SAT-based decision procedures for normal modal logics: a theoretical framework*

Roberto Sebastiani and Adolfo Villafiorita

ITC-IRST, 38050 Pantè di Povo (TN), Italy
{rseba,adolfo}@irst.itc.it

Abstract. Tableau systems are very popular in AI for their simplicity and versatility. In recent papers we showed that tableau-based procedures are intrinsically inefficient, and proposed an alternative approach of building decision procedures on top of SAT decision procedure. We called this approach "SAT-based". In extensive empirical tests on the case study of modal K, a SAT-based procedure drastically outperformed state-of-the-art tableau-based systems. In this paper we provide the theoretical foundations for developing SAT-based decision procedures for many different modal logics.

1 Introduction

By a *tableau* framework for a logic $\mathcal{L}$ we generically denote a refutation system for $\mathcal{L}$ extending Smullyan's propositional framework [13]. Tableau frameworks have been presented for many logics, including propositional and classical first order logic [13], modal logics (see, e.g., [6]), terminological logics (see, e.g., [1]), dynamic logics [5]. The popularity of tableaux is due to many reasons, e.g., their simplicity and versatility. Unfortunately, tableau-based procedures are intrinsically inefficient. In fact, as we showed in [10, 8] – but see also [3] – tableau-based procedures present two intrinsic weaknesses, which drastically affect their performance:

(i) Propositional tableau rules branch syntactically on the disjunctions occurring inside the input well-formed-formula (wff) φ, creating two branches which are mutually consistent. This causes a (possibly huge) number of redundant branches in the search tree. This propositional redundancy propagates with the modal depth of φ.
(ii) Propositional tableaux do not cut branches as soon as they violate constraints of the formulas. Therefore they cannot perform the heavy pruning that standard procedures for propositional satisfiability do.

[9, 10] presented KSAT, a new decision procedure for logic K(m)/$\mathcal{ALC}$ built on top of a decision procedure for propositional satisfiability (SAT). We called this approach "SAT-based". KSAT outperformed by orders of magnitude a distributed tableau-based system in extensive empirical tests. In [8] we showed that

* The authors thank Fausto Giunchiglia for providing valuable feedback.

this performance gap was not by chance, for the SAT-based approach is intrinsically superior to the tableau-based one, as it solves the two weaknesses (i) (ii) described above. Moreover, in [12] an asymptotic complexity analysis proved that KSAT presents better upper bounds than the corresponding tableau-based procedure. Finally [7] presents also an empirical comparison with TA [11], a system based on the translation into first order logics, in which KSAT outperforms TA of orders of magnitude. [2] [3]

KSAT was developed as a case-study for the much more general goal of developing SAT-based decision procedures for many different logics, in particular modal and terminological logics. In this paper we provide the theoretical foundations for making this goal feasible:

(1) starting from the formal framework of labeled tableaux described in [6], we provide a general SAT-based formal framework for all the normal modal logics described there. Then we prove its correctness and completeness by means of a general-purpose schema of SAT-based procedure, according to the guidelines of the corresponding proof in [6];
(2) as most intuitions, definitions and results are independent from the logic considered, we give general suggestions of how to build SAT-based frameworks from Tableau-based ones in other logics.

The work is based on a simple intuition: *SAT-based frameworks are obtained from tableau-based ones by substituting Smullyan's propositional rules with one single rule, which represents the application of a SAT procedure.*

The paper is organized as follows. In Section 2 we recall Fitting's tableau framework [6] for normal modal logics. In Section 3 we describe a SAT-based framework for the same logics, whose correctness and completeness are proved in Appendix A. In Section 4 we describe KSAT and show how this procedure implements the SAT-based framework for the logic K. This gives the guideline for developing efficient SAT-based decision procedures for all the modal logics considered in Section 3. In Section 5 we provide the main guidelines for developing SAT-based frameworks and procedures for other logics.

2 Fitting's tableaux for $\mathcal{N}$

Following [6], we consider the class $\mathcal{N}$ of the normal modal logics K, KB, K4, D, T, DB, B, D4, S4 and S5. We assume all the standard definitions and results for these logics (see, e.g., [2, 6]). We denote by Λ the language of $\mathcal{N}$, i.e., the least set of formulas containing the set of propositional atoms $\mathcal{A} = \{A_1, A_2, \ldots\} \cup \{T, F\}$ closed under the set of connectives $\{\neg, \wedge, \vee\}$ and $\Box$. We assume that all modal wffs are in negation normal form, that is, combinations of $\wedge$, $\vee$, $\Box$, $\neg\Box$, A_i and $\neg A_i$ (the general case follows straightforwardly).

[2] [7] also replies to the critics to KSAT presented in [11], showing that the latter were affected by some crucial errors in the empirical tests presented there.

[3] The papers [9, 10, 8, 12, 7] are available at the first author's WWW page `http://afrodite.itc.it:1024/~rseba/publist.html`.

