Using Resolution for Testing Modal Satisfiability and Building Models

Ullrich Hustadt (u.hustadt@doc.mmu.ac.uk) and Renate A. Schmidt (r.a.schmidt@doc.mmu.ac.uk)
Centre for Agent Research and Development, Department of Computing and Mathematics, Manchester Metropolitan University, Chester Street, Manchester M1 5GD, United Kingdom.

Abstract. This paper presents a translation-based resolution decision procedure for the multi-modal logic $K_{(m)}(\cap, \cup, \smile)$ defined over families of relations closed under intersection, union and converse. The relations may satisfy certain additional frame properties. Different from previous resolution decision procedures which are based on ordering refinements our procedure is based on a selection refinement, the derivations of which correspond to derivations of tableaux or sequent proof systems. This procedure has the advantage that it can be used both as a satisfiability checker and a model builder. We show that tableaux and sequent-style proof systems can be polynomially simulated with our procedure. Furthermore, the finite model property follows for a number of extended modal logics.

Keywords: modal logic, automated theorem proving, resolution decision procedures, tableaux proof systems, satisfiability testing, model generation, simulation, relative proof complexity, relative search space complexity.

1. Introduction

Recently a number of results concerning resolution decision procedures for subclasses of first-order logic have been obtained. The considered subclasses are expressive enough to encompass a variety of non-classical logics, in particular, various extended modal logics as well as description logics. De Nivelle [11] describes a resolution decision procedure for the guarded fragment using a non-liftable ordering refinement. The guarded fragment is a generalisation of the restricted quantifier fragment corresponding to basic modal logic and allows for the embedding of a variety of extended modal logics [1]. Modal logics can also be embedded into the class $\overline{\mathrm{DK}}$ (the dual of Maslov's class K) [25], and its subclasses One-Free [14] and the class of DL-clauses [26]. Again, ordering refinements of resolution provide decision procedures for these classes. A non-standard translation into path logics combined with resolution and arbitrary refinements provide decision procedures for the satisfiability of the modal logics *K*, *KD*, *KT*, *KD4*, *S4* and *S5* [40]. This approach was adopted in the experiments of [23, 27]. Experiments using the standard translation

 To appear in the *SAT 2000* Special Issue of *Journal of Automated Reasoning*, and in a book in the *Frontiers in Artificial Intelligence and Applications* Series of IOS Press.

and a combination of a first-order theorem prover augmented with a finite-model finder are described in [37].

The motivation for this work has been our interest in different refinements for translation-based theorem proving for modal logics. Different refinements result in better performance for different problem sets. Experimental evidence for this is that for some problem sets good performance can be achieved with SPASS using ordered resolution while not so good performance with tableaux proof systems, and for some problem sets the situation is reversed, cf. [22, 24, 27]. However, by competitive testing alone it is near impossible to identify the major factors having a positive or negative influence on the performance of a theorem prover [20]. As long as the theorem provers which are being compared follow different search strategies this difference is likely to have a dominating effect on the overall performance. This has two consequences. One, we can say little about the other factors influencing the performance, for example, fundamental differences between the underlying proof systems or sophisticated redundancy elimination techniques used by the theorem prover. Two, while it is easy to find benchmark problems illustrating the superiority of one theorem prover it is just as easy to find benchmark problems showing the opposite. Therefore, it is always advisable to complement empirical investigations with theoretical study so that some insight about the different approaches is gained.

To this end we study a selection refinement of resolution, called $R^{\mathcal{MOD}}$, which adopts a very different approach to ordered refinements. It provides a decision procedure for satisfiability in the multi-modal logic $K_{(m)}(\cap, \cup, \smile)$ with modalities defined over intersection, union and converse of relations, and any normal modal logic below it, as well as a number of extensions with additional frame properties. We show that on the basis of this proof system we can polynomially simulate tableaux proof systems for the familiar lattice of standard normal modal logics below *S5*. Moreover, for the logics *K*, *KD*, *KT*, *KB*, *KDB* and *KTB*, there is a polynomial reduction of the search space of a tableaux procedure to that of our procedure for any modal formula. We focus on Massacci's single-step prefixed tableaux systems [31], but this choice is arbitrary. The technique described in this paper can also be applied for obtaining simulation results of modal tableaux systems with implicit or explicit accessibility relation and analytic modal KE tableaux [32], or even sequent proof systems.

As a side-effect the procedure described in this paper provides a solution to the problem of backward translation of resolution proofs. In general, the modal form is lost during transformation to clausal form and subsequent deduction. By using a different translation method, for example, translation methods based on the functional translation where accessibility is encoded in terms of paths [2, 19, 36], this problem can be reduced and eliminated for certain logics [9, 12]. There is another well-known technique, called structural transformation or renaming, which enables the preservation of the structure of the original formula in the clausal form. This technique is used in connection with most resolution decision procedures, but the problem is that the connection to the original formula is generally lost after a few inference steps. This makes it difficult to convert resolution proofs into more user friendly tableaux style or natural deduction style proofs. The resolution procedure described in this paper does not have this drawback. It has the property that it is always possible to translate inferred information back to modal logic. An immediate consequence of this property and the termination result is the finite model property. More important, and of practical value, is the direct utilisation of the resolution procedure for automatically generating models for satisfiable input formulae.

The structure of the paper is as follows. We begin with some preliminary definitions in the next section. Section 3 defines the syntax and semantics of the multi-modal logic $K_{(m)}(\cap, \cup, \smile)$ and extensions with first-order frame properties or axiom schemas. The resolution framework which we adopt is described in Section 4. Section 5 recalls the definition of structural transformation. The main parts are Sections 6, 7, 9 and 10. Section 6 defines the selection-based resolution procedure $R^{\mathcal{MOD}}$ and proves termination results for a certain class of extensions of logics in-between K and $K_{(m)}(\cap, \cup, \smile)$. $R^{\mathcal{MOD}}$ immediately gives a procedure for effectively finding models for satisfiable input formulae. This is shown in Section 7. In Section 8 we recall the definition of single-step prefixed tableaux calculi, for which simulation results are presented in Section 9. In Section 10 we describe an implementation of $R^{\mathcal{MOD}}$ and some initial experiments which illustrate differences to ordered resolution. The final section is the Conclusion.

2. Some basic definitions

The polarity of (occurrences of) modal or first-order subformulae is defined as usual: Any occurrence of a proper subformula of an equivalence has *zero polarity*. For occurrences of subformulae not below a '$\leftrightarrow$' symbol, an occurrence of a subformula has *positive polarity* if it is one inside the scope of an even number of (explicit or implicit) negations, and it has *negative polarity* if it is one inside the scope of an odd number of negations.

First-order variables are denoted by x, y and z, terms by s and t, atoms by $A, A_1, A_2, \ldots$, and clauses by C and D. For sets of clauses we use the symbols $N, N_1, N_2, \ldots$. A literal is an atom or the negation of an atom. The former is said to be a *positive literal* and the latter a *negative literal*. As usual *clauses* are assumed to be multisets of literals, and will be denoted by $P(x) \vee P(x) \vee \neg R(x, y)$, for example. The components in the variable partition of a clause are called *split components*, that is, split components do not share variables. Two formulae or clauses are said to be *variants* of each other if one is equal to the other modulo variable renaming.

3. The modal logic $\boldsymbol{K}_{(m)}(\cap, \cup, \smile)$

We have not come across any mention of the logic $K_{(m)}(\cap, \cup, \smile)$ in the literature of modal logic, but it is closely related to a number of known logics. It can be viewed as an extension of the multi-modal version of the basic tense logic K_t with intersection and union of relations. Unlike Boolean modal logic [16] we admit converse relations but not complementation of relations. In the field of knowledge representation, description logics closely related to $K_{(m)}(\cap, \cup, \smile)$ and moderate extensions with inclusion axioms are prominent (see for example [13]).

$K_{(m)}(\cap, \cup, \smile)$ is the multi-modal logic defined over families of binary relations closed under intersection, union and converse. Formally, the language is defined over countably many propositional variables $p, p_1, p_2, \ldots$, and countably many relational variables $r, r_1, r_2, \ldots$. A *propositional atom* is a propositional variable, $\top$ or $\bot$. A *modal formula* is either a propositional atom or a formula of the form $\neg\varphi$, $\varphi \wedge \psi$, $\varphi \vee \psi$, $\langle\alpha\rangle\varphi$ and $[\alpha]\varphi$, where φ is a modal formula and α is a relational formula. A *relational formula* is a relational variable or has one of the following forms:

$$\alpha \wedge \beta \text{ (conjunction)}, \ \alpha \vee \beta \text{ (disjunction)}, \text{ and } \alpha^{\smile} \text{ (converse)},$$

where α and β are relational formulae. Other connectives are defined to be abbreviations, for example, $\varphi \rightarrow \psi = \neg\varphi \vee \psi$. A modal formula is in *negation normal form* iff for every subformula of the form $\neg\varphi$, φ is a propositional variable.

The semantics of $K_{(m)}(\cap, \cup, \smile)$ is defined in terms of relational structures or frames. A frame is a tuple (W, R) of a non-empty set W (of worlds) and a mapping R from relational formulae to binary relations over W satisfying:

$$R(\alpha \wedge \beta) = R(\alpha) \cap R(\beta)$$

Seriality, for $D = \langle r \rangle \top$:	$R(x, f(x))$
Reflexivity, for $T = [r]p \to p$:	$R(x, x)$
Symmetry, for $B = p \to [r]\langle r \rangle p$:	$\neg R(x, y) \vee R(y, x)$
Inclusion, for $\langle r_1 \rangle p \to \langle r_2 \rangle p$:	$\neg R_1(x, y) \vee R_2(x, y)$
For $(\langle r_1 \rangle p \wedge \langle r_2{}^{\smile} \rangle p) \to \langle r_3 \rangle p$:	$\neg R_1(x, y) \vee \neg R_2(y, x) \vee R_3(x, y)$

Figure 1. Some correspondence properties

$$R(\alpha \vee \beta) = R(\alpha) \cup R(\beta)$$
$$R(\alpha^{\smile}) = R(\alpha)^{\smile}.$$

The defining class of frames of a modal logic determines, and is determined by, a corresponding class of models. A model (an interpretation) is given by a triple $\mathcal{M} = (W, R, \iota)$, where (W, R) is a frame and ι is a mapping from modal formulae to subsets of W satisfying:

$$\begin{aligned} \iota(\bot) &= \emptyset \qquad \iota(\top) = W \\ \iota(\varphi \wedge \psi) &= \iota(\varphi) \cap \iota(\psi) \\ &\vdots \quad \text{(as expected for the other propositional connectives)} \\ \iota(\langle \alpha \rangle \varphi) &= \{x \mid \exists y \in W \ \ (x, y) \in R(\alpha) \wedge y \in \iota(\varphi)\} \\ \iota([\alpha]\varphi) &= \{x \mid \forall y \in W \ \ (x, y) \in R(\alpha) \to y \in \iota(\varphi)\}. \end{aligned}$$

A modal formulae is satisfiable if an $\mathcal{M}$ exists such that for some x in W, $x \in \iota(\varphi)$.

By $L\Delta$ we denote the extension of a logic L characterised by the class of frames satisfying the conjunction of relational (frame) properties in Δ. By $L\Sigma$ we denote the extension of L closed under the additional axiom schemas in Σ. Examples of first-order frame properties (in Skolemised clausal form) and the corresponding axiom schemas are given in Figure 1.

Our results will apply to certain classes of logics $L\Delta$ with the L being logics in-between K and $K_{(m)}(\cap, \cup, {}^{\smile})$. Formally, by a logic in-between K and $K_{(m)}(\cap, \cup, {}^{\smile})$ we mean a logic $K_{(m)}(\star_1, \ldots, \star_k)$ where $m \geq 1$, $1 \leq k \leq 3$ and the $\star_i$ are distinct operations from $\{\cap, \cup, {}^{\smile}\}$.

