12-2010

A PSpace Tableau Algorithm for Acyclic Modalized ALC

Jia Tao
Iowa State University, jtao510@gmail.com

Giora Slutzki
Iowa State University, slutzki@iastate.edu

Vasant Honavar
Iowa State University

A PSpace Tableau Algorithm for Acyclic Modalized $\mathcal{ALC}$

Jia Tao, Giora Slutzki and Vasant Honavar

TR 11-01
Dec 2010

Department of Computer Science
226 Atanasoff Hall
Iowa State University
Ames, Iowa 50010-1040, USA

A PSpace Tableau Algorithm for Acyclic Modalized $\mathcal{ALC}$

Jia Tao, Giora Slutzki, and Vasant Honavar

Iowa State University, Ames, IA, USA

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, we are able 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 $\mathbf{K}_{\mathcal{ALC}}$.
keywords Description Logic, $\mathcal{ALC}$, Modal Logic, Tableau Algorithm, PSpace

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 [2–5]. 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 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* [8–10] into DLs [11–16]. 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 [17–19]. In a series of papers, Wolter and Zakharyaschev [20–23] 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 [16].

There are also papers that provide decision procedures for languages that augment $\mathcal{ALC}$ with modal operators. For example, Donini et al. [13, 14] 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 $\mathbf{A}$ [24] such that the resulting language combines the non-first-order features of frame-based systems with default reasoning. Baader and Laux [15] 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 [16]), are interpreted in the classic multi-modal logic $\mathbf{K}_m$. 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 [8]) where all worlds share the same interpretation domain. It has been shown in [20] 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 paper is based on the constant domain assumption.

Lutz et al. [16] 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 [18]. In contrast, the satisfiability problem for $\mathcal{ALC}$ is known to be PSpace-complete [25]. 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.

Given an $\mathcal{ALCK}_m$ *knowledge base* (KB) Σ, as in the case of DL knowledge bases (see [26]), 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. It is well-known that problems (2)-(4) can be reduced to KB-satisfiability in linear time [26]. Following [13, 14], our queries are of the form $C(a)$ or $R(a, b)$. We solve the query answering problem (whether the KB entails the query), i.e., the instance checking problem, by reducing it to the KB-satisfiability problem.

The main contribution of this paper is a novel PSpace implementation for the satisfiability of an $\mathcal{ALCK}_m$ query with respect to an $\mathcal{ALCK}_m$ knowledge base. This extends the result of Hladik and Peñaloza [27] that the satisfiability of $\mathcal{ALC}$ concepts 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. A novel extension of the idea of canonical interpretation [28, 26] that incorporates the TBox definitions into the interpretation of concept names.

To the best of our knowledge, the main result of this paper as well as the technical approach used to arrive at it are new.

The paper is organized as follows. Section 2 introduces the syntax and semantics of $\mathcal{ALCK}_m$. We proceed to develop a sound and complete algorithm for $\mathcal{ALCK}_m$ KB-satisfiability with an acyclic TBox in Section 3, and then provide the solution to the query answering problem in Section 4. Section 5 shows a PSPACE implementation for $\mathcal{ALCK}_m$ KB-satisfiability. Section 6 develops a sound and complete algorithm for $\mathcal{ALCK}_m$ KB-satisfiability w.r.t. the class of $S4$-models and provides a PSPACE implementation for the algorithm. Section 7 concludes the paper.

2 Preliminaries

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*, $C, D \in \mathcal{C}$, $R \in N_\mathcal{R}$ and $i \in N_\mathcal{E}$.

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 [8]. 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 $a \in N_\mathcal{O}, A \in N_\mathcal{C}, R \in N_\mathcal{R}, C \in \mathcal{C}$,

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

$$(\forall R.C)^{\pi(s)} = \{a \in \Delta \mid \forall b : (a, b) \in R^{\pi(s)} \rightarrow b \in C^{\pi(s)}\}$$
$$(\exists R.C)^{\pi(s)} = \{a \in \Delta \mid \exists b : (a, b) \in R^{\pi(s)} \wedge b \in C^{\pi(s)}\}$$

Definition 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:*

$$(\mathbb{M}, s) \models C \quad \Leftrightarrow C^{\pi(s)} \neq \emptyset \qquad (\mathbb{M}, s) \models R(a, b) \Leftrightarrow (a^{\pi(s)}, b^{\pi(s)}) \in R^{\pi(s)}$$
$$(\mathbb{M}, s) \models C(a) \Leftrightarrow a^{\pi(s)} \in C^{\pi(s)} \qquad (\mathbb{M}, s) \models A \doteq C \Leftrightarrow A^{\pi(s)} = C^{\pi(s)}$$

