpret the user input. After this interpretation, the plan-based agent that specifies the domain-specific behavior is responsible for handling the new dialogue state.

In Baskar and Lindgren [6], a domain ontology is exploited by the dialogue manager to provide information on the topic of the dialogue. A task manager is in charge of specifying a list of domain-specific tasks that will feed the planner. The planner is then responsible for determining the execution ordering of these tasks. Lee et al. [66] also rely on a domain ontology for the generation of a dialog support system. However, this ontology is used by the planner to learn which is the goal state of a detected situation and how the components modeled in the ontology affect the environment so an appropriate plan can be generated to change this situation. The generation of a planning problem relies on a framework that is defined as part of the knowledge used by their model. This framework is independent of the domain of the application and it is only during execution time that the domain-specific variables will be mapped to the plan.

The work of Behnke et al. [11] integrates the user through a dialogue on

a plan generation for a companion system. By relying on an ontology as a common knowledge source, separated models are automatically generated (and also extended) for the plan and the dialogue. In this process, ontology and planning aspects are translated through a mapping defined by the authors. The definition of predicates in the planning domain, for example, is mapped to pre-defined OWL properties. Meanwhile, arguments of a predicate can be obtained from annotations in the ontology. By keeping a shared vocabulary and model between ontology and planning problem, this model can reproduce possible domain updates and reduce costs on maintenance. A similar strategy is adopted in [102], that also addresses companion systems. This work exploits a conceptual model (ontology) to provide domain-specific knowledge and a planning model to address procedural knowledge. The planning model is automatically generated by exploiting the most generic concepts within the ontology. The integration of conceptual and procedural knowledge expands the possibility of delivering explanations of the resulting plans.

The ontologies used in the works listed so far represent the knowledge on the domain of the dialogue, but they do not formalize or represent the dialogue interactions themselves. In these works, dialogue interactions have been modeled and handled by additional components. An exception is the work of Bickmore et al. [18], that aimed at automating the generation of dialogue systems that focus on promoting health behavior change. In addition to domain knowledge on behavioral medicine, the ontology includes the description of a few components of the dialogue (e.g. DialogueAction) and a task model exploits the knowledge from the ontology to describe what steps should be taken by the system to achieve a goal. An automated planner interprets the outcome of the task model to plan a task decomposition for each segment of the dialogue. Although the authors have addressed reusability within their model, the focus of this approach

is very restricted to the behavioral change domain.

Section 3.2.1 listed a few ontologies that formalized the dialogue paradigm. However, we found no register of their integration within planning problems. To the best of our knowledge, no other work has combined an ontology for representing the conversational domain with AI planning techniques to support the automated generation of a dialogue manager.

3.3.2 Further Approaches

We were interested in the comprehenion of strategies that were employed by further approaches, i.e. not restricted to the dialogue domain, when integrating ontologies and planning.

One of the strategies identified consists in formalizing the planning technique as an ontology. For example, in [44], an ontology that formalizes HTN plans was proposed. The ontology makes it possible to automatically derive a planning domain and problem. The HTN formalization was provided in the ontology TBox. This way, it is enough to populate its ABox with domain-specific knowledge to generate HTN plans for different domain applications. The work of [14], that focuses on emergency scenarios, also formalizes some planning concepts in the ontology. Such a formalization makes it possible to easily translate the concepts into PDDL to generate a corresponding plan. However, a disadvantage of these approaches is that they require the knowledge engineer to comprehend automated planning to populate the ontology. That is, before instantiating the domain-specific knowledge, the knowledge engineer will have to think about how this domain can be best represented as a planning problem.

Some approaches rely on an external mapping of predefined components of the ontology into planning terms. This mapping can be exploited both ways, that is, either to generate a planning problem based on the ontology or to map planned actions to ontology instances. PORSCE II

[55], for example, is a system to perform automatic semantic web service composition that relies on an ontology that models web services. In this work, the knowledge is easily mapped into PDDL actions since the ontology already describes which are the expected inputs (preconditions) and outputs (effects) of each web service. Planned actions also can be easily mapped back to their corresponding web service during execution. A similar strategy was used in [131], that focuses on the automated construction of knowledge discovery workflows. Specific classes within an ontology that formalizes knowledge discovery are translated into planning domain and problem instances. For example, actions are translated from instances of *NamedAlgorithm*, while their preconditions and effects are generated from ontology properties called *input* and *output*, respectively. Resulting classical plans correspond to the expected workflows and they also can be employed to extract knowledge from the ontology. Such mapping was the most common strategy employed by the works described in Section 3.3.1.

Finally, an alternative is the specification of a planning domain that is independent of the ontology components. However, the knowledge from the ontology is exploited to instantiate the actual values of the planning variables. Different from the previous works that automate the domain instantiation, the work of [63] specifies an abstract planning domain according to the needs of the domain (i.e. e-learning). Then, an ontology that formalizes AI-related learning competencies is exploited for the instantiation of the planning problem. That is, the actual instances of objects, initial and goal states are retrieved from the ontology. A disadvantage of such a strategy is the maintenance aspect. As changes are not automatically reproduced when either the ontology or planning domain needs updates, there is an additional human cost to update systems deploying it.

Several of the works described in this Chapter served as inspiration or guidance during the definition of the approach proposed in this thesis.

Chapter 4

Plan-Based Dialogue Management

4.1 Introduction

Artificial Intelligence (AI) planning [50] consists of a process that selects and structures domain actions in some rational order aiming at achieving a final pre-stated goal. Considering that a conversation also aims to achieve a goal and that dialogue managers deal with the problem of action selection, AI planning is a promising alternative for dialogue management [94, 32, 67]. Planning for human-machine dialogue offers the possibility of reducing the cost of the dialogue by anticipating the outcome of each dialogue action to identify the states that could be achieved. This way, it is possible to choose the best path that leads to the goal of the dialogue. In addition, AI planning provides verifiable agents, and offers policy control without limiting the flexibility of the dialogue.

However, the development of methods that implement AI planning to structure and manipulate effective human-machine dialogue is still in its early stages. Indeed, when compared to other approaches for dialogue management, planning has been used on a quite smaller scale. Several may be the reasons for such low adoption; yet, no detailed analysis of the contributions and challenges of AI planning for dialogue management is available in recent literature.

This chapter presents a systematic literature review (SLR) in recent contributions of plan-based dialogue systems, with a focus on dialogue management. The intention is to provide the reader with a broad picture of the current state of the art in this field. To achieve this goal, the most recent works on the topic were identified and their data was extracted to identify the following aspects: the types of AI planning exploited for dialogue management; the planning characteristics that justify its adoption; and, the challenges posed on the development of plan-based dialogue managers. This SLR is available in [101].

The remainder of this chapter is organized as follows: Section 4.2 provides the overall definition of automated planning. Next, Section 4.3 presents the method used to select the papers and lists the research questions covered by this SLR. Section 4.4 discusses the process conducted to extract the works to compose the final corpus. Section 4.5 discusses the answers to the research questions. Then, Section 4.6 provides some further discussion on the topic. Finally, Section 4.7 concludes the SLR.

4.2 What is Automated Planning?

Automated planning is a sub-field of artificial intelligence [50]. The most common type of AI planning is the classical approach [39], which can be viewed as a search problem that describes the sequence of actions that must be applied in the world in order to achieve a goal. Planning considers that the world has different *states*. A state can be changed to a new one by the so-called *actions*, which have *preconditions* that must be met to be applicable to that state; and present *effects* that describe what changes in the world after its application, i.e. add or delete values. These actions and other universal aspects must be described in an instance of a *planning domain*. An instance of a *planning problem*, that is based on this domain

instance, must also be specified. This instance represents one concrete problem that specifies which is its *initial* and *goal* states. Both instances must be described in a language understandable by *automated planners*, like the Planning Domain Definition Language (PDDL). This way, the planner investigates all possible states that can result from the application of the available actions and a *plan* is a path (sequence of actions) that takes the initial state to the goal state.

4.3 Method

This work follows recognized guidelines for conducting a SLR. We refer mainly to the ones discussed by [61] and by [96]. In this section, we detail the steps that were conducted to accomplish the SLR.

Motivation and Study Goals

AI planning has been reported as an effective and efficient technique for dialogue management [94, 32, 67]. Nevertheless, it still has not gained due attention, presenting low adoption among current dialogue systems. A broad view of how planning strategies can be integrated within dialogue systems is missing in recent literature and the lack of such study led to an underestimation of the potentiality of such techniques. First, the lack of a comprehensive overview of what has been produced so far might limit the evolution of already defined approaches. This might result in the design of redundant strategies to handle issues that have been successfully covered by previous literature. Another consequence is the low adoption of this technique since a shared knowledge of its characteristics and advantages for dialogue management is not available. Such comprehension would make it possible for dialogue engineers to consider the adoption of AI planning on scenarios that require features such as predictability, which are more

effective on knowledge-based approaches when compared to data-driven ones. Finally, not knowing which are the challenges and open problems on plan-based dialogue management might lead to unexpected barriers to the development of plan-based dialogue systems.

The goal of this chapter is to provide a broad overview of recent contributions to plan-based dialogue management. This chapter extracts the planning techniques that have offered a great contribution to dialogue systems. It also to identifies the main characteristics of planning that justify its adoption in dialogue management. The comprehension of these characteristics is expected to instigate a further investigation of plan-based dialogue systems. Lastly, this chapter also identifies of the main challenges on the topic. The finding of such challenges is crucial to decide on the suitability of AI planning for a given dialogue problem, but also to support future work on investigating and overcoming them.

Next, the research questions that were elaborated to motivate this investigation are presented.

Research Questions

Starting from the described goals, three research questions were formulated to be answered within this review:

- **RQ1.** What types of planning have been used to support dialogue management in recent plan-based dialogue systems?
- **RQ2.** What are the most eminent planning characteristics that motivate the adoption of planning for dialogue management?
- **RQ3.** What challenges can be identified in recent plan-based dialogue systems?

The general questions above motivate discussions on the state of the art of plan-based dialogue management. *RQ1* seeks to identify which are the

planning techniques that were exploited in the retrieved works, motivating a discussion on the reasons for their adoption. By describing some planning characteristics that are expected on efficient dialogue systems, *RQ2* helps us to justify the application and benefits of AI planning to support dialogue management. Finally, *RQ3* is devoted to the identification of the main challenges that are still encountered in the development of plan-based dialogue managers.

Search strategy

To help answer the research questions, works were retrieved by using three strategies: (i) applying a search query to available scientific databases; (ii) conducting a manual search, i.e. by starting from the results of point (i), we extracted the most cited authors and checked if all their relevant works were already included; (iii) including additional works that did not appear in the results of the two previous strategies, but that were mentioned by the works retrieved by them.

To query scientific databases, four sources have been selected: *scopus*[1], *acm*[2], *dblp*[3], and *ieee*[4]. The decision on the selection of these repositories was made given their extensive coverage of well-established scientific literature in computer science. In addition, these repositories include works that have been published in peer-reviewed conference and workshop proceedings. As a search strategy, a query was formulated including keywords related to the purpose of this work and the research questions previously listed. The query is shown next:

```
(("planning" OR "plan-based") AND ("dialogue" OR "dialog"
OR "conversation" OR "conversational agents"))
```

[1] https://www.scopus.com/search/
[2] https://dl.acm.org/
[3] https://dblp.uni-trier.de/
[4] https://www.ieee.org/

EC 1	The paper is not available.
EC 2	The paper is not in English.
EC 3	The paper is a technical report.
EC 4	The paper is shorter than 4 pages.
EC 5	The paper was published before 2014.
EC 6	The paper does not cover either dialogue or AI planning.
EC 7	The paper does not address dialogue management.

Table 4.1: Exclusion Criteria.

The resulting query aimed at focusing on a specific portion of works that employ AI planning on dialogue systems. The conjunction "AND" has been employed to avoid the retrieval of works that exploit different dialogue techniques, such as data-driven approaches. The authors have considered expanding the query to include further terms related to dialogue, such as *argumentation* or *interaction*. However, the inclusion of these terms resulted in a huge set of non-related works, revealing a deviation from the focus of our search.

From the authors' previous knowledge of the works of some researchers in this field, several works that have exploited plan-based dialogue did not include the terms present in our query either in their title or abstract (e.g. [21, 33]). This way, we also opted to conduct a manual search on the bibliography of the most cited authors elicited in the previous step in order to include these and similar works. Finally, relevant works mentioned within the selected corpus were also considered for inclusion, as long as they meet our eligibility criteria for article selection and pass the quality assessment.

Article selection

To filter the retrieved works and to focus only on the ones that meet our main objectives, the exclusion criteria (EC) listed in Table 4.1 were defined.

Papers were excluded if they satisfied at least one of the exclusion criteria between EC1 and EC7. These criteria aim to exclude papers that are not available online, are written in some language other than English, are not scientific contributions, or that are too short to provide a complete approach. To keep the focus on the most recent contributions, we also excluded works that have been published before 2014. Papers that do not address either dialogue systems or automated planning are also excluded as they cannot contribute to the purpose of this review. To direct this review to dialogue management, we excluded works whose main focus is on natural language processing. Finally, we opted to not exclude works that referred to the same approach as those, eventually, presented slightly different planning techniques.

Quality assessment

To assess the quality of the investigated studies, we defined six criteria:

- **QA1** Is a well-defined methodology used?
- **QA2** Is the goal of the study clear?
- **QA3** Does the study discuss the state-of-the-art?
- **QA4** Is the approach applicable to different domains?
- **QA5** Does the study include a clear evaluation?
- **QA6** Does the study present an implementation or a clear illustrative problem of the approach?

Each paper was marked with one of three possible scores for each criteria (QA1-QA6): *Yes*, *No*, or *Partially*, which are weighted 1.0, 0.0 and 0.5, respectively. An exception was delineated for *QA5*, for which we defined the weights: 1.0 when an evaluation with real users has been conducted;

0.5 when the evaluation was conducted on synthetic scenarios (use cases); 0 when no evaluation was reported.

Data extraction strategy and analysis

Aiming to extract only works that can answer the research questions, the data was extracted by following three steps. First, all papers are evaluated against the EC by taking into account only their title, abstract and, keywords. Second, the works were carefully evaluated against the EC by taking into account their full content. Lastly, the proposed quality assessment was applied to the remaining papers, keeping only works that get a score equal to or above 3.0.

4.4 Results

This section describes the process of extracting the works for answering the research questions. The search has been conducted according to the methodology described in Section 4.3. The data extraction has been conducted between December 2018 and November 2021. To focus on the most recent contributions while obtaining a satisfying amount of works to answer our research questions, we have limited our results to works that have been published since the year of 2014. Works retrieved in our manual search and from the related literature were also limited to papers starting from 2014.

The search process is detailed in Figure 4.1. Initially, the search query retrieved a total of 631 papers. Given our query definition (Section 4.3), which avoids the retrieval of approaches that are not plan-based, it is not surprising that we started with a small number of works. As discussed by Cohen [32] and by Petrick and Foster [94], plan-based dialogue systems still have not received due attention from the research community.

Next, we applied EC1 to EC5, excluding 92 papers. EC6 and EC7 were

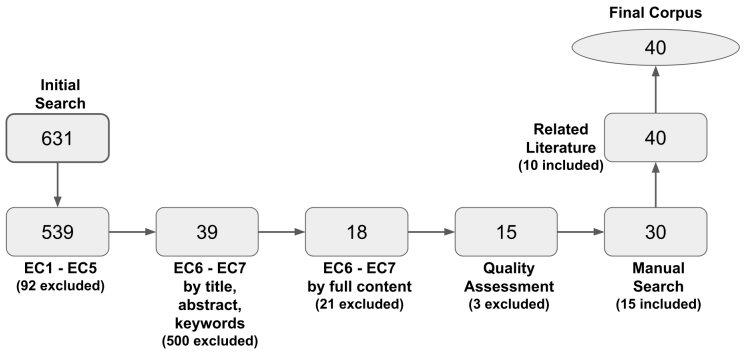


Figure 4.1: Papers selection process.

first used to filter the papers' title, abstract, and keywords. Since *planning* is a rather common word, the query returned several works that do not address the AI planning community but, instead, communities that use this term in different contexts (e.g. task or family planning) or communities that also may be able to conduct some kind of planning, such as machine learning. As those are not the focus of this survey, these papers were excluded. This way, EC6 and EC7 excluded an extra 500. At this point, 39 papers remained. For the remaining papers, we then applied EC6 and EC7 by analyzing their entire content, which excluded another 21 works. In both steps considering EC6 and EC7, manual analysis was conducted.

Following our quality assessment, we further eliminated 3 papers that did not reach the minimum expected quality (3 out of 6.5), described in Section 4.3. Our manual search included another 15 works, which correspond to meaningful works from relevant authors. These works were obtained after analyzing the profile of the most cited authors within the initial corpus. By meaningful, we refer to works that, after full reading, were considered relevant to the topic of this SLR.

Finally, after carefully analyzing the resulting works, 10 papers have been included in our corpus as they were mentioned by the retrieved works and offered a meaningful contribution. Both works from the manual search and works retrieved from the related literature meet all our selection cri-

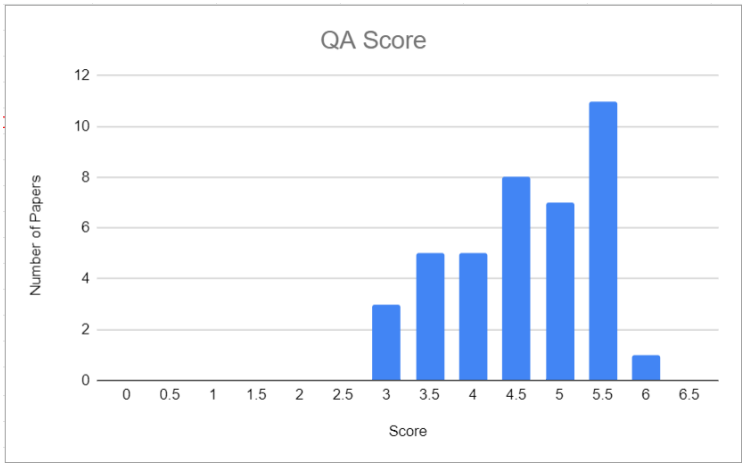


Figure 4.2: Quality assessment: total of papers grouped by score.

teria and our quality assessment. Figure 4.2 reports the total number of papers grouped by the score that they received during the quality assessment. The final corpus is composed of 40 works and it is presented in Tables 4.2 and 4.3, that also describes the papers' publishers.

4.5 Answering the research questions

This section discusses the answers to the RQs proposed in Section 4.3.

4.5.1 RQ1. What types of planning have been used to support dialogue management in recent plan-based dialogue systems?

A range of different planning techniques is available for plan generation and the choice of one or another varies according to the problem approached. In dialogue systems, examples of some aspects to be considered when choosing a planning approach are: (i) are actions deterministic, non-deterministic or probabilities are available for the outcomes? (ii) is it possible to subdivide the dialogue into sub-dialogues or smaller tasks to be accomplished?

Id	Reference	Published at	Publisher
P01	Lison (2014) [72]	IWSDS	Springer
P02	Black, Coles and Bernardini (2014) [20]	CLIMA	Springer
P03	Honold et al. (2014) [58]	IE	IEEE
P04	Black, Coles and Hampson (2017) [21]	AAMAS	ACM
P05	Panisson et al. (2014) [87]	ArgMAS	Springer
P06	Lee et al. (2016) [66]	PlatCon	IEEE
P07	Botea et al. (2019) [24]	DEEP-DIAL	AAAI
P08	Nothdurft, Ultes and Minker (2015) [86]	MMSYM	LiU Electronic Press
P09	Kominis and Geffner (2017) [62]	ICAPS	AAAI
P10	Nasihati et al. (2015) [83]	ICMI	ACM
P11	Biundo and Wendemuth (2016) [19]	KI-Künstliche Intelligenz	Springer
P12	Nothdurft et al. (2015) [84]	SIGDIAL	ACL
P13	Galescu et al. (2018) [45]	SIGDIAL	ACL
P14	Bercher et al. (2014) [12]	ICAPS	AAAI
P15	Geib, Craenen and Petrick (2016) [49]	ICAPS	AAAI
P16	Petrick and Foster (2016) [94]	AAAI	AAAI
P17	Foster and Petrick (2017) [42]	IWSDS	Springer
P18	Nothdurft et al. (2017) [85]	IWSDS	Springer
P19	Behnke et al. (2017) [9]	ICCT	IEEE
P20	Petrick and Foster (2016) [92]	PlanSIG	University College

Table 4.2: Final corpus of surveyed papers (part 1).

Id	Reference	Published at	Publisher
P21	Shams et al. (2016) [104]	IJCAI	AAAI
P22	Baskar and Lindgren (2014) [6]	PAAMS	Springer
P23	Behnke et al. (2017) [10]	Companion Technology	Springer
P24	Zhang and Stone (2015) [132]	AAAI	AAAI
P25	Lu et al. (2017) [74]	IROS	IEEE
P26	Morbini et al. (2014) [80]	IWSDS	Springer
P27	Marques and Rovastos (2015) [76]	HAIDM	HAIDM
P28	Franzoni, Milani and Vallverdú (2017) [43]	WI	ACM
P29	Cohen (2018) [32]	AAAI	AAAI
P30	Behnke et al. (2015) [11]	IJCAI	AAAI
P31	Garcia-Olaya et al. (2018) [46]	WAF	Springer
P32	Teixeira, Dragoni and Eccher [113]	ISWC	Springer
P33	Sreedharan et al. (2020) [108]	ICAPS	AAAI
P34	Muise et al. (2019) [81]	arXiv[5]	-
P35	Petrick and Foster (2020) [95]	Knowledge Engineering Tools and Techniques for AI Planning	Springer
P36	Cohen (2019) [33]	SIGdial	ACL
P37	Pardo and Godo (2018) [90]	Journal of Logic and Computation	Oxford University Press
P38	Behnke et al. (2020) [7]	ICAPS	AAAI
P39	Teixeira, Maran and Dragoni (2021) [114]	SAC	ACM
P40	Teixeira, Maran and Dragoni (2021) [116]	CBMS	IEEE

Table 4.3: Final corpus of surveyed papers (part 2).

(iii) are there efficient planners available for the chosen planning approach?

In answering **RQ1**, we obtain an overview of which planning techniques have been proven suitable to address dialogue management. It is important to note that we do not attempt to cover the whole variety of planning techniques, instead, we cover the ones identified in our corpus only. The planning techniques are listed in Table 4.4 and their description is summarized in the sequence. We would like to note that, for different reasons (e.g. planning was not the main focus of the approach or the paper presented a broad discussion on plan-based dialogue), some works have not specified a planning technique [86, 83, 19, 45, 92, 6].

Planning Technique	**Paper ID**
Classical	[20, 21, 62, 49, 94, 42, 104, 80, 76, 43, 46, 95, 33, 90]
Probabilistic	[72, 20, 21, 49, 132, 74, 80, 32]
HTN	[87, 66, 84, 85, 11, 7]
POMDP	[72, 9, 132, 74]
FOND	[24, 113, 108, 81, 114, 116]
Hybrid	[58, 12, 9, 10]

Table 4.4: Planning techniques.

Classical Planning

The most simple planning technique is the *classical* approach. Traditional classical planners generate (offline) a sequence of deterministic actions (each action has only one outcome) that lead to the goal (Figure 4.3).

A classical planning problem [40] is a 4-tuple `<A,O,I,G>` where A is a finite set of applicable actions, O is a finite set of domain objects, I is the initial state and G is the set of goal variables. Actions are in the form of `<Pre,Add,Del>`, where Pre stands for precondition, and Add and Del

[5]This work is currently under revision, but it was included in our corpus as it presents strong relevance for the domain.

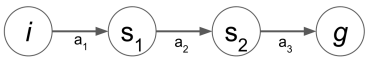


Figure 4.3: Rough representation of a classical plan. Nodes represent the states while edges represent the deterministic actions, being i the initial state and g the goal state.

stands for the predicates to be added and deleted as effects of the action execution. As can be viewed in Table 4.4, classical planning was the most common approach among the surveyed works.

Some works have supported classical planning with strategies that make it possible to adapt the plan according to the domain needs. Examples are: Franzoni et al. [43], that took into account emotional affordances in the planning model to prioritize actions, inform inconsistencies, and acquire some missing information; and Kominis et al. [62], that extended STRIPS with negation, conditional effects, and axioms to translate the dialogue problem with hidden initial state to a classical plan.

Some of the listed approaches, instead, do not focus on planning a dialogue itself but make an integration among planning and dialogue. For example, by evaluating goals and norms, the work of Shams et al. [104] reasons on the best plan and integrates this reasoning into a dialogue that provides explanations about its choice to the agent. In the collaborative approach of Geib et al. [49], after recognizing the plan (goal and subgoals) of the initiator agent, the supporter agent conducts a negotiation dialogue to define which actions it should execute. Only then, a plan is generated to organize the sequence among these actions. Also aiming to address a collaborative scenario, Pardo and Godo [90] propose an algorithm where multiple agents collaborate through dialogue to build a plan. In this work, the agents communicate relevant information to be taken into consideration when choosing an action for the plan.

Probabilistic Planning

The use of probabilistic reasoning was also frequent among the surveyed papers. Probabilistic planning is an extension of non-deterministic planning and it is used to address domains with some certain kind of uncertainty.

A probabilistic planning domain is a 4-tuple (S, A, γ, Pr) where S is a finite set of states, A is a finite set of actions, $\gamma : S \times A \rightarrow 2^S$ is the state-transition function, and Pr is the probability-transition function (in the form of $S \times A \times S \rightarrow [0, 1]$). Figure 4.4 shows a simple example of an action with a probabilistic outcome. That is, when action a is executed, the probability of reaching the state s_1 is 70% and state s_2 is 30%.

Besides being addressed in the effects of an action (e.g. a probability of occurrence is known to each of the different outcomes of the given action), probabilities can also represent the probability of success of a plan, guiding the search for a plan that maximizes the probability of reaching the goals. To some extent, an example of the former can be found in [80], that weights the information states with the probability that they can be reached. The latter, on the other hand, is the case of [21], where the plan *guarantees* a certain probability of success of a persuasion strategy.

The works mentioned above have handled probabilistic problems by using classical planning techniques (Section 4.5.1). Instead of classical planning, some works have managed the probabilistic dialogue problem by using a partially observable Markov decision process (POMDP) [28]. A POMDP supports the state transition based on observations of the world

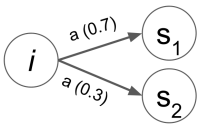


Figure 4.4: Simple representation of an action with a probabilistic outcome, being i the initial state, and s_1 and s_2 the possible resulting states after the execution of action a.

state and their probabilities. Such observations form a belief of the current state (i.e. partial observability) and POMDP is able to generate a policy describing which action is optimal given a belief state. For example, Lison [72] relies on probabilistic rules (which can then be converted to POMDP) for updating the dialogue state, determining the possible actions available at that state and the reward that can be obtained with the execution of an action (each effect is assigned a probability). Instead, the architecture proposed in [9] integrates a POMDP module to monitor the dialogue and identify when a planned action does not correspond to the action that was determined by the POMDP.

Fully Observable Non-Deterministic (FOND) Planning

When the domain presents some level of uncertainty and probabilities are not available or are not convenient, non-deterministic methods can be applied. In fully observable non-deterministic (FOND) planning problems, the actions may have a set of different possible effects. That is, non-determinism is represented in the effects of an action through the *oneof* feature, which indicates that one of the listed effects will be true on execution time. Formally, a FOND planning problem is a 4-tuple (P, O, S_I, ϕ_{goal}) where P is a set of Boolean state propositions, O is an operator set, S_I is the initial state, and ϕ_{goal} is the goal state set.

FOND plans anticipate all possible states that can result from the execution of a non-deterministic action. Figure 4.5 shows a simple example of an action with non-deterministic outcome. Note that, different from the probabilistic action illustrated in Figure 4.4, no probabilities are available for the outcomes. During plan execution, the agent is able to fully observe which effect has actually occurred in the real world.

FOND planning has been successfully adopted in [24], [81] and [108], which are related works. In these approaches, the solution to the FOND

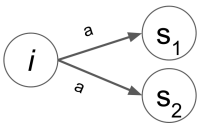


Figure 4.5: Simple representation of an action with a non-deterministic outcome, being i the initial state, and s_1 and s_2 the possible states after the execution of action a.

problem consists of a non-deterministic graph plan with actions to be executed in a dialogue system.

In [113, 114, 116], on the other hand, FOND planning is exploited with the aim of supporting dialogue management in health dialogues. To achieve this goal, domain-specific knowledge is translated into a planning problem. Each dialogue action is treated as a non-deterministic planning action. This way, FOND plans anticipate the different paths that the dialogue can take given the different answers that can be provided by the end-user.

Hierarchical Task Network (HTN)

Unlike classical planning that focuses on accomplishing goals, Hierarchical Task Networks (HTN) [37] aim to accomplish tasks. Tasks are high-level descriptions of some activity to be carried on and HTN provides methods for problem reduction. That is, *abstract tasks* are decomposed into subtasks until the achievement of a primary action that cannot be further decomposed, i.e. a *primitive task* (Figure 4.6).

An HTN problem is a 4-tuple (S_0, T, O, M) where S_0 is the initial state, T is a set of initial tasks which defines the goal, O is a set of operators that

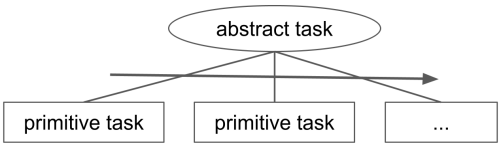


Figure 4.6: Hierarchical Task Network planning: representation of an abstract task and the corresponding primitive tasks achieved through the decomposition process.

define the achievable actions, and M is a set of decomposition methods.

Instead of a search through the state-space, HTN searches through the plan-space and its solutions are partial-order plans. In practice, HTN is one of the most used planning techniques and it was also frequent among the surveyed works. One reason for its substantial adoption within dialogue systems is that HTN assimilates to a human decision strategy, i.e. more abstract tasks are decomposed into smaller and easier actions to be accomplished.

An example of the use of HTN in a dialogue system is the work of Panisson et al. [87] that formalize HTN methods for an argumentative dialogue. By using an HTN planner, this approach can generate both: an immediate response or an optimal approach. Instead, the work of Nothdurft et al. [84] benefits from HTN planning to support the user to accomplish complex goals (tasks). In their approach, the HTN decomposition process of higher level tasks involves the end-user through a dialogue that asks the user to choose the sub-tasks.

Hybrid Planning

Hybrid planning integrates hierarchical actions that are partially adopted from HTN (Section 4.5.1) and Partial-Order Causal-Link (POCL) [13].

A hybrid planning problem is a 6-tuple $(V, N_c, N_p, \delta, M, P^i)$ where V is a finite set of state variables, N_c is a finite set of compound task names, N_p is a finite set of primitive task names, δ is a function mapping the task names to their preconditions and effects, M is a finite set of decomposition methods, and P^i is the initial plan. In a hybrid planning problem, both abstract and primitive tasks can contain so-called *causal links*. This way, a hybrid plan presents a similar structure to an HTN plan (Figure 4.6) with the difference that causal links will add some constraints on the ordering of the tasks (either abstract or primitive). For further details, we refer the

reader to [13].

Similar to HTN planning, this type of planning has been reported as well suited for user-centered planning applications [8]. This happens because, to some extent, hybrid planning imitates the strategies that humans use for solving problems, i.e., in a hierarchical (top-down) manner [26]. However, a further advantage of hybrid planning is that it allows some flexibility to explain its decisions as a consequence of relying on causal reasoning.

An example of its use in a user-centered application is given in [12] for a scenario that supports a user to assemble a home-theater. Hybrid planning contributes to the execution of actions that are semantically connected, instead of actions that reach the goal but in an order that does not make sense to the user (domain-specific knowledge was required). In addition, when differences in the expected state and actual state are detected, the use of causal links helps to identify if replanning should be conducted.

4.5.2 RQ2. What are the most eminent planning characteristics that motivate the adoption of planning for dialogue management?

The comprehension of AI planning characteristics is important to analyze whether this is a suitable strategy to be adopted for a given problem. Indeed, some planning characteristics, such as the possibility of explaining its action choices, are especially attractive to the dialogue community. This way, the aim of **RQ2** is to provide a discussion on the planning characteristics that received significant attention among the surveyed works.

Goal-Driven Plans

Similar to a conversation, a plan aims to achieve a goal. In planning for human-machine dialogue, by anticipating the outcome of each dialogue

action, a planner is capable of identifying the states that could be achieved and of choosing the best path that leads to the goal of the dialogue. In Section 4.5.1, we described the different types of planning approaches that have been used among the surveyed works to find a plan that reaches a goal in conversational scenarios.

Differently from traditional data-driven based dialogue systems, which frequently focus on achieving goals such as booking a restaurant or planning a trip [126], the plan-based surveyed works aimed at accomplishing a range of different goals that addressed tasks such as: supporting a user to assemble a home-theater [12]; building a workout plan [9]; a robot taking orders placed by human agents [132]; a bartender robot serving its customers [94]; car inspection [81]; supporting asthma patients [114]. Furthermore, to achieve the goal of the dialogue, we have observed the frequent use of some conversational strategies, such as collaboration [62, 45, 49, 76, 32, 90, 7], argumentation [20, 21, 87, 104, 90] or negotiation [43, 87, 49]. In fact, AI planning will always try to achieve a goal; the difference is in the strategy applied for it, which may vary according to the type of planning employed.

Classical planning presents a sequence of deterministic actions to reach the goal. However, instead of just selecting the shortest path to the goal, some of the surveyed works have exploited different strategies to define this sequence. For example, in [43], action selection is affected by emotional affordances that might prioritize or exclude some actions. In the problem exploited by Shams et al. [104], on the other hand, selecting the shortest path might not be the most convenient solution as some norms might influence action selection. Non-deterministic plans, instead, take into account the possible different outcomes that can result from an action execution. Probabilistic or POMDP planning can rely on probabilities to identify the best action towards the goal [72]. Meanwhile, as FOND planning treats all possible action outcomes equally and it will only know the result after

action execution, it must anticipate all possible paths to reach the goal [81]. Finally, descriptions of HTN and hybrid planning strategies to reach the goal are given in Session 4.5.1 and 4.5.1, respectively.

Among the surveyed works, only the work of Nasihati et al. [83] could not be categorized as goal-driven, but instead, as a chatbot [59]. Their system uses a dialogue manager that tries to engage infants to interact, maintain this engagement and promote responses from the baby with the aim of facilitating language (and also sign language) learning. The system does not have a final goal that must be achieved within a dialogue session. Finally, in some cases, the goal of the dialogue can change during a dialogue session. Lee et al. [66] exploited situation awareness to recognize possible changes in the goal of the dialogue and, if necessary, to build a new plan to accomplish the new recognized goal.

Addressing Large State Spaces

Traditional approaches for dialogue management, like finite state machines [59], require the dialogue author to handcraft the dialogue tree. Manually building extensive dialogue trees for complex dialogues is a time-consuming process. In addition, as the model size (i.e. number of variables, actions, states) increases, the complexity of the dialogue also increases, making this process error-prone and, in some cases, not feasible. Therefore, one of the main advantages of AI planning for dialogue is the automation of this process. As a consequence, plan-based policies can implement much larger state spaces when compared to traditional approaches.

