Tableau Calculi for Hybrid Logics

Miroslava Tzakova *

Max-Planck-Institut für Informatik

Abstract. Hybrid logics were proposed in [15] as a way of boosting the expressivity of modal logics via a novel mechanism: adding labels for states in Kripke models and viewing these labels as formulae. In addition, hybrid logics may contain quantifiers to bind the labels. Thus, hybrid logics have both Kripke semantics and a first-order binding apparatus. We present prefixed tableau calculi for weak hybrid logics (proper fragments of classical logic) as well as for hybrid logics having full first-order expressive power, and give a general method for proving completeness. For the weak quantifier-free logics we present a tableau-based decision procedure.

1 Introduction

Hybrid logics are extensions of modal logics with labels for states in Kripke models. The labels and the ordinary propositional symbols are treated in a uniform way to construct formulae of these logics. For example, given a label c, where labels are special sort of formulae, $c \wedge \neg\Diamond p$ is a well-formed formula of the hybrid logics. In addition, hybrid logics allow quantification over the set of states in Kripke models. For example, in the formula $\forall x \neg\Diamond x$, the quantifier $\forall x$ should be read as 'for all states'.

As the examples suggest, hybrid logics have a rather novel syntax and semantics. By viewing labels as formulae, they incorporate both Kripke semantics and first-order binding. Hybrid logics greatly increase the expressivity of modal logics, for example, they can express irreflexivity, the *Until* operator and counting modalities; for a discussion on the relevance of hybrid logics for temporal logic and AI, we refer to [12, 7, 6]. In fact, hybrid logics can be seen as fragments of classical logic ranging from strictly weaker systems to systems having full first-order expressive power (see [3, 4] for a hierarchy).

So far, the work on proof systems for hybrid logics has been mostly concerned with Hilbert-style systems (see, for example, [11, 8, 12, 5, 7, 14]), and a better deductive apparatus can be found only in [17, 18, 2]. The latter papers discuss sequent calculi and natural deduction systems for both weak quantifier-free hybrid languages and systems having full first-order expressive power.

In this paper we investigate tableau proof systems for such languages as well as for a variety of weak languages containing powerful quantifiers. Tableau proof

* Max-Planck-Institut für Informatik, Im Stadtwald, 66123 Saarbrücken, Germany. E-mail: tzakova@mpi-sb.mpg.de. Phone: +49 681 9325 226.

Neil V. Murray (Eds.): TABLEAUX'99, LNAI 1617, pp. 278-292, 1999.

systems have been designed for a variety of modal logics and widely used for proving interpolation and other results (see, for example, [10, 16, 13]). However, tableau methods proved important not only from a theoretical point of view, but also they are nowadays successfully used for automated deduction (for an extensive overview we refer to [9]).

We present tableau systems for weak hybrid languages as well as for very expressive languages containing powerful binders, and thus show that hybrid logics behave proof-theoretically well. Tableau systems are important especially for weak systems with quantifiers for which the only known Hilbert-style axiomatizations contain infinite collections of rules of proof (see [5]). Completeness of the proposed calculi is proved in a uniform way using a systematic-tableau-construction argument. Our systematic procedure is based on the procedures in [19, 10], and constructs certain saturated sets that can be satisfied on *hybrid* Kripke models, namely on models in which labels are true at unique states.

For the weak quantifier-free (and decidable) hybrid languages we give a new tableau-based proof procedure that terminates and thus decides the binder-free languages (Section 5).

2 Hybrid logics

We begin by recalling the syntax of propositional modal logic. Given a denumerably infinite set PROP $= \{p, q, r, \ldots\}$ of *propositional symbols*, the well-formed formulae of propositional modal logic (PML) are defined as follows: $\varphi \ := \ p \mid \neg\varphi \mid \varphi \wedge \psi \mid \Box\varphi$. The dual of the $\Box$ operator is $\Diamond\varphi := \neg\Box\neg\varphi$. Other Boolean operators are defined in the standard way.

To define hybrid logics we extend PML in two steps. First, we add two sets of new symbols: a countably infinite set SVAR $= \{x, y, z, \ldots\}$, called *state variables* and a countably infinite set NOM $= \{c, c_1, \ldots\}$, called *nominals*. (In what follows we assume that PROP, SVAR and NOM are fixed.) Second, we introduce operators. In this paper we consider two binding operators (binders): $\forall$ and $\downarrow$, and the operator @. Both state variables and nominals will be interpreted as singletons and thus, will act as 'labels' for the unique states they are satisfied at. The difference is that whereas state variables can be bound by the binders, nominals cannot. We call PROP $\cup$ SVAR $\cup$ NOM the set of *atoms*. The operator $@_l$ allows us to retrieve the information at the state labeled l.

Let $\mathcal{O} \subseteq \{\forall, \downarrow, @\}$ be a set of operators. We define $\mathcal{L}(\mathcal{O})$, the hybrid language over the operators in $\mathcal{O}$, to be the smallest set of formulae containing: (1) each atom a, (2) $\varphi \wedge \psi$ for each φ and ψ in $\mathcal{L}(\mathcal{O})$, (3) $\neg\varphi$ for each φ in $\mathcal{L}(\mathcal{O})$, (4) $\Box\varphi$ for each φ in $\mathcal{L}(\mathcal{O})$, (5) $Ox\varphi$, for each $x \in$ SVAR and each $\varphi \in \mathcal{L}(\mathcal{O})$ if $O \in \mathcal{O} \cap \{\forall, \downarrow\}$, and (6) $@_l\varphi$, for each $l \in$ SVAR $\cup$ NOM and each $\varphi \in \mathcal{L}(\mathcal{O})$ if $@ \in \mathcal{O}$. (Thus, we have eight hybrid languages: one for each choice of $\mathcal{O}$. Whenever $\mathcal{O}$ is clear from the context, we will write $\mathcal{L}$ instead of $\mathcal{L}(\mathcal{O})$.) We denote the simplest language containing no operator from $\mathcal{O}$ by $\mathcal{L}_o$.

The dual of $\forall$ [6TDfl(and)Tjfl16-25000(con)]Tfl12030TDfl[(taining)-25000(no)-27000(

Note that the definition of the hybrid languages treats *all* atoms as *formulae*. For example, $x \wedge \forall x((p \wedge \Diamond c) \rightarrow \neg\Diamond x)$ is a well-formed formula. Free and bound state variables, substitution and other syntactic concepts, are defined as in classical logic (for definitions see [5]). A formula φ is called a *sentence* iff φ does not contain any free occurrence of a state variable. Given a formula φ and state variables x and y, $\varphi[y/x]$ will denote the formula obtained from φ by substituting y for all free occurrences of x.

Now for the semantics. Let $\mathcal{L}$ be any of the hybrid languages defined above. A (Kripke) model $\mathcal{M}$ for $\mathcal{L}$ is a triple (S, R, V) such that S is a non-empty set of states, R a binary relation on S, called *the accessibility relation*, and $V : \mathrm{PROP} \cup \mathrm{NOM} \longrightarrow Pow(S)$. A valuation V is called *hybrid* iff for all nominals $c \in \mathrm{NOM}$, $V(c)$ is a singleton subset of S. A model $\mathcal{M}$ is called *hybrid* iff its valuation is hybrid. (That is: hybrid models treat nominals as labels.)