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

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 knowledge bases 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. *A world $w = (\mathbb{M}, s)$* satisfies *a knowledge base $\Sigma = \langle \mathcal{A}, \mathcal{T} \rangle$, written as $w \models \Sigma$, 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 $\Sigma \models C(a)$, if for all worlds w, $w \models \Sigma \Rightarrow w \models C(a)$.*

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 $\Sigma \models C(a)$,
- NO, if $\Sigma \models \neg C(a)$,
- 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 1. 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. ■

3 Tableau Algorithm

As discussed in Section 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* [13, 15, 14, 16], 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 [16, 29], 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

[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.

$\mathbb{E}$ 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 *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 Fig.1.

Local Expansion Rules:

$\sqcap$-rule: If there is a node n with $a : C_1 \sqcap C_2 \in \mathbb{L}(n)$,
and $\{a : C_1, a : C_2\} \nsubseteq \mathbb{L}(n)$,
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:

$\Diamond\mathcal{C}$-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 $\mathbb{L}(n') := \{a : C\}$;

$\Box\mathcal{C}$-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 $\mathbb{L}(n') := \mathbb{L}(n') \cup \{a : C\}$.

Fig. 1. The local and global expansion rules for $\mathcal{ALCK}_m$

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 [30], 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 [30].

Nebel has shown that the straightforward unfolding of an ABox w.r.t. a TBox may lead to an exponential blowup [31]. To give a PSPACE complexity result for reasoning $\mathcal{ALC}$ with acyclic TBoxes, instead of unfolding iteratively as in [31], the approach in [30] ensures that if an 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 Fig. 2 deal with this issue.

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 $\{a : \neg A, a : B\} \nsubseteq \mathbb{L}(n)$, 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 $\mathbb{L}(n) := \mathbb{L}(n) \cup \{a : \Diamond_i \neg B\}$.

Fig. 2. The terminological expansion rules for $\mathcal{ALCK}_m$

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 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.

Definition 3. *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* $n, n' \in \mathbb{V}$*,*

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

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 3 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 1. *(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,* $\mathbb{M}_\alpha \Vdash \mathbb{G}$*.*

Theorem 1 (proof is given in Appendix A) 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 [28, 26]. In [28], 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 [26] 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 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 B).

Lemma 2. *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. *(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 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. 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 applies to general directed graphs (rather than just trees as, e.g., in [29]). Secondly, it is crucial that $\mathbb{G}$ is complete w.r.t. all the local, global and terminological expansion rules as given in Fig.1 and Fig. 2.

Corollary 1. *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,* $\mathbb{M} \Vdash \mathbb{G} \Longrightarrow (\mathbb{M}_{\mathbb{G}} \Vdash \mathbb{G} \wedge \mathbb{M}_{\mathbb{G}} \models \mathcal{T})$.

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 5, this idea facilitates the PSPACE implementation of the $\mathcal{K}$-tableau algorithm $\Lambda_{\mathcal{K}}$ [2].

4 Query Answering

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

Theorem 3. *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 of 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, this means that for all $(\mathbb{M}, s)$, $(\mathbb{M}, s) \models \Sigma \Rightarrow (\mathbb{M}, s) \models C(a)$. Suppose $\mathbb{G}$ is an open and complete constraint graph generated by $\Lambda_{\mathcal{K}}$ starting from n_0. By Theorem 2, $\mathbb{M}_{\mathbb{G}} \Vdash \mathbb{G}$ and $\mathbb{M}_{\mathbb{G}} \models \mathcal{T}$. By Theorem 1, $(\mathbb{M}_{\mathbb{G}}, n_0) \models \mathbb{L}(n_0)$. 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 $\Sigma \models C(a)$.

[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 C.

Proof of Claim 2. Suppose $\Sigma \not\models C(a)$. By Definition 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 $(\mathbb{M}_0, s_0) \models \mathcal{T}$ 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 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 1, 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 $\mathbb{M} \models \mathcal{T}$. Thus, $\mathbb{M} \Vdash \mathbb{G}$. By Corollary 1, $\mathbb{M}_{\mathbb{G}} \Vdash \mathbb{G}$ and $\mathbb{M}_{\mathbb{G}} \models \mathcal{T}$ where $\mathbb{M}_{\mathbb{G}}$ is the canonical Kripke structure of $\mathbb{G}$. It follows from Theorem 2 that $\mathbb{G}$ is open. ■

We revisit Example 1 to illustrate the use of tableau algorithm to answer queries against a modalized $\mathcal{ALC}$ knowledge base.