It is important to highlight that the planning type employed in the dialogue manager directly affects problem and solution sizes. Probabilistic approaches, for instance, suffer from a limitation on the number of domain variables that are allowed in the model. To handle this limitation, the POMDP approach in [74] has relied on commonsense reasoning to simplify

and guide the dialogue manager. HTN planning, instead, is capable of reducing the problem into smaller tasks thanks to its decomposition feature. Therefore, instead of generating a single plan that includes large amounts of information, partial plans can be generated according to the progress of the dialogue [66]. Classical planning is also capable of addressing problem sizes that are very difficult to be solved by humans. An example is the approach of Black et al. [20], which presents a solution for simple persuasion dialogues that present considerable complexity for hand-crafting. In further work [21], the authors showed a satisfactory plan generation time for argumentative dialogues with up to 13 arguments. While this does not seem a very huge problem size, the authors have verified that the approach covers realistically sized dialogues. FOND planning is another good candidate to handle huge state spaces. In [113], that specifies a health dialogue as a FOND problem, the authors showed how a very simple scenario can result in large state-spaces but that can be efficiently handled by a FOND planner. In a more recent work [116], the authors show the suitability of the approach to generate dialogue policies on real-time scenarios, addressing up to 160 slots. Botea et al. [24], that also exploited FOND planning, demonstrated results of a synthetic scenario in which the solution size remained, in most cases, 4 times the model size. In some extreme cases, it grows up to 16 times the model size, without compromising the efficiency of the plan generation. In a further work [81], the authors showed satisfactory results for plans addressing up to 28 variables and resulting 482 nodes. Such problem size is intractable for human beings.

Although several data-driven approaches are also capable of generating dialogues with great complexity, these approaches require large amounts of training data, being limited to the capabilities of user simulators. In addition, updating the model to include extra variables in the dialogue means that more data must be provided for training.

It is important to note that the specification of the planning problem will affect the planning solution. With the aim of obtaining reasonable solutions for complex dialogues, aspects such as abstraction and the type of planning chosen for the problem must be carefully addressed for the dialogue domain. These aspects can be considered challenges in plan-based dialogue management and they are better discussed in Section 4.5.3. Finally, although planning is capable of implementing larger state spaces than several other techniques, a limitation with respect to scalability has been reported by some approaches and it will be discussed in Section 4.5.3.

Explainability

Some problems require the implementation of agents that can explain their behavior [32]. In recent years, explainable AI planning (XAIP) has received substantial attention [27, 1, 109, 64]. Explainability is a feature of AI planning that can be exploited for all types of planning and it can be provided from two perspectives: the developer and the end-user. From the developer's point of view, planning is explainable in the sense that it is possible to debug the resulting tree to understand each step that leads to the goal achievement. Consequently, planning makes it possible to control the generated policy and to keep track of the behavior of the agent. Regarding the end-user, planning also makes it possible to provide explanations that justify its actions choice. This aspect is especially relevant if we take into account that an optimal solution provided by the planner can still differ from the user's mental model, that is, from an expected course of actions that is based on the user's domain knowledge. When no further explanation is provided in such situations, the plan solution might result in some confusion on why the system is taking a given action, endangering the user's trust in the system.

Among the surveyed works, explainability has been exploited in AI plan-

ning with aims like (i) explaining generated plans [86, 104], (ii) explaining the choice of an action [84, 11, 12, 58, 87], or, less frequently, (iii) explaining the unsolvability of a model [108]. A quick description of the use of explainability in these works is given below.

In Nothdurft et al. [86], the authors investigate appropriate interaction strategies for a proactive plan-based dialogue system. The authors conclude that a system should attempt to explain or justify its decisions to the user to align the perceived mental model to the actual system model. Later, in [84], the authors implemented proactive as well as requested explanations in a plan-based dialog system. The proposed approach builds an exercise plan together with the user, for which explanations can be given on each step, helping the user to understand why an action is being chosen and what should be changed if this is not the desired choice. In addition, as the plan is built dynamically with the user, when dead-ends are found and the planning system needs to roll back some previously made decision, this decision is also explained to the user. In [11], a domain ontology is used both for plan generation and to enrich explanations on the plan steps.

The approach applied in Bercher et al. [12] and Honold et al. [58] deal with plan repair. As changing a plan during its execution might cause confusion to the user, the authors identified the need of providing explanations when requested by the user. By using formal proofs, which are translated with a template to a user-friendly language, their approach generates explanations on *why* a given step is present in the plan (e.g. it is a precondition for a further system action). Causal links that are used to build the plan, also support the generation of the explanations.

To improve the agent's chance of success in finding the best plan to reach the desired outcome within a negotiation, the agent proposed in [87] justifies its actions (negotiation stance) with the arguments that were used to choose this action. Arguments have also been used for generating ex-

planations in the work of Shams et al. [104]. In this normative approach, where norm compliance is weighted against goal achievement, the agent explains why a given plan was chosen as the best one for execution.

Regarding explanations on model unsolvability, Sreedharan et al. [108] aimed at supporting dialogue authors during model acquisition with explanations that help them to understand why the designed model cannot be solved. The purpose of such explanations is to support authors in finding a fix to arrive at the solution.

Handling uncertainty

Among the types of planning presented in Section 4.5.1, probabilistic (Section 4.5.1) and FOND (Section 4.5.1) planning are the most appropriate types to address domains that present some level of uncertainty and for which it is not possible to define deterministic models. However, it is important to note that handling uncertainty in a dialogue problem is not limited to these approaches. In fact, the surveyed works exploited planning in different ways to address uncertainty, being it handled differently according to where it occurs in the dialogue.

When the uncertainty is in the initial state, for example, exploring all possible alternatives for the truth initial state might be necessary. However, as reported by [62], this strategy increases the plan space and might result in scalability problems (Section 4.5.3).

A few different strategies have been applied to address uncertainty on the current state of the dialogue. Bercher et al. [12], for example, rely on a *probability* distribution over the possible world states. A quasi-deterministic model that takes into account the probability that a variable last had is applied to any variable that has not received an updated explicit observation during the last interaction. Lison [72], instead, encodes the dialogue state as a Bayesian Network. The algorithm proposed by

the author relies on high-level probabilistic rules that are based on prior domain knowledge. Instead of probabilities, *non-deterministic* approaches have also been adopted to represent uncertain states within the dialogue. In Botea et al. [24], non-deterministic planning was used to anticipate the different outcomes that can result from an action execution, predicting the possible resulting states and, therefore, the possible dialogue paths. Muise et al. [81] introduce a *determiner* that is in charge of identifying which non-deterministic outcome has actually occurred on execution time. Handling uncertainty on the current state has also been addressed with the use of *replanning* (Section 4.5.2). In Garcia et al. [46], a plan is built by assuming that each action execution will result in the most likely effect. When this is not the case, replanning is conducted.

Uncertainty may also come from the module that precedes the dialogue manager, i.e. the language understanding module. The uncertainty on the information received has been handled with confidence values [9, 93, 132]; properties like the *badASR* property in [93] that indicates low-confidence automatic speech recognition; or with fluents like the MAYBE-* fluent adopted in [24] to identify when a piece of information is uncertain. These strategies instigate the adoption of clarification questions, instead of the reproduction of a previously made question.

Additional techniques have been used to handle uncertainty in other parts of the plan-based dialogue. Common sense, for example, was exploited in the POMDP approaches discussed in [132, 74] to reduce the uncertainty in action outcomes. Aiming to handle the agent's uncertainty on its opponent's beliefs, Black et al. [20, 21] used conformant planning for compiling the uncertainty away and model the domain as a classical plan. Other works that have addressed uncertainty to some extent are [76, 42].

Mixed-initiative

Mixed-initiative is a feature that is not a direct characteristic of AI planning, but that can be implemented and exploited by a plan-based dialogue manager. Mixed-initiative can be achieved with all types of planning and there are different possibilities to implement it; examples are: dialogue actions [108], predicates that work as flags to switch the initiative [80] or through replanning [95].

In the surveyed works, we could extract two types of mixed initiative: mixed-initiative planning (MIP) systems and mixed-initiative dialogues.

The purpose of MIP systems is to collaborate with the user for decision making aiming to obtain high-quality plans. To achieve this goal, the user itself is included in the planning process. In MIP, the system first interacts with the user to define a goal understandable by the planner. In the sequence, the interactive planner applies a strategy to refine the tasks that are required to accomplish this goal and it decides whether to include the user in each decision step. The MIP approach in [84] (also discussed in [85, 9, 10]), for example, includes the user in the process of generating a workout plan. Together with the user, the dialog system, that relies on HTN planning, decomposes the abstract actions into executable sub-actions. This approach reduces the cognitive load on the user to find a solution while building a plan that includes the user's preferences. However, although the plan generation is part of the MIP system, the dialogue that requests the choice of the next plan action, is system initiated.

Mixed-initiative dialogues, on the other hand, are able to support dialogue interactions that are initiated either by the system or by the user. Systems that implement mixed-initiative dialogues are able to keep a more natural conversation when compared to system-initiated ones, that present lower complexity. As previously mentioned, AI planning supports mixed-

initiative and this feature has been exploited by some of the surveyed works in different ways.

The approach of Galescu et al. [45], for example, supports mixed initiative among multi-agents that address collaborative problem solving. Since constant initiative switches can significantly increase the complexity of the dialogue, to facilitate and order the conversation among the agents, the initiative switch is restricted by an act that is called by the dialogue manager only when there are no further tasks pending or in progress.

The dialogue manager proposed by Morbini et al. [80], on the other hand, takes the initiative when it is necessary to obtain some information from the user, but it is also able to respond appropriately to user-initiated utterances. This behavior is achieved through the use of a precondition that informs on which initiative each operator can be activated. For example, operators with the precondition for system initiative mean that this operator can be activated at any moment in the dialogue, while, a precondition for user initiative means that this operator will be used to handle some user-initiated input.

Sreedharan et al. [108] addressed mixed initiative through the specification of a dialogue action to respond to user-initiated utterances. In addition, in their non-deterministic approach, every system action has an additional outcome to switch initiative based on the user input. In [81], mixed-initiative has been addressed similarly.

Finally, a common approach in classical planning to address user-initiated actions, i.e. actions that were not expected within the plan in execution, is replanning. This strategy has been adopted to address mixed-initiative in the bartending scenario developed by Petrick et al. [95]. Replanning is discussed in Section 4.5.2.

Dynamic policy through replanning

Handcrafted dialogue trees limit the flexibility of the dialogue in the sense that the dialogue must stick to a predefined path and, most times, ignore any information that is not expected at the current state. AI planning, on the other hand, presents the possibility of dynamically updating the dialogue policy through replanning.

Replanning can be conducted to all types of planning and upon different circumstances. Goal change, for example, requires a new plan to be built. In the work of [66], dynamic plans are built according to the current situation. When the situation changes, the goal also changes. Therefore, the existing plan must be abandoned and a new plan is generated to address the newly detected situation.

Execution failure also can be managed with replanning. An example is the approach proposed by [76], where each agent has a plan and tries to collaborate with the other agent by the means of a dialogue. When the agent's communicative plan fails (e.g. a belief about the other agent was wrong), it immediately updates its beliefs and replans. In the approach of [46], besides replanning if some error occurs during execution, replanning is also conducted if some unexpected event modifies the planned flow and invalidates the next planned action. In [116], instead, in case of a non-expected input, replanning can be conducted without affecting the ongoing dialogue session. Similarly, in [80], if a received event cannot be handled by the active operator, the dialogue manager simulates future dialogues that would result from the activation of each available operator and selects the most promising operator to handle this event.

In the bartender agent discussed in [95], the plan is also dynamically updated through replanning in a few situations: (i) on action execution failure or a not understood response (low confidence score); (ii) on over-

answering, i.e. the user provides more information than what was asked at that state; and (iii) when the bartender agent notices a new customer that was not present during the generation of the current plan. In this last case, the new plan consists of an extension of the existing plan, where actions that were in progress are maintained and new ones are added to the flow.

Replanning also enables the handling of large state spaces, which can be divided into smaller problems, feasible for runtime generation. This way, after reaching a milestone, a new plan can be built to address new variables. In the online approach proposed by Kominis and Geffner [62], for example, plans are built with a single action and, after executing and observing this action's outcome, replanning is conducted by taking into account the updated knowledge. This strategy is aimed to address multi-agents that collaborate to achieve a common goal.

Horizontal Ingredients

Besides the characteristics discussed above, two further ingredients received significant attention in the surveyed works. These ingredients are not direct features of AI planning, but they are worth some discussion. The first one is the integration of *ontologies* as a knowledge source for the planning problems. The second, the *adaptability* of the dialogue policies. Both aspects are discussed below.

Ontology Several of the surveyed works have exploited the integration of ontologies within their approaches [43, 113, 11, 84, 10, 85, 9, 6, 45, 66]. These integrations have already been discussed in section 3.3.

Adaptability Adaptability is a concern mainly of cognitive or robotic systems and it has been exploited by several of the surveyed works to different extents. Examples of adaptability include: adapting plans to user's

priorities or preferences [6, 12, 84, 9, 11, 7], to dynamic changes in the environment [12, 43, 72] or to the dialogue history [84, 9].

To address adaptability some of these works relied on ontologies, replanning, or online planning [72]. The first two aspects have already been discussed in this Section. Concerning the latter, by considering that adapting a policy once it has been calculated can be quite difficult, online planning is a suitable strategy to address dynamic changes in the environment. Online approaches can more easily adapt to the actual state of the dialogue and plan the next action accordingly. The drawbacks rely on the limited time available for planning as these approaches plan at execution time and, therefore, must meet real-time constraints.

4.5.3 RQ3. What challenges can be identified in recent plan-based dialogue systems?

There are several open problems in planning research [48] and some of them are also extended to plan-based dialogue management. Motivated by **RQ3**, this Section provides a discussion on the main challenges that could be identified in the surveyed works.

Dialogue Modeling

Modeling the behavior of the dialogue as a planning problem is a complex task and it involves expertise in both dialogue modeling and AI planning. The automated generation of efficient policies depends on the specification provided; a poor specification will result in poor policies. Indeed, some authors [94, 32] recognize that research on plan-based dialogue is still in its early stages and some works [81, 46] highlight that plan-based dialogue requires careful analysis on domain modeling to overcome challenges on the representation of complex aspects that concern the dialogue domain in the planning problem (e.g. initiative-switch). By analyzing the surveyed

works, we were able to identify that modeling a dialogue as a planning problem can be challenging from two perspectives: (i) modeling decisions, i.e. which strategies are the most appropriate to generate the expected policy; and, (ii) model acquisition, i.e. how a new planning problem can be generated for a new dialogue domain.

Modeling decisions Considering the different range of dialogue applications that can be instantiated, it is not possible to define a single strategy that would work for all of them. Therefore, decisions on how to model the information as a planning problem must be carefully analyzed according to the purpose of the dialogue application and to what is expected from the resulting agent. In fact, the complexity of the policy might increase according to the modeling strategy adopted. Nesting non-deterministic effects within a FOND planning problem, for example, can result in a longer search time [81]. Meanwhile, several levels of decomposition tasks in an HTN planning problem will give origin to longer policies, which might or might not be desired in the given domain [87].

The *type of planning* employed is, indeed, one of the aspects that must be taken into account when modeling a plan-based dialogue. Planning types employed in the surveyed works have been discussed in Section 4.5.1. Nonetheless, we would like to note that the planning type selected for the dialogue problem might affect the size of the model. Model size, that can be measured by elements such as the number of actions and variables addressed, is a concern of several authors [11, 81, 66, 46] since it may affect the quality of the generated plan and impact the plan generation time, constraining real-time scenarios. HTN approaches, for instance, are able to address relatively large state spaces. POMDP approaches, on the other hand, must restrict the number of variables to remain efficient. Meanwhile, with a declarative specification of the problem, the FOND approach in [81]

showed efficiency for problems with a significant number of both actions and variables. In general, it is possible to identify that significant improvement on the model size can be obtained by abstracting the problem.

The *abstraction level* implemented in the planning problem presents significant relevance. Abstraction can reduce scalability problems, as discussed in Section 4.5.3, and it may also influence the quality of the generated plan. This aspect has been confirmed by some of the surveyed works. An example is given by Garcia et al. [46], that compared a unified and more abstract domain with simplified (less abstract) specific domains. The authors identified that the former was able to find plans much faster than the specific ones.

With the aim of abstracting the domain and reducing the search space for complex and especially huge domains, an efficient strategy might be the adoption of HTN planning. As discussed in Section 4.5.1, HTN generates an abstract plan with abstract actions and relies on subsequent decomposition methods for generating an executable plan, i.e. a plan without abstract actions [12, 10, 66]. However, hierarchical decomposition is not suitable for every domain and, like other aspects, it must be analyzed for each case scenario.

Interesting abstraction strategies can be found in argumentative approaches like [20, 87]. In these scenarios, actions are modeled with a high level of abstraction as the content of the arguments is not relevant for ordering the actions. A great advantage of these models is their reusability in quite different argumentative scenarios.

Another possibility is to abstract the values given to the variables since, as discussed in [24, 81], anticipating all possible values for a variable may become intractable and, in several cases, knowing whether the variable has a value or not is enough to proceed with the next action selection. Of course, such strategy does not apply to domains where these values

influence the next action.

Another aspect to be considered while modeling a plan-based dialogue is the decision on how to *handle errors* and *non-expected states*. Some discussion on this aspect was given together with the discussion on replanning in Section 4.5.2.

In general, most of the surveyed works presented a satisfactory level of *generality*, meaning that the modeling decisions adopted in their definition are capable of addressing different dialogue domains without requiring a complete redefinition.

Model acquisition By analyzing the surveyed works, it is possible to note that model acquisition has not received as much attention as modeling decisions have received. Although automated policy generation can result in richer policies when compared to handcrafted dialogue trees, handcoding the planning problem (or parts of it) for every new dialogue domain may still imply high costs for building a new dialogue system and, considering that not many dialogue authors have knowledge on automated planning, this is likely a reason for the low adoption of such a powerful approach.

In fact, to guarantee reliable policies, some approaches rely on human specification (handcoding) of the whole planning domain or of complex parts of the dialogue. Examples of the former can be found in [95], where the actions were crafted for the bartending domain; and in [83], that uses specific actions to be executed by an avatar/robot for multimodal interactions with infants. Morbini et al. [80], on the other hand, have focused on supporting efficient policy authoring. Aspects such as mixed initiative and topic selection (through a reward function) are automated. However, this approach requires the dialogue author to craft the complex parts of the dialogue (e.g. specific moves within a subtopic) and the authors still report that significant human effort was required for modeling new dialogues.

In the work of Behnke et al. [11], ontology reasoning is used to automatically derive further decomposition methods for hierarchical tasks. However, the approach requires as input an initial planning domain already containing a description of the domain problem. In [114, 116], an ontology is translated into a planning problem. The ontology models the domain-independent knowledge that is common to any goal-oriented dialogue system. Therefore, when generating a new dialogue system, the domain-specific knowledge must be populated into the assertional box of the ontology. The approach of Leet et al. [66], instead, relies on a domain ontology as the knowledge source for decomposition tasks. However, too few details are given on how this decomposition is accomplished.

Some works like [76, 21, 62, 45] specify sets of communicative acts (including their preconditions and effects) that address some specific type of dialogue (e.g. negotiation, argumentation). The modeled actions can be applied to different domains that implement these types of dialogue by loading the variables specific to the conversation topic (e.g. arguments).

Finally, a work that has paid significant attention to model acquisition is the work of Muise et al. [81]. By using an interface for declarative dialogue design, dialogue authors are in control of the behavior of the agent, being able to edit it as desired. Although the proposed approach is capable of automatically generating complex FOND problems for dialogue, dialogue authors are not required to understand AI planning to build a new agent. With the aim of improving explainability during the model acquisition process, concepts from XAIP have been integrated into this approach in [108].

Scalability

Although automatically generated plans can address significantly larger state spaces if compared to handcrafted techniques, scalability restrictions

for large problems is still a topic that gets some of the attention of the AI planning community [48]. As a dialogue can take several paths to achieve its goal, such limitation might also constrain plan-based dialogue managers when dealing with nontrivial dialogues. Indeed, early approaches proposed for plan-based dialogue suffered from the limited performance of the planners available by then. Currently, the amount of information that can be encoded in a plan-based dialogue may vary according to the coding scheme of the dialogue acts and to the planning approach employed.

Limitations on offline approaches are common since these models must compute a whole policy offline. For some planning approaches, this process might become impracticable for large state spaces. Given their uncertain nature, probabilistic planning approaches, for instance, allow the modeling of only a small number of variables in the state space to ensure high-quality policies. An alternative to deal with this limitation and improve the plan performance on probabilistic spoken scenarios was presented in [132] and in [74]. The method employed in these works exploited commonsense with the aim of reducing the number of possible worlds, so a POMDP solver is able to calculate accurate action policies with less uncertainty and at a reasonable time.

In some offline approaches like [20] and [21], that addressed classical planning, the size of the problem and the search space grew exponentially with the number of domain variables, resulting in a concern on *memory usage*. Instead, the offline approach in [24] was able to minimize the scalability issue by modeling the domain variables with a high level of abstraction. The declarative representation employed in this approach benefited the model scalability, as reported in a scalability analysis that was conducted for a synthetic domain. In a more recent work [81], the authors exhibit a further scalability analysis that shows an exponential scale-up on the size of the generated dialogue graph with respect to the declarative

specification. The FOND planner applied has proven to be very efficient in computing the solutions, taking satisfactory generation time. The same planner was used in [116], also delivering feasible results.

Alternatively, some online planning approaches, i.e. approaches that plan only the next action on runtime based on the current state, have been proposed to avoid scalability problems. These approaches are very convenient to dialogues in open-ended domains that change constantly (e.g. robotics) and that require a fast policy update. For instance, by using high-level probabilistic rules based on prior domain knowledge, the online algorithm proposed by Lison [72] aims at speeding up the action selection process. Unfortunately, when dealing with complex problems, the author reports that the proposed approach is not able to scale to real-time requirements (*runtime performance* limitation). Another online approach was proposed in [62], where an online algorithm aims at finding an action sequence for multi-agent problems with a hidden initial state. Although this algorithm shows efficiency for problems with tens of possible initial states, a scalability limitation is still reported for larger problems.

Dynamic planning through replanning (Section 4.5.2) is also an alternative that can minimize scalability issues for dialogue approaches. With replanning, it is possible to avoid the anticipation of everything that can go wrong during a plan execution and, consequently, keep a smaller search space. As an example, for better scalability (among other motivations), Garcia et al. [46] have opted for planning strategies that apply corrective actions instead of relying on probabilities for every action outcome. By *repairing* a plan in case of execution failures, the approach in [12] is also likely to avoid or minimize scalability problems.

To sum up, offline approaches can address a greater number of domain variables when compared to online approaches. However, they suffer from some limitations concerning policy updates and adaptation; it is harder

to change a policy already built. Meanwhile, online approaches restrict the number of domain variables, aiming to address runtime processing. However, as reported in the works mentioned in this section, the plan generation time still requires improvement to address real-world problems.

In general, recent works have shown improvement with respect to scalability problems, but their applicability to domains different from the proposed ones still has to be further explored. This way, how to better exploit current planning techniques to achieve better scalability in conversational scenarios opens the interest to new research perspectives.

Learning over time

Learning from experience is an active topic being exploited over different research fields. While for some risky domains like healthcare, to keep a stable and well-defined problem definition is preferred, some systems, like the ones implemented in companion robots, might become more interesting to the end-user if new actions and behaviors are learned over time.

As previously discussed in Section 4.5.2, plan-based dialogue systems are capable of implementing dynamic policies through online planning [72, 62] or replanning. However, in both cases, the updated policy is limited to a previously specified planning problem and, unless an external learning module is integrated into the system architecture, planning models do not address learning. As a consequence, learning new actions or states from experience remains an open challenge for plan-based dialogue. Possible research directions to overcome this challenge include the integration of planning with reinforcement learning techniques [91].

Evaluation

The evaluation of dialogue systems is well-known as a challenging and subjective task [35]. Similarly, evaluating plan-based dialogue managers is a

challenging task as no benchmarks nor official protocols are available. Making direct comparisons over different dialogue management techniques can be infeasible as their different nature and purposes would make the comparison unfair. Indeed, just a little more than half (52%) of the surveyed works have conducted some type of evaluation.

Among the works that evaluated the proposed strategies, it was possible to identify two types of analysis: with real users or with user simulations. For the latter, interaction errors or uncertain responses were also simulated with the aim of better reproducing real scenarios [46, 74]. Some works have focused on evaluating *technical planning aspects* [20, 21, 11, 46, 81, 108, 116]. Among them, we identified the aspects listed in Table 4.5 as the most relevant measured ones. Some derivations of these metrics were given by the comparison of the *model size* with respect to the *plan generation time* [21, 20, 46, 81, 108, 116], and of the *solution size* with respect to the *model size* [24, 81, 116]. Although the other works might have discussed one or more of these aspects, no evaluation was provided. This brings to attention the need of designing evaluation protocols for validating both the effectiveness and efficiency of plan-based dialogue systems.

A few works, instead, conducted *subjective* evaluations that concerned *dialogue quality* or *user satisfaction* [12, 85, 84, 80, 7, 114]. In these works, questionnaires have been submitted to the users after a certain period of using the system. The results were either compared to different system

Metric	**Paper reference**
average search time	[46, 11, 20, 21, 81, 116]
success rate/probability of success	[74, 21]
number of actions in the executed plan	[46]
number of times it is necessary to replan	[46]
question-asking cost	[74]

Table 4.5: Metrics used for evaluation.

settings (e.g. [12]) or a different version of the same system (e.g. [80, 7]).

Finally, rather than evaluating resulting plans or dialogues, Nasihati et al. [83] have focused on evaluating the effectiveness of their approach with respect to the designated application purpose, i.e. behavior change.

4.6 Discussion

This study has revealed some interesting aspects regarding the research on plan-based dialogue systems. First of all, the relatively low number of works that were retrieved reveals that this topic is still under-investigated. Although our search was limited to works dated since 2014, a greater number of works was expected. In fact, it has already been brought to light by other authors that research on plan-based dialogue systems is still in its early stages [94, 32]. Plan-based dialogue requires further research to overcome challenges such as the representation of complex aspects that concern the dialogue domain in the planning problem. Cohen [32] also highlights the fact that current approaches have yet to be improved to handle dialogues able to keep the context of the conversation and unexpected initiative switches. Such capabilities can be implemented on plan-based approaches. However, as could be observed in most of the surveyed works, they have not yet reached such a maturity level. Furthermore, trending topics in dialogue systems such as empathy [71], emotional information or sentiment analysis [111, 130], and continual learning [73] have yet to be exploited by the community.

Regarding the application domains, we observed that plan-based approaches follow a different line when compared to traditional reinforcement learning approaches, which frequently focus on addressing problems like booking trips or restaurants [91, 129]. Among the surveyed works, most present domain-independent approaches and just a few addressed a strat-

egy that was very specific to the proposed domain [94, 10, 83]. However, some lines of research have gained greater emphasis. They are: *robotics* [72, 83, 49, 94, 42, 132, 74], *companion* technology [58, 86, 19, 84, 12, 85, 9, 10], *cognitive* systems [6, 43, 11], and *healthcare* [6, 80, 46, 114]. A possible reason for that can be associated with the fact that such systems, especially companion, cognitive and health systems, are expected to be predictable to gain the user's trust. Explainability, as discussed in Section 4.5.2, is also a factor that motivates the adoption of planning in such systems. Meanwhile, robotics have long exploited the adoption of AI planning in different contexts [69, 50].

Another important aspect that came to our attention is the lack of a common strategy to evaluate and compare the proposed approaches. As discussed in Section 4.5.3, several works lack evaluation at all and no pattern could be identified among the ones that have conducted some kind of evaluation. This way, we highlight the evaluation of plan-based dialogue systems as an open topic for future research. It is necessary to find ways to evaluate aspects such as the quality of the resulting policies, costs of building plan-based dialogue managers, and costs of re-planning a policy. In addition, some comparisons to different techniques for dialogue management, such as finite-state or data-driven approaches, would provide a more clear picture of the actual contribution of plan-based approaches. Another aspect that would reinforce the quality of plan-based dialogue systems is the user's satisfaction with resulting dialogues. As discussed in [84], dialogue systems are user-centered applications and, building systems that do achieve the goal, but in ways that might confuse the user, may reduce the user's trust in the system. This might have as a consequence the user abandoning the system, especially for domains in which the user has a choice on whether to use the system or not.

Finally, this study presents some limitations. First, the time constraint

defined for the retrieved works might have left out some relevant literature and authors. However, as stated in the goals of this SLR (Section 4.3), our focus was on discussing the latest contributions only. The search query employed might also have limited this SLR to some extent. The inclusion of additional keywords, as previously discussed, would result in a greater number of works. To handle this threat and to try to not leave out some interesting contributions, we manually searched for further works on the topic. However, we are aware that some relevant papers might still be left out. As a last limitation, we highlight possible biases in the selection of the papers and possible imprecisions on the data extraction. These aspects might have emerged as a consequence of the subjectivity of the analysis carried out and due to the authors' previous knowledge on the topic.

4.7 Conclusion

This Chapter presented a systematic literature review of the recent advances on plan-based dialogue managers. This SLR addressed research questions concerning (i) the types of planning recently exploited for dialogue management; (ii) the planning features that present significant relevance for a dialogue manager; and, (iii) the challenges on the development of plan-based dialogue managers.

The analysis conducted in this SLR reveals that the amount of works that have addressed plan-based dialogue management in recent years is considerably small; but these works have addressed relatively complex problems, not being limited to simplistic dialogues (e.g. single-turn interactions). Moreover, by analyzing which planning aspects have been exploited in the retrieved works, it was possible to identify the contributions of AI planning on the generation of robust policies. Some examples are: automated action selection towards a goal, dynamic update of the

dialogue policy, and explanatory dialogues.

On the other hand, relevant challenges like scalability limitations on complex dialogues, dialogue modeling, and the lack of a shared evaluation strategy were identified. This opens up an opportunity for further research efforts to overcome these aspects.

This SLR also discussed some of the application domains within the analyzed works. This analysis revealed that plan-based approaches tend to address goal-oriented domains, but that follow a different line with respect to traditional domains commonly exploited in data-driven approaches (e.g. hotel booking). Robotics, companion, cognitive and health systems were among the most frequent domains in the analyzed corpus. Finally, the limitations of the SLR were discussed. They include the time constraint that was set for the retrieved works (works starting from 2014), limitations in the search query employed, and possible biases in the papers selection or data extraction. These limitations leave the opportunity for an extension of this work, which could consider further aspects that are relevant to the community.

Chapter 5

Convology

5.1 Introduction

An *ontology* is a formal and explicit specification of a set of shared concepts [23]. Ontologies model sets of concepts and their relationship within a domain of knowledge, giving semantics to the concepts within this domain. In addition, ontologies are also used to reason about the properties of the given domain and, possibly, infer new knowledge from the available information. In the literature, several approaches have implemented semantic-aware strategies that rely on ontologies when the system's outcomes might imply some risk to the end-user, like in the healthcare domain.

Together with recent advances in conversational agents, several works have highlighted the need to equip these agents with background knowledge in order to improve their overall effectiveness and efficiency. The interplay of background knowledge with semantics is expected to increase the capacity of a conversational agent, enabling the deployment of more robust systems that are capable of keeping a structured conversation.

This chapter introduces the CONVersational ontOLOGY (Convology), a top-level ontology that models the goal-oriented conversation scenario. Convology, which is represented in the OWL language, supports the modeling of a full (multi-turn) dialogue between a user and a system. The aim

of Convology is twofold: (i) to semantically model and describe the concepts of the dialogue and (ii) to aid the understanding and management of the entire dialogue workflow. This way, Convology can aid both the development of conversational agents and real-time dialogue interactions.

This chapter first describes the process employed for the construction of Convology (Section 5.2). Next, the description of Convology 's main components is provided (Section 5.3). Section 5.4 describes how Convology was instantiated into a real-world scenario, namely a conversational agent for supporting patients affected by asthma. Finally, in Section 5.5, the Chapter is concluded and opportunities for future work are discussed.

5.2 The Construction of Convology

Convology is the result of collaborative work between the Fondazione Bruno Kessler (FBK) with the LINKS Foundation [1]. The collaboration is part of the "Trentino Salute 4.0"[2] framework promoted by Trentino's local government with the aim of providing smart applications (e.g. intelligent chatbots) to citizens for supporting them under different perspectives (e.g. monitoring of chronic diseases, promoting healthy lifestyles, etc.). One of the goals of this framework is to promote the integration of artificial intelligence solutions into digital health platforms with the long-term goal of improving the life quality of citizens. The presented ontology is part of the core technologies used in this framework.

The development of Convology followed the need of providing a metamodel able not only to provide a representation of the (goal-oriented) conversational domain but, also, to support the development of smart applications enabling access to knowledge bases through a conversational

[1] https://linksfoundation.com/en/

[2] https://trentinosalutedigitale.com/en/

paradigm. Such applications aim to reduce users' effort in obtaining information. Therefore, Convology has been modeled by taking into account how it can be extended to be integrated into real-world applications.

The process for building Convology involved knowledge engineers with competencies in dialogue systems and natural language understanding. Since Convology was developed in the context of a project that is related to healthcare, as previously mentioned, health experts (i.e. pulmonologists and psychologists) also participated in discussions related to the development of a conversational agent for the asthma domain. This way, these experts also provided insights on the modeling of Convology. The definition of Convology followed the METHONTOLOGY [38] methodology and this process is summarized next.

Specification. Convology specifies a full human-machine dialogue, being described from the conversational agent's perspective. From the granularity perspective, Convology is modeled with a *low* granularity level; it contains only top-level concepts that represent the main entities to describe a goal-oriented conversation.

Knowledge Acquisition. The acquisition of the knowledge to build Convology was split into two phases: (i) the definition of the terminology box (TBox) containing all the mandatory entities needed to represent conversations, and (ii) the definition of an assertion box (ABox) focused on a specific domain (i.e. asthma), used to validate the Convology itself. While the TBox has been modeled by the *modeling team* having competencies both in knowledge representation and natural language understanding, knowledge defined within the ABox is to be acquired through collaborative work with domain experts. This way, the TBox can be used to store information related to the conversation and also be exploited for reasoning purposes. The rationale behind this choice is to avoid changes in the TBox when Convology is instantiated into a new domain. Thus, when a

new application is developed, the experts in charge of defining all entities involved in the conversation will work only on the ABox, being guided by the TBox concepts.

Conceptualization. The conceptualization of Convology was split into two steps. The first one was covered during the *knowledge acquisition* stage, where most of the terminology has been collected and directly modeled into the ontology. The second step consisted in deciding how to represent, as classes or as individuals, the information we collected from unstructured resources. Then, we modeled both the object and data properties used to support all the requirements. During this stage, we relied on several ontology design patterns (ODP) [57], like the logical patterns *Tree* and *N-Ary Relation*, the alignment pattern *Class Equivalence*, and the content patterns *Parameter*, *Time Interval*, *Action*, *Classification*.

Integration. Although it is not addressed by the TBox, upon an application development Convology opens the possibility of being integrated with external ontologies. A conversational agent supporting people affected by asthma, for example, could align instances of the *Slot* concept with concepts defined into an external medical knowledge base like UMLS[3]. This way, individuals defined within the ABox of Convology may work as a bridge between Convology and the Linked Open Data (LOD) cloud.

