Nominal Substitution at Work with the Global and Converse Modalities

Serenella Cerrito

IBISC, Université d'Evry Val d'Essonne,
Tour Evry 2, 523 Place des Terrasses de l'Agora,
91000 Evry Cedex, France

Marta Cialdea Mayer

Dipartimento di Informatica e Automazione
Università di Roma Tre
Via della Vasca Navale 79, 00146 Roma, Italy

Abstract

This paper represents a continuation of a previous work, where a practical approach to the treatment of nominal equalities in tableaux for basic Hybrid Logic $HL(@)$ was proposed. Its peculiarity is a substitution rule accompanied by nominal deletion. The main advantage of such a rule, compared with other approaches, is its efficiency, that has been experimentally verified for the $HL(@)$ fragment.
The integration of substitution and nominal deletion with more expressive languages is not a trivial task. In this work the previously proposed tableaux calculus for $HL(@)$ is extended to hybrid logic with the global and converse modalities, taking into account also practical considerations. Though termination, in this case, relies on loop checks, the computational advantages of the substitution rule persist in this richer framework.

Keywords: Tableaux, Modal Logic, Hybrid Logic, Automated Deduction

1 Introduction

Kripke structures, providing the semantics of modal logics, can be seen as labelled graphs, or else, transition systems. As such, they are widely used in computer science for modelisation purposes. *Nominal equalities*, which are assertions saying that a and b are different names for the same vertex (state), are typical of hybrid logic. In fact, the hybrid syntax extends the modal one by allowing states to be named by means of *nominals*, i.e. atomic formulae which hold at exactly one state, and it is possible to express that a formula F - possibly a nominal b itself - holds at a state named a by

use of the *satisfaction* operator ($@_a F$). In particular, an equality $@_a b$ states that the nominals a and b are synonymous.

The treatment of nominal equalities in proof systems and provers may raise many redundancies, because when a formula F holds in the world named by synonymous nominals $a_1, ..., a_n$, it can potentially be treated n times. A previous work [6], further revised and extended in [5], presented a tableaux calculus for $HL(@)$ (basic hybrid logic), called H, whose characterizing feature is the treatment of nominal equalities by means of a special substitution rule, that expands an equality $@_a b$ by replacing a with b while deleting the chain of nominals generated by a, thus allowing for a reduction of the number of redundancies possibly induced by nominal equalities. Though embedded in a different context, such a rule is essentially the same as the "merge and prune" method proposed by [14] for the description logic $\mathcal{SHOIQ}$.

The computational advantages of such an approach with respect to expanding $@_a b$ by copying formulae from a to b, like in [2], have been experimentally verified [9,8]. The gain in efficiency is due to the *nominal deletion* mechanism embedded in the substitution rule.

Like other calculi for basic hybrid logic, H enjoys strong termination (every tableau in H is finite, independently of the rule application order), and termination does not need loop checks. Moreover, the system does not use any extra-logical notation (like prefixes), i.e. it is an *internalized* calculus. Tableau calculi for modal logics can in fact be given either in the explicit style, by means of *prefixes*, or in the implicit style (internalized calculi). In the first case, tableau nodes are labelled by *prefixed formulae* of the form $\sigma : F$, where σ is a symbol of the meta-language and F a formula, in the second one by language-level formulae. 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.

In the case of hybrid logic, both styles have been used. A prefixed calculus is presented for instance in [2], which constitutes the theoretical background of the prover HTab [12]. However, 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 both meta-proofs and implementations more complicated. 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 [3].

In this work we show that the approach to nominal equalities proposed in [6,5] can be safely extended to $HL(@, \Diamond^-, \mathsf{E})$, i.e. the language containing, beyond the satisfaction operator, the global modality E (and its dual A) and the converse modalities $\Diamond^-$ and $\Box^-$, preserving the computational advantages of substitution (with nominal deletion). The overall main feature of the proposed calculus is its practical approach, aiming in fact at being as close as possible a theoretical basis of an implemented system. For this reason, a particular attention is paid to efficiency aspects.

In the extended calculus, termination is achieved by means of a loop checking mechanism, partially like in [2,3], i.e. using ancestor equality blocking. The paper also shows that such a mechanism coupled with substitution without nominal deletion does not

ensure termination.

An additional gain in efficiency is obtained by a different treatment of the existential modality: no loop checks are performed when expanding a formula of the form $@_a \mathsf{E} F$, still preserving termination.

The presence of substitution, and its interplay with the nominal generating rules, raise specific technical subtleties in the termination and completeness proofs, which are given here with a fair amount of details.

1.1 Preliminaries

We conclude this section by briefly recalling the syntax and semantics of (multi-modal) hybrid logic with the global and converse modalities, $HL(@, \Diamond^-, \mathsf{E})$.

Given two disjoint sets of atoms, NOM (nominals) and PROP (ordinary atoms), and a set of relation labels REL, formulae are built up from atoms using the classical connectives, the satisfaction operator @, the unary modal operators $\Diamond_\tau$, $\Box_\tau$ and their converses $\Diamond_\tau^-$, $\Box_\tau^-$ (where $\tau \in \mathsf{REL}$), and the global modalities E and A. Formulae are defined by the following grammar, where $p \in \mathsf{PROP}$, $a \in \mathsf{NOM}$, $\tau \in \mathsf{REL}$:

$$F := p \mid a \mid \neg F \mid F \wedge F \mid F \vee F \mid @_a F \mid \Diamond_\tau F \mid \Box_\tau F \mid \Diamond_\tau^- F \mid \Box_\tau^- F \mid \mathsf{A}\, F \mid \mathsf{E}\, F$$

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

If $\mathcal{M} = \langle W, \{R_\tau \mid \tau \in \mathsf{REL}\}, N, I\rangle$ is an interpretation, $w \in W$ and F a formula, the relation $\mathcal{M}_w \models F$ is defined by:

(i) $\mathcal{M}_w \models p$ if $p \in I(w)$, for $p \in \mathsf{PROP}$.

(ii) $\mathcal{M}_w \models a$ if $N(a) = w$, for $a \in \mathsf{NOM}$.

(iii) $\mathcal{M}_w \models \neg F$ if $\mathcal{M}_w \not\models F$.

(iv) $\mathcal{M}_w \models F \wedge G$ if $\mathcal{M}_w \models F$ and $\mathcal{M}_w \models G$.

(v) $\mathcal{M}_w \models F \vee G$ if either $\mathcal{M}_w \models F$ or $\mathcal{M}_w \models G$.