Example 2. (Example 1 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), ¬BasicCourse(cs525)}
$\mathcal{T}$ = { Lecturer $\doteq$ ∀TEACHES.BasicCourse, Advisor $\doteq$ Professor ⊓ A,
$A \doteq$ ∃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 ⊔ or possibility of generating a concept involving ⊔, there is only one complete constraint graph that can be constructed from $\mathcal{A} \cup$ {¬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 : ∃ADVISE.Grad, (john,x) : ADVISE, x : Grad,
john : ¬Professor, cs525 : ¬BasicCourse }
Because of the constraints "john : Professor" and "john : ¬Professor", $\mathbb{L}(n_0)$ has a clash and the constraint graph is closed. Hence, $\Sigma_1 \models$ 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 $\mathcal{A} \cup$ {¬Lecturer(susan)} and end up with an open complete constraint graph $\mathbb{G}_1$ as 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 : ∃ADVISE.Grad, (john,x) : ADVISE, x : Grad,
cs525 : ¬BasicCourse, susan : ¬Lecturer, susan : ∃TEACHES.¬BasicCourse }
$\mathbb{L}(n_1)$ ={ susan : Advisor, susan : Professor, susan : A,
susan : ∃ADVISE.Grad, (susan,y) : ADVISE, y : Grad }
$\mathbb{L}(n_2)$ ={ mary : Grad, susan : Lecturer, susan : ∀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 $\langle \mathcal{A} \cup \{\neg\text{Lecturer(susan)}\}, \mathcal{T} \rangle$. 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 : ∃ADVISE.Grad, (john,x) : ADVISE, x : Grad, cs525 : ¬BasicCourse,
susan : Lecturer, susan : ∀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 1 will be answered in the same way. ■

5 A 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 Fig. 3). The expansion procedure creates two kinds of successors: successors of individuals w.r.t. roles that are created due to the ∃-rule, and successors of the current constraint system that are created due to the $\Diamond\mathcal{C}$-rule.

Within each constraint system, before applying the ∃-rule or the $\Diamond\mathcal{C}$-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 ∃-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 ∃-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 $\Diamond\mathcal{C}$-rule will be applied and a new constraint system $\mathbb{L}^x(n')$ will be created (see Fig. 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 $\Diamond\mathcal{C}$-rule will be applied to another possible constraint and 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.

The following example illustrates the operation of Algorithm 1.

Example 3. 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;

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 $\Box\mathcal{C}$-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"

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.

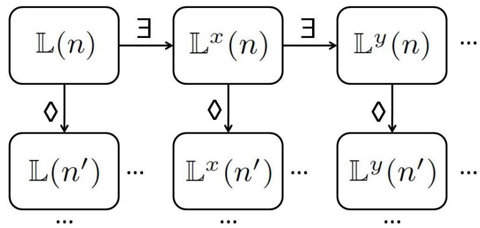

Fig. 3. An Illustration of the Execution of Algorithm 1

Eventually, the algorithm returns "satisfiable". ■

In reference to Fig. 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 Fig. 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 4. *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 $\Diamond\mathcal{C}$-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 $\Diamond\mathcal{C}$-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. ■

6 $S4$-Tableau Algorithm for $\mathcal{ALCK}_m$

In this section, we are interested in models for $\mathcal{ALCK}_m$ knowledge bases and queries that have some special, epistemically motivated, properties. Specifically, the models we want are $S4$-structures.

Definition 5. *A Kripke structure $\mathbb{M} = \langle S, \pi, \mathcal{E}_1, ..., \mathcal{E}_m \rangle$ is* reflexive *if for every $s \in S$, for every $i \in N_\mathcal{E}$, $(s,s) \in \mathcal{E}_i$; $\mathbb{M}$ is* transitive *if for every $s,t,u \in S$, for every $i \in N_\mathcal{E}, (s,t) \in \mathcal{E}_i \wedge (t,u) \in \mathcal{E}_i \Rightarrow (s,u) \in \mathcal{E}_i$. A Kripke structure $\mathbb{M} = \langle S, \pi, \mathcal{E}_1, ..., \mathcal{E}_m \rangle$ is an $S4$-structure if each $\mathcal{E}_i$ is reflexive and transitive.*

The modal logic $S4$ is well-suited to express epistemic knowledge in multiagent environments. This point was argued eloquently in [32]. 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. In particular, analogously to the axioms (A3) and (A4) in [9], we have

(i) (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 $w \models C(a)$.

(ii) (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 $w \models \Box_i \Box_i C(a)$.