Implementation and Documentation. Convology is represented by using the Terse RDF Triple Language [4] (Turtle) language in order to provide a formal representation enabling the check of inconsistencies, the visualization of ontology structure, and the ease of maintenance. The creation and the editing of the ontology are demanded by the Protege tool [5], while the exposure of the ontology is granted by the services available from the Convology website. Since Convology is not available for business purposes,

[3]https://www.nlm.nih.gov/research/umls/
[4]https://www.w3.org/TR/turtle/
[5]https://protege.stanford.edu/

it is licensed under the Creative Commons Attribution-NonCommercial-ShareAlike 4.0[6] and it can be downloaded from the Convology website[7]. Convology is constantly updated due to the project activities that use the ontology as a core component.

The documentation of Convology was generated by using the LODE[8] system and it is available on the ontology website.

Evaluation. The evaluation procedure was conducted by one knowledge engineer and one domain expert who did not participate in the modeling process. They were asked to instantiate Convology within the asthma domain by providing all the domain-specific knowledge (i.e., questions, intents, relevant user status, etc.). The main points of the evaluators' feedback are reported below:

- *Completeness*: for the TBox[9], the evaluators agreed on the completeness of Convology and the lexical representations of its concepts.
- *Accuracy*: within the scale *high*, *good*, *discrete*, *poor*, Convology was judged as *good*.
- *Conciseness*: all axioms included are relevant to the domain and no redundancies were found.
- *Coherence*: little bias in the documentation containing the informal description of the concepts and their formalization was found.
- *Adaptability*: *extension* of the ontology was positively judged as it did not require removing any axiom. However, updates are not always monotonic (e.g. if a user corrects some information, some *UserEvent* individuals may be removed). In any case, *consistency* is preserved.

[6]https://creativecommons.org/licenses/by-nc-sa/4.0/
[7]http://w3id.org/convology
[8]http://www.essepuntato.it/lode
[9]Evaluation of the ABox should be provided for each scenario in which Convology is deployed.

Finally, to verify the *computational efficiency* of Convology, together with the evaluators, we observed how the ontology behaved in the modeled scenario. Indeed, Convology does not contain axioms with a high level of complexity that would represent a criticism for reasoners. On the contrary, the ontology is thought to be used, mainly, to collect user input. Reasoning activities may be demanded by external libraries that perform SPARQL-based reasoning on more complex data and, in turn, update the ontology with new knowledge concerning the status of active conversation sessions.

5.3 Inside Convology

Convology contains six top-level concepts: *Actor*, *ConversationItem*, *Dialog*, *DialogType*, *Event*, and *Status*. These concepts represent likewise branches describing the main elements composing complex conversations. In total, the current version of Convology contains 41 classes and 53 properties (38 object and 15 data). Figure 5.1 shows the concepts defined within the ontology and the subsumption relationships among them.

Since Convology is aimed to be deployed into a server where it can be assessed by several instances of a dialogue agent at the same time, the concept of *abstract* and *session* instances has been adopted for the classes whose individuals are updated at runtime. In particular, the classes that adopt these concepts are: *Intent*, *Slot*, *Subdialog*, and *UserStatus*.

In the next Section, by starting from each top-level concept, we detail each branch of the TBox of Convology. The ABox is discussed in the sequence (Section 5.3.2).

5.3.1 The Terminology Box of Convology

This Section provides the semantic meaning of the most important entities of the TBox of Convology.

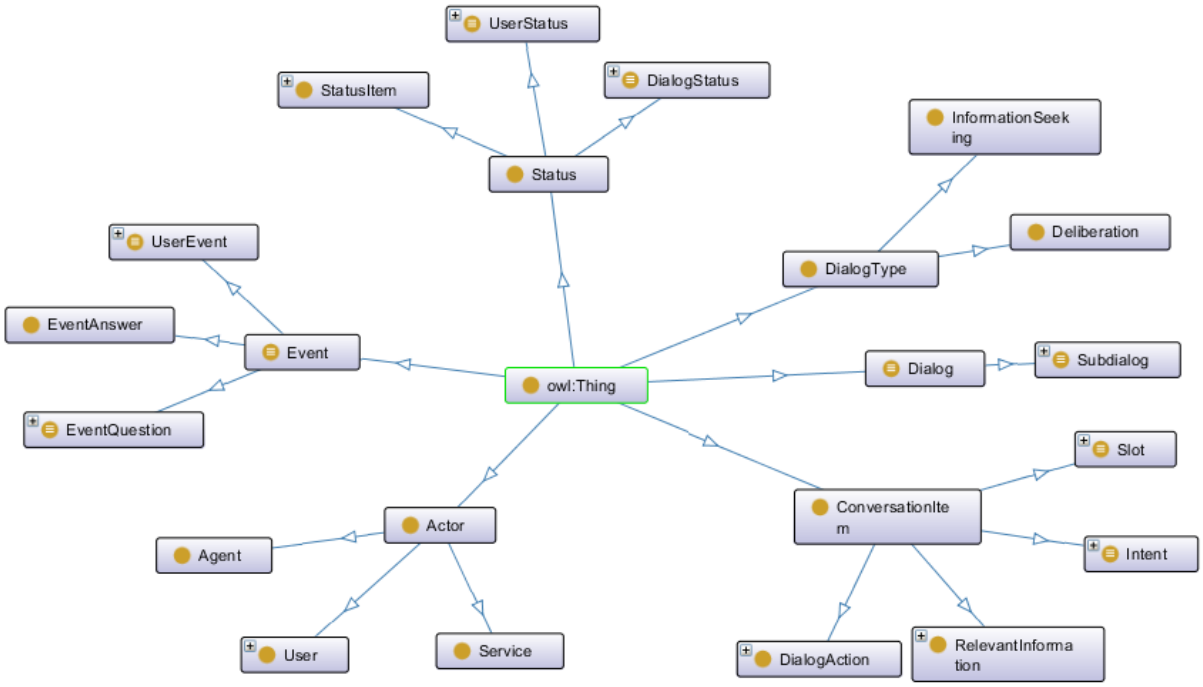


Figure 5.1: Overview of Convology.

Actor The *Actor* concept defines the different roles that can take part in a conversation. Within Convology, two main roles are represented by the subconcepts *Agent* and *User*. When Convology is deployed into an application, instances of the *Agent* concept represent the agents involved in conversations with the users adopting the application. Instances of the *User* concept, instead, represent the actual users that are dialoguing with the conversational agent. Convology maintains a single instance of *User* for each deployment in order to track conversations from the user's perspective, supporting domains that need to trace the whole history of a user. This strategy supports the development of robust conversational agents that are capable of referring to information provided in a previous dialogue session. Such an implementation would avoid repetitive conversations that request information previously discussed or even exploit previous user's preferences before making a recommendation. Of course, an analysis of further costs

related to the implementation of such a strategy would be required.

Dialog and DialogType The *Dialog* concept works as a collector of other concepts, representing a multi-turn interaction between a *User* and an *Agent*. A new instance of the *Dialog* concept must be created when a new conversation session starts. The *Dialog* instance can have its goal identified through the object property *hasGoal*. This information can be exploited by the application implementing Convology for decision-making or reasoning purposes. Furthermore, when the goal of the dialogue is achieved, the property *hasGoalAchieved* should be instantiated. Both properties, *hasGoal* and *hasGoalAchieved* expect an instance of *Status* as their range.

The *Dialog* concept subsumes the *Subdialog* concept. *Dialog* was defined to be exploited as the main class that connects smaller dialogue interactions, so-called, sub-dialogues. Instances of *Subdialog* must be defined by the dialog designer, according to what is expected from the agent. When several instances of *Subdialog* are defined, the property *triggers* must be defined indicating an ordering among them. Further instances (e.g. *Event*) are associated with the *Subdialog* instance through the property *refersToDialog*. Through this association it is possible to track all interactions made during a session. Moreover, *Subdialog* instances inherit the *hasGoal* and *hasGoalAchieved* properties, which should be exploited by the application.

The property *hasDialogType* must be defined for each instance of *Subdialog*, representing the type of dialogue addressed by this instance. The current version of Convology includes two types of dialogue: *InformationSeeking* and *Deliberation*, which are defined as subclasses of *DialogType*. The definition of these types of dialogue was discussed in Section 2.2.1.

ConversationItem A *ConversationItem* is an entity taking part in a conversation and that represents relevant knowledge involved in each interac-

tion. Within Convology, four subclasses of the *ConversationItem* concept were defined: *Slot*, *RelevantInformation*, *DialogAction*, and *Intent*.

The *Slot* concept represents pieces of information to be filled during a dialogue session. *Slot* instances are part of domain-specific knowledge and, therefore, each conversational domain will have different instances. The possible instances must be defined by the dialogue designer before the conversation starts. The value attributed to each instance, instead, will be acquired through the dialogue.

The value given to a *Slot* is represented by the concept *RelevantInformation*. Each *Slot* instance must be associated with instances of *RelevantInformation* by the *hasAllowedRelevantInformation* object property. This association indicates which are the accepted values for the given slot. In addition, the *hasNLURelevantInformation* object property can be instantiated when the NLU component identifies a new value for the slot; and the property *hasRelevantInformation* was defined to hold the actual value set to the slot after the state tracker confirms the NLU output. Note that the value associated with *hasRelevantInformation* should be among the instances defined in *hasAllowedRelevantInformation*.

The *DialogAction* concept (Figure 5.2) is related to actions performed by an *Agent* individual, representing the next message sent to a *User*. The sending of a *DialogAction* is handled by events, described later in this session. In the current version of Convology, *DialogAction* subsumes two types of actions: *Feedback* and *Question*, which must be instantiated as part of domain-specific knowledge. A *Feedback* individual represents a simple sentence that an *Agent* can send to users and for which it does not expect any reply. A *Feedback* can be used to close a conversation as the result of the reasoning process (i.e. when the goal is achieved), to answer a user's request, or simply to send single messages to users without requiring any further interaction (e.g. follow-up messages). An

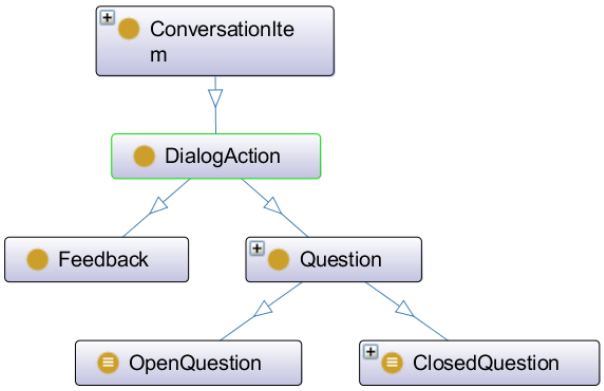


Figure 5.2: Overview of the DialogAction concept.

individual of type *Question*, on the other hand, represents a question that an *Agent* can send to a *User* and for which a new *User* interaction is expected in the sequence. This concept subsumes two further subtypes: (i) *OpenQuestion*: an open question for which the possible answers are not predefined; (ii) *ClosedQuestion*: a question with predefined expected answers. The expected answers to a *ClosedQuestion* are specified in the form of *Intent* instances through the *hasRelevantIntent* object property. Such *Intent* individuals are expected to be recognized in subsequent user input. In an application deploying Convology, this association favors the interpretation of the user's answer and the dialogue state recognition.

An *Intent* represents a piece of relevant information that (i) the reasoner is able to process and that (ii) is recognized by the NLU module from a natural language input provided by a *User*. The output of the NLU module must match an *Intent* among the list of all *Intent* individuals that the agent is able to handle. This list must be previously defined as part of domain knowledge. *Intent* instances can be recognized from the content of: (i) a user-initiated utterance or (ii) a user's answer to a *Question* made by the *Agent* in a previous turn. With the aim of addressing domains that exploit partial observability and to support follow-up questions that aim at confirming some information, a *hasConfidenceValue* object property

can be associated with each *Intent* individual. The confidence value can be exploited by the application by the implementation of additional techniques to either recognize or disregard the NLU output (e.g. by exploiting the last two interactions to try to match the received intent). Finally, *Intent* instances are associated with *Slot* instances through the property *hasSlot*. This means, that pieces of relevant information can be obtained from the user's input.

Event The *Event* concept describes a single event that can be performed by an *Actor* during a conversation. Instances of *Event* can either be created on runtime or they can be predefined as part of the domain knowledge and then recognized during the dialogue session. The property *hasDialogAction* can be instantiated to associate a *DialogAction* instance to the *Event* instance. Within Convology, we identified three kinds of events: *EventQuestion*, *EventAnswer*, and *UserEvent*. Instances of these concepts enable the storage of information within the knowledge repository, trigger the execution of the reasoning process, and allow the retrieval of information for both analysis and debugging purposes.

An *EventQuestion* represents the fact that a question was made in the dialogue. The property *hasQuestion* identifies the *Question* instance that the event refers to. In addition, instances of this concept are associated with further knowledge that identifies the timing that the event occurred (*hasTimestamp* data property), the *Actor* instance that sent the question (*performedBy* object property), and the *Actor* instance that received the question (*receivedQuestion* object property).

On the contrary, the *EventAnswer* concept represents an answer provided by an *Actor*. The timing information associated with individuals of this concept is defined through the *hasTimestamp* datatype property, while the sender and the receiver are defined by the *sentAnswer* and *re-*

ceivedAnswer object properties.

A *UserEvent*, instead, represents an *Event* performed by a specific *User*. The purpose of having a specific *UserEvent* concept instead of inferring *UserEvent* objects from the *EventQuestion* and *EventAnswer* individuals is that a *UserEvent* does not refer only to questions and answers but also to other events that can occur. An example is the presence of one or more *Intent* instances that are recognized within a user-initiated utterance. The relationship between a *UserEvent* and an *Intent* is instantiated through the *hasRecognizedIntent* object property. Finally, instances of *UserEvent* can trigger specific instances as the result of the reasoning process. Triggering events are instantiated through the *hasTriggerQuestion* and *triggers* object properties. While the former allows to put in a relationship a *UserEvent* with a *Question*, i.e. a specific *UserEvent* triggers a specific *Question*; the latter associates a *UserEvent* with a specific *UserStatus*. Such triggers can only be implemented when the *UserEvent* instances are predefined as part of the domain knowledge.

Status The last branch of Convology has the *Status* concept as top-level entity. Instances of the *Status* concept can be associated with the dialogue goal, as previously mentioned. This branch contains concepts describing the possible statuses of the user, through the *UserStatus* and *StatusItem* concepts, and of the dialogue, through the *DialogStatus* concept.

The *UserStatus* concept, whose instances are part of domain-specific knowledge, represents the possible status of a *User* that can be discovered by means of a conversation. Each *Subdialog* can be associated with different instances of *UserStatus* (object property *refersToSubdialog*). This concept is relevant mainly to conversational agents that need to understand the user's current situation to make a decision based on it. An example of its application is the asthma scenario described in Section 5.4,

for which the user's health status can be inferred based on the gravity of the symptoms that are recognized during the conversation. A *UserStatus* individual must be associated with a set of *StatusItem* individuals that represent atomic conditions under which a *UserStatus* can be activated. *StatusItem* individuals can be associated with a *UserStatus* by the property *influencesUserStatus*. As an alternative, when *UserEvent* instances are predefined, a *UserStatus* can be associated with a set of *UserEvent* that, in turn, are associated with *Intent* individuals. This path describes which is the list of *Intent* enabling the classification of a *User* with respect to a specific *UserStatus*. A *StatusItem* instance can be activated upon the recognition of a specific *Intent* that is indirectly associated with it through a *Slot* individual. Generally, not all *StatusItem* has to be activated for inferring, in turn, a *UserStatus*. Different strategies can be applied at reasoning time according to the requirements of the application's domain.

The third subsumed concept is the *DialogStatus* one. A *DialogStatus* individual provides a snapshot of a specific *Subdialog* at a certain time. Individuals of type *DialogStatus* are created at reasoning time after the processing of the *Intent* recognized by the system. Two main entities must be associated with the *DialogStatus* individual: (i) the *Dialog* instance and (ii) the *DialogAction* that has to be performed as the next step. The list of *DialogStatus* instances associated with a *Subialog* instance can be exploited by the mechanism in charge of deciding the most appropriate actions to lead the dialogue to its goal.

5.3.2 The Assertion Box of Convology

The ABox of Convology includes 7 pre-defined instances. These instances are common to any dialogue and, for this reason, we judged that their availability would speed up the development process of a new agent. The instances are summarized in Table 5.1. Additional instances are reserved

Instance name	Class type	Description
initiate	*OpenQuestion*	First *DialogAction* to be executed by the *Agent*
fallback_question	*OpenQuestion*	A *DialogAction* that can be sent when the *Agent* could not interpret the *User*'s input
fallback_intent	*Intent*	An instance to handle non recognized *User*'s input
goal	*Feedback*	*DialogAction* to be executed by the *Agent* to conclude the conversation
acknowledge	*Feedback*	*DialogAction* to be executed by the *Agent* to acknowledge receiving some information
feedback_unexpected_intent	*Feedback*	*DialogAction* to be executed by the *Agent* when receiving some information that was not expected.
information_seeking	*InformationSeeking*	To identify the type of a *Subdialog* instance
deliberation	*Deliberation*	To identify the type of a *Subdialog* instance

Table 5.1: Pre-defined Instances in Convology.

for the domain-specific knowledge that must be provided when Convology is integrated into a conversational application. These instances can be provided either by the dialogue designer or they can be generated as a result of the dialogue interactions, depending on the domain requirements.

The next section describes how Convology was instantiated into a conversational agent for monitoring asthma patients. In a further study, which is described in Chapter 8, the reusability of Convology can be observed in the implementation of a dialogue system for diagnosis.

5.4 Convology in Action

Convology has been deployed into *PuffBot*, a multi-turn goal-oriented conversational agent that addresses the asthma scenario described in Section 2.1. *PuffBot* focuses on acquiring information through dialogue, having as its dialogue goal to reach a real-time classification of the *UserStatus* to output a related recommendation as a *Feedback*. In this Section, we demonstrate the practical use of Convology in this application. It is important to highlight that external modules (e.g. NLU and planning modules) and further mechanisms (e.g. the strategy adopted to classify the user status) were used for the development of *PuffBot*. These aspects are out of the scope of this section but they are discussed along with this thesis.

When a new dialogue session starts in *PuffBot*, a new instance of *Dialog* is created for this session. The *goal* of the overall dialogue is achieved when *PuffBot* is able to infer the *UserStatus*, outputting a recommendation as a *Feedback*. However, as discussed in Section 2.2.1, the asthma scenario presents four different sub-dialogues and each sub-dialogue requires the definition of their own goals. Therefore, the first class to be instantiated in Convology is the *Subdialog* class. Four instances were generated and their corresponding *DialogType* was identified, i.e. two instances are associated with the *information_seeking* individual and the two others with the *deliberation* individual. An ordering among the *Subdialog* instances was identified by the property *triggers*. Next sections describe how Convology is exploited to address the first two sub-dialogues, being the first of type *InformationSeeking* and the second a *Deliberation* sub-dialogue.

5.4.1 Information-Seeking Sub-dialogue

As described in Section 2.2.1, the first sub-dialogue to be conducted within the asthma scenario is an information-seeking sub-dialogue. The goal of

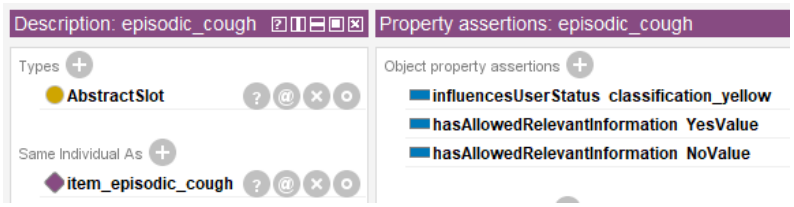


Figure 5.3: An instance of the *Slot* concept in Convology.

this sub-dialogue is to achieve a classification of the user's current situation. For this sub-dialogue, Convology was populated in the following way:

- symptoms are modeled as *Slot* instances;
- asthma classes (e.g. yellow, orange, red) are modeled as *UserStatus*. The property *refersToSubdialog* associates the instances with the correct *Subdialog* instance;
- *Question* and *Intent* instances were generated having as the starting point the *Slot* individuals. This happens because, in this scenario, the agent will mainly ask end-users about their symptoms and the end-users intentions should be directed to inform their symptoms.

An example of a *Slot* individual is provided in Figure 5.3. In the Figure, the *Slot*[10] individual named *episodic_cough* has a property informing that it influences the *UserStatus* instance named *classification_yellow* (as can be seen in Figure 2.1) as well as properties that inform the allowed values for this slot[11]. In addition, in this scenario, *StatusItem* are equivalent to *Slot* instances (*item_episodic_cough* in the image).

The *episodic_cough* instance is also associated with an *Intent* individual (*inform_episodic_cough*) through the property *hasSlot*. Finally, a *ClosedQuestion* individual is created (*request_episodic_cough*) and it is associated with

[10]The use of *Abstract* classes was discussed in Section 5.3.

[11]*Puffbot* addresses propositional values for the slots. The actual value of a slot is retrieved during the conversation.

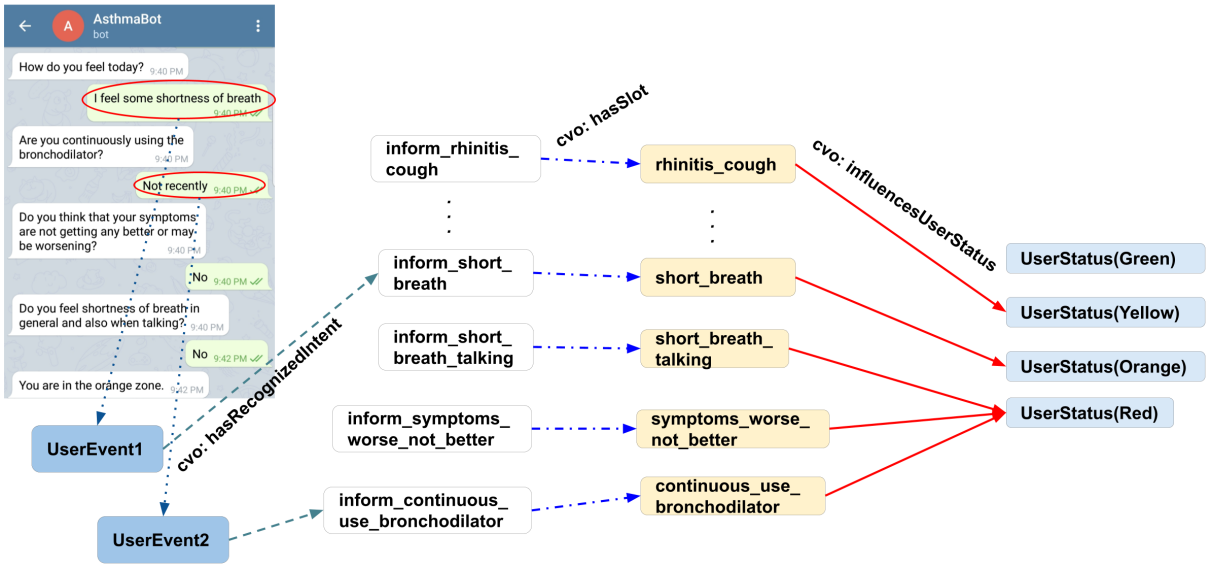


Figure 5.4: Illustration of the Convology Instances Exploited to Handle the User's Input in a Dialogue Session.

this intent through the property *hasRelevantIntent*. This association makes it easier for the agent to recognize the intent of the next user's input after the question is sent.

In Figure 5.4, we provide a sample of an information-seeking conversation between *PuffBot* and a user. To simplify the understanding, we exhibit only a question-answer sequence, but *PuffBot* is capable of identifying intents from user-initiated input, i.e. intents that are not an answer to a previous system question.

When the user inputs some information, this information is first instantiated as an *UserEvent* in Convology (dotted lines in Fig. 5.4). The natural language text provided by the user is sent to the NLU module for interpretation. The NLU module returns the set of detected intents together with a confidence value attached to each intent, representing how certain the NLU module is about the detection of that intent. These intents are expected to be matched with the instances of *Intent* previously modeled in Convology (white blocks in Fig. 5.4). Next, the recognized *Intent* indi-

viduals (if any) are associated with the *UserEvent* individual through the *hasRecognizedIntent* object property (dashed lines in Fig. 5.4).

The next step is state tracking; in order to identify the new state of the dialogue and to set a *Slot* value, *PuffBot* analyzes the *Intent* instances recognized by the NLU module. As a first attempt, to recognize the current dialogue state, *PuffBot* tries to match these instances with the *Intents* that were expected as a consequence of the last system's action. In fact, for short user utterances like "*no*" (Fig. 5.4), the existing association of a *Question* with *Intent* individuals that is provided by Convology favors the recognition of the corresponding *Intent* and, consequently, the *Slot* associated to this instance (yellow blocks in Fig. 5.4). That is, when a *no-intent* is recognized, the state tracker can look back at the last agent's interactions and easily comprehend that the user input was an answer a previously made question. More details on *Puffbot's* strategy to recognize the dialogue state are given on Section 7.3.1.

After selecting an *Intent*, *PuffBot* can finally set the property *hasRelevantInformation* for the corresponding *Slot* with the value extracted by the NLU module (informed in *hasNLURelevantInformation*). As the value given to a *Slot* affect the classification of its associated *UserStatus* (continuous lines in Fig. 5.4), this process takes the dialogue session to a new state. Thus, a new *DialogStatus* instance is created (omitted in Fig. 5.4).

Whenever a new *DialogStatus* is instantiated, the mechanism in charge of deciding the next action to take is triggered. This mechanism instantiates the property *hasdialogAction* that informs the selected *DialogAction*, which can be a *Closed/OpenQuestion* or a *Feedback*. Note that at this point, there are two possibilities: (i) the agent generates the proper individuals to continue the conversation, or (ii) a classification is reached, meaning that the goal of the sub-dialogue was achieved. For both cases, an instance of *Event* is created and associated with the *DialogAction* indi-

vidual. This instance is then processed by the NLG module that generates the actual utterance understandable by the final user.

However, when the latter occurs (goal is achieved), the *Subdialog*'s property *hasGoalAchieved* is instantiated and *Puffbot* checks if a new instance of *Subdialog* should be triggered. In our example, the next instance is a deliberation sub-dialogue and it is discussed in the next Section. Instead, when no more sub-dialogues are available, *Puffbot* will conclude the conversation, setting the property *hasGoalAchieved* with the last goal (*UserStatus*) achieved also to the *Dialog* individual.

5.4.2 Deliberation sub-dialogue

To address the second and fourth sub-dialogues of the scenario described in 2.1, the recommendations were abstracted and modeled as *UserStatus* instances. This strategy was adopted because providing a recommendation is the dialogue goal in a deliberation sub-dialogue. This way, like the goal instances in an information-seeking sub-dialogue, each goal has a corresponding *Feedback* instance to be exploited to output the recommendation.

According to its definition in Section 2.2.1, a deliberation sub-dialogue decides upon the actions that should be taken with respect to the classification made in the previous sub-dialogue. Therefore, instead of just outputing a recommendation to the end-user, it is important to confirm whether this action has already been taken (e.g., user already took the recommended medicine). To address this behavior and, since the recommendation itself is a piece of information, an instance of *StatusItem* is associated with each *UserStatus* instance. In addition, a *ClosedQuestion* and an *Intent* instances are created and they should be exploited to set the value (*RelevantInformation*) of this *StatusItem* instance.

Like in the previous sub-dialogue, on execution time *Puffbot* associates

ClosedQuestion and *Feedback* instances with an instance of *Event*. The NLG component is in charge of converting these instances to a human-understandable utterance.

5.5 Conclusions and Future Work

This chapter introduced Convology: a top-level ontology that represents goal-oriented conversational scenarios, adding semantics to conversational concepts. Convology has been thought to provide a meta-model to aid the complex process that is the development of a conversational agent. The knowledge modeled in Convology derives from the analysis of collaborative work between knowledge engineers with competencies in conversational agents and health experts. The contribution of the latter comes from discussions on the requirements of a conversational system for the asthma domain. This chapter first described the process followed to build the ontology and then it detailed the main concepts and properties that were included. Next, a dialogue system to monitor asthma patients was discussed, showing how the ontology can be deployed in a real-world scenario.

Convology opens the opportunity for several research activities. These activities include the integration of Convology within further conversational scenarios with the aim of further verifying its correctness and completeness. Convology is also expected to be extended to cover additional types of dialogue. As described in this chapter, the current version includes two types (*information-seeking* and *deliberation*) but Walton and Krabbe (1995) [121] categorized six different types, which might be of interest of further conversational applications. Finally, the authors of Convology are aware that *dialogue systems* cover a broad topic that comprises several sublines of research. Therefore, we foresee the possibility of extending Convology to integrate additional topics frequently discussed by the di-

alogue community. Examples are frameworks that address empathetic, collaborative, argumentative or persuasive conversations. Modeling such frameworks within Convology or integrating external ontologies with this purpose is an opportunity for future work.

Chapter 6

Information Usefulness

6.1 Introduction

The different factors that make a health dialogue unique, as listed in [17], increase the complexity of the dialogue and make the choice of what to say/ask next not a trivial task. In task-oriented dialogue systems [59], the decision on the next action is a responsibility of the dialogue manager component [75]. For this component, factors such as *criticality* and *expertise*, as discussed in Section 3.1, must be taken into account when dealing with health dialogues.

Current architectures for dialogue management approach differently the well known challenge of choosing the dialogue action to execute next. *Finite-state* machines [59], for example, keep the control of the conversation with the user in a predetermined fixed order. *Frame-based* approaches [59], instead, select the next unfilled slot in a frame. On the other hand, many recent approaches that deal with complex dialogues use data-driven techniques such as reinforcement learning [123, 89] for action selection. Even if such approaches do not require a large amount of domain data for training the model, they suffer from the limitation of removing some control on the policy, which may not be desired in healthcare. Finally, AI planning is presented as a promising strategy for dialogue management [12, 42, 67], but

it suffers some limitations to properly address the healthcare domain [17]. Therefore, to build more efficient health dialogue systems, further research that covers the health needs into general-purpose strategies for dialogue management is required.

This Chapter introduces an agent-based framework that supports the development of an intelligent goal-driven agent to manage health dialogues. We adapt and extend the idea presented in [30], that models information usefulness for goal-driven agents, to enable its application into a real-world and more complex domain, i.e., healthcare. This way, our framework provides a strategy for the action selection task by comprising components that allow the complex reasoning on the above mentioned health factors (criticality and expertise). This strategy contributes to the generation of a multi-turn dialogue that uses as few questions as possible to retrieve the information, seeking a fast, but also a cautious solution.

It is important to note that, in this Chapter, by *dialogue* we mean the whole interaction process for information acquisition between an agent and the environment, not being limited to a human-machine conversation. That is, although later in this thesis it is integrated into a dialogue agent, the framework discussed here focuses on acquiring useful information regardless of its sources, which might involve practitioners, sensors, or external services, for example. Therefore, the overall result is a multi-turn interaction that uses as few steps as possible to retrieve the useful information. An evaluation of the framework is presented in Section 6.6, showing its effectiveness on a real-world use case concerning diagnosis.

The proposed framework has first been shared with the related community in [117]. Then, an extension of it introduced the principles of *cautiousness* and *fast-solving* solutions; it is available in [115].

6.2 Preliminaries

This section presents some preliminary definitions that compose the usefulness framework. Although some of the terminologies are shared with other research areas (e.g., multi-agent systems), the definitions here provided are reported for the healthcare field.

Definition 1 *An agent a is a discrete entity aiming to classify a patient with respect to a set of classes. An agent a has its own belief base B_a and a set of goals G_a.*

Definition 2 *The agent belief base B_a contains the collection of the truth values (true, false, or unknown) associated with each information unit s, as well as a set of rules R_a supporting the patient classification task by stating which information units are required to achieve each goal.*

B_a is updated after the arrival of a new piece of information that is relevant to change the truth value of an information unit s contained in B_a.

Definition 3 *An information unit s (called hereafter "slot") is a relevant information, increasing the knowledge of the belief base B_a necessary to achieve a goal g.*

The default value of a slot s is *unknown*. During the information acquisition process, the value of a slot s can dynamically change to *true* or *false*, depending on the information received.

This framework considers different types of information: (i) information acquired directly by asking the patient (slots of type S_a), (ii) information acquired directly through sensors (slots of type S_b), (iii) information acquired from the patient's electronic file (slots of type S_c) (iv) information acquired from the history of data provided by the patient (slots of type S_d) and (v) information acquired through external services (slots of type S_e).

The types of information required on an instance of the agent, correspond to the ones presented in the guidelines for the deployed domain.

Definition 4 *A goal $g \in G_a$ is a class for which an agent a has to know if a patient is classified with it or not. The classification task is performed by applying the rules defined in R_a.*

A goal can be characterized thanks to two values: (i) a confidence degree $C(g)$ that represents the amount of true information (i.e., the truth value of a slot s is *true*) that the agent a collected about the goal g, and (ii) $M(g)$ that represents the amount of information that the agent a still have to collect to have a complete knowledge concerning the goal g, i.e., missing information. Let τ be a threshold given by an expert. A goal g can be considered achieved (the patient may be considered as belonging or not to the class) if: (i) $C(g) \geq \tau$ or if (ii) $C(g) + M(g) < \tau$. The former means that the information collected by the agent a is enough to classify the patient as belonging to the class g. The latter means that even if all missing information would be set to *true*, the value of $C(g)$ will not pass the threshold τ. Hence, the agent a can classify the patient as not belonging to the class g. In both cases, the agent a achieves the goal. This is one of the peculiarities of the health domain, since a physician should be supported by intelligent agents able to both detect and exclude undesired situations.

Another aspect is the possibility of giving an importance degree to each goal g, hereafter called "priority".

Definition 5 *The priority o of a goal g is a real value in the interval $[0, 1]$ representing the importance degree that the goal g has within the belief base B_a. The set of all priority values is given by O_{G_a}*

Definition 6 *A rule $r \in R_a$ allows classifying a patient into a class g based on the information collected by an agent a. A rule can be represented*

as follows[1]*:* $s_1 \oplus s_2 \oplus \ldots \oplus s_{n_g} \Rightarrow g$*, where* s_i *represents the* i*th slot related to* g *and* n_g *is the number of slots related to* g*.*

Definition 7 *The premise set* $P(g)$ *is the set of all the slots that help classifying goal* g*, i.e.,* $P(g) = \{s \mid s \in lhs(g)\}$*, where "lhs" is the left hand side of a rule* $r \in R_a$*.*

Definition 8 *Let* g_j *be a goal and* s_i *be a slot. The association between* g_j *and* s_i *is represented through a real value* $w_{ij} \in [0, 1]$*, called "weight". The weight represents how relevant the information contained in the slot* s_i *is to achieve the goal* g_j*. We will also note the association between goal* g *with Slot* s *as* $w(s, g)$*.*

The same slot can appear in more than one rule r since some information units can be shared by more than one goal. Each value w_{ij} is part of domain-specific knowledge and it can be directly provided by domain experts (guidelines) or inferred from background knowledge. Therefore, their computation changes according to the domain for which the agent is deployed and it is out of the scope of this work. The only constraint is that, given the j_{th} goal, it holds the hypothesis $\sum_{i=0}^{n_{g_j}} w_{ij} = 1$, where n_{g_j} is the number of slots associated with the goal g_j. It is important to emphasize that, even if this constraint holds, these values are not probabilities. The rationales behind this constraint are the following. First, we require that if all the slots associated with a specific goal are set to *true*, the agent is "sure" that the patient belongs to that goal. Second, the classification of a patient with respect to a goal relies on the analysis of the agent's confidence on the possibility that a patient belongs to that goal, given the threshold τ. Hence, in order to perform this comparison fairly, the numerical boundaries of the confidence have to be the same for all goals.

