2012

Topics in Knowledge Bases: Epistemic Ontologies and Secrecy-preserving Reasoning

Jia Tao
Iowa State University

Topics in knowledge bases:

Epistemic ontologies and secrecy-preserving reasoning

by

Jia Tao

A dissertation submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Major: Computer Science

Program of Study Committee:

Vasant Honavar, Co-major Professor

Giora Slutzki, Co-major Professor

Ying Cai

Carl Chang

Shashi Gadia

Iowa State University

Ames, Iowa

2012

DEDICATION

This dissertation is dedicated to my parents and my sisters, for their loving support, endless patience, constant encouragement and sage advice without which I would not have been able to complete this work.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ACKNOWLEDGEMENTS

I would like to take this opportunity to express my thanks to those who helped me with various aspects of conducting research and the writing of this dissertation. First and foremost, I offer my sincerest gratitude to my advisors, Dr. Vasant Honavar and Dr. Giora Slutzki, who have guided me throughout my dissertation with their patience and knowledge whilst allowing me the room to work in my own way. Without their insight, guidance and effort, this dissertation would not have taken shape. I am grateful to those endless discussions with them that have inspired me and helped develop algorithms and techniques in this dissertation. I am thankful to Dr. Carl Chang who has always been supportive. With his encouragement, I was able to put new elements in teaching a class and had great experience. I am in debt to Dr. Shashi Gadia, who has taught me to pursue truth and never compromise in front of truth, and to Dr. Ying Cai, who encouraged me to continue with the Ph.D program after my Master's, which helped me make the decision. I am also thankful to Dr. Hridesh Rajan, who shared his passion for research with me.

I would also like to thank my dad Shoushan Tao and my mom Yurui Chen, who raised me and my sisters and encourage us to pursue knowledge. They have always been role models to us. Their integrity, perseverance and loving-kindness are always in our hearts. I am grateful to my two sisters, Jing and Ying, my best friends. Their support has been instrumental during hard times.

I appreciate that I have many friends who have been beside me to support me during the past years with numerous discussions regarding research, life and interaction of these concepts: Mohammed Alabsi, Jie Bao, Jiqiu Chen, Qian Feng, Parichey Gandhi, Youssef Hanna, Khushboo Hemnani, Dimitris Kontogiannis, Glori Larpenter, Harris Lin, Suzhen Lin, Satyadev Nandakumar, Seo-young Noh, Santosh Panchapakesan, Fang Peng, Hongyan Sun, Fadi Towfic, Chuang Wang, Liyuan Xiao, Lu Xin, Li Xue.

ABSTRACT

Applications of ontologies/knowledge bases (KBs) in many domains (healthcare, national security, intelligence) have become increasingly important. In this dissertation, we focus on developing techniques for answering queries posed to KBs under the open world assumption (OWA) [1].

In the first part of this dissertation, we study the problem of query answering in KBs that contain epistemic information, i.e., knowledge of different experts. We study $\mathcal{ALCK}_m$, which extends the description logic $\mathcal{ALC}$ by adding modal operators of the basic multi-modal logic $\mathbf{K}_m$. We develop a sound and complete tableau algorithm $\Lambda_{\mathcal{K}}$ for answering $\mathcal{ALCK}_m$ queries w.r.t. an $\mathcal{ALCK}_m$ knowledge base with an acyclic TBox. We then consider answering $\mathcal{ALCK}_m$ queries w.r.t. an $\mathcal{ALCK}_m$ knowledge base in which the epistemic operators correspond to those of classical multi-modal logic $\mathbf{S4}_m$ and provide a sound and complete tableau algorithm Λ_{S4}. Both algorithms can be implemented in PSPACE.

In the second part, we study problems that allow autonomous entities or organizations (collectively called *querying agents*) to be able to selectively share information. In this scenario, the KB must make sure its answers are informative but do not disclose sensitive information. Most of the work in this area has focused on access control mechanisms that prohibit access to sensitive information (secrets). However, such an approach can be too restrictive in that it prohibits the use of sensitive information in answering queries against knowledge bases even when it is possible to do so without compromising secrets. We investigate techniques for secrecy-preserving query answering (SPQA) against KBs under the OWA. We consider two scenarios of increasing difficulty: (a) a KB queried by a single agent; and (b) a KB queried by multiple agents where the secrecy policies can differ across the different agents and the agents

[1] The closed world assumption is the presumption that a statement that cannot be inferred from the KB, is false. The open world assumption on the other hand is the presumption that a statement that cannot be inferred from the KB is *not necessarily* false.

can selectively communicate the answers that they receive from the KB with each other subject to the applicable answer sharing policies. We consider classes of KBs that are of interest from the standpoint of practical applications (e.g., description logics and Horn KBs). Given a KB and secrets that need to be protected against the querying agent(s), the SPQA problem aims at designing a secrecy-preserving reasoner that answers queries without compromising secrecy under OWA. Whenever truthfully answering a query risks compromising secrets, the reasoner is allowed to hide the answer to the query by feigning ignorance, i.e., answering the query as "Unknown". Under the OWA, the querying agent is not able to infer whether an "Unknown" answer to a query is obtained because of the incomplete information in the KB or because secrecy protection mechanism is being applied. In each scenario, we provide a general framework for the problem. In the single-agent case, we apply the general framework to the description logic $\mathcal{EL}$ and provide algorithms for answering queries as informatively as possible without compromising secrecy. In the multiagent case, we extend the general framework for the single-agent case. To model the communication between querying agents, we use a communication graph, a directed acyclic graph (DAG) with self-loops, where each node represents an agent and each edge represents the possibility of information sharing in the direction of the edge. We discuss the relationship between secrecy-preserving reasoners and envelopes (used to protect secrets) and present a special case of the communication graph that helps construct tight envelopes in the sense that removing any information from them will leave some secrets vulnerable. To illustrate our general idea of constructing envelopes, Horn KBs are considered.

CHAPTER 1. INTRODUCTION

The rapid expansion of the World Wide Web (WWW) and the widespread use of distributed databases and networked information systems offer unprecedented opportunities for productive interaction and collaboration among individuals and organizations in virtually every area of human endeavor. In many domains such as healthcare, with the increasing electronic data, applications that store domain information and manipulate stored information in an automated way have become more and more important. Knowledge representation and reasoning is an area of artificial intelligence research that is concerned with representing knowledge symbolically and manipulating the knowledge representation, by inference and otherwise, to create new knowledge. This dissertation focuses on two topics in this area.

1.1 Query Answering in Epistemic Knowledge Bases

In many applications, due to the specific domain knowledge from different experts, the ability to represent experts knowledge such as 'Dr. Vos *knows that* swine flu is a life threatening disease' rather than just the facts as 'swine flu is a life threatening disease' is desirable. Motivated by such applications, we study Description Logics (DLs) [1] which offer a powerful formalism for representing and reasoning with knowledge in a broad range of applications. Many DLs have been investigated with respect to their expressivity and complexity [2, 3, 4, 5]. Some DLs provide the foundation for powerful practical languages to represent knowledge on the web, e.g., DAML+OIL [6], OWL DL, OWL Lite [7], and reasoners (typically based on the analytic tableau method [4]) can be used to draw inferences from these DL knowledge bases [7]. Because of its inferential feasibility and practical utility, the *terminological* knowledge representation language $\mathcal{ALC}$ [2] is of particular interest. Representing knowledge in such a system

amounts to introducing the *terminology* of the application domain through *definitions* of the relevant concepts, and *assertions* that hold with respect to specific *individuals* in the domain. However, terminological knowledge representation languages such as $\mathcal{ALC}$ lack the expressivity needed to represent *modal* or *epistemic* aspects of knowledge.

Epistemic DLs allow us to address this limitation by providing a means to model as well as reason about the knowledge of different experts using epistemic operators. The resulting logic finds applications in settings where it is useful to be able to attribute specific pieces of knowledge to individual experts.

In Chapter 2, we augment $\mathcal{ALC}$ with an acyclic TBox with modal operators that can appear in front of any concept expressions, yielding a language which we refer to as $\mathcal{ALCK}_m$. We provide two sound and complete algorithms that can be implemented in PSPACE for the satisfiability of an $\mathcal{ALCK}_m$ query with respect to an $\mathcal{ALCK}_m$ knowledge base and the satisfiability of an $\mathcal{ALCS}4_m$ query with respect to an $\mathcal{ALCS}4_m$ knowledge base where $\mathcal{ALCS}4_m$ is an epistemically motivated language whose syntax is identical to that of $\mathcal{ALCK}_m$, but whose semantics is based on the modal logic $S4_m$.

1.2 Secrecy-preserving Query Answering Problem

With the increasing reliance on networked knowledge bases in virtually all areas of human endeavor that involve interactions among organizations, those that provide healthcare (hospitals, pharmacies, insurance providers), governmental agencies (e.g., intelligence, law enforcement, public policy), or independent nations acting on matters of global concern (e.g., counter-terrorism, international finance) call for information sharing between organizations. However, the need to share information often has to be balanced against the need to protect sensitive information or *secrets* from unintended disclosure, e.g., due to copyright, privacy, security, or commercial considerations. Barring few examples (see below), most approaches to information protection, e.g., access control methods in databases [8, 9, 10, 11] and on policy languages (see [12] for a survey), including those that focus on selective access to information on the web [13, 14, 15, 16, 17] simply forbid the use of secret information in answering queries.

As Weitzner et al. [18] have recently noted, "*excessive reliance on secrecy and up-front*

control over information has yielded policies that fail to meet social needs, as well as technologies that stifle information flow". The controlled query evaluation (CQE) framework [19] offers a way to answer database queries without revealing secrets [20, 21, 22]. This work has focused on protecting secrets in (typically relational) databases under the closed world assumption (CWA), using techniques that may rely on lying (i.e., responding to queries with answers that are inconsistent with the knowledge base) in addition to refusing to answer. However, KBs that contain knowledge about the real world, e.g., scientific or medical knowledge are necessarily incomplete. Hence semantics based on OWA is better suited than that based on CWA in such settings. To the best of our knowledge, Bao et al. (2007) were the first to consider the SPQA problem under OWA, albeit in the restricted setting of a hierarchical KB with a single querying agent. In this case, query answering reduces to checking reachability in a directed acyclic graph, and protecting a secret is tantamount to hiding the reachability of a given target node from a given source node. More recently, Tao et al. (2010) provided a solution to the SPQA problem for instance checking in the description logic $\mathcal{EL}$, with a single querying agent.

In this part of this dissertation, techniques for secrecy-preserving query answering against KBs under OWA are investigated in two scenarios of increasing difficulty: (a) a KB queried by a single agent; and (b) a KB queried by multiple agents where the secrecy policies can differ across the different agents and the agents can selectively communicate the answers that they receive from the KB with each other subject to the applicable answer sharing policies.

In Chapter 3, a simplified version of the SPQA problem which consists of a single querying agent is presented. Given a KB Σ and a finite set $\mathbb{S}$ of secrets, called a *secrecy set*, the secrecy-preserving query answering problem aims at designing a *secrecy-preserving* reasoner that answers queries without revealing any secrets. A formal framework modeling most of the aspects of the problem is provided. In general, the answer to a query q against a KB Σ can be "Yes" (q can be inferred from Σ), "No" ($\neg q$ can be inferred from Σ) or "Unknown" (e.g., because of the incompleteness of Σ). We assume a cooperative rather than adversarial scenarios in which the KB does not *lie*. However, whenever truthfully answering a query risks compromising secrets in $\mathbb{S}$, the reasoner associated with the KB is allowed to hide the answer to the query by feigning ignorance, i.e., answering the query as "Unknown".

One way to answer queries while preserving secrecy is to evaluate every query when it is posed to the KB. If the truthful answer to the query together the query history compromises some secret, an "Unknown" will be retrieved. Otherwise, the query will be faithfully answered. This strategy is called a *lazy evaluation*. Since this approach checks the query history every time a query is posed, when the history gets larger, the response time gets longer, and therefore, the approach is getting less and less appealing.

Therefore, we propose an approach in which we precompute a *secrecy envelope* used to protect secrets such that the querying agent who has no access to the contents of the envelope will not be able to deduce any secrets. To compute such an envelope, we try to "disrupt" all proofs of secrets that an envelope needs to protect. To illustrate the basic idea, consider a formula $\alpha \wedge \beta$ in propositional logic. Suppose $\Gamma \vDash \alpha$ and $\Gamma \vDash \beta$. Obviously, $\Gamma \vDash \alpha \wedge \beta$. If we need to protect $\alpha \wedge \beta$, we must protect at least one of α and β, i.e., if $\alpha \wedge \beta$ is a secret, at least one of α and β will be in the corresponding envelope. The formula, α or β, that is relegated to the envelope is "disrupting" a proof of $\alpha \wedge \beta$ from $\{\alpha, \beta\}$. Roughly, an envelope will be built from all the disrupting formulas.

It is easy to see that a secrecy envelope always exists. For instance, $\Sigma^+ \setminus \{\text{tautologies}\}$ constitutes an envelope for any secrecy set $\mathbb{S} \subseteq \Sigma^+$. A key challenge is to *develop strategies that can be used by the KB to respond to queries as informatively as possible (i.e., using an envelope that is as small as possible) without compromising secrets that the KB is obliged to protect*. Unfortunately, it turns out that given a language that is expressive enough, computing a minimum envelope may be NP-hard (see Section 3.5.1). Therefore, we aim at computing envelopes that are *tight* in the sense that removing any information from such an envelope will leave the secrecy set vulnerable. In general, if an envelope is finite, for each element in the envelope, we could test whether it is necessary to protect the secrets. If it is not, it can be removed. After all the elements in the envelope are tested, a tight envelope is obtained. Since computing tight envelopes is an optimization problem, depending on the underlying language, algorithms may be designed to directly build an envelope that is tight.

We apply this approach to $\mathcal{EL}$ [25], which is one of the simpler Description Logics (DLs) that is both computationally tractable [26, 27, 28] and practically useful [25, 29]. For example,

the medical ontology SNOMED CT [30] and large parts of the medical ontology GALEN [31] can be expressed in $\mathcal{EL}$. We provide algorithms to computes envelopes and tight envelopes to answer queries against an $\mathcal{EL}$ knowledge base that use, but do not reveal, the information that is designated as secret.

In Chapter 4, we extend the single-agent framework to multiple querying agents where the querying agents are allowed to communicate by sharing some of the query answers obtained from the KB. Since secrecy sets for different agents may be different, instead of allowing free communication, which essentially eliminates the distinction between different agents, we use a *communication graph*, a directed acyclic graph (except for all the self-loops) to restrict how the agents communicate. In a communication graph, each node represents a querying agent and each edge denotes the permission of information sharing from a querying agent to its successor. We consider a general situation where agents are only allowed to pass their own query answers obtained from the KB rather than "gossip" about the information obtained from their predecessors. This chapter aims at designing a *secrecy-preserving* reasoner that answers queries so that none of the querying agents can infer any of the secrets which the knowledge base is obliged to protect against them.

Similarly to the single-agent case, we utilize OWA and protect secrets by feigning ignorance. If the truthful answer to q risks compromising any of the querying agents' secrets, the answer to q will be censored to be "Unknown" by the KB reasoner. Such a reasoner can be designed using lazy evaluation, which, as we mentioned before, becomes less attractive over time. In view of this, we provide a general approach of precomputing an envelope such that none of the querying agents can deduce any secrets that need to be protected against itself from the queries answers it obtains from both the KB and its predecessors. We provide a general framework for SPQA problem in the multiple querying agents setting and discuss the relationship between a secrecy-preserving reasoner and an envelope that protects secrets. We then use Horn KBs as an example to present the idea of computing envelopes by "disrupting" proofs.

CHAPTER 2. QUERY ANSWERING IN EPISTEMIC KNOWLEDGE BASES

*Based on a paper titled "*PSPACE *Tableau Algorithms for Acyclic Modalized $\mathcal{ALC}$"*

published in Journal of Automated Reasoning 2011

Jia Tao, Giora Slutzki and Vasant Honavar

Abstract

We study $\mathcal{ALCK}_m$, which extends the description logic $\mathcal{ALC}$ by adding modal operators of the basic multi-modal logic $\mathbf{K}_m$. We develop a sound and complete tableau algorithm $\Lambda_{\mathcal{K}}$ for answering $\mathcal{ALCK}_m$ queries w.r.t. an $\mathcal{ALCK}_m$ knowledge base with an acyclic TBox. Defining tableau expansion rules in the presence of acyclic definitions by considering only the concept names on the left-hand side of TBox definitions or their negations allows us to give a PSPACE implementation for $\Lambda_{\mathcal{K}}$. We next consider answering $\mathcal{ALCK}_m$ queries w.r.t. an $\mathcal{ALCK}_m$ knowledge base in which the epistemic operators correspond to those of classical multi-modal logic $\mathbf{S4}_m$. The expansion rules in the tableau algorithm Λ_{S4} are designed to syntactically incorporate the epistemic properties. We also provide a PSPACE implementation for Λ_{S4}. In light of the fact that the satisfiability problem for $\mathcal{ALCK}_m$ with general TBox and no epistemic properties (i.e., $\mathbf{K}_{\mathcal{ALC}}$) is NEXPTIME-complete, we conclude that $\mathcal{ALCK}_m$ offers computationally manageable and practically useful fragment of $\mathbf{K}_{\mathcal{ALC}}$.

2.1 Introduction

Description Logics (DLs) [1] offer a powerful formalism for representing and reasoning with knowledge in a broad range of applications. Many DLs have been investigated with respect to their expressivity and complexity [2, 3, 4, 5]. Some DLs provide the foundation for powerful

practical languages to represent knowledge on the web, e.g., DAML+OIL [6], OWL DL, OWL Lite [7], and reasoners (typically based on the analytic tableau method [4]) can be used to draw inferences from these DL knowledge bases [7]. Because of its inferential feasibility and practical utility, the *terminological* knowledge representation language $\mathcal{ALC}$ [2] is of particular interest. Representing knowledge in such a system amounts to introducing the *terminology* of the application domain through *definitions* of the relevant concepts, and *assertions* that hold with respect to specific *individuals* in the domain. However, terminological knowledge representation languages such as $\mathcal{ALC}$ lack the expressivity needed to represent *modal* or *epistemic* aspects of knowledge. Thus, in a pure terminological system, we can say that 'swine flu is a life threatening disease' but not that 'Dr. Vos *knows that* swine flu is a life threatening disease'. Epistemic DLs allow us to address this limitation by providing a means to model as well as reason about the knowledge of different experts using epistemic operators. The resulting logic finds applications in settings where it is useful to be able to attribute specific pieces of knowledge to individual experts.

Motivated by such applications, there is growing interest in incorporating some features of *epistemic modal logics* [32, 33, 34] into DLs [35, 36, 37, 38, 39, 40]. In general, in DLs augmented with modal operators the interaction between modalities and DL constructs can substantially increase the complexity of reasoning and in some cases, even lead to undecidability [41, 42, 43]. In a series of papers, Wolter and Zakharyaschev [44, 45, 46, 47] showed various decidability results for the satisfiability problem for logics that augment DLs by modal operators. These papers delineate some *syntactical* and *semantical limits* within which DLs augmented with modal operators remain decidable; this line of research was summarized in [40].

There are also papers that provide decision procedures for languages that augment $\mathcal{ALC}$ with modal operators. For example, Donini et al. [37, 38] investigated the addition of an *epistemic operator* to an $\mathcal{ALC}$-based query language and showed that this allows treatment of several features of standard databases such as closed-world reasoning and integrity constraints. The language is further extended by adding the autoepistemic operator **A** [48] such that the resulting language combines the non-first-order features of frame-based systems with default reasoning. Baader and Laux [39] extended $\mathcal{ALC}$ by adding *multi-modal* operators which can

be used both inside concept expressions and in front of assertional (ABox) and terminological (TBox) axioms but not in front of roles. The modal operators in the resulting language (later named $\mathbf{K}_{\mathcal{ALC}}$ in [40]), are interpreted in the classic multi-modal logic K_m. By extending the tableau expansion rules for $\mathcal{ALC}$ to incorporate *accessibility relation* between worlds, they showed that the satisfiability of finite sets of formulae in $\mathbf{K}_{\mathcal{ALC}}$ is decidable under the *increasing domain* assumption (i.e., if a world w' is accessible from a world w, then the domain of w is a subset of the domain of w'). They further showed that their tableau algorithm for $\mathbf{K}_{\mathcal{ALC}}$ is not adequate under the *constant domain assumption* (a.k.a. *common domain assumption* in [32]) where all worlds share the same interpretation domain. It has been shown in [44] that the satisfiability problem w.r.t. models with increasing domains can be reduced to that w.r.t. models with constant domains. Hence, the treatment in this chapter is based on the constant domain assumption.

Lutz et al. [40] assumed a constant domain and a *global* interpretation for all individuals (i.e., all individuals are interpreted identically in all worlds) and provided a tableau decision algorithm for the $\mathbf{K}_{\mathcal{ALC}}$ satisfiability problem. They observed that although infinitely many individuals may be needed to construct a model for a satisfiable $\mathbf{K}_{\mathcal{ALC}}$ formula, only finitely many concepts are involved. Based on this observation, they designed a tableau algorithm that constructs a *quasimodel* wherein each object represents a *type* of individuals (i.e., a set of concepts they belong to) rather than the individuals themselves. The complexity of the resulting tableau algorithm is NEXPTIME which is consistent with the known result that the satisfiability problem for $\mathbf{K}_{\mathcal{ALC}}$ is NEXPTIME-complete [42]. In contrast, the satisfiability problem for $\mathcal{ALC}$ is known to be PSPACE-complete [2, 49]. Hence, it is of interest to explore computationally manageable, yet practically useful fragments of $\mathbf{K}_{\mathcal{ALC}}$. We investigate a subset of $\mathbf{K}_{\mathcal{ALC}}$ obtained by augmenting $\mathcal{ALC}$ with an acyclic TBox with modal operators that can appear in front of any concept expressions, yielding a language which we refer to as $\mathcal{ALCK}_m$. As in the case of $\mathbf{K}_{\mathcal{ALC}}$, $\mathcal{ALCK}_m$ conforms to the constant domain assumption. We provide a sound and complete tableau algorithm for $\mathcal{ALCK}_m$ with an acyclic TBox.

As in the case of DL knowledge bases (see [50]), given an $\mathcal{ALCK}_m$ *knowledge base* (KB) Σ, the following problems are of interest: (1) *KB-satisfiability*: Σ is satisfiable if it has a model;

(2) *Concept satisfiability*: a concept C is satisfiable w.r.t. Σ if there exist a model of Σ in which the interpretation of C is not empty; (3) *Subsumption*: a concept C is subsumed by a concept D w.r.t. Σ if for every model of Σ the interpretation of C is a subset of the interpretation of D; (4) *Instance checking*: a is an instance of C if the assertion $C(a)$ is satisfied in every model of Σ. Instance checking problem can be viewed as a query answering problem. It is well-known that problems (2)-(4) can be reduced to KB-satisfiability in linear time [50]. We solve the query answering problem (whether the KB entails the query) by reducing it to the KB-satisfiability problem.

The main contribution of this chapter is two PSPACE implementations for the satisfiability of an $\mathcal{ALCK}_m$ query with respect to an $\mathcal{ALCK}_m$ knowledge base and the satisfiability of an $\mathcal{ALCS}4_m$ query with respect to an $\mathcal{ALCS}4_m$ knowledge base. This extends the result of Schmidt-Schauß and Smolka [2] that checking satisfiability and subsumption of $\mathcal{ALC}$ concepts can be decided in linear space. Hladik and Peñaloza [51] used automata-theoretic approach to reprove the result that the $\mathcal{ALC}$ concept satisfiability w.r.t. acyclic TBoxes is decidable in PSPACE. Our solution takes advantage of:

1. Tableau expansion rules that can cope with acyclic definitions by considering only the left-hand sides of TBox definitions or their negations. This approach allows us to detect potential clashes and facilitates PSPACE implementation by eliminating the need for backtracking.

2. An extension of the idea of canonical interpretation [52, 50] that incorporates the TBox definitions into the interpretation of concept names.

3. A blocking technique that facilitates the termination of the algorithm in the case of $\mathcal{ALCS}4_m$.

In what follows, we introduce the syntax and semantics of $\mathcal{ALCK}_m$ as well as the framework of query answering problem in Section 2.2. We proceed to develop a sound and complete algorithm for $\mathcal{ALCK}_m$ KB-satisfiability with an acyclic TBox in Section 2.3, and then provide the solution to the query answering problem in Section 2.4. Section 2.5 shows a PSPACE

implementation for $\mathcal{ALCK}_m$ KB-satisfiability. Section 2.6 develops a sound and complete algorithm for $\mathcal{ALCS}4_m$ KB-satisfiability w.r.t. the class of $S4$-models and provides a PSPACE implementation for the algorithm. Section 2.7 summarize the chapter.

2.2 Preliminaries

2.2.1 The Syntax and Semantics

The non-logical signature of the $\mathcal{ALCK}_m$ language includes four mutually disjoint sets: a set of *concept names* $N_{\mathcal{C}}$, a set of *role names* $N_{\mathcal{R}}$, a set of *individual names* $N_{\mathcal{O}}$, all of which are countably infinite and a finite set of *experts* $N_{\mathcal{E}} = \{1, \ldots, m\}$. When we write $\Box_i$ or $\Diamond_i$, the subscript i refers to an expert $i \in N_{\mathcal{E}}$. The syntax of $\mathcal{ALCK}_m$ is defined by specifying $\mathcal{ALCK}_m$ *expressions* $\mathfrak{E}$ and $\mathcal{ALCK}_m$ *formulae* $\mathfrak{F}$. $\mathfrak{E}$ contains the set of *roles names* $N_{\mathcal{R}}$ and a set of *concepts* $\mathcal{C}$ which is recursively defined as follows:

$$C, D \longrightarrow A \mid \top \mid \bot \mid \neg C \mid C \sqcap D \mid C \sqcup D \mid \forall R.C \mid \exists R.C \mid \Diamond_i C \mid \Box_i C$$

where $A \in N_{\mathcal{C}}$, $\top$ is the *top symbol*, $\bot$ is the *bottom symbol*, $C, D \in \mathcal{C}$, $R \in N_{\mathcal{R}}$, $i \in N_{\mathcal{E}}$ and $\Diamond_i C$ is an abbreviation of $\neg\Box_i\neg C$.

In this paper we will consider restricted $\mathcal{ALCK}_m$ formulae $\mathfrak{F}$ of two kinds: the *assertional formulae* of the form $C(a)$ or $R(a, b)$ and the *definitional formulae* of the form $A \doteq C$, where $a, b \in N_{\mathcal{O}}$, $C \in \mathcal{C}$, $R \in N_{\mathcal{R}}$ and $A \in N_{\mathcal{C}}$.

A concept is said to be in *negation normal form* (NNF) if negation occurs only in front of concept names. It is well-known that any concept can be rewritten into an equivalent negation normal form in linear time [2].

The semantics of $\mathcal{ALCK}_m$ language is defined by using Kripke structures [32]. A *relational Kripke structure* for m experts is a tuple $\mathbb{M} = \langle S, \pi, \mathcal{E}_1, ..., \mathcal{E}_m \rangle$ where S is a set of *states*, $\mathcal{E}_i \subseteq S \times S$ are the *accessibility* relations, and π interprets the syntax of $\mathcal{ALCK}_m$, both the expressions in $\mathfrak{E}$ and the formulae in $\mathfrak{F}$ for each state $s \in S$. A *(Kripke) world* is a pair $w = (\mathbb{M}, s)$ where $\mathbb{M}$ is a Kripke structure and s is a state in S. The intuitive interpretation of $(s, t) \in \mathcal{E}_i$ is that in world $(\mathbb{M}, s)$ expert i considers world $(\mathbb{M}, t)$ as a *possible world*. We may further use $\mathcal{E}_i(s)$ to denote the set $\{t \mid (s, t) \in \mathcal{E}_i\}$ of the i-successors of the state s.

For a finite set of symbols $N \subset N_{\mathcal{C}} \cup N_{\mathcal{R}} \cup N_{\mathcal{O}}$, we define a *Kripke structure* $\mathbb{M} = \langle S, \pi, \mathcal{E}_1, ..., \mathcal{E}_m \rangle$ *restricted to* N to be $\mathbb{M}|_N = \langle S, \pi|_N, \mathcal{E}_1, ..., \mathcal{E}_m \rangle$ where $\pi|_N$ denotes the restriction of the function π to N.

All the concepts and roles will be interpreted in a *common (i.e., state-independent) non-empty domain* which we denote by Δ. We do not make the Unique Name Assumption, i.e., distinct individual names can be interpreted identically. The interpretation of concept and role expressions is defined recursively as follows: for all $a \in N_{\mathcal{O}}, A \in N_{\mathcal{C}}, R \in N_{\mathcal{R}}, C \in \mathcal{C}$,

$$
\begin{array}{rclcrcl}
\top^{\pi(s)} & = & \Delta & & (C \sqcup D)^{\pi(s)} & = & C^{\pi(s)} \cup D^{\pi(s)} \\
\bot^{\pi(s)} & = & \emptyset & & (C \sqcap D)^{\pi(s)} & = & C^{\pi(s)} \cap D^{\pi(s)} \\
a^{\pi(s)} & \in & \Delta, & & (\Box_i C)^{\pi(s)} & = & \bigcap_{t \in \mathcal{E}_i(s)} C^{\pi(t)} \\
A^{\pi(s)} & \subseteq & \Delta, & & (\Diamond_i C)^{\pi(s)} & = & \bigcup_{t \in \mathcal{E}_i(s)} C^{\pi(t)} \\
R^{\pi(s)} & \subseteq & \Delta \times \Delta, & & (\neg C)^{\pi(s)} & = & \Delta \setminus C^{\pi(s)} \\
(\forall R.C)^{\pi(s)} & = & \{a \in \Delta \mid \forall b : (a,b) \in R^{\pi(s)} \rightarrow b \in C^{\pi(s)}\} & & & & \\
(\exists R.C)^{\pi(s)} & = & \{a \in \Delta \mid \exists b : (a,b) \in R^{\pi(s)} \wedge b \in C^{\pi(s)}\} & & & &
\end{array}
$$

Definition 2.2.1 *Let C be a concept, $C(a)$ and $R(a,b)$ assertional formulae, and $A \doteq C$ a definitional formula. We define the satisfiability relation as follows:*

$$
\begin{array}{rclcrcl}
(\mathbb{M}, s) \vDash C & \Leftrightarrow & C^{\pi(s)} \neq \emptyset & & (\mathbb{M}, s) \vDash R(a,b) & \Leftrightarrow & (a^{\pi(s)}, b^{\pi(s)}) \in R^{\pi(s)} \\
(\mathbb{M}, s) \vDash C(a) & \Leftrightarrow & a^{\pi(s)} \in C^{\pi(s)} & & (\mathbb{M}, s) \vDash A \doteq C & \Leftrightarrow & A^{\pi(s)} = C^{\pi(s)}
\end{array}
$$

Let φ be a formula (assertional or definitional). Then (i) φ is *satisfiable* if there is a world $w = (\mathbb{M}, s)$ such that $w \vDash \varphi$; (ii) φ is *valid* in a Kripke structure $\mathbb{M} = \langle S, \pi, \mathcal{E}_1, ..., \mathcal{E}_m \rangle$, written as $\mathbb{M} \vDash \varphi$, if $(\mathbb{M}, s) \vDash \varphi$ for all $s \in S$; (iii) φ is *valid*, written as $\vDash \varphi$, if $\mathbb{M} \vDash \varphi$ for all $\mathbb{M}$.

2.2.2 Knowledge Bases and Query Answering

A finite non-empty set of assertional formulae whose concepts and roles belong to the language $\mathcal{ALCK}_m$ is called an *ABox*. A finite set $\mathcal{T}$ of definitional formulae is called a TBox. A concept name A *directly refers* to a concept name B w.r.t. $\mathcal{T}$ if there is a definition $A \doteq C \in \mathcal{T}$ and B occurs in C. Let *refers* be the transitive closure of *directly refers*. Then $\mathcal{T}$ is said to be

acyclic if no concept name refers to itself. In this paper, a *TBox* is assumed to be acyclic such that no defined concept (l.h.s. of a definitional formula) has more than one definition (r.h.s. of a definitional formula). An ABox $\mathcal{A}$ and a TBox $\mathcal{T}$ together form an $\mathcal{ALCK}_m$-knowledge base $\Sigma = \langle \mathcal{A}, \mathcal{T} \rangle$. Note that all the KBs in this paper will be $\mathcal{ALCK}_m$-knowledge bases unless specified otherwise. A knowledge base $\Sigma = \langle \mathcal{A}, \mathcal{T} \rangle$ is called *acyclic* if $\mathcal{T}$ is acyclic. Our *query language* is the set of all assertional formulae over the alphabet of the given knowledge base.

Definition 2.2.2 *A world* $w = (\mathbb{M}, s)$ satisfies *a knowledge base* $\Sigma = \langle \mathcal{A}, \mathcal{T} \rangle$*, written as* $w \vDash \Sigma$*, if* w *satisfies all the assertions in* $\mathcal{A}$ *and all the definitions in* $\mathcal{T}$*. A knowledge base* Σ entails *an assertion* $C(a)$*, written as* $\Sigma \vDash C(a)$*, if for all worlds* w*,* $w \vDash \Sigma \Rightarrow w \vDash C(a)$*.*

In this paper, our motivation is to answer queries of the form $C(a)$ or $R(a, b)$, i.e., whether a is a member of the concept C, or whether (a, b) is a member of the role R. Given a KB Σ, a concept $C \in \mathcal{C}$, and an individual $a \in N_{\mathcal{O}}$, the answer to the query $C(a)$ posed to Σ, is based on the *open world assumption* (OWA) and it is defined as

- YES, if $\Sigma \vDash C(a)$,
- NO, if $\Sigma \vDash \neg C(a)$,
- UNKNOWN, otherwise.

Clearly, given $\Sigma = \langle \mathcal{A}, \mathcal{T} \rangle$, answering the query $C(a)$ is equivalent to checking the non-satisfiability of $\langle \mathcal{A} \cup \{\neg C(a)\}, \mathcal{T} \rangle$ in the following sense. If $\langle \mathcal{A} \cup \{\neg C(a)\}, \mathcal{T} \rangle$ is not satisfiable, the answer to the query is YES. Otherwise, if $\langle \mathcal{A} \cup \{C(a)\}, \mathcal{T} \rangle$ is not satisfiable, then the answer to the query is NO; and if both are satisfiable, the answer to the query will be UNKNOWN.

The query answering framework contains the following components:

- A knowledge base $\Sigma = \langle \mathcal{A}, \mathcal{T} \rangle$.
- Σ includes epistemic statements that contain knowledge of the *experts* expressed using modal operators.

- A *reasoner* that knows every assertion and definition in Σ. In response to a query, it computes answers such as "YES", "NO", or "UNKNOWN" from the information present in Σ and returns the answer to the querying agent.

- A *querying agent* that poses queries of the form $C(a)$ or $R(a, b)$ to Σ. We assume that the querying agent does know the language, $N_{\mathcal{C}}, N_{\mathcal{R}}, N_{\mathcal{O}}, N_{\mathcal{E}}$ as well as the syntax of the language. In particular, the querying agent can ask queries that involve knowledge operators.

In the following example we consider a knowledge base with an ABox and an acyclic TBox with exactly one operator on the right-hand side of each definition.

Example 2.2.3 *Consider the following knowledge base* $\Sigma_1 = \langle \mathcal{A}, \mathcal{T} \rangle$ *where*

$\mathcal{A}$ = { *ADVISE(john, mary), TEACHES(susan, cs525),* $\Diamond_1$*Advisor(susan),*
$\Diamond_2$*Grad(mary),* $\Box_2$*Lecturer(susan), Advisor(john),* $\neg$*BasicCourse(cs525)*}

$\mathcal{T}$ = { *Lecturer* $\doteq \forall$*TEACHES.BasicCourse, Advisor* $\doteq$ *Professor* $\sqcap$ *A,*
A $\doteq \exists$*ADVISE.Grad* }.

Consider the following queries:

Q1: Is john a professor?

Query: Professor(john); Answer: YES.

Q2: Is susan a lecturer?

Query: Lecturer(susan); Answer: NO.

Q3: Is there an Expert 1's successor world where peter is a graduate student?

Query: $\Diamond_1$*Grad(peter); Answer: UNKNOWN.*

Q4: In all Expert 2's successor worlds, is it true that all courses that susan teaches are basic courses?

Query: $\Box_2(\forall$*TEACHES.BasicCourse)(susan); Answer: YES.*

The answer to Q1 is explained by the assertion Advisor(john) and the definition Advisor $\doteq$ *Professor* $\sqcap$ *A. The answer to Q2 comes from the assertions TEACHES(susan, cs525),*

$\neg$BasicCourse(cs525) and the definition Lecturer $\doteq \forall$TEACHES.BasicCourse. To answer Q3, observe that there is an Expert 1's world where Advisor(susan) is true. However, under the OWA, whether there is an Expert 1's world where peter is a graduate student is UNKNOWN. In answering Q4, for any Expert 2's successor world (and there is one in view of $\Diamond_2$Grad(mary)), Lecturer(susan) is true. Since the definition Lecturer $\doteq \forall$TEACHES.BasicCourse is satisfied in any such world, The answer to $\Box_2(\forall$TEACHES.BasicCourse)(susan) is YES. ■

2.3 Tableau Algorithm for $\mathcal{ALCK}_m$

As discussed in Section 2.2.2, answering queries against a knowledge base can be reduced to the problem of checking existence of models. Tableau algorithms are generally used to construct models. Such a model, usually built by using a data structure called a *constraint system* [37, 39, 38, 40], contains a set of constraints and it is constructed by recursively applying *expansion rules*.

In the presence of modal operators, we need to construct a model which eventually is equivalent to a Kripke structure. Intuitively, one world corresponds to one constraint system, and the accessibility relations connect one constraint system to another. Let $\Sigma = \langle \mathcal{A}, \mathcal{T} \rangle$ be a knowledge base. We define the concept of a *constraint graph* by generalizing the idea of a *completion tree* in [53], and build it starting from a single node representing the constraint system obtained from $\mathcal{A}$ and an input query and repeatedly applying expansion rules. The constraints in constraint systems are of the form $a : C$ or $(a, b) : R$, where $a, b \in N_{\mathcal{O}}$, $C \in \mathcal{C}$ and $R \in N_{\mathcal{R}}$. Each assertion $D(a)$ in $\mathcal{A}$ is rewritten into a constraint $a : D'$ where D' is the NNF of D; each $R(a, b)$ in $\mathcal{A}$ is rewritten into a constraint $(a, b) : R$.

Formally, a *constraint graph* [1] is a directed graph $\mathbb{G} = \langle \mathbb{V}, \mathbb{E}, \mathbb{L} \rangle$ where $\mathbb{V}$ is a set of nodes, $\mathbb{E}$ is a set of directed edges and $\mathbb{L}$ is a function that labels each node n with a constraint system and each edge (n, n') in $\mathbb{E}$ with a nonempty subset of $N_{\mathcal{E}}$. If $i \in \mathbb{L}(n, n')$, then n' is an *i-successor* of n, i.e., it is directly accessible from node n by expert i. We denote by $\mathcal{O}_{\mathbb{G}}$ (a subset of $N_{\mathcal{O}}$) the set of all individual names that occur in $\mathbb{G}$. A node $n \in \mathbb{V}$ is said to be

[1] We use constraint graphs, rather than trees, with an eye towards an application to the case of $S4$-structures in which the accessibility relations are reflexive and transitive.

closed if $\mathbb{L}(n)$ contains a *clash*, i.e., $\{a : C, a : \neg C\} \subseteq \mathbb{L}(n)$ or $\{a : \bot\} \subseteq \mathbb{L}(n)$. $\mathbb{G}$ is said to be *closed* if at least one of its nodes is closed. A constraint graph that is not closed is *open*, and it is *complete* if no expansion rule applies.

There are three types of expansion rules: *local* expansion rules which generate new constraints within one constraint system, *global* expansion rules which can add new assertions to constraint systems associated with nodes that are directly accessible from the current node and *terminological expansion rules* which take into consideration both the constraints in the constraint systems and the given set of terminological definitions $\mathcal{T}$. Note that the syntactic construct $\exists R.C$ encodes incomplete information. For example, $\exists$ADVISE.Grad(susan) says that the individual susan advises a graduate student. However, who is this graduate student is left unspecified. Under the OWA and without the Unique Name Assumption, to find a model for the knowledge base containing this kind of assertions, it is sufficient to use a new individual name that has not yet appeared in the constraint graph to denote this unknown person. If using a new individual name causes a clash, then, a fortiori, using any existing individual name will also cause a clash.

We denote by N_Σ ($\mathcal{O}_\Sigma$) the set of all the symbols (individual names) appearing in the knowledge base Σ. Initially, the constraint graph $\mathbb{G}$ contains only the individual names occurring in Σ, i.e., $\mathcal{O}_\mathbb{G} = \mathcal{O}_\Sigma$. With the application of expansion rules, new individual names may be added to $\mathcal{O}_\mathbb{G}$. An individual name is called *fresh* (at any particular time) if it belongs to $N_\mathcal{O} \setminus \mathcal{O}_\mathbb{G}$ (at that time). The local and global expansion rules are listed in Figure 2.1.

We assume that the TBox $\mathcal{T}$ is in *simple form* where the right-hand side of each definition contains exactly one operator, i.e., the right-hand side of each definition is of the form $\neg C, C \sqcap D, C \sqcup D, \exists R.C, \forall R.C, \Diamond_i C$ or $\Box_i C$ where $C, D \in N_\mathcal{C}$ and $R \in N_\mathcal{R}$; moreover, if the right-hand side is of the form $\neg A$, then A does not appear on the left-hand side of any definition in $\mathcal{T}$ (see [54], Definition 6). It can be shown that transforming a given TBox to an equivalent simple form can be done in linear time. The proof is similar to that of Lemma 1 in [54].

Nebel has shown that the straightforward unfolding of an ABox w.r.t. a TBox may lead to an exponential blowup [55]. To give a PSPACE complexity result for reasoning $\mathcal{ALC}$ with acyclic TBoxes, instead of unfolding iteratively as in [55], the approach in [54] ensures that if an

Local Expansion Rules:

$\sqcap$-rule: If there is a node n with $a : C_1 \sqcap C_2 \in \mathbb{L}(n)$,
and $\{a : C_1, a : C_2\} \not\subseteq \mathbb{L}(n)$,
then $\mathbb{L}(n) := \mathbb{L}(n) \cup \{a : C_1, a : C_2\}$;

$\sqcup$-rule: If there is a node n with $a : C_1 \sqcup C_2 \in \mathbb{L}(n)$
and $\{a : C_1, a : C_2\} \cap \mathbb{L}(n) = \emptyset$,
then $\mathbb{L}(n) := \mathbb{L}(n) \cup \{a : C_i\}$ for some $i \in \{1, 2\}$;

$\exists$-rule: If there is a node n with $a : \exists R.C \in \mathbb{L}(n)$,
and there is no $b \in \mathcal{O}_{\mathbb{G}}$ s.t. $\{(a, b) : R, b : C\} \subseteq \mathbb{L}(n)$,
then $\mathbb{L}(n) := \mathbb{L}(n) \cup \{(a, c) : R, c : C\}$ where c is fresh;

$\forall$-rule: If there is a node n with $\{a : \forall R.C, (a, b) : R\} \subseteq \mathbb{L}(n)$,
and $b : C \notin \mathbb{L}(n)$,
then $\mathbb{L}(n) := \mathbb{L}(n) \cup \{b : C\}$;

Global Expansion Rules:

$\Diamond$-rule: If there is a node n with $a : \Diamond_i C \in \mathbb{L}(n)$,
and n has no i-successor l with $a : C \in \mathbb{L}(l)$,
then add a new i-successor n' of n with $\mathbb{L}(n') := \{a : C\}$;

$\Box$-rule: If there is a node n with $a : \Box_i C \in \mathbb{L}(n)$,
and n has an i-successor n' with $a : C \notin \mathbb{L}(n')$,
then $\mathbb{L}(n') := \mathbb{L}(n') \cup \{a : C\}$.

Figure 2.1 The local and global expansion rules for $\mathcal{ALCK}_m$

assertion $a : C$ is in the ABox and a definition $C \doteq D$ is in the TBox, then the assertion $a : D$ is added to the ABox. However, in the case when $C \doteq D_1 \sqcap D_2 \in \mathcal{T}$ and $\{a : D_1, a : D_2, a : \neg C\}$ is a subset of a constraint system, such an approach may not detect the implicit clash. The terminological expansion rules given in Figure 2.2 deal with this issue.

We denote by $\Lambda_{\mathcal{K}}$ the *$\mathcal{K}$-tableau algorithm* which nondeterministically applies the local, global and terminological expansion rules until no further applications are possible. We note again, following up on footnote 1, that the graph-structure constructed by $\Lambda_{\mathcal{K}}$ is actually a tree, referred to as a *constraint tree*. It is also easily seen that in a constraint tree the edge labels are singletons. The following lemma is easy to prove.

Lemma 2.3.1 *All executions of $\Lambda_{\mathcal{K}}$ on an input consisting of a knowledge base and a query terminate.*

The next definition provides a formal interpretation of a constraint graph.

Terminological Expansion Rules:

T-rule: If there is a node n with $a : A \in \mathbb{L}(n), A \doteq D \in \mathcal{T}$, and $a : D \notin \mathbb{L}(n)$
then $\mathbb{L}(n) := \mathbb{L}(n) \cup \{a : D\}$.

N-rule: If there is a node n with $\{a : \neg A, a : B\} \cap \mathbb{L}(n) \neq \emptyset, A \doteq \neg B \in \mathcal{T}$,
and $\{a : \neg A, a : B\} \not\subseteq \mathbb{L}(n)$,
then $\mathbb{L}(n) := \mathbb{L}(n) \cup \{a : \neg A, a : B\}$;

N $\sqcap$-rule: If there is a node n with $a : \neg A \in \mathbb{L}(n), A \doteq B_1 \sqcap B_2 \in \mathcal{T}$,
and $a : \neg B_1 \sqcup \neg B_2 \notin \mathbb{L}(n)$,
then $\mathbb{L}(n) := \mathbb{L}(n) \cup \{a : \neg B_1 \sqcup \neg B_2\}$;

N $\sqcup$-rule: If there is a node n with $a : \neg A \in \mathbb{L}(n), A \doteq B_1 \sqcup B_2 \in \mathcal{T}$,
and $a : \neg B_1 \sqcap \neg B_2 \notin \mathbb{L}(n)$,
then $\mathbb{L}(n) := \mathbb{L}(n) \cup \{a : \neg B_1 \sqcap \neg B_2\}$;

N$\exists$-rule: If there is a node n with $a : \neg A \in \mathbb{L}(n), A \doteq \exists P.B \in \mathcal{T}$,
and $a : \forall P.(\neg B) \notin \mathbb{L}(n)$,
then $\mathbb{L}(n) := \mathbb{L}(n) \cup \{a : \forall P.(\neg B)\}$;

N$\forall$-rule: If there is a node n such that $a : \neg A \in \mathbb{L}(n), A \doteq \forall P.B \in \mathcal{T}$,
and $a : \exists P.(\neg B) \notin \mathbb{L}(n)$,
then $\mathbb{L}(n) := \mathbb{L}(n) \cup \{a : \exists P.(\neg B)\}$;

N$\Diamond$-rule: If there is a node n with $a : \neg A \in \mathbb{L}(n), A \doteq \Diamond_i B \in \mathcal{T}$,
and $a : \Box_i \neg B \notin \mathbb{L}(n)$,
then $\mathbb{L}(n) := \mathbb{L}(n) \cup \{a : \Box_i \neg B\}$.

N$\Box$-rule: If there is a node n with $a : \neg A \in \mathbb{L}(n), A \doteq \Box_i B \in \mathcal{T}$,
and $a : \Diamond_i \neg B \notin \mathbb{L}(n)$,
then $\mathbb{L}(n) := \mathbb{L}(n) \cup \{a : \Diamond_i \neg B\}$.

Figure 2.2 The terminological expansion rules for $\mathcal{ALCK}_m$

Definition 2.3.2 *Let* $\mathbb{G} = \langle \mathbb{V}, \mathbb{E}, \mathbb{L} \rangle$ *be a constraint graph,* $\mathbb{M} = \langle S, \pi, \mathcal{E}_1, ..., \mathcal{E}_m \rangle$ *a Kripke structure, and* σ *a mapping from* $\mathbb{V}$ *to* S. *Then* $\mathbb{M}$ satisfies $\mathbb{G}$ via σ *if, for all* $n, n' \in \mathbb{V}$,

- $i \in \mathbb{L}(n, n') \Longrightarrow \mathcal{E}_i(\sigma(n), \sigma(n'))$
- $a : C \in \mathbb{L}(n) \Longrightarrow (\mathbb{M}, \sigma(n)) \models C(a)$
- $(a, b) : R \in \mathbb{L}(n) \Longrightarrow (\mathbb{M}, \sigma(n)) \models R(a, b)$

We say that $\mathbb{M}$ satisfies $\mathbb{G}$, *denoted as* $\mathbb{M} \Vdash \mathbb{G}$, *if there is a mapping* σ *such that* $\mathbb{M}$ *satisfies* $\mathbb{G}$ *via* σ. *In this case, we also say that* $\mathbb{M}$ *is a model of* $\mathbb{G}$. *Note that* $\mathbb{M} \Vdash \mathbb{G}$ *implies that* $\mathbb{G}$ *is open.*

The idea behind Definition 2.3.2 is that each constraint system is mapped to a state of

$\mathbb{M}$ in which all its constraints are satisfied. Moreover, labeled edges in $\mathbb{G}$ are mapped to the corresponding accessibility relations.

Let $\mathbb{M} = \langle S, \pi, \mathcal{E}_1, ..., \mathcal{E}_m \rangle$ and $\mathbb{M}' = \langle S, \pi', \mathcal{E}_1, ..., \mathcal{E}_m \rangle$ be two Kripke structures, and $N_2 \subseteq N_1$ be finite subsets of $N_\mathcal{C} \cup N_\mathcal{R} \cup N_\mathcal{O}$ such that $N_1 \setminus N_2 \subseteq N_\mathcal{O}$. Then $\mathbb{M}'|_{N_1} = \langle S, \pi'|_{N_1}, \mathcal{E}_1, ..., \mathcal{E}_m \rangle$ is a *semantic extension* of $\mathbb{M}|_{N_2} = \langle S, \pi|_{N_2}, \mathcal{E}_1, ..., \mathcal{E}_m \rangle$ if $(\mathbb{M}'|_{N_1})|_{N_2} = \mathbb{M}|_{N_2}$. The following theorem shows that if a constraint graph has a model, then the constraint graph resulting from the application of any expansion rule also has a model which is a semantic extension of the original model.

Theorem 2.3.3 *(Soundness of the expansion rules) Given a Kripke structure* $\mathbb{M} = \langle S, \pi, \mathcal{E}_1, ..., \mathcal{E}_m \rangle$ *and an acyclic TBox* $\mathcal{T}$ *where* $\mathbb{M} \models \mathcal{T}$*, let* $\mathbb{G}$ *be a constraint graph,* α *a local, global or terminological expansion rule and* $\mathbb{G}_\alpha$ *a constraint graph obtained by applying* α *to* $\mathbb{G}$*. If* $\mathbb{M} \Vdash \mathbb{G}$ *via* σ*, then there exists a semantic extension* $\mathbb{M}_\alpha$ *of* $\mathbb{M}|_{N_\Sigma \cup \mathcal{O}_\mathbb{G}}$ *s.t.* $\mathbb{M}_\alpha \Vdash \mathbb{G}_\alpha$ *via* σ' *(which extends* σ*) and* $\mathbb{M}_\alpha \models \mathcal{T}$*. Furthermore,* $\mathbb{M}_\alpha \Vdash \mathbb{G}$*.*

Theorem 2.3.3 (proof is given in Appendix A.1.1) ensures that applications of expansion rules preserve the existence of models. Unfortunately, it does not specify how to construct such models in the first place. The *canonical interpretation* of a constraint system has been defined in [52, 50]. In [52], no TBox is involved, and the canonical interpretation is defined to be a model for a constraint system that originates from an ABox of an $\mathcal{ALCN}$ knowledge base. The approach in [50] incorporates the subsumptions in the TBox (not necessarily acyclic) into the initial constraint system and then applies expansion rules. A subsumption, $C \sqsubseteq D$, is converted into a constraint $\forall x.x : \neg C \sqcup D$ in which, during the process of expansion, the variable x is substituted by all possible individual names in the constraint system. The resulting algorithm for $\mathcal{ALCNR}$ is in NEXPTIME. In contrast, our tableau algorithm for $\mathcal{ALCK}_m$ incorporates the TBox (in our case, acyclic) into the terminological expansion rules. This is reflected in the following definition of a *canonical Kripke structure* for a constraint graph which takes the TBox into account. It thereby ensures that the TBox is valid in the canonical Kripke structure for an open constraint graph that is complete w.r.t. local, global and terminological expansion rules.

Definition 2.3.4 *Let $\mathbb{G} = \langle \mathbb{V}, \mathbb{E}, \mathbb{L} \rangle$ be a constraint graph and $\mathcal{T}$ a simple acyclic TBox. Let Θ be the set of all the concept names in either $\mathbb{G}$ or $\mathcal{T}$ that do not occur on the left-hand side of any definition in $\mathcal{T}$. The canonical Kripke structure $\mathbb{M}_{\mathbb{G}} = \langle S, \pi, \mathcal{E}_1, ..., \mathcal{E}_m \rangle$ for $\mathbb{G}$ w.r.t. $\mathcal{T}$ is defined as follows.*

- $S := \mathbb{V}$,
- $\mathcal{E}_i := \{e \in \mathbb{E} \mid i \in \mathbb{L}(e)\}$, $1 \leq i \leq m$,
- $\Delta := \mathcal{O}_{\mathbb{G}}$,
- $a^{\pi(n)} := a$ *for all* $a \in \mathcal{O}_{\mathbb{G}}$,
- $R^{\pi(n)} := \{(a, b) \mid (a, b) : R \in \mathbb{L}(n)\}$,
- $A^{\pi(n)} := \{a \mid a : A \in \mathbb{L}(n)\}$, *if* $A \in \Theta$,
- $A^{\pi(n)} := \{a \mid a : A \in \mathbb{L}(n)\} \cup D^{\pi(n)}$, *if* $A \notin \Theta$ *and* $A \doteq D \in \mathcal{T}$.

