Constraint Tableaux for Hybrid Logics
Draft

Jan van Eijck
CWI and ILLC, Amsterdam, Uil-OTS, Utrecht

February 6, 2002

Abstract

Hybrid logics are modal logics with names for worlds. Tableau systems for various hybrid logics were proposed in [6, 7, 20]. We present an improved tableau system for hybrid logic that handles equality for nominals by substitution and generation of inequality constraints. This compiles out the equalities while storing the inequalities, thus allowing for efficient equality reasoning. The proof procedure based on the tableau calculus —the constraint proof engine — uses two other kinds of constraints: box constraints and inverse box constraints. Completeness of the system follows in the usual way from fairness of the proof procedure, together with a model generation argument.

Next, calculus and proof engine are extended to incorporate the universal modality. Among other things, universal modalities allow us to tune the proof engine to specific frame classes. This leads to a general completeness result for the calculus, for all frame classes that can be described by a $A\varphi$ formula. We also propose rules for minimal model generation, and a compression algorithm for generating minimal models from complete open tableau nodes. Finally, we focus on some fragments of hybrid logic that are decided by the constraint proof engine.

A theorem prover for hybrid logic based on the constraint proof engine, *HyLoTab*, has been implemented. A companion paper [11] contains the full code of of this implementation in Haskell [15], in 'literate programming' style [16]. This documented code can be found at `http://www.cwi.nl/~jve/hylotab`.

Keywords: Hybrid logic, tableau reasoning, equality reasoning, model generation, decision methods.

MSC codes: 03B10, 03F03, 68N17, 68T15

1 Hybrid Logic: Syntax and Semantics

Our starting point is $\mathcal{HL}(@,^{\smile},\downarrow)$, a hybrid logic language [1, 2] with the following syntax. Assume p ranges over a set of propositions $\{p_0, p_1, \ldots\}$, c over a set of constant nominals $\{c_0, c_1, \ldots\}$, x over a set of variable nominals $\{x_0, x_1, \ldots\}$, n over the sets of constant nominals $\{c_0, c_1, \ldots\}$ and

variable nominals $\{x_0, x_1, \ldots\}$, and i over a set of relation indices $\{i_0, i_1, \ldots\}$. Then $\mathcal{HL}(@,{}^{\smile},\downarrow)$ is given by:

$$\varphi \quad ::= \quad \top \mid \bot \mid p \mid c \mid x \mid \neg\varphi \mid \varphi \wedge \varphi' \mid \varphi \vee \varphi' \mid \varphi \rightarrow \varphi' \mid [\,i\,]\varphi \mid [\,i\,]^{\smile}\varphi \mid \langle\, i\,\rangle\varphi \mid \langle\, i\,\rangle^{\smile}\varphi \mid @n\varphi \mid \downarrow x.\varphi.$$

$\mathcal{HL}(@,{}^{\smile})$ is the hybrid language that results from leaving out the binders $\downarrow x.\varphi$ from $\mathcal{HL}(@,{}^{\smile},\downarrow)$, and $\mathcal{HL}(@)$ is the hybrid language that results from leaving out the converse modalities $[\,i\,]^{\smile}\varphi$ and $\langle\, i\,\rangle^{\smile}\varphi$ from $\mathcal{HL}(@,{}^{\smile})$.

Models for $\mathcal{HL}(@,{}^{\smile},\downarrow)$ consist of a set of woirld W, a set $R = \{R_i \mid i \in I\}$ of binary relations on W, a valuation function V that maps proposition letters to subsets of W, and constant nominals to singleton subsets of W. Let $\mathcal{M} = (M, R, V)$ be such a model, and let w be a world from $\mathcal{M}$. Let g be a valuation that maps variable nominals to members of W. Then the crucial clauses of the semantics for $\mathcal{HL}(@,{}^{\smile},\downarrow)$ are given by:

$$\begin{array}{rll}
\mathcal{M}, g, w \Vdash p & \text{iff} & w \in V(p) \\
\mathcal{M}, g, w \Vdash c & \text{iff} & \{w\} = V(c) \\
\mathcal{M}, g, w \Vdash x & \text{iff} & w = g(x) \\
\mathcal{M}, g, w \Vdash [\,i\,]\varphi & \text{iff} & \text{for all } w' \text{ with } wR_iw' \text{ it holds that } \mathcal{M}, g, w' \Vdash \varphi \\
\mathcal{M}, g, w \Vdash \langle\, i\,\rangle\varphi & \text{iff} & \text{for some } w' \text{ with } wR_iw' \text{ it holds that } \mathcal{M}, g, w' \Vdash \varphi \\
\mathcal{M}, g, w \Vdash [\,i\,]^{\smile}\varphi & \text{iff} & \text{for all } w' \text{ with } w'R_iw \text{ it holds that } \mathcal{M}, g, w' \Vdash \varphi \\
\mathcal{M}, g, w \Vdash \langle\, i\,\rangle^{\smile}\varphi & \text{iff} & \text{for some } w' \text{ with } w'R_iw \text{ it holds that } \mathcal{M}, g, w' \Vdash \varphi \\
\mathcal{M}, g, w \Vdash @n\varphi & \text{iff} & \mathcal{M}, g, w' \Vdash \varphi \text{ where } \{w'\} = V(n) \\
\mathcal{M}, g, w \Vdash \downarrow x.\varphi & \text{iff} & \mathcal{M}, g^x_w, w \Vdash \varphi.
\end{array}$$

Here g^x_w is like g, except possibly for the fact that it maps x to w.

2 A Tableau Calculus for $\mathcal{HL}(@,{}^{\smile},\downarrow)$

The tableau rules for $\mathcal{HL}(@,{}^{\smile},\downarrow)$ work on the labelled version of this language, with inequalities and access statements added. The tableau rules use the following language ($\varphi \in \mathcal{HL}(@,{}^{\smile},\downarrow)$):

$$\psi \quad ::= \quad m \not\approx n \mid mRn \mid @n\varphi.$$

We assume a linear ordering $<$ on the set of nominals. An inequality $m \not\approx n$ generated by the tableau rules will always satisfy $m \leq n$. Inequalities $m \not\approx m$ are written as $\bot$.

As in [6, 7], the tableau rules are rules for labelled formulas. In fact, the tableau system of this paper is a variation on these calculi, with a different approach to equality reasoning for nominals.

The $[\,i\,]$ and $[\,i\,]^{\smile}$ rules are the only rules that cannot be treated once and for all in the proof procedure. They have the 'standing order' nature of the γ tableau rules in first order logic [19]. In the proof procedure based on the calculus they are translated into constraints (see Section 6).

Conjunctive rules (α rules)

$$\frac{@m(\varphi \wedge \psi)}{\begin{array}{c}@m\varphi\\ @m\psi\end{array}} \qquad \frac{@m\neg(\varphi \vee \psi)}{\begin{array}{c}@m\neg\varphi\\ @m\neg\psi\end{array}} \qquad \frac{@m\neg(\varphi \rightarrow \psi)}{\begin{array}{c}@m\varphi\\ @m\neg\psi\end{array}}$$

Disjunctive rules (β rules)

$$\frac{@m(\varphi \vee \psi)}{@m\varphi \mid @m\psi} \qquad \frac{@m\neg(\varphi \wedge \psi)}{@m\neg\varphi \mid @m\neg\psi} \qquad \frac{@m(\varphi \rightarrow \psi)}{@m\neg\varphi \mid @m\psi}$$

Negation rule As this is a single-sided calculus, the only negation rule we need is the rule for double negation.

$$\frac{@m\neg\neg\varphi}{@m\varphi}$$

$[\,i\,]$ **rules** Like the γ rules in FOL tableaux, these are 'standing orders'.

$$\frac{@m[\,i\,]\varphi, mR_in}{@n\varphi} \qquad \frac{@m\neg\langle\, i\,\rangle\varphi, mR_in}{@n\neg\varphi}$$

$\langle\, i\,\rangle$ **rules**

$$\frac{@n\langle\, i\,\rangle\varphi}{\begin{array}{c}nR_im\\ @m\varphi\end{array}}\ \varphi \text{ not a nominal, } m \text{ fresh} \qquad \frac{@n\neg[\,i\,]\varphi}{\begin{array}{c}nR_im\\ @m\neg\varphi\end{array}}\ \varphi \text{ not a negated nominal, } m \text{ fresh}$$

$[\,i\,]^{\breve{}}$ **rules** Like the γ rules in FOL tableaux, these are 'standing orders'.

$$\frac{@m[\,i\,]^{\breve{}}\varphi, nR_im}{@n\varphi} \qquad \frac{@m\neg\langle\, i\,\rangle^{\breve{}}\varphi, nR_im}{@n\neg\varphi}$$

$\langle\, i\,\rangle^{\breve{}}$ **rules**

$$\frac{@n\langle\, i\,\rangle^{\breve{}}\varphi}{\begin{array}{c}mR_in\\ @m\varphi\end{array}}\ \varphi \text{ not a nominal, } m \text{ fresh} \qquad \frac{@n\neg[\,i\,]^{\breve{}}\varphi}{\begin{array}{c}mR_in\\ @m\neg\varphi\end{array}}\ \varphi \text{ not a negated nominal, } m \text{ fresh}$$

