Journal of Automated Reasoning **24:** 319–364, 2000.
© 2000 *Kluwer Academic Publishers. Printed in the Netherlands.*

Single Step Tableaux for Modal Logics

Computational Properties, Complexity and Methodology

FABIO MASSACCI
Dipartimento di Informatica e Sistemistica, Università di Roma "La Sapienza", via Salaria 113, 00198 Roma, Italy, e-mail: massacci@dis.uniroma1.it

(Received: 24 april 1996; accepted: 9 November 1998)

Abstract. Single Step Tableaux (SST) are the basis of a calculus for modal logics that combines different features of sequent and prefixed tableaux into a simple, modular, strongly analytic, and effective calculus for a wide range of modal logics.

The paper presents a number of the computational results about SST (confluence, decidability, space complexity, modularity, etc.) and compares SST with other formalisms such as translation methods, modal resolution, and Gentzen-type tableaux. For instance, it discusses the feasibility and infeasibility of deriving decision procedures for SST and translation-based methods by replacing loop checking techniques with simpler termination checks.

The complexity of searching for validity and logical consequence with SST and other methods is discussed. Minimal conditions on SST search strategies are proven to yield PSPACE (and NPTIME for S5 and KD45) decision procedures. The paper also presents the methodology underlying the construction of the correctness and completeness proofs.

Key words: modal logics, prefixed tableaux, confluence, complexity, decision procedures, direct and translation methods.

1. Introduction

Modal logics are widely used in AI and computer science. Their applications range from modeling knowledge and belief [10, 22] or distributed systems [21] to non-monotonic formalisms [31]. Thus, a major objective is the design of *effective and simple-to-use* decision procedures.

At one side of the spectrum, *direct deduction* methods use modal formulae and enhance classical deduction mechanism with rules for modal connectives. For instance, Hilbert axiomatizations can be found in [22, 25], Fitch- or Gentzen-type calculi and tableaux in [3, 13, 14, 20, 35], and modal resolution in [9, 35]. Direct systems are closer to the epistemic properties (omniscience, introspection, etc.) a user would like to model. However, they have computational limitations. For instance, Hilbert systems are not analytic and thus cannot be thoroughly automated. Gentzen-type calculi and tableaux [13, 20, 41] are usually analytic. Nevertheless, the reduction of modal connectives requires processing sets of formulae at once (or

whole tableau branches). The absence of a confluence theorem* makes deduction heavily dependent on the search methods [8]. Moreover, logical consequence for symmetric and Euclidean logics, which play a major role in nonmonotonic reasoning [31], is still unsatisfactory. Gentzen calculi treat these logics in an entirely nondeterministic way, since cut has been eliminated only for K45 by Schwartz [45]. For S5 the most effective way is still the explicit construction of Kripke models [25, 22] or the (possibly exponential) reduction to a suitable normal form followed by a specialized calculus [25, 9, 41]. With symmetric logics, the elimination of cut is still an open problem for logical consequence [13, 20].

On the other side, *indirect deduction* methods translate modal formulae into first-order logic, "add" the properties of the underlying Kripke model, and apply a (suitably modified) classical deduction technique [39]. The same process can be carried out with a simplified set theory as the target logic [6]. Some translations, for example, the relational translation plus theory reasoning [18, 42], reflect the underlying semantics of Kripke models but are often inefficient. The use of set theory [6] may also prove to be hard for automated reasoning. Functional translation performs much better and can be easily applied either to matrix proof methods [47] or resolution [38, 39]. This approach requires either a strong equality-handling mechanism or the compilation of logic dependent unification procedures in the prover to cope with the terms corresponding to possible worlds [39]. To overcome this limitation, a mixed approach has been proposed in [36], which captures the most common logics with simple Horn clauses and maintains the original structure of the formula (at least for deontic logics). A common limitation of these proposals is that they seem to need a "user-friendly" interface for proof presentation (see [47] for further references). As we shall see, they also have computational drawbacks, although the presence of effective resolution implementations does not make this immediate.

In the middle of the field we find *labeled deductive systems* [16], such as prefixed tableaux [13, Chapter 8] or [32, 7, 20]. In prefixed tableaux, formulae are labeled with a prefix, to "name" the world where each formula is supposed to hold and rules take into account both formulae and labels. Proofs and proof search presentations are simple, since they closely reflect the Kripke semantics. Yet, we need to understand and use an algebraic or a relational theory of prefixes.

Single Step Tableaux [32], SST for short, lie between Gentzen-type and prefixed tableaux and try to combine user-friendly presentation with effective computational properties. The intuition is to label modal formulae with prefixes, use rules for labels and formulae but without any equational theory. We follow an intuition of Kripke [29, §4]: structure prefixes as a tree, label each node of the tree with a formula, encode the properties of Fitch-style tableaux into rules for single formulae,

* Loosely speaking, confluence means that we can always "converge" to a proof without backtracking. In confluent systems, a systematic strategy for proof search may be computationally bad but not incomplete.

and use rules involving only the immediate predecessor or the successors of each node. The resulting calculus has a number of useful features:

- it is cumulative as Hilbert-style axiomatizations* and can be used in a modular way for a wide range of modal logics;
- it has good computational characteristics;
- can be used to derive results about other systems;
- it can be easily implemented using free variables tableaux [1];
- it can still be used for doing proofs by hand.

1.1. RESULTS PRESENTED IN THIS PAPER

In this paper we focus mainly on the *methodology* underlying the construction of the calculus and its *computational properties* in comparison with other formalisms.

From the methodological perspective, we show how SST can be viewed as a smart combination of some aspects of prefixed tableau calculi à la Fitting [13, 42] and modal Gentzen-type calculi [13, 14, 20, 40]. We show how the algebraic properties of the underlying Kripke model need to be reconstructed only in the completeness proof.

Computational properties of different calculi can be compared in two ways: by extensive experiments on large sets of (possibly random) instances or by theoretical analysis. Both techniques are necessary because experiments may test only the relative efficiency of implementations or be misled by flawed test suites,** whereas theoretical results may not tell us enough about practical cases.

Among computational properties, we focus on complexity theoretic properties such as space requirements, confluence, and decidability.

For the construction of decision procedures, we show how *loop checking for transitive logics can be replaced by a simple and effective termination check* in SST and how this result can be generalized to other systems.‡ We focus on translation methods and give the conditions for using resolution as a decision procedure with functional and mixed translation methods. We also show that any polynomial local check is impossible for logical consequence in nontransitive logics. A general remark can be made w.r.t. the complexity of satisfiability (validity) checking for the logics K45, KD45, and S5 [22]: this problem is known to be NPTIME(CO-NPTIME)-complete but the corresponding proof systems are often based on extension of the logics K or S4 by adding further rules [13, 20, 22, 45], by strengthening the equational constraints [38], or by embedding S5 in S4 [13, 40].

* Note that there is not a 1-1 correspondence between rules and axioms. See Section 6.

** For instance, see how the claims in [19] have been rebutted in [26].

‡ Similar results have been shown independently by [5, 23].

In other cases [25, 41, 9] a potentially* exponential preprocessing step is necessary to avoid cut and prove termination. Simpler in theory, they are harder in practice.

We show two natural SST extensions of K4 for the logics K45, KD45, and S5 that can be used to achieve the complexity lower bound for satisfiability checking [30]. For the other logics, we discuss the design of PSPACE search strategies for validity-checking algorithms.

1.2. PLAN OF THE PAPER

First, we discuss the main computational problems in the design of modal calculi (Section 2). Then, we introduce some preliminaries on modal logics (Section 3) and the proof theory of SST (Section 4) and discuss the intuitions behind the construction of SST from Gentzen-type and prefixed tableaux (Section 5).

Next, we present some computational results about SST and compare SST with translation methods, modal resolution, and Gentzen-type tableaux. We focus on modularity (Section 6), proof confluence (Section 7), decision procedures for SST and translation methods (Section 8), computational complexity, and search strategies (Section 9). Last, we prove SST sound and complete (Section 10) and conclude (Section 11).

The first appendix summarizes some notions about complexity classes, and the second appendix contains the details of the longest proofs.

2. Computational Problems and Objectives

The design of a flexible calculus (with a related theorem prover) for many modal logics faces a number of problems in the attempt to obtain something that is simple, effective, and efficient.

Beside soundness and completeness, other computational properties are important but receive less attention than deserved. We postpone the issue of correctness until Section 10 to discuss these properties first.

With many modal logics, the first objective may be *modularity and cumulativity*, in the same fashion of Hilbert-style axiomatizations:

PROBLEM 1. Can we switch from logic to logic by simply adding or deleting axioms or inference rules in a modular way?

We also want a general methodology for completeness and soundness proofs so that we do not have to make them from scratch for each logic.

Once a logic L is fixed, and soundness and completeness are proved, we must search for a proof, and a calculus may have many rules to choose from. We expect the search strategy to affect time and space complexity, but we would like to avoid more substantial losses.

* Unless one uses the nontrivial definitional translation into CNF by Mints [35].

PROBLEM 2. Does the order in which we select rules or formulae matter for completeness?

The property we are looking for is *proof confluence* (see Section 7 for formal definitions). Loosely speaking, confluence means that the order in which we select the rules does not substantially matter: we can always "converge" to the same result *without backtracking*. We can waste resources, but we cannot lose the possibility of finding a proof. Proof confluence is one of the advantages of resolution in first-order theorem proving.

The propositional modal logics considered in this paper are decidable, and hence our next problem is the following.

PROBLEM 3. Can we devise techniques for blocking the selection of rules and formulae, thus yielding a decision procedure?

Proving that suitable techniques give a *decision method* is the first step (Section 8), but it is not enough. Termination checks can be expensive and may require to handle the whole proof constructed so far [8].

PROBLEM 4. Can we devise termination checks that consider each formula (or each rule) in isolation and require only polynomial time?

Here we ask for a *local termination check*. For validity, we have a positive result: such checks are possible for SST and can be transferred directly to functional and mixed translation methods (Section 8). Unfortunately, this is impossible for logical consequence (Section 9).

The last problem is the *computational complexity* of various search strategies and algorithms for validity and logical consequence (Section 9). Validity checking is "only" CO-NPTIME-complete for the logics K45 and S5, whereas it is PSPACE-complete for the modal logics between K and S4 [30]. Logical consequence is EXPTIME-complete [22]. These facts should be reflected into efficient strategies for theorem proving.

PROBLEM 5. Can we design restrictions on the proof search in the calculus so that the algorithms for checking validity will match the corresponding (optimal) worst-case complexity?

Some calculi may fail to match the lower bounds, and others can be very sensitive to the search strategy. For instance, translation methods based on generic first-order resolution use exponential space rather than polynomial space[⋆] for validity checking.

⋆ The superior performance of translation methods with generic resolution provers shown in [26] may be due to better implementation.

3. Preliminaries on Modal Logics

Some familiarity with the language and the semantics of propositional modal logics is assumed (an introduction can be found in [15, 22, 25]). We construct *modal formulae* A, B, C from propositional letters $p \in \mathcal{P}$ with the connectives $\wedge$, $\neg$, and $\Box$ as follows:

$$A, B \ ::= \ p \mid \neg A \mid A \wedge B \mid \Box A.$$

Other connectives may be seen as abbreviations, e.g., $\Diamond A \equiv \neg\Box\neg A$. Formulae of the kind $\Box A$ are referred as ν-formulae (from necessity) whereas $\neg\Box A$ as π-formulae (from possibility).

The semantics of modal logic is based on *Kripke models*, that is, triples $\langle W, \mathcal{R}, V\rangle$, where W is a nonempty set, whose elements are called *worlds*, $\mathcal{R}$ is a binary *accessibility relation* over W, and $V()$ is a function, called *valuation*, from propositional letters to subsets of worlds. Intuitively the worlds in $V(p) \subseteq W$ are those where p is true.

Different modal logics are obtained by different properties of $\mathcal{R}$. In Table I we list the most common properties and the corresponding axiom schema. The combination of the axiom schemata from K to 5 (or the properties of the accessibility relation) generates the logics K, KB, K4, K5, K45, KD, KDB, KD4, KD5, KD45, KB4, T, B, S4, S5 [15, 22], whereas Cxt is McCarthy's logic for contextual reasoning [2]. Our aim is to provide a modular calculus for these logics, and therefore we refer to L-models, L-tableaux, etc., when referring to models, tableaux, etc., for one of the logics L above. Other logics are discussed in [32].

If $\mathcal{M}$ is an L-model, w is a world, and A is a formula, the entailment relation $\mathcal{M}, w \Vdash A$ is defined in the following way:

$$\begin{array}{lll}
\mathcal{M}, w \Vdash p & \text{iff} & w \in V(p), \\
\mathcal{M}, w \Vdash A \wedge B & \text{iff} & \mathcal{M}, w \Vdash A \text{ and } \mathcal{M}, w \Vdash B, \\
\mathcal{M}, w \Vdash \neg A & \text{iff} & \mathcal{M}, w \nVdash A, \\
\mathcal{M}, w \Vdash \Box A & \text{iff} & \forall v \in W : w\mathcal{R}v \text{ implies } \mathcal{M}, v \Vdash A.
\end{array}$$

When $\mathcal{M}, w \Vdash A$ holds, we say that w *satisfies* A *in* $\mathcal{M}$. If S is a set of formulae, then a world w satisfies S iff for all $A \in S$ one has $\mathcal{M}, w \Vdash A$. For sake of readability, we omit $\mathcal{M}$ whenever it is clear from the context. An L-model $\langle W, \mathcal{R}, V\rangle$ *validates* S iff every world in W satisfies S. A formula A is L-*valid* if every L-model validates A.

For logical consequence we need two sets of modal formulae [13, 15]: global assumptions G and local assumptions U. Intuitively the former are true in every world and the latter in the current world.

DEFINITION 3.1. A formula A is an L-*logical consequence* of global assumptions G and local assumptions U (in symbols $G \models_{\mathsf{L}} U \Rightarrow A$) iff in every L-model $\langle W, \mathcal{R}, V\rangle$ that validates G if a world w satisfies U, then w satisfies A. It is L-*satisfiable* for the global assumptions G and the local assumptions U if there is an L-model $\langle W, \mathcal{R}, V\rangle$ that validates G and a world $w \in W$ that satisfies $U \cup \{A\}$.

Table I. Axioms and accessibility relations

Axiom	Accessibility relation
K : $\Box(A \supset B) \supset (\Box A \supset \Box B)$	any
D : $\Box A \supset \Diamond A$	$\forall w \exists v : w\mathcal{R}v$
T : $\Box A \supset A$	$\forall w : w\mathcal{R}w$
4 : $\Box A \supset \Box\Box A$	$\forall w, v, u : w\mathcal{R}v \ \& \ v\mathcal{R}u \supset w\mathcal{R}u$
B : $\Diamond\Box A \supset A$	$\forall w, v : w\mathcal{R}v \supset v\mathcal{R}w$
5 : $\Diamond\Box A \supset \Box A$	$\forall w, v, z : w\mathcal{R}v \ \& \ w\mathcal{R}z \supset v\mathcal{R}z$
Cxt : $\Diamond\Box A \supset \Box\Box A$	$\forall w, u, v, z : w\mathcal{R}v \ \& \ w\mathcal{R}u \ \& \ v\mathcal{R}z \supset u\mathcal{R}z$

We speak of local (global) logical consequence when $G = \emptyset$ ($G \neq \emptyset$).

Local logical consequence is semantically and computationally identical to validity: $G \models_{\mathsf{L}} U \Rightarrow A$ is equivalent to $G \models_{\mathsf{L}} \emptyset \Rightarrow (\bigwedge U) \supset A$. It can make a difference for the proof theory: the Gentzen calculus for logic B in [13, 20] requires cut when U is used instead of $\bigwedge U$.

Global assumptions make a difference [22]: the decision problem for validity is PSPACE but is EXPTIME for logical consequence.

