An Efficient Approach to Nominal Equalities in Hybrid Logic Tableaux

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

Marta Cialdea Mayer
Dipart. Informatica e Automazione
Università di Roma Tre
Italy

Abstract

Basic hybrid logic extends modal logic with the possibility of naming worlds by means of a distinguished class of atoms (called *nominals*) and the so-called *satisfaction operator*, that allows one to state that a given formula holds at the world named a, for some nominal a. Hence, in particular, hybrid formulae include "equality" assertions, stating that two nominals are distinct names for the same world. The treatment of such *nominal equalities* in proof systems for hybrid logics may induce many redundancies. This paper introduces an internalized tableau system for basic hybrid logic, significantly reducing such redundancies. The calculus enjoys a *strong termination property*: tableau construction terminates without relying on any specific rule application strategy, and no loop-checking is needed. The treatment of nominal equalities specific of the proposed calculus is briefly compared to other approaches. Its practical advantages are demonstrated by empirical results obtained by use of implemented systems. Finally, it is briefly shown how to extend the calculus to include the global and converse modalities.

1 Introduction

The semantics of modal logics is given by means of Kripke structures, that are essentially graphs whose vertices ("worlds" or "states") are classical interpretations. Their standard syntax, however, does not allow one to name vertices, hence one can say that a formula is true at a given world only in the meta-language. The peculiarity of hybrid logics is the extension of the standard modal syntax by means of *nominals* and a *satisfaction* operator (@), which makes it possible to express that a given formula holds at a given state in the object language itself. Nominals are just a distinguished sort of propositional letters, each of which is assumed to be true at exactly one world of the interpretation (a nominal is essentially the name of a world). This more expressive syntax can

be useful, for instance, when modal logics are used to formalise semi-structured databases, such as XML-documents. In [1], just one nominal, *root*, is added to CPDL in order to study path constraints in semistructured databases, while [8] and [19], for instance, use full hybrid languages in order to express this kind of data. In fact, XML-documents describe tree-like structures, or, more generally, (finite) graphs, when references are present. In that framework, attributes having ID type provide node identifiers, while some attributes are pointers to nodes, by making reference to their identifiers; such attributes are said to have the reference type IDREF (or IDREFS). One can observe that node identifiers correspond to nominals.

The satisfaction operator of hybrid logics allows one to move from the current state to a different, not necessarily accessible, one: a formula of the form $@_a F$, where a is a nominal and F any formula, means that F holds at the world named a. Hence, one can "jump" from the current state to a different one. In particular, when F itself is a nominal b, $@_a F$ says that a and b name the same world; for this reason a formula of the form $@_a b$ will be called a *nominal equality*.

Besides @ (and the usual modal operators $\Box$ and $\Diamond$), more expressive hybrid languages may contain state variables and the *binder* ($\downarrow$), that binds a variable to the current world, the converse modal operators $\Box^-$ and $\Diamond^-$, the universal modalities A and its dual E, and the difference modality D, and its dual (see, for instance, [3, 10]). The usual notation for the hybrid logic obtained by addition of operators $o_1, ..., o_k$ to the basic propositional modal logic K is $HL(o_1, ..., o_k)$. The logic $HL(\downarrow)$ is known to be undecidable [3, 2, 20], while any combination of the remaining operators mentioned above (possibly all) keeps the logic decidable.

Several proof systems have been defined for hybrid logics (see, for instance, [4, 9, 11, 12, 14, 15, 26]) and some provers have also been implemented, among which [5, 18, 21, 22, 30]. Since nominals and the satisfaction operator are the main peculiarities of hybrid languages, it is important to devise efficient ways of treating them. Resolution based proof systems and provers [4, 5] handle nominals by means of paramodulation, which can be computationally quite expensive. Also in the context of tableaux methods, where nominals and nominal equalities have been treated in different ways, many redundancies can arise. In fact, when handling a statement of the form $@_a b$, any known property of a can potentially be copied to b (and vice-versa).

This papers focuses on the treatment of nominal equalities in tableaux methods for the basic hybrid logic $HL(@)$, only succinctly indicating how to extend the approach to a more expressive language. Although minimal, the language of $HL(@)$ 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 also due to the fact that the satisfiability problem for a fragment of $HL(@, \downarrow)$, where no (positive) occurrence of $\Box$ dominates a binder, can be reduced to $HL(@)$ by means of a satisfiability preserving translation [20, 31].

It is worth pointing out, moreover, that a desirable property of any modular proof system for hybrid logics is its termination on decidable sublogics, hence also on formulae in $HL(@)$. Therefore, an efficient treatment of nominal equalities is of interest also for systems which can cope with more expressive languages, even if preserving termination in the case of a richer logic may require more costly mechanisms.

This paper describes a tableau system for $HL(@)$, named H, where tableau construction terminates with no need of loop-checks. Moreover, differently from other terminating calculi for $HL(@)$ [29, 31], but similarly to [12] and [26], termination in H does not rely on any specific (and sometimes complicated) tableau construction procedure. In other terms, the calculus enjoys a *strong termination property*. The calculus, originally proposed in [16], has been independently defined approximately at the same time as the calculi presented in [12]. The interest in reconsidering H here is due to the fact that experimental results obtained by means of a recent implementation have shown the efficiency of the algorithm it embodies for the treatment of nominal equalities. In fact, as will be illustrated later on, it reduces the redundancies that may be generated by the "expansion" of nominals actually naming the same world. Being H well suited to treat equalities, which are an essential feature of hybrid reasoning, its current implementation is a promising kernel theorem prover, that can be refined, optimised and extended to cope with other operators.

The system H described in this work is an *internalized* calculus, *i.e.*it deals only with object-language expressions. It is worth noticing that several calculi for hybrid logics make use of *prefixed formulae* of the form $\sigma : F$, where σ is a symbol of the meta-language and F a formula (one of the two calculi in [12], for instance). However, prefixes are useful either when they are complex expressions encoding the relation between states, or when there is no internal (object-language) mechanism to name worlds. The use of (simple) prefixes in the case of hybrid logics seems a useless burden, since nominals and the satisfaction operator can play the same role. In fact, prefixes may sometimes make things more complicated (the reader may wish to compare the two calculi presented in [12] and the respective completeness proofs). Beyond this fact, internalized hybrid calculi have the advantage that the addition of pure axioms automatically yields complete systems for the class of frames they define, although termination may in some cases become a non trivial issue (see [13]).