(vi) $\mathcal{M}_w \models @_a F$ if $\mathcal{M}_{N(a)} \models F$.

(vii) $\mathcal{M}_w \models \Box_\tau F$ if for each w' such that $wR_\tau w'$, $\mathcal{M}_{w'} \models F$.

(viii) $\mathcal{M}_w \models \Diamond_\tau F$ if there exists w' such that $wR_\tau w'$ and $\mathcal{M}_{w'} \models F$.

(ix) $\mathcal{M}_w \models \Box_\tau^- F$ if for each w' such that $w'R_\tau w$, $\mathcal{M}_{w'} \models F$.

(x) $\mathcal{M}_w \models \Diamond_\tau^- F$ if there exists w' such that $w'R_\tau w$ and $\mathcal{M}_{w'} \models F$.

(xi) $\mathcal{M}_w \models \mathsf{A}F$ if for each $w' \in W$, $\mathcal{M}_{w'} \models F$.

(xii) $\mathcal{M}_w \models \mathsf{E}F$ if there exists $w' \in W$ such that $\mathcal{M}_{w'} \models F$.

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 ($F \equiv G$) when, 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:

$$\neg @_a F \equiv @_a \neg F \qquad \neg \Diamond_\tau F \equiv \Box_\tau \neg F \qquad \neg \Box_\tau F \equiv \Diamond_\tau \neg F$$
$$\neg \Diamond_\tau^- F \equiv \Box_\tau^- \neg F \qquad \neg \Box_\tau^- F \equiv \Diamond_\tau^- \neg F \qquad \neg \mathsf{A} F \equiv \mathsf{E} \neg F \qquad \neg \mathsf{E} F \equiv \mathsf{A} \neg F$$

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

2 The tableau calculus H^+

In the calculus H and its extension, that will be named H^+, tableau nodes are labelled by sets of *satisfaction formulae*, i.e. assertions of the form $@_a F$ written as comma separated sequences of formulae. A formula of the form $@_a F$ will be called *labelled by* a. A formula of the form $@_a \Diamond_\tau b$, where b is a nominal, is a *relational formula*.

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. Without loss of generality, formulae are assumed to be in negation normal form. The set S_a is called the *initial set*. Nominals occurring in S_a are called *native nominals* (in the tableau).

Table 1 contains the *logical* rules, i.e. all rules but substitution. A tableau node is closed if it contains either $@_a p$ and $@_a \neg p$ for some nominal a and atom p, or $@_a \neg a$ for some nominal a (otherwise it is open). A tableau branch is open if all its nodes are open, otherwise it is closed. A tableau is closed if all its branches are closed, otherwise it is open.

Note that all the rules of Table 1 are conservative, i.e. they do not "consume" their premises. The $\Diamond_\tau$, $\Diamond_\tau^-$ and E rules are called *nominal generating rules*, and formulae of the form $\Diamond_\tau F$, $\Diamond_\tau^- F$ and $\mathsf{E} F$ *nominal generating formulae*. It is worth pointing out that, contrarily to the $\Diamond_\tau$-rule, the table gives no restriction on the applicability of the $\Diamond_\tau^-$-rule; in fact, it is necessary to expand formulae of the form $@_a \Diamond_\tau^- c$ where $c \in \mathsf{NOM}$, in order to obtain possible premises for the $\Box_\tau$ and $\Box_\tau^-$ rules, of the form $@_c \Diamond_\tau a$.

A formula occurring in a tableau T is called *native* (in T) if and only if it is in the language of the initial set, i.e. it does not contain any non-native nominal. A formula occurring in a tableau node is an *accessibility formula* if it is a relational formula introduced by application of the $\Diamond_\tau$ or $\Diamond_\tau^-$ rule. Accessibility formulae are obviously not native. It is worth pointing out that only the direct modalities $\Diamond_\tau$ occur in accessibility formulae, and not the converse ones.

In order to define the last rule of the system, the substitution rule, the definition of father and children of a nominal, given in [6,5] for H, has to be extended.

Definition 2.1 Let Θ be a tableau branch. If either the $\Diamond_\tau$ or $\Diamond_\tau^-$ rule has been applied in Θ to a formula labelled by a, generating a new nominal b, then $a \prec_\Theta b$ (and we say that b is a child of a, and a is the father of b).

<table>
<tr><td colspan="2">Boolean Rules

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

$\dfrac{@_a(F \vee G), S}{@_a F, @_a(F \vee G), S \quad @_a G, @_a(F \vee G), S}\ (\vee)$</td><td>Label Rule

$\dfrac{@_a @_b F, S}{@_b F, @_a @_b F, S}\ (@)$</td></tr>
<tr><td colspan="3">Rules for the direct and converse modalities

$\dfrac{@_a \Box_\tau F, @_a \Diamond_\tau b, S}{@_b F, @_a \Box_\tau F, @_a \Diamond_\tau b, S}\ (\Box_\tau)$ $\qquad$ $\dfrac{@_a \Diamond_\tau F, S}{@_a \Diamond_\tau b, @_b F, @_a \Diamond_\tau F, S}\ (\Diamond_\tau)$
where b is a new nominal
(not applicable if F is a nominal)

$\dfrac{@_a \Box_\tau^- F, @_b \Diamond_\tau a, S}{@_b F, @_a \Box_\tau^- F, @_b \Diamond_\tau a, S}\ (\Box_\tau^-)$ $\qquad$ $\dfrac{@_a \Diamond_\tau^- F, S}{@_b \Diamond_\tau a, @_b F, @_a \Diamond_\tau^- F, S}\ (\Diamond_\tau^-)$
where b is a new nominal</td></tr>
<tr><td colspan="3">Rules for the global modalities

$\dfrac{@_a \mathsf{A} F, S}{@_c F, @_a \mathsf{A} F, S}\ (\mathsf{A})$ $\qquad$ $\dfrac{@_a \mathsf{E} F, S}{@_b F, @_a \mathsf{E} F, S}\ (\mathsf{E})$
where c occurs in the premise $\qquad$ where b is a new nominal</td></tr>
</table>

Table 1
Logical rules of the tableau system

The relation $\prec_\Theta^+$ is the transitive closure of $\prec_\Theta$ and $\prec_\Theta^*$ the reflexive and 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.

Nominals with no fathers are called *root nominals*.

Note that if a nominal b is newly introduced in a branch by expanding a formula or the form $@_a \mathsf{E} F$, then b is not a child of a.

The substitution rule, which is applicable only if $a \neq b$, is formulated as follows:

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

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

(i) replacing every occurrence of a with b;

(ii) deleting every formula containing a descendant of a.