4. Single Step Tableaux

SST use *prefixed formulae*, that is, pairs $\sigma : A$, where the *prefix* σ is a non-empty sequence of integers and A is a modal formula. Intuitively σ "names" a world that satisfies A. In the sequel, σ is a prefix, $\sigma_0.\sigma_1$ the concatenation of the sequence σ_0 with the sequence σ_1 and $\sigma.n$ the concatenation of σ with n. If $\sigma = n_1.n_2.\ldots.n_{k-1}.n_k$ is a prefix, the length of the prefix σ is k and is denoted by $|\sigma|$.

4.1. SINGLE STEP RULES

A L-*tableau* $\mathcal{T}$ is a (binary) tree whose nodes are labeled with prefixed formulae. A L-*branch* $\mathcal{B}$ is a path from the root to a leaf. Nodes are added and labeled in the usual way by applying the rules [13, 20]: if the antecedent of a rule appears in a branch, we extend the branch (and possibly split it), labeling the new node(s) with the consequent(s). A prefix is L-*present* in a branch $\mathcal{B}$ if there is a prefixed formula in $\mathcal{B}$ with that prefix, and it is *new* if it is not present.

Propositional tableau rules are shown in Figure 1a. Figure 1b shows the rules that do not depend on the particular logic L. The rules that vary from logic to logic are in Figure 1c. They are mostly for ν-formulae.

The prefixes present in a branch form a *tree*, spanning from 1 to 1.1, 1.1.1 and 1.1.2, etc. Loosely speaking, the proof search can be seen as a backward and

$$\alpha: \frac{\sigma : A \wedge B}{\begin{array}{c}\sigma : A \\ \sigma : B\end{array}} \qquad \beta: \frac{\sigma : \neg(A \wedge B)}{\sigma : \neg A \;\; \sigma : \neg B} \qquad \text{dneg}: \frac{\sigma : \neg\neg A}{\sigma : A}$$

Figure 1a. Propositional tableaux rules.

$$Loc: \frac{\vdots}{1 : B} \quad \text{if } B \in U$$

$$Glob: \frac{\vdots}{\sigma : B} \quad \text{if } \sigma \text{ is present in the branch and } B \in G$$

$$\pi: \frac{\sigma : \neg\Box A}{\sigma.n : \neg A} \quad \text{with } \sigma.n \text{ new in the branch}$$

Figure 1b. SST rules common to all logics.

$$K: \frac{\sigma : \Box A}{\sigma.n : A} \qquad D: \frac{\sigma : \Box A}{\sigma : \neg\Box\neg A} \qquad T: \frac{\sigma : \Box A}{\sigma : A}$$

$$4: \frac{\sigma : \Box A}{\sigma.n : \Box A} \qquad 4^R: \frac{\sigma.n : \Box A}{\sigma : \Box A} \qquad B: \frac{\sigma.n : \Box A}{\sigma : A}$$

$$4^\pi: \frac{\sigma.n : \neg\Box A}{\sigma : \neg\Box A} \qquad 4^D: \frac{\sigma.n : \Box A}{\sigma.n.m : \Box A} \qquad Cxt: \frac{\sigma.n : \Box A}{\sigma : \Box\Box A}$$

Where σ, $\sigma.n$ and $\sigma.n.m$ must be present in the branch.

Figure 1c. SST-rules characterizing logics.

forward visit of this tree until a contradiction is found: create a node (π-rule), work inside a node (propositional, *Glob* or *Loc* rules), add a formula *forward* from a parent σ to a child $\sigma.n$ or *backward* from child to parent (ν-rules). For instance, rule (4), if we have in mind the knowledge interpretation of $\Box$, corresponds to positive introspection and allows us to "inherit knowledge forward".

The logics mentioned in Table I are captured by different sets of SST rules as shown in Table II. Logic Cxt is captured by rules $K + Cxt$. In the sequel SST-rules are in italics and between parentheses; for example, K is the logic whereas (K) is the ν-rule. Square-bracketed rules are derivable from the others; for example, rule (K) can be simulated by rules (T) and (4):

Table II. Modal logics and SST-rules

Logic	SST-rules	Logic	SST-rules	Logic	SST-rules
K	K	KD	K, D	T	K, T
KB	K, B	KDB	K, B, D	B	K, T, B
K4	$K, 4$	KD4	$K, D, 4$	S4	$[K], T, 4$
K5	$K, 4^D, 4^R, Cxt$	KD5	$K, D, 4^D, 4^R, Cxt$	KB4	$K, B, 4^\pi, 4^R$
K45	$K, 4, 4^R$	KD45	$K, D, 4, 4^R$	S5	$[K], T, 4, 4^R$
K45	$K, 4^\pi, 4^R$	KD45	$K, D, 4^\pi, 4^R$	S5	$K, T, 4^\pi, 4^R$

$$\sigma : \Box A \xrightarrow{4} \sigma.n : \Box A \xrightarrow{T} \sigma.n : A$$
$$\sigma : \Box A \xrightarrow{\qquad K \qquad} \sigma.n : A$$

Remark. For the logics K5 and KD5 we only need to apply rule (Cxt) to prefixes $\sigma.n$ with $\sigma = 1$.

We have two possible sets of rules for the logics K45, KD45, S5. The first can be seen as the standard cumulative extension of K4 with the addition of extra rules. The second dispatched forward rules in favor of backward ones.* The difference is the way in which rules encode the transitivity axiom. Rule (4) encodes $\Box A \supset \Box\Box A$ while rule (4^π) encodes $\Diamond\Diamond A \supset \Diamond A$. Although the combination $K + 4^\pi$ is not complete for K4, the addition of (4^R) is enough for K45. Surprisingly, the proof search in both calculi has the same computational complexity (Section 9).

4.2. TABLEAU PROOFS AND SATISFIABILITY WITNESSES

DEFINITION 4.1. A *branch* $\mathcal{B}$ is *closed* if there is a σ such that, for some A, both $\sigma : A$ and $\sigma : \neg A$ are present in $\mathcal{B}$. A *tableau* is *closed* if every branch is closed.

DEFINITION 4.2. An L-*tableau proof* for the formula A with global assumptions G and local assumptions U is the closed L-tableau for G and U starting with $1 : \neg A$.

For any logic L among K, KB, K4, K5, K45, KD, KDB, KD4, KD5, KD45, KB4, T, B, S4, S5, Cxt, the calculus is correct (proofs in Section 10):

THEOREM 4.1 (Strong Soundness). *If A has an* L-*tableau proof with local assumptions U and global assumptions G, then $G \models_{\mathsf{L}} U \Rightarrow A$.*

* This calculus has been found independently by Goré [20].

THEOREM 4.2 (Strong Completeness). *If* $G \models_{\mathsf{L}} U \Rightarrow A$, *then* A *has an* L*-tableau proof with local assumptions* U *and global assumptions* G.

To provide a counterexample for the validity of A, we need a branch $\mathcal{B}$ without contradictions. We need some preliminary definitions.

DEFINITION 4.3. A prefixed formula $\sigma : A$ is *reduced* for rule (r) in $\mathcal{B}$

- if (r) has the form $\sigma : A \Rightarrow \sigma' : A'$ and $\sigma' : A'$ is in $\mathcal{B}$;
- if (r) has the form $\sigma : A \Rightarrow \sigma_1 : A_1 \mid \sigma_2 : A_2$ and at least one of $\sigma_1 : A_1$ and $\sigma_2 : A_2$ is in $\mathcal{B}$.

The formula $\sigma : A$ is *fully reduced* in $\mathcal{B}$ if it is reduced for all applicable rules. A prefix σ is (fully) reduced if all prefixed formula $\sigma : A$ are (fully) reduced.

For instance, reduction w.r.t. rule (4^R) is $\sigma.n : \Box A \in \mathcal{B}$ implies $\sigma : \Box A \in \mathcal{B}$. Notice that a formula may be reduced even if no rule has been applied to that particular formula.⋆

DEFINITION 4.4. A *branch* $\mathcal{B}$ is *completed* if all prefixes in $\mathcal{B}$ are fully reduced for $\mathcal{B}$; it is *open* if it is completed and not closed. A *tableau* is *open* if at least a branch is open.

Soundness and completeness can be reformulated as follows.

THEOREM 4.3. *The formula* A *is* L*-satisfiable for local assumptions* U *and global assumptions* G *iff there is an open* L*-tableau for* U *and* G *starting with* $1 : A$.

With Definition 4.4, open branches may well be infinite, as in first-order logic. For instance, if $\Diamond A \in G$, then rules $(Glob)$ and (π) can generate an infinite sequence of prefixes. Rules (π), (K), and (4) generate an infinite branch when applied to formulae like $1 : \Diamond B \wedge \Box\Diamond A$.

If the focus is provability and not decision procedures, a completed branch is simply the result of any systematic and fair search strategy [13, 20]. To obtain decision procedures, we must restrict the applicability of tableau rules. Some techniques are obvious.

Technique 4.1. Apply rule (r) to a prefixed formula $\sigma : A$ in $\mathcal{B}$ only if the formula is not already reduced according to Definition 4.3.

⋆ For instance, there are two cases in which a π-formula $\sigma : \neg\Box A$ can be the reduced for rule (π). The obvious way is to apply the rule to $\sigma : \neg\Box A$ and introduce a new prefixed formula $\sigma.n : \neg A$. However, some other rule may have already introduced a formula $\sigma.n : \neg A$. In this case there is no need to apply rule (π).

This is not enough to tame the examples mentioned above. A solution is a more or less smart form of "loop-checking" [30, 7, 13, 20, 22, 25, 23, 32]: loosely speaking, never reduce a *set of formulae* if we have met and reduced this set beforehand. We discuss this issue in Section 8.

5. From Gentzen-type and Prefixed Tableaux to SST

We can see a tableau proof as a failed attempt to find a countermodel for a valid formula. Loosely speaking, the proof search is a "journey" through a tentative Kripke model, jumping from world to world, chasing a contradiction. The difference between Gentzen-type, prefixed, and single step tableaux is precisely the way they journey.

5.1. FORWARD RULES IN GENTZEN-TYPE AND PREFIXED TABLEAUX

The main intuition behind SST is inherited from Kripke [29].

Intuition. SST views the prefixes within a branch as a tree, labels nodes with a single formula, reduces formulae one by one, and moves formulae only toward a child or a parent node in the tree of prefixes.

Gentzen-type tableaux reduce modal connectives by transforming a whole set of modal formulae S into another set $S^{\#}$. For instance, the logic K4 is characterized by the following rule:

$$\frac{\{\neg\Box A\} \cup S}{\{\neg A\} \cup S^{\#}} \text{ where } S^{\#} \doteq \{B, \Box B \mid \Box B \in S\}.$$

In this way tableaux are trees where each node is labeled by a set of formulae. If we arrange deduction in a way that a single formula labels a node of the tree, then we must use a *global branch modification rule*: delete all nodes in a branch and replace them with others [13].

Intuition. Such transformation corresponds to a jump forward from the world described by S to the world described by $S^{\#}$. It is a jump along one arc of the accessibility relation. We never look back again. SST decompose such a "large" jump into many "leaner" steps that take only one formula as input.

The difference between such a global rule and the SST rules is shown in Figure 2. We can simulate the K4 jump with many subsequent applications of the (K) and (4) SST rules. We delete formulae in the Gentzen calculus while the SST rule α can still be applied to $1 : C \wedge D$.

Soundness is at stake: a complex rule can be globally sound, but its "stepwise computations" may not be such. The key point is that the correctness proofs of Gentzen-type tableaux [12, 13, 20] are stepwise. In a nutshell, if $w\mathcal{R}v$, we pick

K4 Gentzen-type reduction	K4 single step reduction
$\dfrac{\{\neg\Box A, \Box B, C \wedge D\}}{\{\neg A, B, \Box B\}}$	$1 : \neg\Box A \quad (a)$ $1 : \Box B \quad (b)$ $1 : C \wedge D$ $1.1 : \neg A \quad (\pi)$ to a $1.1 : B \quad (K)$ to b $1.1 : \Box B \quad (4)$ to b
The first set describes world 1 the second describes world 1.1.	

Figure 2. Gentzen-type tableau vs SST for K4.

each $A^{\#} \in S^{\#}$ and show that its truth value in v is forced by the truth value in w of a corresponding $A \in S$.

Prefixed tableaux à la Fitting [13] introduce prefixes to "name" worlds and mimic the behavior of Kripke semantics with a syntactic relation $\rhd$ between prefixes. For instance, in the case of K4, if $1 \rhd_{\mathsf{K4}} 1.1$ and $1.1 \rhd_{\mathsf{K4}} 1.1.1$, then we have $1 \rhd_{\mathsf{K4}} 1.1.1$. Deduction is performed by considering a single formula $\sigma : \Box A$ and adding $\sigma^* : A$ for every σ^* such that $\sigma \rhd \sigma^*$.

Intuition. This deduction steps corresponds to a jump from the world named σ to another (possibly far away) world, named σ^*. SST break "long" jumps into sequences of "shorter" steps.

The difference between prefixed tableaux and SST is shown in Figure 3. SST "emulates" the effect of transitivity by first moving to world 1.1 a ν-formula and then using rule (K).

Completeness is under siege: in a prefixed tableau we can jump far away, and in a single step tableau we may lack the intermediate worlds needed to fill the gap. Again, the analysis of the completeness proof for prefixed tableaux [13] reveals that all intermediate worlds are there.

5.2. BACKWARD RULE AND ANALYTIC CUT

Euclidean and symmetric logics have accessibility relations where an arc between two worlds in one direction ("forward") imposes the existence of another arc in the opposite direction ("backward").

Intuition. In these logics, "discovering" a formula in the future forces constraints on the past. For logical consequence Gentzen-type tableaux must use cut* to guess the formulae that will be discovered.

* The cut rule is defined as $\dfrac{S}{S\cup\{A\} \quad S\cup\{\neg A\}}$. The rule makes it possible to guess the truth value of the formula A. See [13] for a discussion.

K4 prefixed tableau	K4 single step tableau
$1 : \Box A$	$1 : \Box A$
|	$11 : \Box A$ (4)
$1.1.1 : A$	$111 : A$ (K)
Since $1 \rhd_{\mathsf{K4}} 1.1.1$	

Figure 3. Prefixed rule vs SST for K4.

Gentzen-type deduction for KB

$$\dfrac{\dfrac{\dfrac{\{\neg\Box\neg\Box A\}}{\{\neg\Box\neg\Box A, \neg A\}} \quad \{\neg\Box\neg\Box A, A\}}{\{\neg\neg\Box A, \neg\Box\neg\neg A\}}}{\dfrac{\{\Box A, \neg\Box\neg\neg A\}}{\{\neg\neg\neg A, A, \neg\Box\neg\Box A\}}}$$

$$\{\neg A, A, \neg\Box\neg\Box A\}$$

$$\bot$$

Single step tableau

$1 : \neg\Box\neg\Box A$	
$1.1 : \neg\neg\Box A$	(π)
$1.1 : \Box A$	$(dneg)$
$1 : A$	(B)

For the Gentzen-type derivation we apply first the cut rule and branch the search. On the left branch we apply $dneg$ rules (not marked) and KB rules. The initial set corresponds to world 1; the first KB step leads to world 1.1 and the next in 1.1.1.

Figure 4. Backward rule rather than cut for K45.

SST use backward rules to bring back newly discovered formulae in previous worlds. We gain some of the computational power of cut without its branching factor

In Figure 4 we compare the Gentzen-type rule for KB with the corresponding backward rule in SST. For the Gentzen calculus the KB-rule transforms $\{\neg\Box A\} \cup S$ into $\{\neg A\} \cup \{B, \neg\Box\neg C \mid \Box B, C \in S\}$ [13, 20].

Notice that cut is necessary for the completeness of local logical consequence in KB. An example is $\models_{\mathsf{KB}} \{A, B\} \Rightarrow \Box\neg\Box\neg(A \wedge B)$ [13].

6. Modularity

The first property of interest is *modularity*. Hilbert axiomatizations are modular "par excellence": once we fix the set of rules and axioms that we want to use, it is clear that the logic KD4 is an extension of (has more theorems than) K. Translation methods based on the relational translation [18] or the functional translation with an explicit equality theory [39] are modular: change the first-order (equality) axioms representing the accessibility relation and change the logic.