This work is organized as follows. After recalling the syntax and semantics of $HL(@)$ (Section 2), the tableau system is presented in Section 3 and its fundamental properties are proved. In Section 4 the proposed approach to nominal equalities is compared to others and the experimental results demonstrating its advantages are briefly presented. Section 5 sketches how the calculus can be extended so as to deal also with the global and converse modalities, though needing loop-checks to ensure termination. Section 6 concludes this work.

2 Syntax and semantics of $HL(@)$

In this section we present the syntax and semantics of the "uni-modal" version of $HL(@)$, its extension to the multimodal case being straightforward.

Let NOM and PROP be disjoint sets of propositional letters. The elements of NOM are called *nominals* and the elements of $\mathsf{NOM} \cup \mathsf{PROP}$ *atoms*. We shall use lowercase letters from the beginning of the alphabet, possibly with indexes, as metavariables for nominals, and p, q, r, possibly with indexes, for elements of PROP. The set of formulae in $HL(@)$ is defined by the following grammar:

$$F := \bot \mid p \mid a \mid \neg F \mid F \wedge F \mid F \vee F \mid \Box F \mid \Diamond F \mid @_a F$$

where $p \in \mathsf{PROP}$ and $a \in \mathsf{NOM}$. The notation $\top$ is a shorthand for $\neg\bot$.

An *interpretation* $\mathcal{M}$ is a quadruple $\langle W, R, N, I\rangle$ where W is a non-empty set (whose elements are the *states* of the interpretation), $R \subseteq W \times W$ (the *accessibility relation*), N is a function $\mathsf{NOM} \to W$ and I a function $W \to 2^{\mathsf{PROP}}$. 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. $\mathcal{M}, w \not\models \bot$.
2. $\mathcal{M}, w \models p$ if $p \in I(w)$, for $p \in \mathsf{PROP}$.
3. $\mathcal{M}, w \models a$ if $N(a) = w$, for $a \in \mathsf{NOM}$.
4. $\mathcal{M}, w \models \neg F$ if $\mathcal{M}, w \not\models F$.
5. $\mathcal{M}, w \models F \wedge G$ if $\mathcal{M}, w \models F$ and $\mathcal{M}, w \models G$.
6. $\mathcal{M}, w \models F \vee G$ if either $\mathcal{M}, w \models F$ or $\mathcal{M}, w \models G$.
7. $\mathcal{M}, w \models \Box F$ if for each w' such that wRw', $\mathcal{M}, w' \models F$.
8. $\mathcal{M}, w \models \Diamond F$ if there exists w' such that wRw' and $\mathcal{M}, w' \models F$.
9. $\mathcal{M}, w \models @_a F$ if $\mathcal{M}, N(a) \models F$.

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 a and formula F:

$$\neg @_a F \equiv @_a \neg F \qquad \neg \Diamond F \equiv \Box \neg F \qquad \neg \Box F \equiv \Diamond \neg F$$

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

3 The tableau system H

The tableau system we are now going to present, and which will be called H, "internalizes" prefixes, like in [9, 11, 13], (partially) in [31], one of the systems in [14] and one of the calculi in [12]. Tableau nodes are in fact labelled by sets of *satisfaction statements*, i.e. assertions of the form $@_a F$. In a formula of such a form, F is said to be *labelled by* a. If $@_a F \in S$, where S is a tableau node, we say that F is true at a in S. A formula of the form $@_a \Diamond b$, where b is a nominal, is a *relational formula*.

In the sequel, sets of formulae will be written as comma separated sequences of formulae. For the sake of simplicity, we assume that formulae are in negation normal form (nnf). Furthermore, we present here just the uni-modal version of the calculus; its extension to the multi-modal case (where different accessibility relations may co-exist) is straightforward.

The initial tableau for a set S of formulae is a node labelled by $S_a = \{@_a F \mid F \in S\}$, where a is a new nominal. S_a is called the *root set*. Nominals occurring

in S_a are called *root nominals*, and, if T is a tableau rooted at S_a, then the set of its root nominals is denoted by C_T:

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

The basic set of expansion rules is reported in Table 1, where S is a set of formulae. They are essentially the same rules of other internalized calculi for $HL(@)$ [12, 9, 11, 31] modulo a reformulation of the calculi from the "nodes as formulae" to the "nodes as sets" style. The boolean, modal and label rules are called *logical rules*. All the logical rules are *non-destructive*, *i.e.*they do not "consume" the expanded formula (they add formulae to the lower node, without modifying or deleting any formula in the upper node). Obviously, some rule must necessarily conserve its premise(s) to ensure completeness. In this work, for the sake of simplicity, the logical rules are formulated so that every expanded formula is kept in memory, with no further distinction. This choice allows for a simple restriction to avoid trivial non-terminating tableau constructions: it is established in fact that a formula is never added to a node where it already occurs. Moreover, it is assumed that the $\Diamond$-rule, which generates a new nominal, is never applied twice to the same premise on the same branch.

Obviously, from a practical point of view, not every formula has to be kept in memory, and tableau nodes can be structured in such a way that membership tests are in fact very light. Details can be found in [18].

The definitions of open/closed tableaux and complete branches are standard: a tableau node S is *closed* if it contains $\bot$ (see the Closure Rules of Table 1). Closed nodes are not expandable. 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 branches are closed, otherwise it is open.

In order to introduce the last rule of the system, treating nominal equalities, the following definitions are needed.

Definition 1. *If a nominal b is introduced in a branch Θ by application of the $\Diamond$-rule to a premise of the form $@_a\Diamond F$, then $a \prec_\Theta b$ (and we say that b is a child of a, and a is the father of b). The notation $children(a, \Theta)$ will be used to denote the set of nominals b such that $a \prec_\Theta b$.*

The relation $\prec_\Theta^+$ is the transitive closure of $\prec_\Theta$. If $a \prec_\Theta^+ b$ we say that b is a descendant of a and a an ancestor of b in the branch Θ.

The system H treats nominal equalities (formulae of the form $@_a b$, where b is a nominal) by means of a destructive rule, *i.e.*a rule with "side effects", the substitution rule (Sub). It is applicable only if $a \neq b$ and is formulated as follows:

$$\frac{@_a b, S}{S^\#[a \mapsto b]} \ (Sub)$$

where $S^\#[a \mapsto b]$ is obtained from S by:

1. deleting every formula containing a descendant of a;
2. replacing every occurrence of a with b.