To bind state variables we will make use of the Tarskian idea of assignment functions. An assignment for $\mathcal{L}$ on $\mathcal{M}$ is a mapping $g : \mathrm{SVAR} \longrightarrow Pow(S)$ such that for all state variables $x \in \mathrm{SVAR}$, $g(x)$ is a singleton subset of S. (That is: assignments treat state variables a00(of)-fl(2-2000(o)]29(x)TjflT11TfflTjfl940TDflt)]

hybrid models $\mathcal{M}$, all assignments g on $\mathcal{M}$, and all states s in $\mathcal{M}$, $\mathcal{M}, g, s \models \varphi$. We write $\mathcal{M}, s \models \varphi$ iff $\mathcal{M}, g, s \models \varphi$ for all assignments g. Note that for every sentence φ, $\mathcal{M}, g, s \models \varphi$ iff $\mathcal{M}, s \models \varphi$.

Let $\mathcal{L}$ be any of the hybrid languages defined above. The hybrid logic of $\mathcal{L}$ is defined to be the set of all valid $\mathcal{L}$-formulae. Hybrid languages greatly increase the expressivity of PML. For example, the weakest hybrid language $\mathcal{L}_o$ is more expressive than PML, since $\mathcal{L}_o$ can express irreflexivity by $c \rightarrow \neg\Diamond c$. Here are two examples of properties that are not definable in PML while definable in hybrid languages.

Example 1. Counting modalities are definable. For example:

$$\mathit{At\text{-}least\text{-}2}(\varphi) \;:=\; \exists x \exists y(\Diamond(x \wedge \neg y \wedge \varphi) \wedge \Diamond(y \wedge \neg x \wedge \varphi)).$$

$\mathit{At\text{-}least\text{-}2}(\varphi)$ is satisfied at a state s iff φ is satisfied in at least two *distinct* successors of s. Read this definition as follows: it is possible to bind the variables x and y to two states in such a way that x is a successor of s and φ is true but y is false at x, and y is a successor of s and φ is true but x is false at y.

Example 2. Until is definable.

$$\mathit{Until}(\varphi, \psi) \;:=\; \downarrow x \Diamond \downarrow y @_x(\Diamond(y \wedge \varphi) \wedge \Box(\Diamond y \wedge \neg y \rightarrow \psi)).$$

Note how this works: we label the current state with x, use $\Diamond$ to move to an accessible state, which we label y, and then use @ to jump us back to x. We then use the modalities to insist that (1) φ holds at the state labeled y, and (2) ψ holds at all successors of the current state that precede this y-labeled state.

In fact, hybrid languages can be seen as fragments of classical logic ranging from strictly weaker systems, like $\mathcal{L}(\downarrow)$, to systems with full first-order expressive power, like $\mathcal{L}(\forall, @)$. For further discussion on expressivity we refer to [3, 4].

Lemma 1 (Substitution Lemma). *Let $\mathcal{M}$ be a hybrid model, let g be an assignment on $\mathcal{M}$, and let φ be a formula of any of the hybrid languages defined above. Then, for every state s in $\mathcal{M}$, if y is a variable that is substitutable for x in φ and c is a nominal then:*

1. *$\mathcal{M}, g, s \models \varphi[y/x]$ iff $\mathcal{M}, g', s \models \varphi$, where $g' \overset{x}{\sim} g$ and $g'(x) = g(y)$.*
2. *$\mathcal{M}, g, s \models \varphi[c/x]$ iff $\mathcal{M}, g', s \models \varphi$, where $g' \overset{x}{\sim} g$ and $g'(x) = V(c)$.*

Proof. By induction on the complexity of φ.

3 Prefixed tableaux

Prefixed tableau systems have been designed for a variety of modal logics (see [10]). The idea is to use prefixes to 'label' states in Kripke models. In contrast to the branch 'modification' techniques, prefixes allow us to keep the information about the past. This is crucial to the hybrid languages as nominals and state variables should be satisfied at unique states. We follow the notation in [10].

Let $\mathcal{L}$ be any of the hybrid languages defined in Section 2. It suffices to construct tableau proofs for sentences in $\mathcal{L}$, since validity of arbitrary formulae can be reduced to that of sentences. As a consequence, tableau proofs will contain only sentences of $\mathcal{L}$ (and not arbitrary formulae). More precisely, tableau proofs will contain two types of formulae: prefixed sentences and accessibility sentences.

Prefixed sentences in $\mathcal{L}$ (prefixed $\mathcal{L}$-sentences) consist of a prefix followed by an $\mathcal{L}$-sentence. More formally, given a countably infinite set of prefixes $\mathrm{PREF} = \{\tau, \sigma, \sigma', \ldots\}$, a prefixed sentence has the form $\sigma\varphi$, where $\sigma \in \mathrm{PREF}$ and φ is an $\mathcal{L}$-sentence. A prefixed sentence $\sigma\varphi$ is typically read 'φ is true at σ'. We refer to prefixed sentences as *atomic* if they are of the form σa or $\sigma\neg a$, where σ is a prefix and a an atom in $\mathcal{L}$.

Recall that nominals are also used to label states in models. Thus, prefixed sentences contain two kinds of labels: nominals and prefixes. The difference is that nominals label states internally, that is, we refer to them in formulae in the object language, whereas prefixes label externally: they are used in the meta-language to keep the information about all states that have been created in the course of the tableau construction.

Accessibility sentences are of the form $\sigma < \sigma'$, where σ and σ' are prefixes.

As it was mentioned before prefixes will later be interpreted as states in models while accessibility sentences will define pairs of states that are in the accessibility relation.

Let Γ be a set of prefixed sentences and accessibility sentences. We use $\mathrm{PREF}(\Gamma)$ (resp. $\mathrm{NOM}(\Gamma)$ and $\mathrm{PROP}(\Gamma)$) to denote the set of all prefixes (resp. nominals and propositional symbols) that occur in some prefixed or accessibility sentence in Γ.

A *tableau rule in* $\mathcal{L}$ consists of a premiss $\mathcal{P}$ and a (finite) set of conclusions $\mathcal{C}_1, \ldots, \mathcal{C}_n$, where $n \in \omega$: $\frac{\mathcal{P}}{\mathcal{C}_1 \mid \ldots \mid \mathcal{C}_n}$. The *premiss* and the *conclusions* are (finite) sets consisting of accessibility sentences and prefixed $\mathcal{L}$-sentences. The tableau rules can be read as follows: 'if all formulae in the premiss $\mathcal{P}$ of a rule are simultaneously satisfiable then so are all formulae in at least one of the conclusions'.

Let $\mathcal{O} \subseteq \{\forall, \downarrow, @\}$ be some set of operators and $\mathcal{L}$ the hybrid language over $\mathcal{O}$. A *tableau calculus* $\mathcal{TC}(\mathcal{O})$ for $\mathcal{L}$ is a finite collection of tableau rules in $\mathcal{L}$. Tableau calculi for the hybrid languages will be defined in the next section.

The definition of prefixed tableaux that we use is standard and can be found in [10]. Throughout the definition both prefixed $\mathcal{L}$-sentences and accessibility sentences are referred to as sentences.

A $\mathcal{TC}(\mathcal{O})$-*prefixed tableau* for $\sigma\varphi$ is a finite tree with root $\sigma\varphi$ each node of which carries either a prefixed $\mathcal{L}$-sentence or an accessibility sentence. We say that a $\mathcal{TC}(\mathcal{O})$-rule is *applicable* to a branch if the branch contains all sentences that occur in the premiss of the rule. The steps for extending the tableau are: (1) choose a branch θ and a rule ρ that is applicable to θ, (2) if ρ has n conclusions, split the end of θ to n branches and for all $k \leq n$, add $\mathcal{C}_k$ to the k-th branch. (That is, for all $k \leq n$ we add as many nodes to the k-th branch as there are sentences in $\mathcal{C}_k$.) All tableaux are constructed in this way.