For testing the satisfiability of a modal formula we utilise the following translation mappings.

$$\begin{aligned} \pi(\top, x) &= \top \\ \pi(\bot, x) &= \bot \\ \pi(p_i, x) &= P_i(x) \end{aligned}$$

$$\begin{aligned}
\pi(\neg\varphi, x) &= \neg\pi(\varphi, x) \\
\pi(\varphi \star \psi, x) &= \pi(\varphi, x) \star \pi(\psi, x) \quad \text{for } \star \in \{\wedge, \vee, \rightarrow, \leftrightarrow\} \\
\pi(\langle\alpha\rangle\varphi, x) &= \exists y\ (\tau(\alpha, x, y) \wedge \pi(\varphi, y)) \\
\pi([\alpha]\varphi, x) &= \forall y\ (\tau(\alpha, x, y) \rightarrow \pi(\varphi, y))
\end{aligned}$$

Relational formulae are translated according to:

$$\begin{aligned}
\tau(r_j, x, y) &= R_j(x, y) \\
\tau(\alpha^{\smile}, x, y) &= \tau(\alpha, y, x) \\
\tau(\alpha \star \beta, x, y) &= \tau(\alpha, x, y) \star \tau(\beta, x, y) \quad \text{for } \star \in \{\wedge, \vee\}
\end{aligned}$$

In the translation each propositional or relational variable (p_i or r_j) is uniquely associated with a unary or binary predicate variable, denoted by the corresponding capital letter (P_i or R_j).

By definition, Π maps any modal formula φ to $\pi(\varphi, \underline{a})$, where $\underline{a}$ denotes a constant.

THEOREM 3.1. *Let Δ be a finite (possibly empty) set of first-order frame properties, and let L be a logic in-between K and $K_{(m)}(\cap, \cup, \smile)$. For any modal formula φ, φ is satisfiable in $L\Delta$ iff $\Delta \wedge \Pi(\varphi)$ is first-order satisfiable.*

Proof. It not difficult to define a first-order interpretation from a modal interpretation, and vice versa.

It will be helpful to keep in mind the following list of equivalences which may be employed as rewrite rules (from left to right), thereby simplifying formulae and possibly transforming them into negation normal form.

$$\begin{array}{ll}
\neg\langle\alpha\rangle\psi \leftrightarrow [\alpha]\neg\psi & \neg[\alpha]\psi \leftrightarrow \langle\alpha\rangle\neg\psi \\
\langle\alpha\rangle(\psi \vee \phi) \leftrightarrow \langle\alpha\rangle\psi \vee \langle\alpha\rangle\phi & [\alpha](\psi \wedge \phi) \leftrightarrow [\alpha]\psi \wedge [\alpha]\phi \\
\langle\alpha \vee \beta\rangle\psi \leftrightarrow \langle\alpha\rangle\psi \vee \langle\beta\rangle\psi & [\alpha \vee \beta]\psi \leftrightarrow [\alpha]\psi \wedge [\beta]\psi \\
\alpha^{\smile\smile} \leftrightarrow \alpha & (\alpha \star \beta)^{\smile} \leftrightarrow \alpha^{\smile} \star \beta^{\smile} \quad \text{for } \star \in \{\wedge, \vee\}
\end{array}$$

4. First-order resolution

We briefly describe the general ordered resolution calculus R (with selection and simplification) of Bachmair and Ganzinger [5, 6, 7, 8].

In the calculus inference rules are parameterised by an admissible ordering $\succ$ on literals and a selection function S. Essentially, an admissible ordering is a total (well-founded) strict ordering on the ground level such that for literals: $\ldots \succ \neg A_n \succ A_n \succ \ldots \succ \neg A_1 \succ A_1$. This is extended to the non-ground level in a canonical manner. For the exact definition the interested reader may refer to [5, 6, 7, 8]. A *selection* function assigns to each clause a possibly empty set of occurrences of negative literals. If C is a clause, then the literal occurrences in $S(C)$ are *selected*. No restrictions are imposed on the selection function.

The calculus consists of general *expansion rules* of the form:

$$\frac{N}{N_1 \mid \cdots \mid N_n}$$

Each represents a finite derivation of alternatives $N_1, \ldots, N_n$ from N. The following rules describe how derivations can be expanded at the leaves.

Deduce: $$\frac{N}{N \cup \{\textsc{Norm}(C)\}}$$

if C is either a resolvent or a factor of clauses in N, $\textsc{Norm}(C)$ is a clause which is logically equivalent to C and $C \succeq \textsc{Norm}(C)$.

Delete: $$\frac{N \cup \{C\}}{N}$$

if C is a tautology or N contains a clause which is a variant of C.

Split: $$\frac{N \cup \{C \vee D\}}{N \cup \{C\} \mid N \cup \{D\}}$$

if C and D are variable-disjoint.

Resolvents and factors are derived by the following rules.

Resolution: $$\frac{C \vee A_1 \quad \neg A_2 \vee D}{(C \vee D)\sigma}$$

where (i) σ is the most general unifier of A_1 and A_2, (ii) no literal is selected in C, and $A_1\sigma$ is strictly $\succ$-maximal with respect to $C\sigma$, and (iii) $\neg A_2$ is either selected, or $\neg A_2\sigma$ is maximal with respect to $D\sigma$ and no literal is selected in D.

The left premise is called the *positive premise* and the right premise is called the *negative premise*. We implicitly assume that the premises have no common variables.

Factoring: $$\frac{C \vee A_1 \vee A_2}{(C \vee A_1)\sigma}$$

where (i) σ is the most general unifier of A_1 and A_2, and (ii) no literal is selected in C and $A_1\sigma$ is $\succ$-maximal with respect to $C\sigma$.

The calculus is refutationally complete and compatible with a general notion of *redundancy* for clauses and inferences, with which additional don't-care non-deterministic simplification and deletion rules can be sanctioned [5, 6, 7, 8]. Essentially, a ground clause is redundant in a set N with respect to the ordering $\succ$ if it follows from smaller instances of clauses in N, and a non-ground clause is redundant in N if all its ground instances are redundant in N. For example, any tautologous clause is redundant, and C is redundant with respect to $N \cup \textsc{Norm}(C)$. By definition, a set of clauses is *saturated up to redundancy* (with respect to ordered inference) if the conclusion of every inference from non-redundant premises in N is either contained in N, or else is redundant in N.

THEOREM 4.1. ([5, 6, 7, 8]). *Let N be a set of clauses. Then, N is unsatisfiable iff the R-saturation (up to redundancy) of N contains the empty clause.*

Commonly, in resolution decision procedures the normalisation function in the "Deduce" is required to be the condensation mapping. The *condensation* $\textsc{Cond}(C)$ of a clause C is a minimal subclause of C which is a factor of C. However, for the purposes of obtaining the decidability and simulation results below no normalisation is required.

5. Structural transformation

Structural transformation, also known as renaming or transformation to definitional form, is a standard technique for transforming formulae into a more suitable normal form, and is used in many areas, including automated deduction [4, 38, 41], modal logic [33, 34] and description logics [3]. In this paper its primary purpose is to keep a tight link between the structure of modal formulae and their translated clausal form.

Let $\mathrm{Pos}(\varphi)$ be the set of positions of a first-order formula φ. If λ is a position in φ, then $\varphi|_\lambda$ denotes the subformula of φ at position λ and $\varphi[\lambda \leftarrow \psi]$ is the result of replacing φ at position λ by ψ.

We associate with each element λ of $\Lambda \subseteq \mathrm{Pos}(\varphi)$ a predicate symbol Q_λ and a literal $Q_\lambda(x_1, \ldots, x_n)$, where $x_1, \ldots, x_n$ are the free variables of $\varphi|_\lambda$, the symbol Q_λ does not occur in φ and two symbols Q_λ and $Q_{\lambda'}$ are equal only if $\varphi|_\lambda$ and $\varphi|_{\lambda'}$ are variant formulae. Let

$$\mathrm{Def}^+_\lambda(\varphi) = \forall x_1 \ldots x_n\ (Q_\lambda(x_1, \ldots, x_n) \rightarrow \varphi|_\lambda) \quad \text{and}$$
$$\mathrm{Def}^-_\lambda(\varphi) = \forall x_1 \ldots x_n\ (\varphi|_\lambda \rightarrow Q_\lambda(x_1, \ldots, x_n)).$$

The *definition* of Q_λ is the formula

$$\mathrm{Def}_\lambda(\varphi) = \begin{cases} \mathrm{Def}^+_\lambda(\varphi) & \text{if } \varphi|_\lambda \text{ has positive polarity} \\ \mathrm{Def}^-_\lambda(\varphi) & \text{if } \varphi|_\lambda \text{ has negative polarity} \\ \mathrm{Def}^+_\lambda(\varphi) \wedge \mathrm{Def}^-_\lambda(\varphi) & \text{otherwise.} \end{cases}$$

The corresponding clauses will be called *definitional clauses*. Now, define $\mathrm{Def}_\Lambda(\varphi)$ inductively by:

$$\mathrm{Def}_\emptyset(\varphi) = \varphi \quad \text{and}$$
$$\mathrm{Def}_{\Lambda\cup\{\lambda\}}(\varphi) = \mathrm{Def}_\Lambda(\varphi[\lambda \leftarrow Q_\lambda(x_1, \ldots, x_n)]) \wedge \mathrm{Def}_\lambda(\varphi),$$

where λ is maximal in $\Lambda \cup \{\lambda\}$ with respect to the prefix ordering on positions. A *definitional form* of φ is $\mathrm{Def}_\Lambda(\varphi)$, where Λ is a subset of all positions of subformulae (usually, non-atomic or non-literal subformulae).

THEOREM 5.1. *Let φ be a first-order formula.* $\mathrm{Def}_\Lambda(\varphi)$ *can be computed in polynomial time and φ is satisfiable iff* $\mathrm{Def}_\Lambda(\varphi)$ *is satisfiable, for any $\Lambda \subseteq \mathrm{Pos}(\varphi)$.*

Consequently (by Theorems 5.1 and 3.1):

COROLLARY 5.2. *Let Δ be a finite (possibly empty) set of first-order frame properties, and let L be a logic in-between K and $K_{(m)}(\cap, \cup, \smile)$. For any given modal formula φ,* $\mathrm{Def}_\Lambda(\Pi(\varphi))$ *can be computed in polynomial time and φ is satisfiable in $L\Delta$ iff $\Delta \wedge \mathrm{Def}_\Lambda(\Pi(\varphi))$ is satisfiable, for any $\Lambda \subseteq \mathrm{Pos}(\Pi(\varphi))$.*

6. A structure preserving resolution decision procedure

In this section we define a selection-based resolution decision procedure which retains the modal structure of formulae.

As only negative literals can be selected and we aim not to destroy the connection to the original modal formulae, we transform modal formulae into negation normal form first. This is achieved by eliminating $\rightarrow$ and $\leftrightarrow$ in the usual way, moving negation symbols inward as far as possible, and deleting double negations along the way. So henceforth we will assume that the given modal formula is already in negation normal form.[1]

[1] An extra transformation, which is described in [26, Sect. 5], is necessary if we do not want to make this assumption.