Axiom	Property of $\mathcal{R}$	Description
B	symmetric	$\forall\ u\ v\ \mathcal{R}(u,v) \Longrightarrow \mathcal{R}(v,u)$
D	serial	$\forall\ u\ \exists\ v\ \mathcal{R}(u,v)$
T	reflexive	$\forall\ u\ \mathcal{R}(u,u)$
4	transitive	$\forall\ u\ v\ w\ \mathcal{R}(u,v)\ e\ \mathcal{R}(v,w) \Longrightarrow \mathcal{R}(u,w)$
5	euclidean	$\forall\ u\ v\ w\ \mathcal{R}(u,v)\ e\ \mathcal{R}(u,w) \Longrightarrow \mathcal{R}(v,w)$

Logic $\mathcal{L} \in \mathcal{N}$	Properties of $\mathcal{R}$
K	—
KB	symmetric
KD	serial
KT = KDT (T)	reflexive
K4	transitive
K5	euclidean
KBD	symmetric and serial
KBT = KBDT (B)	symmetric and reflexive
KB4 = KB5 = KB45	symmetric and transitive
KD4	serial and transitive
KD5	serial and euclidean
KT4 = KDT4 (S4)	reflexive and transitive
KT5 = KBD4 = KBD5 = KBT4 = KBT5 = KDT5 = KT45 = KBD45 = KBT45 = KDT45 = KBDT4 = KBDT5 = KBDT45 (S5)	reflexive, transitive and symmetric (equivalence)
K45	transitive and euclidean
KD45	serial, transitive and euclidean

Table 1. Properties of $\mathcal{R}$ for the various normal modal logics.

We denote any *Kripke structure* for $\mathcal{L} \in \mathcal{N}$ by a tuple $M = <\mathcal{U}, \pi, \mathcal{R}>$, where $\mathcal{U}$ is a set of worlds, π is a function $\pi : \mathcal{A} \times \mathcal{U} \longmapsto \{True, False\}$, and $\mathcal{R}$ is a binary relation on the worlds of $\mathcal{U}$. The different logics $\mathcal{L} \in \mathcal{N}$ differ in the properties of the relation $\mathcal{R}$, as in Table 1. We call a $\mathcal{L}$-*situation* any pair $< M, u >$ (simply "M, u" from now on) so that M is a Kripke structure for $\mathcal{L}$ and $u \in \mathcal{U}$. We extend the definition of $\models_{\mathcal{L}}$ to wff sets $\mu = \{\varphi_1, ..., \varphi_n\}$ as follows:

$$M, u \models_{\mathcal{L}} \mu \iff M, u \models_{\mathcal{L}} \varphi_i, \ \ \textit{for every } \varphi_i \in \mu.$$

We use the prefix "$\mathcal{L}$-" to mean "in the logic $\mathcal{L}$": M, u $\mathcal{L}$-*satisfies* φ, φ is $\mathcal{L}$-*satisfiable*, etc. The binary relation $\models_{\mathcal{L}}$ between a modal formula φ and a $\mathcal{L}$-situation M, u is defined as follows:

$$\begin{array}{lcl}
M, u \models_{\mathcal{L}} A_i,\ A_i \in \mathcal{A} & \iff & \pi(A_i, u) = True; \\
M, u \models_{\mathcal{L}} \neg\varphi_1 & \iff & M, u \not\models_{\mathcal{L}} \varphi_1; \\
M, u \models_{\mathcal{L}} \varphi_1 \wedge \varphi_2 & \iff & M, u \models_{\mathcal{L}} \varphi_1 \textit{ and } M, u \models_{\mathcal{L}} \varphi_2; \\
M, u \models_{\mathcal{L}} \varphi_1 \vee \varphi_2 & \iff & M, u \models_{\mathcal{L}} \varphi_1 \textit{ or } M, u \models_{\mathcal{L}} \varphi_2; \\
M, u \models_{\mathcal{L}} \Box\varphi_1 & \iff & M, v \models_{\mathcal{L}} \varphi_1 \text{ for every } v \in \mathcal{U} \text{ s.t. } \mathcal{R}(u,v) \text{ holds in } M.
\end{array}$$

"$M, u \models_{\mathcal{L}} \varphi$" should be read as "$M, u$ *satisfies* φ in $\mathcal{L}$" (alternatively, "M, u $\mathcal{L}$-satisfies φ"). We say that a formula $\varphi \in \Lambda$ is *$\mathcal{L}$-satisfiable* if and only if there exist a situation M, u so that $M, u \models_{\mathcal{L}} \varphi$.

A *labeled wff* is a pair $\sigma : \varphi$, where φ is a wff in Λ and σ is a sequence of integers, called *label*, labeling a world in a Kripke structure for $\mathcal{L}$. Intuitively, $\sigma : \varphi$ means "the wff φ in the world σ". For every $\mathcal{L} \in \mathcal{N}$, [6] gives a notion of *accessibility relation* between labels and gives the properties for these relations for the various logics $\mathcal{L}$. Essentially, they mirror the accessibility relation between the worlds they label. Notationally, if $\Gamma = \{\varphi_1, \ldots, \varphi_n\}$, we write $\sigma : \Gamma$ for $\{\sigma : \varphi_1, \ldots, \sigma : \varphi_n\}$.