Gentzen calculi and the corresponding tableaux [13, 20] are much less modular since the logic dependent part is compiled into one single rule for π-formulae. To

change the logic, we must change the whole rule. The only modular part is the presence of a single rule for reflexivity.

SST have the same cumulative flavor of Hilbert-style axiomatizations: the set of available rules is given in Table II, and by taking different subsets we obtain different logics. The calculus with rule (K) and (4) is clearly an extension of the calculus using only rule (K).

Notice that there is not a 1-1 mapping between axioms and SST-rules. For instance SST can simulate the effect of three axioms $K + T + 4$ with two rules $(T) + (4)$. On the other hand, rule (4^R) alone cannot simulate the power of axiom 5. In a similar way $(K) + (T) + (4) + (B)$ is not enough to get completeness for S5. Overlooking this fact leads to incomplete formalizations. For instance the rules for K5 are not complete in [32], and the rules for S5 are not complete in [29].

7. Proof Confluence

The study of confluence (Problem 2) has a long tradition in rewrite systems, and we refer to [24] for an introduction.

7.1. PRELIMINARY DEFINITIONS AND COMPARISONS

In the sequel x, y, z, u will be stages of the computation and $\rightarrow$ will be a relation between them. Here, the computation is the proof search, and its stages are tableaux, sets of clauses, etc. For tableaux, $x \rightarrow y$ if the tableau y is obtained from x with an application of a tableau rule. For resolution, x is set of clauses and y the same set plus a resolvent of two clauses in x or minus a subsumed clause. The relation $\rightarrow^*$ is the reflexive, transitive closure of $\rightarrow$ whereas $\rightarrow^\epsilon$ is the reflexive closure.

DEFINITION 7.1 [24, p. 779]. The relation $\rightarrow$ is *strongly confluent* iff

$$\forall xyz.\ \text{if } x \rightarrow y \text{ and } x \rightarrow z \text{ then } \exists u.\ y \rightarrow^* u \ \text{ and } \ z \rightarrow^\epsilon u.$$

This form of confluence is too strict, and it is satisfied only by *modal resolution* [9]. We are usually interested in confluence modulo an equivalence relation. For instance, in first-order resolution we want confluence up to renaming of bound variables. If $\sim$ is an equivalence relation between stages of the computation, we have (see [24, page 802]) the following definition.

DEFINITION 7.2. The relation $\rightarrow$ is *strongly confluent modulo* $\sim$ iff

$$\begin{array}{lll} \forall x_1 x_2 y_1 y_2 & \text{if} & x_1 \sim x_2 \text{ and } x_1 \rightarrow y_1 \text{ and } x_2 \rightarrow y_2 \\ & \text{then} & \exists u_1 u_2.\ u_1 \sim u_2 \text{ and } y_1 \rightarrow^* u_1 \text{ and } y_2 \rightarrow^\epsilon u_2. \end{array}$$

Clearly the *functional and relational translation methods* based on resolution [18, 36, 38, 39] are strongly confluent modulo isomorphic renaming of bound variables. They remain so using subsumption.

Among direct methods *Fitch-style tableaux or Gentzen calculi are not confluent*. The culprit is the π rule (e.g., in Section 5): replace the set $S \cup \{\neg\Box A\}$ with the set $S^{\#} \cup \{\neg A\}$, where $S^{\#}$ is obtained from S by considering a subset of the formulae of S. Typically one focuses on $\Box B \in S$ (or $\neg\Box B$) and deletes the other formulae [12, 13, 20].

Backtracking becomes the only choice, and the search strategy becomes critical [8]. We have a partially negative answer to Problem 2: being a systematic strategy is not enough to be a fair (and hence complete) one. Consider the following selection technique.

Technique 7.1. Select formulae that have been reduced the least number of times; then select π formulae, then β formulae.

This strategy is clearly systematic but is not complete for Gentzen-type tableau: try $\{\Box A \vee \Box B, \neg\Box(A \vee B)\}$. On the contrary, it is complete for SST. More examples of systematic yet incomplete strategies for Gentzen-type tableaux are discussed in [8].

The practical consequence is that we must do a lot of (maybe pointless) case analysis with β-rules before trying any π-rules. This substantially hinders the effectiveness of the proof search [26].

7.2. SST CONFLUENCE

Single step tableaux are strongly confluent modulo a renaming of prefixes. To prove this result, we need some preliminary definitions.

DEFINITION 7.3. An injective and surjective function h from the set of prefixes onto itself is a *renaming* iff $h(1) = 1$ and $h(\sigma n) = h(\sigma)m$ for some integer m.

Intuitively, a renaming permutes the integers and leaves the structures of prefixes unchanged.

DEFINITION 7.4. The sets of prefixed formulae $\mathcal{B}_1$ and $\mathcal{B}_2$ are equivalent modulo a renaming of prefixes iff there are two renamings h_{12} and h_{21} such that $h_{ij}(h_{ji}(\sigma)) = \sigma$ and if $\sigma : A \in \mathcal{B}_i$ then $h_{ij}(\sigma) : A \in \mathcal{B}_j$ for $i, j = 1, 2$.

This definition can be extended to tableaux as sets of branches.

THEOREM 7.1. *If ν-formulae can be reduced more than once, SST rules are strongly confluent modulo isomorphic renaming of prefixes.*

We discuss the key steps of the proof, leaving details in the appendix. The difficulties are due to rules with side conditions, for example, the π-rule (Figure 1b) and the ν-rules (Figure 1c), since conditions impose an ordering.

The first case is when a prefix must be introduced by a (π) rule before we can apply rule (K) to it. This also happens for matrix-based methods (e.g., K-admissible substitution [47, page 113]).

Still, confluence does not require an arbitrary shuffling of rules: it requires that **if** we can choose which rule to apply, **then** the choice is irrelevant. With rules (π) and (K) we cannot choose: if we can apply (K) to a prefix $\sigma.m$ present in $\mathcal{B}$, this prefix is not new. Dually, if we apply a (π)-rule to a branch $\mathcal{B}$ and introduce a new prefix $\sigma.n$, this prefix is not present in $\mathcal{B}$. So we cannot apply (K) to $\sigma.n$ in $\mathcal{B}$.

The other case is when two π-formulae, $\sigma : \neg\Box A$ and $\sigma : \neg\Box B$ can be reduced in a branch $\mathcal{B}$. After the reduction of $\sigma : \neg\Box A$ the prefix $\sigma.n_1$ is no longer new. Thus the reduction of $\sigma : \neg\Box B$ needs to introduce another new prefix $\sigma.m_1$. By changing the order we do not get the same result. The partial computation $\mathcal{B}_1 = \mathcal{B} \cup \{\sigma.n_1 : \neg A,\ \sigma.m_1 : \neg B\}$ is potentially different from $\mathcal{B}_2 = \mathcal{B} \cup \{\sigma.n_2 : \neg A, \sigma.m_2 : \neg B\}$.

We introduce the following renamings h_{12} and h_{21} and we are done.

$$h_{ij}(s) = \begin{cases} \sigma.n_j & \text{if } s = \sigma.n_i, \\ \sigma.m_j & \text{if } s = \sigma.m_i, \\ s & \text{otherwise.} \end{cases}$$

Remark. For the logics K45, KD45 and S5, we can choose the calculus with rule (4^{π}). Then π-formulae must also be reducible more than once.

8. Decision Procedures

There are a number of ways in which *decision procedures* can be designed using a logical calculus. The traditional way is to keep the original calculus and constrain the applicability of rules (e.g., [8]). An alternative approach is to revise the calculus, embedding the constraints in the proof theory (e.g., [23]). Both approaches have their advantages and limitations: the former uses a simple calculus by keeping implicit the termination conditions. The second makes termination and the search strategy explicit features of the (new) calculus at the expenses of its clarity and flexibility.

As mentioned in Section 2, we would like some simple (local) conditions that prevent the application of the rules in the *original* calculus.

This objective seems difficult for *translation methods*, since first-order logic is not decidable: for functional translations [36, 39, 47], the mapping of worlds into terms containing variables makes it possible to generate infinite sets of resolvents even with simple modal formulae. In Figure 5 we show a simple (Horn) satisfiable formula with Nonnegart's translation [36]. The prolog terms $i, i{:}f, i{:}f{:}g$ are called world paths [39]. The corresponding intuition is that there is a an $\mathcal{R}$ path in the model such that $w_i \mathcal{R} w_{i:f} \mathcal{R} w_{i:f:g}$ and so on.

Formula $p \wedge \Box(p \vee \Diamond\neg p) \wedge \Diamond\neg p$ in KD45

Translation clauses	SLD resolvents
$r(U, V : X).$	$\leftarrow p(i : f)$
$p(i).$	$\leftarrow p(i : f : g)$
$p(W) \leftarrow r(i, W), p(W : g)$	$\leftarrow p(i : f : g : g)$
$\leftarrow p(i : f).$	$\leftarrow p(i : f : g : g : g)$
	$\ldots$

Figure 5. Infinite resolvents for translation methods.

We shall see how a simple technique, a local restriction on first-order terms corresponding to worlds, makes translation methods decidable.

To design decision procedures for *direct methods* such as modal resolution [9] or Gentzen-type calculi or tableau [13, 20], one typically uses the property of analyticity. The basic idea is that a calculus is *analytic* iff its rules and axioms use subformulae only of the formula we want to prove [46]. In the case of logical consequence we may need to use also subformulae of the set of local and global assumptions.

If every analytic rule introduces in the consequents only proper subformulae of the formulae of the antecedent, then technique 4.1 is the only thing we need to terminate. This is the case for Gentzen-type calculi and the related tableau methods for the logic K, *if we restrict ourselves to validity*, that is, with an empty set of global assumptions.

In the general case, we must use *loop checking* [8, 13, 20, 25]: before applying any rule we check whether we haven't applied it already to the same antecedent; if this is the case we block the application of the rule. If the calculus is analytic, this technique guarantees the termination of the proof search [8, 20].

A weaker property corresponds to the notion of Fischer–Ladner closure for dynamic logics [11]: we can define a *finite superset* of subformulae of U, G and the formula we want to prove. Then a calculus is *super-analytic* iff the consequent of any rule contains only formulae in that superset [20]. When loop checking is used, this is sufficient for termination.

Implementations must store the trace of previous computations and, more or less often, verify whether a rule has been already applied. This process may be extremely expensive [8].

It is possible to compile the loop-checking technique into the rules of the calculus, as done in [23]: we add a constraint corresponding to previous application of a rule and then modify the rules so that they propagate both formulae and constraints. This approach doesn't avoid the problem of storing previous computations: it simply stores this information in the constraints, although this is done in clever way to store only the minimal relevant information.

An alternative approach [5] is to translate each logic into K, for which blind search always terminates. The difficult part of this techniques is crafting a polynomial translation, which may vary from logic to logic.

Designing decision procedures for *prefixed tableaux*, either à la Fitting or SST, is easier than translation methods (there are no variables) but harder than direct methods (we have "extra" prefixes).

If we neglect prefixes, Fitting's tableaux [13, Chap. 8] are analytic, and SST are also analytic with the exception of the logics with the rule (Cxt). For validity, if the rule (Cxt) is not coupled with rule (4) or (4^D), then it is easy to prove that SST are super-analytic w.r.t. the set $S^* = S \cup \Box S \cup \cdots \Box^d S$, where S is the set of subformulae of $U \cup \{\neg A\}$ and d is the maximum modal nesting of formulae in S. In case of K5 the calculus remains super-analytic if we prove the *once-off* property as in [13]: rule (Cxt) can be applied only once. Indeed this is the case after we have restricted rule Cxt to prefixes $\sigma.n$ with $\sigma = 1$. The superset we are looking for is $S^* = S \cup \Box S$ (see also [20] for a discussion).

8.1. INCORPORATING "LOOP CHECKING" IN SST

The intuition behind loop checking is stopping the search whenever two prefixes are "a different name for the same state". We change the definition of completed branches (Def. 4.3) to incorporate this idea.

We say that a prefix σ is a *copy* of a prefix σ_0 for branch $\mathcal{B}$ if for every formula A one has $\sigma : A \in \mathcal{B}$ if and only if $\sigma_0 : A \in \mathcal{B}$.

DEFINITION 8.1. A prefix is π-reduced in $\mathcal{B}$ if it is reduced for all rules except rule (π). A branch $\mathcal{B}$ is *π-completed* if (i) all prefixes are π-reduced in $\mathcal{B}$, and (ii) for every σ that is not fully reduced there is a fully reduced copy σ_0 shorter than σ.

The intuition is that to avoid infinite computations, π-rules should not be applied to formulae belonging to copies. Any complete search strategy must now prove that it always leads to a π-completed branch. A simple strategy that works together with technique 4.1 is the following.

Technique 8.1. Select prefixed formulae with the shortest prefix.

The final outcome is a π-completed branch although the proof search may not terminate: we only guarantee completion "ad infinitum". To guarantee termination, we may use the following technique.

Technique 8.2. Before reducing a π-formula, check whether the corresponding prefix is not a copy of a shorter prefix.

This is exactly the loop-checking method in [8].

Remark. The difference between SST and Gentzen-type tableaux is that SST are proof confluent. We do not need to backtrack once we find a loop; we leave the "copies" and focus on other parts of the branch.

Strategies that guarantee π-completeness are complete.

THEOREM 8.1. *If the* L*-tableau with local assumptions U and global assumptions G starting with* $1 : A$ *terminates with a π-completed branch, then A is* L*-satisfiable for U and G.*

For logics K4 and S4 it is useful to sharpen the notion of π-completeness by using the notion of *modal copy*: σ is a modal copy of σ_0 in $\mathcal{B}$ if it has the same ν formulae.

DEFINITION 8.2. A branch $\mathcal{B}$ is *π-modal-completed* if (i) all prefixes are π-reduced for $\mathcal{B}$, and (ii) for every prefixed formula $\sigma : \neg\Box A$ that is not reduced there is a shorter modal copy σ_0 of σ such that $\sigma_0 : \neg\Box A$ is reduced.

For K4 or S4 Theorem 8.1 can be extended* to π-modal-completeness.

Once a branch is either π-completeness or π-modal-completed, we can "shorten" it without losing this property. First we introduce the *forward tree* of prefixes rooted at σ in branch $\mathcal{B}$:

$$\text{Ftree}(\sigma) = \{\sigma^* : A \mid \sigma^* : A \in \mathcal{B} \text{ and } \sigma \text{ is an initial subsequence of } \sigma^*\}.$$

The prefix σ is in Ftree(σ), since σ is an initial subsequence of itself.

Second, we say that a branch $\mathcal{B}$ is *pruned* into $\mathcal{B} \setminus \text{Ftree}(\sigma)$ if the set of prefixed formulae $\mathcal{B} \setminus \text{Ftree}(\sigma)$ is obtained from $\mathcal{B}$ by deleting all prefixed formulae in Ftree(σ).

Third, the prefixed formula $\sigma : \neg\Box A$ is *fulfilled* by $\sigma.n$ in $\mathcal{B}$ if $\sigma : \neg\Box A$ is reduced for rule (π) in $\mathcal{B}$ because $\sigma.n : \neg A$ is in $\mathcal{B}$ (see Def. 4.3).

The result we are looking for is the following.

LEMMA 8.2. (Pruning Lemma). *Let $\mathcal{B}$ be a branch and $\sigma.n$ a prefix such that for every $\sigma : \neg\Box A$ that is fulfilled by $\sigma.n$ there is shorter copy* (*resp. modal copy*) *σ_0 such that $\sigma_0 : \neg\Box A$ is reduced in $\mathcal{B}$. If $\mathcal{B}$ is π-completed* (*resp. π-modal-completed*)*, then $\mathcal{B} \setminus Ftree(\sigma.n)$ is π-completed* (*resp. π-modal-completed*).

Proof. This operation only changes the status of the π-formulae $\sigma : \neg\Box A$ in $\mathcal{B} \setminus \text{Ftree}(\sigma.n)$. By definition of π-(modal)-completeness, either $\sigma_0 : \neg\Box A$ is reduced or there is a shorter (modal) copy σ_{00} such that $\sigma_{00} : \neg\Box A$ is reduced. By hypothesis σ is a (modal) copy of σ_0, and in either case we have a reduced (modal) copy of the prefix σ. □

* This is not the case for logics KB and B. For instance, use $G = \{\Diamond(\Box B \wedge \Diamond\Diamond\neg B)\}$ and $A = \top$.

