Institute of Informatics, Warsaw University

Results on Modal Reasoning with Applications to Modal Deductive Databases

Nguyen Anh Linh

Doctoral Thesis
Written under the supervision of Professor Andrzej Szałas
Warsaw October, 1999

This work is partially supported by
Institute of Informatics, Warsaw University, BW-1416/2

To My Parents

Contents

Chapter 1

Introduction

Deductive databases and logic programming are fields that have gained a great deal of attention due to their usefulness and a wide range of applications. These fields are closely related to each other, they both use classical logic rules to represent mutual dependencies between relations. While logic programming is mostly used in artificial intelligence systems, deductive databases are systems in which the amount of data is much greater than the number of rules. Primarily, logic rules in such systems are restricted to the positive form, i.e. to universally quantified Horn clauses of the classical first-order logic. It is known that with such restriction the least model for a program (or a database) is computable in polynomial time and its size is bounded by a polynomial.

In the classical propositional and first-order logics we are not able, or hardly able, to reason about necessity, belief, knowledge, ... To deal with such modalities it is natural to use modal logics. In modal logics we have two additional unary modal connectives $\Box$ and $\Diamond$, which may have different meanings in different logics, for example, $\Box$ can stand for "necessity", and $\Diamond$ for "possibility".

When trying to extend the fields of deductive databases and logic programming to deal with modal logics, a natural attempt is to consider the modal Horn fragment. Unfortunately, it has been shown by Chen and Lin [4] that satisfiability of a set of modal Horn clauses in the modal logics K, KD, T, KB, KDB, B, $K4$, $KD4$, and $S4$ is PSPACE-complete. Due to this fact, modal logic programming has not gained much interest from researchers. Moreover, the modal Horn fragment has been only, as known to us, studied in the context of top-down methods, which are suitable for logic programming but not deductive databases. With bottom-up methods, we often deal with constructing the least models for positive programs. Having the least model of a positive program, all queries for that program can be effectively answered.

The main aim of this work is to select fragments of modal logics that are solvable in PTIME, in the bottom-up manner. Basing on the definition of modal Horn clauses, we define what is a positive modal logic program, in the way that every set of modal Horn clauses can be treated as a positive modal logic program plus a (positive) query.

One of the main keys to our results is an order between Kripke models with the property that if a model M is less than or equal to a model N then for any positive modal formula ϕ, if $M \models \phi$ then $N \models \phi$. In this work, we give algorithms that for a given positive propositional modal logic program P in a serial modal logic L which is one of KD, T, KDB, B, $KD5$, $KD45$, $S5$, $KD4$, and $S4$ will construct the least L-model of P. If L is one of the modal logics $KD5$, $KD45$, and $S5$, then the least L-model can be constructed in PTIME, and its size is bounded by a polynomial in the size of P. For L being KD or T, the preceding claim remains true when the modal depth of P is finitely bounded. If L is KDB or B, and the modal depth of P is finitely bounded, then the least L-model of P can be constructed in PTIME, and treated as a finite model with size bounded by a polynomial. We also prove that each positive propositional modal logic program in any of the almost serial modal logics KB, $K5$, $K45$, or $KB5$ can be characterized by one or two models, which can be effectively constructed (in PTIME for the logics $K5$, $K45$, and $KB5$).

As a modal logic query language, we give one, called *MDatalog*. This language is defined in fixed-domain first-order modal logics over signatures without functions. A rule in a MDatalog program is a universally quantified modal Horn clause. A MDatalog program can be treated as the set of all of propositional instances of its rules. This gives a straight method to estimate complexity of constructing the least models for MDatalog programs. However, some other strategies discussed in this work will result in more efficient algorithms.

We also include in this work our results about clausal tableau systems and space bounds for the propositional modal logics K, KD, T, KB, KDB, B, $K4$, $KD4$, and $S4$. For each of our systems there is a measure such that for each tableau rule of the system the measure of all its denominators is smaller than the measure of its numerator. This property allows depth-first search without loop checking. Analysis of the systems allows us to reduce space requirement for the logics $K4$, $KD4$, and $S4$ from the previously established bound $O(n^2.\log n)$ to $O(n.\log n)$, and to establish a space bound $O(n^2)$ for the logics T, KB, KDB, and B.

This work is a study on modal reasoning, we give improved decision procedures for propositional modal logics, select and study fragments of modal logics that are computable in polynomial time. Our results have not only theoretical values, but they are also useful in applications to modal logic programming and modal deductive databases.

Summing up, our main contributions are:

- clausal tableau systems and improved decision procedures for the propositional modal logics K, KD, T, KB, KDB, B, $K4$, $KD4$, and $S4$;

- establishment of the best currently known upper space bound $O(n.\log n)$ for the propositional modal logics $K4$, $KD4$, and $S4$, and $O(n^2)$ for KB, KDB, and B;

- a new proof for the results of other authors [15, 1] such that the propositional modal logics K and KD are decidable in $O(n.\log n)$-space;

- a new method of estimating upper space bounds for propositional modal logics;

- a formulation of positive modal logic programs and an order between Kripke models such that if a model M is less than or equal to a model N then for any positive modal formula ϕ, if $M \vDash \phi$ then $N \vDash \phi$;

- algorithms of constructing the least models for positive modal logic programs in the modal logics KD, T, KDB, B, $KD5$, $KD45$, $S5$, $KD4$, and $S4$, and conditions that guarantee the algorithms terminating in polynomial time;

- a characterization of positive modal logic programs in KB, $K5$, $K45$, and $KB5$ by one or two models, which can be effectively constructed;

- a theorem such that the problem of checking satisfiability of a set of Horn formulae with finitely bounded modal depth in the propositional modal logics KD, T, KB, KDB, and B is decidable in PTIME;

- a new proof for the results of other authors [7, 4] such that the problem of checking satisfiability of a set of Horn formulae in the propositional modal logics $K5$, $KD5$, $K45$, $KD45$, $KB5$, and $S5$ is decidable in PTIME;

- a method of constructing the least models for positive modal logic programs.

This thesis is organized as follows:

Chapter 2 contains preliminaries about propositional modal logics, Horn clauses, and positive modal logic programs. In this chapter, we also define syntax, semantics, soundness and completeness of clausal tableau systems for propositional modal logics.

Chapter 3 presents clausal tableau systems for the propositional modal logics K, KD, T, KB, KDB, B, $K4$, $KD4$, and $S4$, and proofs of soundness and completeness of the systems. Estimation of space requirement for the logics, and comparison with other work on this subject are given in this chapter.

Chapter 4 deals with positive propositional modal logic programs. In the first section of this chapter, we define a pre-order between propositional Kripke models. In the next sections, we give algorithms of constructing the least models for positive propositional modal logic programs, proofs of their correctness, and analysis of computational complexity. Note that this chapter is independent of Chapter 3, and Sections 4.2 and 4.4 are independent of each other.

Chapter 5 deals with MDatalog programs. Results about existence of the least models for MDatalog programs, as well as computational complexity of constructing them are presented in this chapter. We also discuss how to implement and optimize the algorithms.

Chapter 6 contains conclusions and information about further work.

Chapter 2

Preliminaries

2.1 Syntax and Semantics of Propositional Modal Logics

The sentences of modal logics are built from primitive propositions p_1, p_2, $\ldots$, classical connectives $\wedge$, $\vee$, $\neg$, $\rightarrow$, nonclassical unary modal connectives $\Box$, $\Diamond$, and parentheses), (.

A modal formula, hereafter simply called a *formula*, is any finite sequence of these symbols obtained by applying the following rules: any proposition p_i is a formula, and if ϕ and ψ are formulae then so are $\neg\phi$, $\phi \wedge \psi$, $\phi \vee \psi$, $\phi \rightarrow \psi$, $\Box\phi$, and $\Diamond\phi$.

The symbols $\neg$, $\wedge$, $\vee$ and $\rightarrow$, respectively, stand for logical negation, logical conjunction, logical disjunction and logical (material) implication. The symbols $\Box$ and $\Diamond$ can take various meanings but traditionally stand for "necessity" and "possibility". To enable us to omit parentheses, we adopt the convention that the connectives $\neg$, $\Box$, $\Diamond$ are of equal binding strength but bind stronger than $\wedge$ which binds stronger than $\vee$ which binds stronger than $\rightarrow$.

We use letters p and q to denote primitive propositions, and Greek letters ϕ, ψ, ζ to denote formulae. We denote the set of primitive propositions by $\mathcal{P}$, and the set of formulae by $\mathcal{F}$.

Definition 2.1.1 A *Kripke frame* is a triple $\langle W, \tau, R\rangle$, where W is a nonempty set (of possible worlds), $\tau \in W$ is the actual world (also called the current world), and R is a binary relation on W, called the accessibility relation. If $R(w, u)$ then we say that the world u is accessible from the world w, or that u is reachable from w.

Definition 2.1.2 A *Kripke model* is a tuple $\langle W, \tau, R, h\rangle$, where $\langle W, \tau, R\rangle$ is a Kripke frame, and $h : W \rightarrow P(\mathcal{P})$, $h(w)$ is the set of primitive propositions that are "true" at the world w.

Definition 2.1.3 A *model graph* is a tuple $\langle W, \tau, R, H\rangle$, where $\langle W, \tau, R\rangle$ is a Kripke frame, and $H : W \to P(\mathcal{F})$, $H(w)$ is the set of formulae which should be "true" at the world w. We sometimes treat model graphs as models with h being H restricted to the set of primitive propositions.

Definition 2.1.4 Given some Kripke model $M = \langle W, \tau, R, h\rangle$, some world $w \in W$, define the *satisfiability relation* $\models$ as follows:

$M, w \models p$	iff	$p \in h(w)$;
$M, w \models \neg\phi$	iff	$M, w \not\models \phi$;
$M, w \models \phi \wedge \psi$	iff	$M, w \models \phi$ and $M, w \models \psi$;
$M, w \models \phi \vee \psi$	iff	$M, w \models \phi$ or $M, w \models \psi$;
$M, w \models \phi \to \psi$	iff	$M, w \not\models \phi$ or $M, w \models \psi$;
$M, w \models \Box\phi$	iff	for all $v \in W$ such that $R(w, v)$, $M, v \models \phi$;
$M, w \models \Diamond\phi$	iff	there exists $v \in W$ such that $R(w, v)$ and $M, v \models \phi$.

We say that M *satisfies* ϕ *at* w iff $M, w \models \phi$. We say that M *satisfies* ϕ, or ϕ *is satisfied in* M, and write $M \models \phi$, iff $M, \tau \models \phi$.

Definition 2.1.5 We define the *size* of a Kripke model $M = \langle W, \tau, R, h\rangle$ (resp. a model graph $M = \langle W, \tau, R, H\rangle$) to be the sum of the number of its worlds, the size of its accessibility relation, and the total number of primitive propositions (resp. formulae) from its worlds, i.e. $|W| + |R| + \Sigma_{w\in W}|h(w)|$ (resp. $|W| + |R| + \Sigma_{w\in W}|H(w)|$).

Lemma 2.1.6 *Given a model M and a formula ϕ, the problem of checking whether $M \models \phi$ is solvable in polynomial time.*

Proof. Let $f_M(\phi)$ be the cost function of computing the set U of worlds such that $M, u \models \phi$ iff $u \in U$. We estimate f_M by induction on the construction of ϕ. Let n be the size of M. We have:

$$\begin{aligned}
f_M(p) &\leq O(n) \\
f_M(\neg\phi) &\leq f_M(\phi) + O(n) \\
f_M(\phi \wedge \psi) &\leq f_M(\phi) + f_M(\psi) + O(n) \\
f_M(\phi \vee \psi) &\leq f_M(\phi) + f_M(\psi) + O(n) \\
f_M(\Box\phi) &\leq f_M(\phi) + O(n) \\
f_M(\Diamond\phi) &\leq f_M(\phi) + O(n)
\end{aligned}$$

We conclude that $f_M(\phi) = O(n * |\phi|)$. □

Definition 2.1.7 For a Kripke model $M = \langle W, \tau, R, h\rangle$ (or a model graph $M = \langle W, \tau, R, H\rangle$), for $w \in W$, $w \neq \tau$, we define the *depth* of w to be the least number k such that $R^k(\tau, w)$. The depth of τ is assumed to be 0. The depth of a world that is not directly or indirectly reachable from τ is undefined.

Definition 2.1.8 For a Kripke model $M = \langle W, \tau, R, h \rangle$ (or a model graph $M = \langle W, \tau, R, H \rangle$), we define the *real diameter* of M to be the maximal depth of some non-empty world in M (i.e. a world w with $h(w)$ (resp. $H(w)$) not empty). If there is no upper bound on depths, the real diameter is infinite.

Definition 2.1.9 Let M be a Kripke model (resp. a model graph). For $k \geq 0$, we define $M|_k$ to be the model (resp. model graph) obtained from M by restricting it to the worlds with depth not greater then k.

Definition 2.1.10 We define the *modal depth* of a formula, denoted by $mdepth(\_)$, as follows:

$$\begin{aligned}
&mdepth(p) = 0, \quad mdepth(\neg\phi) = mdepth(\phi),\\
&mdepth(\phi \wedge \psi) = mdepth(\phi \vee \psi) =\\
&mdepth(\phi \rightarrow \psi) = max(mdepth(\phi), mdepth(\psi)),\\
&mdepth(\Box\phi) = mdepth(\Diamond\phi) = mdepth(\phi) + 1.
\end{aligned}$$

Lemma 2.1.11 *Let M be a Kripke model. A positive formula ϕ with modal depth m is satisfied in M iff it is satisfied in $M|_m$.*

This lemma can be easily proved by induction on the construction of ϕ.

Notation If R is a binary relation then we use R^* to denote its transitive closure.

2.2 Modal Logic Correspondences

The simplest normal modal logic (called K) is axiomatized by the standard axioms for the classical propositional logic and the *modus ponens* inference rule together with the axiom schema $\Box(\phi \rightarrow \psi) \rightarrow (\Box\phi \rightarrow \Box\psi)$, which is usually called the *K-axiom*, plus the additional *necessitation rule*

$$\frac{\vdash \phi}{\vdash \Box\phi}$$

It can be shown that a modal logic formula is provable in this axiomatization iff it is satisfied in every Kripke model (i.e. without any special R-properties) [17]. It is known that certain axiom schemata added to this axiomatization correspond to certain properties of the accessibility relation (see also [16, 3]).

Many of such correspondences are definable as formulae of first-order logic where the binary predicate $R(x, y)$ represents the accessibility relation, as shown in Table 2.1. Different modal logics are distinguished by their respective additional axiom schemata. Some of the most popular modal logics together with their axiom schemata are listed in Table 2.2. We refer to properties of the accessibility relation of a modal logic L as *L-frame axioms* or *L-frame restrictions*.

Axiom	Schemata	First-Order Formula
D	$\Box\Phi \rightarrow \Diamond\Phi$	$\forall w\ \exists u\ R(w,u)$
T	$\Box\Phi \rightarrow \Phi$	$\forall w\ R(w,w)$
B	$\Phi \rightarrow \Box\Diamond\Phi$	$\forall w,u\ R(w,u) \rightarrow R(u,w)$
4	$\Box\Phi \rightarrow \Box\Box\Phi$	$\forall w,u,v\ R(w,u) \wedge R(u,v) \rightarrow R(w,v)$
5	$\Diamond\Phi \rightarrow \Box\Diamond\Phi$	$\forall w,u,v\ R(w,u) \wedge R(w,v) \rightarrow R(u,v)$

Table 2.1: Axioms and corresponding conditions on R

Definition 2.2.1 We call a model M a *L-model* if the accessibility relation of M satisfies all L-frame restrictions. We say that ϕ is L-satisfiable if there exists a L-model of ϕ, i.e. a L-model satisfying ϕ. A formula ϕ is a tautology in a logic L if ϕ is satisfied in every L-model. We write $\phi \vDash_L \psi$ to denote that ψ is satisfied in every L-model of ϕ.

We call modal logics that are characterized by classes of Kripke models *normal modal logics*. Given two normal modal logics L and L', we say that L' is a *normal extension* of L, and write $L \leq L'$, if all L-frame restrictions are also L'-frame restrictions.

Let L be one of the modal logics listed in Table 2.2. For a binary relation R', we use $Ext_L(R')$ to denote the least extension of R' that satisfies all L-frame axioms, excluding the axiom D. It is clear that this operator is well defined for such logics.

2.3 Modal Clauses

We call formulae of the form p or $\neg p$ *classical literals* and use letters a, b, c to denote them. We call formulae of the form a, $\Box a$, or $\Diamond a$ and their negations *atoms* and use letters A, B, C to denote them. A *simple clause* is an atom or a disjunction of atoms. We write $[A_1, \ldots, A_k]$ to denote the simple clause $A_1 \vee \ldots \vee A_k$. If ϕ is a simple clause then we call $\Box\phi$ a *$\Box$-simple clause*, and $\Box^s\phi$, where $s \geq 0$, a *clause*. By a *basic clause* we mean a clause of one of the forms $\Box^s\neg\Box a$, $\Box^s[a_1, \ldots, a_k]$, and $\Box^s[a, B]$, where $s \geq 0$, $k \geq 1$, and B is an atom of the form $\Box b$ or $\neg\Box b$.

In this section we show that any set of formulae can be translated to an *equisatisfiable* (see Definition 2.3.3) set of clauses.

Definition 2.3.1 We define the length of a formula ϕ to be the number of connectives and primitive propositions in ϕ. The length of a clause is the length of the formula it stands for. We define the length of a set of formulae to be the sum of the lengths of its elements. For a clause $\phi = \Box^s\psi$, where ψ is a simple clause, we define the *restrictive length* of ϕ to be the length of

Logic	Axioms	Frame Restriction
K	K	no restriction
KD	KD	serial
T	KT	reflexive
KB	KB	symmetric
KDB	KDB	serial and symmetric
B	KTB	reflexive and symmetric
$K4$	K4	transitive
$KD4$	KD4	serial and transitive
$S4$	KT4	reflexive and transitive
$K5$	K5	euclidean
$KD5$	KD5	serial and euclidean
$K45$	K45	transitive and euclidean
$KD45$	KD45	serial, transitive and euclidean
$KB5$	KB5	symmetric and euclidean
$S5$	KT5	reflexive and euclidean

Table 2.2: Modal logics and frame restriction

the clause $\Box\psi$, and denote it by $rlength(\phi)$. We define the restrictive length of a set of clauses to be the sum of the restrictive lengths of its elements.

Restrictive length can be understood as length in the common sense if sequences of $\Box^s$ are considered to be primitive connectives.

Definition 2.3.2 A formula is in the *negative normal form* if each of its negations occurs immediately before a primitive proposition. A formula is called *negative* if in its negative normal form every primitive proposition is prefixed by negation. A formula is called *non-negative* if it is not negative, and *positive* if its negation is a negative formula.

Given a formula, we can translate it to the negative normal form, which is equivalent in any normal modal logic to the former one, by applying the following rules:

$$\begin{aligned}
\phi \to \psi &\equiv \neg\phi \vee \psi \\
\neg\neg\phi &\equiv \phi \\
\neg(\phi \wedge \psi) &\equiv \neg\phi \vee \neg\psi \\
\neg(\phi \vee \psi) &\equiv \neg\phi \wedge \neg\psi \\
\neg\Box\phi &\equiv \Diamond\neg\phi \\
\neg\Diamond\phi &\equiv \Box\neg\phi
\end{aligned}$$

We write $[\phi_1, \dots, \phi_k]$ to denote the disjunction $\phi_1 \vee \dots \vee \phi_k$, and write $\phi_1; \phi_2; \dots; \phi_k$ to denote the set $\{\phi_1, \phi_2, \dots, \phi_k\}$. If X and Y are sets of formulae then we write $X; Y$ to denote the sum of them. For $X = \{\phi_1, \phi_2, \dots, \phi_k\}$, we write $\Box^s X$ to denote $\Box^s\phi_1; \Box^s\phi_2; \dots; \Box^s\phi_k$. A set of formulae is sometimes understood as the conjunction of its formulae.

Definition 2.3.3 We call two sets of formulae X and Y *equisatisfiable* in a logic L iff (X is L-satisfiable iff Y is L-satisfiable).

Lemma 2.3.4 (Mints [22]) *Let p and q be primitive propositions which occur only at the indicated positions. Then the following pairs of sets of formulae are equisatisfiable in any normal modal logic:*

$$X; \Box^s[\phi, \psi] \quad \textit{and} \quad X; \Box^s[\phi, p]; \Box^s[\neg p, \psi] \tag{2.1}$$
$$X; \Box^s[\phi, \psi \vee \zeta] \quad \textit{and} \quad X; \Box^s[\phi, \psi, \zeta] \tag{2.2}$$
$$X; \Box^s[\phi, \psi \wedge \zeta] \quad \textit{and} \quad X; \Box^s[\phi, \psi]; \Box^s[\phi, \zeta] \tag{2.3}$$
$$X; \Box^s[\phi, \Box\psi] \quad \textit{and} \quad X; \Box^s[\phi, \Box p]; \Box^{s+1}[\neg p, \psi] \tag{2.4}$$
$$X; \Box^s[\phi, \Box\psi] \quad \textit{and} \quad X; \Box^s[\phi, \Box\neg q]; \Box^{s+1}[q, \psi] \tag{2.5}$$
$$X; \Box^s[\phi, \Diamond\psi] \quad \textit{and} \quad X; \Box^s[\phi, \Diamond p]; \Box^{s+1}[\neg p, \psi] \tag{2.6}$$
$$X; \Box^s[\phi, \Diamond\psi] \quad \textit{and} \quad X; \Box^s[\phi, \Diamond\neg q]; \Box^{s+1}[q, \psi] \tag{2.7}$$

Proof. $\rightarrow$) Fix one of the pairs. If the left side set is satisfied in a model $M = \langle W, \tau, R, h\rangle$, then the model $M' = \langle W, \tau, R, h'\rangle$ with $h'(u)(x) = h(u)(x)$ for $x \neq p \wedge x \neq q$, and $h'(u)(p) = h(u)(\psi)$, $h'(u)(q) = \neg h(u)(\psi)$, where p and q are new primitive propositions, satisfies the right side set.

$\leftarrow$) Fix one of the pairs. We show that the right side set of formulae implies the left side one in any normal modal logic. This claim is clear for the pairs (2.1), (2.2), and (2.3). The pairs (2.5) and (2.7) are dual to the pairs (2.4) and (2.6), respectively. The assertion about the pair (2.4) holds because the following formulae are tautologies in any normal modal logic:

$\Box(p \wedge (\neg p \vee \psi)) \rightarrow \Box\psi$,

$(\phi \vee \Box p) \wedge \Box(\neg p \vee \psi) \rightarrow (\phi \vee \Box\psi)$.

The assertion about the pair (2.6) holds since the following formulae are tautologies in any normal modal logic:

$\Diamond p \wedge \Box(\neg p \vee \psi) \rightarrow \Diamond(p \wedge (\neg p \vee \psi))$,

$\Diamond p \wedge \Box(\neg p \vee \psi) \rightarrow \Diamond\psi$,

$(\phi \vee \Diamond p) \wedge \Box(\neg p \vee \psi) \rightarrow (\phi \vee \Diamond\psi)$. □

Given a formula ϕ, by translating it to the negative normal form and applying equivalencies given in the preceding lemma, we can obtain a set X of basic clauses (i.e. clauses of one of the forms $\Box^s\neg\Box a$, $\Box^s[a_1, \dots, a_k]$, and $\Box^s[a, B]$, where $s \geq 0$, $k \geq 1$, and B is an atom of the form $\Box b$ or $\neg\Box b$), such that:

- ϕ and X are equisatisfiable in any normal modal logic.

- The modal depth of X is equal to the modal depth of ϕ and the restrictive length of X is linearly bounded by the length of ϕ.

During the translation we treat sequences of $\Box^s$ as primitive connectives, which allows us to do the task and encode X in $O(n. \log n)$-space.

2.4 Positive Propositional Modal Logic Programs

In this section, we give the definition of positive propositional modal logic programs. At a first sight, the definition seems a bit restrictive, however, we will show that for any set X of so called *Horn formulae* (see Definition 2.4.1) in any normal modal logic L there exists a positive propositional modal logic program P and a query Q (i.e. a positive formula) such that X is L-satisfiable iff $P \not\models_L Q$.

Definition 2.4.1 A formula ϕ is called a *Horn formula* iff one of the following conditions holds:

- ϕ is a primitive proposition
- ϕ is a negative formula
- $\phi = \Box\psi$, or $\phi = \Diamond\psi$, or $\phi = \psi \wedge \zeta$, where ψ and ζ are Horn formulae
- ϕ is a disjunction of a negative formula and a Horn formula.

We call a clause a *Horn clause* if it is a Horn formula.

In [7], Fariñas del Cerro and Penttonen have given a definition of Horn clauses. It is different from our definition of Horn clauses and Horn formulae. Each of our Horn clauses is also a Horn clause in the meaning by Fariñas del Cerro and Penttonen, and each of the later is a Horn formula, but not vice versa. However, these definitions are equivalent in the sense that each Horn formula can be translated to an equisatisfiable set of Horn clauses.

Definition 2.4.2 A *positive propositional modal logic program* is a finite set of rules of the following form:

$$\Box^s(B_1 \wedge \ldots \wedge B_k \to A)$$

where $s \geq 0$, $k \geq 0$ and $A, B_1, \ldots, B_k$ are atoms of one of the forms p, $\Box p$, $\Diamond p$, where p is a primitive proposition. We often write program rules in the form:

$$\Box^s(A \leftarrow B_1, \ldots, B_k).$$

Proposition 2.4.3 *For any set X of Horn formulae there exists a set Y of Horn clauses such that:*

- *X and Y are equisatisfiable in any normal modal logic.*
- *The modal depth of Y is equal to the modal depth of X, and the restrictive length of Y is linearly bounded by the length of X.*

Moreover, if we divide Y into two groups P and Q such that P contains only non-negative clauses and Q contains only negative clauses, then P can be treated as a positive propositional modal logic program, and X is L-satisfiable iff $P \nvDash_L \neg Q$, where L is any normal modal logic. The translation from X to Y is solvable in polynomial time.

Proof. We first translate X to the negative normal form, and then apply the equivalencies (2.2) - (2.7) given in Lemma 2.3.4 (in the left to right direction) to the resulting set. In situations when both of the equivalencies (2.4) and (2.5), resp. both (2.6) and (2.7), are applicable, the appropriate one must be chosen in order to guarantee the resulting set containing only Horn formulae. We apply the equivalencies until the set does not change. This process terminates in polynomial time. Let Y be the resulting set. It is easily seen that Y satisfies the two mentioned assertions.

Let Y be divided into two groups P and Q such that P contains only non-negative clauses and Q contains only negative clauses. Y is L-satisfiable iff $\neg(P \wedge Q)$ is not a tautology in L, and iff $P \nvDash_L \neg Q$. □

2.5 Syntax, Soundness, and Completeness of Clausal Modal Tableau Systems

In Chapter 3, we give efficient decision procedures for common propositional modal logics that base on so called *clausal modal tableau systems.* For this reason, in this section, we define the syntax of such deductive systems.

The most popular tableau formulation is due to Smullyan as expounded by Fitting [8]. Following Hintikka [13], Rautenberg [30] and Goré [11], we use a slightly different formulation where formulae are carried from a tableau node to its child because the direct correspondence between sequent systems and tableau systems is easier to be established using this formulation. Our systems are defined only for sets of basic clauses. Given a formula, we can translate it to an equisatisfiable set of basic clauses to be usable by our systems.

Definition 2.5.1 A *tableau rule R* consists of a numerator N above the line and a (finite) list of denominators $D_1, D_2, \ldots, D_k$ (below the line) separated by vertical bars.

$$\frac{N}{D_1 \mid D_2 \mid \ldots \mid D_k}$$

The numerator is a finite set of basic clauses and so is each denominator. A *tableau system* (or *calculus*) $\mathcal{CL}$ is a finite set of tableau rules.

The terms "numerator" and "denominator" are borrowed from Goré [11]. As we shall see later, each rule is read downwards as "if the numerator is L-satisfiable, then so is one of the denominators". A tableau rule can be accompanied by conditions restricting when the rule can be applied. The numerator of each tableau rule contains one or more distinguished clauses, the most left ones, which are called *principal clauses.*

Definition 2.5.2 A *$\mathcal{CL}$-tableau* is a tree with nodes carrying sets of clauses such that if x is a node carrying X and $y_1, y_2, \ldots y_k$ are all child nodes of x that carry $Y_1, Y_2, \ldots Y_k$ respectively, then there exists a $\mathcal{CL}$-rule R of k denominators such that X is an instance of the numerator of R and Y_i, $1 \le i \le k$, is the corresponding instance of the denominator number i of R, and X satisfies all the conditions necessary to applying R. A *$\mathcal{CL}$-tableau for* X is a $\mathcal{CL}$-tableau with the root carrying X.

Note that in our definition, tableaux can be infinite. We say that a tableau system $\mathcal{CL}$ has the *finite tableau property* if every $\mathcal{CL}$-tableau is finite.

Definition 2.5.3 Let X and Y be sets of basic clauses, and let R be a set of tableau rules. We say that Y *can be obtained from* X *by applying the rules from* R if there exists a tableau for X that only uses rules from R, and Y is its node.

Definition 2.5.4 A branch in a tableau is *closed* if it ends with $\perp$. A tableau is *closed* if all of its branches are closed. A tableau is *open* if it is not closed. A set X of basic clauses is said to be *$\mathcal{CL}$-consistent* if every $\mathcal{CL}$-tableau for X is open. If there is a closed $\mathcal{CL}$-tableau for X then we say that X is *$\mathcal{CL}$-inconsistent.*

The following definitions concern soundness and completeness of clausal tableau systems. Note that such systems are defined only for sets of basic clauses.

Definition 2.5.5 A tableau system $\mathcal{CL}$ is said to be *sound* if for any set X of basic clauses, if X is L-satisfiable then X is $\mathcal{CL}$-consistent. A tableau system $\mathcal{CL}$ is said to be *complete* if for any set X of basic clauses, if X is $\mathcal{CL}$-consistent then X is L-satisfiable.

Definition 2.5.6 Let R be one of the rules of $\mathcal{CL}$. We say that R is sound wrt. L if for any instance R' of R, if the numerator of R' is L-satisfiable then so is one of the denominators of R'.

It is clear that if $\mathcal{CL}$ contains only rules sound wrt. L then $\mathcal{CL}$ is sound.

Chapter 3

Clausal Tableau Systems for Propositional Modal Logics

In the 1970s Ladner has shown that complexity of provability in the modal logics K, T, and $S4$ is PSPACE-complete, and in $S5$ is co-NP-complete [18]. Recently, some authors have analyzed space requirement for modal logics. In [15] Hudelmaier translates formulae to clausal form and proposes contraction-free sequent calculi for the logics K, T and $S4$ that are defined only for sets of clauses and have decreasing measures for rules. Using the measures he has shown that provability for $S4$ is decidable in $O(n^2.\log n)$-space, for K and T in $O(n.\log n)$-space. Using labeled sequent systems, Basin, Matthews and Viganò have given decision procedures for the logics K, KD and T that use $O(n.\log n)$-space, and for the logics $K4$, $KD4$ and $S4$ that use $O(n^2.\log n)$-space [1, 34, 2].

In this chapter we present so called clausal tableau systems for the common modal logics K, KD, T, KB, KDB, B, $K4$, $KD4$, and $S4$. Following Hudelmaier, our systems are defined only for sets of basic clauses (hereafter we will use "clause" instead of "basic clause" for short). This simplifies proofs of completeness and gives a good method to estimate space bounds for modal logics. For each of these systems there is a measure such that for each tableau rule of the system the measure of all its denominators is smaller than the measure of its numerator. This is just a nice property of our systems, which allows depth-first search without loop checking. To estimate space bounds for the modal logics we do not use the measures. Analysis of the systems allows us to establish a lower space bound, $O(n.\log n)$, for the logics $K4$, $KD4$, and $S4$, and a new space bound, $O(n^2)$, for the logics KB, KDB, and B. For the logics K and KD we arrive at the same space bound $O(n.\log n)$ as Hudelmaier [15] and Basin et al [1]. We also show that the problem of checking satisfiability in T, KB, KDB, or B for formulae with finitely bounded modal depth is decidable in $O(n.\log n)$-space. We are the first who explicitly estimates space requirement for the logics KB, KDB, and B. Our space bounds are obtained from analysis of a special model, if any, constructed for an input formula.

Our formulation of tableau systems is based on the work of Goré [11]. The method for proving completeness of tableau systems is slightly different here. Instead of constructing a *saturated* L-model graph for a $\mathcal{C}L$-consistent set X of clauses, we construct a L-model graph (that needs not to be saturated) and prove by induction that the model graph satisfies all its formulae at the corresponding worlds.

3.1 Clausal Tableau Rules and Calculi

Tables 3.1 and 3.2 represent clausal tableau rules, from which calculi listed in Table 3.3 are built. In Table 3.1 and Table 3.2, we assume $s \geq 1$.

The connective $\blacksquare$ used in Table 3.1 has the same semantics as $\Box$, i.e. $M, w \vDash \blacksquare\phi$ iff $M, w \vDash \Box\phi$. The connectives $\blacksquare$ and $\boxdot$ used in Table 3.2 have the same semantics as $(\Box\phi; \phi)$, i.e. $M, w \vDash \blacksquare\phi$ iff $M, w \vDash \boxdot\phi$ iff $M, w \vDash (\Box\phi; \phi)$. We write $\blacksquare^s\phi$ to denote $\blacksquare\Box^{s-1}\phi$. Syntactically, clauses of the form $\blacksquare^s\phi$, where $s \geq 1$, are blocked from being principal clauses. The connective $\blacksquare$ has the same role as the connective $\bigcirc$ introduced by Hudelmaier in [15].

When not dealing with the logics $K4$ and $KD4$, we assume that the language does not contain the connective $\boxdot$. The calculi $\mathcal{C}K4$ and $\mathcal{C}KD4$ are built in a similar way as $\mathcal{C}S4$; the connective $\boxdot$ is introduced to simulate the connective $\Box$ of $S4$.

Following Goré [11], we call some rules static and some rules transitional. A rule is called *static* if it does not change world (i.e. one of the denominators of the rule is satisfied at the same world where the numerator). A rule is called *transitional* if it changes world (i.e. if the numerator of the rule is satisfied at a world w then one of the denominators is satisfied at a world reachable from w).

