A Tableaux Based Decision Procedure for a Broad Class of Hybrid Formulae with Binders

Serenella Cerrito

Lab. Ibisc
Université d'Evry Val d'Essonne,
France

Marta Cialdea Mayer

Dipart. Informatica e Automazione
Università di Roma Tre
Italy

Abstract

In this paper we provide the first (as far as we know) direct calculus deciding satisfiability of formulae in negation normal form in the fragment of hybrid logic with the satisfaction operator and the binder, where no occurrence of the $\Box$ operator is in the scope of a binder. A preprocessing step, rewriting formulae into equisatisfiable ones, turns the calculus into a satisfiability decision procedure for the fragment $\mathsf{HL}(@, \downarrow) \setminus \Box\!\downarrow\!\Box$, *i.e.* formulae in negation normal form where no occurrence of the binder is both in the scope of and contains in its scope a $\Box$ operator.

The calculus is based on tableaux, where nominal equalities are treated by means of substitution, and termination is achieved by means of a form of anywhere blocking with indirect blocking. Direct blocking is a relation between nodes in a tableau branch, holding whenever the respective labels (formulae) are equal up to (a proper form of) nominal renaming. Indirect blocking is based on a partial order on the nodes of a tableau branch, which arranges them into a tree-like structure.

1 Introduction

The Hybrid Logic $\mathsf{HL}(@, \downarrow)$ is an extension of modal (propositional, possibly multi-modal) logic K by means of three constructs: *nominals* (propositions which hold in exactly one state of the model), the *satisfaction operator* @ (allowing one to state that a given formula holds at the state named by a given nominal), and the *binder* $\downarrow$, accompanied by *state variables*, which allows one to give a name to the current state (see [2] for an overview of the subject).

The satisfiability problem for formulae of basic Hybrid Logic $\mathsf{HL}(@)$ (without the binder)[1] is decidable, and it stays decidable even with the addition of other

[1]The notation $\mathsf{HL}(Op_1, ..., Op_n)$ is commonly used to denote the extension of modal logic K by means of the operators $Op_1 ... Op_n$. In particular, $\mathsf{HL}(@, \downarrow, \mathsf{E}, \Diamond^-)$ and $\mathsf{HL}(@, \mathsf{E}, \Diamond^-)$

operators, such as the global and converse modalities. On the contrary, an unrestricted addition of the binder causes a loss of decidability [1, 3].

However, similarly to what happens for first order logic, one can obtain decidable fragments of hybrid logic with the binder by imposing syntactic restrictions on the way formulae are built. Some decidability results are proved in [13], which considers full Hybrid Logic $\mathsf{HL}(@, \downarrow, \mathsf{E}, \Diamond^-)$, that will henceforth be abbreviated as FHL. In that work it is proved that the source of undecidability is the occurrence of a specific *modal pattern* in formulae in negation normal form (NNF). A pattern π is a sequence of operators, and a formula is a π-formula, where $\pi = Op_1...Op_n$, if it is in NNF and contains some occurrence of Op_1 that contains in its scope an occurrence of Op_2, that in turn has in its scope an occurrence of Op_3, *etc.* For simplicity, moreover, when the $\Box$ operator is used in a pattern, it actually stands for any *universal operator*, *i.e.* one of the modalities $\Box, \Box^-$ or A. In particular, a $\Box\downarrow$-formula is a hybrid formula in NNF where some occurrence of the binder is in the scope of a universal operator; a $\downarrow\Box$-formula is a hybrid formula in NNF where some occurrence of a universal operator is in the scope of a binder; and a $\Box\downarrow\Box$-formula is a hybrid formula in NNF containing a universal operator in the scope of a binder, which in turn occurs in the scope of a universal operator. Finally, if π is a pattern, the fragment $\mathsf{HL}(Op_1, ..., Op_k) \setminus \pi$ is constituted by the class of NNF hybrid formulae in $\mathsf{HL}(Op_1, ..., Op_k)$ excluding π-formulae.

The main decidability result on syntactic restrictions proved in [13] is the following:

1. The satisfiability problem for $\mathsf{FHL} \setminus \Box\downarrow\Box$ is decidable.

This result is tight, in the sense that there is no pattern π that contains $\Box\downarrow\Box$ as a subsequence and such that the satisfiability problem for $\mathsf{FHL} \setminus \pi$ is still decidable. Therefore, the fragment $\mathsf{FHL} \setminus \Box\downarrow\Box$ is particularly interesting.

For the aim of the present work, it is important to recall the intermediate results allowing [13] to prove 1:

2. The satisfiability problem for $\mathsf{FHL} \setminus \Box\downarrow$ is decidable. This is proved by showing that there exists a satisfiability preserving translation from $\mathsf{FHL} \setminus \Box\downarrow$ to $\mathsf{HL}(@, \mathsf{E}, \Diamond^-)$. The translation is obtained by first replacing any occurrence of the binder by a full existential quantification over states (*i.e.* $\downarrow x.F$ is replaced by $\exists x(x \wedge F)$); in the resulting formula, no existential quantifier is in the scope of a universal operator, so that the existential quantifiers can be moved in front of the formula, and, finally, they are skolemized away by use of fresh nominals.

3. The satisfiability problem for $\mathsf{FHL} \setminus\downarrow\Box$ is decidable. This holds because the standard translation ST of FHL into first order classical logic [1, 13] maps formulae in the considered fragment into *universally guarded formulae* [13], that have a decidable satisfiability problem [8].

Result 1 easily follows from 2 and 3. Let in fact F be any formula in $\mathsf{FHL} \setminus \Box\downarrow\Box$. Any occurrence of the binder that contains in its scope a universal operator is not, in its turn, in the scope of a universal operator. Therefore it

include the existential global modality E (and its dual A) and the converse operator $\Diamond^-$ (and its dual $\Box^-$).

can be skolemized away like in the proof of 2. Repeating this transformation for every $\downarrow\Box$-subformula of F, an equisatisfiable formula F' is obtained, where no occurrence of a universal operator is in the scope of a binder. Satisfiability of F' can be decided because of result 3.

The above sketched approach to proving result 1 shows also that any decision procedure for formulae in $\mathsf{FHL} \setminus \downarrow\Box$ can easily be turned into a decision procedure for formulae in the largest fragment $\mathsf{FHL} \setminus \Box\downarrow\Box$, by preprocessing formulae.

Satisfiability of formulae in the fragment $\mathsf{FHL} \setminus \downarrow\Box$ can be tested by translation, by use of any calculus for the guarded fragment, such as the tableau calculi defined in [9, 10], or the decision procedure based on resolution given in [7]. The translation can be obtained in polynomial time [13], hence the theoretical complexity does not increase. However, in practice, the overhead coming from the translation cannot be completely ingnored. In fact, the standard translation of F is a *universally guarded* formula, which has to be rewritten into an equisatisfiable *guarded* one [8]. Moreover, decision procedures for guarded logic such as the above mentioned ones apply to constant-free formulae. Since formulae obtained from the translation may in general contain constants (deriving from nominals), a further rewriting would be necessary to eliminate them [8, 12].