Let $\mathcal{T}$ be a given TBox and let $\mathbb{G}$ be a constraint graph that is complete w.r.t. local, global and terminological expansion rules. We next prove that $\mathbb{G}$ is open if and only if it has a model. This shows the soundness and completeness of the $\mathcal{K}$-tableau algorithm. Before proving it, we state an auxiliary lemma that specifically deals with negation (proof is given in Appendix A.1.2).

Lemma 2.3.5 *Let $\mathcal{T}$ be an acyclic TBox and let $\mathbb{G}$ be an open complete constraint graph w.r.t. local, global and terminological expansion rules. Then for every $A \in N_{\mathcal{C}}$ and every $a \in \Delta$, $a : \neg A \in \mathbb{L}(n) \Rightarrow (\mathbb{M}_{\mathbb{G}}, n) \vDash \neg A(a)$.*

Theorem 2.3.6 *(Soundness and Completeness of the $\mathcal{K}$-Tableau Algorithm) Let $\mathcal{T}$ be a simple acyclic TBox, and $\mathbb{G}$ be a constraint graph, complete w.r.t. local, global and terminological expansion rules. Then $\mathbb{G}$ is open if and only if $\mathbb{M}_{\mathbb{G}} \Vdash \mathbb{G}$ and $\mathbb{M}_{\mathbb{G}} \vDash \mathcal{T}$.*

Proof It suffices to prove the following:

- **Claim 1.** If $\mathbb{G}$ is open, then $\mathbb{M}_{\mathbb{G}} \Vdash \mathbb{G}$ and $\mathbb{M}_{\mathbb{G}} \vDash \mathcal{T}$.

- **Claim 2.** If $\mathbb{G}$ is closed, then there does not exist a Kripke structure $\mathbb{M}$ such that $\mathbb{M} \Vdash \mathbb{G}$.

Proof of Claim 1. For Claim 1, suppose that the complete constraint graph $\mathbb{G}$ is open. We first prove $\mathbb{M}_{\mathbb{G}} \Vdash \mathbb{G}$.

By the construction of $\mathbb{M}_{\mathbb{G}}$, for every $n, n' \in \mathbb{V}$, $i \in \mathbb{L}(n, n') \Rightarrow \mathcal{E}_i(n, n')$ and $(a, b) : R \in \mathbb{L}(n) \Rightarrow (\mathbb{M}_{\mathbb{G}}, n) \vDash R(a, b)$ where $R \in N_{\mathcal{R}}$. The implication $a : C \in \mathbb{L}(n) \Rightarrow (\mathbb{M}_{\mathbb{G}}, n) \vDash C(a)$ where $C \in \mathcal{C}$, is proved by induction on the structure of C. The base case is when $C \in N_{\mathcal{C}}$. If $C \in \Theta$, by the definition of $C^{\pi(n)}$, $(\mathbb{M}_{\mathbb{G}}, n) \vDash C(a)$. If $C \notin \Theta$, then there is a definition $C \doteq D \in \mathcal{T}$, and again by Definition 2.3.4, $C^{\pi(n)} = \{b \mid b : C \in \mathbb{L}(n)\} \cup D^{\pi(n)}$. Hence, $(\mathbb{M}_{\mathbb{G}}, n) \vDash C(a)$.

With respect to the induction step, the most involved case is that of the negation, which was dealt with in Lemma 2.3.5. The remaining cases, namely, $\sqcap, \sqcup, \exists, \forall, \Diamond$, and $\Box$, are proved below.

1. C is of the form $B_1 \sqcap B_2$. Since $\mathbb{G}$ is complete, $\{a : B_1, a : B_2\} \subseteq \mathbb{L}(n)$. By IH, $a : B_1 \in \mathbb{L}(n)$ and $a : B_2 \in \mathbb{L}(n) \Rightarrow (\mathbb{M}_{\mathbb{G}}, n) \vDash B_1(a)$ and $(\mathbb{M}_{\mathbb{G}}, n) \vDash B_2(a) \Leftrightarrow (\mathbb{M}_{\mathbb{G}}, n) \vDash B_1 \sqcap B_2(a) \Leftrightarrow (\mathbb{M}_{\mathbb{G}}, n) \vDash C(a)$.

2. C is of the form $B_1 \sqcup B_2$. Since $\mathbb{G}$ is complete, $\{a : B_1, a : B_2\} \cap \mathbb{L}(n) \neq \emptyset$. By IH, $a : B_1 \in \mathbb{L}(n)$ or $a : B_2 \in \mathbb{L}(n) \Rightarrow (\mathbb{M}_{\mathbb{G}}, n) \vDash B_1(a)$ or $(\mathbb{M}_{\mathbb{G}}, n) \vDash B_2(a) \Leftrightarrow (\mathbb{M}_{\mathbb{G}}, n) \vDash B_1 \sqcup B_2(a) \Leftrightarrow (\mathbb{M}_{\mathbb{G}}, n) \vDash C(a)$.

3. C is of the form $\exists R.B$. Since $\mathbb{G}$ is complete, there exists b s.t. $\{(a, b) : R, b : B\} \subseteq \mathbb{L}(n)$. Since $(a, b) : R \in \mathbb{L}(n) \Rightarrow (\mathbb{M}_{\mathbb{G}}, n) \vDash R(a, b)$ and by IH, $b : B \in \mathbb{L}(n) \Rightarrow (\mathbb{M}_{\mathbb{G}}, n) \vDash B(b)$, $(\mathbb{M}_{\mathbb{G}}, n) \vDash \exists R.B(a)$.

4. C is of the form $\forall R.B$. Since $\mathbb{G}$ is complete, for every b where $(a, b) : R \in \mathbb{L}(n)$, we have $b : B \in \mathbb{L}(n)$. Since $(a, b) : R \in \mathbb{L}(n) \Rightarrow (\mathbb{M}_{\mathbb{G}}, n) \vDash R(a, b)$ and by IH, $b : B \in \mathbb{L}(n) \Rightarrow (\mathbb{M}_{\mathbb{G}}, n) \vDash B(b)$, $(\mathbb{M}_{\mathbb{G}}, n) \vDash \forall R.B(a)$.

5. C is of the form $\Diamond_i B$. Since $\mathbb{G}$ is complete, there exists $n' \in \mathbb{V}$ s.t. $i \in \mathbb{L}(n, n')$ and $a : B \in \mathbb{L}(n')$. Since $i \in \mathbb{L}(n, n') \Rightarrow \mathcal{E}_i(n, n')$ and by IH, $a : B \in \mathbb{L}(n') \Rightarrow (\mathbb{M}_{\mathbb{G}}, n') \vDash B(a)$, we have $(\mathbb{M}_{\mathbb{G}}, n) \vDash \Diamond_i B(a)$.

6. C is of the form $\Box_i B$. Since $\mathbb{G}$ is complete, then for every $n' \in \mathbb{V}$ where $i \in \mathbb{L}(n, n')$, we have $a : B \in \mathbb{L}(n')$. Since $i \in \mathbb{L}(n, n') \Rightarrow \mathcal{E}_i(n, n')$ and by IH, $a : B \in \mathbb{L}(n') \Rightarrow (\mathbb{M}_\mathbb{G}, n') \vDash B(a)$, we have $(\mathbb{M}_\mathbb{G}, n) \vDash \Box_i B(a)$.

We next show that $\mathcal{T}$ is valid in $\mathbb{M}_\mathbb{G}$. Suppose that there is a node n and a definition $A \doteq D \in \mathcal{T}$ such that $(\mathbb{M}_\mathbb{G}, n) \nvDash A \doteq D$. Since $A \notin \Theta$, $A^{\pi(n)} := \{a \mid a : A \in \mathbb{L}(n)\} \cup D^{\pi(n)}$, and hence, $D^{\pi(n)} \subseteq A^{\pi(n)}$. Suppose that $D^{\pi(n)} \neq A^{\pi(n)}$. Then there is $b \in \mathcal{O}_\mathbb{G}$ such that $b \in A^{\pi(n)}$ and $b \notin D^{\pi(n)}$. This implies that $b \in \{a \mid a : A \in \mathbb{L}(n)\}$. $\mathbb{G}$ being complete and $b : A \in \mathbb{L}(n)$ imply that $b : D \in \mathbb{L}(n)$. We already proved that $\mathbb{M}_\mathbb{G} \Vdash \mathbb{G}$. So $(\mathbb{M}_\mathbb{G}, n) \vDash D(b) \Leftrightarrow b \in D^{\pi(n)}$, which is a contradiction. It follows that for every definition $A \doteq D \in \mathcal{T}$ and for every $n \in \mathbb{V}$, $(\mathbb{M}_\mathbb{G}, n) \vDash A \doteq D$.

Proof of Claim 2. Assume that the complete constraint tree $\mathbb{G}$ is closed. Then there is a node n in $\mathbb{G}$ such that $\{a : C, a : \neg C\} \subseteq \mathbb{L}(n)$ or $\{a : \bot\} \subseteq \mathbb{L}(n)$. Suppose there is a Kripke structure $\mathbb{M}$ and a mapping σ that satisfy $\mathbb{G}$. Then $a^{\pi(\sigma(n))} \in C^{\pi(\sigma(n))}$ and $a^{\pi(\sigma(n))} \in \neg C^{\pi(\sigma(n))}$, or $a^{\pi(\sigma(n))} \in \bot^{\pi(\sigma(n))}$. Either case leads to a contradiction. ■

Remark Firstly, note that Theorem 2.3.6 applies to general directed graphs (rather than just trees as, e.g., in [53]). Secondly, it is crucial that $\mathbb{G}$ is complete w.r.t. all the local, global and terminological expansion rules as given in Figures 2.1 and 2.2.

Corollary 2.3.7 *Given a simple acyclic TBox $\mathcal{T}$, let $\mathbb{G}$ be a constraint graph that is complete w.r.t. local, global and terminological expansion rules, and let $\mathbb{M}$ be an arbitrary Kripke structure. Then,* $\mathbb{M} \Vdash \mathbb{G} \Longrightarrow (\mathbb{M}_\mathbb{G} \Vdash \mathbb{G} \wedge \mathbb{M}_\mathbb{G} \vDash \mathcal{T})$.

Discussion. Designing a set of terminological expansion rules that provide a sound and complete tableau algorithm, and also lead to a PSPACE implementation is rather challenging. Recall the example presented just before Lemma 1: Given a definition $C \doteq D_1 \sqcap D_2$ and a constraint system $\mathbb{L}(n) = \{a : D_1, a : D_2, a : \neg C\}$, to generate a "quick" clash, one may expand $\mathbb{L}(n)$ by adding a constraint $a : C$. This would suggest a terminological expansion rule for the construct $\sqcap$: "If there is a node n with $\{a : B_1, a : B_2\} \subseteq \mathbb{L}(n), A \doteq B_1 \sqcap B_2 \in \mathcal{T}$, and $a : A \notin \mathbb{L}(n)$, then $\mathbb{L}(n) := \mathbb{L}(n) \cup \{a : A\}$". Similar terminological expansion rules could be defined for

other constructs. However, treating $\Diamond$ and $\Box$ analogously would require one to backtrack to the parent node, which would vastly complicate the algorithm. To avoid backtracking, our terminological expansion rules always examine the left-hand side of a definition and expand the right-hand side whenever necessary. As we will see in Section 2.5, this idea facilitates the PSPACE implementation of the $\mathcal{K}$-tableau algorithm $\Lambda_{\mathcal{K}}$ [2].

2.4 Query Answering

In this section we show how to use the tableau algorithm to answer queries.

Theorem 2.4.1 *Let $\Sigma = \langle \mathcal{A}, \mathcal{T} \rangle$ be a knowledge base, C a concept, and $a \in N_{\mathcal{O}}$. Let $\mathbb{L}(n_0)$ be the constraint system obtained from $\mathcal{A} \cup \{\neg C(a)\}$. Then $\Sigma \models C(a)$ if and only if all the complete constraint graphs generated by the tableau algorithm $\Lambda_{\mathcal{K}}$ from n_0 are closed.*

Proof Assume the hypotheses. The proof can be split into two claims:

- **Claim 1.** If $\Sigma \models C(a)$, then all the constraint graphs generated by $\Lambda_{\mathcal{K}}$ from n_0 are closed.
- **Claim 2.** If $\Sigma \not\models C(a)$, then there is an open and complete constraint graph generated by $\Lambda_{\mathcal{K}}$ from n_0.

Proof of Claim 1. Assume that $\Sigma \models C(a)$. By Definition 2.2.2, this means that for all $(\mathbb{M}, s)$, $(\mathbb{M}, s) \models \Sigma \Rightarrow (\mathbb{M}, s) \models C(a)$. Suppose that $\mathbb{G}$ is an open and complete constraint graph generated by $\Lambda_{\mathcal{K}}$ starting from n_0. By Theorem 2.3.6, $\mathbb{M}_{\mathbb{G}} \Vdash \mathbb{G}$ and $\mathbb{M}_{\mathbb{G}} \models \mathcal{T}$. By Theorem 2.3.3, $(\mathbb{M}_{\mathbb{G}}, n_0) \models \mathbb{L}(n_0)$. Because the set of constraints obtained from $\mathcal{A} \cup \{\neg C(a)\}$ is a subset of $\mathbb{L}(n_0)$, we have $(\mathbb{M}_{\mathbb{G}}, n_0) \models \mathcal{A}$ and $(\mathbb{M}_{\mathbb{G}}, n_0) \models \neg C(a)$. It follows that $(\mathbb{M}_{\mathbb{G}}, n_0) \models \Sigma$ and $(\mathbb{M}_{\mathbb{G}}, n_0) \models \neg C(a)$. This contradicts that $\Sigma \models C(a)$.

Proof of Claim 2. Suppose that $\Sigma \not\models C(a)$. By Definition 2.2.2, this means that for some $(\mathbb{M}_0, s_0)$, $(\mathbb{M}_0, s_0) \models \Sigma$ and $(\mathbb{M}_0, s_0) \not\models C(a)$; this implies that $(\mathbb{M}_0, s_0) \models \mathcal{T}$ and $(\mathbb{M}_0, s_0) \models \mathcal{A} \cup \{\neg C(a)\}$. We construct an initial constraint graph $\mathbb{G}_0$ consisting of a single node n_0 with

[2] If the terminological expansion rules go from left to right for definitions involving modalities (to avoid backtracking) and go from right to left for definitions that do not involve modalities, then the resulting tableau algorithm is incomplete. See an example in Appendix A.1.3.

label $\mathbb{L}(n_0)$ obtained from $\mathcal{A} \cup \{\neg C(a)\}$ and set the mapping $\sigma_0(n_0) = s_0$. Obviously, $\mathbb{M}_0 \Vdash \mathbb{G}_0$ via σ_0. By Lemma 1 and repeated application of Theorem 2.3.3, there is an execution of $\Lambda_{\mathcal{K}}$ resulting a complete constraint graph $\mathbb{G}$, a corresponding Kripke structure $\mathbb{M}$ and a mapping σ such that $\mathbb{M}$ is a semantic extension of $\mathbb{M}_0|_{N_\Sigma}$ where $\mathbb{M} \Vdash \mathbb{G}$ (via σ) and $\mathbb{M} \vDash \mathcal{T}$. Thus, $\mathbb{M} \Vdash \mathbb{G}$. By Corollary 2.3.7, $\mathbb{M}_{\mathbb{G}} \Vdash \mathbb{G}$ and $\mathbb{M}_{\mathbb{G}} \vDash \mathcal{T}$ where $\mathbb{M}_{\mathbb{G}}$ is the canonical Kripke structure of $\mathbb{G}$. It follows from Theorem 2.3.6 that $\mathbb{G}$ is open. ■

We revisit Example 2.2.3 to illustrate the use of tableau algorithm to answer queries against an $\mathcal{ALCK}_m$ knowledge base.

Example 2.4.2 *(Example 2.2.3 continued.) Consider the knowledge base* $\Sigma_1 = \langle \mathcal{A}, \mathcal{T} \rangle$ *where*

$\mathcal{A} = \{$ *ADVISE(john, mary), TEACHES(susan, cs525),* $\Diamond_1 Advisor(susan)$,
$\Diamond_2 Grad(mary)$, $\Box_2 Lecturer(susan)$, *Advisor(john),* $\neg BasicCourse(cs525)\}$

$\mathcal{T} = \{$ $Lecturer \doteq \forall TEACHES.BasicCourse$, $Advisor \doteq Professor \sqcap A$,
$A \doteq \exists ADVISE.Grad$ $\}$.

Each query will be answered by constructing a constraint graph.

Q1: Is john a professor? Query: Professor(john).

In this example, since there are no concepts involving the construct $\sqcup$ *or possibility of generating a concept involving* $\sqcup$*, there is only one complete constraint graph that can be constructed from* $\mathcal{A} \cup \{\neg Professor(john)\}$*. The constraint system* $\mathbb{L}(n_0)$ *at the root node* n_0 *is listed below:*

$\mathbb{L}(n_0) = \{$ *(john, mary) : ADVISE, (susan, cs525) : TEACHES,* $susan : \Diamond_1 Advisor$,
$mary : \Diamond_2 Grad$, $susan : \Box_2 Lecturer$, *john : Advisor, john : Professor,*
john : A, $john : \exists ADVISE.Grad$, *(john, x) : ADVISE, x : Grad,*
$john : \neg Professor$, $cs525 : \neg BasicCourse$ $\}$

Because of the constraints "john : Professor" and "john : $\neg$*Professor",* $\mathbb{L}(n_0)$ *has a clash and the constraint graph is closed. Hence,* $\Sigma_1 \vDash Professor(john)$ *and the answer to the query is YES.*

Q2: Is susan a lecturer? Query: Lecturer(susan).

We start by constructing a constraint system from $\mathcal{A} \cup \{\neg$*Lecturer(susan)*$\}$ *and end up with an open complete constraint graph* $\mathbb{G}_1$ *as follows.*

$\mathbb{L}(n_0)$ =$\{$ *(john, mary) : ADVISE, (susan, cs525) : TEACHES, susan :* $\Diamond_1$*Advisor, mary :* $\Diamond_2$*Grad, susan :* $\Box_2$*Lecturer, john : Advisor, john : Professor, john : A, john :* $\exists$*ADVISE.Grad, (john, x) : ADVISE, x : Grad, cs525 :* $\neg$*BasicCourse, susan :* $\neg$*Lecturer, susan :* $\exists$*TEACHES.*$\neg$*BasicCourse* $\}$

$\mathbb{L}(n_1)$ =$\{$ *susan : Advisor, susan : Professor, susan : A, susan :* $\exists$*ADVISE.Grad, (susan, y) : ADVISE, y : Grad* $\}$

$\mathbb{L}(n_2)$ =$\{$ *mary : Grad, susan : Lecturer, susan :* $\forall$*TEACHES.BasicCourse* $\}$

$\mathbb{L}(n_0, n_1) = \{1\}$, $\mathbb{L}(n_0, n_2) = \{2\}$.

The above $\mathbb{G}_1$ *provides a model of* $\langle \mathcal{A} \cup \{\neg$*Lecturer(susan)*$\}, \mathcal{T}\rangle$. *Therefore, we cannot conclude "YES" to the original query. We then go on to construct a constraint graph from* $\mathcal{A} \cup \{$*Lecturer(susan)*$\}$ *and similarly to Q1, there is a clash in* $\mathbb{L}(n_0)$.

$\mathbb{L}(n_0)$ =$\{$ *(john, mary) : ADVISE, (susan, cs525) : TEACHES, susan :* $\Diamond_1$*Advisor, mary :* $\Diamond_2$*Grad, susan :* $\Box_2$*Lecturer, john : Advisor, john : Professor, john : A, john :* $\exists$*ADVISE.Grad, (john, x) : ADVISE, x : Grad, cs525 :* $\neg$*BasicCourse, susan : Lecturer, susan :* $\forall$*TEACHES.BasicCourse, cs525 : BasicCourse* $\}$

Since there is only one constraint graph that can be constructed from $\mathcal{A} \cup \{$ *Lecturer(susan)* $\}$ *and it has a clash, we conclude that* $\Sigma_1 \models \neg$*Lecturer(susan) and therefore the answer to the query is NO.*

The queries Q3 and Q4 in Example 2.2.3 will be answered in the same way. ■

2.5 PSPACE implementation of the Tableau Algorithm $\Lambda_{\mathcal{K}}$

The model constructed by the $\mathcal{K}$-tableau algorithm $\Lambda_{\mathcal{K}}$ may be exponential in the size of input as illustrated by the following set of constraints $a : C_i$ where $C_i = \Diamond_1 A_{i1} \sqcap \Diamond_1 A_{i2} \sqcap \Box_1 C_{i+1}$ $(1 \leq i < n-1)$, and $C_n = \Diamond_1 A_{n1} \sqcap \Diamond_1 A_{n2}$.

We now describe the algorithm $\mathcal{ALCK}_m$-SAT (Algorithm 1), a PSPACE implementation for the tableau algorithm $\Lambda_{\mathcal{K}}$. Given an $\mathcal{ALCK}_m$ KB $\Sigma = \langle \mathcal{A}, \mathcal{T} \rangle$ and an $\mathcal{ALCK}_m$ query $C(a)$, the algorithm $\mathcal{ALCK}_m$-SAT decides whether $C(a)$ is satisfiable with respect to Σ. The algorithm $\mathcal{ALCK}_m$-SAT$(\Sigma, C(a))$ makes use of the recursive subroutine SAT$(n, \mathbb{L}(n))$ that imposes restrictions on the order in which expansion rules are applied so as to maintain only a single path of the constraint tree at all times during its execution.

The algorithm $\mathcal{ALCK}_m$-SAT expands constraint systems in a depth-first manner (see Figure 2.3). The expansion procedure creates two kinds of successors: successors of individuals w.r.t. roles that are created due to the $\exists$-rule, and successors of the current constraint system that are created due to the $\Diamond$-rule.

Within each constraint system, before applying the $\exists$-rule or the $\Diamond$-rule, the algorithm ensures that all the other local and terminological rules are applied exhaustively. Once this process is completed, the resulting constraint system, say $\mathbb{L}(n)$, remains fixed until the time when $\mathbb{L}(n)$ is removed. The algorithm then expands $\mathbb{L}(n)$, by applying the $\exists$-rule to a constraint of the form $b : \exists R.D \in \mathbb{L}(n)$, and creates an R-successor, say x, of the individual b, and constraints $(b, x) : R$, $x : D$ that are put in a "temporary" set $\mathbb{L}^x(n)$. In the presence of $(b, x) : R$ and $x : D$, other expansion rules may become applicable to constraints in $\mathbb{L}(n) \cup \mathbb{L}^x(n)$. So the algorithm then exhaustively applies local and terminological rules, except the $\exists$-rule. All these newly created constraints, except for $(b, x) : R$, are only about the fresh individual x and they are put into the set $\mathbb{L}^x(n)$. Since constraints about x cannot clash with constraints about other individuals, we consider $\mathbb{L}^x(n)$ as an auxiliary constraint system specifically for individual x. The algorithm checks in a depth first manner whether $\mathbb{L}^x(n)$ contains any clash (Line 14-16). During the recursive call (Line 14), new auxiliary constraint systems, e.g., $\mathbb{L}^y(n)$, may be created. Once $\mathbb{L}^y(n)$ was found to be satisfiable, the control returns to $\mathbb{L}^x(n)$ and $\mathbb{L}^y(n)$ is removed. If still $\mathbf{E}(n) \neq \emptyset$, another auxiliary constraint system will be created, and the space previously used by $\mathbb{L}^y(n)$ will be reused. Once $\mathbf{E}(n) = \emptyset$, $\mathbf{D}(n)$ is checked. If $\mathbf{D}(n) \neq \emptyset$, the $\Diamond$-rule will be applied and a new constraint system $\mathbb{L}^x(n')$ will be created (see Figure 2.3). Expansion rules are applied in $\mathbb{L}^x(n')$ the same manner as in $\mathbb{L}(n)$. If $\mathbb{L}^x(n')$ has been fully examined without any clash, the $\Diamond$-rule will be applied to another possible constraint and

Algorithm 1 $\mathcal{ALCK}_m$-SAT$(\Sigma, C(a))$

$\mathcal{ALCK}_m$-SAT$(\Sigma, C(a))$:= SAT$(n_0, \mathbb{L}(n_0))$, where $\Sigma = \langle \mathcal{A}, \mathcal{T} \rangle$ and $\mathbb{L}(n_0)$ is a constraint system obtained from $\mathcal{A} \cup \{C(a)\}$.

SAT$(n, \mathbb{L}(n))$:

while a local or terminological rule, except for the $\exists$-rule, is applicable to $\mathbb{L}(n)$ **do**
  apply the rule (if it is a $\sqcup$-rule, non-deterministically pick one choice),
  add the new constraints to $\mathbb{L}(n)$
end while
if $\mathbb{L}(n)$ contains a clash **then**
  return "not satisfiable"
end if
$\mathbf{E}(n) := \{a : \exists R.C \mid a : \exists R.C \in \mathbb{L}(n)$ and there is no b s.t. $(a, b) : R, b : C \in \mathbb{L}(n)\}$
$\mathbf{D}(n) := \{a : \Diamond_i C \mid a : \Diamond_i C \in \mathbb{L}(n)\}$
while $\mathbf{E}(n) \neq \emptyset$ **do**
  pick one $a : \exists R.C \in \mathbf{E}(n)$ and let $\mathbb{L}^x(n) := \{(a, x) : R, x : C\}$ where x is fresh
  while a local or terminological rule, except for the $\exists$-rule, is applicable to $\mathbb{L}(n) \cup \mathbb{L}^x(n)$ **do**
    apply the rule (if it is a $\sqcup$-rule, non-deterministically pick one choice),
    add the new constraint to $\mathbb{L}^x(n)$
  end while
  if SAT$(n, \mathbb{L}^x(n))$ = "not satisfiable" **then**
    return "not satisfiable"
  end if
  discard $\mathbb{L}^x(n)$
  $\mathbf{E}(n) := \mathbf{E}(n) \setminus \{a : \exists R.C\}$
end while
while $\mathbf{D}(n) \neq \emptyset$ **do**
  pick one $a : \Diamond_i C \in \mathbf{D}(n)$, create a new constraint system $\mathbb{L}(n')$
  let $\mathbb{L}(n') := \{a : C\}$ and $\mathbb{L}(n, n') := \{i\}$
  while the $\Box$-rule is applicable to $\mathbb{L}(n)$ **do**
    apply the rule in $\mathbb{L}(n)$, add corresponding constraints to $\mathbb{L}(n')$
  end while
  if SAT$(n', \mathbb{L}(n'))$ = "not satisfiable" **then**
    return "not satisfiable"
  end if
  discard $\mathbb{L}(n')$
  $\mathbf{D}(n) := \mathbf{D}(n) \setminus \{a : \Diamond_i C\}$
end while
return "satisfiable"

another constraint system will be created using the same space of $\mathbb{L}^x(n')$. When $\mathbf{D}(n) = \emptyset$, if no clash has been detected, $\mathbb{L}^x(n)$ is satisfiable. The control returns to $\mathbb{L}(n)$ and $\mathbb{L}^x(n)$ is removed so that the same space can be reused for another "fresh" individual.

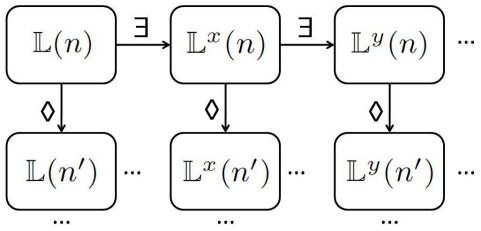

Figure 2.3 An Illustration of the Execution of Algorithm 1

The following example illustrates the operation of Algorithm 1.

Example 2.5.1 *Suppose that we have an initial constraint system* $\mathbb{L}(n) = \{a : \exists R.\Diamond_1 C,\ a : \forall R.\exists R.\Diamond_2 D,\ b : \Diamond_1 D\}$. *The constraint systems and the auxiliary constraint systems are created or removed in the following order:*

1. $\mathbb{L}^x(n) = \{(a, x) : R,\ x : \Diamond_1 C,\ x : \exists R.\Diamond_2 D\}$ *is created;*
2. $\mathbb{L}^y(n) = \{(x, y) : R,\ y : \Diamond_2 D\}$ *is created;*
3. $\mathbb{L}^y(n') = \{y : D\}$ *is created where* $(n, n') = \{2\}$*;*
4. $\mathbb{L}^y(n')$ *is removed;*
5. $\mathbb{L}^y(n)$ *is removed;*
6. $\mathbb{L}^x(n') = \{x : C\}$ *is created where* $(n, n') = \{1\}$*;*
7. $\mathbb{L}^x(n')$ *is removed;*
8. $\mathbb{L}^x(n)$ *is removed;*
9. $\mathbb{L}(n') = \{b : D\}$ *is created where* $(n, n') = \{1\}$*;*
10. $\mathbb{L}(n')$ *is removed.*

Eventually, the algorithm returns "satisfiable". ■

In reference to Figure 2.3, we note that at any one time only one path through the "tree" is maintained. For example, when this path consists of $\cdots, \mathbb{L}(n), \mathbb{L}^x(n), \mathbb{L}^x(n'), \cdots$, the temporary "nodes" $\mathbb{L}^y(n), \mathbb{L}^y(n'), \cdots$ would have already been processed and the space they used can therefore be reused. At that point of time, the node $\mathbb{L}(n')$, in Figure 2.3, has not yet been created.

We now proceed to show that the $\mathcal{ALCK}_m$ satisfiability problem can be solved in PSPACE. It suffices to show that $\mathcal{ALCK}_m$-SAT, the implementation of the tableau algorithm $\Lambda_{\mathcal{K}}$, runs in PSPACE.

Theorem 2.5.2 *The tableau algorithm $\Lambda_{\mathcal{K}}$ can be implemented in* PSPACE.

Proof Referring to the execution of $\mathcal{ALCK}_m$-SAT, within each (possibly auxiliary) constraint system, the algorithm $\mathcal{ALCK}_m$-SAT takes one existential constraint $a : \exists R.C$ at a time and the auxiliary constraint system is reset for the newly created constraints that are all about the witness individual of $a : \exists R.C$. The algorithm reuses the same space for new constraint systems that are successors of the current system. The constraint system $\mathbb{L}(n')$ is reset whenever such a successor of the current constraint system is created.

Since the TBox is acyclic, the depth of the auxiliary constraint systems created due to the $\exists$-rule or $\Diamond$-rule is linearly bounded by the length of the constraints in the original constraint system. Within each constraint system, the total number of constraints is polynomially bounded by the number of constraints in the initial constraint system. Furthermore, in algorithm $\mathcal{ALCK}_m$-SAT, once the $\exists$-rule is applied to a constraint $b : \exists R.D \in \mathbb{L}(n)$, it will not be applicable to the same constraint again (Line 18). Similarly, for constraints of the form $b : \Diamond_i D$, after the $\Diamond$-rule is applied to it, the same rule will not be applicable to this constraint any more (Line 29). It follows that the algorithm terminates and runs in PSPACE. ■

2.6 Tableau Algorithm for $\mathcal{ALCS}4_m$

In this section we study $\mathcal{ALCS}4_m$, an epistemically motivated language whose syntax is identical to that of $\mathcal{ALCK}_m$, but whose semantics is based on the modal logic $S4_m$. The modal logic $S4_m$ is well-suited to express epistemic knowledge in multiagent environments. This point

was argued eloquently in [56]. Given a knowledge base $\Sigma = \langle \mathcal{A}, \mathcal{T} \rangle$ and a query $C(a)$, we would like to know whether $\Sigma \models C(a)$ w.r.t. all $S4$-structures defined as follows.

A Kripke structure $\mathbb{M} = \langle S, \pi, \mathcal{E}_1, ..., \mathcal{E}_m \rangle$ is *reflexive* (*transitive*) if for every $i \in N_{\mathcal{E}}$, the relation $\mathcal{E}_i$ is reflexive (transitive). $\mathbb{M}$ is an $S4$-structure if it is reflexive and transitive. It can be easily shown that $S4$-structures satisfy the following two properties (see the analogous axioms (A3) and (A4) in [33]):

($e1$) (Truth) The facts known by experts are true; formally, for any world w, every $i \in N_{\mathcal{E}}$, if $w \models \Box_i C(a)$, then $w \models C(a)$.

($e2$) (Positive Introspection) If an expert knows something, then he/she knows that he/she knows it; formally, for any world w, every $i \in N_{\mathcal{E}}$, if $w \models \Box_i C(a)$, then $w \models \Box_i \Box_i C(a)$.

As discussed in Section 2.2.2, checking whether $\Sigma \models C(a)$ can be reduced to the problem of checking the existence of models. Given a knowledge base Σ and a query $C(a)$, we would like to build an open and complete constraint graph which can be used to construct an $S4$-structure as per Definition 2.3.4. However, the $\mathcal{K}$-tableau algorithm which utilizes only local, global and terminological expansion rules is not sufficient for this purpose. For example, consider a set of constraints $\mathcal{A} = \{a : \Box_1 C, a : \neg C\}$ with an empty TBox. Clearly, the constraint graph $\mathbb{G}$ consisting of a single node labeled with $\mathcal{A}$ is open and complete w.r.t. local, global and terminological expansion rules. By Theorem 2.3.6, there is a canonical Kripke structure $\mathbb{M}_{\mathbb{G}} = \langle \{s\}, \pi, \emptyset \rangle$ such that $\mathbb{M}_{\mathbb{G}} \Vdash \mathbb{G}$. But $\mathbb{M}_{\mathbb{G}}$ is not an $S4$-structure for $\mathbb{G}$ since it is not reflexive. In fact, due to reflexivity, $\mathbb{G}$ is not satisfiable in any $S4$-structure.

Accessibility Expansion Rules:

A^T-rule: If there is a node n with $a : \Box_i C \in \mathbb{L}(n)$, and $a : C \notin \mathbb{L}(n)$, then $\mathbb{L}(n) = \mathbb{L}(n) \cup \{a : C\}$.

A^4-rule: If there is a node n with $a : \Box_i C \in \mathbb{L}(n)$, and n has an i-successor n' with $a : \Box_i C \notin \mathbb{L}(n')$, then $\mathbb{L}(n') := \mathbb{L}(n') \cup \{a : \Box_i C\}$.

Figure 2.4 The accessibility expansion rules

To address this problem, we adapt the $\mathcal{K}$-tableau algorithm by adding two accessibility expansion rules that implement the two properties ($e1$) and ($e2$) stated above and which will

facilitate the construction of $S4$-models. They are shown in Figure 2.4. However, it turns out that the tableau algorithm with the accessibility rules and the current local, global and terminological rules may not terminate. Consider an initial constraint system $\mathbb{L}(n_0) = \{a : \Box_1\Diamond_1 C\}$. After an application of the A^T-rule, a constraint $a : \Diamond_1 C$ is added to $\mathbb{L}(n_0)$ and leads to the application of $\Diamond$-rule which creates a new constraint system $\mathbb{L}(n_1) = \{a : C\}$ with $\mathbb{L}(n_0, n_1) = \{1\}$. After an application of the A^4-rule followed by another application of the A^T-rule, $\mathbb{L}(n_1) = \{a : C, a : \Box_1\Diamond_1 C, a : \Diamond_1 C\}$. With the current tableau algorithm $\Lambda_{\mathcal{K}}$, a new constraint system, say $\mathbb{L}(n_2)$, will be created and contain the same constraints as $\mathbb{L}(n_1)$; this process will not terminate. Since $S4$-structures are reflexive, any world in the structure is an i-successor of itself ($i \in N_{\mathcal{E}}$) and this suggests that we modify the condition of the $\Diamond$-rule such that the $\Diamond$-rule is not applicable to $a : \Diamond_i C \in \mathbb{L}(n)$ whenever $a : C \in \mathbb{L}(n)$. Unfortunately, this modification by itself is not sufficient to ensure termination as is illustrated in the following example.

Example 2.6.1 *Consider an initial constraint system* $\mathbb{L}(n_0) = \{a : \Box_1\Diamond_1 D,\ a : \Diamond_1 D,\ a : \Box_1\Diamond_1\exists R.\Box_1\Diamond_1 C,\ a : \Diamond_1\exists R.\Box_1\Diamond_1 C\}$. *Applications of several expansion rules may lead to the following constraint systems:* $\mathbb{L}(n_1) = \{a : \exists R.\Box_1\Diamond_1 C,\ (a, x) : R,\ x : \Box_1\Diamond_1 C,\ x : \Diamond_1 C\} \cup \mathbb{L}(n_0)$, $\mathbb{L}(n_2) = \{x : C,\ x : \Box_1\Diamond_1 C,\ x : \Diamond_1 C\} \cup \mathbb{L}(n_0)$, *and* $\mathbb{L}(n_3) = \{a : \exists R.\Box_1\Diamond_1 C,\ (a, y) : R,\ y : \Box_1\Diamond_1 C,\ y : \Diamond_1 C\} \cup \{x : \Box_1\Diamond_1 C,\ x : \Diamond_1 C\} \cup \mathbb{L}(n_0)$ *where* $\mathbb{L}(n_0, n_1) = \mathbb{L}(n_1, n_2) = \mathbb{L}(n_2, n_3) = \{1\}$ *and individuals* x, y *were freshly chosen. Both* $\mathbb{L}(n_1)$ *and* $\mathbb{L}(n_3)$ *were created because of the constraint* $a : \Diamond_1\exists R.\Box_1\Diamond_1 C$. *The constraint system* $\mathbb{L}(n_2)$ *was created because of the constraint* $x : \Diamond_1 C$. *Since the* $\exists$*-rule always picks a fresh individual, the box-assertions for the previously picked individuals will be carried along to the newly created constraint system. So there may be larger and larger sets of constraints with the creation of new constraint systems.* ■

To address this problem, we use a blocking technique. We define $\mathbb{B}_i^n = \{a : \Box_i C \in \mathbb{L}(n) \mid a \in N_{\mathcal{O}}, C \in \mathcal{C}\}$ for $i \in N_{\mathcal{E}}$. In a constraint tree, we say that n_1 is an *i-ancestor* of n_k and that n_k is an *i-descendant* of n_1 if $i \in \bigcap_{j=1}^{k-1} \mathbb{L}(n_j, n_{j+1})$ where $k > 1$. Among all the tableau expansion rules that we use, there are two expansion rules that create new entities - the $\Diamond$-rule and the $\exists$-rule. To enforce termination, we must limit the applicability of these rules. Figure 2.5 lists

the "blocking" versions of these rules: the $\Diamond_b$-rule and the $\exists_b$-rule.

Definition 2.6.2 *An assertion* $a : \Diamond_i C \in \mathbb{L}(n)$ is blocked by a node n' *if (i)* n' *is an* i*-ancestor of* n*, (ii)* $\mathbb{B}_i^n = \mathbb{B}_i^{n'}$*, and (iii)* $a : C \in \mathbb{L}(n')$*. An assertion* $a : \exists r.C \in \mathbb{L}(n)$ is blocked by an individual $x \in \mathcal{O}_\mathbb{G} \setminus \mathcal{O}_\Sigma$ *if there is an ancestor* n' *of* n *such that* $\{a : \exists r.C,\ (a, x) : r,\ x : C\} \subseteq \mathbb{L}(n')$.

$\Diamond_b$**-rule:**	If there is a node n such that none of the expansion rules except the $\Diamond_b$-rule is applicable to $\mathbb{L}(n)$, and 1. $a : \Diamond_i C \in \mathbb{L}(n), a : C \notin \mathbb{L}(n)$, 2. $a : \Diamond_i C$ is not blocked, 3. n has no i-successor l with $a : C \in \mathbb{L}(l)$, then add a new i-successor n' of n with $\mathbb{L}(n') := \{a : C\}$ and $\mathbb{L}(n, n') = \{i\}$.
$\exists_b$**-rule:**	If there is a node n with $a : \exists R.C \in \mathbb{L}(n)$ and there is no $b \in \mathcal{O}_\mathbb{G}$ such that $\{(a, b) : R, b : C\} \subseteq \mathbb{L}(n)$, then (i) if there is an individual $x \in \mathcal{O}_\mathbb{G} \setminus \mathcal{O}_\Sigma$ such that $a : \exists R.C$ is blocked by x, then $\mathbb{L}(n) := \mathbb{L}(n) \cup \{(a, x) : R, x : C\}$; or (ii) if $a : \exists R.C$ is not blocked by any individual in $\mathcal{O}_\mathbb{G} \setminus \mathcal{O}_\Sigma$, then $\mathbb{L}(n) := \mathbb{L}(n) \cup \{(a, c) : R, c : C\}$ where c is fresh.

Figure 2.5 The $\Diamond_b$-rule and the $\exists_b$-rule

The $\sqcap$-, $\sqcup$-, $\exists_b$- and $\forall$-rules (respectively, the $\Diamond_b$- and $\Box$-rules/the A^T- and A^4-rules) are jointly referred to as $S4$-*local rules* (respectively, $S4$-*global rules*/$S4$-*accessibility rules*). The $S4$-local, $S4$-global, terminological and $S4$-accessibility expansion rules together are called $S4$-*rules*. We denote by Λ_{S4} the $S4$-*tableau algorithm* which nondeterministically applies an $S4$-rule until no rule is applicable. As was the case with $\Lambda_\mathcal{K}$, the graph-structure produced by Λ_{S4} will be a tree. The next theorem establishes the soundness of the expansion rules used in Λ_{S4}.

Theorem 2.6.3 *(Soundness of expansion rules) Given an* $S4$*-structure* $\mathbb{M} = \langle S,\ \pi,\ \mathcal{E}_1, ...,\ \mathcal{E}_m \rangle$ *and an acyclic TBox* $\mathcal{T}$ *with* $\mathbb{M} \vDash \mathcal{T}$*, let* $\mathbb{G}$ *be a constraint graph,* α *an* $S4$*-rule and* $\mathbb{G}_\alpha$ *a constraint graph obtained by applying* α *to* $\mathbb{G}$*. If* $\mathbb{M} \Vdash \mathbb{G}$ *via* σ*, then there exists a semantic extension* $\mathbb{M}_\alpha$ *(also an* $S4$*-structure) of* $\mathbb{M}|_{N_\Sigma \cup \mathcal{O}_\mathbb{G}}$ *s.t.* $\mathbb{M}_\alpha \Vdash \mathbb{G}_\alpha$ *via* σ' *(which extends* σ*) and* $\mathbb{M}_\alpha \vDash \mathcal{T}$*. Furthermore,* $\mathbb{M}_\alpha \Vdash \mathbb{G}$.

Proof Assume the hypotheses. It suffices to prove that the $\exists_b$-rule, the $\Diamond_b$-rule and $S4$-accessibility expansion rules preserve the existence of $S4$-models. In the other cases, $\mathbb{M}_\alpha$ is a semantic extension of $\mathbb{M}$ (see proof of Theorem 2.3.3) and hence, if $\mathbb{M}$ is an $S4$-structure, so is $\mathbb{M}_\alpha$.

- If α is an A^T-rule, then there is a node n with $a : \Box_i C \in \mathbb{L}(n)$ and $a : C \notin \mathbb{L}(n)$. After applying α, $a : C \in \mathbb{L}(n)$. Since $\mathbb{M}$ is reflexive, $\mathbb{G}_\alpha$ obtained from $\mathbb{G}$ is satisfied by $\mathbb{M}$ via σ.

- If α is an A^4-rule, then there are two nodes n and n' in $\mathbb{G}$ such that $i \in \mathbb{L}(n, n')$, $a : \Box_i C \in \mathbb{L}(n)$ and $a : \Box_i C \notin \mathbb{L}(n')$. After applying α, $a : \Box_i C$ is added to $\mathbb{L}(n')$. Let n'' be an arbitrary i-successor of n'. Because $\mathbb{M} \Vdash \mathbb{G}$ and $a : \Box_i C \in \mathbb{L}(n)$, we have $(\mathbb{M}, \sigma(n)) \vDash \Box_i C(a)$. Since $\mathbb{M}$ being transitive implies that n'' is also an i-successor of n, we have $(\mathbb{M}, \sigma(n'')) \vDash C(a)$. Because n'' is an arbitrary i-successor of n', $(\mathbb{M}, \sigma(n')) \vDash \Box_i C(a)$. Therefore, $\mathbb{G}_\alpha$ obtained from $\mathbb{G}$ is satisfied by $\mathbb{M}$ via σ.

- If α is a $\Diamond_b$-rule, then there is a node n such that none of the expansion rules except the $\Diamond_b$-rule is applicable, $a : \Diamond_i C \in \mathbb{L}(n)$, $a : \Diamond_i C$ is not blocked, and n does not have an i-successor l with $a : C \in \mathbb{L}(l)$. By Definition 2.3.2, $a : \Diamond_i C \in \mathbb{L}(n)$ implies $(\mathbb{M}, \sigma(n)) \vDash \Diamond_i C(a)$ which means that there is a world s with $(\sigma(n), s) \in \mathcal{E}_i$ and $a^{\pi(s)} \in C^{\pi(s)}$. There are two cases. (i) If $a : C \notin \mathbb{L}(n)$, then after applying the $\Diamond_b$-rule, a new node n' is added to $\mathbb{G}$ with $\mathbb{L}(n') = \{a : C\}$, $\mathbb{L}(n, n') = \{i\}$ and $\mathcal{L}(n, n') = a : \Diamond_i C$. Extend σ to σ' such that $\sigma'(n') = s$. $\mathbb{M}$ satisfies the resulting $\mathbb{G}_\alpha$ via σ'. (ii) When $a : C \in \mathbb{L}(n)$, since $\mathbb{M}$ is reflexive, $(\sigma(n), \sigma(n)) \in \mathcal{E}_i$, then $s = \sigma(n)$ and $a^{\pi(s)} \in C^{\pi(s)}$.

- If α is an $\exists_b$-rule, then there is a node n with $a : \exists R.C \in \mathbb{L}(n)$ and there is no $b \in \mathcal{O}_\mathbb{G}$ such that $\{(a, b) : R, b : C\} \subseteq \mathbb{L}(n)$. There are two cases. (i) If $a : \exists R.C$ is blocked by x, then $x \in \mathcal{O}_\mathbb{G} \setminus \mathcal{O}_\Sigma$ and there is an i-ancestor n' of n such that x is a witness for the assertion $a : \exists R.C \in \mathbb{L}(n')$. Since $a : \exists R.C \in \mathbb{L}(n)$, we have $(\mathbb{M}, \sigma(n)) \vDash \exists R.C(a)$. So there exists an element $d \in \Delta$ such that $(a^{\pi(\sigma(n))}, d) \in R^{\pi(\sigma(n))}$ and $d \in C^{\pi(\sigma(n))}$. Let $x^{\pi(\sigma(n))} = d$. Then we have $x^{\pi(\sigma(n))} \in C^{\pi(\sigma(n))}$ and $(a^{\pi(\sigma(n))}, x^{\pi(\sigma(n))}) \in R^{\pi(\sigma(n))}$. Therefore, the newly

added constraints $(a, x) : R$ and $x : C$ are satisfied. (ii) If $a : \exists R.C$ is not blocked, the proof is same as that in Theorem 2.3.3). ■

In the case of $\mathcal{ALCK}_m$, since the KB is finite and the TBox is acyclic, there are finitely many assertions in each constraint system and so the outdegree of each constraint system is finite. Moreover, since no A^4-rule is involved (and the TBox is acyclic), whenever a new constraint system is created, the length of an assertion in the new constraint system is shorter than the assertion that creates it. It follows that the length of a path is also finite and therefore, the tableau algorithm $\Lambda_{\mathcal{K}}$ terminates. However, in the $\mathcal{ALCS}4_m$ case, because of the A^4-rule, the same assertion may be passed from one constraint system to its successor and so it is not immediately obvious why the tableau algorithm Λ_{S4} terminates. The following lemma deals with this issue.

Lemma 2.6.4 *Given a KB $\Sigma = \langle \mathcal{A}, \mathcal{T} \rangle$ and a query $C(a)$, the S4-tableau algorithm Λ_{S4} that takes $\langle \mathcal{A} \cup \{\neg C(a)\}, \mathcal{T} \rangle$ as input terminates.*

Proof It suffices to show that the constraint tree that Λ_{S4} creates is a finite tree and that each constraint system contains finitely many constraints.

Let l be the total number of sub-expressions of all the concepts and roles that appear in Σ or $C(a)$. Within each constraint system, for each individual $x \in \mathcal{O}_{\mathbb{G}}$, the number of assertions involving individual x is bounded by l. Since the TBox is acyclic, at most one fresh individual is chosen for each $\exists$-assertion and so the total number of individuals in each constraint system is also bounded by l. Therefore, the size of each constraint system is $O(l^2)$. It follows that the number of $\Diamond$-assertions in each constraint system and hence the outdegree of each node in a constraint tree are $O(l^2)$.

Note that starting from the root, each path actually contains a sequence of i-edges followed by a sequence of j-edges ($j \neq i$), and so on. Let i-sequence denote the sequence of nodes as well as the constraint systems associated with them where nodes (except the starting node) are i-descendants of the starting node. Also note that due to the $\exists_b$-rule, for each $\exists$-assertion within an i-sequence, at most one fresh individual will be chosen as witness and so there are $O(l)$ individuals within each i-sequence. It follows that the total number of distinct $\Box$-assertions

and $\Diamond$-assertions along each i-sequence of a constraint tree is $O(l^2)$. To show that the length of each path of a constraint tree is finite, it suffices to show that (1) each i-sequence has finite length, and (2) on each path, there are finitely many such sequences:

(1) Since there are $O(l^2)$ $\Box$-assertions in one i-sequence of a constraint tree, there are $O(l^2)$ $\Box_i$-assertions in each i-sequence. Due to the A^4-rule, every $\Box_i$-assertion is passed to its i-descendants. So there will be one constraint system $\mathbb{L}(n^*)$ in an i-sequence which contains all the $\Box_i$-assertions that appear in this i-sequence. Starting from node n^*, each $\Diamond_i$-assertion will be expanded once and then will remain blocked so that the $\Diamond_b$-rule is not applicable to this assertion any more. Since there are $O(l^2)$ $\Diamond_i$-assertions in an i-sequence, each i-sequence is finite.

(2) If a j-sequence starts from the last node of an i-sequence ($j \neq i$), then no $\Box_k$-assertions ($k \neq j$) can be passed to the constraint systems in the j-sequence (see A^4-rule). There may be two or more different i-sequences in one path. However, since the TBox is acyclic, the constraint systems associated with two different i-sequences will be different and none of such sequences can be repeated in one path. Specifically, every $\Box$-concept appearing in the latter i-sequence is a proper sub-concept of some $\Box$-concept appearing in the former i-sequence (with a shorter length). Therefore, the number of distinct such sequences in one path of a constraint tree is finite.

Since each constraint system has finitely many constraints with a finite outdegree and each path is finite, Λ_{S4} terminates. ■

Let $\mathbb{G}$ be an open and complete constraint tree resulting from Λ_{S4}. To obtain an $S4$-model for $\mathbb{G}$, we need to construct a graph from $\mathbb{G}$ such that whenever $a : \Diamond_i C \in \mathbb{L}(n)$, there is an i-successor n' of n such that $a : C \in \mathbb{L}(n')$. Moreover, for each $i \in N_{\mathcal{E}}$, the set of edges labeled by $\{i\}$ should represent a reflexive and transitive relation. Formally, this is defined as follows.

Definition 2.6.5 *Let $\mathbb{G} = \langle \mathbb{V}, \mathbb{E}, \mathbb{L} \rangle$ be an arbitrary constraint tree resulting from Λ_{S4}. An $S4$* constraint graph $\mathbb{G}^{S4} = \langle \mathbb{V}, \mathbb{E}^*, \mathbb{L}^* \rangle$ *is obtained from $\mathbb{G}$ as follows:*

1. *For every $i \in N_{\mathcal{E}}$,*

   $\mathbb{E}_i := \{(n, n') \mid n, n' \in \mathbb{V}$, *either* $i \in \mathbb{L}(n, n')$ *or* $a : \Diamond_i C \in \mathbb{L}(n)$ *is blocked by* $n'\}$,

$\mathbb{E}_i^*$ *is the reflexive and transitive closure of* $\mathbb{E}_i$,

$\mathbb{E}^* := \bigcup_i \mathbb{E}_i^*$;

2. *For every* $n \in \mathbb{V}$ *and every* $e \in \mathbb{E}^*$,

   $\mathbb{L}^*(n) := \mathbb{L}(n)$,

   $\mathbb{L}^*(e) := N_{\mathcal{E}}$, *if* e *is a self-loop edge,*

   $\mathbb{L}^*(e) := \{i\}$, *if* $e \in \mathbb{E}_i^*$ *and* e *is not a self-loop edge.*

Note that when $i \neq j$, $\mathbb{E}_i \cap \mathbb{E}_j = \emptyset$. Moreover, since $\mathbb{G}$ is a tree, $\mathbb{E}_i^* \cap \mathbb{E}_j^* = \{(n, n) \mid n \in \mathbb{V}\}$. The following lemma shows the relationship between $\mathbb{G}$ and $\mathbb{G}^{S4}$.

Lemma 2.6.6 *If a constraint tree* $\mathbb{G} = \langle \mathbb{V}, \mathbb{E}, \mathbb{L} \rangle$ *is open and complete w.r.t. S4-rules, then* $\mathbb{G}^{S4} = \langle \mathbb{V}, \mathbb{E}^*, \mathbb{L}^* \rangle$ *is also open and complete w.r.t. S4-rules. Moreover,* $\mathbb{G}^{S4}$ *is open and complete w.r.t. local, global and terminological expansion rules.*

Proof Suppose that $\mathbb{G}$ is open and complete w.r.t. $S4$-rules. Let us analyze the applicability of the expansion rules in $\mathbb{G}^{S4}$.