Table III. Input size parameters

Par.	Definition
d	maximum nesting of modal operators in $nnf(U \cup \{\neg A\})$
n	number of ν-subformulae in $nnf(U \cup \{\neg A\})$
p	number of π-subformulae in $nnf(U \cup \{\neg A\})$
d_p	the maximum nesting of possibility operators $\Diamond$
	not under the scope of any $\Box$ operators in $nnf(U \cup \{\neg A\})$
p_G	number of π-subformulae in $nnf(G)$
n_G	number of ν subformulae in $nnf(G)$
f_G	number of formulae of $nnf(G)$.

The intuition is that π-completeness (modal or not) is a minimal requirement: when pruning a branch that satisfies the preconditions of the pruning lemma, we are just deleting useless reductions.

8.2. DECIDABILITY WITHOUT LOOP CHECKING

Loop checking is a very expensive method and does not match our second objective (Problem 4): a simple technique on the original rules of the calculus that checks only the formula to be reduced.

In the rest of the section, we identify such simple techniques when the set of global assumptions is empty and extend it to translation methods. We focus on *validity* or at most local logical consequence.

For sake of simplicity, we assume that all formulae in G, U, and the initial formula $1 : \neg A$ have been reduced to negation normal form with standard transformations such as $nnf(\neg\Box A) \mapsto \Diamond(nnf(\neg A))$. All rules are also transformed in the obvious way. For example, in rule (π) in Figure 1b we replace $\sigma : \neg\Box A$ with $\sigma : \Diamond A$ and $\sigma.n : \neg A$ with A.

We introduce some parameters to evaluate the size of G, U, and A in Table III. The measure d is also called the "modal depth" of the formula. In $\Box(A \wedge \Diamond\Diamond B) \vee \Diamond C$ we have that $d = 3$, $n = 1$, $p = 3$, and $d_p = 1$. It is $p = 3$ because we must count both $\Diamond\Diamond B$ and $\Diamond B$.

Intuition. For each logic L, there is a height bound hb_{L} on the length of prefixes in a branch after which either there are no more modal operators or formulae just repeat themselves with a longer prefix.

Technique 8.3. For every logic L the application of the single step tableau π-rule is limited to prefixes whose length is less than hb_{L}.

Table IV. Bounds for decidability checks

Logic L	Bound hb_{L}
K, D, T, KB, KDB, B	$1 + d$
K4, KD4, S4	$2 + d_p + p \times n$

This is exactly the kind of *local termination check* we are looking for: it requires to look only at single formulae and, if hb_{L} is polynomial in the size of the input formula, it is also efficient. In Table IV we give the upper bound for a first set of logics. The first result is easy.

LEMMA 8.3. *For local* L*-logical consequence* $\models_{\mathsf{L}} U \Rightarrow A$ *in the logic* L *every strategy that applies techniques* 4.1 *and* 8.3 *with the bounds in Table* IV *terminates.*

The difficult part is proving that we preserve completeness. This proof is extremely difficult for Gentzen-type tableaux (see, e.g., [5, 23]). The major problem is that we cannot universally quantify over strategies because the calculus is not proof confluent and there are incomplete strategies (Section 7). As a consequence, termination proofs have an existential flavor "if there is a proof, then there is a proof such that...".

Thanks to proof confluence, we prove something more general.

THEOREM 8.4. *For local* L*-logical consequence* $\models U \Rightarrow A$ *every strategy which applies techniques* 4.1 *and* 8.3 *with the bound of Table* IV *terminates in one of the following conditions:*

(1) *a tableau proof has been found, or*
(2) *in every branch some rule is still applicable, or*
(3) *at least one branch can be pruned into a* π*-completed branch* (π*-modal-completed for* K4 *and* S4).

Proof [Sketch, details in the appendix]. The difficult case is when the strategy terminates and no rule is applicable in at least a branch $\mathcal{B}$. We have to prove that the branch $\mathcal{B}$ can be pruned into a π-(modal)-completed branch. The proof goes through the following steps:

- prove that if $\sigma : \Diamond A$ is not reduced, then σ must be longer than hb_{L} (all other π-formulae are reduced);
- prune subtrees of prefixes until a branch $\mathcal{B}'$ is found that no longer satisfies the preconditions of the pruning lemma;
- show that the longest chain of prefixes 1, $1.n_1$ up to $1.n_1 \ldots n_k$ that does not trigger the pruning lemma is such that $k \leq hb_{\mathsf{L}}$.

The logics K, D, T, KB, KDB, and B can be treated without difficulty whereas K4 and S4 require more work, especially for the third step. We use the following properties of branches reduced for rule (4).

PROPOSITION 8.1. *If the prefix σ_0 is an initial subsequence of σ in the branch $\mathcal{B}$, then $\sigma_0 : \Box A \in \mathcal{B}$ implies $\sigma : \Box A \in \mathcal{B}$.*

PROPOSITION 8.2. *Let all prefixes σ, $\sigma.n_1$ up to $\sigma.n_1 \ldots .n_k$ have the same ν-formulae; then each $\sigma.n_1 \ldots .n_i$ has been generated by a different π-formula $\Diamond A$.*

Then, a counting argument shows that the longest sequence that does not trigger the pruning lemma has length $hb_{\mathsf{L}} = 2 + d + n \times p$. □

We do not need to prune the branch after the proof search terminates. It is enough to know that we can do it. The calculus in [23] corresponds to a visit of the tree of prefixes directly in the pruned branch $\mathcal{B}'$.

We have *replaced loop checking* for SST with a simple local check.

8.3. DECIDABILITY FOR TRANSLATION METHODS

This result can be easily extended to the *free variables SST* proposed by Beckert and Goré [1] and to other methods. It gives a direct decidability check for functional [38, 39] and mixed [36] translation methods.

Observe that there is a 1-1 mapping between prefixes in (single step) prefixed tableaux and ground terms with functional or mixed translation methods [43, 47] for the logics K, KD, T, K4, KD4, and S4. Use the following technique.

Technique 8.4. Delete resolvents with terms corresponding to worlds paths longer than hb_{L} for a logic L among K, KD, T, K4, KD4, and S4.

This is particularly effective for methods that directly employ resolution. It is also much simpler to implement than the exponential upper bound on multiplicity derived in [5].

We can extend the decidability result to translation methods by combining Theorem 8.4 with the mapping in [43, 47, 5]. We need factoring and condensing; otherwise the number of literals in not subsumed clauses can grow without limits. A direct proof can be found in [44].

THEOREM 8.5. *Resolution with factoring, condensing, and technique* 8.4 *is a decision procedure for the modal logics* K, KD, T, K4, KD4, S4, *with the functional and mixed translation methods.*

An S4 model is also an S5 model. So, this technique also works for K45 and S5. In the example at the beginning of the section, we have that $hb_{\mathsf{S4}} = 2+1+2\times 1 = 5$, and thus we can terminate after few resolvents.

9. Complexity and Search Strategies

The next step is proving (or trying to prove) that we can efficiently upgrade this procedure to solve global logical consequence. A further step is finding further restrictions on the proof search to match the theoretical complexity bounds.

In this section we assume a basic knowledge of complexity theory. An introduction can be found in [28, 27], and some concepts of complexity classes are recalled in the appendix. Recall that deciding validity is a PSPACE-complete problem for modal logics between K and S4 [30] and deciding logical consequence is EXPTIME-complete [22] for K. For K45, KD45, and S5, deciding validity is "only" CO-NPTIME-complete [22].

We need some preliminary observations on the size of the smallest model that can satisfy a given formula.

Fact 9.1. If deciding L-satisfiability is NPTIME-complete, for every L-satisfiable formula A there is an L-model $\langle W, \mathcal{R}, V\rangle$ for A such that its size (number of worlds) is polynomially bounded in the size of A.

Fact 9.2. If deciding L-satisfiability is PSPACE-complete, there are L-satisfiable formulae A such that the size (number of worlds) of the smallest L-model for A is exponential in the size of A.

Fact 9.3. If deciding L-satisfiability is PSPACE-complete, for every L-satisfiable formula A there is an L-model $\langle W, \mathcal{R}, V\rangle$ for A where the length of the longest simple $\mathcal{R}$-path* is bounded by a polynomial in the size of A.

The termination check in Section 8 is a reformulation of this property. Theorem 8.4 simply says that the length of the longest $\mathcal{R}$-path is hb_{L}.

Fact 9.4. If deciding L-logical consequence is EXPTIME-complete, there are formulae A such that $G \not\models_{\mathsf{L}} A$ and that the length of the longest simple $\mathcal{R}$-path in the smallest L-countermodel for A is exponential in the size of A and G.

These facts have a direct impact on the complexity of the proof search, in particular for calculi that work by refuting the theorem, that is, by trying to construct a countermodel.

With an NPTIME-complete problem we can generate directly the countermodels: we "know" that some of them are small (although it may take exponential time to find one or prove that there are none).

With a PSPACE-complete problem we cannot generate whole models, since there are formulae that have exponentially large models. Still, we "know" that we can visit some of them using polynomial space.

* A model $\langle W, \mathcal{R}, V\rangle$ is simply a directed graph. A simple path is a sequence of worlds $w_1, \ldots, w_n$ without repetitions such that $w_i \mathcal{R} w_{i+1}$.

Remark. If the completeness of our proof procedure requires the generation of the whole model, the corresponding algorithm will take *exponential space* rather than polynomial space in the size of the input.

9.1. DEALING WITH LOGICAL CONSEQUENCE

The extension of technique 8.3 and 8.4 to logical consequence seems fairly easy using the following procedure:

(1) keep the rules and the strategy of the calculus for validity (whether direct or based on translation into first-order logic);
(2) apply the modal deduction theorem and transform (global) logical consequence into validity (local logical consequence);
(3) recompute the upper bounds for the resulting problem;
(4) use the decision procedure with the new bounds.

The key question is, how efficient is this procedure? Reconsider again the modal deduction theorem [13, 15]:

$$G \models_{\mathsf{L}} U \Rightarrow A \quad \text{iff} \quad \text{for some } n \text{ one has } \models_{\mathsf{L}} U \cup \bigcup_{i=0}^{n} \Box^i G \Rightarrow A, \tag{1}$$

where $\Box^0 = G$ and $\Box^{i+1} G = \{\Box A \mid A \in \Box^i G\}$.

We can reformulate the question in terms of the value of n. If the value of n is polynomially bounded in the size of G, U and A, then this can be a feasible approach. If n is not polynomially bounded, then the decision procedure could receive an intermediate input with an *exponential blow-up* w.r.t. the original input.

One may argue that the translation used in the deduction theorem is too naive. For instance, the decision procedure for S4 proposed in [5] translates S4 into K using two steps: first it generates an S4-satisfiability preserving modal CNF with the techniques of Mints [35] and then uses a naive translation of S4 into K. The naive translation would yield an exponential blow-up but the preprocessing guarantees a polynomial translation. Unfortunately, clever translations are not possible.

THEOREM 9.1. *For the logics* K, KD, T, KB, KDB, *and* B *the existence of a polynomial translation of logical consequence into validity implies* PSPACE = EXPTIME.

This is an obvious consequence of the fact that validity for K is PSPACE-complete whereas logical consequence is EXPTIME-complete.

COROLLARY 9.2. *For the logics* K, KD, T, KB, KDB, *and* B *the value of* n *in* (1) *cannot be bounded by any polynomial in* n.

Proof. Suppose that n is bounded by a polynomial in the size of G, U and A, that is, $n \leq poly(|G|, |U|, |A|)$.

Then observe that the modal deduction theorem is monotone in n: if (1) holds for some n, then it holds for all $m \geq n$. Indeed, $\models_{\mathsf{L}} U \cup \bigcup_{i=0}^{n} \Box^i G \Rightarrow A$ implies $\models_{\mathsf{L}} U \cup \bigcup_{i=0}^{n} \Box^i G \cup \bigcup_{i=n+1}^{m} \Box^i G \Rightarrow A$ by monotonicity of the underlying modal logic L.

For each G, U and A we use directly the upper bound $n_{\max} = poly(|G|, |U|, |A|)$ and apply the deduction theorem with $n = n_{\max}$. The generation of $\bigcup_{i=0}^{n_{\max}} \Box^i G$ can be done in polynomial time in the value of $n_{\max}$ and the size of G. This would yield a polynomial time transformation of logical consequence into validity. □

These negative results can be extended to the decision procedures for SST and functional/mixed translation methods:

COROLLARY 9.3. *For logical consequence $G \models_{\mathsf{L}} U \Rightarrow A$ in the logics* K, KD, T, KB, KDB, *and* B *the value of hb_{L} cannot be bounded by any polynomial in the size of G, U and A.*

If a polynomial upper bound hb_{L} existed, we could set $n = hb_{\mathsf{L}}$ in the modal deduction theorem (1).

The situation is better for K4, KD4, and S4.

THEOREM 9.4. *For global logical consequence $G \models_{\mathsf{L}} U \Rightarrow A$ in logics* K4 *and* S4 *every strategy that applies technique* T.8.3 *with $hb_{\mathsf{L}} = 2 + (d_p + p_G) + (p + p_G) \times (n + n_G + f_G)$ terminates in one of the following conditions:*

(1) *a tableaux proof has been found, or*
(2) *in every branch some rule is still applicable, or*
(3) *at least one branch can be pruned into a π-modal-completed branch.*

Proof. Logics including K4 satisfy a particular form of the deduction theorem [25, 13, 15]: $G \models_{\mathsf{L}} U \Rightarrow A$ if and only if $\models_{\mathsf{L}} G \cup \Box G \cup U \Rightarrow A$. Apply Theorem 8.4 to the new set $U' = G \cup \Box G \cup U$. □

9.2. NPTIME-SEARCH STRATEGIES FOR K45 AND S5 WITH SST

From the preliminary facts we have recalled, proof search for logics like K45 and S5 should be much simpler than for other logics.

Yet, the presence of symmetry makes things harder for most direct methods, either Gentzen-type tableaux or modal resolution. Cut is necessary when faced with local logical consequence for arbitrary formulae. The prototypical example is $\models_{\mathsf{S5}} \{A, B\} \Rightarrow \Box\Diamond(A \wedge B)$. The common solution [9, 25, 40, 41] is to use an equivalence-preserving preprocessing step that reduces the depth of the modal connectives. Then cut-free calculi for the reduced formula can be found. The problem is that the preprocessing step leads to an exponential blow-up. This can be avoided

by using a further preprocessing with the definitional translation into modal CNF by Mints [35]. The immediacy of the translation is now lost, and this procedure requires at least a quadratic increase in size.

An alternative approach is the translation into S4 [14, 8]. This means translating a simpler problem into a harder one.

For translation methods, the main result is in [47]. It gives a linear upper bound on the multiplicities* of every first-order variable.

In Section 4 we proposed two sets of rules for each of K45, KD45, and S5. A simple technique to restrict rule π yields the NPTIME-bound.

Technique 9.1. With the calculus using rule (4) for K45, KD45, and S5, apply rule (π) once to each π-formula $\neg\Box A$ (no matter its prefix).

Technique 9.2. With the calculus using rule (4^{π}) for K45, KD45, and S5, apply rule (π) only to the prefix $\sigma = 1$.

Each technique gives a decision procedure when combined with technique T. 4.1 and provides an equivalent of Theorem. 8.4. We can also weaken the condition on modal completeness: in a *π-weakly-modal-completed* branch the modal copy σ_0 of Definition 8.2 may be longer than σ and may have more ν-formulae than σ.

LEMMA 9.5. *For logical consequence $G \models U \Rightarrow A$ in logics* K45, KD45, *and* S5 *every strategy that applies technique* T.4.1 *and* T.9.1 (T.9.2) *terminates in one of the following conditions:*

(1) *a tableaux proof has been found, or*
(2) *in every branch some rule is still applicable, or*
(3) *at least a branch is π-weakly-modal-completed branch.*

Proof. Reduction wrt rules (4) and (4^R) forces all prefixes to have the same ν-formulae. Every prefix is a modal copy of all other prefix.