For intuition of the rules, as an example, let us consider the rule (T). This is a static rule. The formula ϕ is obtained from $\Box^s\phi$ by s times applying the axiom $\Box\Phi \to \Phi$. The rule takes advantage of a one-step treatment of iterated modality (i.e. $\Box^s$). The intermediate clauses $\Box^{s-1}\phi, \ldots, \Box^2\phi$ are not necessary as they do not contain as much information as $\Box^s\phi$, and they cannot be directly used to introduce $\bot$ as ϕ. Hence these clauses are not present in the denominator, and we restrict ϕ to be a simple clause. Once the clause $\Box^s\phi$ has been used at a world w to introduce ϕ, it cannot bring anything else to w in order to introduce $\bot$, hence we block using it at w by replacing it by $\blacksquare^s\phi$. Clauses of the form $\blacksquare^s\phi$ are unblocked in applications of transitional rules. In the calculus $\mathcal{C}B$, clauses of the form $\Box^s[a, \Box b]$ are solved by the rule (KTB), so we replace (T) by (T'). For specific intuition of other rules, we refer readers to the proof of Lemma 3.2.1 given below.

We use $\boxtimes$ to denote either $\Box$ or $\blacksquare$, for instance we write $\boxtimes^s\phi \in X$ to mean $(\Box^s\phi \in X)$ or $(\blacksquare^s\phi \in X)$. Similarly, we use $\boxplus$ to denote either $\boxdot$ or $\blacksquare$. We define the relations $\in_\boxtimes$ and $\in_\boxplus$ as follows:

$$\phi \in_\boxtimes X \equiv \begin{cases} \phi \in X & \text{if } \phi \text{ is not of the form } \boxtimes\psi \\ \boxtimes\psi \in X & \text{if } \phi = \Box\psi \text{ or } \phi = \blacksquare\psi \end{cases}$$

$$\phi \in_\boxplus X \equiv \begin{cases} \phi \in X & \text{if } \phi \text{ is not of the form } \boxplus\psi \\ \boxplus\psi \in X & \text{if } \phi = \boxdot\psi \text{ or } \phi = \blacksquare\psi \end{cases}$$

For example, $\Box\phi \in_\boxtimes \{\blacksquare\phi\}$.

3.2 Soundness

Lemma 3.2.1 *Each rule in Tables 3.1 and 3.2 is sound wrt. any normal modal logic extending the axioms listed in the rule's name.*

Proof. Fix a rule R from Table 3.1 or Table 3.2. Let L be a normal modal logic extending the axioms listed in the name of R. Let R' be an instance of R. We show that if the numerator of R' is L-satisfiable then so is one of the denominators of R'.

This assertion obviously holds for $(\vee)$ and $(\bot)$.

For (K), let M be a Kripke model and let $M, w \vDash X; \blacksquare Y; \Box Z; \neg\Box a$. There exists a world u reachable from w such that $M, u \vDash \neg a$. It is easily seen that $M, u \vDash Y; Z; \neg a$.

For (KD), let M be a KD-model and let $M, w \vDash X; \blacksquare Y; \Box Z$. Let u be a world reachable from w. It is easily seen that $M, u \vDash Y; Z$.

For (T) and (T'), let M be a T-model and let $M, w \vDash X; \Box^s\phi$. It is easily seen that $M, w \vDash X; \blacksquare^s\phi; \phi$.

For (KB), let M be a KB-model and let $M, w \vDash X; \Box^{2s}\phi; \neg\Box a$. There exists a world u reachable from w. We have $M, u \vDash \Box^{2s-1}\phi$, hence $M, w \vDash \Box^{2s-2}\phi$. Repeat the argument until we obtain $M, w \vDash \phi$. Therefore $M, w \vDash X; \blacksquare^{2s}\phi; \phi; \neg\Box a$.

Similarly, the assertion holds for (KDB).

For (KB_b), let M be a KB-model and let $M, w \vDash X; \Box[a, \Box b]$. Suppose that $M, w \nvDash \Box a$, we show that $M, w \vDash b$. There exists a world u reachable from w such that $M, u \vDash \neg a$. Thus $M, u \vDash \Box b$, hence $M, w \vDash b$.

For (KTB), let M be a B-model and let $M, w \vDash X; \Box^s[a, \Box b]$. We have $M, w \vDash \Box[a, \Box b]; [a, \Box b]$. Reasoning similarly as for (KB_b) we derive $M, w \vDash [\Box a, b]$. Therefore one of the following conditions must hold: $(M, w \vDash \Box a)$, $(M, w \vDash \Box b)$, $(M, w \vDash a; b)$.

For $(K4)$, let M be a $K4$-model and let $M, w \vDash X; \Box Y; \neg\Box a$. There exists a world u reachable from w such that $M, u \vDash \neg a$. It is easily seen that $M, u \vDash \Box Y; \neg a$.

$$(\vee)\ \frac{X; [A_1, \ldots, A_k]}{X; A_1 \mid \ldots \mid X; A_k} \qquad (\bot)\ \frac{X; a; \neg a}{\bot}$$

$$(K)\ \frac{X; \blacksquare Y; \Box Z; \neg\Box a}{Y; Z; \neg a} \qquad (KD)\ \frac{X; \blacksquare Y; \Box Z}{Y; Z}$$

$$(T)\ \frac{X; \Box^s \phi}{X; \blacksquare^s \phi; \phi} \quad \text{where } \phi \text{ is a simple clause}$$

$$(T')\ \frac{X; \Box^s \phi}{X; \blacksquare^s \phi; \phi} \quad \text{where } \phi \text{ is a simple clause not of the form } [a, \Box b]$$

$$(KB)\ \frac{X; \neg\Box a; \Box^{2s} \phi}{X; \neg\Box a; \blacksquare^{2s} \phi; \phi} \qquad (KDB)\ \frac{X; \Box^{2s} \phi}{X; \blacksquare^{2s} \phi; \phi}$$

where ϕ is either a simple clause or a $\Box$-simple clause

$$(KB_b) \frac{X; \Box[a, \Box b]}{X; \blacksquare\phi; \Box a \mid X; \blacksquare\phi; b} \qquad (KTB) \frac{X; \Box^s[a, \Box b]}{X; \blacksquare^s \phi; \Box a | X; \blacksquare^s \phi; \Box b | X; \blacksquare^s \phi; a; b}$$

where $\phi = [a, \Box b]$

$$(K4)\ \frac{X; \Box Y; \neg\Box a}{\Box Y; \neg a} \qquad (T_r)\ \frac{X; \Box^s \phi}{X; \blacksquare^s \phi; \phi} \quad \text{where } \phi = [a_1, \ldots, a_k]$$

$$(S4_a)\ \frac{X; \blacksquare Y; \Box^s \phi}{X; \Box Y; \Box b \mid X; \blacksquare Y; \blacksquare^s \phi; a} \quad \text{where } \phi = [a, \Box b]$$

$$(S4_b)\ \frac{X; \blacksquare Y; \Box Z; \Box^s \phi}{X; \Box Y; \Box Z; \Box a \mid \blacksquare Y; \Box Z; \blacksquare^s \phi; \neg b} \qquad (S4_c)\ \frac{X; \blacksquare Y; \Box Z; \Box^s \neg\Box a}{\blacksquare Y; \Box Z; \blacksquare^s \neg\Box a; \neg a}$$

where $\phi = [a, \neg\Box b]$
In the rules $(S4_a)$ and $(S4_b)$, X does not contain $\blacksquare$

Table 3.1: Clausal tableau rules for propositional modal logics

$$(K4') \ \frac{X;\Box Y;\Box^2 Z;\neg\Box a}{\boxdot Y;\Box Z;\neg a} \qquad (KD4) \ \frac{X;\Box Y;\Box^2 Z;\boxdot U}{\boxdot Y;\Box Z;\boxdot U}$$

where Y is a set of simple clauses,
and for $(KD4)$ $Y \bigcup Z$ is not empty

$$(K_r) \ \frac{X;\boxdot\phi}{X;\blacksquare\phi;\phi} \qquad (K4_a) \ \frac{X;\blacksquare Y;\boxdot\psi}{X;\boxdot Y;\Box b \mid X;\blacksquare Y;\blacksquare\psi;a}$$

where $\phi = [a_1,\ldots,a_k]$ and $\psi = [a,\Box b]$

$$(K4_b) \ \frac{X;\blacksquare Y;\boxdot Z;\boxdot\phi}{X;\boxdot Y;\boxdot Z;\boxdot a \mid \blacksquare Y;\boxdot Z;\blacksquare\phi;\neg b} \qquad (K4_c) \ \frac{X;\blacksquare Y;\boxdot Z;\boxdot\neg\Box a}{\blacksquare Y;\boxdot Z;\blacksquare\neg\Box a;\neg a}$$

where $\phi = [a,\neg\Box b]$
In the rules $(K4_a)$ and $(K4_b)$, X does not contain $\blacksquare$

$$(K4_d) \ \frac{X;\boxdot Y;\Box Z;\Box^2 U;\boxdot\phi}{X;\boxdot Y;\Box Z;\Box^2 U;\boxdot a \mid \boxdot Y;\boxdot Z;\Box U;\boxdot\phi;\neg b}$$

where $\phi = [a,\neg\Box b]$, Z is a set of simple clauses,
and $Z \bigcup U$ is not empty

$$(K4_e) \ \frac{X;\boxdot Y;\Box Z;\Box^2 U;\boxdot\neg\Box a}{\boxdot Y;\boxdot Z;\Box U;\boxdot\neg\Box a;\neg a}$$

where Z is a set of simple clauses, and $Z \bigcup U$ is not empty

Table 3.2: Clausal tableau rules for propositional modal logics, c.d.

$\mathcal{C}L$	Static Rules	Transitional Rules
$\mathcal{C}K$	$(\vee),(\bot)$	(K)
$\mathcal{C}KD$	$(\vee),(\bot)$	$(K),(KD)$
$\mathcal{C}T$	$(\vee),(\bot),(T)$	(K)
$\mathcal{C}KB$	$(\vee),(\bot),(KB),(KB_b)$	(K)
$\mathcal{C}KDB$	$(\vee),(\bot),(KDB),(KB_b)$	$(K),(KD)$
$\mathcal{C}B$	$(\vee),(\bot),(KTB),(T')$	(K)

	Rules
$\mathcal{C}K4$	$(\vee),(\bot),(K_r),(K4_a),(K4_b),(K4_c),(K4_d),(K4_e),(K4')$
$\mathcal{C}KD4$	$(\vee),(\bot),(K_r),(K4_a),(K4_b),(K4_c),(K4_d),(K4_e),(K4'),(KD4)$
$\mathcal{C}S4$	$(\vee),(\bot),(T_r),(S4_a),(S4_b),(S4_c),(K4)$

Table 3.3: Clausal tableau calculi for propositional modal logics

The rule (T_r) is just a restricted version of (T), so the assertion also holds for it.

For $(S4_a)$, let M be a $S4$-model and let $M, w \models X; \blacksquare Y; \Box^s[a, \Box b]$. It is obvious that $M, w \models [a, \Box b]$. If $M, w \not\models a$ then $M, w \models \Box b$. Therefore either $M, w \models X; \blacksquare Y; \blacksquare^s[a, \Box b]; a$ or $M, w \models X; \Box Y; \Box b$.

For $(S4_b)$, let M be a $S4$-model and let $M, w \models X; \blacksquare Y; \Box Z; \Box^s[a, \neg\Box b]$. Suppose that $M, w \not\models \Box a$. There exists a world u reachable from w such that $M, u \models \neg a$. Thus $M, u \models \neg\Box b$, and then there exists a world v reachable from u such that $M, v \models \neg b$. It is easily seen that $M, v \models \blacksquare Y; \Box Z; \blacksquare^s[a, \neg\Box b]; \neg b$.

For $(S4_c)$, let M be a $S4$-model and let $M, w \models X; \blacksquare Y; \Box Z; \Box^s\neg\Box a$. We have $M, w \models \neg\Box a$, hence there exists a world u reachable from w such that $M, u \models \neg a$. It is easily seen that $M, u \models \blacksquare Y; \Box Z; \blacksquare^s\neg\Box a; \neg a$.

Recall that in Table 3.2 formulae of the form $\boxdot\phi$ or $\blacksquare\phi$ have the same semantical meaning as $(\Box\phi; \phi)$. The assertion obviously holds for (K_r). The proofs for $(K4')$ and $(KD4)$ are straightforward.

For $(K4_a)$, let M be a $K4$-model and let $M, w \models X; \blacksquare Y; \boxdot[a, \Box b]$. We have $M, w \models [a, \Box b]$. Hence either $M, w \models a$ or $M, w \models \Box b$.

For $(K4_b)$, let M be a $K4$-model and let $M, w \models X; \blacksquare Y; \boxdot Z; \boxdot[a, \neg\Box b]$. We have $M, w \models \Box[a, \neg\Box b]; [a, \neg\Box b]$. Suppose that $M, w \not\models \boxdot a$. In both of the cases $M, w \not\models a$ or $M, w \not\models \Box a$ we can derive that there exists a world u reachable from w such that $M, u \models \neg b$. It is easy to check that $M, u \models \blacksquare Y; \boxdot Z; \blacksquare[a, \neg\Box b]; \neg b$.

For $(K4_c)$, let M be a $K4$-model and let $M, w \models X; \blacksquare Y; \boxdot Z; \boxdot\neg\Box a$. We have $M, w \models \neg\Box a$. Therefore there exists a world u reachable from w such that $M, u \models \neg a$. It is easy to check that $M, u \models \blacksquare Y; \boxdot Z; \blacksquare\neg\Box a; \neg a$.

The proofs for $(K4_d)$ and $(K4_e)$ are similar to the proofs for $(K4_b)$ and $(K4_c)$, respectively. $\Box$

Corollary 3.2.2 *The calculi listed in Table 3.3 are sound.*

3.3 Estimating Depths of Tableaux

To estimate depths of tableaux in $\mathcal{C}K$, $\mathcal{C}KD$, $\mathcal{C}T$, $\mathcal{C}KB$, $\mathcal{C}KDB$, or $\mathcal{C}B$ we define an integer function f, which takes a clause or a set of clauses as parameter, as follows:

$$f(a) = 0, \quad f(\neg\Box a) = 1$$
$$f([A_1, \ldots, A_k]) = 1 + max\{f(A_i) \mid 1 \leq i \leq k\}, \quad \text{where } k > 1$$
$$f(\Box^s\phi) = (f(\phi) + 2).s + f(\phi), \text{where } s \geq 1 \text{ and } \phi \text{ is a simple clause}$$
$$f(\blacksquare\phi) = f(\phi) + 1$$
$$f(X; Y) = f(X) + f(Y)$$

Observe that if R is one of the rules $(\vee)$, (K), (KD), (T), (KB), (KDB), (KB_b), and (KTB), X is an instance of the numerator of R, and Y is one of the corresponding instances of the denominators of R, then $f(Y) < f(X)$. This leads to the following conclusion:

Lemma 3.3.1 *Let X be a set of clauses, with $rlength(X) = n$ and $mdepth(X) = s$. Then every $\mathcal{C}L$-tableau for X, where $\mathcal{C}L$ is one of the calculi $\mathcal{C}K$, $\mathcal{C}KD$, $\mathcal{C}T$, $\mathcal{C}KB$, $\mathcal{C}KDB$, and $\mathcal{C}B$, has depth of order $O((s+1).n)$.*

Proof. It suffices to show that $f(X) = O((s+1).n)$. Observe that if ϕ is a simple clause then $f(\phi) \leq 3$. If ϕ is a clause then $f(\phi) \leq (3+2)s + 3 < 5(s+1)$. Therefore $f(X) < 5(s+1).n$. $\square$

To estimate depths of $\mathcal{C}S4$-tableaux we fix a set X of clauses, let $n = rlength(X)$, and define an integer function f_X, which takes a clause or a set of clauses as parameter, as follows:

$$
\begin{array}{l}
f_X(a) = 0, \quad f_X(\neg\Box a) = 1 \\
f_X([A_1, \ldots, A_k]) = 1 + max\{f_X(A_i) \mid 1 \leq i \leq k\}, \quad \text{where } k > 1 \\
f_X(\Box^s[a_1, \ldots, a_k]) = 2, \quad f_X(\blacksquare^s[a_1, \ldots, a_k]) = 0, \quad \text{where } s \geq 1 \\
f_X(\Box^s[a, \Box b]) = f_X(\Box^s[a, \neg\Box b]) = f_X(\Box^s\neg\Box a) = 2n+1, \quad \text{where } s \geq 1 \\
f_X(\blacksquare^s[a, \Box b]) = f_X(\blacksquare^s[a, \neg\Box b]) = f_X(\blacksquare^s\neg\Box a) = 2n, \quad \text{where } s \geq 1 \\
f_X(Y; Z) = f_X(Y) + f_X(Z)
\end{array}
$$

Observe that for any $\mathcal{C}S4$-tableau T for X, if z is a child node of y in T, and Y and Z are contents of y and z respectively, then $f_X(Z) < f_X(Y)$. Furthermore, we have $f_X(X) < n.(2n+1)$. This leads to the following conclusion:

Lemma 3.3.2 *Let X be a set of clauses, with $rlength(X) = n$. Then every $\mathcal{C}S4$-tableau for X has depth of order $O(n^2)$.*

To estimate depths of $\mathcal{C}K4$-tableaux and $\mathcal{C}KD4$-tableaux we fix a set X of clauses, let $n = rlength(X)$, and define an integer function g_X, which takes a clause or a set of clauses as parameter, as follows:

$$
\begin{array}{l}
g_X(a) = 0, \quad g_X(\neg\Box a) = 1 \\
g_X([A_1, \ldots, A_k]) = 1 + max\{g_X(A_i) \mid 1 \leq i \leq k\}, \quad \text{where } k > 1 \\
g_X(\boxdot[a_1, \ldots, a_k]) = 2, \quad g_X(\blacksquare[a_1, \ldots, a_k]) = 0 \\
g_X(\boxdot[a, \Box b]) = g_X(\boxdot[a, \neg\Box b]) = g_X(\boxdot\neg\Box a) = 2n+4 \\
g_X(\blacksquare[a, \Box b]) = g_X(\blacksquare[a, \neg\Box b]) = g_X(\blacksquare\neg\Box a) = 2n+3 \\
g_X(\Box^s\phi) = g_X(\boxdot\phi) + s, \quad \text{where } s \geq 1 \text{ and } \phi \text{ is a simple clause} \\
g_X(Y; Z) = g_X(Y) + g_X(Z)
\end{array}
$$

Observe that for any $\mathcal{C}L$-tableau T for X, where $\mathcal{C}L$ is $\mathcal{C}K4$ or $\mathcal{C}KD4$, if z is a child node of y in T, and Y and Z are contents of y and z respectively, then $g_X(Z) < g_X(Y)$. Furthermore, we have $g_X(X) < n.(2n+4)$. This leads to the following conclusion:

Lemma 3.3.3 *Let X be a set of clauses, with $rlength(X) = n$. Then every $\mathcal{C}K4$-tableau and every $\mathcal{C}KD4$-tableau for X has depth of order $O(n^2)$.*

We conclude that all tableau systems given in Table 3.3 have the finite tableau property.

3.4 Completeness of $\mathcal{C}K$, $\mathcal{C}KD$, $\mathcal{C}T$, $\mathcal{C}KB$, $\mathcal{C}KDB$, and $\mathcal{C}B$

In this section, we use L to denote one of the logics K, KD, T, KB, KDB, and B; and we use $\mathcal{C}L$ to denote the corresponding calculus. To show completeness of the calculi we give an algorithm that for a given $\mathcal{C}L$-consistent set X of clauses not containing the connective $\blacksquare$ will construct a L-model graph for X. In the algorithm, saturation is exploited. We saturate a node by applying static $\mathcal{C}L$-rules to it until no changes occur. When a node has been saturated, its successors will be created according to transitional rules. Using saturation we will not have to update contents of the nodes we have processed. The method has been previously used by other authors, e.g. Hintikka [13], Rautenberg [30] and Goré [11].

Definition 3.4.1 We say that a set X of clauses is *$\mathcal{C}L$-saturated* if no static $\mathcal{C}L$-rule is applicable for X.

Definition 3.4.2 Let X be a $\mathcal{C}L$-consistent set of clauses. We say that a set Y of clauses is a *$\mathcal{C}L$-saturation* of X if:

- Y can be obtained from X by applying static $\mathcal{C}L$-rules, and
- Y is $\mathcal{C}L$-consistent and $\mathcal{C}L$-saturated.

Lemma 3.4.3 *For any $\mathcal{C}L$-consistent set X of clauses, there exists a $\mathcal{C}L$-saturation of X.*

Proof. We apply static $\mathcal{C}L$-rules to X until no changes occur. When a rule is applied to X we always choose a $\mathcal{C}L$-consistent denominator. The process always terminates because every $\mathcal{C}L$-tableau for X is finite. The resulting set is a $\mathcal{C}L$-saturation of X. □

Consider Algorithm 3.4.6. Given a $\mathcal{C}L$-consistent set X of clauses, the algorithm constructs a L-model graph $M = \langle W, \tau, R, H \rangle$ for X. The skeleton of M is the tree R_0. The algorithm always goes downward along R_0; it never recomputes contents of the nodes it has passed. If $KB \leq L$ then for any nodes w and u such that $R_0(w, u)$, we have $R(u, w)$. This implies that if $KB \leq L$, $R_0(w, u)$, and $\boxtimes c \in H(u)$, then c should be present in $H(w)$. Hence, if $KB \leq L$ and $R_0(w, u)$, then being at u (and having $H(w)$ fixed) the strategy is to reduce clauses of the form $\boxtimes c$ in u. This leads to the following formulation of *redundants*.

Definition 3.4.4 Let X be a $\mathcal{C}L$-consistent set of clauses, and Y be a $\mathcal{C}L$-saturation of X. We call an atom $\boxtimes c \in Y$ a *redundant* with respect to L and X if $L = KB$ or $L = KDB$ and:

- X contains no clauses of the form $\boxtimes^{2s+1}c$, where $s \geq 0$
- For any formula $\Box^{2s}[a, \Box c] \in X$, where $s \geq 0$, it holds that $a \in Y$
- For any formula $\Box^{2s+1}[c, \Box b] \in X$, where $s \geq 0$, it holds that $b \in Y$;

or $L = B$ and:

- X contains no clauses of the form $\boxtimes^{s}c$, where $s \geq 1$
- For any formula $\Box^{s}[a, \Box c] \in X$, where $s \geq 0$, it holds that $a \in Y$
- For any formula $\Box^{s}[c, \Box b] \in X$, where $s \geq 1$, it holds that $b \in Y$.

We denote the set obtained from Y by deleting all redundants wrt. L and X by $\mathit{RemoveRedundants}_L(Y, X)$.

(Note that if X is a $\mathcal{C}L$-consistent set of clauses, Y is a $\mathcal{C}L$-saturation of X, and $Z = \mathit{RemoveRedundants}_L(Y, X)$, then Z is also a $\mathcal{C}L$-saturation of X. If $Z \neq Y$ then during the process of generating Y from X certain or-branching is not *optimal*.)

We write $\phi \in_\vee X$ to mean that if ϕ is of the form $[A_1, \ldots, A_k]$, with $k > 1$, then $A_i \in_\boxtimes X$ for some $1 \leq i \leq k$, else $\phi \in_\boxtimes X$.

Lemma 3.4.5 *Let X be a $\mathcal{C}L$-consistent set of clauses not containing the connective $\blacksquare$. Let Y be a $\mathcal{C}L$-saturation of X, and let $Z = \mathit{RemoveRedundants}_L(Y, X)$. Then the following assertions hold:*

1. *Z is $\mathcal{C}L$-consistent.*
2. *If $\phi \in X$ then $\phi \in_\vee Z$.*
3. *If $L = T$ and $\boxtimes^s\phi \in Z$, where $s \geq 1$ and ϕ is a simple clause, then $\phi \in_\vee Z$.*
4. *If $L = B$ and $\boxtimes^s\phi \in Z$, where $s \geq 1$ and ϕ is a simple clause of one of the forms $[a_1, \ldots, a_k]$, $[a, \neg\Box b]$, then $\phi \in_\vee Z$.*
5. *If $L = KB$ or $L = KDB$, and $\boxtimes^{2s}\phi \in Z$, where $s \geq 1$ and ϕ is a simple or $\Box$-simple clause, then $\phi \in_\vee Z$.*
6. *If $\phi \in Z$ and $\phi = \boxtimes^s\psi$, where $s \geq 1$ and ψ is a simple clause such that if $s = 1$ then ψ is not a classical literal, then there exists $t \geq 0$ such that $\Box^{s+2t}\psi \in X$.*
7. *If $L = KB$ or $L = KDB$, and $\boxtimes[a, \Box b] \in Z$, then $\boxtimes a \in Z$ or $b \in Z$.*

8. *If $L = B$ and $\boxtimes^s[a, \Box b] \in Z$, where $s \geq 1$, then $(\boxtimes a \in Z \wedge a \in Z)$ or $(\boxtimes b \in Z \wedge b \in Z)$ or $(a \in Z \wedge b \in Z)$.*

Proof. The proofs of 1 and 2 are straightforward.

3) Suppose that $L = T$. The set Z cannot contain clauses of the form $\Box^s\phi$ with $s \geq 1$. Suppose that $\blacksquare^s\phi \in Z$, where $s \geq 1$ and ϕ is a simple clause. It follows that $\Box^s\phi \in X$, and ϕ is introduced during saturation of X. If ϕ is of the form $[A_1, \ldots, A_k]$, then $A_i \in_{\boxtimes} Z$ for some $1 \leq i \leq k$, else $\phi \in Z$. Thus $\phi \in_{\vee} Z$.

4) The proof is similar to the proof of 3.

5) Suppose that $L = KB$ or $L = KDB$. The set Z cannot contain clauses of the form $\Box^{2s}\phi$ with $s \geq 1$. Suppose that $\blacksquare^{2s}\phi \in Z$, where $s \geq 1$ and ϕ is a simple or $\Box$-simple clause. It follows that $\Box^{2s}\phi \in X$, and ϕ is introduced during saturation of X. Consider the following cases:

- If ϕ is of the form $\Box a$, then $\phi \in Z$, because $\Box a$ is not a redundant wrt. L and X.

- If ϕ is of the form $\Box\psi$, where ψ is not a classical literal, then $\phi \in_{\boxtimes} Z$.

- If ϕ is a simple clause not of the form $[a, \Box b]$, then $\phi \in_{\vee} Z$.

- For the last case, suppose that ϕ is of the form $[a, \Box b]$. If $a \notin Z$ then $\Box b \in Y$, and since $\Box b$ is not a redundant wrt. L and X, it yields that $\Box b \in Z$.

For all of these cases we can conclude that $\phi \in_{\vee} Z$.

6) Suppose that the conditions of the assertion hold. If $\phi = \Box^s\psi$ then $\phi \in X$, and $\Box^{s+2t}\psi \in X$ for $t = 0$. Suppose that $\phi = \blacksquare^s\psi$. It follows that $L \in \{T, KB, KDB, B\}$. The assertion clearly holds for $L = KB$ or $L = KDB$. If $L = T$ or $L = B$ then $\Box^s\psi \in X$, and we choose $t = 0$.

7) Suppose that $L = KB$ or $L = KDB$, $\boxtimes[a, \Box b] \in Z$, and $b \notin Z$. We show that $\boxtimes a \in Z$. We have $b \notin Y$ and $\Box^{2s+1}[a, \Box b] \in X$ for some $s \geq 0$. Hence either the rule (KB) or (KDB), and then the rule (KB_b) must be used in the saturation, and $\Box a \in Y$. Since $b \notin Y$, $\Box a$ is not a redundant wrt. L and X. Hence $\Box a \in Z$.

8) Suppose that $L = B$ and $\boxtimes^s[a, \Box b] \in Z$. Suppose that $b \notin Z$, we show that $\boxtimes a \in Z$ and $a \in Z$. We have $b \notin Y$ and $\Box^s[a, \Box b] \in X$. The rule (KTB) must be used in the saturation, and $\Box a$ is introduced. Since $b \notin Y$, the clauses $\Box a$ and $\blacksquare a$ are not redundants wrt. L and X. Hence $\blacksquare a \in Z$ and $a \in Z$. Analogously, if $a \notin Z$ then $\boxtimes b \in Z$ and $b \in Z$. We conclude that the assertion holds. □

In the following algorithm we assume that we have an oracle to compute the step 2b. For how this algorithm can be implemented, see Section 3.7.1.

Algorithm 3.4.6 *Let X be a $\mathcal{C}L$-consistent set of clauses not containing the connective $\blacksquare$. We construct a consistent L-model graph $M = \langle W, \tau, R, H\rangle$ that satisfies X as follows:*

1. *Let $W = \{\tau\}$, $H_0(\tau) = X$, $R_0 = \emptyset$, and mark τ as unsolved.*
2. (a) *Take an unsolved world w from W.*
   (b) *Let Y be a $\mathcal{C}L$-saturation of $H_0(w)$.*
   (c) *Set $H(w) = RemoveRedundants_L(Y, H_0(w))$.*
   (d) *For every atom $\neg\Box a$ from $H(w)$:*
      - *Create a new world w_a, add it to W and mark it as unsolved.*
      - *Set $R_0 = R_0 \bigcup \{(w, w_a)\}$.*
      - *Set $H_0(w_a) = \{\neg a\} \bigcup \{Y \mid \boxtimes Y \in H(w)\}$.*

   (e) *If L = KD or L = KDB, and there are no atoms of the form $\neg\Box a$ in $H(w)$ then*
      - *If $H(w)$ is not empty*
         - *Create a new world w', add it to W and mark it as unsolved.*
         - *Set $R_0 = R_0 \bigcup \{(w, w')\}$.*
         - *Set $H_0(w') = \{Y \mid \boxtimes Y \in H(w)\}$.*
      - *else set $R_0 = R_0 \bigcup \{(w, w)\}$.*

   (f) *Mark w as resolved.*
3. *While there are unsolved worlds, repeat the step 2.*
4. *Set $R = Ext_L(R_0)$.*

The step 2e takes place when $L = KD$ or $L = KDB$, and it connects w to at least one node. The step 2d corresponds to the rule (K), the step 2e to the rule (KD).

Note that the relation R_0 in the above algorithm is a tree. For any nodes w and u such that $R_0(w, u)$, we have $mdepth(H(u)) < mdepth(H(w))$. Therefore the depth of the tree R_0 is bounded by the modal depth of the input X. This also yields that Algorithm 3.4.6 always terminates.

Lemma 3.4.7 *Let X be a $\mathcal{C}L$-consistent set of clauses not containing the connective $\blacksquare$, and let $M = \langle W, \tau, R, H\rangle$ be the model graph constructed by Algorithm 3.4.6 for X. Then M is a consistent L-model graph satisfying X.*

Proof. Since X is $\mathcal{C}L$-consistent, by Lemma 3.4.5:1, M is a consistent L-model graph. By Lemma 3.4.5:2, for any clause $\phi \in X$ we have $\phi \in_\vee H(\tau)$. To prove that M satisfies X it is sufficient to show that for any $w \in W$ and $\phi \in H(w)$, $M, w \vDash \phi$. Denote this assertion by $(\star)$. It clearly holds for the case when ϕ is of the form $\neg\Box a$. We observe that for any $w \in W$, $H(w)$ does not contain clauses of the form $[A_1, \ldots, A_k]$, with $k > 1$.

We now show that $(\star)$ holds for the case $\phi = \boxtimes c$, where c is a classical literal.

The case $L \in \{K, KD, T\}$ is trivial.

Case $L = KB$ or $L = KDB$: Suppose that $\boxtimes c \in H(w)$. We have $\forall u \in W$ $R_0(w, u) \to c \in H(u)$. It remains to show that $\forall u\ R_0(u, w) \to c \in H(u)$. Suppose that $R_0(u, w)$ holds. Since $\boxtimes c \in H(w)$, $\boxtimes c$ is not a redundant wrt. L and $H_0(w)$. Hence one of the following conditions must hold:

- $\boxtimes^{2s+2} c \in H(u)$ for some $s \geq 0$
- $\boxtimes^{2s+1}[a, \Box c] \in H(u)$ for some $a \notin H(w)$ and $s \geq 0$
- $\boxtimes^{2s+2}[c, \Box b] \in H(u)$ for some $b \notin H(w)$ and $s \geq 0$

If the first condition holds, we have $c \in H(u)$, by Lemma 3.4.5:5. Suppose that the second condition holds and $c \notin H(u)$. By Lemma 3.4.5:5, we have $\boxtimes[a, \Box c] \in H(u)$. Hence $\boxtimes a \in H(u)$, by Lemma 3.4.5:7. This contradicts the fact that $a \notin H(w)$. Now assume that the third conditions holds and $c \notin H(u)$. By Lemma 3.4.5:5, we have $[c, \Box b] \in_\vee H(u)$, which implies $\boxtimes b \in H(u)$. This contradicts the fact that $b \notin H(w)$. Hence $c \in H(u)$.

Case $L = B$: Suppose that $\boxtimes c \in H(w)$. We have $\forall u \in W\ R_0(w, u) \to c \in H(u)$. By Lemma 3.4.5:4, we have $c \in H(w)$. It remains to show that $\forall u\ R_0(u, w) \to c \in H(u)$. Suppose that $R_0(u, w)$ holds. Since $\boxtimes c \in H(w)$, $\boxtimes c$ is not a redundant wrt. L and $H_0(w)$. Hence one of the following conditions must hold:

- $\boxtimes^s c \in H(u)$ for some $s \geq 2$
- $\boxtimes^s[a, \Box c] \in H(u)$ for some $a \notin H(w)$ and $s \geq 1$
- $\boxtimes^s[c, \Box b] \in H(u)$ for some $b \notin H(w)$ and $s \geq 2$

If the first condition holds, we have $c \in H(u)$, by Lemma 3.4.5:4. Suppose that the second condition holds and $c \notin H(u)$. By Lemma 3.4.5:8, we have $\boxtimes a \in H(u)$, which contradicts the fact that $a \notin H(w)$. Now assume that the third condition holds and $c \notin H(u)$. By Lemma 3.4.5:8, we have $\boxtimes b \in H(u)$, which contradicts the fact that $b \notin H(w)$. Hence $c \in H(u)$.

It remains to show that $(\star)$ holds for the case $\phi = \boxtimes^s \psi$, where $s \geq 1$ and ψ is a simple clause such that if $s = 1$ then ψ is not a classical literal.

Case $L = K$ or $L = KD$: If $\boxtimes^s \psi \in H(w)$ and $R(w, u)$ hold, then $\Box^{s-1}\psi \in_\vee H(u)$. By induction on s, it yields that if $\phi \in H(w)$ then $M, w \models \phi$.

Case $L = T$: We show by induction on i that for any $u \in W$, if there exists $s \geq i$ such that $\boxtimes^s\psi \in H(u)$ then $M, u \vDash \Box^i\psi$. Case $i = 0$: Suppose that $\boxtimes^s\psi \in H(u)$. By Lemma 3.4.5:3, we have $\psi \in_\vee H(u)$, hence $M, u \vDash \psi$. Now assume that the assertion holds for all $0 \leq j < i$, and $\boxtimes^s\psi \in H(u)$ for some $s \geq i$. We show that $M, u \vDash \Box^i\psi$. By induction, we have $M, u \vDash \Box^{i-1}\psi$. For any v such that $R_0(u, v)$, we have $\boxtimes^{s-1}\psi \in H(v)$, which by induction implies $M, v \vDash \Box^{i-1}\psi$. Thus $M, u \vDash \Box^i\psi$. As a consequence, if $\phi \in H(w)$ then $M, w \vDash \phi$.