Access rules Formulas of the forms $\langle i \rangle n$, $\neg[i]\neg n$, $\langle i \rangle\breve{}\, n$, $\neg[i]\breve{}\neg n$, are called access formulas. They are treated by a separate rule.

$$\frac{@n\langle i \rangle m}{nR_i m} \qquad \frac{@n\neg[i]\neg m}{nR_i m} \qquad \frac{@n\langle i \rangle\breve{}\, m}{mR_i n} \qquad \frac{@n\neg[i]\breve{}\neg m}{mR_i n}$$

Label rules

$$\frac{@m@n\varphi}{@n\varphi} \qquad \frac{@m\neg@n\varphi}{@n\neg\varphi}$$

Nominal substitution Here comes the new element, a substitution rule that makes use of the fact that nominals are unique names. $\boldsymbol{B}^t_s$ is the result of substituting s for t everywhere in tableau branch $\boldsymbol{B}$. The rules make use of the linear order $<$ on nominals.

$$\frac{\boldsymbol{B} + @mm}{\boldsymbol{B}} \qquad \frac{\boldsymbol{B} + @mn}{\boldsymbol{B}^t_s} s = \min(m,n), t = \max(m,n)$$

Inequality generation Write $\bot$ for $m \not\approx m$.

$$\frac{@m\neg m}{\bot} \qquad \frac{@m\neg n}{s \not\approx t}\ s = \min(m,n), t = \max(m,n)$$

$$\frac{@m\varphi, @m\neg\varphi}{\bot} \qquad \frac{@m\varphi, @n\neg\varphi}{s \not\approx t}\ s = \min(m,n), t = \max(m,n)$$

The rule that derives an inequality constraint from $@m\varphi, @n\neg\varphi$ is a so-called admissible rule. It does not change the set of formulas that can be refuted or proved satisfiable, but admitting it to the calculus may shorten some tableau proofs.

Binding rules

$$\frac{@m \downarrow x.\varphi}{@m\varphi^x_m} \qquad \frac{@m\neg \downarrow x.\varphi}{@m\neg\varphi^x_m}$$

Here φ^x_m denotes the result of substituting m for all *free* occurrences of x in φ.

Tableau Closure A tableau branch is closed if it contains $\bot$. A tableau is closed if all its branches are closed. Note that **nominal substitution** can lead to branch closure.

3 Examples of Refutation Proofs, for $\mathcal{HL}(@)$

Figure 1 gives a tableau refutation of (1), with the active formulas of the branch repeated at each node. Figure 1 gives another version of this, without repetition of formulas, but with the substitution instructions indicated along the branch.

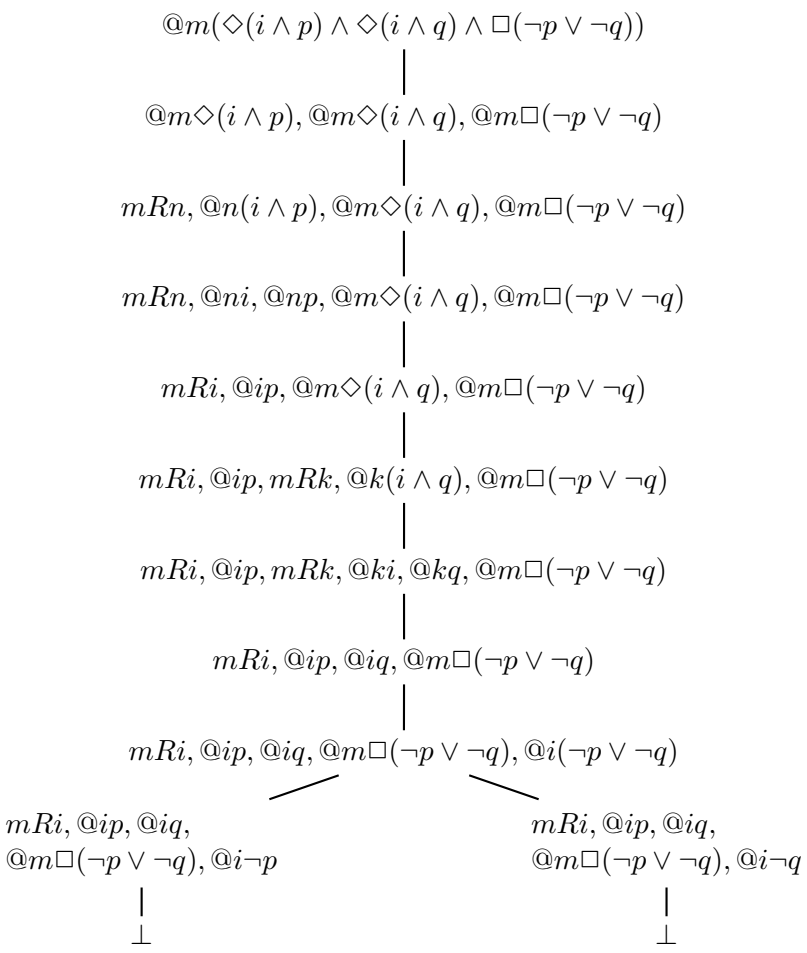

Figure 1: Tableau refutation for (1).

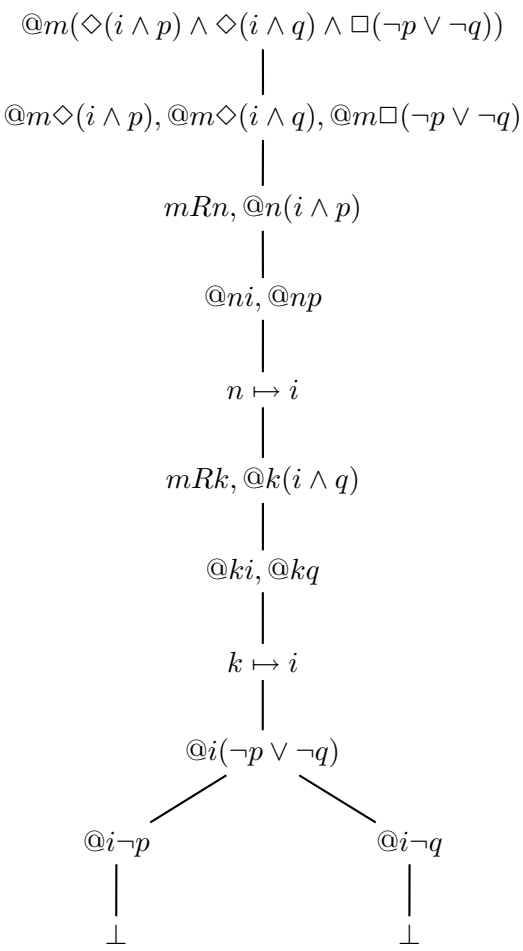

Figure 2: Tableau refutation for (1), without formula repetition.

$$@m(\Diamond(i \wedge p) \wedge \Diamond(i \wedge q) \wedge \Box(\neg p \vee \neg q)). \tag{1}$$

The tableau refutation of (1) proves the validity of $\Diamond(i \wedge p) \wedge \Diamond(i \wedge q) \rightarrow \Diamond(p \wedge q)$, which expresses that if from the current world i is accessible, and at i both p and q hold, then from the current world a $p \wedge q$ world is accessible.

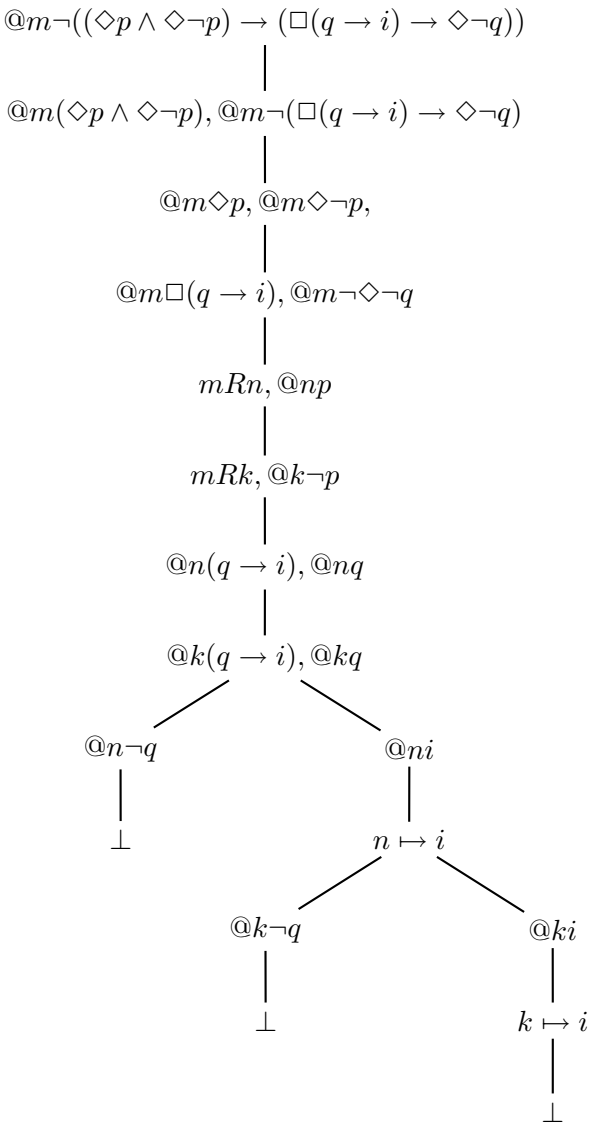

Figure 3: Tableau refutation for $@m\neg((\Diamond p \wedge \Diamond\neg p) \rightarrow (\Box(q \rightarrow i) \rightarrow \Diamond\neg q))$.