For the calculus with rule (4^{π}), all prefixes are reduced w.r.t. rule (4^R), and hence the prefix 1 contains all ν-formulae and all π-formulae. □

We can also show that this is optimal.

THEOREM 9.6. *For logical consequence $G \models U \Rightarrow A$ in logics* K45, KD45, *and* S5 *every strategy with technique* T.9.1 (T.9.2) *generates SST branches with size polynomial in the size of G, U and A.*

Proof. We can generate at most $p + p_G + 1$ different prefixes, which have length at most $1 + p + p_G + 1$. Each prefix can have a number of formulae equal at most

* Loosely speaking, the number of times we need to instantiate each variable that is generated by the translation of modal formulae into first-order ones.

to the number of subformulae of G, U, and A, and this number is linearly bounded by their respective sizes. The theorem follows with a simple multiplication. □

Notice that the number of worlds is *linearly* bounded by the number of π-formulae. This corresponds to the lowest possible bound.

We can extend this technique to translation methods.

Technique 9.3. Let L be one of the logics K45, KD45, and S5; the first-order terms corresponding to worlds must contain at most one occurrence of each Skolem function/constant corresponding to a π-formula.

In Figure 5, we can now stop after the generation of one resolvent. Resolution with technique 9.3 is a decision procedure for K45, KD45, S5.

9.3. SPACE COMPLEXITY OF DIFFERENT CALCULI

The case for the logics K, KD, T, K4, KD4, and S4 is more interesting because deciding validity is PSPACE-complete and therefore we may have formulae only with "large" models.

Gentzen-type calculi require only polynomial space if we use loop checking with backtracking as in [8, 30, 22]. This works if the calculus actually *deletes* the formulae before backtracking to a previous stage. Loosely speaking, we can describe this depth-first search as follows:

(1) start from a set of formulae;
(2) reduce propositional connectives (possibly creating new branches);
(3) if we get an inconsistent set, then stop and pass to a new branch;
(4) choose a π-formula, and save the remaining choices in the stack;
(5) reduce the π-formula, generating a new starting set;
(6) if the same set was already generated, then delete all sets generated after the latest choice point and backtrack; otherwise go to (1).

This strategy guarantees the use of polynomial space but has its disadvantages. The worst problem is the reduction of disjunctions before π-formulae to retain completeness (see Section 7). In practice, this forces a lot of (likely pointless) case analysis [26].

More space, but still a polynomial amount, is required by the calculus in [23]. Indeed the major difference is that the loop-checking mechanism is compiled into the rules. We can now check for loops without looking back in the stack. However, we are simply encoding a part of the stack in each node, so nodes are bigger.

Remark. The usage of polynomial space is not a characteristic of Gentzen systems. It is simply the fact that we are used to consider depth-first search as the "natural" way to search for a proof in Gentzen-type calculi. If we use a breadth-first search for visiting choice points of π-formulae, then we need exponential space.

Tableau methods that use an explicit accessibility relation [3, 29] are bound to take exponential space: for the "bad formulae" they create the exponential models (Fact 9.2).

For the same reasons, *relational translation methods* based on resolution or similar saturation methods [18] require exponential space. The key observation is that the completeness of first-order resolution forces the addition of (nonsubsumed) resolvents to the current set of clauses. If no contradiction is found, the final set of clauses describes the accessibility relation between worlds of the (counter)model. This model may have exponential size (see Fact 9.2 above).

Functional translation methods are slightly better, since the accessibility relation between worlds is embedded in the unification theory. Therefore, we do not need to store (and not even generate) the clauses describing the accessibility relation. This is also true for the mixed translation method [36] where all literals describing the accessibility relation can be resolved away.

Yet, we have the literals describing properties of worlds (recall that a term is just a name for a world). Again, the saturated set of clauses for satisfiable formulae must describe all these worlds and these may be in exponential number. The conclusion is the following.

Fact 9.5. Any first-order complete resolution strategy for relational, functional, and mixed translation methods requires worst-case exponential space.

This may seem strange given the experimental results in [26] where resolution outperforms tableaux. The caveat is that we may be comparing the efficiency of implementations or using benchmarks that are not PSPACE-hard.

It is, of course, possible to recover polynomial space by ad hoc mechanisms for deduction in modal logics. At a guess, this may be done by a combination of specialized rules for deleting (not-subsumed) clauses and selection functions for the next resolvent. In such a way we could mimic the search strategy of Gentzen-type tableaux.

Yet, this would be the very negation of the motivation behind translation methods [37, page 513]: "there is no need to develop specialized theorem provers for modal logic."

Modal resolution [9] also requires exponential space.

The worst-case complexity of the matrix proof method [47] is unclear. An exponential upper bound for the calculus has been found in [4] but it is not possible to derive (straightforward) conclusions regarding its worst-case space requirements.

The algorithms for *prefixed* (*single step*) *tableaux* in [13, 20, 32] require exponential space. This is not by chance: being a saturation method, prefixed tableaux suffer from the same illness of resolution.

Fact 9.6. Any complete strategy for prefixed (single step) tableaux that never deletes formulae requires worst-case exponential space.

9.4. PSPACE-SEARCH STRATEGIES WITH SST

To recover a PSPACE algorithm for validity, we need two techniques:

- delete formulae as soon as they are no longer needed;
- do not generate new prefixes if too many formulae are unreduced.

We know that this is possible: we can always fall back on a depth-first traversal of the tree of prefixes and simulate the depth-first strategy of Gentzen-type tableaux. We would like to exploit proof confluence for a more flexible strategy.

At first, we focus on the logics between K and S4, and then extend the techniques to logics KB, KDB, B. As in Section 8 we assume that formulae are in negation normal form. Some preliminary notions are also needed: the *backward tree* of a prefix in branch $\mathcal{B}$ is the set

$$\text{Btree}(\sigma) = \{\sigma_0 : A \mid \sigma_0 : A \in \mathcal{B} \text{ and } \sigma_0 \text{ is an initial subsequence of } \sigma\}.$$

We need to introduce the concept of *confined formula*.

Intuition. A confined formula with prefix σ does not interfere with the consistency of formulae having different prefixes. So we can delete the formulae of a confined prefix as soon as they are saturated.

Let $\mathcal{B}'$ be a set of prefixed formulae. A prefixed formula $\sigma : \Diamond A \in \mathcal{B}'$ is *confined* in $\mathcal{B}'$. Prefixed formulae $\sigma : A \wedge B$ and $\sigma : A \vee B$ are confined in $\mathcal{B}'$ if they are reduced in $\mathcal{B}'$ (Definition 4.3) or no formula $\Box C$ is a subformula of either A or B. The prefixed formula $\sigma : \Box A$ is confined in $\mathcal{B}'$ iff it is reduced in $\mathcal{B}'$. A formula $A \in U$ is confined in $\mathcal{B}'$ iff $1 : A$ is confined in $\mathcal{B}'$.

DEFINITION 9.1. A prefix is *confined* in a branch $\mathcal{B}$ if all formulae in U and all prefixed formulae in $\text{Btree}(\sigma)$ are confined in $\text{Btree}(\sigma)$.

LEMMA 9.7 (Local Stability). *Let σ be a confined prefix in $\mathcal{B}$. No sequence of rules (α), (β), (Loc), (D), (K), (T), (4), (4^D) applied to formulae with prefixes different from σ can introduce a new prefixed formula $\sigma : A$ in $\mathcal{B}$.*

Proof. The proof is by a double induction on the length of the prefix σ and the length of the sequence. The case for prefix 1 is immediate.

For the first induction, suppose that $\sigma.n$ is confined but there is a sequence of rules that introduces a new prefixed formula $\sigma.n : A$.

The sequence of length 1 can be composed only by a ν-rule. Then, there should be an unreduced $\sigma : \Box A'$ in $\mathcal{B}$. This formula is also in $\text{Btree}(\sigma.n)$, and therefore $\sigma.n$ could not be confined.

For longer sequences, we apply the induction hypothesis to σ: if σ is confined, no sequence of rules applied to prefixes different from σ can introduce a new prefixed $\sigma : \Box A$ in $\mathcal{B}$. Only α and β rules are left, and the corresponding formulae are confined by hypothesis. □

We use this lemma for proving that confined prefixes are preserved by the techniques we have used for decidability.

PROPOSITION 9.1. *A strategy using techniques* T.4.1 *and* T.8.3 *cannot introduce a new prefixed formula* $\sigma : A$ *in a branch* $\mathcal{B}$ *if* σ *is a confined and reduced prefix in* $\mathcal{B}$.

Now, choose a parameter $wb \geq 1$ (width bound) equal to some fixed constant* and apply the following technique.

Technique 9.4. For every integer n and every prefix σ of length n, do not apply rule (π) to prefixed formulae $\sigma : \Diamond A$ if there are more than wb unreduced prefixes of length $n + 1$.

The next step is proving that the unreduced part of each branch has only a polynomial size.

THEOREM 9.8. *For the logics* K, KD, T, K4, KD4, *and* S4 *every strategy using techniques* T.4.1, T.8.3, *and* T.9.4 *the number of unreduced prefixes in a branch is bounded by a polynomial in the size of* U *and* A.

Proof. By technique T.4.1 the number of (unreduced) prefixed formulae is linearly bounded by the number of subformulae of U and A. By technique T.8.3 the maximum length of prefixes is hb_{L}, which is polynomially bounded in the size of U and A (Thm. 8.4). By technique T.9.4 we have at most wb unreduced prefixes of length n for each $n \geq 1$. The claim follows by a multiplication. □

The polynomial space algorithm we are looking for must use this additional technique (beyond T.4.1, T.8.3, T.9.4) for each branch $\mathcal{B}$.

Technique 9.5. Delete all prefixed formulae of every confined and reduced prefix in $\mathcal{B}$. If a prefixed formula $\sigma.n : A$ is deleted in this process, then consider the corresponding $\sigma : \Diamond A$ as still reduced.

We may keep deleted prefixes somewhere to exhibit the model, but then the auxiliary storage is bound to take exponential space (Fact 9.2).

The combination of these techniques preserves soundness (using Th. 9.1 and 7.1) and completeness (Th. 7.1, 8.1, 8.4), provides a decision procedure (Th. 8.4) and guarantees the use of polynomial space in each branch (Theorem 9.8) — this without dropping, but rather exploiting, proof confluence.

We are free to reduce a polynomial number of $\Diamond A$ formulae before reducing a single disjunctive β-formula. Still we do not lose completeness nor the polynomial space bound for the size of a branch.

* In general, we could also use a function $wb \leq poly(|U|, |A|)$.

Moreover, technique T.9.5 is so liberal that the part of the branch we keep in memory may resemble a "tree with gaps". For example, the part of the branch we keep in memory can be composed only by the prefixes 1, 1.1.1, 1.1.2, and 1.1.2.1, because the prefix 1.1 has been deleted. Theorem 9.1 guarantees that such gaps are harmless.

For logics with rule (B) these techniques are not sufficient. To be precise, reckless deletion of prefixes may lead to incomplete search strategies, since $\sigma.n : A$ can be introduce by $\sigma : \Box A$ *and* by $\sigma.n.m : \Box A$. To prove that a prefix can be safely deleted, it is not enough to look only to shorter prefixes; we must also look to longer ones.

At first we need to revise the technique 9.5.

Technique 9.6. Delete all prefixed formulae of every confined and reduced prefix σ in $\mathcal{B}$ such that for every $\sigma.n$ the forward tree Ftree$(\sigma.n)$ is empty (has been previously deleted).

Second, use the following trick.

Technique 9.7. If a new prefixed formula $\sigma : \Box A$ is introduced in $\mathcal{B}$, then for every $\sigma : \Diamond B$ if the corresponding $\sigma.n : B$ has been deleted, then $\sigma : \Diamond B$ must be reduced again.

10. Soundness and Completeness for SST

As a preliminary result we can prove that SST satisfy a proof theoretical analogue of the Generation theorem by Segerberg [15].

DEFINITION 10.1. A set of prefixes Σ is *tree generated* iff

(1) the prefix $1 \in \Sigma$ and it is the only prefix of length 1;
(2) if prefix $n_0.\ldots.n_{i-1}.n_i \in \Sigma$ then also $n_0.\ldots.n_{i-1} \in \Sigma$.

By induction on the rules used to construct a branch, one has the following.

THEOREM 10.1. *Let $\mathcal{B}$ be a branch of a single step tableau; the set of prefixes occurring in the branch $\{\sigma \mid \sigma : A \in \mathcal{B}\}$ is tree generated.*

This explains why the π-rule requires prefixes to be new rather than unrestricted as in [13]. Suppose that $\sigma.n$ is new and yet it is the initial part of another prefix $\sigma.n.n_1 \ldots n_k$. By Theorem 10.1, $\sigma.n$ must also be present, contradiction.

Table V. SST-rules and model properties

Rule form	Semantic property
$\sigma : A \Rightarrow \sigma : B$	$w \Vdash A$ implies $w \Vdash B$
$\sigma : A \Rightarrow \sigma.n : B$	$w\mathcal{R}v$ and $w \Vdash A$ implies $v \Vdash B$
$\sigma.n : A \Rightarrow \sigma : B$	$w\mathcal{R}v$ and $v \Vdash A$ implies $w \Vdash B$
$\sigma.m : A \Rightarrow \sigma.m : B$	$\exists w_0.w_0\mathcal{R}w$ and $w \Vdash A$ implies $w \Vdash B$
$\sigma.m : A \Rightarrow \sigma.m.n : B$	$\exists w_0.w_0\mathcal{R}w$ and $w\mathcal{R}v$ and $w \Vdash A$ implies $v \Vdash B$
$\sigma.m.n : A \Rightarrow \sigma.m : B$	$\exists w_0.w_0\mathcal{R}w$ and $w\mathcal{R}v$ and $v \Vdash A$ implies $w \Vdash B$

All variables except for w_0 are universally quantified.

10.1. STRONG SOUNDNESS

To prove the correctness of prefixed and labeled systems [13, 16], one establishes a mapping between "names" (prefixes) and "things" (possible worlds) and shows that the mapping is indeed an homomorphism preserved by tableau rules. To prove the soundness of Gentzen-type tableaux [13, 20, 22], one shows that if a set of formulae holds in a world, then a related set holds in a world one step away.

Intuition. The soundness proof of SST is a combination of both techniques: set a mapping between prefixes and worlds, do not worry about homomorphisms, and use the stepwise proof of Gentzen-type tableaux.

We need to establish some properties of the logics at hand, in a natural correspondence with its SST rules.

We focus on ν rules (the others can be found in [13]) and divide them into three main classes, according the form of the prefixes that is used in the antecedent or in the consequent of each rule. In *static rules* both the consequent and the antecedent formula have the same prefix. For example, rules (D) and (T) are static. In *forward rules* the prefix of the consequent formula is one step longer than the prefix of the antecedent. Rules (K), (4^D), and (4) are examples of forward rules. In *backward rules* the prefix of the consequent formula is one step shorter than the prefix of the antecedent. Rules (4^R), (B), or (Cxt) are backward rules.

Rules can be *delayed*. This classification is orthogonal to the previous one. It refers to rules applicable only to prefixes longer than two.

Table V identifies the correspondence between rules and properties of L-models needed for the proof (see appendix) of the following lemma.

LEMMA 10.2. *Let $\sigma : A \Rightarrow \sigma^* : B$ be an SST rule for logic* L*; every* L*-model verifies the conditions in Table* V.

The next step is defining a mapping between names and things and introduce the notion of satisfiable branch.

DEFINITION 10.2. Let $\mathcal{B}$ be a branch and $\langle W, \mathcal{R}, V\rangle$ an L-model; an *SST-interpretation* is a mapping from prefixes to worlds $\iota(\sigma) \in W$ such that for all σ and $\sigma.n$ present on $\mathcal{B}$, one has $\iota(\sigma)\mathcal{R}\iota(\sigma.n)$.

SST interpretations do not depend on the logic L, in contrast with prefixed tableaux interpretation [13], which are logic dependent. Loosely speaking, SST interpretations are just K-interpretations.