The left-hand-side tableau in Example 3 below motivates the following:

Definition 1 (Closed branches and tableaux). *Let $\mathcal{T}$ be a prefixed tableau and θ a branch in $\mathcal{T}$. θ is* closed *if either (1) θ contains both $\sigma\varphi$ and $\sigma\neg\varphi$ or, (2) θ contains the following four prefixed sentences σc, $\sigma' c$, $\sigma\varphi$ and $\sigma'\neg\varphi$, where φ is an $\mathcal{L}$-sentence, σ, σ' prefixes and c a nominal. If a branch is not closed it is* open. *The tableau $\mathcal{T}$ is* closed *if all its branches are closed, otherwise $\mathcal{T}$ is* open.

Let $\mathcal{T}$ be a tableau and θ a branch in $\mathcal{T}$. Define a binary relation $<_\theta$, *the accessibility relation*, on the set $\mathrm{PREF}(\theta)$ as follows: $\sigma <_\theta \sigma'$ iff $\sigma < \sigma'$ is on θ.

Let Γ be a set of prefixed sentences, $<_\Gamma$ a binary relation on $\mathrm{PREF}(\Gamma)$, and $\mathcal{M} = (S, R, V)$ a hybrid model for $\mathcal{L}$. A mapping $\mathcal{I} : \mathrm{PREF} \longrightarrow S$ is called an *interpretation of* $(\Gamma, <_\Gamma)$ if for all prefixes σ and σ' in Γ such that $\sigma <_\Gamma \sigma'$ we have $\mathcal{I}(\sigma) R \mathcal{I}(\sigma')$. In particular, if Γ is the set of all prefixed sentences that occur on a branch θ and $<_\Gamma$ is the accessibility relation $<_\theta$, we say that $\mathcal{I}$ is an *interpretation of θ*.

Definition 2 (Satisfiability of branches). *Let Γ be a set of prefixed sentences and $<_\Gamma$ a binary relation on* $\mathrm{PREF}(\Gamma)$. *Further, let $\mathcal{M}$ be a hybrid model for $\mathcal{L}$ and $\mathcal{I}$ an interpretation of $(\Gamma, <_\Gamma)$. We say that $(\Gamma, <_\Gamma)$ is* satisfiable in $\mathcal{M}$ under $\mathcal{I}$ *if for all $\sigma\varphi \in \Gamma$ we have $\mathcal{M}, \mathcal{I}(\sigma) \models \varphi$. A branch θ* is satisfiable in $\mathcal{M}$ under $\mathcal{I}$ *if $(\Gamma, <_\Gamma)$ is satisfiable in $\mathcal{M}$ under $\mathcal{I}$, where Γ is the set of all prefixed sentences that occur on θ and $<_\Gamma$ is the accessibility relation $<_\theta$. $(\Gamma, <_\Gamma)$ is* satisfiable *if there is a hybrid model $\mathcal{M}$ and an interpretation $\mathcal{I}$ such that $(\Gamma, <_\Gamma)$ is satisfiable in $\mathcal{M}$ under $\mathcal{I}$. A branch θ is* satisfiable *if there is a model $\mathcal{M}$ and an interpretation $\mathcal{I}$ such that θ is satisfiable in $\mathcal{M}$ under $\mathcal{I}$.*

It is easy to see that a closed branch θ cannot be satisfiable. A tableau is called *satisfiable* if it has a satisfiable branch. Let $\mathcal{O} \subseteq \{\forall, \downarrow, @\}$ be a set of operators, $\mathcal{L}$ the hybrid language over $\mathcal{O}$ and φ an $\mathcal{L}$-sentence. We say that φ is *provable* in a tableau calculus $\mathcal{TC}(\mathcal{O})$ iff there is a closed tableau for $\sigma\neg\varphi$ where σ is some prefix. In this case, the tableau is called a proof of φ in $\mathcal{TC}(\mathcal{O})$.

4 Tableau calculi for hybrid logics

In this section we define tableau calculi for hybrid languages and prove them complete. The collection of rules $\mathcal{TC}$ below defines a tableau calculus for the weakest hybrid language $\mathcal{L}_o$ that contains no operators from $\{\forall, \downarrow, @\}$. We use c and c_1 to denote nominals, and σ, τ and σ' to denote prefixes.

(α) $\dfrac{\sigma\neg\neg\varphi}{\sigma\varphi}$; $\dfrac{\sigma\varphi\wedge\psi}{\sigma\varphi,\ \sigma\psi}$ $\qquad$ (β) $\dfrac{\sigma\neg(\varphi\wedge\psi)}{\sigma\neg\varphi \mid \sigma\neg\psi}$ $\qquad$ (ν) $\dfrac{\sigma\Box\varphi,\ \sigma{<}\sigma'}{\sigma'\varphi}$

(π) $\dfrac{\sigma\neg\Box\varphi}{\sigma'\neg\varphi,\ \sigma{<}\sigma'}$ σ' is not on the branch

$(\mathit{Labeling})$ $\dfrac{\sigma\varphi}{\sigma c}$ c is not on the branch

$(\mathit{S\text{-}Identifying})$ $\dfrac{\sigma c,\ \tau c,\ \tau{<}\sigma'}{\sigma{<}\sigma'}$

$(\mathit{L\text{-}Identifying})$ $\dfrac{\sigma c,\ \tau c,\ \sigma_1 c_1,\ \tau c_1}{\sigma c_1,\ \sigma_1 c}$

The rules α, β, ν and π are known from tableau calculi for modal logics and apply to hybrid languages as well. The *Labeling* and both *Identifying* rules reflect the hybrid languages. The *Labeling* rule says that whenever we have an external label σ for a state, we can introduce an internal label c for that state. The *S-Identifying* rule allows us to identify the successors of two states if the two states have a common internal label. More precisely, if two states are internally labelled by a label c then, we identify the successors of the first state with those of the second. The *L-Identifying* rule says that we can identify the internal labels of a state σ with those of a state σ_1, if σ and σ_1 share a comon external label, namely τ. The rules for the operators $\forall, \downarrow$ and @ are defined as follows:

$$(\forall 1)\quad \frac{\sigma \forall x\varphi}{\sigma\varphi[c/x]} \qquad c \text{ is on the branch}$$

$$(\forall 2)\quad \frac{\sigma \neg\forall x\varphi}{\sigma\neg\varphi[c/x]} \qquad c \text{ is not on the branch}$$

$$(\downarrow 1)\quad \frac{\sigma \downarrow x\varphi,\ \sigma c}{\sigma\varphi[c/x]} \qquad (\downarrow 2)\quad \frac{\sigma \neg\downarrow x\varphi,\ \sigma c}{\sigma\neg\varphi[c/x]}$$

$$(@1)\quad \frac{\sigma @_c\varphi}{\sigma' c,\ \sigma'\varphi} \qquad (@2)\quad \frac{\sigma \neg@_c\varphi}{\sigma' c,\ \sigma'\neg\varphi}$$

(In the @-rules above, if σc is on the branch, $\sigma' = \sigma$, else σ' is not on the branch.) Let $\mathcal{O} \subseteq \{\forall, \downarrow, @\}$ be any set of operators and $\mathcal{L}$ be the hybrid language over $\mathcal{O}$. We define a *tableau calculus* $\mathcal{TC}(\mathcal{O})$ *for* $\mathcal{L}$ to be the following collection of rules in $\mathcal{L}$: $\mathcal{TC} \cup \{O\text{-rules} \mid O \in \mathcal{O}\}$. (For example, $\mathcal{TC} \cup \{\downarrow 1, \downarrow 2\} \cup \{@1, @2\}$ is the tableau calculus for $\mathcal{L}(\downarrow, @)$.) One can see that if a single tableau rule is applied to a satisfiable tableau, the resulting tableau will be satisfiable too. Therefore, $\mathcal{TC}(\mathcal{O})$ is sound: if an $\mathcal{L}$-sentence φ is provable in $\mathcal{TC}(\mathcal{O})$, then φ is valid.