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.*

It is worth pointing out that, differently from the cases of the label and boolean rules

which could also be formulated as "destructive", [1] the $\Diamond_\tau$ and $\Diamond_\tau^-$ rules must conserve their premises (though they are single-premise rules). In fact, when the descendants of a replaced nominal are deleted, these formulae have to be reused, with the new label, in order to keep completeness.

In order to ensure termination, first of all, trivial re-applications of rules must be ruled out. So, we establish that:

R1. rules that would not change the node (their expansion being already contained in the upper node) are not applicable.

Obviously, in practice, since substitution affects the whole node, such heavy membership tests can be avoided and replaced by a marking mechanism – which can also allow for the boolean and label rules being implemented destructively (see [9] or [8]).

Secondly, useless re-application of nominal generating rules must be avoided. To this aim, the expansion of existential formulae is restricted similarly to [16]. A memory of existential formulae is associated to each tableau node. When a formula of the form $@_a\mathsf{E}F$ is expanded, the "body" of the formula, $\mathsf{E}F$, is saved in the memory of the resulting node. When the substitution rule is applied, it affects also memorized formulae (which are not deleted), and the other rules keep the memory unchanged. The E rule is then subject to the following restriction:

R2. a formula of the form $@_a\mathsf{E}F$ is not expanded in a node S if $\mathsf{E}F$ belongs to the memory of S.

Moreover, when one of the rules $\Diamond_\tau$ or $\Diamond_\tau^-$ is applied, the expanded formula is marked as *inactive* in the lower node. When the substitution rule is fired by an equality $@_a b$, it affects (only) the markings of formulae labelled by a, as follows: every formula of the form $@_a\Diamond_\tau F$ or $@_a\Diamond_\tau^- F$ becomes active again, before replacing a with b and deleting its descendants. This is because, as already remarked, the children of a are deleted, and the formulae resulting from the substitution, $@_b\Diamond_\tau F'$ and $@_b\Diamond_\tau^- F'$ ($F' = F[a \mapsto b]$), need to be expanded again in order to generate the corresponding children of b.

It is necessary to specify how substitution interacts with the active/inactive marking mechanism, i.e. what happens when two differently marked formulae collapse by the effect of substitution. In such cases, the inactive marking takes precedence. In particular, when *Sub* is applied to a node containing an active formula F_1 and an inactive one F_2, then, if the application of the substitution to F_1 and F_2 produces the same formula G, G is marked as inactive in the conclusion of the rule. The same mechanism applies when the *Sub* rule replaces a with b in a node containing both $@_a\Diamond_\tau F_1$ (or $@_a\Diamond_\tau^- F_1$) and an inactive $@_b\Diamond_\tau F_2$ (or $@_b\Diamond_\tau^- F_2$), with $F_1[a \mapsto b] = F_2[a \mapsto b] = G$: the inactive marking takes precedence again in the conclusion of the rule. I.e. substitution does not produce two occurrences of $@_b\Diamond_\tau G$ (or $@_b\Diamond_\tau^- G$), an active occurrence and an inactive one, but a single inactive one.

The $\Diamond_\tau$ and $\Diamond_\tau^-$ rules are then restricted as follows:

[1] The destructive versions of the label and boolean rules would in fact be more faithful to a possible implementation. Though such a choice would affect only technical details, the completeness and termination proofs would however become more fastidious.

R3. the $\Diamond_\tau$ and $\Diamond_\tau^-$ rules are not applicable to inactive formulae.

Note that the simpler restriction that nominal generating formulae are never expanded more than once on a branch (R1) would be weaker than R2 and R3. In fact, in that case, when an already expanded formula of the form $@_a\Diamond_\tau F$, $@_a\Diamond_\tau^- F$ or $@_a\mathsf{E}F$ is changed by a substitution, resulting in a never expanded new formula, it would become expandable again (even if the substitution does not affect a). Moreover, restriction R2 cannot be replaced by marking an expanded $@_a\mathsf{E}F$ as inactive and establishing that $@_b\mathsf{E}F$ is not expanded whenever the node contains an inactive $@_a\mathsf{E}F$ (similarly to restriction R3); in fact an existential inactive formula could be deleted by the *Sub* rule, without its witness being deleted. Nor could restriction R2 be replaced by simply requiring that $@_a\mathsf{E}F$ is not expandable whenever the node contains some $@_bF$, for any nominal b (this would not be enough for the completeness proof given in Section 3.2).

Finally termination needs a blocking mechanism relying on loop checks, which exploits the notion of *twin nominals* (like in [3]):

Definition 2.2 Let T be a tableau, Θ a branch of T and S a node of Θ. Then:

- If a is a nominal occurring in S then

$$Forms_S(a) = \{F \mid @_aF \text{ occurs in } S \text{ and } F \text{ is native in } T\}$$

  i.e. $Forms_S(a)$ contains all the native formulae labelled by a in S.
- Two nominals a and b are said to be twins in S if $Forms_S(a) = Forms_S(b)$.
- A nominal a occurring in S is *directly blocked* by a nominal b in S if $b \prec^+_\Theta a$ and a and b are twins in S. It is *indirectly blocked* by b in S when $b \prec^+_\Theta a$ and b is directly blocked in S. Finally, a is *blocked* in S when it is either directly or indirectly blocked in S.

Note that a nominal a is non-blocked in S if and only if there is no pair of distinct twins b, c occurring in S such that $b, c \prec^*_\Theta a$.

Termination is ensured by the following restriction:

R4. *The rules $\Diamond_\tau$ and $\Diamond_\tau^-$ are not applicable to a formula labelled by a nominal a in a node S if a is blocked in S.*

A tableau node (respectively a tableau branch) is *complete* if no rule can be applied to expand it (possibly because of restrictions R1–R4). A tableau is complete when all its branches are complete.

The blocking mechanism is similar to the dynamic loop checking [2] introduced in [13] for description logics, then adapted to hybrid logic in [2]. However, existential formulae are treated differently, more in the style of [16]. A loop checking mechanism closer to that used in [2,3] – and adopted in the first proposal to extend H to the global and converse modalities, that is [7] – would treat E like the $\Diamond_\tau$ and $\Diamond_\tau^-$ operators: if a new

[2] If blocks on nominals are never undone then blocking is static. Otherwise it is called dynamic. In H^+ blocking is dynamic because the "twin" relation may be destroyed when new formulae are added to the branch.

nominal b is generated by expansion of an existential formula labelled by a, then $a \prec_\Theta b$, and it is established that a formula of the form $@_a\mathsf{E}F$ cannot be expanded when a is blocked. The main difference between the two methods is that, in H^+, any existential formula is expanded at most once, independently of its label, thus avoiding possible redundant computations.

