

Terminating tableaux for $HL(@)$ without loop-checking

Serenella CERRITO, inst1 Marta CIALDEA MAYER
2

¹ IBISC

Tour Évry 2

523 place des terrasses de l'Agora

91000 Évry Cedex, France

Email: Serenella.Cerrito@ibisc.univ-evry.fr

² Dipartimento di Informatica e Automazione

Università Roma 3

Via della Vasca Navale 79, 00146, Italie Email: cialdea@dia.uniroma3.it.
— *Méthodes Formelles* —



RAPPORT DE RECHERCHE

N° 007

15 Fevrier 2007

Serenella CERRITO, inst1 Marta CIALDEA MAYER

Terminating tableaux for HL(@) without loop-checking

20 p.

Les rapports de recherche d'IBISC sont disponibles aux formats PostScript® et PDF® à l'URL :

<http://www.ibisc.univ-evry.fr/Vie/TR>

Research reports from IBISC are available in PostScript® and PDF® formats at the URL:

<http://www.ibisc.univ-evry.fr/Vie/TR>

© Mai 2007 by Serenella CERRITO, inst1 Marta CIALDEA MAYER

Terminating tableaux for $HL(@)$ without loop-checking

Serenella CERRITO, inst1 Marta CIALDEA MAYER

Résumé

This paper introduces a terminating tableau system for hybrid logic with the satisfaction operator, $HL(@)$, that does not need loop-checks. Differently from other calculi for $HL(@)$ enjoying the same property, termination does not rely on any specific – and sometimes complicated – tableau construction procedure.

This document corresponds to a draft which has been completed on 2007, February the 15th h.

Mots-clés additionnels et phrases : Decidability, Hybrid Modal Logic, Proof Systems, Tableaux, Termination

1 Introduction

Hybrid logics extend modal logics with the possibility of naming states by means of a second sort of propositional letters, called *nominals*, each of which is assumed to be true at exactly one state. Beyond this basic feature, hybrid logics are equipped with different operators that endow them with a very rich expressive power and make them well suited to reason on graph structures. In fact, one of the “hot” applications of such logics in recent years concerns the so called *semistructured databases* [1], where data are organized in a finite graph. Even the World Wide Web is often presented as a “giant” semistructured database, where vertexes are uniquely defined by URLs and edges are hypertext links between them.

The logical operators that are peculiar to hybrid languages are the *satisfaction operator*, $@$, and the *binder*, \downarrow . These, beyond the modal operators, are the operators of the hybrid logic named $HL(@, \downarrow)$. The satisfaction operator allows one to “jump” from the current state to a different one : if s is a nominal, a formula of the form $@_s A$ means that A holds at the state named by s . The binder, on the contrary, binds a *state variable* to the current world : $\downarrow x.A$ is true if A is true when x is taken as the name of the current state. For instance, if s is a nominal, the formula $@_s \Box \Box \downarrow x.@_s \Diamond x$ states that every state accessible from a state accessible from s is also accessible directly from s (i.e. the accessibility relation is transitive w.r.t. the state s).

The hybrid logic $HL(@, \downarrow)$ is known to be undecidable (see, for instance, [2, 3, 7]), but some of its sub-logics are decidable, in particular $HL(@)$ (hybrid logic with the satisfaction operator). $HL(@)$ allows one to define many frame properties that are not definable in modal logics, such as, for instance, irreflexivity, asymmetry and antisymmetry. The interest of $HL(@)$ is due also to the fact that the satisfiability problem for a larger fragment of $HL(@, \downarrow)$ can be reduced to $HL(@)$ by means of satisfiability preserving translations [7, 9].

The tableaux system for full (quantified) hybrid logic defined by Blackburn and Marx [4] does not terminate even for formulae in $HL(@)$, but recently some effort has been devoted to devise tableau calculi that ensure termination in the case of decidable hybrid logics. Bolander and Bräuner [6] have shown how to decide satisfiability in $HL(@)$ (equipped also with the universal modality) by use of a loop-checking mechanism during tableau construction.

This paper introduces a terminating tableau system for $HL(@)$ that does not need loop-checks. Differently from other calculi for $HL(@)$ enjoying the same property [8, 9], termination does not rely on any specific (and sometimes complicated) tableau construction procedure. In other terms, the calculus enjoys a *strong termination property*. Moreover, the termination proof is quite simple and is based on general properties of the system whose underlying intuition is easy to understand. As it will be pointed out, the fundamental property ensuring termination fails with the (unrestricted) addition of the binder.

The essential differences among the tableau systems for hybrid logics defined in [4, 5, 6, 8, 9] concern the possible use of prefixes and the way how “equalities” are treated, i.e. formulae of the form $@_s t$, where s and t are nominals, asserting that s and t name the same state. The system presented in this paper is similar to the calculus defined in [9], but with a modification in the formulation of the equality rule that ensures termination independently of the tableau construction strategy.

2 Syntax and semantics of $HL(@)$

In this section, we present the syntax and semantics of the “uni-modal” version of $HL(@)$, but obviously it is straightforward to extend the definitions to the multimodal case.

Let Nom and P be disjoint sets of propositional letters. Elements of Nom are called *nominals*. We

shall use the letters s, t, u, v , possibly with indexes, as metavariables for nominals. An *atom* is an element of $Nom \cup P$. The set of formulae in $HL(@)$ is defined by the following grammar :

$$F := p \mid \neg F \mid F \wedge F \mid F \vee F \mid \Box F \mid \Diamond F \mid @_s F$$

where p is an atom and s a nominal.