Before turning to completeness, we consider three examples of tableau proofs. To simplify the presentation, we use finite sequences of natural numbers as prefixes, and assume that two prefixes $\sigma = k_1 \ldots k_i$ and $\tau = l_1 \ldots l_j$ are in the accessibility relation of a branch θ iff either $j = i+1$ and for all $m \leq i$, $k_m = l_m$ or, θ contains $\sigma < \tau$.

Example 3. The tableau on the left-hand-side below is closed because of Definition 1. The tableau proof on the right-hand-side below uses the *Labeling* rule.

$1.\neg(\Diamond\Diamond(c \wedge \varphi) \to \Box(c \to \varphi))$
$1.\Diamond\Diamond(c \wedge \varphi)$
$1.\neg\Box(c \to \varphi)$
$1.1.\Diamond(c \wedge \varphi)$
$1.2.\neg(c \to \varphi)$
$1.1.1.c$
$1.1.1.\varphi$
$1.2.c$
$1.2.\neg\varphi$
$\bot$

$1.\neg(\Diamond p \to \downarrow x\Diamond\downarrow y@_x@_y p)$	
$1.\Diamond p$	$1, \alpha$
$1.\neg\downarrow x\Diamond\downarrow y@_x@_y p$	$1, \alpha$
$1.1.p$	$2, \pi$
$1.c_1$	*Labeling*
$1.\neg\Diamond\downarrow y@_{c_1}@_y p$	$3, \downarrow 2$
$1.1.\neg\downarrow y@_{c_1}@_y p$	$6, \nu$
$1.1.c_{11}$	*Labeling*
$1.1.\neg@_{c_1}@_{c_{11}}p$	$7, \downarrow 2$
$1.\neg@_{c_{11}}p$	$9, @2$
$1.1.\neg p$	$10, @2$
$\bot$	

Example 4. The following tableau proof uses the *S-Identifying* rule.

$1.\neg(\Diamond(c \wedge \Box(c_1 \rightarrow p)) \rightarrow \neg\Diamond(c \wedge \Diamond(c_1 \wedge \neg p)))$	
$1.\Diamond(c \wedge \Box(c_1 \rightarrow p))$	$1, \alpha$
$1.\Diamond(c \wedge \Diamond(c_1 \wedge \neg p))$	$1, \alpha$
$1.1.c$	$2, \pi, \alpha$
$1.1.\Box(c_1 \rightarrow p)$	$2, \pi, \alpha$
$1.2.c$	$3, \pi, \alpha$
$1.2.\Diamond(c_1 \wedge \neg p)$	$3, \pi, \alpha$
$1.2.1.c_1$	$7, \pi, \alpha$
$1.2.1.\neg p$	$7, \pi, \alpha$
$1.1 < 1.2.1$	$4, 6,$ *S-Identifying*
$1.2.1.c_1 \rightarrow p$	$5, 10, \nu$
$1.2.1.\neg c_1$ $\quad$ $1.2.1.p$	$11, \beta, \alpha$
$\bot$ $\quad$ $\bot$	

4.1 A proof procedure

To prove that the tableau calculi defined above are complete we will use a systematic-tableau-construction argument (see [19, 10]). Given a sentence φ, we will describe a systematic procedure such that if φ is valid, the procedure on input $\neg\varphi$ will construct a tableau proof for φ, otherwise, the procedure will construct a counter-model for φ. Note that in the case of hybrid logics the counter-model should be a hybrid (Kripke) model.

The procedure will closely follow the systematic procedures in [19, 10]. We will work with each occurrence of a prefixed sentence in a tableau exactly once, after which we declare this occurrence finished. We fix an enumeration $E_{nom} = \{c_1, c_2, \ldots\}$ of all nominals in NOM and an enumeration $E_{pref} = \{\sigma_1, \sigma_2, \ldots\}$ of all prefixes in PREF.

The proof procedure. On input $\chi = \neg\varphi$ perform the following steps:

Step 1. Place $\sigma_1\neg\varphi$ at the origin and then add $\sigma_1 c$. (Here c is the first nominal in E_{nom} that is not in χ.) This completes *Step 1.*

Suppose that n steps of the procedure have been completed and the tableau that has been constructed is $\mathcal{T}_n$. Denote by (a) the condition that all occurrences of prefixed sentences in $\mathcal{T}_n$ are finished and by (b) the condition that there is an open branch θ in $\mathcal{T}_n$ such that for all prefixes σ and τ on θ if θ contains σc and τc for some nominal c then, σ and τ have the same $<_\theta$-successors, and for all prefixes σ, τ and σ_1 on θ, if for some nominals c and c_1, $\sigma c, \tau c, \sigma_1 c_1, \tau c_1$ are on θ, then $\sigma c_1, \sigma_1 c$ are on θ. Then, if either $\mathcal{T}_n$ is closed, or both (a) and (b) hold, then stop. Otherwise perform the next step.

Step n+1. This step consists of two substeps, namely of (A) and (B):

(A) For all open branches θ and:

(1) for all prefixes σ and τ on θ, if σc and τc on θ, where c is a nominal then, identify $<_\theta$-successors of σ with $<_\theta$-successors of τ by adding to the end of θ accessibility sentences as follows: for all τ' such that $\tau <_\theta \tau'$ add $\sigma < \tau'$, and for

all σ' such that $\sigma <_\theta \sigma'$ add $\tau <_\theta \sigma'$.
(2) for all prefixes σ, τ and σ_1 on θ; if $\sigma c, \tau c, \sigma_1 c_1, \tau c_1$ are on θ where c and c_1 are nominals, add $\sigma c_1, \sigma_1 c$ to the end of θ.

Having done this for all open branches θ, all nominals and all prefixes on θ, go on to:

(B) Choose an occurrence of a prefixed sentence as high up in the tree as possible (as close to the origin as possible) that has not been finished, say this is $\sigma\gamma$. If $\sigma\gamma$ is atomic, declare $\sigma\gamma$ finished and complete *Step n+1.*

