Terminating Tableau Systems for Modal Logic with Equality

Mark Kaminski and Gert Smolka
Saarland University

April 14, 2008

The paper presents two terminating tableau systems for hybrid logic with the difference modality. Both systems are based on an abstract treatment of equality. They expand formulas with respect to a congruence closure that is not represented explicitly. The first system employs pattern-based blocking. The second system employs chain-based blocking and covers converse modalities. Both systems can handle transitive relations.

1 Introduction

There are two established ways to arrive at modal logic with equality [5, 10, 1]. One approach employs the difference modality D, defined such that a property Ds holds for a state x if there exists a different state y such that s holds for y. The other approach, known as hybrid logic, employs nominals, which are primitive properties holding for exactly one state x. Without further extensions, the difference modality is more powerful than nominals. Once nominals are accompanied by the global modality E (Es holds for x if there exists a y for which s holds), both approaches have the same expressivity [13].

Tableau systems for modal logic were first devised by Kripke [23, 24]. Terminating tableau systems yield decision procedures for satisfiability. Once more, it was Kripke [24] who devised the first terminating tableau system for S4 (modal logic with a reflexive and transitive relation). Work of Horrocks et al [17] on description logic (modal logic adapted to knowledge representation) shows that tableau-based decision procedures can be the most efficient choice for practical applications.

Terminating tableau systems for modal logic with multiple, possibly transitive relations, and global and converse modalities are known [14, 18]. The purpose of this paper is to extend these results to modal logic with nominals and the difference modality. This turns out to be a non-trivial task.

Tableau systems for modal logic with the difference modality are not well-understood so far. In a recent handbook chapter on modal proof theory [12], an unsound tableau calculus for basic modal logic with the difference modality is given. Balbiani and Demri [2] give a sound and complete tableau system for this logic, but their system does not terminate on all inputs (although claimed). Bolander and Braüner [8] devise a terminating tableau system for hybrid logic with the global modality. Building on this work, Bolander and Blackburn [7] give a terminating tableau system for hybrid logic with global and converse modalities, but point out that the difference modality remains a challenge.

To avoid the syntactic limitations of modal logic, in particular as it comes to equality, we start from nominal predicate logic PLN (classical first-order logic without functions) and let the individuals act as states. By adding modal quantifiers [15] to PLN we arrive at an extension we call PLM. For instance, the modally quantified formula $\Diamond rsx$ holds if there exists an r-successor of x for which the property s holds. PLM is a translational extension of PLN, that is, formulas with modal quantifiers can be translated to equivalent formulas without modal quantifiers. For instance, $\Diamond rsx \doteq \exists y.\ rxy \wedge sy$. Full PLM is undecidable, but it is easy to identify a fragment that corresponds to modal logic with equality.

We devise several tableau systems for PLM. In contrast to existing approaches [8, 7], our systems handle equality abstractly and do not commit to a particular representation of the congruence closure induced by the equations. We develop terminating tableau systems for diamond-free formulas (easy), for converse-free formulas, and for formulas with all the modal features mentioned so far. Due to our approach, the treatment of the difference modality is relatively straightforward. The real difficulty are diamond formulas. For the most general system, we use chain-based blocking, which dates back to Kripke [24] and was refined by Horrocks and Sattler [18] for converse modalities (dynamic blocking). For the converse-free system we use a new technique called pattern-based blocking. Pattern-based blocking is simpler than chain-based blocking and seems promising for efficient implementation.

With our abstract treatment of equality, the termination proofs are not difficult. More difficult are the model existence proofs. To avoid large and opaque proofs, we proceed in stages. First we formulate an evidence property (analogous to Hintikka's model sets [16]) and prove the corresponding model existence theorem. Evidence yields a first tableau system. Next we formulate a weaker property, called quasi-evidence, for which we show model existence by reduction to evidence. The important point about quasi-evidence is that it doesn't require the presence of edges rxy (accessibility formulas) for diamond formulas. Quasi-evidence has not been used before. Quasi-evidence yields a tableau system that can prove much more formulas satisfiable than the system based on evidence. To arrive at terminating systems, it now suffices to control diamond expansion with either the pattern- or the chain-based technique.

Preceding this paper, we have written three preliminary papers [21, 20, 22]. The different equality rules used in these papers document our struggle for the right treatment of equality. Pattern-based blocking first appears in [21]. Our first terminating system for hybrid logic with difference appears in [20]. Finally, [22] presents a terminating system for hybrid logic with difference and converse.

The paper is organized as follows. We start with PLM and develop the notion of nominal congruence that underlies our treatment of equality. We then define syntactic satisfiability and evidence and prove the corresponding model existence theorem. Next we introduce abstract tableau systems as a basis for our concrete systems. We then obtain a first concrete system, $\mathcal{T}$, as an operational reformulation of the notion of PLM-evidence. We show that $\mathcal{T}$ terminates for $\Diamond$-free formulas if all $\exists$-subformulas are closed. We then define regular formulas, which yield a fragment of PLM that subsumes modal logic with equality. Next we define quasi-evidence and prove the corresponding model existence theorem. Quasi-evidence yields a system $\mathcal{T}^q$ that can verify all satisfiable regular clauses. Based on quasi-evidence and $\mathcal{T}^q$, we then present two terminating systems $\mathcal{T}^p$ and $\mathcal{T}^c$, which control diamond expansion by pattern-based and chain-based blocking, respectively. We then add the difference modalities to PLM and show that our results carry over. Finally, we add transitive relations and rework the notion of quasi-evidence so that our results carry over.

2 PLM

We start with first-order logic with equality and without function symbols. We call this logic **PLN** for **nominal predicate logic**. It is well-known that modal logic is a translational fragment of PLN. For this purpose, the individuals of PLN are understood as states and the relations between states are modeled as binary predicates. Following [15], we extend PLN with modal quantifiers:

- $\Diamond rsx$: There exists an r-successor of x that satisfies s.
- $\Box rsx$: Every r-successor of x satisfies s.

An **r-successor of x** is a y such that rxy holds. Symmetrically, an **r-predecessor of x** is a y such that ryx holds. We also provide converse modal quantification:

- $\Diamond r^{-}sx$: There exists an r-predecessor of x that satisfies s.
- $\Box r^{-}sx$: Every r-predecessor of x satisfies s.

s	$::=$	$a \mid \neg a \mid s \wedge s \mid s \vee s \mid \exists u \mid \forall u \mid tx$	**formula**
a	$::=$	$px\ldots x \mid x \doteq x$	**atomic formula**
u	$::=$	$\lambda x.s$	**lambda property**
t	$::=$	$u \mid \mu t$	**property**
μ	$::=$	$\Diamond\rho \mid \Box\rho$	**modality**
ρ	$::=$	$r \mid r^{\smallsmile}$	
x	$:$	I	**individual name**
p, r	$:$	$I\ldots IB$ where $r : IIB$	**predicate name**

Figure 1: Syntax of PLM

We model converse modal quantification as modal quantification with respect to the inverse $r^{\smallsmile}$ of a relation r.

We call the extension of PLN with modal quantification **PLM**. We employ a simply typed syntax for PLM where the quantifiers appear as higher-order constants and variable binding is done uniformly through lambda-abstractions. PLM has two base types B (bool) and I (individuals). Terms of type B are called **formulas**. We write functional types $\sigma_1 \to \sigma_2$ as $\sigma_1\sigma_2$ and omit parentheses according to $\sigma_1\sigma_2\sigma_3 \rightsquigarrow \sigma_1(\sigma_2\sigma_3)$. We use the logical constants

$$\neg : BB \qquad \doteq : IIB$$
$$\wedge, \vee : BBB \qquad \exists, \forall : (IB)B$$

with their usual meaning. For modal quantification we provide three additional constants, called **modal constants**:

$\Diamond, \Box :$	$(IIB)(IB)IB$	**diamond, box**
$^{\smallsmile} :$	$(IIB)IIB$	**converse**

We use **individual names** x of type I and **predicate names** p of type $I\ldots IB$. Predicate names may be nullary, that is, have the type B. We reserve the letter r for predicate names of type IIB. We write applicative terms as t_1t_2 and omit parentheses according to $t_1t_2t_3 \rightsquigarrow (t_1t_2)t_3$. Moreover, applications of $\wedge$, $\vee$, and $\doteq$ are written in infix notation, and applications of $^{\smallsmile}$ are written in postfix notation (i.e., $r^{\smallsmile}$ instead of $^{\smallsmile}r$). Based on these assumptions, Figure 1 defines the syntax of PLM.

We write quantified formulas of the form $\exists(\lambda x.s)$ and $\forall(\lambda x.s)$ as $\exists x.s$ and $\forall x.s$. Formulas of the form $x \doteq y$ are called **equations**. We write $x \neq y$ for a negated equation $\neg(x \doteq y)$. Formulas of the form rxy are called **edges**.

Equations in PLM must always be formed with constituents of type I. However, for semantic considerations it is useful to use equations whose constituents have other types. Using this device, we define the semantics of the modal constants in terms of the logical constants:

$$\begin{aligned}
\Diamond &\doteq \lambda rpx.\, \exists y.\, rxy \wedge py \\
\Box &\doteq \lambda rpx.\, \forall y.\, \neg rxy \vee py \\
{}^{-} &\doteq \lambda rxy.\, ryx
\end{aligned}$$

By means of these equations, every formula of PLM can be translated to a formula of PLN using β-reduction.

We shall use the letters s, t and u for terms in general, that is, ignore the role these letters play in the grammar defining the syntax of PLM (shown in Figure 1).

PLM provides only formulas in negation normal form since this simplifies the technical development. General formulas can be translated to negation normal form by means of the following equations:

$$\begin{aligned}
\neg\neg s &\doteq s & & \\
\neg(s \wedge t) &\doteq \neg s \vee \neg t & \neg(s \vee t) &\doteq \neg s \wedge \neg t \\
\neg\exists x.s &\doteq \forall x.\neg s & \neg\forall x.s &\doteq \exists x.\neg s \\
\neg\Diamond rsx &\doteq \Box r(\lambda y.\neg sy)x & \neg\Box rsx &\doteq \Diamond r(\lambda y.\neg sy)x
\end{aligned}$$

The formulas of modal logic appear in PLM as properties, which are terms of type IB. Our syntax for properties is expressive but inconvenient. We have kept the syntax for properties minimal to not burden the technical development with unnecessary syntactic forms. Here are examples showing how the usual modal notation translates to PLM:

$$\begin{aligned}
s \wedge t &\rightsquigarrow \lambda x.\, sx \wedge tx \\
\Diamond r \neg s &\rightsquigarrow \Diamond r(\lambda x.\neg sx) \\
\Box r(a \wedge s) &\rightsquigarrow \Box r(\lambda x.\, a \doteq x \wedge sx) \\
s \wedge @_a t &\rightsquigarrow \lambda x.\, sx \wedge ta \\
\downarrow x.s &\rightsquigarrow \lambda x.sx
\end{aligned}$$

From what we have said about PLM it is clear that modal logic can be formalized in simple type theory. One benefit of this approach is that theorems about modal logic can be proven in type theory, possibly using automated provers, as done by Benzmüller and Paulson [3].

To simplify substitution (cf. Proposition 3.1 (3)), we distinguish between two kinds of individual names, called **variables** and **nominals**. The individual names bound by lambda properties must always be variables. A formula is **closed** if it has no free variable. A formula is **open** if it is not closed. A **PLM-formula** is a closed formula obtained according to the syntax in Figure 1. We will use the letters x, y, z for individual names (i.e., variables or nominals) and reserve a, b, c for nominals. The letter F will always denote a set of PLM-formulas. We write $\mathcal{N}s$ and $\mathcal{N}F$ for the set of all nominals that occur in a term s or in a set of formulas F. We use Nom to denote the set of all nominals.

3 Nominal Congruence

Given a set F of PLM-formulas, the equations in F yield an equivalence relation on the set of nominals called nominal congruence. If we close F under nominal congruence, we obtain a set of PLM-formulas called the congruence closure of F. For instance, if $F = \{px,\ x \doteq y\}$, then the congruence closure of F is $F \cup \{py,\ x \doteq x,\ y \doteq x,\ y \doteq y\}$. Our tableau systems will treat equality through congruence closures.