Beyond the generally recognized interest of having direct calculi for modal logics, we therefore consider that the problem of defining direct decision procedures for decidable fragments of hybrid logics deserves a specific attention.

In this paper we provide the first (as far as we know) direct calculus deciding satisfiability of formulae in $\mathsf{HL}(@,\downarrow) \setminus \downarrow\Box$. A preprocessing step, rewriting a formula into an equisatisfiable one, like explained above, turns the calculus into a satisfiability decision procedure for $\mathsf{HL}(@,\downarrow) \setminus \Box\downarrow\Box$.

The work is organized as follows. In the rest of this section we recall the syntax and semantics of $\mathsf{HL}(@,\downarrow)$. In Section 2 we define the tableau system, and section 3 contains a brief outline of the termination and completeness proofs, whose details can be found in [6]. Section 4 concludes this work, and includes a comparison of some aspects of our work with techniques already present in the literature.

Hybrid Logic.

Let PROP (the set of propositional letters) and NOM (the set of nominals) be disjoint sets of symbols. Let VAR be a set of *state variables*. Hybrid formulae F in $\mathsf{HL}(@,\downarrow)$ are defined by the following grammar:

$$F := \ p \mid a \mid x \mid \neg F \mid F \wedge F \mid F \vee F \mid \Diamond F \mid \Box F \mid t : F \mid \downarrow x.F$$

where $p \in \mathsf{PROP}$, $a \in \mathsf{NOM}$, $x \in \mathsf{VAR}$ and $t \in \mathsf{VAR} \cup \mathsf{NOM}$. In this work, the notation $t : F$ is used rather than the more usual one $@_t F$. We use metavariables a, b, c, possibly with subscripts, for nominals, while x, y, z are used for variables.

A formula of the form $a : F$ is called a *satisfaction statement*, whose *outermost nominal* is a and F is its *body*. The operator $\downarrow$ is a *binder* for state variables. A variable x is *free* in a formula if it does not occur in the scope of a $\downarrow x$. A formula is *ground* if it contains no free variables.

A *subformula* of a formula F is a substring of F (possibly F itself) that is itself a formula. An *instance* of a formula F is an expression obtained by replacing every free variable of F with some nominal.

An *interpretation* $\mathcal{M}$ is a triple $\langle W, R, N, I\rangle$ where W is a non-empty set (whose elements are the *states* of the interpretation), $R \subseteq W \times W$ is a binary relation on W (the *accessibility relation*), N is a function $\mathsf{NOM} \rightarrow W$ and I a function $W \rightarrow 2^{\mathsf{PROP}}$. We shall write wRw' as a shorthand for $\langle w, w'\rangle \in R$.

A *variable assignment* σ for $\mathcal{M}$ is a function $\mathsf{VAR} \rightarrow W$. If $x \in \mathsf{VAR}$ and $w \in W$, the notation σ_x^w stands for the variable assignment σ' such that: $\sigma'(y) = \sigma(y)$ if $y \neq x$ and $\sigma'(x) = w$.

If $\mathcal{M} = \langle W, R, N, I\rangle$ is an interpretation, $w \in W$, σ is a variable assignment for $\mathcal{M}$ and F is a formula, the relation $\mathcal{M}_w, \sigma \models F$ is inductively defined as follows:

1. $\mathcal{M}_w, \sigma \models p$ if $p \in I(w)$, for $p \in \mathsf{PROP}$.
2. $\mathcal{M}_w, \sigma \models a$ if $N(a) = w$, for $a \in \mathsf{NOM}$.
3. $\mathcal{M}_w, \sigma \models x$ if $\sigma(x) = w$, for $x \in \mathsf{VAR}$.
4. $\mathcal{M}_w, \sigma \models \neg F$ if $\mathcal{M}_w, \sigma \not\models F$.
5. $\mathcal{M}_w, \sigma \models F \wedge G$ if $\mathcal{M}_w, \sigma \models F$ and $\mathcal{M}_w, \sigma \models G$.
6. $\mathcal{M}_w, \sigma \models F \vee G$ if either $\mathcal{M}_w, \sigma \models F$ or $\mathcal{M}_w, \sigma \models G$.
7. $\mathcal{M}_w, \sigma \models a : F$ if $\mathcal{M}_{N(a)}, \sigma \models F$, for $a \in \mathsf{NOM}$.
8. $\mathcal{M}_w, \sigma \models x : F$ if $\mathcal{M}_{\sigma(x)}, \sigma \models F$, for $x \in \mathsf{VAR}$.
9. $\mathcal{M}_w, \sigma \models \Box F$ if for each w' such that wRw', $\mathcal{M}_{w'}, \sigma \models F$.
10. $\mathcal{M}_w, \sigma \models \Diamond F$ if there exists w' such that wRw' and $\mathcal{M}_{w'}, \sigma \models F$.
11. $\mathcal{M}_w, \sigma \models {\downarrow} x.F$ if $\mathcal{M}_w, \sigma_x^w \models F$.

A formula F is *satisfiable* if there exist an interpretation $\mathcal{M}$, a variable assignment σ for $\mathcal{M}$ and a state w of $\mathcal{M}$, such that $\mathcal{M}_w, \sigma \models F$. Two formulae F and G are logically equivalent ($F \equiv G$) when, for every interpretation $\mathcal{M}$, assignment σ and state w of $\mathcal{M}$, $\mathcal{M}_w, \sigma \models F$ if and only if $\mathcal{M}_w, \sigma \models G$.

It is worth pointing out that, if $t \in \mathsf{VAR} \cup \mathsf{NOM}$ and F is a formula:

$$\neg(t : F) \equiv t : \neg F \qquad \neg{\downarrow} x.F \equiv {\downarrow} x.\neg F \qquad \neg\Diamond F \equiv \Box\neg F \qquad \neg\Box F \equiv \Diamond\neg F$$

This allows one to restrict attention to formulae in negation normal form (NNF).

2 The Tableau Calculus

A *tableau branch* is a sequence of *nodes* $n_0, n_1, \ldots$, where each node is labelled by a ground satisfaction statement in NNF, and a tableau is a set of branches. If n occurs before m in the branch $\mathcal{B}$, we shall write $n < m$. The label of the node n will be denoted by $label(n)$. The notation $(n)\, a : F$ will be used to denote the node n, and simultaneously say that its label is $a : F$.

A tableau for a formula F is initialized with a single branch, constituted by the single node $(n_0)\, a_0 : F$, where a_0 is a new nominal. The formula $a_0 : F$ is the *initial formula* of the tableau, which is assumed to be ground and in NNF.