Otherwise, extend the tableau as follows. For each open branch θ through this occurrence of $\sigma\gamma$, do the following:
(1) If $\sigma\gamma$ is $\sigma\neg\neg\psi$ (resp. $\sigma\psi_1 \wedge \psi_2$), add $\sigma\psi$ (resp. $\sigma\psi_1$ and $\sigma\psi_2$) to θ.
(2) If $\sigma\gamma$ is $\sigma\neg(\psi_1 \wedge \psi_2)$, split θ to two branches and add $\sigma\neg\psi_1$ to the one branch and $\sigma\neg\psi_2$ to the other.
(3) If $\sigma\gamma$ is of the form $\sigma\Box\psi$, then for each prefix σ' such that σ' appears on θ and $\sigma < \sigma'$, add $\sigma'\psi$ to θ, after which add a fresh occurrence of $\sigma\Box\psi$ to θ.
(4) If $\sigma\gamma$ is of the form $\sigma\neg\Box\psi$, then add $\sigma'\neg\psi$, $\sigma < \sigma'$ and $\sigma' c$ to θ. (Here σ' is the first prefix in the enumeration E_{pref} that is not on θ and c is the first nominal in the enumeration E_{nom} that is not on θ.)
(5) If $\sigma\gamma$ is of the form $\sigma\forall x\psi$, then for all nominals c on θ, add $\sigma\psi[c/x]$ to θ, followed by a fresh occurrence of $\sigma\forall x\psi$.
(6) If $\sigma\gamma$ is of the form $\sigma\neg\forall x\psi$, then add $\sigma\neg\psi[c/x]$ to θ, where c is the first nominal in the enumeration E_{nom} that is not on θ.
(7) If $\sigma\gamma$ is $\sigma\downarrow x\psi$, add $\sigma\psi[c/x]$ to θ. Here c is a nominal such that σc is on θ.
(8) If $\sigma\gamma$ is $\sigma\neg\downarrow x\psi$, add $\sigma\neg\psi[c/x]$ to θ. Here c is a nominal such that σc is on θ.
(9) If $\sigma\gamma$ is $\sigma@_c\psi$, then add $\sigma'\psi$ and $\sigma' c$ to θ, where $\sigma' = \sigma$ if σc is on θ, and σ' is the first prefix in the enumeration E_{pref} that is not on θ otherwise.
(10) If $\sigma\gamma$ is $\sigma\neg@_c\psi$, then add $\sigma'\neg\psi$ and $\sigma' c$ to θ, where $\sigma' = \sigma$ if σc is on θ, and σ' is the first prefix in the enumeration E_{pref} that is not on θ otherwise.

Having done this for all open branches through the current occurrence of $\sigma\gamma$, declare this occurrence finished. This completes *Step n+1.*
End of the procedure.

Clearly, the procedure does not always terminate; for example, on input $\chi = \Diamond(c \wedge \Diamond c \wedge \Box\Diamond c)$. Termination will be discussed at the end of Section 4 as well as in Section 5.

4.2 Completeness

Note that if the procedure on input $\chi = \neg\varphi$ does not produce a closed tableau for $\neg\varphi$ then, it constructs an open branch θ having the properties listed in the following definition. More precisely, if Γ is the set of all prefixed sentences on θ and $<_\Gamma$ is the accessibility relation on θ, then $(\Gamma, <_\Gamma)$ is downward saturated.

Definition 3 (Downward saturated sets). *Let $\mathcal{O} \subseteq \{\forall, \downarrow, @\}$ be a set of operators and $\mathcal{L}$ be the hybrid language over $\mathcal{O}$. Further, let Γ be a non-empty set of prefixed sentences in $\mathcal{L}$ and $<_\Gamma$ a binary relation on the prefixes in Γ. $(\Gamma, <_\Gamma)$ is called* $\mathcal{O}$-downward saturated *iff the following properties from the list*

below hold: for all $n \leq 7$, property (n) and for every operator $O \in \mathcal{O}$, properties (O') and (O'').

(1) *there is no atom a and no prefix σ in Γ such that both $\sigma a \in \Gamma$ and $\sigma\neg a \in \Gamma$; and, there are no prefixes σ and τ in Γ such that for some $c \in \mathrm{NOM}(\Gamma)$ and some atom a, Γ contains $\sigma c, \tau c, \sigma\neg a, \tau a$.*
(2) *for all nominals c and c_1 and all prefixes σ, τ and σ_1, if $\sigma c, \tau c, \sigma_1 c_1, \tau c_1 \in \Gamma$ then, $\sigma_1 c, \sigma c_1 \in \Gamma$*
(3) *for all nominals c and for all prefixes σ and τ, if $\sigma c, \tau c \in \Gamma$ then, σ and τ have the same $<_\Gamma$-successors*
(4) *if $\sigma\varphi \wedge \psi \in \Gamma$ (resp. $\sigma\neg\neg\varphi \in \Gamma$) then, $\sigma\varphi \in \Gamma$ and $\sigma\psi \in \Gamma$ (resp. $\sigma\varphi \in \Gamma$)*
(5) *if $\sigma\neg(\varphi \wedge \psi) \in \Gamma$ then, either $\sigma\neg\varphi \in \Gamma$ or $\sigma\neg\psi \in \Gamma$*
(6) *if $\sigma\Box\varphi \in \Gamma$ then, for all σ' in Γ, $\sigma <_\Gamma \sigma'$ implies $\sigma'\varphi \in \Gamma$*
(7) *if $\sigma\neg\Box\varphi \in \Gamma$ then, $\sigma'\neg\varphi \in \Gamma$ for some σ' in Γ such that $\sigma <_\Gamma \sigma'$*
($\forall'$) *if $\sigma\forall x\varphi \in \Gamma$, then $\sigma\varphi[c/x] \in \Gamma$ for every nominal c in Γ*
($\forall''$) *if $\sigma\neg\forall x\varphi \in \Gamma$, then $\sigma\neg\varphi[c/x] \in \Gamma$ for some nominal c in Γ*
($\downarrow'$) *if $\sigma\downarrow x\varphi \in \Gamma$, then $\sigma c, \sigma\varphi[c/x] \in \Gamma$ for some nominal c in Γ*
($\downarrow''$) *if $\sigma\neg\downarrow x\psi \in \Gamma$, then $\sigma c, \sigma\neg\varphi[c/x] \in \Gamma$ for some nominal c in Γ*
($@'$) *if $\sigma@_c\varphi \in \Gamma$ where c is a nominal, then $\sigma' c, \sigma'\varphi \in \Gamma$ for some prefix σ'*
($@''$) *if $\sigma\neg@_c\varphi \in \Gamma$ where c is a nominal, then $\sigma' c, \sigma'\neg\varphi \in \Gamma$ for some prefix σ'*

To cope with the binders when proving satisfiability of downward saturated sets we need the concept of a labeled model.

Definition 4 (Labeled Models). *Let $\mathcal{M}$ be a hybrid model in a hybrid language $\mathcal{L}$. We say that $\mathcal{M}$ is* labeled *if for all states $s \in \mathcal{M}$, there is a nominal c such that $\mathcal{M}, s \models c$.*

To state the next lemma we fix notation: a set of prefixed sentences Γ is called *labeled* iff for all prefixes σ in Γ, there is a nominal c such that σc is in Γ.

Lemma 2 (Satisfiability of Downward saturated sets). *Let $\mathcal{O} \subseteq \{\forall, \downarrow, @\}$ be a set of operators and $\mathcal{L}$ be the hybrid language over $\mathcal{O}$. Further, let Γ be a set of prefixed sentences in $\mathcal{L}$ and $<_\Gamma$ a binary relation on the prefixes in Γ. If $(\Gamma, <_\Gamma)$ is $\mathcal{O}$-downward saturated and Γ is labeled, then $(\Gamma, <_\Gamma)$ is satisfiable.*

Proof. Let $\sim$ be a binary relation on the set $\mathrm{PREF}(\Gamma)$ defined as follows: $\sigma \sim \tau$ iff there is a nominal c such that $\sigma c, \tau c$ are in Γ. Since Γ is labeled and (2) of Definition 3 holds, $\sim$ is an equivalence relation on $\mathrm{PREF}(\Gamma)$; let S be the set of the equivalence classes. Define $f : \mathrm{PREF}(\Gamma) \longrightarrow S$ to be a function that maps each prefix $\sigma \in \mathrm{PREF}(\Gamma)$ to its equivalence class.