[1]The aggregation operator $\oplus$ is to be implemented as required by the scenario for which the framework is deployed.

Section 6.3 introduces the working example used throughout this Chapter to show how these elements can be instantiated into a real-world case.

6.3 Working Example

This section presents an illustrative example to show how the framework can be instantiated into a real-world scenario. In this example, whose aim is to help the reader to understand the proposed framework, only slots of type S_a are considered, i.e. slots concerning information that is obtained from the patient during a consultation with a practitioner.

Let us define a scenario in which, by means of a dialogue, an agent aims to retrieve the symptoms experienced by a patient with the goal of diagnosing the possibility that one or more diseases may occur. Here, we assume to have a finite set $G_a = \{g_1, g_2, g_3, g_4\}$ of diseases and a finite set $S = \{s_1, ..., s_d\}$ of symptoms related to the different diseases; d corresponds to the number of slots (pieces of information) related to the different diseases, and in our example $d = 12$. We also have the assumptions that (i) each symptom s can be a premise for more than one disease (s_1, s_4, and s_8 in our example), and (ii) the value of each slot s can be 0 (i.e., the symptom has not been experienced) or 1 (i.e., the symptom has been experienced).

Table 6.1 shows the list of diseases, the list of symptoms, and the relevance values w representing the associations between symptoms and diseases. This information composes the belief base B_a that an agent a has during the conversation with a patient. The aim of the agent is to achieve all the set of goals G_a to classify the patient with respect to each disease, i.e., to know which diseases might affect a patient and which not. It is important to notice that within a real-world scenario, in addition to symptoms, further knowledge such as information on the patient's profile and family history can also compose B_a. For the sake of simplicity, we restricted

Table 6.1: Diseases/Symptoms Domain Knowledge

Disease	**Symptoms**											
	Abdominal pain (s_1)	Pruritus (s_2)	Respiratory distress (s_3)	Fever (s_4)	Early awakening (s_5)	Nausea (s_6)	Incontinence (s_7)	Shortness of breath (s_8)	Pain (s_9)	Diarrhea (s_{10})	Asthenia (s_{11})	Yellow sputum (s_{12})
Hepatitis (g_1)	0.36	0.28	0.20	0.16	-	-	-	-	-	-	-	-
Colitis (g_2)	0.09	-	-	0.30	0.26	0.22	0.13	-	-	-	-	-
Tricuspid insufficiency (g_3)	0.11	-	-	0.05	-	0.16	-	0.42	0.26	-	-	-
Kidney failure (g_4)	-	-	-	0.13	-	-	-	0.35	-	0.26	0.22	0.04

this working example only to the symptoms that allow us to identify a disease without considering any other patient-specific knowledge. In addition, note that not all the slots must be filled, i.e., depending on the disease, not all symptoms must be confirmed to reach a diagnosis.

The classification task is performed by applying the rules R_a that allow to classify the patient with respect to each disease.

$$s_1 \oplus s_2 \oplus s_3 \oplus s_4 \Rightarrow g_1$$
$$s_1 \oplus s_4 \oplus s_5 \oplus s_6 \oplus s_7 \Rightarrow g_2$$
$$s_1 \oplus s_4 \oplus s_6 \oplus s_8 \oplus s_9 \Rightarrow g_3$$
$$s_4 \oplus s_8 \oplus s_{10} \oplus s_{11} \oplus s_{12} \Rightarrow g_4$$

Finally, we consider that the domain-specific knowledge might include also information concerning entailments and co-occurrence of symptoms (e.g. a patient with high fever also has "just" fever). In our example, we assume to have the following three entailments:

$$s_6 \rightarrow s_1; \quad s_1 \rightarrow s_9; \quad s_{10} \wedge s_{11} \rightarrow s_4$$

The presentation of the framework in Section 6.4 relies on this example.

6.4 Proposed Framework

This section introduces the new agent-based framework to support the management of a slot-filling interaction related to the health domain. The focus is on information that must be retrieved with the goal of classifying

the end-user (e.g. patient) into some predefined classes. Therefore, in this work, each slot is expected to give origin to a dialogue action that corresponds to a question to retrieve the slot value (confirming or disregarding it). For example, to retrieve the value of the slot s_4 (*fever*), the dialogue action would be translated to an utterance like "Have you had a fever recently?". However, following the architecture of task-oriented dialogue systems [129], it is important to highlight that both, the conversion of a dialogue action into a natural language utterance (*Natural Language Generation*) and the natural language processing of the answer given by the user (*Natural Language Understanding*) are out of the scope of this work. The concern of the framework is to feed the dialogue manager with a strategy to determine which is the dialogue action that should be executed next (i.e. which slot should be retrieved next).

To address the challenge of effectively selecting the next dialogue action that most contributes to a classification, we extend the framework proposed in [30], that discusses the manipulation of pieces of information represented by formulas. The rationales behind this extension are the following: (i) here, the main aim of the agent is to understand which is (or which are) the class to which the patient belongs to according to the values given to the slots; (ii) such values can be viewed as the components in the left-hand side (lhs) of a rule, associating possible slots to a particular class; and (iii) the agent aims to mimic the reasoning task of a physician, who can be considered as a cognitive agent, i.e., someone with background knowledge (beliefs) about the patient and other subjects (e.g. the possible relations or the probability of co-occurrence between the symptoms, etc.).

We consider a propositional language L of which a subset, L_G, is the language used to represent the rules associated with the goals. We will consider a *dialogue agent* that is aware of all the beliefs/knowledge (this includes the pieces of information obtained in the form of slot types S_b, S_c,

S_d, and S_e and, also, information concerning entailments), the patient's answers (slots of type S_a), and the goals of the physician. The dialogue agent a has a goal set G_a (classes) which corresponds then to the physician goals from the language of possible goals L_G. For example, in the context of diagnosis, the goal of the dialogue agent is to understand among the different diseases (g_1, g_2, ..., g_n), which of them correspond to the patient's condition. We would like to highlight that, unlike in [30], where the goal achievement is binary, here we consider a gradual definition to compute the extent to which a diagnosis is established. For this, we consider a threshold (τ) that allows us to decide if the result is positive (the patient may have the disease) or not (the patient may not have the disease).

6.4.1 The Belief Base Composition

The dialogue agent a has a belief base B_a composed of two subsets B_a^m and B_a^g. B_a^m is the set of formulas from $L \setminus L_G$ which represents a's beliefs about the slot-values, e.g., the patient has *pruritus* (s_2), the patient has a *fever* (s_4), *nausea* (s_6), *incontinence* (s_7), etc.. B_a^m may also contain other physician background knowledge like the co-occurrence between the slots, e.g., *Abdominal Pain* implies *Pain* ($s_1 \rightarrow s_9$). This means that if *Abdominal Pain* is reported during the dialogue as one of the experienced symptoms by the patient, B_a is automatically updated with both the values concerning the slots *Abdominal Pain* and *Pain*. B_a^g instead, contains as many rules of the form $s_1 \oplus s_2 \oplus \ldots \oplus s_{n_g} \Rightarrow g$ (Def. 6), where each s_i is a literal of $L \setminus L_G$, representing a slot that influences the achievement of goal g, as there are $g \in G_a$. Such rules represent the beliefs of a about which information is needed to determine a class, achieving then a given goal. These pieces of information are indeed the slot-values of the different types presented in Definition 3. For example, to understand if the patient has *Hepatitis* (g_1), the physician needs to know if the patient has *Pain*

Abdominal (s_1), *Pruritus* (s_2), *Distress respiratory* (s_3), and *Fever* (s_4), i.e., $s_1 \oplus s_2 \oplus s_3 \oplus s_4 \Rightarrow g$. The set of the elements in the left-hand-side of this rule is the premise of g (Def. 7), i.e., $P(g) = \{s_1, \ldots, s_4\}$.

Let us suppose that during the dialogue with a patient, the dialogue agent acquires more and more information about the patient's actual state, approaching the classification into one or more classes. The units of information that are still missing can be represented as follows.

Definition 9 (Missing Information) *Let a be a dialogue agent with its belief base B_a and its goal set G_a. Let $g \in G_a$ be such that $B_a \not\models g$ [2]. The missing information for goal g, $Missing(B_a, g)$, is defined as follows:*

$$Missing(B_a, g) = \{l : l \in P(g) \text{ and } B_a \not\models l\} \tag{6.1}$$

$Missing(B_a, g)$ is the set of all the slots in the premise of g which cannot be deduced from B_a (i.e., which are not yet believed by the agent and therefore the dialogue agent should acquire them).

Remark 1 *In the particular case in which $B_a^m = \emptyset$, $Missing(B_a, g) = P(g)$, i.e., the missing piece of information to achieve g is $P(g)$.*

Because different goals can have missing information in common, we need to introduce the notion of a *multiset of missing information.*

Definition 10 (Multiset of missing information) *Let a be a dialogue agent whose belief base is B_a and whose goal set is G_a. The multiset [3] of missing information to achieve the goals in G_a is:*

$$Missing(B_a, G_a) = \bigcup_{k=1}^{|G_a|} Missing(B_a, g_k) \tag{6.2}$$

where $\bigcup$ represents the union on multisets, $||$ represents the cardinality of a set.

[2] In propositional logic, $\phi \models \psi$ means that ψ is a logical consequence of ϕ. Here, it means that we can classify the patient as having disease g from what we already know/believe ($B_a \models g$, see Defi. 15).

[3] Reminder: a multiset is a set whose elements can have several occurrences, such as $\{p, q, p\}$.

6.4.2 The Overall Importance of a Slot

Let us consider the following definitions:

Definition 11 *Let $G(s)$ be the set of goals related to slot s. We define $G(s) = \{g \in G_a \mid s \in P(g)\}$.*

Definition 12 *Let $W(g)$ be the set of all the weights related to the symptoms associated with goal g. $W(g) = \{w(s,g) \mid s \in P(g)\}$.*

We can now define the overall importance of requiring an answer for the slot s. $N_1(s)$ represents the extent to which s ($\neg s$) would help get closer to classification, with respect to all goals $g \in G_a$ as follows:

$$N_1(s) = \left\{ \begin{array}{ll} \sum_{g \in G(s)} w(s,g) + \sum_{s' | B_a \cup s \models s' \wedge B_a \not\models s'} w(s',g) & \text{if } s = true, \\ \sum_{g \in G(s)} w(s,g) + \sum_{s' | B_a \cup s' \models s \wedge B_a \not\models s} w(s',g) & \text{if } s = false. \end{array} \right\} \tag{6.3}$$

where $w(s',g) = 0$ if $s' \notin P(g)$ and s comes from the multiset of missing information (Def. 10). We can notice that $N_1(s)$ is the gradual definition (and therefore an extension) of the N_1 component proposed in [30].

We can also compute the overall weight, $N_2(s)$, that concerns the slots which are still missing after receiving the value of slot s as follows:

$$N_2(s) = (\sum_{g \in G_a} \sum_{k \in W(g)} k) - N_1(s). \tag{6.4}$$

We can notice that our definition of $N_2(s)$ is a generalization (a gradual counterpart) of the one proposed in [30].

6.4.3 Goal Achievement

To characterize a goal as being achieved or not, we need to know the amount of already known information about the slots related to it and the amount of information which is still missing.

Definition 13 *The amount of information that the agent a collected about the goal g, is given by the union of:*

$$\mathcal{C}(g) = \sum_{(s|B_a \models s) \wedge (s \in P(g))} w(s, g) \tag{6.5}$$

representing the information slot set to true, and:

$$\mathcal{C}(\neg g) = \sum_{(s|B_a \models \neg s) \wedge (s \in P(g))} w(s, g) \tag{6.6}$$

representing the information slot set to false.

Definition 14 *The amount of information that the agent a still has to collect to have a complete knowledge concerning goal g, $\mathcal{M}(g)$ is given by:*

$$\mathcal{M}(g) = \sum_{s' \in Missing(B_a, g)} w(s', g) \tag{6.7}$$

Let τ be a threshold which allows characterizing a goal as being achieved or not, according to the guidelines.

Definition 15 *A goal g is achieved, i.e., the diagnosis can be made, noted $B_a \models g$, if and only if:*

- $\mathcal{C}(g) \geq \tau$*: the amount of information already available is* sufficient *to make a positive diagnosis, i.e., the patient has the disease, Or*
- $\mathcal{C}(g) + \mathcal{M}(g) < \tau$*: the amount of information already available is* sufficient *to make a negative diagnosis, i.e., the patient does not have the disease.*

The value of τ obviously influences the capability of an agent concerning the achievement of goals. High values of τ reduce the possibility of classifying a patient as belonging to a specific goal, but increase the possibility that a patient would not belong to any goal. The opposite occurs for low values of τ.

Remark 2 *We can notice that if there are not missing slots, i.e.,* $\mathcal{M}(g) = 0$*, we have:* $\mathcal{C}(g) + \mathcal{C}(\neg g) = 1$.

This remark shows how the total amount of information is preserved when all slots are filled.

Definition 16 *The set of goals that a slot* s *allows the dialogue agent to achieve is:*

$$E(s) = \{g \in G_a | B_a \cup s \models g \wedge B_a \not\models g\}. \tag{6.8}$$

6.4.4 Next Question Selection

Considering the given definitions, this work provides a function that measures the usefulness of a (not-yet-filled) slot [4]. By comparing the usefulness value of the different slots, it is possible to select the *best candidate question/test/check* for the next information acquisition, i.e., the one that allows us to achieve/get closer to a classification. The usefulness measure takes into account several factors:

- Class' priority: for domains that present urgency or critical situations, a priority can be set for each class. Therefore, whenever priority values are available, slots related to classes with higher priority have their usefulness value increased.
- Information filled by the slot: some slots are capable of filling more information than others, since they are present in more than one class or the domain knowledge reports some co-occurrence.
- Slot's weight: as previously stated, according to the domain, not all pieces of information (slots) have the same relevance to a class. For example, a physician may inform that *abdominal pain* (s_1) is more

[4] Of course, here we are interested in knowing the utility of the slots which are not a logical consequence of the belief base B_a, i.e., which cannot be deduced from previous beliefs/knowledge.

relevant than *fever* (s_4) to determine that the patient has *Hepatitis* (g_1) and, therefore, the latter would present a higher usefulness value to this class. The usefulness function also takes into account that a slot can be present in different classes with different weights.

Due to the different requirements of health domains, when we compute the usefulness value of a slot s, we need to consider what happens in both cases, i.e. if the slots s is set to 1 (the symptom occurred) or to 0 (the symptom did not occur). The resulting equation to compute the usefulness of a slot s is the following:

$$U(s) = \Big[\Big(|E(s)| + \frac{N_1(s)}{N_1(s)+N_2(s)}\Big)^{s=0} + \Big(|E(s)| + \frac{N_1(s)}{N_1(s)+N_2(s)}\Big)^{s=1}\Big]\frac{O^s_{G_a}}{|G^s_a|} \quad (6.9)$$

where:

- $|E(s)|$ is the number of goals that are satisfied thanks to the information about slot s, i.e., the number of diseases for which we can conclude if the patient belongs to them or not after checking the slot s;
- $N_1(s)$ and $N_2(s)$ are computed by taking into account the weights associated with s in all goals;
- $O^s_{G_a}$ is the sum of the priorities associated with the goals having the slot s in their premise;
- $|G^s_a|$ is the number of goals having the slot s in their premise.

The coefficient $\frac{O^s_{G_a}}{|G^s_a|}$ increases the usefulness value of slots that are the premise of *more important* goals.

6.4.5 Cautious and Fast-Solving Approaches

Due to the different requirements of health domains, the choice of the next dialogue action can be determined by using two different approaches:

cautious or *fast-solve*. The definition of usefulness of a slot may slightly change depending on the approach that is more convenient to achieve a classification in the given domain. Each of them has its advantages and disadvantages and choosing which one to implement in the dialogue system is a matter that can be decided by the domain expert, based on the scenario in which the framework is deployed.

The approach discussed so far follows the principle of ***cautiousness***, i.e., it explores the domain as much as possible to investigate different possibilities that may lead to a classification. As defined in the proposed function, this is achieved by increasing the usefulness value of slots that, besides enabling the achievement of as many goals as possible, also cover more information from a single *action for information acquisition* (question/test/check), such those that are common to more than one class or that co-occur. The main advantage of this approach is to empower the cautiousness of the practitioner within sensitive scenarios. As a disadvantage, this approach may be slow to reach the classification when G_a is a large set.

Instead, the ***fast-solve*** approach consists in detecting the slot s which is, for the agent a, at the same time useful and that makes the classification of a patient as belonging to a goal g as much proximate as possible to the threshold. The coefficient of maximum proximity can be defined as follows.

Definition 17 (Coefficient of maximum proximity) *Let B_a the belief base of an agent a and G_a the set of goals contained in B_a. The coefficient of maximum proximity for a slot s is given by:*

$$MP(s) = \begin{cases} 1 & \text{if } E(s)^{s=1} \geq 1, \\ 1 - \frac{\tau - \max(C(g \in G_a))}{\tau} & \text{if } E(s)^{s=1} = 0 \end{cases} \tag{6.10}$$

This coefficient is applied to Equation 6.9 as follows:

$$U_{FS}(s) = U(s) \cdot MP(s) \tag{6.11}$$

Since the aim of the *fast-solve* approach is to quickly classify the patient as belonging to a specific goal, the coefficient of maximum proximity introduces in the usefulness formula a punishment proportional to their non-contribution in classifying a patient as belonging to a specific class.

The main advantage of this approach is to support the agent to quickly reach a threshold, which may be desired in emergencies. On the other hand, it has the disadvantage of possibly missing the retrieval of further slots that would lead to different classes.

The decision on when to stop exploring the domain, that is, stop acquiring information, depends on the needs of the domain and it is the role of the dialogue manager. For a *fast-solve policy*, for example, the dialogue manager ceases the dialogue and reports the obtained information as soon as the first goal is (positively) achieved. While this classification may be correct, depending on the risks that the domain may offer, the output should be further evaluated by a human agent. Conversely, many health domains require caution when making a decision and achieving a goal does not necessarily mean that all the relevant information was obtained. For this reason, an agent that adopts a *cautious policy* would continue the dialogue to obtain a more precise classification by retrieving additional slots that were not asked yet as a result of having a lower usefulness value. This new information will either confirm or exclude further classes (goals).

6.5 The Framework in Action

We discuss here the application of the proposed framework to the working example described in Section 6.3. For demonstration purposes, we set the value of τ to 0.75, we do not use priorities between classes, and we adopt the *cautious* strategy. Let us suppose to have a dialogue started by a patient in which he/she reported a general illness and the *fever* and

nausea symptoms.

Step 1: the agent acquires these two pieces of information and set the truth values for the slots s_4 and s_6 to 1.

Step 2: by analyzing the co-occurrence knowledge, the agent sets also the slots s_1 and s_9 to 1 from the application of both the $s_6 \rightarrow s_1$ and $s_1 \rightarrow s_9$ entailments. $\mathcal{C}(g)$ values, with $g \in G_a$, change in: $\mathcal{C}(g_1) = 0.52$, $\mathcal{C}(g_2) = 0.61$, $\mathcal{C}(g_3) = 0.58$, $\mathcal{C}(g_4) = 0.13$.

Step 3: the agent computes $U(s)$ for the remaining slots. The three most useful slots resulted: $U(s_8) = 4.05$, $U(s_2) = 2.19$, $U(s_5) = 2.18$.

Step 4: the agent asks for the symptom associated with s_8 (i.e. *Shortness of breath*) and the patient replies that he/she does not experience this. The slot s_8 is set to 0 and the $\mathcal{C}(g)$ values, with $g \in G_a$, remain unchanged. However, by checking the values of $\mathcal{C}(g) + \mathcal{M}(g)$ for the goals g_3 and g_4, we can observe the values of 0.58 and 0.65 respectively. This means that the agent can already exclude two possible diseases since even if all missing information would be set to 1, both $\mathcal{C}(g_3)$ and $\mathcal{C}(g_4)$ will be lower than τ.

Step 5: the agent computes $U(s)$ for the remaining slots. The two most useful slots resulted: $U(s_2) = 2.40$, $U(s_5) = 2.37$, while the others have values lower than 0.5.

Step 6: the agent asks for the symptom associated with s_2 (i.e. *Pruritus*) and the patient replies that he/she does experience this. The slot s_2 is set to 1 and the $\mathcal{C}(g)$ values change in: $\mathcal{C}(g_1) = 0.80$, $\mathcal{C}(g_2) = 0.61$, $\mathcal{C}(g_3) = 0.58$, $\mathcal{C}(g_4) = 0.13$. This answer allows the agent to conclude that the patient belongs to g_1.

Here, based on the decision policy applied by the domain expert, i.e. *fast-solve* vs. *cautious*, we can obtain different behaviors. When a *fast-solve* approach is applied, the expert exploits the conclusion provided by the agent as a trigger for further actions. Instead, since we decided to adopt the *cautious* approach, the agent performs other steps as shown below.

Step 7: the agent computes $U(s)$ for the remaining slots. The most useful slot resulted $U(s_5) = 2.47$, while the others continue to have values lower than 0.5.

Step 8: the agent asks for the symptom associated with s_5 (i.e. *Early awakening*) and the patient replies that he/she does not experience this. The slot s_5 is set to 0 and the $\mathcal{C}(g)$ values remain unchanged. Hence, goal g_1 remains the only goal to which the patient belongs. Indeed, by checking the values of $\mathcal{C}(g) + \mathcal{M}(g)$, we can observe a value of 0.74 for the goals g_2. This means that the agent excludes also this disease since it becomes impossible for $\mathcal{C}(g_2)$ to reach the threshold τ.

Differently, in case of a positive answer for the slot s_5, the $\mathcal{C}(g)$ values would change in: $\mathcal{C}(g_1) = 0.80$, $\mathcal{C}(g_2) = 0.87$, $\mathcal{C}(g_3) = 0.58$, $\mathcal{C}(g_4) = 0.13$. Hence, the agent would conclude that the patient may have two possible diseases. Even if this situation introduces uncertainty in the analysis of patients' conditions, it is the duty of the domain expert to decide which actions to take next. This practice is commonly accepted within the health-care domain as intelligent systems aim to support domain experts and not to replace them.

6.6 Evaluation

The previous section discusses a small running example showing in practice which are the execution steps performed by using the proposed framework. The evaluation of the proposed framework has been performed on a scenario related to the diagnosis of a disease by considering the symptoms that are provided by a patient. We built our knowledge by starting from the contingency table associating the list of symptoms relevant for a specific disease presented in [122]. The encoding of this table corresponds to a set of 131 goals and 397 information slots where each goal corresponds

to a specific disease and each information slot to a specific symptom.

We validated the effectiveness of the proposed framework with respect to three baselines:

- Baseline 1 (BS1): at each iteration, the selected slot is the most popular one. This means that the agent will check in its knowledge which is the information slot valued as "unknown" and having the highest number of occurrences.

- Baseline 2 (BS2): at each iteration, the selected slot is the one with the highest overall weight within the knowledge base. The overall weight of a slot s is computed by summing all the weights $w(s, g)$ associated with the slot s in the entire set of goals G_a.

- Baseline 3 (BS3): this baseline is a variation of BS1 where at each iteration, the selected slot is the one having the highest normalized frequency. The normalized frequency of a slot s is computed by dividing the number of occurrences of s by the number of slots valued as "unknown" within the knowledge base.

For both the *cautious* and *fast-solve* settings, we analyzed three metrics: precision, recall, and the average number of steps required by the agent to classify the patient. Precision and recall are computed by checking if the set of goals provided by the agent contains the correct one. Meanwhile, the average number of steps is computed by considering the number of information requests performed by the agent before reaching a conclusion.

The evaluation has been performed by simulating the interactions between the agent and a set of patients and by covering all goals defined in each scenario. Since in this work we did not take into account the analysis of how the value of τ can be set, we run a simulation for each possible value of τ within the interval $[0.00, 1.00]$ with an incremental step of 0.01

between two consecutive simulations. Finally, each simulation starts with a fixed number of slots already set, for which we annotated which goal would be recognized by a human agent. These slots represent the information provided by the patient when she starts the interaction with the system. For each value of τ, we performed three simulations with 1, 2, or 3 slots set on startup, respectively. This parameter made it possible to observe how the effectiveness of our approach changes according to the amount of information provided at the startup.

Figures 6.5, 6.3, and 6.1 show the results observed by using the *cautious* strategy, while Figures 6.6, 6.4, and 6.2 show the results observed by using the *fast-solve* strategy. Within all figures, the x-axis reports the different values of τ, while the y-axis reports the precision and recall observed at each step or the average number of steps. The analysis of the obtained results highlighted four main points of interest.

By observing the graphs shown in Figures 6.1 and 6.2, we can notice a different behavior of the number of steps required by the agent to achieve the goal. For the *fast-solve* policy at a certain point, the number of steps decreases, while for the *cautious* one there is a constant increase and no differences between our method and the baselines. Concerning the *fast-solve* policy, this behavior is quite strange by considering that by increasing the value of τ the expected number of steps needed to pass the threshold should increase as well. A more deep analysis revealed that by adopting our framework to select the next slot to fill, the agent increases the capability of detecting unreachable goals. Hence, it is possible for the agent to reduce the number of steps to classify the patient with respect to the correct goal. While, concerning the *cautious* policy, we observed how a wider exploration by the agent does not reduce the cost of finding candidate goals.

However, even if for the *cautious* policy the cost of finding candidate goals remains more or less the same, the effectiveness of our framework

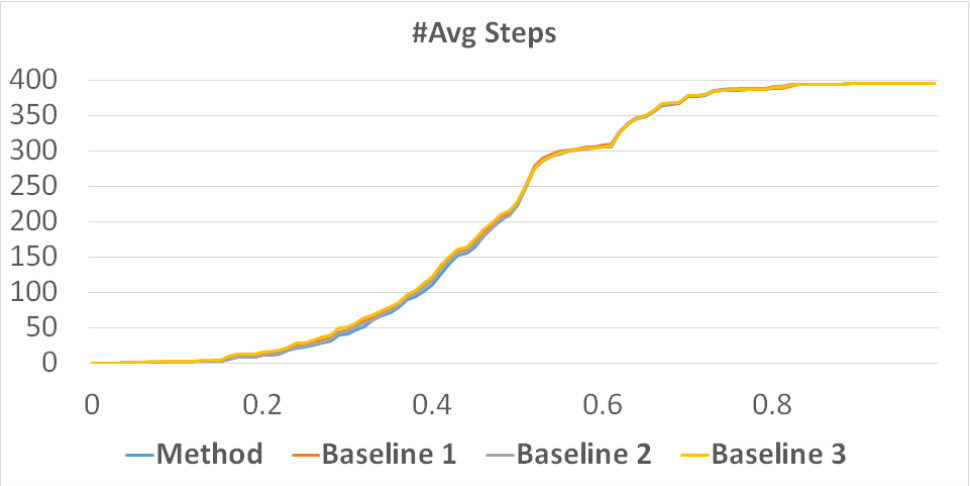


Figure 6.1: Cautious policy: graph of the number of steps required by the agent to reach a conclusion observed for all values of the threshold τ within the interval $[0.0, 1.0]$.

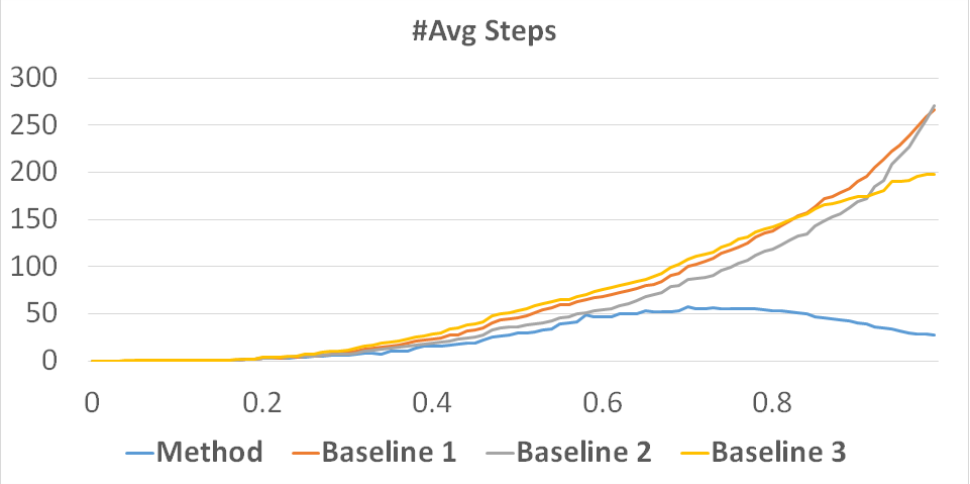


Figure 6.2: Fast-solve policy: graph of the number of steps required by the agent to reach a conclusion observed for all values of the threshold τ within the interval $[0.0, 1.0]$.

Figure 6.3: Cautious policy: graph of the Recall metric observed for all values of the threshold τ within the interval $[0.0, 1.0]$.

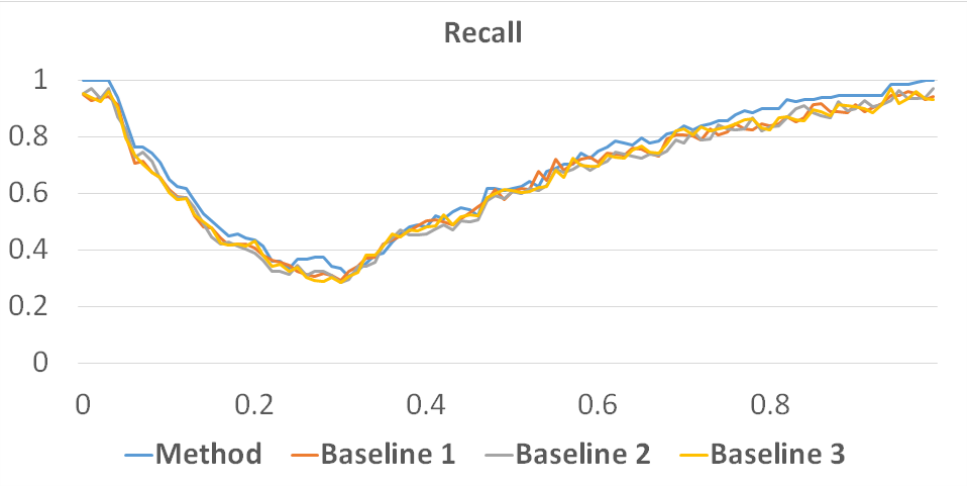


Figure 6.4: Fast-solve policy: graph of the Recall metric observed for all values of the threshold τ within the interval $[0.0, 1.0]$.

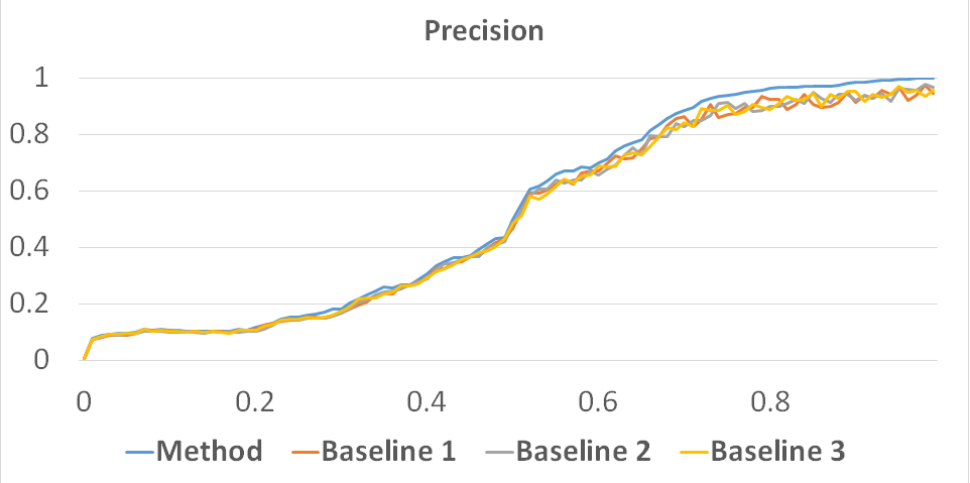


Figure 6.5: Cautious policy: graph of the Precision metric observed for all values of the threshold τ within the interval $[0.0, 1.0]$.

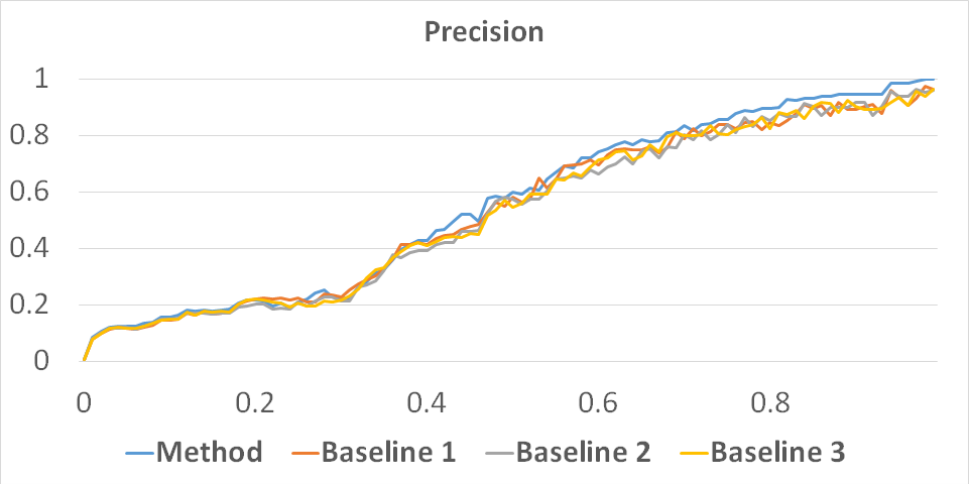


Figure 6.6: Fast-solve policy: graph of the Precision metric observed for all values of the threshold τ within the interval $[0.0, 1.0]$.

increases for both precision and recall values with respect to the baselines. In particular, we can observe how the recall starts to be 1.0 from τ values higher than 0.75.

By focusing on the recall graphs, there are a couple of aspects that attracted our attention: the high recall observed for low τ values and the subsequent drop. The former is justified by the fact that the patient starts the interaction by providing some information, i.e. some slots are filled on startup. Thus, for low values of τ, the possibility that the correct goal is contained within the set of goals passed the threshold is high. On the contrary, the precision values are low since, especially if the filled slots are very popular, many goals can be satisfied by considering only the information provided on startup. The latter is given by considering that, medium values of τ means that the agent is in the middle of acquiring information and it seems that in many cases there are goals, then resulted to be the wrong ones, that are satisfied with this partial knowledge. Future work will consider investigating this behavior.