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

If $\mathcal{M} = \langle W, R, N, I \rangle$ is an interpretation, $w \in W$ and F a formula, the relation $\mathcal{M}, w \models F$ (\mathcal{M} satisfies F at w) is inductively defined as follows :

1. $\mathcal{M}, w \models p$ if $p \in I(w)$, for $p \in P$.
2. $\mathcal{M}, w \models s$ if $N(s) = w$.
3. $\mathcal{M}, w \models \neg F$ if $\mathcal{M}, w \not\models F$.
4. $\mathcal{M}, w \models F \wedge G$ if $\mathcal{M}, w \models F$ and $\mathcal{M}, w \models G$.
5. $\mathcal{M}, w \models F \vee G$ if either $\mathcal{M}, w \models F$ or $\mathcal{M}, w \models G$.
6. $\mathcal{M}, w \models \Box F$ if for each w' such that wRw' , $\mathcal{M}, w' \models F$.
7. $\mathcal{M}, w \models \Diamond F$ if there exists w' such that wRw' and $\mathcal{M}, w' \models F$.
8. $\mathcal{M}, w \models @_s A$ if $\mathcal{M}, N(s) \models A$.

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

It is worth pointing out that, for any nominal s and formula F :

$$\neg @_s F \equiv @_s \neg F \quad \neg \Diamond F \equiv \Box \neg F \quad \neg \Box F \equiv \Diamond \neg F$$

This allows one to restrict attention to formulae in negation normal form (where negation dominates only atoms) without loss of generality.

3 The tableau system for $HL(@)$

The tableau system we are now going to present “internalizes” prefixes, like in [4] and (partially) [9]. Tableau nodes are in fact labelled by sets of *satisfaction statements*, i.e. assertion of the form $@_s F$. A formula of the form $@_s F$ will be called *labelled by s* . For the sake of simplicity, we assume that formulae are in negation normal form.

The initial tableau for a formula F is a node labelled by the singleton $\{@_s F\}$, where s is a new nominal. $@_s F$ is called the *initial formula* of the tableau, and $\{@_s F\}$ the *initial set*. Analogously, the tableau for a set of satisfaction statements S is a node labelled by S (the initial set). If S is the initial set of a tableau T , then T is said to be *rooted at S* .

Since we shall often need to refer to the set of nominals occurring in the initial set, we give it a name : if T is a tableau rooted at S , then

$$C_T = \{t \mid t \text{ is a nominal occurring in } S\}$$

Each tableau branch is equipped with a function γ , that maps each nominal occurring in the branch with a set of nominals, called its *descendants*. The function γ is modified when the \diamond -rule is applied and is used by the rule treating equalities, i.e. formulae of the form $@_s t$ where t is a nominal. The function γ associated to the single (one node) branch of an initial tableau T maps every nominal occurring in the initial set to the empty set, i.e. $\gamma(s) = \emptyset$ for every nominal $s \in C_T$ (i.e. initially nominals have no descendants).

A tableau node S is *closed* if either S contains $@_s p$ and $@_s \neg p$ for some nominal s and atom p , or it contains $@_s \neg s$ for some nominal s . A node is *open* if it is not closed. A tableau branch is open if all its nodes are open (otherwise it is closed) and it is *complete* if no rule can be applied to expand it further. A tableau is closed if all its branch are closed, otherwise it is open.

In the sequel, sets of formulae will be written as comma separated sequences of formulae. The expansion rules are shown in Figure 1. The boolean, modal and label rules are called *logical rules*. In the formulation of the rules, the function γ associated to the expanded branch is not indicated explicitly, since it only changes, as explained below, with applications of the \diamond -rule.

Boolean Rules		
$\frac{@_s(F \wedge G), S}{@_s F, @_s G, @_s(F \wedge G), S} (\wedge)$	$\frac{@_s(F \vee G), S}{@_s F, @_s(F \vee G), S \quad @_s G, @_s(F \vee G), S} (\vee)$	
Modal Rules		Label Rule
$\frac{@_s \Box F, @_s \Diamond t, S}{@_t F, @_s \Box F, @_s \Diamond t, S} (\Box)$	$\frac{@_s \Diamond F, S}{@_s \Diamond t, @_t F, @_s \Diamond F, S} (\Diamond)$ <p style="text-align: center; font-size: small;">where t is a nominal new in the branch</p>	$\frac{@_s @_t F, S}{@_t F, @_s @_t F, S} (@)$
Equality Rule		
$\frac{@_s t, S}{(S[s \mapsto t])^{-\gamma(s)}} (=)$		

FIG. 1 – The expansion rules

Note that the logical rules are all *non-destructive*, i.e. they do not “consume” the expanded formula.¹ They are subjected to the following applicability restrictions :

- The \wedge -rule is applicable only if either $@_s F$ or $@_s G$ do not belong to S .
- The \vee -rule is applicable only if none of $@_s F$ and $@_s G$ belong to S .
- The \Box -rule is applicable only if $@_t F \notin S$.
- The \Diamond -rule is applicable only if F is not a nominal and there is no nominal u such that $@_s \Diamond u \in S$

¹An alternative is considered in Section 6.

and $@_u F \in S$.

- The label rule is applicable only if $@_t F \notin S$.

Moreover, although it is not necessary, we convene that the equality rule is applicable only if $s \neq t$.

When the \diamond -rule is applied to expand a branch \mathcal{B} , the function γ associated to \mathcal{B} is modified. First of all, it is initialized to \emptyset for the newly introduced term t . Moreover, t is recorded as a descendant of s and all its ancestors. Formally, if γ is the function associated to \mathcal{B} before the expansion, then the function associated to \mathcal{B} after the expansion is γ' such that :

$$\gamma'(u) = \begin{cases} \emptyset & \text{if } u = t \\ \gamma(u) \cup \{t\} & \text{if either } u = s \text{ or } s \in \gamma(u) \\ \gamma(u) & \text{otherwise} \end{cases}$$

Intuitively, if $t \in \gamma(s)$, then either t has been newly introduced in the branch by application of the \diamond -rule to a formula of the form $@_s \diamond F$, i.e. t is a “child” of s , or t is a child of a child of s , etc.

It is worth pointing out that, for every nominal s , $\gamma(s)$ does not contain any nominal in C_T in any tableau branch.