We end up this section pointing out that nominal deletion not only allows one to avoid useless expansions, but is necessary in order to ensure termination, as shown by the infinite tableau in Figure 1, where nominals are replaced without deleting their descendants. For the sake or readability, when a rule without side effects is applied, the lower node shows only the newly added formulae. Note that, as an effect of substitution without nominal deletion, each nominal b_{i+1} becomes orphan of its father b_i that, otherwise, would have blocked it.

$$
\begin{array}{cl}
@_a\mathsf{A}\Diamond_\tau(p\wedge a) & \\
\hline
@_a\Diamond_\tau(p\wedge a) & (\mathsf{A}) \\
\hline
@_a\Diamond_\tau b_0, @_{b_0}(p\wedge a) & (\Diamond_\tau) \\
\hline
@_{b_0}p, @_{b_0}a & (\wedge) \\
\hline
@_{b_0}\Diamond_\tau(p\wedge a) & (\mathsf{A}) \\
\hline
@_{b_0}\Diamond_\tau b_1, @_{b_1}(p\wedge a) & (\Diamond_\tau) \\
\hline
@_{b_1}p, @_{b_1}a & (\wedge) \\
\hline
@_a\mathsf{A}\Diamond_\tau(p\wedge a), @_a\Diamond_\tau(p\wedge a), @_a\Diamond_\tau a, @_a(p\wedge a), @_a p, @_a a, @_a\Diamond_\tau b_1, @_{b_1}(p\wedge a), @_{b_1}p, @_{b_1}a & (b_0\mapsto a) \\
\hline
@_{b_1}\Diamond_\tau(p\wedge a) & (\mathsf{A}) \\
\hline
@_{b_1}\Diamond_\tau b_2, @_{b_2}(p\wedge a) & (\Diamond_\tau) \\
\hline
@_{b_2}p, @_{b_2}a & (\wedge) \\
\hline
@_a\mathsf{A}\Diamond_\tau(p\wedge a), @_a\Diamond_\tau(p\wedge a), @_a\Diamond_\tau a, @_a(p\wedge a), @_a p, @_a a, @_a\Diamond_\tau b_2, @_{b_2}(p\wedge a), @_{b_2}p, @_{b_2}a & (b_1\mapsto a) \\
\vdots &
\end{array}
$$

Fig. 1. Substitution without nominal deletion

As a final remark, one can observe that the tableau in Figure 1 would terminate (even without nominal deletion) if *subset blocking* were adopted, i.e. if a nominal were blocked whenever it labels a subset of the formulae labelled by some of its ancestors. In fact, $@_{b_1}\Diamond_\tau(p\wedge a)$ could not be expanded since b_1 labels, at that point, a subset of the formulae labelled by a. Subset blocking is sufficient for converse-free formulae, [3] but does not guarantee completeness for $HL(@, \Diamond^-, \mathsf{E})$.

[3] One of the tableau decision procedures presented in [4] for $HL(@, \mathsf{E})$ uses in fact substitution – without nominal deletion – and (anywhere) subset blocking.

3 Properties of H^+

3.1 Termination

In order to show that H^+ terminates with loop-checks (under any rule application strategy), one can use an argument similar to the one given in [2] for the prefixed calculus. However, there are obvious differences, and some more delicate points, due to the presence of the substitution rule and the different treatment of existential formulae.

In order to state the key property of the system, we need the following definition:

Definition 3.1 If T is a tableau rooted at S_0, then:

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

In other terms, S_0^* contains every formula that can be obtained from a subformula of some formula in the initial set, by replacing nominals with native nominals. Note that S_0^* is necessarily finite and closed with respect to subformulae.

The following result can easily been proved by induction on tableaux.

Lemma 3.2 (Quasi-subformula property) *Let Θ be a branch in a tableau rooted at S_0. Then for every nominal a, the set of native formulae labelled by a in Θ is a subset of the finite set*

$$S_0^* \cup \{\Diamond_\tau b \mid b \text{ is a native nominal and } \tau \text{ is in the language of } S_0\}$$

Moreover, any formula $@_a F$ occurring in Θ with F non-native is a relational formula (i.e. F has the form $\Diamond_\tau b$).

The following properties are direct consequences of Lemma 3.2. If T is a tableau rooted at S_0, then:

(i) If $@_a b$ occurs in a node of T, then b is a native nominal. Therefore, in the applications of the substitution rule, nominals are always replaced by native nominals.

(ii) A non-native nominal b may occur in tableau nodes only in relational formulae of the form $@_a \Diamond_\tau b$ or $@_b \Diamond_\tau a$, or as the label of a formula $@_b F$, where $F \in S_0^*$ is native.

(iii) In particular, b is native in any formula of the form $@_a \Diamond_\tau^- b$, since such formulae are not relational. Therefore, a given nominal a may label only a finite number of formulae of the form $\Diamond_\tau^- b$.

(iv) If there is only a finite number of nominals occurring in a tableau branch Θ, then the set of formulae occurring in Θ and labelled by any fixed nominal a is finite (if there is a finite number of nominals, a can label only a finite number of relational formulae).

Since a given nominal a may label only a finite number of formulae which can be expanded by means of the $\Diamond_\tau$ or $\Diamond_\tau^-$ rule (Lemma 3.2, and its consequence iii), the restrictions on the applicability of such rules allow us to prove:

Lemma 3.3 *If Θ is a tableau branch and a any nominal occurring in Θ, then $\{b \mid a \prec_\Theta b\}$ is finite.*

As a consequence, also the following result holds:

Lemma 3.4 *Let Θ be a tableau branch. If Θ is infinite then there is an infinite chain of nominals*

$$b_1 \prec_\Theta b_2 \prec_\Theta b_3 ... \ .$$

Proof. The presence of the substitution rule makes the argument a little more complicated than the corresponding one given in [2].

First of all we prove that if Θ is infinite, then there is an infinite number of nominals occurring in Θ. If there were only a finite number of nominals, in fact, each of them would label a finite number of formulae (by the consequence iv of Lemma 3.2). Now, since formulae are never added to nodes where they already occur, there should be at least one formula F occurring in a node S_i of Θ which disappears from the branch and then reappears in a node S_j below S_i. F can disappear only because some nominal occurring in it is either replaced or deleted. But when a nominal is replaced or deleted, it can never occur again in the branch below the application of *Sub* which replaces/deletes it.