- A^4-rule: If an A^4-rule is applicable in $\mathbb{G}^{S4}$, then there is a node n with $a : \Box_i C \in \mathbb{L}^*(n) = \mathbb{L}(n)$ and n has an i-successor n' with $a : \Box_i C \notin \mathbb{L}^*(n')$. Since $a : \Box_i C \in \mathbb{L}(n)$ and $\mathbb{G}$ is complete (specifically, the A^4-rule is not applicable), n' cannot be an i-descendant of n in $\mathbb{G}$. In view of the construction of $\mathbb{E}_i^*$ (see Definition 2.6.5), there is an assertion $b : \Diamond_i D \in \mathbb{L}(n)$ that is blocked by a node n'' (an i-ancestor of n in $\mathbb{G}$) which is either n' itself or an i-ancestor of n'. It follows from Definition 2.6.2 that $\mathbb{B}_i^n = \mathbb{B}_i^{n''}$. Moreover, if n'' is an i-ancestor of n' in $\mathbb{G}$, by the the A^4-rule, we have $\mathbb{B}_i^{n''} \subseteq \mathbb{B}_i^{n'}$. Hence, $a : \Box_i C \in \mathbb{L}(n') = \mathbb{L}^*(n')$, which is a contradiction. Therefore, no A^4-rule is applicable in $\mathbb{G}^{S4}$.

- $\Box$-rule: Since no A^4-rule is applicable in $\mathbb{G}^{S4}$ by the previous case, for any two nodes n, n' such that $i \in \mathbb{L}^*(n, n')$, $a : \Box_i C \in \mathbb{L}^*(n) \Rightarrow a : \Box_i C \in \mathbb{L}^*(n')$. Furthermore, since no A^T-rule is applicable in $\mathbb{G}$, $a : \Box_i C \in \mathbb{L}^*(n') = \mathbb{L}(n')$ implies $a : C \in \mathbb{L}(n') = \mathbb{L}^*(n')$. Therefore, no $\Box$-rule is applicable in $\mathbb{G}^{S4}$.

- $\Diamond_b$-rule: Since $\mathbb{G}$ is a subgraph of $\mathbb{G}^{S4}$, if an assertion $a : \Diamond_i C \in \mathbb{L}^*(n)$ is not blocked by any i-ancestor, $a : C \notin \mathbb{L}^*(n)$ and there is no i-successor n' of n such that $a : C \in \mathbb{L}^*(n')$ in $\mathbb{G}^{S4}$, same happens in $\mathbb{G}$. However, this contradicts that $\mathbb{G}$ is complete w.r.t. $\Diamond_b$-rule. Therefore, no $\Diamond_b$-rule is not applicable in $\mathbb{G}^{S4}$.

Since none of the A^4-, $\Box$- and $\Diamond_b$-rules are applicable and all the constraint systems in $\mathbb{G}^{S4}$ remain the same as those in $\mathbb{G}$, if no $S4$-local, terminological expansion rule, or A^T-rule is applicable in $\mathbb{G}$, it is not applicable in $\mathbb{G}^{S4}$ either. Next we analyze the applicability of the $\Diamond$- and $\exists$-rules.

- $\Diamond$-rule: If a $\Diamond$-rule is applicable in $\mathbb{G}^{S4}$, then there is a node n with $a : \Diamond_i C \in \mathbb{L}^*(n) = \mathbb{L}(n)$, $a : C \notin \mathbb{L}^*(n) = \mathbb{L}(n)$ and n does not have an i-successor n' in $\mathbb{G}^{S4}$ such that $a : C \in \mathbb{L}^*(n') = \mathbb{L}(n')$. This implies that in $\mathbb{G}$ the assertion $a : \Diamond_i C \in \mathbb{L}(n)$ was blocked by an i-ancestor n_1 of n. It follows that $a : C \in \mathbb{L}(n_1)$ and $\mathbb{B}_i^{n_1} = \mathbb{B}_i^n$. By Definition 2.6.5, there is an edge $(n, n_1) \in \mathbb{E}_i^*$ in $\mathbb{G}^{S4}$ and so n_1 is an i-successor of n such that $a : C \in \mathbb{L}^*(n_1) = \mathbb{L}(n_1)$, which is a contradiction. Therefore, no $\Diamond$-rule is applicable in $\mathbb{G}^{S4}$.

- $\exists$-rule: All constraint systems in $\mathbb{G}^{S4}$ are same as those in $\mathbb{G}$. Moreover, $\mathbb{G}$ is complete w.r.t. $\exists_b$-rule, i.e., for every $n \in \mathbb{V}$, if $a : \exists R.C \in \mathbb{L}^*(n)$, there is a witness $x \in \mathcal{O}_{\mathbb{G}}$ such that $\{(a, x) : R, x : C\} \subseteq \mathbb{L}^*(n)$. Therefore, no $\exists$-rule is applicable in $\mathbb{G}^{S4}$.

It follows that $\mathbb{G}^{S4}$ is complete w.r.t. local, $S4$-local, global, $S4$-global, terminological and $S4$-accessibility expansion rules. Furthermore, the constraint systems in $\mathbb{G}^{S4}$ are exactly the same as the corresponding ones in $\mathbb{G}$ and since $\mathbb{G}$ is open, so is $\mathbb{G}^{S4}$. ■

Note that the converse implication of Lemma 2.6.6 does not hold. That is, $\mathbb{G}^{S4} = \langle \mathbb{V}, \mathbb{E}^*, \mathbb{L}^* \rangle$ being open and complete (w.r.t. $S4$-rules) does not imply that $\mathbb{G} = \langle \mathbb{V}, \mathbb{E}, \mathbb{L} \rangle$ is open and complete (w.r.t. $S4$-rules). For example, suppose that we have $\mathbb{L}(n_0) = \{a : \Diamond_1 C, a : \Diamond_1 \Diamond_1 C\} = \mathbb{L}^*(n_0)$, $\mathbb{L}(n_1) = \{a : \Diamond_1 C\} = \mathbb{L}^*(n_1)$ and $\mathbb{L}(n_2) = \{a : C\} = \mathbb{L}^*(n_2)$ where $\mathbb{L}(n_0, n_1) = \mathbb{L}(n_1, n_2) = \{1\}$. $\mathbb{G}$ is not complete since n_0 does not have a 1-successor l such that $a : C \in \mathbb{L}(l)$. However, the corresponding $\mathbb{G}^{S4}$ is complete because $(n_0, n_2) \in \mathbb{E}_1^*$.

The canonical Kripke structure $\mathbb{M}_{\mathbb{G}^{S4}}$ is obtained from $\mathbb{G}^{S4}$ using Definition 2.3.4. By Definition 2.6.5, $\mathbb{M}_{\mathbb{G}^{S4}}$ is actually an $S4$-structure. To show the soundness and completeness of the tableau algorithm Λ_{S4}, we need to show that any complete constraint graph $\mathbb{G}$ (w.r.t $S4$-rules) is open if and only if there is an $S4$-structure that satisfies $\mathbb{G}$. The next lemma shows that the canonical Kripke structure $\mathbb{M}_{\mathbb{G}^{S4}}$ is such an $S4$-structure for $\mathbb{G}$.

Lemma 2.6.7 *Let $\mathbb{G} = \langle \mathbb{V}, \mathbb{E}, \mathbb{L} \rangle$ be a constraint tree and $\mathbb{G}^{S4} = \langle \mathbb{V}, \mathbb{E}^*, \mathbb{L}^* \rangle$ the constraint graph obtained from $\mathbb{G}$ by Definition 2.6.5. Let $\mathbb{M}_{\mathbb{G}^{S4}} = \langle S, \pi, \mathcal{E}_1, ..., \mathcal{E}_m \rangle$ be the canonical Kripke structure of $\mathbb{G}^{S4}$. Then $\mathbb{M}_{\mathbb{G}^{S4}} \Vdash \mathbb{G}^{S4} \Rightarrow \mathbb{M}_{\mathbb{G}^{S4}} \Vdash \mathbb{G}$.*

Proof Suppose $\mathbb{M}_{\mathbb{G}^{S4}} \Vdash \mathbb{G}^{S4}$ via σ where σ is an identity function (see Definitions 2.3.2 and 2.3.4). Since for every $n \in \mathbb{V}$, $\mathbb{L}(n) = \mathbb{L}^*(n)$, $\mathbb{E} \subseteq \mathbb{E}^*$ and for every $e \in \mathbb{E}$, $\mathbb{L}(e) = \mathbb{L}^*(e)$, it is clear that $\mathbb{M}_{\mathbb{G}^{S4}} \Vdash \mathbb{G}$ via σ. Hence, $\mathbb{M}_{\mathbb{G}^{S4}} \Vdash \mathbb{G}^{S4} \Rightarrow \mathbb{M}_{\mathbb{G}^{S4}} \Vdash \mathbb{G}$. ■

Theorem 2.6.8 *(Soundness and completeness of Λ_{S4}) Let $\mathbb{G} = \langle \mathbb{V}, \mathbb{E}, \mathbb{L} \rangle$ be a complete constraint tree resulting from Λ_{S4} applied to a KB $\Sigma = \langle \mathcal{A}, \mathcal{T} \rangle$ where $\mathcal{T}$ be a simple acyclic TBox. Then $\mathbb{G}$ is open if and only if $\mathbb{M}_{\mathbb{G}^{S4}} \Vdash \mathbb{G}$ and $\mathbb{M}_{\mathbb{G}^{S4}} \vDash \mathcal{T}$.*

Proof ($\Rightarrow$) Suppose that $\mathbb{G}$ is open and complete w.r.t. $S4$-rules. Let $\mathbb{G}^{S4}$ be the constraint graph constructed from $\mathbb{G}$ by Definition 2.6.5. By Lemma 2.6.6, $\mathbb{G}^{S4}$ is open and complete w.r.t. local, global and terminological expansion rules, and hence, by Theorem 2.3.6, $\mathbb{M}_{\mathbb{G}^{S4}} \Vdash \mathbb{G}^{S4}$ and $\mathbb{M}_{\mathbb{G}^{S4}} \vDash \mathcal{T}$. By Lemma 2.6.7, $\mathbb{M}_{\mathbb{G}^{S4}} \Vdash \mathbb{G}$.

($\Leftarrow$) The proof is exactly the same as the proof of Claim 2 in Theorem 2.3.6. ■

By Lemma 2.6.4, the $S4$-tableau algorithm Λ_{S4} terminates. Based on Λ_{S4}, a PSPACE implementation $\mathcal{ALCS}4_m$-SAT for $\mathcal{ALCS}4_m$-satisfiability can be obtained following the approach of $\mathcal{ALCK}_m$-SAT. The basic idea is to maintain a single path of the constraint tree during the execution by imposing restrictions on the order of application of the expansion rules. The algorithm $\mathcal{ALCS}4_m$-SAT$(\Sigma, C(a))$ (see Algorithm 2) calls the subroutine $S4$-SAT by providing the input arguments n_0 and $\mathbb{L}(n_0)$, where $\Sigma = \langle \mathcal{A}, \mathcal{T} \rangle$ and $\mathbb{L}(n_0)$ is a constraint system obtained from $\mathcal{A} \cup \{C(a)\}$. The subroutine $S4$-SAT differs from the subroutine SAT in Algorithm 1 mainly at the following points:

Algorithm 2 $\mathcal{ALCS}4_m$-SAT

$\mathcal{ALCS}4_m$-SAT$(\Sigma, C(a))$:= $S4$-SAT$(n_0, \mathbb{L}(n_0))$, where $\Sigma = \langle \mathcal{A}, \mathcal{T} \rangle$, $\mathcal{T}$ is a simple acyclic TBox, and $\mathbb{L}(n_0)$ is a constraint system obtained from $\mathcal{A} \cup \{C(a)\}$.

$S4$-SAT$(n, \mathbb{L}(n))$:

1: **while** an $S4$-local, terminological or A^T-rule, except for the $\exists_b$-rule, is applicable to $\mathbb{L}(n)$ **do**
2: apply the rule (if it is a $\sqcup$-rule, non-deterministically pick one choice), add the resulting constraints to $\mathbb{L}(n)$
3: **end while**
4: **if** $\mathbb{L}(n)$ contains a clash **then**
5: **return** "not satisfiable"
6: **end if**
7: $\mathbf{E}(n) := \{a : \exists R.C \mid a : \exists R.C \in \mathbb{L}(n)$ and there is no $b \in \mathcal{O}_{\mathbb{G}}$ s.t. $(a, b) : R, b : C \in \mathbb{L}(n)\}$
8: $\mathbf{D}(n) := \{a : \Diamond_i C \mid a : \Diamond_i C \in \mathbb{L}(n)$ is not blocked and $a : C \notin \mathbb{L}(n)\}$
9: **while** $\mathbf{E}(n) \neq \emptyset$ **do**
10: pick one $a : \exists R.C \in \mathbf{E}(n)$
11: **if** $a : \exists R.C$ is blocked by an individual x **then**
12: $\mathbb{L}^x(n) := \{(a, x) : R, x : C\} \cup \{x : D \mid x : D \in \mathbb{L}(n)\}$ and $\mathbb{L}(n) := \mathbb{L}(n) \setminus \mathbb{L}^x(n)$
13: **else**
14: Let $\mathbb{L}^x(n) := \{(a, x) : R, x : C\}$ where x is fresh
15: **end if**
16: **while** an $S4$-local, terminological or A^T-rule, except for the $\exists_b$-rule, is applicable to $\mathbb{L}(n) \cup \mathbb{L}^x(n)$ **do**
17: apply the rule (if it is a $\sqcup$-rule, non-deterministically pick one choice), add the resulting constraint to $\mathbb{L}^x(n)$
18: **end while**
19: **if** $S4$-SAT$(n, \mathbb{L}^x(n))$ = "not satisfiable" **then**
20: **return** "not satisfiable"
21: **end if**
22: discard $\mathbb{L}^x(n)$
23: $\mathbf{E}(n) := \mathbf{E}(n) \setminus \{a : \exists R.C\}$
24: **end while**
25: **while** $\mathbf{D}(n) \neq \emptyset$ **do**
26: pick one $a : \Diamond_i C \in \mathbf{D}(n)$, create a new constraint system $\mathbb{L}(n')$ let $\mathbb{L}(n') := \{a : C\}$ and $\mathbb{L}(n, n') := \{i\}$
27: **while** the $\Box$- or A^4-rule is applicable to $\mathbb{L}(n)$ **do**
28: apply the rule to $\mathbb{L}(n)$, add corresponding constraints to $\mathbb{L}(n')$
29: **end while**
30: **if** $S4$-SAT$(n', \mathbb{L}(n'))$ = "not satisfiable" **then**
31: **return** "not satisfiable"
32: **end if**
33: discard $\mathbb{L}(n')$
34: $\mathbf{D}(n) := \mathbf{D}(n) \setminus \{a : \Diamond_i C\}$
35: **end while**
36: **return** "satisfiable"

- Lines 10-15 in $S4$-SAT implements the $\exists_b$-rule which corresponds to Line 10 in SAT (see Algorithm 1).

- In Lines 1 and 16, $S4$-SAT tests for the applicability of the $S4$-local, terminological and A^T rules except for the $\exists_b$-rule instead of local and terminological rules except for the $\exists$-rule in SAT (Lines 1 and 11).

- In Line 8, $S4$-SAT chooses constraints of the form $a : \Diamond_i C \in \mathbb{L}(n)$ only under the condition that $a : \Diamond_i C$ is not blocked and $a : C \notin \mathbb{L}(n)$ whereas SAT chooses constraints of the form $a : \Diamond_i C \in \mathbb{L}(n)$ without any restriction.

- In Line 27, $S4$-SAT tests for the applicability of the A^4-rule in addition to the $\Box$-rule in SAT (Line 22).

It is clear that these changes do not affect the space requirements of $\mathcal{ALCS}4_m$-SAT. It follows that the tableau algorithm Λ_{S4} can be implemented in PSPACE.

2.7 Summary and Discussion

In this chapter we studied $\mathcal{ALCK}_m$ and $\mathcal{ALCS}4_m$, knowledge representation languages obtained by augmenting $\mathcal{ALC}$ with modal operators of the basic multi-modal logics K_m and $S4_m$. The resulting logics allow us to represent and reason about the knowledge of multiple experts. We developed sound and complete tableau algorithms $\Lambda_{\mathcal{K}}$ and Λ_{S4} for answering queries w.r.t. corresponding knowledge bases with acyclic TBoxes.

Instead of general concept inclusions allowed in $\mathbf{K}_{\mathcal{ALC}}$ [40] which lead to a NEXPTIME algorithm for satisfiability, the acyclicity restriction on the TBoxes is critical to achieving the PSPACE implementations for both algorithms. In particular, the tableau algorithm $\Lambda_{\mathcal{K}}$ does not need any blocking technique to ensure termination. Furthermore, we have introduced expansion rules that have the following features:

- The expansion rules are quite efficient at detecting clashes in the tableau by avoiding addition of concept memberships that are guaranteed not to lead to a clash during tableau expansion. For example, when $\mathbb{L}(n) = \{a : C, a : D\}$ and $A \doteq C \sqcap D \in \mathcal{T}$, we do

not add $a : A$ into $\mathbb{L}(n)$. The design of the terminological expansion rules aims at detecting clashes only when necessary instead of fully expanding the constraint systems. A consequence of this approach is that not all individuals are categorized as being in or out (of the interpretation) of concept names. In this setting, it turns out that a canonical interpretation, defined analogously to [50], is not sufficient to ensure that the TBox definitions are valid in the model. Therefore, we had to introduce a new definition of a canonical Kripke structure for a constraint graph to address this issue (see Definition 2.3.4).

- In the case of Λ_{S4}, we not only have accessibility expansion rules that are designed to syntactically incorporate the properties of $S4$-structures, but also use a blocking technique that guarantees the termination of the algorithm and facilitates the construction of $S4$-models. In $\mathbf{K}_{\mathcal{ALC}}$ [40], since there is no acyclicity restriction on the formulas, in order to prevent creating infinitely many individuals and ensure the termination of the algorithm, each object is used to represent a type, i.e., a set of concepts that an individual belongs to, rather than the individual itself. Thus, two "individuals" that have the same type are deemed the same. This identification can be viewed as a "blocking" of sorts. In Λ_{S4}, the repetition of the same constraint in different constraint systems is caused because of the epistemic property $(e2)$ that is implemented as the A^4-rule. To prevent creating infinitely many individuals and ensure termination, we limit the generation of new entities as follows. The $\exists_b$-rule limits the creation of a new individual by reusing an existing one created in some previous constraint system and the $\Diamond_b$-rule limits the creation of a new constraint system if there is an existing constraint system that can be used as a successor of the current one. With this blocking technique, when Λ_{S4} terminates, the resulting constraint tree is sufficient to detect clashes (see Definition 2.6.5 and Lemma 2.6.6). If the resulting constraint tree is open and complete, the corresponding canonical $S4$-structure can be constructed by adding edges to the constraint tree without the need to change any constraint system.

The implementations of the tableau algorithms $\Lambda_{\mathcal{K}}$ and Λ_{S4} trace a constraint tree one

path at a time, and within each (possibly auxiliary) constraint system the algorithms deal with constraints about the same "freshly" chosen individual one at a time, thus lending themselves to PSPACE implementations.

Our PSPACE results for the satisfiability of $\mathcal{ALCK}_m$ and $\mathcal{ALCS}4_m$ extend the result of Schmidt-Schauß and Smolka [2] for the satisfiability and subsumption of $\mathcal{ALC}$ concepts. Baader et al. [57] have recently extended the PSPACE result of [2] to $\mathcal{ALC}$ with transitive and inverse roles. In light of this result, we conjecture that query answering against $\mathcal{SIK}$, obtained by replacing $\mathcal{ALC}$ with $\mathcal{SI}$ ($\mathcal{ALC}$ augmented with transitive and inverse roles), can also be implemented in PSPACE.

Acknowledgment. We thank the anonymous referees and Dr. Uli Sattler for their insightful comments which helped us to significantly improve the work. We also thank George Voutsadakis and Jie bao for helpful discussions and Moshe Vardi for encouraging us to consider incorporating epistemic operators into Description Logic knowledge bases. The work of Vasant Honavar and Giora Slutzki was supported in part by grant #0639230 from the National Science Foundation. Jia Tao was supported in part by a research assistantship from the Center for Computational Intelligence, Learning, and Discovery and in part by a teaching assistantship from the Department of Computer Science at Iowa State University. The work of Vasant Honavar while working at the National Science Foundation was supported by the National Science Foundation. Any opinion, finding, and conclusions contained in this article are those of the authors and do not necessarily reflect the views of the National Science Foundation.

CHAPTER 3. SECRECY-PRESERVING QUERY ANSWERING FOR KNOWLEGDE BASES

Abstract

In this chapter, we investigate the secrecy-preserving query answering (SPQA) problem. Given a knowledge base (KB) and a set of secrets that need to be protected against the querying agent, the SPQA problem aims at designing a secrecy-preserving reasoner that answers queries without compromising secrecy under the OWA. Whenever truthfully answering a query risks compromising secrets, the reasoner is allowed to hide the answer to the query by feigning ignorance, i.e., answering the query as "Unknown". Under the OWA, the querying agent is not able to infer whether an "Unknown" answer to a query is obtained because of the incomplete information in the KB or because secrecy protection is being applied.

A simple solution to this problem is to evaluate every query when it is posed to the KB. Since this approach checks the query history each time a query is posed, when the query history gets longer, the response time to a query gets longer as well and hence the approach becomes less and less attractive over time. In our approach, we maintain a secrecy system that is initialized before any query is posed. This system contains a set of assertions that can be inferred from the KB and a secrecy envelope that is used to protect the secrets, both restricted to a finite set of expressions. When a query that was not evaluated during the prequery stage is posed, the secrecy system will be expanded accordingly. The secrecy envelope contains a set of assertions that "disrupt" all proofs of secrets so that none of the secrets can be deduced by the information outside the envelope. To answer queries as informatively as possible, the envelope should be as small as possible. Unfortunately, computing a minimum envelope may be NP-complete. However, we could compute a tight envelope which is irredundant in the sense

that removing any information from it will leave the secrecy set vulnerable.

We provide a general framework for answering queries while preserving secrecy and then study query answering against $\mathcal{EL}$ knowledge bases. Given an $\mathcal{EL}$ knowledge base Σ and a set $\mathbb{S}$ of secrets, we compute an envelope $\mathbb{E}^f \supseteq \mathbb{S}$ by inverting the $\mathcal{EL}$-tableau expansion rules. We then provide two algorithms for computing tight envelopes, a naive approach and an optimized one. We compare the complexity of the two approaches, including experimental results. With the precomputed tight envelope, the answer to a query q posed to Σ will be censored if q can be deduced from Σ and $q \in \mathbb{E}^f$. Our approach allows more flexible information sharing than is possible with traditional access control mechanisms.

3.1 Introduction

The rapid expansion of the WWW and the widespread use of distributed databases and networked information systems offer unprecedented opportunities for productive interaction and collaboration among individuals and organizations in virtually every area of human endeavor. However, the need to share information has to be balanced against the need to protect private, confidential and otherwise sensitive information. The following example illustrates one such scenario.

Example 3.1.1 ***(Healthcare)*** *(a simplified version adapted from [58]): Suppose that Jane's mother Jill had breast cancer. Dr. James, Jane's physician, who is aware of Jane's family history, concludes that Jane has a significant risk of developing breast cancer. He asks her to undergo genetic screening for BRCA1 mutation (which is linked to an increased risk of breast cancer) to determine the extent to which Jane is at risk of developing breast cancer. Suppose Jane tests positive for BRCA1 mutation. Dr. James proceeds to prescribe her a certain drug that he knows is effective at reducing the breast cancer risk for patients with BRCA1 mutation. Jane purchases the medications from her pharmacy and wants to get reimbursed for the cost of her prescription by her insurance company. If her insurance company finds out that she has tested positive for BRCA1 mutation or that she has been prescribed certain drug(s) for breast cancer, Jane risks losing her health insurance. In this setting, the knowledge base (KB) needs to be able to certify to the insurance company that Jane qualifies for reimbursement for a drug that is covered by her insurance policy without revealing the fact that she is on such drugs.* ■

The preceding example illustrates the need for algorithms that can, given a knowledge base Σ and a set $\mathbb{S}$ of secrets (perhaps specified using some secrecy policy[1]), answer queries against Σ, using secrets in the deduction process and providing informative answers, whenever it is possible to do so without compromising confidentiality. In this chapter, we investigate a simplified version of the SPQA problem which consists only one single querying agent. Note that multiple querying agents can be treated as a single querying agent whenever communication between

[1]Upon choosing an underlying language to express the information in the KB, a mechanism is needed to transform the secrecy policies into secrets expressed by the chosen language. Such transformations are beyond the scope of this paper.

the agents is unrestricted. Given a KB Σ and a finite set $\mathbb{S}$ of secrets, called a *secrecy set*, that need to be protected against the querying agent, the secrecy-preserving query answering problem aims at designing a *secrecy-preserving* reasoner that answers queries without revealing any secrets. We provide a general framework of how to answer queries without compromising $\mathbb{S}$. In general, the answer to a query q against a KB Σ can be "Yes" (q can be inferred from Σ), "No" ($\neg q$ can be inferred from Σ) or "Unknown" (e.g., because of the incompleteness of Σ). We assume a cooperative rather than adversarial scenarios in which the KB does not *lie*. However, whenever truthfully answering a query risks compromising secrets in $\mathbb{S}$, the reasoner associated with the KB is allowed to hide the answer to the query by feigning ignorance, i.e., answering the query as "Unknown".

One way to answer queries while preserving secrecy is to evaluate every query when it is posed to the KB. If the truthful answer to the query together the query history compromises some secret, an "Unknown" will be retrieved. Otherwise, the query will be faithfully answered. This strategy is called a *lazy evaluation*. Since this approach checks the query history every time a query is posed, when the history gets larger, the response time gets longer, and therefore, the approach is getting less and less appealing.

Let Σ^+ denote the set of all assertions that can be inferred from Σ. Given a finite secrecy set $\mathbb{S} \subseteq \Sigma^+$, it is clear that answers to queries in $\mathbb{S}$ will be "Unknown". However, since truthful answers to certain queries that are not in $\mathbb{S}$ may reveal information in $\mathbb{S}$, protecting just $\mathbb{S}$ is not sufficient. It follows that we must protect a superset $\mathbb{E}$ of $\mathbb{S}$ ($\mathbb{S} \subseteq \mathbb{E} \subseteq \Sigma^+$), which we call a *secrecy envelope* for $\mathbb{S}$, such that the querying agent who has no access to the envelope will not be able to deduce any information in $\mathbb{S}$. It is easy to see that a secrecy envelope always exists. For instance, $\Sigma^+ \setminus$ {tautologies} constitutes an envelope for any secrecy set $\mathbb{S} \subseteq \Sigma^+$. A key challenge is to *develop strategies that can be used by the KB to respond to queries as informatively as possible (i.e., using an envelope that is as small as possible) without compromising secrets that the KB is obliged to protect.* Unfortunately, it turns out that given a language, computing a minimum envelope may be NP-hard (see Section 3.5.1). Therefore, we aim at computing an envelope that is *tight* in the sense that removing any information from it will leave the secrecy set $\mathbb{S}$ vulnerable. In general, if an envelope is finite, after it is obtained,

for each element in the envelope, we could test whether it is necessary to protect the secrets. If it is not, it will be removed. After all the elements in the envelope are tested, a tight envelope is obtained. Since computing tight envelopes is an optimization problem, depending on the underlying language, algorithms may be designed to directly build an envelope that is tight.

We apply this approach to $\mathcal{EL}$ [25], which is one of the simpler Description Logics (DLs) that is both computationally tractable [26, 27, 28] and practically useful [25, 29]. For example, the medical ontology SNOMED CT [30] and large parts of the medical ontology GALEN [31] can be expressed in $\mathcal{EL}$. We provide algorithms to answer queries against an $\mathcal{EL}$ knowledge base that use, but do not reveal, the information that is designated as secret.

In the case of $\mathcal{EL}$, a KB contains an ABox $\mathcal{A}$, a finite set of assertions and a TBox $\mathcal{T}$, a finite set of subsumptions. Given an $\mathcal{EL}$-KB $\Sigma = \langle \mathcal{A}, \mathcal{T} \rangle$ and a secrecy set $\mathbb{S}$, let $SubE$ be a finite set of expressions, initialized as the set of expressions occurring in Σ or $\mathbb{S}$ and some of their subexpressions (see Section 3.4.1). We shall maintain a *secrecy maintenance system* which we will initialize before any query is posed. It consists of $SubE$, $\mathcal{A}^f$, $\mathcal{T}^f$ and $\mathbb{E}^f$ where $\mathcal{A}^f$ is a set of assertions deducible from Σ, $\mathcal{T}^f$ is the set of subsumptions deducible from Σ, and $\mathbb{E}^f \subseteq \mathcal{A}^f$ is an envelope for $\mathbb{S}$, all restricted to $SubE$. Envelopes restricted in this fashion will be termed *partial envelopes*. The answer to a query $q \in \mathcal{A}^f$ is censored whenever $q \in \mathbb{E}^f$. When a query q whose expression is not in $SubE$ is posed, the secrecy maintenance system needs to be expanded before q can be answered. This includes updating $SubE$, $\mathcal{A}^f$, $\mathcal{T}^f$ and $\mathbb{E}^f$.

We assume that $\mathcal{T}^f$ is precomputed (see [59] for more details). The ABox $\mathcal{A}^f$ is computed using the $\mathcal{EL}$ tableau expansion rules. To compute $\mathbb{E}^f$, we introduce the following idea. From each original expansion rule, we construct a corresponding *inverse expansion rule*. We show that the inverted system of expansion rules generates an envelope for $\mathbb{S}$. To the best of our knowledge, the idea of constructing a secrecy envelope by inverting the tableau expansion rules is novel.

Since computing a minimum envelope in $\mathcal{EL}$ is NP-hard (see Section 3.5.1), instead of computing a minimum envelope, we provide two polynomial time algorithms (one called the *naive* and the other *optimized*) for computing tight envelopes. The naive algorithm first computes an

envelope, then tests every assertion in the envelope whether or not it is redundant, and removes those that are. The optimized algorithm guides the process as it builds up a tight envelope. It avoids much unnecessary work of first putting an assertion in the envelope and then removing it because of redundancy. Even though both approaches run in polynomial time, we show that the optimized algorithm indeed is more efficient for applications whenever the sizes of the TBox and the secrecy set are much smaller than that of the ABox.

The rest of the chapter is organized as follows. Section 3.2 provides the general framework for the SPQA problem. From Section 3.3 to Section 3.6, we study the SPQA problem against $\mathcal{EL}$ knowledge bases. In more detail, Section 3.3 introduces the $\mathcal{EL}$ secrecy-preserving framework. Section 3.4 initializes the $\mathcal{EL}$ secrecy maintenance system. We provide a tableau algorithm for computing $\mathcal{A}^f$ and a tableau algorithm for computing $\mathbb{E}^f$. In Section 3.5, we show that computing a minimum envelope in $\mathcal{EL}$ is NP-complete and give an optimized algorithm for computing tight envelopes. Section 3.6 discusses how to retrieve answers to queries for an $\mathcal{EL}$ KB. Specifically, when a query is not in the current secrecy maintenance system, $\mathcal{A}^f$ and $\mathbb{E}^f$ need to be expanded. We show that such expansion will still lead to a tight envelope provided we start with a tight envelope. Section 3.7 contains description of related work and we conclude in Section 3.8.

3.2 General Framework of Secrecy-preserving Query Answering

3.2.1 Preliminaries

Let $\mathcal{L}$ be some appropriate (logic-based) description language which, for simplicity of notation, we also view as a set of sentences (viz., all sentences expressible in the language $\mathcal{L}$). Let $\models_{\mathcal{L}}$ (or, just $\models$) be a Tarski-style semantic entailment for $\mathcal{L}$ and suppose that $\vdash_{\mathcal{L}}$ (or, just $\vdash$) is an inference system for the language $\mathcal{L}$ that is sound and complete with respect to $\models$.

For $\Gamma \subseteq \mathcal{L}$, we write $\Gamma^+ = \{\alpha \mid \Gamma \vdash_{\mathcal{L}} \alpha\}$ for the *inferential closure* of a set of formulas Γ and we say that $\Gamma \subseteq \mathcal{L}$ is *inferentially closed* if $\Gamma^+ = \Gamma$. We also write $\neg\Gamma = \{\neg\alpha \mid \alpha \in \Gamma\}$. A formula $\alpha \in \mathcal{L}$ is a *tautology* if $\models \alpha$. The set of all tautologies will be denoted by $\mathcal{T}_{aut}$.

Definition 3.2.1 *A* knowledge base *(abbreviated, KB) over $\mathcal{L}$ is a triple $\mathcal{K} \triangleq \langle \Sigma, \mathcal{Q}, \Omega \rangle$ where*

- Σ *is a consistent finite subset* $\Sigma \subseteq_f \mathcal{L}$. *It represents the information that is explicitly stored in* $\mathcal{K}$. Σ^+ *represents all the assertions that the KB can infer. In the sequel, we shall refer to both* $\mathcal{K}$ *and* Σ *as a knowledge base (KB).*

- $\mathcal{Q}$, *the* query space *of* $\mathcal{K}$, *is a subset:* $\Sigma^+ \subseteq \mathcal{Q} \subseteq \mathcal{L}$ *representing the set of all queries that can be "legally" posed against* $\mathcal{K}$. *We do not insist that* $\mathcal{Q} = \mathcal{L}$ *which allows us to account for possible restrictions that may be imposed on the querying agents.*

- Ω, *is the* answer space. *In most cases* $\Omega = \{Y, N, U\}$ *(for "Yes", "No" and "Unknown", respectively). The "classical" answer space is* $\Omega = \{Y, N\}$.

A *secrecy set* $\mathbb{S} \subseteq \Sigma^+$ is a finite set of non-tautological statements that the KB is supposed to protect against the querying agent. Associated with the KB $\mathcal{K}$, there is a *reasoning algorithm*, called a $\mathcal{K}$*-reasoner* $\mathcal{R} : \mathcal{Q} \to \Omega$, which for a query $q \in \mathcal{Q}$ provides an answer $\mathcal{R}(q) \in \Omega$. For any $B \in \Omega = \{Y, N, U\}$, let $\mathcal{Q}_B = \{\alpha \in \mathcal{Q} \mid \mathcal{R}(\alpha) = B\}$, for the set of all queries to which the $\mathcal{K}$-reasoner returns the answer B [2].

Definition 3.2.2 *Given a KB* $\mathcal{K} = \langle \Sigma, \mathcal{Q}, \Omega \rangle$ *and a secrecy set* $\mathbb{S}$, *a secrecy-preserving* $\mathcal{K}$*-reasoner is a* $\mathcal{K}$*-reasoner* $\mathcal{R} : \mathcal{Q} \to \Omega$ *satisfing the following axioms:*

- ***[Yes-Axiom]*** $\mathcal{Q}_Y \subseteq \Sigma^+$;

- ***[No-Axiom]*** $\mathcal{Q}_N = \neg\mathcal{Q}_Y \triangleq \{\neg\alpha \mid \ \alpha \in \mathcal{Q}_Y\}$;

- ***[Closure Axiom]*** $(\mathcal{Q}_Y)^+ = \mathcal{Q}_Y$;

- ***[Secrecy Axiom]*** $(\mathcal{Q}_Y)^+ \cap \mathbb{S} = \emptyset$.

The triple $\langle \mathcal{K}, \mathbb{S}, \mathcal{R} \rangle$ *is called a* secrecy-preserving query answering system *(SQ system).*

The Yes-Axiom ensures that all Y-queries are provable from Σ. The No-Axiom enforces a match between the Y-queries and the N-queries and implies that $\mathcal{Q}_U$ is closed under negation: $\neg\mathcal{Q}_U = \mathcal{Q}_U$. The Closure Axiom requires the set of Y-queries is closed under inference. Finally, the Secrecy Axiom ensures that secrets are "inferentially unreachable" from $\mathcal{Q}_Y$. A trivial

[2] Observe that the set $\{\mathcal{Q}_Y, \mathcal{Q}_N, \mathcal{Q}_U\}$ forms a 3-partition of the query space $\mathcal{Q}$.

example of a secrecy-preserving reasoner is one which for every non-tautological query $\alpha \in \mathcal{Q}$, $\mathcal{R}(\alpha) = U$.

Given a KB $\mathcal{K}$ and a secrecy set $\mathbb{S}$, the *secrecy-preserving querying answering (SPQA) problem* is to design a secrecy-preserving $\mathcal{K}$-reasoner. The secrecy-preserving query answering framework is based on the OWA where what cannot be inferred is considered unknown rather than false. To protect confidential information, we utilize OWA and answer queries that relate to secrets as "Unknown" so that the querying agent is not able to distinguish between (a) the answer to the query is truly unknown, and (b) the answer is being protected for reasons of secrecy. We call the set of queries whose answers are adjusted so as to protect secrets a *secrecy envelope* (or just *envelope*) for the secrecy set. To answer queries as informatively as possible, we aim to have envelopes as small as possible. The idea of envelopes is formalized below in a purely semantic way.

Definition 3.2.3 *Let $\mathcal{K} = \langle \Sigma, \mathcal{Q}, \Omega \rangle$ be a KB and $\mathbb{S}$ a secrecy set on $\mathcal{K}$. A set $\mathbb{E}$, where $\mathbb{S} \subseteq \mathbb{E} \subseteq \Sigma^+ \setminus \mathcal{T}_{aut}$, is called a* secrecy envelope (or envelope) for $\mathbb{S}$ *if for every $\alpha \in \mathbb{E}$, $\Sigma^+ \setminus \mathbb{E} \nvDash \alpha$. A secrecy envelope $\mathbb{E}$ is said to be* tight *if it satisfies an extra condition: for every $\alpha \in \mathbb{E}$, there exists $\beta \in \mathbb{S}$ such that $(\Sigma^+ \setminus \mathbb{E}) \cup \{\alpha\} \vDash \beta$.*

Note that for any secrecy envelope $\mathbb{E}$, its complement is closed under entailment, i.e., $\Sigma^+ \setminus \mathbb{E} \vDash \alpha$ implies $\alpha \in \Sigma^+ \setminus \mathbb{E}$. A secrecy envelope ensures that its contents is "semantically hidden" from the outside. A tight envelope requires, in addition, that the envelope cannot be reduced by changing the answer of a single query in $\mathbb{E}$ without revealing the secret information that needs to be protected against the querying agent. Also note that secrecy envelopes as well as tight envelopes are not unique. For example, $\mathbb{E} = \Sigma^+ \setminus \mathcal{T}_{aut}$ is a secrecy envelope.

The following two theorems present a useful relationship between secrecy envelopes and secrecy-preserving $\mathcal{K}$-reasoners.

Theorem 3.2.4 *Let $\langle \mathcal{K}, \mathbb{S}, \mathcal{R} \rangle$ be an SQ system. Then $\mathbb{E} = \Sigma^+ \setminus \mathcal{Q}_Y$ is an envelope for $\mathbb{S}$.*

Proof We first show that for every $\alpha \in \mathbb{E}$, $\Sigma^+ \setminus \mathbb{E} \nvDash \alpha$. Suppose that there exists $\alpha \in \mathbb{E}$ such that $\Sigma^+ \setminus \mathbb{E} \vDash \alpha$. Since the proof system $\vdash$ is complete w.r.t. $\vDash$, we have $\mathcal{Q}_Y = (\Sigma^+ \setminus \mathbb{E}) \vdash \alpha$. By the Closure Axiom, $\alpha \in \mathcal{Q}_Y$, i.e., $\alpha \in \Sigma^+ \setminus \mathbb{E}$. This contradicts $\alpha \in \mathbb{E}$.

Next we show that $\mathbb{S} \subseteq \mathbb{E}$. If $\alpha \in \mathbb{S} \setminus \mathbb{E}$, then $\alpha \in \Sigma^+ \setminus \mathbb{E} = \mathcal{Q}_Y$, contradicting the Secrecy Axiom. ■

Theorem 3.2.5 *Let $\mathcal{K} = \langle \Sigma, \mathcal{Q}, \Omega \rangle$ be a KB, $\mathbb{S}$ a secrecy set on $\mathcal{K}$ and $\mathbb{E}$ a secrecy envelope for $\mathcal{S}$. Define a function $\mathcal{R}_\mathbb{E}$: $\mathcal{Q} \to \Omega$ by*

$$\mathcal{R}_\mathbb{E}(\alpha) = \begin{cases} Y & \text{if } \alpha \in \Sigma^+ \setminus \mathbb{E}, \\ N & \text{if } \neg\alpha \in \Sigma^+ \setminus \mathbb{E}, \\ U & \text{otherwise.} \end{cases}$$

Then $\mathcal{R}_\mathbb{E}$ is a secrecy-preserving $\mathcal{K}$-reasoner.

Proof We need to show that $\mathcal{R}_\mathbb{E}$ satisfies the four axioms.

- Yes-Axiom: By definition of $\mathcal{R}_\mathbb{E}$, $\mathcal{Q}_Y = \{\alpha \mid \alpha \in \Sigma^+ \setminus \mathbb{E}\} = \Sigma^+ \setminus \mathbb{E} \subseteq \Sigma^+$.
- No-Axiom: By definition of $\mathcal{R}_\mathbb{E}$, $\mathcal{Q}_N = \{\alpha \mid \neg\alpha \in \Sigma^+ \setminus \mathbb{E}\} = \{\neg\alpha \mid \alpha \in \Sigma^+ \setminus \mathbb{E}\} = \neg\mathcal{Q}_Y$.
- Closure Axiom: It suffices to show that $(\mathcal{Q}_Y)^+ \subseteq \mathcal{Q}_Y$. By definition of $\mathcal{R}_\mathbb{E}$, $\mathcal{Q}_Y = \Sigma^+ \setminus \mathbb{E}$. Suppose that $\mathcal{Q}_Y \vdash \alpha$ and $\alpha \notin \mathcal{Q}_Y = \Sigma^+ \setminus \mathbb{E}$. By the soundness of $\vdash$, $\mathcal{Q}_Y \vDash \alpha$ and $\alpha \in \mathbb{E}$. This contradicts the assumption that $\mathbb{E}$ is an envelope.
- Secrecy Axiom: Suppose that $(\mathcal{Q}_Y)^+ \cap \mathbb{S} \neq \emptyset$. Let $\alpha \in \mathbb{S}$ s.t. $\mathcal{Q}_Y \vdash \alpha$. Then, $(\Sigma^+ \setminus \mathbb{E}) \vdash \alpha$, and by the soundness of $\vdash$, $(\Sigma^+ \setminus \mathbb{E}) \vDash \alpha$. This contradicts the assumption that $\mathbb{E}$ is an envelope. ■

3.2.2 A Simple SQ Reasoner – Lazy Evaluation

Given a KB $\mathcal{K}$ and a secrecy set $\mathbb{S}$, a simple approach of answering queries while preserving secrecy is to evaluate each query when it is posed, check whether truthfully answering the query will compromise the secrecy and adjust the answer if necessary. Suppose that queries are posed in an arbitrary but fixed order (history) $H = \{\alpha_k\}_{k=1}^{\infty}$. The history H is said to be *full* if for all $\alpha \in \mathcal{Q}$, α belongs to H. When a query α_k is posed, the algorithm checks whether answers to previous queries combined with the truthful answer to α_k reveal secret information or not.

If they do, α_k will be answered as "Unknown". Otherwise, α_k can be faithfully answered. This approach is greedy in that it makes sure that the secrecy is not compromised each time when answering a query without considering how the current response may constrain answers to future queries. Algorithm 3 implements this lazy approach. For the sake of simplicity of presentation, it omits the actual responses and, instead, concentrates on the construction of the sets $\mathcal{Q}_Y$, $\mathcal{Q}_N$, and $\mathcal{Q}_U$. Note that when the history H is not full, Algorithm 3 does not define a total function $\mathcal{Q} \to \Omega$ and as such it does not satisfy the definition of $\mathcal{K}$-reasoner. The next lemma shows that for full history, Algorithm 3 defines a secrecy-preserving $\mathcal{K}$-reasoner.

Algorithm 3 Lazy Evaluation Algorithm

Input: $\mathcal{K} = \langle \Sigma, \mathcal{Q}, \Omega \rangle$ and $\mathbb{S}$
Initialization: Let $\mathcal{Q}_Y = \mathcal{Q}_N := \emptyset$, $\mathcal{Q}_U := \mathbb{S}$.

while *true* **do**
  input $\alpha \in \mathcal{Q}$
  if $\alpha \notin \mathcal{Q}_Y \cup \mathcal{Q}_N \cup \mathcal{Q}_U$ **then**
    if $\Sigma^+ \cap \{\alpha, \neg\alpha\} = \emptyset$ **then**
      $\mathcal{Q}_U := \mathcal{Q}_U \cup \{\alpha, \neg\alpha\}$
    else
      let $\bar{\alpha} \in \{\alpha, \neg\alpha\}$ such that $\Sigma \vdash \bar{\alpha}$
      if $(\mathcal{Q}_Y \cup \{\bar{\alpha}\})^+ \cap \mathbb{S} \neq \emptyset$ **then**
        $\mathcal{Q}_U := \mathcal{Q}_U \cup \{\alpha, \neg\alpha\}$
      else
        $\mathcal{Q}_Y := \mathcal{Q}_Y \cup \{\bar{\alpha}\}$
        $\mathcal{Q}_N := \mathcal{Q}_N \cup \{\neg\bar{\alpha}\}$
      end if
    end if
  end if
end while

Lemma 3.2.6 *Given a knowledge system $\mathcal{K} = \langle \Sigma, \mathcal{Q}, \Omega \rangle$, a secrecy set $\mathbb{S}$ and a full query history $H = \{\alpha_k\}_{k=1}^{\infty}$, the lazy evaluation algorithm (Algorithm 3) defines a secrecy-preserving $\mathcal{K}$-reasoner.*

Proof It is easy to see that Algorithm 3 satisfies Yes-Axiom and No-Axiom in Definition 3.2.2. To show that the Secrecy Axiom is satisfied, we argue by induction on history of queries. In the pre-query stage, $\mathcal{Q}_Y = \emptyset$ and so, since $\mathcal{T}_{aut} \cap \mathbb{S} = \emptyset$, $(\mathcal{Q}_Y)^+ \cap \mathbb{S} = \emptyset$. For the induction step, suppose that the condition $(\mathcal{Q}_Y)^+ \cap \mathbb{S} = \emptyset$ holds, and consider the next query $\alpha \in \mathcal{Q}$. If

$\Sigma^+ \cap \{\alpha, \neg\alpha\} = \emptyset$, then $\mathcal{Q}_Y$ does not change, and the above condition is maintained. The same holds true if $(\mathcal{Q}_Y \cup \{\bar{\alpha}\})^+ \cap \mathbb{S} \neq \emptyset$. So suppose that $(\mathcal{Q}_Y \cup \{\bar{\alpha}\})^+ \cap \mathbb{S} = \emptyset$. Then $\mathcal{Q}_Y := \mathcal{Q}_Y \cup \{\bar{\alpha}\}$. So with the new value of $\mathcal{Q}_Y$, the same condition holds. Note that whenever a query is assigned to be in $\mathcal{Q}_Y \cup \mathcal{Q}_N \cup \mathcal{Q}_U$, it will not be re-evaluated. Therefore, the property $(\mathcal{Q}_Y)^+ \cap \mathbb{S} = \emptyset$ is an invariant of the algorithm. It follows that the Secrecy Axiom is satisfied.

To show that the Closure Axiom is also satisfied, we assume that there is a query $q \in (\mathcal{Q}_Y)^+$ and $q \notin \mathcal{Q}_Y$. From $q \in (\mathcal{Q}_Y)^+$, we conclude $\Sigma \vdash q$. This means that when q was first queried and evaluated (i.e., when q was not in $\mathcal{Q}_Y \cup \mathcal{Q}_N \cup \mathcal{Q}_U$), Algorithm 3 executes Lines 7-13. Moreover, since $q \notin \mathcal{Q}_Y$, the condition in Line 8 was satisfied. Hence, $(\mathcal{Q}_Y \cup \{\alpha\})^+ \cap \mathbb{S} \neq \emptyset$. This contradicts the invariant of the algorithm. Therefore, $(\mathcal{Q}_Y)^+ \subseteq \mathcal{Q}_Y$. Since, obviously, $\mathcal{Q}_Y \subseteq (\mathcal{Q}_Y)^+$, the Closure Axiom is satisfied. ■

By Theorem 3.2.4, $\mathbb{E} = \Sigma^+ \setminus \mathcal{Q}_Y = \Sigma^+ \cap \mathcal{Q}_U$ where $\mathcal{Q}_Y$ and $\mathcal{Q}_U$ are computed by Algorithm 3 is an envelope. The next theorem shows that $\mathbb{E}$ is actually a tight envelope.

Theorem 3.2.7 *Given an SQ system $\langle \mathcal{K}, \mathcal{S}, \mathcal{R} \rangle$ where $\mathcal{R}$ is defined as in Algorithm 3, every query that can be deduced from Σ and is in $\mathcal{Q}_U$ belongs to a tight envelope.*

Proof By Lemma 3.2.6, for full history, the lazy evaluation algorithm is a secrecy-preserving $\mathcal{K}$-reasoner. It then follows from Theorem 3.2.4 that $\mathbb{E} = \Sigma^+ \setminus \mathcal{Q}_Y = \Sigma^+ \cap \mathcal{Q}_U$ is an envelope where $\mathcal{Q}_Y$ and $\mathcal{Q}_U$ are obtained by Algorithm 3. Suppose that $\mathbb{E}$ is not tight. Then there exists $\alpha \in \mathbb{E} \subseteq \mathcal{Q}_U$ such that for every $\beta \in \mathbb{S}$, $(\Sigma^+ \setminus \mathbb{E}) \cup \{\alpha\} \nvdash \beta$. By the soundness of $\vdash$, we have $(\Sigma^+ \setminus \mathbb{E}) \cup \{\alpha\} \nvdash \beta$, i.e., $\mathcal{Q}_Y \cup \{\alpha\} \nvdash \beta$. This means that $(\mathcal{Q}_Y \cup \{\alpha\})^+ \cap \mathbb{S} = \emptyset$. So when α was queried for the first time, since $\Sigma \vdash \alpha$ and the condition in Line 8 is satisfied, α should have been added to $\mathcal{Q}_Y$. However, this contradicts that $\alpha \in \mathcal{Q}_U$. ■

The lazy evaluation approach is simple, but as the number of queries increases, the set $\mathcal{Q}_Y$ gets larger, checking conditions in Line 8 takes longer time and so answering queries tends to be more time consuming as the system continues to operate. Therefore, the system becomes less and less user-friendly. In the next section, we propose another approach that answers queries by computing envelopes based on the intrinsic properties of secrets.

3.2.3 Computing Envelope

Consider a formula $\alpha \wedge \beta$ in propositional logic. If $\Gamma \models \alpha$ and $\Gamma \models \beta$, then obviously $\Gamma \models \alpha \wedge \beta$, and hence if we need to protect $\alpha \wedge \beta$, it is clear that we must protect at least one of α and β; i.e., if $\alpha \wedge \beta$ is in a secrecy set, at least one of α and β should be in the corresponding envelope. We use a syntactic version of a generalization of this observation. Roughly, the idea is to "disrupt" all proofs of secret information in $\mathbb{S}$. The disrupting formulas will form an envelope for $\mathbb{S}$. Since finding proofs is a syntactic process, to construct an envelope, we invert the inference rules to obtain disrupting formulas (see an example in Section 3.4.3). When an envelope is present, a reasoner can be defined to answer queries while preserving secrecy according to Theorem 3.2.5. In what follows, we formalize these ideas.

Given a language $\mathcal{L}$ and $\alpha \in \mathcal{L}$, we say that a finite set $\Gamma \subseteq_f \mathcal{L}$ is *α-minimal* if $\Gamma \models \alpha$ and for every $\beta \in \Gamma$, $\Gamma \setminus \{\beta\} \not\models \alpha$. Let $\mathcal{F}_\alpha = \{\Gamma \mid \Gamma \text{ is } \alpha\text{-minimal}\}$. If α is to be protected, at least one element in each set in $\mathcal{F}_\alpha$ has to be protected so that α cannot be entailed. Let ϕ_Γ be an arbitrary but fixed element of a given set Γ. Note that, in particular, for $\Gamma \in \mathcal{F}_\alpha$, $\Gamma \models \alpha$, but $\Gamma \setminus \{\phi_\Gamma\} \not\models \alpha$. Also note that for two different sets Γ and Γ', ϕ_Γ and $\phi_{\Gamma'}$ may be same.

Theorem 3.2.8 *Given a secrecy set $\mathbb{S}$, define a sequence of sets where $E_0 = \mathbb{S}$ and $E_{i+1} = \{\phi_\Gamma \mid$ there is $\alpha \in E_i$ and $\Gamma \in \mathcal{F}_\alpha\}$. Let $\mathbb{E} = \bigcup_{i=0}^{\infty} E_i$. Then $\mathbb{E}$ is an envelope for $\mathbb{S}$.*

Proof Suppose that for some $\alpha \in \mathbb{E}$, $\Sigma^+ \setminus \mathbb{E} \models \alpha$. Then for some finite subset $\Gamma \subseteq \Sigma^+ \setminus \mathbb{E}$, $\Gamma \models \alpha$ and Γ is α-minimal; and therefore, $\Gamma \in \mathcal{F}_\alpha$. By the definition of $\mathbb{E}$, suppose $\alpha \in E_i$. Then, $\phi_\Gamma \in \Gamma \cap E_{i+1}$ and so $\Gamma \cap \mathbb{E} \neq \emptyset$, contradicting $\Gamma \subseteq \Sigma^+ \setminus \mathbb{E}$. ■

Once an envelope is computed, queries can be answered according to Theorem 3.2.5 without revealing any secret. Note that $\mathbb{E}$ is not unique and may contain redundant information. We would like to have envelopes as small as possible so that we can answer truthfully as many queries as possible. However, given a language, deciding a minimum secrecy envelope may be NP-complete (see Section 3.5.1). Because of this, we strive to obtain tight envelopes. If an envelope $\mathbb{E}$ is finite, a simple approach is to test for each $\alpha \in \mathbb{E}$, whether or not it could be removed from $\mathbb{E}$ without compromising $\mathbb{S}$. That is, if there does not exist $\beta \in \mathbb{S}$ such that

$(\Sigma^+ \setminus \mathbb{E}) \cup \{\alpha\} \models \beta$, then α is removed from $\mathbb{E}$. After all the elements in the original envelope are tested, one will get a tight envelope. Also note that depending on the order in which elements of $\mathbb{E}$ are tested, different tight envelopes may result. Further optimization techniques may be developed according to the properties of the underlying language $\mathcal{L}$ and its native inference system.