Let's now turn to explain the equality rule. When it is applied to expand a node $@_s t, S$, every occurrence of s in S is replaced by t and every formula containing a descendant of s is deleted. The notations used in the formulation of the rule are defined as follows :

- $S[s \mapsto t]$ is obtained from S replacing every occurrence of the nominal s with t ;
- if Γ is a set of nominals, then $S^{-\Gamma}$ is the subset of S containing only formulae where no nominal in Γ occurs.

Therefore $(S[s \mapsto t])^{-\gamma(s)}$ is obtained from S by replacing s with t , and then deleting all the formulae involving some descendant of s .

When the rule is applied, s is said to be *replaced in the branch* and nominals in $\gamma(s)$ that occur in $S[s \mapsto t]$ are called *deleted in the branch*. Let's point out that nominals in C_T are never deleted.

Note that if nominals in $\gamma(s)$ were not deleted when applying the equality rule, i.e. if the simpler rule is used :

$$\frac{@_s t, S}{S[s \mapsto t]}$$

then tableau construction might not terminate. This is shown in Figure 2, where an infinite tableau is built by use of the simpler equality rule² (and a particular rule application order). What happens there is that each of the nominals t_0, t_2, t_4, \dots , before being replaced by s , generates its child, that begins to live its own life. In the figure, the applied rules and the expanded formulae are shown on the right of inference lines.

Figure 3 shows that the same initial set of formulae and the same rule application order produce a finite tableau when nominals are deleted by the equality rule. The first application of the equality rule deletes t_1 , since $\gamma(t_0) = \{t_1\}$. The next rule reintroduces t_1 as a child of s .³ The second application of the equality rule deletes the nominal $t_3 \in \gamma(t_2)$. The last node S of the tableau cannot be expanded because, except for formulae of the form $@_u \diamond t$, $@_u p$ (where u, t are nominals) and $@_s s$, it contains :

²The rules used in this example, in particular the equality rule, are essentially (reformulations of) the rules of the system defined in [9].

³Although a nominal new in the branch should be used instead of t_1 , we use here the same name for ease of comparison with the tableau in Figure 2.

$@_s \diamond p$ and also $@_s \diamond t_1, @_{t_1} p$
 $@_s (s \wedge \diamond p)$ and also $@_s s$ and $@_s \diamond p$
 $@_{t_1} \diamond (s \wedge \diamond p)$ and also $@_{t_1} \diamond s$ and $@_s (s \wedge \diamond p)$
 $@_s \diamond (s \wedge \diamond p)$ and also $@_s \diamond s$ and $@_s (s \wedge \diamond p)$
 $@_s \square \diamond (s \wedge \diamond p)$ and for all u such that $@_s \diamond u \in S$ (namely t_1 and s), S
contains also $@_u \diamond (s \wedge \diamond p) : @_{t_1} \diamond (s \wedge \diamond p)$ and $@_s \diamond (s \wedge \diamond p)$

$$\begin{array}{c}
\frac{@_s \diamond (s \wedge \diamond p), @_s \square \diamond (s \wedge \diamond p)}{(\diamond : @_s \diamond (s \wedge \diamond p))} \\
\frac{@_s \diamond t_0, @_{t_0} (s \wedge \diamond p), @_s \diamond (s \wedge \diamond p), @_s \square \diamond (s \wedge \diamond p)}{(\wedge : @_{t_0} (s \wedge \diamond p))} \\
\frac{@_{t_0} s, @_{t_0} \diamond p, @_s \diamond t_0, @_{t_0} (s \wedge \diamond p), @_s \diamond (s \wedge \diamond p), @_s \square \diamond (s \wedge \diamond p)}{(\diamond : @_{t_0} \diamond p)} \\
\frac{@_{t_0} \diamond t_1, @_{t_1} p, @_{t_0} s, @_{t_0} \diamond p, @_s \diamond t_0, @_{t_0} (s \wedge \diamond p), @_s \diamond (s \wedge \diamond p), @_s \square \diamond (s \wedge \diamond p)}{(\doteq : @_{t_0} s)} \\
\frac{@_s \diamond t_1, @_{t_1} p, @_s \diamond p, @_s \diamond s, @_s (s \wedge \diamond p), @_s \diamond (s \wedge \diamond p), @_s \square \diamond (s \wedge \diamond p)}{(\square : @_s \diamond t_1, @_s \square \diamond (s \wedge \diamond p))} \\
\frac{@_{t_1} \diamond (s \wedge \diamond p), @_s \diamond t_1, @_{t_1} p, @_s \diamond p, @_s \diamond s, @_s (s \wedge \diamond p), @_s \diamond (s \wedge \diamond p), @_s \square \diamond (s \wedge \diamond p)}{(\diamond : @_{t_1} \diamond (s \wedge \diamond p))} \\
\frac{@_{t_1} \diamond t_2, @_{t_2} (s \wedge \diamond p), @_{t_1} \diamond (s \wedge \diamond p), @_s \diamond t_1, @_{t_1} p, @_s \diamond p, @_s \diamond s, @_s (s \wedge \diamond p), @_s \diamond (s \wedge \diamond p), @_s \square \diamond (s \wedge \diamond p)}{(\wedge : @_{t_2} (s \wedge \diamond p))} \\
\frac{@_{t_2} s, @_{t_2} \diamond p, @_{t_1} \diamond t_2, @_{t_2} (s \wedge \diamond p), @_{t_1} \diamond (s \wedge \diamond p), @_s \diamond t_1, @_{t_1} p, @_s \diamond p, @_s \diamond s, @_s (s \wedge \diamond p), @_s \diamond (s \wedge \diamond p), @_s \square \diamond (s \wedge \diamond p)}{(\diamond : @_{t_2} \diamond p)} \\
\frac{@_{t_2} \diamond t_3, @_{t_3} p, @_{t_2} s, @_{t_2} \diamond p, @_{t_1} \diamond t_2, @_{t_2} (s \wedge \diamond p), @_{t_1} \diamond (s \wedge \diamond p), @_s \diamond t_1, @_{t_1} p, @_s \diamond p, @_s \diamond s, @_s (s \wedge \diamond p), @_s \diamond (s \wedge \diamond p), @_s \square \diamond (s \wedge \diamond p)}{(\doteq : @_{t_2} s)} \\
\frac{@_s \diamond t_3, @_{t_3} p, @_s \diamond p, @_{t_1} \diamond s, @_s (s \wedge \diamond p), @_{t_1} \diamond (s \wedge \diamond p), @_s \diamond t_1, @_{t_1} p, @_s \diamond s, @_s \diamond (s \wedge \diamond p), @_s \square \diamond (s \wedge \diamond p)}{(\square : @_s \diamond t_3, @_s \square \diamond (s \wedge \diamond p))} \\
\vdots
\end{array}$$