1. $Q_\varphi(\underline{a})$
2. $\neg Q_\varphi(x)^+ \vee Q_\Box(x)$
3. $\neg Q_\varphi(x)^+ \vee Q_\neg(x)$
4. $\neg Q_\Box(x)^+ \vee \neg R(x,y) \vee Q_\vee(y)$
5. $\neg Q_\vee(x)^+ \vee P(x) \vee Q_\Diamond(x)$
6. $\neg Q_\Diamond(x)^+ \vee R(x,f(x))$
7. $\neg Q_\Diamond(x)^+ \vee P(f(x))$
8. $\neg Q_\neg(x)^+ \vee \neg P(x)$

Figure 2. The clausal form of $\Xi\Pi(\varphi)$ for $\varphi = \Box(p \vee \Diamond p) \wedge \neg p$.

Let $\varphi' = \Pi(\varphi)$ be the translation of any $K_{(m)}(\cap, \cup, \smile)$-formula φ. To ensure a one-to-one correspondence between clauses of the clausal form of φ' and subformulae of φ we require a particular form of structural transformation. Let Ξ denote the mapping Def_Λ where Λ is the subset of positions in φ' which correspond to non-atomic subformulae in φ. For reasons explained below Ξ introduces the same symbol for variant subformulae with the same polarity. Because by assumption φ is in negation normal form, all occurrences of non-atomic subformulae of φ' with one free variable have positive polarity. This means $\mathrm{Def}_\lambda(\varphi') = \mathrm{Def}^+_\lambda(\varphi')$ for the positions λ associated with these occurrences. But subformulae corresponding to relational formulae, that is, subformulae with two free variables, can occur both positively and negatively. Here, Ξ introduces one symbol for all variant occurrences of subformulae corresponding to non-atomic relational subformulae with positive polarity and a different symbol for all variant occurrences with negative polarity.

For example, Ξ will introduce for the subformula ψ corresponding to $\langle r_j\rangle p_i$ the definition $\forall x\ (Q_\psi(x) \rightarrow \exists y\ (R_j(x,y) \wedge P_i(y)))$. For $\psi = [\alpha]\langle\alpha\rangle p_i$ with $\alpha = r_{j_1} \wedge r_{j_2}$ and $\phi = \langle\alpha\rangle p_i$ it will introduce the definitions:

$$\begin{aligned}
&\forall x\ (Q_\psi(x) \rightarrow \forall y\ (Q^n_\alpha(x,y) \rightarrow Q_\phi(y)))\\
&\forall x\ (Q_\phi(x) \rightarrow \exists y\ (Q^p_\alpha(x,y) \wedge P_i(y)))\\
&\forall xy\ (Q^p_\alpha(x,y) \rightarrow (R_{j_1}(x,y) \wedge R_{j_2}(x,y)))\\
&\forall xy\ ((R_{j_1}(x,y) \wedge R_{j_2}(x,y)) \rightarrow Q^n_\alpha(x,y)).
\end{aligned}$$

Figure 2 gives the clausal form of a small example which will also be used for illustration later.

From a computational perspective, the renaming of formulae associated with $\alpha^{\smile}$ and also negative literals is not necessary. In this case the decidability results of this section remain true. The sole purpose of these renamings is the reconstruction of the corresponding modal formulae from derived clauses (in Section 7), and the simulation of tableaux proof systems (in Section 9).

Note how the exact correspondence between introduced symbols and subformulae is made explicit in our notation: We denote introduced predicate symbols by Q_ψ and Q_α, if Q_ψ represents an occurrence of a modal subformula ψ and Q_α represents an occurrence of a relational subformula α. We also find it convenient to use the notation

$$\begin{array}{rl} \mathcal{P}(s) & \text{for some literal in } \{P_i(s), Q_\psi(s)\}_{i,\psi}, \text{ and} \\ \mathcal{R}(s,t) & \text{for some literal in } \{R_j(s,t), R_j(t,s), Q_\alpha(s,t), Q_\alpha(t,s)\}_{j,\alpha}. \end{array}$$

Two occurrences of $\mathcal{P}(s)$, or $\mathcal{R}(s,t)$, need not be identical. For example, $\neg Q_\psi(x) \vee \mathcal{P}(x) \vee \mathcal{P}(x)$ can represent $\neg Q_\psi(x) \vee P_i(x) \vee Q_\chi(x)$. Also, $\neg Q_\psi(x) \vee \neg\mathcal{R}(x,y) \vee \mathcal{P}(y)$ can represent $\neg Q_\psi(x) \vee \neg R_j(y,x) \vee Q_\chi(y)$ or $\neg Q_\psi(x) \vee \neg Q_\alpha(x,y) \vee Q_\chi(y)$.

The inference calculus, denoted by $R^{\mathcal{MOD}}$, is given by a minimal calculus of resolution and factoring with selection by a function $S_{\mathcal{MOD}}$ (to be defined next) plus splitting, and NORM being the identity function. We assume all derivations in $R^{\mathcal{MOD}}$ are generated by strategies in which no application of the "Deduce" expansion rule with identical premises and identical consequence may occur twice on the same path in any derivation. In addition, "Delete", "Split", and "Deduce" are applied in this order, except that "Split" is not applied to clauses which contain a selected literal. An ordering refinement is optional.

We define a dependency relation $\succ_d$ on the predicate symbols by $S_1 \succ_d S_2$, if there is a definition $\psi \to \phi$ in $\Xi\Pi(\varphi)$ such that S_1 occurs in ψ and S_2 occurs in ϕ. Let $\succ_D$ be any ordering on the predicate symbols in $\Xi\Pi(\varphi)$ which is compatible with the transitive closure of $\succ_d$, that is, $\succ_d^+ \subseteq \succ_D$. Such an ordering can always be found. For this, it was important to introduce different predicate symbols for positive and negative subformulae associated with relational subformulae.

Now, for any clause C which contains an occurrence of a negative literal, the function $S_{\mathcal{MOD}}$ selects a literal $\neg A$ iff either

1. the predicate symbol of A is $\succ_D$-maximal in C, or

2. $\neg A$ is a binary literal and C does not contain a literal which would have been selected according to condition 1.

$Q_\varphi(\underline{a})$	$P_i(\underline{a})$ (when $\varphi = p_i$)
$\neg Q_\psi(x)^+ \vee \mathcal{P}(x)$	$\neg Q_\psi(x)^+ \vee \neg P_i(x)$
$\neg Q_\psi(x)^+ \vee \mathcal{P}(x) \vee \mathcal{P}(x)$	$\neg Q_\psi(x)^+ \vee \neg \mathcal{R}(x,y)\ [\vee\ \mathcal{P}(y)]$
$\neg Q_\psi(x)^+ \vee \mathcal{P}(f(x))$	$\neg Q_\psi(x)^+ \vee \mathcal{R}(x, f(x))$
$\neg Q_\alpha(x,y)^+ \vee \mathcal{R}(x,y)$	$\neg Q_\alpha(x,y)^+ \vee \mathcal{R}(x,y) \vee \mathcal{R}(x,y)$
$Q_\alpha(x,y) \vee \neg \mathcal{R}(x,y)^+$	$Q_\alpha(x,y) \vee \neg \mathcal{R}(x,y) \vee \neg \mathcal{R}(x,y)^+$

Figure 3. Schematic clausal forms for $\Xi\Pi(\varphi)$.

In the sample clause set of Figure 2 selected literals are indicated by $^+$. These are the literals which may be resolved upon, as well as any literals[2] of clauses without selected literals.

LEMMA 6.1. *Let φ be any modal formula. The clausal form of $\Xi\Pi(\varphi)$ is defined by a subset of instances of the schemas of Figure 3. (Again, selected literals are indicated by $^+$.)*

We now consider what happens during deduction with the clausal form of $\Xi\Pi(\varphi)$. Our goal is to show the existence of a term depth bound for all inferred clauses, as well as the existence of a bound on the number of variables occurring in any inferred clause. It then follows that $R^{\mathcal{MOD}}$ is a decision procedure [29].

As all non-unit clauses contain a selected literal no factoring steps are possible and all definitional clauses can only be used as negative premises of resolution steps. We note that the selected literals contain all the variables of any clause except for those in the form

$$(*) \qquad \neg Q_\psi(x)^+ \vee \neg \mathcal{R}(x,y) \vee \mathcal{P}(y).$$

With the exception of

$$\neg Q_\psi(x)^+ \vee \mathcal{P}(f(x)) \quad \text{and} \quad \neg Q_\psi(x)^+ \vee \mathcal{R}(x, f(x))$$

no variables occur as arguments of compound terms.

To begin with the only candidates for positive premises are unit ground clauses. Inference with these produce either ground clauses consisting of unary literals only, which can be split into ground unit clauses, clauses of the form $\mathcal{R}(s, f(s))$, or

$$(**) \qquad \neg \mathcal{R}(s,y)^+\ [\vee\ \mathcal{P}(y)],$$

[2] ... any strictly maximal literals ..., in case an ordering restriction is assumed.

for s a ground term. Since $\neg\mathcal{R}(s, y)$ is a binary literal and it is the only negative literal in $(**)$, it is selected either by condition 1 or condition 2 above. Now, inferences on $(**)$ with ground binary unit clauses produce ground clauses only. Inferences with the definitional clauses associated with relational subformulae produce ground clauses, which can be split into units of the form $(\neg)\mathcal{R}(s, f(s))$.

Thus, assuming φ is any modal formula and N is the clausal form of $\Xi\Pi(\varphi)$, we have:

LEMMA 6.2. *Any derivation by $R^{\mathcal{MOD}}$ from N contains only (i) ground clauses of the form $(\neg)\mathcal{P}(s)$ and $(\neg)\mathcal{R}(s, f(s))$, (ii) clauses satisfying the schematic definition of Figure 3, and (iii) clauses of the form $(**)$.*

LEMMA 6.3. *All clauses occurring in an $R^{\mathcal{MOD}}$-derivation from N contain at most two variables.*

It remains to exhibit a term depth bound. Resolvents may contain terms of larger depth than both their parent clauses. However $R^{\mathcal{MOD}}$ has the property that derived clauses are always strictly smaller than their parent clauses with respect to a well-founded ordering which is based on the ordering $\succ_D$. This implies the number of conclusions is finitely bounded, and consequently, there is an upper bound on the depth of terms. Formally:

LEMMA 6.4. *There is a finite bound on the maximal term depth of clauses derived by $R^{\mathcal{MOD}}$ from N.*

Proof. Define $\succ_d$ as above with the additional requirement that if Q_ψ is the symbol introduced for a diamond formula ψ, and Q_ϕ is the symbol introduced for a box formula ϕ, and ψ and ϕ occur at the same modal depth in φ, then $Q_\psi \succ_d Q_\phi$. Let $\succ'_D$ be any ordering on predicate symbols compatible with $\succ_d^+$. Now, with every clause C which can occur as a positive premise or as a conclusion in a derivation we associate a complexity measure c_C. We let $c_C = (Q, Q)$, if $C = (\neg)Q(s)$. If $C = (\neg)R(s,t)$ then $c_C = (Q, R)$, where Q is the predicate symbol associated with the leading function symbol of the maximal term in $\{s, t\}$. By definition, a predicate symbol Q is *associated with a function symbol* f, written Q_f, if there is a clause $\neg Q(x) \vee \mathcal{R}(x, f(x))$ in N. For example, the measure of a clause $(\neg)\mathcal{R}(s, f(s))$ is (Q_f, R). For clauses of the form $(**)$, let $c_C = (Q_\psi, Q_\phi)$ where $\neg Q_\psi(x) \vee \neg\mathcal{R}(x, y) \vee Q_\phi(y)$ is the negative premise of C and $\psi = [\alpha]\phi$. Complexity measures c_C are compared by the lexicographic combination $\succ_c = (\succ'_D, \succ'_D)$. Now, it is routine to verify that any inference step from a positive premise C

by resolution or factoring will result in a clause D such that $c_C \succ_c c_D$.[3] Termination and the existence of a term depth bound follows.