Given a wff set $\Gamma = \{\varphi_1, \ldots, \varphi_n\}$, a *$\mathcal{L}$-tableau* for Γ is a binary tree of sets of labeled wffs whose root is $\{1 : \Gamma\}$, where 1 is the label of the initial world. In this respect, we see a branch as the union of the wff sets of its nodes. A label σ is called *used* in a branch iff there is at least one wff $\sigma : \varphi$ in the branch; σ is called *unrestricted* iff it is not an initial segment of a label used in the branch; σ' is called a *simple extension* of σ iff $\sigma' = \sigma, n$, for some integer n; σ is *normal* in a branch if $\sigma : \Box\varphi_i$ occurs in the branch for some formula φ_i. Notice that these definitions do not depend on the logic $\mathcal{L}$ considered.

A branch is *closed* iff it contains an atom φ_i and its negation $\neg\varphi_i$, *open* otherwise. A $\mathcal{L}$-tableau is *closed* iff all its branches are closed, *open* otherwise. One branch is *completely expanded* iff no more rules are applicable. [4] A $\mathcal{L}$-tableau is *completely expanded* iff all its branches are completely expanded.

[6] gives a notion of $\mathcal{L}$-satisfiability of a $\mathcal{L}$-tableau, for every $\mathcal{L} \in \mathcal{N}$. Briefly, given a Kripke structure $M =< \mathcal{U}, \pi, \mathcal{R} >$, an *$\mathcal{L}$-interpretation* $\mathcal{I}$ is a map from labels to possible worlds in M so that $\mathcal{R}(\mathcal{I}(\sigma), \mathcal{I}(\sigma'))$ holds if σ' is $\mathcal{L}$-accessible from σ. $\mathcal{I}$ *$\mathcal{L}$-satisfies* a branch θ if $M, \mathcal{I}(\sigma) \models_{\mathcal{L}} \varphi$, for each $\sigma : \varphi \in \theta$. A $\mathcal{L}$-tableau is *$\mathcal{L}$-satisfiable* iff at least one of its branches is $\mathcal{L}$-satisfiable. From now on, with a little abuse of notation, we use σ to represent indifferently the label σ and the labeled world $\mathcal{I}(\sigma)$.

Intuitively, a completely expanded open branch contains a set of labeled wffs which represent a model for the input wff set Γ. Each branch is an attempt of building a model. If it closes, this attempt fails. If all branches close, there are no models. A correctness and completeness theorem states that a wff set $\Gamma = \{\varphi_1, \ldots, \varphi_n\}$ is $\mathcal{L}$-unsatisfiable if and only if there exists a closed $\mathcal{L}$-tableau for $\sigma : \Gamma$, for some label σ. Therefore, the idea underlying tableau-based algorithm is to try to build a closed $\mathcal{L}$-tableau for $1 : \Gamma$. If this succeeds, Γ is $\mathcal{L}$-unsatisfiable, otherwise, it builds a fully expanded open branch, so that Γ is $\mathcal{L}$-satisfiable.

A $\mathcal{L}$-tableau is built as follows. At the first step the root $\{1 : \Gamma\}$ is created. At the i-th step, the current branch is expanded by applying to a chosen wff in

[4] In some logics, like K4 and S4, it is possible to have infinite cyclic open branches. In this case "applicable" must be read as "applicable without generating a cycle". See [6] for details.

the branch the rule corresponding to its main connective among the following:

$$(\wedge\text{-elimination})\ \frac{\sigma:\varphi_1\wedge\sigma:\varphi_2}{\begin{array}{c}\sigma:\varphi_1\\ \sigma:\varphi_2\end{array}}\qquad(\vee\text{-elimination})\ \frac{\sigma:\varphi_1\vee\sigma:\varphi_2}{\sigma:\varphi_1\qquad\sigma:\varphi_2},$$

$$(\neg\Box\text{-elimination})\ \frac{\sigma:\neg\Box\varphi}{\sigma':\neg\varphi}\qquad(\Box\text{-elimination})\ \frac{\sigma:\Box\varphi}{\sigma'':\varphi}$$

The latter two rules are constrained by the following applicability conditions:

- **$\neg\Box$-elimination**: σ' must be an unrestricted simple extension of σ. Intuitively, σ' represents a new world directly accessible from σ.
- **$\Box$-elimination**: σ'' must be used in the branch and must be accessible from σ. Intuitively, σ'' represents an existing world accessible from σ. In the logics KD, T, KBD, B, KD4, S4 and S5, σ'' can alternatively be an unrestricted simple extension of σ.

Any application of the $\vee$-elimination rule splits the branch into two sub-branches. The first two rules are called *propositional rules*, the latter *modal rules* [5].

3 The SAT-based framework for $\mathcal{N}$

3.1 Atoms, assignments and propositional satisfiability

We call an *atom* any formula that cannot be decomposed propositionally, that is, any formula whose main connective is not propositional. A *literal* is either an atom or its negation. Given a formula φ, we call an atom [literal] a *top-level atom [literal]* for φ if it occurs in φ and under the scope of no boxes.