FIG. 2 – An infinite tableau where the equality rule does not delete nominals

$@_s \diamond(s \wedge \diamond p), @_s \square \diamond(s \wedge \diamond p)$		
$@_s \diamond t_0, @_{t_0}(s \wedge \diamond p), @_s \diamond(s \wedge \diamond p), @_s \square \diamond(s \wedge \diamond p)$		$(\diamond : @_s \diamond(s \wedge \diamond p))$
$@_{t_0} s, @_{t_0} \diamond p, @_s \diamond t_0, @_{t_0}(s \wedge \diamond p), @_s \diamond(s \wedge \diamond p), @_s \square \diamond(s \wedge \diamond p)$		$(\wedge : @_{t_0}(s \wedge \diamond p))$
$@_{t_0} \diamond t_1, @_{t_1} p, @_{t_0} s, @_{t_0} \diamond p, @_s \diamond t_0, @_{t_0}(s \wedge \diamond p), @_s \diamond(s \wedge \diamond p), @_s \square \diamond(s \wedge \diamond p)$		$(\diamond : @_{t_0} \diamond p)$
$@_s \diamond p, @_s \diamond s, @_s(s \wedge \diamond p), @_s \diamond(s \wedge \diamond p), @_s \square \diamond(s \wedge \diamond p)$		$(=: @_{t_0} s)$
$@_s \diamond t_1, @_{t_1} p, @_s \diamond p, @_s \diamond s, @_s(s \wedge \diamond p), @_s \diamond(s \wedge \diamond p), @_s \square \diamond(s \wedge \diamond p)$		$(\diamond : @_s \diamond p)$
$@_{t_1} \diamond(s \wedge \diamond p), @_s \diamond t_1, @_{t_1} p, @_s \diamond p, @_s \diamond s, @_s(s \wedge \diamond p), @_s \diamond(s \wedge \diamond p), @_s \square \diamond(s \wedge \diamond p)$		$(\square : @_s \diamond t_1, @_s \square \diamond(s \wedge \diamond p))$
$@_{t_1} \diamond t_2, @_{t_2}(s \wedge \diamond p), @_{t_1} \diamond(s \wedge \diamond p), @_s \diamond t_1, @_{t_1} p, @_s \diamond p, @_s \diamond s, @_s(s \wedge \diamond p), @_s \diamond(s \wedge \diamond p), @_s \square \diamond(s \wedge \diamond p)$		$(\diamond : @_{t_1} \diamond(s \wedge \diamond p))$
$@_{t_2} s, @_{t_2} \diamond p, @_{t_1} \diamond t_2, @_{t_2}(s \wedge \diamond p), @_{t_1} \diamond(s \wedge \diamond p), @_s \diamond t_1, @_{t_1} p, @_s \diamond p, @_s \diamond s, @_s(s \wedge \diamond p), @_s \diamond(s \wedge \diamond p), @_s \square \diamond(s \wedge \diamond p)$		$(\wedge : @_{t_2}(s \wedge \diamond p))$
$@_{t_2} \diamond t_3, @_{t_3} p, @_{t_2} s, @_{t_2} \diamond p, @_{t_1} \diamond t_2, @_{t_2}(s \wedge \diamond p), @_{t_1} \diamond(s \wedge \diamond p), @_s \diamond t_1, @_{t_1} p, @_s \diamond p, @_s \diamond s, @_s(s \wedge \diamond p), @_s \diamond(s \wedge \diamond p), @_s \square \diamond(s \wedge \diamond p)$		$(\diamond : @_{t_2} \diamond p)$
$@_s \diamond p, @_{t_1} \diamond s, @_s(s \wedge \diamond p), @_{t_1} \diamond(s \wedge \diamond p), @_s \diamond t_1, @_{t_1} p, @_s \diamond s, @_s \diamond(s \wedge \diamond p), @_s \square \diamond(s \wedge \diamond p)$		$(=: @_{t_2} s)$
$@_s s, @_s \diamond p, @_{t_1} \diamond s, @_s(s \wedge \diamond p), @_{t_1} \diamond(s \wedge \diamond p), @_s \diamond t_1, @_{t_1} p, @_s \diamond s, @_s \diamond(s \wedge \diamond p), @_s \square \diamond(s \wedge \diamond p)$		$(\wedge : @_s(s \wedge \diamond p))$

FIG. 3 – A finite tableau with nominal deletion

4 Termination

In this section we prove that tableau construction always terminates, independently of the rule application strategy.

We shall make use of the notion of *father* of a nominal $t \notin C_T$ introduced by application of the \diamond -rule in a branch \mathcal{B} : if t is newly introduced in \mathcal{B} by expanding a formula of the form $@_s \diamond F$, then $father(t, \mathcal{B}) = s$, and t is called a *child* of s . The set of children of a nominal in a branch is $children(s, \mathcal{B}) = \{u \mid s = father(u, \mathcal{B})\}$.

Formulae of the form $@_s \diamond t$, for s, t nominals, are called *relational formulae*.