Figure 3 gives a tableau refutation of $@m\neg((\Diamond p \wedge \Diamond\neg p) \rightarrow (\Box(q \rightarrow i) \rightarrow \Diamond\neg q))$. The nominal substitutions $n \mapsto i$ and $k \mapsto i$ act on the formulas $@np$ and $@k\neg p$ to give $@ip, @i\neg p$ on the branch, and therefore branch closure. The tableau refutation proves the validity of $((\Diamond p \wedge \Diamond\neg p) \rightarrow (\Box(q \rightarrow i) \rightarrow \Diamond\neg q))$, a principle which expresses that if from the current world there are at least two worlds accessible, and if i is the only accessible q world, then there has to be an

accessible $\neg q$ world.

4 Examples of Model Generation, for $\mathcal{HL}(@)$

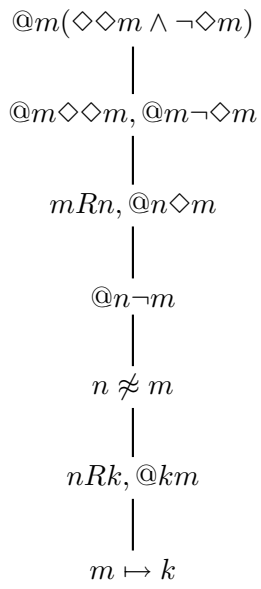

Figure 4: Open tableau for $@m(\Diamond\Diamond m \wedge \neg\Diamond m)$.

Figure 4 gives an open tableau for $@m(\Diamond\Diamond m \wedge \neg\Diamond m)$. Since the formula contains no proposition letters, it defines a frame property. The smallest frame with the property is $\bullet \leftrightarrow \bullet$.

Figure 5 gives an open tableau for $@m(\Diamond i \wedge \Diamond j \wedge \Diamond k \wedge @i\neg j \wedge @j\neg k \wedge @i\neg k)$.

Figure 6 shows that the formula $\Diamond(c \wedge \Diamond c \wedge \Box\Diamond c)$ is satisfiable. Note that this formula causes the first of the two tableau proof procedures in [20] to loop.

5 Examples of the treatment of $\downarrow$

Figure 7 gives a closed tableau for $@m \downarrow x.\Box\Diamond x \wedge \neg(\Diamond\Box p \rightarrow p)$. The formula $\downarrow x.\Box\Diamond x$ holds at the point m if mRn implies nRm. The formula $\Diamond\Box p \rightarrow p$ is true at points where R is symmetric. Therefore, $\downarrow x.\Box\Diamond x \rightarrow (\Diamond\Box p \rightarrow p)$ is valid.

In $\mathcal{HL}(\downarrow, @)$ it is easy to describe situations that only have infinite models. Figure 8 gives an infinite tableau for (2).

$$@s(\Diamond\top \wedge \Box\Box \downarrow x.@s\Diamond x \wedge \Box(\Diamond\top \wedge \varphi)), \tag{2}$$

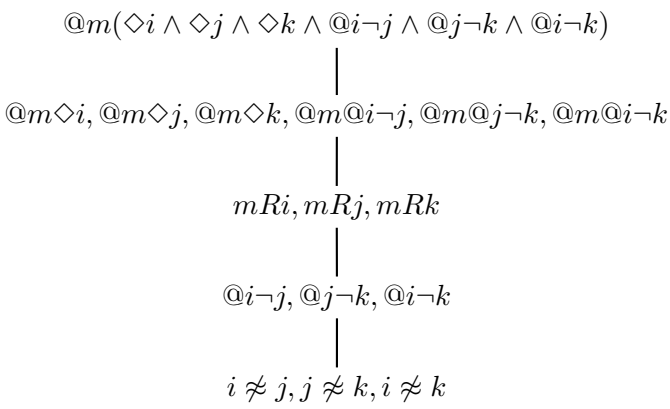

Figure 5: Open tableau for $@m(\Diamond i \wedge \Diamond j \wedge \Diamond k \wedge @i\neg j \wedge @j\neg k \wedge @i\neg k)$.

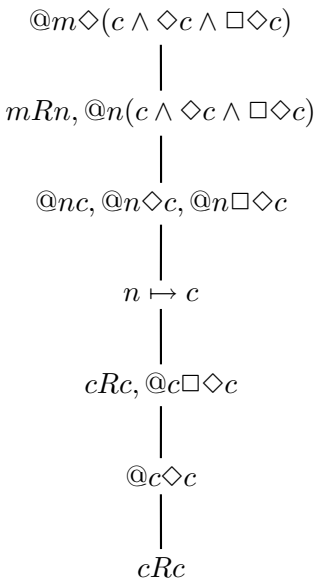

Figure 6: Open tableau for $\Diamond(c \wedge \Diamond c \wedge \Box\Diamond c)$.

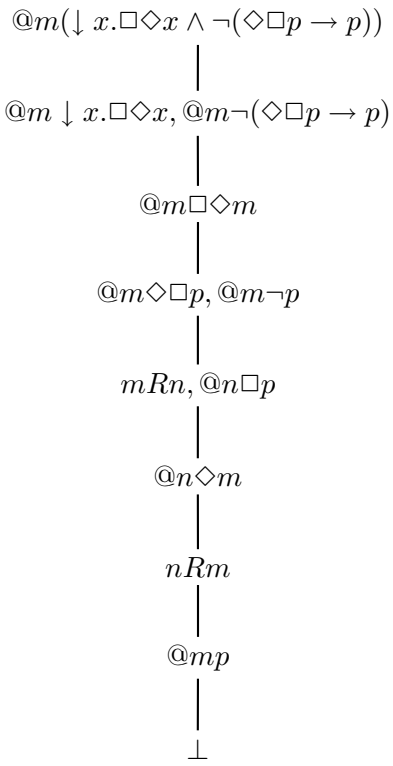

Figure 7: Closed tableau for $@m\ \downarrow x.\Box\Diamond x \wedge \neg(\Diamond\Box p \rightarrow p)$.

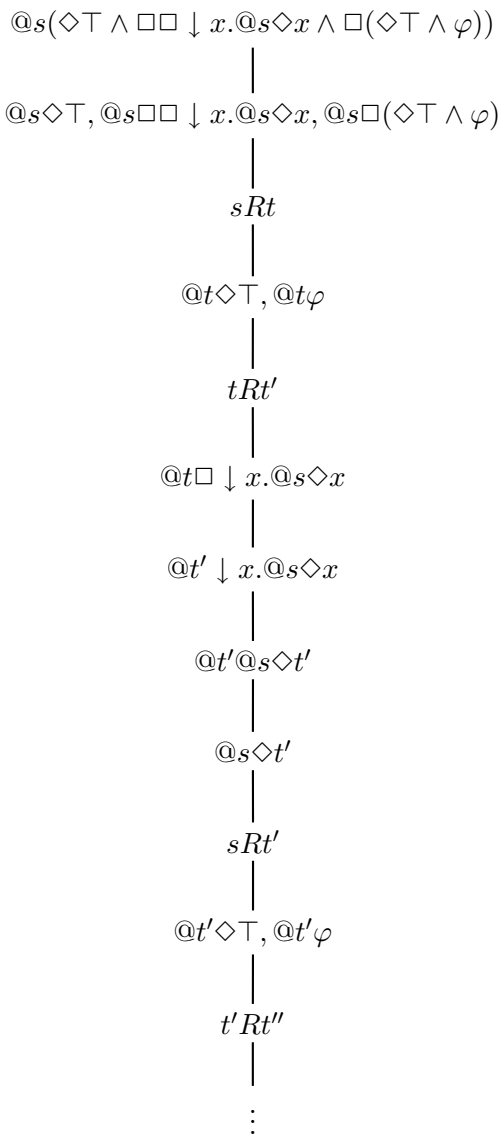

Figure 8: Infinite tableau for (2).

where φ is the formula

$$\downarrow x.\Box\neg x \wedge \downarrow x.\Box\Box\neg x \wedge \downarrow x.\Box\Box \downarrow y.@x\Diamond y.$$

The formula $\downarrow x.\Box\neg x$ expresses irreflexivity, the formula $\downarrow x.\Box\Box\neg x$ asymmetry, the formula $\downarrow x.\Box\Box \downarrow y.@x\Diamond y$ transitivity. Formula (2) expresses that from the spypoint s a serial, irreflexive, asymmetric and transitive, hence infinite, relation is visible.