We have presented a general framework for answering queries posed to a knowledge base containing secret information that need to be protected against the querying agent. For the rest of this paper, we take the description language $\mathcal{EL}$ to illustrate how our secrecy-preserving query answering framework can be applied to a specific language.

3.3 $\mathcal{EL}$ Preliminaries

3.3.1 Syntax and Semantics

The non-logical signature of the $\mathcal{EL}$ description language includes three mutually disjoint sets: a set of *concept names* $N_{\mathcal{C}}$, a set of *role names* $N_{\mathcal{R}}$ and a set of *individual names* $N_{\mathcal{O}}$. The syntax of $\mathcal{EL}$ is defined by specifying *expressions* and *formulae*. $\mathcal{EL}$ expressions consist of the set of *role names* $N_{\mathcal{R}}$ and the set of *concepts* $\mathcal{C}$ which is recursively defined as follows:

$C, D \longrightarrow A \mid \top \mid C \sqcap D \mid \exists r.C$

where $A \in N_{\mathcal{C}}$, $\top$ is the *top symbol*, $C, D \in \mathcal{C}$ and $r \in N_{\mathcal{R}}$. In this paper we will consider three kinds of $\mathcal{EL}$ formulae: *assertions* of the form $C(a)$ or $r(a, b)$, *definitions* of the form $A \doteq D$ and *general concept inclusions (GCI)* of the form $C \sqsubseteq D$ where $a, b \in N_{\mathcal{O}}$, $C, D \in \mathcal{C}$, $r \in N_{\mathcal{R}}$ and $A \in N_{\mathcal{C}}$.

The semantics of $\mathcal{EL}$ is defined by using an *interpretation* $\mathcal{I} = \langle \Delta, \cdot^{\mathcal{I}} \rangle$ where Δ is a non-empty domain and $\cdot^{\mathcal{I}}$ is a function that maps each individual name to an element in Δ, $\top$ to Δ, each concept name to a subset of Δ and each role name to a subset of $\Delta \times \Delta$. The interpretation of concept expressions is extended recursively as follows: for all $r \in N_{\mathcal{R}}$ and $C, D \in \mathcal{C}$, $(C \sqcap D)^{\mathcal{I}} = C^{\mathcal{I}} \cap D^{\mathcal{I}}$ and $(\exists r.C)^{\mathcal{I}} = \{a \in \Delta \mid \exists\, b \in \Delta : (a, b) \in r^{\mathcal{I}} \wedge b \in C^{\mathcal{I}}\}$. For a finite set of symbols $N \subset N_{\mathcal{C}} \cup N_{\mathcal{R}} \cup N_{\mathcal{O}}$ and an interpretation $\mathcal{I} = \langle \Delta, \cdot^{\mathcal{I}} \rangle$, we define an *interpretation $\mathcal{I}$ restricted to N* to be $\mathcal{I}_N = \langle \Delta, \cdot^{\mathcal{I}}|_N \rangle$ where $\cdot^{\mathcal{I}}|_N$ denotes the restriction of the

function $\cdot^{\mathcal{I}}$ to N.

Definition 3.3.1 *An* $\mathcal{EL}$-knowledge base *(abbreviated,* $\mathcal{EL}$*-KB) is a triple* $\mathcal{K} \triangleq \langle \Sigma, \mathcal{Q}, \Omega \rangle$ *such that*

- $\Sigma = \langle \mathcal{A}, \mathcal{T} \rangle$ *where* $\mathcal{A}$ *contains a finite non-empty set of assertions over* $\mathcal{EL}$*, named an* ABox*, and* $\mathcal{T}$ *contains a finite set of definitions and GCIs over* $\mathcal{EL}$*, named a* TBox*. As in the general case (see Definition 3.2.1), we shall refer to both* $\mathcal{K}$ *and* Σ *as a knowledge base.*
- $\mathcal{Q}$*, the* query space *of* $\mathcal{K}$ *is a set of assertions.*
- Ω*, is the* answer space*. Since* $\mathcal{EL}$ *does not have negation,* $\Omega = \{Y, U\}$ *(for "Yes" and "Unknown", respectively).*

A TBox $\mathcal{T}$ is *normalized* [60] if $\mathcal{T}$ contains only GCIs each of which is of the following form: $A \sqsubseteq B$, $A_1 \sqcap ... \sqcap A_k \sqsubseteq B$, $A \sqsubseteq \exists r.B$ or $\exists r.A \sqsubseteq B$ where $A, A_i, B \in N_{\mathcal{C}} \cup \{\top\}$ $(i \in \{1, ..., k\})$. It was shown that transforming a TBox into such a normal form can be accomplished in polynomial time [60]. Henceforth, we will assume that all the subsumptions are in normal form. From now on, we shall denote by $\mathcal{T}^+$ the extended transitive closure of the TBox $\mathcal{T}$ (as computed in [59]). For example, if $D \sqsubseteq E \in \mathcal{T}$, then $\exists r.(C \sqcap D) \sqsubseteq \exists r.E \in \mathcal{T}^+$. Similarly, $\mathcal{A}^+$ will denote the inferential closure of Σ restricted to assertions and is referred to as the *assertional closure* of Σ.

A concept C is said to be *atomic* if $C \in N_{\mathcal{C}}$ or $C = \exists r.D$ where $D \in \mathcal{C}$. Note that the concepts on the right-hand side of subsumptions (in normal form) are all atomic. An assertion $C(a)$ is *atomic* if C is atomic. We denote by $N_\Sigma \subseteq N_{\mathcal{C}} \cup N_{\mathcal{R}} \cup N_{\mathcal{O}}$ the set of all the names appearing in Σ and by $\mathcal{O}_\Sigma$ the set of individual names appearing in Σ. Thus, $\mathcal{O}_\Sigma \subset N_{\mathcal{O}} \cap N_\Sigma$ and $N_\Sigma \setminus \mathcal{O}_\Sigma \subset N_{\mathcal{C}} \cup N_{\mathcal{R}}$.

Definition 3.3.2 *Let* $\Sigma = \langle \mathcal{A}, \mathcal{T} \rangle$ *be a knowledge base,* $\mathcal{I} = \langle \Delta, \cdot^{\mathcal{I}} \rangle$ *an interpretation,* $C, D \in \mathcal{C}$*,* $r \in N_{\mathcal{R}}$ *and* $a, b \in N_{\mathcal{O}}$*. We say that* $\mathcal{I}$ satisfies $C(a)$*,* $r(a, b)$*, or* $C \sqsubseteq D$ *if, respectively,* $a^{\mathcal{I}} \in C^{\mathcal{I}}$*,* $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in r^{\mathcal{I}}$*, or* $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$*.* $\mathcal{I}$ *is a* model *of* Σ *if it satisfies all the assertions in* $\mathcal{A}$ *and all the*

GCIs in $\mathcal{T}$. Let α be an assertion or a GCI. We say that Σ entails α*, written as $\Sigma \models \alpha$, if all models of Σ satisfy α.*

3.3.2 The Secrecy-preserving Query Answering Problem in $\mathcal{EL}$

The scenario described in Example 3.1.1 can be more formally specified as an $\mathcal{EL}$-KB.

Example 3.3.3 *(Example 3.1.1, cont.) Let $\Sigma_1 = \langle \mathcal{A}_1, \mathcal{T}_1 \rangle$ be a KB that contains information about the patients, their health history, the prescriptions that they get from the physicians and their insurance information.*

1. $\exists is\_child.A \sqsubseteq CancerRisk$
2. $HasMutBRCA1 \sqsubseteq \exists has\_pres.CancerDrug$
3. $\exists has\_pres.CancerDrug \sqsubseteq CancerRisk$
4. $\exists has\_pres.CoveredDrug \sqsubseteq Reimburse$
5. $CancerDrug \sqsubseteq CoveredDrug$
6. $A \sqsubseteq Woman$
7. $A \sqsubseteq HasCancer$
8. $Woman \sqcap HasCancer \sqsubseteq A$
9. $Woman(Jill)$
10. $HasCancer(Jill)$
11. $is\_child(Jane, Jill)$
12. $HasMutBRCA1(Jane)$

The GCIs 1-8 form a subset of $\mathcal{T}_1$ (in normal form) and the assertions 9-12 form a subset of $\mathcal{A}_1$. The secrecy set is $\mathbb{S}_1 = \{CancerRisk(Jane)\}$. In order for Jane to get reimbursed, when the query Reimburse(Jane) is posed to the KB, the answer should be "Yes". To protect Jane's privacy, the query CancerRisk(Jane) should be answered "Unknown". ■

Given an $\mathcal{EL}$-KB and a finite secrecy set $\mathbb{S}$, the basic goal is to answer queries as informatively as possible while preserving secrecy. It is obvious that protecting just secrets in $\mathbb{S}$ is not enough to preserve secrecy. For instance, in Example 3.3.3, in order to protect Jane's privacy, the query CancerRisk(Jane) should be answered "Unknown". However, by keeping just CancerRisk(Jane) secret, the fact that Jane has cancer risk can still be inferred from statements 12, 2 and 3.

Since the SPQA problem is to design a secrecy-preserving $\mathcal{K}$-reasoner, it follows from the discussion in Section 3.2 that we could either (a) use the lazy evaluation outlined in Section 3.2.2, or (b) be somewhat more proactive and precompute a (partial) envelope to answer queries

as per Theorem 3.2.5. In what follows, we focus on providing algorithms to compute envelopes for the given knowledge system and secrecy set.

In the following, we make explicit some assumptions about the $\mathcal{EL}$-SPQA problem.

- The $\mathcal{EL}$ reasoner $\mathcal{R}$ has an underlying inference system $\vdash_{\mathcal{EL}}$ that is complete in the sense that it can infer (prove) an assertion α whenever $\Sigma \vDash \alpha$ (such inference systems exist, see [2], and Section 3.4.2). Specifically, the inference system $\vdash$ should be able to decide:
  - whether a given subsumption follows from the TBox $\mathcal{T}$. Since a sound and complete proof system dealing with this issue is given in [59], we shall use it without any further ado.
  - whether any given assertion can be inferred from Σ.
- The querying agent has the same inference capacity as $\mathcal{R}$. Since we assume that $\vdash_{\mathcal{EL}}$ is complete, this is not a restriction. The querying agent may log the history of all the answers to its queries and draw conclusions from it.
- The querying agent has full access to the TBox $\mathcal{T}$. This implies that the querying agent, given any subsumption, can decide whether or not it follows from $\mathcal{T}$.
- Queries in the query space $\mathcal{Q}$ are of the form $C(a)$ or $r(a, b)$. We assume that the querying agent has computational access only to the signature of the knowledge base, i.e., all its queries are over N_Σ.
- A secrecy set $\mathbb{S}$ is a finite set of assertions over N_Σ.

The reason for the last two assumptions is that, if an assertion α contains symbols not in N_Σ, based on OWA, the answer to α is "Unknown" and asking such queries or protecting such queries is of no interest. We stress that the secrecy set $\mathbb{S}$ is not assumed to be a subset of the ABox $\mathcal{A}$. However, the individual names that occur in $\mathbb{S}$ do belong to $\mathcal{O}_\Sigma$.

3.4 Initializing $\mathcal{EL}$ SQ System

In this section, we discuss the initialization of the secrecy maintenance system in detail. Since the set of formulas that can be deduced from a given KB as well as an envelope may be infinite, we compute a finite set $SubE$ of some subexpressions of all concepts and roles appearing in $\Sigma = \langle \mathcal{A}, \mathcal{T} \rangle$ or $\mathbb{S}$. Then we restrict all the relevant sets to $SubE$. For example, the assertional closure of Σ is denoted by $\mathcal{A}^+$, and its restriction to $SubE$ will be denoted by $\mathcal{A}^f$. Similarly, $\mathcal{T}^f$ will denote the restriction of $\mathcal{T}^+$ to $SubE$. The set $\mathcal{A}^f$ is called the *assertional closure* of Σ (restricted to $SubE$). Obviously, since $\mathbb{S} \subseteq \Sigma^+$ is finite and all the concepts and roles in $\mathbb{S}$ are also in $SubE$, $\mathbb{S} \subseteq \mathcal{A}^f$.

In [2], authors gave a sound and complete inference system for $\mathcal{ALC}$. We have chosen another inference system (see Section 3.4.2) which is fashioned explicitly for $\mathcal{EL}$ and is more amenable to rule inversion, a technique we shall use to construct envelopes (see Section 3.4.3). To compute envelopes, we apply the idea of "disrupting" proofs discussed in Section 3.2.3; the inverted rules help us find formulas that will disrupt all the proofs of secrets in $\mathbb{S}$.

3.4.1 Computing $SubE$

In the prequery stage, $SubE$ is the set of certain subexpressions of all the concepts and roles appearing in Σ or $\mathbb{S}$, and it is defined formally as follows:

- if $C(a) \in \mathcal{A} \cup \mathbb{S}$, then $C \in SubE$; if $r(a, b) \in \mathcal{A} \cup \mathbb{S}$, then $r \in SubE$;
- if $C \sqsubseteq D \in \mathcal{T}$, then $\{C, D\} \subseteq SubE$;
- if $C_1 \sqcap \cdots \sqcap C_k \in SubE$ (all C_i are atomic), then $C_i \in SubE$ $(1 \leq i \leq k)$;
- if $\exists r.C \in SubE$, then $\{r, C\} \subseteq SubE$;

Note that $SubE$ does not contain all the subexpressions of concepts appearing in Σ or $\mathbb{S}$. For example, if $C_1 \sqcap C_2 \sqcap C_3(a) \in \mathcal{A}$, then $\{C_1, C_2, C_3, C_1 \sqcap C_2 \sqcap C_3\} \subseteq SubE$. However, $C_1 \sqcap C_2 \notin SubE$ unless it is added to $SubE$ in another way, for example, if $\exists r.(C_1 \sqcap C_2) \in SubE$ or $C_1 \sqcap C_2 \sqsubseteq D \in \mathcal{T}$.

When a query $C(a)$ comes along with $C \notin SubE$, it (and, possibly, some of its subconcepts) will be added into $SubE$. As such, following an initial, pre-query phase, the system is built up gradually depending on the history of queries. Also note that the initial size of $SubE$ is linear in the size of the KB Σ plus the size of the secrecy set $\mathbb{S}$.

Example 3.4.1 *(Example 3.1.1-3.4.7, cont.) The set of subexpressions of* Σ_1 *and* $\mathbb{S}_1$ *is:*

$SubE_1 = \{\exists is\_child.A,\ CancerRisk,\ HasMutBRCA1,\ \exists has\_pres.CancerDrug,$
$\exists has\_pres.CoveredDrug,\ Reimburse,\ CancerDrug,\ CoveredDrug,\ A,$
$Woman \sqcap HasCancer,\ Woman,\ HasCancer,\ is\_child,\ has\_pres\}.$ ■

3.4.2 Computing $\mathcal{A}^f$

The ABox $\mathcal{A}^f$ is initialized as $\mathcal{A}$ and expanded by recursively applying *assertion expansion rules* listed in Figure 3.1. We say that $\mathcal{A}^f$ is *assertionally closed*, or that it is an *assertional closure of* Σ, w.r.t. $SubE$, if no assertion expansion rule is applicable. The set of all the individual names appearing in $\mathcal{A}^f$ is denoted by $\mathcal{O}^f$. It is initialized as $\mathcal{O}_\Sigma$. New individuals are *introduced* with the application of the $\exists_2^{\mathcal{A}}$-rule. We stipulate that all individuals in $\mathcal{O}_\Sigma$ are introduced at the same time and before any individual in $\mathcal{O}^f \setminus \mathcal{O}_\Sigma$. An individual a is said to be *fresh* (at a particular time during the expansion process) if $a \in N_\mathcal{O} \setminus \mathcal{O}^f$ (at that time). An individual $a \in \mathcal{O}^f \setminus \mathcal{O}_\Sigma$ is *blocked* by an individual $b \in \mathcal{O}^f$ if b was introduced earlier than a (during the expansion process), and $\{C \mid C(a) \in \mathcal{A}^f\} \subseteq \{C' \mid C'(b) \in \mathcal{A}^f\}$.

$\sqsubseteq^{\mathcal{A}}$ -rule:	if $C \sqsubseteq D \in \mathcal{T}^f, C(a) \in \mathcal{A}^f$ and $D(a) \notin \mathcal{A}^f$, then $\mathcal{A}^f := \mathcal{A}^f \cup \{D(a)\}$;
$\sqcap^{\mathcal{A}}$ -rule:	if $C_1 \sqcap \cdots \sqcap C_k \in SubE, \{C_1(a), ..., C_k(a)\} \subseteq \mathcal{A}^f$ and $C_1 \sqcap \cdots \sqcap C_k(a) \notin \mathcal{A}^f$, then $\mathcal{A}^f := \mathcal{A}^f \cup \{C_1 \sqcap \cdots \sqcap C_k(a)\}$;
$\exists_1^{\mathcal{A}}$ -rule:	if $\exists r.C \in SubE, \{r(a,b), C(b)\} \subseteq \mathcal{A}^f$ and $\exists r.C(a) \notin \mathcal{A}^f$, then $\mathcal{A}^f := \mathcal{A}^f \cup \{\exists r.C(a)\}$;
$\exists_2^{\mathcal{A}}$ -rule:	if none of the $\sqsubseteq^{\mathcal{A}}$-, $\sqcap^{\mathcal{A}}$- and $\exists_1^{\mathcal{A}}$-rules are applicable, and 1. $\exists r.C(a) \in \mathcal{A}^f$, 2. a is not blocked, and 3. for all $b \in \mathcal{O}^f, \{r(a,b), C(b)\} \nsubseteq \mathcal{A}^f$, then $\mathcal{A}^f := \mathcal{A}^f \cup \{r(a,c), C(c)\}$ where c is fresh, and $\mathcal{O}^f := \mathcal{O}^f \cup \{c\}$.

Figure 3.1 Assertion Expansion Rules

We denote by Λ the tableau algorithm which nondeterministically applies assertion expansion rules until no further applications are possible. Since each expansion rule can be applied polynomially many times (in the size of $SubE$), the computation of $\mathcal{A}^f$ can be completed in polynomial time. When an execution of Λ terminates, we have an assertionally closed ABox $\mathcal{A}^f$. The following are observations about the individuals occurring in $\mathcal{A}^f$.

f_1. For any role assertion $r(a,b) \in \mathcal{A}^f$, $b \in \mathcal{O}_\Sigma \Rightarrow a \in \mathcal{O}_\Sigma$ and $r(a,b) \in \mathcal{A}$.

f_2. Once an individual a is found to be blocked by an individual a' while attempting to apply the $\exists_2^{\mathcal{A}}$-rule (item 2), the set $\{C \mid C(a) \in \mathcal{A}^f\}$ will remain fixed and will not change for the remainder of the tableau algorithm Λ. In particular, a will always remain blocked by a'.

f_3. For all $a \in \mathcal{O}^f \setminus \mathcal{O}_\Sigma$, if a was introduced via an application of the $\exists_2^{\mathcal{A}}$-rule to an assertion $\exists r.C(b)$, then for any $D \in \{D' \mid D'(a) \in \mathcal{A}^f\}$, $C \sqsubseteq D$. The proof of this observation can be found in Appedix A.2.1.

Remark. Without blocking, the $\exists_2^{\mathcal{A}}$-rule may introduce infinitely many individuals for the same existential concept expression as in this example: $A \sqsubseteq \exists r.A \in \mathcal{T}^f$ and $A(a) \in \mathcal{A}$. With blocking and the condition that the $\exists_2^{\mathcal{A}}$-rule has to be considered only when no other assertion expansion rule is applicable, for each existential concept assertion $\exists s.B(d) \in \mathcal{A}^f$ with $d \in \mathcal{O}^f \setminus \mathcal{O}_\Sigma$, the expansion process will generate at most one new individual (besides d). A typical example is the following. Suppose that we have $\{A \sqsubseteq B, B \sqsubseteq C, C \sqsubseteq \exists r.A\} \subseteq \mathcal{T}^f$ and $B(a) \in \mathcal{A}$. From $B(a) \in \mathcal{A}$, we conclude that $C(a), \exists r.A(a) \in \mathcal{A}^f$. Since $a \in \mathcal{O}_\Sigma$, a is not blocked and the $\exists_2^{\mathcal{A}}$-rule is applicable to $\exists r.A(a)$. Suppose that a fresh individual b is introduced and $r(a,b), A(b)$ are added to $\mathcal{A}^f$. As a consequence, $B(b)$, $C(b)$, and $\exists r.A(b)$ are also added into $\mathcal{A}^f$. There are two cases: (1) If b is blocked by a, by observation (f_2), the $\exists_2^{\mathcal{A}}$-rule will never be applicable to $\exists r.A(b)$, or (2) If b is not blocked by a (e.g., if $A(a) \notin \mathcal{A}^f$), the $\exists_2^{\mathcal{A}}$-rule is applicable to $\exists r.A(b)$ and a new individual c is introduced which will be blocked by b (since the set of concepts that c belongs to is a subset of concepts that b belongs to and b was introduced earlier than c).

Example 3.4.2 *Continuing Examples 3.1.1-3.4.1 with the KB* $\Sigma_1 = \langle \mathcal{A}_1, \mathcal{T}_1 \rangle$ *and secrecy set* $\mathbb{S}_1$*, by applying* Λ*, we can obtain the assertional closure of* Σ_1*, denoted by* $\mathcal{A}_1^f$*, as follows.*

$\mathcal{A}_1^f = \mathcal{A}_1 \cup \{$ *Woman*$\sqcap$*HasCancer(Jill), A(Jill),* $\exists$*is_child.A(Jane), CancerRisk(Jane),*
$\exists$*has_pres.CancerDrug(Jane), has_pres(Jane, a), CancerDrug(a), CoveredDrug(a),*
$\exists$*has_pres.CoveredDrug(Jane), Reimburse(Jane)* $\}$. ■

Let $\mathcal{I}^1 = \langle \Delta, \cdot^{\mathcal{I}^1} \rangle$, $\mathcal{I}^2 = \langle \Delta, \cdot^{\mathcal{I}^2} \rangle$ be two interpretations, and $N_2 \subseteq N_1$ be two finite subsets of $N_\mathcal{C} \cup N_\mathcal{R} \cup N_\mathcal{O}$ such that $N_1 \setminus N_2 \subseteq N_\mathcal{O}$. Then, $\mathcal{I}^1_{N_1} = \langle \Delta, \cdot^{\mathcal{I}^1}|_{N_1} \rangle$ is a *semantic extension* of $\mathcal{I}^2_{N_2} = \langle \Delta, \cdot^{\mathcal{I}^2}|_{N_2} \rangle$ if $(\cdot^{\mathcal{I}^2}|_{N_2}) = (\cdot^{\mathcal{I}^1}|_{N_2})$. The following theorem shows the soundness of the assertion expansion rules. The proof is in Appendix A.2.2.

Theorem 3.4.3 *(Soundness of the Assertion Expansion Rules) Let* $\mathcal{A}^f$ *be an assertionally closed ABox obtained from* Σ *by applying the tableau algorithm* Λ*. For any* $C \in \mathcal{C} \cap SubE$ *and any* $a \in \mathcal{O}^f$*, if* $C(a) \in \mathcal{A}^f$*, then for every model* $\mathcal{I} = \langle \Delta, \cdot^{\mathcal{I}} \rangle$ *of* Σ*, there is a semantic extension of* $\mathcal{I}_{N_\Sigma}$ *that satisfies* $C(a)$*. In particular, if* $a \in \mathcal{O}_\Sigma$*, then* $\Sigma \models C(a)$*.*

The following Theorem shows the completeness of the tableau algorithm Λ and its proof is in Appendix A.2.3.

Theorem 3.4.4 *(Completeness) Let SubE be the set of subexpressions obtained from a KB* $\Sigma = \langle \mathcal{A}, \mathcal{T} \rangle$ *and a finite set of assertions S (see Section 3.4.1). Let* $\mathcal{A}^f$ *be an assertionally closed ABox obtained from* Σ *by applying the tableau algorithm* Λ*. Then for every concept* $C \in \mathcal{C} \cap SubE$ *and for every individual* $a \in \mathcal{O}^f$*,* $\Sigma \models C(a) \Rightarrow C(a) \in \mathcal{A}^f$*.*

Note that the $\exists_2^\mathcal{A}$-rule helps us to build a model for the assertional closure $\mathcal{A}^f$. The $\sqsubseteq^\mathcal{A}$-rule, as we will see in Theorem 3.4.6, leads to a nice feature regarding different assertional closures of the same Σ. Example 3.4.5 below shows that two different executions of Λ may lead to two different assertional closures $\mathcal{A}^f$.

Example 3.4.5 *Consider a KB* $\Sigma = \langle \mathcal{A}, \emptyset \rangle$ *where* $\mathcal{A} = \{\exists r.C(a), \exists r.D(a), \exists r.(C \sqcap D)(a)\}$*. One computation of the assertional closure* $\mathcal{A}^f$ *of* Σ *is obtained by:*

1. *applying the* $\exists_2^\mathcal{A}$*-rule to* $\exists r.(C \sqcap D)(a)$ *and introducing* $r(a, b), C \sqcap D(b)$*;*

2. *applying the* $\sqsubseteq^{\mathcal{A}}$*-rule to* $C \sqcap D(b)$ *and introducing* $C(b), D(b)$*;*

resulting $\mathcal{A}^f = \mathcal{A} \cup \{r(a,b), C \sqcap D(b), C(b), D(b)\}$*. Another computation of* $\mathcal{A}^f$ *is:*

1. *applying the* $\exists_2^{\mathcal{A}}$*-rule to* $\exists r.C(a)$ *and introducing* $r(a,d_1), C(d_1)$*;*

2. *applying the* $\exists_2^{\mathcal{A}}$*-rule to* $\exists r.D(a)$ *and introducing* $r(a,d_2), D(d_2)$*;*

3. *applying the* $\exists_2^{\mathcal{A}}$*-rule to* $\exists r.(C \sqcap D)(a)$ *and introducing* $r(a,e), C \sqcap D(e)$*;*

4. *applying the* $\sqsubseteq^{\mathcal{A}}$*-rule to* $C \sqcap D(e)$ *and introducing* $C(e), D(e)$*;*

resulting $\mathcal{A}^f = \mathcal{A} \cup \{r(a,d_1), C(d_1), r(a,d_2), D(d_2), r(a,e), C \sqcap D(e), C(e), D(e)\}$. ■

As illustrated in Example 3.4.5, even though different assertional closures of Σ can be obtained from different executions of Λ, they differ only in assertions about individual names that have been freshly chosen during the executions. This observation is formulated more precisely in the next theorem. Let $\mathcal{A}^f_\Sigma$ denote the assertional closures $\mathcal{A}^f$ restricted to the individuals in $\mathcal{O}_\Sigma$.

Theorem 3.4.6 *Given a KB* $\Sigma = \langle \mathcal{A}, \mathcal{T} \rangle$ *and a secrecy set* $\mathbb{S}$*, let* $\mathcal{A}_1^f$ *and* $\mathcal{A}_2^f$ *be two assertional closures of* Σ *w.r.t. SubE obtained by different executions of* Λ*. Then* $\mathcal{A}_{1\Sigma}^f = \mathcal{A}_{2\Sigma}^f$ *where* $\mathcal{A}_{1\Sigma}^f$ *(*$\mathcal{A}_{2\Sigma}^f$*) denotes* $\mathcal{A}_1^f$ *(*$\mathcal{A}_2^f$*) restricted to the individuals in* $\mathcal{O}_\Sigma$*.*

Proof By symmetry, it suffices to prove that $\mathcal{A}_{1\Sigma}^f \subseteq \mathcal{A}_{2\Sigma}^f$. If $C(a) \in \mathcal{A}_{1\Sigma}^f$, then we have $a \in \mathcal{O}_\Sigma$ and $C \in \mathcal{C} \cap SubE$. It follows from Theorem 3.4.3 that $\Sigma \models C(a)$. By Theorem 3.4.4, $C(a) \in \mathcal{A}_2^f$. Since $a \in \mathcal{O}_\Sigma$, $C(a) \in \mathcal{A}_{2\Sigma}^f$. ■

3.4.3 Computing Envelopes in $\mathcal{EL}$-KBs

In this section, we illustrate how we apply the general idea of computing envelopes presented in Section 3.2.3. The basic idea is to start from secrets, protect at least one premise from each proof of each secret, and answer "Unknown" to the protected information. As we shall see, this can be achieved by inverting the normal inference rules. Because of OWA, a querying agent cannot distinguish between an answer "Unknown" that results from the reasoner's incomplete

information and an "Unknown" resulting from the reasoner's need to protect secret information. Since we have assumed that the querying agent can only ask queries over the vocabulary N_Σ, the information the reasoner needs to protect against need not include assertions about individuals that are not in $\mathcal{O}_\Sigma$.

Remark. Axiom pinpointing [61, 62] was introduced to find out proofs for a given consequence by computing a boolean formula that encodes all the axioms used for obtaining the consequence. Specifically, axiom pinpointing in $\mathcal{EL}$ [62] aims at finding all the possible subsumptions used to deduce the subsumption relation between two concepts. It has been shown in [63] that axiom pinpointing is hard. Different from axiom pinpointing that finds minimal subsets of the original KB that have the given consequence, to protect a secret, we need to find all the sets (some of them may not be a subset of the original KB since our secrets are assertions rather than subsumptions) that the given secret can be deduced from, and disrupt every such set so that the secret is not deducible.

Example 3.4.7 *(Example 3.4.2, cont.) For the given knowledge base $\Sigma_1 = \langle \mathcal{A}_1, \mathcal{T}_1 \rangle$, consider the secrecy set $\mathbb{S}_1 = \{CancerRisk(Jane)\}$. Here, the querying agent is the insurance company and the queries include CancerRisk(Jane) and Reimburse(Jane). Because CancerRisk(Jane) can be inferred from statements 12, 2 and 3 in $\mathcal{A}_1$, at least one of these assertions, e.g., statement 12, should be put into the envelope. Thus, HasMutBRCA1(Jane)$\in \mathbb{E}_1$ for $\mathbb{S}_1$.* ■

The following definition specializes the general definition of envelopes and tight envelopes to $\mathcal{EL}$-KBs (see Definition 3.2.3).

Definition 3.4.8 *Given an $\mathcal{EL}$-KB $\Sigma = \langle \mathcal{A}, \mathcal{T} \rangle$ and a finite secrecy set $\mathbb{S} \subseteq \mathcal{A}^+_\Sigma$, a* secrecy envelope *(or* envelope*) for $\mathbb{S}$, denoted by $\mathbb{E}$, is a superset of $\mathbb{S}$ and a subset of $\mathcal{A}^+_\Sigma \setminus \mathcal{T}_{aut}$ such that for every $\alpha \in \mathbb{E}$, $\mathcal{A}^+_\Sigma \setminus \mathbb{E} \not\models \alpha$ where $\mathcal{A}^+$ is the assertional closure of Σ and $\mathcal{A}^+_\Sigma$ is $\mathcal{A}^+$ restricted to the individuals in $\mathcal{O}_\Sigma$. An envelope $\mathbb{E}$ is* tight *if for every $\alpha \in \mathbb{E}$, there exists $\beta \in \mathbb{S}$ such that $(\mathcal{A}^f_\Sigma \setminus \mathbb{E}) \cup \{\alpha\} \models \beta$.*

Remark. (1) The computational idea behind this definition is that if the reasoner $\mathcal{R}$ answers every query in $\mathbb{E}$ with "Unknown" and every query in $\mathcal{A}^+_\Sigma \setminus \mathbb{E}$ with "Yes", the querying agent

will not be able to deduce any assertions in $\mathbb{S}$. Note that the computation of a secrecy envelope, as described above, happens in the pre-query stage. (2) When constructing a knowledge base that involves secret or private information, the designer is faced with the problem of resolving the tradeoff between the two goals of a good design: secrecy and informativeness. In order to be as informative as possible, we aim to make $\mathbb{E}$ as small as possible. A tight envelope $\mathbb{E}$ is irredundant in the sense that removing any assertion α from $\mathbb{E}$ (i.e., answering it with "Yes" instead of "Unknown") would leave the secrecy set $\mathbb{S}$ vulnerable. Thus, the problem of how to efficiently compute tight envelopes is of much interest. In what follows, we first focus on the primary goal of preserving secrecy. Based on the techniques used to achieve this goal, we go on to present some methods that can be used to achieve the second goal of improved informativeness.

To compute an envelope, we introduce the technique of inverting assertion expansion rules. The intuitive idea behind inverting rules is that: *if the conclusion of an expansion rule is to be secret, then some of its premises must be secret as well.* There are four assertion expansion rules in Figure 3.1. Among these four rules, even if $\exists_2^{\mathcal{A}}$-rule is applicable to $(\mathcal{A}_\Sigma^f \setminus \mathbb{E})^+$, due to OWA, the querying agent can only conclude that there exists an individual d that is the witness for $\exists r.C(a)$ and that $d \notin \mathcal{O}_\Sigma$. However, since the querying agent has no computational access to individual names that are not in $\mathcal{O}_\Sigma$ (except, of course, the names introduced for its own use), inverting $\exists_1^{\mathcal{A}}$- and $\exists_2^{\mathcal{A}}$-rules can be restricted to $\mathcal{O}_\Sigma$ (without considering any "fresh" individuals). Therefore, we do not need to invert the $\exists_2^{\mathcal{A}}$-rule. Figure 3.2 lists a set of secrecy closure rules that can be used to compute an envelope, which we name *Secrecy Closure Rules*. Note that the SQ system only maintains a finite part of an envelope that is a subset of $\mathcal{A}^f$.

$\sqsubseteq^S$ -rule:	if $C(a) \in \mathcal{A}^f \setminus \mathbb{E}^f, C \sqsubseteq D \in \mathcal{T}^f$ and $D(a) \in \mathbb{E}^f$, then $\mathbb{E}^f := \mathbb{E}^f \cup \{C(a)\}$;
$\sqcap^S$ -rule:	if $C_1 \sqcap \cdots \sqcap C_k(a) \in \mathbb{E}^f$ and $\{C_1(a), ..., C_k(a)\} \cap \mathbb{E}^f = \emptyset$, then $\mathbb{E}^f := \mathbb{E}^f \cup \{C_i(a)\}$ where $1 \leq i \leq k$ and C_i is atomic;
$\exists^S$ -rule:	if $\exists r.C(a) \in \mathbb{E}^f$ and $\{r(a,b), C(b)\} \subseteq \mathcal{A}^f \setminus \mathbb{E}^f$ with $b \in \mathcal{O}_\Sigma$, then $\mathbb{E}^f := \mathbb{E}^f \cup \{r(a,b)\}$ or $\mathbb{E}^f := \mathbb{E}^f \cup \{C(b)\}$.

Figure 3.2 Secrecy Closure Rules

We denote by Λ_S the tableau algorithm which repeatedly and nondeterministically applies

the secrecy closure rules where $\mathbb{E}^f$ is initialized as the given set $\mathbb{S}$. When no further rules are applicable, we say that $\mathbb{E}^f$ is *closed w.r.t.* $\mathbb{S}$. It is clear that all executions of Λ_S, on an input consisting of the assertional closure $\mathcal{A}^f$ of a KB and a finite set of assertions $\mathbb{S}$, terminate. It is also easy to see that Λ_S takes polynomial time in the size of its input.

Example 3.4.9 *For the given KB Σ_1 and the secrecy set $\mathbb{S}_1$, with the assertional closure $\mathcal{A}_1^f$ of Σ_1, by applying Λ_S, an envelope can be obtained and is listed as follows:*
$\mathbb{E}'_1 = \{$*CancerRisk(Jane), ∃has_pres.CancerDrug(Jane), HasMutBRCA1(Jane),*
∃is_child.A(Jane), is_child(Jane, Jill) $\}$. ■

Theorem 3.4.10 *Let $\Sigma = \langle \mathcal{A}, \mathcal{T} \rangle$ be a KB and $\mathbb{S} \subseteq \mathcal{A}_\Sigma^f$ be a finite secrecy set. If $\mathbb{E}^f$ is closed w.r.t. $\mathbb{S}$, then $\mathbb{E}^f$ is an envelope for $\mathbb{S}$ restricted to SubE.*

Proof Since $\mathbb{E}^f$ is initialized as $\mathbb{S}$, we have $\mathbb{S} \subseteq \mathbb{E}^f$. To show that $\mathbb{E}^f$ is an envelope, we need to show that for every $\alpha \in \mathbb{E}^f$, $\mathcal{A}_\Sigma^f \setminus \mathbb{E}^f \not\models \alpha$. Since the tableau algorithm Λ is sound, it suffices to show that for every $\alpha \in \mathbb{E}^f$, $\mathcal{A}_\Sigma^f \setminus \mathbb{E}^f \not\vdash \alpha$, or, equivalently, $(\mathcal{A}_\Sigma^f \setminus \mathbb{E}^f)^f \cap \mathbb{E}^f = \emptyset$.

Let $\mathbb{A} = \mathcal{A}_\Sigma^f \setminus \mathbb{E}^f$. We prove that $\mathbb{A}^f \cap \mathbb{E}^f = \emptyset$ by induction on the construction of $\mathbb{A}^f$. We use $\mathbb{A}'$ ($\mathbb{A}''$) and $\mathcal{O}'$ ($\mathcal{O}''$) to denote the ABox before (after) the application of an expansion rule and the set of individual names appearing in $\mathbb{A}'$ ($\mathbb{A}''$), respectively. Since $\mathcal{A}_\Sigma^f \setminus \mathbb{E}^f \subseteq \mathcal{A}^f$, it follows from Theorem 3.4.6 that, $\mathbb{A}_\Sigma^f$, the assertional closure of $\langle \mathcal{A}_\Sigma^f \setminus \mathbb{E}^f, \mathcal{T} \rangle$ restricted to $\mathcal{O}_\Sigma$, is a subset of $\mathcal{A}_\Sigma^f$. Therefore, for each assertion $C(a)$: $C(a) \in \mathbb{A}''$, $a \in \mathcal{O}_\Sigma \Rightarrow C(a) \in \mathcal{A}^f$.

The base case is when none of the assertion expansion rules has been applied yet, $\mathbb{A}' = \mathbb{A}$. Clearly, $\mathbb{A} \cap \mathbb{E}^f = \emptyset$. We assume that $\mathbb{A}' \cap \mathbb{E}^f = \emptyset$ and show that $\mathbb{A}'' \cap \mathbb{E}^f = \emptyset$.

- If the $\sqsubseteq^{\mathcal{A}}$-rule is applicable, then $C(a) \in \mathbb{A}'$, $C \sqsubseteq D \in \mathcal{T}^f$ and $D(a) \notin \mathbb{A}'$. After the application of the $\sqsubseteq^{\mathcal{A}}$-rule, $D(a) \in \mathbb{A}''$. Suppose $D(a) \in \mathbb{E}^f$. Because individuals appearing in $\mathbb{E}^f$ are in $\mathcal{O}_\Sigma$, we have $a \in \mathcal{O}_\Sigma$, and so by Theorem 3.4.6, $C(a), D(a) \in \mathcal{A}^f$. However, since $\mathbb{E}^f$ is closed, by the $\sqsubseteq^S$-rule, $C(a) \in \mathbb{E}^f$, which contradicts $\mathbb{A}' \cap \mathbb{E}^f = \emptyset$. Therefore, $\mathbb{A}'' \cap \mathbb{E}^f = \emptyset$.

- If the $\sqcap^{\mathcal{A}}$-rule is applicable, then $\{C_1(a), ..., C_k(a)\} \subseteq \mathbb{A}'$, $C_1 \sqcap \cdots \sqcap C_k(a) \notin \mathbb{A}'$ and $C_1 \sqcap \cdots \sqcap C_k \in SubE$. $\mathbb{A}' \cap \mathbb{E}^f = \emptyset$ implies that $\{C_1(a), ..., C_k(a)\} \cap \mathbb{E}^f = \emptyset$. After the

application of the $\sqcap^{\mathcal{A}}$-rule, $C_1 \sqcap \cdots \sqcap C_k(a) \in \mathbb{A}''$. If $\mathbb{A}'' \cap \mathbb{E} \neq \emptyset$, then $C_1 \sqcap \cdots \sqcap C_k(a) \in \mathbb{E}^f$. Since $\mathbb{E}^f$ is closed, by $\sqcap^S$-rule, there is an i $(1 \leq i \leq k)$ such that $C_i(a) \in \mathbb{E}^f$, contradicting $\mathbb{A}' \cap \mathbb{E}^f = \emptyset$. Therefore, $\mathbb{A}'' \cap \mathbb{E}^f = \emptyset$.

- If the $\exists_1^{\mathcal{A}}$-rule is applicable, then $\{r(a,b), C(b)\} \subseteq \mathbb{A}'$, $\exists r.C \in SubE$ and $\exists r.C(a) \notin \mathbb{A}'$. After the application of the rule, $\exists r.C(a) \in \mathbb{A}''$. By IH, $\mathbb{A}' \cap \mathbb{E}^f = \emptyset$, so in particular, $\{r(a,b), C(b)\} \subseteq \mathbb{A}' \setminus \mathbb{E}^f$. There are two cases:
  - $b \in \mathcal{O}_\Sigma$. By (f_1), $a \in \mathcal{O}_\Sigma$ and $r(a,b) \in \mathcal{A}$. It then follows from Theorem 3.4.6 that $r(a,b), C(b) \in \mathcal{A}_\Sigma^f$. Suppose $\exists r.C(a) \in \mathbb{E}^f$. Since $\mathbb{E}^f$ is closed, by the $\exists^S$-rule, either $r(a,b) \in \mathbb{E}^f$ or $C(b) \in \mathbb{E}^f$, contradicting $\mathbb{A}' \cap \mathbb{E}^f = \emptyset$.
  - $b \notin \mathcal{O}_\Sigma$. Suppose $\exists r.C(a) \in \mathbb{E}^f$ (and hence, $a \in \mathcal{O}_\Sigma$). Since $b \notin \mathcal{O}_\Sigma$, b was introduced earlier via an application of the $\exists_2^{\mathcal{A}}$-rule to some assertion in $\mathbb{A}'$. In view of $r(a,b) \in \mathbb{A}'$, this assertion must be of the form $\exists r.D(a)$. We have, by (f_3), $D \sqsubseteq C \in \mathcal{T}^f$ and so, $\exists r.D \sqsubseteq \exists r.C \in \mathcal{T}^f$. Because $a \in \mathcal{O}_\Sigma$, it follows from Theorem 3.4.6 that $\exists r.D(a) \in \mathcal{A}^f$. Since $\mathbb{E}^f$ is closed, by the $\sqsubseteq^S$-rule, $\exists r.D(a) \in \mathbb{E}^f$. However, this contradicts $\mathbb{A}' \cap \mathbb{E}^f = \emptyset$.

  Therefore, $\mathbb{A}'' \cap \mathbb{E}^f = \emptyset$.
- An application of the $\exists_2^{\mathcal{A}}$-rule introduces a fresh individual, say $c \notin \mathcal{O}_\Sigma$, and adds two assertions involving c to $\mathbb{A}''$. On the other hand, the tableau algorithm Λ_S puts in $\mathbb{E}^f$ only assertions involving individuals in $\mathcal{O}_\Sigma$. It follows that $\mathbb{A}'' \cap \mathbb{E}^f = \emptyset$. ■

Note that the whole initialization of the SQ system (including the computation of $SubE$, $\mathcal{A}^f$ and $\mathbb{E}^f$) is easily seen to be doable in polynomial time in the size of the KB Σ plus the size of the given secrecy set $\mathbb{S}$.

3.5 Tight Envelopes

Depending on the execution of Λ_S, we may have different secrecy envelopes. Furthermore, the construction of $\mathbb{E}$ resulting from Λ_S may not result a tight envelope as shown in the following example.

Example 3.5.1 *Suppose that the secrecy set* $\mathbb{S}$ *is* $\{C \sqcap D(a), D \sqcap E(a)\}$ *and* $\Sigma = \langle \mathcal{A}, \emptyset \rangle$ *where* $\mathcal{A} = \{C(a), D(a), E(a)\}$. *Depending on the choice made in the application of the* $\sqcap^S$*-rule when computing the secrecy envelope, we may have several envelopes:*

- $\mathbb{E}_1 = \mathbb{S} \cup \{C(a), D(a)\}$, $\mathbb{E}_2 = \mathbb{S} \cup \{D(a), E(a)\}$ — *not tight;*
- $\mathbb{E}_3 = \mathbb{S} \cup \{C(a), E(a)\}$ — *tight;*
- $\mathbb{E}_4 = \mathbb{S} \cup \{D(a)\}$ — *minimum (and tight).*

Since our goal is to answer queries as informatively as possible while preserving secrecy, we would prefer to have a minimum envelope, i.e., an envelope of the smallest cardinality. Unfortunately, to compute a minimum envelope is hard. Specifically, we will show that the decision version of the problem of computing minimum envelopes is NP-complete.

3.5.1 Deciding a Minimum Secrecy Envelope is NP-complete

An instance of the Minimum Secrecy Envelope (MSE) problem contains a triple $\langle \Sigma = \langle \mathcal{A}, \mathcal{T} \rangle, \mathbb{S}, k \rangle$ where Σ is a knowledge base, $\mathcal{A}^+$ is the assertional closure of Σ restricted to $SubE$, $\mathbb{S} \subseteq \mathcal{A}^+$ is a secrecy set, and $k \leq |\mathcal{A}^+|$ is a nonnegative integer. The question is "Is there a secrecy envelope $\mathbb{E}$ such that $\mathbb{S} \subseteq \mathbb{E} \subseteq \mathcal{A}^+$ and $|\mathbb{E} \setminus \mathbb{S}| \leq k$?"

Given a set of assertions $\mathbb{E}' \supseteq \mathbb{S}$, we can verify (a) whether $\mathbb{E}'$ is an envelope by recalculating $(\mathcal{A}^+_\Sigma \setminus \mathbb{E}')^+$ and checking that it contains no assertions in $\mathbb{S}$, and (b) whether $|\mathbb{E}' \setminus \mathbb{S}| \leq k$. Both tasks are doable in polynomial time and therefore MSE belongs to NP.

To show that the MSE problem is NP-hard, we reduce the *Hitting Set* (HS) problem to the MSE problem. An instance of HS consists of a collection $\mathcal{M}$ of subsets of a finite set S and a positive integer $k \leq |S|$. The question is "Is there a subset $S' \subseteq S$ with $|S'| \leq k$ such that S' contains at least one element from each set in $\mathcal{M}$?" W.l.o.g., we may assume that every set in $\mathcal{M}$ has at least two elements.

Given an instance of HS, we construct an instance of MSE, using the same constant k, as follows:

- $N_\mathcal{O} = \{a\}$, $N_\mathcal{R} = \emptyset$, $N_\mathcal{C} = S$

- $\mathbb{S} = \{A_1 \sqcap \cdots \sqcap A_m(a) \mid \{A_1, ..., A_m\} \in \mathcal{M}\}$
- $\mathcal{A} = \{A(a) \mid A \in S\} \cup \mathbb{S}$
- $\Sigma = \langle \mathcal{A}, \emptyset \rangle$

Claim. S has a subset S' with $|S'| \leq k$ that hits every subset in $\mathcal{M}$ iff there is a secrecy envelope $\mathbb{E}$ such that $\mathbb{S} \subseteq \mathbb{E} \subseteq \mathcal{A}^+$ and $|\mathbb{E} \setminus \mathbb{S}| \leq k$.

Proof Suppose that S has a subset S' with $|S'| \leq k$ that hits every set in $\mathcal{M}$. Then for each set $\{A_1, ..., A_m\} \in \mathcal{M}$, there is an element $A_j \in S'$ $(1 \leq j \leq m)$. Let $\mathbb{E} = S' \cup \mathbb{S}$. It follows that for every $A_1 \sqcap \cdots \sqcap A_m(a) \in \mathbb{S}$, there is an assertion $A_j(a) \in \mathbb{E}$ $(1 \leq j \leq m)$. By construction, Σ does not involve any roles and $SubE = \{C \mid C(a) \in \mathcal{A}\}$. Therefore $\mathcal{A}^+_\Sigma = \mathcal{A}$, and so $\mathcal{A}^+_\Sigma \setminus \mathbb{E} = \mathcal{A} \setminus \mathbb{E}$. Note that $\mathcal{A} \setminus \mathbb{E}$ contains only assertions of the form $C(a)$ where $C \in N_\mathcal{C}$ and $a \in \mathcal{O}_\Sigma$. Consequently, none of the assertion expansion rules is applicable to $\mathcal{A} \setminus \mathbb{E}$, implying that $(\mathcal{A}^+_\Sigma \setminus \mathbb{E})^+ \cap \mathbb{E} = (\mathcal{A} \setminus \mathbb{E}) \cap \mathbb{E} = \emptyset$. It follows that $\mathbb{E}$ is an envelope with $|\mathbb{E} \setminus \mathbb{S}| \leq |S'| \leq k$.

Conversely, suppose that there is a secrecy envelope $\mathbb{E}$ such that $\mathbb{S} \subseteq \mathbb{E} \subseteq \mathcal{A}^+$ and $|\mathbb{E} \setminus \mathbb{S}| \leq k$. Then, by Definition 3.4.8, for every $A_1 \sqcap \cdots \sqcap A_m(a) \in \mathbb{S}$, there is $A_j(a) \in \mathbb{E}$ where $1 \leq j \leq m$. $\mathbb{E} \setminus \mathbb{S}$ contains only assertions of the form $C(a)$ where $C \in N_\mathcal{C}$. This shows that S contains a subset $S' = \{C \mid C(a) \in \mathbb{E} \setminus \mathbb{S}\}$ that hits every set in $\mathcal{M}$ and $|S'| = |\mathbb{E} \setminus \mathbb{S}| \leq k$. ■

3.5.2 A Naive Algorithm for Computing Tight Envelope

Since computing a minimum envelope is NP-hard, in what follows, we consider polynomial time algorithms for computing tight envelopes. Such a tight envelope is minimal in the sense of being irredundant, but, as shown in Example 3.5.1, it need not be minimum.

By Definition 3.4.8, a naive algorithm for computing tight envelope can simply take an envelope $\mathbb{E}^f$ obtained by applying Λ_S to the secrecy set $\mathbb{S}$, and check each assertion α in $\mathbb{E}^f$ whether it is redundant: if the intersection $((\mathcal{A}^f_\Sigma \setminus \mathbb{E}^f) \cup \{\alpha\})^f \cap \mathbb{S}$ is empty, α is redundant and moved from $\mathbb{E}^f$ to $\mathcal{A}^f_\Sigma \setminus \mathbb{E}^f$. Otherwise, α remains in $\mathbb{E}^f$.

Since $\mathcal{A}^f$ and $\mathbb{E}^f$ can be computed in polynomial time and $((\mathcal{A}^f_\Sigma \setminus \mathbb{E}^f) \cup \{\alpha\})^f \cap \mathbb{S}$ can also be computed in polynomial time, it follows that the whole naive algorithm runs in polynomial

time. The following example illustrates some of the computations in the execution of the naive algorithm.

Example 3.5.2 *Given a KB* $\Sigma = \langle \mathcal{A}, \mathcal{T} \rangle$ *and a secrecy set* $\mathbb{S}$ *where*

- $\mathcal{A} = \{F(a),\ C(a),\ \exists r.\exists s.(D \sqcap D')(a),\ C \sqcap C_1(a),\ C \sqcap C_2(a),\ ...,\ C \sqcap C_k(a),\ E_1 \sqcap E_2(a)\}$,
- $\mathcal{T} = \{E_1 \sqcap E_2 \sqsubseteq C,\ \exists r.M \sqsubseteq F,\ \exists s.D \sqsubseteq M\}$, *and*
- $\mathbb{S} = \{F(a),\ C \sqcap \exists r.\exists s.(D \sqcap D')(a)\}$,

the set of subexpressions $SubE$ *and a corresponding assertional closure* $\mathcal{A}^f$ *are listed as follows:*

- $SubE = \{F, C, \exists r.\exists s.(D \sqcap D'), r, \exists s.(D \sqcap D'), s, D \sqcap D', D, D', C \sqcap C_1, C \sqcap C_2,\ ...,\ C \sqcap C_k, C, C_1, C_2, ..., C_k, E_1 \sqcap E_2, E_1, E_2, \exists r.M, M, \exists s.D, C \sqcap \exists r.\exists s.(D \sqcap D')\}$.
- $\mathcal{A}^f = \mathcal{A} \cup \mathbb{S} \cup \{r(a,x),\ \exists s.(D \sqcap D')(x),\ s(x,y),\ D \sqcap D'(y),\ D(y),\ D'(y),\ \exists s.D(x),\ M(x),\ \exists r.M(a),\ C_1(a),\ C_2(a),\ ...,\ C_k(a),\ C(a),\ E_1(a),\ E_2(a)\}$, *where* x *and* y *were freshly chosen individuals during the computation of* $\mathcal{A}^+$.

Suppose that an envelope $\mathbb{E}^f$ *is obtained by*

1. *applying the* $\sqcap^S$*-rule to* $C \sqcap \exists r.\exists s.(D \sqcap D')(a)$ *and choosing* $C(a)$,
2. *applying the* $\sqsubseteq^S$*-rule to* $C(a)$ *and obtaining* $E_1 \sqcap E_2(a)$,
3. *applying the* $\sqcap^S$*-rule to* $E_1 \sqcap E_2(a)$ *and choosing* $E_1(a)$,
4. *applying the* $\sqsubseteq^S$*-rule to* $F(a)$ *and obtaining* $\exists r.M(a)$,
5. *applying the* $\sqsubseteq^S$*-rule to* $\exists r.M(a)$ *and obtaining* $\exists r.\exists s.(D \sqcap D')(a)$. *Note that* $\exists s.(D \sqcap D') \sqsubseteq \exists s.D \sqsubseteq M$ *and so* $\exists r.\exists s.(D \sqcap D') \sqsubseteq \exists r.M$.
6. *applying the* $\sqsubseteq^S$*-rule to* $C(a)$ *repeatedly and obtaining* $C \sqcap C_1(a), ..., C \sqcap C_k(a)$.