The key result that ensures termination is a kind of ‘‘subformula property’’ enjoyed by the calculus. In order to state it, we define the set S_T^* containing every formula that can be obtained from a subformula of some formula in the initial set, by replacing nominals with other nominals still occurring in the initial set : when T is a tableau rooted at S ,

$$S_T^* = \{F \mid F = G[t_1 \mapsto u_1, \dots, t_n \mapsto u_n] \text{ for some subformula } G \text{ of some } A \in S, \text{ and } u_1, \dots, u_n \in C_T\}$$

(t_1, \dots, t_n are nominals occurring in G). Note that, obviously, every nominal t occurring in a formula of S_T^* occurs in the initial set (i.e. $t \in C_T$). Moreover, S_T^* is finite and closed with respect to subformulae.

The key property of the system, that will be extensively used in the sequel, is the following :

Lemma 1 (Quasi-subformula property) *If T is a tableau, and $@_s F$ is a formula occurring in some node of T , then either $@_s F$ is relational or $F \in S_T^*$.*

Therefore any tableau contains a finite number of non-relational formulae labelled by the same nominal. I.e. for every nominal s , the set

$$\{@_s F \mid @_s F \text{ occurs in some node of } T \text{ and } @_s F \text{ is not relational}\}$$

is finite.

The result can easily be proved by induction on tableaux. The following properties are direct consequences of Lemma 1 :

1. If $@_s t$ occurs in a node of T , then $t \in C_T$. Therefore, in the applications of the equality rule, nominals are always replaced by nominals in C_T .
2. A nominal $t \notin C_T$ may occur in tableau nodes only in relational formulae, i.e. in the form $@_s \diamond t$, or as the label of a formula $@_t F$ (where $F \in S_T^*$ does not contain t).

Next, we have to prove that the number of nominals occurring in a tableau branch is finite. First of all we note that, in a tableau branch \mathcal{B} , a nominal s cannot generate more children than the number of (non relational) formulae of the form $@_s \diamond F$ that occur in \mathcal{B} , and they are finite (Lemma 1). In fact, let us assume that a child t of s is generated from $@_s \diamond F$, with $@_s \diamond t$ and $@_t F$, in a node S_n of \mathcal{B} . Obviously, if $@_s \diamond t$ and $@_t F$ keep on being present in all the nodes following S_n in \mathcal{B} , then the restriction on the applicability of the \diamond -rule prevents $@_s \diamond F$ from generating a new child. Otherwise such formulae disappear in a node S_{n+k} in \mathcal{B} . This can happen for two reasons : either because t is replaced by another nominal u , and in this case the presence of $@_s \diamond u$ and $@_u F$ prevents a new expansion of $@_s \diamond F$; or else because s itself disappears (either because it is replaced or because it is deleted), and in this case the nominal s cannot be re-used in \mathcal{B} to generate a new occurrence of $@_s \diamond F$, since nominals generated by the \diamond -rule are always new in the branch.

As a consequence :

Corollary 1 For every branch \mathcal{B} and nominal s , $children(s, \mathcal{B})$ is finite.

Thanks to nominal deletion in the equality rule, the following result holds :

Lemma 2 If s and t are nominals, $t \notin C_T$, and $@_s \diamond t$ occurs in some node of a tableau branch \mathcal{B} , then $s = father(t, \mathcal{B})$.

Proof. The result is proved by induction on the number of nodes between the root of \mathcal{B} and the node where $@_s \diamond t$ occurs for the first time. The base of the induction is vacuously true, since every nominal occurring in the initial formula belongs to C_T .

The only non-trivial case in the induction step is when the equality rule is applied :

$$\frac{@_u v, S}{(S[u \mapsto v])^{-\gamma(u)}} (=)$$

Let $@_s \diamond t = @_s \diamond t'[u \mapsto v]$ be a formula occurring in $(S[u \mapsto v])^{-\gamma(u)}$ and not in S . Hence, either $s' = u$ or $t' = u$ (or both). If $t' = u$, then $@_s \diamond t = @_s \diamond u[u \mapsto v] = @_s \diamond v$; since $@_u v$ occurs in a tableau node, by Lemma 1, $v \in C_T$ and there is nothing to prove.

So we are left with the case where $s' = u$ and $t' \neq u$, i.e. $@_s \diamond t = @_u \diamond t[u \mapsto v] = @_v \diamond t$, with $t \notin C_T$. By the induction hypothesis, since $@_u \diamond t \in S$, $u = father(t, \mathcal{B})$; hence $t \in \gamma(u)$ is deleted by the equality rule and $@_s \diamond t$ cannot occur in $(S[u \mapsto v])^{-\gamma(u)}$.

The property stated in Lemma 2 ensures that nominals that do not occur in the initial formula can only inherit formulae from their fathers. I.e. it cannot happen that a formula of the form $@_t F$ (for $t \notin C_T$) appears in a tableau because it is obtained by application of the \square -rule from some $@_u \square F$ and $@_u \diamond t$ where u is not the father of t .

Let us now define the *maximal modal degree* of a nominal in a branch \mathcal{B} as follows : if $degree(F)$ is the modal degree of F , then

$$Deg(s, \mathcal{B}) = \max\{degree(F) \mid @_s F \text{ occurs in } \mathcal{B}\}$$

It is worth pointing out that $Deg(s, \mathcal{B}) \in \mathbb{N}$, because the set of formulae labelled by s in \mathcal{B} is finite, by Lemma 1. From Lemma 2, it is easily proved that, for any tableau branch \mathcal{B} and any chain of nominals s_0, s_1, s_2, \dots with $s_i = father(s_{i+1}, \mathcal{B})$, the maximal modal degree of s_i is strictly decreasing. This is what is stated by next Lemma, whose proof is again an induction on tableau.

Lemma 3 Let T be a tableau and \mathcal{B} a branch in T . If $t \notin C_T$ and $father(t, \mathcal{B}) = s$, then $Deg(t, \mathcal{B}) < Deg(s, \mathcal{B})$.