The infinite number of nominals occurring in Θ can be arranged by $\prec_\Theta$ in a forest of trees rooted at root nominals, and there are finitely many such trees. In fact, only native nominals and nominals generated by the E rule have no fathers, and their number is finite (only a finite number of – necessarily native – formulae of the form $\mathsf{E}F$ may occur in the branch, and each of them is expanded at most once, independently of its label).

Moreover, each tree is finitely branching because any nominal can generate only a finite number of new ones, by Lemma 3.3. By König's Lemma, if one of such trees is infinite, it has an infinite branch, i.e. there is an infinite chain of nominals $b_1 \prec_\Theta b_2 \prec_\Theta b_3 ... \ .$ □

Theorem 3.5 (Termination) *Every tableau is finite.*

Proof. By Lemma 3.4, if an infinite branch Θ exists, then there is an infinite chain of nominals

$$b_1 \prec_\Theta b_2 \prec_\Theta b_3 ... \ .$$

By Lemma 3.2, if $@_a F$ occurs in Θ and F is native, then F is an element of the finite set $\Sigma = S_0^* \cup \{\Diamond_\tau b \mid b$ is a native nominal and τ is in the language of $S_0\}$, where S_0 is the initial set.

Let n be the cardinality of Σ and consider the initial sub-chain:

$$b_1 \prec_\Theta b_2 \prec_\Theta ... \prec_\Theta b_{2^n+1} \prec_\Theta b_{2^n+2}$$

Let Θ' be the initial segment of Θ up to, but not including, the nominal-generating inference ($\Diamond_\tau$ or $\Diamond_\tau^-$) producing b_{2^n+2}, and let S_k be the last node of Θ'.

Since b_{2^n+1} occurs in S_k, all its ancestors occur in S_k, too (if some of them had been either replaced or deleted above S_k, b_{2^n+1} would have been deleted by the same application of the substitution rule). Since b_{2^n+1} is the father of b_{2^n+2} in Θ, and it generates b_{2^n+2} by expanding S_k, then b_{2^n+1} is not blocked in S_k, i.e. S_k does not contain two distinct twins $b_i, b_j \prec^* b_{2^n+1}$.

Because of the choice of n, however, at least two nominals b_i and b_j among $b_1, ..., b_{2^n+1}$ must be twins in S_k (i.e. they must label the same set of native formulae).

□

3.2 Completeness

In this section we prove that if Θ is a complete and open branch of an H^+ tableau rooted at S_0, then S_0 is satisfiable. The overall structure of the completeness proof is the same as the corresponding one for H [5], and exploits the termination theorem: we first consider the set labelling the last node of Θ, that is *downward saturated* (in some sense), and we show that any such set has a model. Then the model existence property is propagated upward to the root node. However, in the presence of substitution, the model existence argument is technically subtler, because of the interplay between Sub and the nominal generating rules, accompanied by the blocking mechanism.

The following notion of nominal representatives is used in order to define saturation.

Definition 3.6 Let Θ be a tableau branch, S a node of Θ and b a nominal occurring in S. The *representative* of b in S, written $\rho_S(b)$, is the nominal $a \prec^*_\Theta b$ such that a is a twin of b and a is not blocked, if it exists (undefined otherwise).

Note that $\rho_S(b)$ may be undefined. Consider in fact a situation where $a_1 \prec^+_\Theta a_2 \prec^+_\Theta b$ and a_1 and a_2 become twins after the generation of b (by effect of the converse rules). It may happen that, in the chain leading to b, there is no ancestor of b that is a twin of b (because of different choices in the expansion of disjunctive formulae). In such cases, b is blocked and $\rho_S(b)$ does not exist either. It is worth pointing out also that there is at most one non-blocked nominal $a \prec^*_\Theta b$ that is a twin of b. In fact, if a_1 and a_2 are distinct nominals such that $a_1 \prec^*_\Theta b$, $a_2 \prec^*_\Theta b$ and both a_1, a_2 are twins of b, then a_1 is also a twin of a_2, hence at least one among a_1, a_2 has a twin ancestor and is blocked.

The following result establishes useful properties of nominal representatives.

Lemma 3.7 *Let Θ be a tableau branch and S a node of Θ. Then:*

(i) *For any nominal a occurring in S, a is non-blocked in S if and only if $\rho_S(a) = a$. In particular, if a is a root nominal, $\rho_S(a) = a$.*

(ii) *Let a be a non-blocked nominal in S and b a nominal occurring in S. If either $a \prec_\Theta b$ or $b \prec_\Theta a$, then $\rho_S(b)$ is defined.*

(iii) *If $\rho_S(a) = b$ and F is native, then $@_a F \in S$ if and only if $@_b F \in S$.*

Here finally follows the notion of saturation, that is relative to a tableau node, since

clauses ix–xi make reference to non-blocked nominals and nominal representatives. Such notions, in turn, depend on the branch, since the relation $\prec^*_\Theta$ is branch-dependent.

Definition 3.8 A node S of a tableau branch Θ is *downward saturated* if and only if the following conditions hold:

(i) S contains neither a formula of the form $@_a\neg a$, nor two formulae of the form $@_a p$ and $@_a\neg p$ for some atom p.

(ii) If $@_a(F \wedge G) \in S$, then $@_a F, @_a G \in S$.

(iii) If $@_a(F \vee G) \in S$, then either $@_a F \in S$ or $@_a G \in S$.

(iv) If $@_a @_b F \in S$, then $@_b F \in S$.

(v) If $@_a b \in S$ then $a = b$.

(vi) If $@_a \Diamond_\tau b, @_a \Box_\tau F \in S$, then $@_b F \in S$.

(vii) If $@_b \Diamond_\tau a, @_a \Box^-_\tau F \in S$, then $@_b F \in S$.

(viii) If $@_a \mathsf{A} F \in S$, then for all nominals b occurring in S, $@_b F \in S$.

(ix) If $@_a \Diamond_\tau F \in S$, F is not a nominal and a is not blocked in S, then there is a nominal b such that $\rho_S(b)$ is defined and $@_a \Diamond_\tau b, @_b F \in S$.

(x) If $@_a \Diamond^-_\tau F \in S$ and a is not blocked in S, then there is a nominal b such that $\rho_S(b)$ is defined and $@_b \Diamond_\tau a, @_b F \in S$.

(xi) If $@_a \mathsf{E} F \in S$, then there is a nominal b such that $\rho_S(b) = b$ and $@_b F \in S$.