The resulting envelope is $\mathbb{E}^f = \{F(a),\ C \sqcap \exists r.\exists s.(D \sqcap D')(a),\ C(a),\ E_1 \sqcap E_2(a),\ E_1(a),\ \exists r.M(a),\ \exists r.\exists s.(D \sqcap D')(a),\ C \sqcap C_1(a),\ ...,\ C \sqcap C_k(a)\}$. *The naive algorithm tests every assertion in* $\mathbb{E}^f$ *and removes it if it is redundant. Note that in this case the presence of* $\exists r.\exists s.(D \sqcap D')(a)$ *not*

only makes $C(a)$ *as well as all of* $C \sqcap C_1(a), ..., C \sqcap C_k(a)$ *redundant, but also makes* $E_1 \sqcap E_2(a)$ *and* $E_1(a)$ *redundant since they were obtained because of* $C(a)$*. This suggests that further optimization can be done to compute a tight envelope during the construction of an envelope by using D-secrecy closure rules.* ■

3.5.3 $\exists_d^S$-Secrecy Closure Rule

Instead of repeatedly checking redundancy as done by the naive approach, we provide an optimized approach that guides the application of secrecy closure rules as it builds up a tight envelope.

Among the secrecy closure rules (Figure 3.2), the $\exists^S$-rule is the only one that, when applied to an assertion for one individual, may cause an assertion for (possibly) a different individual to become redundant. Consider an example where $\{r(a,b), C(b), D(b)\} \subseteq \mathcal{A}^f \setminus \mathbb{S}$ and $\mathbb{S} = \{\exists r.C(a), C \sqcap D(b)\}$. An application of the $\sqcap^S$-rule may lead to $\mathbb{E}^f = \{\exists r.C(a), C \sqcap D(b), D(b)\}$. Then, since $\{r(a,b), C(b)\} \subseteq \mathcal{A}^f \setminus \mathbb{E}^f$ and $\exists r.C(a) \in \mathbb{E}^f$, the $\exists^S$-rule is applicable and $\mathbb{E}^f$ may be expanded to $\{\exists r.C(a), C \sqcap D(b), D(b), C(b)\}$. At this point, $D(b)$ becomes redundant. To reduce such kinds of "interactions" between assertions for different individuals, we will use a deterministic version of $\exists^S$-rule. The rule, named $\exists_d^S$-rule, is listed in Figure 3.3.

$\sqsubseteq^S$ -rule:	if $C(a) \in \mathcal{A}^f \setminus \mathbb{E}^f, C \sqsubseteq D \in \mathcal{T}^f$ and $D(a) \in \mathbb{E}^f$, then $\mathbb{E}^f := \mathbb{E}^f \cup \{C(a)\}$;
$\sqcap^S$ -rule:	if $C_1 \sqcap \cdots \sqcap C_k(a) \in \mathbb{E}^f$ and $\{C_1(a), ..., C_k(a)\} \cap \mathbb{E}^f = \emptyset$, then $\mathbb{E}^f := \mathbb{E}^f \cup \{C_i(a)\}$ where $1 \leq i \leq k$ and C_i is atomic;
$\exists_d^S$ -rule:	if $\exists r.C(a) \in \mathbb{E}^f$ and $\{r(a,b), C(b)\} \subseteq \mathcal{A}^f \setminus \mathbb{E}^f$ with $b \in \mathcal{O}_\Sigma$, then $\mathbb{E}^f := \mathbb{E}^f \cup \{r(a,b)\}$.

Figure 3.3 D-secrecy closure rules

The set of rules obtained from the deterministic secrecy closure rules by replacing the $\exists^S$-rule with the $\exists_d^S$-rule will be referred to as the *D-secrecy closure rules*.

In the following discussion, when we say that an assertion is redundant (irredundant), during the construction of an envelope, we mean that this assertion is a candidate of being removed (retained) in the resulting envelope. Here are some observations regarding the D-secrecy closure rules:

- $\sqcap^S$-rule: Some of the non-deterministic choices in its applications may create redundant assertions when computing an envelope (see Examples 3.5.1 and 3.5.2).

- $\exists_d^S$-rule: It cannot trigger applications of the $\sqcap^S$-rule (since its applications create role assertions) and so if we can restrict its applications to irredundant assertions, the $\exists_d^S$-rule will not create any redundancy.

- $\sqsubseteq^S$-rule: An application of the $\sqsubseteq^S$-rule may possibly trigger applications of the $\sqcap^S$-rule. If it does, since the non-deterministic choice in the $\sqcap^S$-rule may create redundancy, the $\sqsubseteq^S$-rule may participate in creating redundancy. However, for any assertion $C(a) \in \mathbb{E}$ obtained by applying the $\sqsubseteq^S$-rule, if there is an irredundant assertion $E(a) \in \mathbb{E}$ such that $C \sqsubseteq E$, then $C(a)$ is also irredundant.

Taking these observations into account, the general idea of the optimized approach is to

(i) obtain a set of irredundant assertions that contains the given secrecy set and is closed under the $\sqcap^S$-rule, and

(ii) exhaustively apply the $\sqsubseteq^S$- and $\exists_d^S$-rules (which, as we will show in Theorem 3.5.6, will not create any redundancy).

In step (i), we need to consider not only applications of the $\sqcap^S$-rule, but also applications of the $\sqsubseteq^S$-rule because those applications, in turn, may trigger applications of the $\sqcap^S$-rule (see Step 2 and 3 in Example 3.5.2). In this step, the applications of the $\sqsubseteq^S$-rule will be restricted to the set of subexpressions of $\mathbb{S}$ and $\mathcal{T}$, denoted by $SubE(\mathbb{S}, \mathcal{T})$ and computed in the same way as $SubE$ (see Section 3.4.1). This is important as the set $\mathbb{S} \cup \mathcal{T}$ may be substantially smaller than $\mathcal{A} \cup \mathbb{S} \cup \mathcal{T}$.

In a little more detail, the optimized approach first removes the redundancy from a set, referred to as $\bigcup_{a \in \mathcal{O}_\Sigma} \mathbb{E}_a^{(1)}$ in Section 3.5.4, that is closed under the $\sqcap^S$-rule and the $\sqsubseteq^S$-rule restricted to $SubE(\mathbb{S}, \mathcal{T})$. The removal procedure ensures that the resulting set, referred to as the basic set $\mathbb{B}_\mathbb{S}$, is a superset of the given secrecy set $\mathbb{S}$ and is closed under the $\sqcap^S$-rule. The optimized approach then completes the envelope by repeated applications of the $\sqsubseteq^S$-rule,

followed by repeated applications of the $\exists_d^S$-rule. This procedure, named TIGHT, results a tight envelope, as we shall see in Section 3.5.5.

Note that the $\sqcap^S$-rule in fact deals with some implicit subsumptions that can be deduced from an empty TBox. For example, $D \sqcap D_1 \sqsubseteq D$ is such a subsumption even if it may not appear in $\mathcal{T}$. If an assertion $D \sqcap D_1(a)$ is to be protected, one of $D(a)$ and $D_1(a)$ must be protected. This differs from an subsumption such as $C \sqsubseteq D$. In this case, if $D(a)$ is to be protected, then $C(a)$ needs to be protected. However, if $C(a)$ is to be protected, it's not necessary to protect $D(a)$ because a "Yes" answer to $D(a)$ does not directly lead to the "Yes" answer to $C(a)$. Therefore, we would like to distinguish these two kinds of subsumptions during the process of building a tight envelope. We say that a concept D *trivially subsumes* (or *covers*) a concept C, or C is *trivially subsumed* (or is *covered*) by D, denoted by $C \preceq D$, if $C = C_1 \sqcap ... \sqcap C_k$, $D = D_1 \sqcap ... \sqcap D_m$ and $\{D_1, \cdots, D_m\} \subseteq \{C_1, \cdots, C_k\}$. We stipulate that every concept is trivially subsumed by $\top$. The following list two observations about the trivial subsumptions.

(s_1) If (i) $C \in SubE$ is non-atomic, (ii) $C \sqsubseteq D \in \mathcal{T}^f$ and $C \neq D$, and (iii) C does not occur in the TBox, then there exists a concept $E \in SubE$ such that $E \neq C$, $C \preceq E$ and $E \sqsubseteq D \in \mathcal{T}^f$.

(s_2) If (i) $\exists r.C \in SubE$ does not occur in the TBox, and (ii) $\exists r.C \sqsubseteq D \in \mathcal{T}^f$ where $\exists r.C \neq D$, then there exists a concept $\exists r.E \in SubE$ such that $\{\exists r.C \sqsubseteq \exists r.E, \exists r.E \sqsubseteq D, C \sqsubseteq E\} \subseteq \mathcal{T}^f$ and $C \neq E$.

For observation (s_1), it is obvious that when $C \preceq D$, the claim is true (by letting $E = D$). If $C \not\preceq D$, then there must exist $C' \sqsubseteq D' \in \mathcal{T}$ (which cannot be deduced from an empty TBox) where $C' \not\preceq D'$ such that $C \preceq C'$ and $D' \sqsubseteq D \in \mathcal{T}^f$. Because $C' \sqsubseteq D' \in \mathcal{T}$, by the computation of $SubE$, $C' \in SubE$. Letting $E = C'$, the claim holds. For observation (s_2), since $\exists r.C$ is atomic and does not occur in the TBox, for $\exists r.C \sqsubseteq D$ to be a subsumption that is not deduced from an empty TBox, we can only have $C \sqsubseteq E$ where $C \not\preceq E$ to deduce $\exists r.C \sqsubseteq \exists r.E$ and through $\exists r.E \sqsubseteq D$ to obtain $\exists r.C \sqsubseteq D$.

We next provide an efficient algorithm for computing a *basic set*, denoted by $\mathbb{B}_{\mathbb{S}}$, that will play an important role in constructing a tight envelope for $\mathbb{S}$.

3.5.4 Computing the Basic Set $\mathbb{B}_{\mathbb{S}}$

For each $a \in \mathcal{O}_\Sigma$, let $\mathbb{S}_a = \{C(a) \mid C(a) \in \mathbb{S}\}$. We apply the $\sqcap^S$-rule and the $\sqsubseteq_1^S$-rule (the $\sqsubseteq^S$-rule restricted to $SubE(\mathbb{S}, \mathcal{T})$, see Figure 3.4) to $\mathbb{S}_a$ until none of them are applicable. The resulting set is denoted by $\mathbb{E}_a^{(1)}$.

$\sqsubseteq_1^S$ -rule:	if $C \in SubE(\mathbb{S}, \mathcal{T}), C(a) \in \mathcal{A}^f \setminus \mathbb{E}^f, C \sqsubseteq D \in \mathcal{T}^f$ and $D(a) \in \mathbb{E}^f$, then $\mathbb{E}^f := \mathbb{E}^f \cup \{C(a)\}$.

Figure 3.4 $\sqsubseteq_1^S$-Rule, the $\sqsubseteq^S$-Rule restricted to $SubE(\mathbb{S}, \mathcal{T})$

It turns out that the set $\mathbb{E}_a^{(1)}$ may be too large for our purposes. We compute a subset $\mathbb{B}_a \subseteq \mathbb{E}_a^{(1)}$ termed the *basic set for* $\mathbb{S}_a$ *w.r.t.* $\mathbb{E}_a^{(1)}$. Then we define $\mathbb{B}_{\mathbb{S}} := \bigcup_{a \in \mathcal{O}_\Sigma} \mathbb{B}_a$. To compute $\mathbb{B}_a$, we first construct a directed graph $G_a = \langle V_a, A_a^{\mathcal{T}}, A_a^{\sqcap} \rangle$ where $V_a = \mathbb{E}_a^{(1)}$ and $A_a^{\mathcal{T}}$ and $A_a^{\sqcap}$ consist of two types of edges constructed as follows:

(g_1) If $C(a), D(a) \in V_a$, $C \preceq D$, $C \neq D$, and there does not exist $E(a) \in V_a$ such that $C \neq E \neq D$ and $C \preceq E \preceq D$, then $A_a^{\sqcap} := A_a^{\sqcap} \cup \{(C(a), D(a))\}$.

(g_2) If $C(a), D(a) \in V_a$, $C \sqsubseteq D \in \mathcal{T}^f$ and $C \not\preceq D$, then $A_a^{\mathcal{T}} := A_a^{\mathcal{T}} \cup \{(D(a), C(a))\}$ [3].

We call a node $n \in V_a$ *atomic* if n is an atomic assertion and *non-atomic* otherwise. For any two nodes $n_1, n_2 \in V_a$, if $(n_1, n_2) \in A_a^{\sqcap}$ ($A_a^{\mathcal{T}}$), we say that n_1 is a $\sqcap$-*predecessor* ($\sqsubseteq$-*predecessor*) of n_2 and n_2 is a $\sqcap$-*successor* ($\sqsubseteq$-*successor*) of n_1. A node n_1 is a *predecessor* of a node n_2 if n_1 is either a $\sqcap$-predecessor or a $\sqsubseteq$-predecessor of n_2. The edges in G_a encode two kinds of dependencies between assertions. For every $C(a) \in V_a$, if $C(a)$ needs to be protected, then at least one of its $\sqcap$-successors and all of its $\sqsubseteq$-successors need to be protected. Note that $A_a^{\sqcap} \cap A_a^{\mathcal{T}} = \emptyset$.

Some edges in G_a represent applications of rules used to compute $\mathbb{E}_a^{(1)}$; other edges represent "accidental" subsumptions as illustrated in the following example. Let $\mathbb{E}_a^{(1)} = \{C \sqcap D(a), D \sqcap E(a), C(a), D(a)\}$ where $C(a)$ is obtained by an application of the $\sqcap^S$-rule to $C \sqcap D(a)$ and $D(a)$ is obtained by an application of the $\sqcap^S$-rule to $D \sqcap E(a)$. In the corresponding graph

[3] Note that the edges in $A_a^{\sqcap}$ and $A_a^{\mathcal{T}}$ represent subsumptions in opposite directions.

G_a, the edge $(C \sqcap D(a), D(a))$ does not correspond to an actual application of a rule used to compute $\mathbb{E}_a^{(1)}$.

The algorithm BASIC-SET checks for each node/assertion whether it is necessary in order to protect $\mathbb{S}_a$. If the tested node/assertion is not needed, it will be removed by the procedure REMOVE (Line 4, Procedure 5). Some ancestors of the tested node/assertion will also be removed (recursively) by the procedure REMOVE (Line 5-16, Procedure 5). Note that the variable W is used to keep track of all the tested nodes so that none is tested more than once. Moreover, it is easy to see that when the algorithm terminates, $V_a = W$. There are three criteria for removing predecessors:

c_1. If $(C(a), D(a)) \in A_a^{\mathcal{T}}$ (e.g., $C = \exists r.C_1$, $D = \exists r.D_1$ and $D_1 \sqsubseteq C_1 \in \mathcal{T}^f$), removing $D(a)$ leads to the removal of $C(a)$. This is because if $D(a)$ is not protected, neither can be $C(a)$.

c_2. If $(C(a), D(a)) \in A_a^{\sqcap}$ and $D(a)$ is the only $\sqcap$-successor of $C(a)$ in G_a, removing $D(a)$ will directly lead to the removal of $C(a)$ (e.g., $C = D \sqcap D'$ and $D'(a)$ is not in G_a. In this case, if none of $D(a)$ and $D'(a)$ is protected, $D \sqcap D'(a)$ cannot be protected and hence should be removed.); otherwise, when $C(a)$ has an $\sqcap$-successor other than $D(a)$, removing $D(a)$ will not lead to the removal of $C(a)$.

c_3. If $(C(a), D(a)) \in A_a^{\sqcap}$ and $C(a)$ is the only predecessor of $D(a)$ in G_a, removing $C(a)$ will lead to the removal of $D(a)$; otherwise, if $D(a)$ has a predecessor other than $C(a)$, removing $C(a)$ will not lead directly to the removal of $D(a)$.

Regarding criterion (c_3), note that we may have $\{(D \sqcap D'(a), D(a)), (D \sqcap D'(a), D'(a)\} \subseteq A_a^{\sqcap}$, and $(E(a), D'(a)) \in A_a^{\mathcal{T}}$, in which case, removing $D \sqcap D'(a)$ will lead to the removal of $D(a)$, but not $D'(a)$. After computing $\mathbb{B}_a$, if $D'(a) \in \mathbb{B}_a$, then $D \sqcap D'(a)$ will be brought back into the tight envelope by applying the $\sqsubseteq^S$-rule (see Section 3.5.5).

The procedure REMOVE-DESC (Procedure 6) further optimizes the process by repeatedly removing nodes that are only descendants of the removed node.

Example 3.5.3 *In Example 3.5.2, we have given a KB $\Sigma = \langle \mathcal{A}, \mathcal{T} \rangle$ with*

Algorithm 4 BASIC-SET($G_a = \langle V_a, A_a^{\mathcal{T}}, A_a^{\sqcap}\rangle$, $\mathbb{S}_a$)

$W := \mathbb{S}_a$ {The set W is used to maintain the tested irredundant nodes}
$V := V_a \setminus W$
while V is not empty **do**
  pick a node $v \in V$
  $G' = \langle V'_a, A'^{\mathcal{T}}_a, A'^{\sqcap}_a\rangle :=$ REMOVE($G_a = \langle V_a, A_a^{\mathcal{T}}, A_a^{\sqcap}\rangle$, v)
  if $W \subseteq V'_a$ **then** {Removing v will not compromise $\mathbb{S}_a$}
    $G_a := G'$
    $V := V'_a$
  else {v cannot be removed}
    $W := W \cup \{v\}$
  end if
  $V := V \setminus W$
end while
return $G_a = \langle V_a, A_a^{\mathcal{T}}, A_a^{\sqcap}\rangle$

- $\mathcal{A} = \{F(a),\ C(a),\ \exists r.\exists s.(D \sqcap D')(a),\ C \sqcap C_1(a),\ C \sqcap C_2(a),\ \ldots,\ C \sqcap C_k(a),\ E_1 \sqcap E_2(a)\}$,
- $\mathcal{T} = \{E_1 \sqcap E_2 \sqsubseteq C,\ \exists r.M \sqsubseteq F,\ \exists s.D \sqsubseteq M\}$, *and*
- $\mathbb{S} = \{F(a),\ C \sqcap \exists r.\exists s.(D \sqcap D')(a)\}$.

The set of subexpressions of $\mathbb{S}$ *and* $\mathcal{T}$ *is* $SubE(\mathbb{S}, \mathcal{T}) = \{F,\ C \sqcap \exists r.\exists s.(D \sqcap D'),\ C,\ \exists r.\exists s.(D \sqcap D'),\ r,\ \exists s.(D \sqcap D'),\ s,\ D \sqcap D',\ D,\ D',\ E_1 \sqcap E_2,\ E_1,\ E_2,\ \exists r.M,\ M,\ \exists s.D\}$.

Suppose that applications of the $\sqcap^S$- *and* $\sqsubseteq_1^S$-*rules are exactly the same as those in Example 3.5.2. Then we have* $\mathbb{E}_a^{(1)} = \mathbb{S} \cup \{C(a),\ E_1 \sqcap E_2(a),\ E_1(a),\ \exists r.M(a),\ \exists r.\exists s.(D \sqcap D')(a)\}$.

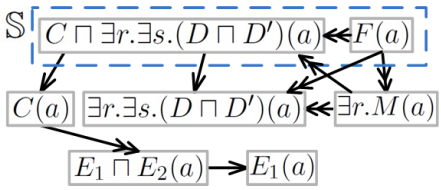

Figure 3.5 G_a *in Example 3.5.3*

Note that Step 6 in Example 3.5.2 is not applied since those concepts are not in $SubE(\mathbb{S}, \mathcal{T})$. *Figure 3.5 is the graph* $G_a = \langle V_a, A_a^{\mathcal{T}}, A_a^{\sqcap}\rangle$ *built from* $\mathbb{E}_a^{(1)}$. *In* G_a, *there are single-head arrows and two-head arrows which denote the edges in* $A_a^{\sqcap}$ *and* $A_a^{\mathcal{T}}$, *respectively. The nodes* $\exists r.M(a)$ *and* $\exists r.\exists s.(D \sqcap D')(a)$ *cannot be removed because removing any of them leads to the removal*

Procedure 5 REMOVE($G_a = \langle V_a, A_a^{\mathcal{T}}, A_a^{\sqcap}\rangle$, v)

$V_v^{\mathcal{T}} := \{u \mid (u, v) \in A_a^{\mathcal{T}}\}$
$V_v^{\sqcap} := \{u \mid (u, v) \in A_a^{\sqcap}\}$ {All the predecessors of v in G_a due to trivial subsumptions.}
$suc_v^{\sqcap} := \{u \mid (v, u) \in A_a^{\sqcap}\}$ {All the successors of v in G_a due to trivial subsumptions.}
$G_a := G_a \setminus \{v\}$ {Including all the incident edges. }
for all $u \in V_v^{\mathcal{T}}$ **do**
    $G_a = \langle V_a, A_a^{\mathcal{T}}, A_a^{\sqcap}\rangle :=$ REMOVE($G_a = \langle V_a, A_a^{\mathcal{T}}, A_a^{\sqcap}\rangle$, u) {See criterion (c_1).}
end for
for all $u \in V_v^{\sqcap}$ **do**
    if there is no vertex u' s.t. $(u, u') \in A_a^{\sqcap}$ **then**
        $G_a = \langle V_a, A_a^{\mathcal{T}}, A_a^{\sqcap}\rangle :=$ REMOVE($G_a = \langle V_a, A_a^{\mathcal{T}}, A_a^{\sqcap}\rangle$, u) {See criterion (c_2).}
    end if
end for
for all $u \in suc_v^{\sqcap} \cap V_a$ **do**
    $G_a = \langle V_a, A_a^{\mathcal{T}}, A_a^{\sqcap}\rangle :=$ REMOVE-DESC($G_a = \langle V_a, A_a^{\mathcal{T}}, A_a^{\sqcap}\rangle$, u)
end for
return $G_a = \langle V_a, A_a^{\mathcal{T}}, A_a^{\sqcap}\rangle$

Procedure 6 REMOVE-DESC($G_a = \langle V_a, A_a^{\mathcal{T}}, A_a^{\sqcap}\rangle$, v)

if v has a predecessor in G_a **then** {v will not be removed. See criterion (c_3).}
    return $G_a = \langle V_a, A_a^{\mathcal{T}}, A_a^{\sqcap}\rangle$
end if
$suc_v^{\sqcap} := \{u \mid (v, u) \in A_a^{\sqcap}\}$ {All the successors of v in G_a due to trivial subsumptions.}
$G_a := G_a \setminus \{v\}$ {Including all the incident edges.}
for all $u \in suc_v^{\sqcap}$ **do**
    $G_a = \langle V_a, A_a^{\mathcal{T}}, A_a^{\sqcap}\rangle :=$ REMOVE-DESC($G_a = \langle V_a, A_a^{\mathcal{T}}, A_a^{\sqcap}\rangle$, u)
end for
return $G_a = \langle V_a, A_a^{\mathcal{T}}, A_a^{\sqcap}\rangle$

of $F(a) \in \mathbb{S}$. After the execution of the Algorithm BASIC-SET*, the nodes $C(a)$, $E_1 \sqcap E_2(a)$ and $E_1(a)$ will be removed. Eventually, the basic set $\mathbb{B}_a = \mathbb{S} \cup \{\exists r.M(a), \exists r.\exists s.(D \sqcap D')(a)\}$.* ■

We denote by $\mathbb{G}_a = \langle \mathbb{V}_a, \mathbb{A}_a^{\mathcal{T}}, \mathbb{A}_a^{\sqcap}\rangle$ the graph obtained from $G_a = \langle V_a, A_a^{\mathcal{T}}, A_a^{\sqcap}\rangle$ by executing the algorithm BASIC-SET. Define $\mathbb{B}_a := \mathbb{V}_a$. The following lists some observations about $\mathbb{B}_a$:

b_1. $\mathbb{S}_a \subseteq \mathbb{B}_a$.

The set W in Algorithm 4 is initialized as $\mathbb{S}_a$ and none of the elements in W was removed once it is added to W. When the algorithm terminates, $W = \mathbb{V}_a = \mathbb{B}_a$. Therefore, $\mathbb{S}_a \subseteq \mathbb{B}_a$.

b_2. (a) For every assertion $C(a) \in \mathbb{B}_a \setminus \mathbb{S}_a$, there is a path to $C(a)$ in $\mathbb{G}_a$ that starts from an assertion in $\mathbb{S}_a$.

If an assertion $C(a) \in \mathbb{B}_a \setminus \mathbb{S}_a$ is not reachable from $\mathbb{S}_a$, then it is also not reachable from nodes in W in Algorithm 4 (because all nodes in W are reachable from $\mathbb{S}_a$), and so $C(a)$ would have been removed (Algorithm 4, Lines 5-11), contradicting the assumption.

(b) Moreover, given an assertion $D(a) \in \mathbb{B}_a \setminus \mathbb{S}_a$, there is a path p from $\mathbb{S}_a$ to $D(a)$ such that all non-atomic nodes on p have a unique $\sqcap$-successor in $\mathbb{G}_a$ (not necessarily on p).

Proof Let $D(a) \in \mathbb{B}_a \setminus \mathbb{S}_a$ be an assertion for which the claim does not hold. Then for every path p from $\mathbb{S}_a$ to $D(a)$, there is a non-atomic node u_p with more than one $\sqcap$-successor. W.l.o.g., we assume that u_p is the closest such node to $D(a)$ (among all the non-atomic nodes on p with more than one $\sqcap$-successor). Then, at the time when $D(a)$ was tested for removal (see Algorithm 4, Line 4), none of the nodes on p starting from u_p's $\sqcap$-successor and up to $D(a)$ were tested yet because if they were, they would have been removed (see criteria (c_1) and (c_2) as implemented by Algorithm 4, Lines 5, and Procedure 5). It follows that none of these nodes were in W at that time and all of them could be removed, including $D(a)$ (when $D(a)$ was tested for removal). This yields a contradiction to $D(a) \in \mathbb{V}_a = \mathbb{B}_a$. Therefore, the claim (b_2) part(b) holds. ■

b_3. $\mathbb{B}_a$ is closed under the $\sqcap^S$-rule. This is equivalent to the claim that every non-atomic node in $\mathbb{B}_a$ has a $\sqcap$-successor.

Proof Suppose that after the execution of the algorithm Basic-set, a non-atomic assertion, say $C \sqcap D \sqcap E(a) \in \mathbb{B}_a$, does not have any $\sqcap$-successor in $\mathbb{G}_a$. Since $\mathbb{E}_a^{(1)}$ is closed under the $\sqcap^S$-rule, when the graph G_a was initially defined, at least one of $C(a)$, $D(a)$ and $E(a)$ was in V_a. W.l.o.g., assume that all of them were in V_a. Since $C \sqcap D \sqcap E(a)$ does not have any $\sqcap$-successor in $\mathbb{G}_a$, then $C(a)$, $D(a)$ and $E(a)$ were all removed during the execution. Suppose that among $C(a)$, $D(a)$ and $E(a)$, $C(a)$ was removed after the removal of $D(a)$ and $E(a)$. Right before the removal of $C(a)$, $C(a)$ was the only $\sqcap$-successor of $C \sqcap D \sqcap E(a)$. By criterion (c_2) (Procedure 5, Line 12-16), removing $C(a)$

will lead to the removal of $C \sqcap D \sqcap E(a)$, contradicting that $C \sqcap D \sqcap E(a) \in \mathbb{B}_a$. Therefore, the claim (b_3) holds. ■

3.5.5 The Procedure Tight

The whole procedure of computing a tight envelope is given in Figure 3.6. We will show that by utilizing the basic set computed as in Section 3.5.4, the resulting set $\mathbb{F}^{(3)}$ is a tight envelope.

TIGHT:

(1) Compute the basic set $\mathbb{B}_{\mathbb{S}}$ for $\mathbb{S}$ according to Section 3.5.4. Let $\mathbb{F}^{(1)} := \mathbb{S} \cup \mathbb{B}_{\mathbb{S}}$. Note that $\mathbb{F}^{(1)} \setminus \mathbb{B}_{\mathbb{S}}$ contains all and only the role assertions in $\mathbb{S}$ (see (b_1) and the definition of $\mathbb{S}_a$).

(2) Apply the $\sqsubseteq^S$-rule exhaustively until no longer applicable [a]. The resulting set is denoted by $\mathbb{F}^{(2)}$.

(3) Apply $\exists^S_d$-rule exhaustively to $\mathbb{F}^{(2)}$ until no longer applicable. The resulting set is denoted by $\mathbb{F}^{(3)}$.

[a] Note that if $C \sqsubseteq D, D \sqsubseteq E \in \mathcal{T}^f$ and $E(a) \in \mathbb{F}^{(1)}$, then $C(a)$ could have been added to $\mathbb{F}^{(2)}$ either via an application of the $\sqsubseteq^S$-rule to $E(a) \in \mathbb{F}^{(1)}$ or via an application of the $\sqsubseteq^S$-rule to $D(a)$ after $D(a)$ is added to $\mathbb{F}^{(2)}$ via an application of the $\sqsubseteq^S$-rule to $E(a)$. Furthermore, since $\mathbb{F}^{(1)}$ is closed under the $\sqcap^S$-rule, for every $C(a) \in \mathbb{F}^{(2)} \setminus \mathbb{F}^{(1)}$, there is an atomic assertion $D(a) \in \mathbb{F}^{(1)}$ such that $C \sqsubseteq D \in \mathcal{T}^f$.

Figure 3.6 Procedure TIGHT

Some observations regarding the procedure TIGHT are listed below:

t_1. $\mathbb{F}^{(1)}$ is closed under the $\sqcap^S$-rule by (b_3) and since $\mathbb{B}_{\mathbb{S}} := \bigcup_{a \in \mathcal{O}_\Sigma} \mathbb{B}_a$.

t_2. $\mathbb{F}^{(2)}$ is closed under the $\sqsubseteq^S$-rule. Furthermore, if $C(a) \in \mathbb{F}^{(2)}$ is atomic and $C \in SubE(\mathbb{S}, \mathcal{T})$, then $C(a) \in \mathbb{F}^{(1)}$.

Proof Suppose that there is an atomic assertion $C(a) \in \mathbb{F}^{(2)} \setminus \mathbb{F}^{(1)}$ where $C \in SubE(\mathbb{S}, \mathcal{T})$. Then (by footnote (a)), there is an atomic assertion $D(a) \in \mathbb{F}^{(1)}$ such that $C \sqsubseteq D \in \mathcal{T}^f$. Both C and D being atomic and distinct implies that $C \not\equiv D$. Because $C \in SubE(\mathbb{S}, \mathcal{T})$, $D(a) \in \mathbb{E}^{(1)}_a$ and $C \sqsubseteq D \in \mathcal{T}^f$, both $C(a)$ and $D(a)$ are in $\mathbb{E}^{(1)}_a$. Moreover, since $C \not\equiv D$, by (g_2), we have $(D(a), C(a)) \in \mathcal{A}^{\mathcal{T}}_a$ in the initial graph G_a that we construct for computing the basic set. However, $C(a) \notin \mathbb{F}^{(1)}$, so $C(a)$ must have been removed by the algorithm

BASIC-SET during its execution. By criterion (c_1) (implemented by Procedure 5, Lines 1, 5-11), $D(a)$ would also have been removed, contradicting $D(a) \in \mathbb{F}^{(1)}$. Therefore, $C(a) \in \mathbb{F}^{(1)}$. ■

t_3. $\mathbb{F}^{(3)}$ is closed under $\exists_d^S$-rule and $\mathbb{F}^{(3)} \setminus \mathbb{F}^{(2)}$ contains only role assertions.

Lemma 3.5.4 *For every non-atomic assertion $C(a) \in \mathbb{F}^{(2)} \setminus \mathbb{F}^{(1)}$ where $C \in SubE \setminus SubE(\mathbb{S}, \mathcal{T})$, there is an assertion $D(a) \in \mathbb{F}^{(2)}$, $C \neq D$, such that $C \preceq D$.*

Proof Since $C(a) \in \mathbb{F}^{(2)} \setminus \mathbb{F}^{(1)}$, there is an assertion $E(a) \in \mathbb{F}^{(1)}$ such that $C \sqsubseteq E \in \mathcal{T}^f$. Furthermore, because $C \in SubE \setminus SubE(\mathbb{S}, \mathcal{T})$, C does not appear in the TBox. Since $C(a)$ is non-atomic, by (s_1), there is a concept $D \in SubE$, $D \neq C$, such that $C \preceq D \sqsubseteq E$. It then follows from (t_2) that $D(a) \in \mathbb{F}^{(2)}$. ■

Lemma 3.5.5 $\mathbb{F}^{(3)}$ *obtained from the procedure* TIGHT *is a secrecy envelope.*

Proof We argue that $\mathbb{F}^{(3)}$ is closed, i.e., none of the Secrecy Closure Rules rules is applicable to $\mathbb{F}^{(3)}$. By Theorem 3.4.10, $\mathbb{F}^{(3)}$ is a secrecy envelope.

- $\sqsubseteq^S$-rule: By (t_2) and (t_3) above, $\mathbb{F}^{(3)}$ is closed under the $\sqsubseteq^S$-rule.
- $\sqcap^S$-rule: Since $\mathbb{F}^{(2)}$ is obtained by exhaustively applying the $\sqsubseteq^S$-rule, for every $C(a) \in \mathbb{F}^{(2)} \setminus \mathbb{F}^{(1)}$, there is an assertion $D(a) \in \mathbb{F}^{(1)}$ such that $C \sqsubseteq D$. If $C(a)$ is atomic, then the $\sqcap^S$-rule is not applicable to $C(a)$. If C is non-atomic, then either $C \in SubE(\mathbb{S}, \mathcal{T})$ or $C \in SubE \setminus SubE(\mathbb{S}, \mathcal{T})$.

  (a) $C \in SubE(\mathbb{S}, \mathcal{T})$: If $C \preceq D$, then since $D(a) \in \mathbb{F}^{(1)}$ and $\mathbb{F}^{(1)}$ is closed under the $\sqcap^S$-rule, the $\sqcap^S$-rule is not applicable to $C(a)$. Now suppose that $C \not\preceq D$. Because $\mathbb{E}_a^{(1)}$ is closed under the $\sqsubseteq_1^S$-rule (the $\sqsubseteq^S$-rule restricted to $SubE(\mathbb{S}, \mathcal{T})$), and since $C \sqsubseteq D$ and $D(a) \in \mathbb{F}^{(1)}$ (in particular, $D(a) \in \mathbb{B}_a \subseteq \mathbb{E}_a^{(1)}$), it follows that $C(a)$ must be in the initial graph G_a. Then by the construction of G_a, in particular (g_2), we have $(D(a), C(a)) \in \mathcal{A}_a^{\mathcal{T}}$. Since $C(a) \notin \mathbb{F}^{(1)}$, it was removed by the algorithm BASIC-SET. However, in that case by criterion (c_1) (implemented by Procedure 5,

Lines 1, 5-11), $D(a)$ would also have to be removed. This contradicts $D(a) \in \mathbb{F}^{(1)}$. Therefore, the $\sqcap^S$-rule is not applicable to $C(a)$.

(b) $C \in SubE \backslash SubE(\mathbb{S}, \mathcal{T})$: By Lemma 3.5.4, there is an assertion $C_1(a) \in \mathbb{F}^{(2)}$, $C \neq C_1$, such that $C \preceq C_1$.

* If C_1 is atomic, then the $\sqcap^S$-rule is not applicable to $C(a)$.
* If C_1 is non-atomic and $C_1 \in SubE(\mathbb{S}, \mathcal{T})$, it then follows from case (a) that the $\sqcap^S$-rule is not applicable to $C_1(a)$ and so it is also not applicable to $C(a)$ in view of $C \preceq C_1$.
* If C_1 is non-atomic and $C_1 \in SubE \setminus SubE(\mathbb{S}, \mathcal{T})$, again by Lemma 3.5.4, there is an assertion $C_2(a) \in \mathbb{F}^{(2)}$, $C_1 \neq C_2$, such that $C_1 \preceq C_2$, and hence $C \preceq C_2$ and $C \neq C_2$. Note that the length of C_2 is smaller than that of C_1. Applying the same reasoning to C_2, eventually, there is an atomic assertion $C_k(a) \in \mathbb{F}^{(2)}$, $C \neq C_k$, such that $C \preceq C_k$. Therefore, the $\sqcap^S$-rule is not applicable to $C(a)$.

We have shown that $\mathbb{F}^{(2)} \setminus \mathbb{F}^{(1)}$ is closed under the $\sqcap^S$-rule. It then follows from (t_1) and (t_3) that $\mathbb{F}^{(3)}$ is closed under the $\sqcap^S$-rule as well.

- $\exists^S$-rule: Since $\mathbb{F}^{(3)}$ is closed under the $\exists^S_d$-rule by (t_3), it is also closed under the $\exists^S$-rule.

■

Theorem 3.5.6 $\mathbb{F}^{(3)}$ *obtained from the procedure* TIGHT *is a tight envelope.*

Proof By Lemma 3.5.5, $\mathbb{F}^{(3)}$ is an envelope. It suffices to show that for every $\alpha \in \mathbb{F}^{(3)}$, $((\mathcal{A}^f_\Sigma \setminus \mathbb{F}^{(3)}) \cup \{\alpha\})^f \cap \mathbb{S} \neq \emptyset$. Note that $\mathbb{F}^{(3)} = \mathbb{S} \cup (\mathbb{B}_\mathbb{S} \setminus \mathbb{S}) \cup (\mathbb{F}^{(2)} \setminus \mathbb{F}^{(1)}) \cup (\mathbb{F}^{(3)} \setminus \mathbb{F}^{(2)})$.

1. If $\alpha \in \mathbb{S}$, then $\alpha \in ((\mathcal{A}^f_\Sigma \setminus \mathbb{F}^{(3)}) \cup \{\alpha\})^f \cap \mathbb{S}$, and hence the intersection is not empty.

2. Let $\alpha = C(a) \in \mathbb{B}_\mathbb{S} \setminus \mathbb{S}$. By (b_2), there is a path p in $\mathbb{G}_a$ from $\mathbb{S}_a$ to $C(a)$ s.t. all non-atomic nodes on p have a unique $\sqcap$-successor (not necessarily on p). Suppose that the path p starts from $D(a) \in \mathbb{S}_a$ and that nodes on p are $C_1(a), ..., C_k(a)$ where $C_1(a) = D(a)$ and $C_k(a) = C(a)$. Then an edge $(C_i(a), C_{i+1}(a))(1 \leq i \leq k-1)$ is in either $\mathbb{A}^\mathcal{T}_a$ or $\mathbb{A}^\sqcap_a$.

(a) If $(C_i(a), C_{i+1}(a)) \in \mathbb{A}_a^{\mathcal{T}}$, then we have $C_{i+1} \sqsubseteq C_i \in \mathcal{T}^f$. With an application of the $\sqsubseteq^{\mathcal{A}}$-rule (see Figure 3.1), $C_i(a) \in ((\mathcal{A}_\Sigma^f \setminus \mathbb{F}^{(3)}) \cup \{C_{i+1}(a)\})^f$.

(b) If $(C_i(a), C_{i+1}(a)) \in \mathbb{A}_a^{\sqcap}$, then $C_i(a)$ is non-atomic and $C_i \preceq C_{i+1}$. Let $\mathbb{H}$ be the set of atomic concepts that trivially subsumes C_i but not C_{i+1}. Since $C_{i+1}(a)$ is the only $\sqcap$-successor of $C_i(a)$ in $\mathbb{G}_a$, for every atomic $E \in \mathbb{H}$, $E(a) \notin \mathbb{F}^{(1)}$. In view of $C_i(a) \in \mathbb{B}_{\mathbb{S}}$, we have $C_i \in SubE(\mathbb{S}, \mathcal{T})$ and so $\mathbb{H} \subseteq SubE(\mathbb{S}, \mathcal{T})$. It follows from observations (t_2) and (t_3) that for every $E \in \mathbb{H}$, $E(a) \notin \mathbb{F}^{(2)}$ and $E(a) \notin \mathbb{F}^{(3)} \setminus \mathbb{F}^{(2)}$. Therefore, $E(a) \in \mathcal{A}_\Sigma^f \setminus \mathbb{F}^{(3)}$. With an application of the $\sqcap^{\mathcal{A}}$-rule, $C_i(a) \in ((\mathcal{A}_\Sigma^f \setminus \mathbb{F}^{(3)}) \cup \{C_{i+1}(a)\})^f$.

Iterating these two cases, we conclude that $D(a) \in ((\mathcal{A}_\Sigma^f \setminus \mathbb{F}^{(3)}) \cup \{C(a)\})^f \cap \mathbb{S}$. Therefore, when $\alpha \in (\mathbb{B}_{\mathbb{S}} \setminus \mathbb{S})$, $((\mathcal{A}_\Sigma^f \setminus \mathbb{F}^{(3)}) \cup \{\alpha\})^f \cap \mathbb{S} \neq \emptyset$.

3. Let $\alpha = C(a) \in \mathbb{F}^{(2)} \setminus \mathbb{F}^{(1)}$. Then there is an assertion $D(a) \in \mathbb{F}^{(1)}$ such that $C \sqsubseteq D \in \mathcal{T}^f$. With an application of the $\sqsubseteq^{\mathcal{A}}$-rule, $D(a) \in ((\mathcal{A}_\Sigma^f \setminus \mathbb{F}^{(3)}) \cup \{\alpha\})^f$. However, by cases 1 and 2, $((\mathcal{A}_\Sigma^f \setminus \mathbb{F}^{(3)}) \cup \{D(a)\})^f \cap \mathbb{S} \neq \emptyset$. Since $((\mathcal{A}_\Sigma^f \setminus \mathbb{F}^{(3)}) \cup \{D(a)\})^f \subseteq ((\mathcal{A}_\Sigma^f \setminus \mathbb{F}^{(3)}) \cup \{\alpha\})^f$, we obtain $((\mathcal{A}_\Sigma^f \setminus \mathbb{F}^{(3)}) \cup \{\alpha\})^f \cap \mathbb{S} \neq \emptyset$.

4. If $\alpha \in \mathbb{F}^{(3)} \setminus \mathbb{F}^{(2)}$, then $\alpha = r(a, b)$ $(b \in \mathcal{O}_\Sigma)$ was obtained by an application of $\exists_d^S$-rule to an assertion $\exists r.C(a) \in \mathbb{F}^{(2)}$ with $\{r(a, b), C(b)\} \subseteq \mathcal{A}_\Sigma^f$. Hence, $C(b) \in \mathcal{A}_\Sigma^f \setminus \mathbb{F}^{(3)}$. By case 3, $\exists r.C(a) \in \mathbb{F}^{(2)}$ implies that $((\mathcal{A}_\Sigma^f \setminus \mathbb{F}^{(3)}) \cup \{\exists r.C(a)\})^f \cap \mathbb{S} \neq \emptyset$. An application of $\exists_1^{\mathcal{A}}$-rule to α and $C(b)$ (both in $(\mathcal{A}_\Sigma^f \setminus \mathbb{F}^{(3)}) \cup \{\alpha\}$) would result $\exists r.C(a) \in ((\mathcal{A}_\Sigma^f \setminus \mathbb{F}^{(3)}) \cup \{\alpha\})^f$. Thus, $((\mathcal{A}_\Sigma^f \setminus \mathbb{F}^{(3)}) \cup \{\exists r.C(a)\})^f \subseteq ((\mathcal{A}_\Sigma^f \setminus \mathbb{F}^{(3)}) \cup \{\alpha\})^f$, and therefore $((\mathcal{A}_\Sigma^f \setminus \mathbb{F}^{(3)}) \cup \{\alpha\})^f \cap \mathbb{S} \neq \emptyset$.

■

3.5.6 Experimental Comparison Result

We have presented two algorithms for computing tight envelopes, a naive one and an optimized one, the Procedure TIGHT. It is easy to see that both algorithms run in polynomial time. However, to compute a tight envelope, the naive approach uses the whole ABox, for each assertion in an envelope, to check whether it is necessary to be protected whereas the optimized algorithm depends much less on the ABox. Because of this, the optimized algorithm should run faster when the sizes of the TBox and the secrecy set are much smaller than that of the

ABox. We implemented Λ_S for computing envelopes, and the two algorithms for computing tight envelopes to see how they run in practice. Next we compare the experiment results for the algorithms that build envelopes or tight envelopes during the pre-query stage.

Let $N_{\mathcal{T}}$ be the number of subsumptions (in normal form) in $\mathcal{T}$, $N_{\mathcal{A}}$ be the number of assertions in $\mathcal{A}$, N_I be the number of individuals in $\mathcal{A}$ and $N_{\mathbb{S}}$ be the number of assertions in $\mathbb{S}$. Below we list experiment results showing the running time (in seconds) of computing an envelope, computing a tight envelope using the naive approach and computing a tight envelope using Procedure TIGHT. The experiments were run on a personal computer with Intel®Core$^{\text{TM}}$2 duo processor at 2.40GHz and 3GB of RAM.

Table 3.1 Experiments result

Line#	$N_{\mathcal{T}}$	$N_{\mathcal{A}}$	N_I	$N_{\mathbb{S}}$	$t_{envelope}$	t_{naive}	t_{TIGHT}
1	45	120	2	25	0.19	6.56	0.16
2	45	210	12	14	0.02	46.29	0.37
3	103	210	12	14	0.05	104.70	0.14
4	103	210	12	56	0.18	162.07	0.55
5	45	240	2	11	0.04	22.17	0.05
6	45	418	22	13	0.02	34.29	0.01
7	133	418	24	13	0.31	295.95	0.08
8	133	418	24	101	0.55	409.01	0.28
9	45	400	2	11	0.06	109.62	0.07
10	45	400	40	11	0.02	42.09	0.02
11	173	400	40	11	0.03	47.55	0.01
12	173	400	40	165	0.11	685.86	0.14
13	45	2340	40	11	0.06	31524.47	0.05

From the table above, we can see that the time for Procedure TIGHT for computing a tight envelope is comparable with the time for computing an envelope (not necessarily tight). However, computing a tight envelope using the naive approach is much more costly. In general, when the size of $\mathbb{S}$ increases, the time of computation increases since the corresponding envelope is larger (see lines 3 vs 4, 7 vs 8, 11 vs 12). When the size of the ABox $\mathcal{A}$ increases and the size of the secrecy set $\mathbb{S}$ decreases, the time for computing an envelope or a tight envelope using Procedure TIGHT decreases, but the time for computing a tight envelope using the naive

approach increases because the naive approach depends more on the ABox $\mathcal{A}$ (see lines 1, 2, 5, 6). An extreme example is shown in Line 13.

3.6 Queries

Recall that we have assumed that given a KB Σ and a secrecy set $\mathbb{S}$, the assertional closure $\mathcal{A}^f$, the subsumption set $\mathcal{T}^f$ and the initial secrecy envelope $\mathbb{E}^f$ for $\mathbb{S}$ are precomputed at the pre-query stage (as described in detail in Section 3.4). Thus, when the concept C in a query $C(a)$ is in $SubE$, the answer of the query is "Yes" if $C(a) \in \mathcal{A}^f_\Sigma$ and $C(a) \notin \mathbb{E}^f$ and it is "Unknown" otherwise. It is obvious that such a query can be answered in linear time in the size of $\mathcal{A}^f_\Sigma$.

When $C \notin SubE$, $SubE$ will be expanded by adding the subexpressions of C, the assertional closure $\mathcal{A}^f$ and then the secrecy envelope $\mathbb{E}^f$ will be updated. The corresponding procedure SPQA is listed in Algorithm 7. The input to SPQA includes the TBox $\mathcal{T}^f$ in normal form, the precomputed assertional closure $\mathcal{A}^f$, the query $C(a)$ and the precomputed secrecy envelope $\mathbb{E}^f$.

Algorithm 7 SPQA($\mathcal{T}^f, \mathcal{A}^f, C(a), \mathbb{E}^f$)

1: **if** $C \notin SubE$ **then**
2: compute $sub(C)$ (see Section 3.4.1)
3: $SubE = SubE \cup sub(C)$
4: expand $\mathcal{A}^f$ and $\mathcal{T}^f$ to $SubE$
5: expand the secrecy envelope $\mathbb{E}^f$ to $SubE$ by exhaustively applying the $\sqsubseteq^S$-rule
6: **end if**
7: **if** $C(a) \in \mathcal{A}^f_\Sigma$ and $C(a) \notin \mathbb{E}^f$ **then**
8: **return** "Yes"
9: **end if**
10: **return** "Unknown"

Lines 1-6 of SPQA deal with the case when the concept expression of a query is not in $SubE$. Line 2 computes the set of subexpressions of the concept C as defined in Section 3.4.1 and Line 3 expands $SubE$ by adding expressions in $sub(C) \setminus SubE$. The expanded $SubE$ is then used to update $\mathcal{A}^f$ by applying assertion expansion rules (Figure 3.1) until none of them is applicable. $\mathcal{T}^f$ is also updated accordingly, as indicated in Line 4. As a consequence, some secrecy closure rules may become applicable, implying that the current $\mathbb{E}^f$ may no longer be a

secrecy envelope. Therefore, we apply the $\sqsubseteq^S$-rule exhaustively in Line 5. We next show that once the execution of Line 5 is completed, we obtain a tight envelope. Let $SubE'$ be the newly expanded set of subexpressions after the application of Line 3, $\mathcal{A}'$ the assertional closure of Σ after the application of Line 4 and $\mathbb{E}'$ the expansion of $\mathbb{E}^f$ after the application of Line 5. We first argue that $\mathbb{E}'$ is an envelope.

- It is obvious that the $\sqsubseteq^S$-rule is not applicable.
- Suppose that the $\sqcap^S$-rule is applicable to an assertion $C(a) \in \mathbb{E}'$. Since $\mathbb{E}^f$ is closed under the $\sqcap^S$-rule, $C(a) \in \mathbb{E}' \setminus \mathbb{E}^f$. Therefore, $C \in SubE' \setminus SubE$ and C does not appear in the TBox $\mathcal{T}$. $C(a)$ being non-atomic and not a role assertion implies that $C(a)$ is obtained by an application of the $\sqsubseteq^S$-rule. So there is an assertion $D(a) \in \mathbb{E}^f$ such that $C \sqsubseteq D \in \mathcal{T}^f$. It then follows from (s_1) (see Section 3.5.3) that there is a concept $E \in SubE'$ such that $C \neq E$ and $C \preceq E \sqsubseteq D$. If $E \in SubE$, then $E(a) \in \mathbb{E}^f$ because $\mathbb{E}^f$ is closed under the $\sqsubseteq^S$-rule w.r.t. $SubE$. Since $\mathbb{E}^f$ is also closed under the $\sqcap^S$-rule w.r.t. $SubE$, there is an atomic assertion $E'(a) \in \mathbb{E}^f$ such that $E \preceq E'$, and hence, the $\sqcap^S$-rule is not applicable to $C(a)$. If $E \in SubE' \setminus SubE$, applying the same reasoning to E, we can eventually conclude that the $\sqcap^S$-rule is not applicable to ($E(a)$ and) $C(a)$. It follows that the $\sqcap^S$-rule is not applicable to $\mathbb{E}'$.
- Suppose that the $\exists^S_d$-rule is applicable to an assertion $\exists r.C(a)$. By the previous case, the $\sqcap^S$-rule is not applicable to any assertion in $\mathbb{E}'$. Hence, $\exists r.C(a)$ was obtained by an application of the $\sqsubseteq^S$-rule and there is an assertion $D(a) \in \mathbb{E}^f$ such that $\exists r.C \sqsubseteq D \in \mathcal{T}^f$. Since $\mathbb{E}^f$ is closed under the $\exists^S_d$-rule, $\exists r.C(a) \in \mathbb{E}' \setminus \mathbb{E}^f$. Therefore, $\exists r.C \in SubE' \setminus SubE$ and hence $\exists r.C$ does not appear in the TBox $\mathcal{T}$. It follows from (s_2) (see Section 3.5.3) that there is a concept $\exists r.E \in SubE'$ such that $\exists r.C \sqsubseteq \exists r.E \sqsubseteq D$, $C \sqsubseteq E$ and $C \neq E$. If $\exists r.E \in SubE$, then $\exists r.E(a) \in \mathbb{E}^f$ because $\mathbb{E}^f$ is closed under the $\sqsubseteq^S$-rule w.r.t. $SubE$ and $D(a) \in \mathbb{E}^f$. Since the $\exists^S_d$-rule is applicable to $\exists r.C(a)$, there is an individual $b \in \mathcal{O}_\Sigma$ such that $r(a,b), C(b) \in \mathcal{A}'$. By the observation (f_1) in Section 3.4.2, $r(a,b) \in \mathcal{A}$. Moreover, $C(b) \in \mathcal{A}'$ implies $E(b) \in \mathcal{A}'$. Since $E \in SubE$, we have $E(b) \in \mathcal{A}^f$. Because $\mathbb{E}^f$ is closed under the $\exists^S_d$-rule, $r(a,b) \in \mathbb{E}^f$. This contradicts the applicability of the $\exists^S_d$-rule

to $\exists r.C(a)$. Therefore, the $\exists_d^S$-rule is not applicable to $\exists r.C(a)$. If $\exists r.E \in SubE' \setminus SubE$, applying the same reasoning to $\exists r.E$, we can eventually conclude that the $\exists_d^S$-rule is not applicable to $\exists r.E(a)$ and so it is not applicable to $\exists r.C(a)$. It follows that the $\exists_d^S$-rule is not applicable to $\mathbb{E}'$.