Case $L = KB$ or $L = KDB$: We show by induction on i that for any $u \in W$, if there exists $k \geq 0$ such that $\boxtimes^{i+2k}\psi \in H(u)$ then $M, u \vDash \Box^i\psi$. Case $i = 0$: Suppose that $\boxtimes^{2k}\psi \in H(u)$. By Lemma 3.4.5:5, we have $\psi \in_\vee H(u)$, hence $M, u \vDash \psi$. Now assume that the assertion holds for all $0 \leq j < i$, and $\boxtimes^{i+2k}\psi \in H(u)$ for some $k \geq 0$. We show that $M, u \vDash \Box^i\psi$. For any v such that $R_0(u, v)$, we have $\boxtimes^{i+2k-1}\psi \in H(v)$, which by induction implies $M, v \vDash \Box^{i-1}\psi$. For any v such that $R_0(v, u)$, by Lemma 3.4.5:6, we have $\boxtimes^{i+2k+2t+1}\psi \in H(v)$ for some $t \geq 0$, which by induction implies $M, v \vDash \Box^{i-1}\psi$. Thus $M, u \vDash \Box^i\psi$. As a consequence, if $\phi \in H(w)$ then $M, w \vDash \phi$.

Case $L = B$: We show by induction on i that for any $u \in W$, if there exists $s \geq i$ such that $\boxtimes^s\psi \in H(u)$ then $M, u \vDash \Box^i\psi$. Case $i = 0$: Suppose that $\boxtimes^s\psi \in H(u)$. If ψ is of the form $[a, \Box b]$ then by Lemma 3.4.5:8 we have $a \in H(u)$ or $\boxtimes b \in H(u)$, hence $M, u \vDash \psi$. If ψ is not of the form $[a, \Box b]$ then by Lemma 3.4.5:4 we have $\psi \in_\vee H(u)$, which implies $M, u \vDash \psi$. Now assume that the assertion holds for all $0 \leq j < i$, and $\boxtimes^s\psi \in H(u)$ for some $s \geq i$. We show that $M, u \vDash \Box^i\psi$. By induction, we have $M, u \vDash \Box^{i-1}\psi$. For any v such that $R_0(u, v)$, we have $\boxtimes^{s-1}\psi \in H(v)$, which by induction implies $M, v \vDash \Box^{i-1}\psi$. For any v such that $R_0(v, u)$, by Lemma 3.4.5:6, we have $\boxtimes^{s+2t+1}\psi \in H(v)$ for some $t \geq 0$, which by induction implies $M, v \vDash \Box^{i-1}\psi$. Thus $M, u \vDash \Box^i\psi$. As a consequence, if $\phi \in H(w)$ then $M, w \vDash \phi$. □

The following theorem immediately follows from Corollary 3.2.2 and Lemma 3.4.7

Theorem 3.4.8 *The calculi $\mathcal{C}K$, $\mathcal{C}KD$, $\mathcal{C}T$, $\mathcal{C}KB$, $\mathcal{C}KDB$ and $\mathcal{C}B$ are sound and complete.*

3.5 Completeness of $\mathcal{C}S4$

Between the rules of the calculus $\mathcal{C}S4$, we can categorize the rules $(\vee)$, (T_r), and $(S4_a)$ to be static, the rules $(K4)$ and $(S4_c)$ to be transitional. The rule $(S4_b)$ is an odd one, its first denominator is static, while its second denominator is transitional. For $\mathcal{C}S4$ we define two kinds of saturation as follows:

Definition 3.5.1 Let X be a $\mathcal{C}S4$-consistent set of clauses. Let $Y = X$ and repeat the following steps until Y does not change:

- If Y is an instance of the numerator of the rule $(\vee)$ then one of the corresponding instances of denominators of $(\vee)$, lets say Z, must be $\mathcal{C}S4$-consistent. Set $Y = Z$.
- Let R be $(S4_a)$ or $(S4_b)$. If Y is an instance of the numerator of R and the corresponding instance Z of the first denominator of R is $\mathcal{C}S4$-consistent, then set $Y = Z$.

It is easily seen that this process always terminates. We call Y a *first kind* $\mathcal{C}S4$*-saturation* of X.

Definition 3.5.2 Let X be a $\mathcal{C}S4$-consistent set of clauses. Let $Y = X$ and repeat the following steps until Y does not change:

- Let R be $(\vee)$ or (T_r). If Y is an instance of the numerator of R then one of the corresponding instances of denominators of R, lets say Z, must be $\mathcal{C}S4$-consistent. Set $Y = Z$.
- If Y is an instance of the numerator of the rule $(S4_a)$ then set Y to be the corresponding instance of the second denominator of $(S4_a)$.

It is easily seen that this process always terminates. We call Y a *second kind* $\mathcal{C}S4$*-saturation* of X.

Lemma 3.5.3 *Let X be a $\mathcal{C}S4$-consistent set of clauses not containing the connective $\blacksquare$. Let Y be a first kind $\mathcal{C}S4$-saturation of X, and Z be a second kind $\mathcal{C}S4$-saturation of Y. Then the following assertions hold:*

1. *Y and Z are $\mathcal{C}S4$-consistent.*
2. *If $[A_1, \ldots, A_k] \in X$ then there exists $1 \leq i \leq k$ such that $A_i \in Y$.*
3. *If $\Box^s[a, \Box b] \in X$, where $s \geq 1$, then $\Box b \in Y$ or $\Box^s[a, \Box b] \in Y$.*
4. *If $\Box^s[a, \neg\Box b] \in X$, where $s \geq 1$, then $\Box a \in Y$ or $\Box^s[a, \neg\Box b] \in Y$.*
5. *If $\phi \in X$ and ϕ is not of any of the forms $[A_1, \ldots, A_k]$, $\Box^s[a, \Box b]$, $\Box^s[a, \neg\Box b]$, where $s \geq 1$, then $\phi \in Y$.*
6. *If $\Box^s[a_1, \ldots, a_k] \in Y$, where $s \geq 0$, then $a_i \in Z$ for some $1 \leq i \leq k$.*
7. *If $\Box^s[a, \Box b] \in Y$ then $a \in Z$.*
8. *If $\phi \in Y$ and ϕ is not of the form $[A_1, \ldots, A_k]$, then $\phi \in_{\boxtimes} Z$.*

The proof of this lemma is straightforward.

In the following algorithm we assume that we have oracles to compute the steps 2b and 4.

Algorithm 3.5.4 *Let X be a $\mathcal{C}S4$-consistent set of clauses not containing the connective $\blacksquare$. We construct a consistent $S4$-model graph $M = \langle W, \tau, R, H \rangle$ that satisfies X as follows:*

1. *Set $W = \{\tau\}$, $H_0(\tau) = X$, $R_0 = R_1 = \emptyset$, and mark τ as unsolved.*

2. (a) *Take an unsolved world w from W.*

   (b) *Let $H_1(w)$ be a first kind $\mathcal{C}S4$-saturation of $H_0(w)$.*

   (c) *For every atom $\neg\Box a$ from $H_1(w)$:*
   - *Create a new world w_a, add it to W, and mark it as unsolved.*
   - *Set $R_0 = R_0 \bigcup \{(w, w_a)\}$.*
   - *Set $H_0(w_a) = \{\neg a\} \bigcup \{\Box Y | \Box Y \in H_1(w)\}$.*

   (d) *For every clause $\phi \in H_1(w)$ such that $\phi = \Box^s\psi$, where $s \geq 1$ and ($\psi = [a, \neg\Box b]$ or $\psi = \neg\Box b$):*
   - *Create a new world w_ϕ, add it to W, and mark it as unsolved.*
   - *Set $R_0 = R_0 \bigcup \{(w, w_\phi)\}$.*
   - *Set* $H_0(w_\phi) = \{\neg b, \blacksquare^s\psi\} \bigcup \{\blacksquare Y \mid \blacksquare Y \in H_1(w)\}$
     $\bigcup \{\Box Z \mid \Box Z \in H_1(w)\}$.

   (e) *For every clause $\phi \in H_1(w)$ such that $\phi = \blacksquare^s\psi$, where $s \geq 1$ and ($\psi = [a, \neg\Box b]$ or $\psi = \neg\Box b$):*
   - *Let $w_\phi \in W$ be the node on the path from τ to w (in the tree generated by R_0) such that for any node v on the path from w_ϕ to w we have $\phi \in H_1(v)$, and that for v being the node immediately above w_ϕ we have $\phi \notin H_1(v)$, i.e. :*

     $$\begin{array}{l} w_\phi = w \vee R_0^*(w_\phi, w) \\ \forall v \; (v = w_\phi \vee R_0^*(w_\phi, v)) \wedge R_0^*(v, w) \rightarrow \phi \in H_1(v) \\ \forall v \; R_0(v, w_\phi) \rightarrow \phi \notin H_1(v) \end{array}$$

   - *Set $R_1 = R_1 \bigcup \{(w, w_\phi)\}$.*

   (f) *Mark w as resolved.*

3. *While there are unsolved worlds, repeat the step 2.*

4. *For every $w \in W$, let $H(w)$ be a second kind $\mathcal{C}S4$-saturation of $H_1(w)$.*

5. *Set $R = Ext_{S4}(R_0 \bigcup R_1)$.*

At the step 2e, the world w_ϕ exists because $H_1(\tau)$ does not contain the connective $\blacksquare$. At the step 2d, we use R_0^* to denote the transitive closure of R_0. The step 2c corresponds to the rule $(K4)$, the step 2d to the rules $(S4_b)$ and $(S4_c)$. It follows from Lemma 3.3.2 that this algorithm always terminates.

Lemma 3.5.5 *Consider the step 2e in the above algorithm. We have* $(\neg b) \in H_1(w_\phi)$.

Proof. Let u be the world such that $R_0(u, w_\phi)$ (such u exists because $H_1(w_\phi)$ contains ϕ while $H_1(\tau)$ does not). It is easily seen that w_ϕ was created from u at the step 2d with respect to $\Box^s\psi$. Therefore $(\neg b) \in H_1(w_\phi)$. $\Box$

Lemma 3.5.6 *Let M be the model graph constructed by Algorithm 3.5.4. M has the following properties:*

1. *For any $w \in W$, if $\blacksquare^s\phi \in H_1(w)$, where ϕ is a simple clause, then ϕ is of one of the forms $[a, \neg\Box b]$, $\neg\Box a$.*
2. *For any $w, u \in W$ such that $R_1(w, u)$, for any formula ψ, if $\boxtimes\psi \in H_1(w)$ then $\boxtimes\psi \in H_1(u)$.*

The proof of this lemma is straightforward.

Lemma 3.5.7 *Let M be the model graph constructed by Algorithm 3.5.4. Then M is a consistent S4-model graph, and for any $w \in W$ and $\phi \in H(w)$, $M, w \models \phi$.*

Proof. It follows from Lemma 3.5.3:1 that M is consistent. It is clear that R satisfies all $S4$-frame restrictions. Suppose that $\phi \in H(w)$, we show that $M, w \models \phi$.

It is easily seen that ϕ cannot be of any of the forms: $[A_1, \ldots, A_k]$ with $k > 1$, $\Box^s[a_1, \ldots, a_k]$ with $s \geq 1$, $\Box^s[a, \Box b]$ with $s \geq 1$.

Case $\phi = \neg\Box a$: We have $R_0(w, w_a)$ and $(\neg a) \in H_0(w_a)$. Hence $M, w \models \phi$.

Case $\phi = \blacksquare^s[a_1, \ldots, a_k]$, where $s \geq 1$: Denote $\psi = \Box^s[a_1, \ldots, a_k]$. We claim that for any $u \in W$ such that $\psi \in H_1(u)$ the following assertions hold:

$$\begin{array}{l} M, u \models [a_1, \ldots, a_k] \\ \forall v\ R_0(u, v) \to \psi \in H_1(v) \\ \forall v\ R_1(u, v) \to \psi \in H_1(v) \\ \forall v\ R(u, v) \to (M, v \models [a_1, \ldots, a_k]) \end{array}$$

The first assertion follows from Lemma 3.5.3:6, the second one follows from Lemma 3.5.3:5, the third one follows from Lemma 3.5.6, and the last one follows from the first three ones. Hence $\forall u \in W\ \psi \in H_1(u) \to (M, u \vDash \psi)$. Since $\phi \in H(w)$, it is easily seen that $\psi \in H_1(w)$. Therefore $M, w \vDash \phi$.

Case $\phi = \blacksquare^s[a, \Box b]$, where $s \geq 1$: Denote $\psi = \Box^s[a, \Box b]$. We claim that for any $u \in W$ such that $\psi \in H_0(u)$ the following assertions hold:

$$\begin{array}{l}\blacksquare b \in H(u) \vee a \in H(u)\\ \forall v\ R_0(u,v) \to \psi \in H_0(v) \vee \blacksquare b \in H(v)\\ \forall v\ R_1(u,v) \to \psi \in H_0(v) \vee \blacksquare b \in H(v)\end{array}$$

The first assertion follows from Lemma 3.5.3, items 3 and 7. For the second assertion: Suppose that $R_0(u, v)$ holds. We have $\psi \in H_1(u)$ or $\Box b \in H_1(u)$, hence $\psi \in H_0(v)$ or $\Box b \in H_0(v)$, and finally $\psi \in H_0(v)$ or $\blacksquare b \in H(v)$. For the third assertion: Suppose that $R_1(u, v)$ holds. We have $\psi \in H_1(u)$ or $\Box b \in H_1(u)$, hence $\psi \in H_1(v)$ or $\Box b \in H_1(v)$, by Lemma 3.5.6. It follows that $\psi \in H_0(v)$ or $\blacksquare b \in H(v)$.

Observe that for any $v \in W$, if $\blacksquare b \in H(v)$ then $M, v \vDash \Box b$ and $M, v \vDash \Box[a, \Box b]$. We conclude that $\forall u \in W\ \psi \in H_0(u) \to (M, u \vDash \psi)$. Since $\phi \in H(w)$, it is easily seen that $\psi \in H_0(w)$. Therefore $M, w \vDash \phi$.

Case $\phi = \boxtimes^s[a, \neg\Box b]$, where $s \geq 1$: We claim that for any $u \in W$ such that $\phi \in H_1(u)$ the following assertions hold:

$$\begin{array}{l}\exists v\ R(u,v) \wedge (\neg b) \in H(v)\\ \forall v\ R_0(u,v) \to \boxtimes^s[a, \neg\Box b] \in H_1(v) \vee \Box a \in H_1(v)\\ \forall v\ R_1(u,v) \to \boxtimes^s[a, \neg\Box b] \in H_1(v)\end{array}$$

The first assertion is clear. For the second assertion: Suppose that $R_0(u, v)$ holds. We have $\boxtimes^s[a, \neg\Box b] \in H_0(v)$, which implies $\boxtimes^s[a, \neg\Box b] \in H_1(v)$ or $\Box a \in H_1(v)$, by Lemma 3.5.3:4. The third assertion follows from Lemma 3.5.6. Therefore

$$\begin{array}{l}M, u \vDash [a, \neg\Box b]\\ \forall v\ R_0(u,v) \to \boxtimes^s[a, \neg\Box b] \in H_1(v) \vee M, v \vDash \Box[a, \neg\Box b]\\ \forall v\ R_1(u,v) \to \boxtimes^s[a, \neg\Box b] \in H_1(v)\end{array}$$

We conclude that $\forall u \in W\ \phi \in H_1(u) \to M, u \vDash \phi$. Since $\phi \in H(w)$, we have $\boxtimes^s[a, \neg\Box b] \in H_1(w)$, which implies $M, w \vDash \phi$.

Case $\phi = \boxtimes^s\neg\Box a$, where $s \geq 1$: Reasoning similarly as for the preceding case, we derive that for any $u \in W$ such that $\phi \in H_1(u)$, the following assertions hold:

$$\begin{array}{l}M, u \vDash \neg\Box a\\ \forall v\ R_0(u,v) \to \boxtimes^s\neg\Box a \in H_1(v)\\ \forall v\ R_1(u,v) \to \boxtimes^s\neg\Box a \in H_1(v)\end{array}$$

It follows that $\forall u \in W\ \phi \in H_1(u) \to M, u \vDash \phi$. Since $\phi \in H(w)$, we have $\boxtimes^s\neg\Box a \in H_1(w)$, which implies $M, w \vDash \phi$. $\Box$

Corollary 3.5.8 *Let X be a $\mathcal{C}S4$-consistent set of clauses not containing the connective $\blacksquare$. Let M be the model graph constructed by Algorithm 3.5.4 for the input X. Then M is a $S4$-model of X.*

Proof. By Lemma 3.5.7, M is a consistent $S4$-model graph and $M \models H(\tau)$. Since $H(\tau)$ is a second kind $\mathcal{C}S4$-saturation of $H_1(\tau)$, which is a first kind $\mathcal{C}S4$-saturation of X, by Lemma 3.5.3, items 2, 3, 4, 5, 6, and 8, we conclude that $M \models X$. □

The following theorem immediately follows from Corollaries 3.2.2 and 3.5.8.

Theorem 3.5.9 *The calculus $\mathcal{C}S4$ is sound and complete.*

3.6 Completeness of $\mathcal{C}K4$ and $\mathcal{C}KD4$

Between the rules of the calculi $\mathcal{C}K4$ and $\mathcal{C}KD4$, we can categorize the rules $(\vee)$, (K_r), and $(K4_a)$ to be static, the rules $(K4')$, $(KD4)$, $(K4_c)$, and $(K4_e)$ to be transitional. The rules $(K4_b)$ and $(K4_d)$ are odd ones, their first denominators are static, while their second denominators are transitional.

In this section we use L to denote $K4$ or $KD4$, and we write $\mathcal{C}L$ to denote the corresponding calculus. We define two kinds of $\mathcal{C}L$-saturation in the same way as for $\mathcal{C}S4$, as follows:

Definition 3.6.1 Let X be a $\mathcal{C}L$-consistent set of clauses. Let $Y = X$ and repeat the following steps until Y does not change:

- If Y is an instance of the numerator of the rule $(\vee)$ then one of the corresponding instances of denominators of $(\vee)$, lets say Z, must be $\mathcal{C}L$-consistent. Set $Y = Z$.
- Let R be one of the rules $(K4_a)$, $(K4_b)$, and $(K4_d)$. If Y is an instance of the numerator of R and the corresponding instance Z of the first denominator of R is $\mathcal{C}L$-consistent then set $Y = Z$.

It is easily seen that this process always terminates. We call Y a *first kind $\mathcal{C}L$-saturation* of X.

Definition 3.6.2 Let X be a $\mathcal{C}L$-consistent set of clauses. Let $Y = X$ and repeat the following steps until Y does not change:

- Let R be $(\vee)$ or (K_r). If Y is an instance of the numerator of R then one of the corresponding instances of denominators of R, lets say Z, must be $\mathcal{C}L$-consistent. Set $Y = Z$.
- If Y is an instance of the numerator of the rule $(K4_a)$ then set Y to be the corresponding instance of the second denominator of $(K4_a)$.

It is easily seen that this process always terminates. We call Y a *second kind $\mathcal{C}L$-saturation* of X.

Lemma 3.6.3 *Let X be $\mathcal{C}L$-consistent set of clauses not containing the connective $\blacksquare$. Let Y be a first kind $\mathcal{C}L$-saturation of X, and Z be a second kind $\mathcal{C}L$-saturation of Y. Then the following assertions hold:*

1. *Y and Z are $\mathcal{C}L$-consistent.*
2. *If $[A_1, \ldots, A_k] \in X$ then there exists $1 \leq i \leq k$ such that $A_i \in Y$.*
3. *If $\boxdot[a, \Box b] \in X$ then $\Box b \in Y$ or $\boxdot[a, \Box b] \in Y$.*
4. *If $\boxdot[a, \neg\Box b] \in X$ then $\boxdot a \in Y$ or $\boxdot[a, \neg\Box b] \in Y$.*
5. *If $\phi \in X$ and ϕ is not of any of the forms $[A_1, \ldots, A_k]$, $\boxdot[a, \Box b]$, $\boxdot[a, \neg\Box b]$ then $\phi \in Y$.*
6. *If $\boxdot[a_1, \ldots, a_k] \in Y$ then $a_i \in Z$ for some $1 \leq i \leq k$.*
7. *If $\boxdot[a, \Box b] \in Y$ then $a \in Z$.*
8. *If $\phi \in Y$ and ϕ is not of the form $[A_1, \ldots, A_k]$ then $\phi \in_{\boxplus} Z$.*

The proof of this lemma is straightforward.

In the following algorithm we assume that we have oracles to compute the steps 2b and 4. We will use $CreateANewWorldFrom(w, x)$ to denote the following:

- Let x be a new world, add it to W, and mark it as unsolved.
- Set $R_0 = R_0 \bigcup \{(w, x)\}$.

Algorithm 3.6.4 *Let X be a $\mathcal{C}L$-consistent set of clauses not containing the connectives $\boxdot$, $\blacksquare$. We construct a consistent L-model graph $M = \langle W, \tau, R, H \rangle$ that satisfies X as follows:*

1. *Set $W = \{\tau\}$, $H_0(\tau) = X$, $R_0 = R_1 = \emptyset$, and mark τ as unsolved.*
2. *(a) Take an unsolved world w from W.*
   (b) Let $H_1(w)$ be a first kind $\mathcal{C}L$-saturation of $H_0(w)$.
   (c) For every atom $\neg\Box a$ from $H_1(w)$:
      - *CreateANewWorldFrom(w, w_a).*
      - *Set $H_0(w_a) = \{\neg a\} \bigcup \{\Box Z \mid \Box^2 Z \in H_1(w)\} \bigcup$ $\bigcup \{\boxdot Y \mid \Box Y \in H(w)$ and Y is a set of simple clauses$\}$*

(d) *If the connective* $\Box$ *occurs in* $H_1(w)$ *then:*
For every clause $\phi \in H_1(w)$ *such that* $\phi = \boxdot\psi$*, where* $s \geq 1$ *and* $(\psi = [a, \neg\Box b]$ *or* $\psi = \neg\Box b)$*:*

- *CreateANewWorldFrom*(w, w_ϕ).
- *Set* $H_0(w_\phi) = \{\neg b\} \bigcup \{\Box U \mid \Box^2 U \in H_1(w)\} \bigcup$
  $\{\boxdot Y \mid \boxdot Y \in H_1(w) \vee \Box Y \in H_1(w)$ *and* Y *is a set of simple clauses*$\}$.

(e) *If the connective* $\Box$ *does not occurs in* $H_1(w)$ *then:*

i. *For every clause* $\phi \in H_1(w)$ *such that* $\phi = \boxdot\psi$*, where* $s \geq 1$ *and* $(\psi = [a, \neg\Box b]$ *or* $\psi = \neg\Box b)$*:*
   - *CreateANewWorldFrom*(w, w_ϕ).
   - *Set* $H_0(w_\phi) = \{\neg b, \blacksquare\psi\} \bigcup \{\blacksquare Y \mid \blacksquare Y \in H_1(w)\}$
     $\bigcup \{\boxdot Z \mid \boxdot Z \in H_1(w)\}$.

ii. *For every clause* $\phi \in H_1(w)$ *such that* $\phi = \blacksquare\psi$*, where* $s \geq 1$ *and* $(\psi = [a, \neg\Box b]$ *or* $\psi = \neg\Box b)$*:*
   - *Let* $w_\phi \in W$ *be the node on the path from* τ *to* w *(in the tree generated by* R_0*) such that for any node* v *on the path from* w_ϕ *to* w *we have* $\phi \in H_1(v)$*, and that for* v *being the node immediately above* w_ϕ *we have* $\phi \notin H_1(v)$*, i.e. :*

     $$w_\phi = w \vee R_0^*(w_\phi, w)$$
     $$\forall v\ (v = w_\phi \vee R_0^*(w_\phi, v)) \wedge R_0^*(v, w) \rightarrow \phi \in H_1(v)$$
     $$\forall v\ R_0(v, w_\phi) \rightarrow \phi \notin H_1(v)$$

   - *Set* $R_1 = R_1 \bigcup \{(w, w_\phi)\}$.

(f) *If L is KD4 and there is no* u *such that* $R_0(w, u)$ *or* $R_1(w, u)$ *then*

- *If the connective* $\Box$ *occurs in* $H_1(w)$ *then*
  - *CreateANewWorldFrom*(w, w').
  - *Set* $H_0(w') = \{\Box Z \mid \Box^2 Z \in H_1(w)\} \bigcup$
    $\{\boxdot Y \mid \Box Y \in H_1(w) \vee \boxdot Y \in H_1(w)$ *and* Y *is a set of simple clauses*$\}$.
- *else set* $R_0 = R_0 \bigcup \{(w, w)\}$.

(g) *Mark* w *as resolved.*

3. *While there are unsolved worlds, repeat the step 2.*

4. *For every* $w \in W$*, let* $H(w)$ *be a second kind CL-saturation of* $H_1(w)$.

5. *Set* $R = Ext_{K4}(R_0 \bigcup R_1)$.

At the step 2(e)ii, the node w_ϕ exists because $H_1(\tau)$ does not contain the connective $\blacksquare$. The step 2c corresponds to the rule $(K4')$, the step 2d to the rules $(K4_d)$ and $(K4_e)$, the step 2(e)i to the rules $(K4_b)$ and $(K4_c)$, and the step 2f to the rule $(KD4)$. It follows from Lemma 3.3.3 that this algorithm always terminates.

Lemma 3.6.5 *Consider the step 2(e)ii in the above algorithm. We have* $(\neg b) \in H_1(w_\phi)$.

Proof. Let u be a world such that $R_0(u, w_\phi)$ (such u exists because $H_1(w_\phi)$ contains ϕ while $H_1(\tau)$ does not). It is easily seen that w_ϕ was created from u at the step 2(e)i with respect to $\boxdot\psi$. Therefore $(\neg b) \in H_1(w_\phi)$. $\Box$

Lemma 3.6.6 *Let M be the model graph constructed by Algorithm 3.6.4. M has the following properties:*

1. *For any $w \in W$, if $\blacksquare\phi \in H_1(w)$, where ϕ is a simple clause, then ϕ is of one of the forms $[a, \neg\Box b]$, $\neg\Box a$.*

2. *For any $w, u \in W$ such that $R_1(w, u)$, for any formula ψ, if $\boxplus\psi \in H_1(w)$ then $\boxplus\psi \in H_1(u)$.*

3. *For any $w \in W$, either the connective $\Box$ or $\blacksquare$ does not occurs in $H_1(w)$.*

4. *If $H_1(w)$ contains a formula of the form $\neg\Box a$ then $w = \tau$.*

5. *Neither the connective $\blacksquare$ nor $\boxdot$ occurs in $H_1(\tau)$.*

The proof of this lemma is straightforward.

Lemma 3.6.7 *Let M be the model graph constructed by the Algorithm 3.6.4. Then M is a consistent L-model graph and for any $w \in W$ and $\phi \in H(w)$, $M, w \vDash \phi$.*

Proof. It follows from Lemma 3.6.3:1 that M is consistent. It is clear that R satisfies all L-frame restrictions. Suppose that $\phi \in H(w)$, we show that $M, w \vDash \phi$.

It is easily seen that ϕ cannot be of any of the forms: $[A_1, \ldots, A_k]$ with $k > 1$, $\boxdot[a_1, \ldots, a_k]$, $\boxdot[a, \Box b]$.

Case $\phi = \neg\Box a$: We have $R_0(w, w_a)$ and $(\neg a) \in H_0(w_a)$. Hence $M, w \vDash \phi$.

Case $\phi = \Box a$: It is easily seen that $\phi \in H_1(w)$. By Lemma 3.6.6:3, there is no u such that $R_1(w, u)$. For any $u \in W$ such that $R_0(w, u)$, we have $\boxdot a \in H_0(v)$ and $a \in H(v)$. Hence $M, w \vDash \phi$.

Case $\phi = \blacksquare[a_1, \ldots, a_k]$: Denote $\psi = \boxdot[a_1, \ldots, a_k]$. We claim that for any $u \in W$ such that $\psi \in H_1(u)$ the following assertions hold:

$$\begin{array}{l} M, u \models [a_1, \ldots, a_k] \\ \forall v\ R_0(u,v) \to \psi \in H_1(v) \\ \forall v\ R_1(u,v) \to \psi \in H_1(v) \\ \forall v\ R(u,v) \to (M, v \models [a_1, \ldots, a_k]) \end{array}$$

The first assertion follows from Lemma 3.6.3:6, the second one follows from Lemma 3.6.3:5, the third one follows from Lemma 3.6.6, and the last one follows from the first three ones. Hence $\forall u \in W\ \psi \in H_1(u) \to (M, u \models \psi)$. Since $\phi \in H(w)$, it is easily seen that $\psi \in H_1(w)$. Therefore $M, w \models \phi$.

Case $\phi = \blacksquare[a, \Box b]$: Denote $\psi = \boxdot[a, \Box b]$. We claim that for any $u \in W$ such that $\psi \in H_0(u)$ the following assertions hold:

$$\begin{array}{l} \Box b \in H(u) \lor a \in H(u) \\ \forall v\ R_0(u,v) \to \psi \in H_0(v) \lor \blacksquare b \in H(v) \\ \forall v\ R_1(u,v) \to \psi \in H_0(v) \end{array}$$

The first assertion follows from Lemma 3.6.3, items 3 and 7. For the second assertion: Suppose that $R_0(u,v)$ holds. We have $\psi \in H_1(u)$ or $\Box b \in H_1(u)$, hence $\psi \in H_0(v)$ or $\boxdot b \in H_0(v)$, and finally $\psi \in H_0(v)$ or $\blacksquare b \in H(v)$. For the third assertion: Suppose that $R_1(u,v)$ holds. The connective $\blacksquare$ thus occurs in $H_1(u)$. By Lemma 3.6.6:3, we derive that $\psi \in H_1(u)$. Hence $\psi \in H_1(v)$, by Lemma 3.6.6. Therefore $\psi \in H_0(v)$.

Observe that for any $v \in W$, if $\blacksquare b \in H(v)$ then we have $M, v \models \Box b$ and $M, v \models \boxdot[a, \Box b]$. We conclude that $\forall u \in W\ \psi \in H_0(u) \to (M, u \models \psi)$. Since $\phi \in H(w)$, it is easily seen that $\psi \in H_0(w)$. Therefore $M, w \models \phi$.

Case $\phi = \boxplus[a, \neg\Box b]$: We claim that for any $u \in W$ such that $\phi \in H_1(u)$ the following assertions hold:

$$\begin{array}{l} \exists v\ R(u,v) \land (\neg b) \in H(v) \\ \forall v\ R_0(u,v) \to \boxplus[a, \neg\Box b] \in H_1(v) \lor \boxdot a \in H_1(v) \\ \forall v\ R_1(u,v) \to \boxplus[a, \neg\Box b] \in H_1(v) \end{array}$$

The first assertion is clear. For the second assertion: Suppose that $R_0(u,v)$ holds. We have $\boxplus[a, \neg\Box b] \in H_0(v)$, which implies $\boxplus[a, \neg\Box b] \in H_1(v)$ or $\boxdot a \in H_1(v)$, by Lemma 3.6.3:4. The third assertion follows from Lemma 3.6.6:2. Therefore

$$\begin{array}{l} M, u \models [a, \neg\Box b] \\ \forall v\ R_0(u,v) \to \boxplus[a, \neg\Box b] \in H_1(v) \lor M, v \models \boxdot[a, \neg\Box b] \\ \forall v\ R_1(u,v) \to \boxplus[a, \neg\Box b] \in H_1(v) \end{array}$$

We conclude that $\forall u \in W\ \phi \in H_1(u) \to M, u \models \phi$. Since $\phi \in H(w)$, it is easily seen that $\boxplus[a, \neg\Box b] \in H_1(w)$. Therefore $M, w \models \phi$.

Case $\phi = \boxplus\neg\Box a$: Reasoning similarly as for the preceding case, we derive that for any $u \in W$ such that $\phi \in H_1(u)$, the following assertions hold:

$$\begin{array}{l} M, u \models \neg\Box a \\ \forall v\ R_0(u,v) \to \boxplus\neg\Box a \in H_1(v) \\ \forall v\ R_1(u,v) \to \boxplus\neg\Box a \in H_1(v) \end{array}$$

It follows that $\forall u \in W\ \phi \in H_1(u) \to M, u \models \phi$. Since $\phi \in H(w)$, it is easily seen that $\boxplus\neg\Box a \in H_1(w)$. Therefore $M, w \models \phi$.

Case $\phi = \Box^s\psi$, where $s \geq 1$: We have $\phi \in H_1(w)$. By Lemma 3.6.6:3, there is no u such that $R_1(w,u)$. For any $u \in W$ such that $R_0(w,u)$, we have $(s = 1 \wedge \boxdot\psi \in H_0(u))$ or $(s > 1 \wedge \Box^{s-1}\psi \in H_1(u))$. It is easy to prove by induction that $M, w \models \phi$. □

Corollary 3.6.8 *Let X be a $\mathcal{C}L$-consistent set of clauses not containing the connectives $\boxdot$, $\blacksquare$. Let M be the model graph constructed by Algorithm 3.6.4 for X. Then M is a L-model of X.*

Proof. By Lemma 3.6.7, M is a consistent L-model graph and $M \models H(\tau)$. Since $H(\tau)$ is a second kind $\mathcal{C}L$-saturation of $H_1(\tau)$, which is a first kind $\mathcal{C}L$-saturation of X, by Lemma 3.6.3, items 2, 3, 4, 5, 6, and 8, we conclude that $M \models X$. □

The following theorem immediately follows from Corollaries 3.2.2 and 3.6.8.

Theorem 3.6.9 *The calculi $\mathcal{C}K4$ and $\mathcal{C}KD4$ are sound and complete.*

3.7 Space Bounds for the Logics

In this section, we estimate space bounds for the considered modal logics. We will translate the algorithms given in the preceding section to nondeterministic ones, in order to overcome the assumption about oracles, and then simulate the later by deterministic algorithms.