When the substitution rule is applied, a is said to be *replaced in the branch* and the descendants of a are called *deleted in the branch.* Let's point out that root

Boolean Rules

$$\frac{@_a(F \wedge G), S}{@_a F, @_a G, @_a(F \wedge G), S}\ (\wedge)$$

$$\frac{@_a(F \vee G), S}{@_a F, @_a(F \vee G), S \qquad @_a G, @_a(F \vee G), S}\ (\vee)$$

Label Rule

$$\frac{@_a @_b F, S}{@_b F, @_a @_b F, S}\ (@)$$

Modal rules

$$\frac{@_a \Box F, @_a \Diamond b, S}{@_b F, @_a \Box F, @_a \Diamond b, S}\ (\Box) \qquad \frac{@_a \Diamond F, S}{@_a \Diamond b, @_b F, @_a \Diamond F, S}\ (\Diamond)$$

where b is a new nominal
(not applicable if F is a nominal)

Closure rules

$$\frac{@_a p, @_a \neg p, S}{\bot}\ (\bot_1) \qquad \frac{@_a \neg a, S}{\bot}\ (\bot_2)$$

Table 1: Basic expansion rules

nominals are never deleted (they cannot be children of any nominal), although obviously they may be replaced.

Note that, without nominal deletion in the substitution rule, *i.e.*by use of the simpler rule:

$$\frac{@_a b, S}{S[a \mapsto b]}\ (Sub^*)$$

tableau construction might not terminate. This is shown in Figure 1, where an infinite tableau is built by use of the simpler substitution rule Sub^* (and a particular rule application order, which delays substitutions: after the first application of the $\Diamond$-rule, the rules are applied in the order $\Box, \Diamond, \wedge, Sub^*$).[1] In the figure, the applied rules and the expanded formulae are shown on the right of inference lines, and the newly added formulae (if any) are displayed as the first ones in each node. The initial set is $\{@_a \Diamond(a \wedge \top), @_a \Box \Diamond(a \wedge \top)\}$; the subformula $\Diamond(a \wedge \top)$ cannot be replaced by $\Diamond a$ because of the restriction on the application of the $\Diamond$-rule.

What happens there is that each of the nominals $b_0, b_1, b_2, ...$, before being replaced by a, generates a child, that begins to live its own life. In particular, it is "adopted" by a, and therefore inherits its boxed formula $\Diamond(a \wedge \top)$.

[1]The rules used in this example, in particular Sub^*, are essentially (reformulations of) the rules of the system defined in [31].

$$
\begin{array}{c}
@_a\Diamond(a\wedge\top),\ @_a\Box\Diamond(a\wedge\top) \\
\hline
\quad (\Diamond : @_a\Diamond(a\wedge\top)) \\
@_a\Diamond b_0,\ @_{b_0}(a\wedge\top), \\
@_a\Diamond(a\wedge\top),\ @_a\Box\Diamond(a\wedge\top) \\
\hline
\quad (\Box : @_a\Diamond b_0, @_a\Box\Diamond(a\wedge\top)) \\
@_{b_0}\Diamond(a\wedge\top),\ @_a\Diamond b_0,\ @_{b_0}(a\wedge\top), \\
@_a\Diamond(a\wedge\top),\ @_a\Box\Diamond(a\wedge\top) \\
\hline
\quad (\Diamond : @_{b_0}\Diamond(a\wedge\top)) \\
@_{b_0}\Diamond b_1,\ @_{b_1}(a\wedge\top),\ @_{b_0}\Diamond(a\wedge\top), \\
@_a\Diamond b_0,\ @_{b_0}(a\wedge\top),\ @_a\Diamond(a\wedge\top),\ @_a\Box\Diamond(a\wedge\top) \\
\hline
\quad (\wedge : @_{b_0}(a\wedge\top)) \\
@_{b_0}a,\ @_{b_0}\top, \\
@_{b_0}\Diamond b_1,\ @_{b_1}(a\wedge\top), @_{b_0}\Diamond(a\wedge\top),\ @_a\Diamond b_0, \\
@_{b_0}(a\wedge\top),\ @_a\Diamond(a\wedge\top),\ @_a\Box\Diamond(a\wedge\top) \\
\hline
\quad (Sub^* : @_{b_0}a) \\
@_a\top,\ @_a\Diamond b_1,\ @_{b_1}(a\wedge\top), @_a\Diamond(a\wedge\top), \\
@_a\Diamond a,\ @_a(a\wedge\top),\ @_a\Box\Diamond(a\wedge\top) \\
\hline
\quad (\Box : @_a\Diamond b_1, @_a\Box\Diamond(a\wedge\top)) \\
@_{b_1}\Diamond(a\wedge\top), \\
@_a\top,\ @_a\Diamond b_1,\ @_{b_1}(a\wedge\top), @_a\Diamond(a\wedge\top), \\
@_a\Diamond a,\ @_a(a\wedge\top),\ @_a\Box\Diamond(a\wedge\top) \\
\hline
\quad (\Diamond : @_{b_1}\Diamond(a\wedge\top)) \\
@_{b_1}\Diamond b_2,\ @_{b_2}(a\wedge\top), \\
@_{b_1}\Diamond(a\wedge\top),\ @_a\top,\ @_a\Diamond b_1,\ @_{b_1}(a\wedge\top), @_a\Diamond(a\wedge\top), \\
@_a\Diamond a,\ @_a(a\wedge\top),\ @_a\Box\Diamond(a\wedge\top) \\
\hline
\quad (\wedge : @_{b_1}(a\wedge\top)) \\
@_{b_1}a,\ @_{b_1}\top,\ @_{b_1}\Diamond b_2,\ @_{b_2}(a\wedge\top), \\
@_{b_1}\Diamond(a\wedge\top),\ @_a\top, \\
@_a\Diamond b_1,\ @_{b_1}(a\wedge\top), \\
@_a\Diamond(a\wedge\top),\ @_a\Diamond a,\ @_a(a\wedge\top),\ @_a\Box\Diamond(a\wedge\top) \\
\hline
\quad (Sub^* : @_{b_1}a) \\
@_a\top,\ @_a\Diamond b_2,\ @_{b_2}(a\wedge\top), @_a\Diamond(a\wedge\top), \\
@_a\Diamond a,\ @_a(a\wedge\top),\ @_a\Box\Diamond(a\wedge\top) \\
\hline
\quad (\Box : @_a\Diamond b_2, @_a\Box\Diamond(a\wedge\top)) \\
\vdots
\end{array}
$$