Now, 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 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, 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 $\mathbb{L}(n) = \mathbb{L}(n) \cup \{a : C\}$.
$A^4\mathcal{C}$-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 $\mathbb{L}(n') := \mathbb{L}(n') \cup \{a : \Box_i C\}$.

Fig. 4. The accessibility expansion rules

To address this problem, we adapt the $\mathcal{K}$-tableau algorithm by adding accessibility expansion rules that will facilitate the construction of $S4$-structures. They are shown in Fig. 4. Unfortunately, a tableau algorithm with the accessibility expansion rules and the current local, global and terminological rules may not terminate [3]. 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 $\Diamond\mathcal{C}$-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 $A^4\mathcal{C}$-rule followed by an 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 keep creating the same constraint system without terminating. However, since $S4$-structures are reflexive, any world in the structure is an i-successor of itself ($i \in N_{\mathcal{E}}$). This suggests a way to modify the condition of the $\Diamond\mathcal{C}$-rule originally used to generate new constraint systems. The modified rule, called the $\Diamond_b\mathcal{C}$-rule, is listed in Fig. 5. The $\Diamond_b\mathcal{C}$- and $\Box\mathcal{C}$-rules are jointly called the b-global rules.

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

Fig. 5. The $\Diamond_b\mathcal{C}$-rule

[3] We thank the referee for this observation.

We denote by Λ_{S4} the $S4$-*tableau algorithm* which nondeterministically applies a local, b-global, terminological, or accessibility expansion 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 accessibility expansion rules and the b-global expansion rules.

Theorem 5. *(Soundness of expansion rules) Given an $S4$-structure $\mathbb{M} = \langle S, \pi, \mathcal{E}_1, ..., \mathcal{E}_m\rangle$ and an acyclic TBox $\mathcal{T}$ where $\mathbb{M} \vDash \mathcal{T}$, let $\mathbb{G}$ be a constraint graph, α a local, b-global, terminological, or accessibility 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 $\mathbb{M}_\alpha$ (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 accessibility expansion rules and the $\Diamond_b\mathcal{C}$-rule preserve the existence of $S4$ models. In the other cases, $\mathbb{M}_\alpha$ is a semantic extension of $\mathbb{M}$ (see proof of Theorem 1) 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 $A^4\mathcal{C}$-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 $a : \Box_i C \notin \mathbb{L}(n')$. 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 $\Diamond_b\mathcal{C}$-rule, then 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 $a^{\pi(s)} \in C^{\pi(s)}$. There are two cases. (i) If $a : C \notin \mathbb{L}(n)$, then after applying the $\Diamond_b\mathcal{C}$-rule, a new node n' is added to $\mathbb{G}$ 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'. (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 $a^{\pi(s)} \in C^{\pi(s)}$. ■

Let $\mathbb{G}$ be an open and complete constraint tree resulting from Λ_{S4}. The corresponding canonical Kripke structure $\mathbb{M}_\mathbb{G}$ is a model of $\mathbb{G}$ which however is not an $S4$-structure. To obtain an $S4$-structure for $\mathbb{G}$, we use the following definition.

Definition 6. *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 $i \in N_\mathcal{E}$,*
   $\mathbb{E}_i := \{e \in \mathbb{E} \mid i \in \mathbb{L}(e)\}$,
   $\mathbb{E}_i^*$ is the reflexive and transitive closure of $\mathbb{E}_i$ [4],
   $\mathbb{E}^* := \bigcup_i \mathbb{E}_i^*$;
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.*

The following lemma shows the relationship between $\mathbb{G}$ and $\mathbb{G}^{S4}$.

[4] Note that when $i \neq j$, $\mathbb{E}_i \cap \mathbb{E}_j = \emptyset$. Moreover, since $\mathbb{G}$ is a tree, $\mathbb{E}_i^* \cap \mathbb{E}_j^* = \{(n, n) \mid n \in \mathbb{V}\}$.

Lemma 3. *If a constraint tree $\mathbb{G} = \langle \mathbb{V}, \mathbb{E}, \mathbb{L} \rangle$ is open and complete (w.r.t. local, b-global, terminological and accessibility expansion rules), then so is $\mathbb{G}^{S4} = \langle \mathbb{V}, \mathbb{E}^*, \mathbb{L}^* \rangle$.*

Proof. Suppose that $\mathbb{G}$ is open and complete w.r.t. local, b-global, terminological and accessibility expansion rules. By Definition 6, $\mathbb{G}^{S4}$ is constructed from $\mathbb{G}$ by adding edges and labeling these edges so that $\mathbb{G}^{S4}$ is reflexive and transitive w.r.t. each expert $i \in N_{\mathcal{E}}$. We prove that $\mathbb{G}^{S4}$ is complete by showing that newly added edges do not cause any expansion rule to become applicable, i.e., the constraint graph $\mathbb{G}^{S4}$ is also complete w.r.t. local, b-global, terminological and accessibility expansion rules.