We call a *total truth assignment* μ for a modal formula φ a set of literals

$$\mu=\{\Box\alpha_1,\ldots,\Box\alpha_N,\neg\Box\beta_1,\ldots,\neg\Box\beta_M,A_1,\ldots,A_R,\neg A_{R+1},\ldots,\neg A_S\},$$

such that every top-level atom of φ occurs either positively or negatively in μ. μ is interpreted as a truth value assignment to all the top-level atoms of φ: $\Box\alpha_i\in\mu$ means that $\Box\alpha_i$ is assigned to $True$, $\neg\Box\beta_i\in\mu$ means that $\Box\beta_i$ is assigned to $False$. We say that $M,u\models_{\mathcal{L}}\mu$ if $M,u\models_{\mathcal{L}} l_i$, for every literal $l_i\in\mu$, and that μ is $\mathcal{L}$-satisfiable iff $M,u\models_{\mathcal{L}} l_i$, for some *$\mathcal{L}$-situation* M,u.

We say that a total truth assignment μ for φ *propositionally satisfies* φ, written $\mu\models_p\varphi$, if and only if it makes φ evaluate to $True$, that is, for all sub-formulas φ_1,φ_2 of φ:

$$\begin{array}{ll}
\mu\models_p\varphi_1,\ \varphi_1 \text{ top-level atom of } \varphi & \Longleftrightarrow \varphi_1\in\mu;\\
\mu\models_p\neg\varphi_1 & \Longleftrightarrow \textit{not }\mu\models_p\varphi_1;\\
\mu\models_p\varphi_1\wedge\varphi_2 & \Longleftrightarrow \mu\models_p\varphi_1 \textit{ and } \mu\models_p\varphi_2.\\
\mu\models_p\varphi_1\vee\varphi_2 & \Longleftrightarrow \mu\models_p\varphi_1 \textit{ or } \mu\models_p\varphi_2.
\end{array}$$

[5] $\wedge$-elimination, $\vee$-elimination, $\Box$-elimination and $\neg\Box$-elimination and their equivalent versions are often called α, β, ν and π rules respectively [13, 6].

For every φ_1 and φ_2, we say that $\varphi_1 \models_p \varphi_2$ iff $\mu \models_p \varphi_1$ implies $\mu \models_p \varphi_2$ for every total assignment μ. It is easy to verify that $\varphi_1 \models_p \varphi_2$ iff $\models_p \neg\varphi_1 \vee \varphi_2$. We also say that $\models_p \varphi$ (φ is *propositionally valid*) iff $\mu \models_p \varphi$ for every total assignment μ for φ. It is easy to verify that $\models_p \varphi$ iff $\neg\varphi$ is propositionally unsatisfiable. It is important to notice that, if we consider a wff φ as a propositional wff in its top-level atoms, than $\models_p$ is the standard satisfiability in propositional logic. Notice also that $\models_p$ is stronger than $\models_{\mathcal{L}}$, that is, if $\varphi_1 \models_p \varphi_2$, then $\varphi_1 \models_{\mathcal{L}} \varphi_2$, but the vice-versa is not true. For instance, $\Box(A_1 \wedge A_2) \models_{\mathcal{L}} \Box A_1 \wedge \Box A_2$, but $\Box(A_1 \wedge A_2) \not\models_p \Box A_1 \wedge \Box A_2$.

We call a *partial* truth assignment μ for φ a truth assignment to a proper subset of the top-level atoms of φ. If $\mu_2 \subseteq \mu_1$, then we say that μ_1 *extends* μ_2 and μ_2 *subsumes* μ_1. We say that a partial truth assignment μ *propositionally satisfies* φ if and only if all the total assignments for φ which extend μ propositionally satisfy φ. For instance, if $\varphi = \Box\varphi_1 \vee \neg\Box\varphi_2$, then the partial assignment $\mu = \{\Box\varphi_1\}$ is such that $\mu \models_p \varphi$. In fact, both $\{\Box\varphi_1, \Box\varphi_2\}$ and $\{\Box\varphi_1, \neg\Box\varphi_2\}$ propositionally satisfy φ. Obviously, if $\mu_1 \supseteq \mu_2$, then $\mu_1 \models_p \mu_2$. We call $Assigns(\varphi)$ the set of all possible assignments for φ, either total or partial.

We say that a collection $\mathcal{M} = \{\mu_1, \ldots, \mu_n\}$ of (possibly partial) assignments satisfying φ is *complete* iff

$$\models_p \varphi \equiv \bigvee_j \mu_j,$$

where each μ_j is written as a conjunction of its elements. $\mathcal{M}$ is complete in the sense that, for every total assignment η so that $\eta \models_p \varphi$, there exists $\mu_j \in \mathcal{M}$ so that $\eta \supseteq \mu_j$. Therefore $\mathcal{M}$ is a compact representation of the whole set of total assignments which propositionally satisfy φ. Notice that $\models_{\mathcal{L}} \varphi \equiv \bigvee_j \mu_j$, for every $\mathcal{L} \in \mathcal{N}$, as $\models_p$ is stronger than $\models_{\mathcal{L}}$.

Theorem 1. *Let φ be a modal formula and let $\mathcal{M} = \{\mu_1, \ldots, \mu_n\}$ be a complete collection of truth assignments satisfying φ. Then, for a given $\mathcal{L}$-situation M, u, $M, u \models_{\mathcal{L}} \varphi$ if and only if at least one $\mu_j \in \mathcal{M}$ is such that $M, u \models_{\mathcal{L}} \mu_j$.*

Proof.