6 From Tableau Calculus to Proof Engine

A proof procedure (or proof engine) for φ is a systematic search for a Kripke model satisfying φ, by means of a search tree starting out from a node *start* φ. Our proof engine for hybrid logic based on the calculus given above is presented in Figure 9. The proof rules work on tuples (U, A, I, P, N, C, F) consisting of a node universe (or node domain) U, a list of accessibilities A, a list of inequality constraints I, a list of positive propositional attributions P, a list of negative propositional attributions N, a list of box constraints and converse box constraints C, and a list of pending formulas F. Write $\varphi : F$ for the list with head φ and tail F, and $F_1 +\!\!+ F_2$ for the result of concatenating lists F_1, F_2. Write [] for the empty list, $[\varphi]$ for the unit list with φ as its element, and $[\varphi_1, \ldots \varphi_n]$ for the list consisting of φ_1 through φ_n. We will assume throughout that the lists do not contain duplicates, i.e., that the lists behave as ordered sets. In particular, U^s_t means that s gets replaced by t in U, and the duplicate of t that this may generate is removed.

The table uses abbreviations $\varphi \in \alpha, \varphi \in \beta, \varphi \in [\,i\,], \varphi \in \langle\, i\,\rangle, \varphi \in [\,i\,]^{\smile}, \varphi \in \langle\, i\,\rangle^{\smile}$ for "φ is an α formula", and so on. We do not count $\langle\, i\,\rangle n$ or $\langle\, i\,\rangle^{\smile} n$ or $\neg[\,i\,]\neg n$ or $\neg[\,i\,]^{\smile}\neg n$ as $\langle\, i\,\rangle$ or $\langle\, i\,\rangle^{\smile}$ formulas, as these 'access formulas' are treated by a separate rule. If $\varphi \in \alpha \cup \beta$, the components of φ are referred to as $\varphi_1, \ldots, \varphi_n$. If $\varphi \in [\,i\,] \cup \langle\, i\,\rangle \cup [\,i\,]^{\smile} \cup \langle\, i\,\rangle^{\smile}$, its component is referred to as φ'.

The key to understanding the table is the following invariant of the rule applications: the constraint store of the node contains only constraints that have been combined with all access relations of the node. This invariant may get destroyed when a new access relation gets added, when a new constraint gets added, or when a substitution is applied to a node. In all such cases, the invariant is restored by applying the appropriate constraints, thus generating extra material on the pending formula list.

Here is what has to happen in the case of applying a substitution to a node:

- If the substitution results in new access relations, these get combined with all box and converse box constraints of the node.
- If the substitution results in new box constraints, these get combined with all access relations of the node.
- If the substitution results in new converse box constraints, these get combined with all access relations of the node.

In the substitution rule in the table, this procedure is abbreviated as 'add stuff to restore the invariant'.

$$\frac{\text{start } \varphi}{[m], [], [], [], [], [], [@m\varphi]}\ m \text{ fresh}$$

$$\frac{U, A, I, P, N, C, F}{\text{closure}}\ \bot \in I \vee P \cap N \neq \emptyset \qquad \frac{U, A, I, P, N, C, []}{\text{success}}\ \bot \notin I, P \cap N = \emptyset$$

$$\frac{U, A, I, P, N, C, (@m\varphi : F)}{U, A, I, P, N, C, [@m\varphi_1, \ldots, @m\varphi_n] +\!\!+ F}\ \varphi \in \alpha$$

$$\frac{U, A, I, P, N, C, (@m\varphi : F)}{U, A, I, P, N, C, (@m\varphi_1 : F) \ \mid \ \cdots \ \mid \ U, A, I, P, N, C, (@m\varphi_n : F)}\ \varphi \in \beta$$

$$\frac{U, A, I, P, N, C, (@m\neg\neg\varphi : F)}{U, A, I, P, N, C, (@m\varphi : F)}$$

$$\frac{U, A, I, P, N, C, (@mp : F)}{U, A, I, (@mp : P), N, C, F} \qquad \frac{U, A, I, P, N, C, (@m\neg p : F)}{U, A, I, P, (@np : N), C, F}$$

$$\frac{U, A, I, P, N, C, (@mn : F)}{U^t_s, A^t_s, I^t_s, P^t_s, N^t_s, C^t_s, F^t_s +\!\!+ F'}\ s = \min(m, n), t = \max(m, n), F' = \text{ stuff to restore invariant}$$

$$\frac{U, A, I, P, N, C, (@m\neg n : F)}{U, A, (s \not\approx t : I), P, N, C, F}\ s = \min(m, n), t = \max(m, n)$$

$$\frac{U, A, I, P, N, C, (@m@n\varphi : F)}{U, A, I, P, N, C, (@n\varphi : F)} \qquad \frac{U, A, I, P, N, C, (@m\neg@n\varphi : F)}{U, A, I, P, N, C, (@n\neg\varphi : F)}$$

$$\frac{U, A, I, P, N, C, (@m\varphi : F)}{U, A, I, P, N, (@m\varphi : C), F +\!\!+ [@n\varphi' \mid mR_in \in A]}\ \varphi \in [\,i\,]$$

$$\frac{U, A, I, P, N, C, (@m\langle\, i\,\rangle n : F)}{U, (mR_in : A), I, P, N, C, F +\!\!+\!\!+ \ [@n\psi \mid @m[\,i\,]\psi \in C] +\!\!+ [@m\psi \mid @n[\,i\,]^{\smile}\psi \in C]}$$

$$\frac{U, A, I, P, N, C, (@m\varphi : F)}{(n : U), (mR_in : A), I, P, N, C, (@n\varphi' : F) +\!\!+ [@n\psi \mid @m[\,i\,]\psi \in C]}\ \varphi \in \langle\, i\,\rangle, n \text{ fresh}$$

$$\frac{U, A, I, P, N, C, (@m\varphi : F)}{U, A, I, P, N, (@m\varphi : C), F +\!\!+ [@n\varphi' \mid nR_im \in A]}\ \varphi \in [\,i\,]^{\smile}$$

$$\frac{U, A, I, P, N, C, (@m\langle\, i\,\rangle^{\smile} n : F)}{U, (nR_im : A), I, P, N, C, F +\!\!+ [@m\psi \mid @n[\,i\,]\psi \in C]F +\!\!+ [@n\psi \mid @m[\,i\,]^{\smile}\psi \in C]}$$

$$\frac{U, A, I, P, N, C, (@m\varphi : F)}{(n : U), (nR_im : A), I, P, N, C, (@n\varphi' : F) +\!\!+ [@n\psi \mid @m[\,i\,]^{\smile}\psi \in C]}\ \varphi \in \langle\, i\,\rangle^{\smile}, n \text{ fresh}$$

$$\frac{U, A, I, P, N, C, (@m \downarrow x.\varphi : F)}{U, A, I, P, N, C, (@m\varphi^x_m : F)} \qquad \frac{U, A, I, P, N, C, (@m\neg \downarrow x.\varphi : F)}{U, A, I, P, N, C, (@m\neg\varphi^x_m : F)}$$

Figure 9: Constraint Proof Engine for $\mathcal{HL}(@, ^{\smile}, \downarrow)$.

Closure, Success A node (U, A, I, P, N, C, F) is closed if either $\bot \in I$ or $P \cap N \neq \emptyset$, otherwise it is open. An open node (U, A, I, P, N, C, F) is a success node if it has $F = []$. A node is *incomplete* if it is neither closed nor a success node.

Selection Rule To develop an incomplete node, use the table from Figure 9. Since an incomplete node contains a non-empty list of pending formulas, and the table has exactly one rule that applies to the head φ of the pending formula list, this specifies the *selection rule* of the proof engine.

Search Rule To select an incomplete leaf node to develop, use any breadth first search method for tree traversal. This specifies a *search rule* for the proof engine.

Termination Condition The proof procedure ends when there are no more nodes to develop. The proof procedure can be used both for satisfiablity checking and for refutation. For satisfiability checking, the satisfiability procedure outputs *true* when a success node is encountered, and outputs *false* if the proof procedure ends with closed nodes at all leaves. For refutation, the refutation procedure outputs *false* when a success node is encountered, and outputs *true* if the proof procedure ends with closed nodes at all leaves.

7 Soundness, Fairness of Proof Engine, Completeness

Theorem 1 *The tableau calculus for $\mathcal{HL}(@, {}^{\smile}, \downarrow)$ is sound.*

Proof. By an easy inspection, all tableau rules are sound. Soundness of the calculus follows from this by induction on tableau structure. □

For completeness, we need a fair proof procedure P. Consider the (possibly infinite) set of tableaux for formula φ, according to procedure P. Since P determines which rule (selection) to apply to which node (search), this tableau set is ordered. The successor of a tableau $\boldsymbol{T}$ is tableau $\boldsymbol{T}'$ computed from $\boldsymbol{T}$ according to P. Let $\boldsymbol{T}_0$ be the initial tableau for φ. Then there is a possibly infinite ascending chain $\boldsymbol{T}_0, \ldots$ of tableaux for φ. By Zorn's lemma, this chain has a supremum $\boldsymbol{T}_\infty$.