DEFINITION 10.3. A tableau branch $\mathcal{B}$ with local assumptions U and global assumptions G is L-SAT iff there is an L-model $\langle W, \mathcal{R}, V\rangle$ and an SST-interpretation $\iota()$ such that

(1) the L-model $\langle W, \mathcal{R}, V\rangle$ validates G;
(2) the world $\iota(1)$ satisfies U in $\langle W, \mathcal{R}, V\rangle$;
(3) for every prefixed formula $\sigma : A$ in $\mathcal{B}$, the world $\iota(\sigma)$ satisfies A.

A tableau is L-SAT if at least one branch is such.

Now we have the machinery to prove a *safe extension lemma*.

THEOREM 10.3. *Let $\mathcal{T}$ be an* L*-SAT tableau; the tableau $\mathcal{T}'$ obtained by an application of an SST rule for* L *is also* L*-SAT.*

Proof. If $\mathcal{T}$ is L-SAT, there is a branch $\mathcal{B}$ that is L-SAT with an SST-interpretation $\iota()$ and a model $\langle W, \mathcal{R}, V\rangle$. The cases where $\mathcal{T}'$ has been obtained by applying a rule to a branch different from $\mathcal{B}$, or the rule applied to a prefixed formula in $\mathcal{B}$ is an α, β, Loc or $Glob$ rule are standard [13]. The different cases are ν-rules and the π-rule.

For SST-rules characterizing L, consider whether they are static, backward, or forward. If a *static rule* has been applied, then the prefix σ is present in $\mathcal{B}$ and $\iota()$ is defined on it. By inductive hypothesis $\iota(\sigma) \Vdash A$ and by Lemma 10.2 we are done. *Forward and backward rules* involve prefixes of the form σ and $\sigma.n$ both already present on the branch. By Definition 10.2 one has $\iota(\sigma)\mathcal{R}\iota(\sigma n)$, and the claim follows by Lemma 10.2.

Delayed rules require an "ancestor" world w_0: by Theorem 10.1, if $\sigma.n$ and $\sigma.n.m$ are in the branch, then σ is there too. Hence $\iota()$ is defined on all three prefixes and by inductive hypothesis $\iota(\sigma)\mathcal{R}\iota(\sigma.n)$ and $\iota(\sigma.n)\mathcal{R}\iota(\sigma.n.m)$. Apply Lemma 10.2.

The π-rule introduces a new prefix, and $\iota()$ must be extended. By hypothesis the original branch $\mathcal{B}$ is L-SAT and thus $\iota(\sigma) \Vdash \neg\Box A$. The semantics forces the existence of a world $w \in W$ such that $\iota(\sigma)\mathcal{R}w$ and $w \Vdash \neg A$. Then extend $\iota()$ as follows:

$$\jmath(s) = \begin{cases} w & \text{if } s = \sigma.n, \\ \iota(s) & \text{otherwise.} \end{cases}$$

Table VI. Conditions on the syntactic relation $\rhd$

Logic	Conditions on the accessibility relation over prefixes $\sigma \rhd \sigma^*$
K	$\sigma \rhd \sigma.n$
KB	$\sigma \rhd \sigma.n$ and $\sigma.n \rhd \sigma$
K4	$\sigma \rhd \sigma.\sigma'$ with $\|\sigma'\| \geq 1$
K5	$\sigma \rhd \sigma.n$ and $\sigma.\sigma' \rhd \sigma.\sigma''$ with $\|\sigma'\| \geq 1$ and $\|\sigma''\| \geq 1$
K45	$\sigma.\sigma' \rhd \sigma.\sigma''$ with $\|\sigma''\| \geq 1$
KD-logics	as K-logics provided that either there is an n such that $\sigma \rhd \sigma.n$ or $\sigma \rhd \sigma$
T	$\sigma \rhd \sigma.n$ and $\sigma \rhd \sigma$
B	$\sigma \rhd \sigma.n$ and $\sigma.n \rhd \sigma$ and $\sigma \rhd \sigma$
S4	$\sigma \rhd \sigma.\sigma'$
S5	$\sigma.\sigma' \rhd \sigma.\sigma''$
Cxt	$\|\sigma^*\| = \|\sigma\| + 1$

We must prove that $\jmath()$ is an SST-interpretation. On every prefix s different from $\sigma.n$ the mapping $\jmath(s)$ coincides with $\iota(s)$, which is an SST-interpretation. Moreover, $\jmath(\sigma)\mathcal{R}\jmath(\sigma.n)$ by construction. Since $\sigma.n$ is new, there is no prefix $\sigma.n.m$ in the branch that requires us to prove $\jmath(\sigma.n)\mathcal{R}\jmath(\sigma.n.m)$. The branch is L-SAT on the same L-model with SST-interpretation $\jmath()$. □

The soundness theorem (Theorem 4.1) is now standard [13, 20].

10.2. STRONG COMPLETENESS

The completeness proof follows the general ideas of [13] adapted to "shorter rules". For simplicity, we give the proof using completed branches (Section 4). The extension to π-(modal)-completed branches (Section 8) can be done along the same lines of the completeness proofs in [7] for prefixed tableaux or in [20] for completeness via model graphs.

Intuition. Apply a systematic strategy to the tableau and use an open branch to build a model for the initial formula $\neg A$, that is, a countermodel for A. Identify prefixes present in the branch with worlds (so that $\iota()$ is the identity function), and show that if $\sigma : A$ occurs in the branch, then also $\sigma \Vdash A$.

The "easy" part is the construction of a syntactic relation $\rhd$ between prefixes with the properties of the semantic relation $\mathcal{R}$ in L-models. Each logic L has its relation $\rhd_{\mathsf{L}}$ as shown in Table VI.

THEOREM 10.4. *For every logic* L*, the syntactic relation* $\rhd$ *over prefixes has the same properties of the semantic relation* $\mathcal{R}$ *over worlds.*

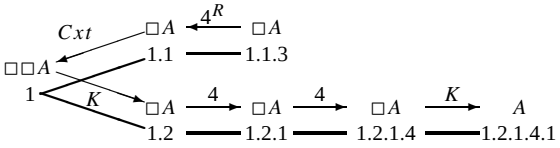

Figure 6. Traveling throughout prefixes in K5.

The proof is long but easy: replace possible worlds with prefixes in the properties of $\mathcal{R}$ (Table I), use for $\rhd$ the conditions of Table VI, and reason by cases* (see appendix).

The "hard" part is the proof of *complete reduction*; that is, if $\sigma : \Box A$ is in the branch and $\sigma \rhd \sigma^*$, then $\sigma^* : A$ is also there.

Intuition. Prefixed tableaux [13, 42] use $\rhd$ in the calculus and have only one ν-rule: if $\sigma : \Box A$ is present and $\sigma \rhd \sigma^*$ with σ^* also present, then add $\sigma^* : A$. The proof procedure itself grants complete reduction.

However, complete reduction is too powerful because a set can be completely reduced without being the result of a tableau proof; for example, $\{1 : \Box p, 1.1.1.1 : p\}$ is completely reduced for K4 but doesn't correspond to any tableaux proof.

Intuition. SST can recover complete reduction from SST-reductions by letting a formula "travel" along prefixes.

A simple example is shown in Figure 6 for K5, where $\{1.1.3.1\} \rhd_{\mathsf{K5}} \{1.2.1.4.1\}$ and thus we should have $\{1.1.3.1 : \Box A\}$ implies $\{1.2.1.4.1 : A\}$.

We classify each logic according the "ability" of its syntactic accessibility relation $\rhd$. The relation $\rhd$ connects

immediate neighbors if $\rhd$ connects prefixes that are one integer shorter or longer (logics K, KB, and their serial, or reflexive variants);

far forward prefixes where the relation $\rhd$ can link a prefix σ with a another prefix σ^* that can be longer than σ but still σ must be an initial subsequence of σ^* (logic K4 or its variants);

far backward prefixes $\rhd$ can link a long prefix σ with a shorter σ^* such that σ^* is the initial subsequence of σ (logic K5 or its variants).

A logic can also satisfy a combination of these properties, for example, K45.

Hence, we need to prove a *complete reduction lemma* (see the appendix):

LEMMA 10.5. *Let $\mathcal{B}$ be a branch of a tableau, if $\mathcal{B}$ is completed for the logic* L, *then it is also completely reduced.*

* Note that the relation $\rhd$ for K5 in [32] is incorrect and thus the original system is incomplete. See also [20] for a detailed proof of K5 completeness.

Proof. If the syntactic relation $\triangleright$ can access only *close neighbors*, then complete reduction coincides with SST reduction.

If $\triangleright$ accesses *forward prefixes*, use rule (4) for copying $\Box A$ from σ to $\sigma.n$ and forward. Repeat until we arrive at the immediate predecessor of σ^*. Then apply rule (K).

For logics whose syntactic relation accesses *backward prefixes*, use rule (4^R) for copying $\Box A$ from $\sigma.n$ to σ and backward. Repeat the process until we are at the immediate predecessor of σ^* and then apply (K).

Other logics combine these techniques; for example, for Cxt, apply (Cxt) down to 1 and (K) up to σ^*. □

A further requirement is that $\mathcal{B}$ should be an open branch, that is, for every σ and every A, there is no pair $\sigma : A$ and $\sigma : \neg A$ in $\mathcal{B}$. Then we can prove a *strong model existence theorem*.

THEOREM 10.6. *If $\mathcal{B}$ is open branch for the rules of logic* L, *then there is an* L*-model for G and U where $\mathcal{B}$ is satisfiable.*

Proof. Construct the model as follows:

$$\begin{array}{rcl} W & \doteq & \{\sigma \ : \ \sigma \text{ is present in } \mathcal{B}\}, \\ \sigma \mathcal{R} \sigma^* & \text{iff} & \sigma \rhd \sigma^*, \\ V(p) & \doteq & \{\sigma \mid \sigma : p \in \mathcal{B}\}. \end{array}$$

If L is serial, extend $\mathcal{R}$ by setting $\sigma \mathcal{R} \sigma$ if there are no formula of the form $\sigma : \neg\Box A$ in $\mathcal{B}$. This guarantees that $\sigma : \Box B \notin \mathcal{B}$. Otherwise, since σ is reduced for (D), we would have had $\sigma : \neg\Box\neg A \in \mathcal{B}$, contradiction.

By Theorem 10.4, $\mathcal{R}$ satisfies the properties of L-accessibility relations. We must show that $\mathcal{B}$ is L-SAT on model $\langle W, \mathcal{R}, V \rangle$ with an SST-interpretation $\iota()$ (here $\iota()$ is the identity function): for every A we must prove that if $\sigma : A \in \mathcal{B}$, then $\sigma \Vdash A$ by induction on the construction of A.

We focus on modal connectives. If $\sigma : \neg\Box A \in \mathcal{B}$, we have $\sigma.n : \neg A \in \mathcal{B}$ for some $\sigma.n$, since $\sigma : \neg\Box A$ must be reduced for rule (π). By inductive hypothesis $\sigma.n \Vdash \neg A$ and $\sigma \mathcal{R} \sigma.n$ by construction. Therefore $\sigma \Vdash \neg\Box A$. If $\sigma : \Box A \in \mathcal{B}$, then, by Lemma 10.5, for every σ^* present in $\mathcal{B}$ such that $\sigma \rhd \sigma^*$, one has $\sigma^* : A \in \mathcal{B}$. By inductive hypothesis $\sigma^* \Vdash A$ and $\sigma \mathcal{R} \sigma^*$ by construction. Therefore $\sigma \Vdash \Box A$.

For every $A \in G$ and every σ in $\mathcal{B}$, the prefixed formula $\sigma : A$ is present in the branch $\mathcal{B}$ because σ is reduced for the $(Glob)$-rule in $\mathcal{B}$. Then $\sigma \Vdash A$. With the same argument, if $A \in U$, then $1 \Vdash A$. □

The *strong completeness theorem* (Theorem 4.2) is now standard [13].

11. Conclusions

As pointed out by Catach [3, page 508]:

> ... the most interesting features of modal logics are the *expressivity* and *modularity* and also their *possible-world semantics* which is both general and intuitive.

From this viewpoint Single Step Tableaux provide a calculus that reflects these characteristics: modularity, simplicity, and intuitive rules that "pass" knowledge (or necessity) between worlds.

A further advantage is their effectiveness. SST are proof confluent and can be transformed into decision procedures that use polynomial space (nondeterministic time for K45 and S5). The use of SST makes also possible to derive simple bounds and termination checks for translation methods based on resolution [36, 39].

Given their modularity, as pointed out in [20], SST can be extended to multi-modal logics and, with more work, to dynamic logics [7].

In the quest for effective implementations, a version of SST based on free variables, as proposed in [1], which simulates the techniques for yielding PSPACE decision methods of Sections 8 and 9, may be one of the most effective modal provers. We leave this open for future investigations.

Appendix

A. Terminology about Complexity Classes

PTIME(NPTIME) is the class of decision problems solvable in polynomial time by a deterministic (nondeterministic) Turing machine. For instance, satisfiability for the propositional calculus [27] and the modal logics S5 and K45 is NPTIME-complete [22, 30] together with the logic Cxt which remains in NPTIME also for the multi-modal case [33].

Problems in EXPTIME can be solved by a deterministic Turing machine using exponential time, that is, time bounded by $O(2^{poly(n)})$, where $poly(n)$ is a polynomial in the size n of the input. Deciding logical consequence for K_m and for $\mathsf{S4}_m$ for $m \geq 2$ is EXPTIME-complete; validity for propositional dynamic logic is EXPTIME-complete [22].

As for PSPACE and EXPSPACE, the machine is deterministic and works in polynomial (exponential) space. For instance, deciding satisfiability and validity for all modal logics between K and Section four and for all multimodal logics between K_m and $\mathsf{S5}_m$ with $m \geq 2$ is PSPACE-complete [30, 22].

It is known that PTIME $\subseteq$ NPTIME $\subseteq$ PSPACE $\subseteq$ EXPTIME $\subseteq$ EXPSPACE. All containment relations are conjectured strict [27].

Table VII. Reduction rules for SST

Reduction precondition	Reduction relation
$\sigma : A \wedge B \in \mathcal{B}$	$\mathcal{B} \xrightarrow{\alpha} \mathcal{B} \cup \{\sigma : A, \sigma : B\}$
$\sigma : \neg(A \wedge B) \in \mathcal{B}$	$\mathcal{B} \xrightarrow{\beta} \mathcal{B} \cup \{\sigma : \neg A\} \mid \mathcal{B} \cup \{\sigma : \neg B\}$
$\sigma : \Box A \in \mathcal{B}$ and $\exists C\ \sigma.n : C \in \mathcal{B}$	$\mathcal{B} \xrightarrow{K} \mathcal{B} \cup \{\sigma.n : A\}$
$\sigma : \neg\Box A \in \mathcal{B}$ and $\forall C\ \sigma.n : C \notin \mathcal{B}$	$\mathcal{B} \xrightarrow{\pi} \mathcal{B} \cup \{\sigma.n : \neg A\}$

B. Proofs

B.1. PROOFS OF SECTION 7

Proof (Theorem 7.1). We show a stronger claim, namely

$$\begin{array}{lll} \forall x, x' yz & \text{if} & x \sim x' \ \&\ x \to y \ \&\ x' \to z \\ & \text{then} & \exists u, u_e\ y \to u \ \&\ z \to u_e \ \&\ u \sim u_e. \end{array}$$

In words, if we apply a rule to the tableau x yielding y or another rule yielding z, then we can apply another rule to y to obtain u and similarly to z obtaining u_e. The two possible results u and u_e must be identical up to renaming of prefixes.

The standard Knuth–Bendix method for proving confluence [24, page 797] is to prove that each critical pair satisfies the above mentioned claim. For SST, a critical pair can only be formed when we reduce two formulae in the same branch, since reductions in different branches do not interact.⋆

For clarity of exposition, we consider first the basic logic K and the case where $x = x'$. This is also the most difficult case.

The cases of superpositions can be seen more easily if we reformulate the rules as in Table VII, where $\mathcal{B}$ is a branch (set of prefixed formulae).