If: Let μ'_j be the extension of μ_j to all top-level atoms of φ so that $M, u \models_{\mathcal{L}} \mu'_j$. Then $\mu'_j \models_p \varphi$. Then, as in the proof of Theorem 2 in [8], $M, u \models_{\mathcal{L}} \varphi$.

Only if: If $M, u \models_{\mathcal{L}} \varphi$, then $M, u \models_{\mathcal{L}} \bigvee_j \mu_j$, and thus $M, u \models_{\mathcal{L}} \mu_j$, for some j. Q.E.D.

Theorem 1 reduces the $\mathcal{L}$-satisfiability of a formula φ to the $\mathcal{L}$-satisfiability of a complete collection of its truth assignments. Notice that this result is not committed to $\mathcal{L} \in \mathcal{N}$, but it can be easily extended to any logic whose semantic gives a standard interpretation to the propositional connectives.

3.2 The SAT-based framework for $\mathcal{N}$

Definition 2 propositional decider. We call a **propositional decider** a total function f which maps any wff $\varphi \in \Lambda$ into a complete collection of assignments satisfying φ, that is, $f(\varphi) = \{\mu_1, \ldots, \mu_n\}$, so that $\models_p \varphi \equiv \bigvee_j \mu_j$.

Notice that, if we consider any wff $\varphi \in \Lambda$ as a propositional formula in its top-level atoms, then the notion of propositional decider matches many state-of-the-art SAT procedures.

Definition 3 $\mathcal{L}$-tableau$_f$. Given a logic $\mathcal{L} \in \mathcal{N}$ and a propositional decider f, a $\mathcal{L}$-tableau$_f$ is a formalism obtained from $\mathcal{L}$-tableau by substituting all the propositional rules with the single rule:

$$(f\text{-application}) \frac{\sigma : \varphi}{\sigma : \mu_1 \quad \sigma : \mu_2 \quad \ldots \quad \sigma : \mu_n}, \qquad \{\mu_1, \ldots, \mu_n\} = f(\varphi).$$

All the other definitions related to $\mathcal{L}$-tableau hold also for $\mathcal{L}$-tableau$_f$. Intuitively, we use the propositional decider to decompose "one shot" a wff φ into a complete collection of assignments satisfying φ. To this extent, e.g., we call $\mathcal{L}$-tableau$_{DPLL}$ the $\mathcal{L}$-tableau$_f$ obtained with Davis-Putnam-Longemann-Loveland procedure (DPLL), $\mathcal{L}$-tableau$_{PTAB}$ the $\mathcal{L}$-tableau$_f$ obtained with Propositional Tableaux (Ptab) [13], and so on. A noteworthy exception is DPLL with pure-literal rule, as the set of assignment generated is not complete.

Notice that $\mathcal{L}$-tableau$_{PTAB}$ is trivially correct and complete, as it is just a subcase of standard $\mathcal{L}$-tableau. Each $\mathcal{L}$-tableau$_f$ differs from $\mathcal{L}$-tableau$_{PTAB}$ only in the way it performs the embedded propositional reasoning, that is, for the different complete collection of assignments $\mathcal{M}$ it generates for each non-literal formula φ it reasons on. As stated before, different $\mathcal{M}$'s are just different compact representations of the same global set of total assignments. This suggests that the correctness and completeness of all $\mathcal{L}$-tableau$_f$'s are a consequence of the correctness and completeness of $\mathcal{L}$-tableau$_{PTAB}$, as stated in the following theorem.

Theorem 4. *Given $\mathcal{L} \in \mathcal{N}$, a propositional decider f and a wff set Γ, Γ is $\mathcal{L}$-unsatisfiable if and only if there exists a closed $\mathcal{L}$-tableau$_f$ for Γ.*

The proof is given in Appendix A. It mirrors step by step the equivalent proof for $\mathcal{L}$-tableau in Chapter 8 of [6], introducing only the slight modifications needed for handling f-application rules instead of $\wedge/\vee$-elimination rules.s

4 An example of SAT-based procedure: KSAT

Ksat is a state-of-the-art decision procedure for logics K(m)/$\mathcal{ALC}$ presented in [9, 10, 8]. In its basic version, Ksat is reported in Figure 1. [6] Ksat takes a modal propositional wff φ as input and returns a truth value asserting whether φ is K-satisfiable or not. Ksat invokes Ksat$_W$ (where "$_W$" stands for "Wff"), passing as arguments φ and the empty assignment $\emptyset$. Ksat$_W$ is a variant of (a non-CNF version of) DPLL [4]. Unlike DPLL, whenever an assignment μ has been found ("base" step), Ksat$_W$ invokes Ksat$_A(\mu)$ instead of returning

[6] The actual Ksat algorithm is more sophisticated. See [9, 8, 7] for details.

```
function KSAT(φ)
    return KSAT_W(φ, ∅);

function KSAT_W(φ, μ)
    if φ = T                                              /* base      */
        then return KSAT_A(μ);
    if φ = F                                              /* backtrack */
        then return False;
    if {a unit clause (l) occurs in φ}                    /* unit      */
        then return KSAT_W(assign(l, φ), μ ∪ {l});
    l := choose-literal(φ);                               /* split     */
    return  KSAT_W(assign(l, φ), μ ∪ {l})  or
            KSAT_W(assign(¬l, φ), μ ∪ {¬l});

function KSAT_A({□α_1, ..., □α_N, ¬□β_1, ..., ¬□β_M, A_1, ..., ¬A_S})
    for each conjunct "¬□β_j" do
        φ^j := ⋀_i α_i ∧ ¬β_j;                            /* ¬□/□-elimination */
        if not KSAT(φ^j)
            then return False;
    return True;
```

Fig. 1. The basic version of KSAT algorithm.