Our model $\mathcal{M}$ is a triple (S^+, R, V), where $S^+ = S$ if for all $c \in \mathrm{NOM}(\Gamma)$ there is σ such the $\sigma c \in \Gamma$, and $S^+ = S \cup \{*\}$ otherwise. (Here $*$ is an entity that is not an equivalence class.) For every two states $s_1, s_2 \in S^+$, $s_1 R s_2$ iff there are $\sigma_1, \sigma_2 \in \mathrm{PREF}(\Gamma)$ such that $\sigma_1 <_\Gamma \sigma_2$, $f(\sigma_1) = s_1$ and $f(\sigma_2) = s_2$; for all $p \in \mathrm{PROP}(\Gamma)$, $V(p) = \{s \mid \exists\sigma : \ f(\sigma) = s \ \&\ \sigma p \in \Gamma\}$, for all $c \in \mathrm{NOM}(\Gamma)$, $V(c) = \{s \mid \exists\sigma : \ f(\sigma) = s \ \&\ \sigma c \in \Gamma\}$ if this set is non-empty and $V(c) = \{*\}$

otherwise; for all $p \in \mathrm{PROP}$ such that $p \notin \mathrm{PROP}(\Gamma)$ and all $c \in \mathrm{NOM}$ such that $c \notin \mathrm{NOM}(\Gamma)$, $V(p) = V(c) = \{*\}$.

First, note that $\mathcal{M}$ is a hybrid model, that is, for all $c \in \mathrm{NOM}$, $V(c)$ is a singleton set. If $c \notin \mathrm{NOM}(\Gamma)$ this is obvious, so let $c \in \mathrm{NOM}(\Gamma)$ and suppose that there are $s_1, s_2 \in S^+$ and $\{s_1, s_2\} \subseteq V(c)$. Then, there exist σ_1, σ_2 such that $f(\sigma_1) = s_1, f(\sigma_2) = s_2, \sigma_1 c \in \Gamma$ and $\sigma_2 c \in \Gamma$, hence $s_1 = s_2$.

Second, $\mathcal{M}$ is a labeled model, as the set Γ is labeled and if $\mathcal{M}$ contains the state $*$ then, $\mathcal{M}, * \models c$ for some nominal c in Γ.

Third, consider an interpretation $\mathcal{I} : \mathrm{PREF} \longrightarrow S^+$ which is an arbitrary extension of the function $f : \mathrm{PREF}(\Gamma) \longrightarrow S^+$ to PREF. To show that $(\Gamma, <_\Gamma)$ is satisfiable in $\mathcal{M}$ under $\mathcal{I}$, we will prove by induction on the complexity of φ, that for every prefixed sentence $\sigma\varphi$, if $\sigma\varphi \in \Gamma$ then $\mathcal{M}, f(\sigma) \models \varphi$.

The base case, that is, for φ an atom a or $\neg a$, follows from the definition of $\mathcal{M}$ and property (1) of Definition 3.

Now, assume that $\sigma\varphi \in \Gamma$ and for all prefixed sentences $\sigma\psi \in \Gamma$ if ψ has lower complexity than φ then $\mathcal{M}, f(\sigma) \models \psi$. We consider different cases for φ.

If φ is $\neg\neg\psi$, $\psi \wedge \chi$, $\neg(\psi \wedge \chi)$ or $\neg\Box\psi$ we use (4), (5) and (7) of Definition 3.

Let φ be $\Box\psi$ and $\sigma\varphi \in \Gamma$. We have to prove that for all $s \in S^+$, $f(\sigma)Rs$ implies $\mathcal{M}, s \models \psi$. So, suppose that $f(\sigma)Rs$ for some $s \in S^+$. Note that, by the definition of $\mathcal{M}$, $f(\sigma) \in S$ and, since $*$ is not a successor of any state in S, s cannot be $*$, and so $s \in S$ too.

Then, there are σ_1, σ_2 such that $\sigma_1 <_\Gamma \sigma_2$, $f(\sigma_1) = f(\sigma)$ and $f(\sigma_2) = s$. Hence $\sigma <_\Gamma \sigma_2$. (If $\sigma \neq \sigma_1$, this follows from the fact that, since $f(\sigma_1) = f(\sigma)$, there is $c \in \mathrm{NOM}(\Gamma)$ such that $\sigma_1 c, \sigma c \in \Gamma$, and by (3) of Definition 3, σ_1 and σ have the same set of successors.) Now, as $\sigma\Box\psi \in \Gamma$ and $\sigma <_\Gamma \sigma_2$, by (6) of Definition 3, $\sigma_2\psi \in \Gamma$. By the inductive hypothesis, $\mathcal{M}, f(\sigma_2) \models \psi$ and hence equivalently, $\mathcal{M}, s \models \psi$. We have shown that for all states $s \in S^+$, $f(\sigma)Rs$ implies $\mathcal{M}, s \models \psi$, and hence $\mathcal{M}, f(\sigma) \models \Box\psi$.

Let φ be $\forall x\psi$ and $\sigma\varphi \in \Gamma$. We have to show that $\mathcal{M}, f(\sigma) \models \forall x\psi$, that is, for all states $s \in S^+$, if g is an assignment such that $g(x) = \{s\}$ then, $\mathcal{M}, g, f(\sigma) \models \psi$. Since $\mathcal{M}$ is a labeled model, s is labeled by some nominal c. Moreover, as $\sigma\forall x\psi \in \Gamma$ and $(\Gamma, <_\Gamma)$ is downward saturated (see $(\forall')$ of Definition 3), $\sigma\psi[c/x] \in \Gamma$ and hence, by the inductive hypothesis, $\mathcal{M}, f(\sigma) \models \psi[c/x]$. Therefore, by Substitution Lemma, $\mathcal{M}, g, f(\sigma) \models \psi$ for all $s \in S^+$.

Let φ be $\downarrow x\psi$ and $\sigma\varphi \in \Gamma$. We have to show that $\mathcal{M}, f(\sigma) \models \downarrow x\psi$, that is $\mathcal{M}, g, f(\sigma) \models \psi$ where g is an assignment such that $g(x) = \{f(\sigma)\}$. As $(\Gamma, <_\Gamma)$ is downward saturated (see $(\downarrow')$ of Definition 3) and $\sigma\downarrow x\psi \in \Gamma$, for some nominal c in Γ we have $\sigma c, \sigma\psi[c/x] \in \Gamma$. Then, by the inductive hypothesis, $\mathcal{M}, f(\sigma) \models c$ and $\mathcal{M}, f(\sigma) \models \psi[c/x]$. Hence $V(c) = \{f(\sigma)\}$. Therefore, by Substitution Lemma, $\mathcal{M}, g, f(\sigma) \models \psi$.

Suppose that φ is $@_c\psi$ and $\sigma\varphi \in \Gamma$. We have to show that $\mathcal{M}, f(\sigma) \models @_c\psi$. Since $\sigma@_c\psi \in \Gamma$, by $(@')$ of Definition 3, there is σ' such that $\sigma' c, \sigma'\psi \in \Gamma$. Hence, by the inductive hypothesis, $\mathcal{M}, f(\sigma') \models c$ and $\mathcal{M}, f(\sigma') \models \psi$, and therefore $\mathcal{M}, f(\sigma') \models @_c\psi$.

The cases when φ is $\neg\forall x\psi$, $\neg\downarrow x\psi$ and $\neg@_c\psi$ can be proved similarly using $(\forall'')$, $(\downarrow'')$ and $(@'')$ of Definition 3.