From this property it follows that :

Corollary 2 In any tableau branch \mathcal{B} , every chain of nominals s_0, s_1, s_2, \dots , with $s_i = father(s_{i+1}, \mathcal{B})$, is finite.

An immediate consequence of Corollaries 1 and 2 (and the fact that C_T is finite) is :

Corollary 3 The number of nominals that occur in a tableau branch is finite. In particular, the “father-of” relation arranges the set of nominals occurring in a tableau branch in a finite set of well-founded and finitely branching trees.

Therefore, in any tableau the number of relational formulae labelled by the same nominal s , i.e. formulae of the form $@_s \diamond t$, is finite. From this fact and Lemma 1, it follows that :

Corollary 4 *For every tableau branch \mathcal{B} and nominal s , the set*

$$\{F \mid @_s F \text{ occurs in } \mathcal{B}\}$$

is finite.

We have now all we need to prove that the system enjoys the strong termination property :

Theorem 1 (Termination) *Every tableau is finite.*

Proof. By Corollaries 3 and 4, if a tableau has an infinite branch \mathcal{B} , then (by the applicability restrictions on the rules and the fact that the logical rules do not consume their premisses) there is at least a formula $@_s F$ that is either deleted or modified by application of the equality rule, and then reintroduced in \mathcal{B} . If $@_s F$ is deleted, it is because s is deleted and the same nominal cannot be used again in the branch. If it is modified, i.e. it becomes $(@_s F)[u \mapsto t]$ for some u and t , then every occurrence of u is replaced in the same node, therefore, again, a formula containing u cannot be reintroduced by any expansion rule.⁴ Therefore, no formula can be deleted or modified in a branch, and reintroduced later on, so every tableau branch is finite.

Let us now consider a possible extension of the system to $HL(@, \downarrow)$, obtained by adding the following rule :

$$\frac{@_s \downarrow x.A, S}{@_s A[x \mapsto s], @_s \downarrow x.A, S}$$

(applicable only if $A[x \mapsto s] \notin S$). Now, obviously, Lemma 1 fails. This corresponds to the fact that the binder “enables us to create a name for the here-and-now” [3]. Therefore, with unrestricted use of the binder, an infinite number of “subformulae” of a given formula F can appear in a tableau branch, each using different nominals to instantiate state variables, even though F refers to the corresponding states only implicitly.

5 Soundness and Completeness

Soundness can easily be proved in the standard way, by showing that each rule preserves satisfiability. In particular, the soundness of the equality rule is due to the following form of the *substitution theorem* : if F is a formula, $\mathcal{M} = \langle W, R, N, I \rangle$ an interpretation and $w \in W$, then for all nominals s, t such that $N(s) = N(t)$, $\mathcal{M}, w \models F$ if and only if $\mathcal{M}, w \models F[s \mapsto t]$.

The completeness proof also follows the usual technique of showing that :

Lemma 4 *If \mathcal{B} is a complete and open branch of a tableau rooted at S_0 , then S_0 is satisfiable.*

Proof. Standard completeness proofs work by observing that the union of all the node labels in \mathcal{B} is *downward saturated*, and then showing that any such set has a model. In our case, the guideline is the same, but deletion and replacement of nominals raise some difficulties, that are overcome by exploiting

⁴Note that we exploit here the fact that nominals introduced by the \diamond -rule are *new in the branch*. Obviously, once termination is proved, this restriction can be weakened to require only that such nominals are new in the node.

the fact that \mathcal{B} is finite. So we first consider the set labelling the last node of \mathcal{B} , that is shown to be downward saturated and, consequently, satisfiable. Then we show that satisfiability propagates upward to the root node, if nominals that are deleted in \mathcal{B} , and every formula containing them, are ignored. This is sufficient because nominals occurring in the initial set are never deleted.

In what follows, we say that a formula F occurs in \mathcal{B} (and \mathcal{B} contains F) to mean that F occurs in some node of \mathcal{B} .

Since \mathcal{B} is finite by Theorem 1, $\mathcal{B} = S_0, S_1, \dots, S_k$ for some k . If \mathcal{B} is open and complete, then S_k is *downward saturated*, i.e. :

1. S_k does not contain any formula of the form $@_s \neg s$, and it does not contain two formulae of the form $@_s p$ and $@_s \neg p$ for some atom p .
2. If $@_s(A \wedge B) \in S_k$, then $@_s A, @_s B \in S_k$.
3. If $@_s(A \vee B) \in S_k$, then either $@_s A \in S_k$ or $@_s B \in S_k$.
4. If $@_s \diamond A \in S_k$ (for A that is not a nominal), then there is a nominal t with $@_s \diamond t, @_t A \in S_k$.
5. If $@_s \diamond t, @_s \Box A \in S_k$, then $@_t A \in S_k$.
6. If $@_s @_t A \in S_k$, then $@_t A \in S_k$.
7. If $@_s t \in S_k$ then $s = t$, otherwise the equality rule would be applicable to expand \mathcal{B} .

We define the interpretation $\mathcal{M}' = \langle W, R, N', I \rangle$ as follows :

$$\begin{aligned} W &= \{s \mid s \text{ occurs in } S_k\}; \\ R &= \{(s, t) \mid @_s \diamond t \in S_k\}; \\ &\text{for every nominal } s \text{ occurring in } S_k, N'(s) = s; \\ I(s) &= \{p \mid @_s p \in S_k\}. \end{aligned}$$

Exploiting the fact that S_k is downward saturated, it can easily be proved by induction on A that if $@_s A \in S_k$, then $\mathcal{M}', N'(s) \models A$.

Next, we define an equivalence relation on nominals (with respect to the branch \mathcal{B}) as follows : $s \sim t$ if $@_s t$ occurs in \mathcal{B} . The relation \approx is the reflexive, symmetric and transitive closure of \sim .