A tableau is expanded by application of the rules in Table 1, which are applied to a given branch. Their reading is standard: a rule is applicable if the branch contains a node (two nodes) labelled by the formula(e) shown as premiss(es) of the rules. The rules $\wedge, @, \downarrow, \Box$ and $\Diamond$ add one or two nodes to the branch, labelled by the conclusion(s); the rule $\vee$ replaces the current branch $\mathcal{B}$ with two branches, each of which is obtained by adding $\mathcal{B}$ a new node, labelled, respectively, by the formula shown on the left and right below the inference line.

$$\frac{a:(F\wedge G)}{\begin{array}{c}a:F\\ a:G\end{array}}\ (\wedge) \qquad\qquad \frac{a:(F\vee G)}{a:F \quad\mid\quad a:G}\ (\vee)$$

$$\frac{a:b:F}{b:F}\ (@) \qquad\qquad \frac{a:{\downarrow}x.F}{a:F[a/x]}\ (\downarrow) \qquad\qquad \begin{array}{c}[\mathcal{B}]\\ \dfrac{a:b}{\mathcal{B}[b/a]}\ (=)\\ \text{(not applicable if } a=b\text{)}\end{array}$$

$$\frac{a:\Box F \quad a:\Diamond b}{b:F}\ (\Box) \qquad\qquad \begin{array}{c}\dfrac{a:\Diamond F}{\begin{array}{c}a:\Diamond b\\ b:F\end{array}}\ (\Diamond)\\ \text{where } b \text{ is a new nominal}\\ \text{(not applicable if } F \text{ is a nominal)}\end{array}$$

Table 1: Expansion rules

The $\Box$ rule has two premisses, which must both occur in the branch, in any order. The leftmost premiss of the $\Box$ rule is called its *major premiss*, the rightmost one its *minor premiss*. The minor premiss is a *relational formula*, *i.e.* a satisfaction statement of the form $a : \Diamond b$ (where b is a nominal). A formula of the form $\Box F$ is called a *universal formula*. The $\Diamond$ rule is called *blockable rule*, a formula of the form $a : \Diamond F$, where F is not a nominal, is a *blockable formula* and a node labelled by a blockable formula is a *blockable node*.

If F is a formula, the notation $F[a/x]$ is used to denote the formula that is obtained from F replacing a for every free occurrence of the variable x. Analogously, if a and b are nominals, $F[b/a]$ is the formula obtained from F replacing b for every occurrence of a. The *equality rule* $(=)$ does not add any node to the branch, but modifies the labels of its nodes. The schematic formulation of this rule in Table 1 indicates that it can be fired whenever a branch $\mathcal{B}$ contains a *nominal equality* of the form $a : b$ (with $a \neq b$); as a result of the application of the rule, every node label F in $\mathcal{B}$ is replaced by $F[b/a]$.

The first node of a branch $\mathcal{B}$ is called the *top node* and its label the *top formula* of $\mathcal{B}$. The nominals occurring in the top formula are called *top nominals*. Note that the notion of top nominal is relative to a tableau branch. In fact, applications of the equality rule may change the top formula, hence the set of top nominals.

In the following definition, the current branch is left implicit, so as to lighten the notation.

Definition 1. *If a node n is added to a branch by application of the rule $\mathcal{R}$ to the node m then we write $m \leadsto^{\mathcal{R}} n$. In the case of rules with two conclusions, we write $m \leadsto^{\mathcal{R}} (n, k)$, or, sometimes, $m \leadsto^{\mathcal{R}} n$ and $m \leadsto^{\mathcal{R}} k$. In the case of the two premisses rule $\Box$ we write $(m, k) \leadsto^{\Box} n$.*

Note that the application of the equality rule does not change nodes, but only their labels, therefore it does not change the relation $\leadsto^{\mathcal{R}}$ between *nodes*, for any rule $\mathcal{R}$.

We say that a formula $a : F$ occurs in a tableau branch $\mathcal{B}$ (or $a : F \in \mathcal{B}$) if for some node n of the branch, $label(n) = a : F$. Similarly, a nominal occurs in a branch $\mathcal{B}$ if it occurs in the label of some node of $\mathcal{B}$. Finally, a nominal a *labels* a formula F in $\mathcal{B}$ if $a : F \in \mathcal{B}$.

Termination is achieved by means of a loop-checking mechanism using nominal renaming. In fact, in the presence of the binder, non-top nominals may occur in the body of any node label. In order to define this mechanism, some preliminary definitions are necessary.

Definition 2 (Nominal compatibility and mappings)**.** *If $\mathcal{B}$ is a tableau branch and a is a nominal occurring in $\mathcal{B}$, then*

$$\Phi_{\mathcal{B}}(a) = \{p \mid p \in \mathsf{PROP} \textit{ and } a : p \in \mathcal{B}\} \cup \{\Box F \mid a : \Box F \in \mathcal{B}\}$$

If a and b are nominals occurring in a tableau branch $\mathcal{B}$, then a and b are compatible *in $\mathcal{B}$ if $\Phi_{\mathcal{B}}(a) = \Phi_{\mathcal{B}}(b)$,* i.e. *if they label the same propositions in* PROP *and the same universal formulae.*

A mapping *π for a branch $\mathcal{B}$ is an injective function from* non-top *nominals to* non-top *nominals such that for all a, a and $\pi(a)$ are compatible in $\mathcal{B}$.*

A mapping π for $\mathcal{B}$ maps *a formula F to a formula G if:*

1. *$\pi(F) = G$;*
2. *π is the identity for all nominals which do not occur in F.*

A formula F can be mapped to a formula G in $\mathcal{B}$ if there exists a mapping π for $\mathcal{B}$ mapping F to G.

Since a mapping π is the identity almost everywhere, it can be represented by a finite set of pairs of the form $\{b_1/a_1, ..., b_n/a_n\}$ where $a_i \neq b_i$, whenever $\pi(a_i) = b_i$ and $\pi(c) = c$ for all $c \notin \{a_1, ..., a_n\}$.

The application of the blockable rule is restricted by blocking conditions: a direct blocking condition, which forbids the application of the blockable rule to a node n, whenever the label of a previous node can be mapped to $label(n)$; and also an indirect blocking condition. In fact, since a node may be (directly) blocked in a branch after that it has already been expanded, all the nodes which, in some sense, depend from that expansion must be blocked too. So, a notion of indirect blocking is needed, which in turn requires a new partial order on nodes. The following definition introduces a binary relation on nodes, which organizes them into a family of trees.

Definition 3. *Let $\mathcal{B}$ be a tableau branch. The relation $n \prec_{\mathcal{B}} m$ between nodes of $\mathcal{B}$ is inductively defined as follows:*

Base case *If $n \leadsto^{\Diamond} (m, k)$, then $n \prec_{\mathcal{B}} m$ and $n \prec_{\mathcal{B}} k$;*

Inductive cases *If $m \prec_{\mathcal{B}} n$, then:*

1. *if $n \rightsquigarrow^{\mathcal{R}} k$, where $\mathcal{R} \in \{\vee, @, \downarrow, \wedge\}$, then $m \prec_{\mathcal{B}} k$;*
2. *if $label(n)$ is a relational formula and for some n', $(n', n) \rightsquigarrow^{\Box} k$, then $m \prec_{\mathcal{B}} k$.*

If $m \prec_{\mathcal{B}} n$ then n is said to be a child *of m* w.r.t. *$\prec_{\mathcal{B}}$, and m the* parent *of n. A node n in $\mathcal{B}$ is called a* root node *if it has no parent. Two nodes n and k are called* siblings *if either both of them are root nodes, or for some m, $m \prec_{\mathcal{B}} n$ and $m \prec_{\mathcal{B}} k$.*

The relation $\prec_{\mathcal{B}}^{+}$ is the transitive closure of $\prec_{\mathcal{B}}$. If $n \prec_{\mathcal{B}}^{+} m$, then n is an ancestor *of m and m a descendant of n* w.r.t. *$\prec_{\mathcal{B}}$.*

In other terms, when the blockable rule is applied to a node n, a first pair of children of n *w.r.t.* $\prec_{\mathcal{B}}$ is generated. The application of rules other than $\Diamond$ generates siblings, where, in the case of the two premisses rule $\Box$, it is the minor premiss which is added a sibling. Intuitively, when $n \prec_{\mathcal{B}} m$, n is the node which is taken to be the main "responsible" of the presence of m in the branch. In fact, the first "children" of a node n are nodes obtained from n by application of the blockable rule. And, if a node m is obtained from m' (as the minor premiss, in the case of the $\Box$ rule) by means of applications of non-blockable rules, then they are "siblings" *w.r.t.* $\prec_{\mathcal{B}}$.