Nominal congruences will be represented through normalizers. A **normalizer** is an idempotent function $\varphi \in \mathit{Nom} \to \mathit{Nom}$ (i.e., $\varphi(\varphi x) = \varphi x$ for all $x \in \mathit{Nom}$). Let φ be a normalizer. We define (slight abuse of notation):

- $\mathit{Dom}\,\varphi := \{\, x \in \mathit{Nom} \mid \varphi x \neq x \,\}$
- $\mathit{Ran}\,\varphi := \{\, \varphi x \mid x \in \mathit{Dom}\,\varphi \,\}$
- φs is the term obtained from the term s by applying φ as a substitution.
- $\varphi F := \{\, \varphi s \mid s \in F \,\}$.

Proposition 3.1 Let φ is a normalizer and s be a term. Then:

1. $\mathcal{N}(\varphi s) \cap \mathit{Dom}\,\varphi = \emptyset$
2. $\varphi(\varphi s) = \varphi s$
3. $\varphi(\lambda x.s) = \lambda x.\varphi s$
4. $\varphi(s^x_y) = (\varphi s)^x_{\varphi y}$ if x is a variable

Note that (2) holds since variables and nominals are disjoint, λ binds variables, and normalizers replace nominals by nominals.

Let $\sim$ be an equivalence relation on Nom. A **normalizer for** $\sim$ is a normalizer φ such that $\forall x, y \in \mathit{Nom}$: $x \sim y \iff \varphi x = \varphi y$. Note that the normalizers for an equivalence relation can be obtained by choosing a representative for every class and mapping all members of a class to the chosen representative.

Lemma 3.2 Let $\sim_1$ and $\sim_2$ be equivalence relations on Nom and φ_1 and φ_2 be normalizers for $\sim_1$ and $\sim_2$, respectively. Then $\varphi_2(\varphi_1 s) = \varphi_2 s$ for all terms s if $\sim_1 \subseteq \sim_2$.

Proof By induction on the size of terms. ■

We define:

- $\sim_F$ is the least equivalence relation on *Nom* such that $\forall (x \doteq y) \in F\colon\ x \sim_F y$.
- **φ is a normalizer for F** if φ is a normalizer for $\sim_F$.
- $s \approx_F t \;:\iff\; \exists$ normalizer φ for $F\colon\ \varphi s = \varphi t$ **nominal congruence**
- $\tilde{F} \;:=\; \{\, s \mid \exists t \in F\colon\ s \approx_F t \,\}$ **nominal congruence closure of F**

Note that $\approx_F$ is an equivalence relation. An equation is **trivial** if it has the form $x \doteq x$. A set F is **basic** if every equation $s \in F$ is trivial. If F is basic, then $\sim_F$ and $\approx_F$ are identity relations, $\tilde{F} = F$, and the unique normalizer of F is the identity function.

Proposition 3.3 Let φ be a normalizer for F. Then:

1. $\tilde{F} = \{\, s \mid \exists t \in F\colon\ \varphi s = \varphi t \,\}$
2. $s \in \tilde{F} \iff \varphi s \in \varphi F$
3. $\varphi \tilde{F} = \varphi F$

Proof

1. Clearly $\{\, s \mid \exists t \in F\colon\ \varphi s = \varphi t \,\} \subseteq \tilde{F}$, so it suffices to show the converse inclusion. Let $s \in \tilde{F}$. Then $\exists t \in F\, \exists$ normalizer φ' for $F\colon\ \varphi' s = \varphi' t$. Hence, by Lemma 3.2, $\varphi s = \varphi(\varphi' s) = \varphi(\varphi' t) = \varphi t$. The claim follows.
2. $s \in \tilde{F} \overset{(1)}{\iff} \exists t \in F\colon\ \varphi s = \varphi t \iff \varphi s \in \varphi F$
3. $s \in \varphi \tilde{F} \iff \exists t \in \tilde{F}\colon\ s = \varphi t \overset{(1)}{\iff} \exists t' \in F\colon\ s = \varphi(\varphi t') = \varphi t' \iff s \in \varphi F$ ■

Proposition 3.4 Let φ be a normalizer for F. Then:

1. F basic $\iff \forall s\colon\ \varphi s = s$
2. φF is basic
3. $\sim_F \subseteq \approx_F$
4. $s \approx_F t \iff \varphi s = \varphi t$

Proof Claim (1) holds since F is basic if and only if $\sim_F$ is the identity relation. Claim (2) follows by (1) and Proposition 3.1 (2). Claim (3) is immediate, as is the direction from right to left in Claim (4). The other direction follows by Lemma 3.2. ■

Proposition 3.5 Let $x \notin \mathcal{N}F$. Then $x \sim_F y$ iff $x = y$.

Proof Let $x \notin \mathcal{N}F$. Clearly $x = y$ implies $x \sim_F y$. So assume for contradiction that $x \sim_F y$ for some $y \neq x$. Let $\sim \,:= \{(x,x)\} \cup \{\, (x',y') \in \sim_F \mid x' \neq x \wedge y' \neq x \,\}$. Clearly $\sim \subsetneq \sim_F$. It can be shown that $\sim$ is an equivalence relation. Moreover, as $x \notin \mathcal{N}F$, $x' \sim y'$ holds for every equation $x' \doteq y' \in F$, which contradicts the definition of $\sim_F$. ∎

We define Id $:= \{\, x \doteq x \mid x \in \mathit{Nom} \,\}$ (the set of all trivial equations).

Proposition 3.6

1. $F \subseteq \tilde{F}$
2. $F_1 \subseteq F_2 \implies \sim_{F_1} \subseteq \sim_{F_2}$
3. $F_1 \subseteq F_2 \implies \tilde{F_1} \subseteq \tilde{F_2}$
4. $\mathcal{N}F = \mathcal{N}\tilde{F}$
5. F finite $\iff$ $\tilde{F}$ finite
6. $x \sim_F y \iff (x \doteq y) \in \tilde{F} \cup \mathrm{Id}$
7. $\sim_F \, = \, \sim_{\tilde{F}}$

Proof

1. Follows by the reflexivity of $\approx_F$.
2. By contradiction. Assume $x \sim_{F_1} y$ and $x \not\sim_{F_2} y$. Let $\sim \,:= \sim_{F_1} \cap \sim_{F_2}$. It can be shown that $\sim$ is an equivalence relation such that $\forall (x \doteq y) \in F_1 \colon x \sim y$. Since $\sim \subsetneq \sim_{F_1}$, this is a contradiction to the minimality of $\sim_{F_1}$.
3. Follows by Claim (2) and Lemma 3.2.
4. Clearly $\mathcal{N}F \subseteq \mathcal{N}\tilde{F}$, so it suffices to show the converse inclusion. Let $x \in \mathcal{N}\tilde{F}$, meaning there is some formula $s \in \tilde{F}$ such that $x \in \mathcal{N}s$. Then there is some formula $t \in F$ such that $s \approx_F t$. Consequently, there is some variable $y \in \mathcal{N}t$ such that $x \sim_F y$. Since $y \in \mathcal{N}F$, Proposition 3.5 implies $x \in \mathcal{N}F$.
5. Follows by Claim (4) and the fact that every two terms s and t such that $s \approx_F t$ have the same size.
6. The direction from left to right is proven similarly to Proposition 3.5. As for the other direction, it suffices to show that $(x \doteq y) \in \tilde{F}$ implies $x \sim_F y$. So, let $(x \doteq y) \in \tilde{F}$ and φ be a normalizer for F. By Proposition 3.3 (2), $\varphi x \doteq \varphi y \in \varphi F$. Then there are nominals x' and y' such that $x' \doteq y' \in F$, $\varphi x' = \varphi x$ and $\varphi y' = \varphi y$. The claim follows.
7. $x \sim_{\tilde{F}} y \overset{(6)}{\iff} (x \doteq y) \in \tilde{\tilde{F}} \cup \mathrm{Id} \iff (x \doteq y) \in \tilde{F} \cup \mathrm{Id} \overset{(6)}{\iff} x \sim_F y$ ∎

Proposition 3.7 Let φ be a normalizer for F. Then:

1. $\varphi F \subseteq \tilde{F}$

2. $\mathcal{N}(\varphi F) = \varphi(\mathcal{N}F) = \mathcal{N}F - \mathit{Dom}\,\varphi$

Proof

1. Let $s \in F$. It suffices to show that $\varphi s \in \tilde{F}$. This is the case because, by Proposition 3.4 (4) and 3.1 (2), $\varphi s \approx_F s$.
2. The first equality states a fundamental property of substitution. The second equality follows by Proposition 3.1 (1) and the observation that $x \notin \mathit{Dom}\,\varphi$ implies $x \in \mathcal{N}(\varphi F) \iff x \in \mathcal{N}F$. ■

A set F is **equationally expanded** if $F = \tilde{F}$. Note that basic sets and nominal congruence closures are always equationally expanded.

4 Syntactic Satisfiability

An **interpretation** $\mathcal{I}$ interprets B as the set $\{0, 1\}$, I as a non-empty set, the logical constants $\doteq$, $\neg$, $\wedge$, $\vee$, $\forall$, $\exists$ as usual (1 takes the role of true), the modal constants according to their defining equations, and the individual and predicate names according to their types. Given an interpretation $\mathcal{I}$ and a term s, we write $\mathcal{I}s$ for the object $\mathcal{I}$ assigns to s. An interpretation $\mathcal{I}$ **satisfies a formula** s if $\mathcal{I}s = 1$. An interpretation **satisfies a set of formulas** F if it satisfies every formula $s \in F$. An interpretation is a **model of** F if it satisfies F. A set F is **satisfiable** if it has a model, and **unsatisfiable** otherwise.

An interpretation $\mathcal{I}$ is **syntactic for** F if $\mathcal{I}I = \mathcal{I}(\mathcal{N}F)$ (i.e, $\mathcal{I}I = \{\,\mathcal{I}x \mid x \in \mathcal{N}F\,\}$) and for every atomic formula a the following holds: if $\mathcal{N}a \subseteq \mathcal{N}F$, then $\mathcal{I}$ satisfies a if and only if $a \in \tilde{F} \cup \mathrm{Id}$. An interpretation is a **syntactic model of** F if it satisfies F and is syntactic for F. A set F is **syntactically satisfiable** if it has a syntactic model. A set F is **finitely satisfiable** if F has a model $\mathcal{I}$ such that $\mathcal{I}I$ is a finite set.

Proposition 4.1

1. An interpretation $\mathcal{I}$ is syntactic for a set F if and only if it is syntactic for $\tilde{F}$.
2. If F is syntactically satisfiable and finite, then F is finitely satisfiable.
3. If $\mathcal{I}$ is syntactic for F and $x, y \in \mathcal{N}F$, then $\mathcal{I}$ satisfies $x \doteq y$ if and only if $x \sim_F y$.

Proof The last claim follows with Proposition 3.6 (6). ■

Let $\mathcal{I}$ be an interpretation and φ be a normalizer. We define $\mathcal{I}_\varphi$ as the interpretation that is obtained from $\mathcal{I}$ by (possibly) changing $\mathcal{I}$ on the nominals in $\mathit{Dom}\,\varphi$ such that $\mathcal{I}_\varphi x = \mathcal{I}(\varphi x)$.

Proposition 4.2 Let φ be a normalizer for F and $\mathcal{I}$ be an interpretation that is syntactic for φF. Then $\mathcal{I}_\varphi$ is syntactic for F.