Finally, by comparing the two policies, even if the proposed framework outperforms the baselines in both of them, it resulted to be more effective in the *cautious* one. A possible reason is that by using policies like the *fast-solve* one aiming to reduce as much as possible the number of steps to achieve a goal, strategies relying on frequencies can be more effective, especially for lower values of τ. However, it is important to remark two aspects about the *fast-solve* policy results. First, real-world scenarios within the healthcare domain consider precision values in the range of $0.4 - 0.6$ not acceptable by domain experts. Hence, the possibility of adopting low values of τ in practice is very low. Second, the proposed framework demonstrated to be more effective with respect to the baselines by exploiting a dramatically lower number of interactions. The latter is a very important aspect within the healthcare domain since it allows to understand which is

the current healt status of a patient and, at the same time, to reduce the effort to provide the knowledge required to reach a conclusion.

6.7 Final Remarks

This Chapter presented a goal-based framework to support dialogue management within the healthcare domain. It discussed the concept of information usefulness for an agent that conducts health-related dialogues and it formalized the metrics that make it possible to compute the usefulness value of a missing slot. A running example about diagnosis was used to show how the framework can be instantiated.

The proposed usefulness measure has as one of its main benefits the generation of a "short" information acquisition process due to the careful selection of the slot that most contributes with information at each iteration of the dialogue. As further benefits, this approach avoids the need of large historical databases and the costs of training the dialogue system to obtain a reliable output.

By proposing this approach, we intend to contribute as a heuristic measure that covers the needs of the health domain to support the choice of the next dialogue action. This heuristic can be integrated with some of the strategies currently used for dialogue management (e.g. automated planning [10] or finite state machines [67]), which also can exploit the belief base B_a to determine the dialogue state.

This work opens several research directions to be accomplished in the future. The use of fuzzy sets, for example, would be an interesting extension of the proposed framework as it would allow the end-user to confirm a slot to a certain degree. Also, the integration of further strategies that elicit the retrieval of information should be investigated. The use of a mixed-initiated design, associated with the manipulation of open answers is expected to

elicit spontaneous input of (useful) information by the patient. However, this hypothesis must be validated through extensive studies.

Finally, this approach was integrated into a plan-based architecture for the management of health dialogues. This integration is a key feature of the approach proposed in this thesis and it is discussed in Chapter 7.

Chapter 7

OntoPlanDM: An Ontology and Plan-Based Dialogue Manager

This chapter introduces *OntoPlanDM*, the approach designed for the automated generation of dialogue managers, the main subject of this thesis. *OntoPlanDM* is composed of three modules, which are detailed in the next sections of this chapter. Figure 7.1 presents an overview of these modules.

7.1 Model Acquisition

The first module is dedicated to the acquisition of knowledge on the topic of the dialogue system. At this stage, all domain-specific knowledge that is relevant for the dialogue session must be populated in Convology. This process occurs in two steps. First, a *dialogue author* with knowledge of the dialogue domain is responsible for populating a few main classes of the ontology. Second, this knowledge is exploited by a *populator* that instantiates further classes in the Convology ABox.

The knowledge to be instantiated by the dialogue author corresponds to the following classes:

- *Subdialog*: as many instances as the necessary sub-dialogues to cover the domain;

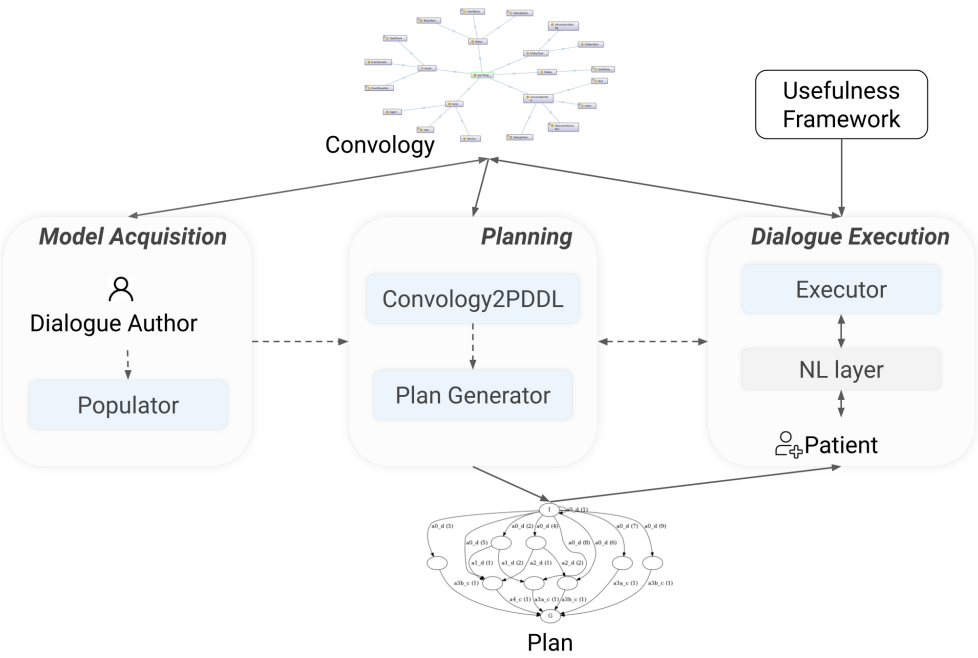


Figure 7.1: The overall picture of the process to generate a dialogue manager.

- *UserStatus*: instances to define the possible goals for each sub-dialogue;
- *Slot* and *StatusItem*: for an information-seeking dialogue, slots are instances of everything that can be asked/informed during a dialogue session. A deliberation dialogue, instead, requires an instance of *StatusItem* to keep the information to be communicated.

A few properties of each of these classes and some relations among them must also be informed by the dialogue author. Detailed instructions for dialogue authors are available in Appendix A.

Following the dialogue author's task, a *populator* is triggered. This populator has the aim of reducing the burden of the dialogue author when instantiating Convology. This is possible since some instances are part of knowledge that is common to any dialogue and, therefore, can be automatically generated. It is important to note that, although these instances are common, they are directly related to domain knowledge and can be derived only from the information provided by the dialogue author. But

before describing these instances, let's take a look at the list of system and user actions that are supported by *OntoPlanDM*.

As mentioned in Section 2.2.1, *OntoPlanDM* covers *information-seeking* and *deliberation* dialogue types. The corresponding list of the *dialogue actions*, i.e. system actions, covered by *OntoPlanDM* in an information-seeking sub-dialogue is given in Table 7.1 and in a deliberation sub-dialogue in Table 7.2. For each action, the Tables indicate which subclass of *DialogAction* represents it in Convology. Table 7.3, instead, lists the *intents* that can be extracted from user actions and interpreted by *OntoPlanDM*. All intents are represented by the *Intent* class in Convology and the Table signs the type of sub-dialogue that can address it. All Tables include a short description of each item.

Name	***DialogAction* Subclass**	**Description**
request	*ClosedQuestion*	Makes a request to the user
initiate	*OpenQuestion*	Initiates the dialogue session
fallback	*OpenQuestion*	Asks the user for clarification as something was not understood
report	*Feedback*	Reports a classification
acknowledge	*Feedback*	Acknowledges an information received

Table 7.1: Dialogue actions covered in an information-seeking sub-dialogue.

Name	***DialogAction* Subclass**	**Description**
feedback	*Feedback*	Provides a recommendation to the user
confirm	*ClosedQuestion*	Confirms that a recommendation was followed
recommend	*Feedback*	Emphasizes a recommendation to the user
recommend_next_step	*Feedback*	Provides the user with the next recommendation

Table 7.2: Dialogue actions covered in a deliberation sub-dialogue.

Name	Information seeking	Deliberation	Description
inform	X	X	User wishes to provide some information
fallback	X		User provides an information not understandable to the agent

Table 7.3: Intents recognized by *OntoPlanDM*.

Based on the definitions provided in these Tables and on the knowledge instantiated by the dialogue author, the following classes are instantiated by the populator:

- *Intent*: each *Slot* instance gives origin to an intent, meaning that the user can *inform* this slot during the session;
- *ClosedQuestion*: each *Slot* instance gives origin to a *request*, i.e. a question about the slot to be asked by the system;
- *RelevantInformation*: two instances (*YesValue* and *NoValue*) are generated to define boolean values as the possible values of a slot. These instances are then associated to each *Slot* instance through the property *hasAllowedRelevantInformation*;
- *Feedback*: each *UserStatus* instance gives origin to a *Feedback* instance that will output the goal achieved. In addition, an instance of *Feedback* is generated for each *StatusItem* instance associated with a *Subdialog* of type deliberation. Finally, an instance called *acknowledge* is created to acknowledge a piece of recognized information provided by the user.

Table 7.4 illustrates an example of instances related to the *Slot* instance named *chest_tightness*. While *chest_tightness* is informed by the dialogue author, the further instances and relations are generated by the Convology populator.

Instance Name	Class	Properties and Range
YesValue	RelevantInformation	
NoValue	RelevantInformation	
chest_tightness	Slot	hasAllowedRelevantInformation {YesValue,NoValue}
inform_chest_tightness	Intent	hasSlot chest_tightness
request_chest_tightness	ClosedQuestion	hasRelevantIntent inform_chest_tightness

Table 7.4: Convology instances related to a *Slot* instance.

The knowledge derived in this module must be translated into a planning problem that will be compiled to provide the dialogue policy. This process happens in the next module of *OntoPlanDM*, the *planning module*, which is discussed in the next section.

7.2 Planning Module

The role of the *planning* module is to specify the components of the dialogue in a way that it can be compiled by a state-of-the-art planner, with the aim of automatically generating the dialogue policy (i.e. plan) to be integrated into the dialogue system. To successfully achieve the expected policy, a few planning aspects must be considered. First, this work exploits *dynamic* planning [50]. That is, the generation of a plan occurs first offline, i.e. when the dialogue session starts. However, to accommodate changes in the real-world knowledge, the policy might require updates on interaction time, which implies updates in the planning problem. These aspects are discussed in section 7.3.3.

Second, the planning type adopted in this work is the FOND planning, which was discussed in section 4.5.1. Non-determinism is suitable to the kind of problem exploited in this work since, although a set of possibilities exists, it is not possible to know the exact outcome (effects) of

most dialogue actions before their execution. By addressing dialogue actions as non-deterministic, the plan will cover all the different alternative paths that the dialogue might take. Meanwhile, as state tracking in this work does not require high complexity (if compared to a spoken dialogue system), *full observability* is obtained by adopting a threshold to the confidence score attached to the *Intents* received from the NLU output (data property *hasConfidenceScore* in Convology).

Finally, within FOND planning, we address *strong-cyclic planning* (SCP) [34]. Through this kind of solution, it is possible to guarantee that there is always a path to reach the dialogue goal (task-completion), repeating an action when necessary information is still missing (e.g. clarification questions can be made).

Prior to actual planning, the planning module has a pre-processing phase, which concerns the translation of the knowledge within Convology into planning terms. This translation is based on correspondences that can be observed in both techniques and that address the dialogue needs (e.g. *DialogAction* in Convology and planning actions). To keep an interface between both, the specification of the planning problem replicates the vocabulary used in Convology. However, while some aspects within the planning specification are mapped directly from concepts or instances in Convology, others correspond to modeling decisions that are necessary to obtain the expected plan. Examples of these modeling decisions is the handling of deadends or strategies that integrate the usefulness framework; both are discussed along this chapter. Like in Convology, the specification of the planning problem is formulated in a way that it can be reused to different scenarios within healthcare dialogues.

The planning specification is given in PDDL, making it possible to invoke an external automated planner to generate the solution that will be used to manage the dialogue session. Besides obtaining a structured pol-

icy that selects the most appropriate actions towards the dialogue goal, the translation of Convology to PDDL will easily reflect in the plan any possible update made in the domain-specific knowledge (e.g. add or delete slots). The two PDDL instances, namely planning *domain* and *problem* (section 4.2), are described in the next sections.

7.2.1 Planning Domain Specification

This section specifies the components of the planning *domain* instance.

Types. A type can be specified for each object in PDDL. The *types* within the planning domain are mapped from three concepts that manage domain-specific information within Convology: *Slot*, *RelevantInformation*, and *UserStatus* (see below). These types are available for both information-seeking and deliberation dialogues.

```
(:types
Slot
RelevantInformation
UserStatus
)
```

Constants and variables. Constansts refer to a specific object in the modeled world. Variables are used to define any applicable object of a given type from the modeled world. In this work, *objects* are specified within the planning *problem* (Section 7.2.2) and they are addressed in the planning *domain* as either constants or variables, according to the need.

Predicates. Predicates are atomic statements concerning the problem. They can take both constants and variables as arguments. Table 7.5 lists all predicates specified within the planning domain, also identifying their arguments (if any), the type of sub-dialogue for which they are available and a short description of their purpose.

Table 7.5: Predicates specified within the planning domain.

Predicate	Arguments	Information seeking	Deliberation	Description
hasRelevantInformation	?x - Slot ?v - RelevantInformation		X	Indicates which value is given to a slot
has_value	?s - Slot	X		Indicates whether a slot has a value
fallback_intent		X		A flag to indicate that the user's input was not understood
unexpected_intent_received		X		A flag to indicate that the user informed something that was not expected at the current state
initiate		X		A flag to indicate that no information was provided yet
is_most_useful	?s - Slot	X		Identifies the most useful slot
search_most_useful		X		A flag to notify when the action that searches for the most useful slot should be called
possible_classification_us	?us - UserStatus	X		A flag to indicate that a classification might have been achieved for the argument ?us
GOAL_ACHIEVED		X	X	A flag to indicate that the dialogue goal was achieved
enable_confirmation	?x - Slot		X	A flag to force the agent to confirm if the given recommendation was already followed

Actions. Actions are used to change the world state. The specification of an action must include its *parameters* (if any), the *preconditions* that must be met for the action to be applied; and the *effects* that result when this action is applied.

The planning actions defined within *OntoPlanDM* model the dialogue actions and they are classified into two types: DIALOGUE and SERVICE, which were borrowed from [24]. Actions of type DIALOGUE correspond to actions that are originated from *DialogAction* instances in Convology. These actions result in interactions with the end-user, such as questions or feedbacks that are converted to a natural language output. SERVICE actions, on the other hand, correspond to actions that do not result in natural language output. These actions are specified to manage aspects of the dialogue session that occurr in the background. Examples are actions to set constraints that avoid deadends (Section 7.2.4) and the abstract action *search_most_useful_slot*, whose aim is to integrate the usefulness framework to identify the most useful slot at the current dialogue state (Section 7.2.3).

Each *Subdialog* type (i.e. *InformationSeeking* and *Deliberation*) contains a different set of actions. They are described next.

Information-Seeking Actions

The specification of the planning domain of a *Subdialog* instance of type *InformationSeeking* includes 4 main types of actions:

initiate : This action of type DIALOGUE is translated from the *Open-Question* instance named *initiate* in Convology (Table 5.1). The purpose of this action is to start the conversation with an open question (e.g. "*How do you feel today?*") that aims to retrieve any of the available *Slots*. Therefore, the precondition for this action is that all *Slot* instances in Convology have no value. This condition is abstracted by the predicate *initiate*, which

will be set by the parser in the planning problem only when no information was received yet. Note that this action could have a non-deterministic effect specifying that one or more of the empty *Slots* received a value (after its execution). However, this would result in a huge state space for domains that include a large number of *Slot* instances. Therefore, the cost of anticipating all possibilities is avoided with the modeling of a deterministic effect that leads the plan to its goal. This way, the information received is handled by the manager and a replanning operation is conducted to generate a new policy (see Section 7.3.3). An example of the *initiate* action[1] in PDDL is given below:

```
:action initiate
       :precondition (initiate)
       :effect (GOAL_ACHIEVED)
```

request and search_most_useful_slot : All *ClosedQuestion* instances related to a *Subdialog* instance are translated to a *request* action in the planning domain. Request actions are also of type DIALOGUE as they will be translated to a question that is made to the end-user. The specification of *request* actions integrates the usefulness framework described in chapter 6 and a SERVICE action named *search_most_useful_slot* supports the information retrieval process. The specification of both actions is detailed later in this chapter, in section 7.2.3.

report : All *UserStatus* instances related to a *Subdialog* instance are translated to a *report* action in the planning domain. These actions are used for the classification of the *UserStatus*, after retrieving the expected information. Like the actions for information retrieval (i.e. *request*), the specification of *report* actions is based on the usefulness framework and it is detailed in section 7.2.3.

[1] All PDDL actions displayed in this chapter were simplified for easier comprehension.

In addition to these main actions, the two actions described next were specified to address an information-seeking sub-dialogue:

enable_fallback_question : This action is translated from the *OpenQuestion* instance named *fallback_question* in Convology. This action is intended to be used when no intent could be recognized with satisfying confidence. That is, when the user's input cannot be matched to any intent, the plan includes this action to inform the users that the last input was not understood and invite them to repeat the information in a different way. Therefore, this is an action of type DIALOGUE. The precondition of this action is the predicate *fallback_intent*. To avoid the cost of anticipating all possible outcomes of this open-question and force a replanning operation (section 7.3.3), the action has as its effect the predicate *GOAL_ACHIEVED*.

feedback_unexpected_intent : This action is translated from the *Feedback* instance also named *feedback_unexpected_intent* in Convology. This action is used to acknowledge some (non-expected) information provided by the end-user. Its precondition is the predicate *unexpected_intent_received*, which works like a flag to force the execution of this action. As effects, the flag is removed and the predicate *search_most_useful* will lead the agent to continue the information retrieval process. A simplified snippet of the *feedback_unexpected_intent* instance translated to this action in PDDL is given below:

```
(:action feedback_unexpected_intent
       :precondition (unexpected_intent_received)
       :effect(and
              (not(unexpected_intent_received))
              (search_most_useful)
       )
)
```

Deliberation Actions

In Convology, the output of an information-seeking sub-dialogue, i.e. the classified *UserStatus*, has a trigger to a *UserStatus* instance associated with the subsequent deliberation sub-dialogue. To address a deliberation sub-dialogue, the parser exploits this *UserStatus* instance and its associated *Feedback* instances to specify 4 types of DIALOGUE actions in PDDL:

feedback: This action will give origin to an output (i.e. a recommendation) that informs the end-user how to proceed, given the received classification. The precondition of this action specifies that this recommendation was not given yet. This information is obtained from the association of the *UserStatus* instance with a *StatusItem* instance (see section 5.4.2), that is, the precondition requires that the *StatusItem* has no value (*RelevantInformation*). Its deterministic effect will enable a flag to the next action, a *confirm* action. A sample of the *feedback* action is given below.

```
(:action feedback
       :precondition (hasRelevantInformation recommendation_x empty)
       :effect (enable_confirmation recommendation_x)
)
```

confirm: This action was specified with the aim of giving the possibility to the end-user to confirm whether the given recommendation is an action already taken or not (e.g. the patient already took the recommended medicine). As its preconditions, it has the flag (i.e. predicate) *enable_confirmation* and the requirement that the associated *StatusItem* is empty. As effects, (i) this flag is disabled and (ii) the corresponding *StatusItem* is identified as not empty anymore. The latter is a non-deterministic effect since the value given to the *StatusItem* affects the next action to be taken.

```
(:action confirm
       :precondition (and
              (hasRelevantInformation recommendation_x empty)
              (enable_confirmation recommendation_x)
       )
       :effect (and
              (not(enable_confirmation recommendation_x))
              (not(hasRelevantInformation recommendation_x empty))
              (oneof
                     (hasRelevantInformation recommendation_x NoValue)
                     (hasRelevantInformation recommendation_x YesValue)
              )
       )
)
```

recommend: This action was specified to emphasize the recommendation for the received classification when the end-user confirms that this action has not been taken yet. This action achieves the goal of the deliberation sub-dialogue.

```
(:action recommend
       :precondition (hasRelevantInformation recommendation_x NoValue)
       :effect (GOAL_ACHIEVED)
)
```

recommend_next_step: On contrary to the previous action, this action was specified to address the situation when the provided recommendation was already addressed by the end-user. This way, the deliberation sub-dialogue has its goal achieved and the next sub-dialogue should be initiated right away.

```
(:action recommend_next_step
       :precondition (hasRelevantInformation recommendation_x YesValue)
       :effect (GOAL_ACHIEVED)
)
```

7.2.2 Planning Problem Specification

The *problem* instance, which is based on the domain instance, is also specified according to the knowledge available in Convology. This instance defines (i) the *initial* state, the (ii) *goal* state, and (iii) the set of *objects* of the modeled world.

The *initial state* is defined according to the type of sub-dialogue. When no information was retrieved yet for an information-seeking sub-dialogue, for example, the initial state includes only the predicate *initiate*. Upon replanning, the initial state is updated to include all information available in the current state of the dialogue (section 7.3.3). The initial state of a deliberation sub-dialogue, on the other hand, specifies the predicate *hasRelevantInformation* for each *StatusItem* instance associated with the active *UserStatus* (e.g. *hasRelevantInformation recommendation_yellow empty*[2]).

For both sub-dialogue types, the *goal state* is abstracted with the predicate *GOAL_ACHIEVED*. Since different combinations of parameters lead to the dialogue goal, it is enough to specify this predicate as an effect of actions that lead to the goal. In some cases, when goal-achievement is a possibility but it is not guaranteed upon action execution, like *report* actions (section 7.2.3), this predicate is associated with a non-deterministic effect.

Finally, the finite list of *objects* represents the pieces of information that are available within the problem. This list is obtained from the instances of the three classes that were mapped to planning types: *RelevantInformation*, *Slot*, and *UserStatus*. Consequently, each object is associated with its type. If replanning is required at some point of the dialogue, it is enough to update the initial state with updated values for these objects.

[2]Empty values mean that the recommendation was not given yet.

7.2.3 Integrating Planning and Information Usefulness

The specification of the dialogue as a planning problem must take into account some peculiarities of the healthcare domain. As previously mentioned in this thesis, a health dialogue system cannot rely on automatic responses that may be valid but suboptimal, leading a patient to risky situations. Non-binary classification and obtaining information that allows the agent to both detect and exclude undesired situations, as discussed in chapter 6, are aspects to be considered also in the planning problem. Therefore, the specification of information-seeking sub-dialogues implemented definitions from the *usefulness framework* introduced in chapter 6.

Since the generation of the planning problem is grounded on the information available in Convology, the properties below were added to the ontology to address the usefulness framework; their usage is discussed along this section:

- *searchMostUseful*: a data property associated with a *Subdialog* instance;
- *hasMostUsefulSlot*: an object property associated with a *Subdialog* instance. Its range, a *Slot* instance, represents the current most useful slot;
- *hasWeight*: a data property to address the weight given to a *Slot*;
- *hasThreshold*: a data property to specify the threshold required for the classification within a *Subdialog*;
- *hasPriority*: a data property to specify priorities among *UserStatus* instances.

In summary, three types of planning actions are based on the usefulness framework: *report*, *request*, and *search_most_useful_slot*. Details on

the definition of these actions as well as the predicates included in the planning domain to address the usefulness framework are described in the next sections.

Report actions

Report actions correspond to classification actions, that is, actions to manage the goal achievement of the information-seeking sub-dialogue. The classification specified within the usefulness framework (section 6.4.3) is dynamic, taking into account aspects such as the slot-values acquired and their associated weights. That is, instead of predefining a set of *Slot* that classify a given *UserStatus*, different combinations of slot-values might lead to a classification. This way, specifying every possible combination of *Slot* instances as preconditions of classification actions is not a feasible solution and the adoption of a more abstract specification is required.

To provide such an specification, the modeling strategy adopted in this work includes the predicate *possible_classification_us* (Table 7.5). This predicate abstracts the preconditions of a report action, working as a flag that indicates that the agent must check if a classification is achieved for the corresponding *UserStatus*. Since every new information received by the agent might lead to a classification, this flag is raised as an effect of a *request* action (section 7.2.3). As a consequence of this abstraction, it is only during dialogue execution, under the deployment of the usefulness framework, that it is possible to determine whether the just-received information leads the agent to a classification or not. This way, *report* actions can be considered a mix of SERVICE and DIALOGUE actions and they have a non-deterministic effect. While the former is given when no classification is reached, the latter is a consequence of a classification, which will result in an output to the end-user. A sample of a classification action for a *UserStatus* instance (*classification_yellow*) is shown next:

```
:action report_classification_yellow
:precondition possible_classification_us classification_yellow
:effect (oneof
           (and
              (not(GOAL_ACHIEVED))
              (search_most_useful)
           )
           (GOAL_ACHIEVED)
         )
```

Note that, when the goal is not achieved, the report action has an additional effect with the predicate *search_most_useful*. This predicate is discussed in the next section.

Information Retrieval Actions

Request actions are designed to guide the agent on information retrieval. To address the needs of the health domain in this process, the usefulness framework, whose validation was discussed in section 6.6, was integrated to these actions.

Some important aspects had to be observed to model request actions. As shown in chapter 6, the most useful slot changes based on the information retrieved in each dialogue turn. Therefore, like in a classification, the definition of the "most useful slot" can only be given in real-time, during dialogue execution. Anticipating usefulness values in the preconditions of the request action is not feasible mainly because the same slot would present different usefulness values in different states. An alternative to address this scenario would be the adoption of *online planning*, that is, interleave planning and execution in each state. However, as reported in the literature [72], online planning is not a feasible solution for huge problems, which is the case in most health scenarios. Therefore, the FOND planning problem must be specified considering its *offline* generation. But how can the (online) usefulness framework be addressed in this specification?

Given the requirements and restrictions discussed, the solution for the problem of integrating the online usefulness measure and offline plan generation relies on *abstraction*. That is, the planning problem includes abstract actions that instruct the agent to *search* and *request* the most useful slot, anticipating the paths for the different alternatives. However, it is only during execution time that the agent deploys the usefulness framework and determines the most useful slot. This strategy also makes it possible to separate domain knowledge reasoning from dialogue management, supporting the flexibility and portability of the approach [2].

The predicates *search_most_useful* and *is_most_useful* (Table 7.5) were specified in the PDDL to support the abstraction of the usefulness framework. They are represented in Convology by the the data property *searchMostUseful* and the object property *hasMostUsefulSlot*, respectively.

A SERVICE action was defined to state that a search for the most useful slot is the next step to be conducted by the agent. The precondition for this action is the predicate *search_most_useful* and as effect, it identifies the most useful slot. However, this identification is non-deterministic, which implies that any of the listed slots could be the most useful one. The actual value can be determined only upon action execution. A simplified example of this action is given below:

```
:action search_most_useful_slot
:precondition search_most_useful
:effect (oneof
           (is_most_useful s1)
           (is_most_useful s2)
           (is_most_useful s3)
        )
```

Finally, *request* actions, which are translated from *ClosedQuestion* instances in Convology (section 7.2.1), specify two main preconditions: (i)

the associated slot[3] is empty and (ii) the slot is the most useful. The effect of a *request* action, on the other hand, is non-deterministic since its execution may result in the following:

- (i) identify that a value was received for the slot (i.e. it is no longer empty) and (ii) inform which possible classifications[4] might have been achieved as a consequence of filling this information;
- a *fallback_intent* is reported, meaning that the user's answer was not understood by the agent.

A simplified example of a *request* action is given below:

```
:action request_s1
:precondition (and
                (is_most_useful s1)
                (not(has_value s1))
              )
:effect (oneof
           (fallback_intent)
           (and (has_value s1) (possible_classification_us x))
)
```

Figure 7.2 illustrates a short graph of the resulting plan for the information retrieval process of a single slot. While each node represents a dialogue state, the edges represent actions applied to that state[5]. Note that the policy starts by searching for the most useful slot. Then it is followed by a request action that can lead either to a fallback question or to a state where the information was acquired. When the information is acquired, the policy checks for a classification (report action). At this point the dialogue goal might be achieved. As this is a short example and no further information can be acquired, the other outcome of the report

[3]Obtained from the association with the *Intent* (*hasRelevantIntent*).
[4]Note that the precondition for a *report* action is now satisfied.
[5]Note that in FOND planning a single action might result in different states (e.g. *request_slot1*).

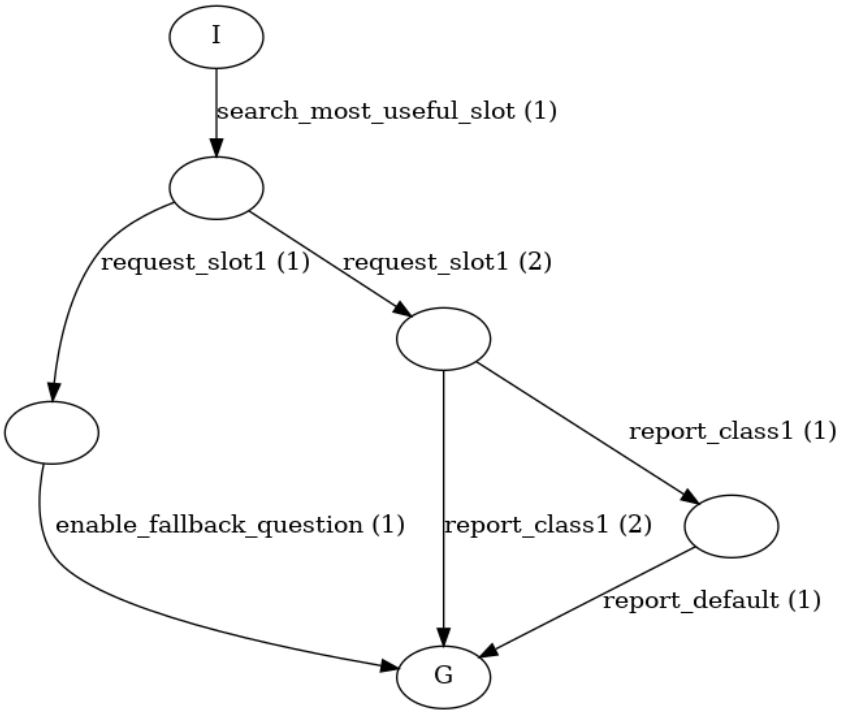


Figure 7.2: Sample of resulting policy to retrieve one slot.

action is to report a default classification. However, in a domain with further slots, the next action would correspond to a new search for the most useful slot.

7.2.4 Handling Deadends

The planning specification for information-seeking sub-dialogues given so far would reach two deadends and not result in the SCP mentioned in section 7.2. Therefore, this section introduces the strategies employed to avoid these deadends.

First, let's observe that a *request* action has as preconditions that a slot is empty and it is the most useful one. At a certain point, the plan graph would reach a node that says that slot $s2$, for example, is the most useful but it is not empty anymore, resulting in a deadend. This condition is logically possible, but it would never happen in real-world situations as

the usefulness measure states that the most useful slot belongs to the set of missing slots (Definition 9). To manage this issue logically and still reach an SCP, the action *service_deadend* (type: SERVICE) was included in the planning domain. This action catches in its preconditions the variables that would cause the deadend and its effect redirects the plan to a new search for the most useful slot:

```
:action service_deadend
:parameters (?s - Slot)
:precondition (and
                (is_most_useful ?s)
                (has_value ?s)
              )
:effect (and
           (not(is_most_useful ?s))
           (search_most_useful)
        )
```

The second deadend might occur when no classification can be reached from the information retrieved. If no extra slots are available to be filled (i.e. all have a value) and, therefore, no request action can be executed, a default classification action was defined to avoid a deadend and guarantee that the plan still has a solution. Different from the deadend discussed previously, this is a situation that might occur in real-world situations (e.g. when high thresholds are defined). Therefore, this situation must be addressed not only in the plan but also in the real-world variables. This way, the data property *isDefaultStatus* was defined in Convology to inform which *UserStatus* is the default classification, that is, which *UserStatus* will be considered the goal when no other can be classified. This definition is domain-specific and, therefore, must be provided as part of the domain knowledge for each information-seeking sub-dialogue. The following DIALOGUE action was specified in the planning domain to address these scenarios:

```
:action report_default
:precondition (and
                (has_value s1)...(has_value sn)
                (not(is_most_useful s1))...(not(is_most_useful sn))
                (not(possible_classification_us x1))...
                (not(possible_classification_us xn))
              )
:effect (GOAL_ACHIEVED)
```

7.2.5 Plan Generator

The resulting PDDL files that correspond to the planning domain and problem are to be compiled by an automated planner. An efficient state-of-the-art planner for generating strong-cyclic FOND plans is the Planner for Relevant Policies (PRP)[6] [82]. In this work, PRP is exploited to generate a plan, that corresponds to the dialogue policy to be integrated into the dialogue system. The next section describes the dialogue execution, specifying when the planner is triggered.

7.3 Execution Module

The execution module is part of the agent's online reasoning. Its role is to interpret the policy generated in the previous stage, using it to manage the ongoing dialogue session. This module integrates the natural language layers that are in charge of interacting with the end-user by (i) interpreting the user's input and (ii) translating the system's response to a human-understandable output. The execution module also exploits two other components: (i) Convology, which is used as a knowledge base for the execution algorithm and (ii) the usefulness framework, which extracts the most useful information to be asked and supports classification.

[6]https://github.com/QuMuLab/planner-for-relevant-policies

The dialogue executor invokes the NLU module to interpret the input received from the end-user. Then, it populates Convology with the new information and identifies the new state. At each state of the dialogue, the executor will consult the dialogue policy to learn which action should be executed next. Note that the action informed by the plan is a high-level definition of the actual implementation of the action. The planning action named *'search_most_useful_slot'*, for example, must be mapped to a code that deploys the usefulness framework. This work adopts the term *transformer* from [24] when refering to the mapping of the planned action into its actual implementation. When the action results in a *DialogAction* instance, it is then sent to the NLG component in charge of generating the human-understandable response given by the system.

By adopting the asthma scenario (section 2.1) as the dialogue domain, this section discusses the whole dialogue management process from the dialogue executor perspective. While section 7.3.1 discusses the expected flow of the resulting dialogue policy, sections 7.3.2 and 7.3.3 detail how deviations from the planned policy are handled. Differently from section 5.4, that described only how the dialogue agent named *PuffBot* exploits Convology, this section addresses the integration of the planned policy with Convology to manage the dialogue. Figure 7.3 shows a sample conversation between *PuffBot* and an end-user, illustrating *PuffBot*'s behavior with respect to state-action selection. The plan shown in the Figure is an abstraction, as the policy is replanned a couple of times. Replanning occurrences are omitted in the Figure, but are discussed in the next sections.