A proof procedure for hybrid logic is fair if the following hold for all open branches $\boldsymbol{B}$ in $\boldsymbol{T}_\infty$:

1. All formulas of type $\alpha \cup \beta \cup (\neg\neg) \cup @ \cup \downarrow \cup \langle i \rangle \cup \langle i \rangle^{\smile}$ on $\boldsymbol{B}$ were used to expand $\boldsymbol{B}$.
2. All formulas of type $[i]$ or $[i]^{\smile}$ on $\boldsymbol{B}$ were combined with all mR_in accessibilities on $\boldsymbol{B}$ to expand $\boldsymbol{B}$.

Theorem 2 *The Constraint Proof Engine of Section 6 is fair.*

Proof. Inspection of the table in Figure 9 makes clear that (1) is satisfied. That (2) is also satisfied follows from an induction argument: it can be proved by induction on path length that

the constraint sets B and C at each node N consist of exactly the $[\,i\,] \cup [\,i\,]^{\smile}$ formulas at the node that were used to expand all appropriate mR_in accessibilities introduced on the path along $\boldsymbol{B}$ from the root up to N. Finally, note that formulas resulting from combining an accessibility relation mR_in and a constraint $@m[\,i\,]\varphi$ or $@n[\,i\,]^{\smile}\varphi$ are appended to the list of pending formulas, thus ensuring that every formula on the pending formula list will eventually get treated. □

Call a tableau *finished* if it does not contain incomplete end nodes. It is easy to see that each limit tableau $\boldsymbol{T}_\infty$ is finished. Every open branch of a finished tableau for φ yields a Kripke model for φ: take the nominals as worlds, put $m \xrightarrow{i} n$ if a mR_in relation occurs along the branch, make all proposition letters p with $@mp$ along the branch true at m, and all other proposition letters false at m. From the facts that the tableau is finished and that the branch is open it follows that this model is well-defined, and from the fact that the tableau rules are truth preserving in both directions it follows that it is indeed a model for φ.

Theorem 3 *The tableau calculus for $\mathcal{HL}(@,{}^{\smile},\downarrow)$ is complete.*

Proof. Immediate from Theorem 2, plus the fact that every open branch of a finished tableau for φ yields a Kripke model for φ. □

8 Adding the Universal Modality

Let us now extend the language with the universal modality A and its dual E, with the following intended meanings:

$$\begin{array}{lll} \mathcal{M},g,w \Vdash A\varphi & \text{iff} & \text{for all } w' \in M \text{ it holds that } \mathcal{M},g,w' \Vdash \varphi \\ \mathcal{M},g,w \Vdash E\varphi & \text{iff} & \text{for some } w' \in M \text{ it holds that } \mathcal{M},g,w' \Vdash \varphi \end{array}$$

It is well known that binding together with universal modality makes it possible to express full quantification. The definition of the quantifiers is as follows:

$$\begin{array}{lll} \mathcal{M},g,w \Vdash \forall x\varphi & \text{iff} & \text{for all } g' \text{ with } g \overset{x}{\sim} g' \text{ it holds that } \mathcal{M},g',w \Vdash \varphi \\ \mathcal{M},g,w \Vdash \exists x\varphi & \text{iff} & \text{for some } g' \text{ with } g \overset{x}{\sim} g' \text{ it holds that } \mathcal{M},g',w \Vdash \varphi \end{array}$$

It is not hard to see that $\forall x\varphi$ can be taken as shorthand for $\downarrow y.A \downarrow x.@y\varphi$, and $\exists x\varphi$ as shorthand for $\downarrow y.E \downarrow x.@y\varphi$. So we have full quantification once we know how to deal with the modalities A and E. Tableau rules for A and E can look like this:

$$\frac{@mA\varphi}{@n\varphi}\ n \text{ on the branch} \qquad \frac{@m\neg E\varphi}{@n\neg\varphi}\ n \text{ on the branch}$$

$$\frac{@mE\varphi}{@n\varphi}\ n \text{ fresh} \qquad \frac{@m\neg A\varphi}{@n\neg\varphi}\ n \text{ fresh}$$

Since the tableau rules for A and E are obviously sound, we get by induction on tableau structure:

Theorem 4 *The tableau calculus for $\mathcal{HL}(@,{}^{\smile},\downarrow,A)$ is sound.*

$$\frac{U, A, I, P, N, C, (@m\varphi : F)}{(n : U), A, I, P, N, C, (@n\varphi' : F) +\!\!+ [@n\psi \mid A\psi \in C]}\ \varphi \in E, n \text{ fresh}$$

$$\frac{U, A, I, P, N, C, (@m\varphi : F)}{U, A, I, P, N, (\varphi : C), F +\!\!+ [@n\varphi' \mid n \in U]}\ \varphi \in A$$

$$\frac{U, A, I, P, N, C, (@m\langle\, i\,\rangle n : F)}{U, (mR_i n : A), I, P, N, C, F +\!\!+ [@n\psi \mid @m[\, i\,]\psi \in C] +\!\!+ [@m\psi \mid @n[\, i\,]\breve{}\psi \in C] +\!\!+ [@m\psi \mid A\psi \in C]}$$

$$\frac{U, A, I, P, N, C, (@m\varphi : F)}{(n : U), (mR_i n : A), I, P, N, C, (@n\varphi' : F) +\!\!+ [@n\psi \mid @m[\, i\,]\psi \in C] +\!\!+ [@n\psi \mid A\psi \in C]}\ \varphi \in \langle\, i\,\rangle, n \text{ fresh}$$

$$\frac{U, A, I, P, N, C, (@m\langle\, i\,\rangle\breve{}n : F)}{U, (nR_i m : A), I, P, N, C, F +\!\!+ [@n\psi \mid @m[\, i\,]\breve{}\psi \in C] +\!\!+ [@n\psi \mid A\psi \in C]}$$

$$\frac{U, A, I, P, N, C, (@m\varphi : F)}{(n : U), (nR_i m : A), I, P, N, C, (@n\varphi' : F) +\!\!+ [@n\psi \mid @m[\, i\,]\breve{}\psi \in C] +\!\!+ [@n\psi \mid A\psi \in C]}\ \varphi \in \langle\, i\,\rangle\breve{}, n \text{ fresh}$$

Figure 10: Modified Constraint Proof Engine for $\mathcal{HL}(@,\breve{},\downarrow, A)$.

Figure 10 lists the modifications in the constraint proof engine that are necessary to accommodate the universal modality. The conditions $\varphi \in A$, $\varphi \in E$ have the obvious meanings. A-type formulas are of the forms $A\psi$, with component ψ, and $\neg E\psi$, with component $\neg\psi$. E-type formulas are of the forms $E\psi$, with component ψ, and $\neg A\psi$, with component $\neg\psi$. The constraint store C now also contains universal formulas $A\psi$, and the corresponding constraints have to be imposed on *all* nominals that turn up at the branch.

The results about fairness and completeness extend to the new calculus and proof engine:

Theorem 5 *The modified Constraint Proof Engine of Figure 10 is fair.*

Theorem 6 *The tableau calculus for $\mathcal{HL}(@,\breve{},\downarrow, A)$ is complete.*

9 Tuning the Engine: Proof Procedures for Frame Classes

The calculus and proof engine give a systematic search for Kripke models, without imposing any constraint on the kind of Kripke model. In other words, it interprets 'validity' as 'validity in models based on the class of *all* Kripke frames'. It is straightforward to adapt our reasoning engine for hybrid logic to other frame classes than the universal frame class. Since we have a way of dealing with the universal modality, the only thing we have to do is modify the start rule, by storing the appropriate constraint for the frame class we want. Here are some examples:

Irreflexive Frames for R_i Modify the start rule to:

$$\frac{\text{start } \varphi}{[m], [], [], [], [], [A \downarrow x.[\, i\,]\neg x], [@m\varphi]}\ m \text{ fresh}$$

Reflexive Frames for R_i Modify the start rule to:

$$\frac{\text{start } \varphi}{[m], [], [], [], [], [A \downarrow x.\langle i \rangle x], [@m\varphi]} \; m \text{ fresh}$$

Transitive Frames for R_i Modify the start rule to:

$$\frac{\text{start } \varphi}{[m], [], [], [], [], [A \downarrow x.[i][i]\langle i \rangle \breve{} x], [@m\varphi]} \; m \text{ fresh}$$

S4 Frames for R_i Modify the start rule to:

$$\frac{\text{start } \varphi}{[m], [], [], [], [], [A \downarrow x.(\langle i \rangle x \wedge [i][i]\langle i \rangle \breve{} x)], [@m\varphi]} \; m \text{ fresh}$$

It is obvious that this can be done for any frame class that can be described (hence characterized) by a formula in the first order correspondence language of hybrid logic. This is so since we have the full power of quantification available now; see [10]. This gives the following:

Theorem 7 (General Completeness) *The tableau calculus for $\mathcal{HL}(@, \breve{}, \downarrow, A)$ is complete for any frame class that can be described in the first order correspondence language of $\mathcal{HL}(@, \breve{}, \downarrow, A)$.*

10 Modifying the Calculus for Minimal Model Generation

A model $\mathcal{M}$ is minimal for φ if there are g, w with $\mathcal{M}, g, w \Vdash \varphi$, and for all $\mathcal{N}, h, w'$ with $\mathcal{N}, h, w' \Vdash \varphi$ it holds that $|N| \geq |M|$. E.g., the formula $\neg c \wedge \Box\Diamond\top$ has minimal models of size 2.