True. Essentially, DPLL is used to generate truth assignments μ's, whose K-satisfiability is recursively checked by KSAT_A. [7] $\text{KSAT}_A(\mu)$ invokes KSAT on $\varphi^j = \bigwedge_i \alpha_i \wedge \neg\beta_j$ for any conjunct $\neg\Box\beta_j$ occurring in μ. This is repeated until either KSAT returns a negative value (in which case $\text{KSAT}_A(\mu)$ returns *False*) or no more $\neg\Box\beta_j$'s are available (in which case $\text{KSAT}_A(\mu)$ returns *True*).

According to the SAT-based framework of Section 3.2, KSAT is the result of applying a control algorithm to the rules of K-tableau$_{DPLL}$, that is, DPLL-application and $\neg\Box/\Box$-elimination. The labels σ's are left implicit. Every wff set is treated as the conjunction of its elements.

- Assume we start from a (implicit) world σ. KSAT_W plays the role of the DPLL-application rule, generating one by one all the assignments in $\mathcal{M} = \{\mu_1, \ldots, \mu_n\} = DPLL(\varphi)$, each time invocating KSAT_A on μ_j. (Notice that $\models_p \varphi \equiv \bigvee_j \mu_j$.) If $\text{KSAT}_A(\mu_j)$ returns *True*, this means that the branch associated with μ_j is open. Thus KSAT returns *True*. If $\text{KSAT}_A(\mu_j)$ returns *False*, this means that the branch associated with μ_j is closed. Thus KSAT_W looks for the next assignment μ_{j+1}. If no more assignment in $\mathcal{M}$ is available, KSAT returns *False*. Notice that, if φ is propositionally inconsistent (e.g., if it contains a contradiction $\psi \wedge \neg\psi$), then $\mathcal{M} = \emptyset$, and KSAT returns *False*.

[7] Notice that the pure literal rule cannot be added to DPLL, as it generates incomplete assignment sets.

– $\textsc{Ksat}_A$ plays the role of the $\neg\Box/\Box$-elimination rules. For each $\neg\Box\beta_j$ in μ, $\textsc{Ksat}_A$ applies $\neg\Box$-elimination, generating $\neg\beta_j$ in an (implicit) new world σ_j accessible from σ. Then, for every $\Box\alpha_i$ in μ, it applies $\Box$-elimination, adding α_i to σ_j. As a result, $\Gamma_j = \{\alpha_1, \ldots, \alpha_n, \neg\beta_j\}$ holds in σ_j. Γ_j is then expanded by invoking recursively $\textsc{Ksat}$ on $\bigwedge_i \alpha_i \wedge \neg\beta_j$. If $\textsc{Ksat}$ returns *True* for every j, this means that an open K-tableau$_{DPLL}$ has been spanned. Thus $\textsc{Ksat}_A$ returns *True*. If $\textsc{Ksat}$ returns *False* for some j, this means that no open K-tableau$_{DPLL}$ exists. Thus $\textsc{Ksat}_A$ returns *False*.

Notice that using the $\textsc{Ksat}$ control strategy Kripke models are spanned depth-first, each time working on one single world and keeping only parent worlds in the stack. As no confusion can arise between worlds, there is no need to keep labels explicit. See Chapter 2 of [6] for the analogous situation with tableaux.

5 Future work: beyond $\mathcal{N}$

The method used in the previous section to pass from $\mathcal{L}$-tableau's to $\mathcal{L}$-tableau$_f$'s suggests a generalized approach.

Consider a generic logic $\mathcal{L}$, whose semantics gives the standard interpretation to the propositional connectives. Suppose there exists a correct and complete $\mathcal{L}$-tableau framework, given by the following rules:

$$\left\{ (\wedge\text{-elimin.}) \frac{\sigma : \varphi_1 \wedge \sigma : \varphi_2}{\begin{array}{c}\sigma : \varphi_1 \\ \sigma : \varphi_2\end{array}},\ (\vee\text{-elimin.}) \frac{\sigma : \varphi_1 \vee \sigma : \varphi_2}{\sigma : \varphi_1 \qquad \sigma : \varphi_2} \right\} \quad \cup \quad R'.$$

where each rule $r \in R'$ is in the general form

$$(r)\ \frac{\sigma : \psi}{\ldots}$$

where ψ is a *literal*, in the sense defined in Section 3.1.

Intuitively, the rules are subdivided in *purely propositional* (standard Smullyan's rules) and *purely non-propositional* (the rules in R'). As with Fitting's tableaux, our proposal is thus to define $\mathcal{L}$-tableau$_f$ by substituting the propositional rules with the application of a propositional decider f:

$$\left\{ (f\text{-application})\ \frac{\sigma : \varphi}{\sigma : \mu_1 \quad \sigma : \mu_2 \quad \ldots \quad \sigma : \mu_n} \right\} \quad \cup \quad R',$$

being $f(\varphi) = \{\mu_1, \mu_2 \ldots\}$.