At first, formulae with different prefixes do not interact.

PROPOSITION B.1. *Let $\sigma_1 : A_1$ and $\sigma_2 : A_2$ be prefixed formulae with $\sigma_1 \neq \sigma_2$ and let $\mathcal{B}'$ be the reduction of $\mathcal{B}$ using rule (r_1) on $\sigma_1 : A_1$. If rule (r_2) can be applied to $\sigma_2 : A_2$ in $\mathcal{B}$, then it can be applied in $\mathcal{B}'$.*

This reduces the cases of superposition to the following four:

(1) the prefixed formulae $\sigma : \Box A$, $\sigma.n : C$ and $\sigma.m : D$ are in $\mathcal{B}$;
(2) the prefixed formulae $\sigma : \Box A$, $\sigma : \Box B$ and $\sigma.n : C$ are in $\mathcal{B}$;
(3) the prefixed formulae $\sigma : \Box A$, $\sigma : \neg\Box B$ and $\sigma.n : C$ are in $\mathcal{B}$ but no prefixed formula $\sigma.m : D$ is present in $\mathcal{B}$ ($\sigma.m$ is new);

⋆ This is quite different from free variables tableaux where variables span over branches and therefore the reduction (unification) in a branch affects other branches.

(4) the prefixed formula $\sigma : \neg\Box A$ and $\sigma : \neg\Box B$ are in $\mathcal{B}$ and the prefixes $\sigma.n$ and $\sigma.m$ are new.

For case (1) we apply rule (K) either for $\sigma.n$ or for $\sigma.m$. Suppose we apply first (K) to $\sigma.n$ yielding $\mathcal{B} \to \mathcal{B} \cup \{\sigma.n : A\}$. Since $\sigma.m : C$ is still present in the new branch and ν formulae can be reused, we apply (K) and obtain $\mathcal{B} \cup \{\sigma.n : A, \sigma.m : A\}$. If we apply (K) first to $\sigma.m$ and then to $\sigma.n$, we obtain $\mathcal{B} \cup \{\sigma.m : A.\sigma.n : A\}$. The result is the same.

For case (2), use the same argument. The final outcome of both reduction paths is equal to $\mathcal{B} \cup \{\sigma.n : A, \sigma.n : B\}$.

For (3) we can use rule (K) on $\sigma.n$ or introduce a new prefix with (π). If we reduce the branch using rule (K), we do not introduce new prefixes and $\sigma.m$ would still be new in $\mathcal{B} \cup \{\sigma.n : A\}$. In other words, if for no C we have $\sigma.m : C \in \mathcal{B}$, then $\sigma.m : C \notin \mathcal{B} \cup \{\sigma.n : A\}$. Thus we use rule (π) and obtain $\mathcal{B} \cup \{\sigma.n : A, \sigma.m : \neg B\}$.

If we apply (π) first, we obtain $\mathcal{B} \cup \{\sigma.m : \neg B\}$. By hypothesis, for all D one has $\sigma.m : D \notin \mathcal{B}$. Since $\sigma.n : C \in \mathcal{B}$, one has $\sigma.m \neq \sigma.n$. We can apply rule (K) obtaining the same branch $\mathcal{B} \cup \{\sigma.n : A, \sigma.m : \neg B\}$.

For case (4) we must use a renaming to prove confluence. Suppose we reduce first $\sigma : \neg\Box A$ and obtain $\mathcal{B} \cup \{\sigma.n_1 : \neg A\}$. In the new branch, the prefix $\sigma.n_1$ is no longer new. So the next reduction forces the use of $\sigma.m_1$. The final result is the branch $\mathcal{B}_1 = \mathcal{B} \cup \{\sigma.n_1 : \neg A,\ \sigma.m_1 : \neg B\}$.

If we reduce $\sigma : \neg\Box B$ with $\sigma.n_2$, we obtain the branch $\mathcal{B} \cup \{\sigma.n_2 : \neg B\}$. A further (π)-reduction yields $\mathcal{B}_2 = \mathcal{B} \cup \{\sigma.n_2 : \neg B,\ \sigma.m_2 : \neg A\}$. At this stage we know only that there are two new prefixes $\sigma.n$ and $\sigma.m$. It can be that $n_1 = n \neq m = n_2$ and $A \neq B$ and thus $\mathcal{B}_1 \neq \mathcal{B}_2$.

Then we define two renamings h_{12} and h_{21}:

$$h_{ij}(s) = \begin{cases} \sigma.n_j & \text{if } s = \sigma.n_i, \\ \sigma.m_j & \text{if } s = \sigma.m_i, \\ s & \text{otherwise.} \end{cases}$$

Now we can prove that $\sigma : A \in \mathcal{B}_i$ implies $h_{ij}(\sigma) : A \in \mathcal{B}_j$, and that $h_{ij}(h_{ji}(\sigma)) = \sigma$.

The proof for logics other than K is a repetition of arguments (1), (2), and (3) for the various ν-rules; just replace the prefixed formulae $\sigma : \Box A$ and $\sigma.n : A$ of rule (K) with the antecedent and the consequent of each ν-rule. Each ν-formula must be reducible more than once because each logic requires more than one ν-rule and all rules must be applicable.

The proof of the general case, $x \sim x'$, follows the same pattern, and it is simply notationally heavy, since we have two branches $\mathcal{B}_1 \sim \mathcal{B}_2$ and two renamings h_{12} (from $\mathcal{B}_1$ to $\mathcal{B}_2$) and h_{21} (in the other direction).

The first step is the following.

PROPOSITION B.2. *A rule (r) can be applied to $\sigma : A$ in $\mathcal{B}_i$ if and only if it can be applied to $h_{ij}(\sigma) : A$ in $\mathcal{B}_j$.*

Second, we reformulate Proposition B.1 as follows.

PROPOSITION B.3. *Let $\sigma_1 : A \in \mathcal{B}_1$ and $\sigma_2 : B \in \mathcal{B}_2$ be prefixed formulae such that $h_{12}(\sigma_1) \neq \sigma_2$. Let $\mathcal{B}_1^r$ be the reduction of $\mathcal{B}_1$ using rule (r) on $\sigma_1 : A$ and $\mathcal{B}_2^r$ be the reduction of $\mathcal{B}_2$ using (r) on $h_{12}(\sigma_1) : A$. If rule (r_2) can be applied to $\sigma_2 : B$ in $\mathcal{B}_2$, then it can be applied in $\mathcal{B}_2^r$.*

The cases of superposition can be reformulated along the same lines. The proof is substantially unchanged. Case (4) is the only difficult one.

For $\mathcal{B}_1$ we have the prefixed formulae $\sigma_1 : \neg\Box A$, $\sigma_1 : \neg\Box B$ and the prefixes $\sigma_1.n_1$ and $\sigma_1.m_1$ are new. The same conditions (changing subscript) hold for $\mathcal{B}_2$. We also have $h_{ij}(\sigma_i) = \sigma_j$. We have no constraints on $\sigma_i.n_i$ and $\sigma_i.m_i$ because they are new.

As in the proof for $x = x'$, renamings must be updated when the branch $\mathcal{B}_1$ reduces to $\mathcal{B}_1 \cup \{\sigma_1.n_1 : \neg A, \sigma_1.m_1 : \neg B\}$ and the branch $\mathcal{B}_2$ reduces to $\mathcal{B}_2 \cup \{\sigma_2.n_2 : \neg A, \sigma_2.m_2 : \neg B\}$.

$$h'_{ij}(s) = \begin{cases} h_{ij}(\sigma_i).n_j & \text{if } s = \sigma_i.n_i, \\ h_{ij}(\sigma_i).m_j & \text{if } s = \sigma_i.m_i, \\ h_{ij}(s) & \text{otherwise.} \end{cases}$$

By hypothesis, one has $h_{ij}(\sigma_i) = \sigma_j$ and the new mappings h'_{ij} gives the desired renamings. □

B.2. PROOFS OF SECTION 8

Proof (Theorem 8.4). If a tableau proof is found, Theorem 4.1 does the job. If some rules are still applicable, then the strategy itself was not systematic or possibly incomplete. In this case continue the reduction of the applicable rules with a systematic strategy. By proof confluence (Theorem 7.1) we must end up in one of the remaining cases.

Finally, if the strategy terminates and no rule is applicable in at least one branch $\mathcal{B}$, then we must prove that the branch $\mathcal{B}$ can be pruned into a π-completed branch (π-modal-completed for K4 and S4).

At first, a π-formula $\sigma : \Diamond A$ may not be reduced according to Definition 8.2 only if $|\sigma| = hb_{\mathsf{L}}$. Indeed, suppose that $\sigma : \Diamond A$ is not reduced but there is no shorter modal copy σ_0 where it is reduced. If $|\sigma| < hb_{\mathsf{L}}$, then rule π would be still applicable according to technique T.8.3 and this contradicts the assumption that no rule is applicable in $\mathcal{B}$.

The case for logics K, D, T, KB, KDB, and B is simple: every rule that increases the length of the prefix strictly reduces the number of modal connectives. So any prefix with length equal to $hb_{\mathsf{L}} = 1 + d$ has no modal connectives. We obtain directly a π-completed branch.

The case for K4 and S4 is more interesting, because rule (4) increases the length of the prefix without decreasing the number of modal connectives. The first observation is that the number of ν-formulae increases monotonically with the length of the prefix:

PROPOSITION 8.1 *If the prefix σ_0 is an initial subsequence of σ in the branch $\mathcal{B}$, then $\sigma_0 : \Box A \in \mathcal{B}$ implies $\sigma : \Box A \in \mathcal{B}$.*

If $\sigma : \Box A$ is not present, then we could apply rule (4) a suitable number of times until $\sigma : \Box A$ is introduced. This is against the hypothesis that no rule is applicable to branch $\mathcal{B}$.

Now, we start pruning the tree: delete from the branch all members of Ftree$(\sigma.n)$ such that for every $\sigma : \Diamond A$ that is fulfilled by $\sigma.n$ there is a shorter modal copy σ_0 where $\sigma_0 : \Diamond A$ is reduced for rule (π).

Denote by $\mathcal{B}'$ the final set of prefixed formulae that cannot be pruned anymore. Proposition 8.1 still holds for $\mathcal{B}'$, and the Pruning Lemma 8.2 guarantees the following property:

PROPOSITION B.4. *Let $\sigma : \Diamond A$ be a prefixed formula in $\mathcal{B}'$. Then either $|\sigma| = hb_{\mathsf{L}}$ or $\sigma : \Diamond A$ is reduced for rule (π) in $\mathcal{B}'$ or there is a shorter modal copy σ_0 of σ such that $\sigma_0 : \Diamond A$ is reduced for rule (π) in $\mathcal{B}'$.*

The only π-formulae that may violate the definition of π-modal-completeness are those with length hb_{L}.

Since $\mathcal{B}'$ cannot be pruned anymore, we have the following.

PROPOSITION B.5. *Let $\sigma : \Diamond A$ be a prefixed formula in $\mathcal{B}'$ that is not reduced for (π). For every σ_0 shorter than σ, either σ_0 is not a modal copy of σ, or $\sigma_0 : \Diamond A$ is not present in $\mathcal{B}'$.*

Then we only have to prove that the longest prefix σ on the $\mathcal{B}'$ that can satisfy Proposition B.5 has length $hb_{\mathsf{L}} - 1 = 1 + d_p + p \times n$. We need the following preliminary results.

PROPOSITION 8.2. *If every prefix from σ_0, $\sigma_0.n_1$ up to $\sigma_0.n_1 \ldots n_k$ is a modal copy of σ_0, then each $\sigma_0 \ldots n_i$ fulfills a different π-formula $\Diamond A$ in $\mathcal{B}'$.*

Otherwise we could apply the pruning lemma to $\sigma.n_1 \ldots n_i$ and delete the whole subtree generated by it. This contradicts the hypothesis that $\mathcal{B}'$ is the final result of the pruning.

Given Propositions 8.1 and 8.2 the worst-case longest sequence of prefixes we can build without violating Proposition B.5 is shown below:

$$\underbrace{1 \rhd 1.n_1 \rhd \rhd 1.n_1 \ldots n_{p-1}}_{0\ \nu\text{–formulae}} \rhd 1 \ldots n_{p-1} n_p \rhd \underbrace{1 \ldots n_p.n_{p+1} \rhd \rhd 1 \ldots n_{2p-1}}_{1\ \nu\text{–formula}} \rhd$$

$$\rhd \underbrace{1 \ldots n_{p+p} \rhd \rhd 1 \ldots n_{p+2p}}_{2\ \nu\text{–formulae}} \rhd \quad \rhd \quad \rhd \underbrace{1. \ldots n_{p+(n-1)p+1} \rhd \rhd 1 \ldots n_{p+np}}_{n\ \nu\text{–formulae}}$$

Consider the first sequence with zero ν formulae: they are clearly modal copies of each other. Each time we pass from $1 \ldots n_i$ to $1 \ldots n_i.n_{i+1}$ there is a different π-formula that is fulfilled (Prop. 8.2). We have at most p different π-subformulae, and we can arrive only till $1 \ldots n_p$ before triggering the pruning lemma. Suppose that no ν-formula is present in $1 \ldots n_p$. Then, for every $1 \ldots n_p : \Diamond A$ there is a shorter modal copy $1 \ldots n_i$ where $1 \ldots n_i : \Diamond A$ is reduced. By Proposition B.5, there is no longer prefix $1 \ldots n_p.n_{p+1}$ in $\mathcal{B}'$.

In the worst case, only one ν-formula will be freshly introduced in $1 \ldots n_p$. This formula will continue to be present from now on. Again, before finding another ν-formula we can arrive at most to $1. \ldots n_{p+p}$.

We can continue this reasoning until we arrive at $1 \ldots n_{p+np}$. This prefix is the longest prefix that respects Proposition B.5 (the length is $1 + p + n \times p$ because of the initial 1). If we add a new prefix it will have the same modal formulae of $1 \ldots n_{p+(n-1)p+1}$ and every π-formula will be already reduced in one of the prefixes between $1 \ldots n_{p+(n-1)p+1}$ and $\sigma_{1+(n+1)p}$. Any further reduction would trigger the pruning lemma.

For every prefix σ_{hb} of length $1+p+n \times p+1$, and every unreduced π-formula $\sigma_{hb} : \Diamond A$, there is a shorter modal copy σ_0 such that $\sigma_0 : \Diamond A$ is reduced. Hence the pruned branch $\mathcal{B}'$ is π-modal-completed.

The first part of the sequence, with zero ν-formulae, can be generated only by π-formulae not under the scope of a necessity operator. Hence the modal depth decreases after each rule application and the first part can be bounded by d_p. This yields the final upper bound of $hb_{\mathsf{L}} = 2 + d_p + n \times p$. □

B.3. PROOFS OF SECTION 10

Proof (Lemma 10.2). The case for (K), (D) and (T) is immediate.

Rule (4) is used for transitive models. Assume that $w \Vdash \Box A$ and $w \mathcal{R} v$. Then, for any $u \in W$ such that $v \mathcal{R} u$ one has $w \mathcal{R} u$, by transitivity. Hence $u \Vdash A$. Since u is arbitrary, also $v \Vdash \Box A$. For rule (4^π) suppose that $w \mathcal{R} v$ and $v \Vdash \neg\Box A$. The semantics forces the existence of a world u such that $v \mathcal{R} u$ and $u \nVdash A$. By transitivity $w \mathcal{R} u$ and thus $w \Vdash \neg\Box A$.

Rule (4^R) is used for Euclidean relations. Suppose that $w \Vdash \Box A$ and $w \mathcal{R} v$. For any u such that $w \mathcal{R} u$, one has $v \mathcal{R} u$, since $\mathcal{R}$ is Euclidean. Thus, if $v \Vdash \Box A$, one has $u \Vdash A$ and then $w \Vdash \Box A$.

For rule (B), assume that $w \mathcal{R} v$ and $v \Vdash \Box A$. By symmetry one has $v \mathcal{R} w$ and therefore $w \Vdash A$.