Figure 1: An infinite tableau in H without nominal deletion

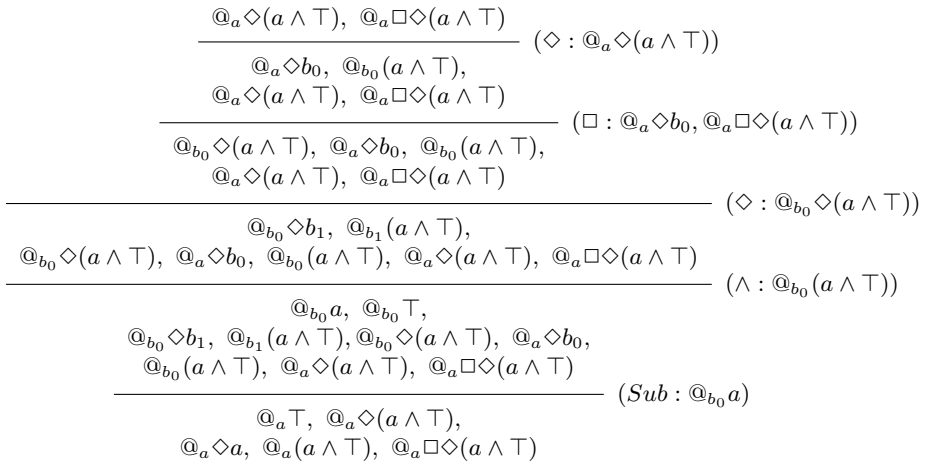

Figure 2: A finite tableau with nominal deletion

Figure 2 shows that the same initial set of formulae and the same rule application order produce a finite tableau when nominals are deleted by the substitution rule. The first (and unique) application of the substitution rule deletes b_1, since it is a child of b_0. None of the formulae in the leaf S can be expanded because $@_a\Diamond(a \wedge \top)$ has been already expanded in the branch; $@_a a$, $@_a\top$ and $a\Diamond a$ do not fire any rule (the latter formula is relational); applying the $\Box$-rule to $@_a\Box\Diamond(a \wedge \top)$ and $@_a\Diamond a$ would produce a formula already in the node; and the same would happen by expanding $@_a(a \wedge \top)$.

It is worth pointing out that a mechanism similar to substitution with nominal deletion (called *merging* and *pruning*) is used in a tableau algorithm for a rich description logic (with transitive and inverse roles, role hierarchies, qualified number restrictions and nominals) [24]. Termination is however guaranteed only when following a particular rule application strategy – beyond needing cycle detection, since such a logic enjoys neither the finite model property nor the tree model model one.

3.1 Termination

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

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 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_0,

$$S_0^* = \{F \mid F = G[b_1 \mapsto c_1, ..., b_n \mapsto c_n] \text{ for some subformula } G \text{ of some } A \in S_0, \text{ and } c_1, ..., c_n \in C_T\}$$

where $b_1, \cdots, b_n$ are nominals occurring in G. Note that, obviously, every nominal c occurring in a formula of S_0^* occurs in the initial set (*i.e.*$c \in C_T$). Moreover,

S_0^* 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 rooted at S_0, and $@_a F$ is a formula occurring in some node of T, then either $@_a F$ is relational or $F \in S_0^*$.*

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

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

is finite.

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

1. If $@_a b$ occurs in a node of T, then $b \in C_T$. Therefore, in the applications of the substitution rule, nominals are always replaced by root nominals.
2. A nominal $b \notin C_T$ may occur in a tableau node only in relational formulae, *i.e.*in the form $@_a \Diamond b$, or as the label of a formula $@_b F$ (where $F \in S_0^*$ does not contain b).

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 Θ, a nominal a cannot generate more children than the number of (non relational) formulae of the form $@_a \Diamond F$ that occur in Θ, and they are finite (Lemma 1). As a consequence:

Corollary 1. *For every branch Θ and nominal a, $children(a, \Theta)$ is finite.*

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

Lemma 2. *If c and d are nominals, $d \notin C_T$, and $@_c \Diamond d$ occurs in some node of a tableau branch Θ, then d is a child of c in Θ.*

Proof. The result is proved by induction on the number of nodes between the root of Θ and the node where $@_c \Diamond d$ 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 substitution rule is applied:

$$\frac{@_a b, S}{S^\#[a \mapsto b]} \ (Sub)$$

Let $@_c \Diamond d = @_{c'} \Diamond d'[a \mapsto b]$ be a formula occurring in $S^\#[a \mapsto b]$ and not in S. Hence, either $c' = a$ or $d' = a$ (or both). If $d' = a$, then $@_c \Diamond d = @_{c'} \Diamond a[a \mapsto b] = @_c \Diamond b$; since $@_a b$ occurs in a tableau node, by Lemma 1, $b \in C_T$ and there is nothing to prove.

So we are left with the case where $c' = a$ and $d' \neq a$, *i.e.*$@_c \Diamond d = @_a \Diamond d[a \mapsto b] = @_b \Diamond d$, with $d \notin C_T$. By the induction hypothesis, since $@_a \Diamond d \in S$, d is a child of a in Θ, thus it is deleted by the substitution rule and $@_c \Diamond d$ cannot occur in $S^\#[a \mapsto b]$. □

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 $@_b F$ (for $b \notin C_T$) appears in a tableau because it is obtained by application of the $\Box$-rule from some $@_a \Box F$ and $@_a \Diamond b$ where a is not the father of b.

Note that Lemma 2 would hold also if only the children of a replaced nominal, rather than its complete descent, were deleted, and this would suffice to make the whole termination argument work. However, as we will argue in Section 4, deleting all the descendants of a replaced nominal has a positive practical impact.

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

$$Deg(a, \Theta) = max\{degree(F) \mid @_a F \text{ occurs in } \Theta\}$$

It is worth pointing out that $Deg(a, \Theta) \in \mathbb{N}$, because the set of formulae labelled by a in Θ is finite, by Lemma 1. From Lemma 2, it is easily proved that, for any tableau branch Θ and any chain of nominals $a_0 \prec_\Theta a_1 \prec_\Theta a_2 \prec_\Theta ...$, the maximal modal degree of a_i is strictly decreasing. This is what is stated by next Lemma, whose proof is again an induction on tableaux.