Due to the great variety of tableau frameworks available in literature, so far it has not been possible to provide a common proof of correctness/completeness for general tableau$_f$'s. Notice however that in the literature the "hard" parts of the correctness/completeness proofs for tableaux are typically those involving the rules in R'. Thus we believe that, similarly to what happens in the proof in Appendix A, in most cases correctness/completeness proofs for SAT-based frameworks are straightforward variants of the proofs for the corresponding tableaux.

References

1. P. Bresciani, E. Franconi, and S. Tessaris. Implementing and testing expressive Description Logics: a preliminary report. In *Proc. International Workshop on Description Logics*, Rome, Italy, 1995.
2. B. F. Chellas. *Modal Logic – an Introduction.* Cambridge University Press, 1980.
3. M. D'Agostino and M. Mondadori. The Taming of the Cut. *Journal of Logic and Computation*, 4(3):285–319, 1994.
4. M. Davis, G. Longemann, and D. Loveland. A machine program for theorem proving. *Journal of the ACM*, 5(7), 1962.
5. G. DeGiacomo and F. Massacci. Tableaux and Algorithms for Propositional Dynamic Logic with Converse. In *Proc. of the 5th International Conference on Principles of Knowledge Representation and Reasoning - KR'96*, Cambridge, MA, USA, November 1996.
6. M. Fitting. *Proof Methods for Modal and Intuitionistic Logics.* D. Reidel Publishg, 1983.
7. E. Giunchiglia, F. Giunchiglia, R. Sebastiani, and A. Tacchella. More evaluation of decision procedures for modal logics. In *Proc. of the 6th International Conference on Principles of Knowledge Representation and Reasoning - KR'98*, Trento, Italy, November 1997.
8. F. Giunchiglia and R. Sebastiani. Building decision procedures for modal logics from propositional decision procedures - the case study of modal K(m). Technical Report 9611-06, IRST, Trento, Italy, 1996.
9. F. Giunchiglia and R. Sebastiani. Building decision procedures for modal logics from propositional decision procedures - the case study of modal K. In *Proc. of the 13th Conference on Automated Deduction*, Lecture Notes in Artificial Intelligence, New Brunswick, NJ, USA, August 1996. Springer Verlag. Also DIST-Technical Report 96-0037 and IRST-Technical Report 9601-02.
10. F. Giunchiglia and R. Sebastiani. A SAT-based decision procedure for ALC. In *Proc. of the 5th International Conference on Principles of Knowledge Representation and Reasoning - KR'96*, Cambridge, MA, USA, November 1996. Also DIST-Technical Report 9607-08 and IRST-Technical Report 9601-02.
11. U. Hustadt and R.A. Schmidt. On evaluating decision procedures for modal logic. In *Proc. of the 15th International Joint Conference on Artificial Intelligence*, 1997.
12. R. Sebastiani and D. McAllester. New upper bounds for satisfiability in modal logics - the case-study of modal K. Technical Report 9710-15, IRST, Trento, Italy, October 1997.
13. R. M. Smullyan. *First-Order Logic.* Springer-Verlag, NY, 1968.

A Correctness and completeness of $\mathcal{L}$-tableau$_f$

To prove the correctness and completeness of $\mathcal{L}$-tableau$_f$, we follow step by step the equivalent proofs for $\mathcal{L}$-tableau in Chapter 8 of [6], introducing only the slight modifications for handling f-application rules instead of $\wedge/\vee$-elimination rules. As the proofs are mostly identical, we will briefly sum up the parts which are identical, and explain explicitly only the modified parts. To prove the correctness, we introduce a modified version of Lemma 3.1 in [6].

```
Place 1 : φ in the origin;
Repeat
    Choose a not-finished wff occurrence σ : φ as high up in the tree as possible;
    If (φ is not a propositional literal)
    then
        For each open branch θ through the occurrence of σ : φ do
            Case φ of
                non-literal:
                    Find f(φ) = {μ1 ... μn};
                    Split the end of θ into n sub-branches θ1 ... θn;
                    Add each element of μi to θi, for every i;
                □φ1:
                    For each prefix σ' used in θ so that σ' is L-accessible from σ do:
                        add σ' : φ1 to θ;
|[L deontic:]       if (there is no prefix σ' used in θ so that σ' is L-accessible from σ)
|                   then
|                       let (k be the smallest integer so that σ, k : φ1 unrestricted in θ)
|                           add σ, k : φ1 to θ;
                    Add a fresh occurrence of σ : φ at the end of θ;
                ¬□φ1:
                    let (k be the smallest integer so that σ, k : φ1 unrestricted in θ)
                        add σ, k : ¬φ1 to θ;
    Declare σ : φ finished;
Until (all branches closed) or (all labelled wff occurrences are finished);
if (all branches closed)
then
    return φ unsatisfiable;
else
    return φ satisfiable;
```

Fig. 2. Schema of a systematic $\mathcal{L}$-tableau$_f$ procedure. The lines labelled with "|" relate only to deontic logics.