3.7.1 Space Bounds for *K*, *KD*, *T*, *KB*, *KDB*, and *B*

In this subsection, we use L to denote one of the logics K, KD, T, KB, KDB, and B; and we use $\mathcal{C}L$ to denote the corresponding calculus. We need the following auxiliary lemma:

Lemma 3.7.1 *Let X be an input to Algorithm 3.4.6, with restrictive length n. Then for any node w of the resulting model graph, the restrictive length of $H(w)$ is of order $O(n)$.*

Proof. We define the set $sc(X)$, which contains significant subclauses of X, to be the smallest set satisfying the following. Let $\phi \in X$, $\phi = \Box^s\psi$, where $s \geq 0$ and ψ is a simple clause. Then the followings clauses belong to $sc(X)$: ϕ, $\blacksquare^s\psi$, $\Box\psi$, $\blacksquare\psi$, ψ, and:

- case $\psi = [a_1, \ldots, a_k]$: $a_1, \ldots, a_k$
- case $\psi = [a, \Box b]$: $\Box a$, $\blacksquare a$, a, $\Box b$, $\blacksquare b$, b
- case $\psi = [a, \neg\Box b]$: a, $\neg\Box b$, $\neg b$
- case $\psi = \neg\Box b$: $\neg b$.

Observe that the restrictive length of $sc(X)$ is of order $O(n)$. For any node w of the resulting model graph we have $H(w) \subseteq sc(X)$, hence the restrictive length of $H(w)$ is of order $O(n)$. $\Box$

Definition 3.7.2 Let X and Y are sets of clauses. We say that Y is a candidate for $\mathcal{C}L$-saturation of X if Y can be obtained from X by applying static $\mathcal{C}L$-rules, and Y is $\mathcal{C}L$-saturated (i.e. we reject the condition demanding Y to be $\mathcal{C}L$-consistent).

We translate Algorithm 3.4.6 to a nondeterministic one, to overcome the assumption about the oracle, by ignoring the computation of W and R, and replacing the step 2b by:

"Nondeterministically generate a candidate for $\mathcal{C}L$-saturation of $H_0(w)$, and assign it to $H(w)$. If $H(w)$ is inconsistent in the sense that it contains $\perp$, then reject the computation."

Observe that for any set X of clauses not containing the connective $\blacksquare$, X is L-satisfiable iff the new algorithm has an unrejected computation for X.

We then simulate the obtained algorithm by a deterministic one, denoted by $\mathcal{A}$, by backtracking. During computation of $\mathcal{A}$ we only need to keep information about the current path from the root τ to the current node. We record all and-or branching on the current path, and keep the contents of the root node and the current node.

Let n and s be the restrictive length and the modal depth of X, respectively, and assume that $s \leq n$. The contents of the root node and the current node can be coded using $O(n.\log n)$-space; each sequence $\boxtimes^k$ takes $O(\log n)$-space, each connective $O(1)$-space, and each primitive proposition $O(\log n)$-space.

We do not keep contents of other nodes on the current path as they can be reconstructed from the content of the root node and the information about and-or branching on the path. Let u be a node on the current path. Suppose that w is a predecessor of u, i.e. $R_0(w, u)$, and $H(w)$ has been reconstructed.

The set $H(u)$ is reconstructed as follows. Using the information about and-branching from w to u, we compute the set $H_0(u)$, as defined in the steps 2d and 2e of Algorithm 3.4.6. We then repeatedly apply static $\mathcal{C}L$-rules to $H_0(u)$ accordingly to the information about or-branching at u until no more changes occur. The resulting set is $H(u)$. Note that the order in which we apply static $\mathcal{C}L$-rules to $H_0(u)$ is not important; in any order we obtain the same set. The task of computing $H(u)$ from $H(w)$ requires $O(n. \log n)$-space (mainly to store $H(u)$). Hence the content of any node on the current path can be reconstructed in $O(n. \log n)$-space.

When and-branching from one node to another in the search tree, we use $O(\log n)$-space to record the position of the clause $\neg\Box a$ (see the step 2d of Algorithm 3.4.6) in the *sequent* X, treating sequences of $\boxtimes^k$ as primitive connectives. The length of every path is bounded by s, hence we need $O(s. \log n)$-space to keep all information about and-branching on the current path.

Case $L = K$ or $L = KD$: Each formula of the form $\Box^k[A_1, \ldots, A_h]$, where $k \geq 0$ and $h > 1$, causes or-branching at the depth k by choosing some A_i, $1 \leq i \leq h$. For each formula of that form, we need $O(\log n)$-space to keep the index i. Therefore we need $O(n. \log n)$-space to keep all information about or-branching on the current path. Summing up, when $L = K$ or $L = KD$, the algorithm $\mathcal{A}$ runs in $O(n. \log n)$-space.

Case $L \in \{T, KB, KDB, B\}$: We claim that for any $w \in W$ all information about or-branching at w can be coded in $O(n)$-space. We reserve two bits for every sequence of the form $\Box^k$, and for every other symbol occurring in $H_0(w)$. Thus we have an array of length $O(n)$ of pieces of two bits. To keep information about or-branching caused by a clause of the form $\Box^k[a, \Box b]$, $k \geq 0$, we write into the piece reserved for the sequence $\Box^k$ (or for the comma if $k = 0$) the position (1, 2, or 3) of the chosen denominator. To keep information about or-branching caused by a clause of the form $[A_1, \ldots, A_h]$, but not of the form $[a, \Box b]$, we set the piece reserved for the chosen disjunct to 1, and the other pieces to 0. Since the length of every path is bounded by s, we conclude that it needs $O((s+1).n)$-space to keep all information about or-branching on the current path. Summing up, when $L \in \{T, KB, KDB, B\}$, the algorithm $\mathcal{A}$ runs in $O(n.(s + \log n))$-space.

Theorem 3.7.3 *The logics K and KD are decidable in $O(n. \log n)$-space, and the logics T, KB, KDB, and B are decidable in $O(n^2)$-space. The problem of checking satisfiability in T, KB, KDB, or B for formulae with finitely bounded modal depth is decidable in $O(n. \log n)$-space. (Here n is the length of an input formula.)*

The space bound $O(n. \log n)$ for the logic K was first established by Hudelmaier [15], and for the logic KD by Basin, Matthews, and Viganò [1].

Proof. It suffices to recall that for any formula ϕ with length n and modal depth s (here we have $s < n$), we can use $O(n. \log n)$-space to translate ϕ to an equisatisfiable, in any normal modal logic, set X of clauses such that the restrictive length of X is linearly bounded by n, and the modal depth of X is equal to s. □

3.7.2 A Space Bound for $S4$

We define a calculus $\mathcal{C}S4'$ as $\mathcal{C}S4$ with all occurrences of $\blacksquare$ in all of the rules replaced by $\Box$. The calculus $\mathcal{C}S4'$ is a traditional one, which does not use extra modalities and does not have the finite tableau property either.

It is easy to show that $\mathcal{C}S4'$ is sound.

Let the first and the second kind of $\mathcal{C}S4'$-saturation be defined in the same way as for $\mathcal{C}S4$, but with respect to $\mathcal{C}S4'$. Consider the following algorithm, which is a modified version of Algorithm 3.5.4:

Algorithm 3.7.4
Input: A $\mathcal{C}S4'$-consistent set X of clauses[1].
Steps:

1. *Set* $W = \{\tau\}$, $H_0(\tau) = X$, $R_0 = R_1 = \emptyset$, *and mark* τ *as unsolved.*
2. (a) *Take an unsolved world* w *from* W.
   (b) *Let* $H_1(w)$ *be a first kind* $\mathcal{C}S4'$*-saturation of* $H_0(w)$.
   (c) *For every atom* $\neg\Box a$ *from* $H_1(w)$:
      - *Create a new world* w_a, *add it to* W, *and mark it as unsolved.*
      - *Set* $R_0 = R_0 \bigcup \{(w, w_a)\}$.
      - *Set* $H_0(w_a) = \{\neg a\} \bigcup \{\Box Y | \Box Y \in H_1(w)\}$.
   (d) *For every clause* $\phi \in H_1(w)$ *such that* $\phi = \Box^s \psi$, *where* $s \geq 1$ *and* ($\psi = [a, \neg\Box b]$ *or* $\psi = \neg\Box b$):
      - *Let* $Y = \{\neg b\} \bigcup \{\Box Z \mid \Box Z \in H_1(w)\}$.
      - *If there exists a world* $w_\phi \in W$ *such that* $w_\phi = w$ *or* $R_0^*(w_\phi, w)$ *and* $Y \subseteq H_1(w_\phi)$, *then set* $R_1 = R_1 \bigcup \{(w, w_\phi)\}$;
      - *else*
        - *Create a new world* w_ϕ, *add it to* W, *and mark it as unsolved.*
        - *Set* $R_0 = R_0 \bigcup \{(w, w_\phi)\}$, *and* $H_0(w_\phi) = Y$.
   (e) *Mark* w *as resolved.*
3. *While there are unsolved worlds, repeat the step 2.*

[1] In this subsection we assume that the language does not contain the connectives $\blacksquare$ and $\boxdot$

4. *For every $w \in W$, let $H(w)$ be a second kind $\mathcal{C}S4'$-saturation of $H_1(w)$.*

5. *Set $R = Ext_{S4}(R_0 \bigcup R_1)$.*

Lemma 3.7.5 *Let X be a $\mathcal{C}S4'$-consistent set of clauses. Let M be the model graph constructed by Algorithm 3.7.4 for X. Then M is a $S4$-model of X.*

The proof of this lemma is very similar to the proof of Corollary 3.5.8, so we omit it; see [25] for details. As a consequence, the calculus $\mathcal{C}S4'$ is sound and complete.

To estimate space bound for $S4$ we need the following lemma:

Lemma 3.7.6 *Let X be a $\mathcal{C}S4'$-consistent set of clauses. Let n be the restrictive length of X. Let R_0 be the relation computed by Algorithm 3.7.4 for X. Then R_0 is a tree with depth bounded by n.*

Proof. Suppose that the depth of R_0 is greater than n. It is easily seen that there exist two different worlds w, u such that: $R_0^*(\tau, w)$, $R_0^*(w, u)$, $H_1(w) = \{\neg b\} \bigcup Y$, $H_1(u) = \{\neg b\} \bigcup Z$, for some b, Y, Z such that $Y \neq Z$ and Y and Z contain only clauses of the form $\Box^s \phi$, where $s \geq 1$. Since $Y \neq Z$, either the rule $(S4_a)$ or $(S4_b)$ must be applied on the path from w to u. Observe that if $\Box^s \phi \in H_1(w)$, where $s \geq 1$, then:

- If ϕ is of the form $[a_1, \ldots, a_k]$ then $\Box^s \phi \in H_1(u)$.
- If ϕ is of the form $[a, \Box b]$ then either $\Box^s \phi$ or $\Box b$ belongs to $H_1(u)$.
- If ϕ is of the form $[a, \neg \Box b]$ then either $\Box^s \phi$ or $\Box a$ belongs to $H_1(u)$.

This contradicts the fact that $H_1(w)$ is a first kind $\mathcal{C}S4'$-saturation of $H_0(w)$. □

Consider the nondeterministic algorithm obtained from Algorithm 3.7.4 by deleting the steps 4 and 5, ignoring the computation of W and R_1, and replacing the step 2b by:

- Nondeterministically choose a candidate for first kind $\mathcal{C}S4'$-saturation of $H_0(w)$ and assign it to $H_1(w)$.
- If all candidates for second kind $\mathcal{C}S4'$-saturation of $H_1(w)$ are inconsistent in the sense that they contain both p and $\neg p$ for some p, or if the depth of w in the tree R_0 is greater than n, then reject the computation.

It is easily seen that for any set X of clauses, X is $S4$-satisfiable iff the new algorithm has an unrejected computation for X.

We can simulate the mentioned algorithm by a deterministic one, denoted by $\mathcal{B}$, by backtracking. During computation of $\mathcal{B}$ for X we only need to keep information about the current path from the root τ to the current world.

We will treat sequences of $\Box^s$ as primitive connectives. Let n be the restrictive length of X. Without difficulty one can show that for any node w of the constructed model graph, the restrictive lengths of $H_0(w)$ and $H_1(w)$ are of order $O(n)$. For each world on the current path, we need $O(\log n)$-space to keep the information about and-branching from its parent to it, i.e. to keep the position either of the atom $\neg\Box a$, in the case of the step 2c, or of the clause ϕ, in the case of the step 2d, in the sequential display of X. For each formula ϕ of the form $\Box^s[a, \Box b]$ (resp. $\Box^s[a, \neg\Box b]$), where $s \geq 1$, we need $O(\log n)$-space to keep the depth at which it is replaced by $\Box b$ (resp. $\Box a$) as a result of applying the (modified) rule $(S4_a)$ (resp. $(S4_b)$), i.e. to keep the information about or-branching caused by the formula. Note that formulae of the form $[A_1, \ldots, A_k]$, where $k > 1$, can occur only in the root τ.

We do not keep the sets $H_0(w)$ and $H_1(w)$, except for w being the current node or the root node. For any node w on the current path, from the content of the root node and the information about and-or branching (on whole of the path) we can reconstruct the sets $H_0(w)$ and $H_1(w)$, using $O(n. \log n)$-space. For this, it suffices to show that for any worlds w and u on the current path such that $R_0(w, u)$, having $H_1(w)$ and the information about and-or branching we can construct $H_0(u)$ and $H_1(u)$ in $O(n. \log n)$-space. It is clear that having $H_1(w)$ and the information about and-branching from w to u we can construct $H_0(u)$ using $O(n. \log n)$-space. Let k be the depth of u in the tree R_0. We have $1 \leq k \leq n$. The set $H_1(u)$ is obtained from $H_0(u)$ by replacing every clause of the form $\Box^s[a, \Box b]$ (resp. $\Box^s[a, \neg\Box b]$) - for which there is information that it is replaced by $\Box b$ (resp. $\Box a$) at the depth k - by $\Box b$ (resp. $\Box a$). The task can be done in $O(n. \log n)$-space.

For the two special worlds, the root and the current, we can use $O(n. \log n)$-space to keep information about them, and to test consistency of candidates for second kind $\mathcal{C}S4'$-saturation of the current world. Since every path has depth bounded by n, we conclude that the algorithm $\mathcal{B}$ can be computed in $O(n. \log n)$-space.

Theorem 3.7.7 *The logic S4 is decidable in $O(n. \log n)$-space.*

Proof. It suffices to recall that using $O(n. \log n)$-space we can translate any formula Φ to a $S4$-equisatisfiable set X of clauses such that the restrictive length of X is linearly bounded by the length of Φ. $\Box$

3.7.3 A Space Bound for $K4$ and $KD4$

We define a calculus $\mathcal{C}K4'$ (resp. $\mathcal{C}KD4'$) as $\mathcal{C}K4$ (resp. $\mathcal{C}KD4$) with all occurrences of $\blacksquare$ in all of the rules replaced by $\boxdot$.

It is easy to show that the calculi $\mathcal{C}K4'$ and $\mathcal{C}KD4'$ are sound.

In this subsection we use L to denote $K4$ or $KD4$, and write $\mathcal{C}L$ and $\mathcal{C}L'$ to denote the corresponding calculi.

Let the first and the second kind of $\mathcal{C}L'$-saturation be defined in the same way as for $\mathcal{C}L$, but with respect to $\mathcal{C}L'$.

Let $\mathcal{C}$ be the algorithm obtained from Algorithm 3.6.4 by replacing the term *first* (resp. *second*) *kind $\mathcal{C}L$-saturation* by *first* (resp. *second*) *kind $\mathcal{C}L'$-saturation*, and replacing the step 2e by:

> If the connective $\Box$ does not occur in $H_1(w)$ then:
> For every clause $\phi \in H_1(w)$ such that $\phi = \boxdot\psi$, where $s \geq 1$ and $(\psi = [a, \neg\Box b]$ or $\psi = \neg\Box b)$:
>
> - Let $Y = \{\neg b, \boxdot\psi\} \bigcup \{\boxdot Z \mid \boxdot Z \in H_1(w)\}$.
> - If there exists a world $w_\phi \in W$ such that $w_\phi = w$ or $R_0^*(w_\phi, w)$, and $Y \subseteq H_1(w_\phi)$, then set $R_1 = R_1 \bigcup \{(w, w_\phi)\}$;
> - else
>   - CreateANewWorldFrom(w, w_ϕ).
>   - Set $H_0(w_\phi) = Y$.

Lemma 3.7.8 *Let X be a $\mathcal{C}L'$-consistent set of clauses. Let M be the model graph constructed by the algorithm $\mathcal{C}$ for X. Then M is a L-model of X.*

The proof of this lemma is very similar to the proof of Corollary 3.6.8, so we omit it. As a consequence, the calculi $\mathcal{C}K4'$ and $\mathcal{C}KD4'$ are sound and complete.

Similarly as for $\mathcal{C}S4'$, we have the following lemma:

Lemma 3.7.9 *Let X be a $\mathcal{C}L'$-consistent set of clauses. Let n and s be the restrictive length and the modal depth of X, respectively. Let R_0 be the relation computed by the algorithm $\mathcal{C}$ for X. Then R_0 is a tree with depth bounded by $2n + s$.*

Proof. On any path of the tree R_0, there is at most one world created at the step 2c (with $w = \tau$), and at most n worlds created at the steps 2d or 2f, with depth greater than s. Hence the number of worlds created at the steps 2c, 2d and 2f on any path is bounded by $s + n$. Suppose that the depth of R_0 is greater than $2n + s$. There then exist two different worlds w, u such that: $R_0^*(\tau, w)$, $R_0^*(w, u)$, $H_1(w) = \{\neg b\} \bigcup Y$, $H_1(u) = \{\neg b\} \bigcup Z$, for some b, Y, Z such that $Y \neq Z$ and Y and Z contain only clauses of the form $\boxdot\phi$, where ϕ is a simple clause. Since $Y \neq Z$, either the rule $(K4_a)$ or $(K4_b)$ must be applied on the path from w to u. Observe that if $\boxdot\phi \in H_1(w)$ then:

- If ϕ is of the form $[a_1, \ldots, a_k]$ then $\boxdot\phi \in H_1(u)$.
- If ϕ is of the form $[a, \Box b]$ then either $\boxdot\phi$ or $\boxdot b$ belongs to $H_1(u)$.
- If ϕ is of the form $[a, \neg\Box b]$ then either $\boxdot\phi$ or $\boxdot a$ belongs to $H_1(u)$.

This contradicts the fact that $H_1(w)$ is a first kind $\mathcal{C}L'$-saturation of $H_0(w)$. $\Box$

Consider the nondeterministic algorithm obtained from the algorithm $\mathcal{C}$ by deleting the steps 4 and 5, ignoring the computation of W and R_1, and replacing the step 2b by:

- Nondeterministically choose a candidate for first kind $\mathcal{C}L'$-saturation of $H_0(w)$ and assign it to $H_1(w)$.
- If all candidates for second kind $\mathcal{C}L'$-saturation of $H_1(w)$ are inconsistent in the sense that they contain both p and $\neg p$ for some p, or if the depth of w in the tree R_0 is greater than $2n+s$, where $n = rlength(X)$ and $s = mdepth(X)$, then reject the computation.

It is easily seen that for any set X of clauses, X is L-satisfiable iff the new algorithm has an unrejected computation for X.

We can simulate the mentioned algorithm by a deterministic one, denoted by $\mathcal{D}$, by backtracking. During computation of $\mathcal{D}$ for X we only need to keep information about the current path from the root τ to the current world. For each world in the current path we need $O(\log n)$-space to keep the information about and-branching from its parent to it. For each formula ϕ of the form $\boxdot[a, \Box b]$ (resp. $\boxdot[a, \neg\Box b]$), we need $O(\log(n+s))$-space to keep the information about the depth at which it is replaced by $\Box b$ (resp. $\boxdot a$), i.e. to keep the information about or-branching caused by the formula. Note that formulae of the form $[A_1, \ldots, A_k]$, where $k > 1$, can occur only in the root τ. For the two special worlds, the root and the current, we can use $O(n.\log(n+s))$-space to keep information about them, and to test consistency of candidates for second kind $\mathcal{C}L'$-saturation of the current world. Since every path has depth bounded by $2n+s$, we conclude that the algorithm $\mathcal{D}$ can be computed in $O((n+s).\log(n+s))$-space.

Theorem 3.7.10 *The logics K4 and KD4 are decidable in $O(n.\log n)$-space.*

Proof. Let Φ be an arbitrary formula, and let $n = length(\Phi)$. Using $O(n.\log n)$-space, we can translate Φ to an equisatisfiable, in any normal modal logic, set X of clauses such that the restrictive length and the modal depth of X are linearly bounded by n. Therefore we can check whether Φ is $K4$-satisfiable (resp. $KD4$-satisfiable) using $O(n.\log n)$-space. $\Box$

3.8 Comparison with Other Work

While the work of Hudelmaier [15] and the works of Basin, Matthews and Viganò [1, 34, 2] base on sequent systems, our work bases on tableau systems. The proofs of completeness of the sequent systems deal with proof theory, the proof of completeness of our tableau systems deal with constructing model graphs. At this aspect, which method is preferable is a matter of taste.

Hudelmaier [14, 15] was the first pioneer who concentrated in estimating upper space bounds for modal logics. The idea of translating formulae to clausal form and defining deductive systems only for sets of clauses is due to Hudelmaier. Our connective $\blacksquare$ has the same role as the connective $\bigcirc$ introduced by Hudelmaier in [14, 15]. Our systems $\mathcal{C}K$, $\mathcal{C}T$ and $\mathcal{C}S4$ are very similar to the ones of his. The differences lie in proofs of completeness and in analysis of space requirement for the logics.

As we have stated, our space bounds for the logics $K4$, $KD4$, and $S4$ are lower than the previously established ones. We are the first who explicitly establish space requirement for the logics KB, KDB and B. The space bound $O(n.\log n)$ for the logic K was first established by Hudelmaier [15], and later also proved by Basin et al [1]. The space bound $O(n.\log n)$ for the logic KD is due to Basin et al [1].

In [15] Hudelmaier gave the space bound $O(n.\log n)$ for the logic K and T. However, he has accepted some wrong assumption. Namely, he claimed that when a formula is translated to clausal form, linearly many connectives are added to the output sequent. The correct statement should be that when a formula of length n is translated to clausal form, the number of connectives in the output sequent is of order $O(n^2)$. To see this, consider formulae $\phi_n = \Box^n(\Box p_1 \vee \Box p_2 \vee \ldots \vee \Box p_n)$. By Mints translation, ϕ_n is first translated to $(\Box^n[\Box p_1, q_1];\ \Box^n[\neg q_1, \Box p_2, q_2];\ \ldots;\ \Box^n[\neg q_{n-2}, \Box p_{n-1}, q_{n-1}];$ $\Box^n[\neg q_{n-1}, \Box p_n])$, and then to $(\Box^n[q_1, \Box p_1]; \Box^n[\neg q_1, q_2, r_2]; \Box^n[\neg r_2, \Box p_2]; \ldots;$ $\Box^n[\neg q_{n-2}, q_{n-1}, r_{n-1}]; \Box^n[\neg r_{n-1}, \Box p_{n-1}]; \Box^n[\neg q_{n-1}, \Box p_n])$. Denote the later sequent by ψ_n. Now note that while the number of connectives in ϕ_n is of order $O(n)$, the number of connectives in ψ_n is greater than n^2 (note that the number of connectives in $\Box^n$ is n). Therefore "the maximal length of a deduction" of a sequent in $C_1(T)$ is of order $O(n^2)$, and space bound obtained by Hudelmaier method for the logic T should be $O(n^2.\log n)$. For the logic K, the space bound $O(n.\log n)$ for the algorithm given by Hudelmaier is true, however, the proof needs a correction. A possible solution is as follows:

"Define $d(s) = rlength(s) + mdepth(s)$. When a formula ϕ is translated to clausal form, to a sequent X, $d(X)$ is linearly bounded by the length of ϕ. The measure d decreases under backwards application of the rule $CK\Box_0$. Thus every deduction of a sequent in $C_0(K)$ has length of order $O(n)$."

In this work, we have introduced *restrictive length* of a set of clauses in order to overcome the problem.

In [1], Basin, Matthews and Viganò gave the space bound $O(n.\log n)$ for the logic T. In a later work [2] by the same authors, only space bound $O(n^2.\log n)$ is proved for T. In this work, we have given a decision procedure for T that runs in $O(n^2)$-space. If we concentrate only on estimating space bounds for modal logics, then our presentation is much shorter than the work on this problem by Basin et al [2].

3.9 Conclusions

We have presented clausal tableau systems for the common propositional modal logics K, KD, T, KB, KDB, B, $K4$, $KD4$, and $S4$. For each of these systems there is a measure such that for each tableau rule of the system the measure of all its denominators is smaller than the measure of its numerator. This property allows depth-first search without loop checking. Basing on the systems, we have given improved decision procedures for the considered logics, which use $O(n.\log n)$-space for the logics K, KD, $K4$, $KD4$, and $S4$, and $O(n^2)$-space for the logics T, KB, KDB, and B. We have also shown that the problem of checking satisfiability in T, KB, KDB, or B for formulae with finitely bounded modal depth is decidable in $O(n.\log n)$-space. We are the first who explicitly estimates space requirement for the logics KB, KDB, and B. Our space bounds for the logics $K4$, $KD4$, and $S4$ are lower than those hitherto known. Our space bounds for the logics K and KD are equal to the ones previously established by Hudelmaier [15] and Basin et al [1].

Although our decision procedures for the considered modal logics have some nice properties, it cannot change the fact that provability in such logics is PSPACE-complete. For real applications, we must select fragments of modal logics that are computable in PTIME, and develop decision procedures for them. The next two chapters deal with this aim.

Chapter 4

Constructing the Least Models for Positive Propositional Modal Logic Programs

It is well known that for any positive classical logic program P there exists the least model of P. Moreover, the least model can be constructed in PTIME, and its size is bounded by a polynomial in the size of P. How can we extend these results for modal logics ?

In [18] Ladner showed that the complexity of provability in the logics K, T, and $S4$ is PSPACE-complete, in the logic $S5$ is co-NP-complete. We can raise a question: What fragments of modal logics are computable in polynomial time? Some authors have studied the case [7, 4, 12]. In [7] Fariñas del Cerro and Penttonen showed that the problem of checking satisfiability of a set of modal Horn formulae in $S5$ is decidable in PTIME. In [4] Chen and Lin showed that the similar problem for a normal modal logic L which is a normal extension of $K5$, write $K5 \leq L$, is also solvable in PTIME. Chen and Lin also proved that for a normal modal logic L such that $K \leq L \leq S4$ or $K \leq L \leq B$, the problem is PSPACE-complete. In [12] Halpern showed that if we restrict to finite modal depth, and restrict the language to having only finitely many primitive propositions, then the complexity of the problem of checking satisfiability of a set of modal formulae goes down to linear time for the modal logics K, T, and $S4$. Although the latter restriction is very strong, the idea is interesting. In this paper, we show that the combination of restriction to Horn formulae and restriction to having finitely bounded modal depth gives polynomial time complexity for the logics KD, T, KB, KDB, and B. As mentioned in Chapter 2, there are correspondences between Horn formulae and positive logic programs.

In this paper, we show that for any positive modal logic program P

in a modal logic $L \in \{KD, T, KDB, B, KD5, KD45, S5, KD4, S4\}$, there exists the least L-model of P. The above listed modal logics are, as a characterization, serial and their frame restrictions, excluding D, are Horn formulae. For the modal logics KB, $K5$, $K45$, and $KB5$, which are *almost serial* (see Definition 4.3.2), we show that if P has no *flat* models (see Definition 4.3.1) then P has the least model in any such logic. The algorithms of constructing the least models for positive modal logic programs are very useful from the point of view of modal deductive databases.

The idea of how to construct the least model for a positive modal logic program P in a modal logic L is the following: We construct a L-model graph realizing P (the method of constructing model graphs can be found in [30, 11]). To guarantee the constructed model graph to be the smallest, for each created world, we connect it, at the time of its creation, to an empty world.

In this chapter, we only deal with *propostional* modal logics and *propositional* modal logic programs. We will omit the word "propositional" for convenience.

4.1 Ordering Kripke Models

In this section, we define an order between Kripke models. The order has the property that if a model M is less than or equal to N, then for any positive formula ϕ, if ϕ is satisfied in M then ϕ is also satisfied in N.

Definition 4.1.1 Let $M = \langle W, \tau, R, h\rangle$ and $N = \langle W', \tau', R', h'\rangle$ be Kripke models. We say that M is less than or equal to N wrt. a binary relation $r \subseteq W \times W'$, and write $M \leq N$ wrt. r, if:

1. $r(\tau, \tau')$
2. $\forall x, x' \;\; r(x, x') \rightarrow (h(x) \subseteq h'(x'))$
3. $\forall x, x', y \;\; R(x, y) \wedge r(x, x') \rightarrow \exists y' \; R'(x', y') \wedge r(y, y')$
4. $\forall x, x', y' \;\; R'(x', y') \wedge r(x, x') \rightarrow \exists y \; R(x, y) \wedge r(y, y')$

We say that a model M is less than or equal to N, and write $M \leq N$, if $M \leq N$ wrt. some r.

Definition 4.1.2 Let P be a positive program in a normal modal logic L. We say that M is the least L-model of P if M is a L-model of P and M is less than or equal to every L-model of P.

Lemma 4.1.3 *If $M \leq N$ wrt. r, then for every positive formula ϕ and for every u, u' such that $r(u, u')$, if $M, u \vDash \phi$ then $N, u' \vDash \phi$. In particular if $M \leq N$, then for every positive formula ϕ, if $M \vDash \phi$ then $N \vDash \phi$.*

Proof. Let $M \leq N$ wrt. r. We prove the lemma by induction on the construction of ϕ. Suppose that $r(u, u')$ and $M, u \vDash \phi$ hold.

Case $\phi = p$: We have $p \in h(u)$. By the condition 2, we have $p \in h'(u')$, hence $N, u' \vDash p$.

The case $\phi = \psi \wedge \zeta$ or $\phi = \psi \vee \zeta$ is trivial.

Case $\phi = \Box\psi$: By the condition 4, we have $\forall v' \quad R'(u', v') \rightarrow \exists v \; r(v, v') \wedge R(u, v) \wedge (M, v \vDash \psi)$. By the assumption of induction, we have $\forall v' \; R'(u', v') \rightarrow (N, v' \vDash \psi)$, hence $N, u' \vDash \Box\psi$.

Case $\phi = \Diamond\psi$: Since $M, u \vDash \Diamond\psi$, it follows that $\exists v \; R(u, v) \wedge (M, v \vDash \psi)$. By the condition 3, we have $\exists v, v' \; R'(u', v') \wedge r(v, v') \wedge (M, v \vDash \psi)$. By the assumption of induction, it yields that $\exists v' \; R'(u', v') \wedge (N, v' \vDash \psi)$. Therefore $N, u' \vDash \Diamond\psi$. □

Proposition 4.1.4 *The relation $\leq$ between Kripke models is a pre-order.*

Proof. Let $M = \langle W, \tau, R, h\rangle$, $M' = \langle W', \tau', R', h'\rangle$, and $M'' = \langle W', \tau', R', h'\rangle$ be Kripke models. We have $M \leq M$ wrt. the identity relation. Let $M \leq M'$ wrt. r_1, and $M' \leq M''$ wrt. r_2. Let $r_3 = r_1 * r_2$. We show that $M \leq M''$ wrt. r_3.

It is obvious that $r_3(\tau, \tau'')$ and $\forall u, u'' \; r_3(u, u'') \rightarrow (h(u) \subseteq h''(u''))$.

For the condition 3 : Suppose that $R(u, v) \wedge r_3(u, u'')$ holds. By the definition of r_3, there exists u' such that $R(u, v) \wedge r_1(u, u') \wedge r_2(u', u'')$. By Definition 4.1.1, the following formulae hold:

$$\begin{array}{l}
\exists v' \; R'(u', v') \wedge r_1(v, v') \wedge r_2(u', u'') \\
\exists v'' \; R''(u'', v'') \wedge r_2(v', v'') \wedge r_1(v, v') \\
\exists v'' \; R''(u'', v'') \wedge r_3(v, v'').
\end{array}$$

For the condition 4 : Suppose that $R''(u'', v'') \wedge r_3(u, u'')$ holds. By the definition of r_3 there exists u' such that $R''(u'', v'') \wedge r_1(u, u') \wedge r_2(u', u'')$. By Definition 4.1.1, the following formulae hold:

$$\begin{array}{l}
\exists v' \; R'(u', v') \wedge r_2(v', v'') \wedge r_1(u, u') \\
\exists v \; R(u, v) \wedge r_1(v, v') \wedge r_2(v', v'') \\
\exists v \; R(u, v) \wedge r_3(v, v'').
\end{array}$$

□

4.2 Constructing the Least Models for Positive Programs in *KD*, *T*, *KDB*, *B*, *KD*5, *KD*45, and *S*5

In this section we give an algorithm that for a given positive modal logic program P will construct the least L-model for P, where L can be one of the modal logics KD, T, KDB, B, $KD5$, $KD45$, and $S5$. The complexity of the algorithm will be analyzed, and some corollaries will be presented.

To illustrate the idea of the algorithm, let us consider an example. Let $P = \{\Diamond p, \Diamond q\}$ be a positive program. $M = \langle \{\tau, u, v\}, \tau, R, h \rangle$, with $R = \{(\tau, u), (\tau, v), (u, u), (v, v)\}$, $h(\tau) = \{\}$, $h(u) = \{p\}$, and $h(v) = \{q\}$, is a KD-model of P. But M is not the least KD-model of P. Why not? The reason is that $M \vDash \Box(p \vee q)$ but $\Diamond p \wedge \Diamond q \nvDash_{KD} \Box(p \vee q)$. How can we change M to make it the least KD-model of P? First, note that if we connect τ to a new empty world, then M no longer satisfies $\Box(p \vee q)$. But it still satisfies, for example, $\Diamond \Box p$, which does not follow from $\Diamond p \wedge \Diamond q$ in KD. Now connect u to a new empty world, and note that M no longer satisfies $\Diamond \Box p$. Generalizing these observations, one can hope that connecting each newly created world to an empty world is the key to construct the least models for positive modal logic programs.

The method of constructing the least L-model for a positive program P is the following: We construct a L-model graph for P. Formulae from contents of worlds of the model graph are treated as requirements to be realized. We solve formulae of the form $(B \leftarrow C_1, \ldots, C_k)$ or $\Box \psi$ in a usual way (see Algorithm 4.2.1). To realize a formula $\Diamond \psi$ at a world u we connect u to a new world containing ψ. To guarantee the model graph to be the smallest, we connect each newly created world to an infinite chain of empty worlds. For the logics $KD5$, $KD45$, and $S5$, these infinite chains can be replaced by two special worlds. From time to time, we extend the accessibility relation in the way to satisfy the frame restrictions.

In the algorithm given below, we use the following data structures:

- W : a set of worlds,
- $R : W \times W$ - the accessibility relation,
- H : a function that maps each world $u \in W$ to a set of formulae which should hold at u.