Lemma 3. *Let T be a tableau and Θ a branch in T. If $a \prec_\Theta b$, then $Deg(b, \Theta) < Deg(a, \Theta)$.*

From this property it follows that:

Corollary 2. *In any tableau branch Θ, every chain of nominals $a_0 \prec_\Theta a_1 \prec_\Theta a_2 \prec_\Theta ...$ 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 Θ is finite. In particular, the relation $\prec_\Theta$ arranges the set of nominals occurring in Θ in a finite set of finite trees.*

Therefore, in any tableau the number of relational formulae labelled by the same nominal a, *i.e.*formulae of the form $@_a \Diamond b$, is finite. From this fact and Lemma 1, it follows that:

Corollary 4. *For every tableau branch Θ and nominal a, the set*

$$\{F \mid @_a F \text{ occurs in } \Theta\}$$

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 Θ, then (by the condition stating that a formula is never added to a node where it already occurs and the fact that no rule, but for substitution, consumes or modifies

its premise) there is at least a formula $@_a F$ that is either deleted or modified by application of the substitution rule, and then reintroduced in Θ. If $@_a F$ is deleted, it is because a is deleted and the same nominal cannot be used again in the branch. If it is modified, *i.e.*it becomes $(@_a F)[b \mapsto c]$ for some b and c, then every occurrence of b is replaced in the same node, therefore, again, a formula containing b cannot be reintroduced by any expansion rule. Therefore, no formula can be either 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:

$$\frac{@_a \downarrow x.A, S}{@_a A[x \mapsto a], @_a \downarrow x.A, S}$$

(applicable only if $A[x \mapsto a] \notin S$). Now, obviously, Lemma 1 fails. This corresponds to the fact that the binder "enables us to create a name for the here-and-now" [2]. 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.

It is worth pointing out that our termination proof fails also if the following rule for the universal modality is added to the system:

$$\frac{@_a AF, S}{@_c F, @_a AF, S}$$
where c occurs in $\{@_a AF\} \cup S$

In fact, in this case, Lemma 3 fails.

3.2 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 substitution 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 a, b such that $N(a) = N(b)$, $\mathcal{M}, w \models F$ if and only if $\mathcal{M}, w \models F[a \mapsto b]$.

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

Lemma 4. *If Θ 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 Θ 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 the fact that Θ is finite. So we first consider the set labelling the last node of Θ, that is shown to be downward saturated and, consequently, satisfiable. Then we show that satisfiability propagates upward to the root node, provided that nominals that are deleted in Θ, 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 Θ (and Θ contains F) to mean that F occurs in some node of Θ.

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

1. S_k does not contain any formula of the form $@_a\neg a$, and it does not contain two formulae of the form $@_a p$ and $@_a\neg p$ for some atom p.
2. If $@_a(F \wedge G) \in S_k$, then $@_a F, @_a G \in S_k$.
3. If $@_a(F \vee G) \in S_k$, then either $@_a F \in S_k$ or $@_a G \in S_k$.
4. If $@_a\Diamond F \in S_k$ (for F that is not a nominal), then there is a nominal b with $@_a\Diamond b, @_b F \in S_k$.
5. If $@_a\Diamond b, @_a\Box F \in S_k$, then $@_b F \in S_k$.
6. If $@_a@_b F \in S_k$, then $@_b F \in S_k$.
7. If $@_a b \in S_k$ then $a = b$, otherwise the substitution rule would be applicable to expand Θ.

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

$W = \{a \mid a \text{ occurs in } S_k\}$;
$R = \{(a, b) \mid @_a\Diamond b \in S_k\}$;
for every nominal a occurring in S_k, $N'(a) = a$;
$I(a) = \{p \mid @_a p \in S_k\}$.

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

Next, we define an equivalence relation on nominals (with respect to the branch Θ) as follows: $a \sim b$ if $@_a b$ occurs in Θ. 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 Θ. Let w^* be any fixed element of W. Then N is the extension of N' such that for all nominals c occurring in Θ:

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

It is clear that if $@_a b$ occurs in Θ, then $N(a) = N(b)$.

If $\mathcal{M} = \langle W, R, N, I\rangle$, obviously, it still holds that for every $@_a F \in S_k$, $\mathcal{M}$ satisfies F at $N(a)$.

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

($\bullet$) if $\mathcal{M}$ is a Θ-model of S_{i+1}, then $\mathcal{M}$ is a Θ-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 substitution rule, where:

$$\begin{aligned} S_i = &\quad @_a b, S' \\ S_{i+1} = &\quad S'^{\#}[a \mapsto b] \end{aligned}$$

By the induction hypothesis, for every formula $@_c F \in S_{i+1} = S'^{\#}[a \mapsto b]$ containing only nominals that are never deleted in the branch, $\mathcal{M}, N(c) \models F$. Note that all the descendants of a are deleted in Θ. Since $N(a) = N(b)$, by definition, $\mathcal{M}, N(a) \models b$.

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

Completeness follows directly from Lemma 4.

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

4 Comparison with other approaches

Some early attempts to obtain terminating calculi for decidable hybrid logics are [29, 31]. In both papers, however, the termination arguments rely on specific (and sometimes complicated) tableau construction procedures. As already remarked in Section 3, our approach to nominal equalities is closed to [31]. In fact, both calculi use forms of substitution, but the substitution rule in [31] does not delete formulae (*i.e.*it corresponds to the rule called Sub^* in Section 3), and the infinite tableau of Figure 1 is a counter-example to the strong termination property when such a rule is used.

The tableau calculus proposed for first-order $HL(@, \forall, \exists)$ in [11] (an extension of [9], which treats only the propositional case) has quite "natural" rules for treating nominal equalities (reflexivity, a rule which "passes" formulae from a nominal to an equal one and a rule that makes worlds, equal to a state seen by a given one, also visible by it). However, such a treatment of equalities, although very intuitive and natural, leads to a non terminating calculus even for the propositional (decidable) fragment $HL(@)$. As a matter of fact, devising a strongly terminating (and complete) calculus for propositional $HL(@)$ is not straightforward.