Theorem 1 (Completeness). *Let $\mathcal{O} \subseteq \{\forall, \downarrow, @\}$ be a set of operators and $\mathcal{L}$ be the hybrid language over $\mathcal{O}$. If an $\mathcal{L}$-sentence φ is valid, then φ has a systematic tableau proof in $\mathcal{TC}(\mathcal{O})$.*

Proof. The proof is standard. If φ is not provable, a systematic attempt at proving φ will produce an open branch θ, such that the set Γ of all prefixed sentences on θ will be labeled and $(\Gamma, <_\theta)$ will be $\mathcal{O}$-downward saturated. Hence, by Lemma 2, $(\Gamma, <_\theta)$ will be satisfiable, and therefore φ cannot be valid.

Soundness and completeness imply that the systematic procedure is a proof procedure for the hybrid languages:

Corollary 1. *Let $\mathcal{O} \subseteq \{\forall, \downarrow, @\}$ be a set of operators and $\mathcal{L}$ be the hybrid language over $\mathcal{O}$. Then, for every $\mathcal{L}$-sentence φ, if φ is provable in $\mathcal{TC}(\mathcal{O})$, then φ has a systematic tableau proof in $\mathcal{TC}(\mathcal{O})$.*

Any hybrid language that contains either the binder $\forall$ or the binder $\downarrow$ is undecidable (see [3, 4]).[1] There are only two binder-free languages: $\mathcal{L}_o$ and $\mathcal{L}(@)$. Both languages are decidable, since they can be embedded into the guarded fragment defined in [1]. However, the proof procedure we introduced in Section 4.1 does not terminate for these two decidable languages; $\chi = \Diamond(c \wedge \Diamond c \wedge \Box\Diamond c)$ is an example of non-termination. The reason for the non-termination here is the occurrence of an infinite alternation of the π and the *S-Identifying* rule.

In the next section, we define a procedure that uses a new rule called the *S-Identifying'* rule instead of the previously introduced *S-Identifying* rule. Moreover, we allow applying the π rule only with proviso. As a result, for the new procedure we will be able to prove termination and moreover, that it is in fact a decision procedure for $\mathcal{L}_o$ and $\mathcal{L}(@)$.

5 A terminating proof procedure

The proof procedure we define in this section will use only the following rules: α, β, ν, π, *L-Identifying* and @-rules from Section 4, and in addition the rule:

$$(\textit{S-Identifying}') \quad \frac{\sigma c,\ \tau c,\ \sigma\Box\varphi \quad \tau{<}\tau'}{\tau'\varphi}$$

(Here c is a nominal and σ, τ and τ' are prefixes.) Obviously, the latter rule is correct.

In the tableau construction below when the π-rule is applied, the prefix that is introduced is new to the entire tableau. As a consequence, for each prefix τ in the tableau there is a unique sequence of prefixes $\tau_1, \ldots, \tau_n$ such that $\tau_n = \tau$,

[1] Decidable hybrid logics containing binders exist and can be obtained by restricting the classes of models (see [4]).

for each $i < n$, $\tau_i < \tau_{i+1}$ is on the tableau and, there is no prefix σ such that $\sigma < \tau_1$ is on the tableau. We call n the *depth of the prefix* τ. As before we will work with each occurrence of a prefixed sentence that is not of the form $\sigma\Box\varphi$ exactly once, after which we declare this occurrence finished. No prefixed sentence should be added to a branch if the sentence is already on the branch. Let $E_{pref} = \{\sigma_1, \sigma_2, \ldots\}$ and $E_{nom} = \{c_1, c_2, \ldots\}$ be enumerations of respectively all prefixes in PREF and all nominals in NOM.

The terminating procedure. On input χ perform the following steps:

Step 1. Place $\sigma_1\chi$ at the origin and add $\sigma_1 c$, where c is the first nominal in E_{nom} that is not in χ. While there is an unfinished occurrence of a prefixed sentence $\sigma\varphi$ that is not of the form $\tau\neg\Box\gamma$ and $\tau\Box\gamma$, do the following:
For all branches θ throughout $\sigma\varphi$, if φ is:

– atomic, do not extend θ;

– $\neg\neg\psi$ (resp. $\psi_1 \wedge \psi_2$), add $\sigma\psi$ (resp. $\sigma\psi_1$ and $\sigma\psi_2$) to θ;

– $\neg(\psi_1 \wedge \psi_2)$, split θ to two branches and add $\sigma\neg\psi_1$ and $\sigma\neg\psi_2$ respectively to the first and to the second branch;

– $@_c\psi$ (resp. $\neg@_c\psi$) then, if there is a prefix τ such that τc is on θ, add $\tau\psi$ (resp. $\tau\neg\psi$) to θ; else, add τc and $\tau\psi$ (resp. τc and $\tau\neg\psi$) to θ; (Here τ is the first prefix in the enumeration E_{pref} that is not on the tableau.)

Declare $\sigma\varphi$ finished.
For all open branches θ, if c and c_1 are nominals and σ, τ and σ_1 prefixes such that $\sigma c, \tau c, \sigma_1 c_1, \tau c_1$ are on θ, add $\sigma_1 c, \sigma c_1$ to θ, and declare them finished.

Complete *Step 1.* Denote the tableau that has been constructed by $\mathcal{T}_1$.

Suppose that n steps of the procedure have been completed, and the tableau that has been constructed is denoted by $\mathcal{T}_n$. If $\mathcal{T}_n$ is a closed tableau or all occurrences of prefixed sentences of the form $\tau\neg\Box\gamma$ are finished, stop. Otherwise:

Step $n+1$. Consider the following three conditions:
(A) There is an unfinished occurrence of a prefixed sentence that is not of the form $\tau\Box\gamma$ or $\tau'\neg\Box\gamma$, where τ' has depth $n+1$;
(B) There is a branch θ, a prefixed sentence $\sigma\Box\varphi$ and a prefix σ' such that both $\sigma\Box\varphi$ and $\sigma < \sigma'$ are on θ, but $\sigma'\varphi$ is not on θ;
(C) There is a branch θ such that the *S-Identifying'* rule is applicable to θ. (That is, there is a branch θ such that θ contains σc, τc, $\sigma\Box\varphi$, and $\tau < \tau'$ and θ does not contain $\tau'\varphi$.)
(D) There is a branch θ such that for some nominals c and c_1 and some prefixes σ, τ and σ_1, $\sigma c, \tau c, \sigma_1 c_1, \tau c_1$ are on θ but $\sigma_1 c, \sigma c_1$ are not on θ.

While at least one of (A), (B), (C) or (D) holds do the following:
(1) If $\sigma\varphi$ is an unfinished occurrence of a prefixed sentence such that $\sigma\varphi$ is not of the form $\tau\Box\gamma$ and $\tau'\neg\Box\gamma$, where τ' has depth $n+1$ then, for all branches θ throughout $\sigma\varphi$, if φ is of the form:

– atomic, do not extend θ;

– $\neg\neg\psi$ (resp. $\psi_1 \wedge \psi_2$), add $\sigma\psi$ (resp. $\sigma\psi_1$ and $\sigma\psi_2$) to θ;

– $\neg(\psi_1 \wedge \psi_2)$, split θ to two branches and add $\sigma\neg\psi_1$ and $\sigma\neg\psi_2$ respectively to the first and to the second branch;

– $\neg\Box\psi$ and there are no τ and c such that σc, τc and $\tau\neg\Box\psi$ are on θ and

$\tau\neg\Box\psi$ is finished, add $\sigma < \sigma'$, $\sigma'\neg\psi$ and $\sigma' c$ to θ. (Here σ' is the first prefix in E_{pref} that is not on the tableau and c is the first nominal in E_{nom} that is not on the tableau.)