We will use M to denote the model graph $\langle W, \tau, R, H \rangle$. We sometimes refer to M as the *model* $\langle W, \tau, R, h \rangle$, with $h(x) = \{p : p$ is a primitive proposition belonging to $H(x)\}$. We write $M, u \vDash \phi$ to mean that the formula ϕ is satisfied at the world u in the *model* M; and write $M \leq N$ to denote that the *model* M is less than or equal to N.

We will use the procedure $CreateEmptyTail_L(x_0)$, which is defined as follows:

- If L is one of the logics KD, T, KDB, and B, then let $(x_1, x_2, \ldots)$ be an infinite chain of new empty worlds, add them to W and set $R = R \bigcup \{(x_i, x_{i+1}) \mid i \geq 0\}$. (Note that the chain can be coded as a finite chain, which will be dynamically expanded when necessary.)

- If L is one of the logics $KD5$, $KD45$, and $S5$, and x_0 is τ, then let ρ and ϖ be new empty worlds, add them to W, and add the edges (τ, ρ) and (ρ, ϖ) to R.

Algorithm 4.2.1
Input: *A positive modal logic program* P
in a modal logic $L \in \{KD, T, KDB, B, KD5, KD45, S5\}$
Output: *The least* L*-model* $M = \langle W, \tau, R, h \rangle$ *of* P.
Steps:

1. *Set* $W = \{\tau\}$, $H(\tau) = P$, $R = \emptyset$.
   $CreateEmptyTail_L(\tau)$.
   Set $R = Ext_L(R)$.

2. *For every world* $u \in W$ *with* $H(u)$ *not empty, and for every formula* $\phi \in H(u)$:

   (a) *Case* $\phi = (B \leftarrow C_1, \ldots, C_k)$, *where* $k \geq 1$, *and* $M, u \vDash C_i$ *for all* $1 \leq i \leq k$: *Set* $H(u) = H(u) \bigcup \{B\}$.

   (b) *Case* $\phi = \Box\psi$: *For every world* $v \in W$ *satisfying* $R(u, v)$, *set* $H(v) = H(v) \bigcup \{\psi\}$.

   (c) *Case* $\phi = \Diamond\psi$, *and* $\neg(\exists x\ R(u, x) \wedge \psi \in H(x))$:

      Let v *be a new world.*
      Set $W = W \bigcup \{v\}$, $H(v) = \psi$, $R = R \bigcup \{(u, v)\}$.
      $CreateEmptyTail_L(v)$.
      Set $R = Ext_L(R)$.

3. *While some change occurred, repeat the step 2.*

We give below two auxiliary lemmas. The first one is about frame properties of the model graph constructed by the above algorithm. The second one is needed for the proof of Lemma 4.2.4.

Lemma 4.2.2 *The following assertions are invariant properties of Algorithm 4.2.1 (wrt. the numerated steps):*

1. R *satisfies all* L*-frame axioms.*

2. *Every* x *in* W *is directly or indirectly reachable from the root* τ, *i.e.* $\forall x \in W\ x = \tau \vee R^*(\tau, x)$.

The proof of this lemma is straightforward. When $KD5 \leq L$, the frame of the model graph satisfies the condition $\forall x, y \in W\ (x \neq \tau \wedge y \neq \tau) \rightarrow R(x, y)$. Moreover, if $L = S5$ then the frame satisfies $\forall x, y \in W\ R(x, y)$; if $L = KD45$ then the frame satisfies $\forall x \in W\ x \neq \tau \rightarrow R(\tau, x)$.

Lemma 4.2.3 *Consider the step 2c. Let R denote itself at the beginning of execution of the step, and let R_2 denote R at the end of execution of the step. If $KD5 \leq L$ then the following assertion holds:*

$$\forall x\ x \neq v \wedge (R_2(v,x) \vee (x \neq \tau \wedge R_2(x,v))) \rightarrow R^*(u,x)$$

The formula $R^*(u,x)$ in the above assertion can be replaced by $(R(u,x) \vee R^2(u,x))$. The lemma can be verified using Lemma 4.2.2.

We give below the main lemma of this section. It informally states that during execution of Algorithm 4.2.1 the model graph M is always less than or equal to any L-model of P. What we really want to obtain is the first four assertions, however, to prove them the assertions 5 and 6 are also needed.

Lemma 4.2.4 *Let $N = \langle W', \tau', R', h' \rangle$ be an arbitrary L-model of P. It is an invariant property of the algorithm (wrt. the numerated steps) that there exists a relation $r \subseteq W \times W'$ such that the following assertions hold:*

1. $r(\tau,\tau') \wedge (\forall x'\ r(\tau,x') \rightarrow x' = \tau')$
2. $\forall x, x'\ \forall \zeta \in H(x)\ \ r(x,x') \rightarrow N, x' \vDash \zeta$
3. $\forall x, y, x'\ R(x,y) \wedge r(x,x') \rightarrow \exists y'\ R'(x',y') \wedge r(y,y')$
4. $\forall x, x', y'\ R'(x',y') \wedge r(x,x') \rightarrow \exists y\ R(x,y) \wedge r(y,y')$

   When $L \in \{KD5, KD45, S5\}$, this assertion has a stronger form:
   $\forall x, x', y'\ R'(x',y') \wedge r(x,x') \wedge x \neq \tau \rightarrow R(x,\varpi) \wedge r(\varpi,y')$
   $\wedge\ \forall x', y'\ R'(x',y') \wedge r(\tau,x') \rightarrow R(\tau,\rho) \wedge r(\rho,y')$

5. $\forall x, y, y'\ R(x,y) \wedge r(y,y') \rightarrow \exists x'\ R'(x',y') \wedge r(x,x')$
6. *If $KD5 \leq L$ then*
   $\forall x, x', x''\ r(x,x') \wedge r(x,x'') \rightarrow x' = x'' \vee R'(x',x'')$

Proof. First, we prove the lemma for the step 1.

Case $L \in \{KD, T, KDB, B\}$:

Let $(v_1, v_2, \ldots)$ be the infinite chain of worlds created by the call $CreateEmptyTail_L(\tau)$. Denote $v_0 = \tau$.

Let $r = \{(\tau,\tau')\} \bigcup \{(v_i, v_i{}') \mid R'^i(\tau', v_i{}')\ for\ i \geq 1\}$.

It is obvious that the assertions 1 and 2 hold.

3) Suppose that $R(x,y) \wedge r(x,x')$ holds. We show that $\exists y'\ R'(x',y') \wedge r(y,y')$. There exists $i \geq 0$ such that ($i \geq 1$, $KDB \leq L$, $x = v_i$, and $y = v_{i-1}$) or ($T \leq L$ and $x = y = v_i$) or ($x = v_i$ and $y = v_{i+1}$). If $x = v_i$, there exist $v_0{}'$, $v_1{}'$, ..., $v_{i+1}{}'$ such that $v_0{}' = \tau'$, $v_i{}' = x'$, and $\forall j\ 0 \leq j \leq i$ $R'(v_j{}', v_{j+1}{}')$. It is easy to check that if $y = v_j$ for some $i-1 \leq j \leq i+1$, then by choosing $y' = v_j{}'$ we have $R'(x',y') \wedge r(y,y')$.

4) Suppose that $R'(x', y') \wedge r(x, x')$ holds. There exists $i \geq 0$ such that $x = v_i$. It is easy to check that for $y = v_{i+1}$ we have $R(x, y) \wedge r(y, y')$.

5) The proof of the assertion is similar to the proof of the assertion 3.

Case $L \in \{KD5, KD45, S5\}$: Let
$r = \{(\tau, \tau')\} \bigcup \{(\rho, \rho') \mid R'(\tau', \rho')\} \bigcup \{(\varpi, \varpi') \mid \exists \rho' \; R'(\tau', \rho') \wedge R'(\rho', \varpi')\}$.

It is easy to check that the assertions 1, 2, 4, 6 hold.

3) Suppose that $R(x, y) \wedge r(x, x')$ holds. We show that $\exists y' \; R'(x', y') \wedge r(y, y')$. Case $y = \tau$: We have $L = S5$. For $y' = \tau'$, we have $R'(x', y') \wedge r(y, y')$. Case $y = \rho$: If $x = \rho$, we have $R'(\tau', x')$, hence $R'(x', x')$; for $y' = x'$, we have $R'(x', y') \wedge r(y, y')$. If $x = \tau$ or $x = \varpi$, then for y' such that $R'(\tau', y')$, it is easy to check that $R'(x', y') \wedge r(y, y')$ holds. Case $y = \varpi$: Let y' be a world such that $R'(x', y')$. It is easy to check that $r(y, y')$ holds.

5) Suppose that $R(x, y) \wedge r(y, y')$ holds. We show that $\exists x' \; R'(x', y') \wedge r(x, x')$. Case $x = \tau$: Let $x' = \tau'$. We have $r(x, x')$. If $y = \tau$, we have $L = S5$, hence $R'(x', y')$. If $y = \rho$, we have $R'(\tau', y')$, hence $R'(x', y')$. If $y = \varpi$, we have $KD45 \leq L$, hence from $r(y, y')$ we derive $R'(\tau', y')$, and $R'(x', y')$. Case $x = \rho$ and $y = \varpi$: Since $r(y, y')$, there exists x' such that $R'(\tau', x')$ and $R'(x', y')$, which implies $R'(x', y') \wedge r(x, x')$. Case $x = \varpi$ and $y = \rho$: By the assertion 3, there exist ρ' and ϖ' such that $R'(\tau', \rho') \wedge R'(\rho', \varpi') \wedge r(\varpi, \varpi')$. Since $r(y, y')$ and $y = \rho$, we have $R'(\tau', y')$. It is easily seen that for $x' = \varpi'$ we have $R'(x', y') \wedge r(x, x')$. Case $x = y \neq \tau$: Let $x' = y'$. We have $r(x, x')$. It is easily seen that $R'(x', y')$.

Fix one of the steps 2a, 2b, 2c, and suppose that before executing the step, M with r satisfies all the assertions 1 - 6. Let $M_2 = \langle W_2, \tau, R_2, H_2 \rangle$ be the model graph obtained by executing the step. We construct a relation r_2 such that M_2 with r_2 satisfies all the mentioned assertions.

In the remainder of this proof, the phrase "by the assumption about the assertion i", where $1 \leq i \leq 6$, is understood as "because the assertion i is assumed to hold for M and r". When we are proving the assertion i for M_2 and r_2, the phrase "by assumption" in most of cases stands for "by the assumption about the assertion i".

For the step 2a: Let $r_2 = r$. We only need to prove the assertion 2 for $x = u$ and $\zeta = B$. Suppose that $r(u, x')$ holds. We show that $N, x' \vDash \zeta$. Assumptions 1 - 4 guarantee that $M \leq N$, hence for every $1 \leq i \leq k$ we have $N, x' \vDash C_i$. On the other side, it follows from $(B \leftarrow C_1, \ldots, C_k) \in H(u)$ and $r(u, x')$ that $N, x' \vDash (B \leftarrow C_1, \ldots, C_k)$. Therefore $N, x' \vDash B$, and $N, x' \vDash \zeta$.

For the step 2b: Let $r_2 = r$. We only need to prove the assertion 2 for $x = v$ and $\zeta = \psi$. Suppose that $r(v, x')$ holds. We show that $N, x' \vDash \zeta$. By the assumption 5, we have $\exists u' \; R'(u', x') \wedge r(u, u')$. Then by the assumption

2, we derive $\exists u'\ R'(u',x') \wedge (N,u' \vDash \phi)$, which implies $N,x' \vDash \zeta$.

For the step 2c:

Case $L \in \{KD, T, KDB, B\}$:

Let v be the world mentioned in the algorithm; we denote it also by v_0. Let $(v_1, v_2, \ldots)$ be the chain of worlds created by $CreateEmptyTail_L(v_0)$. Let

$$r_2 = r \bigcup \ \{(v_i, v_i{}') \mid i \geq 0 \text{ and } \exists u', v_0{}' \in W'\ r(u,u') \wedge R'(u', v_0{}') \wedge N, v_0{}' \vDash \psi \wedge R'^{i}(v_0{}', v_i{}')\}.$$

The assertion 1 obviously holds.

2) It suffices to verify the assertion for $x = v$ and $\zeta = \psi$; but this case is also trivial, by the definition of r_2.

3) Suppose that $R_2(x,y) \wedge r_2(x,x')$ holds. We show that $\exists y'\ R'(x',y') \wedge r_2(y,y')$. If ($x \neq u$ and $\forall i \geq 0\ x \neq v_i$) or ($x = u$ and $y \neq v_0$), we have $R(x,y) \wedge r(x,x')$, and then by assumption, there exists y' such that $R'(x',y') \wedge r(y,y')$, which implies $R'(x',y') \wedge r_2(y,y')$.

- Case $x = u$ and $y = v_0$: We have $r(u,x')$ since $r_2(x,x')$ and $x = u$. Consequently, $N,x' \vDash \Diamond\psi$, and hence there exists y' such that $R'(x',y')$ and $N,y' \vDash \psi$. For such y' we have $R'(x',y') \wedge r_2(y,y')$.

- Case $x = v_i$, $i \geq 0$: Denote $v_{-1} = u$. Since $R_2(x,y)$, we have $y = v_{i-1}$ or $y = v_i$ or $y = v_{i+1}$. Since $r_2(x,x')$, there exist u' and $v_0{}'$ such that $r(u,u') \wedge R'(u',v_0{}') \wedge N, v_0{}' \vDash \psi \wedge R'^{i}(v_0{}', x')$. Let $v_1{}'$, ...,$v_{i+1}{}'$ be worlds such that $v_i{}' = x'$ and $R'(v_j{}', v_{j+1}{}')$, for $0 \leq j \leq i$. Denote $v_{-1}{}' = u'$. We have $r_2(v_j, v_j{}')$, for $-1 \leq j \leq i+1$. It is easily seen that for $y = v_j$, $i-1 \leq j \leq i+1$, by choosing $y' = v_j{}'$, we have $R'(x',y') \wedge r_2(y,y')$.

4) Suppose that $R'(x',y') \wedge r_2(x,x')$ holds. We show that $\exists y\ R_2(x,y) \wedge r_2(y,y')$. If $\forall i \geq 0\ x \neq v_i$, we have $r(x,x')$, and by assumption, $\exists y\ R(x,y) \wedge r(y,y')$, which implies $\exists y\ R_2(x,y) \wedge r_2(y,y')$. If $x = v_i$ for some $i \geq 0$, then by choosing $y = v_{i+1}$, we have $R_2(x,y) \wedge r_2(y,y')$.

5) Suppose that $R_2(x,y) \wedge r_2(y,y')$ holds. We show that $\exists x'\ R'(x',y') \wedge r_2(x,x')$.

Case $L = KDB$ or $L = B$: We have $R_2(y,x) \wedge r_2(y,y')$, and it follows from the assertion 3 that there exists x' such that $R'(y',x') \wedge r_2(x,x')$. Since R' is symmetric, we also have $R'(x',y')$.

Case $L = KD$ or $L = T$: If ($x \neq u$ and $\forall i \geq 0\ x \neq v_i$) or ($x = u$ and $y \neq v_0$), we have $R(x,y) \wedge r(y,y')$, and then by assumption, there exists x' such that $R'(x',y') \wedge r(x,x')$, which implies $R'(x',y') \wedge r_2(x,x')$. Denote $v_{-1} = u$. Now assume that $x = v_i$ for some $i \geq -1$, and that if $x = v_{-1}$ then $y = v_0$. Since $R_2(x,y)$, we have $y = v_j$ for $j = i$ or $j = i+1$. Since $r_2(y,y')$, there exist u' and $v_0{}'$ such that $r(u,u') \wedge R'(u',v_0{}') \wedge N, v_0{}' \vDash \psi \wedge R'^{j}(v_0{}', y')$.

Let ${v_1}'$, ...,${v_j}'$ be worlds such that ${v_j}' = y'$ and $R'({v_{k-1}}', {v_k}')$, for $1 \leq k \leq j$. Denote ${v_{-1}}' = u'$. We have $r_2(v_k, {v_k}')$, for $-1 \leq k \leq j$. It is easily seen that for $x = v_i$, by choosing $x' = {v_i}'$, we have $R'(x', y') \wedge r_2(x, x')$.

Case $L \in \{KD5, KD45, S5\}$:
Let $r_2 = r \bigcup \{(v, v') \mid \exists u' \; r(u, u') \wedge R'(u', v') \wedge N, v' \vDash \psi\}$.
The assertion 1 holds by assumption.

2) It suffices to verify the assertion for $x = v$ and $\zeta = \psi$; but this case is also trivial, by the definition of r_2.

3) Suppose that $R_2(x, y) \wedge r_2(x, x')$ holds. We show that $\exists y' \; R'(x', y') \wedge r_2(y, y')$.

- Case $x \neq v$ and $y \neq v$: The assertion holds by assumption.

- Case $x = v$ and $y = v$: It is easy to check that for $y' = x'$ we have $R'(x', y') \wedge r_2(y, y')$.

- Case $x = v$ and $y \neq v$: There exists u' such that $r(u, u') \wedge R'(u', x') \wedge (N, x' \vDash \psi)$. Since $R_2(x, y)$, by Lemma 4.2.3, we have $R^*(u, y)$. Applying the assumption 3 one or more times for $R^*(u, y) \wedge r(u, u')$, we conclude that there exists y' such that $R'^*(u', y') \wedge r(y, y')$. Since $R'(u', x')$ and $KD5 \leq L$ and $r \subseteq r_2$, it follows that $R'(x', y') \wedge r_2(y, y')$.

- Case $y = v$ and $x \neq v$: By Lemma 4.2.2:2 and the assumption 3, there exists u' such that $((R'^*(\tau', u') \wedge u \neq \tau) \vee (u' = \tau' \wedge u = \tau)) \wedge r(u, u')$. Hence, by the assumption 2, we have $N, u' \vDash \phi$. Hence there exists y' such that $R'(u', y')$ and $N, y' \vDash \psi$. Thus we have $r_2(y, y')$. We now show that $R'(x', y')$ holds.

  Case $x = \tau$: By the assumption 1, we have $x' = \tau'$. If $u = \tau$, we have $u' = \tau' = x'$, and $R'(x', y')$. Suppose that $u \neq \tau$. In this case, from $R_2(x, y)$, $x = \tau$, and $y = v$, we claim that $KD45 \leq L$. It follows that $R'(\tau', y')$, hence $R'(x', y')$.

  Case $x \neq \tau$: Since $R_2(x, y)$, by Lemma 4.2.3, we have $R^*(u, x)$. Applying the assumption 3 one or more times for $R^*(u, x) \wedge r(u, u')$, we conclude that there exists x'' such that $R'^*(u', x'') \wedge r(x, x'')$, which implies $R'(x'', y')$, since $R'(u', y')$ and $KD5 \leq L$. Since $r(x, x') \wedge r(x, x'')$, by the assumption 6, we have $x' = x'' \vee R'(x'', x')$. This together with $R'(x'', y')$ implies $R'(x', y')$.

4) It is sufficient to show that for any x, x', y' belonging to W, the following implication holds: $R'(x', y') \wedge r_2(x, x') \wedge x \neq \tau \to r_2(\varpi, y')$. Suppose that the antecedent of this implication holds. The case $x \neq v$ is trivial, so assume that $x = v$. Since $r_2(x, x')$, there exists u' such that $r(u, u') \wedge R'(u', x') \wedge N, x' \vDash \psi$. If $u = \tau$, by assumption, we derive $r(\rho, x')$,

and then $r(\varpi, y')$. If $u \neq \tau$, we have $R'(u', x') \wedge r(u, u') \wedge u \neq \tau$, which by assumption implies $r(\varpi, x')$, and then $r(\varpi, y')$. Hence $r_2(\varpi, y')$ holds.

5) Suppose that $R_2(x, y) \wedge r_2(y, y')$ holds. We show that $\exists x'\ R'(x', y') \wedge r_2(x, x')$.

- Case $x \neq v$ and $y \neq v$: The assertion holds by assumption.
- Case $x = v$ and $y = v$: It is easy to check that for $x' = y'$ we have $R'(x', y') \wedge r_2(x, x')$.
- Case $x \neq v$ and $y = v$: Since $r_2(y, y')$, there exists u' such that $r(u, u') \wedge R'(u', y') \wedge N, y' \vDash \psi$.

  Case $x = \tau$: Let $x' = \tau'$. We have $r_2(x, x')$, and need to prove that $R'(\tau', y')$ holds. If $u = \tau$, we have $u' = \tau'$, and $R'(\tau', y')$. Suppose that $u \neq \tau$. From $R_2(x, y)$, $x = \tau$, and $y = v$, we claim that $KD45 \leq L$. By Lemma 4.2.2:2, we have $R^*(\tau, u)$, and hence $R(\tau, u)$. Applying the assumption 3 for $R(\tau, u) \wedge r(\tau, \tau')$, we derive that there exists u'' such that $R'(\tau', u'') \wedge r(u, u'')$. From the assumption 6, we derive that $u' = u'' \vee R'(u', u'')$. This together with $R'(\tau', u'')$, $R'(u', y')$, and $KD45 \leq L$ implies $R'(\tau', y')$.

  Case $x \neq \tau$: Since $R_2(x, y)$, by Lemma 4.2.3, we have $R^*(u, x)$. Once or more times applying the assumption 3 for $R^*(u, x) \wedge r(u, u')$, we derive that there exists x' such that $R'^*(u', x') \wedge r(x, x')$. We have $R'(x', y')$ since $R'(u', y')$ and $KD5 \leq L$.
- Case $x = v$ and $y \neq v$: We have $r(y, y')$. It is easily seen that there exists u' such that $r(u, u')$. By the assumption 2, we have $N, u' \vDash \phi$, hence there exists v' such that $R'(u', v')$ and $N, v' \vDash \psi$. This implies $r_2(v, v')$. Since $R_2(x, y)$, by Lemma 4.2.3, we have $R^*(u, y)$. One or more times applying the assumption 3 for $R^*(u, y) \wedge r(u, u')$, we conclude that there exists y'' such that $R'^*(u', y'') \wedge r(y, y'')$. Consequently, $R'(v', y'')$ and $R'(y'', v')$ hold, since $R'(u', v')$ and $KD5 \leq L$. Since $r(y, y') \wedge r(y, y'')$, by the assumption 6, we have $y' = y'' \vee R'(y'', y')$. As a consequence, we have $R'(v', y')$, since $R'(v', y'')$, $R'(y'', v')$, and $KD5 \leq L$. Therefore, for $x' = v'$, we have $R'(x', y') \wedge r_2(x, x')$.

6) Suppose that $r_2(x, x') \wedge r_2(x, x'')$ holds. We show that $x' = x'' \vee R'(x', x'')$. If $x \neq v$, the assertion holds by assumption. Suppose that $x = v$. There exist u' and u'' such that $(r(u, u') \wedge R'(u', x') \wedge N, x' \vDash \psi)$ and $(r(u, u'') \wedge R'(u'', x'') \wedge N, x'' \vDash \psi)$. Since $r(u, u') \wedge r(u, u'')$, by assumption, we have $u' = u'' \vee R'(u', u'')$. If $u' = u''$, we derive $R'(x', x'')$ from $R'(u', x')$ and $R'(u'', x'')$. If $R'(u', u'')$, we have $R'(u'', x')$, since $R'(u', x')$ and $KD5 \leq L$; hence $R'(x', x'')$ since $R'(u'', x'')$ and $KD5 \leq L$. □

We now estimate the complexity of the algorithm and the size of the resulting model.

Lemma 4.2.5 *Let n and k be the size (i.e. the length) and the modal depth of a program P, respectively. Let M be the model constructed by Algorithm 4.2.1 for P. If L is one of the logics KD, T, KDB, and B, then the algorithm terminates in $O(n^{2k+2})$ steps, the real diameter of M is less than or equal to k, M can be coded using $O(n^{k+1})$-space, and for any m, the model $M|_m$ has no more than $m.n^k$ worlds. If L is one of the logics $KD5$, $KD45$, and $S5$, then the algorithm terminates in $O(n^4)$ steps, and the size of M is of order $O(n^2)$.*

Proof. Case $L \in \{KD, T, KDB, B\}$: It is easily seen that, during execution of the algorithm, the real diameter of M is always less than or equal to k. The number of worlds in $M|_k$ is bounded by n^k, the number of edges in $M|_k$ are bounded by $3n^k$, and the number of formulae at each world is bounded by n. Hence the size of $M|_k$ is of order $O(n^{k+1})$. It is easily seen that checking whether the steps 2b, 2a, and 2c can extend the model graph for some u and ψ is solvable in $O(n^{k+1})$ steps. Because that each execution of the step 2 either increases the size of $M|_k$ at least by 1 or certifies that no more changes can be made to M, we conclude that the algorithm terminates in $O(n^{2k+2})$ steps. For each world $w \in W$, there is at most one infinite chain of empty worlds starting from w, hence the resulting model M can be coded using $O(n^{k+1})$-space, and for any $m \geq 1$, the model $M|_m$ has no more than $m.n^k$ worlds.

Case $L \in \{KD5, KD45, S5\}$: It is easily seen that, during execution of the algorithm, there are no more than $n + 3$ worlds. The size of each world is bounded by n. Hence the size of M is of order $O(n^2)$. Checking whether the step 2 can make some change to M can be solved in $O(n^2)$ steps. Each execution of the step 2 either increases the size M at least by 1 or certifies that no more changes can be made to the model graph. Hence the algorithm terminates in $O(n^4)$ steps. □

Here is the main theorem of this section:

Theorem 4.2.6 *For any positive modal logic program P of size n, in a modal logic $L \in \{KD, T, KDB, B, KD5, KD45, S5\}$, there exists the least L-model M of P. If $L \in \{KD5,\ KD45,\ S5\}$, then M can be constructed in $O(n^4)$ steps and its size is of order $O(n^4)$. If $L \in \{KD, T, KDB, B\}$ and the modal depth of P is finitely bounded, then M can be constructed in PTIME, and for any $m \geq 1$, the size of the model $M|_m$ is bounded by a polynomial in n and m.*

Proof. Let M be the model constructed by Algorithm 4.2.1 for P in L. It is easy to prove by induction (on the construction of ϕ) that for any world u, and any formula $\phi \in H(u)$, $M, u \vDash \phi$. Hence $M \vDash P$. Lemma 4.2.2:1 asserts that M is a L-model. These together with Lemma 4.2.4 assert that M is the least L-model of P. The estimation of the complexity of the algorithm and the size of M follows from Lemma 4.2.5. □

Corollary 4.2.7 *The problem of checking satisfiability of a set of Horn formulae with finitely bounded modal depth in KD, T, KDB, or B is decidable in PTIME.*

Corollary 4.2.8 *(Fariñas del Cerro and Penttonen [7], Chen and Lin [4]) The problem of checking satisfiability of a set of Horn formulae in $KD5$, $KD45$, or $S5$ is decidable in PTIME.*

These corollaries immediately follow from Proposition 2.4.3, Theorem 4.2.6, Lemmas 2.1.11 and 2.1.6.

The model M constructed by Algorithm 4.2.1 for a program P in a logic $L \in \{KD, T, KDB, B\}$ is infinite. By Lemmas 4.2.5 and 2.1.11, M can be treated as a finite model. For L being KD or T, we show that M can be actually transformed to an equivalent finite model.

Lemma 4.2.9 *Let $M = \langle W, \tau, R, h\rangle$ be a connected[1] KD-model. Suppose that W can be divided into two groups W_1 and W_2 such that W_1 contains τ, W_2 is a non-empty set of empty worlds, and there is no edges connecting W_2 to W_1. Let w_2 be a new world, and let*

$$\begin{aligned}
&W' = W_1 \bigcup \{w_2\},\\
&R' = R - \{(x,y) \mid x \in W_1 \wedge y \in W_2\}\\
&\qquad \bigcup \{(x,w_2) \mid x \in W_1 \wedge \exists y \in W_2\ R(x,y)\} \bigcup \{(w_2,w_2)\},\\
&h'(x) = h(x)\ for\ x \neq w_2, \quad h'(w_2) = \emptyset,\\
&M' = \langle W', \tau, R', h'\rangle,\\
&r = \{(x,x) \mid x \in W_1\} \bigcup \{(y,w_2) \mid y \in W_2\},\\
&s = \{(x,x) \mid x \in W_1\} \bigcup \{(w_2,y) \mid y \in W_2\}.
\end{aligned}$$

Then $M' \leq M$ wrt. s, and $M \leq M'$ wrt. r. Moreover, for any formula ϕ, $M \vDash \phi$ iff $M' \vDash \phi$.

The proof of the first assertion is straightforward. The second assertion can be easily proved by induction on the construction of ϕ.

Let M be the model constructed by Algorithm 4.2.1 for a program P in a logic $L \in \{KD, T\}$. By the above lemma, if we replace all infinite chains of empty worlds in M by a new empty world w_2, which is connected to itself, then we obtain the least L-model of P. We arrive at:

Corollary 4.2.10 *For any positive program P with a finitely bounded modal depth, the least KD-model and the least T-model of P can be constructed in PTIME, and their sizes are bounded by a polynomial in the size of P.*

[1] every world from W is directly or indirectly reachable from the root τ

4.3 Characterizations of Positive Programs in $\boldsymbol{KB}$, $\boldsymbol{K5}$, $\boldsymbol{K45}$, and $\boldsymbol{KB5}$

In this section, we show that for any positive program P in a modal logic $L \in \{KB, K5, K45, KB5\}$ we can construct one or two L-models such that a positive formula ϕ follows from P iff ϕ is satisfied in these models.

Definition 4.3.1 We call a model $\langle W, \tau, R, h\rangle$ *flat* if $W = \{\tau\}$ and $R = \emptyset$.

Definition 4.3.2 Let L be a normal modal logic. We say that L is *almost serial* if all L-frames $\langle W, \tau, R\rangle$ satisfy the formula
$\exists x\ R(\tau, x) \rightarrow (\forall y\ R^*(\tau, y) \rightarrow \exists z\ R(y, z))$.

The logics KB, $K5$, $K45$, and $KB5$ are almost serial. Hence any non-flat *connected* frame in KB (resp. $K5$, $K45$, $KB5$) is a frame in KDB (resp. $KD5$, $KD45$, $S5$). We say that KDB, $KD5$, $KD45$, and $S5$ are serial logics respectively corresponding to KB, $K5$, $K45$, and $KB5$. For $L \in \{KB, K5, K45, KB5\}$, we use LD to denote the serial logic corresponding to L.

Theorem 4.3.3 *Let P be a positive program in a modal logic $L \in \{KB, K5, K45, KB5\}$. Let M be the least LD-model of P constructed by Algorithm 4.2.1, and let $N = \langle W', \tau', R', h'\rangle$ be a non-flat L-model of P. Then $M \leq N$.*

The proof of this theorem is straightforward.

Theorem 4.3.4 *The problem of checking whether a positive modal logic program P has flat models is decidable in PTIME. If P has flat models then it has the least flat model[2], which can be constructed in PTIME and has size bounded by a polynomial in the size of P.*

Proof. Let P' be the program obtained from P by replacing every formula $\Box\phi$ in P by *true*, and every atom $\Diamond p$ by a new primitive proposition p'. The program P' is a positive program in the classical logic. Therefore it has the least model M, which can be constructed in PTIME and has size bounded by a polynomial in the size of P. If there is a primitive proposition p' satisfied in M then P has no flat models; for otherwise let N be a flat model of P, we would have $\Diamond p$ satisfied in N, which is impossible. If there is no primitive proposition p' satisfied in M then M is a flat model of P. If N is a flat model of P then it is easily seen that it is also a model of P', hence $M \leq N$. $\Box$

[2] a flat model M such that M is less than or equal to any flat model of P

Corollary 4.3.5 *Let P be a positive program in a modal logic $L \in \{KB, K5, K45, KB5\}$. Let M be the least LD-model of P. If P has no flat models then M is the least L-model of P. If P has flat models then it has the least flat model M', and for every model N of P, if N is flat then $M' \leq N$, otherwise $M \leq N$.*

This corollary follows from Theorems 4.3.3 and 4.3.4.

Corollary 4.3.6 *The problem of checking satisfiability of a set of Horn formulae with finitely bounded modal depth in KB is decidable in PTIME.*

Corollary 4.3.7 *(Chen and Lin [4]) The problem of checking satisfiability of a set of Horn formulae in $K5$, $K45$, or $KB5$ is decidable in PTIME.*

These corollaries immediately follow from Proposition 2.4.3, Theorems 4.3.3 and 4.3.4, and Lemmas 2.1.11 and 2.1.6.

4.4 Constructing the Least Models for Positive Programs in $KD4$ and $S4$

Algorithm 4.2.1 cannot be applied for the logics $KD4$ and $S4$, because it will not terminate, for example, when P contains a formula $\Box\Diamond p$. Before giving an algorithm of constructing the least models for positive programs in $KD4$ and $S4$, let us introduce some notations.

Data structures W, R, H, and notations M and h have the same meanings as given in Section 4.2. Additionally, we use the following data structures:

- $StatusW : W \to \{normal, minimal\}$,
- $StatusR : R \to \{permanent, temporal\}$,
- $Next : W \times \mathcal{F} \to W$, which is interpreted as follows: $Next(u, \Diamond\phi) = v$ means $R(u, v)$, $\phi \in H(v)$, and the world u *realizes* the requirement $\Diamond\phi$ by going to v.

We define the binary relation *Permanent* on W as follows

$$Permanent(x, y) \equiv (R(x, y) \wedge StatusR(x, y) = permanent),$$

and use $Permanent^*$ to denote its transitive closure. We also need the following notations:

$$\begin{array}{rcl} H^+(x) & = & \{\phi \mid \exists y\ Permanent^*(y, x) \text{ and} \\ & & (\Box\phi \in H(y) \text{ or } (\phi \in H(y) \text{ and } \phi = \Box\psi \text{ for some } \psi))\} \\ H^*(x) & = & H^+(x) \bigcup \{\phi \mid \\ & & \Box\phi \in H(x) \text{ or } (\phi \in H(x) \text{ and } \phi = \Box\psi \text{ for some } \psi)\} \end{array}$$

where $H^+(x)$ can be interpreted as the potentiality inherited from the predecessors of x, and $H^*(x)$ - the potentiality inherited from x and its predecessors.

Consider Algorithm 4.4.1 given below. We build the least L-model for P by constructing a L-model graph for it. The relation $Permanent$, without self-referencing edges, founds a tree with root τ, which is a skeleton of the model graph. Consider the step 2c. If we create a new world to realize the formula $\Diamond p$, then it should contain p and $H^*(u)$. We first try to find a world v on the path from τ to u (in the tree generated by $Permanent$) such that it contains $\Diamond p$, and $H^*(v) = H^*(u)$. If such v exists and the formula $\Diamond p$ has been realized in v by a connection from v to w, then we just connect u to w; but this edge is not *permanent*, because that if a new formula $\Box\zeta$ is later added to u while ζ or $\Box\zeta$ is not present in w then we will delete the edge. If such v exists but the formula $\Diamond p$ has not been realized in v, then we do nothing; this means that the formula $\Diamond p$ will be realized in v before in u. If such v does not exist, then we realize the formula $\Diamond p$ in u by creating a new world, denoted by v, assigning p and $H^*(u)$ to it, and connecting u to v, and v to a new *minimal* world. *Minimal* worlds are elements that make M the L-least model of P, they have the same role as infinite chains of empty worlds used in Algorithm 4.2.1. In the step 2(b)i, we do not add ψ to u because of the nature of *minimal* worlds.