Consequently (by Lemmas 6.1 to 6.4 and Theorem 4.1):

THEOREM 6.5. *Let φ be any modal formula and let N be the clausal form of $\Xi\Pi(\varphi)$. Then,*

1. *any $R^{\mathcal{MOD}}$-derivation from N (with or without redundancy elimination) terminates, and*

2. *φ is unsatisfiable in $K_{(m)}(\cap, \cup, \smile)$ iff the $R^{\mathcal{MOD}}$-saturation (up to redundancy) of N contains the empty clause.*

Strictly, splitting is optional for this result to hold, but it is essential for the subsequent sections.

Procedures based on $R^{\mathcal{MOD}}$ can be used to decide also extensions of $K_{(m)}$ below $K_{(m)}(\cap, \cup, \smile)$ in which the R_j satisfy additional frame properties. It goes beyond the scope of this paper to investigate which extensions $K_{(m)}(\cap, \cup, \smile)\Delta$ can be decided with $R^{\mathcal{MOD}}$. However it is not difficult to verify the following result, which proves that inference with the properties of Figure 1 does not lead to unbounded computation.

In the following theorem, the notation $\mathcal{R}(s,t)$ represents either an atom $R_j(s,t)$ or an atom $R_j(t,s)$ for some j.

THEOREM 6.6. *Theorem 6.5 is true for any extension $L\Delta$ of a logic L in-between K and $K_{(m)}(\cap, \cup, \smile)$, for which the $R^{\mathcal{MOD}}$-saturation (up to redundancy) of Δ is a conjunction of universal closures of relational properties:*

1. *$R_j(x,x)$,*

2. *$\mathcal{R}(\underline{b}, \underline{c})$, where $\underline{b}$ and $\underline{c}$ denote any constants,*

3. *$\mathcal{R}(x, f(x))$, or*

4. *any clause built from literals of the forms $\mathcal{R}(x,y)$, $\mathcal{R}(x,x)$, $\mathcal{R}(y,y)$, or their negations, containing at least one negative literal $\neg\mathcal{R}(x,y)$.*

Proof. Consider inferences between Δ clauses and clauses of N containing only binary predicate symbols. The latter contain at least one selected negative literal. Thus, no inferences are possible between theory clauses satisfying condition 4 and clauses in N. Resolving clauses $Q^n_\alpha(x,y) \vee \neg\mathcal{R}(x,y)^+ \; [\vee \neg\mathcal{R}'(x,y)]$ and $\mathcal{R}(\underline{b}, \underline{c})$ produce ground clauses

[3] Notice that the ordering $\succ_c$ is merely used to show termination, not to restrict inferences.

of the same form. Non-ground conclusions are produced by resolving $\mathcal{R}(x, f(x))$, or $\mathcal{R}(x, x)$, with $Q^n_\alpha(x, y) \vee \neg\mathcal{R}(x, y)^+ \; [\vee \neg\mathcal{R}'(x, y)]$. The outcome are clauses of the form: $Q^n_\alpha(x, f(x))$, $Q^n_\alpha(x, x)$,

$$(') \quad Q^n_\alpha(x, f(x)) \vee \neg\mathcal{R}'(x, f(x))^+ \quad \text{and} \quad Q^n_\alpha(x, x) \vee \neg\mathcal{R}'(x, x)^+.$$

The superscript of Q^n_α makes explicit that it is a name linked with a negative occurrence of the formula α. Recall, that we have introduced different predicate symbols for positive and negative occurrences of the same relational formula. No inference will be possible between $Q^n_\alpha(x, f(x))$, or $Q^n_\alpha(x, x)$, and clauses satisfying condition 4. Inferences between clauses $(')$ and theory clauses are only possible if $\mathcal{R}'$ is a non-introduced primitive symbol R_j. Such inferences produce $Q^n_\alpha(x, f(x))$, $Q^n_\alpha(x, x)$ or $Q^n_\alpha(\underline{b}, \underline{b})$.

$('')$ The saturation Δ' of Δ and the subset of clauses in N which contain only binary predicate symbols is bounded.

This is immediate since inferred clauses contain at most two variables and there is no increase of the term depth.

We now establish that, except for $(**)$, conclusions of further inferences are ground. Resolution with positive premises from Δ' and clauses of the form $(**)$ produces ground conclusions. Inferences with clauses $(')$ and clauses not in Δ' are with ground clauses, and produce ground clauses. The remaining inferences are as in Lemma 6.2. It follows that split ground conclusions have the form

$$(\neg)P(s), \quad (\neg)\mathcal{R}(s, f(s)), \quad (\neg)\mathcal{R}(s, s) \quad \text{or} \quad \mathcal{R}(\underline{b}, \underline{c}).$$

Termination of any derivation from $N \cup \Delta$ is now shown as follows. Let T be a new symbol which does not occur in either N or Δ. Let this symbol be the largest symbol with respect to $\succ_d$. In addition, let

$$\Gamma(T) = \{(\neg)R(s, t) \mid s, t \in T \text{ and } R \text{ is a binary predicate symbol}\}.$$

Let N_C denote the set of clauses derived prior to C, and C itself. Now, define a measure (a subset of) clauses in a derivation by:

- $c_C = (Q, Q, \emptyset)$, if $C = (\neg)Q(s)$,
- $c_C = (Q_\psi, Q_\phi, \emptyset)$, if C has the form $(**)$, $\neg Q_\psi(x) \vee \neg\mathcal{R}(x, y) \vee Q_\phi(y)$ is the negative premise of C, and $\psi = [\alpha]\phi$, and
- $c_C = (Q, Q, \Gamma(\{s, t\}) \setminus N_C)$, if $C = (\neg)R(s, t)$ and Q is the predicate symbol associated with the the leading function symbol of the maximal term in $\{s, t\}$, whenever such a symbol exists, and T otherwise. Here, maximality is with respect to the proper subterm ordering.

9.	$R(x,x)$	
10.	$Q_{\Box}(\underline{a})$	[1, 2]
11.	$Q_{\neg}(\underline{a})$	[1, 3]
12.	$\neg R(\underline{a},y)^{+} \vee Q_{\vee}(y)$	[4, 10]
13.	$Q_{\vee}(\underline{a})$	[9, 12]
14.	$P(\underline{a}) \vee Q_{\diamond}(\underline{a})$	[5, 13]
15.	$\neg P(\underline{a})$	[11, 8]

16.1.1. $P(\underline{a})$	[14, split]	16.2.1. $Q_{\diamond}(\underline{a})$	[14, split]
16.1.2. $\emptyset$	[15, 16.1.1]	16.2.2. $R(\underline{a}, f(\underline{a}))$	[6, 16.2.1]
		16.2.3. $P(f(\underline{a}))$	[7, 16.2.1]
		16.2.4. $Q_{\vee}(f(\underline{a}))$	[12, 16.2.2]
		16.2.5. $P(f(\underline{a})) \vee Q_{\diamond}(f(\underline{a}))$	[5, 16.2.4]

16.2.5.1.1. $P(f(\underline{a}))$	[16.2.5, split]	16.2.5.1.2. $Q_{\diamond}(f(\underline{a}))$	[16.2.5, split]

Figure 4. Derivation for $\varphi = \Box(p \vee \Diamond p) \wedge \neg p$.

Let $\succ'_D$ be the ordering specified in the proof of Lemma 6.4. The ordering on the complexity measures c_C of positive premises and conclusions is defined to be the lexicographic combination of $\succ'_D$, $\succ'_D$, and the proper superset relationship. This ordering is well-founded. Now we need to verify that split ground conclusions are strictly smaller than their positive premises. As this is tedious, we omit the details. Termination follows.

A characteristic of the clausal class associated with the procedure $R^{\mathcal{MOD}}$ is that (at any point after exhaustive splitting) positive information is given by ground unit clauses, and that additionally, inferred clauses contain at most one variable. For being able to systematically construct a model in the tableaux manner, it is crucial, in particular, that any unary unit clause is ground, for otherwise, the conclusion of an inference with a clause $(*)$ or $(**)$ associated with a box formula would contain more than one variable. This is the reason for having excluded relational negation and the universal modality from consideration. In these cases also non-ground unary unit clauses would have been produced.

Finally, let us consider the example from Figure 2 and give a sample derivation, see Figure 4. We assume R is reflexive, for otherwise not many inference steps are possible. Notice how the correspondence to modal subformulae is retained during inference in $R^{\mathcal{MOD}}$. For example, 16.1.1 and 16.2.1 say that p and $\Diamond p$ are true in the initial world $\underline{a}$, 16.2.2, 16.2.3 and 16.2.4 say that p and $p \vee \Diamond p$ are true in a successor world of $\underline{a}$. The correspondence is elaborated on in the next section. Evidently, a similar derivation would have been obtained with a tableaux procedure, which will be formally established in Section 9.

7. Automatic model generation

Any clause set N_∞ saturated up to redundancy which does not include the empty clause provides a characterisation of models for the input set N. On the basis of the model construction of the completeness proof it is possible to effectively build a model from N_∞ for N. However, this model will in general be infinite. We now consider how finite models can be generated by using our selection-based procedure.

By a *model* of a clause set we mean a set I of ground atoms. The presence of an atom A in I means A is true in I, and the absence of A means $\neg A$ is true in I. In general, a clause C is true in I iff for all ground substitutions σ there is a literal L in $C\sigma$ which is true in I. Falsehood is defined dually.

LEMMA 7.1. *Let φ be a $K_{(m)}(\cap, \cup, \smile)$-formula. Let N be the clausal form of $\Xi\Pi(\varphi)$, and let N_∞ denote the $R^{\mathcal{MOD}}$-saturated (up to redundancy) clause set derivable from N. Let I be the set of positive ground unit clauses in N_∞. If N_∞ does not contain the empty clause then I is a model of N_∞ and N.*

Proof. As was noted before, during the derivation only ground unit clauses and clauses of the form $\neg\mathcal{R}(s, y) \vee \mathcal{P}(y)$ (with s ground) are generated. To prove that I is a model of N_∞ we have to show that any ground instance of a clause C in N_∞ is true in I. This obviously holds for any of the positive ground unit clauses in N_∞. Next, consider a negative ground unit clause $\neg A$ in N_∞. Then, N_∞ does not contain the positive ground unit clause A (and $A \notin I$), since otherwise the empty clause would also be in N_∞. So, $\neg A$ is true in I. Let $C\sigma = \neg A\sigma \vee D\sigma$ be any ground instance of a clause C in N_∞ with selected literal $\neg A$. If $A\sigma$ is not in I, then $C\sigma$ is true. Otherwise, $A\sigma$ is an element of I and hence of N_∞. This implies we have derived $D\sigma$ at one stage of the derivation. Consequently, one of the split components of $D\sigma$ is in N_∞. The split components of a ground clause are unit ground clauses, for which we have already shown that they are true in I. It follows that $C\sigma$ is true in I. Hence I is a model of N_∞, and also N, by completeness of the procedure (Theorem 4.1).

Now it is an easy matter to construct a modal model $\mathcal{M} = (W, R, \iota)$ for φ from I. Essentially, the set of worlds is defined by the set of ground terms occurring in I. The valuation mapping for relational formulae is determined by the set of R_i and Q_α literals in I. More precisely, if $R_i(s,t)$ (resp. $Q_\alpha(s,t)$) is in I then $(s,t) \in R(r_i)$ (resp. $(s,t) \in R(\alpha)$), which can be extended to a homomorphism for complex relational formulae. Similarly, the valuation mapping for modal formulae is defined: for any unary literal $P_i(s)$ (resp. $Q_\psi(s)$) in I, $s \in \iota(p_i)$ (resp. $s \in \iota(\psi)$), that is, p_i (resp. ψ) is true in the world s. Again, this is homomorphically extended in the expected way.