Example 1. *As an example, consider the tableau branch for*

$$F = a : (\Diamond p \wedge \Box{\downarrow}x.\Diamond(p \wedge \neg x \wedge {\downarrow}y.a : \Diamond y))$$

represented in Figure 1. Node numbering reflects the order in which nodes are added to the branch. The right column reports the $\rightsquigarrow^{\mathcal{R}}$ relation justifying the addition of the corresponding node to the branch. W.r.t. *the relation $\prec_{\mathcal{B}}$, $0, 1$ and 3 are root nodes with no children; 2 is also a root node, with children $4, 5, 6$ and 7; nodes 8–17 are all children of 7.*

(0) $a_0 : a : (\Diamond p \wedge \Box{\downarrow}x.\Diamond(p \wedge \neg x \wedge {\downarrow}y.a : \Diamond y))$	
(1) $a : (\Diamond p \wedge \Box{\downarrow}x.\Diamond(p \wedge \neg x \wedge {\downarrow}y.a : \Diamond y))$	$0 \rightsquigarrow^{@} 1$
(2) $a : \Diamond p$	$1 \rightsquigarrow^{\wedge} 2$
(3) $a : \Box{\downarrow}x.\Diamond(p \wedge \neg x \wedge {\downarrow}y.a : \Diamond y)$	$1 \rightsquigarrow^{\wedge} 3$
(4) $a : \Diamond b$	$2 \rightsquigarrow^{\Diamond} 4$
(5) $b : p$	$2 \rightsquigarrow^{\Diamond} 5$
(6) $b : {\downarrow}x.\Diamond(p \wedge \neg x \wedge {\downarrow}y.a : \Diamond y)$	$(3, 4) \rightsquigarrow^{\Box} 6$
(7) $b : \Diamond(p \wedge \neg b \wedge {\downarrow}y.a : \Diamond y)$	$6 \rightsquigarrow^{\downarrow} 7$
(8) $b : \Diamond c$	$7 \rightsquigarrow^{\Diamond} 8$
(9) $c : p \wedge \neg b \wedge {\downarrow}y.a : \Diamond y$	$7 \rightsquigarrow^{\Diamond} 9$
(10) $c : p \wedge \neg b$	$9 \rightsquigarrow^{\wedge} 10$
(11) $c : {\downarrow}y.a : \Diamond y$	$9 \rightsquigarrow^{\wedge} 11$
(12) $c : p$	$10 \rightsquigarrow^{\wedge} 12$
(13) $c : \neg b$	$10 \rightsquigarrow^{\wedge} 13$
(14) $c : a : \Diamond c$	$11 \rightsquigarrow^{\downarrow} 14$
(15) $a : \Diamond c$	$14 \rightsquigarrow^{@} 15$
(16) $c : {\downarrow}x.\Diamond(p \wedge \neg x \wedge {\downarrow}y.a : \Diamond y)$	$(3, 15) \rightsquigarrow^{\Box} 16$
(17) $c : \Diamond(p \wedge \neg c \wedge {\downarrow}y.a : \Diamond y)$	$16 \rightsquigarrow^{\downarrow} 17$

Figure 1: A tableau branch for $a : (\Diamond p \wedge \Box{\downarrow}x.\Diamond(p \wedge \neg x \wedge {\downarrow}y.a : \Diamond y))$

The relation $\prec_{\mathcal{B}}$ enjoys the following important properties:

1. For each node n in a tableau branch $\mathcal{B}$, there exists at most one node m such that $m \prec_{\mathcal{B}} n$. Therefore, there is exactly one maximal chain
$$n_1 \prec_{\mathcal{B}} n_2 \prec_{\mathcal{B}} ... \prec_{\mathcal{B}} n_k = n$$
where n_1 is any root node.

2. If for some n, $m \prec_{\mathcal{B}} n$, then m is a blockable node. Therefore, for any chain
$$n_1 \prec_{\mathcal{B}} n_2 \prec_{\mathcal{B}} ... \prec_{\mathcal{B}} n_k \prec_{\mathcal{B}} n_{k+1}$$
$n_1, ..., n_k$ are all blockable nodes.

Consequently, $\prec_{\mathcal{B}}$ arranges the nodes of a branch into a forest of trees, where non-terminal nodes are blockable nodes.

We can now define the notions of direct and indirect blocking.

Definition 4 (Direct and indirect blocking)**.** *A node* $(n)\, a : \Diamond F$ *is* directly blocked *by* $(m)\, b : \Diamond G$ *in* $\mathcal{B}$ *if*

- $m < n$*,* m *is neither directly blocked in* $\mathcal{B}$ *nor it has any ancestor* w.r.t. $\prec_{\mathcal{B}}$ *which is directly blocked in* $\mathcal{B}$*;*
- $b : G$ *can be mapped to* $a : F$ *in* $\mathcal{B}$*.*

A node n *is directly blocked in* $\mathcal{B}$ *if it is blocked by some* m *in* $\mathcal{B}$*, and it is* indirectly blocked *in* $\mathcal{B}$ *if it has an ancestor* w.r.t. $\prec_{\mathcal{B}}$ *which is directly blocked in* $\mathcal{B}$*. An indirectly blocked node is called a* phantom node *(or, simply, a phantom).*

The tableau branch $\mathcal{B}$ represented in Figure 1 represents a blocking case: node 17 is blocked by 7, because b e c are compatible ($\Phi_{\mathcal{B}}(b) = \Phi_{\mathcal{B}}(c) = \{p\}$).

It must be remarked that the blocking relation is dynamic, *i.e.* blockings are not established forever, since they are relative to a tableau branch, and can be undone when expanding the branch. In fact, a node may be blocked in a branch $\mathcal{B}$ and then unblocked after expanding $\mathcal{B}$, because the addition of new nodes or changes in node labels may destroy nominal compatibility. Possibly, a new blocking can be introduced (but compatibilities must be checked again), by means of a different mapping.