Since all the constraint systems in $\mathbb{G}^{S4}$ remain the same as those in $\mathbb{G}$, if no local, terminological expansion rule, or A^T-rule is applicable in $\mathbb{G}$, no such rule is applicable in $\mathbb{G}^{S4}$ either. Now let us analyze the applicability of the other expansion rules in $\mathbb{G}^{S4}$.

- If an $A^4\mathcal{C}$-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, w.r.t. the $A^4\mathcal{C}$-rule), for every i-descendant n'' of n, $a : \Box_i C \in \mathbb{L}(n'') = \mathbb{L}^*(n'')$. In particular, $a : \Box_i C \in \mathbb{L}^*(n')$, which is a contradiction. Therefore, no $A^4\mathcal{C}$-rule is applicable in $\mathbb{G}^{S4}$.
- If a $\Box\mathcal{C}$-rule is applicable in $\mathbb{G}^{S4}$, then there are two nodes n and n' such that $i \in \mathbb{L}^*(n, n')$, $a : \Box_i C \in \mathbb{L}^*(n)$ and $a : C \notin \mathbb{L}^*(n')$. Because for each $n \in \mathbb{V}$, $\mathbb{L}(n) = \mathbb{L}^*(n)$, we have $a : \Box_i C \in \mathbb{L}(n)$ and $a : C \notin \mathbb{L}(n')$. This means that n' is not an i-successor of n in $\mathbb{G}$. Hence, $(n, n') \in \mathbb{E}^*_i \setminus \mathbb{E}_i$.
  If $n' = n$, then, since $a : \Box_i C \in \mathbb{L}(n)$ and $\mathbb{G}$ is complete, $a : C \in \mathbb{L}(n) = \mathbb{L}^*(n')$ due to A^T-rule. Suppose $n' \neq n$. Since no $A^4\mathcal{C}$-rule is applicable in $\mathbb{G}^{S4}$ (see above), $a : \Box_i C \in \mathbb{L}^*(n')$. Again by A^T-rule, $a : C \in \mathbb{L}^*(n')$. Both cases contradict the assumption $a : C \notin \mathbb{L}^*(n')$. Therefore, no $\Box\mathcal{C}$-rule is applicable.
- If a $\Diamond_b\mathcal{C}$-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' such that $a : C \in \mathbb{L}^*(n') = \mathbb{L}(n')$. This means that $\Diamond_b\mathcal{C}$-rule is applicable in $\mathbb{G}$ as well, which contradicts the assumption that $\mathbb{G}$ is complete. Thus, no $\Diamond_b\mathcal{C}$-rule is applicable in $\mathbb{G}^{S4}$.

Because no expansion rule (local, b-global, terminological or accessibility) is applicable, $\mathbb{G}^{S4}$ is complete. 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 3 does not hold. That is, $\mathbb{G}^{S4} = \langle \mathbb{V}, \mathbb{E}^*, \mathbb{L}^* \rangle$ being open and complete (w.r.t. local, b-global, terminological and accessibility expansion rules) does not imply that $\mathbb{G} = \langle \mathbb{V}, \mathbb{E}, \mathbb{L} \rangle$ is open and complete (w.r.t. local, b-global, terminological and accessibility expansion 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 $\mathbb{G}^{S4}$ by using Definition 4. By Definition 5 and 6, $\mathbb{M}_{\mathbb{G}^{S4}}$ 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 local, b-global, terminological and accessibility expansion 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 4. *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 6. 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 $\mathbb{M}_{\mathbb{G}^{S4}} \Vdash \mathbb{G} \Leftrightarrow \mathbb{M}_{\mathbb{G}^{S4}} \Vdash \mathbb{G}^{S4}$.*

Proof. ($\Leftarrow$) Suppose $\mathbb{M}_{\mathbb{G}^{S4}} \Vdash \mathbb{G}^{S4}$ via σ where σ is an identity function (see Definitions 3 and 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, $\mathbb{M}_{\mathbb{G}^{S4}} \Vdash \mathbb{G}^{S4} \Rightarrow \mathbb{M}_{\mathbb{G}^{S4}} \Vdash \mathbb{G}$.