This proves:

THEOREM 7.2. *For any modal formula satisfiable in* $K_{(m)}(\cap, \cup, \smile)$ *a finite modal model can be effectively constructed on the basis of* $R^{\mathcal{MOD}}$.

On the side we remark that the satisfiability of some complex formulae is already explicit in I and N_∞. For example, if $Q_\psi(f(g(s)))$ is in $I \subseteq N_\infty$ then ψ is true in the world $f(g(s))$, $\langle\beta_g\rangle\psi$ is true in $g(s)$, and $\langle\beta_f\rangle\langle\beta_g\rangle\psi$ is true in s, where β_f and β_g are the relational formulae associated with f and g. Clauses in $N_\infty \setminus I$ give information about the satisfiability of box formulae. For example, if $\neg Q_\alpha(f(s), y) \vee Q_\psi(y)$ is contained in N_∞ then this implies $[\alpha]\psi$ is true in $f(s)$, and $\langle\beta_f\rangle[\alpha]\psi$ is true in s.

The previous result can be strengthened to the following.

THEOREM 7.3. *For any modal formula which is satisfiable in any logic* $L\Delta$ *as defined in Theorem 6.6 a finite modal model can be effectively constructed on the basis of* $R^{\mathcal{MOD}}$.

Proof. Construct a modal model $\mathcal{M}$ from the set of positive ground unit clauses in N_∞, as described above, and consistently complete $\mathcal{M}$ in accordance with the background theory Δ. Because Δ contains no clauses requiring the creation of new worlds this is always possible. For example, if $\Delta = \{R_i(x, f(x))\}$ then add $(s,s) \in R(r_i)$ for each dead end world s.

COROLLARY 7.4. *Let* L *be any logic in-between* K *and* $K_{(m)}(\cap, \cup, \smile)$, *and let* Δ *be defined as in Theorem 6.6. Then,* $L\Delta$ *has the finite model property.*

8. Tableaux systems for $K\Sigma$

We consider the single-step prefixed tableaux of Massacci [31] (with minor corrections by [18]) for the lattice of normal uni-modal logics defined over the axiom schemas K, D, T, B, 4 and 5.

The basic entities are formulae labelled with prefixes. A labelled (prefixed) formula has the form $\sigma : \varphi$, where σ is a sequence of positive integers and φ is a modal formula. We think of σ as representing a world in which φ is true.

Tableaux proofs have a tree structure. Assuming that we are interested whether a given formula φ is satisfiable, the root node contains the formula, $1 : \varphi$. Successor nodes are constructed by the application of expansion rules. The classical expansion rules for formulae in negation normal form are:

$$(\bot)\ \frac{\sigma : \psi,\ \sigma : \neg\psi}{\sigma : \bot} \qquad (\wedge)\ \frac{\sigma : \psi \wedge \phi}{\sigma : \psi,\ \sigma : \phi} \qquad (\vee)\ \frac{\sigma : \psi \vee \phi}{\sigma : \psi \mid \sigma : \phi}$$

The modal expansion rules include the diamond rule

$$(\Diamond)\ \frac{\sigma : \Diamond\psi}{\sigma.n : \psi} \text{ with } \sigma.n \text{ new to the current branch}$$

and for the logics we consider the following box rules:

$$(\Box)\ \frac{\sigma : \Box\psi}{\sigma.n : \psi} \qquad (D)\ \frac{\sigma : \Box\psi}{\sigma : \Diamond\psi} \qquad (T)\ \frac{\sigma : \Box\psi}{\sigma : \psi}$$

$$(B)\ \frac{\sigma.n : \Box\psi}{\sigma : \psi} \qquad (4)\ \frac{\sigma : \Box\psi}{\sigma.n : \Box\psi} \qquad (4^r)\ \frac{\sigma.n : \Box\psi}{\sigma : \Box\psi}$$

$$(4^d)\ \frac{\sigma.n : \Box\psi}{\sigma.n.m : \Box\psi} \qquad (5)\ \frac{1.n : \Box\psi}{1 : \Box\Box\psi}$$

The prefixes occurring in the conclusions of the box rules are required to be present in the current branch.

The tableaux calculi of the standard logics below *S5* are given by the classical rules plus the modal rules of Figure 5.[4]

THEOREM 8.1. ([31, 18]). *A formula φ is satisfiable in a logic $K\Sigma$ iff a tableaux containing a branch X can be constructed by the tableaux calculus for $K\Sigma$ such that X does not contain the falsum and no more rules can be applied.*

[4] Remember $KT = KDT$, $S4 = KT4$, $KB4 = KB5$, $S5 = KTB4 = KDB4 = KT5$.

For K:	$(\diamond), (\Box)$
For KD:	$(\diamond), (\Box), (D)$
For KT:	$(\diamond), (\Box), (T)$
For KB:	$(\diamond), (\Box), (B)$
For $K4$:	$(\diamond), (\Box), (4)$
For $K5$:	$(\diamond), (\Box), (4^r), (4^d), (5)$
For KDB:	$(\diamond), (\Box), (D), (B)$
For $KD4$:	$(\diamond), (\Box), (D), (4)$
For $KD5$:	$(\diamond), (\Box), (D), (4^r), (4^d), (5)$
For KTB:	$(\diamond), (\Box), (T), (B)$
For $S4$:	$(\diamond), (\Box), (T), (4)$
For $KB4$:	$(\diamond), (\Box), (B), (4), (4^r)$
For $K45$:	$(\diamond), (\Box), (4), (4^r), (4^d)$
For $KD45$:	$(\diamond), (\Box), (D), (4), (4^r), (4^d)$
For $S5$:	$(\diamond), (\Box), (T), (4), (4^r)$

Figure 5. Modal tableaux rules

9. Simulation of tableaux for $\boldsymbol{K}\Sigma$

In this section we prove that inference in the tableaux calculi described in the previous section can be polynomially simulated by the selection-based decision procedure of Section 6. More specifically, we consider the relative proof and search space complexity of the two systems.

In general, when concerned with the relative proof complexity of two systems $\mathcal{A}$ and $\mathcal{B}$ the task is to determine whether one proof system is able to polynomially simulate the other proof system. This is to say, there is a function g, computable in polynomial time, mapping proofs in $\mathcal{B}$ for any given formula φ to proofs in $\mathcal{A}$ for φ. Then, $\mathcal{A}$ *polynomially simulates* (*p-simulates*) the system $\mathcal{B}$ [10]. More generally, a system $\mathcal{A}$ *p-simulates derivations* (not only proofs) of a system $\mathcal{B}$ iff there is a function g, computable in polynomial time, such that for any formula φ, g maps derivations from φ in $\mathcal{B}$ to derivations from φ in $\mathcal{A}$. Clearly, if a system p-simulates derivations in another system, then it also p-simulates the other system. For determining the relative size of the search space we must determine the potential number of inference steps performed until a proof is found [39]. For this purpose we introduce the notion of p-simulation of search. A system $\mathcal{A}$ *p-simulates search* of a system $\mathcal{B}$ iff there is a polynomial function g such that for any formula φ, g maps derivations from φ in $\mathcal{A}$ to derivations from φ in $\mathcal{B}$. Then, $\mathcal{A}$ does not perform any inference steps for which no corresponding inference steps exist in $\mathcal{B}$.

In our simulation the first-order background theories for the different axiom schemas are determined by the following clauses:[5]

$$
\begin{array}{rlrl}
T_K &= \emptyset & T_{KD4} &= T_{KD} \cup T_{K4} \\
T_{KD} &= \{R(x, f(x))\} & T_{KD5} &= T_{KD} \cup T_{K5} \\
T_{KT} &= \{R(x,x)\} & T_{KTB} &= T_{KT} \cup T_{KB} \\
T_{KB} &= \{\neg R(x,y)^+ \vee R(y,x)\} & T_{S4} &= T_{KT} \cup T_{K4} \\
T_{K4} &= \{\neg R(x,y)^+ \vee \neg R(y,z) \vee R(x,z)\} & T_{KB4} &= T_{KB} \cup T_{K4} \\
T_{K5} &= \{\neg R(x,y) \vee \neg R(x,z) \vee R(y,z), & & \\
 & \quad \neg R(x,y) \vee R(y,y)\} & T_{K45} &= T_{K4} \cup T_{K5} \\
T_{KDB} &= T_{KD} \cup T_{KB} & T_{S5} &= T_{S4} \cup T_{KB}
\end{array}
$$

The previous selection function $S_{\mathcal{MOD}}$ needs to be restricted somewhat, as we do not want to select certain negative binary literals which are generated by inference steps with theory clauses. Here, we define $S_{\mathcal{TAB}}$ to be the selection function which selects

1. a unary literal $\neg A$ in a clause C, if the predicate symbol of A is $\succ_D$-maximal in C,
2. a binary literal, if it is of the form $\neg\mathcal{R}(s, y)$ and no ground term distinct from s occurs in the clause, and
3. the indicated literals in the symmetry clause and the transitivity clause.

The restriction of condition 2, that no ground term distinct from s occurs in the clause, is needed for the simulation of the rules (4) and (4^r), see below.

A subset of the rules of the general resolution calculus R, and also $R^{\mathcal{MOD}}$, are required for the simulation. Let $R^{\mathcal{TAB}}$ be the resolution calculus consisting of the rules "Split" and "Deduce" parameterised by the selection function $S_{\mathcal{TAB}}$, no ordering, and $\textsc{Norm}(C) = C$.[6] The rules are applied in this order and it is assumed that no derivation step is performed twice on one clause (or any of its variants).

The exact correspondence between the inference steps of the single-step prefixed tableaux procedure and our resolution procedure is not hard to see. In more detail, as was noted before, each non-atomic subformula or modal literal ψ of the given modal formula is uniquely associated with a predicate symbol Q_ψ in the clausal form, and each

[5] Note that we do not select any literal in the theory clauses for 5.

[6] It turns out, we do not even need factoring. This is because all positive premises are ground and subject to the application of the splitting rule.

atomic subformula (p_i or r_j) is uniquely mapped to a predicate symbol by the translation mapping. Furthermore, every prefix σ can be uniquely associated with ground term t_σ.

- The clause associated with $1 : \varphi$ is the ground unit clause $Q_\varphi(\underline{a})$ and $t_1 = \underline{a}$.
- A derivation of falsum by the $(\bot)$ rule corresponds to the derivation of the empty clause from $Q_\psi(t_\sigma)$, $Q_{\neg\psi}(t_\sigma)$ and $\neg Q_{\neg\psi}(x)^+ \vee \neg Q_\psi(x)$, the definition of $\neg\psi$.
- An application of the $(\wedge)$ rule corresponds to two resolution steps with a ground clause $Q_{\psi\wedge\phi}(t_\sigma)$ and clauses $\neg Q_{\psi\wedge\phi}(x)^+ \vee Q_\psi(x)$ and $\neg Q_{\psi\wedge\phi}(x)^+ \vee Q_\phi(x)$, producing resolvents $Q_\psi(t_\sigma)$ and $Q_\phi(t_\sigma)$.
- An application of the $(\vee)$ rule corresponds to one resolution step between clauses $Q_{\psi\vee\phi}(t_\sigma)$ and $\neg Q_{\psi\vee\phi}(x)^+ \vee Q_\psi(x) \vee Q_\phi(x)$. The conclusion is $Q_\psi(t_\sigma) \vee Q_\phi(t_\sigma)$, to which we can apply the splitting rule. The two newly created branches contain $Q_\psi(t_\sigma)$ and $Q_\phi(t_\sigma)$, respectively.
- For the simulation of the $(\Diamond)$ rule assume $Q_{\Diamond\psi}(t_\sigma)$ is present in the clauses set. Then an application of the $(\Diamond)$ rule corresponds to resolving $Q_{\Diamond\psi}(t_\sigma)$ with $\neg Q_{\Diamond\psi}(x)^+ \vee R(x, f(x))$, and $\neg Q_{\Diamond\psi}(x)^+ \vee Q_\psi(f(x))$. This will add $R(t_\sigma, f(t_\sigma))$ and $Q_\psi(f(t_\sigma))$ to the clause set. The term $f(t_\sigma)$ corresponds to the new prefix $\sigma.n$, that is, $t_{\sigma.n} = f(t_\sigma)$.