The application of the expansion rules is restricted by the following conditions:

Definition 5 (Restrictions on the expansion rules)**.** *The expansion of a tableau branch* $\mathcal{B}$ *is subject to the following restrictions:*

R1. *no node labelled by a formula already occurring in* $\mathcal{B}$ *as the label of a* non-phantom *node is ever added to* $\mathcal{B}$*;*

R2. *a blockable node* n *cannot be expanded if there are* $k_0, k_1 \in \mathcal{B}$ *such that* $n \leadsto^{\Diamond} (k_0, k_1)$*;*

R3. *a phantom node cannot be expanded by means of a single-premiss rule, nor can it be used as the minor premiss of the* $\Box$ *rule;*

R4. *a blockable node cannot be expanded if it is directly blocked in* $\mathcal{B}$*.*

It is worth pointing out that termination would not be guaranteed if restriction **R1** were replaced by the condition that a node (or pair of nodes) is never expanded more than once on the branch.

A branch is *closed* whenever it contains, for some nominal a, either a pair of nodes $(n)\, a : p$, $(m)\, a : \neg p$ for some $p \in \mathsf{PROP}$, or a node $(n)\, a : \neg a$. As usual, we assume that a closed branch is never expanded further on. A branch which is not closed is *open.* A branch is *complete* when it cannot be further expanded. For instance, the tableau branch represented in Figure 1 is complete and open.

This section concludes with some further examples. In each of them, $\mathcal{B}$ denotes the considered branch, and the notation $\mathcal{B}_n$ is used to denote the branch segment up to node n, while Φ_n abbreviates $\Phi_{\mathcal{B}_n}$.

Example 2. *Figure 2 represents a closed one-branch tableau for*

$$F = (\Diamond{\downarrow}x.\Diamond(x : p)) \land (\Diamond{\downarrow}y.\Diamond(y : \neg p)) \land (\Diamond{\downarrow}z.(\Diamond(z : p) \land \Diamond(z : \neg p)))$$

where the first applications of the $\land$-rule are collapsed into one.

(0)	$a_0 : F$		(14)	$b_1 : b : \neg p$	$12 \rightsquigarrow^{\Diamond} 14$
(1)	$a_0 : \Diamond{\downarrow}x.\Diamond x : p$	$0 \rightsquigarrow^{\land} 1$	(15)	$b : \neg p$	$14 \rightsquigarrow^{@} 15$
(2)	$a_0 : \Diamond{\downarrow}y.\Diamond y : \neg p$	$0 \rightsquigarrow^{\land} 2$	(16)	$a_0 : \Diamond c$	$3 \rightsquigarrow^{\Diamond} 16$
(3)	$a_0 : \Diamond{\downarrow}z.(\Diamond(z : p)$ $\land \Diamond(z : \neg p))$	$0 \rightsquigarrow^{\land} 3$	(17)	$c : {\downarrow}z.(\Diamond(z : p)$ $\land \Diamond(z : \neg p))$	$3 \rightsquigarrow^{\Diamond} 17$
(4)	$a_0 : \Diamond a$	$1 \rightsquigarrow^{\Diamond} 4$	(18)	$c : \Diamond c : p \land \Diamond c : \neg p$	$17 \rightsquigarrow^{\downarrow} 18$
(5)	$a : {\downarrow}x.\Diamond x : p$	$1 \rightsquigarrow^{\Diamond} 5$	(19)	$c : \Diamond c : p$	$18 \rightsquigarrow^{\land} 19$
(6)	$a : \Diamond a : p$	$5 \rightsquigarrow^{\downarrow} 6$	(20)	$c : \Diamond c : \neg p$	$18 \rightsquigarrow^{\land} 20$
(7)	$a : \Diamond a_1$	$6 \rightsquigarrow^{\Diamond} 7$	(21)	$c : \Diamond c_1$	$19 \rightsquigarrow^{\Diamond} 21$
(8)	$a_1 : a : p$	$6 \rightsquigarrow^{\Diamond} 8$	(22)	$c_1 : c : p$	$19 \rightsquigarrow^{\Diamond} 22$
(9)	$a : p$	$8 \rightsquigarrow^{@} 9$	(23)	$c : p$	$22 \rightsquigarrow^{@} 23$
(10)	$a_0 : \Diamond b$	$2 \rightsquigarrow^{\Diamond} 10$	(24)	$c : \Diamond c_2$	$20 \rightsquigarrow^{\Diamond} 24$
(11)	$b : {\downarrow}y.\Diamond y : \neg p$	$2 \rightsquigarrow^{\Diamond} 11$	(25)	$c_2 : c : \neg p$	$20 \rightsquigarrow^{\Diamond} 25$
(12)	$b : \Diamond b : \neg p$	$11 \rightsquigarrow^{\downarrow} 12$	(26)	$c : \neg p$	$25 \rightsquigarrow^{@} 26$
(13)	$b : \Diamond b_1$	$12 \rightsquigarrow^{\Diamond} 13$			

Figure 2: Example 2

The relation $\prec_{\mathcal{B}}$ in this branch can be described as follows: 0–3 *are root nodes,* $1 \prec_{\mathcal{B}} \{4, 5, 6\}$*,* $6 \prec_{\mathcal{B}} \{7, 8, 9\}$*,* $2 \prec_{\mathcal{B}} \{10, 11, 12\}$*,* $12 \prec_{\mathcal{B}} \{13, 14, 15\}$*,* $3 \prec_{\mathcal{B}} \{16, 17, 18, 19, 20\}$*,* $19 \prec_{\mathcal{B}} \{21, 22, 23\}$*,* $20 \prec_{\mathcal{B}} \{24, 25, 26\}$*.*[2]

The branch is closed because of nodes 23 and 26. In $\mathcal{B}_{20}$, node 19 is not blocked by 6, since $a : \Diamond a : p$ cannot be mapped to $c : \Diamond c : p$ because c and a are not compatible in $\mathcal{B}_{20}$ ($\Phi_{20}(c) = \emptyset \neq \{p\} = \Phi_{20}(a)$); therefore, node 19 can be expanded. In the same branch segment, on the contrary, node 20 is blocked by 12, because $\Phi_{20}(c) = \emptyset = \Phi_{20}(b)$.

When the construction proceeds, expanding the non-blocked node 19, and nodes 21–23 are added to the branch, c and b are no more compatible ($\Phi_{23}(c) = \{p\}$ while $\Phi_{23}(b)$ is still empty), so node 20 is unblocked and it is expanded, producing 24–26 and the branch closes.

Note moreover that, after the addition of node 23, a and c become compatible, so that in $\mathcal{B}_{23}$ node 19 is blocked by 6, and 21–23 are phantom nodes. Since 20 is not a descendant of 19 w.r.t. *$\prec_{\mathcal{B}}$, it is not a phantom, thus it can be expanded.*

[2] $n \prec_{\mathcal{B}} \{m_1, ..., m_k\}$ abbreviates $n \prec_{\mathcal{B}} m_1$ and ... $n \prec_{\mathcal{B}} m_k$.

Example 3. *This example shows the need of indirect blocking (restriction* **R3***) to ensure termination. Let*

$$F = a : ((\Box{\downarrow}x.\Diamond{\downarrow}y.(x : p \wedge a : \Diamond y)) \wedge \Diamond q)$$

Figure 3 shows a complete branch in a tableau for F.