($\Rightarrow$) Suppose $\mathbb{M}_{\mathbb{G}^{S4}} \Vdash \mathbb{G}$ via σ. Since for every $n \in \mathbb{V}$, $\mathbb{L}^*(n) = \mathbb{L}(n)$, the constraints in $\mathbb{G}^{S4}$ are also satisfied in the corresponding states in $\mathbb{M}_{\mathbb{G}^{S4}}$. We need to show that for all $n, n' \in \mathbb{V}, i \in \mathbb{L}^*(n, n') \Rightarrow \mathcal{E}_i(\sigma(n), \sigma(n'))$.

For all the edges $(n, n') \in \mathbb{E}$, since $\mathbb{M}_{\mathbb{G}^{S4}} \Vdash \mathbb{G}$, we have $\mathcal{E}_i(\sigma(n), \sigma(n'))$. For the edges $(n, n') \in \mathbb{E}^* \setminus \mathbb{E}$:

- If $n = n'$, $\mathbb{L}^*(n, n') = N_{\mathcal{E}}$. Since $\mathbb{M}_{\mathbb{G}^{S4}}$ is reflexive, we have for every $i \in N_{\mathcal{E}}$, $\mathcal{E}_i(\sigma(n), \sigma(n))$.
- If $n \neq n'$, suppose $(n, n') \in \mathbb{E}_i^* \setminus \mathbb{E}_i$, $i \in N_{\mathcal{E}}$. Since $\mathbb{M}_{\mathbb{G}^{S4}}$ is transitive, we have $\mathcal{E}_i(\sigma(n), \sigma(n'))$.

Therefore, $\mathbb{M}_{\mathbb{G}^{S4}} \Vdash \mathbb{G} \Rightarrow \mathbb{M}_{\mathbb{G}^{S4}} \Vdash \mathbb{G}^{S4}$. ■

Theorem 6. *(Soundness and completeness of Λ_{S4}) Let $\mathcal{T}$ be a simple acyclic TBox and $\mathbb{G} = \langle \mathbb{V}, \mathbb{E}, \mathbb{L} \rangle$ be a complete constraint tree (w.r.t. local, b-global, terminological and accessibility expansion rules). 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. local, b-global, terminological and accessibility expansion rules. Let $\mathbb{G}^{S4}$ be the constraint graph constructed from $\mathbb{G}$ by Definition 6. Then by Lemma 3, $\mathbb{G}^{S4}$ is open and complete w.r.t. local, b-global, terminological and accessibility expansion rules, and hence, by Theorem 2, $\mathbb{M}_{\mathbb{G}^{S4}} \Vdash \mathbb{G}^{S4}$ and $\mathbb{M}_{\mathbb{G}^{S4}} \vDash \mathcal{T}$. By Lemma 4, $\mathbb{M}_{\mathbb{G}^{S4}} \Vdash \mathbb{G}$.

($\Leftarrow$) Suppose 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 $\{a : \bot\} \subseteq \mathbb{L}(n)$. Let $\mathbb{M}$ be a Kripke structure such that $\mathbb{M} \Vdash \mathbb{G}$ via σ. 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. It follows that there does not exist a Kripke structure $\mathbb{M}$ such that $\mathbb{M} \Vdash \mathbb{G}$. ■

Based on the $S4$-tableau algorithm Λ_{S4}, a PSPACE implementation $\mathcal{ALCK}_m$-$S4$-SAT for $\mathcal{ALCK}_m$ $S4$-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 $\mathcal{ALCK}_m$-$S4$-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:

- In Line 1 and 11, $S4$-SAT tests for the applicability of the $A^4\mathcal{C}$-rule in addition to the other rules in SAT.
- In Line 8, $S4$-SAT chooses constraints of the form $a : \Diamond_i C \in \mathbb{L}(n)$ only under the condition that $a : C \notin \mathbb{L}(n)$ whereas SAT chooses constraints of the form $a : \Diamond_i C \in \mathbb{L}(n)$ without any restriction.
- In Line 22, $S4$-SAT tests for the applicability of the A^T-rule in addition to the $\Box\mathcal{C}$-rule in SAT.

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

7 Summary and Discussion

In this paper we studied $\mathcal{ALCK}_m$, a knowledge representation language obtained by augmenting $\mathcal{ALC}$ with modal operators of the basic multi-modal logics $\mathbf{K}_m$ and $\mathbf{S4}_m$. The resulting logic allows 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 $\mathcal{ALCK}_m$ queries w.r.t. an $\mathcal{ALCK}_m$ knowledge base with an acyclic TBox.

Algorithm 2 $\mathcal{ALCK}_m$-$S4$-SAT

$\mathcal{ALCK}_m$-$S4$-SAT$(\Sigma, C(a))$:= $S4$-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 $\mathcal{A} \cup \{C(a)\}$.