The following lemma proves the adequacy of the set of expansion rules.

Lemma 3.9 *Any complete and open tableau node is downward saturated.*

Proof. The delicate cases of the proofs are the items concerning the nominal generating formulae.

ix. Let us assume that $@_a \Diamond_\tau F \in S$, F is not a nominal, and a is not blocked in S. Then, since S is complete, $@_a \Diamond_\tau F$ is inactive in S. This means that $@_a \Diamond_\tau F$ is obtained by a number of substitutions (possibly none) from some $@_a \Diamond_\tau G$ which has been expanded above S,[4] i.e. $F = G[c_1 \mapsto d_1, ..., c_n \mapsto d_n]$, $n \geq 0$. The expansion of $@_a \Diamond_\tau G$ has generated some $@_a \Diamond_\tau b$ and $@_b G$. Since a still occurs in S, it has not been replaced or deleted, and consequently b has not been deleted. Since the same substitutions modifying $\Diamond_\tau G$ also affect b and G, S contains $@_a \Diamond_\tau b[c_1 \mapsto d_1, ..., c_n \mapsto d_n]$ and $@_b G[c_1 \mapsto d_1, ..., c_n \mapsto d_n]$. If b has not been replaced, then $@_a \Diamond_\tau b, @_b F \in S$, and, by Lemma 3.7.ii, $\rho_S(b)$ is defined. If $b = c_j$ for some j, then S contains $@_a \Diamond_\tau c_j$ and $@_{c_j} F$; since c_j is a native nominal, $\rho_S(b)$ is defined by Lemma 3.7.i.

Case x is treated similarly.

xi. Let us assume that $@_a \mathsf{E} F \in S$. Then, since S is complete, the memory of S contains $\mathsf{E} F$, and this means that some $@_c \mathsf{E} G$ has been expanded in a node S' above S, where

[4] Note that, here, $\Diamond_\tau G$ is necessarily labelled by a, because of the of the interaction between substitution and the active/inactive markings.

F is obtained from G by means of a number of substitutions (those applied in the sub-branch leading from S' to S); i.e. $F = G[c_1 \mapsto d_1, ..., c_n \mapsto d_n]$, $n \geq 0$. The expansion of $@_c\mathsf{E}G$ has generated some $@_dG$. Now, since d is a root nominal, it cannot be deleted (though it can be replaced), therefore S contains $(@_dG)[c_1 \mapsto d_1, ..., c_n \mapsto d_n] = @_bF$ for some $b \in \{d, d_1, ..., d_n\}$. Since b is in any case a root nominal (either $b = d$ or b is native), $\rho_S(b) = b$, by Lemma 3.7.i.

□

The following lemma defines a model (in some sense) of any open and complete tableau node S. Note that S necessarily contains at least one root nominal a_0, which is non-blocked in it, and, by Lemma 3.7.i, $\rho_S(a_0) = a_0$.

Lemma 3.10 *Let S be an open and complete tableau node and a_0 any root nominal occurring in S. Let $\mathcal{M}^*$ be the interpretation defined as follows:*

$$W = \{a \mid a \text{ is a non-blocked nominal occurring in } S\};$$

$$R_\tau = \{(\rho_S(a), \rho_S(b)) \mid @_a\Diamond_\tau b \in S \text{ and both } \rho_S(a), \rho_S(b) \text{ are defined}\};$$

$$\text{For every nominal } a \text{ occurring in } S : N^*(a) = \begin{cases} \rho_S(a) & \text{if } \rho_S(a) \text{ is defined} \\ a_0 & \text{otherwise} \end{cases}$$

$$I(a) = \{p \mid @_ap \in S\} \text{ for all } a \in W$$

If $a \in W$, $@_aF \in S$ and F has not the form $\Diamond_\tau b$ for some b such that $\rho_S(b)$ is undefined, then $\mathcal{M}^*_a \models F$.

Proof. We remark beforehand that since S is open and complete, it is downward saturated by Lemma 3.9. Let us assume that $a \in W$ and $@_aF \in S$, for $F \neq \Diamond_\tau b$ with $\rho_S(b)$ undefined. Since a is not blocked, $\rho_S(a)$ is defined, and $\rho_S(a) = a$ (by Lemma 3.7.i). Therefore $N^*(a) = a$. The proof that $\mathcal{M}^*{}_a \models F$ is by induction on F.

Base We distinguish three cases.

(i) F is a literal. If F is a propositional letter or its negation, then the result is true by construction. In fact, if $@_ap \in S$, for $p \in \mathsf{PROP}$, $\mathcal{M}^*_a \models p$ by definition, because $N^*(a) = a$. If $@_a\neg p \in S$, since S is open, $@_ap \notin S$ and again $\mathcal{M}^*_a \models \neg p$ by construction.

(ii) F is a nominal b. Then necessarily $b = a$, since S is saturated, and the result is trivial, since $N^*(a) = a$.

(iii) F is $\neg b$, for some nominal b. Since S is open, $b \neq a$. By Lemma 3.2, b is a native nominal, hence it is non-blocked and belongs to W, so that $N^*(b) = b \neq a$. Therefore $\mathcal{M}^*_a \models \neg b$.

Induction Step Several cases have to be considered, according to the form of F. Here follows the treatment of some of them, the others being either very simple or similar to those shown below.

(i) $F = \mathsf{E}G$. If $@_a\mathsf{E}G \in S$, since S is saturated, there is a nominal b such that $\rho_S(b) = b$ and $@_bG \in S$. By Lemma 3.7.i, b is not blocked in S, so that $b \in W$.

By the inductive hypothesis, $\mathcal{M}_b^* \models G$, thus $\mathcal{M}_a^* \models \mathsf{E}G$.

(ii) F is $\Diamond_\tau G$. We distinguish two cases.

(a) G is a nominal b. By hypothesis, $\rho_S(b)$ is defined and belongs to W. So, let $\rho_S(b) = c$. By construction of $\mathcal{M}^*$, we have: $N^*(a) = a$, $N^*(b) = \rho_S(b) = c$ and $aR_\tau c$. Hence $\mathcal{M}_a^* \models \Diamond_\tau b$.

(b) G is not a nominal, and therefore it is native. Since S is saturated and $a \in W$ is not blocked, there is a nominal b such that $\rho_S(b)$ is defined and $@_a\Diamond_\tau b, @_b G \in S$. Since $\rho_S(b)$ is a twin of b and G is native, if $c = \rho_S(b)$ we have $@_c G \in S$ by Lemma 3.7.iii. By construction of $\mathcal{M}^*$, $aR_\tau c$, and, by the inductive hypothesis, $\mathcal{M}_c^* \models G$. Hence $\mathcal{M}_a^* \models \Diamond_\tau G$.