7.3.1 Plan-Based Dialogue Management

In this work, the resulting plan corresponds to the dialogue policy that dictates the behavior of the dialogue agent. Every action executed by the agent must be informed by the dialogue policy, that is, it must be planned

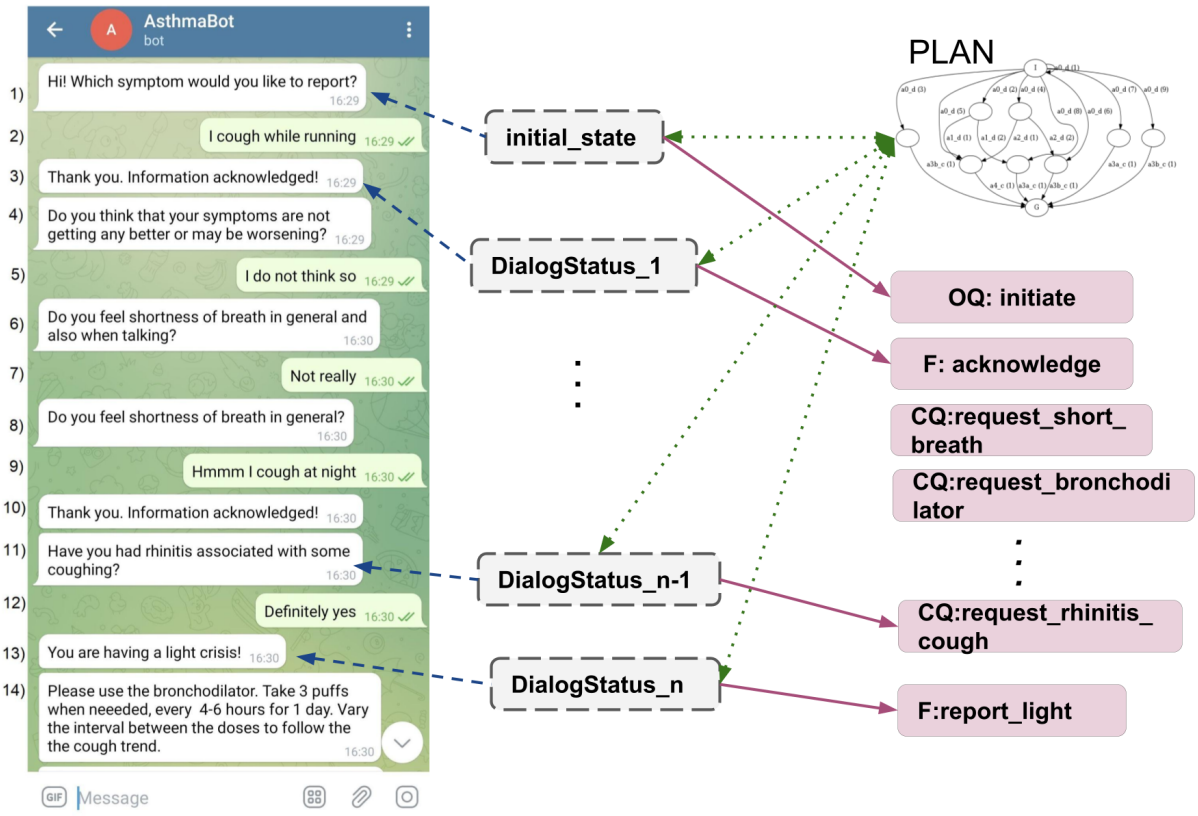


Figure 7.3: Illustration of the behavior of *PuffBot* to decide the next system action in a dialogue session.

in advance. At the beginning of each sub-dialogue, an initial dialogue policy is generated by the planner according to the information available in Convology at that stage. This policy is dynamically replanned during the dialogue session to adapt to updates in the available knowledge. Both replanning operations and sub-dialogue switches are transparent to the end-user.

The first dialogue policy is generated right after the triggering of a new conversation. This triggering depends on the implementation of the agent. For the asthma agent, an information-seeking sub-dialogue is triggered after the end-user sends the first message to the agent (*start* command omitted in Fig. 7.3). When the sub-dialogue starts, an instance of *DialogStatus*

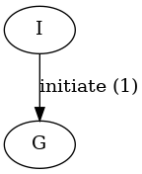


Figure 7.4: Initial dialogue policy.

is created in Convology (dashed boxes in Fig. 7.3) and the information on which *DialogAction* should be executed in this state (as well as in all further states) is retrieved from the dialogue policy (dotted lines in Fig. 7.3).

At the beginning of the dialogue session, it is common that the agent has no information about the current situation. Therefore, according to the specifications provided in section 7.2.1, the initial dialogue policy includes only the *initiate* action (Figure 7.4). As per its definition, *initiate* is a deterministic action that is mapped by the *transformer* to an *OpenQuestion* instance of Convology (box "OQ:initiate" in Fig. 7.3). This dialogue action (and all further) is processed by the NLG component that generates the system's utterances on the left side of the dialogue sample in Fig. 7.3. The initial question is open to leave the possibility to the end-user to provide some relevant information that will give the agent some direction on the current situation. Outcomes are not anticipated for this action as, at this point, all alternatives are possible. For not simplistic domains (with not only two slots, for example), this would result in a huge search space, slowing down the agent's response. As a consequence of the non-anticipation of the action's possible outcomes, the user's response (Fig. 7.3 line 2) will be treated as a non-expected input, which is followed by a replanning operation. The handling of this kind of input is discussed in section 7.3.2 and the *Feedback* shown in line 3 is discussed in section 7.3.3.

After recognizing the user's answer to the initial question, a new *DialogStatus* is created in Convology (Fig. 7.3 *DialogStatus_1*) and the agent checks the updated policy to learn the next action to take. Figure 7.5

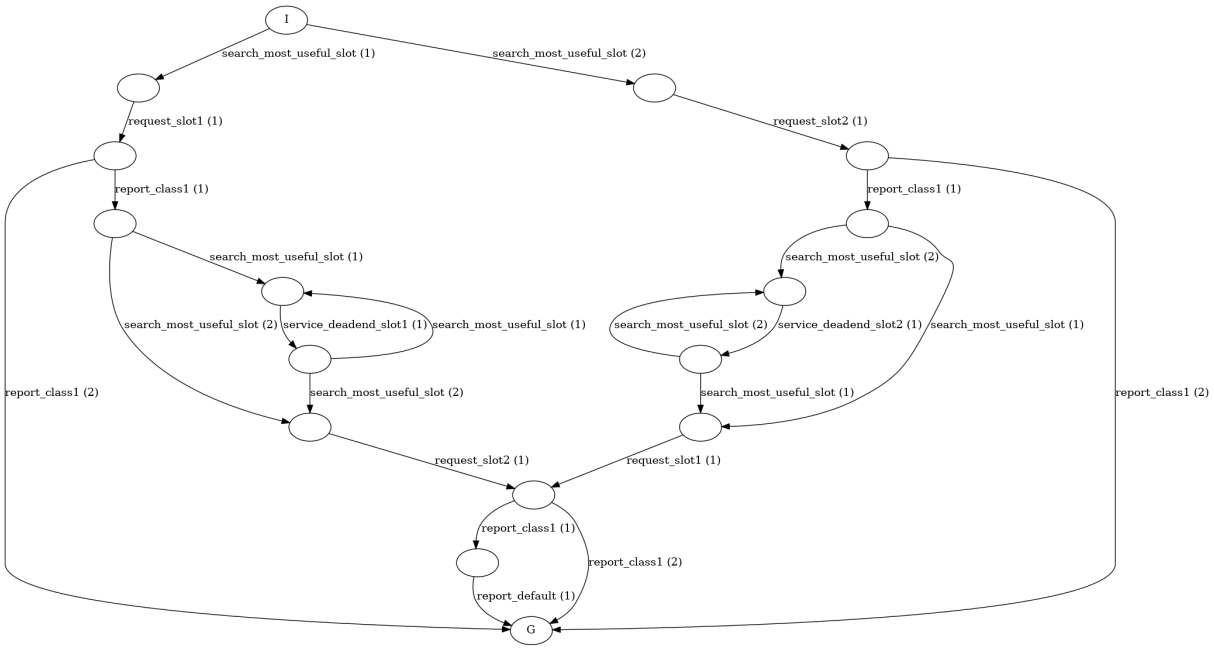


Figure 7.5: Dialogue policy for a problem with two slots.

illustrates a simplified[7] version of the updated policy.

As at this point the agent has gained some information, the planned policy now focuses on acquiring information based on usefulness values. As previously discussed, the plan abstracts the usefulness framework (section 7.2.3). That is, it presents a high-level action that tells the agent that its next step is to find the most useful slot, but the determination of which is the most useful slot is made by the *transformer* that deploys the usefulness function. The identified slot is then flagged in Convology and the agent must check the next planned action.

Following the specification of the planning actions made in section 7.2.1, the identification of the most useful slot satisfies the preconditions of a *request* action. The *transformer* maps this action to a *ClosedQuestion* instance (e.g. translated to the utterance in line 4), resulting in an *EventQuestion*. For readability sake, *Event* instances are omitted in Fig. 7.3.

[7]This policy was generated for a problem including only two slots and deterministic *request* actions.

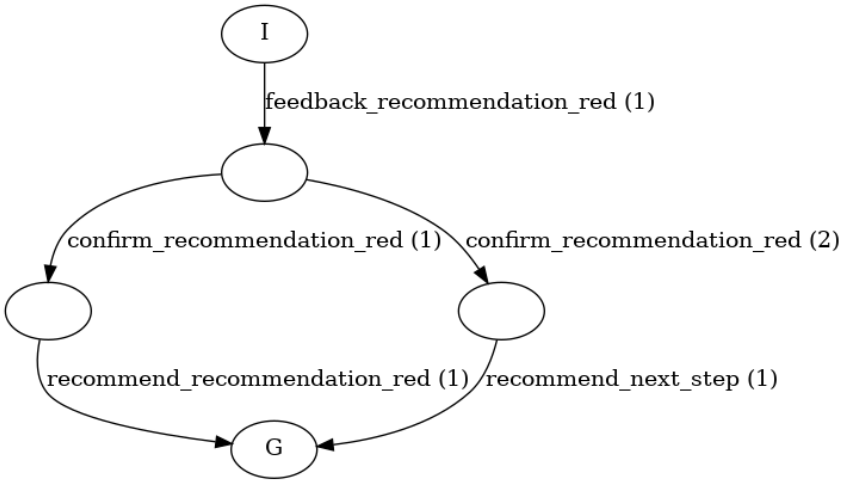


Figure 7.6: Sample of resulting policy for a deliberation sub-dialogue.

A user's input (*UserEvent*) is expected to follow the system's question, taking the dialogue to a new state. State recognition is discussed in the next section. When the agent processes the user's input and is able to recognize some information on it[8], the next planned action should be a classification action. The handling of classification actions by the executor is also discussed in a following section.

In the asthma scenario, the subsequent deliberation sub-dialogue starts. As in any sub-dialogue, the agent must then check the new policy (Figure 7.6) to learn the action to execute in the initial dialogue state. Line 14 in Figure 7.3 is the output of a *feedback* action, which presentes a recommendation to the patient with respect to the previous classification. As can be seen in Figure 7.6, after this message, the policy presentes two further actions that will be translated to natural language: (i) a *confirm* action that will inquiry the patient on whether the provided recommendation was already followed (e.g. "*Have you already done that?*"), and (ii) a *recommend* action that will either reinforce the already given recommendation (e.g. "*Then, please follow this recommendation.*") or provide a further recom-

[8]The procedure adopted when no intent can be recognized is discussed in Section 7.3.2.

mendation (e.g. "*In this case, I recommend you to...*"). The conclusion of this sub-dialogue will trigger the next sub-dialogue or conclude the session.

Dialogue State Tracking

In *OntoPlanDM*, the non-deterministic actions foresee a set of different states that can be reached after their execution. Therefore, the FOND plan anticipates all states that are possible within a sub-dialogue. However, it is only during dialogue execution that it is possible to determine what changed in the real-world as a consequence of the execution of the last planned action. This change is what takes the dialogue to a new state.

The definition of the possible effects of an action supports dialogue state recognition since the agent has a direction on which variables might change upon execution. For example, to recognize the new state after executing the planning action named '*search_most_useful_slot*', the agent will iterate through the action possible outcomes and query the corresponding instances in Convology. Consequently, the agent identifies which *Slot* was set as the most useful. After having this information, the planned action for this state can be executed.

For '*request*' actions, on the other hand, the agent tries to match the *Intent* received from the NLU module with the *Intent* instances associated with the last *DialogAction* executed by the system. Only when this match is possible and it achieves the minimum confidence value set, the agent is able to handle the information received (e.g. store *Slot* value) and proceed with the next planned action. However, it is important to note that a non-expected intent also might be received. That is, the user's input cannot be matched to any information expected at the current state. The causes and procedure to handle this situation are discussed in section 7.3.2.

Finally, the possibility of checking the dialogue history available in Convology also supports the comprehension of the user input and consequent

state recognition, as discussed in section 6.5.

Runtime Classification

As discussed in section 7.2.3, the classification strategy is abstracted by the usefulness framework. The planned action works like a flag that notifies the agent that it should check for a specific classification that might have been achieved after receiving the last information. This operation happens in the background and, if no classification is reached, the agent will proceed to the next planned action.

As defined in chapter 6, the usefulness framework also takes into account a *negative classification*, i.e. the agent can identify when a given status will no longer be achieved even if all missing information is confirmed. To address this situation, the Convology property *hasNegativeGoalAchieved* is used when a *UserStatus* has a negative classification. A negative classification is followed by a replanning operation that excludes from the plan graph all the missing information associated with this *UserStatus* as it is no longer relevant to the agent.

On the other hand, a *positive classification* (i.e. a *UserStatus* is activated) indicates that the plan has reached its goal. For a *cautious* solution (section 6.4.3), the agent can also conduct a replanning operation to exclude the information related to the achieved *UserStatus* and continue the dialogue to retrieve further information. In a *fast-solve* solution, instead, a *Feedback* is provided to the end-user (Fig. 7.3 line 13), concluding this sub-dialogue. In this case, the agent generates a new plan to manage the next sub-dialogue or, if no further sub-dialogue is defined, the dialogue session is concluded.

7.3.2 Mixed Initiative

To keep the dialogue system flexible, *OntoPlanDM* supports *mixed-initiative*[9], a research problem frequently discussed by the dialogue community [119]. The planned dialogue keeps predominantly a system-initiated flow, where the dialogue actions executed by the agent anticipate which information should be included in the intents that are expected in the next user input. However, the user may input some non-expected information at any moment. In the dialogue sample in Figure 7.3, for example, a user-initiated input can be observed in line 9.

Addressing user-initiated actions adds extra complexity to the dialogue manager since it is difficult to predict what will be the user input. In fact, considering that the same sort of alternatives may appear in several places, it is not reasonable trying to anticipate all possible intents that may occur in every dialogue turn. Although this approach deals with non-deterministic actions, adopting such a strategy would lead to a scalability problem of the planning solution. This way, this work has adopted alternative strategies to handle mixed-initiative.

To interpret the user's input, the agent first exploits the restricted set of intents that are related to its last executed action, as discussed in section 7.3.1. However, as a consequence of a user-initiated action, it is possible that none of the expected intents can be recognized and, in this case, the agent must exploit further alternatives to update the dialogue state. The first strategy is trying to match the NLU output with the n previously expected intents. That is, the dialogue history available in Convology (i.e. *DialogStatus* instances) is checked to understand if the user's input is an answer to a question made in a previous turn or the modification of a slot-value recently discussed. When no intent can be matched by applying

[9]It is important to note that *OntoPlanDM* handles mixed-initiative within information-seeking sub-dialogues. On deliberation sub-dialogues, on the other hand, user-initiated input is ignored.

this strategy, the remaining *Intent* instances within Convology are finally considered. Notice that this is a more costly alternative, but it avoids the agent to ignore some relevant information. Finally, if previous strategies still were not able to recognize an intent, a *fallback_intent* should be activated in Convology to give origin to a fallback question.

It is important to note that receiving an information that was not expected at the current state changes the flow of the dialogue. Therefore, before checking the next action to be executed, the dialogue policy must be updated through *replanning* (section 7.3.3).

7.3.3 Replanning

Before replanning, the *initial state* must be updated in the *planning problem* to match the current dialogue state. Only then, the automated planner is called to generate an updated policy that can be consulted by the executor to give continuity to the ongoing dialogue session.

Depending on what triggered the replanning operation, corrective actions must be included in the updated plan. Corrective actions are actions to fix what went wrong in the previous plan before continuing the dialogue flow. This work includes two corrective actions and they make it possible to avoid the costs of predicting everything that can go wrong in the initial plan.

The first is the *feedback_unexpected_intent* action. For non-expected intents, the initial state is updated to force a feedback action that is provided to the user to acknowledge that the system has processed the received information before continuing the information retrieval (lines 3 and 10 in Fig. 7.3). This way, the system outputs two dialogue actions in a sequence: (i) a *Feedback* action to acknowledge the information received and (ii) a *Feedback* if the goal is achieved or, on the contrary, a *ClosedQuestion* with the next question.

Although the action *enable_fallback_question* is included in the initial plan, it can also be treated as a corrective action. This action enables the agent to provide a *Feedback* on what went wrong and, then, replan the policy having an *OpenQuestion* as the next action. The aim of this question is to invite the user to try to repeat the last information in a different way. For domains that include a huge number of slots, it is recommended to remove the non-deterministic effect, i.e. the predicate *fallback_intent*, from *request* actions. Instead, when the user's input is not understood, this predicate can be added to the updated initial state before replanning.

In section 8.2, that describes two early evaluations of the approach, the suitability of *OntoPlanDM* for replanning on real-time scenarios is discussed.

Chapter 8

Experimental results

8.1 Introduction

The evaluation of dialogue systems is known as a challenging process. There is no consensus on an evaluation strategy that is fair to assess dialogue systems built for the most varied purposes. In fact, there is no consensus in what comprises a good dialogue system as it depends on several factors such as the dialogue purpose, target users, and flexibility.

On one hand, human evaluation is still the main strategy to assess the quality of a dialogue system. However, such evaluation is time-consuming and implies high costs. On the other hand, no benchmarks or official protocols are available to guide an evaluation that does not require human subjects. Another factor that brings additional complexity to the evaluation of these systems is the need for a separate evaluation of the system's components. For example, in a human evaluation, it is quite challenging for the end-users to focus only on the dialogue manager features (e.g. correct question) and leave aside the natural language aspects of the system. This way, different studies might be required to assess a dialogue system.

This Chapter first presents an early evaluation that was conducted in early stages of this research (Section 8.2). This evaluation consisted of two independent studies that assessed an early version of *Puffbot*, the asthma

support dialogue system. While the first study involved domain experts that provided a qualitative assessment of the agent, the second consisted of a synthetic evaluation that assessed the scalability and suitability of *OntoPlanDM* for real-time plan generation. These studies contributed to the improvement and definition of the final approach.

Section 8.3 presents a user evaluation that was conducted with the final version of *Puffbot*[1]. This evaluation is composed of three studies[2] involving human collaboration that were conducted to assess different aspects of *OntoPlanDM*[3]. The studies involved three classes of participants:

- Health experts: health professionals with some expertise in respiratory diseases evaluated *Puffbot*;
- End-users: participants with some knowledge in respiratory diseases (e.g. asthma patients) as these are the end-users of *Puffbot*;
- Dialogue authors: participants with some technical background authored a prototype for a domain different than the one in *Puffbot*.

The studies involving *Puffbot* assessed qualitative and quantitative aspects related to its efficiency, effectiveness and usability. The study that involved dialogue authors, instead, evaluated the reusability of *OntoPlanDM* as well as its costs and feasibility for non-experts authoring. Finally, Section 8.4 presents an overall discussion on our findings.

[1]The implementation of *Puffbot* is described in Appendix B.

[2]The studies were conducted in the context of the EU project WideHealth (https://widehealth.eu/).

[3]As the studies were conducted during the COVID19 pandemic crisis, we counted with both in-person (75%) and remote (25%) collaboration. The remote testing was mediated by video calling systems and it followed the same procedure of the in-presence sessions.

8.2 Early Evaluation

This Section describes two early studies that were conducted with the purpose of improving the final version of the approach. First, Section 8.2.1 presents a qualitative assessment and discusses some lessons learned during the development of the first version of *Puffbot*, the dialogue system that supports asthma patients. Next, Section 8.2.2 presents an evaluation of the scalability of also an early version of the approach, as well as its applicability in real-time scenarios, given the replanning operations.

8.2.1 Qualitative Assessment and Lessons Learned

One of the main aims of this work is to provide a mechanism for effectively and efficiently modeling the underlying intelligent layer of the dialogue management system. Therefore, a prototype system was developed and an evaluation of the dialogue manager component has been performed by means of qualitative analysis from the domain expert's perspective. This analysis was reported in [114].

The developed prototype, which is considered the first version of *Puffbot*, partially implemented *OntoPlanDM* and it had limited features:

- the agent conducted only the first information-seeking sub-dialogue, i.e. it asked about asthma symptoms and classified the current situation. No deliberation sub-dialogue was included;
- apart from the initial (open) question, unexpected inputs from the user (user-initiative) in other stages of the interaction were ignored;
- all slots were listed as possible outcomes of the initial action (different from the initial action specified in Section 7.2.1), therefore no replanning was required after the initial question;

- the usefulness framework was not integrated into the agent. Instead, the planning domain included hand-crafted rules to support action selection. These rules would explicitly specify which slots were needed to classify each asthma risk group, for example. Less risky groups (e.g. yellow zone) required that information regarding slots from riskier groups (e.g. red zone) was already provided.

The qualitative evaluation of the system was conducted by a group of three domain experts invited to provide feedback about both the internal structure of the system and the bot. From this feedback, a few lessons were learned, which were extracted to drive the research activities that were conducted afterwards. The main lessons are described below, followed by a short discussion of some of the measures taken afterwards to address the experts' feedback.

Dynamic design of dialogue strategies. Policies are dynamically generated from the information in the ontology ABox and it is possible to replan to update the dialogue tree. This behavior makes it possible to continue the dialogue even when the user inputs something non-expected, while it avoids the generation of unreasonable dialogue trees that would result from anticipating every possible user input. While this aspect was positively evaluated by the experts, they highlighted the need for an evaluation of the costs of this strategy on runtime.

Relevant effort reduction. The initial policy generated by the planner for *PuffBot* resulted in 106 nodes. Handcrafting a dialogue policy of this size can be challenging and error prone for humans; this shows a real benefit of automation. Meanwhile, dialogue authoring with our approach requires expert knowledge on the dialogue domain (e.g. asthma) and some knowledge on conversational aspects (e.g. intent, slot), but no knowledge on AI planning is required, which would be a factor to limit its adoption, as highlighted by the experts.

Real-time suitability. The proposed system is aimed to be deployed in a real-time context. Feedback and recommendations have to be provided timely to users based on the evolution of their health status. Hence, together with the domain experts, we observed the performance of the whole process of selecting the next action by the dialogue manager. The focus on the optimization process brought us to the deployment of a solution able to support an efficient real-time generation of recommendations and guidelines for the final user. Domain experts appreciated how the integration of the ontology for supporting the generation of dialogue policies kept the system efficient on recognizing the state of the dialogue and choosing appropriate dialogue actions, and both effective and efficient in producing a proper output to be shown to the final user. The very low response time (lower than 2 seconds) in providing feedback to the final user has been judged as more than acceptable by the domain experts for considering the adoption of the system into daily practice.

Perception about personalization. We consider the perception that the users had about the possibility of personalization of the proposed platform by asking them how the system could be improved with respect to personalized interactions. A common request was related to the possibility of exploiting external parameters that can affect the user‘s condition such as the air quality. This information was relevant to consider the overall improvement and customization of the conversational agent by taking into account also environmental factors, which can be modeled as *Slots* and whose values can be obtained through external querying and inference.

Finally, the last aspect that resulted in some relevant discussion among the experts is the use of open text by the user. This discussion was expected by us since mixed initiative is a trending topic. Although we have covered this aspect in Section 7.3.2, the questions brought by the experts raised the question of whether it is feasible to allow the user to freely input text at

any moment in the dialogue, since this behavior would result in constant replanning and increase the system's cost. The measures taken to address this and the aspects previously discussed are reported in the next Section.

Approach Improvements

Some measures were taken to address the experts' feedback reported in Section 8.2.1. First, to address the experts' comments within the discussion of the *"dynamic design of dialogue strategies"* and to sustain the conclusion made in the discussion of the *"real-time suitability"*, an evaluation of the scalability and suitability of the approach to realistic sized problems was conducted afterwards. This evaluation is reported in Section 8.2.2.

Regarding the policy size mentioned within the discussion about *"relevant effort reduction"*, it is important to note that the last version of *Puffbot*, whose implementation is presented in Appendix B, is more dynamic: it builds a simple initial policy and the size of the replanned policy will depend on the information provided after the initial question. However, the aspects discussed (e.g. no knowledge on AI planning is required) were preserved in this new version.

The suggestions provided within the discussion of *"perception about personalization"* oriented the translation of the information modeled within Convology into the planning problem. That is, aspects such as the "*air quality*", previously mentioned, can be modeled as *Slot* instances. Consequently, they will be translated into planning constants, representing information to be obtained during the dialogue session. However, as adding a personalization component to retrieve information from external sources is not a trivial task, it is left as an opportunity for future work.

Finally, to avoid additional costs with replanning, the interpretation of user-initiated actions was limited to situations that expect to receive new information but do not require a strict answer. More precisely, information-

seeking sub-dialogues were kept open, allowing the user to provide new information at any moment. Meanwhile, as deliberation sub-dialogues do not require the user to provide information (other than a close-ended confirmation), they ignore user-initiated input.

8.2.2 Evaluation of Scalability and Real-time Policy Generation

The feasibility of modeling the dialogue as a planning problem was tested for different problem sizes and it was reported in [116]. This evaluation was conducted for the early version of *Puffbot*, whose features and limitations were presented in Section 8.2.1. However, an updated evaluation is provided following this Section.

To conduct this evaluation, Convology was first populated with data from the first information-seeking sub-dialogue of the original scenario (Section 2.1), which contains a set of 10 slots and 4 possible classification status. Then, to evaluate the scalability of the planning problem to different specification sizes, this scenario was expanded with synthetic data to consider different numbers of slots. The number of classification alternatives, instead, was kept always the same. The FOND automated planner PRP was used to generate the policies. For the different problem sizes, we analyzed the time required to generate a new policy, which is a sum of the time taken to translate Convology to PDDL and the search time reported by the planner. To obtain an average generation time, the experiments were repeated 100 times for each specification size. The scalability of the solution size (in terms of nodes and edges) with respect to the specification size (in terms of variables and actions) has also been analyzed.

The results (Table 8.1) reveal that the time necessary to generate a policy was significantly short even for large problem sizes, staying far below 1 second. This shows the efficiency of this solution to generate a dialogue policy in real-time for real-world instances. The feasibility of dynamically

Table 8.1: Relations between the number of slots, specification and solution sizes and the average time required to generate dialogue policies

Number of slots	Specification size	Solution size	Time to generate (secs)
10	29	56	0.07
20	49	106	0.10
40	89	206	0.11
80	169	406	0.16
160	329	806	0.32

updating the policy through replanning in the case of non-expected inputs is also guaranteed. In addition, as can be observed in Table 8.1, the plan scaled well to different problem sizes, with solution up to 2.4 times greater than the specification size.

While the first two or three examples are not impossible for human design (although a human would take much longer than seconds), the additional policies are far beyond a specification that could be easily built by hand. Besides requiring high human effort, handcrafting such policies becomes an error-prone process. This evaluation did not include handcrafted baselines since it would not be feasible to compare the seconds required to automatically generate a policy to the time required by a human agent to build a policy, which would require representation in hours. Indeed, such comparison has not been provided by other plan-based approaches [81, 94].

Alternatively, learning such policies requires lots of conversational data on the specific domain. Given some well-known constraints of the health domain, like privacy constraints that limit retrieving information from real-world dialogues [98], having the assistance of a domain expert to populate the ontology can be considered a more realistic strategy for health dialogue generation. In addition, as all planning strategies are abstracted within *OntoPlanDM*, neither the dialogue author nor a domain expert is required

to understand AI planning. This shows that this approach is capable of simplifying to a great scale the automated generation of complex health dialogues.

Updated Scalability Evaluation

The evaluation reported in Section 8.2.2 refers to the early version of *Puffbot*, which did not fully cover *OntoPlanDM*. Therefore, instead of abstracting the usefulness framework for action selection, the planning domain included hand-crafted rules. Classification actions also expressed hand-crafted preconditions. The advantage of such a planning problem is that, as reported in the scalability evaluation, it proves to be very efficient to cover large domains, highlighting the contribution of automated strategies to generate dialogue trees. However, the planning domain represents a domain-specific problem that is not reusable to different scenarios within healthcare. Although some efficiency loss is expected, a reusable[4] and more generic solution can be obtained by employing the full approach. To evaluate how the plan suffers from the employment of this solution, we reproduced the previous evaluation with an updated version of the planning problem.

As can be seen in Table 8.2, the updated solution was significantly affected. The solution size grew exponentially in the specification size. This aspect is common when handling uncertainty within AI planning [50]. To our problem, this solution reinforces the contribution of using an automated technique to generate such complex policies, which are far beyond what could be manually done. Moreover, we can observe its benefit when it is necessary to update any domain variable, since small changes affect large portions of the final solution. The updated results show the feasibility of computing plans with specification size up to 39, which, in our

[4]An evaluation of the reusability of the approach is provided in Section 8.3.3.

Table 8.2: Relations between the number of slots, specification and solution sizes and the average time required to generate dialogue policies by using the full approach.

Number of slots	Specification size	Solution size	Time to generate (secs)
10	35	51214	0.38
12	39	238340	1.92
13	42	509695	4.26

example, corresponded to 12 slots and 4 classification groups that resulted in 23 actions. The time to generate this solution was below 2 seconds, an interval considered acceptable by the experts from the qualitative assessment reported in Section 8.2.1. We recognize that significant efficiency was lost with the aim of delivering a reusable approach; optimization is pointed as a subject of future work. However, as *OntoPlanDM* does not present a limit on the number of sub-dialogues implemented, an alternative to handle this limitation is through the implementation of sub-dialogues. That is, huge scenarios that present a huge number of slots, for example, can be divided into smaller problems addressed by different instances of information-seeking sub-dialogues. Moreover, the specification of an open initial question (e.g. "*How are you feeling?*") opens up the possibility that the end-user will provide some relevant information at this step. This strategy reduces the search space for the replanning operation that follows.

8.3 User Evaluation

8.3.1 Health Experts Evaluation

A qualitative study involving health professionals was conducted with the goal of assessing the perceived usefulness and barriers of *Puffbot* from the health experts' perspective.

Participants

The study counted with the participation of four physicians, all females, three with expertise in allergy and immunology, and one with expertise in pediatric pulmonology. None of these experts was involved in the process described in Section 2.1 to acquire domain-specific knowledge. While three physicians participated all together in an in-person meeting, a remote session was conducted with the fourth one.

Method

The study consisted of sessions where the health experts were invited to interact with *Puffbot*. A session started with an introduction to the research objectives, followed by a description of the purpose of *Puffbot*. Then, the asthma action plan (i.e. the guideline) used as the knowledge source was shown to the experts. We asked the physicians to think of one of their patients and of an asthma crisis that this patient could have. Then, the experts were provided with smartphones with the *Telegram* app pre-installed and were instructed to simulate this patient, trying to interact with the bot based on what they thought that the patient would say. That is, the description of the symptoms should be limited to the patient's vocabulary and to the knowledge that the patient has of the disease. We instructed the participants to use the method '*think out loud*' [70], that is, they were invited to say out loud any thought that they had while interacting with the chatbot. The purpose of using this method was to elicit discussions that could help us to understand the positive and negative points of *Puffbot*. With the consent of the participants, the sessions were audio-recorded to collect qualitative data from comments made by the participants while using *Puffbot*. After conducting a few interactions with *Puffbot*, a questionnaire was provided to each participant.

The questionnaire first included a demographic session containing an open question about the participants' expertise and two questions regarding their level of familiarity with (i) respiratory diseases and (ii) chatbots for the healthcare domain. Both questions could be answered on a five-point *Likert* scale from 1 ('*No knowledge*') to 5 ('*Expert*'). As the experts played the role of end-users, we were also interested on their feedback about the system usability. Therefore, the next section of the questionnaire included the System Usability Scale (SUS) [25] questionnaire. The SUS questionnaire is a simple but widely exploited tool to measure subjective aspects that make it possible to assess the usability of a system, in our case, the conversational agent. SUS consists of a list of ten items and each item is presented as a statement for which the participants are asked to assess their agreement on a five-point *Likert* scale from 1 ('*Strongly Disagree*') to 5 ('*Strongly Agree*'). The last section of our questionnaire asked the participants to rate their trust in recommending *Puffbot* as an additional resource for an asthma patient. This question also used a *Likert* scale, where 1 means '*No trust*' and 5 '*Strongly trust*'. Finally, an open comment could be left by the participants.

Results

According to the instructions given, the participants conducted a few dialogue sessions, simulating different asthma crises by informing different symptoms. Recommendations were provided by *Puffbot* at the end of each dialogue session. The participants made comments during the interactions and filled out the provided questionnaires.

To better understand the experts' feedback about *Puffbot*, we analyzed the results of the questionnaires and also reviewed the recordings of the sessions, taking notes of the main comments. In the demographic section of the questionnaire, all participants declared themselves as experts with

asthma or other respiratory diseases. Regarding their knowledge with chatbots for healthcare, 2 declared to have no knowledge, and 2 informed 3, meaning average knowledge.

From the analysis of the questionnaire results together with the notes extracted from the sessions with the experts, we extracted five major themes, which are described next.

***Questions appropriateness*:** In both in-person and remote sessions, there was plenty of discussion regarding the number of questions made by the bot. Although the experts agreed that the questions made by the bot were appropriate, all of them also agreed that *Puffbot* should make additional questions regarding the symptoms. As highlighted by Expert1, "*it is very difficult to objectively reach a conclusion*". The experts emphasize that, when a patient calls to report an asthma attack, extra questions must be made to fully understand the situation. Examples of further questions include: how the symptoms have started (e.g. was the patient doing any activity?), how long the patient has the symptoms, and what has been already done since the symptoms started.

Expert1: "*There must be more questions!*"

Expert2: "*Maybe more questions were needed, no? (...) here it makes more questions, it's better*"

Expert3: "*How long the patient had the symptoms (...)*" (referring to additional questions)

Expert4: "*yes, the questions yes*" (when asked if questions seemed appropriate) "*I don't think it makes too many questions (...) I think it's worth asking if... already used the bronchodilator and even corticosteroids, which is something we use too...*"

***System adoption*:** Opinions on using *Puffbot* frequently were divided, the average for this statement in SUS is 3. From the discussions with the participants, we understand that, as asthma crises tend to be similar and recurrent, the patient usually knows how to proceed. For this reason, the experts do not see the need for frequent use of *Puffbot*. Nonetheless, the experts recognize that exceptions might occur when an unprecedented crisis occurs and, then, the patient might not be sure which actions to take. Also, as highlighted by *Expert4*, it is common that some patients do not have any crisis for long periods (even years). In this case, when a new crisis occurs, the patient might not remember how to proceed and, for such situations, the expert recognized the great contribution of *Puffbot*.

Expert1: "*the patients already know what to do in these situations*"

Expert4: "*asthma even though it is a chronic disease, is a disease that has outbreaks. Years might pass without any crisis and later, suddenly, there is a crisis... we see it happening and sometimes the patient didn't even remember how many puffs to do. It doesn't seem redundant that it says what to do (...) the person already lost the copy* (referring to the guideline)"

***Adaptability*:** From the discussions with the experts, we could notice that adaptability would increase the acceptance of such systems. The experts agreed that the action plan (guideline) used as the knowledge source is a standardized and validated action plan that is given to the patient to consult in case of crisis. However, they emphasize that sometimes it is necessary to adapt this plan to the patient's case. That is, some patients receive an action plan with recommendations slightly modified with respect to the ones presented in the guideline and *Puffbot* should be able to cover these scenarios.

Expert3: "*we adapt it later*" (referring to the guideline) "*...each patient*

is a patient and we evaluate also according to each one the therapy to institute (...) but it makes sense, it is based on something that is already standardized and validated"

Expert4: "*the recommendation is a little hard because the dosage of bronchodilators varies a lot. So it is hard to say if it is correct or not*"

***Goal Achievement*:** The opinions were divided regarding possible inconsistencies in *Puffbot* (SUS mean: 3). We understand that this might have been a consequence of the classification made by *Puffbot* during some dialogue sessions. For some sessions, as a consequence of the *threshold* defined for *Puffbot* together with the adoption of the *fast-solve* method, *Puffbot* was not able to reach a classification. In these cases, although some symptom was reported, the amount of data received was not enough for *Puffbot* to reach a conclusion and, therefore, the default class (no risk) was triggered. The experts disagreed with *Puffbot*'s classification and pointed up that more questions should be made before reaching such a conclusion.