(1)	$a_0 : F$	
(2)	$a : ((\Box{\downarrow}x.\Diamond{\downarrow}y.$ $(x : p \wedge a : \Diamond y)) \wedge \Diamond q)$	$1 \rightsquigarrow^{@} 2$
(3)	$a : \Box{\downarrow}x.\Diamond{\downarrow}y.$ $(x : p \wedge a : \Diamond y)$	$2 \rightsquigarrow^{\wedge} 3$
(4)	$a : \Diamond q$	$2 \rightsquigarrow^{\wedge} 4$
(5)	$a : \Diamond b$	$4 \rightsquigarrow^{\Diamond} 5$
(6)	$b : q$	$4 \rightsquigarrow^{\Diamond} 6$
(7)	$b : {\downarrow}x.\Diamond{\downarrow}y.$ $x : p \wedge a : \Diamond y)$	$(3, 5) \rightsquigarrow^{\Box} 7$
(8)	$b : \Diamond{\downarrow}y.(b : p \wedge a : \Diamond y)$	$7 \rightsquigarrow^{\downarrow} 8$
(9)	$b : \Diamond b_1$	$8 \rightsquigarrow^{\Diamond} 9$
(10)	$b_1 : {\downarrow}y.(b : p \wedge a : \Diamond y)$	$8 \rightsquigarrow^{\Diamond} 10$
(11)	$b_1 : (b : p \wedge a : \Diamond b_1)$	$10 \rightsquigarrow^{\downarrow} 11$
(12)	$b_1 : b : p$	$11 \rightsquigarrow^{\wedge} 12$
(13)	$b_1 : a : \Diamond b_1$	$11 \rightsquigarrow^{\wedge} 13$
(14)	$b : p$	$12 \rightsquigarrow^{@} 14$
(15)	$a : \Diamond b_1$	$13 \rightsquigarrow^{@} 15$
(16)	$b_1 : {\downarrow}x.\Diamond{\downarrow}y.$ $(x : p \wedge a : \Diamond y)$	$(3, 15) \rightsquigarrow^{\Box} 16$
(17)	$b_1 : \Diamond{\downarrow}y.(b_1 : p \wedge a : \Diamond y)$	$16 \rightsquigarrow^{\downarrow} 17$
(18)	$b_1 : \Diamond b_2$	$17 \rightsquigarrow^{\Diamond} 18$
(19)	$b_2 : {\downarrow}y.(b_1 : p \wedge a : \Diamond y)$	$17 \rightsquigarrow^{\Diamond} 19$
(20)	$b_2 : (b_1 : p \wedge a : \Diamond b_2)$	$19 \rightsquigarrow^{\downarrow} 20$
(21)	$b_2 : b_1 : p$	$20 \rightsquigarrow^{\wedge} 21$
(22)	$b_2 : a : \Diamond b_2$	$20 \rightsquigarrow^{\wedge} 22$
(23)	$b_1 : p$	$21 \rightsquigarrow^{@} 23$
(24)	$a : \Diamond b_2$	$22 \rightsquigarrow^{@} 24$
(25)	$b_2 : {\downarrow}x.\Diamond{\downarrow}y.$ $(x : p \wedge a : \Diamond y)$	$(3, 24) \rightsquigarrow^{\Box} 25$
(26)	$b_2 : \Diamond{\downarrow}y.(b_2 : p \wedge a : \Diamond y)$	$25 \rightsquigarrow^{\downarrow} 26$
(27)	$b_2 : \Diamond b_3$	$26 \rightsquigarrow^{\Diamond} 27$
(28)	$b_3 : {\downarrow}y.(b_2 : p \wedge a : \Diamond y)$	$26 \rightsquigarrow^{\Diamond} 28$
(29)	$b_3 : (b_2 : p \wedge a : \Diamond b_3)$	$28 \rightsquigarrow^{\downarrow} 29$
(30)	$b_3 : b_2 : p$	$29 \rightsquigarrow^{\wedge} 30$
(31)	$b_3 : a : \Diamond b_3$	$29 \rightsquigarrow^{\wedge} 31$
(32)	$b_2 : p$	$30 \rightsquigarrow^{@} 32$
(33)	$a : \Diamond b_3$	$31 \rightsquigarrow^{@} 33$

Figure 3: Example 3.

The relation $\prec_{\mathcal{B}}$ in this branch can be described as follows: the root nodes are 1–4, $4 \prec_{\mathcal{B}} \{5, ..., 8\}$, $8 \prec_{\mathcal{B}} \{9, ..., 17\}$, $17 \prec_{\mathcal{B}} \{18, ..., 26\}$ and $26 \prec_{\mathcal{B}} \{27, ..., 33\}$.

In $\mathcal{B}_{17}$ node 17 is not blocked by 8 because $\Phi_{17}(b) = \{q, p\} \neq \emptyset = \Phi_{17}(b_1)$. And it is not blocked by 8 in $\mathcal{B}_n$ for any $n \geq 23$ either, where $\Phi_n(b) = \{q, p\} \neq \{p\} = \Phi_n(b_1)$. Moreover in $\mathcal{B}_{26}$ node 26 is blocked neither by 8 nor by 17, because $\Phi_{26}(b) = \{q, p\}$, $\Phi_{26}(b_1) = \{p\}$, and $\Phi_{26}(b_2) = \emptyset$.

But in $\mathcal{B}_{33}$ node 26 is blocked by 17, because $\Phi_{33}(b_1) = \{p\} = \Phi_{33}(b_2)$. Therefore, its children w.r.t. *$\prec_{\mathcal{B}_{33}}$,* i.e. *27–33 are all phantom nodes, and, in particular, node 33 cannot participate, with node 3, to an expansion via the $\Box$ rule.*

Without restriction **R3***, the construction of the branch would go on forever. In fact, the following nodes could be added:*

(34)	$b_3 : {\downarrow}x.\Diamond{\downarrow}y.(x : p \wedge a : \Diamond y)$	$(3, 33) \rightsquigarrow^{\Box} 34$
(35)	$b_3 : \Diamond{\downarrow}y.(b_3 : p \wedge a : \Diamond y)$	$34 \rightsquigarrow^{\downarrow} 35$

In $\mathcal{B}_{35}$ node 35 would not be blocked, because $\Phi_{35}(b_3) = \emptyset$, while $\Phi_{35}(b_1) = \Phi_{35}(b_2) = \{p\}$. So a sequence of new nodes could be added, with labels obtained from the labels of 27–34, by renaming b_2 with b_3 and b_3 with a new nominal b_4. A neverending story ...

3 Properties of the Calculus

The tableau calculus defined in Section 2 is trivially sound. Moreover it terminates and can be proved complete, *provided that the initial formula is in the*

$\mathit{HL}(@, \downarrow) \setminus {\downarrow}\Box$ *fragment.* Since space restrictions do not allow for a full account of the termination and completeness proofs, which are in some points quite subtle, this section gives only a brief and simplified outline of these proofs. Their details can be found in [6].

For the purposes of proving termination and completeness, the main property of the considered fragment is that, if $\Box G$ is a subformula of the initial formula, it contains no free variable. As a consequence, for any node label of the form $a : \Box G$, G does not contain any non-top nominal. In other terms, $\Box G$ is a subformula of the top formula of the branch (*strong subformula property*).