Later on, [14] presents a calculus deciding satisfiability for $HL(@, A, E)$, *i.e.*hybrid logic with the global modalities, by use of a loop-checking mechanism during tableau construction. While loop-checking is necessary in order to treat

the global and converse modalities, it can be avoided when the logic is restricted to $HL(@)$. And in fact, [12] presents (propositional) calculi in two different styles (prefixed and "internalized") which – like H – enjoy the strong termination property in the case of $HL(@)$, with no need for loop-checks. Note that the prefixed calculus in [12] extends also, in a modular way, to $HL(@, A, E, \Box^-, \Diamond^-)$, *i.e.*hybrid logic with the global and converse modalities; the internalized one, as originally proposed, only deals with $HL(@)$, but in [13] it is extended to the global and converse modalities.[2]

As a matter of fact, the internalized calculus proposed in [12] is very similar to H, the two systems differing only in the way how nominal equalities are treated. Still another approach to equality, highly declarative, is represented by [27].

The central point of this section is the comparison of the different treatments of equalities in H and the internalized calculus in [12], that will be called P. In [18] a still more thorough comparison between P and H, taking into account also implementation issues, can be found.

Beyond the fact that the calculus P is formulated in the "nodes as formulae" style (since rules never modify existing formulae) and other minor differences, the essential difference between H and P consists in the expansion rules for equalities. In P, such formulae are expanded by means of the two premises rule Id:

$$\frac{@_a F \quad @_a b}{@_b F}\ (Id)$$

The rule is applicable only if $@_a F$ is not an *accessibility* formula, *i.e.*a relational formula derived by an application of the $\Diamond$-rule.

The two rules for the treatment of equalities are apparently very different, Id being declaratively more elegant and simple than Sub. However, they bear strong similarities, that can also be recognized by inspection of the respective termination and completeness proofs for H and P. Such proofs rely in fact on the same key properties of Sub and Id.

First of all, both rules are 'directional": $@_a b$ is not the same as $@_b a$. And in both Id and Sub, the nominal b occurring in the premise $@_a b$ is a root nominal, therefore only root nominals can "inherit" formulae from other (equal) ones, in both calculi.

Secondly, forbidding the application of the Id rule to accessibility formulae (which is essential to ensure termination) means that the nominal b inherits all formulae true at a in the branch, except for those representing outcoming links to its children. The substitution rule directly replaces (instead of copying) a with b, but, again, its children are not "adopted" by b. They are deleted, together with all their descendants. So, in simple words we can say that the main difference between the two systems is that P is more tolerant than H: even when the descendants of a nominal are of no use any longer (and in fact P's completeness proof does not exploit them), they are left alive, since they are not harmful, either. H, on the contrary, is radical and bloody: when a nominal becomes useless, it is killed with all its descent.

Nominal deletion in H avoids the employment of resources to expand formulae labelled by "useless" nominals. It is worth pointing out that, in P, it can

[2]Also the systems proposed in [29] and [14] use the explicit formulation style of modal rules, by use of prefixes.

be proved that whenever $@_a b$, $@_b c$, $@_d a$ occur in a branch for some nominals a, b, c, d (*i.e.*a, b and c are *right nominals*), then also $@_b a, @_c b, @_a c$ and $@_c a$ occur in the branch, so that the Id rule copies formulae from any of the three nominals a, b, c to all the others.

Thus, abstract considerations lead to the hypothesis that H, though theoretically more complex than P, has a practical advantage, allowing a more efficient management of equalities. In order to check such an hypothesis, we have implemented both calculi: the system Pilate ("What crime has he committed?") implements P, while Herod is the implementation of the slaughter of the innocents represented by H. The two systems are implemented in Objective Caml [28] and are available at Herod web page[3]. Their implementations are described in [18], where also a first evaluation of the performances of Pilate and Herod is is carried out, on the base of a set of formulae generated by a naive random generator.

Here, new experimental results are briefly described, obtained by running the two provers on a set of 1600 sets of formulae, randomly generated by hGen [6], and approximately equally partitioned into satisfiable and unsatisfiable. The sets of formulae used for the benchmarks and the parameters used for their generation can be found at Herod web page. The experiments were run on an Intel Pentium 4 3GHz, with 3Gb RAM, running under Linux, and the provers were given one minute timeout.

The new experiments confirm what already described in [18]: considering the 1274 tests that both Pilate and Herod solved in the allowed time, Pilate is in the average more than 50 times slower than Herod, and the median run time of Pilate is more than 5 times Herod's one. Moreover, Pilate runs out of time almost 50% more often than Herod.

Maybe more interesting is the comparison between the two systems on a set of hand-written formulae (already reported in [18]), which involve many $\Diamond$'s and equalities, where the differences in treating equalities should be pushed to the limit. The formulae we have used have the form:

$$@_{a_1} \Diamond^n (@_{a_1} a_2 \wedge ... \wedge @_{a_n} a_{n+1} \wedge \Diamond F)$$

where $\Diamond^n$ is a sequence of n $\Diamond$'s, dominating n nominal equalities, and F is a (non trivially) unsatisfiable formula. The size of such formulae is taken to be n. Pilate can only solve problems up to size 100 in the allowed time of one minute, and its execution time increases exponentially. On the contrary, Herod can solve problems up to the maximal tested size (600), and its running times seem to increase linearly.

The empirical results shown above reflect what one could have expected, *i.e.*that Pilate consumes, in general, more resources than Herod. It is important to point out, moreover, that the different performances are effectively due to the different treatment of equalities. This fact is witnessed by testing the two systems on modal formulae (with no nominals): run on a set of 400 modal formulae randomly generated by hGen, the two provers had the same cases of timeouts and the same average and median execution times. The same results, showing that that all the difference between the systems is due to their treatment of @, are obtained when running the two provers on the hand-tailored collection

[3]Herod web page: `http://cialdea.dia.uniroma3.it/herod/`

of modal formulae proposed in [7], where in fact Pilate and Herod show the same behaviour.

Although Herod is only at its first stage of development, it has been compared with other more mature provers for hybrid logic, based on tableau calculi: HTab [22], which implements the prefixed calculus presented in [12], and Spartacus [21], based on the calculi presented in [25] and previous works by the same authors. Both provers can be considered to be much more mature than Herod, since they implement many important optimization strategies, such as, for instance, backjumping. And in fact, the relative performances of the three provers, Herod, HTab and Spartacus, run on the set of hand-tailored collection of modal formulae presented in [7] (with no nominals), show that Herod is far beyond the other two provers.

In order to compare the provers on their treatment of equalities, they have been run on the same sets of formulae used for the comparison with Pilate. Although the number of Herod's failures is the highest one, its median run time is better than HTab's and Spartacus's. Moreover, the average execution times on the problems solved by both Herod and HTab are in favour of Herod. Finally, in the average, the execution times of Herod and Spartacus, on the problems solved by both provers, are comparable.