Expert1: "*cough might be a sign of an asthma crisis and here it says it's not*"

Expert3: "*we think it was too much (...) it says it is an intense crisis and recommends (...) it is important to understand how long he has the symptom*s"

***Trustworthiness*:** When asked if they would trust *Puffbot* as an additional resource for an asthma patient, the expert's opinion was also divided. For this question, 1 meant '*no trust*' and 5 meant '*strongly trust*'. While 2 experts voted 4, the two others informed 1 and 2. From their comments, it is possible to notice some resistance in the use of *Puffbot*, although they recognize it has its usefulness to deliver the content of the guideline.

Expert1: "*it is a good thing (...) the machine will never cover every-*

thing"

Expert2: "*if the patient cannot reach the physician, it is always better to follow the recommendation*"

SUS Analysis The SUS score for *Puffbot* was 68.75. A score of 68% has been used in the literature as the usability threshold for interactive systems. This way, *Puffbot* still can be considered above average, although there is lots of room for improvements. By taking a closer look into the SUS results, we can highlight the following positive aspects: (i) the experts did not find *Puffbot* unnecessarily complex (mean 1.75), (ii) they felt confident using *Puffbot* (mean 3.75), (iii) they think people would learn to use *Puffbot* very quickly (mean 4.5), (iv) it was easy to use (mean 4), and (v) no support from a technical person is necessary for someone to use *Puffbot* (mean 2.25). The aspects that presented divided opinions were discussed in the previous Section and they bring our attention to possible improvements of the agent. While some of these aspects are related to settings within the agent (e.g. classification), others open the opportunity to refine the approach (e.g. to include adaptation).

8.3.2 End-users Evaluation

A living lab was conducted to provide both a quantitative and a qualitative evaluation of *Puffbot*. By employing different settings, this study assessed the performance of the dialogue manager and the usability of the agent.

Participants

The participants for this study consist of people with some knowledge of respiratory diseases. A total of 27 participants were recruited, of which 3 were excluded due to a technical failure of a setting. Therefore, each valid system setting was tested by 8 participants. Participants were 35 years

old on average, 42% female, and all had some degree of higher education. Most participants (79%) declared to have asthma, 13% reported that some family member has it, and 8% declared just to have some knowledge of the disease. Finally, on average the participants declared to have good knowledge of health dialogue systems (3.6 out of 5).

Method

To conduct a controlled study, a health expert was asked to describe 3 scenarios of asthma crises, one for each possible risk group described in the asthma action plan (Section 2.1).

The testing sessions were individual and they were conducted either remotely or in a dedicated room. At the beginning of each session, the session admin quickly introduced the research goals and described *Puffbot*'s purpose. To interact with *Puffbot*, participants were asked to use their personal Telegram account on their smartphones or, whenever this was not possible, they were provided with a smartphone with the Telegram app pre-installed. Afterward, the session admin randomly selected one of the asthma scenarios and described it to the participant. The participant was then asked to interact with *Puffbot* by trying to simulate the scenario presented. The participants were instructed to try to provide the information from the scenario as they find it more convenient, i.e. by either answering *Puffbot*'s questions or by trying to use open text; but they were asked to try to stick to the symptoms and situation presented. The simulation was followed by a short questionnaire with questions about the interaction. This process was repeated for the 2 other scenarios, that were also randomly distributed. After concluding the three simulations, an additional questionnaire had to be completed. This questionnaire first asked some demographic information, including questions about the users' familiarity with the asthma disease and chatbots for healthcare. The next

section of the questionnaire included the SUS questionnaire, also used in the study with the health experts (Section 8.3.1). Like in the study involving experts, the last section of this questionnaire asked the participants to rate their trust in recommending *Puffbot* as an additional resource for an asthma patient. This question also used a *Likert* scale, where 1 means '*No trust*' and 5 '*Strongly trust*'. Finally, an open comment could be left by the participants.

Each participant is randomly assigned to one of the three[5] following settings:

- *S1 - full approach*: this setting implements all the features of *OntoPlanDM*;
- *S2 - no replanning*: this setting implements the approach used in S1 but without replanning operations;
- *S3 - system-initiated agent*: this setting implements the approach used in S1 restricted to system-initiative.

S2 included two main differences from S1. First, the initial action was modified to anticipate all possible intents that can be provided by the end-user. As a consequence, the generation of the policy when the session starts takes longer than in S1. However, the agent will not devote additional time during the session to replan the policy. Second, *corrective actions* (Section 7.3.3) that handle non-expected input are not addressed in S2. Therefore, although the agent processes non-expected input, no feedback is provided to the end-user. S3 also presented a modification in the initial action with respect to S1. While S1 starts with an open question, S3 starts with a closed question, i.e. the most useful one. During the session, S3 will ignore any user input that is not an answer to its last question. A comparison of our approach is made by taking into account these three different settings.

[5]A fourth setting was excluded due to technical issues during the first three sessions that employed it.

As highlighted by the dialogue community, it is difficult to make direct comparisons between different systems as the interactions vary according to systems' purpose [31]. However, to better understand the values achieved, we confront our results with those from the work of Beveridge and Fox [15]. Their approach relies on a domain ontology and on a framework of conversational games to manage health-related dialogues. Their results come from a study that included six participants who evaluated a decision support system that conducts dialogues related to breast cancer, the CR–UK cancer dialogue system.

Results

A total of 72 dialogue sessions were conducted, of which only 2 did not achieve the dialogue goal (the user quit the dialogue before it ended).

Our dialogue manager was evaluated based on three aspects: (i) task success, (ii) dialogue costs, and (iii) system usability. The analysis of the results partially followed the criteria defined by PARADISE [120], which is an established framework designed to evaluate the performance of spoken dialogue systems. However, it is important to note that PARADISE cannot be properly applied to our problem for two main reasons: (i) as it was designed to evaluate spoken dialogue systems, some metrics are not present in our data; (ii) PARADISE aims at evaluating the whole agent, while our purpose is to assess the dialogue manager component only. This limits the comparison of our approach to other works evaluated with PARADISE. This way, as previously mentioned, the performance of our dialogue manager is compared to the different settings of *Puffbot* and the CR-UK dialogue agent.

Task success Task success was analyzed according to the following metrics: (i) number of users that completed the three dialogue sessions, (ii) the

Table 8.3: Task Success

	S1	S2	S3	CR–UK
Users that completed all sessions (%)	100	87.5	87.5	80.8
Data accuracy (%)	97.5	93	97	97.6
Transaction success	95.6	95.6	100	85.7

percentage of data correctly acquired from the user input (data accuracy), and (iii) the correctness of the classification made by the agent (transaction success). The results are displayed in Table 8.3 and discussed below.

For all settings, the majority of users completed all sessions. Only two exceptions occurred, one for S2 and one for S3. In the former, the participant decided to leave the conversation during the first information-seeking sub-dialogue, after the agent did not recognize the information provided two consecutive times. In the second case, instead, the participant quit the conversation during the second information-seeking sub-dialogue. As this sub-dialogue is initiated after a deliberation sub-dialogue, i.e. a recommendation had just been received, we believe that the participant might have misinterpreted the agent, believing that the session was over. The high percentage achieved in all settings shows that, even if some errors occur during the session, they mostly do not prevent dialogue completion. The lower completeness achieved by CR-UK was (in parts) a consequence of failures in the speech recognizer. This brings attention to the need for efficient speech recognizers when dealing with the health domain.

Data accuracy was analyzed by taking into account only the sessions that achieved the final goal. To obtain the correctness of the acquired information, dialogue logs were manually analyzed and the natural user input was confronted with the values registered in Convology. All settings presented a high degree of accuracy. While both S1 and S3 had only 3 incorrect values, S2 presented 7 errors. In S2, although non-expected information is not part of the plan, the dialogue manager is still able to

handle it. However, as no replanning is conducted, no corrective actions are added to provide the user with feedback. The lack of feedback might prevent the user from correcting information misunderstood by the agent. We attribute the reduced error rate in S3 to the fact that non-expected information is ignored. This way, any error that occurred was carried along with the dialogue manager as a consequence of misinterpretation of the user's answer by the NLU module. S1 obtained almost the same accuracy as CR–UK. However, it is important to note that the high accuracy obtained by CR–UK was a consequence of a verification strategy implemented to check the correctness of the data before making a classification. Although this strategy benefits the accuracy, the verification prompts can become over-long in some cases, as identified by the authors.

Finally, transaction success was analyzed for all sessions that reached the final goal. Here, it is important to note that thanks to the deployment of the usefulness framework, classification is knowledge based and it matches the data received. This aspect was already demonstrated in Section 6.6. In fact, all settings present high transaction success. However, since errors might occur during data acquisition, the classification might be consequently affected. Therefore, to assess the correctness of the classification provided by the agent, we manually analyzed the sessions that had errors in the user input recognition. Out of 9 sessions that presented some error (all settings considered), only 2 (22.2%) were miss-classified (one in S1 and one in S2). This represents only 2.86% of all sessions conducted.

Dialogue Costs The analysis of the dialogue costs took into account the following metrics: system response time, overall session duration, the number of turns of a session, and the proportion of turns to correct errors (i.e. fallback questions). Table 8.4 presents the medium values for each of these aspects and their description is given next.

Table 8.4: Dialogue Cost

	S1	S2	S3	CR–UK
System response time (s)	1.43	1.22	1.34	0.53
Session duration (s)	102	70	46	277
Number of turns	15	14	9	50
Correction rate	0	0	0	2

Response time indicates the time that was spent from receiving a user input (*UserEvent* instantiation) to sending the user a response (*Event* or *EventQuestion* instantiation). For all settings, higher values occur at the start of a dialogue session, when no information was retrieved yet. As expected, S2 presented a lower response time than the other settings as no replanning operation was conducted. However, this value was not significantly lower for two main reasons: (i) the initial action anticipates all possible user responses, taking longer to generate the initial (and only) plan, and (ii) as there was no replanning to exclude non-achievable goals, the search for the most useful slot kept checking all empty slots. S3 was also expected to present a lower response time than S1 as it does not interpret non-expected input. In general, response time was significantly higher than in CR-UK, raising a flag for the need for optimization. However, as demonstrated in our early evaluation (Section 8.2), such a response time is still considered acceptable by experts (less than 2 seconds). This confirms that the efficiency loss obtained to deliver a more generic approach (Section 8.2.2) did not affect the final solution.

For all settings, the session duration correlates with the number of turns, which includes both system and user turns. Once again, the system-initiated setting, i.e. S3, was quicker to achieve the dialogue goal (at the cost of ignoring non-expected information). Since CR-UK addresses a different domain, its session duration and the number of turns are just demonstrative and are not comparable to our approach.

Finally, the medium number of turns to correct errors (correction rate) is 0 for all settings. However, their ranges were 0-3 for S1, and 0-2 for both S2 and S3. The average proportion of turns spent to correct errors, instead, was 4.78% (S1), 5.8% (S2), and 3.43% (S3). These values indicate that, for all settings, the system had a low cost regarding the overall time spent to address non-recognized information. It is important to highlight that, instead of just quitting the interaction when errors occurred, the agent was robust enough to recover from these errors and continue the dialogue session. The difference is in how the settings handled these errors. While S1 and S3 update the policy through replanning, S2 already anticipated possible errors. However, as demonstrated in Section 8.2.2, huge domains suffer from such anticipation and, for those, replanning is a more appropriate strategy.

System Usability The usability of *Puffbot* was evaluated based on the SUS score (Table 8.5[6]) and on the feedback provided by the participants. The SUS scores revealed a non-expected outcome: the main setting (S1) presented an overall score relatively lower than the other settings, being slightly below the SUS threshold (i.e. 68%). There was not much consensus among the participants, as the individual scores for S1 varied from 35 to 92.5. To understand the reasons for the overall score, we analyzed the answers given to the SUS questionnaire and identified the two items that received the worst rating: "*I needed to learn a lot of things before I could get going with Puffbot*" and "*I thought there was too much inconsistency in Puffbot*". Both items received an overall rating of 2.75 on a 5 point scale (1 = "*Strongly disagree*" and 5 = "*Strongly agree*"). The comments left by the participants suggest that this rating might be related to limitations

[6]No comparison to CR-UK is made as the authors used different strategies to evaluate the usability of their system.

Table 8.5: System Usability Score

S1	S2	S3
67.5	79.1	73.1

on the way that information can be provided by the user. For example, some participants tried to exploit the bot capabilities by providing long texts or different words. Thus, some users complained that the bot did not recognize long sentences or a too varied vocabulary:

Participant #6: *"I tried to explain better the situation, writing long sentences, with more symptoms and the bot didn't understand."*

Participant #10: *"You need to use some keywords...it might be harder for some people (children or elderly)."*

While the first aspect needs to be exploited in future work to address multiple intents in a single turn, the second can be improved with the deployment of a more robust NLU module. Despite the negative aspects, the SUS item "*I thought Puffbot was easy to use*" received a high overall rating (4.25) and, when asked if they would trust *Puffbot* as an additional resource for an asthma patient, all participants selected 3 or above, being 4 the overall score for this aspect in S1. The overall score for this aspect considering all settings was 3.96.

As previously discussed, given the absence of replanning operations, S2 lacks feedback after user-initiated input and it takes longer to generate the initial policy, but it is quicker than S1 to answer during the ongoing session. However, the absence of feedback did not seem to be a problem to the participants as no comment was left on this matter and, like in S1, the dialogue length was rated negatively in only two sessions.

Finally, S3 did not present significant differences from S1 in SUS items. We believe that one of the reasons for a higher SUS score in S3 is related to the agent's goal. While in 83.3% of the sessions in S3 the user reported

that "*The objective of the bot was clear*", in S1, this value was 75%. In fact, system-initiated agents have the advantage of being more objective. Nonetheless, some of the well-known disadvantages of these systems could also be observed in S3. For example, two participants complained that the agent did not recognize the symptom *chest tightness* when they tried to inform it, although it was a part of the given scenario. Indeed, the participants tried to inform the symptom in a user-initiated turn and, as it was not expected at the current state of the dialogue, the agent ignored it. Regarding this situation, participant #23 commented "*I think someone not so calm when using the bot, might not continue with the other questions*". Another important limitation within this setting was the impossibility to update information already provided. A participant highlighted that she made a mistake when inputting some information and that it could not be fixed later in the dialogue. This aspect brings attention to the need for a confirmation step before providing a classification.

In summary, it is not possible to draw a conclusion on what influenced the difference in the SUS scores of the 3 settings. In fact, our baselines were very strong and they also implemented our approach, demonstrating its effectiveness. A more extensive study addressing also qualitative aspects should be conducted to better understand the reasons for this difference. This study should address a different scenario and include a greater number of participants aiming for statistical significance among the samples.

8.3.3 Dialogue Authors Evaluation

This study was conducted to assess the reusability and costs of *OntoPlanDM* as well as the feasibility of non-experts to author a dialogue system by using it. The participants of this study played the role of a dialogue author (DA), i.e. they were expected to build a multi-turn goal-oriented dialogue system for the diagnosis of acute sinusitis. The purpose of the

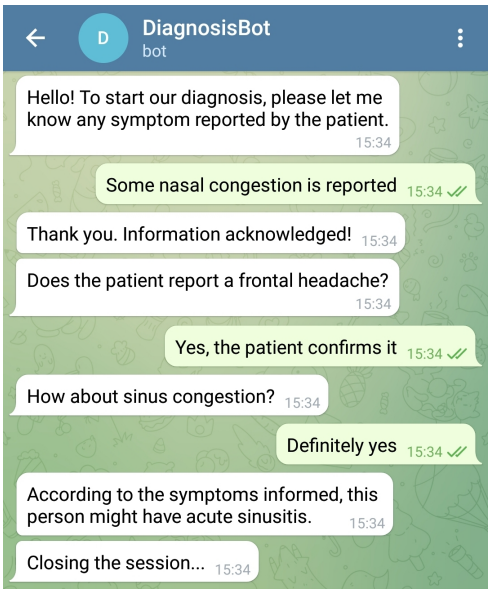


Figure 8.1: Sample of a conversation between the agent and an end-user of the dialogue system for acute sinusitis diagnosis.

proposed dialogue system is to acquire some information (symptoms related to acute sinusitis) with the goal of classifying the patient as having or not acute sinusitis. An example of the expected system can be seen in Figure 8.1. The domain-specific knowledge used in this study was obtained (and adapted) from *symcat*[7] and it is available in the Appendix C.

Participants

The participants for this study were required to have some technical skills like programming. Some knowledge of dialogue authoring and/or ontologies was also desired, although we opted to recruit participants with different skills to better understand the perceived complexity of the approach. No knowledge of automated planning was required.

Eight participants were recruited for the study; all had a graduate degree related to informatics. The average knowledge of the participants on AI planning was 2 (in a 1-5 scale), meaning that they did not have much

[7]http://www.symcat.com/welcome

Table 8.6: Participants divided into two groups according to their expertise

	Expertise in DS	Expertise in ontologies	Expertise in AI Planning
Group 1 (5DAs)	1	2.4	1.2
Group 2 (3DAs)	4.3	4	3.3
Overall (8DAs)	2.2	3	2

knowledge of this area. In fact, 4 participants (50%) declared to have no knowledge at all in AI planning. However, as previously discussed, this was just an informative item, as no knowledge in planning was required from the participants. Regarding knowledge of dialogue systems authoring, 5 participants declared to have no knowledge, 2 were experts and the last one declared to have some experience (rating: 3). Coincidently, the first 5 also did not have much experience with ontologies on average (2.4), while the last 3 had relevant experience (4). For this reason, to better analyse our results, we opted to divide the participants into two groups: Group 1, containing the non-expert DAs (with little experience in ontologies) and Group 2, with expert DAs (with experience in ontologies). Table 8.6 summarises the DAs' expertise.

Method

The study was conducted in person in a dedicated room and the sessions were held either individually or in small groups. Participants were provided with computers pre-configured with all libraries and tools necessary for the development of the new dialogue system. As we focus on evaluating the generation of the dialogue manager component only, we pre-configured the approach with *DialogFlow*[8] as the NLU layer. For NLG, instead, we asked the participants to input utterances as a property of Convology. The *Telegram* platform was also pre-configured.

[8]https://dialogflow.cloud.google.com/

At the beginning of each session, a short presentation describing *OntoPlanDM* was given. This presentation had the aim of introducing the knowledge necessary for a DA to use this approach. Next, the participants were provided with the documentation containing the directions to build a new system step-by-step (Appendix A). Then, they could start the implementation. For the implementation, the DA is required to instantiate a few classes in Convology and to define some settings at the code level. Although some technical knowledge is required for this implementation, no deep skills in ontologies or coding are required. To keep our study feasible, the participants were required to build only one information-seeking sub-dialogue. The timeframe of one and a half hours was defined for each session.

At the end of each session, a questionnaire was provided for each participant. This questionnaire first contained some demographic questions to understand the degree of expertise of the participants on the techniques used in this study. These questions could be answered on a five-point Likert scale from 1 ('*No knowledge*') to 5 ('*Expert*'). The next section of the questionnaire included items that were chosen to capture aspects related to the participant's perception of the complexity of using *OntoPlanDM* to build a new dialogue system. After concluding each session, the session admin filled a checklist of the items accomplished by each participant.

Results

The participants were able to follow the instructions and conclude the implementation within the timeframe specified, delivering a working dialogue system at the end of the session. Although this was a very simplified version of a dialogue system, the bot was able to retrieve information and provide the end-user with a diagnosis, either positive or negative regarding acute sinusitis. There was an exception of two dialogue systems that did

not fully work. However, we opted to not exclude these two results since the dialogue systems did not work (or partially worked) due to technical errors not related to the approach and, in this case, we consider the feedback of these participants as relevant for our study. Next, we discuss the outcomes according to the goals that were established for this study.

Feasibility of non-experts to author a dialogue system To discuss the feasibility of non-experts to author a dialogue manager by using *OntoPlanDM*, we first analyzed the answers provided by Group 1, the less experienced group. All members of this group were able to conclude the creation of the dialogue system[9]. The participants' feedback was quite positive and they seemed to not find the instructions too complex to follow as only two of them requested support from the session admin and it happened a maximum of two times (per participant).

Interestingly, different from Group1, Group2 found the approach more complex to use. This information was obtained by analyzing answers to the question "*What is your perception on the overall complexity of creating a new dialogue system by applying the proposed approach?*", which could be answered in a five-point Likert scale from 1 ('*Not complex at all*') to 5 ('*Very complex*'). While the overall complexity perceived by the former group was 2.2, the experienced DAs had an average of 3.3. In addition, two out of the three members of Group2 requested some support from the session admin. Based on their feedback, we understand that the software used to populate the ontology together with the instructions provided, which abstract several aspects of the dialogue system, limited their ability to create something from scratch. Besides, given the huge size of the resulting plan graphs, they were not exhibited to the authors; the outcome in the study

[9]One of the resulting systems worked partially as a consequence of a technical error that occurred during the session.

was only the running system. As a consequence, experienced authors felt somehow frustrated to not understand the impact of their actions on the overall agent. Finally, the use of more intuitive or user-friendly tools to populate the ontology was suggested by some participants.

In summary, the overall perceived complexity of the approach to build a new dialogue system is 2.6. Based on this information and the discussion above, we conclude that *OntoPlanDM* is feasible for non-experts, although there is room for improvement.

Reusability of the approach As previously described, excluding technical errors, all participants achieved a successfull implementation, being able to develop a working dialogue system. For feasibility purposes, the dialogue system was quite simplified, addressing only a short information-seeking sub-dialogue. However, it was fully able to conduct dialogue interactions with the end-user and achieve the dialogue goal. Rather than the specification of the domain-specific knowledge, no further adaptation was required in the approach to develop the new agent. The resulting agents included all features described in *OntoPlanDM*; they were able to replan the dialogue policy, and, it was possible to deploy the usefulness framework. Finally, 75% of the participants reported that given their experience after building the diagnosis agent, they think that it would be easy to generate a new dialogue system for a different domain.

Costs of building a new system In general, very few requests for assistance during the development of the agent were made by the participants. Therefore, we evaluate the costs of the approach based on the time taken to build the new agent and the expertise required.

Considering that the development of a real instance of a dialogue system would take at least a few days, the timeframe defined for each session was

relatively short (one and a half hours). Nonetheless, as previously mentioned, all participants were able to conclude the implementation within this timeframe. We also analyzed the participants' feedback regarding this matter. All participants from Group1 have judged the time taken to build the diagnosis dialogue system as "*appropriate*". In Group2, on the other hand, there was some divergence. While two participants consider the time "*short*", the other participant reports it as "*too long*". We believe that, similarly to the perceived complexity, the experienced participants had further expectations about the dialogue system and they were willing to further exploit the features of the approach. Meanwhile, the non-experienced authors were satisfied with the working agent.

Previously, we discussed the feasibility of non-experts authoring a dialogue system. Given our results, we identify that the degree of expertise required to build a new agent by employing this approach is relatively low. First, the non-requirement of expertise in automated planning is a positive aspect since not many dialogue authors have knowledge of this field. Next, considering that the overall expertise of the participants in dialogue authoring was 2.2 and all participants were able to complete the task, we also conclude that expertise in this aspect is not a requirement. Regarding knowledge of ontologies, 63% of the participants voted 4 or above when asked how confident they felt to populate Convology (average is 3.1). Errors during Convology population constituted only 15.6% of all knowledge populated and they were related to properties that were missing since the dialogue author did not comprehend or forgot to add them. This way, we conclude that some knowledge is welcome but no expertise in ontologies is required to use *OntoPlanDM*.

8.4 Discussion

The studies conducted to evaluate *OntoPlanDM* were aimed to provide some insight on aspects that are considered relevant for the health domain, such as runtime feasibility, task completion, and data accuracy. Some other aspects evaluated might have less relevance for the health domain, but they were considered for computational and optimization purposes. For example, a dialogue session with several turns does not necessarily mean a negative aspect if the agent decided to make further questions to reduce its uncertainty. But it is important to observe whether the agent is "wasting" interactions on questions that do not lead it to the goal.

The results of our studies revealed the need for improvement in some features of *Puffbot*. While some of these features can be easily implemented with the current approach, others require it to be improved. For example, additional questions, as suggested by the experts, can be treated as further slots (pieces of information to be acquired) even if these questions correspond to actions already done by the patient. Some end-users also argued that more questions should be made as they disagreed with the early classification made by the agent. In fact, low scores given by some experts regarding trustworthiness might be related to what they judged as *inconsistent results*. That is, in some sessions, the bot did not classify the crisis according to their expectation. Fortunately, this aspect is covered by the approach and such classifications can be easily adjusted by adopting the *cautious* instead of the *fast-solve* solution, which will result in additional questions to reduce the agent's uncertainty. In addition, tunning the classification threshold would lead to more precise results.

On the other hand, the strategy adopted to address scenarios where no conclusion can be reached requires a modification with the aim of avoiding the wrong assessment of risky situations. That is, instead of treating a

default classification as "no risk", the agent should inform the end-user that the outcome is *inconclusive* and an appropriate recommendation should be provided, e.g. "*Please, refer to your health provider for further assistance*". This strategy also does not require changes in the approach but, instead, the domain-specific knowledge must be updated by the dialogue designer.

Another matter directly related to domain-specific knowledge is the frustration demonstrated by some end-users with respect to the recommendations received; some of them suggested that additional instructions should be given by the agent. However, the agent was limited to the guideline presented in Section 2.1 and, as long as further instructions are defined by experts (and consequently modeled in Convology), they will automatically be suggested by the agent. Adaptation of the questions based on aspects such as the user profile, instead, is a feature that requires the integration of a new module, leaving an opportunity for future work.

Finally, as demonstrated by [105], other factors other than objective measures affect the user's satisfaction. In fact, *Puffbot* had very high scores for objective aspects such as task completion and accuracy. However, the usability analysis reveals that, although the overall score[10] (72.1%) is above the SUS threshold (68%), there is room for improvement in usability aspects. As the NLU and NLG components affect the user's perception of the agent, further studies using more robust natural language layers should be conducted to evaluate their influence on the usability score. Moreover, we believe that experience in the domain (i.e. asthma) had a strong influence on the users' judgment of the agent. In fact, instead of a tool for patients with frequent asthma episodes, both end-users and experts foresee *Puffbot* as a useful information tool for new patients or patients without a recent history of asthma attacks.

[10]Overall score considering the expert's score and the three settings applied to the end-user study.

Chapter 9

Conclusion

In this thesis, we presented *OntoPlanDM*, an approach that integrates automated planning and information management to support the modeling and management of health dialogues. *OntoPlanDM* aimed to cover some of the challenges in the development of dialogues for the health domain, such as prioritizing actions that lead to the identification of emergencies and delivering a predictable and explainable solution. While a conversational ontology was proposed to support knowledge management on both domain-independent and specific aspects, AI planning was exploited to automate the generation of the dialogue policy. Resulting dialogue managers (i) deliver a more natural flow by adopting mixed-initiated actions; (ii) divide the dialogue session in sub-dialogues according to their type (i.e. information-seeking or deliberation); (iii) select the most useful dialogue action on each state according to healthcare aspects (e.g. data priority and possible risks); (iv) replan the dialogue policy in case of error or non-expected events.

OntoPlanDM was evaluated from different perspectives. First, for an early version of the approach, we obtained feedback from domain experts. For this version, we also evaluated the computational performance of the planning solution. This analysis was repeated later to address the final ver-

sion of *OntoPlanDM*. Finally, three different user studies were conducted, for which we recruited health experts, asthma patients, and dialogue authors. While two studies exploited a prototype for the asthma domain; the study involving dialogue authors analyzed the feasibility and costs of using the approach for a new dialogue domain (i.e. diagnosis).

This work advances the state of the art in health dialogue management by automating the generation of efficient dialogue managers with a reduced cost since they do not require handcrafting of the dialogue policy or large conversational datasets. As further contributions, this thesis presented a survey of the state-of-the-art on plan-based approaches for dialogue management and a conversational agent for the self-monitoring of the asthma disease. A list of publications that resulted from this research is available in section 1.2. Two other works that report the final version of the approach and our user studies are currently under submission.

Limitations. Although *OntoPlanDM* was able to deliver efficient dialogue managers, we identify a few limitations in this research. First, a scalability limitation of the planning model when addressing huge scale domains was identified for the last version of the approach. As previously discussed, this limitation can be addressed through the implementation of additional sub-dialogues. However, additional research focusing on the optimization of the planning solution can be conducted.

Next, some limitations regarding the user studies were identified. First, the domain addressed in our prototype was chosen given its relatively limited complexity. However, as pointed by some subjects of our user study, asthma patients usually already know how to proceed in a crisis. This aspect might have influenced the user perception of the system's usefulness and, consequently, the SUS score. For this reason, a user study for a different domain should be conducted to compare possible differences in the SUS scores. Similarly, a more extensive prototype that retrieves

additional information regarding the current situation should be tested to evaluate the user perception of the agent. Moreover, the sample size involving human subjects might not have challenged the approach enough to show its deficiencies. Therefore, after implementing the lessons learned in these experiments, this work could benefit from additional experiments that include a greater sample of subjects. However, as well known, such experiments are expensive in terms of time and financial aspects.

Finally, as discussed in this thesis, no official protocols or benchmarks are available for the evaluation of such systems. This limits the comparison of this approach to other works in the literature, limiting the assessment of its contribution. This way, our evaluation is expected to serve as a benchmark for future conversational agents addressing similar problems.

Future work. Besides addressing the limitations discussed above, we foresee additional opportunities for future research. First, extending *OntoPlanDM* to cover further dialogue types from the categorization of [121] is an aspect to be covered in future work. For example, an *inquiry* dialogue can be exploited to confirm agent's beliefs regarding the current situation and a *persuasion* dialogue can be employed on scenarios that require behavior change. Second, *adaptability* is also seen as an opportunity for future work since we noticed that the integration of a personalization component that retrieves information from external sources is not a trivial task. A very important aspect that must be exploited in future work is the handling of multiple intents in a single turn. The integration of Convology with domain-specific ontologies (e.g. a diabetes ontology) is also an open aspect. This integration would reduce even more the burden on the dialogue author deploying *OntoPlanDM* in a new scenario. Finally, addressing *empathetic* or *affective* frameworks is an open problem in several fields, including the health domain, and its inclusion in the proposed approach is to be investigated in future research.

Bibliography

[1] James Allen, Elisabeth André, Philip R Cohen, Dilek Hakkani-Tür, Ronald Kaplan, Oliver Lemon, and David Traum. Challenge discussion: advancing multimodal dialogue. In *The Handbook of Multimodal-Multisensor Interfaces: Language Processing, Software, Commercialization, and Emerging Directions-Volume 3*, pages 191–217. 2019.

[2] James Allen, George Ferguson, and Amanda Stent. An architecture for more realistic conversational systems. In *Proceedings of the 6th international conference on Intelligent user interfaces*, pages 1–8, 2001.

[3] Duygu Altinok. An ontology-based dialogue management system for banking and finance dialogue systems. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, 2018.

[4] Muhammad Amith, Kirk Roberts, and Cui Tao. Conceiving an application ontology to model patient human papillomavirus vaccine counseling for dialogue management. *BMC bioinformatics*, 20(21):1–16, 2019.

[5] John Langshaw Austin. *How to do things with words*. Oxford university press, 1975.

[6] Jayalakshmi Baskar and Helena Lindgren. Cognitive architecture of an agent for human-agent dialogues. In *International Conference on Practical Applications of Agents and Multi-Agent Systems*, pages 89–100. Springer, 2014.

[7] Gregor Behnke, Pascal Bercher, Matthias Kraus, Marvin Schiller, Kristof Mickeleit, Timo Häge, Michael Dorna, Michael Dambier, Dietrich Manstetten, Wolfgang Minker, et al. New developments for robert–assisting novice users even better in diy projects. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 30, pages 343–347, 2020.

[8] Gregor Behnke, Daniel Höller, Pascal Bercher, and Susanne Biundo. Change the plan—how hard can that be? In *Twenty-Sixth International Conference on Automated Planning and Scheduling*, 2016.

[9] Gregor Behnke, Florian Nielsen, Marvin Schiller, Pascal Bercher, Matthias Kraus, Wolfgang Minker, Birte Glimm, and Susanne Biundo. Sloth—the interactive workout planner. In *2017 International Conference on Companion Technology (ICCT)*, pages 1–6. IEEE, 2017.

[10] Gregor Behnke, Florian Nielsen, Marvin Schiller, Denis Ponomaryov, Pascal Bercher, Birte Glimm, Wolfgang Minker, and Susanne Biundo. To plan for the user is to plan with the user: Integrating user interaction into the planning process. In *Companion Technology*, pages 123–144. Springer, 2017.

[11] Gregor Behnke, Denis Ponomaryov, Marvin Schiller, Pascal Bercher, Florian Nothdurft, Birte Glimm, and Susanne Biundo. Coherence across components in cognitive systems—one ontology to rule them all. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*, 2015.

[12] Pascal Bercher, Susanne Biundo, Thomas Geier, Thilo Hoernle, Florian Nothdurft, Felix Richter, and Bernd Schattenberg. Plan, repair, execute, explain—how planning helps to assemble your home theater. In *Twenty-Fourth International Conference on Automated Planning and Scheduling*, 2014.

[13] Pascal Bercher, Daniel Höller, Gregor Behnke, and Susanne Biundo. More than a name? on implications of preconditions and effects of compound htn planning tasks. In *Proceedings of the Twenty-second European Conference on Artificial Intelligence*, pages 225–233. IOS Press, 2016.

[14] Julita Bermejo-Alonso, Jorge Salvador, and Ricardo Sanz. Towards an ontology for task and planning in autonomous systems: An emergency scenario. In *Iberian Robotics conference*, pages 429–440. Springer, 2017.

[15] Martin Beveridge and John Fox. Automatic generation of spoken dialogue from medical plans and ontologies. *Journal of biomedical informatics*, 39(5):482–499, 2006.

[16] Timothy Bickmore and Toni Giorgino. Some novel aspects of health communication from a dialogue systems perspective. In *AAAI Fall Symposium on Dialogue Systems for Health Communication*, pages 275–291, 2004.

[17] Timothy Bickmore and Toni Giorgino. Health dialog systems for patients and consumers. *Journal of biomedical informatics*, 39(5):556–571, 2006.

[18] Timothy W Bickmore, Daniel Schulman, and Candace L Sidner. A reusable framework for health counseling dialogue systems based on a behavioral medicine ontology. *Journal of biomedical informatics*, 44(2):183–197, 2011.

[19] Susanne Biundo and Andreas Wendemuth. Companion-technology for cognitive technical systems. *KI-Künstliche Intelligenz*, 30(1):71–75, 2016.

[20] Elizabeth Black, Amanda Coles, and Sara Bernardini. Automated planning of simple persuasion dialogues. In *International Workshop on Computational Logic and Multi-Agent Systems*, pages 87–104. Springer, 2014.

[21] Elizabeth Black, Amanda J Coles, and Christopher Hampson. Planning for persuasion. In *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems*, pages 933–942. International Foundation for Autonomous Agents and Multiagent Systems, 2017.