A looser subformula property holds for node labels in general: if $a : F$ is the label of some node in a tableau branch $\mathcal{B}$, then F is an *instance* of some subformula F' of the top formula of $\mathcal{B}$, i.e. F is obtained from F' by replacing the free variables occurring in F' with nominals (*loose subformula property*).

Termination is proved by showing that the nodes of a branch $\mathcal{B}$ are arranged by $\prec_{\mathcal{B}}$ into a bounded sized set of trees, each of which has bounded width and bounded depth. Hence any tableau branch $\mathcal{B}$ has a number of nodes that is bounded by a function of the size N of the initial formula.

The above statement is proved by use of the following intermediate results:

1. The number of siblings *w.r.t.* $\prec_{\mathcal{B}}$ of any node n is bounded by a function of N. This is not as trivial a task as it may appear at first sight. In fact, it is not sufficient to show that the number of formulae that can label the siblings of a given node is bounded, because, in principle, a given formula might be the label of an infinite number of nodes. In fact, notwithstanding restriction **R1**, distinct node labels can become equal by effect of substitution.

   However, it can be shown that the label of any sibling of n has a *matrix* taken from a bounded stock of formulae, that can be built in the language of the branch *at the time n is added to it.* Node labels with a same matrix are always equal, at any construction stage of the branch, since they are obtained from the same formula (matrix) by application of the same nominal renaming. Since siblings always have the same phantom/non-phantom state, restrictions **R1** and **R3** ensure that any node can only produce a bounded number of siblings.

   Note that the above sketched reasoning would not work if it were the major premiss of the $\Box$ rule to be added a sibling *w.r.t.* $\prec_{\mathcal{B}}$. In fact, a node labelled by a universal formula can in principle produce an infinite number of expansions.

2. The length of any chain of nodes $n_1 \prec_{\mathcal{B}} n_2 \prec_{\mathcal{B}} \dots \prec_{\mathcal{B}} n_k$ is bounded by a function of N. This is due to the "loose" subformula property and the fact that the set of elements in $\mathsf{PROP} \cup \{\Box G \mid \Box G$ occurs in some node label$\}$ is bounded by N (by the "strong" subformula property). Therefore, restrictions **R2** and **R4** ensure that the blockable rule cannot be applied to extend any chain of nodes beyond a given depth.

It is worth pointing out that the considerations underlying the termination argument in [6] establish a doubly exponential upper bound on the number of nodes in a tableau branch. Therefore, the decision procedure defined in this paper is not worst-case optimal, since the satisfiability problem for $\mathsf{HL}(@, \downarrow) \setminus {\downarrow}\Box$ is in 2-ExpTime [13].

Completeness is proved in the standard way, by showing how to define a model of the initial formula from a complete and open tableau branch. However, for the calculus defined in this work, the fact that the labels of blocked and blocking nodes are not identical must be taken into account. A model cannot be simply built from a set of states consisting of equivalence classes of nominals, and establishing that two nominals are in the same class whenever some blocking mapping maps one to the other. In fact, it might be the case that a nominal a is mapped to a nominal b to block a given node, although the branch contains a node labelled by $a : \neg b$ (like in Example 1).

Thus, a different approach is followed, showing that a (possibly infinite) model can be built out of a complete and open branch $\mathcal{B}$ by means of a preliminary infinitary extension $\mathcal{N}^{\infty}_{\mathcal{B}}$ of a subset $\mathcal{N}_0$ of $\mathcal{B}$. More precisely, $\mathcal{N}_0$ is the union of the non-phantom nodes in $\mathcal{B}$ and the nodes $(n)\, a : F$ where either $F \in \mathsf{PROP}$ or F has the form $\Box G$.

$\mathcal{N}^{\infty}_{\mathcal{B}}$ is built by stages, as the union of a (possibly infinite) series of extensions $\mathcal{N}_0 \subseteq \mathcal{N}_1 \subseteq \mathcal{N}_2 ....$ of $\mathcal{N}_0$. The purpose of each stage is the creation of a *witness* for a given blockable node, where a nominal b is called a witness for a node labelled by a blockable formula of the form $a : \Diamond F$ if there exist nodes labelled, respectively, by $a : \Diamond b$ and $b : F$. Each sequence of (labelled) nodes $\mathcal{N}_i$ is associated a *blocking relation* $\mathbf{B}_i$, containing triples of the form (n, m, π), where n and m are nodes, $m < n$ and π is an injective mapping such that $\pi(label(m)) = label(n)$. The construction ensures that:

1. for any $(n, m, \pi) \in \mathbf{B}_i$:
   (a) n and m are labelled by blockable formulae;
   (b) $m \in \mathcal{N}_0$ and it has a witness in $\mathcal{N}_0$.
2. For any blockable node $n \in \mathcal{N}_i$, if n has no witness in $\mathcal{N}_i$, then $(n, m, \pi) \in \mathbf{B}_i$, for some m and π.

Each extension $\mathcal{N}_i$ is built so as to add a *witness* to the first "blocked" node $n \in \mathcal{N}_{i-1}$, *i.e.* such that for some m and π, $(n, m, \pi) \in \mathbf{B}_{i-1}$. The label of each new node added to $\mathcal{N}_i$ is obtained from a node in $\mathcal{N}_0$ by suitably renaming non-top nominals. Specifically, an injective mapping θ_i is defined, that will guide the construction of the new nodes of $\mathcal{N}_i$. The mapping θ_i is the identity except for the following cases:

- if a occurs in $label(m)$, then $\theta_i(a) = \pi(a)$;
- if b is the witness of m and b does not occur in $label(m)$, then $\theta_i(b) = b^i$, where b^i is a fresh nominal. Note that, at the time the witness of m was added to the branch by an application of the $\Diamond$ rule, it was obviously fresh *w.r.t.* to the current branch, but it may subsequently have been replaced by the equality rule.

The sequence $\mathcal{N}_i$ is then obtained from $\mathcal{N}_{i-1}$ by adding new nodes, labelled by $\theta_i(label(k))$ for each $k \in \mathcal{N}_0$ such that $\theta_i(label(k))$ does not already occur in $\mathcal{N}_{i-1}$. Hence, in particular, a pair of nodes is added, representing the fact that b^i is the witness of n in $\mathcal{N}_i$.

Consistently, the triple (n, m, π) is removed from the blocking relation. Possibly, new nodes with no witness are created; for each of them, a blocking node and blocking mapping are defined, and the corresponding triple is added to $\mathbf{B}_i$.

Each of the sets of nodes $\mathcal{N}_i$ enjoys a form of *saturation property*: it is consistent (there are no labels of the form $a : \neg a$, or both $a : p$ and $a : \neg p$), it does not contain non-trivial equalities ($a : b$ with $a \neq b$, so that the equality rule does not need to be taken into account), and, for any node or pair of nodes in $\mathcal{N}_i$ that could be the premiss(es) of some expansion rule other than $\Diamond$, its expansion(s) are also in $\mathcal{N}_i$.