Rule (Cxt) can be used for Euclidean and contextual models. Suppose the model is Euclidean and that $w \mathcal{R} v$ with $v \Vdash \Box A$. Then let $w \mathcal{R} u$ and $u \mathcal{R} t$. The properties of $\mathcal{R}$ forces $u \mathcal{R} v$ and then $v \mathcal{R} t$. Thus $t \Vdash A$. Since t is arbitrary, one has $u \Vdash \Box A$ and $w \Vdash \Box\Box A$.

If the L-model is a contextual model (see [2] or Table I), then we have directly $v\mathcal{R}t$ and the same reasoning applies.

Delayed rules require the same arguments plus $w_0\mathcal{R}w$. □

Proof (Theorem 10.4 (Cxt)). For the logic of contextual reasoning we must prove the following property [2]: "if $\sigma \rhd \sigma_a$ and $\sigma \rhd \sigma_b$ and $\sigma_a \rhd \sigma_c$, then $\sigma_b \rhd \sigma_c$". The conditions on $\rhd$ for Cxt are the following:

$$\begin{array}{ll} \sigma \rhd \sigma_a & \text{iff} \quad |\sigma_a| = |\sigma| + 1 \\ \sigma \rhd \sigma_b & \text{iff} \quad |\sigma_b| = |\sigma| + 1 \\ \sigma_a \rhd \sigma_c & \text{iff} \quad |\sigma_c| = |\sigma_a| + 1 \end{array}$$

By substitution one has $|\sigma_c| = |\sigma_a| + \ = |\sigma| + 2 = |\sigma_b| + 1$. □

Proof (Theorem 10.4 (K4)). Transitivity equals "$\sigma_a \rhd \sigma_b$ and $\sigma_b \rhd \sigma_c$ implies $\sigma_a \rhd \sigma_c$". The two initial conditions impose the properties below on $\rhd$ (Table VI):

$$\begin{array}{l} \sigma_a \rhd \sigma_b \ \text{ iff } \ \sigma_b = \sigma_a.\sigma' \ \text{ with } \ |\sigma'| \geq 1 \\ \sigma_b \rhd \sigma_c \ \text{ iff } \ \sigma_c = \sigma_b.\sigma'' \ \text{ with } \ |\sigma''| \geq 1 \end{array}$$

By concatenation one has $\sigma_c = \sigma_a.\sigma'.\sigma''$ and $|\sigma'.\sigma''| \geq 1$. □

Proof (Theorem 10.4 (K5)). An Euclidean relation is characterized by the property "$\sigma_a \rhd \sigma_b$ and $\sigma_a \rhd \sigma_c$ implies $\sigma_b \rhd \sigma_c$". The conditions for K5 impose that

$$\begin{array}{l} \sigma_a \rhd \sigma_b \ \text{ iff } \ \sigma_a = \sigma.\sigma' \text{ and } \sigma_b = \sigma.\sigma'' \quad \text{ with } \ |\sigma'| \geq 1 \text{ and } \ |\sigma''| \geq 1 \\ \sigma_a \rhd \sigma_c \ \text{ iff } \ \sigma_a = \sigma_1.\sigma_1' \text{ and } \sigma_c = \sigma_1.\sigma_1'' \ \text{ with } \ |\sigma_1'| \geq 1 \text{ and } \ |\sigma_1''| \geq 1 \end{array}$$

By definition both σ_1 and σ are longer than 1. Since both σ and σ_1 are initial parts of the same prefix σ_a then we only have three cases: $\sigma = \sigma_1$ or $\sigma_1 = \sigma.\tau$ or $\sigma = \sigma_1.\tau$ for some prefix τ such that $|\tau| \geq 1$.

$\sigma = \sigma_1$: this condition implies $\sigma_b = \sigma.\sigma''$ and $\sigma_c = \sigma.\sigma_1$. Since $|\sigma''| \geq 1$ and $|\sigma_1''| \geq 1$ we have directly $\sigma_b \rhd \sigma_c$;

$\sigma_1 = \sigma.\tau$: we substitute σ_1 in the equation of σ_c and obtain $\sigma_b = \sigma.\sigma''$ together with $\sigma_c = \sigma.\tau.\sigma_1$. Since $|\sigma''| \geq 1$ and $|\tau.\sigma_1| \geq 1$, we obtain $\sigma_b \rhd \sigma_c$.

The remaining case is similar. □

Proof (Lemma 10.5). If the syntactic relation $\rhd$ accesses only *close neighbors*, the proof is trivial: complete reduction coincides with SST reduction. For instance, for the logic B there are three cases: $\sigma \rhd_{\mathsf{B}} \sigma$, $\sigma \rhd_{\mathsf{B}} \sigma.n$, and $\sigma.n \rhd_{\mathsf{B}} \sigma$. Apply respectively SST reduction for rule (T), (K), and (B).

If $\rhd$ connects *forward prefixes*, then $\sigma \rhd \sigma^*$ implies that σ^* has the form $\sigma.n_1 \ldots n_k$ for $k \geq 1$ (or $k \geq 0$ if the logic is reflexive). By Theorem 10.1, for every

$i = 1 \ldots k$ also $\sigma.n_1 \ldots n_i$ is present. Apply reduction w.r.t. (4) or (4^D) to construct the chain $\sigma : \Box A \in \mathcal{B}$ implies $\sigma.n_1 : \Box A \in \mathcal{B}$, implies $\sigma.n_1.n_2 : \Box A \in \mathcal{B}$ etc. Once we have $\sigma.n_1 \ldots n_{k-1} : \Box A \in \mathcal{B}$, apply SST reduction w.r.t. (K) to obtain $\sigma^* : A \in \mathcal{B}$.

When *backward prefixes* are connected by $\triangleright$, then $\sigma \triangleright \sigma^*$ implies that for some prefix σ_0 we have $\sigma = \sigma_0.n_1 \ldots n_k$ and $\sigma^* = \sigma_0.m$ (or $\sigma^* = \sigma_0$ if the logic is reflexive). By Theorem 10.1, for every $i = 1 \ldots k$ also $\sigma_0.n_1 \ldots n_i$ is in $\mathcal{B}$. For every i, if $\sigma_0.n_1 \ldots n_{i+1} : \Box A \in \mathcal{B}$ then $\sigma_0.n_1 \ldots n_i : \Box A \in \mathcal{B}$, because $\sigma_0.n_1 \ldots n_{i+1} : \Box A \in \mathcal{B}$ is reduced for rule (4^R) by hypothesis. Therefore, $\sigma_0 : \Box A$ is present in $\mathcal{B}$. Then we apply the reduction for rule (K) and obtain $\sigma_0.m : A \in \mathcal{B}$.

For K5 we combine these technique: we apply reduction for rule (4^R) until we can conclude that $1.n : \Box A$ is present. If $\sigma^* = 1.m$ then we use reduction for rule (4^R) followed by (K). Otherwise, first (Cxt) obtaining $1 : \Box\Box A$, and then (K). From $1.m : \Box A \in \mathcal{B}$, the reasoning is identical to that for syntactical relations $\triangleright$ connecting forward prefixes: first we consider reduction for rule (4^D) and last reduction for rule (K).

For the logic Cxt apply reduction for rule (Cxt) until we have shown that $1 : \Box^n A \in \mathcal{B}$; then apply reduction for (K) up to σ^*.

The tableaux for K45, KD45, S5 with rule (4^π) have prefixes of the form 1 and $1.n$, since we restricted the application of the π-rule. For moving $\Box A$ we apply (4^R) once and then either (K) or (T). □

Acknowledgments

I thank L. Carlucci Aiello and F. Pirri for their encouragement and support. I benefited from an ongoing discussion on tableau methods with R. Goré and F. Donini. The precise comments of two referees greatly improved the quality of this paper. This work has been supported by ASI, CNR, and MURST grants.

References

1. Beckert, B. and Goré, R.: Free variable tableaux for propositional modal logics, in [17], 1997, pp. 91–108.
2. Buvač, S., Buvač, V. and Mason, I.: Metamathematics of contexts, *Fundamenta Inform.* **23**(3) (1995), 263–301.
3. Catach, L.: TABLEAUX, a general theorem prover for modal logics, *J. Automated Reasoning* **7** (1991), 489–510.
4. Cerrito, S. and Cialdea Mayerm M.: Hintikka multiplicities in matrix decision methods for some propositional modal logics, in [17], 1997, pp. 138–152.
5. Cerrito, S. and Cialdea Mayer, M.: A polynomial translation of S4 into T and contraction-free tableaux for S4, *J. Interest Group in Pure Appl. Logic* **5**(2) (1997), 287–300.
6. D'Agostino, G., Montanari, A. and Policriti, A.: A set-theoretic translation method for polymodal logics, *J. Automated Reasoning* **15** (1995), 317–337.

7. De Giacomo, G. and Massacci, F.: Tableaux and algorithms for propositional dynamic logic with converse, in [34], 1996, pp. 613–628.
8. Demri, S.: Uniform and non uniform strategies for tableaux calculi for modal logics, *J. Appl. Non-Classical Logics* **5**(1) (1995), 77–96.
9. Enjalbert, P. and Fariñas del Cerro, L.: Modal resolution in clausal form, *Theoret. Comput. Sci.* **65** (1989), 1–33.
10. Fagin, R., Halpern, J., Moses, Y. and Vardi, M.: *Reasoning about Knowledge*, The MIT Press, 1995.
11. Fischer, N. and Ladner, R.: Propositional dynamic logic of regular programs, *J. Comput. System Sci.* **18** (1979), 194–211.
12. Fitch, F.: Tree proofs in modal logic, *J. Symbolic Logic* **31** (1966).
13. Fitting, M.: *Proof Methods for Modal and Intuitionistic Logics*, Reidel, 1983.
14. Fitting, M.: First-order modal tableaux, *J. Automated Reasoning* **4** (1988), 191–213.
15. Fitting, M.: Basic modal logic, in D. Gabbay, C. Hogger, and J. Robinson (eds), *Handbook of Logic in Artificial Intelligence and Logic Programming*, Volume 1, Oxford Univ. Press, 1993, pp. 365–448.
16. Gabbay, D.: Labelled deductive systems, Technical Report MPI-I-94-223, Max Plank Institute für Informatik (MPII), Saarbrüken, Germany, 1994. To appear as a book by Oxford Univ. Press.
17. Galmiche, D. (ed.): *Proc. of the Internat. Conf. on Analytic Tableaux and Related Methods (TABLEAUX-97)*, LNAI 1227, Springer-Verlag, 1997.
18. Gent, I.: Theory matrices (for modal logics) using alphabetical monotonicity, *Studia Logica* **52** (1993), 233–257.
19. Giunchiglia, F. and Sebastiani, R.: Building decision procedures for modal logics from propositional decision procedures – the case study of modal K, in M. McRobbie and J. Slaney (eds.), *Proc. of the 13th Internat. Conf. on Automated Deduction (CADE-96)*, LNAI 1104, Springer-Verlag, 1996, pp. 583–597.
20. Goré, R.: Tableaux method for modal and temporal logics, Technical Report TR-ARP-15-5, Australian Nat. University, 1995. To appear as chapter on the *Handbook of Tableau Methods* by Kluwer.
21. Halpern, J. and Fagin, R.: Modelling knowledge and action in distributed systems, *Distrib. Comput.* **3**(4) (1989), 159–177.
22. Halpern, J. and Moses, Y.: A guide to completeness and complexity for modal logics of knowledge and belief, *Artif. Intell.* **54** (1992), 319–379.
23. Heuerding, A., Seyfried, M. and Zimmermann, H.: Efficient loop-check for backward proof search in some non-classical logics, in *Proc. of the 5th Workshop on Theorem Proving with Analytic Tableaux and Related Methods (TABLEAUX-96)*, LNAI 1071, Springer-Verlag, 1996, pp. 210–225.
24. Huet, G.: Confluent reductions: Abstract properties and applications to term rewriting systems, *J. ACM* **27**(4) (1980), 797–821.
25. Hughes, G. and Cresswell, M.: *An Introduction to Modal Logic*, Methuen, 1968.
26. Hustadt, U. and Schmidt, R.: On evaluating decision procedure for modal logic, in M. Pollack (ed.), *Proc. of the 15th Internat. Joint Conf. on Artificial Intelligence (IJCAI-97)*, 1997, pp. 202–207.
27. Johnson, D.: A catalog of complexity classes, in J. van Leeuwen (ed.), *Handbook of Theoretical Computer Science*, Elsevier Science Publishers (North-Holland), Amsterdam, 1990, pp. 67–162.
28. Johnson, D. and Garey, M.: *Computers and Intractability: A Guide to the Theory of NP-Completeness*, Freeman, San Francisco, 1979.
29. Kripke, S.: Semantical analysis of modal logic I: normal propositional calculi, *Z. Math. Logik Grundlag. Math.* **9** (1963), 67–96.

30. Ladner, R.: The computational complexity of provability in systems of modal propositional logic, *SIAM J. Comput.* **6**(3) (1977), 467–480.
31. Marek, W., Schwarz, S. and Truszczynski, M.: Modal nonmonotonic logics: Ranges, characterization, computation, *J. ACM* **40**(4) (1993), 963–990.
32. Massacci, F.: Strongly analytic tableaux for normal modal logics, in A. Bundy (ed.), *Proc. of the 12th Internat. Conf. on Automated Deduction (CADE-94)*, LNAI 814, Springer-Verlag, 1994, pp. 723–737.
33. Massacci, F.: Contextual resoning is NP-complete, in W. Clancey and D. Weld (eds.), *Proc. of the Nat. (US) Conf. on Artificial Intelligence (AAAI-96)*, AAAI/MIT Press, 1996, pp. 621–626.
34. McRobbie, M. and Slaney, J. (eds.): *Proc. of the 13th Internat. Conf. on Automated Deduction (CADE-96)*, LNAI 1104, Springer Verlag, 1996.
35. Mints, G.: Gentzen-type systems and resolution rules, in *Internat. Conf. on Computer Logic (COLOG)*, LNCS 417, Springer-Verlag, 1988, pp. 198–231.
36. Nonnengart, A.: First-order modal logic theorem proving and functional simulation, in *Proc. of the 13th Internat. Joint Conf. on Artificial Intelligence (IJCAI-93)*, Morgan Kaufmann, 1993, pp. 80–85.
37. Ohlbach, H.: A resolution calculus for modal logic, in *Proc. of the 9th Internat. Conf. on Automated Deduction (CADE-88)*, LNCS 310, Springer-Verlag, 1988, pp. 500–516.
38. Ohlbach, H.: Semantic-based translation methods for modal logics, *J. Logic Comput.* **1**(5) (1991), 691–746.
39. Ohlbach, H.: Translation methods for non-classical logics – an overview, *J. Interest Group in Pure Appl. Logic* **1**(1) (1993), 69–89.
40. Ohnishi, M. and Matsumoto, K.: Gentzen method in modal calculi, *Osaka Math. J.* **9** (1957), 113–130.
41. Ohnishi, M. and Matsumoto, K.: Gentzen method in modal calculi, II, *Osaka Math. J.* **11** (1959), 115–120.
42. Russo, A.: Generalising propositional modal logic using labelled deductive systems, in *Proceedings of the Internat. Workshop on Frontiers of Combining Systems (FroCoS-96)*, LNAI, Springer-Verlag, 1996.
43. Schmitt, S. and Kreitz, C.: Converting non-classical matrix proofs into sequent-style systems, in [34], 1996, pp. 418–432.
44. Schmidt, R.: Resolution is a decision procedure for many propositional modal logics, in M. Kracht, M. de Rijke, H. Wansing, and M. Zakharyaschev (eds.), *Advances in Modal Logic*, Vol. 1, Lecture Notes 87, CSLI Publications, Stanford, pp. 189–208.
45. Schwarz, G.: Gentzen style systems for K45 and KD45, in A. Meyer and M. Taitslin (eds.), *Logic at Botik '89, Symposium on Logical Foundations of Computer Science*, LNAI 363, Springer-Verlag, 1989.
46. Smullyan, R.: *First Order Logic*, Springer-Verlag, 1968. Republished by Dover, New York, in 1995.
47. Wallen, L.: *Automated Deduction in Nonclassical Logics*, The MIT Press, 1990.