[22] Dan Bohus and Alexander I Rudnicky. The ravenclaw dialog management framework: Architecture and systems. *Computer Speech & Language*, 23(3):332–361, 2009.

[23] Willem Nico Borst. Construction of engineering ontologies for knowledge sharing and reuse. 1999.

[24] Adi Botea, Christian Muise, Shubham Agarwal, Oznur Alkan, Ondrej Bajgar, Elizabeth Daly, Akihiro Kishimoto, Luis Lastras, Radu Marinescu, Josef Ondrej, et al. Generating dialogue agents via automated planning. *arXiv preprint arXiv:1902.00771*, 2019.

[25] John Brooke et al. Sus-a quick and dirty usability scale. *Usability evaluation in industry*, 189(194):4–7, 1996.

[26] Richard Byrne. Planning meals: Problem-solving on a real data-base. *Cognition*, 5(4):287–332, 1977.

[27] Michael Cashmore, Anna Collins, Benjamin Krarup, Senka Krivic, Daniele Magazzeni, and David Smith. Towards explainable ai planning as a service. In *ICAPS Workshop on Explainable AI Planning (XAIP)*, 2019.

[28] Anthony Rocco Cassandra. *Exact and approximate algorithms for partially observable Markov decision processes*. Brown University, 1998.

[29] Hamid R Chinaei and Brahim Chaib-Draa. An inverse reinforcement learning algorithm for partially observable domains with application on healthcare dialogue management. In *2012 11th International Conference on Machine Learning and Applications*, volume 1, pages 144–149. IEEE, 2012.

[30] Laurence Cholvy and Célia da Costa Pereira. Usefulness of information for goal achievement. In *PRIMA*, volume 11873 of *Lecture Notes in Computer Science*, pages 123–137. Springer, 2019.

[31] G Churcher, Eric Stevan Atwell, and Clive Souter. *Dialogue management systems: a survey and overview*. University of Leeds, School of Computing Research Report 1997.06. 1997., 1997.

[32] Philip R Cohen. Back to the future for dialogue research: A position paper. *arXiv preprint arXiv:1812.01144*, 2018.

[33] Philip R Cohen. Foundations of collaborative task-oriented dialogue: What's in a slot? In *Proceedings of the 20th Annual SIGdial Meeting on Discourse and Dialogue*, pages 198–209, 2019.

[34] Marco Daniele, Paolo Traverso, and Moshe Y Vardi. Strong cyclic planning revisited. In *European Conference on Planning*, pages 35–48. Springer, 1999.

[35] Jan Deriu, Alvaro Rodrigo, Arantxa Otegi, Guillermo Echegoyen, Sophie Rosset, Eneko Agirre, and Mark Cieliebak. Survey on evaluation methods for dialogue systems. *Artificial Intelligence Review*, pages 1–56, 2020.

[36] Francesca Di Massimo, Valentina Carfora, Patrizia Catellani, and Marco Piastra. Applying psychology of persuasion to conversational agents through reinforcement learning: an exploratory study. 2019.

[37] Kutluhan Erol, James Hendler, and Dana S Nau. Htn planning: Complexity and expressivity. In *AAAI*, volume 94, pages 1123–1128, 1994.

[38] M. Fernández-López, A. Gómez-Pérez, and N. Juristo. Methontology: from ontological art towards ontological engineering. In *Proc. Symposium on Ontological Engineering of AAAI*, 1997.

[39] Richard E Fikes and Nils J Nilsson. Strips: A new approach to the application of theorem proving to problem solving. *Artificial intelligence*, 2(3-4):189–208, 1971.

[40] Michael David Fisher, Dov M Gabbay, and Lluis Vila. *Handbook of temporal reasoning in artificial intelligence*, volume 1. Elsevier, 2005.

[41] Annika Flycht-Eriksson and Arne Jönsson. Some empirical findings on dialogue management and domain ontologies in dialogue systems - implications from an evaluation of birdquest. In *Proceedings of the Fourth SIGdial Workshop on Discourse and Dialogue*, pages 158–167, 2003.

[42] Mary Ellen Foster and Ronald PA Petrick. Separating representation, reasoning, and implementation for interaction management: Lessons from automated planning. In *Dialogues with Social Robots*, pages 93–107. Springer, 2017.

[43] Valentina Franzoni, Alfredo Milani, and Jordi Vallverdú. Emotional affordances in human-machine interactive planning and negotiation. In *Proceedings of the International Conference on Web Intelligence*, pages 924–930. ACM, 2017.

[44] Artur Freitas, Daniela Schmidt, Alison Panisson, Felipe Meneguzzi, Renata Vieira, and Rafael H Bordini. Semantic representations of agent plans and planning problem domains. In *International Workshop on Engineering Multi-Agent Systems*, pages 351–366. Springer, 2014.

[45] Lucian Galescu, Choh Man Teng, James Allen, and Ian Perera. Cogent: A generic dialogue system shell based on a collaborative problem solving model. In *Proceedings of the 19th Annual SIGdial Meeting on Discourse and Dialogue*, pages 400–409, 2018.

[46] Angel García-Olaya, Raquel Fuentetaja, Javier García-Polo, José Carlos González, and Fernando Fernández. Challenges on the application of automated planning for comprehensive geriatric assessment using an autonomous social robot. In *Workshop of Physical Agents*, pages 179–194. Springer, 2018.

[47] Albert Gatt and Emiel Krahmer. Survey of the state of the art in natural language generation: Core tasks, applications and evaluation. *J. Artif. Intell. Res.*, 61:65–170, 2018.

[48] Hector Geffner and Blai Bonet. A concise introduction to models and methods for automated planning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 8(1):1–141, 2013.

[49] Christopher Geib, Bart Craenen, and Ronald PA Petrick. Generating collaborative behaviour through plan recognition and planning. In *Proceedings of the ICAPS 2016 Workshop on Distributed and Multi-Agent Planning (DMAP)*, pages 98–105, 2016.

[50] Malik Ghallab, Dana Nau, and Paolo Traverso. *Automated Planning: theory and practice*. Elsevier, 2004.

[51] Toni Giorgino, Ivano Azzini, Carla Rognoni, Silvana Quaglini, Mario Stefanelli, Roberto Gretter, and Daniele Falavigna. Automated spoken dialogue system for hypertensive patient home management. *International Journal of Medical Informatics*, 74(2-4):159–167, 2005.

[52] David Griol, Zoraida Callejas, Ramón López-Cózar, and Giuseppe Riccardi. A domain-independent statistical methodology for dialog management in spoken dialog systems. *Computer Speech & Language*, 28(3):743–768, 2014.

[53] Thomas R Gruber. A translation approach to portable ontology specifications. *Knowledge acquisition*, 5(2):199–220, 1993.

[54] Jaakko Hakulinen, Markku Turunen, Cameron Smith, Marc Cavazza, and Daniel Charlton. A model for flexible interoperability between dialogue management and domain reasoning for conversational spoken dialogue systems. In *Fourth International Workshop on Human-Computer Conversation, Bellagio, Italy*, 2008.

[55] Ourania Hatzi, Dimitris Vrakas, Nick Bassiliades, Dimosthenis Anagnostopoulos, and Ioannis Vlahavas. The porsce ii framework: Using ai planning for automated semantic web service composition. *The Knowledge Engineering Review*, 28(2):137–156, 2013.

[56] Tobias Heinroth, Dan Denich, Alexander Schmitt, and Wolfgang Minker. Efficient spoken dialogue domain representation and interpretation. In *LREC*, 2010.

[57] Pascal Hitzler, Aldo Gangemi, Krzysztof Janowicz, Adila Krisnadhi, and Valentina Presutti, editors. *Ontology Engineering with Ontology Design Patterns - Foundations and Applications*, volume 25 of *Studies on the Semantic Web*. IOS Press, 2016.

[58] Frank Honold, Pascal Bercher, Felix Richter, Florian Nothdurft, Thomas Geier, Roland Barth, Thilo Hörnle, Felix Schüssel, Stephan Reuter, Matthias Rau, et al.

Companion-technology: towards user-and situation-adaptive functionality of technical systems. In *2014 International Conference on Intelligent Environments*, pages 378–381. IEEE, 2014.

[59] Daniel Jurafsky and James H Martin. Dialog systems and chatbots. *Speech and language processing*, 3, 2017.

[60] Dipesh Kadariya, Revathy Venkataramanan, Hong Yung Yip, Maninder Kalra, Krishnaprasad Thirunarayanan, and Amit Sheth. Kbot: Knowledge-enabled personalized chatbot for asthma self-management. In *2019 IEEE International Conference on Smart Computing (SMARTCOMP)*, pages 138–143. IEEE, 2019.

[61] Barbara Kitchenham. Procedures for performing systematic reviews. *Keele, UK, Keele University*, 33(2004):1–26, 2004.

[62] Filippos Kominis and Hector Geffner. Multiagent online planning with nested beliefs and dialogue. In *Twenty-Seventh International Conference on Automated Planning and Scheduling*, 2017.

[63] Efstratios Kontopoulos, Dimitris Vrakas, Fotis Kokkoras, Nick Bassiliades, and Ioannis Vlahavas. An ontology-based planning system for e-course generation. *Expert Systems with Applications*, 35(1-2):398–406, 2008.

[64] Benjamin Krarup, Senka Krivic, Daniele Magazzeni, Derek Long, Michael Cashmore, and David E. Smith. Contrastive explanations of plans through model restrictions. *J. Artif. Int. Res.*, 72:533–612, November 2021.

[65] Liliana Laranjo, Adam G Dunn, Huong Ly Tong, Ahmet Baki Kocaballi, Jessica Chen, Rabia Bashir, Didi Surian, Blanca Gallego, Farah Magrabi, Annie YS Lau, et al. Conversational agents in healthcare: a systematic review. *Journal of the American Medical Informatics Association*, 25(9):1248–1258, 2018.

[66] Keonsoo Lee, Jae Kwan Kim, Myon Woong Park, and Laehyun Kim. Situation based dynamic planning for dialog support with hierarchical knowledge. In *2016 International Conference on Platform Technology and Service (PlatCon)*, pages 1–4. IEEE, 2016.

[67] Keonsoo Lee, Yang Sun Lee, and Yunyoung Nam. A model of fsm-based planner and dialogue supporting system for emergency call services. *The Journal of Supercomputing*, 74(9):4603–4612, 2018.

[68] Kyusong Lee, Hongsuck Seo, Junhwi Choi, Sangjun Koo, and Gary Geunbae Lee. Conversational knowledge teaching agent that uses a knowledge base. In *Proceedings of the 16th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 139–143, 2015.

[69] Daniel Leidner, Christoph Borst, Alexander Dietrich, Michael Beetz, and Alin Albu-Schäffer. Classifying compliant manipulation tasks for automated planning in robotics. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1769–1776. IEEE, 2015.

[70] Clayton Lewis. *Using the" thinking-aloud" method in cognitive interface design*. IBM TJ Watson Research Center Yorktown Heights, NY, 1982.

[71] Zhaojiang Lin, Peng Xu, Genta Indra Winata, Farhad Bin Siddique, Zihan Liu, Jamin Shin, and Pascale Fung. Caire: An end-to-end empathetic chatbot. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 13622–13623, 2020.

[72] Pierre Lison. Towards online planning for dialogue management with rich domain knowledge. In *Natural Interaction with Robots, Knowbots and Smartphones*, pages 111–123. Springer, 2014.

[73] Bing Liu and Sahisnu Mazumder. Lifelong and continual learning dialogue systems: learning during conversation. *Proceedings of AAAI-2021*, 2021.

[74] Dongcai Lu, Shiqi Zhang, Peter Stone, and Xiaoping Chen. Leveraging commonsense reasoning and multimodal perception for robot spoken dialog systems. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6582–6588. IEEE, 2017.

[75] Susann LuperFoy, Dan Loehr, David Duff, Keith Miller, Florence Reeder, and Lisa Harper. An architecture for dialogue management, context tracking, and pragmatic adaptation in spoken dialogue systems. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics-Volume 2*, pages 794–801. Association for Computational Linguistics, 1998.

[76] Tânia Marques and Michael Rovatsos. Toward domain-independent dialogue planning.

[77] Georgios Meditskos, Efstratios Kontopoulos, Stefanos Vrochidis, and Ioannis Kompatsiaris. Converness: Ontology-driven conversational awareness and context understanding in multimodal dialogue systems. *Expert Systems*, 37(1):e12378, 2020.

[78] Manish Mehta and Andrea Corradini. Developing a conversational agent using ontologies. In *International Conference on Human-Computer Interaction*, pages 154–164. Springer, 2007.

[79] David Milward. Ontologies and the structure of dialogue. In *Proceedings of the 8th Workshop on the Semantics and Pragmatics of Dialogue (Catalog*, pages 69–77, 2004.

[80] Fabrizio Morbini, David DeVault, Kenji Sagae, Jillian Gerten, Angela Nazarian, and David Traum. Flores: a forward looking, reward seeking, dialogue manager. In *Natural interaction with robots, knowbots and smartphones*, pages 313–325. Springer, 2014.

[81] Christian Muise, Tathagata Chakraborti, Shubham Agarwal, Ondrej Bajgar, Arunima Chaudhary, Luis A Lastras-Montano, Josef Ondrej, Miroslav Vodolan, and Charlie Wiecha. Planning for goal-oriented dialogue systems. *arXiv preprint arXiv:1910.08137*, 2019.

[82] Christian Muise, Sheila A McIlraith, and Vaishak Belle. Non-deterministic planning with conditional effects. In *Twenty-Fourth International Conference on Automated Planning and Scheduling*, 2014.

[83] Setareh Nasihati Gilani, David Traum, Arcangelo Merla, Eugenia Hee, Zoey Walker, Barbara Manini, Grady Gallagher, and Laura-Ann Petitto. Multimodal dialogue management for multiparty interaction with infants. In *Proceedings of the 2018 on International Conference on Multimodal Interaction*, pages 5–13. ACM, 2018.

[84] Florian Nothdurft, Gregor Behnke, Pascal Bercher, Susanne Biundo, and Wolfgang Minker. The interplay of user-centered dialog systems and ai planning. In *Proceedings of the 16th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 344–353, 2015.

[85] Florian Nothdurft, Pascal Bercher, Gregor Behnke, and Wolfgang Minker. User involvement in collaborative decision-making dialog systems. In *Dialogues with Social Robots*, pages 129–141. Springer, 2017.

[86] Florian Nothdurft, Stefan Ultes, and Wolfgang Minker. Finding appropriate interaction strategies for proactive dialogue systems—an open quest. In *Proceedings of the 2nd European and the 5th Nordic Symposium on Multimodal Communication, August 6-8, 2014, Tartu, Estonia*, number 110, pages 73–80. Linköping University Electronic Press, 2015.

[87] Alison R Panisson, Giovani Farias, Artur Freitas, Felipe Meneguzzi, Renata Vieira, and Rafael H Bordini. Planning interactions for agents in argumentation-based negotiation. In *Proc. of 11th Int. Workshop on Argumentation in Multi-Agent Systems*, 2014.

[88] Alexandros Papangelis, Georgios Galatas, Konstantinos Tsiakas, Alexandros Lioulemes, Dimitrios Zikos, and Fillia Makedon. A dialogue system for ensuring safe rehabilitation. In *International Conference on Universal Access in Human-Computer Interaction*, pages 349–358. Springer, 2014.

[89] Alexandros Papangelis, Robert Gatchel, Vangelis Metsis, and Fillia Makedon. An adaptive dialogue system for assessing post traumatic stress disorder. In *Proceedings of the 6th International Conference on Pervasive Technologies Related to Assistive Environments*, page 49. ACM, 2013.

[90] Pere Pardo and Lluís Godo. A temporal argumentation approach to cooperative planning using dialogues. *Journal of Logic and Computation*, 28(3):551–580, 2018.

[91] Baolin Peng, Xiujun Li, Jianfeng Gao, Jingjing Liu, Kam-Fai Wong, and Shang-Yu Su. Deep dyna-q: Integrating planning for task-completion dialogue policy learning. *arXiv preprint arXiv:1801.06176*, 2018.

[92] Ronald Petrick and Mary Ellen Foster. Action selection for interaction management: Opportunities and lessons for automated planning. 2016.

[93] Ronald PA Petrick and Fahiem Bacchus. A knowledge-based approach to planning with incomplete information and sensing. In *AIPS*, volume 2, pages 212–222, 2002.

[94] Ronald PA Petrick and Mary Ellen Foster. Using general-purpose planning for action selection in human-robot interaction. In *2016 AAAI Fall Symposium Series*, 2016.

[95] Ronald PA Petrick and Mary Ellen Foster. Knowledge engineering and planning for social human–robot interaction: A case study. In *Knowledge Engineering Tools and Techniques for AI Planning*, pages 261–277. Springer, 2020.

[96] Mark Petticrew and Helen Roberts. *Systematic reviews in the social sciences: A practical guide*. John Wiley & Sons, 2008.

[97] Louisa Pragst, Juliana Miehle, Wolfgang Minker, and Stefan Ultes. Challenges for adaptive dialogue management in the kristina project. In *Proceedings of the 1st ACM SIGCHI International Workshop on Investigating Social Interactions with Artificial Agents*, pages 11–14, 2017.

[98] W Price and II Nicholson. Black-box medicine. *Harv. JL & Tech.*, 28:419, 2014.

[99] Abdul Quamar, Chuan Lei, Dorian Miller, Fatma Ozcan, Jeffrey Kreulen, Robert J Moore, and Vasilis Efthymiou. An ontology-based conversation system for knowledge bases. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, pages 361–376, 2020.

[100] Lina Maria Rojas-Barahona and Toni Giorgino. Adaptable dialog architecture and runtime engine (adarte): a framework for rapid prototyping of health dialog systems. *international journal of medical informatics*, 78:S56–S68, 2009.

[101] Milene Santos Teixeira and Mauro Dragoni. A review of plan-based approaches for dialogue management. *Cognitive Computation*, pages 1–20, 2022.

[102] Marvin Schiller, Gregor Behnke, Mario Schmautz, Pascal Bercher, Matthias Kraus, Michael Dorna, Wolfgang Minker, Birte Glimm, and Susanne Biundo. A paradigm for coupling procedural and conceptual knowledge in companion systems. In *2017 International Conference on Companion Technology (ICCT)*, pages 1–6. IEEE, 2017.

[103] Daniel Schulman, Timothy Bickmore, and Candace Sidner. An intelligent conversational agent for promoting long-term health behavior change using motivational interviewing. In *2011 AAAI Spring Symposium Series*, 2011.

[104] Zohreh Shams, Marina De Vos, Nir Oren, and Julian Padget. Normative practical reasoning via argumentation and dialogue. In *Proceedings of the 25th International Joint Conference on Artificial Intelligence (IJCAI-16)*. AAAI Press, 2016.

[105] Annika Silvervarg and Arne Jönsson. Subjective and objective evaluation of conversational agents in learning environments for young teenagers. In *Proceedings of the 7th IJCAI Workshop on Knowledge and Reasoning in Practical Dialogue Systems*, volume 84. Citeseer, 2011.

[106] Cameron Smith, Marc Cavazza, Daniel Charlton, Li Zhang, Markku Turunen, and Jaakko Hakulinen. Integrating planning and dialogue in a lifestyle agent. In *International Workshop on Intelligent Virtual Agents*, pages 146–153. Springer, 2008.

[107] Daniel Sonntag, Christian Schulz, Christian Reuschling, and Luis Galarraga. Radspeech's mobile dialogue system for radiologists. In *Proceedings of the 2012 ACM international conference on Intelligent User Interfaces*, pages 317–318. ACM, 2012.

[108] Sarath Sreedharan, Tathagata Chakraborti, Christian Muise, Yasaman Khazaeni, and Subbarao Kambhampati. –d3wa+–a case study of xaip in a model acquisition task for dialogue planning. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 30, pages 488–497, 2020.

[109] Sarath Sreedharan, Siddharth Srivastava, David Smith, and Subbarao Kambhampati. Why can't you do that hal? explaining unsolvability of planning tasks. In *Proc. IJCAI*, 2019.

[110] Mario Stefanelli and Lina Maria Rojas Barahona. Health care dialogue systems: Practical and theoretical approaches to dialogue management. 2009.

[111] Hiroaki Takatsu, Ryota Ando, Yoichi Matsuyama, and Tetsunori Kobayashi. Sentiment analysis for emotional speech synthesis in a news dialogue system. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 5013–5025, 2020.

[112] Paul Tarau and Elizabeth Figa. Knowledge-based conversational agents and virtual storytelling. In *Proceedings of the 2004 ACM symposium on Applied computing*, pages 39–44, 2004.

[113] Milene S Teixeira, Mauro Dragoni, and Claudio Eccher. A planning strategy for dialogue management in healthcare. In *SWH 2019 - Second International Workshop on Semantic Web Meets Health Data Management*, 2019.

[114] Milene S Teixeira, Vinicius Maran, and Mauro Dragoni. The interplay of a conversational ontology and ai planning for health dialogue management. In *The 36th ACM/SIGAPP Symposium On Applied Computing*, 2021.

[115] Milene Santos Teixeira, Célia da Costa Pereira, and Mauro Dragoni. Information usefulness as a strategy for action selection in health dialogues. In *2020*

IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT), pages 323–330. IEEE, 2020.

[116] Milene Santos Teixeira, Vinícius Maran, and Mauro Dragoni. Towards semantic-awareness for information management and planning in health dialogues. In *2021 IEEE 34th International Symposium on Computer-Based Medical Systems (CBMS)*, pages 372–377. IEEE, 2021.

[117] Milene Santos Teixeira, Célia da Costa Pereira, and Mauro Dragoni. A goal-based framework for supporting medical assistance: The case of chronic diseases. In *International Conference on Principles and Practice of Multi-Agent Systems*, pages 281–298. Springer, 2020.

[118] Pedro Elkind Velmovitsky, Marx Viana, Elder Cirilo, Ruy Luiz Milidiu, Plinio Pelegrini Morita, and Carlos José Pereira de Lucena. Promoting reusability and extensibility in the engineering of domain-specific conversational systems. In *2019 8th Brazilian Conference on Intelligent Systems (BRACIS)*, pages 473–478. IEEE, 2019.

[119] Marilyn Walker and Steve Whittaker. Mixed initiative in dialogue: An investigation into discourse segmentation. *arXiv preprint cmp-lg/9504007*, 1995.

[120] Marilyn A Walker, Diane J Litman, Candace A Kamm, and Alicia Abella. Paradise: A framework for evaluating spoken dialogue agents. *arXiv preprint cmp-lg/9704004*, 1997.

[121] Douglas Walton and Erik CW Krabbe. *Commitment in dialogue: Basic concepts of interpersonal reasoning*. SUNY press, 1995.

[122] Xiaoyan Wang, Amy E. Chused, Noémie Elhadad, Carol Friedman, and Marianthi Markatou. Automated knowledge acquisition from clinical narrative reports. In *AMIA 2008, American Medical Informatics Association Annual Symposium, Washington, DC, USA, November 8-12, 2008*. AMIA, 2008.

[123] Zhongyu Wei, Qianlong Liu, Baolin Peng, Huaixiao Tou, Ting Chen, Xuanjing Huang, Kam-Fai Wong, and Xiangying Dai. Task-oriented dialogue system for automatic diagnosis. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 201–207, 2018.

[124] Henry Weld, Xiaoqi Huang, Siqi Long, Josiah Poon, and Soyeon Caren Han. A survey of joint intent detection and slot-filling models in natural language understanding. *CoRR*, abs/2101.08091, 2021.

[125] Michael Wessel, Girish Acharya, James Carpenter, and Min Yin. Ontovpa—an ontology-based dialogue management system for virtual personal assistants. In *Advanced Social Interaction with Agents*, pages 219–233. Springer, 2019.

[126] Jason Williams, Antoine Raux, and Matthew Henderson. The dialog state tracking challenge series: A review. *Dialogue & Discourse*, 7(3):4–33, 2016.

[127] Ugan Yasavur. Statistical dialog management for health interventions. 2014.

[128] Steve Young, Milica Gašić, Blaise Thomson, and Jason D Williams. Pomdp-based statistical spoken dialog systems: A review. *Proceedings of the IEEE*, 101(5):1160–1179, 2013.

[129] Steve J Young. Probabilistic methods in spoken–dialogue systems. *Philosophical Transactions of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences*, 358(1769):1389–1402, 2000.

[130] Tom Young, Vlad Pandelea, Soujanya Poria, and Erik Cambria. Dialogue systems with audio context. *Neurocomputing*, 388:102–109, 2020.

[131] Monika Žáková, Petr Křemen, Filip Železnỳ, and Nada Lavrač. Automating knowledge discovery workflow composition through ontology-based planning. *IEEE Transactions on Automation Science and Engineering*, 8(2):253–264, 2010.

[132] Shiqi Zhang and Peter Stone. Corpp: Commonsense reasoning and probabilistic planning, as applied to dialog with a mobile robot. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

Appendix A

Instructions for Dialogue Authors

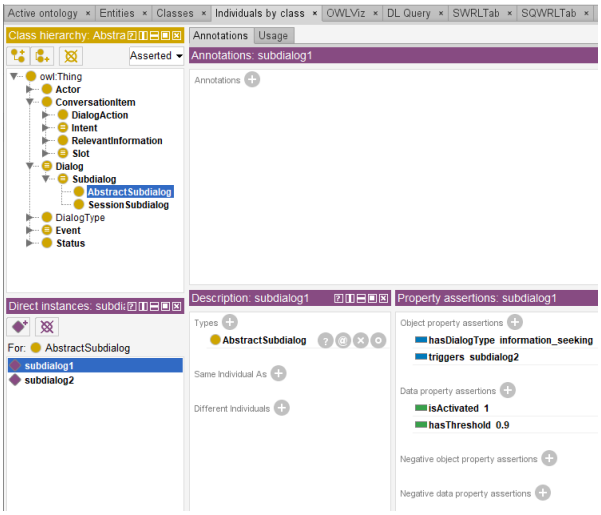
Convology Population

Steps to develop a new bot:

1. **Populate Convology**:
   The dialogue author must instantiate some knowledge in Convology as follows:

- ***AbstractSubdialog***: Instantiate as many subdialogs as necessary.
  - Properties required for each AbstractSubdialog instance:
    - *isActivated=1:* MUST be defined for the 1st subdialog only
    - *hasDialogType:* range is either *information_seeking* or *deliberation*
    - *hasThreshold:* a value in the interval [0...1] indicating the minimal confidence that is necessary to classify a status
    - *triggers:* to indicate some ordering among the subdialogs; range AbstractSubdialog (e.g. subdialog1 triggers subdialog2)

  Example:

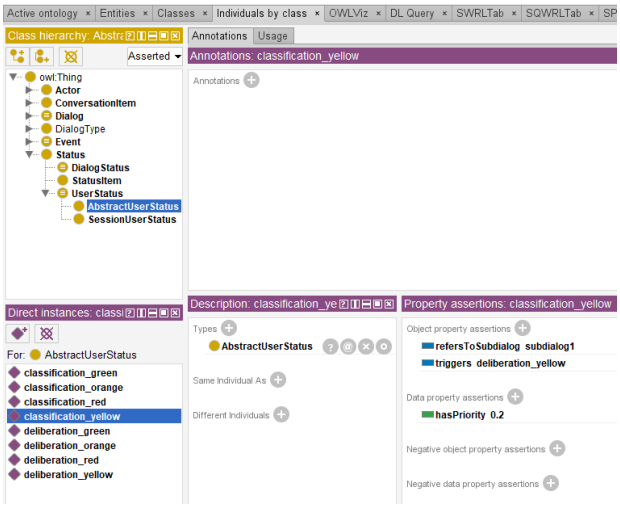


- ***AbstractUserStatus****:* Instantiate all possible classifications (goals) that can be made in each subdialog. *A deliberation subdialog also should have at least one AbstractUserStatus, that is related to the classification made in the previous subdialog (information_seeking).
  - Properties required **for each** AbstractUserStatus:
    - *refersToSubdialog*: indicate to which subdialog instance this UserStatus refers to (e.g. subdialog1)

- *triggers*: range AbstractUserStatus. Define this property if this AbstractUserStatus instance is related to any other AbstractUserStatus in the ***next*** subdialog (e.g. inf_seeking_us1 triggers deliberation_us1)
- *hasPriority* (optional): define this property if this AbstractUserStatus instance has any priority with respect to other AbstractUserStatus instances (if any) in the same subdialog. *Sum of all priorities must be 1!!

○ *isDefaultStatus*: for an information-seeking subdialog, make sure to specify an AbstractUserStatus instance as default (to be used when the information received during the dialogue session cannot reach any classification). The default status can be either one of the already created AbstractUserStatus instances (as long as it makes sense) or a new one can be created just for this purpose. If a new one is created, please set all the properties related to *AbstractUserStatus (*listed above*).*

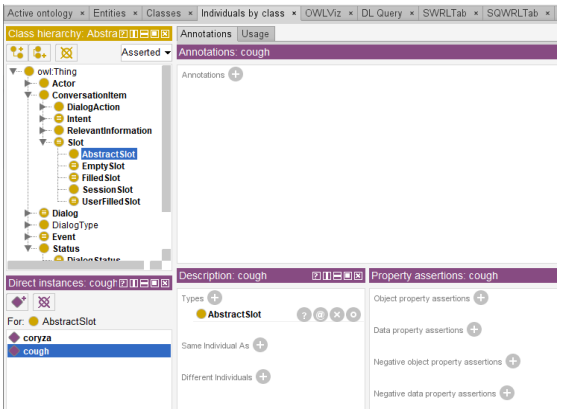
Example:

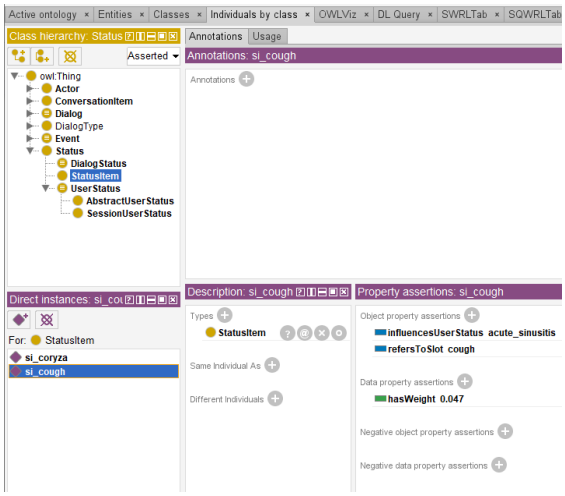


- ***AbstractSlot*** *and* ***StatusItem****:*
  For an *information seeking* dialogue, slots are instances of everything that can be asked/informed during a dialogue session (e.g. symptoms).
  For a *deliberation* dialogue, a single AbstractSlot instance MUST be created to be used as the outcome/recommendation to be given.
  *As *weights* are available for this domain (*accute_sinusitis*), please also define a *StatusItem* instance for each AbstractSlot that you define.
  AbstractSlot does not require any property to be instantiated, but the following properties should be defined for each StatusItem:

- *influencesUserStatus*: range AbstractUserStatus. Indicate which AbstractUserStatus this slot/status_item has influence in (e.g. which disease the symptom affects)
- *refersToSlot*: range AbstratSlot. Indicate to which slot this StatusItem refers to (e.g. si_cough(StatusItem) refersTo cough(AbstractSlot)).
- *hasWeight:* indicate the weight (*available in the domain knowledge chart).

Example:





- ***WEIGHTS***
  If weights are available for the slots, please follow the instructions below. Also, it is important to note that slots are unique instances, they should not be duplicated. In the case that a slot influences more than one UserStatus with possibly different weights (e.g. a symptom can help in the diagnosis of more than one disease), it should be handled in the following way:
    - A single slot instance must be defined (by the DA)
    - In case the domain REQUIRES weights, the DA should also define StatusItem instances (as many as necessary), instantiating the properties: *refersToSlot, influencesUserstatus, hasWeight*
    - If no weights/statusitem are defined by the DA, the populator is responsible for generating them and also for deriving the weights*

*Weights are part of domain-specific knowledge and, therefore, a strategy for deriving them is out of our scope. It is important to note that such strategy changes according to the domain for which the DS is generated.

Appendix B

Implementation

This Appendix describes the implementation of *Puffbot*, the prototype that deploys *OntoPlanDM*, the approach proposed in this thesis. *Puffbot* is a multi-turn goal-oriented conversational agent that aims at supporting patients affected by asthma. *Puffbot* acquires information through dialogue, having as its dialogue goal to reach the real-time classification of patients' status. Then, it provides them with recommendations that aim to improve their current health condition. In other words, *Puffbot* asks a patient about present symptoms, classifies the current situation, and, based on this classification, provides a direction on what the patient should do next. The latest version of *Puffbot* implements the full approach discussed in chapter 7. Some details of its implementation are given below.

Model Acquisition The domain-specific knowledge deployed into *Puffbot* was acquired with the collaboration of domain experts (pulmonologists) from the Trentino Healthcare Department and it was described in chapter 2. The ABox of Convology was instantiated according to this knowledge and, on code level, the Python library *Owlready2*[1] was used to handle the ontology. The deployment of the approach requires the definition of parameters that were not described within the asthma action plan. However, these parameters were defined during model acquisition (Table B.1).

Planning Module The planning parser described in section 7.2 (*Convology2PDDL*) models the dialogue as a planning problem. To write the resulting PDDL file, we exploited a third-party parser, which is available at *https://github.com/boompig/pddl-parser-2*. FOND plans are generated (and updated) by the automated Planner for Relevant

[1]https://owlready2.readthedocs.io

Attribute	Value	Description
classification_threshold	0.6	Threshold to consider a goal achieved (part of the usefulness framework described in Chapter 6)
intent_confidence	0.5	Threshold that defines the minimun confidence value on the interpretation of an intent
n_previous_dialog_actions	3	Number of previous states to visit when trying to recognize an intent

Table B.1: Attributes defined for *Puffbot*.

Policies (PRP)[2].

Execution Module *Puffbot* was deployed as a *Telegram* bot. The python library available at *https://python-telegram-bot.readthedocs.io* was used for its implementation. Execution requires the integration of the generated dialogue manager with the natural language layers. This way, for the natural language understanding (NLU) component, we exploited Dialogflow[3]. Intents were created in Dialogflow matching the intent instances in Convology and, for each intent, a few sentences were added to train its recognition. For natural language generation (NLG), instead, we relied on predefined utterances that were added directly to Convology instances (i.e. property *hasUtterance*). For future work, the use of a more robust NLG layer is intended.

[2]https://github.com/QuMuLab/planner-for-relevant-policies
[3]https://cloud.google.com/dialogflow/

Appendix C

Domain-specific Knowledge: Acute Sinusitis

Disease name	acute_sinusitis		
Symptom	**weight**		
cough	0.047		
nasal_congestion	0.044		
sore_throat	0.039		
headache	0.039		
frontal_headache	0.328		
coryza	0.03		
fever	0.03		
ear_pain	0.029		
sinus_congestion	0.217		
painful_sinuses	0.17		
facial_pain	0.015		
coughing_up_spu	0.013		
Common Tests and Procedures			
Patients with acute sinusitis often receive other diagnostic procedures (interview; evaluation; consultation), cat scan of head, influenzavirus antibody assay, other respiratory therapy, physical therapy exercises; manipulation; and other procedures and tracheoscopy and laryngoscopy with biopsy .			