Since no D-secrecy closure rules are applicable, $\mathbb{E}'$ is closed and so it is an envelope. Furthermore, since all assertions in $\mathbb{E}' \setminus \mathbb{E}^f$ are obtained by applications of the $\sqsubseteq^S$-rule, for every $C(a) \in \mathbb{E}' \setminus \mathbb{E}^f$, there is $D(a) \in \mathbb{E}^f$ such that $C \sqsubseteq D$, and so $((\mathcal{A}' \setminus \mathbb{E}') \cup \{D(a)\})^f \subseteq ((\mathcal{A}' \setminus \mathbb{E}') \cup \{C(a)\})^f$. Since $\mathbb{E}^f$ is an envelope w.r.t. $SubE$ and $D(a) \in \mathbb{E}^f$, $((\mathcal{A}^f \setminus \mathbb{E}^f) \cup \{D(a)\})^f \cap \mathbb{S} \neq \emptyset$. Moreover, because $(\mathbb{E}' \setminus \mathbb{E}^f) \subseteq (\mathcal{A}' \setminus \mathcal{A}^f)$, we have $\mathcal{A}^f \setminus \mathbb{E}^f \subseteq \mathcal{A}' \setminus \mathbb{E}'$ and so $((\mathcal{A}' \setminus \mathbb{E}') \cup \{C(a)\})^f \supseteq ((\mathcal{A}' \setminus \mathbb{E}') \cup \{D(a)\})^f \supseteq ((\mathcal{A}^f \setminus \mathbb{E}^f) \cup \{D(a)\})^f$. It follows that $((\mathcal{A}' \setminus \mathbb{E}') \cup \{C(a)\})^f \cap \mathbb{S} \neq \emptyset$. Hence, $\mathbb{E}'$ is tight.

Example 3.6.1 *Recall that we have a knowledge base* $\Sigma_1 = \langle \mathcal{A}_1, \mathcal{T}_1 \rangle$ *and the secrecy set* $\mathbb{S}_1 = \{CancerRisk(Jane)\}$ *in our running Examples 3.1.1, 3.3.3, 3.4.7, 3.4.1, 3.4.2 and 3.4.9. The assertional closure* $\mathcal{A}_1^f$ *of* Σ_1 *is listed in Example 3.4.2. The envelope* $\mathbb{E}_1^f$ *of* $\mathbb{S}_1$ *in Example 3.4.9 is a tight envelope. If the querying agent asks the query Reimburse(Jane), Reimburse(Jane)*$\in \mathcal{A}_{1\Sigma_1}^f \setminus \mathbb{E}_1^f$*, the answer to the query is "Yes". If the querying agent asks the query CancerRisk(Jane), since CancerRisk(Jane)*$\notin \mathcal{A}_{1\Sigma_1}^f \setminus \mathbb{E}_1^f$*, the answer to the query is "Unknown".* ■

Example 3.6.2 *Recall that in Examples 3.5.2 and 3.5.3, we have a KB* $\Sigma = \langle \mathcal{A}, \mathcal{T} \rangle$ *with*

- $\mathcal{A} = \{F(a),\ C(a),\ \exists r.\exists s.(D \sqcap D')(a),\ C \sqcap C_1(a),\ C \sqcap C_2(a),\ ...,\ C \sqcap C_k(a),\ E_1 \sqcap E_2(a)\}$,
- $\mathcal{T} = \{E_1 \sqcap E_2 \sqsubseteq C,\ \exists r.M \sqsubseteq F,\ \exists s.D \sqsubseteq M\}$, *and*
- $\mathbb{S} = \{F(a),\ C \sqcap \exists r.\exists s.(D \sqcap D')(a)\}$.

Starting with the basic set $\mathbb{B}_a$ *computed in Example 3.5.3, we obtain a tight envelope* $\mathbb{E}^f = \mathbb{S} \cup \{\exists r.M(a),\ \exists r.\exists s.(D \sqcap D')(a)\}$. *Suppose that the querying agent asks the query* $\exists r.\exists s.D(a)$. *Note that* $\exists r.\exists s.D \notin SubE$ *(see Example 3.5.2). We add* $\exists r.\exists s.D$ *into SubE and accordingly,* $\exists r.\exists s.D(a)$ *is added into* $\mathcal{A}^f$ *in view of* $r(a,x), \exists s.D(x) \in \mathcal{A}^f$. *Since* $\exists r.M(a) \in \mathbb{E}^f$ *and* $\exists r.\exists s.D \sqsubseteq \exists r.M$, $\exists r.\exists s.D(a)$ *is added into* $\mathbb{E}'$. *The final answer to the query* $\exists r.\exists s.D(a)$ *is "Unknown".* ■

3.7 Related Work

In this section, we outline some related work in the area of privacy and security in information systems, including web-based information systems.

Early work on information protection led to the creation of a multi-level security model for mandatory access control (MAC) [8, 9] where each data object belongs to a security class and each user is assigned a clearance for a security class. By posing restrictions on reads and writes of all data objects, such models are built to ensure that no secret information flows between different users. While MAC solves the problem of an unauthorized user tricking an authorized user into disclosing sensitive data that a discretionary access control (DAC) model may have, it restricts the security granularity at object level. Role-Based Access Control (RBAC) [64] is an alternative approach to both DAC and MAC which provides authorization on operations (rather than objects). The primary focus of these work has been on access control mechanisms that prohibit access to sensitive information.

Chin [65] studied the problem of unintended disclosure of information about particular individuals that can be inferred from statistics made available about groups of individuals. Grau and Horrocks [66] have recently introduced a framework that combines logic and probabilistic approaches to privacy-preserving query answering from databases. A growing body of work on data linkage [67, 68] addresses the problem of disclosure of personal data from aggregate information or from separately released, non-confidential information about an individual. Work on privacy preserving data mining [69, 70, 71] addresses the design of algorithms for constructing predictive models that describe shared characteristics of groups of individuals, e.g., patients in a clinical trial, without revealing information about specific individuals, e.g., clinical records of individual participants in clinical trials. The primary focus of such work is on preventing inference that is typically of a probabilistic nature about individual records from statistical or aggregate information about a population. Our paper focuses on guaranteeing that secret information is not compromised by queries answered using deductive inferences that are of a purely logical nature. It remains to be seen (i.e., it is an open question) whether these two approaches can be usefully combined in the context of knowledge bases with the open world

assumption.

Research on encryption of sensitive information focuses on preventing unauthorized access to such information using cryptographic protocols. Giereth [72] has studied techniques for hiding a fragment of an RDF document by encrypting it while the rest of the document remains publicly readable. Abel et al. [73] have proposed a policy-based control of access to RDF stores. Baader et al. [74] have recently introduced an approach to reasoning with ontologies in the presence of access restrictions on specific axioms. Access control policies and encryption techniques can be used to prohibit access to sensitive information. More recent work on logic-based authorization frameworks [64, 75, 76] focuses on policy languages that go beyond traditional access control methods to address obligation, provision, and delegation of authorization as a basis for protecting sensitive information in computer systems, databases and networked information systems (see [12] for a survey). Most of the work on policy languages for the web [13, 14, 15, 77, 16, 78, 17, 79, 80] focuses on specifying syntax-based restrictions on access to specific resources or operations on the web. Halpern and Weissman [81] have proposed a first order logic based approach to reasoning about policies. The main focus of these models is the control of direct access to sensitive information. Our paper focuses on logic-based mechanisms for answering queries using secret information whenever it is possible to do so without compromising their confidentiality.

Farkas et al. [58] have proposed a privacy information flow model to represent information flow and privacy requirements that are enforced by a privacy mediator which guarantees that users cannot logically infer information that violates the privacy requirements. They assume a tree-like, semistructured data model, selection-projection queries and domain knowledge, represented as Horn clause constraints, and a domain KB consisting of assertions in the form of Horn clauses. Jain and Farkas [11] have proposed an elegant RDF authorization model that can selectively control access to RDF triples using a pre-specified set of patterns that can be used to assign a secrecy label to each (stored or inferred) RDF triple. Our approach can be generalized to many KBs that are equipped with a sound and complete reasoning algorithm.

Several authors have explored the use of "cover stories" i.e., lies introduced into a multilevel database in order to protect some secret information in the database [82, 83, 84]. The

controlled query evaluation (CQE) framework first introduced by Sicherman et al. [19] offers a mechanism for answering database queries without revealing secrets. This framework which has been explored extensively by Biskup and colleagues in a series of papers [85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96] includes: (a) policies that specify secrets (queries for which all possible answers must be protected), potential secrets (queries for which only some of the answers must be protected); (b) policies for answering queries so as to protect secrets and potential secrets including lying in response to a query, refusing to answer a query, and their combinations; and (c) alternative assumptions regarding whether or not the querying agent is aware of the queries whose answers the KB is trying to protect. Recent work within this framework has introduced techniques using SAT-solvers and constraint solvers for preprocessing the databases so that the resulting database can answer queries in a manner that is consistent with the specified policies. However, with the exception of [94], this work has focused on protecting secrets in (typically relational) databases under the closed world assumption. We focus on answering queries against KBs under the open world assumption with emphasis on scenarios where lying is either not desirable or prohibited (e.g., in the case of the physician, the pharmacy, and the insurance company sharing information with each other in the scenario described in Example 1, or different government agencies sharing information with each other under the law).

3.8 Conclusion and Future Work

In this chapter, we studied the problem of answering queries against a knowledge base that contains secret information. Based on the OWA, we designed reasoners that hide truthful answers to the queries that if faithfully answered, may compromise the secrecy. One such a reasoner was designed using the lazy evaluation. This approach checks the query history each time a query is posed, when the query history gets longer, the response time to a query gets longer as well and hence the approach becomes less and less appealing over time. Because of this, we proposed to maintain a secrecy system that precomputes an envelope used to protect secret information so that the information outside the envelope cannot deduce any secret. Once an envelope is present, a query will be truthfully answered if it is outside the envelope. A general framework for the solution to the problem was provided. We discussed the relationship

between a secrecy-preserving reasoner and an envelope. To answer queries as informatively as possible, we aim at computing an envelope as small as possible. Unfortunately, given a language, deciding the smallest envelope may be NP-complete. Depending on the underlying language and its native inference system, optimization can be designed to compute a tight envelope in the sense that removing any information from it will compromise secrecy.

We applied the general framework to the Description Logic $\mathcal{EL}$. Given an $\mathcal{EL}$ knowledge base Σ and a secrecy set $\mathbb{S}$, in order to answer queries, we first construct a secrecy maintenance system which contains a finite set of consequences $\mathcal{A}^f$ of Σ and a secrecy envelope $\mathbb{E}^f$ which is used to protect $\mathbb{S}$. Both $\mathcal{A}^f$ and $\mathbb{E}^f$ are restricted to $SubE$, a finite set of some subexpressions in Σ or $\mathbb{S}$. To answer queries as informatively as possible, we aimed to make $\mathbb{E}^f$ as small as possible. Since computing the smallest envelope in $\mathcal{EL}$ is also NP-complete, we have presented two algorithms for computing tight envelopes: a naive algorithm and an optimized version, procedure TIGHT. We compared the complexities of these two algorithms, designed experiments and concluded that the optimized algorithm indeed is more efficient for applications whenever the sizes of the TBox and the secrecy set are much smaller than that of the ABox, which is typical in many applications. When a query $C(a)$ is posed to Σ, we first check whether the concept $C \in SubE$. If it doesn't, we expand the whole secrecy maintenance system by adding the subexpressions of C. We showed that after the expansion of the maintenance system, the resulting envelope is still tight. Then the answer of a query α is "Yes" if $\alpha \in \mathcal{A}^f \setminus \mathbb{E}^f$ and "Unknown" otherwise.

Instead of forbidding the use of the secret information in answering queries as is done in access control methods, our approach uses secrets in the deduction process while providing informative answers, whenever it is possible to do so without compromising secrecy.

In this chapter, we have focused on how to answer queries while preserving secrecy given a secrecy set. A future extension of the current work can be developing strategies to specify secrets (policy specifications) and generate secrecy sets in an automated way. For example, a policy like "Whether or not a patient x is at risk of developing cancer must be kept secret" specifies a requirement about a whole range of assertions which must be protected (rather than a single assertion). This is rather different than most of the work on policy languages for the

web which focuses on specifying syntax-based restrictions on access to specific resources or operations on the web (see Section 3.7). We have also assumed that the secrecy set to be finite in the current work. To develop strategies that can deal with infinite secrecy set is also a future direction.

CHAPTER 4. OPEN WORLD SECRECY-PRESERVING QUERY ANSWERING: THE MULTIPLE QUERYING AGENTS SETTING

Abstract

Many applications require a knowledge base (KB) that contains secrets to answer queries posed against it using secrets, whenever it is possible to do so, without revealing secrets. We consider this problem under the OWA in a setting with multiple querying agents $M_1, ..., M_m$ that can pose queries against the KB $\mathcal{K}$ and selectively share answers that they receive from $\mathcal{K}$ with one or more other querying agents. We assume that for each M_i, the KB has a pre-specified set of secrets S_i that need to be protected from M_i. Communication between querying agents is modeled by a communication graph, a DAG with self-loops. We introduce a general framework and propose an approach to secrecy-preserving query answering. The idea is to hide the truthful answer from a querying agent M_i, by feigning ignorance without lying, i.e., to provide the answer 'Unknown' to a query q if it needs to be protected. Under the OWA, a querying agent cannot distinguish whether q is being protected or it cannot be inferred from $\mathcal{K}$. We precompute a set of envelopes E_1, ..., E_m (restricted to a finite set Φ of formulae that are entailed by $\mathcal{K}$) such that $S_i \subseteq E_i$ and a query α posed by agent M_i can be answered truthfully only if $\alpha \notin E_i$. The envelope is updated as needed. We illustrate this approach in the case of Propositional Horn KBs.

4.1 Introduction

This chapter considers the SPQA problem under the OWA in a setting with multiple querying agents. Given a KB $\mathcal{K}$ and a set of querying agents $\mathcal{M} = \{M_1, M_2, ..., M_m\}$, we assume that for each M_i, there is a pre-specified set of secrets S_i that $\mathcal{K}$ needs to protect from M_i. We further assume that each M_i can selectively share answers that it receives (in response to queries posed by it) from $\mathcal{K}$ with one or more other querying agents. We model communication (answer sharing) between querying agents using a *communication graph* (CG), a DAG with self-loops, in which a node in the CG corresponds to a querying agent and a directed edge from node M_i to M_j in CG denotes the ability of M_i to share with M_j the answers it receives from $\mathcal{K}$ (but not answers shared with it by other querying agents, unless they happen to be also received directly from $\mathcal{K}$). We introduce a general framework and propose a solution to the SPQA problem in this setting. Under OWA, the answer to a query q posed by an agent M_i against $\mathcal{K}$ can be "Yes" (q can be deduced from $\mathcal{K}$), "No" ($\neg q$ can be deduced from $\mathcal{K}$), or "Unknown" (Neither q nor $\neg q$ can be deduced from $\mathcal{K}$). The basic idea is to hide the truthful answer from M_i, when it is necessary to do so by feigning ignorance without lying; i.e., to provide the answer 'Unknown' to a query q whenever providing to M_i, the truthful answer to q would compromise any secret that the KB $\mathcal{K}$ is obliged to protect from *any* of the querying agents in $\mathcal{M}$. Under the OWA, a querying agent cannot distinguish between the following two scenarios: the answer to q (i) is being protected; and (ii) cannot be inferred from $\mathcal{K}$.

A simple way of defining a secrecy-preserving reasoner is to maintain a history that, for each agent, logs the sequence of queries and the corresponding answers. When a new query q is posed by an agent M_i, the reasoner tests whether the truthful answer to q together with answers to previous queries that M_i has obtained from directly by querying the KB $\mathcal{K}$ or indirectly from other querying agents (its predecessors in the CG) compromises a secret that $\mathcal{K}$ is obliged to protect against any of the querying agents in $\mathcal{M}$. If it does, M_i receives the answer "Unknown" in response to the query q. Otherwise, q will be truthfully answered. A "Yes" or "No" answer can be shared by M_i with its successors in CG. We call this approach *lazy evaluation*. Because lazy evaluation requires checking the answer to each query posed by

each querying agent against a query history, it takes more and more time to answer a query as the size of the history grows over time. Hence, we propose a different approach: we precompute a *secrecy envelope* (or simply *envelope*) $\mathbb{E} = \{E_1, E_2, ..., E_m\}$ (restricted to a finite set Φ of formulae that are entailed by the KB) such that $S_i \subseteq E_i$ and a query α posed by agent M_i can be answered truthfully only if $\alpha \notin E_i$. The envelope is updated as needed as queries are answered. Our notion of an envelope is illustrated by the following example (in the simplified setting of a single querying agent): Consider a formula $\alpha \wedge \beta$ in propositional logic that needs to be protected from the querying agent. Suppose that both α and β are entailed by the KB. Obviously, $\alpha \wedge \beta$ is entailed by the KB. If we need to protect $\alpha \wedge \beta$ from the querying agent, we must disrupt *every* proof of $\alpha \wedge \beta$, protect either α or β, i.e., if $\alpha \wedge \beta$ is a secret, either α or β must be protected, and hence placed in the envelope. It is easy to show that an envelope always exists. The challenge is to construct an envelope that is guaranteed to protect secrets (in the sense described above, in the setting with multiple querying agents that can selectively share answers with other querying agents) while allowing queries to be answered as informatively as possible (feigning ignorance only when doing so is necessary to protect a secret). This requires constructing an envelope that is as small as possible. Unfortunately, in general, computing the smallest envelope is NP-hard or worse (see Section 4.5). Hence, we settle on computing and maintaining a *tight envelope*, i.e., an envelope that is *minimal* in that no formula can be removed from it without risk of a secret being compromised. When an envelope is finite, a general way of obtaining a tight envelope is to evaluate every assertion in the envelope. If removing an assertion still results an envelope, it will be removed from the envelope. This process ends with a tight envelope. Since computing a tight envelope is an optimization problem, depending on the language, and/or properties of the communication graph, some strategy may be designed to guide the computation of an envelope so that when an envelope is constructed, it is tight. we consider an example of a communication graph that is an inverted forest (with self-loops) in which a strategy leads to a tight envelope.

The rest of the chapter is organized as follows: Section 4.2 formally introduces a general framework for SPQA with multiple querying agents under OWA. An algorithm for lazy evaluation is provided in Section 4.3. Section 4.4 proves some properties of envelopes that are useful

in computing and updating envelopes, and in using envelopes to answer queries and considers an interesting special case of communication graphs (inverted forests). Section 4.5 illustrates an application of the framework and results of preceding sections to solve the SPQA problem in the simple yet practically useful case of Propositional Horn KBs.

4.2 Multiagent Secrecy-preserving Framework

Let $\mathcal{L}$ be some appropriate (logic-based) description language which, for simplicity of notation, we also view as a set of sentences (viz., all sentences expressible in the language $\mathcal{L}$). Let $\models_{\mathcal{L}}$ (or, just $\models$) be a Tarski-style semantic entailment for $\mathcal{L}$ and suppose that $\vdash_{\mathcal{L}}$ (or, just $\vdash$) is an inference system for the language $\mathcal{L}$ that is sound and complete, with respect to $\models$.

For $\Gamma \subseteq \mathcal{L}$, we write $\Gamma^+ = \{\alpha \mid \Gamma \vdash_{\mathcal{L}} \alpha\}$ for the *inferential closure* of a set of formulas Γ and we say that Γ is *inferentially closed* if $\Gamma^+ = \Gamma$. A formula $\alpha \in \mathcal{L}$ is a *tautology* if $\models \alpha$. The set of all tautologies will be denoted by $\mathcal{T}_{aut}$.

Definition 4.2.1 *A* knowledge base *(abbreviated, KB) over $\mathcal{L}$ is a triple $\mathcal{K} \triangleq \langle K, \mathcal{Q}, \Omega \rangle$ where*

- *K is a consistent finite subset $K \subseteq_f \mathcal{L}$. It represents the information that is explicitly stored in $\mathcal{K}$. K^+ represents all the information ("knowledge") that the KB $\mathcal{K}$ can infer. In the sequel, we shall refer to both $\mathcal{K}$ and K as a knowledge base (KB).*

- *$\mathcal{Q}$, the* query space *of $\mathcal{K}$, is a subset: $K^+ \subseteq \mathcal{Q} \subseteq \mathcal{L}$ representing the set of all queries that can be "legally" posed against $\mathcal{K}$. We do not insist that $\mathcal{Q} = \mathcal{L}$ which allows us to account for possible restrictions that may be imposed on the querying agents.*

- *Ω, is the* answer space. *In most cases $\Omega = \{Y, N, U\}$ (for "Yes", "No" and "Unknown", respectively). The "classical" answer space is $\Omega = \{Y, N\}$.*

Let $\mathcal{K} = \langle K, \mathcal{Q}, \Omega \rangle$ be a KB and let $\mathcal{M} = \{M_1, M_2, ..., M_m\}$ be an m-set of *querying agents* who may pose queries to the KB. For each querying agent M_i there is a corresponding *secrecy set* consisting of non-tautological statements which the KB is supposed to protect against agent M_i. The querying agents may share the answers they obtain from $\mathcal{K}$ with other querying agents. The sharing is constrained by means of a *communication graph* $(\mathcal{M}, \mathcal{A})$ which

is an acyclic directed graph (except for all the self-loops) such that the existence of an edge $(M_i, M_j) \in \mathcal{A}$ means that querying agent M_i shares with agent M_j all the non-U answers he receives from the KB [1]. As a technicality, we stipulate that the communication graph includes all the self-loops; that is, for every $M_i \in \mathcal{M}$, $(M_i, M_i) \in \mathcal{A}$.

Definition 4.2.2 *A* secrecy structure on *a KB $\mathcal{K}$ is a triple $\mathcal{S} \triangleq \langle \mathcal{M}, \mathbb{S}, \mathcal{G} \rangle$ where*

- $\mathcal{M} = \{M_1, M_2, ..., M_m\}$ *is an m-set of querying agents who may pose queries to $\mathcal{K}$,*
- $\mathbb{S} = \{S_1, S_2, ..., S_m\}$ *is a collection of secrecy sets, one for each querying agent, where for all $1 \leq i \leq m$, $S_i \subseteq K^+ \setminus \mathcal{T}_{aut}$, and*
- $\mathcal{G} = (\mathcal{M}, \mathcal{A})$ *is a directed communication graph which is acyclic except for all the self-loops.*

Protection of secrets is accomplished by means of $\mathcal{K}$*-reasoner* $\mathcal{R} : \mathcal{Q} \times \mathcal{M} \to \Omega$, a *reasoning algorithm* "attached" to $\mathcal{K}$, which for a query $q \in \mathcal{Q}$ and a querying agent $M_i \in \mathcal{M}$ provides an answer $\mathcal{R}(q, M_i) \in \Omega$ back to M_i. For any $B \in \Omega = \{Y, N, U\}$, we write $\mathcal{Q}_B^i = \{\alpha \in \mathcal{Q} \mid \mathcal{R}(\alpha, M_i) = B\}$, for the set of all M_i-queries to which the $\mathcal{K}$-reasoner returns the answer B; similarly, for any $B \in \{Y, N\}$, we write $\mathcal{P}_B^i = \bigcup_{j:(M_j, M_i) \in \mathcal{A}} \mathcal{Q}_B^j$, for the set of all B-queries that agent M_i obtains from its predecessors. Note that, as stated above, an agent $M_i \in \mathcal{M}$ can pass to its successors only answers to queries in $\mathcal{Q}_Y^i$ or $\mathcal{Q}_N^i$; for a query $\alpha \in \mathcal{Q}_U^i \cap (\mathcal{P}_Y^i \cup \mathcal{P}_N^i)$, even though agent M_i can infer the answer to α, it is not allowed to pass that answer to its successors. The following definition attempts to capture and formalize the whole secrecy framework as discussed above.

Definition 4.2.3 *A* multi-agent secrecy-preserving query answering (MSQ) *system is a triple $\langle \mathcal{K}, \mathcal{S}, \mathcal{R} \rangle$ where*

- $\mathcal{K} = \langle K, \mathcal{Q}, \Omega \rangle$ *is a KB,*
- $\mathcal{S} = \langle \mathcal{M}, \mathbb{S}, \mathcal{G} \rangle$ *is a secrecy structure on $\mathcal{K}$, and*

[1] The case when an agent is allowed to share query-answers obtained from other agents instead of only answers obtained from the KB can be reduced to the current problem by using the transitive closure of the communication graph rather than the original graph.

- $\mathcal{R}$ *is a* $\mathcal{K}$*-reasoner satisfying the following axioms: for all* $1 \leq i \leq m$*,*

  - ***[Yes-Axiom]*** $\mathcal{Q}_Y^i \subseteq K^+$*;*
  - ***[No-Axiom]*** $\mathcal{Q}_N^i = \{\neg\alpha \mid \ \alpha \in \mathcal{Q}_Y^i\}$*;*
  - ***[Closure Axiom]*** $(\mathcal{Q}_Y^i)^+ = \mathcal{Q}_Y^i$*;*
  - ***[Secrecy Axiom]*** $(\mathcal{P}_Y^i)^+ \cap S_i = \emptyset$*.*

  A $\mathcal{K}$*-reasoner satisfying the above four axioms is termed a* secrecy-preserving reasoner*.*

The Yes-Axiom ensures that every Y-query is provable from K. The No-Axiom enforces a match between the Y-queries and the N-queries. Note that the No-Axiom implies that $\mathcal{Q}_U^i$ is closed under negation: $\neg\mathcal{Q}_U^i = \mathcal{Q}_U^i$. The Closure Axiom requires that any consequence of a set of Y-queries that a querying agent obtains from the KB be a Y-query, i.e., $\mathcal{Q}_Y^i$ is inferentially closed. Finally, the Secrecy Axiom ensures that any combination of Y-answers that agent M_i obtains from its predecessors does not compromise any secrets that need to be protected against it. A trivial example of a secrecy-preserving reasoner is one which for every $M_i \in \mathcal{M}$ and every non-tautological query $\alpha \in \mathcal{Q}$, $\mathcal{R}(\alpha, M_i) = U$. At the other extreme, a reasoner $\mathcal{R}$ who answers truthfully all queries except for $\alpha \in S_i$ may fail to satisfy the Closure and/or Secrecy Axioms and hence is not a secrecy-preserving reasoner.

Definition 4.2.4 *Let* $\mathcal{K} = \langle K, \mathcal{Q}, \Omega\rangle$ *be a KB and* $\mathcal{S} = \langle \mathcal{M}, \mathbb{S}, \mathcal{G}\rangle$ *a secrecy structure on* $\mathcal{K}$*. A collection* $\mathbb{E} = \{E_1, E_2, ..., E_m\}$*, where for* $1 \leq i \leq m$*,* $S_i \subseteq E_i \subseteq K^+ \setminus \mathcal{T}_{aut}$*, is called a* (secrecy) envelope for $\mathcal{S}$ *if the following two axioms are satisfied for every* $1 \leq i \leq m$*:*

- ***[E1]*** *for every* $\alpha \in E_i$*,* $K^+ \setminus E_i \nvdash \alpha$*;*
- ***[E2]*** *for every* $\alpha \in S_i$*,* $\bigcup_{j:(M_j,M_i)\in\mathcal{A}}(K^+ \setminus E_j) \nvdash \alpha$*.*

The collection $\mathbb{E}$ *is called a* weak envelope for $\mathcal{S}$ *if it only satisfies Axiom E2. A secrecy envelope* $\mathbb{E}$ *is said to be* tight *if it satisfies an extra minimality axiom:*

- ***[TE]*** *for every* $M_i \in \mathcal{M}$ *and every* $\alpha \in E_i$*, there exist an edge* $(M_i, M_j) \in \mathcal{A}$ *and* $\beta \in S_j$ *such that* $\bigcup_{k:(M_k,M_j)\in\mathcal{A}}(K^+ \setminus E_k) \cup \{\alpha\} \vDash \beta$*.*

Note that every envelope is a weak envelope. Given an envelope $\mathbb{E} = \{E_1, E_2, ..., E_m\}$, we say that E_i is an envelope for the secrecy set S_i. Axiom E1 requires that no information in the envelope E_i is entailed from $K^+ \setminus E_i$. Axiom E2 ensures that no combination of query answers obtained from an agent's predecessors entails any secrets protected against this agent. Axiom TE requires that none of the queries in any of the envelopes in $\mathbb{E}$ can be removed without compromising the overall secrecy (not necessarily of its own secrecy set). Specifically, answering $\alpha \in E_i$ with Y (instead of U) would allow one of M_i's successors to conclude some of the secrets that need to be protected against it using the information passed to it from its predecessors. It is easy to see that (1) for each $M_i \in \mathcal{M}$, the complement of E_i is closed under entailment, i.e., $K^+ \setminus E_i \models \alpha$ implies $\alpha \in K^+ \setminus E_i$; and (2) for each $(M_j, M_i) \in \mathcal{A}$, $S_i \subseteq E_j$.

Lemma 4.2.5 *If a weak envelope $\mathbb{E} = \{E_1, E_2, ..., E_m\}$ for $\mathcal{S}$ is tight, then $\mathbb{E}$ is a tight envelope for $\mathcal{S}$.*

Proof We need to show that $\mathbb{E}$ satisfies Axiom E1. Suppose, by contradiction, that there exists $\alpha \in E_i$ s.t. $K^+ \setminus E_i \models \alpha$. Since $\mathbb{E}$ satisfies Axiom TE, there exist an edge $(M_i, M_j) \in \mathcal{A}$ and $\beta \in S_j$ s.t. $\bigcup_{k:(M_k,M_j)\in\mathcal{A}}(K^+ \setminus E_k) \cup \{\alpha\} \models \beta$. However, since $K^+ \setminus E_i \models \alpha$, we have $\bigcup_{k:(M_k,M_j)\in\mathcal{A}}(K^+ \setminus E_k) \models \alpha$, and so $\bigcup_{k:(M_k,M_j)\in\mathcal{A}}(K^+ \setminus E_k) \models \beta$. This contradicts the fact that $\mathbb{E}$ satisfies Axiom E2. Therefore, $\mathbb{E}$ satisfies Axiom E1. Since $\mathbb{E}$ satisfies Axioms E1, E2 and TE, it is a tight envelope for $\mathcal{S}$. ■

Secrecy envelopes (as well as tight envelopes) are not unique. For example, $\mathbb{E} = \{K^+ \setminus \mathcal{T}_{aut}, ..., K^+ \setminus \mathcal{T}_{aut}\}$ is always a secrecy envelope.

Let $\mathcal{K} = \langle K, \mathcal{Q}, \Omega \rangle$ be a KB. For a secrecy structure $\mathcal{S} = \langle \mathcal{M}, \mathbb{S}, \mathcal{G} \rangle$ on $\mathcal{K}$ define a set of induced single-agent secrecy structures, one for each $M_i \in \mathcal{M}$: $\mathcal{S}_i = \langle \{M_i\}, \{S_i\}, \langle \{M_i\}, \{(M_i, M_i)\}\rangle\rangle$, $1 \leq i \leq m$. Let E'_i be a (weak) envelope for $\mathcal{S}_i$ (as per Definition 4.2.4) and for each $1 \leq i \leq m$, define $E^*_i = \bigcup_{j:(M_i,M_j)\in\mathcal{A}} E'_j$. Even though $\mathbb{E}' = \{E'_1, ..., E'_m\}$ need not be a (weak) envelope for $\mathcal{S}$, we have the following result.

Theorem 4.2.6 $\mathbb{E}^* = \{E^*_1, ..., E^*_m\}$ *is a (weak) envelope for $\mathcal{S}$.*

Proof To show that $\mathbb{E}^*$ is an envelope for $\mathcal{S}$, we need to verify that $\mathbb{E}^*$ satisfies Axioms E1 and E2.

- **[E1]**: Suppose that for $\alpha \in E_i^*$, $K^+ \setminus E_i^* \models \alpha$. Then $\alpha \in E_j'$ for some j with $(M_i, M_j) \in \mathcal{A}$. Since $K^+ \setminus E_i^* \subseteq K^+ \setminus E_j'$, every model of $K^+ \setminus E_j'$ is a model of $K^+ \setminus E_i^*$, and so $K^+ \setminus E_j' \models \alpha$. This contradicts the definition of E_j'.

- **[E2]**: Suppose that for some i and $\alpha \in S_i$, we have $\bigcup_{j:(M_j,M_i)\in\mathcal{A}}(K^+ \setminus E_j^*) \models \alpha$. This is equivalent to $K^+ \setminus (\bigcap_{j:(M_j,M_i)\in\mathcal{A}} E_j^*) \models \alpha$. By the definition of $\mathbb{E}^*$, we have $E_i' \subseteq \bigcap_{j:(M_j,M_i)\in\mathcal{A}} E_j^*$. It follows that $K^+ \setminus (\bigcap_{j:(M_j,M_i)\in\mathcal{A}} E_j^*) \subseteq K^+ \setminus E_i'$, and so every model of $K^+ \setminus E_i'$ is a model of $K^+ \setminus (\bigcap_{j:(M_j,M_i)\in\mathcal{A}} E_j^*)$. This implies that $K^+ \setminus E_i' \models \alpha$, contradicting the definition of E_i'. ■

Note that by performing only local changes, Theorem 4.2.6 can be used to integrate existing MSQs into one. The next theorem shows that given a secrecy-preserving $\mathcal{K}$-reasoner, there is a natural way to define a corresponding envelope.

Theorem 4.2.7 *Given an MSQ system $\langle \mathcal{K}, \mathcal{S}, \mathcal{R} \rangle$, define a set $\mathbb{E}' = \{E_1', E_2', ..., E_m'\}$ where $E_i' = K^+ \setminus \mathcal{Q}_Y^i$ $(1 \leq i \leq m)$. Then $\mathbb{E}'$ is a secrecy envelope for $\mathcal{S}$.*

Proof We need to show that $\mathbb{E}'$ satisfies Axioms E1 and E2.

- **[E1]**: Suppose that there exist $M_i \in \mathcal{M}$ and $\alpha \in E_i'$ such that $K^+ \setminus E_i' \models \alpha$. Since the proof system $\vdash$ is complete w.r.t. $\models$, we have $\mathcal{Q}_Y^i = (K^+ \setminus E_i') \vdash \alpha$. It follows from the Closure Axiom that $\alpha \in \mathcal{Q}_Y^i$, i.e., $\alpha \in K^+ \setminus E_i'$. This contradicts $\alpha \in E_i'$.

- **[E2]**: Suppose that there is $\alpha \in S_i$ s.t. $\bigcup_{j:(M_j,M_i)\in\mathcal{A}} (K^+ \setminus E_j') \models \alpha$. It follows from the definition of E_i' that $\bigcup_{j:(M_j,M_i)\in\mathcal{A}} \mathcal{Q}_Y^j \models \alpha$. Since the proof system $\vdash$ is complete w.r.t. $\models$, we have $\bigcup_{j:(M_j,M_i)\in\mathcal{A}} \mathcal{Q}_Y^j \vdash \alpha$, i.e., $\mathcal{P}_Y^i \vdash \alpha$. This contradicts the Secrecy Axiom. ■

The following theorem gives the opposite direction: given a KB and an envelope $\mathbb{E}$, a corresponding secrecy-preserving $\mathcal{K}$-reasoner can be defined and the answer of a query α can be obtained by checking whether α can be deduced from the KB and its membership status w.r.t. $\mathbb{E}$.

Theorem 4.2.8 *Let $\mathcal{K} = \langle K, \mathcal{Q}, \Omega \rangle$ be a KB, $\mathcal{S} = \langle \mathcal{M}, \mathbb{S}, \mathcal{G} \rangle$ a secrecy structure on $\mathcal{K}$ and $\mathbb{E} = \{E_1, E_2, ..., E_m\}$ a secrecy envelope for $\mathcal{S}$. Define a function $\mathcal{R}_{\mathbb{E}}$: $\mathcal{Q} \times \mathcal{M} \to \Omega$ by*

$$\mathcal{R}_{\mathbb{E}}(\alpha, M_i) = \begin{cases} Y & \text{if } \alpha \in K^+ \setminus E_i, \\ N & \text{if } \neg\alpha \in K^+ \setminus E_i, \\ U & \text{otherwise.} \end{cases}$$

Then $\mathcal{R}_{\mathbb{E}}$ is a secrecy-preserving $\mathcal{K}$-reasoner.

Proof We need to show that $\mathcal{R}_{\mathbb{E}}$ satisfies the four axioms.

- **Yes-Axiom**: By definition of $\mathcal{R}_{\mathbb{E}}$, $\mathcal{Q}^i_Y = \{\alpha \mid \alpha \in K^+ \setminus E_i\} = K^+ \setminus E_i \subseteq K^+$.
- **No-Axiom**: By definition of $\mathcal{R}_{\mathbb{E}}$, $\mathcal{Q}^i_N = \{\alpha \mid \neg\alpha \in K^+ \setminus E_i\} = \{\neg\alpha \mid \alpha \in K^+ \setminus E_i\} = \neg\mathcal{Q}^i_Y$.
- **Closure Axiom**: It suffices to show that $(\mathcal{Q}^i_Y)^+ \subseteq \mathcal{Q}^i_Y$. By definition of $\mathcal{R}_{\mathbb{E}}$, $\mathcal{Q}^i_Y = K^+ \setminus E_i$. Suppose that $\mathcal{Q}^i_Y \vdash \alpha$ and $\alpha \notin \mathcal{Q}^i_Y = K^+ \setminus E_i$. By the soundness of $\vdash$, $\mathcal{Q}^i_Y \vDash \alpha$ and $\alpha \in E_i$. This contradicts Axiom E1 in Definition 4.2.4.
- **Secrecy Axiom**: Suppose that $(\mathcal{P}^i_Y)^+ \cap S_i \neq \emptyset$. Let $\alpha \in S_i$ s.t. $\mathcal{P}^i_Y \vdash \alpha$. Then we have $\bigcup_{j:(M_j,M_i)\in\mathcal{A}}(K^+ \setminus E_j) \vdash \alpha$. Because of the soundness of $\vdash$, we obtain $\bigcup_{j:(M_j,M_i)\in\mathcal{A}}(K^+ \setminus E_j) \vDash \alpha$. This contradicts our assumption that $\mathbb{E}$ is an envelope. ■

4.3 A Simple MSQ Algorithm - Lazy Evaluation

Given a KB $\mathcal{K}$ and a secrecy structure $\mathcal{S}$, a natural way of defining a secrecy preserving reasoner is to assume that queries are posed in an arbitrary but fixed order (history) $H = \{(\alpha_k, M_{i_k})\}_{k=1}^{\infty}$ where for all k, $1 \leq i_k \leq m$. The history H is said to be *full* if for all $(\alpha, M_i) \in \mathcal{Q} \times \mathcal{M}$, (α, M_i) belongs to H, i.e., all the queries are asked by every agent $M_i \in \mathcal{M}$. When agent M_{i_k} poses a query, the reasoner takes into account its answers to previous queries and responds (to M_{i_k}) according to whether or not a true answer may reveal secret information to M_{i_k} or to one of its successors (via communication as per the given communication graph).

This greedy approach is implemented in Algorithm 8. For the sake of simplicity of presentation, the algorithm omits the actual responses to the querying agents and, instead, concentrates on the construction of the sets $\mathcal{Q}_Y^i$, $\mathcal{Q}_N^i$, and $\mathcal{Q}_U^i$. Note that when the history H is not full, Algorithm 8 does not define a total function $\mathcal{Q} \times \mathcal{M} \to \Omega$ and as such it is not a "true" $\mathcal{K}$-reasoner according to our definition. The next lemma shows that for a full history, Algorithm 8 defines a secrecy-preserving $\mathcal{K}$-reasoner.

Algorithm 8 Lazy Evaluation Algorithm

Input: $\mathcal{K} = \langle K, \mathcal{Q}, \Omega \rangle$ and $\mathcal{S} = \langle \mathcal{M}, \mathbb{S}, \mathcal{G} \rangle$

Initialization:

For $1 \leq i \leq m$, let $\mathcal{Q}_Y^i = \mathcal{P}_Y^i = \mathcal{Q}_N^i = \mathcal{P}_N^i := \emptyset$, $\mathcal{Q}_U^i := \bigcup_{j:(M_i,M_j)\in\mathcal{A}} S_j$.

while *true* **do**
  input $\alpha \in \mathcal{Q}$, $M_i \in \mathcal{M}$
  if $\alpha \notin \mathcal{Q}_Y^i \cup \mathcal{Q}_N^i \cup \mathcal{Q}_U^i$ **then**
    if $K^+ \cap \{\alpha, \neg\alpha\} = \emptyset$ **then**
      $\mathcal{Q}_U^i := \mathcal{Q}_U^i \cup \{\alpha, \neg\alpha\}$
    else
      let $\bar{\alpha} \in \{\alpha, \neg\alpha\}$ such that $K \vdash \bar{\alpha}$
      if there exists j where $(M_i, M_j) \in \mathcal{A}$ such that $(\mathcal{P}_Y^j \cup \{\bar{\alpha}\})^+ \cap S_j \neq \emptyset$ **then**
        $\mathcal{Q}_U^i := \mathcal{Q}_U^i \cup \{\alpha, \neg\alpha\}$
      else
        $\mathcal{Q}_Y^i := \mathcal{Q}_Y^i \cup \{\bar{\alpha}\}$
        $\mathcal{Q}_N^i := \mathcal{Q}_N^i \cup \{\neg\bar{\alpha}\}$
        for all j where $(M_i, M_j) \in \mathcal{A}$, $\mathcal{P}_Y^j := \mathcal{P}_Y^j \cup \{\bar{\alpha}\}$ and $\mathcal{P}_N^j := \mathcal{P}_N^j \cup \{\neg\bar{\alpha}\}$
      end if
    end if
  end if
end while

Lemma 4.3.1 *Given a knowledge base $\mathcal{K} = \langle K, \mathcal{Q}, \Omega \rangle$, a secrecy structure $\mathcal{S} = \langle \mathcal{M}, \mathbb{S}, \mathcal{G} \rangle$ and a full query history $H = \{(\alpha_{i_k}, M_{i_k})\}_{k=1}^{\infty}$, the lazy evaluation algorithm (Algorithm 8) is a secrecy-preserving $\mathcal{K}$-reasoner.*

Proof We need to show that Algorithm 8 satisfies the four axioms in Definition 4.2.3. It is easy to see that Yes-Axiom and No-Axiom are satisfied.

To show that the Secrecy Axiom is satisfied, we argue by induction on history of queries. In the pre-query stage, for every i, $\mathcal{P}_Y^i = \emptyset$ and so, since $\mathcal{T}_{aut} \cap S_i = \emptyset$, $(\mathcal{P}_Y^i)^+ \cap S_i = \emptyset$. Now

suppose that the condition $(\mathcal{P}_Y^j)^+ \cap S_j = \emptyset$ holds for all j, and consider the next query $\alpha \in \mathcal{Q}$ posed by agent M_i. If $K^+ \cap \{\alpha, \neg\alpha\} = \emptyset$, then $\mathcal{P}_Y^j$ does not change, and the above condition is maintained. The same holds true if for some edge $(M_i, M_j) \in \mathcal{A}$, $(\mathcal{P}_Y^j \cup \{\bar{\alpha}\})^+ \cap S_j \neq \emptyset$. So suppose that for all $(M_i, M_j) \in \mathcal{A}$, $(\mathcal{P}_Y^j \cup \{\bar{\alpha}\})^+ \cap S_j = \emptyset$. Then for $(M_i, M_j) \in \mathcal{A}$, $\mathcal{P}_Y^j := \mathcal{P}_Y^j \cup \{\bar{\alpha}\}$. So with the new value of $\mathcal{P}_Y^j$, the same condition holds. Note that whenever a query is assigned to be in $\mathcal{Q}_Y^i \cup \mathcal{Q}_N^i \cup \mathcal{Q}_U^i$ for some $M_i \in \mathcal{M}$, it will not be re-evaluated for M_i. Therefore, the property $(\mathcal{P}_Y^i)^+ \cap S_i = \emptyset$ $(1 \leq i \leq m)$ is an invariant of the algorithm. It follows that the Secrecy Axiom is satisfied.

To show that the Closure Axiom is also satisfied, we assume that there is a query $q \in (\mathcal{Q}_Y^i)^+$ and $q \notin \mathcal{Q}_Y^i$. From $q \in (\mathcal{Q}_Y^i)^+$, we conclude $K \vdash q$. This means that when q was first queried by M_i and evaluated (i.e., when q was not in $\mathcal{Q}_Y^i \cup \mathcal{Q}_N^i \cup \mathcal{Q}_U^i$), the scope of the execution was from Lines 7-14. Moreover, in view that $q \notin \mathcal{Q}_Y^i$, the condition in Line 8 was satisfied. Hence, there existed j where $(M_i, M_j) \in \mathcal{A}$ and $\beta \in S_j$ s.t. $(\mathcal{P}_Y^j \cup \{\alpha\})^+ \cap S_j \neq \emptyset$. Therefore, there existed $\beta \in S_j$ such that $\mathcal{P}_Y^j \cup \{\alpha\} \vdash \beta$. Since $q \in (\mathcal{Q}_Y^i)^+$, $\mathcal{Q}_Y^i \vdash q$ and so $\bigcup_{k:(M_k,M_j)\in\mathcal{A}} \mathcal{Q}_Y^k = \mathcal{P}_Y^j \vdash \beta$. This contradicts the invariant of the algorithm. Hence, $(\mathcal{Q}_Y^i)^+ \subseteq \mathcal{Q}_Y^i$. It then follows from the fact that $\mathcal{Q}_Y^i \subseteq (\mathcal{Q}_Y^i)^+$ that the Closure Axiom is satisfied. ■

By Theorem 4.2.7, $\mathbb{E}' = \{E_1', ..., E_m'\}$ with $E_i' = K^+ \setminus \mathcal{Q}_Y^i$ and $\mathcal{Q}_Y^i$ as computed by Algorithm 8 is an envelope. The next theorem shows that $\mathbb{E}'$ is actually a tight envelope.

Theorem 4.3.2 *Let $\langle \mathcal{K}, \mathcal{S}, \mathcal{R}_H \rangle$ be an MSQ system where $\mathcal{R}_H$ is a $\mathcal{K}$-reasoner resulting from Algorithm 8 applied to a full history H. Define $\mathbb{E} = \{E_1, ..., E_m\}$ where $E_i = K^+ \cap \mathcal{Q}_U^i$ for $1 \leq i \leq m$. Then $\mathbb{E}$ is a tight envelope for $\mathbb{S}$.*

Proof By Lemma 4.3.1, for full history, the lazy evaluation algorithm is a secrecy-preserving $\mathcal{K}$-reasoner. It then follows from Theorem 4.2.7 that $\mathbb{E} = \{E_1, ..., E_m\}$ is an envelope. Suppose that $\mathbb{E}$ is not tight. Then by Axiom TE, there exist $M_i \in \mathcal{M}$ and $\alpha \in E_i \subseteq \mathcal{Q}_U^i$ such that for every edge $(M_i, M_j) \in \mathcal{A}$ and every $\beta \in S_j$, we have $\bigcup_{k:(M_k,M_j)\in\mathcal{A}}(K^+ \setminus E_k) \cup \{\alpha\} \nvdash \beta$. By the soundness of $\vdash$, we have $\bigcup_{k:(M_k,M_j)\in\mathcal{A}}(K^+ \setminus E_k) \cup \{\alpha\} \nvdash \beta$, i.e., $\mathcal{P}_Y^j \cup \{\alpha\} \nvdash \beta$ and hence $(\mathcal{P}_Y^j \cup \{\alpha\})^+ \cap S_j = \emptyset$. Note that in this case, $\bar{\alpha} = \alpha$. So when α was posed as a query by agent

M_i (for the first time), since $K \vdash \alpha$ and the condition in Line 8 is not satisfied, α should have been added to $\mathcal{Q}_Y^i$. However, this contradicts the choice of $\alpha \in \mathcal{Q}_U^i$. ■

The lazy evaluation approach is rather simple, but as the number of queries increases, the sets $\mathcal{Q}_Y^i$ get larger and checking condition in Line 8 takes longer time. Thus, answering queries will tend to be more time consuming as the KB continues to operate. Therefore, as an alternative, we propose to precompute an envelope and then utilize Theorem 4.2.8 to answer queries. We formalize this idea in the next section.

4.4 Computing Envelopes

In this section, we provide a general approach to compute envelopes. Given a KB $\mathcal{K}$ and a secrecy structure $\mathcal{S}$, as indicated in Theorem 4.2.6 (and the paragraph before it), the basic task is to construct an envelope for a single secrecy set. Our basic idea is to find a set of proof-disrupting assertions (of secrets) and put these in an envelope. We utilize the normal inference rules (that are native to the underlying language) and look for such disrupting formulas by inverting the inference rules. Next we formalize these ideas.

For a given KB $\mathcal{K} = \langle K, \mathcal{Q}, \Omega \rangle$ and a formula $\alpha \in K^+$, we say that a finite set $\Gamma \subseteq_f K^+$ is *α-minimal* if $\Gamma \models \alpha$ and for every $\beta \in \Gamma$, $\Gamma \setminus \{\beta\} \not\models \alpha$. Let $\mathcal{F}_\alpha = \{\Gamma \mid \Gamma \text{ be } \alpha\text{-minimal}\}$. If α needs to be protected, then at least one element in each set in $\mathcal{F}_\alpha$ has to be protected so that α cannot be entailed. Denote by ϕ_Γ an arbitrary but fixed element of a given set Γ. The following theorem indicates a general way for obtaining an envelope for a secrecy structure.

Theorem 4.4.1 *Given a secrecy structure $\mathcal{S} = \langle \mathcal{M}, \mathbb{S}, \mathcal{G} \rangle$ where $\mathbb{S} = \{S_1, S_2, ..., S_m\}$, for each $1 \leq i \leq m$, define a sequence of sets where $E_i^0 = S_i$ and $E_i^{k+1} = \{\phi_\Gamma \mid$ there is $\alpha \in E_i^k$ and $\Gamma \in \mathcal{F}_\alpha\}$. Let $E_i = \bigcup_{k=0}^{\infty} E_i^k$ and $E_i^* = \bigcup_{j:(M_i,M_j)\in\mathcal{A}} E_j$. Then $\mathbb{E}^* = \{E_1^*, ..., E_m^*\}$ is an envelope for $\mathcal{S}$.*

Proof For the given secrecy structure $\mathcal{S}$, define a set of induced single-agent secrecy structures, one for each $M_i \in \mathcal{M}$: $\mathcal{S}_i = \langle \{M_i\}, \{S_i\}, \langle \{M_i\}, \{(M_i, M_i)\} \rangle \rangle$, $1 \leq i \leq m$. By Theorem 4.2.6, it suffices to show that for $1 \leq i \leq m$, E_i is an envelope for $\mathcal{S}_i$. Suppose that for some $\alpha \in E_i$,

$K^+ \setminus E_i \models \alpha$. Then there is a finite set $\Gamma \subseteq K^+ \setminus E_i$ s.t. $\Gamma \models \alpha$ and Γ is α-minimal. Hence, $\Gamma \in \mathcal{F}_\alpha$. According to the definition of E_i, there exists k such that $\alpha \in E_i^k$. It follows that $\phi_\Gamma \in \Gamma \cap E_i^{k+1}$ and so $\Gamma \cap E_i \neq \emptyset$. This contradicts the fact that $\Gamma \subseteq K^+ \setminus E_i$. Therefore, E_i is an envelope for $\mathcal{S}_i$ for every $1 \leq i \leq m$. ■

Since we have assumed that given a language $\mathcal{L}$, $\vdash$ is sound and complete w.r.t. $\models$, Theorem 4.4.1 in essence indicates a recursive procedure of computing an envelope (see an example in Section 4.5). Once an envelope is computed, queries can be answered according to Theorem 4.2.8 without compromising secrecy, which is the basic goal of solving the SPQA problem. As mentioned before, we would like to compute envelopes that are as small as possible so that queries can be answered as informatively as possible. However, since given a language, deciding a minimum envelope may be NP-hard (see Section 4.5), we aim at computing tight envelopes. In general, when an envelope is finite, we could obtain a tight envelope by checking every formula in the envelope to see whether removing it compromises any secrets. If it does, the formula should be kept. Otherwise, it can be removed. After all the formulas in the original envelope are checked, a tight envelope is obtained. When an envelope $\mathbb{E} = \{E_1, ..., E_m\}$ is infinite, we may not obtain a tight envelope by removing assertions from it one by one. Given two weak envelopes $\mathbb{E} = \{E_1, ..., E_m\}$ and $\mathbb{E}' = \{E'_1, ..., E'_m\}$ for $\mathcal{S}$, we say that $\mathbb{E}' = \{E'_1, ..., E'_m\}$ is a *weak sub-envelope of* $\mathbb{E}$, denoted by $\mathbb{E}' \subseteq \mathbb{E}$, if for each $1 \leq i \leq m$, $E'_i \subseteq E_i$. A weak sub-envelope $\mathbb{E}'$ of $\mathbb{E}$ is *proper* if there exists $1 \leq i \leq m$ such that $E'_i \subset E_i$. Let $\mathbb{E}'$ and $\mathbb{E}$ be envelopes for $\mathcal{S}$. If $\mathbb{E}'$ is a weak sub-envelope of $\mathbb{E}$, then we say that $\mathbb{E}'$ is a *sub-envelope* of $\mathbb{E}$. We next show that given an envelope $\mathbb{E}$, there always exists a tight sub-envelope $\mathbb{E}'$ of $\mathbb{E}$.