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

while a local, terminological or A^T 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. $(a, b) : R, b : C \in \mathbb{L}(n)\}$
$\mathbf{D}(n) := \{a : \Diamond_i C \mid a : \Diamond_i C \in \mathbb{L}(n)$ and $a : C \notin \mathbb{L}(n)\}$
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, terminological or A^T 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 $\Box\mathcal{C}$- or $A^4\mathcal{C}$-rule is applicable to $\mathbb{L}(n)$ **do**
        apply the rule to $\mathbb{L}(n)$, 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"

Instead of the general concept inclusions allowed in $\mathbf{K}_{\mathcal{ALC}}$ [16] which lead to a NEXPTIME algorithm for satisfiability, TBoxes in $\mathcal{ALCK}_m$ are restricted to be acyclic. This restriction is critical to achieving the PSPACE implementations for both algorithms. Furthermore, we have introduced expansion rules that have the following features:

- The expansion rules are somewhat 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 [26], 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 4).
- In the case of Λ_{S4}, the expansion rules are designed to syntactically incorporate the properties of $S4$-structures. Therefore, when the tableau algorithm Λ_{S4} terminates, all the constraint systems in the resulting constraint tree are sufficient to detect clashes (see Definition 6 and Lemma 3). 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 implementation.

Our PSpace result for the satisfiability of $\mathcal{ALCK}_m$ extends the result of Hladik and Peñaloza [27] for the satisfiability of $\mathcal{ALC}$ concepts w.r.t. acyclic TBoxes. Baader et al. [33] have recently extended the PSpace result of [27] 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 for their insightful comments which helped us to significantly improve the paper. We also thank George Voutsadakis for helpful discussions on the topics of this paper and Moshe Vardi for encouraging us to incorporate epistemic operators into Description Logic knowledge bases. Vasant Honavar and Giora Slutzki were supported in part by grant #0639230 from 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.

References

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

9. John-Jules Ch. Meyer and Wiebe Van Der Hoek. *Epistemic Logic for AI and Computer Science*. Cambridge University Press, 2004.
10. 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.
11. 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.
12. 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. [34].
13. Francesco M. Donini, Maurizio Lenzerini, Daniele Nardi, Andrea Schaerf, and Werner Nutt. Adding epistemic operators to concept languages. In *KR*, pages 342–353, 1992.
14. 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.
15. Franz Baader and Armin Laux. Terminological logics with modal operators. In *IJCAI (1)*, pages 808–815, 1995.
16. Carsten Lutz, Holger Sturm, Frank Wolter, and Michael Zakharyaschev. A tableau decision algorithm for modalized alc with constant domains. *Studia Logica*, 72(2):199–232, 2002.
17. Franz Baader and Hans Jürgen Ohlbach. A multi-dimensional terminological knowledge representation language. *Journal of Applied Non-Classical Logics*, 5(2), 1995.
18. 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.
19. Frank Wolter and Michael Zakharyaschev. Multi-dimensional description logics. In Thomas Dean, editor, *IJCAI*, pages 104–109. Morgan Kaufmann, 1999.
20. Frank Wolter and Michael Zakharyaschev. Satisfiability problem in description logics with modal operators. In *KR*, pages 512–523, 1998.
21. 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.
22. Frank Wolter and Michael Zakharyaschev. Modal description logics: Modalizing roles. *Fundam. Inform.*, 39(4):411–438, 1999.
23. Frank Wolter and Michael Zakharyaschev. Decidable fragments of first-order modal logics. *J. Symb. Log.*, 66(3):1415–1438, 2001.
24. 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.
25. Stephan Tobies. Complexity results and practical algorithms for logics in knowledge representation. *CoRR*, cs.LO/0106031, 2001.
26. Martin Buchheit, Francesco M. Donini, and Andrea Schaerf. Decidable reasoning in terminological knowledge representation systems. *J. Artif. Intell. Res. (JAIR)*, 1:109–138, 1993.
27. Jan Hladik and Rafael Peñaloza. Pspace automata for description logics. In Parsia et al. [34].
28. Franz Baader and Werner Nutt. Basic description logics. In Baader et al. [1], pages 43–95.
29. 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.
30. 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.
31. Bernhard Nebel. Terminological reasoning is inherently intractable. *Artif. Intell.*, 43(2):235–249, 1990.
32. 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.
33. Franz Baader, Jan Hladik, and Rafael Peñaloza. Automata can show pspace results for description logics. *Inf. Comput.*, 206(9-10):1045–1056, 2008.