Proof The first condition of being syntactic is obviously satisfied. To show the second condition, let a be an atomic formula such that $\mathcal{N}a \subseteq \mathcal{N}F$. Then the following statements are equivalent:

$\mathcal{I}_\varphi$ satisfies a	
$\mathcal{I}$ satisfies φa	substitution lemma
$\varphi a \in \varphi F \cup \mathrm{Id}$	$\mathcal{I}$ syntactic for φF
$a \in \tilde{F} \cup \mathrm{Id}$	Proposition 3.3 ■

5 Evidence

Tableau systems rest on a semantically motivated notion of evidence first described by Hintikka [16]. To define evidence for PLM, we need some technical definitions. A **literal formula** is an atomic formula or a negated atomic formula. We write s^x_y for the term obtained from s by capture-free replacement of the free occurrences of x by y. If y is a nominal, s^x_y is obtained without renaming of bound variables. By abuse of notation we mean by ρxy the formula rxy if $\rho = r$ and ryx if $\rho = r^-$. We say that a formula is **evident in F** if it is not literal and if it satisfies the **evidence condition** corresponding to its form:

$$\begin{array}{rll}
s_1 \wedge s_2 & \text{evident in } F \text{ if} & s_1 \in F \wedge s_2 \in F \\
s_1 \vee s_2 & \text{evident in } F \text{ if} & s_1 \in F \vee s_2 \in F \\
\exists s & \text{evident in } F \text{ if} & \exists x\colon\ sx \in F \\
\forall s & \text{evident in } F \text{ if} & \forall x \in \mathcal{N}F\colon\ sx \in F \\
(\lambda x.s)y & \text{evident in } F \text{ if} & s^x_y \in F \\
\Diamond \rho s x & \text{evident in } F \text{ if} & \exists y\colon\ \rho xy \in F \wedge sy \in F \\
\Box \rho s x & \text{evident in } F \text{ if} & \forall y\colon\ \rho xy \in F \Longrightarrow sy \in F
\end{array}$$

Lemma 5.1 (Soundness) Let $\mathcal{I}$ be a syntactic model of F and $F = \tilde{F}$. Then $\mathcal{I}$ satisfies s if s is evident in F and $\mathcal{N}s \subseteq \mathcal{N}F$.

Proof Since $\mathcal{I}$ is syntactic for F we have $\mathcal{I}I = \mathcal{I}(\mathcal{N}F)$. We refer to this property as (S). Let s be evident in F and $\mathcal{N}s \subseteq \mathcal{N}F$. We show that $\mathcal{I}$ satisfies s. If s is a conjunction, disjunction, $\exists$-, λ-, or $\Diamond$-formula, the claim is obvious. For $\forall$-formulas the claim follows with (S). If $s = \Box \rho t x$, because of (S) it suffices to show that $\mathcal{I}$ satisfies $\rho xy \to ty$ if $y \in \mathcal{N}F$. Let $\mathcal{I}$ satisfy ρxy and $y \in \mathcal{N}F$. We show that $\mathcal{I}$ satisfies ty. We have $x \in \mathcal{N}F$ since $\mathcal{N}s \subseteq \mathcal{N}F$. Hence $\rho xy \in \tilde{F} = F$ since $\mathcal{I}$ is syntactic for F. Thus $ty \in F$ since s is evident in F. Hence $\mathcal{I}$ satisfies s since $\mathcal{I}$ satisfies F. ■

Proposition 5.2 (Stability) Let φ be a normalizer for F. Then a formula s is evident in $\tilde{F}$ if and only if φs is evident in φF.

Proof Let s be a non-literal formula. Case analysis.

Case $s = s_1 \wedge s_2$. We have to show that $s_1, s_2 \in \tilde{F} \iff \varphi s_1, \varphi s_2 \in \varphi F$, which holds by Proposition 3.3 (2).

Case $s = (\lambda x.t)y$. We have to show that $t^x_y \in \tilde{F} \iff (\varphi t)^x_{\varphi y} \in \varphi F$. This holds by Proposition 3.1 (4) and Proposition 3.3 (2).

Case $s = \forall t$. Let $x \in \mathcal{N}\tilde{F}$. We have to show that $tx \in \tilde{F} \iff (\varphi t)(\varphi x) \in \varphi F$, which holds by Proposition 3.3 (2).

Case $s = \Diamond\rho tx$. We have to show that $(\exists y\colon\ \rho xy, sy \in \tilde{F}) \iff (\exists y\colon\ \rho(\varphi x)y, (\varphi s)y \in \tilde{F})$, which holds by Proposition 3.3 (2) since $\mathcal{N}(\varphi F) = \varphi(\tilde{\mathcal{N}}F)$.

The remaining cases are similar. ■

Proposition 5.3 (Compatibility) Let $s \approx_F s'$. Then s is evident in $\tilde{F}$ if and only if s' is evident in $\tilde{F}$.

Proof Let φ be a normalizer for F. Then $\varphi s = \varphi s'$. Hence the claim follows with Proposition 5.2. ■

A set F is **locally consistent** if there is no negated formula $\neg s \in F$ such that $s \in F \cup \text{Id}$.

Lemma 5.4 (Local Consistency) $\tilde{F}$ is locally consistent if and only if there is no formula s such that $\neg s \in F$ and $s \in \tilde{F} \cup \text{Id}$.

Proof The direction from left to right is obvious. We show the other direction by contraposition. Let $\tilde{F}$ be locally inconsistent. Then there exists a formula $\neg t \in \tilde{F}$ such that $t \in \tilde{F} \cup \text{Id}$. Let φ be a normalizer for F. Then there exists a formula $\neg s \in F$ such that $\varphi s = \varphi t$. We show $s \in \tilde{F} \cup \text{Id}$ by contradiction. Suppose, $s \notin \tilde{F}$ and $s \notin \text{Id}$. Then $t \notin \tilde{F}$ and hence $t \in \text{Id}$. Thus there exist two nominals $x \neq y$ such that $s = (x \doteq y)$ and $\varphi x = \varphi y$. Hence $s \in \tilde{F}$ by Proposition 3.6 (6), which contradicts our assumption. ■

A set F is **evident** if it is equationally expanded and locally consistent, contains a least one nominal, and every non-literal formula $s \in F$ is evident in F. For instance, the set $\{\Diamond r^{-}pa, rba, pb\}$ is evident and the set $\{\Box r^{-}pa, rba\}$ is not evident. Hintikka [16] considers evident sets for pure predicate logic (no functions, no equality) and calls them model sets. Some authors, e.g., [14, 18], call evident sets tableaux. We see it as an advantage of PLM that evident sets can be represented as sets of PLM-formulas. If native modal syntax is used, additional syntax is needed (e.g., for edges rxy).

Proposition 5.5 (Stability) Let φ be a normalizer for F. Then:

1. $\tilde{F}$ is locally consistent if and only if φF is locally consistent.
2. $\tilde{F}$ is evident if and only if φF is evident.

Proof Claim (1) follows with the statements (2) and (3) of Proposition 3.3. Claim (2) follows from claim (1), Proposition 5.2, and the fact that φF is basic and hence equationally expanded. ■

Theorem 5.6 (Model Existence)
Every evident set is syntactically satisfiable.

Proof Let F be evident. By Propositions 5.5 and 4.2 we assume without loss of generality that F is basic. Also without loss of generality we assume that $(x \doteq x) \in F$ for every $x \in \mathcal{N}F$ (to meet the variable requirement of the Soundness Lemma). Now we choose an interpretation $\mathcal{I}$ such that $\mathcal{I}I = \mathcal{N}F$ and $\mathcal{I}x = x$ for all nominals $x \in \mathcal{N}F$. Moreover, we choose $\mathcal{I}$ such that it satisfies atomic formulas $a = px_1 \dots x_n$ if and only if $a \in F$. Since F is basic, $\mathcal{I}$ is syntactic for F. Since F is locally consistent, $\mathcal{I}$ satisfies all literal formulas in F. With the Soundness Lemma 5.1 and induction on the size of formulas it now follows that $\mathcal{I}$ satisfies all non-literal formulas in F. For the induction the size of formulas must be defined such that sx is smaller then $\exists s$ and $\forall s$. ■

6 Tableau Systems

The evidence conditions for non-literal formulas say which formulas are needed to render a formula evident. A tableau system adds these formulas step by step until either an evident set is reached or a local inconsistency appears. Tableau systems originated with Beth [4] and Hintikka [16]. It was Beth [4] who first used the term tableau.

We first introduce abstract tableau systems not committed to a particular logic. To do so, we assume that a set of formulas and a class of interpretations are given, and that for every interpretation $\mathcal{I}$ and every formula s it is defined whether or not $\mathcal{I}$ satisfies s. We make no further assumptions about formulas and interpretations.

A **clause** is a finite and non-empty set of formulas. A **clause set** is a set of clauses. The letter Γ will always denote a clause, and S will always denote a clause set. An interpretation $\mathcal{I}$ **satisfies a clause** Γ if it satisfies every formula in Γ. An interpretation $\mathcal{I}$ **satisfies a clause set** S if it satisfies at least one clause in S. Hence the empty clause set is unsatisfiable. Moreover, Γ is satisfiable if and only if $\{\Gamma\}$ is satisfiable.

We formalize the rules of tableau systems through moves, where a rule yields at least one move for every clause to which it applies. A **move** takes the form $(\Gamma, \{\Gamma_1, \ldots, \Gamma_n\})$ where $n \geq 0$ and $\Gamma \subsetneq \Gamma_i$ for all $i \in \{1, \ldots, n\}$. A move (Γ, S) is

- **refuting** if S is empty.
- **expansive** if S is non-empty.
- **branching** if S contains at least two clauses.
- **sound** if S is satisfiable if Γ is satisfiable.

A **tableau system** (Cla, Mov) consists of a set Cla of clauses and a set Mov of moves such that every move $(\Gamma, S) \in Mov$ satisfies the following conditions:

1. $\Gamma \in Cla$ and $S \subseteq Cla$
2. $(\Gamma, \emptyset) \in Mov \implies S = \emptyset$
3. (Γ, S) sound.

Proposition 6.1 If (Γ, S) is a move of a tableau system, then Γ is satisfiable if and only if at least one clause in S is satisfiable. Hence Γ is unsatisfiable if S is empty.

A tableau system yields a set of **proof trees**. Every single clause is a primitive proof tree. Using the moves of the tableau system we obtain composed proof trees. An example may look as follows:

$$\begin{tikzcd}
 & \Gamma \arrow[d, dash] & \\
 & \Gamma_1 \arrow[dl, dash] \arrow[dr, dash] & \\
\Gamma_2 & & \Gamma_3 \arrow[d, dash] \\
 & & \Gamma_4
\end{tikzcd}$$

This tree is obtained from the moves $(\Gamma, \{\Gamma_1\})$, $(\Gamma_1, \{\Gamma_2, \Gamma_3\})$ and $(\Gamma_3, \{\Gamma_4\})$. As we go down on a branch of a proof tree, the clauses contain more and more formulas. For instance, $\Gamma \subsetneq \Gamma_1 \subsetneq \Gamma_3 \subsetneq \Gamma_4$. A **tableau** exploits this property and represents a proof tree by giving at each node only the formulas that are added. Because of the soundness of the moves, the root clause of a proof tree is satisfiable if and only if at least one of the leaf clauses is satisfiable.

The following definitions and results are parameterized with respect to a tableau system (Cla, Mov). We define two **expansion relations**:

- $\Gamma \to \Gamma'$ if there exists a move $(\Gamma, S') \in Mov$ such that $\Gamma' \in S'$.

- $S \rightarrow S'$ if $S \subseteq \mathit{Cla}$ and there exists a move $(\Gamma, S'') \in \mathit{Mov}$ such that $\Gamma \in S$ and $S' = (S - \{\Gamma\}) \cup S''$.

We write $\Gamma \rightarrow^* \Gamma'$ and $S \rightarrow^* S'$ for the reflexive and transitive closures of the expansion relations. An **expansion** is a pair (Γ, Γ') or (S, S') such that $\Gamma \rightarrow \Gamma'$ or $S \rightarrow S'$. Note that a derivation $\Gamma_1 \rightarrow \cdots \rightarrow \Gamma_n$ corresponds to a path in a proof tree, and that $\{\Gamma\} \rightarrow^* S$ holds if and only if there is a proof tree with Γ as root clause and the clauses in S as leaf clauses.

A clause $\Gamma \in \mathit{Cla}$ is

- **refuted** if Mov contains a refuting move for Γ.
- **refutable** if $\{\Gamma\} \rightarrow^* \emptyset$.
- **verified** if Γ is not refuted and there is no clause Γ' such that $\Gamma \rightarrow \Gamma'$.
- **verifiable** if there is a verified clause Γ' such that $\Gamma \rightarrow^* \Gamma'$.

A clause is refutable if it has a proof tree whose leaves are all refuted, and verifiable if it has a proof tree with at least one verified leaf. Since every clause is by itself a proof tree, refuted clauses are refutable, and verified clauses are verifiable.

Proposition 6.2

1. If $\Gamma \rightarrow \Gamma'$, then $\Gamma \subsetneq \Gamma'$ and Γ is not refuted.
2. If $S \rightarrow S'$, then S is satisfiable if and only if S' is satisfiable.

Proposition 6.3 Refutable clauses are unsatisfiable.