Lemma 4.4.2 *Given a KB $\mathcal{K}$ and a secrecy structure $\mathcal{S}$ on $\mathcal{K}$, for every weak envelope $\mathbb{E} = \{E_1, ..., E_m\}$ for $\mathcal{S}$, if $\mathbb{E}$ does not contain a proper weak sub-envelope, then $\mathbb{E}$ is a tight envelope for $\mathcal{S}$.*

Proof Since $\mathbb{E}$ does not contain a proper weak sub-envelope, for every $M_i \in \mathcal{M}$ and every $\alpha \in E_i$, there exist an edge $(M_i, M_j) \in \mathcal{A}$ and $\beta \in S_j$ such that $\bigcup_{k \neq i:(M_k,M_j)\in\mathcal{A}}(K^+ \setminus E_k) \cup (K^+ \setminus (E_i \setminus \{\alpha\})) \models \beta$. This amounts to $\bigcup_{k:(M_k,M_j)\in\mathcal{A}}(K^+ \setminus E_k) \cup \{\alpha\} \models \beta$. Therefore, $\mathbb{E}$ satisfies Axiom TE. It then follows from Lemma 4.2.5 that $\mathbb{E}$ is a tight envelope for $\mathcal{S}$. ■

Given a KB $\mathcal{K}$ and a secrecy structure $\mathcal{S}$ on $\mathcal{K}$, for every weak envelope $\mathbb{E}$ for $\mathcal{S}$, either it has a sub-envelope $\mathbb{E}'$, or it does not. In the latter case, $\mathbb{E}$ is a tight envelope for $\mathcal{S}$ by Lemma 4.4.2. Given a weak envelope $\mathbb{E}$ for $\mathcal{S}$, let $\mathbb{E} = \mathbb{E}^0 \supseteq \mathbb{E}^1 \supseteq \cdots \supseteq \mathbb{E}^n \supseteq \cdots$ be a descending chain of weak envelopes for $\mathcal{S}$ where $\mathbb{E}^k = \{E_1^k, ..., E_m^k\} (k \geq 0)$. The next lemma holds.

Lemma 4.4.3 $\bigcap_{k=0}^{\infty} \mathbb{E}^k = \{\bigcap_{k=0}^{\infty} E_1^k, ..., \bigcap_{k=0}^{\infty} E_m^k\}$ *is a weak envelope for* $\mathcal{S}$.

Proof Suppose that $\bigcap_{k=0}^{\infty} \mathbb{E}^k$ is not a weak envelope for $\mathcal{S}$. Then $\bigcup_{j:(M_j,M_i)\in\mathcal{A}}(K^+ \backslash \bigcap_{k=0}^{\infty} E_j^k) \models \alpha$ for some $\alpha \in S_i$. This means that there is a finite subset $\Phi \subseteq_f \bigcup_{j:(M_j,M_i)\in\mathcal{A}}(K^+ \setminus \bigcap_{k=0}^{\infty} E_j^k)$ such that $\Phi \models \alpha$. Since Φ is finite, for some (large enough) n, $\Phi \subseteq \bigcup_{j:(M_j,M_i)\in\mathcal{A}}(K^+ \setminus E_j^n)$, implying $\bigcup_{j:(M_j,M_i)\in\mathcal{A}}(K^+ \setminus E_j^n) \models \alpha$. This contradicts the assumption that $\mathbb{E}^n$ is a weak envelope for $\mathcal{S}$. ■

A weak envelope $\mathbb{E}$ is *minimal* if it does not contain a proper weak sub-envelope. Let $\mathbb{E}$ be a weak envelope for $\mathcal{S}$ and $\mathcal{C}_{\mathbb{E}}$ be the collection of all weak sub-envelopes of $\mathbb{E}$. Since the binary relation $\subseteq$ between weak envelopes of a given $\mathbb{E}$ is a partial order on $\mathcal{C}_{\mathbb{E}}$, by Lemma 4.4.3 and (the dual of) Zorn's Lemma, $\mathcal{C}_{\mathbb{E}}$ contains a minimal weak envelope $\mathbb{E}'$. It follows from Lemma 4.4.2 that $\mathbb{E}'$ is a tight envelope for $\mathcal{S}$. Since every envelope is a weak envelope, this implies that every envelope has a tight sub-envelope (for the same $\mathcal{S}$).

Depending on the native inference system $\vdash$ of a language, and/or properties of the communication graph, some strategy may be designed to guide the computation of an envelope so that when an envelope is constructed, it is tight. In what follows, we consider an example of a communication graph that is an inverted forest (with self-loops) in which a strategy leads to a tight envelope.

Definition 4.4.4 *Given a formula* α *and a set* D *of formulas, for each* $\Gamma \in \mathcal{F}_\alpha$, *let* $\Gamma_D = \Gamma$ *if* $\Gamma \cap D = \emptyset$ *and* $\Gamma_D = \Gamma \cap D$ *otherwise. Define* $\mathcal{F}_{\alpha,D} = \{\Gamma_D \mid \Gamma \in \mathcal{F}_\alpha\}$. *For each* $\Gamma_D \in \mathcal{F}_{\alpha,D}$, *let* ϕ_{Γ_D} *be an arbitrary but fixed element in* Γ_D. *Given a secrecy structure* $\mathcal{S} = \langle \mathcal{M} = \{M_1, ..., M_m\}, \{S_1, ..., S_m\}, \langle \mathcal{M}, \mathcal{A} \rangle \rangle$ *and a set* D, *define* $E_i[D] = \{\phi_{\Gamma_D} \mid \alpha \in S_i$ *and* $\Gamma_D \in \mathcal{F}_{\alpha,D}\}$.

For any set D, $E_i[D]$ is a weak envelope for the induced single-agent secrecy structure $\mathcal{S}_i$: If there is $\alpha \in S_i$ s.t. $K^+ \setminus E_i[D] \models \alpha$, then there is $\Gamma \subseteq_f K^+ \setminus E_i[D]$ s.t. Γ is α-minimal. However, by Definition 4.4.4, there is $\phi_{\Gamma_D} \in \Gamma$ s.t. $\phi_{\Gamma_D} \in E_i[D]$, implying $\Gamma \cap E_i[D] \neq \emptyset$. This contradicts the fact that $\Gamma \subseteq_f K^+ \setminus E_i[D]$. Thus, $E_i[D]$ is a weak envelope for $\mathcal{S}_i$.

Lemma 4.4.5 *Given a KB $\mathcal{K}$ and a secrecy structure $\mathcal{S} = \langle \mathcal{M} = \{M_1, M_2\}, \{S_1, S_2\}, \langle \mathcal{M}, \mathcal{A} \rangle \rangle$ on $\mathcal{K}$ where $\mathcal{A} = \{(M_1, M_1), (M_2, M_2), (M_1, M_2)\}$, define two induced single-agent secrecy structure: $\mathcal{S}_i = \langle \{M_i\}, \{S_i\}, \langle \{M_i\}, \{(M_i, M_i)\} \rangle \rangle$ $(1 \leq i \leq 2)$. Suppose that E_2 is a tight envelope for $\mathcal{S}_2$. Let $E_1[E_2]$ be obtained from Definition 4.4.4. Suppose that $E_1 \subseteq E_1[E_2]$ is a tight envelope for $\mathcal{S}_1$. Then $\mathbb{E}^* = \{E_1^*, E_2^*\}$ where $E_i^* = \bigcup_{j:(M_i,M_j)\in\mathcal{A}} E_j$ $(1 \leq i \leq 2)$ is a tight envelope for $\mathcal{S}$.*

Proof We first show that $\mathbb{E}^*$ satisfies Axiom TE.

- For M_2: Since $E_2^* = E_2$ and E_2 is a tight envelope for $\mathcal{S}_2$, for every $\alpha \in E_2^*$, there exists $\beta \in S_2$ such that $(K^+ \setminus E_2^*) \cup \{\alpha\} \models \beta$. Since $E_2^* = E_2 \subseteq E_1^*$, $\bigcup_{k:(M_k,M_2)\in\mathcal{A}}(K^+ \setminus E_k^*) \cup \{\alpha\} = K^+ \setminus (E_1^* \cap E_2^*) \cup \{\alpha\} = K^+ \setminus E_2^* \cup \{\alpha\} \models \beta$. This shows that Axiom TE holds for M_2.

- For M_1:

  - Consider $\alpha \in E_1^* \cap E_2 = E_2$. Since E_2 is a tight envelope for $\mathcal{S}_2$, there exists $\beta \in S_2$ such that $(K^+ \setminus E_2) \cup \{\alpha\} \models \beta$. Then $\bigcup_{k:(M_k,M_2)\in\mathcal{A}}(K^+ \setminus E_k^*) \cup \{\alpha\} = K^+ \setminus (E_1^* \cap E_2^*) \cup \{\alpha\} = K^+ \setminus E_2^* \cup \{\alpha\} = K^+ \setminus E_2 \cup \{\alpha\} \models \beta$.

  - Consider $\alpha \in E_1^* \setminus E_2 = E_1 \setminus E_2$. Since E_1 is a tight envelope for $\mathcal{S}_1$, there exists $\beta \in S_1$ such that $K^+ \setminus E_1 \cup \{\alpha\} \models \beta$ and $K^+ \setminus E_1 \not\models \beta$. Since $\alpha \in E_1 \subseteq E_1[E_2]$, by Definition 4.4.4, there exists a finite set $\Gamma \subseteq K^+ \setminus E_1 \cup \{\alpha\}$ such that $\alpha = \phi_{\Gamma_{E_2}}$, $\Gamma \models \beta$ and $\Gamma_{E_2} \in \mathcal{F}_{\beta,E_2}$. If $\Gamma \cap E_2 \neq \emptyset$, then $\alpha = \phi_{\Gamma_{E_2}} \in E_2$ by the construction of $E_1[E_2]$ using Definition 4.4.4. However, this contradicts the assumption that $\alpha \in E_1 \setminus E_2$. Therefore $\Gamma \cap E_2 = \emptyset$. Moreover, since $\Gamma \subseteq K^+ \setminus E_1 \cup \{\alpha\} \models \beta$, we have $(K^+ \setminus E_1) \setminus E_2 \cup \{\alpha\} \models \beta$. Also, since $\bigcup_{k:(M_k,M_1)\in\mathcal{A}}(K^+ \setminus E_k^*) = K^+ \setminus E_1^* = K^+ \setminus (E_1 \cup E_2) = (K^+ \setminus E_1) \setminus E_2$, we have $\bigcup_{k:(M_k,M_1)\in\mathcal{A}}(K^+ \setminus E_k^*) \cup \{\alpha\} \models \beta$.

These show that the TE condition holds for M_1.

It follows from Theorem 4.2.6 that $\mathbb{E}^*$ is a weak envelope. By Lemma 4.2.5, $\mathbb{E}^*$ is a tight envelope for $\mathcal{S}$. ■

Theorem 4.4.6 *Given a KB $\mathcal{K}$ and a secrecy structure $\mathcal{S} = \langle \mathcal{M}, \{S_1, ..., S_m\}, \mathcal{G} = \langle \mathcal{M}, \mathcal{A} \rangle \rangle$ on $\mathcal{K}$ where each node in $\mathcal{G}$ has only one successor besides itself, i.e., $\mathcal{G}$ is an inverted forest with self-loops. For $1 \leq i \leq m$, in a bottom-up fashion according to $\mathcal{G}$, define an envelope $E_i[E_j]$ for an induced single-agent secrecy structure: $\mathcal{S}_i = \langle \{M_i\}, \{S_i\}, \langle \{M_i\}, \{(M_i, M_i)\} \rangle \rangle$ as per Definition 4.4.4, $(M_i, M_j) \in \mathcal{A}$ and E_j is a tight envelope for $\mathcal{S}_j$. Suppose that $E_i \subseteq E_i[E_j]$ is a tight envelope for $\mathcal{S}_i$. Then $\mathbb{E}^* = \{E_1^*, ..., E_m^*\}$ where $E_i^* = \bigcup_{j:(M_i,M_j)\in\mathcal{A}} E_j$ is a tight envelope for $\mathcal{S}$.*

Proof By Theorem 4.2.6, $\mathbb{E}^*$ is a weak envelope. It suffices to show that $\mathbb{E}^*$ satisfies Axiom TE. Suppose, by contradiction, that there are $M_i \in \mathcal{M}$ and $\alpha \in E_i^*$ s.t. for every j and every $\beta \in S_j$ where $(M_i, M_j) \in \mathcal{A}$, $\bigcup_{k:(M_k,M_j)\in\mathcal{A}} (K^+ \setminus E_k^*) \cup \{\alpha\} \nvDash \beta$. Consider $(M_i, M_j) \in \mathcal{A}$ where $i \neq j$ (when $i = j$ the argument is easy). There are two cases:

- $E_i^* \setminus E_j \neq \emptyset$. By Lemma 4.4.5, for every $\alpha' \in E_i^* \setminus E_j$, there is $\gamma \in S_i$ s.t. $K^+ \setminus E_i^* \cup \{\alpha'\} \vDash \gamma$ (see proof of Lemma 4.4.5 case (ii) for M_1). Since $\bigcap_{k:(M_k,M_j)\in\mathcal{A}} E_k^* \subseteq E_i^*$ (in view that $(M_i, M_j) \in \mathcal{A}$), every model of $K^+ \setminus \bigcap_{k:(M_k,M_j)\in\mathcal{A}} E_k^*$ is a model of $K^+ \setminus E_i^*$, and so $(K^+ \setminus \bigcap_{k:(M_k,M_j)\in\mathcal{A}} E_k^*) \cup \{\alpha'\} \vDash \gamma$.

- $E_i^* \setminus E_j = \emptyset$, i.e., $E_i^* = E_j$. By Lemma 4.4.5, $\{E_i^*, E_j\}$ is a tight envelope for the induced two-agent secrecy structure: $\mathcal{S}_{ij} = \langle \{M_i, M_j\}, \{S_i, S_j\}, \langle \{M_i, M_j\}, \{(M_i, M_i), (M_j, M_j), (M_i, M_j)\} \rangle$. Hence, for every $\alpha' \in E_i^*$, there is $\gamma \in S_i \cup S_j$ s.t. $(K^+ \setminus E_i^*) \cup \{\alpha'\} \vDash \gamma$ or $(K^+ \setminus E_i^*) \cup (K^+ \setminus E_j) \cup \{\alpha'\} \vDash \gamma$, i.e. $(K^+ \setminus E_i^*) \cup \{\alpha'\} \vDash \gamma$. Since $\bigcap_{k:(M_k,M_j)\in\mathcal{A}} E_k^* \subseteq E_i^*$, we have $\bigcup_{k:(M_k,M_j)\in\mathcal{A}} (K^+ \setminus E_k^*) \cup \{\alpha'\} \vDash \gamma$.

Both cases contradict the assumption that Axiom TE is not satisfied. ■

Note that both Lemma 4.4.5 and Theorem 4.4.6 consider communication graphs where each node has only one successor besides itself. In both cases, for any node M_i, the computation of

its envelope E_i is governed by the envelope E_j where $(M_i, M_j) \in \mathcal{A}$ so that as much information in E_j is reused for E_i as possible and as less redundant information is put into E_i as possible. In fact, Lemma 4.4.5 not only helps prove Theorem 4.4.6, but also is a special case of Theorem 4.4.6.

4.5 Horn MSQ System

In this section we illustrate multi-agent secrecy-preserving query answering in the context of Horn knowledge bases. This is of some interest as Horn theories are widely used in AI and Data Bases [97] and more recently KBs [98]. The main reason for this interest is that the satisfiability and inference problems for Horn theories can be solved very efficiently [99]. In fact, precisely for that same reason, non-Horn KBs are often "approximated" by Horn KBs [100, 98].

Recall that a (propositional) *Horn clause* is a clause containing at most one positive literal, i.e., generally, it is of the form: $x_1 \wedge \cdots \wedge x_k \rightarrow \eta$ where x_1, x_2, ..., x_k are propositional names (a.k.a. positive literals) and η is either a propositional name, in which case the Horn clause is called a *rule* (or a *definite clause*), or it is $\perp$, in which case it is called a *constraint*. In this paper we shall have no further use of constraints. A Horn clause is called a *fact* if $k = 0$ and $\eta \neq \perp$, i.e. it consists of a single positive literal. We shall assume a single underlying inference rule, the *forward chaining*, which is well-known to be sound and complete for Horn logic with respect to the usual semantics of propositional logic,

$$\textsc{Forward Chaining}\ (FC):\qquad \frac{\{l_1 \wedge l_2 \wedge \cdots \wedge l_k \rightarrow p\},\quad l_1,\quad l_2,\quad ...,\quad l_k}{p}$$

where $p, l_1, l_2, ..., l_k$ are all propositional names.

A *Horn KB* is a triple $\mathcal{K} = \langle K, \mathcal{Q}, \Omega \rangle$ where K is a finite set of Horn clauses (no constraints), $\mathcal{Q}$ is the set of all (relevant) facts (the query space), and $\Omega = \{Y, N, U\}$ is the answer space. The set of clauses K can be further partitioned $K = F \cup R$ where F is the set of facts in K and R is the set of rules in K. By F^+ we will denote the set of all facts derivable by applying

the FC-rule with assumptions in F and the rules in R: $F^+ = \{p \mid K \vdash_{FC} p \text{ and } p \text{ is a fact}\}$. Obviously, if K is finite, so is F^+.

Given a collection of querying agents $\mathcal{M} = \{M_1, M_2, ..., M_m\}$, a corresponding collection of secrecy sets $\mathbb{S} = \{S_1, S_2, ..., S_m\}$, and a communication graph $\mathcal{G}$ we have a secrecy structure $\mathcal{S} = \langle \mathcal{M}, \mathbb{S}, \mathcal{G} \rangle$. Note that the secrecy sets are subsets of F^+. To compute an envelope for $\mathcal{S} = \langle \mathcal{M}, \mathbb{S}, \mathcal{G} \rangle$, we can use the approach suggested in Theorem 4.2.6: for $1 \leq i \leq m$, compute an envelope E_i^1 for the induced single-agent secrecy structure $\mathcal{S}_i = \langle \{M_i\}, \{S_i\}, \langle \{M_i\}, \{(M_i, M_i)\} \rangle \rangle$. Then, letting $E_i^* = \bigcup_{j:(M_i,M_j)\in\mathcal{A}} E_j^1$, an envelope $\mathbb{E}^* = \{E_1^*, ..., E_m^*\}$ is obtained for the secrecy structure $\mathcal{S}$.

It remains to show how to compute an envelope for the single-agent secrecy structure $\mathcal{S}_i$. We will use a methodology developed by the authors in [24]. The idea is to invert the inference rules, in this case just the FC-rule, into new rules that enforce the intuitively obvious requirement: *whenever the conclusion of an inference rule is to be secret so must be at least one of its premises.* The inverted version of FC is denoted by FC^I and it is formulated as follows for each $1 \leq i \leq m$:

FC^I-RULE:

$$\frac{p \in E_i', \qquad l_1 \wedge l_2 \wedge ... \wedge l_k \rightarrow p \in R, \qquad l_1, l_2, ..., l_k \in F^+ \setminus E_i'}{E_i' := E_i' \cup \{l\}, \text{ for some } l \in \{l_1, l_2, ..., l_k\}}$$

The actual computation of the envelopes proceeds by initializing $\mathbb{E}' = \{E_1', E_2', ..., E_m'\}$ with $E_i' = S_i$ $(1 \leq i \leq m)$. The FC^I-rule is then applied repeatedly until it is no longer applicable. Denote by $\mathbb{E}^1 = \{E_1^1, E_2^1, ..., E_m^1\}$ the resulting collection of the sets E_1^1, E_2^1, ..., E_m^1. To prove the correctness of our procedure we must show that for each $1 \leq i \leq m$, E_i^1 is an envelope for $\mathcal{S}_i$.

Claim 4.5.1 *E_i^1 is a secrecy envelope for $\mathcal{S}_i$.*

Proof It suffices to show that E_i^1 satisfies Axiom E1: for every $\alpha \in E_i^1$, $F^+ \setminus E_i^1 \not\models \alpha$. Since $\vdash$ is complete w.r.t. $\models$, we shall argue instead that for every $\alpha \in E_i^1$, $F^+ \setminus E_i^1 \nvdash \alpha$. First note that for a fixed i, once a rule $l_1 \wedge l_2 \wedge ... \wedge l_k \rightarrow p \in R$ is used in an application of FC^I-rule for computing E_i^1, it is no longer applicable (for that fixed i). Thus, after at most $|R|$ applications

of the FC^I-rule (for that i) the computation of the set E_i^1 is complete. This means that for any rule $l_1 \wedge l_2 \wedge ... \wedge l_k \rightarrow p \in R$ with $l_1, ..., l_k \in F^+ \setminus E_i^1$, $p \notin E_i^1$ and hence $p \in F^+ \setminus E_i^1$. This implies the claim. ■

Since $\mathbb{E}^*$ is an envelope, according to Theorem 4.2.8, a query can be safely answered by checking whether it can provable from the given KB and its membership status w.r.t. $\mathbb{E}^*$. The envelope $\mathbb{E}^* = \{E_1^*, ..., E_m^*\}$ resulting from the single-agent "slices" in a manner indicated above, need not be tight.

Example 4.5.2 *Given a Horn KB* $\mathcal{K} = \langle K, \mathcal{S}, \mathcal{R} \rangle$ *where* $K = \langle F = \{l_1, l_2, s\}, R = \{l_1 \wedge l_2 \rightarrow s\} \rangle$, $\mathcal{S} = \langle \{M_1, M_2\}, \{S_1, S_2\}, \langle \{M_1, M_2\}, \{(M_1, M_1), (M_2, M_2), (M_1, M_2)\} \rangle \rangle$ *and* $S_1 = S_2 = \{s\}$. *Suppose that* $E_1^1 = \{s, l_1\}$ *and* $E_2^1 = \{s, l_2\}$. *It is easy to check that* E_i^1 *is a tight envelope for* $\mathcal{S}_i$, $1 \leq i \leq 2$. *Let* $E_1^* = E_1^1 \cup E_2^1 = \{s, l_1, l_2\}$ *and* $E_2^* = E_2^1$. *Then* $\mathbb{E}^* = \{E_1^*, E_2^*\}$ *is an envelope for* $\mathcal{S}$. *However,* $\mathbb{E}^*$ *is not tight because we could remove* l_1 *from* E_1^* *and still have an envelope for* $\mathcal{S}$.

In fact, there are two tight envelopes for $\mathcal{S}$: $\mathbb{E}_1^* = \{\{s, l_1\}, \{s, l_1\}\}$ *and* $\mathbb{E}_2^* = \{\{s, l_2\}, \{s, l_2\}\}$.

As mentioned before, we aim at computing envelopes that are as small as possible. Unfortunately, the decision problem associated with finding a smallest cardinality envelope is NP-complete. We specify the *Minimum Envelope* problem (*ME*) by the pair $\langle \langle \mathcal{K}, \mathcal{S} \rangle, N \rangle$ where $\mathcal{K} = \langle K, \mathcal{Q}, \Omega \rangle$ is a Horn KB, $\mathcal{S} = \langle \mathcal{M}, \mathbb{S}, \mathcal{G} \rangle$ is a secrecy structure for $\mathcal{K}$ and N is a positive integer. The decision problem is to determine whether $\mathcal{S}$ has a secrecy envelope $\mathbb{E} = \{E_1, ..., E_m\}$ satisfying $|\bigcup_{1 \leq i \leq m} E_i| \leq N$. It is easy to see that the problem is in NP as this only involves checking that $\mathbb{E}$ satisfies the axioms E1 and E2.

We use the Hitting Set (HS) to show NP-hardness. Given a finite set X, a finite collection of non-empty sets $\mathcal{C} = \{C_1, ..., C_k\} \subseteq \mathbb{P}(X)$ and an integer $0 \leq N \leq |X|$, the problem is to determine whether or not there is a subset $Y \subseteq X$ such that $|Y| \leq N$ and for every $C \in \mathcal{C}$, $C \cap Y \neq \emptyset$. Given such an instance of HS, we construct an instance of ME as follows:

- $\mathcal{M} = \{M_1\}$, a single querying agent;
- the communication graph consists of a single self-loop on M_1;

- $\mathbb{S} = \{S_1\}$, with $S_1 = \{s_i \mid C_i \in \mathcal{C}\}$, where s_i's are new symbols;
- $N' = N + |\mathcal{C}|$;
- $K = F \cup R$ where $F = X \cup S_1$ and $R = \{l_1 \wedge ... \wedge l_r \rightarrow s_i \mid C_i = \{l_1, ..., l_r\} \in \mathcal{C}\}$.

Claim 4.5.3 *$\mathcal{C}$ has a hitting set $Y \subseteq X$ with $|Y| \leq N$ if and only if $\langle \mathcal{K}, \mathcal{S} \rangle$ has a secrecy envelope $\mathbb{E} = \{E_1\}$ such that $S_1 \subseteq E_1 \subseteq F^+$ and $|E_1| \leq N + |\mathcal{C}|$.*

Proof Suppose that $Y \subseteq X$ is a hitting set for $\mathcal{C}$ with $|Y| \leq N$. Define the set $E_1 := Y \cup S_1$. Since $Y \cap S_1 = \emptyset$, $|E_1| = |Y| + |S_1| \leq N + |\mathcal{C}|$. Moreover, for every $C \in \mathcal{C}$, $C \cap Y \neq \emptyset$. Therefore, none of the rules in K can be used in applying the FC-rule to $F^+ \setminus E_1$. It follows that E_1 is a secrecy envelope for $\mathcal{S}$.

Conversely, let $\mathbb{E} = \{E_1\}$ be a secrecy envelope for $\mathcal{S}$ such that $S_1 \subseteq E_1 \subseteq F^+$ and $|E_1 \setminus S_1| \leq N$. By Axiom E1 and the soundness of the FC-rule, this implies that for every $\alpha \in E_1 : F^+ \setminus E_1 \nvdash_{FC} \alpha$. We show that the HS instance $\mathcal{C} = \{C_1, ..., C_k\} \subseteq \mathbb{P}(X)$, together with an integer $0 < N \leq |X|$, has a hitting set of size at most N. Define $Y := E_1 \setminus S_1$. It now suffices to show that for every $C_i \in \mathcal{C}$, $Y \cap C_i \neq \emptyset$. Let $C_i = \{l_1, ..., l_r\}$; by the definition of the reduction, this implies that $l_1 \wedge ... \wedge l_r \rightarrow s_i$ belongs to R. If none of the $l_j (1 \leq j \leq r)$ belongs to Y, then they all belong to $F^+ \setminus Y$ and hence also to $F^+ \setminus E_1$ because $C_i \cap S_1 = \emptyset$. Therefore, $F^+ \setminus E_1 \vdash_{FC} s_i \in E_1$, which leads to a contradiction. ■

4.6 Conclusion and Future Work

Many applications require a KB that contains secrets to answer queries using secrets, whenever it is possible to do so, without revealing secrets. We considered this problem under the OWA in a setting with multiple querying agents, where associated with each agent is a secrecy set that the KB is obliged to protect from it, and the agents can selectively share the answers that they receive with other agents. We introduced the notion of a secrecy envelope and prove some results that are helpful in computing and updating the envelope, and for using the envelope to protect secrets while answering queries. We also provided a strategy for constructing tight envelopes in the interesting special case where the communication graphs are inverted

forests. We illustrated an application of this general approach in the case of propositional Horn KBs.

We have assumed that the secrecy sets are finite. It would be interesting to consider how to relax cases where secrecy sets have finite descriptions that can be expressed in a suitable policy language. Other interesting directions for future work include consideration of more expressive communication graphs e.g., those that place additional restrictions on the answers that can be shared between agents e.g., by attaching predicates to edges, and design of efficient algorithms for constructing and maintaining envelopes and answering queries using envelopes for KBs based on tractable yet practically useful knowledge representation languages.

CHAPTER 5. SUMMARY AND DISCUSSION

This dissertation focuses on developing techniques to answer queries posed to KBs. The first topic we studied is query answering in KBs that contain epistemic information. This topic is presented in Chapter 2. The second topic (Chapters 3 and 4) we studied requires autonomous entities or organizations to be able to selectively share information without disclosing sensitive information. We investigate techniques for secrecy-preserving query answering against KBs under the OWA. We consider two scenarios of increasing difficulty: (a) a KB queried by a single agent; and (b) a KB queried by multiple agents where the secrecy policies can differ across the different agents and the agents can selectively communicate the answers that they receive from the KB with each other subject to the applicable answer sharing policies. Specific contributions are listed below:

- In Chapter 2, we studied $\mathcal{ALCK}_m$ and $\mathcal{ALCS}4_m$, knowledge representation languages obtained by augmenting $\mathcal{ALC}$ with modal operators of the basic multi-modal logics K_m and $S4_m$. The resulting logics allow us to represent and reason about the knowledge of multiple experts. We developed sound and complete tableau algorithms $\Lambda_{\mathcal{K}}$ and Λ_{S4} for answering queries w.r.t. corresponding knowledge bases with acyclic TBoxes.

  Instead of general concept inclusions allowed in $\mathbf{K}_{\mathcal{ALC}}$ [40] which lead to a NEXPTIME algorithm for satisfiability, the acyclicity restriction on the TBoxes is critical to achieving the PSPACE implementations for both algorithms. Our PSPACE results for the satisfiability of $\mathcal{ALCK}_m$ and $\mathcal{ALCS}4_m$ extend the result of Schmidt-Schauß and Smolka [2] for the satisfiability and subsumption of $\mathcal{ALC}$ concepts.

- In Chapter 3, we studied the problem of answering queries against a knowledge base that contains secret information. Based on the OWA, we designed reasoners that hide truthful

answers to the queries that if faithfully answered, may compromise the secrecy. One such reasoner was designed using the lazy evaluation. Since this approach becomes less and less appealing over time, we proposed to maintain a secrecy system that precomputes an envelope. Once an envelope is present, a query will be truthfully answered if it is outside the envelope. A general framework for the solution to the problem was provided. We discussed the relationship between a secrecy-preserving reasoner and an envelope. We applied the general framework to the Description Logic $\mathcal{EL}$ and provided an algorithm for constructing an envelope. To answer queries as informatively as possible, we aimed to make envelopes as small as possible. Since computing the smallest envelope in $\mathcal{EL}$ is also NP-complete, we have presented two algorithms for computing tight envelopes: a naive algorithm and an optimized version, procedure TIGHT. We compared the complexities of these two algorithms, designed experiments and concluded that the optimized algorithm indeed is more efficient for applications whenever the sizes of the TBox and the secrecy set are much smaller than that of the ABox, which is typical in many applications.

- In Chapter 4, we extended the general framework for secrecy-preserving query answering problem in the single-agent case to multiple querying agents where a knowledge base contains secret information for each agent and the agents are allowed to share query answers in a constrained fashioned. We designed a secrecy-preserving reasoner that answers queries without compromising the overall secrecy, analyzed the relationship between a secrecy-preserving reasoner associated with a KB and a secrecy envelope used to protect secrets, and discussed a special case of restricted communication for constructing tight envelopes. The idea of constructing envelopes was illustrated using Horn KBs.

There are several directions for extending this dissertation.

- For the first topic, Baader et al. [57] have recently extended the PSPACE result of [2] to $\mathcal{ALC}$ with transitive and inverse roles. In light of this result, we conjecture that query answering against $\mathcal{SIK}_m$, obtained by replacing $\mathcal{ALC}$ (in $\mathcal{ALCK}_m$) with $\mathcal{SI}$ ($\mathcal{ALC}$ augmented with transitive and inverse roles), can also be implemented in PSPACE. Another

direction would be investigating algorithms for the satisfiability of $\mathcal{ALCS}5_m$ whose syntax is identical to that of $\mathcal{ALCK}_m$, but whose semantics is based on the modal logic $S5_m$.

- For the second topic in single-agent case, strategies may be designed to specify secrets (policy specifications) and generate secrecy sets in an automated way. For example, a policy like "Whether or not a patient x is at risk of developing cancer must be kept secret" specifies a requirement about a whole range of assertions which must be protected (rather than a single assertion). This is rather different than most of the work on policy languages for the web which focuses on specifying syntax-based restrictions on access to specific resources or operations on the web (see Section 3.7).

- In the multiagent case, one direction could be applying our multiagent framework to specific DLs such as $\mathcal{ALC}$. Another direction would be to make the communication graph more flexible and realistic. For instance, we could add restrictions on the information sharing by putting predicates on the edges of the communication graph, or allow more general graph structures (not necessarily DAG with self-loops).

APPENDIX A. ADDITIONAL MATERIAL

A.1 Additional Material for Chapter 2

A.1.1 Proof of Theorem 2.3.3

Theorem 2.3.3 *(Soundness of the expansion rules) Given a Kripke structure* $\mathbb{M} = \langle S, \pi, \mathcal{E}_1, ..., \mathcal{E}_m \rangle$ *and an acyclic TBox* $\mathcal{T}$ *where* $\mathbb{M} \vDash \mathcal{T}$*, let* $\mathbb{G}$ *be a constraint graph,* α *a local, global or terminological expansion rule and* $\mathbb{G}_\alpha$ *a constraint graph obtained by applying* α *to* $\mathbb{G}$*. If* $\mathbb{M} \Vdash \mathbb{G}$ *via* σ*, then there exists a semantic extension* $\mathbb{M}_\alpha$ *of* $\mathbb{M}|_{N_\Sigma \cup \mathcal{O}_\mathbb{G}}$ *s.t.* $\mathbb{M}_\alpha \Vdash \mathbb{G}_\alpha$ *via* σ' *(which extends* σ*) and* $\mathbb{M}_\alpha \vDash \mathcal{T}$*. Furthermore,* $\mathbb{M}_\alpha \Vdash \mathbb{G}$*.*

Proof Assume the hypotheses.

1. If α is a $\sqcap$-rule, then there is a constraint $a : C_1 \sqcap C_2 \in \mathbb{L}(n)$ in $\mathbb{G}$ and $\{a : C_1, a : C_2\} \nsubseteq \mathbb{L}(n)$. After applying $\sqcap$-rule, $\mathbb{L}(n) = \mathbb{L}(n) \cup \{a : C_1, a : C_2\}$. By Definition 3, $a : C_1 \sqcap C_2 \in \mathbb{L}(n)$ implies $(\mathbb{M}, \sigma(n)) \vDash C_1 \sqcap C_2(a)$. It follows that $a^{\pi(\sigma(n))} \in (C_1 \sqcap C_2)^{\pi(\sigma(n))}$, which means that $a^{\pi(\sigma(n))} \in C_1^{\pi(\sigma(n))}$ and $a^{\pi(\sigma(n))} \in C_2^{\pi(\sigma(n))}$. Hence, $(\mathbb{M}, \sigma(n)) \vDash C_1(a)$ and $(\mathbb{M}, \sigma(n)) \vDash C_2(a)$. Thus, $\mathbb{G}_\alpha$ obtained by application of $\sqcap$-rule from $\mathbb{G}$ is satisfied by $\mathbb{M}$ via σ.

2. If α is a $\sqcup$-rule, then there is a constraint $a : C_1 \sqcup C_2 \in \mathbb{L}(n)$ in $\mathbb{G}$ and $\{a : C_1, a : C_2\} \cap \mathbb{L}(n) = \emptyset$. By Definition 3, $(\mathbb{M}, \sigma(n)) \vDash C_1 \sqcup C_2(a)$ and therefore $a^{\pi(\sigma(n))} \in (C_1 \sqcup C_2)^{\pi(\sigma(n))}$. This means that $a^{\pi(\sigma(n))} \in C_1^{\pi(\sigma(n))}$ or $a^{\pi(\sigma(n))} \in C_2^{\pi(\sigma(n))}$. Hence, $(\mathbb{M}, \sigma(n))$ satisfies $C_1(a)$ or $C_2(a)$ (or both). It follows that $\sqcup$-rule can be applied in a way such that $\mathbb{G}_\alpha$ is satisfied by $\mathbb{M}$ via σ.

3. If α is an $\exists$-rule, then there is a constraint $a : \exists R.C \in \mathbb{L}(n)$ in $\mathbb{G}$. Since $(\mathbb{M}, \sigma(n)) \vDash \exists R.C(a)$ (by Definition 3), there must be an element $d \in \Delta$ such that $(a^{\pi(\sigma(n))}, d) \in$

$R^{\pi(\sigma(n))}$ and $d \in C^{\pi(\sigma(n))}$. After applying the $\exists$-rule, a fresh individual name c is picked and $\mathbb{L}(n) := \mathbb{L}(n) \cup \{(a, c) : R, c : C\}$. Define the interpretation π' as π except for the fresh individual name c: $c^{\pi'(\sigma(n))} = d$. The resulting $\mathbb{G}_\alpha$ is satisfied by $\mathbb{M}_\alpha$ via σ where $\mathbb{M}_\alpha = \langle S, \pi', \mathcal{E}_1, ..., \mathcal{E}_m \rangle$ is a semantic extension of $\mathbb{M}|_{N_\Sigma \cup \mathcal{O}_\mathbb{G}}$.

4. If α is a $\forall$-rule, then there is a node n with $\{a : \forall R.C, (a, b) : R\} \subseteq \mathbb{L}(n)$ and $b : C \notin \mathbb{L}(n)$. By Definition 3, $a : \forall R.C \in \mathbb{L}(n)$ implies $(\mathbb{M}, \sigma(n)) \vDash \forall R.C(a)$, which means that for all $d \in \Delta$, $(a^{\pi(\sigma(n))}, d) \in R^{\pi(\sigma(n))}$ implies $d \in C^{\pi(\sigma(n))}$. Moreover, $(a, b) : R \in \mathbb{L}(n)$ implies $(\mathbb{M}, \sigma(n)) \vDash R(a, b)$, which means $(a^{\pi(\sigma(n))}, b^{\pi(\sigma(n))}) \in R^{\pi(\sigma(n))}$. After applying the $\forall$-rule, $b : C$ is added to $\mathbb{L}(n)$. The resulting $\mathbb{G}_\alpha$ is satisfied by $\mathbb{M}$ via σ.

5. If α is a $\Diamond\mathcal{C}$-rule, there is a constraint $a : \Diamond_i C \in \mathbb{L}(n)$ in $\mathbb{G}$ and n does not have an i-successor l such that $a : C \in \mathbb{L}(l)$. By Definition 3, $a : \Diamond_i C \in \mathbb{L}(n)$ implies $(\mathbb{M}, \sigma(n)) \vDash \Diamond_i C(a)$ which means that there is a world s with $(\sigma(n), s) \in \mathcal{E}_i$ and $a^{\pi(s)} \in C^{\pi(s)}$. After applying the $\Diamond\mathcal{C}$-rule, a new node n' is generated with $\mathbb{L}(n') = \{a : C\}$ and $\mathbb{L}(n, n') = \{i\}$. Extend σ to σ' such that $\sigma'(n') = s$. $\mathbb{M}$ satisfies the resulting $\mathbb{G}_\alpha$ via σ'.

6. If α is a $\Box\mathcal{C}$-rule, then there are two nodes n and n' in $\mathbb{G}$ such that $i \in \mathbb{L}(n, n')$, $a : \Box_i C \in \mathbb{L}(n)$ and $a : C \notin \mathbb{L}(n')$. By Definition 3, $a : \Box_i C \in \mathbb{L}(n)$ implies $(\mathbb{M}, \sigma(n)) \vDash \Box_i C(a)$ which means that for all s with $(\sigma(n), s) \in \mathcal{E}_i$, $(\mathbb{M}, s) \vDash C(a)$. It follows that $(\mathbb{M}, \sigma(n')) \vDash C(a)$. After applying the $\Box\mathcal{C}$-rule, $a : C \in \mathbb{L}(n')$. $\mathbb{G}_\alpha$ obtained from $\mathbb{G}$ is satisfied by $\mathbb{M}$ via σ.

7. If α is a T-rule, then there is a constraint $a : A \in \mathbb{L}(n)$, a definition $A \doteq D \in \mathcal{T}$ and $a : D \notin \mathbb{L}(n)$. After applying α, $\mathbb{L}(n) = \mathbb{L}(n) \cup \{a : D\}$. Since $\mathbb{M} \Vdash \mathbb{G}$ and $\mathbb{M} \vDash \mathcal{T}$, $a^{\pi(\sigma(n))} \in A^{\pi(\sigma(n))} = D^{\pi(\sigma(n))}$. Therefore, $(\mathbb{M}, \sigma(n)) \vDash D(a)$ and hence, $\mathbb{M} \Vdash \mathbb{G}_\alpha$ via σ.

8. If α is an N-rule, then $\{a : \neg A, a : B\} \cap \mathbb{L}(n) \neq \emptyset$, $A \doteq \neg B \in \mathcal{T}$ and $\{a : \neg A, a : B\} \nsubseteq \mathbb{L}(n)$. Since $\mathbb{M} \Vdash \mathbb{G}$ and $\mathbb{M} \vDash \mathcal{T}$, we have $(\mathbb{M}, \sigma(n)) \vDash A \doteq \neg B$ and therefore $a^{\pi(\sigma(n))} \notin A^{\pi(\sigma(n))} \Leftrightarrow a^{\pi(\sigma(n))} \in B^{\pi(\sigma(n))}$. Because only one of $a : \neg A$ and $a : B$ is in $\mathbb{L}(n)$, after applying the N-rule, the other constraint is added to $\mathbb{L}(n)$ and it is satisfied by $(\mathbb{M}, \sigma(n))$. Therefore, $\mathbb{M} \Vdash \mathbb{G}_\alpha$ via σ.

9. If α is an N$\sqcap$-rule, then $a : \neg A \in \mathbb{L}(n)$, $A \doteq B_1 \sqcap B_2 \in \mathcal{T}$, and $a : \neg B_1 \sqcup \neg B_2 \notin \mathbb{L}(n)$. Since $\mathbb{M} \Vdash \mathbb{G}$ and $\mathbb{M} \vDash \mathcal{T}$, we have $(\mathbb{M}, \sigma(n)) \vDash \neg A(a)$, $(\mathbb{M}, \sigma(n)) \vDash A \doteq B_1 \sqcap B_2$ and therefore $a^{\pi(\sigma(n))} \notin A^{\pi(\sigma(n))} \Leftrightarrow a^{\pi(\sigma(n))} \notin (B_1 \sqcap B_2)^{\pi(\sigma(n))} \Leftrightarrow a^{\pi(\sigma(n))} \in \Delta \setminus (B_1^{\pi(\sigma(n))} \cap B_2^{\pi(\sigma(n))}) \Leftrightarrow a^{\pi(\sigma(n))} \in (\Delta \setminus B_1^{\pi(\sigma(n))}) \cup (\Delta \setminus B_2^{\pi(\sigma(n))}) \Leftrightarrow a^{\pi(\sigma(n))} \in (\neg B_1)^{\pi(\sigma(n))} \cup (\neg B_2)^{\pi(\sigma(n))}$. This means that $a^{\pi(\sigma(n))} \in (\neg B_1 \sqcup \neg B_2)^{\pi(\sigma(n))}$. After applying α, $a : \neg B_1 \sqcup \neg B_2 \in \mathbb{L}(n)$ and it is satisfied by $(\mathbb{M}, \sigma(n))$. Therefore, $\mathbb{M} \Vdash \mathbb{G}_\alpha$ via σ.

10. If α is an N$\sqcup$-rule, then $a : \neg A \in \mathbb{L}(n)$, $A \doteq B_1 \sqcup B_2 \in \mathcal{T}$, and $a : \neg B_1 \sqcap \neg B_2 \notin \mathbb{L}(n)$. Since $\mathbb{M} \Vdash \mathbb{G}$ and $\mathbb{M} \vDash \mathcal{T}$, we have $(\mathbb{M}, \sigma(n)) \vDash \neg A(a)$, $(\mathbb{M}, \sigma(n)) \vDash A \doteq B_1 \sqcup B_2$ and therefore $a^{\pi(\sigma(n))} \notin A^{\pi(\sigma(n))} \Leftrightarrow a^{\pi(\sigma(n))} \notin (B_1 \sqcup B_2)^{\pi(\sigma(n))} \Leftrightarrow a^{\pi(\sigma(n))} \in \Delta \setminus (B_1^{\pi(\sigma(n))} \cup B_2^{\pi(\sigma(n))}) \Leftrightarrow a^{\pi(\sigma(n))} \in (\Delta \setminus B_1^{\pi(\sigma(n))}) \cap (\Delta \setminus B_2^{\pi(\sigma(n))}) \Leftrightarrow a^{\pi(\sigma(n))} \in (\neg B_1)^{\pi(\sigma(n))} \cap (\neg B_2)^{\pi(\sigma(n))}$. This means that $a^{\pi(\sigma(n))} \in (\neg B_1 \sqcap \neg B_2)^{\pi(\sigma(n))}$. After applying α, $a : \neg B_1 \sqcap \neg B_2 \in \mathbb{L}(n)$ and it is satisfied by $(\mathbb{M}, \sigma(n))$. Therefore, $\mathbb{M} \Vdash \mathbb{G}_\alpha$ via σ.

11. If α is an N$\exists$-rule, then $a : \neg A \in \mathbb{L}(n)$, $A \doteq \exists R.B \in \mathcal{T}$, and $a : \forall R.\neg B \notin \mathbb{L}(n)$. Since $\mathbb{M} \Vdash \mathbb{G}$ and $\mathbb{M} \vDash \mathcal{T}$, we have $(\mathbb{M}, \sigma(n)) \vDash \neg A(a)$, $(\mathbb{M}, \sigma(n)) \vDash A \doteq \exists R.B$ and therefore $a^{\pi(\sigma(n))} \notin A^{\pi(\sigma(n))} \Leftrightarrow a^{\pi(\sigma(n))} \notin (\exists R.B)^{\pi(\sigma(n))} \Leftrightarrow a^{\pi(\sigma(n))} \notin \{c \in \Delta \mid \exists b : (c, b) \in R^{\pi(s)} \wedge b \in B^{\pi(s)}\} \Leftrightarrow a^{\pi(\sigma(n))} \in \{c \in \Delta \mid \forall b : (c, b) \in R^{\pi(s)} \rightarrow b \notin B^{\pi(s)}\} \Leftrightarrow a^{\pi(\sigma(n))} \in (\forall R.\neg B)^{\pi(\sigma(n))}$. After applying α, $a : \forall R.\neg B \in \mathbb{L}(n)$ and it is satisfied by $(\mathbb{M}, \sigma(n))$. Therefore, $\mathbb{M} \Vdash \mathbb{G}_\alpha$ via σ.

12. If α is an N$\forall$-rule, then $a : \neg A \in \mathbb{L}(n)$, $A \doteq \forall R.B \in \mathcal{T}$, and $a : \exists R.\neg B \notin \mathbb{L}(n)$. Since $\mathbb{M} \Vdash \mathbb{G}$ and $\mathbb{M} \vDash \mathcal{T}$, we have $(\mathbb{M}, \sigma(n)) \vDash \neg A(a)$, $(\mathbb{M}, \sigma(n)) \vDash A \doteq \forall R.B$ and therefore $a^{\pi(\sigma(n))} \notin A^{\pi(\sigma(n))} \Leftrightarrow a^{\pi(\sigma(n))} \notin (\forall R.B)^{\pi(\sigma(n))} \Leftrightarrow a^{\pi(\sigma(n))} \notin \{c \in \Delta \mid \forall b : (c, b) \in R^{\pi(s)} \rightarrow b \in B^{\pi(s)}\} \Leftrightarrow a^{\pi(\sigma(n))} \in \{c \in \Delta \mid \exists b : (c, b) \in R^{\pi(s)} \wedge b \notin B^{\pi(s)}\} \Leftrightarrow a^{\pi(\sigma(n))} \in (\exists R.\neg B)^{\pi(\sigma(n))}$. After applying α, $a : \exists R.\neg B \in \mathbb{L}(n)$ and it is satisfied by $(\mathbb{M}, \sigma(n))$. Therefore, $\mathbb{M} \Vdash \mathbb{G}_\alpha$ via σ.

13. If α is an N$\Diamond$-rule, then $a : \neg A \in \mathbb{L}(n)$, $A \doteq \Diamond_i B \in \mathcal{T}$, and $a : \Box_i \neg B \notin \mathbb{L}(n)$. Since $\mathbb{M} \Vdash \mathbb{G}$ and $\mathbb{M} \vDash \mathcal{T}$, we have $(\mathbb{M}, \sigma(n)) \vDash A \doteq \Diamond_i B$, $(\mathbb{M}, \sigma(n)) \vDash \neg A(a)$ and therefore $a^{\pi(\sigma(n))} \notin A^{\pi(\sigma(n))} \Leftrightarrow a^{\pi(\sigma(n))} \notin (\Diamond_i B)^{\pi(\sigma(n))} \Leftrightarrow a^{\pi(\sigma(n))} \in \Delta \setminus (\Diamond_i B)^{\pi(\sigma(n))}$ where $\Delta \setminus$

$(\Diamond_i B)^{\pi(\sigma(n))} = \Delta \setminus \bigcup_{t\in\mathcal{E}_i(\sigma(n))} B^{\pi(t)} = \bigcap_{t\in\mathcal{E}_i(\sigma(n))}(\Delta \setminus B^{\pi(t)}) = \bigcap_{t\in\mathcal{E}_i(\sigma(n))}(\neg B)^{\pi(t)} = (\Box_i \neg B)^{\pi(\sigma(n))}$. Hence, $a^{\pi(\sigma(n))} \in (\Box_i \neg B)^{\pi(\sigma(n))}$. After applying α, $a : \Box_i \neg B$ is added into $\mathbb{L}(n)$ and is satisfied by $(\mathbb{M}, \sigma(n))$. Therefore, $\mathbb{M} \Vdash \mathbb{G}_\alpha$ via σ.

14. If α is an N$\Box$-rule, then $a : \neg A \in \mathbb{L}(n)$, $A \doteq \Box_i B \in \mathcal{T}$, and $a : \Diamond_i \neg B \notin \mathbb{L}(n)$. Since $\mathbb{M} \Vdash \mathbb{G}$ and $\mathbb{M} \vDash \mathcal{T}$, we have $(\mathbb{M}, \sigma(n)) \vDash A \doteq \Box_i B$, $(\mathbb{M}, \sigma(n)) \vDash \neg A(a)$ and therefore $a^{\pi(\sigma(n))} \notin A^{\pi(\sigma(n))} \Leftrightarrow a^{\pi(\sigma(n))} \notin (\Box_i B)^{\pi(\sigma(n))} \Leftrightarrow a^{\pi(\sigma(n))} \in \Delta \setminus (\Box_i B)^{\pi(\sigma(n))}$ where $\Delta \setminus (\Box_i B)^{\pi(\sigma(n))} = \Delta \setminus \bigcap_{t\in\mathcal{E}_i(\sigma(n))} B^{\pi(t)} = \bigcup_{t\in\mathcal{E}_i(\sigma(n))}(\Delta \setminus B^{\pi(t)}) = \bigcup_{t\in\mathcal{E}_i(\sigma(n))}(\neg B)^{\pi(t)} = (\Diamond_i \neg B)^{\pi(\sigma(n))}$. Hence, $a^{\pi(\sigma(n))} \in (\Diamond_i \neg B)^{\pi(\sigma(n))}$. After applying α, $a : \Diamond_i \neg B$ is added into $\mathbb{L}(n)$ and is satisfied by $(\mathbb{M}, \sigma(n))$. Therefore, $\mathbb{M} \Vdash \mathbb{G}_\alpha$ via σ.

It follows that after the application of every expansion rule, the resulting constraint graph $\mathbb{G}_\alpha$ is satisfied by $\mathbb{M}_\alpha$ which, except after applying an $\exists$-rule, is the same as $\mathbb{M}$. When α is an $\exists$-rule, $\mathbb{M}_\alpha$ differs from $\mathbb{M}$ only in the interpretation of the newly picked individual name. Therefore, $\mathcal{T}$ is valid in $\mathbb{M}_\alpha$. Since $\mathbb{M}_\alpha$ is a semantic extension of $\mathbb{M}$ restricted to $N_\Sigma \cup \mathcal{O}_\mathbb{G}$, it is obvious that $\mathbb{M}_\alpha$ satisfies the constraint graph $\mathbb{G}$. ■

A.1.2 Proof of Lemma 2.3.5

Lemma 2.3.5 *Let $\mathcal{T}$ be an acyclic TBox and let $\mathbb{G}$ be an open complete constraint graph w.r.t. local, global and terminological expansion rules. Then for every $A \in N_\mathcal{C}$ and every $a \in \Delta$, $a : \neg A \in \mathbb{L}(n) \Rightarrow (\mathbb{M}_\mathbb{G}, n) \vDash \neg A(a)$.*

Proof There are two cases, and for both, since $\mathbb{G}$ is open, $a : A \notin \mathbb{L}(n)$.

(1) When $A \in \Theta$, $a : \neg A \in \mathbb{L}(n) \Rightarrow a : A \notin \mathbb{L}(n) \Rightarrow a \notin A^{\pi(n)} \Leftrightarrow a \in (\neg A)^{\pi(n)} \Leftrightarrow (\mathbb{M}_\mathbb{G}, n) \vDash \neg A(a)$. The first implication is due to the fact that $\mathbb{G}$ is open. The second implication is by Definition 2.3.4 and the rest equivalences are because of the semantics.

(2) When $A \notin \Theta$, i.e., there is a definition $A \doteq D \in \mathcal{T}$, we will prove by induction on the structure of D. For the base case where the concept names involved in D are elements in Θ, we have the following cases:

1. D is of the form $\neg B$ where $B \in \Theta$. Since $\mathbb{G}$ is complete, $a : B \in \mathbb{L}(n)$. By Definition 2.3.4, $a \in B^{\pi(n)} \Leftrightarrow a \notin (\neg B)^{\pi(n)}$. Since $\mathbb{G}$ is open, $a : \neg A \in \mathbb{L}(n) \Rightarrow a : A \notin \mathbb{L}(n)$. However, $A^{\pi(n)} = \{b \mid b : A \in \mathbb{L}(n)\} \cup (\neg B)^{\pi(n)}$. This implies that $a \notin A^{\pi(n)} \Leftrightarrow a \in (\neg A)^{\pi(n)} \Leftrightarrow (\mathbb{M}_{\mathbb{G}}, n) \vDash \neg A(a)$.

2. D is of the form $B_1 \sqcap B_2$ where $\{B_1, B_2\} \subseteq \Theta$. Since $\mathbb{G}$ is complete, $a : \neg B_1 \sqcup \neg B_2 \in \mathbb{L}(n)$ and $a : \neg B_1$ or $a : \neg B_2$ is in $\mathbb{L}(n)$. W.l.o.g., suppose $a : \neg B_1 \in \mathbb{L}(n)$. Since $\mathbb{G}$ is open, $a : B_1 \notin \mathbb{L}(n)$. Because $B_1 \in \Theta$, $a \notin B_1^{\pi(n)} \Leftrightarrow a \in (\neg B_1)^{\pi(n)} \Rightarrow a \notin (B_1 \sqcap B_2)^{\pi(n)}$. However, $A^{\pi(n)} = \{b \mid b : A \in \mathbb{L}(n)\} \cup (B_1 \sqcap B_2)^{\pi(n)}$ and $a : A \notin \mathbb{L}(n)$. Hence, $a \notin A^{\pi(n)} \Leftrightarrow a \in (\neg A)^{\pi(n)} \Leftrightarrow (\mathbb{M}_{\mathbb{G}}, n) \vDash \neg A(a)$.