The efficiency of Herod's treatment of equalities is even more apparent when comparing the three systems on the set of hand-written formulae earlier defined, up to size 600. The interest of such tests relies on the fact that they are meant not so much to compare the provers in themselves, rather their different treatments of nominal equalities. Herod's run time grows up to 0.11 seconds, while HTab reaches 0.28 seconds and Spartacus 0.24. This result suggests that, although an important refinement and optimisation work still has to be done, Herod's approach to nominal equalities deserves some interest.

5 An extension of H

Like it has been done for P [13], H can be extended to deal with the global modalities A and E and the converse modalities $\Box^-$ and $\Diamond^-$, provided that loop-checks are performed so as to ensure termination. We recall that the semantics of the global and converse modalities is defined as follows:

- $\mathcal{M}, w \models AF$ if $\mathcal{M}, w' \models F$ for each $w' \in W$;
- $\mathcal{M}, w \models \Box^- F$ if $\mathcal{M}, w' \models F$ for each $w' \in W$ such that $w'Rw$.

The operator E is the dual of A and $\Diamond^-$ is the dual of $\Box^-$.

The rules for these additional operators are given in Table 2.

It is worth pointing out that there is no restriction on the applicability of the $\Diamond^-$-rule; in fact, it is necessary to expand formulae of the form $@_a\Diamond^- b$ in order to obtain possible premises for the $\Box$ and $\Box^-$-rules, of the form $@_b\Diamond a$.

The blocking mechanism is similar to that already used in tableaux for description logics [23], then adapted to hybrid logic by [12]. We borrow here the same terminology used in the latter. In the present context, however, since the substitution rule is destructive and tableaux are to be formulated in the "nodes as sets" style, the basic notions are somewhat different.

Definition 2. *Let T be a tableau, Θ a branch of T and S a node of Θ. Then:*

Converse rules

$$\frac{@_a \Box^- F, @_b \Diamond a, S}{@_b F, @_a \Box^- F, @_b \Diamond a, S}\ (\Box^-) \qquad \frac{@_a \Diamond^- F, S}{@_b \Diamond a, @_b F, @_a \Diamond^- F, S}\ (\Diamond^-)$$

where b is a new nominal

Global Rules

$$\frac{@_a AF, S}{@_c F, @_a AF, S}\ (A)$$

where c occurs in the premise

$$\frac{@_a EF, S}{@_b F, @_a EF, S}\ (E)$$

where b is a new nominal

Table 2: Rules for the converse and global modalities

1. *A formula occurring in a tableau T is called* native *(in T) iff it is in the language of the root set, i.e. it does not contain any non-root nominal.*
2. *If a is a nominal occurring in S then $Forms_S(a)$ contains all the native formulae labelled by a in S:*
$$Forms_S(a) = \{F \mid @_a F \text{ occurs in } S \text{ and } F \text{ is native in } T\}$$
3. *Two nominals a and b are said to be* twins *in S if $Forms_S(a) = Forms_S(b)$.*
4. *$\prec_\Theta^*$ denotes the* reflexive *and transitive closure of $\prec_\Theta$.*
5. *If a is a nominal occurring in S, a is* blocked *in S if there is a pair of distinct twins $b, c \in S$ such that $b, c \prec_\Theta^* a$.*

In other terms, a and b are twins in a tableau node S if they label exactly the same set of native formulae, and a nominal a is blocked if either a is a twin of one of its ancestors, or it is a descendant of two distinct twin nominals.

Termination is ensured by the following *blocking condition*:

> *The rules $\Diamond, \Diamond^-$ and E are only allowed to be applied to a formula $@_a F$ of a tableau node S if a is not blocked in S.*

Although the only significant difference with respect to the internalized calculus proposed in [13] is the substitution rule, the termination and completeness proofs of the extended calculus presented in this work conceal many subtleties. Details can be found in [17].

We conclude this section with some observations on the loop checking mechanism. First of all, blocking is *dynamic*, i.e. a nominal a may be blocked in a node S and then un-blocked below S. In fact, S might contain two distinct twins $b, c \prec_\Theta^* a$, which later on are distinguished by the addition of some new formula to one of them.

In the presence of the global and converse modalities, moreover, the weaker restriction where a nominal a is blocked if it has a twin ancestor would not be enough to ensure termination. In fact, let C be a generation chain in a tableau branch Θ:

$$C = a_0 \prec_\Theta a_1 \prec_\Theta a_2 ...$$

It can happen that, for *any* finite initial segment

$$a_0 \prec_\Theta a_1 \prec_\Theta a_2 \prec_\Theta ... \prec_\Theta a_k$$

of C, a_k has no twins, although there might be distinct twins among $a_0, ..., a_{k-1}$. In such cases, with the weaker restriction every a_k can generate a_{k+1}. This is exactly what happens, for instance, when a tableau for $@_{a_0} A \Diamond \Box^- p$ is built: for any $i > 0$, the two nominals a_i and a_{i+1} of the generation chain become twins only after that a_{i+1} generates its child, which passes p back to a_{i+1} by application of the $\Box^-$ rule.

6 Concluding remarks

This paper describes an internalized tableau system for hybrid logic with the satisfaction operator, named H, that can be used to decide validity/satisfiability of formulae in $HL(@)$, and proves its essential properties. Any tableau in H is finite, independently of any rule application strategy (in other terms, the calculus enjoys strong termination), and finiteness does not depend on loop-checks. Loop checking becomes however necessary when extending the system to treat the global and converse modalities, like briefly sketched above.

The focus of this work being the treatment of nominal equalities, the system is compared with a calculus introduced in [12], which treats formulae of the form $@_a b$ in a different way. Both theoretical considerations, as well as experimental results briefly described in Section 4, show that H embodies a treatment of equalities that avoids the employment of resources to expand formulae labelled by "useless" nominals, thereby significantly reducing redundancies introduced by equalities.

The experimental results obtained are encouraging, and it it seems worthwhile to go on and refine the implementation of the system, i.e. Herod, both by some routine work that still can be done, as well as by studying and experimenting more effective rule application strategies and the implementation of basic optimization techniques. Moreover, its extension to handle a more expressive language have to be implemented.

A subject of future work is the study of extensions able to to cope with the transitive closure of accessibility relations (similarly to propositional dynamic logic) and some restricted use of the binder (still allowing for terminating tableaux), Here, as far as we know, really new ideas and techniques are needed.