  Observe that if the set already contains $Q_\psi(t_{\sigma.n})$ no inference step is necessary. Similarly, for all other rules: Whenever the conclusion of an inference is already present in the clause set then the inference is redundant.

For the box rules we will assume that the current set of clauses includes a clause $\neg R(s, y)^+ \vee Q_\psi(y)$ of the appropriate form (s is either t_σ or $t_{\sigma.n}$). If it does not, then it can be derived from the given clause $Q_{\Box\psi}(s)$ and the definitional clause $\neg Q_{\Box\psi}(x)^+ \vee \neg R(x, y) \vee Q_\psi(y)$.

- For the $(\Box)$ rule: Resolving $\neg R(t_\sigma, y)^+ \vee Q_\psi(y)$ with $R(t_\sigma, t_{\sigma.n})$ gives $Q_\psi(t_{\sigma.n})$.
- For the (D) rule: A resolution step with $\neg R(t_\sigma, y)^+ \vee Q_\psi(y)$ and the seriality clause $R(x, f(x))$ generates $Q_\psi(f(t_\sigma))$ ($= Q_\psi(t_{\sigma.n})$ for some n already present in the branch).
- For the (T) rule: A resolution step between $\neg R(t_\sigma, y)^+ \vee Q_\psi(y)$ and the reflexivity clause $R(x, x)$ generates $Q_\psi(t_\sigma)$.

- For the (B) rule: Resolve the given clause $R(t_\sigma, t_{\sigma.n})$ with the symmetry clause, which gives $R(t_{\sigma.n}, t_\sigma)$. With $\neg R(t_{\sigma.n}, y)^+ \vee Q_\psi(y)$ we can derive $Q_\psi(t_\sigma)$.
- For the (4) rule: A resolution step between $R(t_\sigma, t_{\sigma.n})$ and the transitivity clause gives $\neg R(t_{\sigma.n}, z) \vee R(t_\sigma, z)$. This resolves with $\neg R(t_\sigma, y)^+ \vee Q_\psi(y)$ so that we get $\neg R(t_{\sigma.n}, z) \vee Q_\psi(z)$.[7]
- For the (4^r) rule, we distinguish two cases:

  a. 5 is a theorem. Inference with $R(t_\sigma, t_{\sigma.n})$ and Euclideanness results in $\neg R(t_\sigma, z) \vee R(t_{\sigma.n}, z)$, which gives $\neg R(t_\sigma, y)^+ \vee Q_\psi(y)$ when resolved with $\neg R(t_{\sigma.n}, y)^+ \vee Q_\psi(y)$.

  b. B and 4 are theorems. Resolving $R(t_\sigma, t_{\sigma.n})$ with symmetry produces $R(t_{\sigma.n}, t_\sigma)$, and with transitivity we derive $\neg R(t_\sigma, z) \vee \neg R(t_{\sigma.n}, z)$. Resolving with $\neg R(t_{\sigma.n}, y)^+ \vee Q_\psi(y)$ produces the clause $\neg R(t_\sigma, z)^+ \vee Q_\psi(z)$.
- For the (4^d) rule: We can assume 5 is a theorem and $\neg R(t_{\sigma.n}, y)^+ \vee Q_\psi(y)$ is available. Resolving this twice with Euclideanness gives

  $$\neg R(x, t_{\sigma.n})^+ \vee \neg R(x, y) \vee \neg R(y, z) \vee Q_\psi(z).$$

  Now, we resolve with $\neg R(x, y) \vee R(y, y)$, producing $\neg R(x, t_{\sigma.n})^+ \vee \neg R(t_{\sigma.n}, y)^+ \vee \neg R(y, z) \vee Q_\psi(z)$. Inference with the given clauses $R(t_\sigma, t_{\sigma.n})$ and $R(t_{\sigma.n}, t_{\sigma.n.m})$ produces $\neg R(t_{\sigma.n.m}, z)^+ \vee Q_\psi(z)$, as required.
- For the (5) rule: Two resolution steps upon $R(t_1, t_{1.n})$, Euclideanness and $\neg R(t_{1.n}, y)^+ \vee Q_\psi(y)$ generates $\neg R(t_1, z)^+ \vee Q_\psi(z)$.

All these inference steps strictly adhere to the restrictions enforced by the selection function $S_{\mathcal{TAB}}$. It turns out that each application of a tableaux rule can be simulated by at most six inference steps in $R^{\mathcal{TAB}}$.

The above proves that every tableaux derivation (or proof) can be polynomially mapped to a resolution derivation (or proof). That is:

THEOREM 9.1. *For $K\Sigma$ with $\Sigma \subseteq \{D, T, B, 4, 5\}$, derivations in the single-step prefixed tableaux calculi can be p-simulated by the selection-based resolution procedure $R^{\mathcal{TAB}}$.*

COROLLARY 9.2. *The single-step prefixed tableaux calculi for $K\Sigma$ with $\Sigma \subseteq \{D, T, B, 4, 5\}$ can be p-simulated by a selection-based resolution procedure.*

[7] It is here, for example, where we do not want to have the first literal in $\neg R(t_{\sigma.n}, z) \vee R(t_\sigma, z)$ selected.

By Theorem 6.6 termination is certain for both procedures.

COROLLARY 9.3. *Procedures based on* $R^{\mathcal{TAB}}$*, and the single-step prefixed tableaux calculi provide decision procedures for* $K\Sigma$ *with* $\Sigma \subseteq \{D, T, B\}$.

Now, we may ask whether the tableaux procedure also p-simulates search in $R^{\mathcal{TAB}}$. The answer is yes, for $K\Sigma$ with $\Sigma \subseteq \{D, T, B\}$, and modulo the addition of a factoring rule. The tableaux rules constitute in effect macro inference steps in the resolution calculus. As previously mentioned factoring is not essential for completeness with respect to the concerned clausal class. We can therefore conclude:

THEOREM 9.4. $R^{\mathcal{TAB}}$ *p-simulates search in single step prefix tableaux for* $K\Sigma$ *with* $\Sigma \subseteq \{D, T, B\}$.

In effect, by Theorem 9.1 and 9.4, there is a bisimulation relationship between the two methods. It follows that, the soundness and completeness of the tableaux procedure is a consequence of the soundness and completeness of $R^{\mathcal{TAB}}$ (Theorem 4.1).

COROLLARY 9.5. *The single-step prefix tableaux calculi for* $K\Sigma$ *with* $\Sigma \subseteq \{D, T, B\}$ *are sound and complete.*

Moreover:

COROLLARY 9.6. *Derivations based on the single-step prefixed tableaux calculi for* $K\Sigma$ *with* $\Sigma \subseteq \{D, T, B\}$ *can be simulated by a resolution procedure with the same time and space complexity.*

The above results generalise in the expected way to independent joins of countably many $K\Sigma$, where $\Sigma \subseteq \{D, T, B, 4, 5\}$.

10. Implementation and experiments

So as to obtain a first impression of the behaviour of an implementation of the selection-based resolution calculus we modified the theorem prover SPASS. SPASS is a resolution theorem prover for first-order logic with equality developed by Weidenbach et al. [43]. SPASS contains, amongst others, the inference rules of the general ordered resolution calculus R described in Section 4. Thus, to realise the calculi $R^{\mathcal{MOD}}$ and $R^{\mathcal{TAB}}$ we have extended SPASS (Version 0.95) with an implementation of the selection functions $S_{\mathcal{MOD}}$ and $S_{\mathcal{TAB}}$.

For the later discussion of the experimental results we need some basic understanding of the proof search strategy, heuristics and optimisations implemented in SPASS, which we briefly describe now.

The inference loop in SPASS is controlled by two sets of clauses: The set of worked-off clauses and the set of usable clauses. At the beginning all input clauses are usable. The theorem prover starts by choosing, according to some heuristic, a clause from the usable clause set and moving it to the worked-off clause set. Then all inference rules are applied to the chosen clause and clauses from the worked-off clause set. The derived clauses are subjected to the reduction rules, for example, subsumption deletion and clause reduction, and non-redundant clauses are added to the usable clause set. If the empty clause is derived or the set of usable clauses is empty, the theorem prover stops, having derived a contradiction or a satisfiable clause set. Otherwise, the next clause is chosen from the usable clause set, moved to the worked-off clause set, and the loop is repeated. The heuristic used by SPASS for choosing clauses is based on the number of symbols in a clause and its depth in the search space.

Among the inference rules, splitting has priority. If splitting is applicable to a clause, a heuristic is used to determine which subclauses are generated and in which order they are considered. The heuristic used by SPASS will simply branch on the first (in the order literals are stored internally) positive literal of the chosen clause.

While the heuristics implemented in SPASS work reasonably well for first-order problems, they provide little guidance on the randomly generated modal formulae we used in the comparison detailed below. Therefore, we have replaced these by new heuristics. If the set of usable clauses contains either a unit clause or a clause containing selected literals, then one of these is chosen. Otherwise, a clause is chosen which is shortest and which contains the literal L for which the function $JW(L)$ is maximal. $JW(L)$ is defined to be the sum of 2^{-n_i} over all clauses C_i containing L, where n_i is the number of literals in C_i.

For the clause sets under consideration, all non-ground clauses contain a selected literal and will therefore be chosen before any non-unit ground clause. Furthermore, all non-ground clauses are indecomposable. If the set of usable clauses contains no clauses with selected literals, then it consists of ground clauses only. On such clause sets the heuristic described above is identical to the Jeroslow-Wang heuristic for propositional problems [28]. Splitting is only applicable if the chosen clause is ground. Thus, the chosen clause contains a literal L for which the function $JW(L)$ is maximal and the splitting heuristic branches on this particular literal.

SPASS provides a variety of reduction rules. We will concentrate on subsumption deletion and unit propagation, which is an instance of clause reduction. A clause C *subsumes* a clause D if there exists a substitution σ such that $C\sigma$ is a subclause of D. For example, the clause $P_1(\underline{a}) \vee \neg P_2(\underline{a})$ subsumes $P_1(\underline{a}) \vee \neg P_2(\underline{a}) \vee P_3(\underline{a})$. A clause C *reduces* a clause D to D' if D' is a (condensed) resolvent of C and D, and D' is a subclause of D. For example, the clause $P_1(\underline{a}) \vee \neg P_2(\underline{a})$ reduces $P_1(\underline{a}) \vee P_2(\underline{a})$ to $P_1(\underline{a})$. In Figure 4, subsumption deletion would have avoided the second splitting (because clause 16.2.3 subsumes clause 16.2.5), and unit propagation (of 15 into 14) followed by subsumption deletion would have avoided also the first splitting of the derivation.

Subsumption deletion and unit propagation correspond to familiar inference rules and notions in the context of tableaux calculi. A (prefixed) disjunction Φ is *fulfilled* on a branch if there is a (prefixed) proper subdisjunction Φ' of Φ on the branch. A tableau is *regular* if the $(\vee)$ rule is not applied to a (prefixed) disjunction which is fulfilled on a branch. Thus, if $\sigma : \varphi$ and $\sigma : \varphi \vee \psi$ occur on a branch, the $(\vee)$ rule will not be applied to the second formula, since this application would produce an irregular tableaux. If $\sigma : \varphi$ and $\sigma : \neg\varphi \vee \psi$ are present on a branch, we can apply the $(\vee)$ rule to the second formula which results in a branch containing $\sigma : \neg\varphi$ and a branch containing $\sigma : \psi$. Obviously, the first branch can be closed with the $(\perp)$ rule. Considering proof length only, both subsumption deletion (regularity) and unit propagation do not improve the computational behaviour of the procedure. The shortest proof for a formula contains no application of subsumption deletion and any application of unit propagation can be simulated by applications of the $(\vee)$ and $(\perp)$ rules, as just explained.