3. D is of the form $B_1 \sqcup B_2$ where $\{B_1, B_2\} \subseteq \Theta$. Since $\mathbb{G}$ is complete, $a : \neg B_1 \sqcap \neg B_2 \in \mathbb{L}(n)$ and $\{a : \neg B_1, a : \neg B_2\} \subseteq \mathbb{L}(n)$. Since $\mathbb{G}$ is open, $a : B_1 \notin \mathbb{L}(n)$ and $a : B_2 \notin \mathbb{L}(n)$. Because $\{B_1, B_2\} \subseteq \Theta$, $a \notin B_1^{\pi(n)}$ and $a \notin B_2^{\pi(n)} \Leftrightarrow a \notin (B_1 \sqcup B_2)^{\pi(n)}$. However, $A^{\pi(n)} = \{b \mid b : A \in \mathbb{L}(n)\} \cup (B_1 \sqcup B_2)^{\pi(n)}$ and $a : A \notin \mathbb{L}(n)$. Therefore, $a \notin A^{\pi(n)} \Leftrightarrow a \in (\neg A)^{\pi(n)} \Leftrightarrow (\mathbb{M}_{\mathbb{G}}, n) \vDash \neg A(a)$.

4. D is of the form $\exists R.B$ where $B \in \Theta$. Since $\mathbb{G}$ is complete, $a : \forall R.\neg B \in \mathbb{L}(n)$ and for every b, $(a, b) : R \in \mathbb{L}(n) \Rightarrow b : \neg B \in \mathbb{L}(n)$. Suppose $(a, b) : R \in \mathbb{L}(n)$. Then, $b : \neg B \in \mathbb{L}(n)$, and since $B \in \Theta$ and $\mathbb{G}$ is open, we have $b \notin B^{\pi(n)}$. Moreover, since $R \in N_{\mathcal{R}}$, we have $(a, b) \in R^{\pi(n)}$. It follows that for every b, $(a, b) \in R^{\pi(n)} \Rightarrow b \notin B^{\pi(n)}$. So $a \in (\forall R.\neg B)^{\pi(n)}$ and therefore $a \notin (\exists R.B)^{\pi(n)}$. However, $A^{\pi(n)} = \{c \mid c : A \in \mathbb{L}(n)\} \cup (\exists R.B)^{\pi(n)}$ and $a : A \notin \mathbb{L}(n)$. Hence, $a \notin A^{\pi(n)} \Leftrightarrow a \in (\neg A)^{\pi(n)} \Leftrightarrow (\mathbb{M}_{\mathbb{G}}, n) \vDash \neg A(a)$.

5. D is of the form $\forall R.B$ where $B \in \Theta$. Since $\mathbb{G}$ is complete, $a : \exists R.\neg B \in \mathbb{L}(n)$ and there exists b s.t. $(a, b) : R \in \mathbb{L}(n)$ and $b : \neg B \in \mathbb{L}(n)$. Since $B \in \Theta$ and $\mathbb{G}$ is open, we have $b \notin B^{\pi(n)}$. And since $R \in N_{\mathcal{R}}$, we have $(a, b) \in R^{\pi(n)}$. Therefore, there exists b s.t. $(a, b) \in R^{\pi(n)} \wedge b \notin B^{\pi(n)}$. Thus, $a \in (\exists R.\neg B)^{\pi(n)}$ and hence, $a \notin (\forall R.B)^{\pi(n)}$. However, $A^{\pi(n)} = \{c \mid c : A \in \mathbb{L}(n)\} \cup (\forall R.B)^{\pi(n)}$ and $a : A \notin \mathbb{L}(n)$. Therefore, $a \notin A^{\pi(n)} \Leftrightarrow a \in (\neg A)^{\pi(n)} \Leftrightarrow (\mathbb{M}_{\mathbb{G}}, n) \vDash \neg A(a)$.

6. D is of the form $\Diamond_i B$ where $B \in \Theta$. Since $\mathbb{G}$ is complete, $a : \Box_i \neg B \in \mathbb{L}(n)$ and for

each n' with $i \in \mathbb{L}(n, n')$, $a : \neg B \in \mathbb{L}(n')$. Since $B \in \Theta$ and $\mathbb{G}$ is open, we have $a \notin B^{\pi(n')}$ whenever $i \in \mathbb{L}(n, n')$. Therefore, we have $a \in \bigcap_{n' \in \mathcal{E}_i(n)} (\neg B)^{\pi(n')} \Leftrightarrow a \in (\Box_i \neg B)^{\pi(n)} \Leftrightarrow a \notin (\Diamond_i B)^{\pi(n)}$. However, $A^{\pi(n)} = \{b \mid b : A \in \mathbb{L}(n)\} \cup (\Diamond_i B)^{\pi(n)}$ and $a : A \notin \mathbb{L}(n)$. Hence, $a \notin A^{\pi(n)} \Leftrightarrow a \in (\neg A)^{\pi(n)} \Leftrightarrow (\mathbb{M}_{\mathbb{G}}, n) \vDash \neg A(a)$.

7. D is of the form $\Box_i B$ where $B \in \Theta$. Since $\mathbb{G}$ is complete, $a : \Diamond_i \neg B \in \mathbb{L}(n)$ and there exists n' s.t. $i \in \mathbb{L}(n, n')$ and $a : \neg B \in \mathbb{L}(n')$. Since $B \in \Theta$ and $\mathbb{G}$ is open, we have $a \notin B^{\pi(n')}$. Therefore, we have $a \in \bigcup_{n' \in \mathcal{E}_i(n)} (\neg B)^{\pi(n')} \Leftrightarrow a \in (\Diamond_i \neg B)^{\pi(n)} \Leftrightarrow a \notin (\Box_i B)^{\pi(n)}$. However, $A^{\pi(n)} = \{a \mid a : A \in \mathbb{L}(n)\} \cup (\Box_i B)^{\pi(n)}$ and $a : A \notin \mathbb{L}(n)$. Hence, $a \notin A^{\pi(n)} \Leftrightarrow a \in (\neg A)^{\pi(n)} \Leftrightarrow (\mathbb{M}_{\mathbb{G}}, n) \vDash \neg A(a)$.

Note that for the first five cases, the correctness of the implication $a : \neg A \in \mathbb{L}(n) \Rightarrow (\mathbb{M}_{\mathbb{G}}, n) \vDash \neg A(a)$ depends on the fact that the constraint graph $\mathbb{G}$ has no applicable local or terminological expansion rules. For the last two cases, the correctness of the implication depends on the fact that $\mathbb{G}$ has no applicable global or terminological expansion rules.

The induction step is similar to the corresponding base case, except that in the general case, in order to show that $a \notin D^{\pi(n)}$, we use the induction hypothesis rather than relying on the membership in Θ when none of the concept names occurring in D belong to Θ, and we use both induction hypothesis and the membership in Θ when some of the concept names occurring in D belong to Θ and some don't. ■

A.1.3 An Example for Footnote 2

One may wonder what would happen if the terminological expansion rules go from left to right for definitions involving modalities (to avoid backtracking) and go from right to left for definitions that do not involve modalities. It turns out that using this approach causes the tableau algorithm to become incomplete as is illustrated in Example A.1.1.

Example A.1.1 *Consider a set of expansion rules that contains (i) local and global expansion rules as given in Fig. 2.1, and (ii) terminological expansion rules that contain the T-, N$\Diamond$-, and N$\Box$-rules as given in Fig. 2.2. Suppose that there are also five other rules (corresponding to the N-, N$\sqcap$-, N$\sqcup$-, N$\exists$- and N$\forall$-rules in Fig. 2.2) that examine the right-hand sides of definitions*

in the TBox. For example, the rule "If there is a node n with $\{a : B_1, a : B_2\} \cap \mathbb{L}(n) \neq \emptyset, A \doteq B_1 \sqcup B_2 \in \mathcal{T}$*, and* $a : A \notin \mathbb{L}(n)$*, then* $\mathbb{L}(n) := \mathbb{L}(n) \cup \{a : A\}$*" corresponds to the* $\mathcal{N}\sqcup$*-rule in Fig. 2.2. Now consider a Tbox* $\mathcal{T} = \{A \doteq C_1 \sqcup C_2, C_1 \doteq \Diamond_1 B\}$ *and a constraint tree* $\mathbb{G}$ *containing the constraint systems* $\mathbb{L}(n_0) = \{a : \neg A, a : \Box_1 B, b : \Diamond_1 C\}$ *and* $\mathbb{L}(n_1) = \{b : C, a : B\}$ *where* $1 \in \mathbb{L}(n_0, n_1)$*. With respect to this set of expansion rules,* $\mathbb{G}$ *is complete and open. Suppose that there is a model* $\mathbb{M} \Vdash \mathbb{G}$ *via* σ *and* $\mathbb{M} \models \mathcal{T}$*. Then we have* $(\mathbb{M}, \sigma(n_1)) \models B(a)$ *and* $\mathcal{E}_1(\sigma(n_0), \sigma(n_1))$*, which implies* $(\mathbb{M}, \sigma(n_0)) \models \Diamond_1 B(a)$*. Since* $\mathbb{M} \models \mathcal{T}$ *and* $C_1 \doteq \Diamond_1 B \in \mathcal{T}$*, we have* $(\mathbb{M}, \sigma(n_0)) \models C_1(a)$*. Furthermore, because* $\mathbb{M} \models A \doteq C_1 \sqcup C_2$*, we have* $(\mathbb{M}, \sigma(n_0)) \models A(a)$*. However, the fact that* $\mathbb{M} \Vdash \mathbb{G}$ *and* $a : \neg A \in \mathbb{L}(n_0)$ *implies that* $(\mathbb{M}, \sigma(n_0)) \models \neg A(a)$*, and this contradicts* $(\mathbb{M}, \sigma(n_0)) \models A(a)$*. Hence, there does not exist a model that satisfies* $\mathbb{G}$*. Thus, due to the inability to generate* $a : \neg A$ *in* $\mathbb{L}(n_0)$*, this set of expansion rules fails to detect a potential clash.* ■

A.2 Additional Material for Chapter 3

A.2.1 Proof of Observation (f_3)

f_3. *For all* $a \in \mathcal{O}^f \setminus \mathcal{O}_\Sigma$*, if* a *was introduced via an application of the* $\exists_2^{\mathcal{A}}$*-rule to an assertion* $\exists r.C(b)$*, then for any* $D \in \{D' \mid D'(a) \in \mathcal{A}^f\}$*,* $C \sqsubseteq D \in \mathcal{T}^f$*.*

Proof We prove this by induction on the application of the assertion expansion rules. Let $\mathcal{A}'$ ($\mathcal{A}''$) be the ABox before (after) the application of an expansion rule. We assume that for any individual $a \in \mathcal{O}^f \setminus \mathcal{O}_\Sigma$ and for all $D \in \{D' \mid D'(a) \in \mathcal{A}'\}$, $C \sqsubseteq D \in \mathcal{T}^f$ and we argue that $C \sqsubseteq E \in \mathcal{T}^f$ for all $E \in \{D' \mid D'(a) \in \mathcal{A}''\}$. The base case is when a was just introduced as a result of applying the $\exists_2^{\mathcal{A}}$-rule to $\exists r.C(b)$ and no other rule was applied yet. Then we have $\{C' \mid C'(a) \in \mathcal{A}'\} = \{C\}$ and obviously, $C \sqsubseteq C \in \mathcal{T}^f$.

If the $\sqsubseteq^{\mathcal{A}}$-rule is applied to an assertion $E(a)$ where $E \in \{D' \mid D'(a) \in \mathcal{A}'\}$ and $E \sqsubseteq F \in \mathcal{T}^f$, then we add $F(a) \in \mathcal{A}''$. By IH, $C \sqsubseteq E \in \mathcal{T}^f$. It follows that $C \sqsubseteq F \in \mathcal{T}^f$.

If the $\sqcap^{\mathcal{A}}$-rule is applied to assertions $E_1(a), ..., E_k(a)$ where $E_1, ..., E_k \in \{D' \mid D'(a) \in \mathcal{A}'\}$ and $E_1 \sqcap \cdots \sqcap E_k \in SubE$, then we have $E_1 \sqcap \cdots \sqcap E_k(a) \in \mathcal{A}''$. By IH, $C \sqsubseteq E_i \in \mathcal{T}^f$ $(i = 1, ..., k)$. It follows that $C \sqsubseteq E_1 \sqcap \cdots \sqcap E_k \in \mathcal{T}^f$.

If the $\exists_2^{\mathcal{A}}$-rule is applied to an assertion $\exists r.E(a)$ where $\exists r.E \in \{D' \mid D'(a) \in \mathcal{A}'\}$, then a fresh individual c and new assertions, $r(a,c)$ and $E(c)$, are introduced. Since for every $b \in \mathcal{O}^f \setminus \mathcal{O}_\Sigma$ where $b \neq c$, $\{D' \mid D'(b) \in \mathcal{A}'\} = \{D' \mid D'(b) \in \mathcal{A}''\}$, the claim holds for individual b. For the fresh individual c, we have $\{C' \mid C'(c) \in \mathcal{A}''\} = \{E\}$ and obviously, $E \sqsubseteq E \in \mathcal{T}^f$.

If the $\exists_1^{\mathcal{A}}$-rule is applied to assertions $r(a,c), E(c) \in \mathcal{A}'$, then $\exists r.E(a) \in \mathcal{A}''$ and $\exists r.E \in \{D' \mid D'(a) \in \mathcal{A}''\}$. Since $a \in \mathcal{O}^f \setminus \mathcal{O}_\Sigma$, it follows from (f_1) that $c \in \mathcal{O}^f \setminus \mathcal{O}_\Sigma$ and so, in view of $r(a,c) \in \mathcal{A}'$, the individual c was introduced as a result of applying the $\exists_2^{\mathcal{A}}$-rule to an assertion, say $\exists r.F(a) \in \mathcal{A}'$. By IH, $C \sqsubseteq \exists r.F \in \mathcal{T}^f$ and $F \sqsubseteq E \in \mathcal{T}^f$. It then follows that $C \sqsubseteq \exists r.E \in \mathcal{T}^f$. ■

A.2.2 Proof of Theorem 3.4.3

Theorem 3.4.3 *(Soundness of the Assertion Expansion Rules) Let $\mathcal{A}^f$ be an assertionally closed ABox obtained from Σ by applying the tableau algorithm Λ. For any $C \in \mathcal{C} \cap SubE$ and any $a \in \mathcal{O}^f$, if $C(a) \in \mathcal{A}^f$, then for every model $\mathcal{I} = \langle \Delta, \cdot^{\mathcal{I}} \rangle$ of Σ, there is a semantic extension of $\mathcal{I}_{N_\Sigma}$ that satisfies $C(a)$. In particular, if $a \in \mathcal{O}_\Sigma$, then $\Sigma \models C(a)$.*

Proof Let $\mathcal{I} = \langle \Delta, \cdot^{\mathcal{I}} \rangle$ be an arbitrary model of Σ. We need to show that after applying each of the expansion rules, there is a semantic extension of $\mathcal{I}_{N_\Sigma}$ that satisfies the new assertion(s) being added to $\mathcal{A}^f$. We prove this by induction on the construction of $\mathcal{A}^f$. The base case is when $C(a) \in \mathcal{A}$. Since $\mathcal{I}$ is a model of Σ, so is $\mathcal{I}_{N_\Sigma}$, and hence $\mathcal{I}_{N_\Sigma}$ satisfies $C(a)$. For the induction step, we use $\mathcal{A}'$, $\mathcal{O}'$ and $\mathcal{I}' = \langle \Delta, \cdot^{\mathcal{I}'} \rangle$ to denote the ABox before the application of an expansion rule, the set of individual names appearing in $\mathcal{A}'$, and a model of $\langle \mathcal{A}', \mathcal{T} \rangle$, respectively, where, by IH, $\mathcal{I}' = \mathcal{I}'_{N_\Sigma \cup \mathcal{O}'}$ is a semantic extension of $\mathcal{I}_{N_\Sigma}$. We also denote by $\mathcal{A}''$ the ABox after the application of the expansion rule and by $\mathcal{O}''$ the set of individual names appearing in $\mathcal{A}''$. Note that in the first three cases below, $\mathcal{O}'' = \mathcal{O}'$.

1. If $\sqsubseteq^{\mathcal{A}}$-rule is applicable, then $C(a) \in \mathcal{A}'$, $C \sqsubseteq D \in \mathcal{T}^f$ and $D(a) \notin \mathcal{A}'$. After applying the rule, $D(a) \in \mathcal{A}''$. By IH, $C(a) \in \mathcal{A}'$ implies that $a^{\mathcal{I}'} \in C^{\mathcal{I}'}$. Since $C \sqsubseteq D$ implies $C^{\mathcal{I}'} \subseteq D^{\mathcal{I}'}$, we have $a^{\mathcal{I}'} \in D^{\mathcal{I}'}$. It follows that the newly added assertion $D(a)$ is satisfied by $\mathcal{I}'$.

2. If $\sqcap^{\mathcal{A}}$-rule is applicable, then $\{C_1(a), ..., C_k(a)\} \subseteq \mathcal{A}'$, $C_1 \sqcap \cdots \sqcap C_k(a) \notin \mathcal{A}'$ and $C_1 \sqcap \cdots \sqcap C_k \in SubE$. After applying the rule, $C_1 \sqcap \cdots \sqcap C_k(a) \in \mathcal{A}''$. By IH, $a^{\mathcal{I}'} \in C_i^{\mathcal{I}'} (1 \leq i \leq k)$ which implies $a^{\mathcal{I}'} \in C_1^{\mathcal{I}'} \cap \cdots \cap C_k^{\mathcal{I}'} = (C_1 \sqcap \cdots \sqcap C_k)^{\mathcal{I}'}$. It follows that $\mathcal{I}'$ satisfies the newly added assertion $C_1 \sqcap \cdots \sqcap C_k(a)$.

3. If $\exists_1^{\mathcal{A}}$-rule is applicable, then $\{r(a,b), C(b)\} \subseteq \mathcal{A}'$, $\exists r.C \in SubE$ and $\exists r.C(a) \notin \mathcal{A}'$. After applying the rule, $\exists r.C(a) \in \mathcal{A}''$. By IH, $\{r(a,b), C(b)\} \subseteq \mathcal{A}'$ implies that $(a^{\mathcal{I}'}, b^{\mathcal{I}'}) \in r^{\mathcal{I}'}$ and $b^{\mathcal{I}'} \in C^{\mathcal{I}'}$. It follows that $a^{\mathcal{I}'} \in (\exists r.C)^{\mathcal{I}'}$. So, $\mathcal{I}'$ satisfies the newly added assertion $\exists r.C(a)$.

4. If $\exists_2^{\mathcal{A}}$-rule is applicable, then $\exists r.C(a) \in \mathcal{A}'$, a is not blocked and for all $b \in \mathcal{O}'$, $\{r(a,b), C(b)\} \nsubseteq \mathcal{A}'$. In applying the rule, a fresh individual name c is introduced and after the application, $\{r(a,c), C(c)\} \subseteq \mathcal{A}''$. By IH, $\exists r.C(a) \in \mathcal{A}'$ implies that there is an individual $d \in \Delta$ such that $(a^{\mathcal{I}'}, d) \in r^{\mathcal{I}'}$ and $d \in C^{\mathcal{I}'}$. We define an interpretation $\mathcal{J}$ such that $c^{\mathcal{J}} = d$ and $\mathcal{J}_{N_\Sigma \cup \mathcal{O}'} = \mathcal{I}'_{N_\Sigma \cup \mathcal{O}'}$. Let $\mathcal{O}'' := \mathcal{O}' \cup \{c\}$. It is obvious that $\mathcal{J}_{N_\Sigma \cup \mathcal{O}''}$ is a semantic extension of ${\mathcal{I}'}_{N_\Sigma \cup \mathcal{O}'}$ and we have $(a^{\mathcal{J}}, c^{\mathcal{J}}) \in r^{\mathcal{J}}$ and $c^{\mathcal{J}} \in C^{\mathcal{J}}$.

Next we show that if $a \in \mathcal{O}_\Sigma$, then $\Sigma \vDash C(a)$. The fact that $a \in \mathcal{O}_\Sigma$ implies that $C(a)$ can only be obtained by an application of the $\sqsubseteq^{\mathcal{A}}$-, $\sqcap^{\mathcal{A}}$-, or $\exists_1^{\mathcal{A}}$-rule. There are two cases: (a) If $C(a)$ is obtained through a sequence of applications of the assertion expansion rules without the $\exists_2^{\mathcal{A}}$-rule from $\mathcal{A}$, since at each step, no new individual is created, the semantic extension of $\mathcal{I}$ is itself, and hence $C(a)$ is satisfied by $\mathcal{I}$. Since $\mathcal{I}$ is an arbitrary model of Σ, we have $\Sigma \vDash C(a)$. (b) If in the sequence of applications of obtaining $C(a)$, the $\exists_2^{\mathcal{A}}$-rule has been applied to an assertion $D(a)$, in view that $a \in \mathcal{O}_\Sigma$, by (f_3) and the (EX)-rule in [59], $D \sqsubseteq C \in \mathcal{T}^f$. Therefore, $C(a)$ could have been obtained from $C(a)$ by an application of the $\sqsubseteq^{\mathcal{A}}$-rule. It then follows from case (a) that $\Sigma \vDash C(a)$. ■

A.2.3 Proof of Theorem 3.4.4

To prove the completeness of Λ, we define a *canonical interpretation* $\mathcal{J}_{\mathcal{A}^f} = \langle \Delta, \cdot^{\mathcal{J}} \rangle$ for the assertionally closed ABox $\mathcal{A}^f$ as follows:

- $\Delta := \mathcal{O}^f$;

- $a^{\mathcal{J}} := a$, for every $a \in \mathcal{O}^f$;
- $A^{\mathcal{J}} := \{a \mid A(a) \in \mathcal{A}^f\}$ where $A \in N_{\mathcal{C}} \cap SubE$;
- $r^{\mathcal{J}} := \{(a,b) \mid r(a,b) \in \mathcal{A}^f$ or $r(c,b) \in \mathcal{A}^f$ where $c \in \mathcal{O}^f$ blocks a and is not blocked by other individuals$\}$
- $\mathcal{J}_{\mathcal{A}^f}$ is extended to all of $SubE$ as usual.

Because of blocking, some assertions of the form $\exists r.C(a) \in \mathcal{A}^f$ may not have a witness $d \in \mathcal{O}^f$ such that $\{r(a,d), C(d)\} \subseteq \mathcal{A}^f$. In this case, the canonical interpretation of $\exists r.C$ is obtained by utilizing an individual, say c, that blocks a and is not blocked by other individuals. Then we have $\{\exists r.C(c), r(c,b), C(b)\} \subseteq \mathcal{A}^f$ for some $b \in \mathcal{O}^f$. The individual b can be used as an r successor of the individual a when constructing the interpretation of $\exists r.C$ and therefore a becomes an element of $(\exists r.C)^{\mathcal{J}}$. This is reflected in the definition of $r^{\mathcal{J}}$.

Lemma A.2.1 *For every concept $C \in \mathcal{C} \cap SubE$ and for every individual $a \in \mathcal{O}^f$, $C(a) \in \mathcal{A}^f \Leftrightarrow \mathcal{J}_{\mathcal{A}^f} \vDash C(a)$.*

Proof ($\Longrightarrow$) Assume that $C(a) \in \mathcal{A}^f$. We argue by induction on the structure of C. The base case is when $C \in N_{\mathcal{C}} \cap SubE$. By the definition of $\mathcal{J}_{\mathcal{A}^f}$, $a^{\mathcal{J}} \in C^{\mathcal{J}}$ and hence $\mathcal{J}_{\mathcal{A}^f} \vDash C(a)$.

If $C = C_1 \sqcap \cdots \sqcap C_k$, by the computation of $SubE$, $C_i \in SubE$ $(1 \leq i \leq k)$. Since $\mathcal{A}^f$ is assertionally closed, $\{C_1(a), ..., C_k(a)\} \subseteq \mathcal{A}^f$ due to the $\sqsubseteq^{\mathcal{A}}$-rule. By IH, $C_i(a) \in \mathcal{A}^f \Rightarrow \mathcal{J}_{\mathcal{A}^f} \vDash C_i(a) \Rightarrow a^{\mathcal{J}} \in C_i^{\mathcal{J}}$, $1 \leq i \leq k$. Hence $a^{\mathcal{J}} \in C_1^{\mathcal{J}} \cap \cdots \cap C_k^{\mathcal{J}} = (C_1 \sqcap \cdots \sqcap C_k)^{\mathcal{J}} = C^{\mathcal{J}}$. Therefore, $\mathcal{J}_{\mathcal{A}^f} \vDash C(a)$.

If $C = \exists r.C_1$, there are two cases:

- $\exists r.C_1(a)$ has a witness $b \in \mathcal{O}^f$ such that $\{r(a,b), C_1(b)\} \subseteq \mathcal{A}^f$. Then by the definition of $\mathcal{J}_{\mathcal{A}^f}$, $(a^{\mathcal{J}}, b^{\mathcal{J}}) \in r^{\mathcal{J}}$. By IH, $b^{\mathcal{J}} \in C_1^{\mathcal{J}}$. It follows that $a^{\mathcal{J}} \in (\exists r.C_1)^{\mathcal{J}} = C^{\mathcal{J}}$ and hence, $\mathcal{J}_{\mathcal{A}^f} \vDash C(a)$.
- $\exists r.C_1(a)$ does not have a witness $b \in \mathcal{O}^f$ such that $\{r(a,b), C_1(b)\} \subseteq \mathcal{A}^f$. Then there must exist an individual $c \in \mathcal{O}^f$ that blocks a and $\{r(c,d), C_1(d), \exists r.C_1(c)\} \subseteq \mathcal{A}^f$ for

some $d \in \mathcal{O}^f$. By the definition of $\mathcal{J}_{\mathcal{A}^f}$, $(a^\mathcal{J}, d^\mathcal{J}) \in r^\mathcal{J}$. By IH, $d^\mathcal{J} \in C_1^\mathcal{J}$. It follows that $a^\mathcal{J} \in (\exists r.C_1)^\mathcal{J} = C^\mathcal{J}$. Therefore, $\mathcal{J}_{\mathcal{A}^f} \models C(a)$.

($\Longleftarrow$) It suffices to show that for every concept $C \in \mathcal{C} \cap SubE$ and individual $a \in \mathcal{O}^f$, if $C(a) \notin \mathcal{A}^f$, then $\mathcal{J}_{\mathcal{A}^f} \not\models C(a)$. We again argue by induction on the structure of C. The base case is when $C \in N_\mathcal{C} \cap SubE$. If $C(a) \notin \mathcal{A}^f$, by the definition of $\mathcal{J}_{\mathcal{A}^f}$, $a^\mathcal{J} \notin C^\mathcal{J}$. Therefore, $\mathcal{J}_{\mathcal{A}^f} \not\models C(a)$.

If $C = C_1 \sqcap \cdots \sqcap C_k$, then since $\mathcal{A}^f$ is assertionally closed, $C(a) \notin \mathcal{A}^f$ implies $\{C_1(a), ..., C_k(a)\} \nsubseteq \mathcal{A}^f$ due to the $\sqcap^\mathcal{A}$-rule. So there is a C_i such that $C_i(a) \notin \mathcal{A}^f$, $1 \leq i \leq k$. By IH, $\mathcal{J}_{\mathcal{A}^f} \not\models C_i(a)$. It follows that $a^\mathcal{J} \notin C_i^\mathcal{J}$ and hence $a^\mathcal{J} \notin C_1^\mathcal{J} \cap \cdots \cap C_k^\mathcal{J} = (C_1 \sqcap \cdots \sqcap C_k)^\mathcal{J} = C^\mathcal{J}$. Therefore, $\mathcal{J}_{\mathcal{A}^f} \not\models C(a)$.

If $C = \exists r.C_1$, then since $\mathcal{A}^f$ is assertionally closed, $C(a) \notin \mathcal{A}^f$ implies that there does not exist an individual $b \in \mathcal{O}^f$ such that $\{r(a,b), C_1(b)\} \subseteq \mathcal{A}^f$ due to the $\exists_1^\mathcal{A}$-rule. By the definition of $r^\mathcal{J}$ and IH, for every $b \in \mathcal{O}^f$, either $(a^\mathcal{J}, b^\mathcal{J}) \notin r^\mathcal{J}$ or $b^\mathcal{J} \notin C_1^\mathcal{J}$. It follows that $a^\mathcal{J} \notin (\exists r.C_1)^\mathcal{J} = C^\mathcal{J}$ and hence, $\mathcal{J}_{\mathcal{A}^f} \not\models C(a)$. ■

Corollary A.2.2 *In the canonical interpretation $\mathcal{J}_{\mathcal{A}^f}$, for every $C \in \mathcal{C} \cap SubE$, $C^\mathcal{J} = \{b \in \mathcal{O}^f \mid C(b) \in \mathcal{A}^f\}$.*

Lemma A.2.1 says in effect that the canonical interpretation is a model of the ABox $\mathcal{A}^f$. Next lemma shows that $\mathcal{J}_{\mathcal{A}^f}$ is also a model of the TBox $\mathcal{T}^f$.

Lemma A.2.3 *For all concepts $C, D \in \mathcal{C} \cap SubE$, $C \sqsubseteq D \in \mathcal{T}^f \Rightarrow \mathcal{J}_{\mathcal{A}^f} \models C \sqsubseteq D$.*

Proof The claim is an easy consequence of the $\sqsubseteq^\mathcal{A}$-rule and Lemma A.2.1. For any subsumption $C \sqsubseteq D \in \mathcal{T}^f$ and any $a \in \mathcal{O}^f$, $C(a) \in \mathcal{A}^f \Rightarrow D(a) \in \mathcal{A}^f$ by the $\sqsubseteq^\mathcal{A}$-rule. By Corollary A.2.2, $C^\mathcal{J} = \{b \in \mathcal{O}^f \mid C(b) \in \mathcal{A}^f\}$ and similarly for $D^\mathcal{J}$. It follows that $C^\mathcal{J} \subseteq D^\mathcal{J}$. ■

Theorem 3.4.4 *(Completeness) Let $SubE$ be the set of subexpressions obtained from a KB $\Sigma = \langle \mathcal{A}, \mathcal{T} \rangle$ and a finite set of assertions S (see Section 3.4.1). Let $\mathcal{A}^f$ be an assertionally closed ABox obtained from Σ by applying the tableau algorithm Λ. Then for every concept $C \in \mathcal{C} \cap SubE$ and for every individual $a \in \mathcal{O}^f$, $\Sigma \models C(a) \Rightarrow C(a) \in \mathcal{A}^f$.*

Proof Suppose that $C(a)$ holds in all models of Σ. By Lemma A.2.1, the canonical interpretation $\mathcal{J}_{\mathcal{A}^f}$ is a model of $\mathcal{A}^f$ and hence of $\mathcal{A}$. By Lemma A.2.3, $\mathcal{J}_{\mathcal{A}^f}$ is a model of $\mathcal{T}^f$ and hence of $\mathcal{T}$. It follows that $C(a)$ holds in $\mathcal{J}_{\mathcal{A}^f}$. By Lemma A.2.1, $C(a) \in \mathcal{A}^f$. ■

BIBLIOGRAPHY

[1] Franz Baader, Diego Calvanese, Deborah L. McGuinness, Daniele Nardi, and Peter F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, 2003.

[2] Manfred Schmidt-Schauß and Gert Smolka. Attributive concept descriptions with complements. *Artif. Intell.*, 48(1):1–26, 1991.

[3] Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, and Daniele Nardi. Reasoning in expressive description logics. In John Alan Robinson and Andrei Voronkov, editors, *Handbook of Automated Reasoning*, pages 1581–1634. Elsevier and MIT Press, 2001.

[4] Franz Baader and Ulrike Sattler. An overview of tableau algorithms for description logics. *Studia Logica*, 69(1):5–40, 2001.

[5] Ian Horrocks, Ulrike Sattler, and Stephan Tobies. Practical reasoning for very expressive description logics. *Logic Journal of the IGPL*, 8(3), 2000.

[6] Ian Horrocks. Daml+oil: a description logic for the semantic web. *IEEE Data Eng. Bull.*, 25(1):4–9, 2002.

[7] Ian Horrocks, Peter F. Patel-Schneider, and Frank van Harmelen. From shiq and rdf to owl: the making of a web ontology language. *J. Web Sem.*, 1(1):7–26, 2003.

[8] David E. Bell and Len LaPadula. Secure computer systems: Mathematical foundations. Technical Report ESD-TR-73-278, Vol. 1, MITRE Corp., Bedford, MA, 1974.

[9] David E. Bell and Len LaPadula. Secure computer systems: A mathematical model. Technical Report ESD-TR-73-278, Vol. 2, MITRE Corp., Bedford, MA, 1974.

[10] Sushil Jajodia. Database security and privacy. *ACM Comput. Surv.*, 28(1):129–131, 1996.

[11] Amit Jain and Csilla Farkas. Secure resource description framework: an access control model. In David F. Ferraiolo and Indrakshi Ray, editors, *SACMAT*, pages 121–129. ACM, 2006.

[12] Elisa Bertino, L. R. Khan, Ravi S. Sandhu, and Bhavani M. Thuraisingham. Secure knowledge management: confidentiality, trust, and privacy. *IEEE Transactions on Systems, Man, and Cybernetics, Part A*, 36(3):429–438, 2006.

[13] Piero A. Bonatti and Daniel Olmedilla. Rule-based policy representation and reasoning for the semantic web. In Grigoris Antoniou, Uwe Aßmann, Cristina Baroglio, Stefan Decker, Nicola Henze, Paula-Lavinia Patranjan, and Robert Tolksdorf, editors, *Reasoning Web*, volume 4636 of *Lecture Notes in Computer Science*, pages 240–268. Springer, 2007.

[14] Gianluca Tonti, Jeffrey M. Bradshaw, Renia Jeffers, Rebecca Montanari, Niranjan Suri, and Andrzej Uszok. Semantic web languages for policy representation and reasoning: A comparison of kaos, rei, and ponder. In *International Semantic Web Conference*, pages 419–437, 2003.

[15] Piero A. Bonatti, Claudiu Duma, Norbert Fuchs, Wolfgang Nejdl, Daniel Olmedilla, Joachim Peer, and Nahid Shahmehri. Semantic web policies - a discussion of requirements and research issues. In York Sure and John Domingue, editors, *ESWC*, volume 4011 of *Lecture Notes in Computer Science*, pages 712–724. Springer, 2006.

[16] Lalana Kagal, Tim Berners-Lee, Dan Connolly, and Daniel J. Weitzner. Using semantic web technologies for policy management on the web. In *AAAI*. AAAI Press, 2006.

[17] Lalana Kagal, Timothy W. Finin, and Anupam Joshi. A policy based approach to security for the semantic web. In *International Semantic Web Conference*, pages 402–418, 2003.

[18] Daniel J. Weitzner, Harold Abelson, Tim Berners-Lee, Joan Feigenbaum, James A. Hendler, and Gerald J. Sussman. Information accountability. *Commun. ACM*, 51(6):82–87, 2008.

[19] George L. Sicherman, Wiebren de Jonge, and Reind P. van de Riet. Answering queries without revealing secrets. *ACM Trans. Database Syst.*, 8(1):41–59, 1983.

[20] Joachim Biskup and Piero A. Bonatti. Controlled query evaluation with open queries for a decidable relational submodel. *Ann. Math. Artif. Intell.*, 50(1-2):39–77, 2007.

[21] Joachim Biskup and Lena Wiese. A sound and complete model-generation procedure for consistent and confidentiality-preserving databases. *Theor. Comput. Sci.*, 412(31):4044–4072, 2011.

[22] Joachim Biskup. History-dependent inference control of queries by dynamic policy adaption. In Yingjiu Li, editor, *DBSec*, volume 6818 of *Lecture Notes in Computer Science*, pages 106–121. Springer, 2011.

[23] Jie Bao, Giora Slutzki, and Vasant Honavar. Privacy-preserving reasoning on the semantic web. In *Web Intelligence*, pages 791–797. IEEE Computer Society, 2007.

[24] Jia Tao, Giora Slutzki, and Vasant Honavar. Secrecy-preserving query answering for instance checking in $\mathcal{EL}$. In Pascal Hitzler and Thomas Lukasiewicz, editors, *RR*, volume 6333 of *Lecture Notes in Computer Science*, pages 195–203. Springer, 2010.

[25] Franz Baader. Terminological cycles in a description logic with existential restrictions. In *IJCAI'03: Proceedings of the 18th international joint conference on Artificial intelligence*, pages 325–330, San Francisco, CA, USA, 2003. Morgan Kaufmann Publishers Inc.

[26] Sebastian Brandt. Polynomial time reasoning in a description logic with existential restrictions, gci axioms, and - what else? In Ramon López de Mántaras and Lorenza Saitta, editors, *ECAI*, pages 298–302. IOS Press, 2004.

[27] F. Baader. Computing the least common subsumer in the description logic EL wrt terminological cycles with descriptive semantics. In *Proceedings of the 11th International Conference on Conceptual Structures, ICCS*, volume 2746, pages 117–130. Springer, 2003.

[28] Adila Krisnadhi and Carsten Lutz. Data complexity in the el family of dls. In Calvanese et al. [101].

[29] N. Novakovic. Proof-theoretic Approach to Deciding Subsumption and Computing Least Common Subsumer in EL wrt Hybrid TBoxes. In *Logics in Artificial Intelligence: 11th European Conference, JELIA 2008, Dresden, Germany, September 28-October 1, 2008. Proceedings*, page 311. Springer-Verlag New York Inc, 2008.

[30] K. Spackman. Managing clinical terminology hierarchies using algorithmic calculation of subsumption: Experience with SNOMED-RT. *J. of the Amer. Med. Informatics Assoc.*, 2000.

[31] A. Rector and I. Horrocks. Experience building a large, re-usable medical ontology using a description logic with transitivity and concept inclusions. In *Proceedings of the Workshop on Ontological Engineering, AAAI Spring Symposium (AAAI97), Stanford, CA*, pages 321–325, 1997.

[32] Ronald Fagin, Joseph Y. Halpern, Yoram Moses, and Moshe Y. Vardi. *Reasoning about Knowledge*. MIT Press, 1995.

[33] John-Jules Ch. Meyer and Wiebe Van Der Hoek. *Epistemic Logic for AI and Computer Science*. Cambridge University Press, 2004.

[34] Patrick Blackburn, Johan F. A. K. van Benthem, and Frank Wolter. *Handbook of Modal Logic, Volume 3 (Studies in Logic and Practical Reasoning)*. Elsevier Science Inc., New York, NY, USA, 2006.

[35] Riccardo Rosati. On the semantics of epistemic description logics. In Lin Padgham, Enrico Franconi, Manfred Gehrke, Deborah L. McGuinness, and Peter F. Patel-Schneider,

editors, *Description Logics*, volume WS-96-05 of *AAAI Technical Report*, pages 185–188. AAAI Press, 1996.

[36] Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati. Epistemic first-order queries over description logic knowledge bases. In Parsia et al. [102].

[37] Francesco M. Donini, Maurizio Lenzerini, Daniele Nardi, Andrea Schaerf, and Werner Nutt. Adding epistemic operators to concept languages. In *KR*, pages 342–353, 1992.

[38] Francesco M. Donini, Maurizio Lenzerini, Daniele Nardi, Werner Nutt, and Andrea Schaerf. An epistemic operator for description logics. *Artif. Intell.*, 100(1-2):225–274, 1998.

[39] Franz Baader and Armin Laux. Terminological logics with modal operators. In *IJCAI (1)*, pages 808–815, 1995.

[40] Carsten Lutz, Holger Sturm, Frank Wolter, and Michael Zakharyaschev. A tableau decision algorithm for modalized $\mathcal{ALC}$ with constant domains. *Studia Logica*, 72(2):199–232, 2002.

[41] Franz Baader and Hans Jürgen Ohlbach. A multi-dimensional terminological knowledge representation language. *Journal of Applied Non-Classical Logics*, 5(2):153–197, 1995.

[42] Milenko Mosurovic and Michael Zakharyaschev. On the complexity of description logics with modal operators. In George Koletsos and Phokion G. Kolaitis, editors, *2nd PLS*, pages 166–171, 1999.

[43] Frank Wolter and Michael Zakharyaschev. Multi-dimensional description logics. In *IJCAI*, pages 104–109. Morgan Kaufmann, 1999.

[44] Frank Wolter and Michael Zakharyaschev. Satisfiability problem in description logics with modal operators. In *KR*, pages 512–523, 1998.

[45] Frank Wolter and Michael Zakharyaschev. Dynamic description logics. In Michael Zakharyaschev, Krister Segerberg, Maarten de Rijke, and Heinrich Wansing, editors, *Advances in Modal Logic*, pages 431–446. CSLI Publications, 1998.

[46] Frank Wolter and Michael Zakharyaschev. Modal description logics: Modalizing roles. *Fundam. Inform.*, 39(4):411–438, 1999.

[47] Frank Wolter and Michael Zakharyaschev. Decidable fragments of first-order modal logics. *J. Symb. Log.*, 66(3):1415–1438, 2001.

[48] Francesco M. Donini, Daniele Nardi, and Riccardo Rosati. Description logics of minimal knowledge and negation as failure. *ACM Transactions on Computational Logic (TOCL)*, 3(2):225, 2002.

[49] Stephan Tobies. Complexity results and practical algorithms for logics in knowledge representation. *CoRR*, cs.LO/0106031, 2001.

[50] Martin Buchheit, Francesco M. Donini, and Andrea Schaerf. Decidable reasoning in terminological knowledge representation systems. *J. Artif. Intell. Res. (JAIR)*, 1:109–138, 1993.

[51] Jan Hladik and Rafael Peñaloza. PSPACE automata for description logics. In *2006 International Workshop on Description Logics DL'06*, page 74, 2006.

[52] Franz Baader and Werner Nutt. Basic description logics. In Baader et al. [1], pages 43–95.

[53] Ian Horrocks, Ullrich Hustadt, Ulrike Sattler, and Renate Schmidt. Computational modal logic. In Patrick Blackburn, Johan van Benthem, and Frank Wolter, editors, *Handbook of Modal Logic*, chapter 4, pages 181–245. Elsevier, 2006.

[54] Carsten Lutz. Complexity of terminological reasoning revisited. In Harald Ganzinger, David A. McAllester, and Andrei Voronkov, editors, *LPAR*, volume 1705 of *Lecture Notes in Computer Science*, pages 181–200. Springer, 1999.

[55] Bernhard Nebel. Terminological reasoning is inherently intractable. *Artif. Intell.*, 43(2):235–249, 1990.

[56] Moshe Y. Vardi. A model-theoretic analysis of monotonic knowledge. In *Proc. Ninth International Joint Conference on Artificial Intelligence (IJCAI'85)*, pages 509–512, 1985.

[57] Franz Baader, Jan Hladik, and Rafael Peñaloza. Automata can show PSPACE results for description logics. *Inf. Comput.*, 206(9-10):1045–1056, 2008.

[58] Csilla Farkas, Alexander Brodsky, and Sushil Jajodia. Unauthorized inferences in semistructured databases. *Information Sciences*, 176:3269–3299, 2006.

[59] Martin Hofmann. Proof-theoretic approach to description-logic. In *LICS*, pages 229–237. IEEE Computer Society, 2005.

[60] S. Brandt. Reasoning in ELH wrt General Concept Inclusion Axioms. Technical report, Dresden, TU, 2004.

[61] Franz Baader and Rafael Peñaloza. Axiom pinpointing in general tableaux. In Nicola Olivetti, editor, *TABLEAUX*, volume 4548 of *Lecture Notes in Computer Science*, pages 11–27. Springer, 2007.

[62] Franz Baader, Rafael Peñaloza, and Boontawee Suntisrivaraporn. Pinpointing in the description logic el. In Calvanese et al. [101].

[63] Rafael Peñaloza and Baris Sertkaya. Axiom pinpointing is hard. In Bernardo Cuenca Grau, Ian Horrocks, Boris Motik, and Ulrike Sattler, editors, *Description Logics*, volume 477 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2009.

[64] Sylvia Osborn, Ravi Sandhu, and Qamar Munawer. Configuring role-based access control to enforce mandatory and discretionary access control policies. *ACM Trans. Inf. Syst. Secur.*, 3(2):85–106, 2000.

[65] F.Y. Chin. Security in statistical databases for queries with small counts. *ACM Transactions on Database Systems*, 3:92–104, 1978.

[66] Bernardo Cuenca Grau and Ian Horrocks. Privacy-preserving query answering in logic-based information systems. In *Proc. of the 18th Eur. Conf. on Artificial Intelligence (ECAI 2008)*, 2008.

[67] Christine M. O'Keefe, Ming Yung, Lifang Gu, and Rohan Baxter. Privacy-preserving data linkage protocols. In *WPES '04: Proceedings of the 2004 ACM workshop on Privacy in the electronic society*, pages 94–102, New York, NY, USA, 2004. ACM.

[68] Lifang Gu, Rohan Baxter, Deanne Vickers, and Chris Rainsford. Record linkage: Current practice and future directions. Technical report, CSIRO Mathematical and Information Sciences CMIS Technical Report No. 03/83, 2003.

[69] Rakesh Agrawal and Ramakrishnan Srikant. Privacy-preserving data mining. In *Proc. of the ACM SIGMOD Conference on Management of Data*, pages 439–450. ACM Press, May 2000.

[70] Yehuda Lindell and Benny Pinkas. Privacy Preserving Data Mining. *Journal of Cryptology*, 15(3):177–206, 2002. An extended abstract appeared at the CRYPTO 2000 conference.

[71] C. Clifton, M. Kantarcioglu, J. Vaidya, X. Lin, , and M. Zhu. Tools for Privacy Preserving Distributed Data Mining. *ACM SIGKDD Explorations*, 4(2), December 2002.

[72] Mark Giereth. On partial encryption of rdf-graphs. In Yolanda Gil, Enrico Motta, V. Richard Benjamins, and Mark A. Musen, editors, *International Semantic Web Conference*, volume 3729 of *Lecture Notes in Computer Science*, pages 308–322. Springer, 2005.

[73] Fabian Abel, Juri Luca De Coi, Nicola Henze, Arne Wolf Koesling, Daniel Krause, and Daniel Olmedilla. Enabling advanced and context-dependent access control in rdf stores. In Karl Aberer, Key-Sun Choi, Natasha Fridman Noy, Dean Allemang, Kyung-Il Lee, Lyndon J. B. Nixon, Jennifer Golbeck, Peter Mika, Diana Maynard, Riichiro Mizoguchi, Guus Schreiber, and Philippe Cudré-Mauroux, editors, *ISWC/ASWC*, volume 4825 of *Lecture Notes in Computer Science*, pages 1–14. Springer, 2007.

[74] Franz Baader, Martin Knechtel, and Rafael Peñaloza. A generic approach for large-scale ontological reasoning in the presence of access restrictions to the ontology's axioms. In Abraham Bernstein, David R. Karger, Tom Heath, Lee Feigenbaum, Diana Maynard, Enrico Motta, and Krishnaprasad Thirunarayan, editors, *International Semantic Web Conference*, volume 5823 of *Lecture Notes in Computer Science*, pages 49–64. Springer, 2009.

[75] Sabrina De Capitani di Vimercati, Pierangela Samarati, and Sushil Jajodia. Policies, models, and languages for access control. In Subhash Bhalla, editor, *DNIS*, volume 3433 of *Lecture Notes in Computer Science*, pages 225–237. Springer, 2005.

[76] Sushil Jajodia, Pierangela Samarati, Maria Luisa Sapino, and V. S. Subrahmanian. Flexible support for multiple access control policies. *ACM Trans. Database Syst.*, 26(2):214–260, 2001.

[77] Daniel J. Weitzner, Jim Hendler, Tim Berners-Lee, and Dan Connolly. Creating a policy-aware web: Discretionary, rule-based access for the world wide web. In E. Ferrari and B.Thuraisingham, editors, *Web and Information Security*. Idea Group, In press.

[78] Lalana Kagal, Massimo Paolucci, Naveen Srinivasan, Grit Denker, Timothy W. Finin, and Katia P. Sycara. Authorization and privacy for semantic web services. *IEEE Intelligent Systems*, 19(4):50–56, 2004.

[79] Simon Godik and Tim Moses (ed.). Oasis extensible access control markup language (xacml). OASIS Committee Secification cs-xacml-specification-1.0, November 2002, http://www.oasis-open.org/committees/xacml/, 2002.

[80] Vladimir Kolovski, James A. Hendler, and Bijan Parsia. Analyzing web access control policies. In Carey L. Williamson, Mary Ellen Zurko, Peter F. Patel-Schneider, and Prashant J. Shenoy, editors, *WWW*, pages 677–686. ACM, 2007.

[81] Joseph Y. Halpern and Vicky Weissman. Using first-order logic to reason about policies. *ACM Trans. Inf. Syst. Secur.*, 11(4), 2008.

[82] B. Thuraisingham. The use of conceptual structures for handling the inference problem, and cover stories for database security. In *Proc. Fifth IFIP WG*, volume 11.

[83] Frédéric Cuppens and Alban Gabillon. Cover story management. *Data Knowl. Eng.*, 37(2):177–201, 2001.

[84] Marianne Winslett, Kenneth Smith, and Xiaolei Qian. Formal query languages for secure relational databases. *ACM Trans. Database Syst.*, 19(4):626–662, 1994.

[85] Piero A. Bonatti, Sarit Kraus, and V. S. Subrahmanian. Foundations of secure deductive databases. *IEEE Trans. Knowl. Data Eng.*, 7(3):406–422, 1995.

[86] Joachim Biskup. For unknown secrecies refusal is better than lying. *Data Knowl. Eng.*, 33(1):1–23, 2000.

[87] Joachim Biskup and Piero A. Bonatti. Lying versus refusal for known potential secrets. *Data Knowl. Eng.*, 38(2):199–222, 2001.

[88] Joachim Biskup and Piero A. Bonatti. Confidentiality policies and their enforcement for controlled query evaluation. In Dieter Gollmann, Günter Karjoth, and Michael Waidner, editors, *ESORICS*, volume 2502 of *Lecture Notes in Computer Science*, pages 39–54. Springer, 2002.

[89] Joachim Biskup and Piero A. Bonatti. Controlled query evaluation for known policies by combining lying and refusal. *Ann. Math. Artif. Intell.*, 40(1-2):37–62, 2004.

[90] Joachim Biskup and Piero A. Bonatti. Controlled query evaluation for enforcing confidentiality in complete information systems. *Int. J. Inf. Sec.*, 3(1):14–27, 2004.

[91] Joachim Biskup and Lena Wiese. On finding an inference-proof complete database for controlled query evaluation. In Ernesto Damiani and Peng Liu, editors, *DBSec*, volume 4127 of *Lecture Notes in Computer Science*, pages 30–43. Springer, 2006.

[92] Joachim Biskup, Dominique Marc Burgard, Torben Weibert, and Lena Wiese. Inference control in logic databases as a constraint satisfaction problem. In Patrick Drew McDaniel

and Shyam K. Gupta, editors, *ICISS*, volume 4812 of *Lecture Notes in Computer Science*, pages 128–142. Springer, 2007.

[93] Joachim Biskup and Lena Wiese. Combining consistency and confidentiality requirements in first-order databases. In Pierangela Samarati, Moti Yung, Fabio Martinelli, and Claudio Agostino Ardagna, editors, *ISC*, volume 5735 of *Lecture Notes in Computer Science*, pages 121–134. Springer, 2009.

[94] Joachim Biskup and Torben Weibert. Keeping secrets in incomplete databases. *Int. J. Inf. Sec.*, 7(3):199–217, 2008.

[95] Joachim Biskup, Jens Seiler, and Torben Weibert. Controlled query evaluation and inference-free view updates. In Ehud Gudes and Jaideep Vaidya, editors, *DBSec*, volume 5645 of *Lecture Notes in Computer Science*, pages 1–16. Springer, 2009.

[96] Joachim Biskup, Christian Gogolin, Jens Seiler, and Torben Weibert. Requirements and protocols for inference-proof interactions in information systems. In Michael Backes and Peng Ning, editors, *ESORICS*, volume 5789 of *Lecture Notes in Computer Science*, pages 285–302. Springer, 2009.

[97] Johann A. Makowsky. Why horn formulas matter in computer science: Initial structures and generic examples. *Journal of Computer and System Sciences*, 34(2-3):266–292, 1987.

[98] Markus Krötzsch, Sebastian Rudolph, and Pascal Hitzler. Complexity boundaries for horn description logics. In *AAAI*, pages 452–457. AAAI Press, 2007.

[99] William F. Dowling and Jean H. Gallier. Linear-time algorithms for testing the satisfiability of propositional horn formulae. *J. Log. Program.*, 1(3):267–284, 1984.

[100] Dimitris J. Kavvadias, Christos H. Papadimitriou, and Martha Sideri. On horn envelopes and hypergraph transversals. In Kam-Wing Ng, Prabhakar Raghavan, N. V. Balasubramanian, and Francis Y. L. Chin, editors, *ISAAC*, volume 762 of *Lecture Notes in Computer Science*, pages 399–405. Springer, 1993.

[101] Diego Calvanese, Enrico Franconi, Volker Haarslev, Domenico Lembo, Boris Motik, Anni-Yasmin Turhan, and Sergio Tessaris, editors. *Proceedings of the 2007 International Workshop on Description Logics (DL2007), Brixen-Bressanone, near Bozen-Bolzano, Italy, 8-10 June, 2007*, volume 250 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2007.

[102] Bijan Parsia, Ulrike Sattler, and David Toman, editors. *Proceedings of the 2006 International Workshop on Description Logics (DL2006), Windermere, Lake District, UK, May 30 - June 1, 2006*, volume 189 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2006.