A tableau system is

- **verification-sound** if every verified clause is satisfiable.
- **verification-complete** if every satisfiable clause is verifiable.
- **refutation-complete** if every unsatisfiable clause in *Cla* is refutable.
- **complete** if it is verification- and refutation-complete.
- **terminating** if the expansion relation $\Gamma \rightarrow \Gamma'$ is terminating.

Proposition 6.4

1. A refutation-complete tableau system is verification-sound.
2. If a clause is verifiable in a verification-sound tableau system, it is satisfiable.
3. A verification-sound and terminating tableau system is complete.
4. A verification-sound and terminating tableau system yields a decision procedure for the satisfiability of clauses.

The decision procedure starts from $\{\Gamma\}$ and applies moves until either a verified clause is reached or no clause is left. If a verified clause Γ' is reached, Γ is

$$\mathcal{R}_\neg\ \frac{\neg s}{\emptyset}\ s \in \tilde{\Gamma} \cup \text{Id} \qquad \mathcal{R}_\wedge\ \frac{s_1 \wedge s_2}{s_1,\ s_2} \qquad \mathcal{R}_\vee\ \frac{s_1 \vee s_2}{s_1 \mid s_2} \qquad \mathcal{R}_\lambda\ \frac{(\lambda x.s)y}{s^x_y}$$

$$\mathcal{R}_\exists\ \frac{\exists s}{sx}\ x \notin \mathcal{N}\Gamma \wedge \exists s \text{ not evident in } \tilde{\Gamma} \qquad \mathcal{R}_\forall\ \frac{\forall s}{sx}\ x \in \mathcal{N}\Gamma$$

$$\mathcal{R}_\Diamond\ \frac{\Diamond\rho s x}{\rho x y,\ s y}\ y \notin \mathcal{N}\Gamma \wedge \Diamond\rho s x \text{ not evident in } \tilde{\Gamma} \qquad \mathcal{R}_\Box\ \frac{\Box\rho s x}{s y}\ \rho x y \in \tilde{\Gamma}$$

Figure 2: Rules for $\mathcal{T}$

verifiable and hence satisfiable since the tableau system is verification-sound. If no clause is left, Γ is refutable and hence unsatisfiable.

7 A Tableau System for PLM

We are now construct a tableau system $\mathcal{T}$ for PLM. $\mathcal{T}$ is obtained as an operational reformulation of the notion of PLM-evidence. As clauses we take all finite sets Γ of PLM-formulas such that Γ contains at least one nominal. A clause Γ will be refuted in $\mathcal{T}$ if and only if $\tilde{\Gamma}$ is locally inconsistent, and verified if and only if $\tilde{\Gamma}$ is evident. By Lemma 5.4 we know that it suffices that $\mathcal{T}$ has a refuting move for every clause Γ such that $\neg s \in \Gamma$ and $s \in \tilde{\Gamma} \cup \text{Id}$. The expansive moves (Γ, S) of $\mathcal{T}$ are chosen such that there is no refuting move for Γ and $\tilde{\Gamma} \subsetneq \tilde{\Delta}$ for every $\Delta \in S$. By Compatibility (Proposition 5.3) we know that it suffices if there is an expanding move for every clause Γ containing a non-literal formula that is not evident in $\tilde{\Gamma}$. Thus the expansive moves are operational reformulations of the evidence conditions for the different types of non-literal formulas. Figure 2 represents all moves of $\mathcal{T}$ through rules. $\mathcal{R}_\neg$ represents the refuting moves. The other rules correspond to the evidence conditions for non-literal formulas and describe expansive moves. A rule can only be applied to a clause Γ if the premise of the rule is in Γ and the side conditions of the rule are satisfied by Γ. If this is the case, the rule adds the formulas appearing as its conclusions to Γ.

$\mathcal{R}_\vee$ is the only rule that describes branching moves. An example of a branching move obtained from $\mathcal{R}_\vee$ is $(\{p \vee q\},\ \{\{p \vee q,\ p\},\ \{p \vee q,\ q\}\})$. $\mathcal{R}_\vee$ also yields non-branching moves, for instance, $(\{p \vee p\}, \{\{p \vee p,\ p\}\})$. Note that $\mathcal{R}_\vee$ does not yield a move for $\{p \vee q,\ p\}$. Here are examples of verified clauses:

$$\{\forall x.px,\ (\lambda x.px)a,\ pa\}$$
$$\{\forall x.px,\ (\lambda x.px)a,\ (\lambda x.px)b,\ pa,\ pb\}$$
$$\{\forall x.px,\ (\lambda x.px)b,\ pa,\ a \doteq b\}$$
$$\{\exists x.px,\ (\lambda x.px)a,\ pb,\ a \doteq b\}$$
$$\{\Diamond r(\lambda x.px)a,\ rbb, (\lambda x.px)a,\ pb,\ a \doteq b\}$$
$$\{\Diamond r(\Box r^{\smile}(\lambda x.px))a,\ rab,\ \Box r^{\smile}(\lambda x.px)b,\ (\lambda x.px)a,\ pa\}$$

Proposition 7.1 $\mathcal{T}$ is a tableau system.

Proof It is straightforward to verify the soundness of the moves of $\mathcal{T}$. The other properties hold by construction. ■

Lemma 7.2 (Evidence) If Γ is verified in $\mathcal{T}$, then $\tilde{\Gamma}$ is evident.

Proof Let Γ be verified in $\mathcal{T}$. Certainly, $\tilde{\Gamma}$ is equationally expanded. Since $\mathcal{R}_\neg$ does not apply to Γ, we know by Lemma 5.4 that $\tilde{\Gamma}$ is locally consistent. Moreover, $\tilde{\Gamma}$ contains a nominal since every clause of $\mathcal{T}$ contains a nominal.

It remains to show that all non-literal formulas in $\tilde{\Gamma}$ are evident in $\tilde{\Gamma}$. By Compatibility (Proposition 5.3) it suffices to show that all non-literal formulas in Γ are evident in $\tilde{\Gamma}$. The tableau rules for non-literal formulas are designed such that a rule is applicable if its premise formula is not evident in $\tilde{\Gamma}$. Hence all non-literal formulas in Γ are evident in $\tilde{\Gamma}$ since no rule applies to Γ. ■

Theorem 7.3 (Verification-Soundness)
If a clause is verified in $\mathcal{T}$, then it is syntactically satisfiable.

Proof Follows with Lemma 7.2, Theorem 5.6, and Proposition 4.1. ■

Example 7.4 $\mathcal{T}$ does not terminate for the satisfiable formula $\forall x.\exists y.rxy$:

$\forall x.\exists y.rxy,\ a \doteq a$	initial clause
$(\lambda x.\exists y.rxy)a,\ \exists y.ray$	$\mathcal{R}_\forall,\ \mathcal{R}_\lambda$
$(\lambda y.ray)b,\ rab$	$\mathcal{R}_\exists,\ \mathcal{R}_\lambda$
$(\lambda x.\exists y.rxy)b,\ \exists y.rby$	$\mathcal{R}_\forall,\ \mathcal{R}_\lambda$
$\dots$	

The non-termination comes from the interplay of $\mathcal{R}_\forall$ and $\mathcal{R}_\exists$ and the fact that $\mathcal{R}_\exists$ introduces new nominals. □

Example 7.5 Here is a derivation verifying a satisfiable clause. We use syntactic sugar to hide lambda abstractions.

$\Diamond rpa,\ \Box r(a \wedge \Diamond rp)a$	initial clause
$rab,\ (\lambda x.px)b,\ pb$	$\mathcal{R}_\Diamond,\ \mathcal{R}_\lambda$
$(a \wedge \Diamond rp)b$	$\mathcal{R}_\Box$
$a \doteq b \wedge \Diamond rpb$	$\mathcal{R}_\lambda$
$a \doteq b,\ \Diamond rpb$	$\mathcal{R}_\wedge$

The final clause Γ is verified since $\Diamond rpb$ is evident in $\tilde{\Gamma}$ (rbb and $(\lambda x.px)b$ are in $\tilde{\Gamma}$). $\Box$

8 Straight Termination

Since satisfiability of PLN-formulas is not semi-decidable (a single binary predicate suffices, see [9]), decidable fragments of PLM must restrict the use of the quantifiers. We begin with a tableau-decidable fragment of PLM that excludes $\Diamond$-formulas.

We distinguish between ordinary, existential and universal formulas. Formulas of the form $\exists s$ or $\Diamond rsx$ are called **existential**, and formulas of the form $\forall s$ or $\Box rsx$ are called **universal**. Formulas that are neither universal nor existential are called **ordinary**.

The rules $\mathcal{R}_\exists$ and $\mathcal{R}_\Diamond$ for existential formulas introduce fresh nominals, that is, nominals that do not appear in the clauses the rules are applied to. For this reason we refer to $\mathcal{R}_\exists$ and $\mathcal{R}_\Diamond$ as **generative rules**. $\mathcal{R}_\exists$ and $\mathcal{R}_\Diamond$ have to introduce fresh nominals since otherwise they would not be sound.

A **subterm of a set** F is a term that appears as subterm in a formula $s \in F$. We write **Sub** F for the set of all subterms of F. We say that F **contains** t if $t \in \text{Sub}\, F$.

Proposition 8.1 $\mathcal{T}$ terminates on clauses not containing existential formulas.

For the proof of the proposition we need some terminology. The **height** of a clause Γ is the size of the largest formula in Γ. If $\Gamma \to \Delta$, then Γ and Δ have the same height. In other word, expansion preserves the height of clauses.

The **breadth** of a clause Γ is the number of elements of Γ (as a set). If $\Gamma \to \Delta$, then the breadth of Δ is larger than the breadth of Γ. In other words, expansion increases the breadth of clauses.

The **vocabulary** of a clause Γ consists of all nominals and all predicate names that occur in Γ. If $\Gamma \to \Delta$, then the vocabulary of Γ is a subset of the vocabulary of Δ. While generative rules enlarge the vocabulary of a clause, non-generative rules preserve the vocabulary of a clause.

The **stock** of a clause Γ consists of all PLM-formulas whose size is at most the height of Γ and that contain only nominals and predicate names in the vocabulary of Γ. The stock of a clause is finite since the vocabulary of a clause is finite. All non-generative rules preserve the stock of a clause.

The **slack** of a clause Γ is the number of formulas in the stock of Γ that are not in Γ. Every non-generative rule decreases the slack of a clause. Hence we know that an infinite derivation must employ a generative rule. Thus we have a proof of Proposition 8.1.

Proposition 8.2 (Monotonicity) Expansion preserves evidence of non-universal formulas. That is, if a non-universal formula s is evident in Γ and $\Gamma \subseteq \Delta$, then s is evident in Δ. Moreover, the evidence of a universal formula can only be lost by an expansion with a generative rule (i.e. $\mathcal{N}\Gamma \subsetneq \mathcal{N}\Delta$).

An **$\exists$-formula** is is a formula of the form $\exists s$. A clause Γ is **straight** if it is diamond-free (i.e., $\Diamond \notin \operatorname{Sub}\Gamma$) and every $\exists$-formula in $\operatorname{Sub}\Gamma$ is closed. We will show that $\mathcal{T}$ terminates on straight clauses.

Proposition 8.3 If Γ is straight and $\Gamma \to \Delta$ in $\mathcal{T}$, then Δ is straight.

Only $\mathcal{R}_\lambda$ can introduce new $\exists$-subformulas, and only if there are open $\exists$-subformulas. Consider $(\lambda x.\exists y.rxy)a$ as an example. Each time $\mathcal{R}_\exists$ is applied it renders an $\exists$-formula evident that was not evident before (ensured by the side condition of $\mathcal{R}_\exists$). Since evidence of $\exists$-formulas is preserved by all rules, $\mathcal{R}_\exists$ cannot cause divergence if all $\exists$-formulas are closed.

The **$\exists$-power of a clause** Γ is the number of $\exists$-formulas in $\operatorname{Sub}\Gamma$ that are not evident in $\tilde{\Gamma}$. Obviously, $\mathcal{R}_\exists$ decreases the $\exists$-power of a clause. Moreover, if all $\exists$-formulas are closed, no rule increases the $\exists$-power of a clause.

Theorem 8.4 (Straight Termination)
$\mathcal{T}$ terminates on straight clauses.

Proof We summarize the facts that prove the theorem.