SPASS was augmented with the translation mapping defined in Section 3, as well as the structural transformation defined in Section 6. The implementation differs from the definitions in that during the structural transformation only non-literal subformula occurrences are renamed, and for any pair of unary predicate symbols Q_ψ and Q_ϕ such that ψ is the negation normal form of $\neg\phi$, the first-order formula $\forall x\ (\neg Q_\psi(x) \vee \neg Q_\phi(x))$ is added. With the latter extension contradictions are found earlier which improves the performance. This corresponds to normalisation optimisations and techniques for early detection of contradictions in tableaux provers [21].

In the following, the modified version of SPASS using selection will be called TABSPASS. It remains a sound and complete theorem prover for first-order logic with equality.

The ordered resolution procedure with which we compare TABSPASS is a realisation in SPASS of the ordering refinement described in [26]. The ordering is a liftable ordering determined by the multiset extension of the strict subterm ordering on ground arguments. It is realised in SPASS by specifying a suitable precedence on the vocabulary for the built in extended Knuth Bendix ordering.

The experiments were performed on randomly generated formulae analysed in [23, 24]. The generated formulae are in modal conjunctive normal form determined by five parameters: N, the number of propositional variables, M, the number of modalities, K, the number of disjuncts, L, the number of conjuncts, D, the modal degree, and P, the probability of being a propositional variable. The graphs in Figures 6 and 7 are for the parameter settings

PS0 (N=5, M=1, K=3, D=2, P=0.5) and
PS1 (N=4, M=1, K=3, D=1, P=0.0).

PS0 was generated using the original generator of [17]. PS1 was generated in accordance with the guidelines of [24]. The parameter L (the number of conjuncts) ranges from N to $40N$. For each value of the ratio L/N a set of 100 random modal formulae of degree D is generated. For small L the generated formulae are most likely satisfiable and for larger L the generated formulae are most likely unsatisfiable. The tests were performed under Sun Solaris 2.6 on a cluster of PCs equipped with 300 MHz Pentium II processors and 256 MB main memory. Each figure includes three percentile graphs of TABSPASS with and without reductions, and SPASS employing the ordering refinement with reductions. Each figure also includes a graph displaying the median CPU time plots.

It is important to note that formulae generated using the parameter settings PS0 and PS1 have particular properties which have to be carefully taken into account in an analysis of the experimental results [24]. However, they are well-suited to illustrate some basic differences between various decision procedures for modal logic.

As is evident in Figures 6 and 7, enabling the reduction rules in TABSPASS greatly enhances its performance. Of particular interest here are ratios L/N greater than 18 for PS0 and greater than 17 for PS1 for which more than half of the formulae are unsatisfiable. As mentioned above, theoretical considerations would lead us to expect little difference in the performance of TABSPASS with and without reductions on these formulae. However, for PS1 there is a visible improvement for the reduction rules.

Consider a clause set containing a unit clause $Q_\varphi(t)$ and ground clauses $\bigcup_j \{\neg Q_\varphi(t) \vee C_j\}$. If $Q_\varphi(t)$ is the chosen clause in the inference loop and the reduction rules are enabled, then TABSPASS will replace each ground clause $\neg Q_\varphi(t) \vee C_j$ by C_j. To achieve the same effect if the reduction rules are disabled, we need to chose the clauses $\neg Q_\varphi(t) \vee C_j$

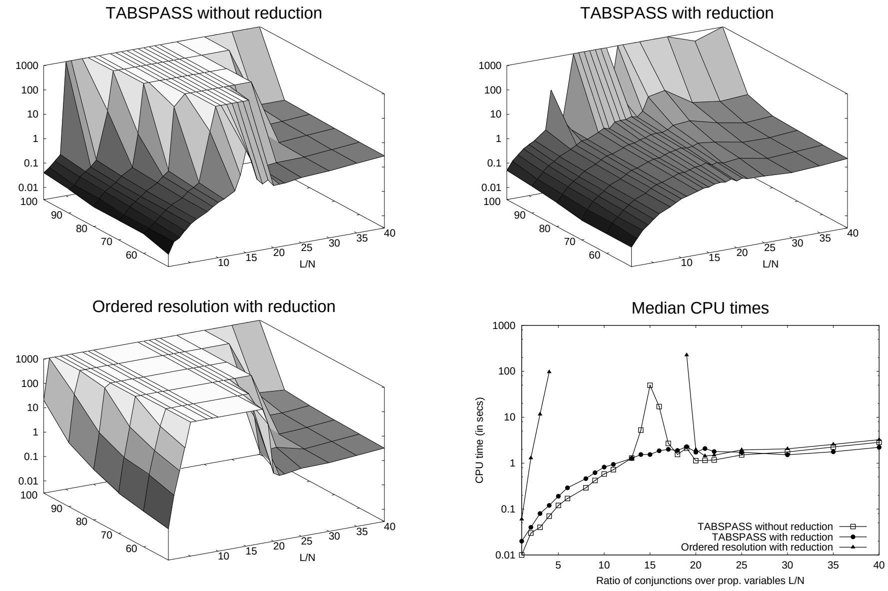

Figure 6. Graphs for PS0 (N=5, M=1, K=3, D=2, P=0.5)

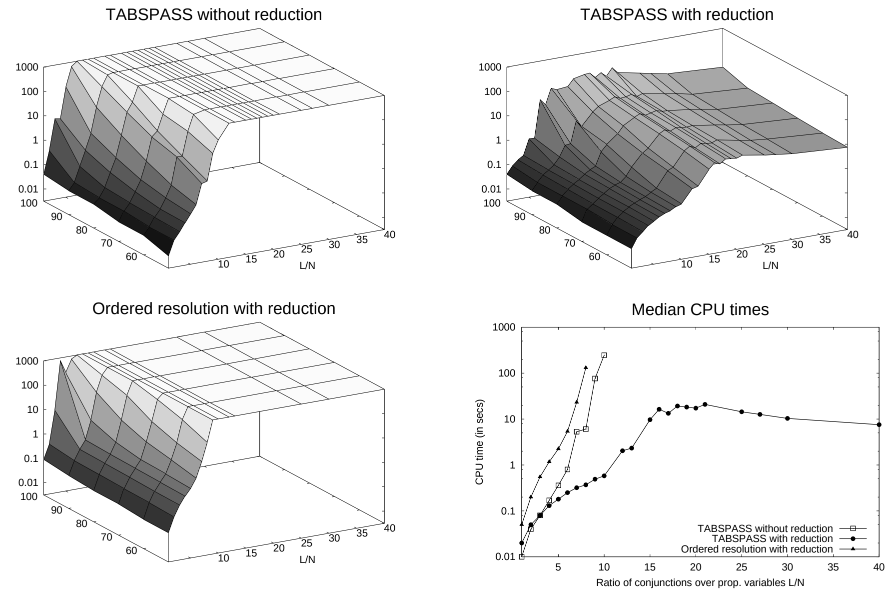

Figure 7. Graphs for PS1 (N=4, M=1, K=3, D=1, P=0.0)

one after the other, apply the splitting rule in such a way that we obtain a branch contain $\neg Q_\varphi(t)$ and one containing C_j, and close the first branch by a resolution inference step. However, the choice of clauses and the application of splitting is determined by the heuristics described above. For PS0, it happens that due to the particular probability with which literals occur in clause sets, the heuristics will actually achieve the desired effect. For PS1 this is not the case.

To understand why TABSPASS with reduction performs better than ordered resolution with reduction on PS0 and PS1 we take a closer look at the way ordered resolution deals with the clause sets under consideration. Due to the structural transformation performed, these clause sets contain a considerable number of clauses of the form

$$(\dagger) \qquad \neg Q_{\diamond\varphi}(x) \vee R(x, f(x))_* \quad \text{and}$$

$$(\dagger\dagger) \qquad \neg Q_{\Box\psi}(x) \vee \neg R(x, y)_* \vee Q_\psi(y).$$

The maximal literals are marked by a $_*$. Recall, that in the framework presented in Section 6, $\neg Q_{\diamond\varphi}(x)$ and $\neg Q_{\Box\psi}(x)$ are selected literals, which prevents any inference steps between such clauses. Here, however, whenever a clause of the form (†) is chosen, resolvents with *all* clauses of form (††) will be computed. Consequences of these inference steps have the form $\neg Q_{\diamond\varphi}(x) \vee \neg Q_{\Box\psi}(x) \vee Q_\psi(f(x))_*$. Further inference steps can lead to clauses of the form $\neg Q_{\diamond\varphi}(x) \vee \neg Q_{\Box\psi}(x) \vee (\neg)P_i(f(x))_*$. Maximal literals with a predicate symbol associated with a propositional variable potentially lead to very many inference steps.

Recall, that the satisfiability of a clause set is established only after all possible inference steps have been performed. Proof search as described above leads to a large number of derived clauses before this is the case. Thus, on all but the smallest problem this approach will not terminate within the given time limit. Even on unsatisfiable problems, the fact that we always have to compute *all* consequences with the chosen clause and the fact that the heuristics of SPASS provide very little guidance for the proof search, results in unsatisfactory performance.

The only exceptions are the clause sets obtained from the random modal formulae for the ratios L/N greater than 19 for PS0. Most of these formulae contain propositional subproblems which are unsatisfiable. The unsatisfiability of these formulae is easily detected by the procedure based on ordered resolution without any inference steps between clauses of the form (†) and (††).

It should be stressed that these experimental results are only preliminary and do not allow any strong conclusions to be drawn for or against either selection-based approaches or ordered resolution approaches. In many cases ordered resolution is easier to use, and it would be interesting to investigate whether a different organisation of the proof search could improve the performance of the ordered resolution procedure for unsatisfiable formulae. A notable difference between the ordered resolution procedure and the selection-based resolution procedure is that for transitive modal logics the former can be extended to a terminating resolution procedure which does not require a loop detection mechanism [15].

11. Conclusion

We have described a new resolution decision procedure for certain extended modal logics. Due to careful transformation to clausal form and the implementation of a tableaux-style refinement, the procedure provides also a tool for automatically building models, and additionally, proofs can be easily translated back into (tableaux proofs of) the original source logic. Logical and model-theoretic implications are decidability and the finite model property. These results are not all new, for example, Theorems 6.5 and 6.6 and Corollary 7.4 follow also from results for more expressive logics, for example [16, 35]. However, the significance of the results is that the method of obtaining them is new. Moreover, we believe the methods described here have the potential to be used for obtaining new results for less well-studied logics. For instance, one could imagine that by studying selection-based resolution refinements a tableaux proof system could be obtained for the guarded fragment.[8]

An important contribution of this paper is on the issue of relative efficiency of proof. We have focused on a particular tableaux proof system and showed how this system can be simulated with polynomial overhead in the context of resolution. As already mentioned, we expect similar results can also be obtained for other forms of tableaux proof systems or sequent calculi. The results provide new insight into the relative proof and search complexity of tableaux and resolution proof systems, similar to corresponding results for propositional logic [42]. With respect to the runtime behaviour, the difference between modal tableaux systems and selection-based resolution procedures is at most a polynomial factor. Naturally, good implementations incorporate various optimisations. A standard optimisation, in both resolution and tableaux procedures, is the utilisation of unit propagation and subsumption deletion for minimising the number of applications of the splitting rule, thereby reducing the size of the search space. It follows

[8] It so happens that during the final revision of this paper we became aware of a tableaux system for a fragment of the guarded fragment, proposed in [30]. This system can be p-simulated (and generalised) by the method described in this paper.

from our results that optimisation techniques can be readily transferred between the corresponding tableaux and resolution proof systems.