To generate minimal models, replace the $\langle i \rangle$ and $\langle i \rangle \breve{}$ rules by the following trial-and-error versions:

$\langle i \rangle$ rules, trial-and-error version

$$\frac{@n\langle i \rangle\varphi}{\begin{array}{c} nRk_1 \\ @k_1\varphi \end{array} \mid \cdots \mid \begin{array}{c} nRk_s \\ @k_s\varphi \end{array} \mid \begin{array}{c} nRm \\ @m\varphi \end{array}} \; k_1, \ldots k_s \text{ all nominals on the branch, } m \text{ fresh}$$

$$\frac{@n\neg[i]\varphi}{\begin{array}{c} nRk_1 \\ @k_1\neg\varphi \end{array} \mid \cdots \mid \begin{array}{c} nRk_s \\ @k_s\neg\varphi \end{array} \mid \begin{array}{c} nRm \\ @m\neg\varphi \end{array}} \; k_1, \ldots k_s \text{ all nominals on the branch, } m \text{ fresh}$$

$$\frac{@n\neg[i]\varphi}{\begin{array}{c} nRk_1 \\ @k_1\neg\varphi \end{array} \mid \cdots \mid \begin{array}{c} nRk_s \\ @k_s\neg\varphi \end{array} \mid \begin{array}{c} nRm \\ @m\neg\varphi \end{array}} \; k_1, \ldots k_s \text{ all nominals on the branch, } m \text{ fresh}$$

$\langle i \rangle^{\breve{}}$ rules, trial-and-error version

$$\frac{@n\langle i \rangle^{\breve{}}\varphi}{\begin{array}{c|c|c} k_1Rn & \cdots & Rk_sRn \\ @k_1\varphi & & @k_s\varphi \end{array} \;\Big|\; \begin{array}{c} mRn \\ @m\varphi \end{array}} \; k_1, \ldots k_s \text{ all nominals on the branch, } m \text{ fresh}$$

$$\frac{@n\neg[i]^{\breve{}}\varphi}{\begin{array}{c|c|c} k_1Rn & \cdots & k_sRn \\ @k_1\neg\varphi & & @k_s\neg\varphi \end{array} \;\Big|\; \begin{array}{c} mRn \\ @m\neg\varphi \end{array}} \; k_1, \ldots k_s \text{ all nominals on the branch, } m \text{ fresh}$$

Figure 11 gives an infinite tableau expansion for Formula (3).

$$@s\Diamond\neg s \wedge @s\Box \downarrow x.\Diamond(\neg s \wedge \neg x) \wedge @s\Box\Box \downarrow x.@s\Diamond x \tag{3}$$

Still, the formula has finite models, and we can find finite models of minimal size by means of (the proof engine based on) the trial-and-error version of the tableau system. See Figure 12.

Again, the new rules are obviously sound, so we get:

Theorem 8 *The minimal model calculus for $\mathcal{HL}(@,^{\breve{}},\downarrow,A)$ is sound.*

The rules lead to an obvious modification of the proof engine, which gives us a fair proof procedure for minimal model generation. From this, by the same reasoning as above:

Theorem 9 *The minimal model calculus for $\mathcal{HL}(@,^{\breve{}},\downarrow,A)$ is complete.*

11 Generation of Minimal Models Through Node Compression

Open branches in finished tableaux in general correspond to partial models rather than full models. E.g., in the propositional case, it does not matter what truth value one assigns to proposition letters not mentioned along open branches [5]. The same phenomenon presents itself in the case of hybrid logic. In general, an open tableau node of the form $(U, A, I, P, N, C, [])$ for φ can be used to generate a whole range of Kripke models for φ, including minimal models. The following node compression algorithm accomplishes this.

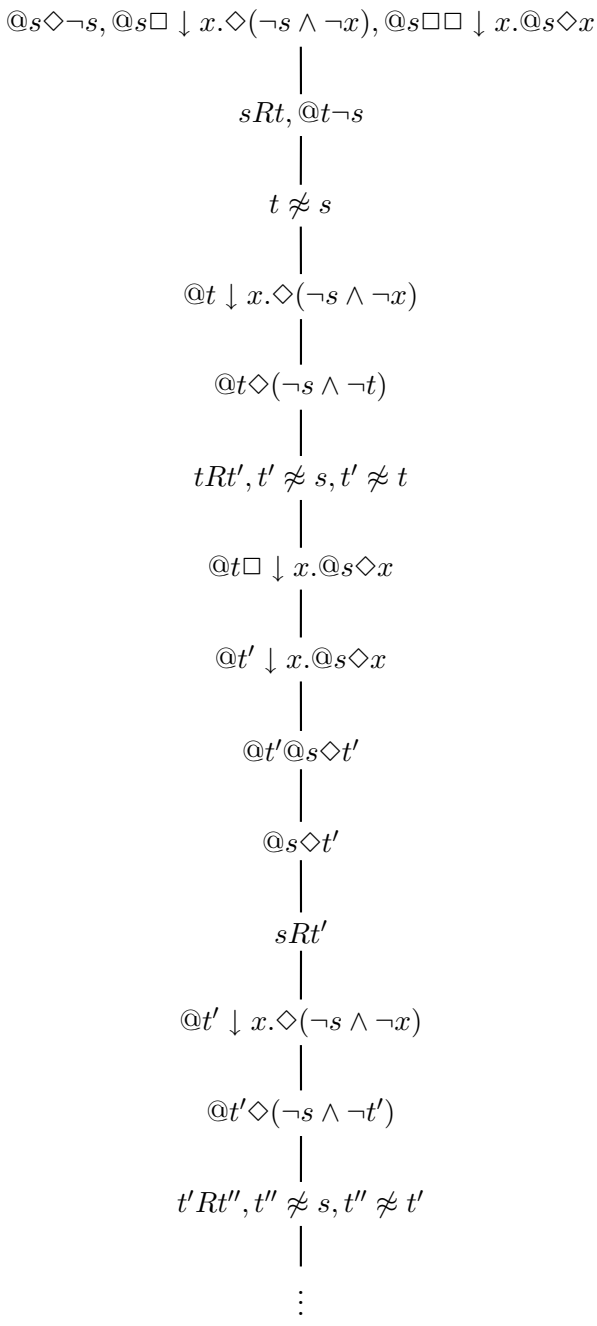

Figure 11: Tableau for (3).