(iii) F is $\Diamond_\tau^- G$. Observe that, differently from the above case, G is necessarily native (Lemma 3.2), because $@_a\Diamond_\tau^- c$ is not a relational formula, and this case is treated similarly to the second item of case ii.

(iv) $F = \Box_\tau^- G$. Let b be any element of W such that $bR_\tau a$. By definition, there are two nominals c and d such that $a = \rho_S(c)$, $b = \rho_S(d)$ and $@_d\Diamond_\tau c \in S$. Since a and c are twins and $\Box_\tau^- G$ is native, $@_c\Box_\tau^- G \in S$ by Lemma 3.7.iii. And since S is saturated, $@_d G \in S$, so that also $@_b G \in S$ because b and d are twins and G is native (Lemma 3.7.iii again). By the induction hypothesis, then $\mathcal{M}_b^* \models G$. Therefore $\mathcal{M}_a^* \models \Box_\tau^- G$.

□

Completeness is proved by lifting the model existence property upwards to the root set.

Theorem 3.11 (Completeness) *If S is unsatisfiable, then every complete tableau for S is closed.*

Proof. The proof consists in showing that if Θ is a complete and open branch of a tableau rooted at S_0, then S_0 is satisfiable.

Since Θ is finite by Theorem 3.5, $\Theta = S_0, S_1, ..., S_k$ for some k, where S_k is a complete and open node.

Let $\mathcal{M}^* = \langle W, \{R_\tau \mid \tau \in \mathsf{REL}\}, N^*, I\rangle$ be the model of S_k given by Lemma 3.10. 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. In order to do it, we define an equivalence relation on nominals (with respect to the branch Θ) as follows: $a \sim b$ if some node of Θ contains $@_a b$. The relation $\approx$ is the reflexive, symmetric and transitive closure of $\sim$. If a_0 is any native nominal occurring in S_k, then N is the extension of N^* such that for all nominals c occurring in Θ:

$$N(c) = \begin{cases} N^*(c) & \text{if } c \in W, \text{ i.e. } c \text{ occurs in } S_k \\ N^*(d) & \text{if for some } d \in W,\ c \approx d \\ a_0 & \text{otherwise} \end{cases}$$

It is clear that if some node of Θ contains $@_a b$, then $N(a) = N(b)$.

If $\mathcal{M} = \langle W, \{R_\tau \mid \tau \in \mathsf{REL}\}, N, I\rangle$, obviously, it still holds that for every $@_a F \in S_k$, if $a \in W$ and F has not the form $\Diamond_\tau b$ for some b with $\rho_{S_k}(b)$ undefined, then $\mathcal{M}_{N(a)} \models F$. We now prove that the satisfaction property propagates upwards, restricting our attention to nominals that are not deleted in Θ. Let us say that a formula $@_a F$ is *relevant* (w.r.t. Θ) if and only if either F is native, or both the following conditions hold:

- $@_a F$ contains only nominals that are never deleted in Θ, and
- F has not the form $\Diamond_\tau b$ for some b with $\rho_{S_k}(b)$ undefined.

Let us say that an interpretation $\mathcal{M}$ is a Θ-model of a node S of Θ if for every relevant formula $@_a F \in S$, $\mathcal{M}_{N(a)} \models F$. Obviously the specific interpretation $\mathcal{M}$ defined above is a Θ-model of S_k. 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 set obviously contains only native (hence relevant) formulae.

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

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

Since $N(a) = N(b)$ (by definition), $\mathcal{M}_{N(a)} \models b$, therefore $\mathcal{M}_{N(a)} \models @_a b$.

Let now $@_c F$ be any relevant formula in $S' = S_i \setminus \{@_a b\}$ such that $@_c F \neq (@_c F)[a \mapsto b] \in S_{i+1}$. If $@_c F$ is relevant then also $(@_c F)[a \mapsto b]$ is relevant. In fact:

- if $@_c F$ is native, then also $(@_c F)[a \mapsto b]$ is native because b is a native nominal (Lemma 3.2).
- If $@_c F$ contains only nominals that are never deleted in Θ, then the same holds for $(@_c F)[a \mapsto b]$. In fact, the only nominal possibly occurring in $(@_c F)[a \mapsto b]$ and not in $@_c F$ is b, and b is native (Lemma 3.2), so it cannot be deleted.
- If $@_c F$ is not a relational formula, obviously $(@_c F)[a \mapsto b]$ is not a relational formula either. So let us assume that $F = \Diamond_\tau d$ where $\rho_{S_k}(d)$ is defined. If $d \neq a$ there is nothing to prove (in fact, $(@_c \Diamond_\tau d)[a \mapsto b] = @_{c[a \mapsto b]} \Diamond_\tau d$ and $\rho_{S_k}(d)$ is defined). If $d = a$, then $(@_c \Diamond_\tau d)[a \mapsto b] = @_{c[a \mapsto b]} \Diamond_\tau b$. Since $\Diamond_\tau b$ is native, $(@_c F)[a \mapsto b]$ is relevant.

Therefore by the inductive 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. 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 $\mathcal{M}_{N(c)} \models F$. □

4 Concluding remarks

In this work we have extended the treatment of nominal equalities by means of substitution and nominal deletion, proposed in [6,5] for $HL(@)$, to the global and converse modalities. The approach followed in this work is *procedural*, in that the formal system embodies algorithmic choices. As a consequence, the proposed calculus, proved to be sound, complete and terminating, can be almost directly "synthesized" into a running prover, with a minimal need of checking that the good properties of the system are preserved.

The features of the calculus defined in this paper can be summarized into the following main points:

(i) No use of prefixes: the calculus is *internalized*.

(ii) A practical approach to the treatment of nominal equalities: they are handled by means of substitution, accompanied by the deletion of the chain of nominals generated by the replaced one.

(iii) Ancestor equality blocking for the $\Diamond_\tau$ and $\Diamond_\tau^-$ rules: formulae of the form $@_a\Diamond_\tau F$ or $@_a\Diamond_\tau^- F$ are not expanded when either there exists an ancestor b of a which labels the same set of native formulae as a (a is directly blocked) or the father of a is blocked (a is indirectly blocked).

(iv) A mechanism to prevent useless re-applications of the E-rule: a formula of the form $@_a\mathsf{E}F$ is not expanded whenever any $@_b\mathsf{E}F$ has already been expanded.