References

[1] N. Alechina, S. Demri, and M. de Rijke. A modal perspective on path constraints. *Journal of Logic and Computation*, 13(6):936–956, 2003.

[2] C. Areces, B. Blackburn, and M. Marx. Hybrid logics: characterization, interpolation and complexity. *The Journal of Symbolic Logic*, 66(3):977–1010, 2001.

[3] C. Areces, P. Blackburn, and M. Marx. A road-map on complexity for hybrid logics. In *Proceedings of the 13th International Workshop and 8th*

Annual Conference of the EACSL on Computer Science Logic, number 1683 in LNCS, pages 307–321. Springer-Verlag, 1999.

[4] C. Areces, H. de Nivelle, and M. de Rijke. Resolution in modal, description and hybrid logic. *Journal of Logic and Computation*, 11(5 (Special Issue on Hybrid Logics)):717–736, 2001.

[5] C. Areces and J. Heguiabehere. Hylores 1.0: Direct resolution for hybrid logics. In A. Voronkov, editor, *Proceedings of CADE-18*, pages 156–160, Copenhagen, Denmark, July 2002.

[6] C. Areces and J. Heguiabehere. `hGen`: A random CNF formula generator for hybrid languages. In *Methods for Modalities 3 (M4M-3)*, Nancy, France, 2003.

[7] P. Balsiger, A. Heuerding, and S. Schwendimann. A benchmark method for the propositional modal logics K, KT, S4. *Journal of Automated Reasoning*, 24(3):297–317, 2000.

[8] N. Bidoit, S. Cerrito, and V. Thion. A first step toward modelling semistructured data in hybrid multi-modal logic. *Journal of Applied Non-classical Logics*, 14(4):447–475, 2004.

[9] P. Blackburn. Internalizing labelled deduction. *Journal of Logic and Computation*, 10(1):137–168, 2000.

[10] P. Blackburn, M. de Rijke, and Y. Venema. *Modal Logic*. Cambridge University Press, 2001.

[11] P. Blackburn and M. Marx. Tableaux for quantified hybrid logic. In U. Egly and C. Fermüller, editors, *Automated Reasoning with Analytic Tableaux and Related Methods (TABLEAUX 2002)*, number 2381 in LNAI, pages 38–52. Springer Verlag, 2002.

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

[13] T. Bolander and P. Blackburn. Terminating tableau calculi for hybrid logics extending K. *Electronic Notes in Theoretical Computer Science*, 231 (Proceedings of the 5th Workshop on Methods for Modalities, 2007):21–39, 2009.

[14] T. Bolander and T. Braüner. Tableau-based decision procedures for hybrid logic. *Journal of Logic and Computation*, 16(6):737–763, 2006.

[15] T. Braüner. Natural deduction for hybrid logics. *Journal of Logic and Computation*, 14(3):329–353, 2004.

[16] S. Cerrito and M. Cialdea Mayer. Terminating tableaux for HL(@) without loop-checking. Technical Report IBISC-RR-2007-07, Ibisc Lab., Université d'Evry Val d'Essonne, 2007. (`http://www.ibisc.univ-evry.fr/Vie/TR/2007/IBISC-RR2007-07.pdf`).

[17] S. Cerrito and M. Cialdea Mayer. Tableaux with substitution for hybrid logic with the global and converse modalities. Technical Report RT-DIA-155-2009, Dipartimento di Informatica e Automazione, Università di Roma Tre, 2009. (`http://web.dia.uniroma3.it/ricerca/rapporti/rt/2009-155.pdf`).

[18] M. Cialdea Mayer, S. Cerrito, E. Benassi, F. Giammarinaro, and C. Varani. Two tableau provers for basic hybrid logic. Technical Report RT-DIA-145-2009, Dipartimento di Informatica e Automazione, Università di Roma Tre, 2009. (`http://web.dia.uniroma3.it/ricerca/rapporti/rt/2009-145.pdf`).

[19] M. Franceschet and M. de Rijke. Model checking for hybrid logics. In *Methods for Modalities 3 (M4M-3)*, pages 109–123, Nancy, France, 2003.

[20] M. Franceschet and B. ten Cate. On the complexity of hybrid logics with binders. In L. Ong, editor, *Proceedings of Computer Science Logic 2005*, volume 3634 of *Lecture Notes in Computer Science*, pages 339–354. Springer Verlag, 2005.

[21] D. Götzmann, M. Kaminski, and G. Smolka. Spartacus: A tableau prover for hybrid logic. Technical report, Saarland University, 2009.

[22] G. Hoffmann and C. Areces. HTab: A terminating tableaux system for hybrid logic. *Electronic Notes in Theoretical Computer Science*, 231 (Proceedings of the 5th Workshop on Methods for Modalities, 2007):3–19, 2009.

[23] I. Horrocks and U. Sattler. A description logic with transitive and inverse roles and role hierarchies. *Journal of Logic and Computation*, 9(3):385–410, 1999.

[24] I. Horrocks and U. Sattler. A tableau decision procedure for $\mathcal{SHOIQ}$. *Journal of Automated Reasoning*, 39(3):249–276, 2007.

[25] M. Kaminski and G. Smolka. Terminating tableaux for hybrid logic with the difference modality and converse. In A. Armando, P. Baumgartner, and G. Dowek, editors, *Proc. of IJCAR 2008*, volume 5195 of *LNAI*, pages 210–225. Springer, 2008.

[26] M. Kaminski and G. Smolka. Hybrid tableaux for the difference modality. *Electronic Notes in Theoretical Computer Science*, 231 (Proceedings of the 5th Workshop on Methods for Modalities, 2007):241–257, 2009.

[27] 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.

[28] X. Leroy. The Objective Caml system, release 3.11. Documentation and user's manual. (`http://caml.inria.fr/`), 2008.

[29] M. Tzakova. Tableau calculi for hybrid logics. In N. Murray, editor, *Automated Reasoning with Analytic Tableaux and Related Methods (TABLEAUX 1999)*, volume 1617 of *LNAI*, pages 278–292. Springer Verlag, 1999.

[30] J. van Eijck. HyLoTab – Tableau-based theorem proving for hybrid logics. Manuscript, CWI, Amsterdam (`http://www.cwi.nl/~jve/hylotab/Hylotab.pdf`), 2002.

[31] J. van Eijck. Constraint tableaux for hybrid logics. Manuscript, CWI, Amsterdam, 2002.