In Algorithm 4.4.1, we use a procedure $CreateNextMinNode(x)$, which is defined as:

> Let y be a new world.
> Set $W = W \bigcup \{y\}$, $StatusW(y) = minimal$, $H(y) = H^*(x)$,
> $R = R \bigcup \{(x,y),(y,y)\}$, $StatusR(x,y) = permanent$.
> If $L = S4$, set $StatusR(y,y) = permanent$,
> otherwise, set $StatusR(y,y) = temporal$.

Algorithm 4.4.1
Input: *A positive modal logic program P in a modal logic $L \in \{KD4, S4\}$*
Output: *The least L-model $M = \langle W, \tau, R^*, h\rangle$ of P.*
Steps:

1. *Set $W = \{\tau\}$, $StatusW(\tau) = normal$, $H(\tau) = P$.*
   If $L = S4$, set $R = \{(\tau,\tau)\}$ and $StatusR(\tau,\tau) = permanent$, otherwise, set $R = \emptyset$.
   $CreateNextMinNode(\tau)$.

2. *For every world $u \in W$, and for every formula $\phi \in H(u)$,*

   (a) *Case $\phi = B \leftarrow C_1, \ldots, C_k$ for some $k \geq 1$, and $M, u \vDash C_i$ for all $1 \leq i \leq k$:*

   Set $H(u) = H(u) \bigcup \{B\}$.

(b) *Case* $\phi = \Box\psi$ *for some* ψ *:*

   i. *If* $StatusW(u) = minimal$, $StatusR(u,u) = temporal$, *and* $\psi \notin H(u)$, *then*

      Set $R = R - \{(u,u)\}$, $StatusW(u) = normal$,
      $CreateNextMinNode(u)$.

   ii. *Otherwise,*

      For every v *such that* $Permanent^*(u,v)$,
         Set $H(v) = H(v) \bigcup \{\phi, \psi\}$,
      For every v *such that* $R(u,v)$, $StatusR(u,v) = temporal$, *and* $\{\phi, \psi\} \nsubseteq H(v)$,
         Set $R = R - \{(u,v)\}$,
         For every $\Diamond p$ *such that* $Next(u, \Diamond p) = v$,
            Set $Next(u, \Diamond p)$ *undefined.*

(c) *Case* $\phi = \Diamond p$ *for some primitive proposition* p, *and* $Next(u, \Diamond p)$ *is undefined:*

   i. *If there exist* v *and* w *such that:*

      $$Permanent^*(v,u) \wedge H^*(v) = H^*(u) \wedge \phi \in H(v) \wedge Permanent(v,w) \wedge Next(v, \Diamond p) = w,$$

      Set $R = R \bigcup \{(u,w)\}$, $StatusR(u,w) = temporal$,
         $Next(u, \Diamond p) = w$.

   ii. *Otherwise, if* $\neg(\exists x \;\; x \neq u \wedge Permanent^*(x,u) \wedge H^*(x) = H^*(u) \wedge \phi \in H(x))$,

      Let v *be a new world.*
      Set $Next(u, \Diamond p) = v$.
      Set $W = W \bigcup \{v\}$, $StatusW(v) = normal$,
         $H(v) = H^*(u) \bigcup \{p\}$,
         $R = R \bigcup \{(u,v)\}$, $StatusR(u,v) = permanent$.
      If $L = S4$, *set* $R = R \bigcup \{(v,v)\}$,
            $StatusR(v,v) = permanent$.
      $CreateNextMinNode(v)$.

3. *While some change occurred, repeat the step 2.*

We give below an auxiliary lemma. Its proof is straightforward.

Lemma 4.4.2 *The following assertions are invariant properties of the algorithm (wrt. the numerated steps):*

1. $L = KD4 \rightarrow$
   $$\forall x \;\; (StatusW(x) = normal \rightarrow \neg R(x,x)) \wedge (StatusW(x) = minimal \rightarrow StatusR(x,x) = temporal)$$

2. $L = S4 \rightarrow \forall x \; Permanent(x,x)$

3. $\forall x\ Permanent^*(x,x) \rightarrow Permanent(x,x)$

4. *For every formula* $\Diamond q$,
$$\forall x, y\ Next(x, \Diamond q) = y \rightarrow R(x,y) \wedge q \in H(y) \wedge (Permanent(x,y) \vee \\ (\exists z\ Permanent^*(z,x) \wedge H^*(z) = H^*(x) \wedge \Diamond q \in H(z) \\ \wedge\ Permanent(z,y) \wedge Next(z, \Diamond q) = y))$$

5. $(L = KD4 \rightarrow \forall x\ \exists y\ R(x,y)) \wedge (L = S4 \rightarrow \forall x\ R(x,x))$

Lemma 4.4.3 *If Algorithm 4.4.1 terminates, then the resulting model* M *is a* L*-model of* P.

Proof. We prove that for every world u, and a formula $\phi \in H(u)$, $M, u \models \phi$. It is easily seen that the assertion holds for the case when ϕ is of the form $\Box\psi$ or $(B \leftarrow C_1, \ldots C_k)$, with $k \geq 1$. Suppose that for some u and $\Diamond q \in H(u)$, $Next(u, \Diamond q)$ is undefined. Without loss of generality we suppose that for every world x such that $x \neq u$ and $Permanent^*(x,u)$, $Next(x, \Diamond p)$ is defined for every $\Diamond p \in H(x)$. Since no more changes can be made, there exists v such that $v \neq u \wedge Permanent^*(v,u) \wedge H^*(v) = H^*(u) \wedge \Diamond q \in H(v)$. We have $Next(v, \Diamond q)$ defined. Let $Next(v, \Diamond q) = w$. By Lemma 4.4.2:4, we have:

$$R(v,w) \wedge q \in H(w) \wedge (Permanent(v,w) \vee \\ (\exists z\ Permanent^*(z,v) \wedge H^*(z) = H^*(v) \wedge \Diamond q \in H(z) \wedge \\ Permanent(z,w) \wedge Next(z, \Diamond q) = w)).$$

If $Permanent(v,w)$ holds, M can be extended by the step 2(c)i, and we have a contradiction. If there is z such that $Permanent^*(z,v) \wedge (H^*(z) = H^*(v)) \wedge \Diamond q \in H(z) \wedge Permanent(z,w) \wedge Next(z, \Diamond q) = w$, then we have $Permanent^*(z,u) \wedge H^*(z) = H^*(u) \wedge \Diamond q \in H(z) \wedge Permanent(z,w) \wedge Next(z, \Diamond q) = w$, hence the step 2(c)i can be applied to extend M, which is a contradiction. Therefore for any world $u \in W$ and $\Diamond p \in H(u)$, $Next(u, \Diamond p)$ is defined, which by Lemma 4.4.2:4 implies that $N, u \models \Diamond p$. The consequence is that $M, \tau \models P$, since $P \subseteq H(\tau)$. Lemma 4.4.2:5 asserts that R^* satisfies all L-frame axioms. $\Box$

We now turn to the main lemma given below. What we really want to claim by this lemma is that the model M constructed by Algorithm 4.4.1 is less than or equal to any L-model of P. One would expect only the first four assertions of the lemma, in the simpler form:

1. $r(\tau, \tau')$

2. $\forall x, x'\ \forall\ \zeta \in H(x)\ \ r(x,x') \rightarrow N, x' \models \zeta$

3. $\forall x, y, x'\ R^*(x,y) \wedge r(x,x') \rightarrow \exists y'\ R'(x',y') \wedge r(y,y')$

4. $\forall x, x', y'\ R'(x',y') \wedge r(x,x') \rightarrow \exists y\ R^*(x,y) \wedge r(y,y')$

We have made the assertions stronger and added additional ones in order to prove themselves.

Lemma 4.4.4 *Let $N = \langle W', \tau', R', h' \rangle$ be an arbitrary L-model of P. It is an invariant property of the algorithm (wrt. the numerated steps) that there exists a relation $r \subseteq W \times W'$ such that the following assertions hold:*

1. $r(\tau, \tau')$
2. $\forall x, x' \;\; \forall \zeta \in H(x) \bigcup H^+(x) \quad r(x, x') \to N, x' \vDash \zeta$
3. $\forall x, y, x' \; R^*(x, y) \land r(x, x') \to \exists y' \; R'^*(x', y') \land r(y, y')$
4. $\forall x, x', y' \; R'^*(x', y') \land r(x, x')$
   $\to \exists y \; (Permanent^*(x, y) \lor (y = x \land \; R(x, y))) \land r(y, y')$
5. $\forall x, y, y' \; x \neq y \land Permanent^*(x, y) \land r(y, y') \to \exists x' \; R'^*(x', y') \land r(x, x')$
6. *For every formula* $\Diamond q$,
   $\forall x, y, x', y' \; (Permanent(x, y) \land Next(x, \Diamond q) = y$
   $\land \; r(x, x') \land R'^*(x', y') \land N, y' \vDash q)$
   $\to r(y, y')$

Note that $R'^* \equiv R'$ since R' is transitive. We write R'^* instead of R' for symmetry.

Proof. For the step 1:

Let u be the world created by $CreateNextMinNode(\tau)$.

Let $r = \{(\tau, \tau')\} \bigcup \{(u, x') \mid R'^*(\tau', x')\}$.

It is obvious that the assertion 1 holds. The assertion 6 holds because the function $Next$ is totally undefined at this step.

2) Suppose that $\zeta \in H(x) \bigcup H^+(x)$ and $r(x, x')$ holds. We show that $N, x' \vDash \zeta$.

Case $x = \tau$: We have $x' = \tau'$ since $r(x, x')$. Case $L = KD4$: We have $\neg R(\tau, \tau)$, hence $H^+(\tau) = \emptyset$ and $\zeta \in H(\tau)$. Since $H(\tau) = P$ and $N \vDash P$, it follows that $N, x' \vDash \zeta$. Case $L = S4$: We have $H(\tau) \bigcup H^+(\tau) = P \bigcup \{\phi \mid \Box\phi \in P\}$. If $\zeta \in P$, we have $N, x' \vDash \zeta$ since $N \vDash P$. If $\Box\zeta \in P$, we have $N, x' \vDash \Box\zeta$ since $N \vDash P$; and $N, x' \vDash \zeta$ since $L = S4$.

Case $x = u$: We have $R'^*(\tau', x')$. Since $\zeta \in H(u) \bigcup H^+(u)$, there are cases: $\Box\zeta \in H(\tau)$, or ($\zeta \in H(\tau)$ and $\zeta = \Box\psi$ for some ψ), or ($\Box\Box\zeta \in H(\tau)$ and $L = S4$). Case $\Box\zeta \in H(\tau)$: We have $N, \tau' \vDash \Box\zeta$, since $H(\tau) = P$ and $N, \tau' \vDash P$. It follows that $N, x' \vDash \zeta$, since $R'^*(\tau', x')$. Case $\zeta = \Box\psi$ and $\zeta \in H(\tau)$: We have $N, \tau' \vDash \Box\psi$, since $H(\tau) = P$ and $N, \tau' \vDash P$. Hence $N, \tau' \vDash \Box\Box\psi$, since N is a $K4$-model. It follows that $N, x' \vDash \zeta$, since $R'^*(\tau', x')$. Case $\Box\Box\zeta \in H(\tau)$ and $L = S4$: We have $N, \tau' \vDash \Box\Box\zeta$, since $H(\tau) = P$ and $N, \tau' \vDash P$. Hence $N, x' \vDash \Box\zeta$, since $R'^*(\tau', x')$. It follows that $N, x' \vDash \zeta$, since N is a $S4$-model.

3) Suppose that $R^*(x,y) \wedge r(x,x')$ holds. We show that $\exists y'\ R'^*(x',y') \wedge r(y,y')$.

Case $x = y = \tau$: It follows that $L = S4$. We have $x' = \tau'$ since $r(x,x')$. For $y' = \tau'$, we have $R'^*(x',y') \wedge r(y,y')$.

Case $x = \tau$ and $y = u$: We have $x' = \tau'$ since $r(x,x')$. There exists y' such that $R'(\tau',y')$, since $KD \le L$. For such y' we have $R'^*(x',y') \wedge r(y,y')$.

Case $x = u$ and $y = u$: We have $R'^*(\tau',x')$ since $r(x,x')$. There exists y' such that $R'(x',y')$, hence $R'^*(\tau',y')$. For such y' we have $R'^*(x',y') \wedge r(y,y')$.

4) Suppose that $R'^*(x',y') \wedge r(x,x')$ holds. We show that $\exists y\ (Permanent^*(x,y) \vee (y = x \wedge R(x,y))) \wedge r(y,y')$.

Case $x = \tau$: It follows that $x' = \tau'$. For $y = u$, we have $Permanent^*(x,y)$ and $r(y,y')$.

Case $x = u$: We have $R'^*(\tau',x')$ since $r(x,x')$, hence $R'^*(\tau',y')$. For $y = u$, we have $y = x \wedge R(x,y) \wedge r(y,y')$.

5) Suppose that $x \neq y \wedge Permanent^*(x,y) \wedge r(y,y')$ holds. We show that $\exists x'\ R'^*(x',y') \wedge r(x,x')$. It is true that $x = \tau$ and $y = u$. We have $R'^*(\tau',y')$ since $r(y,y')$. For $x' = \tau'$, we have $R'^*(x',y') \wedge r(x,x')$.

Fix one of the steps 2a, 2(b)i, 2(b)ii, 2(c)i, 2(c)ii, and suppose that before executing the step, M with $StatusW$, $StatusR$, $Next$, and r satisfies all of the assertions 1 - 6. Let $M_2 = \langle W_2, \tau, R_2, H_2 \rangle$ be the model graph obtained by executing the step. Let $StatusW_2$, $StatusR_2$, $Next_2$ be the functions corresponding to $StatusW$, $StatusR$, $Next$, obtained by executing the step. We construct a relation r_2 such that all of the above mentioned assertions hold for $M_2, StatusW_2, StatusR_2, Next_2, r_2$.

In the remainder of this proof, the phrase "by the assumption about the assertion i", where $1 \le i \le 6$, is understood as "because the assertion i is assumed to hold for M, $StatusW$, $StatusR$, $Next$, and r". When we are proving the assertion i for M_2, $StatusW_2$, $StatusR_2$, $Next_2$, and r_2, the phrase "by assumption" in most of cases stands for "by the assumption about the assertion i".

For the step 2a:

Let $r_2 = r$. We only need to prove the assertion 2.

Suppose that $\zeta \in H_2(x) \bigcup {H_2}^+(x)$ and $r(x,x')$ holds. We show that $N, x' \vDash \zeta$. The assumptions 1 - 4 assert that $M \le N$.

- Case $x = u$ and $\zeta = B$: We have $N, x' \vDash \phi$ and $N, x' \vDash C_i$ for all $1 \le i \le k$, since $r(x,x')$. Therefore $N, x' \vDash B$, and $N, x' \vDash \zeta$.
- Case $x \neq u$ or $\zeta \neq B$: It follows that $\zeta \in H_2(x) \to \zeta \in H(x)$. If B is not of the form $\Box\xi$, or $\neg Permanent^*(u,x)$, or ($B = \Box\xi$, $\zeta \neq \xi$, and $\zeta \neq B$), then we have $\zeta \in {H_2}^+(x) \to \zeta \in H^+(x)$, hence $\zeta \in H(x) \bigcup H^+(x)$, and finally $N, x' \vDash \zeta$. Suppose that $B = \Box\xi$ for some ξ, $Permanent^*(u,x)$, and ($\zeta = \xi$ or $\zeta = B$).

Case $x \neq u$: By the assumption about the assertion 5, there exists u' such that $R'^*(u', x') \wedge r(u, u')$. Thus $N, u' \vDash \phi$ and $\forall i\ 1 \leq i \leq k$ $N, u' \vDash C_i$. It follows that $N, u' \vDash \Box\xi$, hence $N, x' \vDash \xi$ and $N, x' \vDash \Box\xi$, since $R'^*(u', x')$ and $K4 \leq L$. Hence $N, x' \vDash \zeta$.

Case $x = u$: Since $Permanent^*(u, x)$ holds, it follows from Lemma 4.4.2, items 3 and 1 that $L = S4$. Similarly as for the case $x = u$ and $\zeta = B$, we derive $N, x' \vDash B$. The consequence is that $N, x' \vDash \zeta$.

For the step 2(b)i:

By Lemma 4.4.2:2, we have $L = KD4$ since $StatusR(u, u) = temporal$.

Let v be the world created by $CreateNextMinNode(u)$.

Let $r_2 = r \bigcup \{(v, v') \mid \exists u'\ r(u, u') \wedge R'^*(u', v')\}$.

1) The assertion holds by assumption.

2) Suppose that $\zeta \in H_2(x) \bigcup {H_2}^+(x)$ and $r_2(x, x')$ holds. We show that $N, x' \vDash \zeta$.

Case $x \neq v$: We have $H_2(x) \bigcup {H_2}^+(x) = H(x) \bigcup H^+(x)$ and $r(x, x')$. Hence, by assumption, $N, x' \vDash \zeta$.

Case $x = v$: Since $r_2(x, x')$, there exists u' such that $r(u, u') \wedge R'^*(u', x')$. We have ${H_2}^+(v) = H_2(v) = H^*(u)$. Therefore $\Box\zeta \in H(u) \bigcup H^+(u)$ or ($\zeta \in H(u) \bigcup H^+(u)$ and $\zeta = \Box\xi$ for some ξ). Case $\Box\zeta \in H(u) \bigcup H^+(u)$: By assumption, we have $N, u' \vDash \Box\zeta$. Hence $N, x' \vDash \zeta$, since $R'^*(u', x')$. Case $\zeta \in H(u) \bigcup H^+(u)$ and $\zeta = \Box\xi$: By assumption, we have $N, u' \vDash \Box\xi$. Since $K4 \leq L$, we have $N, u' \vDash \Box\Box\xi$. Hence $N, x' \vDash \zeta$, since $R'^*(u', x')$.

3) Suppose that $R_2^*(x, y) \wedge r_2(x, x')$ holds. We show that $\exists y'\ R'^*(x', y') \wedge r_2(y, y')$.

- Case $x \neq v$ and $y \neq v$: We have $R^*(x, y) \wedge r(x, x')$. By assumption and that $r \subseteq r_2$, we derive $\exists y'\ R'^*(x', y') \wedge r_2(y, y')$.

- Case $x = v$: We have $y = v$ and $\exists u'\ r(u, u') \wedge R'^*(u', x')$. Let y' be a world such that $R'(x', y')$. Thus $\exists u'\ r(u, u') \wedge R'^*(u', y')$, and $r_2(v, y')$. Therefore $\exists y'\ R'^*(x', y') \wedge r_2(y, y')$.

- Case $y = v$ and $x \neq v$: Since ${R_2}^*(x, y)$, we have $x = u$ or $R^*(x, u)$.

  Case $x = u$: We have $r(u, x')$ since $r_2(x, x')$. Let y' be a world such that $R'(x', y')$. We have $r(u, x') \wedge R'(x', y')$, hence $r_2(v, y')$. Therefore $\exists y'\ R'^*(x', y') \wedge r_2(y, y')$.

  Case $R^*(x, u)$: We have $r(x, x')$ since $r_2(x, x')$ and $x \neq v$. Hence, by assumption, there exists u' such that $R'^*(x', u') \wedge r(u, u')$. Since $KD \leq L$, there exists y' such that $R'(u', y')$. Thus we have $r(u, u') \wedge R'^*(u', y')$, hence $r_2(v, y')$, and finally $R'^*(x', y') \wedge r_2(y, y')$.

4) Suppose that $R'^*(x', y') \wedge r_2(x, x')$ holds. We show that $\exists y\ ({Permanent_2}^*(x, y) \vee (y = x \wedge R_2(x, y))) \wedge r_2(y, y')$.

Case $x \neq v$ and $x \neq u$: We have $r(x,x')$ since $r_2(x,x')$. The assertion holds by assumption.

Case $x = v$: We have $\exists u'\ r(u,u') \wedge R'^*(u',x')$ since $r_2(x,x')$. Hence $\exists u'\ r(u,u') \wedge R'^*(u',y')$, since $R'^*(x',y')$. It follows that $r_2(v,y')$, and by choosing $y = x$ we have $R_2(x,y) \wedge r_2(y,y')$.

Case $x = u$: We have $R'^*(x',y') \wedge r(u,x')$. Hence $r_2(v,y')$, and by choosing $y = v$ we have $Permanent_2{}^*(x,y) \wedge r_2(y,y')$.

5) Suppose that $x \neq y \wedge Permanent_2{}^*(x,y) \wedge r_2(y,y')$ holds.
We show that $\exists x'\ R'^*(x',y') \wedge r_2(x,x')$. We have $x \neq v$.

Case $y \neq v$: We have $Permanent^*(x,y) \wedge r(y,y')$. By assumption, it follows that $\exists x'\ R'^*(x',y') \wedge r(x,x')$. Hence $\exists x'\ R'^*(x',y') \wedge r_2(x,x')$, since $r \subseteq r_2$.

Case $y = v$: We have $x = u$ or $Permanent^*(x,u)$, since $Permanent_2{}^*(x,y)$. Since $r_2(y,y')$, there exists u' such that $r(u,u') \wedge R'^*(u',y')$. Case $x = u$: For $x' = u'$, we have $R'^*(x',y') \wedge r(x,x')$. Thus $\exists x'\ R'^*(x',y') \wedge r_2(x,x')$. Case $x \neq u$: We have $x \neq u \wedge Permanent^*(x,u) \wedge r(u,u')$, hence, by assumption, $\exists x'\ R'^*(x',u') \wedge r(x,x')$. Thus $\exists x'\ R'^*(x',y') \wedge r(x,x')$, which implies $\exists x'\ R'^*(x',y') \wedge r_2(x,x')$.

6) Suppose that $Permanent_2(x,y) \wedge Next_2(x,\Diamond q) = y \wedge r_2(x,x') \wedge R'^*(x',y') \wedge (N,y' \vDash q)$ holds. We show that $r_2(y,y')$. Since $Next_2(x,\Diamond q) = y$, we have $y \neq v$ and $x \neq v$. It follows that $Permanent(x,y) \wedge Next(x,\Diamond q) = y \wedge r(x,x')$. By assumption, we derive $r(y,y')$, and $r_2(y,y')$.

For the step 2(b)ii:

Since $StatusW(u) = normal$ or $StatusR(u,u) = permanent$ or $\psi \in H(u)$, by Lemma 4.4.2:1, we have $\neg R(u,u)$ or $L = S4$ or $\psi \in H(u)$.

Let $r_2 = r$. It is easily seen that $Permanent_2{}^* = Permanent^*$. We also have $R_2{}^* \subseteq R^*$, and $Next$ is an extension of $Next_2$, i.e. for any $\Diamond p$, $\forall x\ \forall y\ Next_2(x,\Diamond p) = y \rightarrow Next(x,\Diamond p) = y$. Therefore, by assumption, the assertions 1, 3, 5, and 6 hold. It is easily seen that $\forall x\ R(x,x) \rightarrow R_2(x,x)$, hence the assertion 4 also holds.

2) Suppose that $\zeta \in H_2(x) \bigcup H_2{}^+(x)$ and $r(x,x')$ holds. We show that $N,x' \vDash \zeta$.

- Case $\neg Permanent^*(u,x)$: We have $\zeta \in H(x) \bigcup H^+(x)$, and then by assumption, $N,x' \vDash \zeta$.

- Case $Permanent^*(u,x)$:
  We have $H_2(x) \bigcup H_2{}^+(x) \subseteq H(x) \bigcup H^+(x) \bigcup \{\xi \mid \psi = \Box\xi\}$.
  If $\zeta \in H(x) \bigcup H^+(x)$, then by assumption, we have $N,x' \vDash \zeta$. Suppose that $\zeta \notin H(x) \bigcup H^+(x)$ and $\psi = \Box\zeta$. We have $\psi \notin H(u)$.

  - Case $x = u$: By Lemma 4.4.2:3, we have $Permanent(u,u)$, and $L = S4$. By assumption, we have $N,x' \vDash \phi$ since $r(x,x')$. Hence $N,x' \vDash \zeta$.

- Case $x \neq u$: There are two cases: $L = S4$, or $\exists y\ y \neq u \wedge y \neq x \wedge Permanent^*(u, y) \wedge Permanent^*(y, x)$; (for otherwise we would have $\zeta \in H(x) \bigcup H^+(x)$).

  Case $L = S4$: By the assumption about the assertion 5, we have $\exists u'\ R'^*(u', x') \wedge r(u, u')$. By assumption, this implies $\exists u'\ R'^*(u', x') \wedge N, u' \models \phi$. Thus $N, x' \models \zeta$, since $L = S4$.

  The second case : By the assumption about the assertion 5, there exists y' such that $R'^*(y', x') \wedge r(y, y')$, and u' such that $R'^*(u', y') \wedge r(u, u')$. By assumption, we have $N, u' \models \phi$, hence $N, x' \models \zeta$.

For the step 2(c)i:

Let $r_2 = r$. It is easily seen that $Permanent_2{}^* = Permanent^*$. Therefore the assertions 1, 2, 4, 5, and 6 hold by assumption.

3) Suppose that $R_2{}^*(x, y) \wedge r(x, x')$ holds. We show that $\exists y'\ R'^*(x', y') \wedge r(y, y')$. If $R^*(x, y)$ holds, by assumption we have $\exists y'\ R'^*(x', y') \wedge r(y, y')$. Suppose that $R^*(x, y)$ does not hold. Since $R_2{}^*(x, y)$, we have ($x = u$ or $R^*(x, u)$) and ($y = w$ or $R^*(w, y)$). If $x = u$, let $u' = x'$, we have $r(u, u')$. If $R^*(x, u)$, by assumption, there exists u' such that $R'^*(x', u') \wedge r(u, u')$. We have $v \neq u$ because $Next(u, \phi)$ is undefined while $Next(v, \phi)$ is defined. By the assumption about the assertion 5, there exists v' such that $R'^*(v', u') \wedge r(v, v')$. We have $N, u' \models \phi$ since $r(u, u')$ and $\phi \in H(u)$. Therefore there exists w' such that $R'(u', w')$ and $N, w' \models p$. We have $Permanent(v, w) \wedge Next(v, \Diamond p) = w \wedge r(v, v') \wedge R'^*(v', w') \wedge N, w' \models p$, hence, by the assumption about the assertion 6, $r(w, w')$ holds.

If $y = w$, we have $\exists y' = w'\ R'^*(x', y') \wedge r(y, y')$. If $y \neq w$, we have $R^*(w, y)$, hence, by assumption, $\exists y'\ R'^*(w', y') \wedge r(y, y')$, and finally $\exists y'\ R'^*(x', y') \wedge r(y, y')$.

For the step 2(c)ii:

Let v be the world mentioned in the step, and let w be the world created by $CreateNextMinNode(v)$. Let

$$\begin{aligned} r_2 = r\ &\bigcup \{(v, v') \mid \exists u'\ r(u, u') \wedge R'^*(u', v') \wedge N, v' \models p\} \\ &\bigcup \{(w, w') \mid \exists u', v'\ r(u, u') \wedge R'^*(u', v') \wedge N, v' \models p \wedge R'^*(v', w')\} \end{aligned}$$

1) The assertion holds by assumption.

2) Suppose that $\zeta \in H_2(x) \bigcup H_2{}^+(x)$ and $r_2(x, x')$ holds. We show that $N, x' \models \zeta$.

- Case $x \neq v$ and $x \neq w$: We have $H_2(x) \bigcup H_2{}^+(x) = H(x) \bigcup H^+(x)$ and $r(x, x')$. Therefore, by assumption, $N, x' \models \zeta$.

- Case $x = v$: Since $r_2(x, x')$, there exists u' such that $r(u, u') \wedge R'^*(u', x') \wedge N, x' \models p$. Because $\zeta \in H_2(x) \bigcup {H_2}^+(x)$ and $H_2(v) = H^*(u) \bigcup \{p\}$, there are the following cases:
  - Case $\zeta \in H(u) \bigcup H^+(u)$ and $\zeta = \Box\xi$ for some ξ : By assumption, we have $N, u' \models \Box\xi$. It follows that $N, u' \models \Box\Box\xi$, since $K4 \leq L$. Hence $N, x' \models \zeta$, since $R'^*(u', x')$.
  - Case $\Box\zeta \in H(u) \bigcup H^+(u)$: By assumption, we have $N, u' \models \Box\zeta$. Hence $N, x' \models \zeta$, since $R'^*(u', x')$.
  - Case $\Box\Box\zeta \in H(u) \bigcup H^+(u)$ and $L = S4$: By assumption, we have $N, u' \models \Box\Box\zeta$. Hence $N, u' \models \Box\zeta$, since $L = S4$. Consequently, $N, x' \models \zeta$, since $R'^*(u', x')$.
  - Case $\zeta = p$: We have $N, x' \models \zeta$ since $N, x' \models p$.
- Case $x = w$: Since $r_2(x, x')$, there exist u' and v' such that $r(u, u') \wedge R'^*(u', v') \wedge N, v' \models p \wedge R'^*(v', x')$. Since $\zeta \in H_2(x) \bigcup {H_2}^+(x)$, there are the following cases:
  - Cases ($\zeta \in H(u) \bigcup H^+(u)$ and $\zeta = \Box\xi$ for some ξ) or ($\Box\zeta \in H(u) \bigcup H^+(u)$): Reasoning as for the case $x = v$, we obtain $N, x' \models \zeta$.
  - Case $\Box\Box\zeta \in H(u) \bigcup H^+(u)$: By assumption, we have $N, u' \models \Box\Box\zeta$. Hence $N, x' \models \zeta$, since $R'^*(u', v')$ and $R'^*(v', x')$.
  - Case $\Box^3\zeta \in H(u) \bigcup H^+(u)$ and $L = S4$: By assumption, we have $N, u' \models \Box^3\zeta$. Hence $N, u' \models \Box\zeta$, since $L = S4$. Consequently, $N, x' \models \zeta$, since $R'^*(u', x')$.

3) Suppose that ${R_2}^*(x, y) \wedge r_2(x, x')$ holds. We show that $\exists y'\ R'^*(x', y') \wedge r_2(y, y')$.

- Case $x \neq v$ and $x \neq w$: It follows that $r(x, x')$.

  Case $y \neq v$ and $y \neq w$: It follows that $R^*(x, y)$. By assumption, we have $\exists y'\ \ R'^*(x', y') \wedge r(y, y')$, which implies $\exists y'\ R'^*(x', y') \wedge r_2(y, y')$.

  Case $y = v$ or $y = w$: Since ${R_2}^*(x, y)$, we have $x = u$ or $R^*(x, u)$. If $x = u$ then let $u' = x'$, and we have $r(u, u')$. If $R^*(x, u)$, then by assumption, there exists u' such that $R'^*(x', u') \wedge r(u, u')$. Hence, in both of the cases, $N, u' \models \phi$. Therefore there exists v' such that $R'(u', v')$ and $N, v' \models p$. Let w' be a world such that $R'(v', w')$. We have $r_2(v, v')$ and $r_2(w, w')$. If $y = v$ then choose $y' = v'$, if $y = w$ then choose $y' = w'$. Thus $R'^*(x', y') \wedge r_2(y, y')$.
- Case $x = v$: We have $y = v$ or $y = w$. Since $r_2(x, x')$, we have $\exists u'\ r(u, u') \wedge R'^*(u', x') \wedge N, x' \models p$. Case $y = v$: Since ${R_2}^*(x, y)$, we have $L = S4$. Therefore, for $y' = x'$, we have $R'^*(x', y') \wedge r_2(y, y')$.

Case $y = w$: Let y' be a world such that $R'(x', y')$. It is easily seen that $R'^*(x', y') \wedge r_2(y, y')$ holds.

- Case $x = w$: We have $y = w$, and $\exists u', v'\ r(u, u') \wedge R'^*(u', v') \wedge (N, v' \vDash p) \wedge R'^*(v', x')$. Let y' be a world such that $R'(x', y')$. It is easily seen that $R'^*(x', y') \wedge r_2(y, y')$ holds.

4) Suppose that $R'^*(x', y') \wedge r_2(x, x')$ holds. We show that $\exists y\ (Permanent_2{}^*(x, y) \vee (y = x \wedge R_2(x, y))) \wedge r_2(y, y')$.

Case $x \neq v$ and $x \neq w$: The assertion holds by assumption.

Case $x = v$: Since $r_2(x, x')$, we have $\exists u'\ r(u, u') \wedge R'^*(u', x') \wedge (N, x' \vDash p)$. Hence $r_2(w, y')$, since $R'^*(x', y')$. Therefore $\exists y = w\ Permanent_2{}^*(x, y) \wedge r_2(y, y')$.

Case $x = w$: Since $r_2(x, x')$, we have $\exists u', v'\ r(u, u') \wedge R'^*(u', v') \wedge (N, v' \vDash p) \wedge R'^*(v', x')$. Hence $r_2(w, y')$, since $R'^*(x', y')$. By choosing $y = w$ we have $y = x \wedge R_2(x, y) \wedge r_2(y, y')$.

5) Suppose that $x \neq y \wedge Permanent_2{}^*(x, y) \wedge r_2(y, y')$ holds. We show that $\exists x'\ R'^*(x', y') \wedge r_2(x, x')$.

Case $y \neq v$ and $y \neq w$: The assertion holds by assumption.

Case $y = v$: We have $x = u$ or $Permanent^*(x, u)$. Since $r_2(y, y')$, there exists u' such that $r(u, u') \wedge R'^*(u', y') \wedge N, y' \vDash p$. If $x = u$, for $x' = u'$, we have $R'^*(x', y') \wedge r_2(x, x')$. If $x \neq u$, we have $Permanent^*(x, u)$, hence, by assumption, $\exists x'\ R'^*(x', u') \wedge r(x, x')$, and finally $\exists x'\ R'^*(x', y') \wedge r_2(x, x')$.

Case $y = w$: We have $x = v$ or $x = u$ or $Permanent^*(x, u)$. Since $r_2(y, y')$, there exist u' and v' such that $r(u, u') \wedge R'^*(u', v') \wedge (N, v' \vDash p) \wedge R'^*(v', y')$. If $x = v$ then choose $x' = v'$, if $x = u$ then choose $x' = u'$; we will have $R'^*(x', y') \wedge r_2(x, x')$. Now suppose that $x \neq v$ and $x \neq u$. We have $Permanent^*(x, u)$. Hence, by assumption, $\exists x'\ R'^*(x', u') \wedge r(x, x')$, and finally $\exists x'\ R'^*(x', y') \wedge r_2(x, x')$.