1. A derivation issuing from a straight clause involves only straight clauses.
2. No rule increases the $\exists$-power of a straight clause.
3. $\mathcal{R}_\exists$ decreases the $\exists$-power of a straight clause.
4. $\mathcal{R}_\Diamond$ is not applicable.
5. All rules but $\mathcal{R}_\exists$ and $\mathcal{R}_\Diamond$ decrease the slack of a clause. ∎

Corollary 8.5 $\mathcal{T}$ decides the satisfiability of straight clauses.

Proof Follows from Theorems 7.3 and 8.4. ∎

Corollary 8.6 Straight clauses are finitely satisfiable if they are satisfiable.

The termination result for straight clauses can be generalized by admitting open $\exists$-formulas whose free variables are existentially quantified (i.e., by $\exists$). To do this, the definition of $\exists$-power must be adapted. The insight is that new closed $\exists$-formulas are introduced only after a larger $\exists$-formula has been made evident with $\mathcal{R}_\exists$.

9 Regular Clauses

To arrive at decidable fragments of PLM that admit $\Diamond$-formulas, we must restrict the use of modal quantification. It turns out that we must also disallow formulas of the form $\neg rxy$ if r is used with a modal quantifier. A set F of PLM-formulas is

- **well-quantified** if $\mathrm{Sub}\,F$ contains no open formula of the form $\exists s$, $\Diamond \rho s x$, or $\Box \rho s x$.
- **edge-positive** if $\neg rxy \in \mathrm{Sub}\,F$ implies that $\mathrm{Sub}\,F$ contains no modality containing r.
- **regular** if F is well-quantified and edge-positive.
- **converse-free** if $\mathrm{Sub}\,F$ contains no term of the form $r^{\smallsmile}$.

The translation of formulas in ordinary modal syntax to PLM always yields regular formulas. The exception is the down-arrow binder of hybrid logic, whose presence renders hybrid logic undecidable [1]. We define the **modal terms of a set F** as follows:

$$\begin{aligned}\mathrm{Mod}\,F \;:=\; & \{\,\Diamond\rho s \mid \Diamond\rho s \in \mathrm{Sub}\,\Gamma\,\} \;\cup\; \{\,\Box\rho s \mid \Box\rho s \in \mathrm{Sub}\,\Gamma\,\} \\ & \cup \{\, s \mid \exists\rho\colon \Diamond\rho s \in \mathrm{Sub}\,\Gamma \;\vee\; \Box\rho s \in \mathrm{Sub}\,\Gamma\,\}\end{aligned}$$

To have an example, we list the modal terms of the clause $\{\Diamond r(\Box r(\lambda x.px))a\}$: $\Diamond r(\Box r(\lambda x.px))$, $\Box r(\lambda x.px)$, $\lambda x.px$.

Proposition 9.1 Let $\Gamma \to \Delta$ in $\mathcal{T}$. Then:

1. If Γ is well-quantified, then $\mathrm{Mod}\,\Gamma = \mathrm{Mod}\,\Delta$.
2. If Γ is well-quantified, then Δ is well-quantified.
3. If Γ is regular, then Δ is regular.

Proof New modal subterms can only be introduced by $\mathcal{R}_\lambda$ and only if there are open modal terms. However, well-quantifiedness excludes the presence of open modal terms. This proves claim (1). The other claims are easy to verify. ■

Well-quantifiedness will be crucial for the termination proofs to come since it ensures that the tableau rules preserve the modal terms of a clause (as stated by the above proposition). Note that $\mathrm{Mod}\,\Gamma$ is finite since clauses are finite. Edge-positiveness is of no relevance for termination, but is essential for verification-soundness.

The tableau system in this paper all have the property that they can only verify clauses that are finitely satisfiable. Hence a class of clauses can only be tableau-decidable with our approach if it excludes satisfiable clauses that are not finitely satisfiable. Such clauses can be obtained by specifying a binary relation that is total, irreflexive, and transitive.

Example 9.2 (TIT) The strict order $<$ on $\mathbb{N}$ is a relation that is total, irreflexive, and transitive (TIT). It is easy to see that there is no finite relation that is TIT. Here is a PLM-clause specifying that a relation r is TIT:

$\forall x.\ \Diamond r(\lambda x.x\doteq x)x$	totality
$\forall x.\ \neg rxx$	irreflexivity
$\forall xyz.\ \neg rxy \vee \neg ryz \vee rxz$	transitivity

Note that the clause is well-quantified but not edge-positive. Thus the example explains why we require edge-positiveness.

Here is another PLM-clause specifying that a relation r is TIT:

$\forall x.\ \Diamond r(\lambda x.x\doteq x)x$	totality
$\forall x.\ \Box r(\neg x)x$	irreflexivity
$\forall xyz.\ \Box r(\neg y)x \vee \Box r(\neg z)y \vee \Diamond rzx$	transitivity

This time we use the expressive features of hybrid logic. The example is written with some notational sugar. For instance, the notation $\Diamond rz$ stands for the modal term $\Diamond r(\lambda x.\,z\doteq x)$, and $\Box r(\neg x)$ stands for $\Box r(\lambda y.\,x\not\doteq y)$. Note that the clause is edge-positive but not well-quantified since it contains open modal terms (e.g., $\Diamond r(\lambda x.\,z\doteq x)$). $\square$

Regularity takes away most of the extra-expressivity PLM has over hybrid logic. One expressive feature that remains is reflexivity: $\forall x.rxx$.

10 Quasi-Evidence

$\mathcal{T}$ diverges on most clauses that involve diamonds. The reason is that the evidence condition for diamond formulas requires for $\Diamond\rho sx$ the presence of an edge ρxy, which $\mathcal{R}_\Diamond$ can only add by introducing a new nominal. We will now define a weaker quasi-evidence property for diamond formulas that doesn't require the presence of edges.

Example 10.1 Consider the formula $\forall x.\Diamond rpx$. It is finitely satisfiable since

$$\{\forall x.\Diamond rpx,\ (\lambda x.\Diamond rpx)a,\ \Diamond rpa,\ raa,\ (\lambda x.px)a,\ pa\}$$

is an evident clause. However, $\mathcal{T}$ diverges on this formula since it tries to build a tree-like model:

$\forall x.\Diamond rpx,\ pa$	initial clause
$(\lambda x.\Diamond rpx)a,\ \Diamond rpa$	$\mathcal{R}_\forall,\ \mathcal{R}_\lambda$
$rab,\ (\lambda x.px)b,\ pb$	$\mathcal{R}_\Diamond,\ \mathcal{R}_\lambda$
$(\lambda x.\Diamond rpx)b,\ \Diamond rpb$	$\mathcal{R}_\forall,\ \mathcal{R}_\lambda$
$rbc,\ (\lambda x.px)c,\ pc$	$\mathcal{R}_\Diamond,\ \mathcal{R}_\lambda$
$\dots$	

□

We say that an edge rxy is **safe in** F if $rxy \in F$, or if the following conditions are satisfied:

1. $\forall t\colon\ \Box rtx \in F \Longrightarrow ty \in F$
2. $\forall t\colon\ \Box r\check{}ty \in F \Longrightarrow tx \in F$

For diamond formulas we have a **quasi-evidence condition**:

$$\Diamond\rho sx \quad \text{quasi-evident in } F \quad \exists y\colon\ \rho xy \text{ safe in } F\ \wedge\ sy \in F$$

An edge-positive set F is **quasi-evident** if it satisfies all the conditions for evident sets except the one for diamond formulas, for which it satisfies the quasi-evidence condition. For instance, the set $\{\Diamond rpa,\ (\lambda x.px)b,\ pb\}$ is quasi-evident since rab is safe.

Proposition 10.2 Let F be quasi-evident. Then $F \cup \{rxy\}$ is quasi-evident if and only if rxy is safe in F.

Proposition 10.3

1. If a diamond formula is evident in F, then it is also quasi-evident in F.
2. If a set is evident, then it is also quasi-evident.

Proposition 10.4 (Stability) Let φ be a normalizer for F. Then:

1. An edge rxy is safe in $\tilde{F}$ if and only if $r(\varphi x)(\varphi y)$ is safe in φF.
2. A formula $\Diamond\rho xy$ is quasi-evident in $\tilde{F}$ if and only if $\Diamond\rho(\varphi x)(\varphi y)$ is quasi-evident in φF.

3. $\tilde{F}$ is quasi-evident if and only if φF is quasi-evident.

Proof Claim (1) follows with Proposition 3.3 (2). Claim (2) follows with Claim (1). Claim (3) follows with Claim (2) and Propositions 5.2 and 5.5. ■

Lemma 10.5 (Safe Edges) Let F be quasi-evident and basic, and let R be the set of all edges s such that s is safe in F and $\mathcal{N}s \subseteq \mathcal{N}F$. Then $F \cup R$ is evident.

Proof The addition of safe edges preserves the local consistency and the equational expansion of F (recall that quasi-evident sets are edge-positive). Let $t \in F$ be a non-literal formula. It remains to show that t is evident in $F \cup R$. If t is neither a diamond nor a box formula, then t is evident in $F \cup R$ since it is evident in F.

If t is a box formula, a certain condition must be satisfied by every edge that is adjacent to s. For the edges in F this condition is satisfied since t is evident in F. For the edges in R the condition is satisfied since they are safe in F.

If $t = \Diamond\rho s x \in F$, we distinguish two cases. If t is evident in F, then t is also evident in $F \cup R$. If t is not evident in F, there exists a nominal y such that $sy \in F$ and $\rho x y$ is safe in F. Hence $\rho x y \in R$. Thus t is evident in $F \cup R$. ■

Theorem 10.6 (Model Existence)
Every quasi-evident set is syntactically satisfiable.

Proof By Lemma 10.4 (3) and Proposition 4.2 we can assume without loss of generality that F is basic. Now the claim follows with Lemma 10.5 and Theorem 5.6. ■

Quasi-evidence yields a verification-sound tableau system $\mathcal{T}^q$, which is obtained as follows. As clauses $\mathcal{T}^q$ takes all edge-positive clauses. The moves of $\mathcal{T}^q$ are defined analogously to $\mathcal{T}$ except that for diamond formulas we now employ the rule

$$\mathcal{R}^q_\Diamond \ \frac{\Diamond\rho s x}{\rho x y,\ s y}\ y \notin \mathcal{N}\Gamma \wedge \Diamond\rho s x \text{ not quasi-evident in } \tilde{\Gamma}$$

Proposition 10.7 $\mathcal{T}$ is a verification-sound tableau system.

Proof Analogous to the proofs of Lemma 7.2 and Theorem 7.3. ■

We will show that $\mathcal{T}^q$ can verify every converse-free regular clause that is satisfiable. Example 10.1 shows that $\mathcal{T}$ doesn't have this property. The following example shows that $\mathcal{T}^q$ doesn't terminate.

Example 10.8 Here is a diverging $\mathcal{T}^q$-derivation starting from a satisfiable regular clause (some formulas are omitted):

$\forall x.\Diamond rpx \wedge \Box rqx,\ pa$	initial clause
$\Diamond rpa,\ \Box rqa$	$\mathcal{R}_\forall,\ \mathcal{R}_\lambda,\ \mathcal{R}_\wedge$
$rab,\ pb$	$\mathcal{R}^q_\Diamond,\ \mathcal{R}_\lambda$
$\Diamond rpb,\ \Box rqb$	$\mathcal{R}_\forall,\ \mathcal{R}_\lambda,\ \mathcal{R}_\wedge$
$\ldots$	

If we prioritize $\mathcal{R}_\Box$ over $\mathcal{R}_\Diamond$, we terminate with a quasi-evident clause after a few steps.

$\forall x.\Diamond rpx \wedge \Box rqx,\ pa$	initial clause
$\Diamond rpa,\ \Box rqa$	$\mathcal{R}_\forall,\ \mathcal{R}_\lambda,\ \mathcal{R}_\wedge$
$rab,\ pb,\ qb$	$\mathcal{R}^q_\Diamond,\ \mathcal{R}_\lambda,\ \mathcal{R}_\Box$
$\Diamond rpb,\ \Box rqb$	$\mathcal{R}_\forall,\ \mathcal{R}_\lambda,\ \mathcal{R}_\wedge$

Note that $\Diamond rpa$ is evident and $\Diamond rpb$ is quasi-evident since rbb is safe. $\Box$

11 A Pattern-Based System

We will now present a terminating tableau system that decides the satisfiability of converse-free regular clauses. One way to get such a system is to take the subsystem of $\mathcal{T}^q$ that prioritizes $\mathcal{R}_\Box$ over $\mathcal{R}^q_\Diamond$ (i.e., $\mathcal{R}^q_\Diamond$ can only be applied to clauses to which $\mathcal{R}_\Box$ does not apply). However, we take a slightly different approach, which enjoys simpler proofs and extends to transitive relations (see §14). We refer to this approach as *pattern-based approach* (patterns will play an essential role in the termination proof).