The proof of such a saturation property exploits the following (non trivial) properties of the construction:

- If $i > 0$ and θ_i is the mapping used to extend $\mathcal{N}_{i-1}$ to $\mathcal{N}_i$, then for any nominal a, a and $\theta_i(a)$ are compatible in $\mathcal{N}_i$;
- for every triple $(n, m, \pi) \in \mathbf{B}_i$ and for any nominal a, a and $\pi(a)$ are compatible in $\mathcal{N}_i$.

In the union $\mathcal{N}_{\mathcal{B}}^{\infty} = \bigcup_{i \in \mathbb{N}} \mathcal{N}_i$ every blockable node has a witness, and a model can be defined from it, made up of a state for each nominal occurring in $\mathcal{N}_{\mathcal{B}}^{\infty}$. Such a model can easily be extended to a model of the initial formula.

4 Concluding Remarks

In this work a tableau calculus for $\mathsf{HL}(@, \downarrow)$ is defined, which is provably terminating (independently of the rule application strategy) and complete for formulae belonging to the fragment $\mathsf{HL}(@, \downarrow) \setminus {\downarrow}\Box$. A preprocessing step transforming formulae into equisatisfiable ones turns the calculus into a satisfiability decision procedure for $\mathsf{HL}(@, \downarrow) \setminus \Box{\downarrow}\Box$.

The main features of the calculus can be summarized as follows. A tableau branch is a sequence of nodes, each of which is labelled by a satisfaction statement. Since nominal equalities are dealt with by means of substitution, different occurrences of the same formula may occur as labels of different nodes in a branch. The fact that when two formulae become equal by the effect of substitution the corresponding nodes do not collapse, allows for the definition of a binary relation $\prec_{\mathcal{B}}$ on nodes which organizes them into a family of trees. Each tree of the family has a bounded width, and this is due to the fact that, when applying the two premisses $\Box$ rule, it is the minor premiss, labelled by a relational formula, which is taken to be the "main responsible" of the expansion.

The fact that each tree has a bounded depth is guaranteed by a blocking mechanism which forbids the application of the $\Diamond$ rule to a node n whenever it has already been applied to another node whose label is equal to the label of n, *modulo non-top nominal renaming* (accompanied by suitable restrictions). Renaming is essential, because, in the presence of the binder, non-top nominals may occur in the body of any node label. The blocking mechanism is *anywhere* blocking, paired with indirect blocking, relying on the relation $\prec_{\mathcal{B}}$.

This mechanism differs from [4, 5], where calculi for hybrid logic with the global and converse modalities (and no binders) are defined. In fact, such calculi adopt ancestor blocking, where *nominals* (and not nodes) are blocked, and indirect blocking relies on a partial order on nominals (instead of nodes). Differently from [5], moreover, the calculus defined in this work does not require nominal deletion to ensure termination. This is due, again, to the fact that a branch is not a set of formulae, but a sequence of nodes.

Also the tableau system defined in [11] for hybrid logic with the difference and converse modalities makes use of ancestor blocking, relying on an ancestor relation among nominals. The blocking mechanism used for converse free formulae in the same work is different and more similar to ours. In fact, an existential formula, such as, for instance, $a : \Diamond F$, is blocked (independently of its outermost nominal a) whenever there exists a nominal b labelling both F and every formula G such that $a : \Box G$ is in the branch. However, the sub-calculus does not terminate unless applications of the $\Box$ rule are prioritized.

A tableau calculus testing satisfiability of formulae in the *constant-free clique guarded fragment* has been proposed in [9]. A restriction of the algorithm to the guarded fragment has been defined and implemented [10]. A tableau branch, in these calculi, is a tree of nodes, and the label of each node is a set of formulae. A node is directly blocked by a previously created node if, essentially, their labels are the same modulo constant renaming. Our comparison modulo renaming method was in fact originally inspired by [9, 10] (although there are some differences). A further contact point between these calculi and ours is anywhere blocking coupled with indirect blocking (which, in [9, 10] relies on the ancestor relation in the tree).

We are presently working at the next natural step, *i.e.* the extension of the calculus to the global and converse modalities, so as to obtain a tableau based decision procedure for the fragment $\mathsf{HL}(@, \downarrow, \mathsf{E}, \Diamond^-) \setminus \Box{\downarrow}\Box$.

References

[1] C. Areces, P. Blackburn, and M. Marx. A road-map on complexity for hybrid logics. In J. Flum and M. Rodríguez-Artalejo, editors, *Computer Science Logic*, volume 1683 of *LNCS*, pages 307–321. Springer, 1999.

[2] C. Areces and B. ten Cate. Hybrid logics. In P. Blackburn, F. Wolter, and J. van Benthem, editors, *Handbook of Modal Logics*, pages 821–868. Elsevier, 2007.

[3] P. Blackburn and J. Seligman. Hybrid languages. *Journal of Logic, Language and Information*, 4:251–272, 1995.

[4] T. Bolander and P. Blackburn. Termination for hybrid tableaus. *Journal of Logic and Computation*, 17(3):517–554, 2007.

[5] S. Cerrito and M. Cialdea Mayer. Nominal substitution at work with the global and converse modalities. In L. Beklemishev, V. Goranko, and V. Shehtman, editors, *Advances in Modal Logic*, volume 8, pages 57–74. College Publications, 2010.

[6] S. Cerrito and M. Cialdea Mayer. A calculus for a decidable fragment of hybrid logic with binders. Technical Report RT-DIA-181-2011, Dipartimento di Informatica e Automazione, Università di Roma Tre, 2011. Available at `http://www.dia.uniroma3.it/Plone/ricerca/technical-reports/2011`.

[7] H. Ganzinger and H. De Nivelle. A superposition decision procedure for the guarded fragment with equality. In *Proc. 14th Symposium on Logic in Computer Science*, pages 295–305. IEEE Computer Society Press, 1999.

[8] E. Grädel. On the restraining power of guards. *Journal of Symbolic Logic*, 64:1719–1742, 1998.

[9] C. Hirsch and S. Tobies. A tableau algorithm for the clique guarded fragment. In F. Wolter, H. Wansing, M. de Rijke, and M. Zakharyaschev, editors, *Advances in Modal Logic*, volume 3, pages 257–277. CSLI Publications, 2001.

[10] J. Hladik. Implementation and evaluation of a tableau algorithm for the guarded fragment. In U. Egly and C. G. Fermüller, editors, *Proc. of the Int. Conf. on Automated Reasoning with Analytic Tableaux and Related Methods (TABLEAUX 2002)*, volume 2381 of *LNAI*, pages 145–159. Springer, 2002.

[11] M. Kaminski and G. Smolka. Terminating tableau systems for hybrid logic with difference and converse. *Journal of Logic, Language and Information*, 18(4):437–464, 2009.

[12] B. ten Cate and M. Franceschet. Guarded fragments with constants. *Journal of Logic, Language and Information*, 14:281–288, 2005.

[13] B. ten Cate and M. Franceschet. On the complexity of hybrid logics with binders. In L. Ong, editor, *Proc. of Computer Science Logic 2005*, volume 3634 of *LNCS*, pages 339–354. Springer, 2005.