34. 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.

A Proof of Theorem 1

Theorem 1 *(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, $\mathbb{M}_\alpha \Vdash \mathbb{G}$.*

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 $(\mathbb{M}, \sigma(n)) \models C_1 \sqcap C_2(a)$. 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, $(\mathbb{M}, \sigma(n)) \models C_1(a)$ and $(\mathbb{M}, \sigma(n)) \models C_2(a)$. 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, $(\mathbb{M}, \sigma(n)) \models C_1 \sqcup C_2(a)$ 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 $(\mathbb{M}, \sigma(n)) \models \exists R.C(a)$ (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 $(\mathbb{M}, \sigma(n)) \models \forall R.C(a)$, 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 $(\mathbb{M}, \sigma(n)) \models R(a, b)$, 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 $(\mathbb{M}, \sigma(n)) \models \Diamond_i C(a)$ 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 $(\mathbb{M}, \sigma(n)) \models \Box_i C(a)$ which means that for all s with $(\sigma(n), s) \in \mathcal{E}_i$, $(\mathbb{M}, s) \models C(a)$. It follows that $(\mathbb{M}, \sigma(n')) \models C(a)$. 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 $\mathbb{M} \models \mathcal{T}$, $a^{\pi(\sigma(n))} \in A^{\pi(\sigma(n))} = D^{\pi(\sigma(n))}$. Therefore, $(\mathbb{M}, \sigma(n)) \models D(a)$ 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 $\{a:\neg A,\ a:B\}\not\subseteq\mathbb{L}(n)$. Since $\mathbb{M}\Vdash\mathbb{G}$ and $\mathbb{M}\models\mathcal{T}$, we have $(\mathbb{M},\sigma(n))\models A\doteq\neg B$ 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 $\mathbb{M}\models\mathcal{T}$, we have $(\mathbb{M},\sigma(n))\models\neg A(a)$, $(\mathbb{M},\sigma(n))\models A\doteq B_1\sqcap B_2$ 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 $\mathbb{M}\models\mathcal{T}$, we have $(\mathbb{M},\sigma(n))\models\neg A(a)$, $(\mathbb{M},\sigma(n))\models A\doteq B_1\sqcup B_2$ 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 $\mathbb{M}\models\mathcal{T}$, we have $(\mathbb{M},\sigma(n))\models\neg A(a)$, $(\mathbb{M},\sigma(n))\models A\doteq\exists R.B$ 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 $\mathbb{M}\models\mathcal{T}$, we have $(\mathbb{M},\sigma(n))\models\neg A(a)$, $(\mathbb{M},\sigma(n))\models A\doteq\forall R.B$ 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 $\mathbb{M}\models\mathcal{T}$, we have $(\mathbb{M},\sigma(n))\models A\doteq\Diamond_i B$, $(\mathbb{M},\sigma(n))\models\neg A(a)$ and therefore $a^{\pi(\sigma(n))}\notin A^{\pi(\sigma(n))}\Leftrightarrow a^{\pi(\sigma(n))}\notin(\Diamond_i B)^{\pi(\sigma(n))}\Leftrightarrow a^{\pi(\sigma(n))}\in\Delta\backslash(\Diamond_i B)^{\pi(\sigma(n))}$ where $\Delta\backslash(\Diamond_i B)^{\pi(\sigma(n))}=\Delta\backslash\bigcup_{t\in\mathcal{E}_i(\sigma(n))}B^{\pi(t)}=\bigcap_{t\in\mathcal{E}_i(\sigma(n))}(\Delta\backslash B^{\pi(t)})=\bigcap_{t\in\mathcal{E}_i(\sigma(n))}(\neg B)^{\pi(t)}=(\Box_i\neg B)^{\pi(\sigma(n))}$. 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 $\mathbb{M}\models\mathcal{T}$, we have $(\mathbb{M},\sigma(n))\models A\doteq\Box_i B$, $(\mathbb{M},\sigma(n))\models\neg A(a)$ 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 $\mathbb{G}$. ■

B Proof of Lemma 2

Lemma 2 *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 $a\in\Delta$, $a:\neg A\in\mathbb{L}(n)\Rightarrow(\mathbb{M}_\mathbb{G},n)\models\neg A(a)$.*

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 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 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. ■

C 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 4.

Example 4. Consider a set of expansion rules that contains (i) local and global expansion rules as given in Fig. 1, and (ii) terminological expansion rules that contain the T-, N$\Diamond$-, and N$\Box$-rules as given in Fig. 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) 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 N$\sqcup$-rule in Fig. 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. ■