A formula $\Diamond rsx$ is **pattern-evident** in a set F if there exist nominals x', y such that $rx'y,\ sy \in F$ and $\forall t\colon \Box rtx \in F \Longrightarrow \Box rtx' \in F$. An edge-positive and converse-free set F is **pattern-evident** if it satisfies all the conditions for evident sets except the one for diamond formulas, for which it satisfies pattern-evidence. A set F is **box-evident** if every box-formula in F is evident in F.

Proposition 11.1 If a converse-free diamond formula is evident in a set F, then it is pattern-evident in F.

Proposition 11.2 Let F be pattern-evident. Then a converse-free diamond formula is quasi-evident in F if it is pattern-evident in F.

Proof Let $\Diamond rsx \in F$ be pattern-evident in F. Then there exist nominals x', y such that $rx'y$, $sy \in F$ and $\forall t\colon \Box rtx \in F \Longrightarrow \Box rtx' \in F$. Since all box-formulas are evident in F, we have $\forall t\colon \Box rtx \in F \Longrightarrow ty \in F$. Hence rxy is safe in F since F is edge-positive and converse-free. ■

Proposition 11.3 Every pattern-evident set is quasi-evident.

Proof Follows with Proposition 11.2. ■

Proposition 11.4 (Stability) Let φ be a normalizer for F. Then a diamond formula s is pattern-evident in $\tilde{F}$ if and only if φs is pattern-evident in φF.

Proposition 11.5 (Compatibility) Let $s \approx_F s'$. Then s is pattern-evident in $\tilde{F}$ if and only if s' is pattern-evident in $\tilde{F}$.

Proof Let φ be a normalizer of F, $\Diamond r(\varphi s)(\varphi x) = \Diamond r(\varphi s')(\varphi x')$, and let $\Diamond rsx$ be pattern-evident in $\tilde{F}$. Then, by Stability, $\Diamond r(\varphi s)(\varphi x)$ is pattern-evident in φF. Hence $\Diamond r(\varphi s')(\varphi x')$ is pattern-evident in φF. Thus, by Stability, $\Diamond rs'x'$ is pattern-evident in $\tilde{F}$. ■

We now define a tableau system $\mathcal{T}^p$. As clauses $\mathcal{T}^p$ takes all converse-free regular clauses. The moves of $\mathcal{T}^p$ are defined analogously to $\mathcal{T}$ except that for diamond formulas we now employ the rule

$$\mathcal{R}^p_\Diamond \quad \frac{\Diamond rsx}{rxy,\ sy} \quad y \notin \mathcal{N}\Gamma \wedge \Diamond rsx \text{ not pattern-evident in } \tilde{\Gamma}$$

Check that $\mathcal{T}^p$ terminates after 2 steps on the clause in Example 10.1, and that it also terminates on the clause in Example 10.8.

Proposition 11.6 $\mathcal{T}^p$ is a tableau system.

Lemma 11.7 (Evidence) If Γ is verified in $\mathcal{T}^p$, then $\tilde{\Gamma}$ is quasi-evident.

Proof Let Γ be verified in $\mathcal{T}^p$. By reuse of the proof of Lemma 7.2, we know that $\tilde{\Gamma}$ satisfies all evidence conditions but possibly the condition for diamond formulas. Since $\mathcal{R}^p_\Diamond$ is not applicable to Γ, we know that all diamond formulas in Γ are pattern-evident in $\tilde{\Gamma}$. Hence all diamond formulas in $\tilde{\Gamma}$ are pattern-evident in $\tilde{\Gamma}$ by Compatibility (Proposition 11.5). Since by assumption Γ is edge-positive and converse-free, we now know that $\tilde{\Gamma}$ is pattern-evident. Hence $\tilde{\Gamma}$ is quasi-evident by Proposition 11.3. ■

Theorem 11.8 (Verification-Soundness)
If a clause is verified in $\mathcal{T}^p$, then it is syntactically satisfiable.

Proof Follows with Lemma 11.7, Theorem 10.6, and Proposition 4.1. ∎

We now start with the termination proof for $\mathcal{T}^p$. A property for clauses is **monotone** if, if it holds for a set F, it also holds for all supersets of F. Recall from §8 that evidence of non-universal formulas is monotone. Pattern-evidence of diamond formulas, however, is not monotone. Hence we need some other property we can base the termination argument on. A **pattern** P is a set $\{\Diamond rs, \Box rs_1, \dots, \Box rs_n\}$ of terms where $n \geq 0$. A pattern is **realized in** F if there are nominals x, y such that the formulas rxy, sy, and $\Box rs_1x, \dots, \Box rs_nx$ are in F.

Proposition 11.9 (Monotonicity)
If a pattern P is realized in F and $F \subseteq F'$, then P is realized in F'.

Proposition 11.10 A formula $\Diamond rsx$ is pattern-evident in $\tilde{\Gamma}$ if and only if the pattern $\{\Diamond rs\} \cup \{\, \Box rt \in \operatorname{Mod}\Gamma \mid \Box rtx \in \tilde{\Gamma}\,\}$ is realized in $\tilde{\Gamma}$.

Proof The direction $\Rightarrow$ is easy to verify. To show the other direction, let the given pattern be realized in $\tilde{\Gamma}$ and let φ be a normalizer for Γ. By Proposition 11.4 it suffices to show that $\Diamond r(\varphi s)(\varphi x)$ is pattern-evident in $\varphi\Gamma$. Since the pattern is realized in $\tilde{\Gamma}$, there exist x', y such that $r(\varphi x')(\varphi y)$ and $(\varphi s)(\varphi y)$ are in $\varphi\Gamma$ and $\Box r(\varphi t)(\varphi x') \in \varphi\Gamma$ for all $\Box rt \in \operatorname{Mod}\Gamma$ such that $\Box r(\varphi t)(\varphi x) \in \varphi\Gamma$. Thus $\Diamond r(\varphi s)(\varphi x)$ is pattern-evident in $\varphi\Gamma$. ∎

We define the **$\Diamond$-power** of a clause Γ as the number of patterns $P \subseteq \operatorname{Mod}\Gamma$ that are not realized in Γ. Because all rules of $\mathcal{T}^p$ preserve $\operatorname{Mod}\Gamma$ (Proposition 9.1) and realization of patterns is a monotone property, no rule of $\mathcal{T}^p$ increases the $\Diamond$-power of a clause. However, by Proposition 11.10 and Proposition 11.1 we know that $\mathcal{R}^p_\Diamond$ decreases the $\Diamond$-power of a clause.

Theorem 11.11 (Termination)
$\mathcal{T}^p$ is terminating.

Proof The claim follows from the following facts. The facts concerning the slack and the $\exists$-power of clauses were shown in §8.

1. Only well-quantified clauses are involved.
2. No rule increases the $\exists$- or the $\Diamond$-power of a well-quantified clause.
3. $\mathcal{R}_\exists$ decreases the $\exists$-power of a well-quantified clause.
4. $\mathcal{R}^p_\Diamond$ decreases the $\Diamond$-power of a well-quantified clause.
5. All rules but $\mathcal{R}_\exists$ and $\mathcal{R}_\Diamond$ decrease the slack of a clause. ∎

Corollary 11.12 $\mathcal{T}^p$ decides the satisfiability of converse-free regular clauses.

Proof Follows from Theorems 11.8 and 11.11. ∎

Corollary 11.13 Converse-free regular clauses are finitely satisfiable if they are satisfiable.

For an efficient decision procedure it seems advantageous to prioritize $\mathcal{R}_\Box$ over $\mathcal{R}^p_\Diamond$, that is, to apply $\mathcal{R}^p_\Diamond$ only to box-evident clauses. Moreover, $\mathcal{R}^p_\Diamond$ should only be applied to diamond formulas that are not quasi-evident. By Proposition 11.2 we know that this will result in a complete decision procedure. Note that such a procedure checks for quasi-evidence of diamond formulas rather than pattern-evidence.

Corollary 11.14 $\mathcal{T}^q$ is verification-complete for converse-free regular clauses.

Example 11.15 We close with a example that suggests that the pattern-based technique does not extend to regular clauses containing converse modalities. Here is an unsatisfiable regular clause that is pattern-evident except for the fact that it contains converse modalities:

$$\begin{aligned}\{\ &(\Diamond r(\Box r^{\smile}\neg p))a,\ \ raa,\ \ \Box r^{\smile}(\neg p)a,\ \ (\lambda x.\neg px)a,\ \ \neg pa\\ &(\Diamond r(\Box r^{\smile}\neg p))b,\ \ pb\ \}\end{aligned}$$

The diamond formula at a is evident, and the diamond formula at b is pattern-evident. □

12 A Chain-Based System

In this section we present a terminating tableau system that decides the satisfiability of regular clauses possibly containing converse modalities. The system builds on a well-known technique we call **chain-based blocking**. Chain-based blocking exploits a finiteness condition that also appears with the model-theoretic technique of filtration [5]. Chain-based blocking first appears in Kripke [24] with a terminating tableau system deciding S4 (modal logic with a reflexive and transitive relation). In Hughes and Cresswell [19] chain-based blocking appears as *rule of repeating chains*. Horrocks and Sattler [18] use chain-based blocking to cope both with transitive relations and converse modalities. They observe that with converse relations chain-based blocking becomes dynamic. This is also the case with equations. Bolander and Blackburn [7] employ chain-based blocking for a hybrid logic with converse modalities.

Example 12.1 Here is a diverging $\mathcal{T}$-derivation starting from a satisfiable regular clause (λ-formulas are omitted):

$\forall x.\Diamond r(\Box r^- p)x,\ \ a\dot{=}a$	initial clause
$\Diamond r(\Box r^- p)a$	$\mathcal{R}_\forall,\ \mathcal{R}_\lambda$
$rab,\ \ \Box r^- pb$	$\mathcal{R}_\Diamond$
pa	$\mathcal{R}_\Box,\ \mathcal{R}_\lambda$
$\Diamond r(\Box r^- p)b$	$\mathcal{R}_\forall,\ \mathcal{R}_\lambda$
$rbc,\ \ \Box r^- pc$	$\mathcal{R}_\Diamond$
pb	$\mathcal{R}_\Box,\ \mathcal{R}_\lambda$
$\dots$	

Let Γ be the clause that is reached after pb has been added, and let $\Gamma' \subseteq \Gamma$ be the clause obtained from Γ by deleting the two formulas containing the nominal c. Now note that Γ' is quasi-evident since rbb is safe in Γ'. The example shows that $\mathcal{T}$ can diverge although it has produced a quasi-evident subclause containing the initial clause. $\Box$

Chain-based blocking builds on a notion of modal equivalence. We define the **label set of x in F** as $\mathcal{L}_F x := \{\, t \in \mathrm{Mod}\, F \mid tx \in \tilde{F}\,\}$. Note that $\mathcal{L}_F x = \emptyset$ if $x \notin \mathcal{N}F$. We say that two nominals x, y are **modally equivalent in F** if $\mathcal{L}_F x = \mathcal{L}_F y$.

Chain-based blocking records the ancestors of the nominals introduced with the diamond rule through a relation $\prec$ such that $x \prec y$ holds if and only if the nominal y was introduced to expand a diamond formula at x. So for every nominal y we know the complete ancestor chain $x \prec \cdots \prec y$. An ancestor chain is *repeating* if it contains two different nominals that are modally equivalent. Chain-based blocking now disallows diamond expansions of nominals whose ancestor chain is repeating. Since $\mathrm{Mod}\,\Gamma$ is finite and fixed for derivations issuing from well-quantified clauses, the diamond rule can add nominals only up to a certain ancestor depth. This proviso suffices for termination and preserves completeness.