As regard to point iii, indirect blocking and equality loop checks (as opposed to subset blocking) are a necessity in the presence of the converse modalities. The paper shows that, in such a context, a simple substitution rule with no side effects yields a non-terminating calculus. And in fact, although substitution-based decision procedures have been defined for $HL(@, \mathsf{E})$, to our knowledge none of them has been extended to deal with the converse modalities.

Point ii is quite similar to the "merge and prune" mechanism used in [14], where a tableau calculus for description logic with nominals, number restrictions and transitive and inverse roles is defined. The overall context is however quite different: there, a tableau branch is a graph of nodes, labelled by sets of formulae and describing worlds in the model. Nodes containing a nominal in their label are called *nominal nodes*, the others are *blockable nodes*. When two nodes contain the same nominal, they are merged into one of them and the blockable successors of the other node (which is deleted) are "pruned". The blocking mechanism used there only affects blockable nodes, and requires *pairwise blocking* (pairwise blocking seems to be a necessity when both inverse roles and number restrictions are present). The possibility and usefulness of exporting some of the ideas in our approach to the context of description logics have to be explored.

The first tableau-based decision procedures for $HL(@, \mathsf{E})$ were proposed in [4], where systems defined in [17,18,1] are reformulated, and the global modalities are added. The first two systems (reformulations of [17] and [18]) use prefixes, while the third one is internalized (like in point i of our approach). The treatment of nominal equalities in the three systems is different: either by copying formulae from a "world" to another, or by

means of substitution (similarly to point ii, but without nominal deletion), or else by means of a set of quite "natural" rules for equality. The procedures involve loop-checks based on subset blocking (which is sufficient in the absence of the converse modalities).

The first tableau-based decision procedure for hybrid logic including both the global and converse modalities was given in [2]. Ancestor equality blocking (point iii above) is the same blocking mechanism used *for every nominal generating rule* in the prefixed calculus defined there (and the internalized one given in [3]). The main difference between H and H^+ and the internalized calculi for the corresponding logics in [2,3] is represented by ii. In fact, in the latter systems, an equality $@_a b$ is handled by copying formulae labelled by a to b, except for accessibility formulae. When the copied formula generates a new nominal, two copies of such a nominal are therefore expanded, which can in turn generate copies of the same nominals, and so on. The redundancy of such a treatment of equalities and the computational gain of substitution with nominal deletion has been experimentally verified. The results of the comparison are reported in [9,8], which also compares the implementation of H with HTab [12], the more mature prover based on the prefixed calculus defined in [2].

As already remarked, the treatment of existential formulae in H^+ (point iv above) recalls the mechanism used in [16], which defines another procedure including the converse and difference modalities (which subsume the global ones). A similar mechanism (called *pattern-based blocking*) is used to block the expansions of $\Diamond$-formulae for the sublogic with no converse operators. In fact, pattern-based blocking yields a terminating system only in the absence of the converse modalities, and provided that applications of the $\Box$-rule are prioritized. When the converse modalities are present, chain-based blocking (with indirect blocking) has to be used. In [16], equalities are dealt with in an abstract and declarative way, constituting a still different approach w.r.t. point ii above. Algorithmic choices about the concrete treatment of such formulae are left open. A corresponding procedural approach is presented in [15], where a substitution rule, without nominal deletion, is used, in the context of the difference modality, but no converse operators. Differently from H^+ (point iii), pattern-based blocking is applied there to block the application of any nominal generating rule. Spartacus [11,10], the prover implementing tableaux for $HL(@, \mathsf{E})$ on the basis of the works in question, processes nominal equalities by merging the content of the corresponding "nodes", and electing one of them as the representative of both. Again, nominal generating formulae are there treated by pattern-based blocking.

Guidelines for future work include, on the practical side, the extension of the already mentioned prover [9,8], which at present implements only the restricted calculus H, and its refinement so as to include some basic optimization techniques. This would allow for an experimental verification of the fact that the extended calculus still benefits of the computational advantages of substitution with nominal deletion. On the theoretical side, the integration of the substitution mechanism used in this work with still more expressive languages can be studied.

Acknowledgements. The authors thank the anonymous referees of this work, who, with their useful comments and suggestions, gave the opportunity to make the presen-

tation clearer and more accurate.

References

[1] Blackburn, P. 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)*, LNAI **2381** (2002), pp. 38–52.

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

[3] Bolander, T. and P. Blackburn, *Terminating tableau calculi for hybrid logics extending K*, Electronic Notes in Theoretical Computer Science **231** (2009), pp. 21–39, proceedings of the 5th Workshop on Methods for Modalities (M4M-5), 2007.

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

[5] Cerrito, S. and M. Cialdea Mayer, *An efficient approach to nominal equalities in hybrid logic tableaux*, Journal of Applied Non-classical Logics (To appear).

[6] Cerrito, S. 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`).

[7] Cerrito, S. 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`).

[8] Cialdea Mayer, M. and S. Cerrito, *Herod and Pilate: two tableau provers for basic hybrid logic*, in: J. Giesl and R. Hähnle, editors, *Proceedings of IJCAR 2010*, LNAI **6173** (2010), pp. 255–262.

[9] Cialdea Mayer, M., 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`).

[10] Götzmann, D., "Spartacus: A Tableau Prover for Hybrid Logic," Master's thesis, Saarland University (2009).

[11] Götzmann, D., M. Kaminski and G. Smolka, *Spartacus: A tableau prover for hybrid logic*, in: *M4M6*, number 128 in Computer Science Research Reports, Roskilde University, 2009, pp. 201–212.

[12] Hoffmann, G. and C. Areces, *HTab: A terminating tableaux system for hybrid logic*, Electronic Notes in Theoretical Computer Science **231** (2009), pp. 3–19, proceedings of the 5th Workshop on Methods for Modalities (M4M-5), 2007.

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

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

[15] Kaminski, M. and G. Smolka, *Hybrid tableaux for the difference modality*, Electronic Notes in Theoretical Computer Science **231** (2009), pp. 241–257, proceedings of the 5th Workshop on Methods for Modalities (M4M-5), 2007.

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

[17] Tzakova, M., *Tableau calculi for hybrid logics*, in: N. Murray, editor, *Automated Reasoning with Analytic Tableaux and Related Methods (TABLEAUX 1999)*, LNAI **1617** (1999), pp. 278–292.

[18] van Eijck, J., *Constraint tableaux for hybrid logics* (2002), manuscript, CWI, Amsterdam.