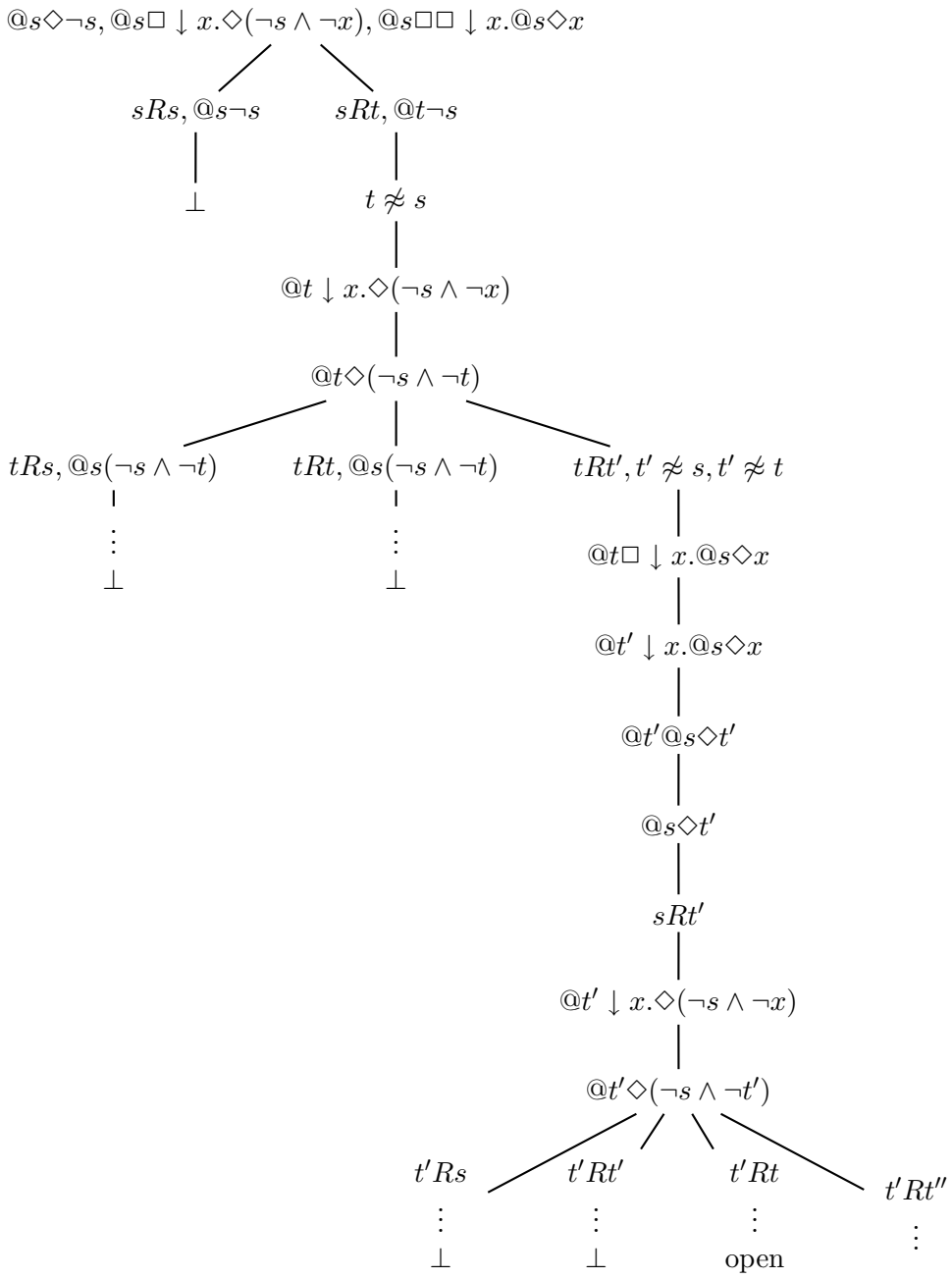

Figure 12: Trial-and-error tableau for (3).

Node Compression Algorithm

Let a finished open constraint tableau node $(U, A, I, P, N, C, [])$ be given.

- If there are no pairs of nominals m, n with m preceding n and $m \not\approx n \notin I$ then return the unit list $[(U, A, I, P, N, C, [])]$.
- Otherwise, nondeterministically pick a pair of nominals m, n with m preceding n and $m \not\approx n \notin I$.

  Compression Step Develop the tableau $[(U^n_m, A^n_m, I^n_m, P^n_m, N^n_m, [], C^n_m)]$.
  - If this remains open, then compress the finished open nodes and collect the results.
  - If it closes, then compress $(U, A, (m \not\approx n : I), P, N, C, [])$.

Theorem 10 (Soundness of Node Compression Algorithm) *If the node compression algorithm is applied to a finished open tableau node for φ, then all finished open tableau nodes that result from the algorithm satisfy φ.*

Proof. Induction on the number of nominal pairs m, n with m preceding n and $m \not\approx n \notin I$ on a finished open tableau node for φ. □

Theorem 11 (Completeness of Node Compression Algorithm) *If the node compression algorithm is applied to a finished open tableau node for φ it yields at least one open node*

$$(U, A, I, P, N, C, [])$$

that corresponds to a minimal model for φ.

Proof. If the algorithm is called with a finite node, termination is ensured, for every step either removes a pair m, n from the list of candidates for merging, or merges the pair. Also, upon termination, there are open nodes, for if $(U, A, I, P, N, C, [])$ is open then

$$(U, A, (m \not\approx n : I), P, N, C, [])$$

is open. Since the resulting open nodes cannot be compressed any further, at least one of them is minimal. □

To define the notion of a minimal model for a fragment of hybrid logic per se, we can use the notion of a hybrid bisimulation [1, 2], with the obvious extension to take care of the converse modalities. A Kripke model $\mathcal{M}$ is minimal for a hybrid language if for all models $\mathcal{N}$, all sequences $\bar{m} \in {}^kM$ and $\bar{n} \in {}^kN$, any hybrid bisimulation (for that language) $\overset{\omega}{\sim}$ with $(\mathcal{M}, \bar{m}) \overset{\omega}{\sim} (\mathcal{N}, \bar{n})$ and $\bar{m}(i) = \bar{m}(j)$ satisfies $\bar{n}(i) = \bar{n}(j)$. Intuitively, $\mathcal{M}$ is minimal for some hybrid logic language if no distinct worlds in $\mathcal{M}$ are ever identified by a hybrid bisimulation for that language. We leave the investigation of this notion for a future occasion.

12 Deciding Some Fragments of $\mathcal{HL}(\downarrow, {}^{\smile}, @)$

It is well known that $\mathcal{HL}(@)$ and $\mathcal{HL}(@, {}^{\smile})$ are decidable. Theorem 12 states that the constraint proof engine from Section 6 decides hybrid logic formulas without binders.

Theorem 12 *The constraint proof engine for $\mathcal{HL}(@, {}^{\smile}, \downarrow)$ decides satisfiability of $\mathcal{HL}(@, {}^{\smile})$ formulas.*

Proof. The result follows from a modal depth argument. $\mathcal{HL}(@, {}^{\smile})$ has a notion of finite degree ([9], Ch. 7); in particular, modal depth for its formulas can be defined as follows:

$$\begin{aligned} d(p) = d(c) = d(x) &= 0 \\ d(\varphi * \psi) &= \max(d(\varphi), d(\psi)), * \in \{\wedge, \vee, \rightarrow\} \\ d(\neg\varphi) &= d(\varphi) \\ d(M\varphi) &= d(\varphi) + 1, M \in \{[\,i\,], \langle\, i\,\rangle, [\,i\,]^{\smile}, \langle\, i\,\rangle^{\smile}\} \\ d(@m\varphi) &= d(\varphi) \end{aligned}$$

Every non-literal formula $\varphi \notin \{[\,i\,], [\,i\,]^{\smile}\}$ gets decomposed by the rule that applies to it. Every formula $\varphi \in \{[\,i\,], [\,i\,]^{\smile}\}$ generates a constraint. When a constraint $@m\Box\varphi$ is combined with an mRn relation, it generates a new formula $@n\varphi$ on the list of pending formulas, but $d(\varphi) = d(\Box\varphi) - 1$. Thus, the box formula $@m\Box\varphi$ gets replaced by a finite number of $@n\varphi$ formulas, and each of these is appended to the list of pending formulas, so fairness of the procedure is preserved. □

At first sight, there is something puzzling about the undecidability of $\mathcal{HL}(@, {}^{\smile}, \downarrow)$, especially since the so-called standard translation given in [1, 2] seems to suggest that hybrid sentences (hybrid formulas without unbound world variables) are in the two-variable bounded fragment of first order logic. In fact, the translation instruction has a flaw, as can be seen when we use it to translate the transitivity sentence $\downarrow u.\Box\Box \downarrow w.@u\Diamond w$. Carrying out the instruction starting out from $\mathrm{ST}_x(\downarrow u.\Box\Box \downarrow w.@u\Diamond w)$, we end up with an injunction to substitute x for u in $\forall y(Rxy \rightarrow \forall x(Ryx \rightarrow \exists y(Ruy \wedge y = x)))$, with capture of x by $\forall x$ as a result. The standard translation can be repaired by replacing the instructions for the modalities and the binders as follows:

$$\begin{aligned} \mathrm{ST}_x(\langle\, i\,\rangle\varphi) &:= \exists y(R_i xy \wedge \mathrm{ST}_y\varphi) \text{ with } y \text{ a fresh variable,} \\ \mathrm{ST}_x(\downarrow w.\varphi) &:= (\mathrm{ST}_y\varphi)^w_y \text{ with } y \text{ a fresh variable.} \end{aligned}$$

A fresh variable, of course, is a variable that has not been used before in the translation. This makes clear that the translation does not ensure a bound on the number of variables that are needed. E.g., the translation of the transitivity sentence needs as least three different variables.

We will now demonstrate that it is the presence of $\downarrow$ in the scope of box modalities that causes undecidability of $\mathcal{HL}(@, {}^{\smile}, \downarrow)$. Consider the version of $\mathcal{HL}(@, {}^{\smile}, \downarrow)$ without $[\,i\,], [\,i\,]^{\smile}, \rightarrow$ that can be got by paraphrasing d with the help of $[\,i\,]\varphi \leftrightarrow \neg\langle\, i\,\rangle\neg\varphi$, $[\,i\,]^{\smile}\varphi \leftrightarrow \neg\langle\, i\,\rangle^{\smile}\neg\varphi$, and $(\varphi \rightarrow \psi) \leftrightarrow \neg\varphi \vee \psi$. A subformula ψ of φ is *existential* in φ if every $\Diamond$ that outscopes ψ in φ is in the scope of an even number of negation operators. Thus, ψ is existential in $\Diamond\neg\psi$ and $@m\neg(\Diamond m \wedge \neg\Diamond\psi)$, but not in $\neg\Diamond\psi$. Call a formula φ *innocent* if every binder subformula $\downarrow x.\psi$ of φ is existential in φ.