6) Suppose that $Permanent_2(x, y) \wedge Next_2(x, \Diamond q) = y \wedge r_2(x, x') \wedge R'^*(x', y') \wedge (N, y' \vDash q)$ holds. We show that $r_2(y, y')$ holds. Since $Next_2(x, \Diamond q)$ is defined, we have $x \neq v$ and $x \neq w$. It follows that $r(x, x')$. Case $x \neq u$ or ($x = u$ and $y \neq v$): We have $Next(x, \Diamond q) = Next_2(x, \Diamond q) = y$, and $Permanent(x, y)$. Hence, by assumption, $r(y, y')$, and then $r_2(y, y')$. Case $x = u$ and $y = v$: We have $q = p$, hence $r_2(y, y')$. □

Corollary 4.4.5 *Let M be the model obtained by executing Algorithm 4.4.1 for a positive program P in a modal logic L. Then M is less than or equal to every L-model of P.*

We now estimate the complexity of the given algorithm.

Definition 4.4.6 We define the p-height of a model $M = \langle W, \tau, R, h \rangle$ wrt. a binary relation $StatusR : W \times W \rightarrow \{permanent, temporal\}$ to be the maximal number k such that there are k different worlds $w_1, \ldots, w_k$ from

W such that:
$Permanent(\tau, w_1) \wedge Permanent(w_1, w_2) \wedge \ldots \wedge Permanent(w_{k-1}, w_k)$.

In the remainder of this section we assume that at the step 2 of Algorithm 4.4.1, every world x such that $Permanent^*(x, u)$ has been previously processed.

Lemma 4.4.7 *Let $M = \langle W, \tau, R^*, h\rangle$ and StatusR be obtained as a result of executing Algorithm 4.4.1 for a positive program P. Let k be the modal depth of P, n be the number of rules in P of the form $\Box^s(\Box A \leftarrow B_1, \ldots, B_p)$, where $s \geq 1$, $p \geq 1$, and let m be the number of rules in P of the form $\Box^s(\Diamond A \leftarrow B_1, \ldots, B_p)$, where $s \geq 1$, $p \geq 1$. Then the p-height of M wrt. StatusR is less than $k + (n+1).(2m+1)$.*

Proof. When a new world x is created, define $f(x)$, $f_1(x)$, $f_2(x)$, and $H_0(x)$ as follows:

$$\begin{aligned}
H_0(x) &= \{\Box\phi \mid \exists y\ (y = x \vee Permanent^*(y, x)) \wedge \Box\phi \in H(y)\} \\
f_1(x) &= \#\{y \in W \mid\ (y = x \vee Permanent^*(y, x)) \wedge H_0(y) = H_0(x) \\
&\qquad\qquad\qquad \text{and } y \text{ was created at the step 2c } \} \\
f_2(x) &= \#\{y \in W \mid\ (y = x \vee Permanent^*(y, x)) \\
&\qquad\qquad\qquad \text{and } y \text{ was created at the step 2(b)i } \} \\
f(x) &= |H_0(x)|.2m + f_1(x) + f_2(x)
\end{aligned}$$

For a world x, we define $depth(x)$ to be the minimal number i such that there are i different worlds $w_1, \ldots, w_i$ such that: $w_i = x \wedge Permanent(\tau, w_1) \wedge Permanent(w_1, w_2) \wedge \ldots \wedge Permanent(w_{i-1}, w_i)$.

For every $x \in W$ such that x is created at the step 2c, at the moment of creating x we have $\forall y\ Permanent^*(y, x) \wedge H_0(y) = H_0(x) \rightarrow H^*(y) = H^*(x)$. We claim that $\forall x\ depth(x) \geq k \rightarrow f_1(x) \leq 2m$. As a consequence, we have $\forall x, y\ \ depth(x) \geq k \wedge Permanent(x, y) \rightarrow f(x) < f(y)$.

We now show that $\forall x, y\ depth(x) \geq k \wedge Permanent^*(x, y) \rightarrow (f_2(y) - f_2(x) \leq n)$. Suppose that $depth(u) \geq k$ and it is the time we are creating a new world v at the step 2(b)i by $CreateNextMinNode(u)$. Since $depth(u) \geq k \wedge \Box\psi \in H(u) \wedge \psi \notin H(u)$, we claim that: ψ must be a primitive proposition, lets say p, and $\Box p$ has been added to $H(u)$ at the step 2a. Moreover, by the assumption made immediately above this lemma, we have $\forall x\ \ Permanent^*(x, u) \rightarrow \Box p \notin H(x)$. We conclude that $\forall x, y\ depth(x) \geq k \wedge Permanent^*(x, y) \rightarrow (f_2(y) - f_2(x) \leq n)$.

Let $t > k$, and let w_k and w_t be different worlds such that $Permanent^k(\tau, w_k)$ and $Permanent^{t-k}(w_k, w_t)$.
We have $|H_0(w_t) - H_0(w_k)| \leq n$, hence
$f(w_t) - f(w_k) \leq (n+1).2m + f_2(w_t) - f_2(w_k) < (n+1).(2m+1)$.
Since $\forall x, y\ depth(x) \geq k \wedge Permanent(x, y) \rightarrow f(x) < f(y)$, we derive that $t - k \leq f(w_t) - f(w_k) < (n+1).(2m+1)$, and finally $t < k + (n+1).(2m+1)$. This completes our proof. $\Box$

Lemma 4.4.8 *If during execution of Algorithm 4.4.1, the p-height of M wrt. StatusR is always less than t, then the algorithm terminates in $O(n^{3t+2})$ steps. (Here n is the size of P).*

Proof. The number of worlds in M is always bounded by n^t. For any action of adding a formula $\Box\phi$ to the model graph, we need no more than $O(n^{t+1})$ units of time to update the model graph with respect to the action. Therefore the total time of executions of the step 2b is bounded by $O(n^{2t+2})$.

We pay $O(n^{2t+1})$ units of time to extend the model graph at the step 2a or to certify that no more changes can be made by the step 2a at the moment.

We pay $O(t * n^{t+1})$ units of time to extend the model graph at the step 2c or to satisfy all formulae of the form $\Diamond q$ at every worlds.

There are no more than $O(n^{t+1})$ times the model graph is extended by the steps 2a and 2c, therefore the algorithm terminates in $O(n^{3t+2})$ steps. $\Box$

Here is the main result of this section:

Theorem 4.4.9 *For any positive program P in a modal logic $L \in \{KD4, S4\}$, there exists the least L-model of P. Moreover, if the modal depth of P and the number of rules of the form $\Box^s(A \leftarrow B_1, \ldots, B_p)$, where $s \geq 1$, $p \geq 1$, and A is of the form $\Box p$ or $\Diamond p$, are finitely bounded, then the least L-model can be constructed in PTIME and its size is bounded by a polynomial in the size of P.*

Note that the modal depth of any program in $S4$ can be assumed to be less than or equal to 2, since in $S4$ we have $\Box\Box\phi \equiv \Box\phi$. The theorem immediately follows from Lemmas 4.4.7, 4.4.8, 4.4.3, and Corollary 4.4.5.

Since in $S4$ we have $\Box\Box\phi \equiv \Box\phi$, it follows that the problem of checking satisfiability of a set of Horn formulae with finitely bounded modal depth in $S4$ is PSPACE-complete. It is easily seen that if the least model constructed by Algorithm 4.4.1 always had polynomial size, then the algorithm would terminate in polynomial time, and the problem of checking satisfiability of a set of Horn formulae with finitely bounded modal depth in $S4$ would be decidable in PTIME. That is why the least model constructed by Algorithm 4.4.1 can be large.

4.5 A Problem with Positive Programs in *K* and *K*4

In this section, we show that there are positive programs that cannot be *characterized* (see Definition 4.5.1) in the logic K by a finitely bounded number of models. We also give a positive program that cannot be characterized in $K4$ by a finite set of models. In particular, there is a positive program that has neither the least K-model nor the least $K4$-model.

Definition 4.5.1 We say that a positive program P is *characterized* in a normal modal logic L by a set $\mathcal{M}$ of models if $\mathcal{M}$ contains only L-models of P, and for any positive formula ϕ, $P \vDash_L \phi$ iff ϕ is satisfied in every model from $\mathcal{M}$.

Definition 4.5.2 For a Kripke model $M = \langle W, \tau, R, h \rangle$, we define the *short diameter* of M to be the least number k such that there exists a world $w \in W$ such that $R^k(\tau, w)$ and there are no worlds reachable from w. If such k does not exist then we define the short diameter of M to be infinite.

We show that the program $P = \{\Box p, \Box^2 p, \ldots, \Box^n p\}$, $n \geq 1$, cannot be characterized in K by less than $n+1$ models. Suppose oppositely that it can. Without loss of generality we can assume that P is characterized in K by exactly n models, denoted by $M_i = \langle W_i, \tau_i, R_i, h_i \rangle$, for $1 \leq i \leq n$. Let $k \geq 0$ be the least number such that k is not the short diameter of any model M_i, for $1 \leq i \leq n$. It is clear that $k \leq n$. Let $\phi = \Box q \vee \Diamond \Box q \vee \ldots \vee \Diamond^{k-1} \Box q \vee \Diamond^{k+1} p$. It is easy to check that $M_i \vDash \phi$ for all $1 \leq i \leq n$, hence $P \vDash_K \phi$. Consider the model $M = \langle \{\tau, 1, 2, \ldots, k\}, \tau, R, h \rangle$ with $R = \{(\tau, 1), (1, 2), \ldots, (k-1, k)\}$ and $h(w) = \{p\}$ for any $w \in W$. It is easily seen that $M \vDash P$ but $M \nvDash \phi$, hence $P \nvDash_K \phi$, which is a contradiction.

We now show that the program $\{\Box p\}$ cannot be characterized in $K4$ by a finite set of models. Suppose oppositely that $\{\Box p\}$ is characterized in $K4$ by n models, denoted by $M_i = \langle W_i, \tau_i, R_i, h_i \rangle$, for $1 \leq i \leq n$. If in each of these models there exists an infinite path starting from the actual world, then $M_i \vDash \Diamond p$ for all $1 \leq i \leq n$, and hence $\Box p \vDash_{K4} \Diamond p$, which is a contradiction. So, we can assume that between the models there are some without infinite paths starting from the actual world; let all of them be $M_{i_1}, \ldots, M_{i_k}$, where $k \geq 1$ and $1 \leq i_1 < \ldots < i_k \leq n$. Let $m > 0$ be a number such that there is no h, $1 \leq h \leq k$, and no world $w \in W_{i_h}$ such that ${R_{i_h}}^m(\tau_{i_h}, w)$. Let $\phi = \Box^m \Diamond q \vee \Diamond^{m+1} p$. It is easily seen that $M_i \vDash \phi$ for all $1 \leq i \leq n$. It follows that $\Box p \vDash_{K4} \phi$, which is false (to see this just consider the model $M = \langle \{\tau, 1, 2, \ldots, m\}, \tau, R, h \rangle$ with $R = Ext_{K4}(\{(\tau, 1), (1, 2), \ldots, (m-1, m)\})$ and $h(w) = \{p\}$ for any $w \in W$). Therefore the program $\{\Box p\}$ cannot be characterized in $K4$ by a finite set of models.

As a consequence, the program $\{\Box p\}$ has neither the least K-model nor the least $K4$-model.

4.6 Conclusions

We have given algorithms that for a given positive program P in a modal logic $L \in \{KD, T, KDB, B, KD5, KD45, S5, KD4, S4\}$ will construct the least L-model of P. If $L \in \{KD5, KD45, S5\}$, or $L \in \{KD, T\}$ and the modal depth of P is finitely bounded, then the least L-model of P can be constructed in PTIME, and has size bounded by a polynomial. If $L \in \{KDB, B\}$ and the modal depth of P is finitely bounded, then the least L-model of P can be constructed in PTIME, and treated as a finite model with size bounded by a polynomial. We also show that if $L \in \{K5, K45, KB5\}$, or L is KB and the modal depth of P is finitely bounded, then any positive modal logic program P can be characterized by one or two L-models, which can be constructed in PTIME, in the sense that a positive formula follows from P iff it is satisfied in these models. As a consequence, the problem of checking satisfiability of a set of modal Horn formulae with finitely bounded modal depth in KD, T, KB, KDB, or B is decidable in PTIME. The results of other authors [7, 4] such that the problem of checking satisfiability of a set of Horn formulae in the modal logics $K5$, $KD5$, $K45$, $KD45$, $KB5$, and $S5$ is decidable in PTIME has been also studied in this work by a different method.

The results of this chapter will be used in the next one to develop a real modal logic query language, to construct algorithms for answering queries, and to analyze complexity of the algorithms.

Chapter 5

A Modal Logic Query Language

In the previous chapter we have dealt with positive *propositional* modal logic programs. A real query language, however, should allow *universal first-order* rules. Such query language for modal logics, called MDatalog, is proposed in this chapter. We recall in the first section the definition of fixed-domain first-order modal logics over signatures without functions. MDatalog programs will be interpreted in such logics. In Section 5.3 we define an order between first-order Kripke models in a similar way as we do for propositional Kripke models. Existence of the least models of MDatalog programs and computational complexity for answering queries are analyzed in Section 5.4. In the next section we discuss how algorithms of constructing the least models for MDatalog programs can be implemented, and propose a technique that optimizes both execution time and memory for the algorithms. Some examples of application are given in the last section.

5.1 Fixed-domain First-order Modal Logics over Signatures without Functions

Our language contains a signature without functions Σ, a countable set of variables, classical connectives $\neg$, $\vee$, $\wedge$ and $\to$, modal operators $\Box$ and $\Diamond$, quantifiers $\forall$ and $\exists$, and parentheses), (.

A *term* is either a constant symbol or a variable. We call $R_i(t_1, \ldots, t_n)$, where R_i is a relation symbol of arity n and $t_1, \ldots, t_n$ are terms, a *classical atom*.

A *first-order modal formula*, hereafter simply called a formula if there is no ambiguity, is any finite sequence of symbols of our language obtained by applying the following rules: any classical atom is a formula, and if ϕ and ψ are formulae, and x is a variable, then $\neg\phi$, $\phi \vee \psi$, $\phi \wedge \psi$, $\phi \to \psi$, $\Box\phi$, $\Diamond\phi$, $\forall x\ \phi$, and $\exists x\ \phi$ are formulae.

A formula is called *ground* if it contains no quantifiers and no variables. *Herbrand base* is the set of all ground classical atoms.

In this chapter, we use letters a, b, c to denote constant symbols.

Definition 5.1.1 A *fixed-domain first-order Kripke model over a signature without functions* Σ, hereafter simply called first-order Kripke model or just Kripke model if there is no ambiguity, is a tuple $M = \langle D, W, \tau, R, m\rangle$, where D is a set called *domain*, $\langle W, \tau, R\rangle$ is a Kripke frame, and m is an interpretation of constant symbols and relation symbols. For a constant symbol c_i, $m(c_i)$ is an element of D, denoted also by ${c_i}^M$. For a relation symbol R_j of arity n and a world $w \in W$, $m(w)(R_j)$ is a n-ary relation on D, denoted also by ${R_j}^{M,w}$.

Definition 5.1.2 A *variable assignment* on a Kripke model M is a function V that maps each variable x to an element of the domain of M. The value of a term t wrt. a variable assignment V is denoted by $V(t)$ and defined as follows: If t is a constant symbol c_i then $V(t) = {c_i}^M$; if t is a variable x then $V(t) = V(x)$.

Definition 5.1.3 The *satisfiability relation* $M, V, w \vDash \phi$, where $M = \langle D, W, \tau, R, m\rangle$ is a Kripke model, V is a variable assignment, $w \in W$, and ϕ is a formula, is recursively defined as follows

$$
\begin{array}{lll}
M,V,w \vDash R_j(t_1,\dots,t_n) & \text{iff} & {R_j}^{M,w}(V(t_1),\dots,V(t_n)); \\
M,V,w \vDash \neg\phi & \text{iff} & M,V,w \nvDash \phi; \\
M,V,w \vDash \phi \wedge \psi & \text{iff} & M,V,w \vDash \phi \text{ and } M,V,w \vDash \psi; \\
M,V,w \vDash \phi \vee \psi & \text{iff} & M,V,w \vDash \phi \text{ or } M,V,w \vDash \psi; \\
M,V,w \vDash \phi \to \psi & \text{iff} & M,V,w \nvDash \phi \text{ or } M,V,w \vDash \psi; \\
M,V,w \vDash \Box\phi & \text{iff} & \text{for all } u \in W \text{ such that } R(w,u),\ M,V,u \vDash \phi; \\
M,V,w \vDash \Diamond\phi & \text{iff} & \text{there exists some } u \in W \text{ such that} \\
 & & R(w,u) \text{ and } M,V,u \vDash \phi; \\
M,V,w \vDash \forall x\ \phi(x) & \text{iff} & \text{for all } a \in D,\ (M,V',w \vDash \phi(a)), \\
 & & \text{where } V'(x) = a \text{ and } V'(y) = V(y) \text{ for } y \neq x; \\
M,V,w \vDash \exists x\ \phi(x) & \text{iff} & \text{there exists } a \in D \text{ such that } M,V',w \vDash \phi(a), \\
 & & \text{where } V'(x) = a \text{ and } V'(y) = V(y) \text{ for } y \neq x.
\end{array}
$$

If $M, V, w \vDash \phi$ then we say that ϕ is satisfied at w in M wrt. V. We say that ϕ is satisfied in M wrt. V, and write $M, V \vDash \phi$, if $M, V, \tau \vDash \phi$. We say that M satisfies ϕ, or ϕ is satisfied in M, and write $M \vDash \phi$, if for any variable assignment V we have $M, V \vDash \phi$. If ϕ is a ground formula then we write $M, w \vDash \phi$ to denote that $M, V, w \vDash \phi$ for some V (note that what V does not matter).

Let L be the name of some propositional normal modal logic, e.g. KD. To define what is the *first-order modal logic* L we define the class of admissible interpretations to be the class of Kripke models $M = \langle D, W, \tau, R, m\rangle$ with $\langle W, \tau, R\rangle$ being a L-frame.

The definitions of the modal depth of a first-order formula, L-satisfiability, and L-models, for a first-order normal modal logic L, are straightforward.

If X is a set of formulae and ϕ is a formula then we write $X \vDash_L \phi$ to denote the fact that for any L-model M, if $M \vDash X$ then $M \vDash \phi$.

5.2 The Modal Query Language MDatalog

In this section we extend the definition of positive propositional modal logic programs to formulate first-order ones, called MDatalog programs. We also define what is a query to a MDatalog program.

Definition 5.2.1 A *MDatalog program* is a finite set of rules of the form

$$\forall \overline{x}\ \Box^s(A \leftarrow B_1, \ldots, B_k)$$

where $s \geq 0$, $k \geq 0$, and

- A, B_1, ..., B_k are formulae of the forms $\Box\phi$, $\Diamond\phi$, ϕ, with ϕ being a classical atom.
- $\overline{x}$ are all variables occurring in $(A \leftarrow B_1, \ldots, B_k)$.
- If $k = 0$ then the rule is a ground fact, i.e. there is no quantifiers prefixing the rule.
- All variables occurring in A also occur in $(B_1, \ldots, B_k)$.

Definition 5.2.2 A *query* is a formula of the form $\exists \overline{x}\ \phi(\overline{x})$, where ϕ is a positive formula without quantifiers, and $\overline{x}$ is a vector of all variables occurring in ϕ, and can be empty.

Fix a modal logic L. Given a MDatalog program P and a query $\exists \overline{x}\ \phi(\overline{x})$, the goal is to check whether $P \vDash_L \exists \overline{x}\ \phi(\overline{x})$, and to find tuples of constant symbols $\overline{c}$ such that $P \vDash_L \phi(\overline{c})$.

5.3 Ordering First-order Kripke Models

We can order first-order Kripke models similarly as we do for propositional Kripke models. The conditions about frame, i.e. the first three ones in the following definition, remain unchanged.

Definition 5.3.1 Let $M = \langle D, W, \tau, R, m\rangle$ and $N = \langle D', W', \tau', R', m'\rangle$ be Kripke models. We say that M is less than or equal to N with respect to a function $f : D \to D'$ and a binary relation $r \subseteq W \times W'$ if

1. $r(\tau, \tau')$
2. $\forall x, x', y \;\; R(x,y) \wedge r(x,x') \rightarrow \exists y' \; R'(x',y') \wedge r(y,y')$
3. $\forall x, x', y' \;\; R'(x',y') \wedge r(x,x') \rightarrow \exists y \; R(x,y) \wedge r(y,y')$
4. For any constant symbol c_i, $f({c_i}^M) = {c_i}^N$.
5. For any $x \in W$ and $x' \in W'$ such that $r(x,x')$, for any n-ary relation symbol R_j and elements $a_1, \ldots, a_n$ of D, if ${R_j}^{M,x}(a_1, \ldots, a_n)$ then ${R_j}^{N,x'}(f(a_1), \ldots, f(a_n))$.

We say that a model M is less then or equal to N, and write $M \leq N$, if M is less then or equal to N with respect to some f and some r.

Lemma 5.3.2 *Let $M = \langle D, W, \tau, R, m \rangle$ and $N = \langle D', W', \tau', R', m' \rangle$ be Kripke models. Let $M \leq N$ wrt. f and r. Let $u \in W$ and $u' \in W'$ be worlds such that $r(u,u')$. Then for any positive ground formula ϕ, if $M, u \models \phi$ then $N, u' \models \phi$.*

Proof. It suffices to prove this lemma for the case when ϕ is a classical atom; for other cases, see the proof of Lemma 4.1.3. We have:

$$\begin{array}{rl} & M, u \models R_j(c_{k_1}, \ldots, c_{k_n}) \\ \Longrightarrow & {R_j}^{M,u}({c_{k_1}}^M, \ldots, {c_{k_n}}^M) \\ \Longrightarrow & {R_j}^{N,u'}(f({c_{k_1}}^M), \ldots, f({c_{k_n}}^M)) \\ \Longrightarrow & {R_j}^{N,u'}({c_{k_1}}^N, \ldots, {c_{k_n}}^N) \\ \Longrightarrow & N, u' \models R_j(c_{k_1}, \ldots, c_{k_n}) \end{array}$$

□

Corollary 5.3.3 *If $M \leq N$ and ϕ is a positive ground formula satisfied in M, then $N \models \phi$.*

The relation $\leq$ defined in this section is a pre-order. The proof of this fact is similar to the proof of Proposition 4.1.4.

5.4 Instantiating MDatalog Programs

Given a MDatalog program, one can translate it to a positive propositional modal logic program by generating all ground instances of its rules. Theorems about existence of the least models and complexity results for positive propositional modal logic programs can be then reformulated for MDatalog programs.

Lemma 5.4.1 *Let $M_0 = \langle W, \tau, R, h\rangle$ be a propositional Kripke model over the language that takes Herbrand base as the set of primitive propositions, and let $M = \langle D, W, \tau, R, m\rangle$ be a first-order Kripke model such that for any n-ary relation symbol R_j and constant symbols c_{k_1}, ..., c_{k_n}, and for any world $w \in W$*

$$R_j{}^{M,w}(c_{k_1}{}^M, \ldots, c_{k_n}{}^M) \equiv h(w)(R_j(c_{k_1}, \ldots, c_{k_n})).$$

Then for any ground formula ϕ, $M \vDash \phi$ iff $M_0 \vDash \phi$.

Proof. We show that for any world $w \in W$ and for any ground formula ϕ, $M, w \vDash \phi$ iff $M_0, w \vDash \phi$. It suffices to show this for ϕ being a classical atom; other cases can be easily proved by induction. We have:

$$\begin{array}{ll} & M, w \vDash R_j(c_{k_1}, \ldots, c_{k_n}) \\ \Longleftrightarrow & R_j{}^{M,w}(c_{k_1}{}^M, \ldots, c_{k_n}{}^M) \\ \Longleftrightarrow & h(w)(R_j(c_{k_1}, \ldots, c_{k_n})) \\ \Longleftrightarrow & M_0, w \vDash R_j(c_{k_1}, \ldots, c_{k_n}) \end{array}$$

□

Corollary 5.4.2 *Let $M_0 = \langle W, \tau, R, h\rangle$ be a model such as in the preceding lemma. Let $M_1 = \langle Const(\Sigma), W, \tau, R, m\rangle$ be the first-order Kripke model such that $Const(\Sigma)$ is the set of all constant symbols of Σ, $c_i{}^{M_1} = c_i$ for any constant symbol c_i, and that for any n-ary relation symbol R_j and constant symbols c_{k_1}, ..., c_{k_n}, for any world $w \in W$,*

$$R_j{}^{M_1,w}(c_{k_1}, \ldots, c_{k_n}) \equiv h(w)(R_j(c_{k_1}, \ldots, c_{k_n})).$$

Then for any ground formula ϕ, $M_1 \vDash \phi$ iff $M_0 \vDash \phi$.

In fact, the models M_0 and M_1 in the above corollary are the same model, looked at different views. So, we can treat any propositional Kripke model over the language that takes Herbrand base as the set of primitive propositions as a first-order Kripke model.

Definition 5.4.3 For a MDatalog program P, we define the *ground instantiation of P*, denoted by P_0, to be the set of all ground instances of all rules of P, substituted over the set of constant symbols occurring in P. We can treat P_0 as a positive propositional modal logic program that takes Herbrand base as the set of primitive propositions.

Theorem 5.4.4 *Let P be a MDatalog program and P_0 be its ground instantiation. Let L be some normal modal logic. If M_0 is the least propositional L-model of P_0 then M_0 is the least first-order L-model of P.*

Proof. Let $M_0 = \langle W, \tau, R, h\rangle$, and let M_1 be the model defined as in Corollary 5.4.2. We show that M_1 is the least L-model of P. Let $N = \langle D', W', \tau', R', m'\rangle$ be an arbitrary first-order L-model of P. Let $N_0 = \langle W', \tau', R', h'\rangle$ be the propositional model such that for any n-ary relation symbol R_j and constant symbols $c_{k_1}, \ldots, c_{k_n}$, for any world $w \in W$

$$h'(w)(R_j(c_{k_1}, \ldots, c_{k_n})) \equiv {R_j}^{N,w}({c_{k_1}}^N, \ldots, {c_{k_n}}^N).$$

By Lemma 5.4.1, N_0 is a model of P_0. Hence there exists a relation $r \subseteq W \times W'$ such that $M_0 \leq N_0$ wrt. r. Let $f : Const(\Sigma) \to D'$ be the function defined by $f(c_i) = {c_i}^N$, for any constant symbol c_i. We show that $M_1 \leq N$ wrt. f and r. It suffices to show that for any $x \in W$ and $x' \in W'$ such that $r(x, x')$, for any n-ary relation symbol R_j and constant symbols $c_{k_1}, \ldots, c_{k_n}$, if ${R_j}^{M_1,x}(c_{k_1}, \ldots, c_{k_n})$ then ${R_j}^{N,x'}(f(c_{k_1}), \ldots, f(c_{k_n}))$. We have

$$\begin{array}{ll} & {R_j}^{M_1,x}(c_{k_1}, \ldots, c_{k_n}) \\ \Longrightarrow & h(x)(R_j(c_{k_1}, \ldots, c_{k_n})) \\ \Longrightarrow & h(x')(R_j(c_{k_1}, \ldots, c_{k_n})) \\ \Longrightarrow & {R_j}^{N,x'}({c_{k_1}}^N, \ldots, {c_{k_n}}^N) \\ \Longrightarrow & {R_j}^{N,x'}(f(c_{k_1}), \ldots, f(c_{k_n})). \end{array}$$

This completes our proof. □

Corollary 5.4.5 *Let L be one of the modal logics KD, T, KDB, B, $KD5$, $KD45$, $S5$, $KD4$, and $S4$. Let P be a MDatalog program. There exists the least L-model of P and it can be effectively constructed.*

Moreover, if the quantifier depth of P is finitely bounded then

- *If L is one of the logics $KD5$, $KD45$, and $S5$, then the least L-model of P can be constructed in PTIME and its size is bounded by a polynomial in the size of P.*

- *If L is KD or T, and the modal depth of P is finitely bounded then the least L-model of P can be constructed in PTIME and its size is bounded by a polynomial in the size of P.*

- *If L is KDB or B, and the modal depth of P is finitely bounded then the least L-model M of P can be constructed in PTIME, and for any $m \geq 1$ the size of the model $M|_m$ is bounded by a polynomial in the size of P and m.*

Proof. Let P_0 be the ground instantiation of P. Just note that if the quantifier depth of P is finitely bounded then the size of P_0 is bounded by a polynomial in the size of P, and the modal depth of P_0 is equal to the modal depth of P. This corollary then follows from Theorems 4.2.6, 4.4.9, and 5.4.4. □

We call a model $\langle D, W, \tau, R, m\rangle$ a *flat model* if $W = \{\tau\}$ and $R = \emptyset$.

Theorem 5.4.6 *Let P be a MDatalog program and P_0 be its ground instantiation. If M_0 is the least flat model of P_0 then it is also the least flat model of P.*

This theorem can be proved in the same way as for Theorem 5.4.4. As a consequence, we have the following corollary for the almost serial modal logics KB, $K5$, $K45$, and $KB5$.

Corollary 5.4.7 *Let L be one of the modal logics KB, $K5$, $K45$, and $KB5$. Let P be a MDatalog program. Let M be the least LD-model of P. If P has no flat models then M is the least L-model of P. If P has a flat model then it has the least flat model M', and for every model N of P, if N is flat then $M' \leq N$, otherwise $M \leq N$.*

Moreover, if the quantifier depth of P is finitely bounded then

- *The model M', if exists, can be constructed in PTIME and its size is bounded by a polynomial in the size of P.*
- *If L is one of the logics $K5$, $K45$, and $KB5$, then the model M can be constructed in PTIME and its size is bounded by a polynomial in the size of P.*
- *If L is KB and the modal depth of P is finitely bounded then the model M can be constructed in PTIME, and for any $m \geq 1$, the size of $M|_m$ is bounded by a polynomial in the size of P and m.*

For the same reasons as for Corollary 5.4.5, this corollary follows from Theorems 4.2.6, 4.3.4, 5.4.4, 5.4.6, and Corollary 4.3.5.

Theorem 5.4.8 *Let L be one of the modal logics KD, T, KB, KDB, B, $K5$, $KD5$, $K45$, $KD45$, $KB5$, and $S5$. Given a MDatalog program P and a query $\exists \overline{x}\ \phi(\overline{x})$ such that the quantifier depths of P and the query are finitely bounded, and that the modal depth of P is finitely bounded in the case when L is not a normal extension of $K5$, then the problem of checking whether $P \vDash_L \exists \overline{x}\ \phi(\overline{x})$ and finding all tuples $\overline{c}$ of constant symbols such that $P \vDash_L \phi(\overline{c})$ is solvable in PTIME.*

Proof. If L is one of the serial modal logics then let M be the least L-model of P. If L is one of the almost but not serial modal logics then let M be the least LD-model of P, and if P has a flat model[1] then let M' be the least flat model of P. Let m be the modal depth of $\phi(\overline{x})$. Under the assumptions of the theorem, the models M and M' can be constructed in PTIME, the

[1] This can be checked in polynomial time.

size of M' is bounded by a polynomial in the size of P, and the size of $M|_m$ is bounded by a polynomial in the size of P and m.

Observe that $P \vDash_L \exists \overline{x}\ \phi(\overline{x})$ iff there exists a tuple $\overline{c}$ of constant symbols occurring in P such that $P \vDash_L \phi(\overline{c})$. The number of such tuples is bounded by a polynomial in the size of P.

If L is one of the serial modal logics or P has no flat models, then for any tuple $\overline{c}$ of constant symbols, $P \vDash_L \phi(\overline{c})$ iff $M|_m \vDash \phi(\overline{c})$. If L is one of the almost but not serial modal logics and P has a flat model, then for any tuple $\overline{c}$ of constant symbols, $P \vDash_L \phi(\overline{c})$ iff $\phi(\overline{c})$ is satisfied in both $M|_m$ and M'. Thus, for both of these cases, the problem of checking whether $P \vDash_L \phi(\overline{c})$ is solvable in polynomial time. Therefore the theorem is true. □

5.5 Discussion on Implementation

In this section we discuss how algorithms of constructing the least models for MDatalog programs can be implemented, and describe a technique that optimizes both execution time and memory for the algorithms.

5.5.1 Algorithms of Constructing the Least Models

Given a MDatalog program P, we will rather not generate the ground instantiation of P instantly, because doing so would generate a large program that could contain many irrelevant rules. We can apply traditional methods that have been used for positive classical logic programs (i.e. Datalog programs) to avoid generating instances of rules.

Recall that for Algorithms 4.2.1 and 4.4.1 that construct the least propositional Kripke models for positive propositional modal logic programs, in general we have to deal with three cases: a) the considered formula is of the form $(A \leftarrow B_1, \ldots, B_k)$, b) the considered formula is of the form $\Box\phi$, c) the considered formula is of the form $\Diamond\phi$. When trying to rewrite those algorithms for MDatalog programs, if we convert all of the rules to the form $\Box^s\ \forall\overline{x}\ (A \leftarrow B_1, \ldots, B_k)$ then the steps for the cases b) and c) need not to be changed; we only have to rewrite the step numbered 2a) in those algorithms to solve the case when the considered formula is of the form $\forall\overline{x}\ (A \leftarrow B_1, \ldots, B_k)$, with $k \geq 1$.

To keep classical atoms in memory we use new relation symbols $R_1', R_2', \ldots$ in the following way: if an atom $R_j(c_{k_1}, \ldots, c_{k_n})$ is present at a world w then we record this fact by keeping $R_j'(w, c_{k_1}, \ldots, c_{k_n})$. So, if we pay attention only to classical atoms then we have a classical relational database. Denote this database by RDB. Let RDB contain also the accessibility relation.