Since N' is undefined for nominals that do not occur in S_k , we can safely extend it to interpret all the nominals occurring in \mathcal{B} . Let w^* be any fixed element of W . Then N is the extension of N' such that for all nominals u occurring in \mathcal{B} :

$$N(u) = \begin{cases} N'(u) & \text{if } u \in W, \text{ i.e. } u \text{ occurs in } S_k \\ N'(r) & \text{if for some } r \in W, u \approx r \\ w^* & \text{otherwise} \end{cases}$$

It is clear that if $@_s t$ occurs in \mathcal{B} , then $N(s) = N(t)$, and if u is deleted in \mathcal{B} , then $N(u) = w^*$.

If $\mathcal{M} = \langle W, R, N, I \rangle$, obviously, it still holds that for every $@_s A \in S_k$, $\mathcal{M}, N(s) \models A$.

We now prove that the satisfaction property propagates upwards, restricting our attention to nominals that are not deleted in \mathcal{B} . Let us say that \mathcal{M} is a \mathcal{B} -model of a node S of \mathcal{B} if for every formula $@_s F \in S$ that contains only nominals that are never deleted in \mathcal{B} , $\mathcal{M}, N(s) \models F$. Then we show that, for every $i = 0, \dots, k-1$:

- (•) if \mathcal{M} is a \mathcal{B} -model of S_{i+1} , then \mathcal{M} is a \mathcal{B} -model of S_i .

When $i = 0$ this is what we want, because the initial formula of the tableau contains only nominals in C_T that are never deleted.

In order to prove (\bullet) , the cases where S_{i+1} is obtained from S_i by applying a logical rule are trivial, since $S_i \subseteq S_{i+1}$. So the only non-trivial case is the equality rule, where :

$$\begin{aligned} S_i &= @_s t, S' \\ S_{i+1} &= (S'[s \mapsto t])^{-\gamma(s)} \end{aligned}$$

By the induction hypothesis, for every formula $@_u F \in S_{i+1} = (S'[s \mapsto t])^{-\gamma(s)}$ containing only nominals that are never deleted in the branch, $\mathcal{M}, N(u) \models F$. Note that all $t_i \in \gamma(s)$ occurring in S_i are deleted in \mathcal{B} .

Since $N(s) = N(t)$, by definition, $\mathcal{M}, N(s) \models t$.

Let now $@_u A$ be a formula in $S' = S_i \setminus \{@_s t\}$, containing only nominals that are never deleted in the branch. Then also $(@_u A)[s \mapsto t]$ contains only nominals that are never deleted in \mathcal{B} ; in fact, the only nominal possibly occurring in $(@_u A)[s \mapsto t]$ and not in $@_u A$ is t , and $t \in C_T$ (Lemma 1), so it cannot be deleted. Hence, by the induction hypothesis $\mathcal{M}, N(u[s \mapsto t]) \models A[s \mapsto t]$, where $u[s \mapsto t] = t$ if $u = s$, and $u[s \mapsto t] = u$ otherwise. By the substitution theorem, since $N(s) = N(t)$, $\mathcal{M}, N(u[s \mapsto t]) \models A$. If $u[s \mapsto t] = u$, we are done. Otherwise, if $u[s \mapsto t] = t$ then $u = s$, so $N(u[s \mapsto t]) = N(t) = N(s) = N(u)$. Hence, also in this case $\mathcal{M}, N(u) \models A$.

Completeness follows directly from Lemma 4.

Theorem 2 (Completeness) *If S is unsatisfiable, then every complete tableau for S is closed.*

6 Towards an implementation of the system

The expansion rules given in Section 3 have been formulated so as to keep their presentation simple, and to ease the termination and completeness proofs. However, the applicability restrictions on the logical rules are rather expensive, since they involve set-membership tests. In this section we briefly show how the rules can be modified, getting closer to an implementation of the system.

To this aim, let us clarify the interplay between the rule applicability restrictions and non-destructiveness. In the case of the modal rules, copying the expanded formulae in the lower nodes is essential to guarantee completeness, since it may be necessary to expand them more than once. On the contrary, in the boolean and label rules, one could well mark the expanded formula as “used” and do not expand it any more. The reason why the formula is copied in the lower node is to ensure termination, avoiding unnecessary re-applications of the modal rules.⁵ If the expanded formulae were consumed by such rules, in fact, an infinite tableau could be built for as simple a formula as $@_s \diamond(p \wedge q)$. In fact such a formula could be expanded infinitely many times if the \wedge -rule consumed its premise $@_{t_i}(p \wedge q)$:

$$\begin{array}{c} @_s \diamond(p \wedge q) \\ \hline @_s \diamond t_0, @_{t_0}(p \wedge q), @_s \diamond(p \wedge q) \\ \hline @_{t_0} p, @_{t_0} q, @_s \diamond t_0, @_s \diamond(p \wedge q) \\ \hline @_s \diamond t_1, @_{t_1}(p \wedge q), @_{t_0} p, @_{t_0} q, @_s \diamond t_0, @_s \diamond(p \wedge q) \\ \hline \vdots \end{array} \begin{array}{l} (\diamond) \\ (\wedge) \\ (\diamond) \\ (\wedge) \end{array}$$

⁵Actually, it is necessary to keep only (boolean or satisfaction) formulae that are in turn produced by the application of a modal rule.

However, instead of marking boolean and label formulae as expanded, a different approach can be followed. The boolean and label rules become destructive :

Boolean Rules		Label Rule
$\frac{\@_s(F \wedge G), S}{\@_s F, \@_s G, S} (\wedge)$	$\frac{\@_s(F \vee G), S}{\@_s F, S \quad \@_s G, S} (\vee)$	$\frac{\@_s \@_t F, S}{\@_t F, S} (@)$

The \Box -rule stays unchanged, since, obviously, its premisses must not be consumed. In order to avoid unnecessary re-applications of such a rule, it is sufficient to keep track of the nominals t which have “inherited” F from a given formula of the form $\@_s \Box F$, and establish that the \Box -rule is never applied more than once to expand the same pair of formulae (this is common practice in labelled modal systems).