– $@_c\psi$ (resp. $\neg@_c\psi$) then, if there is a prefix τ such that τc is on θ, add $\tau\psi$ (resp. $\tau\neg\psi$) to θ; else, add $\tau' c$ and $\tau'\psi$ (resp. $\tau' c$ and $\tau'\neg\psi$) to θ; (Here τ' is the first prefix in the enumeration E_{pref} that is not on the tableau.)

Declare $\sigma\varphi$ finished.

(2) If $\sigma\Box\varphi$ is an occurrence of a prefixed sentence then, for all branches θ throughout $\sigma\Box\varphi$, if σ' is such that $\sigma < \sigma'$ is on θ, but $\sigma'\varphi$ is not on θ, add $\sigma'\varphi$ to θ.

(3) If θ is a branch such that the *S-Identifying*$'$ rule is applicable to θ, apply the rule. (That is, if θ contains σc, τc, $\sigma\Box\varphi$ and $\tau < \tau'$ and, θ does not contain $\tau'\varphi$, add $\tau'\varphi$ to θ.)

(4) If θ is an open branch such that for some nominals c and c_1 and some prefixes σ, τ and σ_1, $\sigma c, \tau c, \sigma_1 c_1, \tau c_1$ are on θ, add $\sigma_1 c, \sigma c_1$ to θ.

Complete *Step* $n+1$. Denote the tableau that has been constructed by $\mathcal{T}_{n+1}$.

End of the procedure.

Theorem 2. *The above procedure terminates and is a decision procedure for the languages* $\mathcal{L}_o$ *and* $\mathcal{L}(@)$.

Proof. First, for the sake of a contradiction suppose that there is a sentence χ such that on input χ the procedure will not terminate. Then, by König's lemma (cf. [10]), there will be an infinite branch, say θ. Let θ^n be the part of θ constructed in *Step* n, that is θ^n is $\theta \cap \mathcal{T}_n$. Note that all prefixes that occur on θ^n have depth at most n. Define $c(\theta^n) = \max_\varphi\{c(\varphi) \mid \sigma\varphi \text{ on } \theta^n \ \&\ \sigma \text{ has depth } n\}$ to be the maximal complexity of a sentence φ that occurs on θ^n prefixed by some σ of depth n. (Complexity $c(\varphi)$ of a sentence φ is the number of occurrences of Boolean connectives and operators in φ.) Define $b(\theta^n)$ and $d(\theta^n)$ to be the cardinality of respectively $\bigcup_c \text{ in } \chi\{\Box\psi \mid \exists\sigma : \sigma c \text{ and } \sigma\Box\psi \text{ are on } \theta^n\}$ and $\bigcup_c \text{ in } \chi\{\neg\Box\psi \mid \exists\sigma : \sigma c \text{ and } \sigma\neg\Box\psi \text{ are on } \theta^n \ \&\ \sigma\neg\Box\psi \text{ is finished on } \theta^n\}$. Let $s(\theta^n) = b(\theta^n) + d(\theta^n)$.

For all n, $s(\theta^n) \leq s(\theta^{n-1})$. Moreover, if in Step n no *S-Identifying*$'$ rule that results in extending θ^{n-1} has been applied, then $c(\theta^n) = c(\theta^{n-1}) - 1$. Otherwise, it is possible that $c(\theta^n) \geq c(\theta^{n-1})$. However, if in Step n, the *S-Identifying*$'$ rule has been applied at least once, then $s(\theta^n) \leq s(\theta^{n-1}) - 1$. This contradicts the assumption that θ is infinite.

Second, let χ be an arbitrary sentence of either $\mathcal{L}_o$ or $\mathcal{L}(@)$. From the above we know that the procedure on input χ will terminate, say after completing *Step* n. Then, the tableau $\mathcal{T}_n$ that has been constructed is either closed, or contains an open branch, say θ. Note that the set of all prefixed sentences on θ is labeled. Moreover, θ is saturated in the following sense: θ satisfies conditions (1), (2), (4), (5), (6), (@$'$) and (@$''$) of Definition 3 as well as conditions (3)$'$ and (7)$'$ defined below:

(3)$'$ for all nominals c, all prefixes σ, τ and τ', if σc, τc, $\sigma\Box\varphi$ and $\tau < \tau'$ are on θ, then $\tau'\varphi$ is on θ too;

(7)$'$ if $\sigma\neg\Box\psi$ is on θ then, there are prefixes τ and τ', and a nominal c such that

σc, τc, $\tau < \tau'$ and $\tau' \neg \psi$ are on θ.
Then, similarly to the proof of Lemma 2, we can show that θ is satisfiable.

Acknowledgments

I would like to thank the referees and Andreas Nonnengart for their comments and suggestions.

References

1. H. Andréka, J. van Benthem, and I. Németi. Modal languages and bounded fragments of predicate logic. *Journal of Philosophical Logic*, 27(3):217–274, 1998.
2. P. Blackburn. Internalizing labeled deduction. Manuscript, 1998.
3. P. Blackburn and J. Seligman. Hybrid languages. *Journal of Logic, Language and Information*, 4:251–272, 1995.
4. P. Blackburn and J. Seligman. What are hybrid languages? In M. Kracht, M. de Rijke, H. Wansing, and M. Zakharyaschev, editors, *Advances in Modal Logic, vol. 1*, pages 41–62. CSLI Publications, Stanford University, 1998.
5. P. Blackburn and M. Tzakova. Hybrid completeness. *Logic Journal of the IGPL*, 6(4):625–650, 1998.
6. P. Blackburn and M. Tzakova. Hybridizing concept languages. *Annals of Mathematics and Artificial Intelligence*, 24(1–4):23–49, 1998.
7. P. Blackburn and M. Tzakova. Hybrid languages and temporal logic. *Logic Journal of the IGPL*, 7(1):27–54, 1999.
8. R. Bull. An approach to tense logic. *Theoria*, 36:282–300, 1970.
9. M. D'Agostino, D. Gabbay, R. Hähnle, and J. Possega, editors. *Handbook of Tableau Methods*. Kluwer, Dordrecht, 1998. To appear.
10. M. Fitting. *Proof methods for modal and intuitionistic logic*. Synthese Library. Reidel, Dordrecht, 1983.
11. G. Gargov and V. Goranko. Modal logic with names. *Journal of Philosophical Logic*, 22(6):607–636, 1993.
12. V. Goranko. Hierarchies of modal and temporal logics with reference pointers. *Journal of Logic, Language and Information*, 5(1):1–24, 1996.
13. R. Goré. Tableau methods for modal and temporal logics. Technical Report TR-ARP-15-95, ARP, Australian National University, 1995.
14. S. Passy and T. Tinchev. An essay in combinatory dynamic logic. *Information and Computation*, 93:263–332, 1991.
15. A. Prior. *Past, present and future*. Oxford University Press, Oxford, 1967.
16. W. Rautenberg. Modal tableau calculi and interpolation. *Journal of Philosophical Logic Vol. 12 No. 4*, 12(4):403–424, 1983.
17. J. Seligman. A cut-free sequent calculus for elementary situated reasoning. Technical Report HCRC-RP 22, HCRC, University of Edinburgh, 1991.
18. J. Seligman. The logic of correct description. In M. de Rijke, editor, *Advances in Intensional Logic*, Applied Logic Series. Kluwer, Dordrecht, 1997.
19. R. M. Smullyan. *First-order logic*. Dover Publications, New York, 2nd edition, 1995.