Lemma 5. *Consider a logic $\mathcal{L} \in \mathcal{N}$ and a propositional decider f. Suppose T is a $\mathcal{L}$-satisfiable $\mathcal{L}$-tableau$_f$. Let T' be a $\mathcal{L}$-tableau$_f$ obtained from T by a single application of a $\mathcal{L}$-tableau$_f$ rule R. Then T' is $\mathcal{L}$-satisfiable.*

Proof. If the rule R is an f-application, then T' is $\mathcal{L}$-satisfiable by Theorem 1. If R is a modal rule, the proof is identical to Lemma 3.1 in [6]. Q.E.D.

It follows straightforwardly from Lemma 5 that $\mathcal{L}$-tableau$_f$'s are correct.

To prove the completeness, in Figure 2 we present a systematic $\mathcal{L}$-tableau$_f$-based procedure. This procedure is identical to the procedure described in Chapter 8 of [6], except that for the "non-literal" case which substitutes the $\wedge/\vee$ cases. We introduce then a modified version of the definition of $\mathcal{L}$-downward saturated sets.

Definition 6 $\mathcal{L}$-downward saturated$_f$ set. Given a logic $\mathcal{L} \in \mathcal{N}$ and a propositional decider f, a set S of labeled formulas is **$\mathcal{L}$-downward saturated$_f$** iff:

- No propositional atom A_i is such that $\sigma : A_i \in S$ and $\sigma : \neg A_i \in S$;
- If a non-literal $\sigma : \varphi$ is in S, then there is an assignment μ_i in $f(\varphi)$ so that $\sigma : \mu_i \subseteq S$;
- If $\sigma : \Box\varphi_1 \in S$, then $\sigma' : \varphi_1 \in S$, for all σ' $\mathcal{L}$-accessible from σ; Moreover, if $\mathcal{L}$ is deontic, then $\sigma' : \varphi_1 \in S$, for some σ' $\mathcal{L}$-accessible from σ;
- If $\sigma : \neg\Box\varphi_1 \in S$, then $\sigma' : \neg\varphi_1 \in S$, for some σ' $\mathcal{L}$-accessible from σ;

As above, this definition differs from the analogous in [6] only for the "non-literal" case. We introduce now a modified version of Lemma 6.1 in [6].

Lemma 7. *Consider a logic $\mathcal{L} \in \mathcal{N}$ and a propositional decider f. If S is a $\mathcal{L}$-downward saturated$_f$ set, then S is $\mathcal{L}$-satisfiable in a model whose worlds are simply the prefixes occurring in members of S.*

Proof. Suppose S is $\mathcal{L}$-downward saturated$_f$. As in Lemma 6.1 in [6], the $\mathcal{L}$-model $M =< \mathcal{U}, \pi, \mathcal{R} >$ is built as follows. Let $\mathcal{U}$ be the set of labels occurring in S. For every label pair $\sigma, \sigma' \in \mathcal{U}$, let $\mathcal{R}(\sigma, \sigma')$ hold iff σ' is $\mathcal{L}$-accessible from σ. This states the frame $< \mathcal{U}, \mathcal{R} >$. As the frame involves only worlds and accessibility relations between them, the proof that $< \mathcal{U}, \mathcal{R} >$ is a $\mathcal{L}$-frame is identical to Lemma 6.1 in [6].

We define π so that $\pi(A_i, \sigma) = True$ iff $\sigma : A_i \in S$. By induction on the degree of φ, if $\sigma : \varphi \in S$, then $M, \sigma \models_{\mathcal{L}} \varphi$:

- φ literal: identical to Lemma 6.1 in [6].
- φ non-literal: by Def. 6, there exists $\mu_i \in f(\varphi)$ so that $\sigma : \mu_i \subseteq S$. Thus, by inductive hypothesis on the elements of μ_i, $M, \sigma \models_{\mathcal{L}} \mu_i$. By Theorem 1, $M, \sigma \models_{\mathcal{L}} \varphi$. Q.E.D.

We have consequently the following completeness theorem.

Theorem 8. *Given a logic $\mathcal{L}$, a propositional decider f and a wff set Γ, if Γ is $\mathcal{L}$-unsatisfiable, then there exists a closed $\mathcal{L}$-tableau$_f$ for Γ.*

Proof. Identical to Theorem 6.2 in [6], substituting "$\mathcal{L}$-downward saturated" with "$\mathcal{L}$-downward saturated$_f$", and "Lemma 6.1" with "Lemma 7". Intuitively, the proof in [6] shows that, if there exists no closed $\mathcal{L}$-tableau for Γ, then the procedure will generate an open branch which is a $\mathcal{L}$-downward saturated set – and thus is $\mathcal{L}$-satisfiable – so that Γ is $\mathcal{L}$-satisfiable. Q.E.D.

Merging Lemma 5 and Theorem 8 we finally obtain Theorem 4.

For the issue of decidability, [6] (Chapter 8, Section 7) proposes a slight modification of the procedure mentioned above to ensure termination for every logic $\mathcal{L}$, in particular for these logics, like K4 and S4, requiring loop checking. These arguments hold identically for the procedure in Figure 2.

This article was processed using the LaTeX macro package with LLNCS style