Theorem 13 *The innocent fragment of $\mathcal{HL}(@^{\breve{}}, \downarrow)$ is decidable.*

Proof. Consider the following translation from (located formulas of) $\mathcal{HL}(@, ^{\breve{}}, \downarrow)$ to $\mathcal{HL}(@, ^{\breve{}})$.

$$\begin{array}{rcl}
(@mp)^T & := & @mp \\
(@mn)^T & := & @mn \\
(@m\neg\varphi)^T & := & \neg(@m\varphi)^T \\
(@m\varphi \wedge \psi)^T & := & (@m\varphi)^T \wedge (@m\psi)^T \\
(@m\varphi \vee \psi)^T & := & (@m\varphi)^T \vee (@m\psi)^T \\
(@m@n\varphi)^T & := & (@n\varphi)^T \\
(@m\langle i \rangle\varphi)^T & := & @m\langle i \rangle(n \wedge (@n\varphi)^T),\ n \text{ a fresh nominal} \\
(@m\langle i \rangle\breve{}\varphi)^T & := & @m\langle i \rangle\breve{}(n \wedge (@n\varphi)^T),\ n \text{ a fresh nominal} \\
(@m \downarrow x.\varphi)^T & := & (@m\varphi[x := m])^T
\end{array}$$

Check by induction on formula structure that this translation is truth-preserving for all formulas in the innocent fragment. □

Another way of seeing this is by observing that the tableaux for φ and φ^T are essentially the same, which leads immediately to:

Theorem 14 *The constraint proof engine for $\mathcal{HL}(@, ^{\breve{}}, \downarrow)$ decides the innocent fragment.*

Consider the following fragment of *existential* formulas of $\mathcal{HL}(@, ^{\breve{}})$:

$$\psi \quad ::= \quad n \mid \neg n \mid \neg\langle i \rangle n \mid \neg\langle i \rangle\breve{}n \mid \psi \wedge \psi' \mid \psi \vee \psi' \mid \langle i \rangle\psi \mid \langle i \rangle\breve{}\psi$$

Use this to define $\mathcal{HL}(@, ^{\breve{}}, \downarrow^{\exists})$, by changing the definition of binder formulas from $\downarrow x.\varphi$ to $\downarrow x.\psi$. In other words, binding is only allowed over existential formulas. If, moreover, we only allow one binding variable w, we get [17]:

Theorem 15 (M. Marx) *The language $\mathcal{HL}(@, ^{\breve{}}, (\downarrow w)^{\exists})$ is decidable in EXPTIME.*

Proof. Using a standard translation we can map this language to the universal guarded fragment (where universal quantifiers are guarded, but existential quantifiers need not be) in three variables x, y, w. By a remark in [14], existential guards can be dispensed with, for the sentence $(\forall \mathbf{x}.\alpha)\exists \mathbf{y}\varphi(\mathbf{xy})$, with $\exists \mathbf{y}$ unguarded, is satisfiable if and only if the properly guarded sentence $(\forall \mathbf{x}.\alpha)\exists \mathbf{y} R\mathbf{xy} \wedge (\forall \mathbf{x}.R\mathbf{xy})\varphi(\mathbf{xy})$, with R a new relation symbol of the appropriate arity, is satisfiable. Satisfiability of the universal guarded fragment in a bounded number of variables is shown to be decidable in EXPTIME in [14]. □

If we allow an unrestricted number of binding variables $x_1, \ldots$ the fragment is still decidable, but since there is no bound now on the number of variables employed in the translation, the complexity is in 2EXPTIME [14].

Theorem 16 *The constraint proof engine for $\mathcal{HL}(@, ^{\breve{}}, \downarrow)$ decides satisfiability of $\mathcal{HL}(@, ^{\breve{}}, \downarrow^{\exists})$ formulas.*

Proof. Suppose the proof engine is invoked to check satisfiability of φ. We only need to be concerned about formulas of the forms $[\,i\,] \downarrow x.\psi$, $\langle\, i\,\rangle\neg \downarrow x.\psi$, $[\,i\,]^{\smile} \downarrow x.\psi$ $\langle\, i\,\rangle^{\smile}\neg \downarrow x.\psi$ that turn up along a tableau branch during the processing of φ, for these are the formulas that cause binders to appear in the $[\,i\,]$ and $[\,i\,]^{\smile}$ constraints. Suppose $@m\Box \downarrow x.\psi$ is in the box constraint store, and gets combined with an mRn accessibility along the branch. Then $@n \downarrow x.\psi$ will get appended to the pending formula list. When $@n \downarrow x.\psi$ gets decomposed, it gets replaced by ψ^x_n, and further decomposition will not affect the constraint stores because ψ is existential. □

13 Related Work

Comparison with other tableau systems Tableau calculi for hybrid logic are still in their infancy. A prefixed tableau calculus along the lines of the prefixed tableau style theorem proving of [13] is given in [20]. This does not yet make full use of the possibility to let the role of prefixes be played by nominals. In [6] it is pointed out that nominals can play the role of labels in labelled deduction style theorem proving, and a tableau calculus is proposed that handles equality reasoning on nominals by means of rewrite rules that express reflexivity, symmetry and transitivity of equality, and that allows substitution of equal nominals. However, since this rewrite system is not normalizing, equality reasoning in the resulting calculus is awkward, and proof engines based on it will spend too much effort on pointless rewrite steps for equality [8].

Comparison with resolution for hybrid logic The resolution approach to modal logic dates back to at least [18, 12]. A prefixed resolution calculus for description and hybrid logic was proposed in [3], and implemented in [4]. Detailed efficiency comparison with this is work in progress.

Acknowledgement Many thanks to the Dynamo team (Wim Berkelmans, Balder ten Cate, Juan Heguiabehere, Breanndán Ó Nualláin) and to Carlos Areces, Patrick Blackburn and Maarten Marx, for useful comments and fruitful discussion. This work was carried out as part of the INRIA funded partnership between CALG (Computational and Applied Logic Group, University of Amsterdam) and LED (Langue et Dialogue, LORIA, Nancy).

References

[1] Areces, C. *Logic Engineering: The Case of Description and Hybrid Logics.* PhD thesis, ILLC, Amsterdam, 2000.

[2] Areces, C., Blackburn, P., and Marx, M. Hybrid logics: Characterization, interpolation and complexity. *Journal of Symbolic Logic* (2001).

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

[4] Areces, C., and Heguiabehere, J. Hylores: Direct resolution for hybrid logics. In *Proceedings of Methods for Modalities 2* (Amsterdam, The Netherlands, November 2001), C. Areces and M. de Rijke, Eds.

[5] Benthem, J. v. Partiality and nonmonotonicity in classical logic. *Logique et Analyse 29* (1986).

[6] Blackburn, P. Internalizing labelled deduction. *Journal of Logic and Computation 10* (2000), 136–168.

[7] Blackburn, P. Representation, reasoning, and relational structures: a hybrid logic manifesto. *Logic Journal of the IGPL 8* (2000), 339–365.

[8] Blackburn, P., Burchard, A., and Walter, S. Hydra: a tableaux-based prover for basic hybrid logic. In *Proceedings of Methods for Modalities 2* (Amsterdam, The Netherlands, November 2001), C. Areces and M. de Rijke, Eds.

[9] Blackburn, P., de Rijke, M., and Venema, Y. *Modal Logic*. Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 2001.

[10] Blackburn, P., and Seligman, J. What are hybrid languages? In *Advances in Modal Logic*, M. Kracht, M. de Rijke, H. Wansing, and M. Zakharyaschev, Eds., vol. 1. CSLI Publications, Stanford University, 1998, pp. 41–62.

[11] Eijck, J. v. HyLoTab — Tableau-based theorem proving for hybrid logics. Manuscript, CWI, Amsterdam, 2002.

[12] Enjalbert, P., and Fariñas del Cerro, F. Modal resolution in clausal form. *Theoretical Computer Science 14* (1989), 1–33.

[13] Fitting, M. *Proof Methods for Modal and Intuitionistic Logics*. Reidel, Dordrecht, 1983.

[14] Grädel, E. On the restraining power of guards. *Journal of Symbolic Logic 64*, 4 (1999), 1719–1742.

[15] Jones, S. P., Hughes, J., et al. Report on the programming language Haskell 98. Available from the Haskell homepage: `http://www.haskell.org`, 1999.

[16] Knuth, D. *Literate Programming*. CSLI Lecture Notes, no. 27. CSLI, Stanford, 1992.

[17] Marx, M. Narcissists, stepmothers and spies. Manuscript, ILLC, 2002.

[18] Ohlbach, H. A resolution calculus for modal logics. In *CADE'88* (1988), pp. 500–516.

[19] Smullyan, R. *First-order logic*. Springer, Berlin, 1968.

[20] Tzakova, M. Tableau calculi for hybrid logics. In *Proceedings of the International Conference TABLEAUX'99 - Analytic Tableaux and Related Methods, June 7-11, 1999, Saratoga Springs, NY USA* (1999), N. V. Murray, Ed., vol. 1617 of *LNAI*, Springer.