Suppose that it is time we want to realize some rule $\forall\overline{x}\ (A \leftarrow B_1, \ldots, B_k)$, where $k \geq 1$, at some world u. We will compute tuples $\overline{c}$ of constants

that in the place of $\overline{x}$ satisfy the set $\{B_1, \ldots, B_k\}$ at u. Denote this set by X. We then replace each atom of the form $R_j(t_{k_1}, \ldots, t_{k_n})$ in X by $R_j{}'(u, t_{k_1}, \ldots, t_{k_n})$, each atom of the form $\Diamond R_j(t_{k_1}, \ldots, t_{k_n})$ by $R(u, y) \wedge R_j{}'(y, t_{k_1}, \ldots, t_{k_n})$ (where y is a new variable), and each atom of the form $\Box R_j(t_{k_1}, \ldots .t_{k_n})$ by
$R_j{}'(v_1, t_{k_1}, \ldots, t_{k_n}) \wedge R_j{}'(v_2, t_{k_1}, \ldots, t_{k_n}) \wedge \ldots \wedge R_j{}'(v_m, t_{k_1}, \ldots, t_{k_n})$,
where $v_1, v_2, \ldots, v_m$ are all worlds with the property that $R(u, v_i), 1 \leq i \leq m$; (if we are working with the logic $KD4$ or $S4$, then the symbol R here is understood as its transitive closure R^*). Thus, the resulting set contains only classical atoms; denote it by $FO\{B_1, \ldots, B_k\}$. Denote the set of the new variables by $NewVar\{B_1, \ldots, B_k\}$; at the moment we also use $\overline{y}$ to denote this set. It is clear that a tuple $\overline{c}$ of constants in the place of $\overline{x}$ satisfies the set $\{B_1, \ldots, B_k\}$ at u iff it satisfies $\exists \overline{y}\ FO\{B_1, \ldots, B_k\}$ in the classical relational database RDB. To compute such tuples $\overline{c}$ we can use known methods that have been used for positive classical logic programs (i.e. Datalog programs). For each found tuple $\overline{c}$, we add the atom A with $\overline{x}$ substituted by $\overline{c}$ to $H(u)$.

In the following we present, for example, an algorithm that for a given MDatalog program P in a modal logic $L \in \{KD, T, KDB, B, KD5, KD45, S5\}$ will construct the least L-model of P. The algorithm is an extended version of Algorithm 4.2.1. The constructed model will be a propositional L-model over the language that takes Herbrand base as the set of primitive propositions; however, we can treat it as a first-order L-model. Data structures W, R, H, and h have the same meanings as for Algorithm 4.2.1. The procedure $CreateEmptyTail_L(x)$ remains unchanged.

For $M = \langle W, \tau, R, H\rangle$ being a model graph, let $RDB(M)$ be the set of classical atoms which is defined as follows:

$$\begin{aligned} RDB(M) = {} & \{R_j{}'(u, c_{k_1}, \ldots, c_{k_n}) \mid R_j(c_{k_1}, \ldots, c_{k_n}) \in H(u)\} \\ & \bigcup \{R(x, y) \mid x \in W \wedge y \in W \wedge R(x, y)\}. \end{aligned}$$

The following algorithm differs from Algorithm 4.2.1 only by the step 2a and the first command of the step 1.

Algorithm 5.5.1
Input: *A MDatalog program P*
in a modal logic $L \in \{KD, T, KDB, B, KD5, KD45, S5\}$
Output: *The least L-model $M = \langle W, \tau, R, h\rangle$ of P.*
Steps:

1. *Change every rule $\forall \overline{x}\ \Box^s(A \leftarrow B_1, \ldots, B_k)$ in P to the form $\Box^s\ \forall \overline{x}\ (A \leftarrow B_1, \ldots, B_k)$.*
   Set $W = \{\tau\}$, $H(\tau) = P$, $R = \emptyset$.
   $CreateEmptyTail_L(\tau)$.
   Set $R = Ext_L(R)$.

2. *For every world $u \in W$ with $H(u)$ not empty, and for every formula $\phi \in H(u)$:*

   (a) *Case $\phi = \forall \overline{x}\ (A \leftarrow B_1, \ldots, B_k)$, where $k \geq 1$:*

   - *If the vector $\overline{x}$ is empty and $M, u \vDash B_i$ for all $1 \leq i \leq k$, then set $H(u) = H(u) \bigcup \{A\}$.*
   - *If the vector $\overline{x}$ is not empty then for every tuple $\overline{c}$ of constant symbols that in the place of $\overline{x}$ satisfies $\exists \overline{y}\ FO\{B_1, \ldots, B_k\}$ in $RDB(M)$, where $\overline{y} = NewVar\{B_1, \ldots, B_k\}$, add A with $\overline{x}$ substituted by $\overline{c}$ to $H(u)$.*

   (b) *Case $\phi = \Box\psi$: For every world $v \in W$ satisfying $R(u, v)$, set $H(v) = H(v) \bigcup \{\psi\}$.*

   (c) *Case $\phi = \Diamond\psi$, and $\neg(\exists x\ R(u, x) \wedge \psi \in H(x))$:*

   Let v be a new world.
   Set $W = W \bigcup \{v\}$, $H(v) = \psi$, $R = R \bigcup \{(u, v)\}$.
   $CreateEmptyTail_L(v)$.
   Set $R = Ext_L(R)$.

3. *While some change occurred, repeat the step 2.*

5.5.2 An Optimization Technique

In this section we present a technique that optimizes both execution time and memory for the algorithms that construct the least models for MDatalog programs.

Let L be one of the serial modal logics considered in this work. Let P be a MDatalog program and M be the model graph constructed by the corresponding algorithm that constructs the least L-model for P. Since M is the least L-model of P, there exists a binary relation $r_0 \subseteq W \times W$ such that $M \leq M$ wrt. some f and r_0, and that r_0 is recursively built as described in the proof of Lemma 4.2.4 - if $L \in \{KD, T, KDB, B, KD5, KD45, S5\}$, or of Lemma 4.4.4 - if L is $KD4$ or $S4$.

One can see that for each world $u \in W$ there exists a world reachable from u, denoted by $f(u)$, such that $\forall u', v\ (u' = u \vee r_0(u, u')) \wedge R(u', v) \rightarrow r_0(f(u), v)$. (If $L \in \{KD, T, KDB, B\}$ then $f(u)$ is the first world in the chain of empty worlds created for u; if $L \in \{KD5, KD45, S5\}$ then $f(u)$ is ρ in the case $R(\tau, u)$, and ϖ in the case $\neg R(\tau, u) \wedge R^2(\tau, u)$; if L is $KD4$ or $S4$ then $f(u)$ is the v such that $R(u, v) \wedge StatusW(v) = minimal$). It is easily seen that $\forall u, v, w\ f^n(u) = w \wedge R^n(u, v) \rightarrow r_0(w, v)$, for all $n \geq 1$.

We claim that $\forall w, v\ r_0(w, v) \rightarrow H(w) \subseteq H(v)$. This can be proved by induction, however, we will not present the proof.

When we create a new world y from a world x, we call x the parent of y, and keep $Parent(y) = x$ in memory. Now observe that for all $u \in W$, if

n is the number such that $Parent^n(u) = \tau$ then

$$\bigcup_{i=1}^{n} H(f^i(Parent^i(u))) \subseteq H(u)$$

This leads to the following strategy:

- For any world $u \in W$, we will *solve* the world $f(u)$ and its successors before solving other successors of u.
- For any world $u \in W$, $u \neq \tau$, we treat $H(u)$ as the sum of itself plus the set $\bigcup_{i=1}^{n} H(f^i(Parent^i(u)))$.

This strategy saves us a lot of time and memory from recomputing and storing known facts.

5.6 Examples of Application

Examples about Belief

We can use modal logics to reason about belief. Between the axioms D, T, 4, 5 and B, the axioms D and 4 are widely accepted for belief. The axiom D : $\Box\phi \to \Diamond\phi$ says that if we believe in ϕ then we do not believe that ϕ does not hold (note that $\Diamond\phi \equiv \neg\Box\neg\phi$); this axiom can be also interpreted as if we believe in ϕ then it is possible that ϕ holds. We will then sometimes read the formula $\Diamond\phi$ as "it is possible that ϕ holds". The axiom 4 : $\Box\phi \to \Box\Box\phi$ says that if we believe in ϕ then we believe that we believe in ϕ. Thus, the logic $KD4$ is one of desirable logics for reasoning about belief. However, as we know, computational complexity of provability in $KD4$ is very high (PSPACE-complete), and modal logic programming in $KD4$ is also very complex (there is no known explicit and acceptable conditions which guarantee a MDatalog program having the least $KD4$-model with polynomial size).

The axiom 5 : $\Diamond\phi \to \Box\Diamond\phi$ can be written as $\neg\Box\psi \to \Box\neg\Box\psi$ and interpreted as if we do not believe in ψ then we believe that we do not believe in ψ. This axiom is somehow as intuitive as the axiom 4. However, one can say that this axiom is a bit strong, claiming that: if we believe in something then this information is somehow kept in our mind, on the other side, there are a lot of things we do not believe in, and generally there is no information kept in our mind about that we believe that we do not believe in that things. From the viewpoint of modal logic programming, the axiom 5 is very useful due to the fact that for L being one of the logics $KD5$, $KD45$, and $S5$, any positive propositional modal logic program has the least L-model with size bounded by a polynomial. For this reason, the modal logic $KD45$ is a very attractive logic for reasoning about belief.

In the following, we give two examples about belief.

Example 5.6.1 *Consider the MDatalog program given below, in which quantifiers are omitted. The first rule says that if X has graduated then we believe that he is employed; the second rule says that we believe that if X lives in a developed country and is employed then we believe that he has a car; the third rule says that we believe that if we believe that X is employed then it is possible that X has a car. The next five rules say that we believe in the facts we have. Given some facts, the queries are: "Who is believed to have a car ?", and "Who possibly has a car ?".*

$P = \{$
$\Box employed(X) \leftarrow graduated(X),$
$\Box[\Box has\_a\_car(X) \leftarrow employed(X) \wedge lives\_in\_a\_developed\_country(X)],$
$\Box[\Diamond has\_a\_car(X) \leftarrow \Box employed(X)],$
$\Box graduated(X) \leftarrow graduated(X),$
$\Box student(X) \leftarrow student(X),$
$\Box employed(X) \leftarrow employed(X),$
$\Box lives\_in\_a\_developed\_country(X) \leftarrow lives\_in\_a\_developed\_country(X),$
$\Box has\_a\_car(X) \leftarrow has\_a\_car(X),$
$graduated(John),\ graduated(Tomek),\ student(Kamil),$
$lives\_in\_a\_developed\_country(John),\ has\_a\_car(Jacek)$
$\}$
$?X \ \ \Box has\_a\_car(X)$
$?X \ \ \Diamond has\_a\_car(X)$

We first consider the program P in the modal logic $KD45$. The $KD45$-model graph that Algorithm 5.5.1 builds for P is specified as follows:

$W = \{\tau, \rho, \varpi, u\},\ R = W * \{\rho, \varpi, u\},$
$H(\tau) = P \bigcup \{$
$\Box employed(John), \Box employed(Tomek),$
$\Box graduated(John), \Box graduated(Tomek),\ \Box student(Kamil),$
$\Box lives\_in\_a\_developed\_country(John),\ \Box has\_a\_car(Jacek)$
$\}$
$H(\varpi) = H(\rho) = H(\tau) \bigcup \{$
$\Box has\_a\_car(X) \leftarrow employed(X) \wedge lives\_in\_a\_developed\_country(X),$
$\Diamond has\_a\_car(X) \leftarrow \Box employed(X),$
$employed(John), employed(Tomek),$
$\Box has\_a\_car(John), has\_a\_car(John),$
$\Diamond has\_a\_car(John), \Diamond has\_a\_car(Tomek)$
$\}$
$H(u) = H(\rho) \bigcup \{has\_a\_car(Tomek)\}$

This model graph founds the least $KD45$-model of P, which has the same frame as the model graph and is specified by:

$h(\tau) = \{$

$graduated(John), graduated(Tomek),\ student(Kamil),$
$lives\_in\_a\_developed\_country(John),\ has\_a\_car(Jacek)$
$\}$
$h(\rho) = h(\tau) \bigcup \{employed(John), employed(Tomek), has\_a\_car(John)\}$
$h(\varpi) = h(\rho)$
$h(u) = h(\rho) \bigcup \{has\_a\_car(Tomek)\}$

To find answers for the first query $?X\ \Box has\_a\_car(X)$, we generate ground instances of this formula and check whether they are satisfied in the constructed model. Answers for this query are 'Jacek' and 'John'. Similarly, answers for the second query $?X\ \Diamond has\_a\_car(X)$ are 'Jacek', 'John', and 'Tomek'. Note that these answers follow from the program P *in the logic KD45*.

Now consider the program P in the modal logic $KD4$. The $KD4$-model graph that the first-order version of Algorithm 4.4.1 builds for P is specified as follows:

$W = \{\tau, 1, 2, 3, 4, 5, 6\},$
$Minimal = \{w \in W \mid StatusW(w) = minimal\} = \{1, 2, 4, 6\},$
$Permanent = \{(\tau, 1), (1, 2), (1, 3), (3, 4), (1, 5), (5, 6)\},$
$R = Permanent \bigcup \{(2, 2), (4, 4), (6, 6)\} \bigcup \{2, 3, 4, 5, 6\} * \{3, 5\},$

$Next(x, \Diamond has\_a\_car(John)) = 3$, for $x \in \{1, 2, 3, 4, 5, 6\},$
$Next(x, \Diamond has\_a\_car(Tomek)) = 5$, for $x \in \{1, 2, 3, 4, 5, 6\},$

$H(\tau) = P \bigcup \{$
$\Box employed(John), \Box employed(Tomek),$
$\Box graduated(John), \Box graduated(Tomek),\ \Box student(Kamil),$
$\Box lives\_in\_a\_developed\_country(John),\ \Box has\_a\_car(Jacek)$
$\}$
$H(1) = H(\tau) \bigcup \{$
$\Box has\_a\_car(X) \leftarrow employed(X) \wedge lives\_in\_a\_developed\_country(X),$
$\Diamond has\_a\_car(X) \leftarrow \Box employed(X),$
$employed(John), employed(Tomek),$
$\Box has\_a\_car(John),$
$\Diamond has\_a\_car(John), \Diamond has\_a\_car(Tomek)$
$\}$
$H(4) = H(3) = H(2) = H(1) \bigcup \{has\_a\_car(John)\}$
$H(6) = H(5) = H(2) \bigcup \{has\_a\_car(Tomek)\}$

This model graph founds the least $KD4$-model of P, which has the same frame as the model graph and is specified by:

$h(\tau) = \{$
$graduated(John), graduated(Tomek),\ student(Kamil),$
$lives\_in\_a\_developed\_country(John),\ has\_a\_car(Jacek)$
$\}$
$h(1) = h(\tau) \bigcup \{employed(John), employed(Tomek)\}$

$h(4) = h(3) = h(2) = h(1) \bigcup \{has\_a\_car(John)\}$
$h(6) = h(5) = h(2) \bigcup \{has\_a\_car(Tomek)\}$

Using this model, the unique answer for the query $?X\ \Box has\_a\_car(X)$ is 'Jacek'; answers for the query $?X\ \Diamond has\_a\_car(X)$ still are 'Jacek', 'John', and 'Tomek'; answers for the query $?X\ \Box\Box has\_a\_car(X)$ are 'Jacek' and 'John'. Note that these answers follow from the program P *in the logic KD4*.

Example 5.6.2 *With the MDatalog given below, we want to answer the queries: "Who is believed to be good in maths ?", and "Who is possibly good in maths ?".*

$P = \{$
$\Box good\_maths(X) \leftarrow maths\_teacher(X),$
$\Box[\Box good\_maths(X) \leftarrow \Box mathematician(X)],$
$\Box[\Diamond good\_maths(X) \leftarrow maths\_student(X)],$
$\Box[\Diamond good\_physics(X) \leftarrow physics\_student(X)],$
$\Box[\Diamond good\_maths(X) \leftarrow good\_physics(X)],$
$\Box maths\_teacher(X) \leftarrow maths\_teacher(X),$
$\Box mathematician(X) \leftarrow mathematician(X),$
$\Box maths\_student(X) \leftarrow maths\_student(X),$
$\Box physics\_student(X) \leftarrow physics\_student(X),$
$maths\_teacher(John),\ mathematician(Tom),$
$maths\_student(Peter),\ physics\_student(Mike),$
$\}$
$?X\ \ \Box good\_maths(X)$
$?X\ \ \Diamond good\_maths(X)$

The $KD45$-model graph that Algorithm 5.5.1 builds for this program is specified as follows:

$W = \{\tau, \rho, \varpi, 1, 2, 3\},\ R = W * \{\rho, \varpi, 1, 2, 3\},$
$H(\tau) = P \bigcup \{$
$\Box maths\_teacher(John),\ \Box mathematician(Tom),$
$\Box maths\_student(Peter),\ \Box physics\_student(Mike),$
$\Box good\_maths(John)$
$\}$
$H(\varpi) = H(\rho) = \{$
$\Box[\Box good\_maths(X) \leftarrow \Box mathematician(X)]$
$\Box[\Diamond good\_maths(X) \leftarrow maths\_student(X)]$
$\Box[\Diamond good\_physics(X) \leftarrow physics\_student(X)]$
$\Box[\Diamond good\_maths(X) \leftarrow good\_physics(X)]$
$\Box good\_maths(X) \leftarrow \Box mathematician(X)$
$\Diamond good\_maths(X) \leftarrow maths\_student(X)$

$\Diamond good\_physics(X) \leftarrow physics\_student(X)$
$\Diamond good\_maths(X) \leftarrow good\_physics(X)$
$\Box maths\_teacher(John)$, $\Box mathematician(Tom)$,
$\Box maths\_student(Peter)$, $\Box physics\_student(Mike)$,
$\Box good\_maths(John)$,
$maths\_teacher(John)$, $mathematician(Tom)$,
$maths\_student(Peter)$, $physics\_student(Mike)$,
$good\_maths(John)$,
$\Box good\_maths(Tom), good\_maths(Tom)$,
$\Diamond good\_maths(Peter), \Diamond good\_physics(Mike)$
}
$H(1) = H(\rho) \bigcup \{good\_maths(Peter)\}$
$H(2) = H(\rho) \bigcup \{good\_physics(Mike), \Diamond good\_maths(Mike)\}$
$H(3) = H(\rho) \bigcup \{good\_maths(Mike)\}$

The least $KD45$-model of P then has the same frame as the model graph and is specified by:

$h(\tau) = \{$
$maths\_teacher(John)$, $mathematician(Tom)$,
$maths\_student(Peter)$, $physics\_student(Mike)$
}
$h(\varpi) = h(\rho) = h(\tau) \bigcup \{good\_maths(John), good\_maths(Tom)\}$
$h(1) = h(\rho) \bigcup \{good\_maths(Peter)\}$
$h(2) = h(\rho) \bigcup \{good\_physics(Mike)\}$
$h(3) = h(\rho) \bigcup \{good\_maths(Mike)\}$

Answers for the query $?X\ \Box good\_maths(X)$, in the logics $KD45$, are 'John' and 'Tom'; for the query $?X\ \Diamond good\_maths(X)$ are all 'John', 'Tom', 'Peter', 'Mike'.

Now consider the program P in the logic $KD4$. The $KD4$-model graph that the first-order version of Algorithm 4.2.1 builds for this program is specified as follows:

$W = \{\tau, 1, 2, 3, 4, 5, 6, 7, 8\}$,
$Minimal = \{w \in W \mid StatusW(w) = minimal\} = \{1, 2, 4, 6, 8\}$,
$Permanent = \{(\tau, 1), (1, 2), (1, 3), (3, 4), (1, 5), (5, 6), (5, 7), (7, 8)\}$,
$R = Permanent \bigcup \{(2, 2), (4, 4), (6, 6), (8, 8)\} \bigcup \{2, 3, 4, 5, 6, 7, 8\} * \{3, 5\}$,
$Next(x, \Diamond good\_maths(Peter)) = 3$, for $x \in \{1, 2, 3, 4, 5, 6, 7, 8\}$,
$Next(x, \Diamond good\_physics(Mike)) = 5$, for $x \in \{1, 2, 3, 4, 5, 6, 7, 8\}$,
$Next(5, \Diamond good\_maths(Mike)) = 7$,
$H(\tau) = P \bigcup \{$
$\Box maths\_teacher(John)$, $\Box mathematician(Tom)$,
$\Box maths\_student(Peter)$, $\Box physics\_student(Mike)$,
$\Box good\_maths(John)$
}

$H(1) = \{$
$\Box[\Box good\_maths(X) \leftarrow \Box mathematician(X)],$
$\Box[\Diamond good\_maths(X) \leftarrow maths\_student(X)],$
$\Box[\Diamond good\_physics(X) \leftarrow physics\_student(X)],$
$\Box[\Diamond good\_maths(X) \leftarrow good\_physics(X)],$

$\Box good\_maths(X) \leftarrow \Box mathematician(X),$
$\Diamond good\_maths(X) \leftarrow maths\_student(X),$
$\Diamond good\_physics(X) \leftarrow physics\_student(X),$
$\Diamond good\_maths(X) \leftarrow good\_physics(X),$

$\Box maths\_teacher(John), \Box mathematician(Tom),$
$\Box maths\_student(Peter), \Box physics\_student(Mike),$
$\Box good\_maths(John),$
$maths\_teacher(John), mathematician(Tom),$
$maths\_student(Peter), physics\_student(Mike),$
$good\_maths(John),$

$\Box good\_maths(Tom),$
$\Diamond good\_maths(Peter), \Diamond good\_physics(Mike)$
$\}$

$H(2) = H(1) \bigcup \{good\_maths(Tom)\}$
$H(3) = H(2) \bigcup \{good\_maths(Peter)\}$
$H(5) = H(2) \bigcup \{good\_physics(Mike), \Diamond good\_maths(Mike)\}$
$H(7) = H(2) \bigcup \{good\_maths(Mike)\}$
$H(4) = H(6) = H(8) = H(2)$

The least $KD4$-model of P then has the same frame as the model graph and is specified by:

$h(\tau) = \{$
$maths\_teacher(John), mathematician(Tom),$
$maths\_student(Peter), physics\_student(Mike)$
$\}$
$h(1) = h(\tau) \bigcup \{good\_maths(John)\}$
$h(2) = h(1) \bigcup \{good\_maths(Tom)\}$
$h(3) = h(2) \bigcup \{good\_maths(Peter)\}$
$h(5) = h(2) \bigcup \{good\_physics(Mike)\}$
$h(7) = h(2) \bigcup \{good\_maths(Mike)\}$
$h(4) = h(6) = h(8) = h(2)$

The unique answer for the query $?X\ \Box good\_maths(X)$, in the logics $KD4$, is 'John'; answers for the query $?X\ \Diamond good\_maths(X)$ are all 'John', 'Tom', 'Peter', 'Mike'; answers for the query $?X\ \Box\Box good\_maths(X)$ are 'John' and 'Tom'.

An Example about Knowledge

One crucial thing that makes knowledge different to belief is that knowledge is true while belief may be false. The axiom T : $\Box\phi \to \phi$ is thus accepted for knowledge. The axiom 4 : $\Box\phi \to \Box\Box\phi$ is also admissible.

In the case of belief, we can reason about what we believe in, but we will rather not reason about *our* knowledge, because what is true (known) to us is our knowledge. It makes sense, however, to reason about mutual knowledge; for example, we can reason about what the person A knows, and if he knows what we know, ... To reason about mutual knowledge we need to use a multiple modal logic. With a single modality, in this subsection, we will give only a simple example.

Suppose that John is a friend of Peter, whose parents are Christ and Jane. Suppose that John knows Peter's father, but there is no information about whether he knows Peter's mother or not. On the question "who does John know as Peter's parent", one would expect only 'Christ'. How can we formulate this situation ?

Suppose that in the following, the connective $\Box$ stands for "John knows". We formulate the above situation by:

$\Box[parent(X,Y) \leftarrow father(X,Y)]$,
$\Box[parent(X,Y) \leftarrow mother(X,Y)]$,
$father(Christ, Peter)$, $mother(Jane, Peter)$,
$\Box father(Christ, Peter)$
$?X \;\; \Box parent(X, Peter)$

The $S4$-model graph that the first-order version of Algorithm 4.4.1 builds for this program is specified by:

$W = \{\tau, \upsilon\}$, $R = Permanent = \{(\tau,\tau), (\tau,\upsilon), (\upsilon,\upsilon)\}$,
$Minimal = \{w \in W \mid StatusW(w) = minimal\} = \{\upsilon\}$,
$H(\tau) = P \bigcup \{$
$parent(X,Y) \leftarrow father(X,Y)$,
$parent(X,Y) \leftarrow mother(X,Y)$,
$parent(Christ, Peter)$, $parent(Jane, Peter)$
$\}$
$H(\upsilon) = \{$
$\Box[parent(X,Y) \leftarrow father(X,Y)]$,
$\Box[parent(X,Y) \leftarrow mother(X,Y)]$,
$parent(X,Y) \leftarrow father(X,Y)$,
$parent(X,Y) \leftarrow mother(X,Y)$,
$\Box father(Christ, Peter)$,
$father(Christ, Peter)$, $parent(Christ, Peter)$
$\}$

The least $S4$-model for the program then has the same frame as the model graph and is specified by:

$h(\tau) = \{$
$father(Christ, Peter), mother(Jane, Peter),$
$parent(Christ, Peter), parent(Jane, Peter)$
$\}$
$h(\upsilon) = \{father(Christ, Peter), parent(Christ, Peter)\}$

Thus, there is only one answer for the query $?X\ \Box parent(X, Peter)$ (in $S4$), and it is 'Christ'.

Chapter 6

Conclusions

This work is a study on modal reasoning. We have given improved procedures for propositional modal logics, selected and studied fragments of modal logics that are computable in polynomial time. In the following, we list our main results of this work and give information about further work.

We have presented clausal tableau systems for the common propositional modal logics K, KD, T, KB, KDB, B, $K4$, $KD4$, and $S4$. For each of these systems there is a measure such that for each tableau rule of the system the measure of all its denominators is smaller than the measure of its numerator. This property allows depth-first search without loop checking. Basing on the systems, we have given improved decision procedures for the considered logics, which use $O(n.\log n)$-space for the logics K, KD, $K4$, $KD4$, and $S4$, and $O(n^2)$-space for the logics T, KB, KDB, and B. We have also shown that the problem of checking satisfiability in T, KB, KDB, or B for formulae with finitely bounded modal depth is decidable in $O(n.\log n)$-space. We are the first who explicitly estimates space requirement for the logics KB, KDB, and B. Our space bounds for the logics $K4$, $KD4$, and $S4$ are lower than those hitherto known. Our space bounds for the logics K and KD are equal to the ones previously established by Hudelmaier [15] and Basin et al [1].

We have given algorithms that for a given positive program P in a modal logic $L \in \{KD, T, KDB, B, KD5, KD45, S5, KD4, S4\}$ will construct the least L-model of P. If $L \in \{KD5, KD45, S5\}$, or $L \in \{KD, T\}$ and the modal depth of P is finitely bounded, then the least L-model of P can be constructed in PTIME, and has size bounded by a polynomial. If $L \in \{KDB, B\}$ and the modal depth of P is finitely bounded, then the least L-model of P can be constructed in PTIME, and treated as a finite model with size bounded by a polynomial. We also show that if $L \in \{K5, K45, KB5\}$, or L is KB and the modal depth of P is finitely bounded, then any positive modal logic program P can be characterized by one or two L-models, which can be constructed in PTIME, in the sense that a positive formula follows from P iff it is satisfied in these models. As a consequence, the problem of checking satisfiability of a set of modal Horn formulae with finitely bounded modal depth in KD, T, KB, KDB, or B is decidable in PTIME. The results of other authors [7, 4] such that the problem of checking satisfiability of a set of Horn formulae in the modal logics $K5$, $KD5$, $K45$, $KD45$, $KB5$, and $S5$ is decidable in PTIME has been also studied in this work by a different method.

We have defined the query language MDatalog. A MDatalog program consists of universally quantified positive modal logic rules. The language is defined in fixed-domain first-order modal logics over signatures without functions. We have shown that if P is a MDatalog program with a finitely bounded quantifier depth, and if L is one of the modal logics $KD5$, $KD45$, $S5$ – or L is KD or T (resp. KDB or B), and the modal depth of P is finitely bounded – then the least L-model of P can be constructed in PTIME and its size is bounded by a polynomial in the size of P (resp. the least L-model of P can be treated as a finite model with size bounded by a polynomial). We have been also discussed how to implement and optimize the algorithms.

We believe that our results about modal logic programming shed new lights on the field and found an underlying ground for developing modal deductive databases. Our methods for both of the problems of estimating space requirement for modal logics and of constructing the least models for positive modal logic programs are applicable not only for the logics considered in this work.

As a further work, we intend to discover more knowledge about implementation, optimization and maintenance of modal deductive databases. We also intend to study modal logic programs with negation, to define semantics for negation in normal modal logic programs.

Acknowledgements

In the first place I would like to thank my supervisor, Professor Andrzej Szałas, for his guidance from my years of undergraduate study. He brought me to the fields of modal logics and deductive databases. He pointed out the problem for which this work has been developed. Without his constant support and encouragement this work would not have been possible.

I am also indebted to the authorities of Faculty of Mathematics, Informatics, and Mechanics, Warsaw University who accepted me to the postgraduate study and offered me a scholarship.

I would like also to thank a number of anonymous referees for helpful comments.

In addition, I would like to thank Martti Penttonen and Jörg Hudelmaier for sending me their papers.

Last but not least, many thanks to Thai Linh for her encouragement.

Bibliography

[1] D. Basin, S. Matthews, and L. Viganò. A new method for bounding the complexity of modal logics. In *G. Gottlob, A. Leitsch, and D. Mundici, editors, Proceedings of the 5th Kurt Gödel Colloquium on Computational Logic and Proof Theory (KGC'97), LNCS 1289*, pages 89–102. Springer, 1997.

[2] D. Basin, S. Matthews, and L. Viganò. *Complexity Bounds for Propositional Modal Logics.* Technical Report, Institut für Informatik, Albert-Ludwigs-Universität Freiburg, Germany. To appear. Available at http://www.informatik.uni-freiburg.de/~luca, 1999.

[3] B.F. Chellas. *Modal Logic: An Introduction.* Cambridge Univ. Press, 1980.

[4] C.C. Chen and I.P. Lin. The computational complexity of the satisfiability of modal Horn clauses for modal propositional logics. *Theoretical Computer Science*, 129:95–121, 1994.

[5] P. Doherty, W. Łukaszewicz, and A. Szałas. Computing circumscription revisited: A reduction algorithm. *Journal of Automated Reasoning*, 18(3):297–336, 1997.

[6] P. Doherty, W. Łukaszewicz, and A. Szałas. Declarative PTIME queries to relational databases using quantifier elimination. *Journal of Logic and Computation*, 9(97):1–23, 1999.

[7] L. Fariñas del Cerro and M. Penttonen. A note on the complexity of the satisfiability of modal Horn clauses. *Logic Programming*, 4:1–10, 1987.

[8] M. Fitting. *Proof Methods for Modal and Intuitionistic Logics.* volume 169 of Synthese Library. D. Reidel, Dordrecht, Holland, 1983.

[9] M. Gelfond and V. Lifschitz. The stable model semantics for logic programming. In *Proceedings of the Fifth Logic Programming Symposium*, pages 1070–1080. MIT Press, 1988.

[10] M.R. Genesereth and N.J. Nilsson. *Logical Foundations of Artificial Intelligence.* Morgan Kaufmann Publishers, 1987.

[11] R. Goré. *Tableau Methods for Modal and Temporal Logics.* Technical Report TR-ARP-15-95, Australian National University, Available via http://arp.anu.edu.au on WWW, 1995.

[12] J.Y. Halpern. The effect of bounding the number of primitive propositions and the depth of nesting on the complexity of modal logic. *Artificial Intelligence*, 75, 1995.

[13] K.J.J. Hintikka. Form and content in quantification theory. *Acta Philosophica Fennica*, 8:3–55, 1955.

[14] J. Hudelmaier. A contraction-free sequent calculus for S4. In *H. Wansing, editor, Proof Theory of Modal Logic*, pages 3–15. Kluwer, Dordrecht, 1996.

[15] J. Hudelmaier. Improved decision procedures for the modal logics K, T, S4. In *H. Kleine Büning, editor, Proceedings of CLS'95, LNCS 1092*, pages 320–334. Springer, 1996.

[16] G.E. Hughes and M.J. Cresswell. *An Introduction to Modal Logic.* Methuen, 1968.

[17] S.A. Kripke. Semantical analysis of modal logic, I. normal modal propositional calculus. *Z. Math. Logic Grundlag*, 9:67–96, 1963.

[18] R. Ladner. The computational complexity of provability in systems of modal propositional logic. *SIAM Journal of Computing*, 6:467–480, 1977.

[19] W. Łukaszewicz. *Non-monotonic Reasoning, Formalization of Commonsense Reasoning.* E.Horwood, 1990.

[20] J. McCarthy. Circumscription - a form of non-monotonic reasoning. *Journal of Artificial Intelligence*, 13:27–39, 1980.

[21] J. Minker. Logic and databases: A 20-year retrospective. In *International Workshop LID'96*, pages 3–57. Verlag Springer, 1996.

[22] G.E. Mints. Gentzen-type systems and resolution rules. In *P.Martin-Löf, G. Mints (eds.): COLOG-88, LNCS 417*, pages 198–231. Springer, 1988.

[23] L.A. Nguyen. Clausal tableau systems and space bounds for the modal logics K, KD, T, KB, KDB, and B. *Submitted to Journal of Logic and Computation*, 1999.

[24] L.A. Nguyen. Constructing the least models for positive modal logic programs. *Technical Report TR 99-04(259), Warsaw University, 1999. To appear also in Fundamenta Informaticae.*

[25] L.A. Nguyen. A new space bound for the modal logics K4, KD4 and S4. In *M. Kutylowski and L. Pacholski, editors, Proceedings of MFCS'99.* Springer, 1999.

[26] A. Nonnengart. *A Resolution-Based Calculus for Temporal Logics.* PhD thesis, Universität des Saarlandes, Saabrücken, Germany, 1995.

[27] A. Nonnengart and A. Szałas. A fixpoint approach to second-order quantifier elimination with applications to correspondence theory. In *Logic at Work. Essays Dedicated to the Memory of Helena Rasiowa, E. Orlowska (ed.)*, pages 89–108. Physica Verlag, 1998.

[28] H.J. Ohlbach. A resolution calculus for modal logics. In *Proceedings of CADE-88, LNCS310.* Springer, 1988.

[29] T.C. Przymusinski. The well-founded semantics coincides with the three-valued stable semantics. *Fundamenta Informaticae*, 13:445–464, 1990.

[30] W. Rautenberg. Modal tableau calculi and interpolation. *Journal of Philosophical Logic*, 12:403–423, 1983.

[31] K.A. Ross. *The Semantics of Deductive Databases.* PhD thesis, Standford University, 1991.

[32] A. Szałas. Temporal logic: A standard approach. In *Time and Logic. A Computational Approach*, pages 1–50. UCL Press Ltd., London, 1995.

[33] A. Van Gelder, K.A. Ross, and J.S. Schlipf. The well-founded semantics for general logic programs. *Journal of the ACM*, 38:620–650, 1991.

[34] L. Viganò. *A framework for non-classical logics.* PhD thesis, Universität des Saarlandes, Saabrücken, Germany, 1997.