Let us consider the \Diamond -rule. The reason why a formula of the form $\@_s \Diamond F$ may need to be re-used is due to the equality rule : if s is replaced by t , then the children of s are deleted and information about such “lost” children must be reproduced by expanding $\@_t \Diamond F[s \mapsto t]$. The same behaviour can be obtained by a different marking mechanism : when $\@_s \Diamond F$ is expanded, it stays in the lower node, but marked as “used” (and marked formulae are never expanded). When the equality rule is applied to expand $\@_s t$, every formula of the form $\@_t \Diamond F[s \mapsto t]$ is unmarked in the lower node. The equivalence between the system defined in Section 3 and its modified version presented here is straightforward.

As a final remark, we point out that when a nominal t is deleted by the equality rule, it is not necessary to perform an “occur check” on every formula of the set ; in fact, thanks to Lemma 1, only formulae labelled by t and relational formulae of the form $\@_s \Diamond t$ are to be deleted.

7 Concluding remarks

This paper introduces a tableau system for hybrid logic with the satisfaction operator that can be used to decide validity/satisfiability of formulae in $HL(@)$. The particular feature of the system is that any tableau is finite, independently of any rule application strategy (in other terms, the calculus enjoys strong termination), and finiteness does not depend on loop-checks. The same system can be used to decide satisfiability of formulae in the fragment $HL(@, \downarrow) \setminus \{\Box, \downarrow\}$, namely the fragment of hybrid logic with the satisfaction operator and the binder, where the binder never occurs in the scope of a \Box modality. In fact, there exist satisfiability preserving translations of formulae in such a fragment into formulae in $HL(@)$ [7, 9].

Other terminating tableau calculi for $HL(@)$ with no loop-checking mechanisms have been defined in the literature. In [8] a prefixed tableau system is defined, where “equalities” are treated by means of two (four-premisses) rules. Termination is ensured only when a specific (and quite elaborated) tableau construction strategy is followed. The calculus defined in [9] “internalizes” prefixes by means of the satisfaction operator, yet the accessibility relation between states is represented by meta-linguistic expressions. Equalities are here treated by means of a substitution rule, which is the same as ours, but for the fact that nothing is deleted (and a given order on nominals is assumed) : it essentially corresponds to the “simple” equality rule used in the infinite tableau of Figure 2. In this case too, the termination of tableau construction is guaranteed only when following a rather complicated procedure. In both the above mentioned works, moreover, only sketchy termination proofs are given, whose underlying intuitions are rather obscure.

[6] gives modified versions of the systems in [8] and [9] (in both cases using prefixes and meta-

linguistic expressions to represent the accessibility relation), as well as the system defined in [4].⁶ For all the three calculi, termination of tableau construction relies on a loop checking mechanism, a standard technique used in connection with tableau systems for modal logics. Except for the treatment of equalities, the third of the three calculi mentioned above is similar to ours and it would be interesting to evaluate trade-offs and benefits of loop-checking on an empirical basis. As a next step of this work, we are planning to build implementations of the systems with and without loop-checks, in order to be able to compare their performances.

Références

- [1] S. ABITEBOUL, P. BUNEMANN, et D. SUCIU. *Data on the Web : From Relations to Semistructured Data and XML*. Morgan Kaufman, 2000.
- [2] C. ARECES, P. BLACKBURN, et M. MARX. A road-map on complexity for hybrid logics. In J. FLUM et M. RODRÍGUEZ-ARTALEJO, réds., *Computer Science Logic*, number 1683 in LNCS, pages 307–321. Springer, 1999. Proceedings of the 8th Annual Conference of the EACSL, Madrid, September 1999.
- [3] C. ARECES, P. BLACKBURN, et M. MARX. Hybrid logics : characterization, interpolation and complexity. *The Journal of Symbolic Logic*, 66(3) :977–1010, 2001.
- [4] P. BLACKBURN et M. MARX. Tableaux for quantified hybrid logic. In U. EGLY et C. FERMÜLLER, réds., *Automated Reasoning with Analytic Tableaux and Related Methods (TABLEAUX 2002)*, number 2381 in LNAI, pages 38–52. Springer Verlag, 2002.
- [5] T. BOLANDER et T. BRAÜNER. Two tableau-based decision procedures for hybrid logic. In H. SCHLINGLOFF, réd., *4th Methods for Modalities Workshop (M4M), Informatik-Bericht Nr. 194*, pages 79–96. Humboldt-Universität zu Berlin, 2005.
- [6] T. BOLANDER et T. BRAÜNER. Tableau-based decision procedures for hybrid logic. To appear in the *Journal of Logic and Computation*. Published online on August 12, 2006.
- [7] B. ten CATE et M. FRANCESCHET. On the complexity of hybrid logics with binders. In L. ONG, réd., *Proceedings of Computer Science Logic 2005*, volume 3634 of *Lecture Notes in Computer Science*, pages 339–354. Springer Verlag, 2005.
- [8] M. TZAKOVA. Tableau calculi for hybrid logics. In N. MURRAY, réd., *Automated Reasoning with Analytic Tableaux and Related Methods (TABLEAUX 1999)*, volume 1617 of *LNAI*, pages 278–292. Springer Verlag, 1999.
- [9] J. van EIJCK. Constraint tableaux for hybrid logics. Manuscript. CWI, Amsterdam, 2002.

⁶The three calculi are also extended to treat the universal modality.

Terminating tableaux for $HL(@)$ without loop-checking

Serenella CERRITO, inst1 Marta CIALDEA MAYER

Résumé

This paper introduces a terminating tableau system for hybrid logic with the satisfaction operator, $HL(@)$, that does not need loop-checks. Differently from other calculi for $HL(@)$ enjoying the same property, termination does not rely on any specific – and sometimes complicated – tableau construction procedure.

This document corresponds to a draft which has been completed on 2007, February the 15th.

Mots-clés additionnels et phrases : Decidability, Hybrid Modal Logic, Proof Systems, Tableaux, Termination