We model the **ancestor relation** through an a priori given binary relation $\prec$ on the set of all nominals. If $x \prec y$, we say that x is a **predecessor** of y, and that y is a **successor** of x. A nominal is **initial** if it doesn't have a predecessor. We assume that the ancestor relation satisfies the following conditions:

1. Every nominal has at most one predecessor.
2. There are no infinite chains $\cdots \prec x_3 \prec x_2 \prec x_1$.
3. There are infinitely many initial nominals.
4. Every nominal has infinitely many successors.

An **ancestor chain** is a tuple $(x_1, \ldots, x_n)$ of nominals such that $x_1 \prec \cdots \prec x_n$. The **ancestor chain of a nominal** x is the unique ancestor chain of maximal length that ends at x. The **depth of a nominal** x is the length of its ancestor chain. We write $\operatorname{depth} x$ for the depth of x. An ancestor chain is **repeating in** F if it contains two different nominals that are modally equivalent in F. A nominal x is **repeating in** F if its ancestor chain is repeating in F. We write $\mathcal{R}F$ for the set of all nominals that are repeating in F. Note that $\mathcal{R}F$ does not contain initial nominals. We define the **kernel of** F as $\mathcal{K}F := \{\, s \in F \mid \mathcal{N}s \cap \mathcal{R}F = \emptyset \,\}$.

Proposition 12.2

1. $x \notin \mathcal{R}\Gamma \implies \operatorname{depth} x \leq |\mathcal{P}(\operatorname{Mod}\Gamma)|$
2. $\mathcal{L}_F x = \mathcal{L}_F y \iff \mathcal{L}_{\tilde{F}} x = \mathcal{L}_{\tilde{F}} y$
3. $\mathcal{R}F = \mathcal{R}\tilde{F}$

Proof Claim (1) is a straightforward consequence of our definitions. Claim (3) follows from claim (2). To show claim (2), let φ be a normalizer for F. Then $\varphi(\operatorname{Mod} F) = \varphi(\operatorname{Mod} \tilde{F})$. Together with Proposition 3.3 this yields claim (2). ■

A formula $\Diamond\rho s x$ **is chain-evident in a set** F if there exist x' and y such that $x \sim_F x'$, $x' \notin \mathcal{R}F$, $x' \prec y$, and $\rho x' y$, $sy \in F$. Note that chain-evidence is not a monotone property.

Proposition 12.3 (Compatibility) Let $\Diamond\rho s x \approx_F \Diamond\rho s' x'$. Then $\Diamond\rho s x$ is chain-evident in $\tilde{F}$ if and only if $\Diamond\rho s' x'$ is chain-evident in $\tilde{F}$.

A set F is **chain-evident** if it is edge-positive, $\mathcal{K}F$ contains at least one nominal, and F satisfies all the conditions for evident sets except the ones for $\exists$- and $\Diamond$-formulas, for which F satisfies the following conditions:

1. $\exists s \in \mathcal{K}F \implies \exists x\colon\ sx \in \mathcal{K}F$
2. $\Diamond\rho s x \in \mathcal{K}F \implies \Diamond\rho s x$ chain-evident in F

Lemma 12.4 Let F be chain-evident and $\mathcal{L}_F y = \mathcal{L}_F z$. Then $\rho x y$ is safe in F if and only if $\rho x z$ is safe in F.

Proof Let rxy be safe in F. We show that rxz is safe in F. Let $\Box r s x \in F$. Since rxy is safe in F and F is box-evident, $sy \in F$. Since $s \in \mathcal{L}_F y = \mathcal{L}_F z$, $sz \in \tilde{F} = F$. Now let $\Box r^{-} s z \in F$. Since $\Box r^{-} s \in \mathcal{L}_F z = \mathcal{L}_F y$, we have $\Box r^{-} s y \in \tilde{F} = F$. Hence $sx \in F$ since rxy is safe in F and F is box-evident.

Let ryx be safe in F. Then we have to show that rzx is safe in F. This follows with the same arguments as above. ■

Lemma 12.5 If F is chain-evident, then $\mathcal{K}F$ is quasi-evident.

$$\mathcal{R}^c_\exists \ \frac{\exists s}{sx} \ x \notin \mathcal{N}\Gamma \land x \text{ initial} \land \neg\exists x\colon\ x \text{ initial} \ \land\ sx \in \tilde{\Gamma}$$

$$\mathcal{R}^c_\Diamond \ \frac{\Diamond\rho sx}{\rho x'y,\ sy} \ \Diamond\rho sx \text{ not chain-evident in } \tilde{\Gamma} \land x \sim_\Gamma x' \land x' \notin \mathcal{R}\Gamma \land y \notin \mathcal{N}\Gamma \land x' {\prec} y$$

Figure 3: New Rules for $\mathcal{T}^c$

Proof Let F be chain-evident. Then $\mathcal{K}F$ contains at least a nominal. Moreover, $\mathcal{K}F \subseteq F$ is locally consistent since F is locally consistent. To show that $\mathcal{K}F$ is equationally expanded, let φ be a normalizer for $\mathcal{K}F$, $\varphi s = \varphi t$, and $s \in \mathcal{K}F$. It suffices to show that $t \in \mathcal{K}F$. Since F is equationally expanded and $\mathcal{K}F \subseteq F$, we have $t \in F$. Since $Dom\ \varphi \cup Ran\ \varphi \subseteq \mathcal{N}(\mathcal{K}F)$, we have $\mathcal{N}t \subseteq \mathcal{N}(\mathcal{K}F)$. Hence $t \in \mathcal{K}F$.

Let $t \in \mathcal{K}F$ be a non-literal formula. It remains to show that t is either evident or quasi-evident in $\mathcal{K}F$. If t is neither an $\exists$- nor a $\Diamond$-formula, then t is evident in F and hence is t evident in $\mathcal{K}F$. If t is an $\exists$-formula, the chain-evidence of F ensures that t is evident in $\mathcal{K}F$.

Otherwise, let $t = \Diamond\rho sx$. We show that t is quasi-evident in $\mathcal{K}F$. Since F is chain-evident, t is chain-evident in F. Hence there are x' and y such that $x \sim_F x'$, $x' \notin \mathcal{R}F$, $x' \prec y$, and $\rho x'y, sy \in F$. Then $\rho xy \in F$ since $x \sim_F x'$. If $y \notin \mathcal{R}F$, then t is evident in $\mathcal{K}F$ and hence quasi-evident in $\mathcal{K}F$. Otherwise, $y \in \mathcal{R}F$. Since $x' \prec y$ and $x' \notin \mathcal{R}F$, there is a $z \notin \mathcal{R}F$ such that $\mathcal{L}_F z = \mathcal{L}_F y$. Since $F = \tilde{F}$, we have $sz \in F$. Hence $sz \in \mathcal{K}F$. Moreover, since $\rho xy \in F$, we know by Lemma 12.4 that ρxz is safe in F. Then ρxz is also safe in $\mathcal{K}F$. Hence t is quasi-evident in $\mathcal{K}F$. ■

We now construct a tableau system $\mathcal{T}^c$ that expands a satisfiable regular clause Γ_0 until a clause Γ is reached such that $\tilde{\Gamma}$ is chain-evident. If Γ_0 contains only initial nominals, we have $\Gamma_0 \subseteq \mathcal{K}\tilde{\Gamma}$, which by Lemma 12.5 and Theorem 10.6 proves that Γ_0 is finitely satisfiable. We define the tableau system $\mathcal{T}^c$ as follows. As clauses $\mathcal{T}^c$ takes all regular clauses Γ such that $\mathcal{K}\tilde{\Gamma}$ contains at least one nominal. The moves of $\mathcal{T}^c$ are defined analogously to $\mathcal{T}$ except that for $\exists$- and $\Diamond$-formulas we now employ the rules $\mathcal{R}^c_\exists$ and $\mathcal{R}^c_\Diamond$ shown in Figure 3. Check that $\mathcal{T}^c$ terminates on the clause in Example 12.1 since after a few additional steps b and c will be modally equivalent.

Proposition 12.6 $\mathcal{T}^c$ is a tableau system.

Lemma 12.7 (Evidence) If Γ is verified in $\mathcal{T}^c$, then $\mathcal{K}\tilde{\Gamma}$ is quasi-evident.

Proof Let Γ be verified in $\mathcal{T}^c$. By Lemma 12.5 it suffices to show that $\tilde{\Gamma}$ is chain-evident. By reuse of the proof of Lemma 7.2, we know that $\tilde{\Gamma}$ satisfies all evidence conditions but possibly the conditions for $\exists$- and $\Diamond$-formulas.

Let $\exists s \in \mathcal{K}\tilde{\Gamma}$. It suffices to show that there is an initial x such that $sx \in \tilde{\Gamma}$. By the assumption we have a formula $\exists s' \in \Gamma$ such that $s' \approx_\Gamma s$. Since $\mathcal{R}^c_\exists$ does not apply to Γ, there there is an initial x such that $sx \in \tilde{\Gamma}$.

Let $\Diamond\rho sx \in \mathcal{K}\tilde{\Gamma}$. It suffices to show that $\Diamond\rho sx$ is chain-evident in $\tilde{\Gamma}$. By the assumption we have $s' \approx_\Gamma s$ and $x' \sim_\Gamma x$ such that $\Diamond\rho s'x' \in \Gamma$. Since $x \notin \mathcal{R}\Gamma$ and $\mathcal{R}^c_\Diamond$ does not apply to Γ, $\Diamond\rho s'x'$ is chain-evident in $\tilde{\Gamma}$. Hence $\Diamond\rho sx$ is chain-evident in $\tilde{\Gamma}$ by Compatibility (Proposition 12.3). ■

We now start with the termination proof for $\mathcal{T}^c$. The **breadth of a nominal in a clause** is defined as $\text{breadth}_\Gamma\, x := |\{\, y \in \mathcal{N}\Gamma \mid x \prec y \,\}|$. A clause Γ is **chain-admissible** if for every nominal $x \in \mathcal{N}\Gamma$ the following conditions are satisfied:

1. $\text{depth}\, x \;\le\; |\mathcal{P}(\text{Mod}\,\Gamma)| + 1$
2. $\text{breadth}_\Gamma\, x \;\le\; |\{\, \Diamond\rho s \in \text{Mod}\,\Gamma \mid \Diamond\rho sx \text{ chain-evident in } \tilde{\Gamma} \,\}| \;\le\; |\text{Mod}\,\Gamma|$

Proposition 12.8 A chain-admissible clause Γ contains at most $I \cdot M^{2^M}$ nominals, where I is the number of initial nominals in Γ and M is the cardinality of $\text{Mod}\,\Gamma$.

Proposition 12.9 Let $\Gamma \to \Delta$ in $\mathcal{T}^c$. Then:

1. $\text{Mod}\,\Gamma = \text{Mod}\,\Delta$.
2. If Γ is chain-admissible, then Δ is chain-admissible.

Proof Claim (1) follows from the fact that the clauses of $\mathcal{T}^c$ are well-quantified. The depth part of claim (2) follows with Proposition 12.2 (1). ■

A formula $\exists s$ is **initially evident** in F if there is an initial nominal x such that $sx \in F$. We redefine the **$\exists$-power** of a clause Γ as the number of $\exists$-formulas in $\text{Sub}\,\Gamma$ that are not not initially evident in $\tilde{\Gamma}$ (cf. §8).

We redefine the **$\Diamond$-power** of a clause Γ as $I \cdot M^{2^M} - N$, where where I is the number of initial nominals in Γ, M is the cardinality of $\text{Mod}\,\Gamma$, and N is the cardinality of $\mathcal{N}\Gamma$. By Proposition 12.8 we know that the $\Diamond$-power of a chain-admissible clause is not negative.

Theorem 12.10 (Termination)
$\mathcal{T}^c$ terminates on chain-admissible clauses.

Proof

1. Only well-quantified and chain-admissible clauses are involved.
2. All rules preserve $\text{Mod}\,\Gamma$.

3. No rule increases the $\exists$-power of a clause, and $\mathcal{R}^c_\exists$ decreases it.
4. No rule but $\mathcal{R}^c_\exists$ increases the $\Diamond$-power of a clause, and $\mathcal{R}^c_\Diamond$ decreases it.
5. All remaining rules decrease the slack of a clause. ■

A clause is **initial** if it contains only initial nominals.