The resolution decision procedure described in this paper offers just one of many possible refinements. Other resolution refinements utilised in the literature, mentioned in the Introduction and the previous section, are implemented by ordering restrictions. Yet another alternative is hyperresolution, which is closely related to a selection-based procedure based on maximal selection [6, 8]. Consequently, the results of this paper also hold for hyperresolution. The main reason for using the Bachmair-Ganzinger framework is that it allows for all the various refinements to be presented uniformly in a common formalism.

As there is much room for further refinement, this paper may be regarded as a starting point for further in depth analysis, including an extended theoretical and empirical comparison of selection-based refinements and ordering refinements. Going on from there, a goal of further work is to establish guidelines for good search strategies for different classes of modal problems.

ACKNOWLEDGEMENTS

We thank the anonymous reviewers for their helpful comments. Also, we are grateful for the permission to perform the experiments on the PC cluster of the Max-Planck-Institut für Informatik in Saarbrücken, Germany. The work of the first author was supported by EPSRC Grant GR/K57282.

References

1. Andréka, H., J. van Benthem, and I. Németi: 1995, 'Back and Forth Between Modal Logic and Classical Logic'. *Bulletin of the IGPL* **3**(5), 685–720.
2. Auffray, Y. and P. Enjalbert: 1992, 'Modal Theorem Proving: An Equational Viewpoint'. *Journal of Logic and Computation* **2**(3), 247–297.
3. Baader, F., B. Hollunder, B. Nebel, H.-J. Profitlich, and E. Franconi: 1992, 'An Empirical Analysis of Optimization Techniques for Terminological Representation Systems, or: Making KRIS get a move on'. In: B. Nebel, C. Rich, and W. Swartout (eds.): *Proceedings of the Third International Conference on Principles of Knowledge Representation and Reasoning (KR'92).* Morgan Kaufmann, pp. 270–281.
4. Baaz, M., C. Fermüller, and A. Leitsch: 1994, 'A Non-Elementary Speed-Up in Proof Length by Structural Clause Form Transformation'. In: *Proceedings of the Ninth Annual IEEE Symposium on Logic in Computer Science (LICS'94).* IEEE Computer Society Press, pp. 213–219.
5. Bachmair, L. and H. Ganzinger: 1994, 'Rewrite-Based Equational Theorem Proving with Selection and Simplification'. *Journal of Logic and Computation* **4**(3), 217–247.

6. Bachmair, L. and H. Ganzinger: 1997, 'A Theory of Resolution'. Research Report MPI-I-97-2-005, Max-Planck-Institut für Informatik, Saarbrücken, Germany.
7. Bachmair, L. and H. Ganzinger: 1998, 'Equational Reasoning in Saturation-Based Theorem Proving'. In: *Automated Deduction—A Basis for Applications. Volume I.* Kluwer, Chapt. 11, pp. 353–397.
8. Bachmair, L. and H. Ganzinger: 2000, 'Resolution Theorem Proving'. To appear in J. A. Robinson and A. Voronkov (eds.): *Handbook of Automated Reasoning*, Elsevier.
9. Caferra, R. and S. Demri: 1993, 'Cooperation Between Direct Method and Translation Method in Non Classical Logics: Some Results in Propositional S5'. In: R. Bajcsy (ed.): *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence (IJCAI'93).* Morgan Kaufmann, pp. 74–79.
10. Cook, S. A. and R. A. Reckhow: 1979, 'The Relative Efficiency of Propositional Proof Systems'. *Journal of Symbolic Logic* **44**(1), 36–50.
11. de Nivelle, H.: 1998, 'A Resolution Decision Procedure for the Guarded Fragment'. In: C. Kirchner and H. Kirchner (eds.): *Automated Deduction: CADE-15*, Vol. 1421 of *Lecture Notes in Artificial Intelligence.* Springer, pp. 191–204.
12. Demri, S.: 1995, 'A Hierarchy of Backward Translations: Applications to Modal Logics'. In: M. de Glas and Z. Pawlak (eds.): *Proceedings of the Second World Conference on the Fundamentals of Artificial Intelligence (WOCFAI 95).* Paris: Angkor, pp. 121–132.
13. Donini, F. M., M. Lenzerini, D. Nardi, and A. Schaerf: 1996, 'Reasoning in Description Logics'. In: G. Brewka (ed.): *Principles in Knowledge Representation*, Studies in Logic, Language and Information. Stanford: CSLI Publications, pp. 191–236.
14. Fermüller, C., A. Leitsch, T. Tammet, and N. Zamov: 1993, *Resolution Method for the Decision Problem*, Vol. 679 of *Lecture Notes in Computer Science.* Springer.
15. Ganzinger, H., U. Hustadt, C. Meyer, and R. A. Schmidt: 1999, 'A Resolution-Based Decision Procedure for Extensions of K4'. To appear in *Advances in Modal Logic, Volume 2*, Stanford: CSLI Publications.
16. Gargov, G. and S. Passy: 1990, 'A Note on Boolean Modal Logic'. In: P. P. Petkov (ed.): *Mathematical Logic: Proceedings of the 1988 Heyting Summerschool.* New York: Plenum Press, pp. 299–309.
17. Giunchiglia, F. and R. Sebastiani: 1996, 'Building Decision Procedures for Modal Logics From Propositional Decision Procedures: The Case Study of Modal K'. In: M. A. McRobbie and J. K. Slaney (eds.): *Automated Deduction: CADE-13*, Vol. 1104 of *Lecture Notes in Artificial Intelligence.* Springer, pp. 583–597.
18. Goré, R.: 1995, 'Tableau Methods for Modal and Temporal Logics'. Technical Report TR-ARP-15-95, Australian National University, Canberra. To appear in M. D'Agostino, D. Gabbay, R. Hähnle and J. Posegga (eds.): *Handbook of Tableau Methods*, Kluwer.
19. Herzig, A.: 1989, 'Raisonnement automatique en logique modale et algorithmes d'unification.'. Ph.D. thesis, Univ. Paul-Sabatier, Toulouse.
20. Hooker, J. N.: 1996, 'Testing heuristics: We have it all wrong'. *Journal of Heuristics* **1**, 33–42.

21. Horrocks, I.: 1998, 'Using an Expressive Description Logic: FaCT or Fiction?'. In: A. Cohn, L. Schubert, and S. Shapiro (eds.): *Proceedings of the Sixth International Conference on Principles of Knowledge Representation and Reasoning (KR-98)*. Morgan Kaufmann, pp. 636–647.
22. Horrocks, I. and P. F. Patel-Schneider: 1998, 'FaCT and DLP'. In: H. de Swart (ed.): *Automated Reasoning with Analytic Tableaux and Related Methods: TABLEAUX'98*, Vol. 1397 of *Lecture Notes in Computer Science*. Springer, pp. 27–30.
23. Hustadt, U. and R. A. Schmidt: 1997, 'On Evaluating Decision Procedures for Modal Logics'. In: M. Pollack (ed.): *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence (IJCAI'97)*. Morgan Kaufmann, pp. 202–207.
24. Hustadt, U. and R. A. Schmidt: 1999a, 'An Empirical Analysis of Modal Theorem Provers'. *Journal of Applied Non-Classical Logics* **9**(4).
25. Hustadt, U. and R. A. Schmidt: 1999b, 'Maslov's Class K Revisited'. In: H. Ganzinger (ed.): *Automated Deduction—CADE-16*, Vol. 1632 of *Lecture Notes in Artificial Intelligence*. Springer, pp. 172–186.
26. Hustadt, U. and R. A. Schmidt: 2000, 'Issues of Decidability for Description Logics in the Framework of Resolution'. In: R. Caferra and G. Salzer (eds.): *Automated Deduction in Classical and Non-Classical Logics*, Vol. 1761 of *Lecture Notes in Artificial Intelligence*. pp. 192–206.
27. Hustadt, U., R. A. Schmidt, and C. Weidenbach: 1998, 'Optimised Functional Translation and Resolution'. In: H. de Swart (ed.): *Automated Reasoning with Analytic Tableaux and Related Methods: TABLEAUX'98*, Vol. 1397 of *Lecture Notes in Computer Science*. Springer, pp. 36–37.
28. Jeroslow, R. and J. Wang: 1990, 'Solving Propositional Satisfiability Problems'. *Annals of Mathematics and Artificial Intelligence* **1**, 167–187.
29. Joyner Jr., W. H.: 1976, 'Resolution Strategies as Decision Procedures'. *Journal of the ACM* **23**(3), 398–417.
30. Lutz, C., U. Sattler, and S. Tobies: 1999, 'A Suggestion of an n-ary Description Logic'. In: P. Lambrix, A. Borgida, M. Lenzerini, R. Möller, and P. Patel-Schneider (eds.): *Proc. of Intern. Workshop on Description Logics'99*. Linköping University, pp. 81–85.
31. Massacci, F.: 1994, 'Strongly Analytic Tableaux for Normal Modal Logics'. In: *Automated Deduction: CADE-12*, Vol. 814 of *Lecture Notes in Artificial Intelligence*. Springer, pp. 723–737.
32. Massacci, F.: 1998, 'Simplification: A General Constraint Propagation Technique for Propositional and Modal Tableaux'. In: H. de Swart (ed.): *Automated Reasoning with Analytic Tableaux and Related Methods: TABLEAUX'98*, Vol. 1397 of *Lecture Notes in Artificial Intelligence*. Springer, pp. 217–231.
33. Mints, G.: 1986, 'A Resolution Method for Non-Classical Logics'. *Semiotika and Informatika* **25**, 120–135.
34. Mints, G.: 1989, 'Resolution Calculi for Modal Logics'. *American Mathematical Society Translations* **143**, 1–14.
35. Mortimer, M.: 1975, 'On Languages with two Variables'. *Zeitschrift für mathematische Logik und Grundlagen der Mathematik* **21**, 135–140.
36. Ohlbach, H. J. and R. A. Schmidt: 1997, 'Functional Translation and Second-Order Frame Properties of Modal Logics'. *Journal of Logic and Computation* **7**(5), 581–603.

37. Paramasivam, M. and D. A. Plaisted: 1998, 'Automated Deduction Techniques for Classification in Description Logic Systems'. *Journal of Automated Reasoning* **20**(3), 337–364.
38. Plaisted, D. A. and S. Greenbaum: 1986, 'A Structure-Preserving Clause Form Translation'. *Journal of Symbolic Computation* **2**, 293–304.
39. Plaisted, D. A. and Y. Zhu: 1997, *The Efficiency of Theorem Proving Strategies*. Vieweg.
40. Schmidt, R. A.: 1999, 'Decidability by Resolution for Propositional Modal Logics'. *Journal of Automated Reasoning* **22**(4), 379–396.
41. Tseitin, G. S.: 1970, 'On the complexity of derivations in propositional calculus'. In: A. O. Slisenko (ed.): *Studies in Constructive Mathematics and Mathematical Logic, Part II*. New York: Consultants Bureau:, pp. 115–125.
42. Urquhart, A.: 1995, 'The Complexity of Propositional Proofs'. *Bulletin of Symbolic Logic* **1**(4), 425–467.
43. Weidenbach, C. et al.: 1999, 'System Description: SPASS Version 1.0.0'. In: H. Ganzinger (ed.): *Automated Deduction—CADE-16*, Vol. 1632 of *Lecture Notes in Artificial Intelligence*. pp. 378–382.