Theorem 12.11 (Correctness) Let Γ be an initial regular clause. Then:

1. $\mathcal{T}^c$ terminates on Γ.
2. If Γ is verifiable in $\mathcal{T}^c$, then Γ is finitely satisfiable.
3. If Γ is refutable in $\mathcal{T}^c$, then Γ is unsatisfiable.

Proof Claim (1) follows by Theorem 12.10 since initial clauses are chain-admissible. Claim (3) follows immediately from the fact that $\mathcal{T}^c$ is a tableau system. To show Claim (2), let $\Gamma \to^* \Delta$ such that Δ is verified in $\mathcal{T}^c$. Then $\Gamma \subseteq \mathcal{K}\tilde{\Delta}$ since Γ is initial. By Lemma 12.7 we know that $\mathcal{K}\tilde{\Delta}$ is quasi-evident. Hence $\mathcal{K}\tilde{\Delta}$ is syntactically satisfiable by Theorem 10.6. Since $\Gamma \subseteq \mathcal{K}\tilde{\Delta}$, Γ is finitely satisfiable. ■

Corollary 12.12 $\mathcal{T}^c$ decides the satisfiability of initial regular clauses.

Corollary 12.13 Regular clauses are finitely satisfiable if they are satisfiable.

Example 12.14 In contrast to our other tableau systems, $\mathcal{T}^c$ is not verification-sound (as defined in §6). This is not in conflict with the Correctness Theorem 12.11 since verification-soundness only fails for non-initial clauses. Consider the unsatisfiable clause

$$\Gamma = \{pa,\ rab,\ pb,\ rbc,\ \Diamond rpc,\ \Box r(\neg p)c\}$$

We assume that a is initial and $a \prec b \prec c$. Then $\mathcal{R}\Gamma = \{b, c\}$ and Γ is chain-evident and verified in $\mathcal{T}^c$. And, as claimed by the Evidence Lemma 12.7, $\mathcal{K}\tilde{\Gamma} = \{pa\}$ is quasi-evident. □

13 Difference Modalities

Adding the difference modalities to basic modal logic is the most elegant way to arrive at modal logic with equality [10]. Informally, we can describe the difference modalities as follows:

- Dpx: at least one individual different from x satisfies p.
- $\bar{D}px$: all individuals different from x satisfy p.

We now extend PLM and our tableau systems with the difference modalities. We add two modal constants $D, \bar{D}\colon (IB)IB$ and define their semantics in terms of the logical constants of PLN:

$$\begin{array}{lll} D & \doteq & \lambda px.\ \exists y.\, x\neq y \wedge py \qquad\qquad \textbf{existential difference} \\ \bar{D} & \doteq & \lambda px.\ \forall y.\, x \doteq y \vee py \qquad\qquad \textbf{universal difference} \end{array}$$

Next, we extend the syntax of PLM (see Figure 1) by adding the new modalities:

$$\mu \ ::= \ \Diamond\rho \ \mid \ \Box\rho \ \mid \ D \ \mid \ \bar{D}$$

Now we define evidence conditions (cf. §5) for the new formulas:

$$\begin{array}{lll} Dsx & \text{evident in } F \text{ if} & \exists y:\ y \not\sim_F x \wedge sy \in F \\ \bar{D}sx & \text{evident in } F \text{ if} & \forall y \in \mathcal{N}F:\ y \not\sim_F x \Longrightarrow sy \in F \end{array}$$

We observe that Soundness (Lemma 5.1), Stability (Propositions 5.2 and 5.5), Compatibility (Proposition 5.3), and Model Existence (Theorem 5.6) also hold for PLM with difference modalities. Now we are ready to extend $\mathcal{T}$ with rules for the difference modalities:

$$\mathcal{R}_D\ \frac{Dsx}{x\neq y,\ sy}\ y \notin \mathcal{N}\Gamma \wedge Dsx \text{ not evident in } \tilde{\Gamma} \qquad \mathcal{R}_{\bar{D}}\ \frac{\bar{D}sx}{x \doteq y \mid sy}\ y \in \mathcal{N}\Gamma \wedge y \not\sim_\Gamma x$$

$\mathcal{R}_D$ is an operational reformulation of the evidence condition for D-formulas. $\mathcal{R}_{\bar{D}}$ is a slight surprise since it is branching. Having only sy as conclusion as suggested by the evidence condition for $\bar{D}$-formulas does not yield sound moves. It is not difficult to verify that Evidence (Lemma 7.2) and hence Verification-Soundness (Theorem 7.3) still hold for $\mathcal{T}$ with difference modalities.

As it comes to termination, the difference modalities behave similar to the ordinary quantifiers and are much easier than the classical modalities $\Diamond$ and $\Box$. We extend the definition of straightness (see §8) with the proviso that every D-formula in $\operatorname{Sub}\Gamma$ is closed. As is the case for $\forall$-formulas, $\bar{D}$-formulas may be open. Extended $\mathcal{T}$ preserves straightness of clauses and still terminates on straight clauses. The termination proof must deal with the complication that evidence of D-formulas may be lost if equations are added (i.e., evidence of D-formulas is not a monotone property). We account for this complication by defining the **D-power** of a clause Γ as the following natural number:

$$|\{\, Ds \in \operatorname{Sub}\Gamma \mid \neg\exists x\colon\ sx \in \tilde{\Gamma} \,\}| \ + \ |\{\, Ds \in \operatorname{Sub}\Gamma \mid \neg\exists x,y\colon\ sx, sy, x\neq y \in \tilde{\Gamma} \,\}|$$

Since for straight clauses $\mathcal{T}$ does not add new terms Ds to $\operatorname{Sub}\Gamma$, no rule increases the D-power of a clause. Moreover, application of $\mathcal{R}_D$ decreases the

D-power of a clause. The interesting case is that $Dsx \in \Gamma$ is not evident in $\tilde{\Gamma}$ but $sx \in \tilde{\Gamma}$. We can assume that $\tilde{\Gamma}$ is locally consistent since $\mathcal{R}_D$ does not apply to refuted clauses. Then there cannot be a y such that $sx, sy, x{\neq}y \in \tilde{\Gamma}$. Application of $\mathcal{R}_D$ will add sy and $x{\neq}y$ and thus yield a clause whose D-power is smaller than the D-power of Γ.

The definition of well-quantifiedness in §9 is extended such that it disallows open formulas of the form Dsx. With this update the results about quasi-evidence and the pattern-based system stay valid.

For the chain-based system, $\mathcal{R}_D$ needs to be modified in the same way $\mathcal{R}_\exists$ was modified before:

$$\mathcal{R}_D^c \ \frac{Dsx}{x{\neq}y,\ sy} \ y \notin \mathcal{N}\Gamma \ \land \ y \text{ initial } \land \ \neg\exists y{:}\ \ y \text{ initial } \land \ y \not\prec_\Gamma x \ \land \ sy \notin \tilde{\Gamma}$$

Accordingly, the definition of D-power must be updated for the termination proof to only consider initial witnesses, that is, be defined as the number

$$|\{\, Ds \in \operatorname{Sub}\Gamma \mid \neg\exists x \in \mathit{Ini}{:}\ \ sx \in \tilde{\Gamma}\,\}| \\ \quad + \ |\{\, Ds \in \operatorname{Sub}\Gamma \mid \neg\exists x, y \in \mathit{Ini}{:}\ \ sx, sy, x{\neq}y \in \tilde{\Gamma}\,\}|$$

where Ini is the set of all initial nominals. With these updates the results about the chain-based system stay valid.

14 Transitive Relations

In many applications the relations used for modal quantification are transitive. Tableau-based decision procedures for transitive modal quantification are well known [19, 14, 18]. It turns out that the usual tableau rule for transitive modal relations [14, 18] also works in our setting. It is enlightening to see what form the correctness arguments for transitive modal relations take in our framework.

We have seen in Example 9.2 that PLM can express that a relation r is transitive. However, transitivity cannot be expressed in regular PLM since regular clauses exclude negated edges. To circumvent the problem, we extend PLM with a logical constant T so that the formula Tr states that r is transitive. We then admit formulas Tr in regular clauses and show how our main results generalize. We define the semantics of $T : (IIB)B$ by reduction to PLN:

$$T \quad \doteq \quad \lambda r.\, \forall x \forall y \forall z.\ \neg rxy \lor \neg ryz \lor rxz$$

We extend the syntax of PLM so that it provides for formulas of the form Tr. Negated formulas $\neg Tr$ are not allowed. The **evidence condition for T-formulas** follows from the definition of T:

$$Tr \quad \text{evident in } F \text{ if} \quad \forall x, y, z \colon\ rxy, ryz \in F \Rightarrow rxz \in F$$

We observe that Soundness (Lemma 5.1), Stability (Propositions 5.2 and 5.5), Compatibility (Proposition 5.3), and Model Existence (Theorem 5.6) also hold for PLM with T-formulas. We now extend $\mathcal{T}$ with the canonical rule for T-formulas:

$$\mathcal{R}_T \ \frac{Tr,\ rxy}{rxz}\ ryz \in \tilde{\Gamma}$$

Evidence (Lemma 7.2), Verification-Soundness (Theorem 7.3), and Straight Termination (Theorem 8.4) still hold for $\mathcal{T}$ with T-formulas.

The definition of regular clauses now allows T-formulas but stays unchanged otherwise. Next the notion of quasi-evidence needs to be generalized. It is here where things really become interesting. Recall that model existence for quasi-evident sets is crucial for both the pattern- and chain-based tableau systems. It turns out that we need a special quasi-evidence condition for T-formulas. As a consequence, we have to replace the tableau $\mathcal{R}_T$ for T-formulas with a new rule $\mathcal{R}_T^q$ based on the quasi-evidence condition.

Example 14.1 Consider the unsatisfiable clause

$$\{Tr,\ \Diamond rpa,\ \Box rpa,\ (\lambda x.px)b,\ pb,\ rbc,\ \neg pc\}$$

which, if we ignore Tr, is quasi-evident with the definition from §10 since rab is a safe edge. The problem shows if we add rab. $\mathcal{R}_T$ then adds rac, $\mathcal{R}_\Box$ adds $(\lambda x.px)c$, and $\mathcal{R}_\lambda$ adds pc, which leaves us with a locally inconsistent clause. $\Box$

The problem is that in the presence of Tr the old notion of safe r-edges is too weak. The solution of the problem builds on the fact that the formula

$$Tr \wedge rxy \wedge \Box rpx \rightarrow \Box rpy$$

holds in all interpretations. Based on this formula we redefine edge safeness as follows. An edge **rxy is safe in F** if $rxy \in F$ or all of the following conditions are satisfied:

1. $\forall t\colon\ \Box rtx \in F \implies ty \in F$
2. $\forall t\colon\ \Box \breve{r}ty \in F \implies tx \in F$
3. $Tr \in F \implies \forall t\colon\ \Box rtx \in F \implies \Box rty \in F$
4. $Tr \in F \implies \forall t\colon\ \Box \breve{r}ty \in F \implies \Box \breve{r}tx \in F$

Conditions (3) and (4) are new. Condition (4) accounts for the fact that $r^{\check{}}$ is transitive if r is transitive. With the new definition, rab is no longer safe for the clause of Example 14.1.

We are not done yet. It is essential that an edge that is an element of a quasi-evident set remains safe if is removed (Proposition 10.2). The set $\{Tr,\ \Box rpa,\ rab,\ (\lambda x.px)b,\ pb\}$ shows that is not the case with our current definition, since rab becomes unsafe after removal. We fix the problem with a **quasi-evidence condition for T-formulas**, which requires the properties (3) and (4) for unrealized safe edges also for realized edges:

$$\begin{array}{llll} Tr & \text{quasi-evident in } F \text{ if} & \forall x,y,t\colon & (rxy,\ \Box rtx \in F \implies \Box rty \in F) \\ & & \wedge & (rxy,\ \Box r^{\check{}}ty \in F \implies \Box r^{\check{}}tx \in F) \end{array}$$

We now define that an edge-positive set is **quasi-evident** if it satisfies all the conditions for evident sets except the ones for diamond and T-formulas, for which it satisfies the respective quasi-evidence conditions. It is not difficult to verify that our previous results for quasi-evidence in §10 still hold. For the verification of Lemma 10.5, the following proposition is essential.

Proposition 14.2 Let F be quasi-evident. If $Tr \in F$ and rxy and ryz are safe in F, then rxz is safe in F.

We now extend the tableau system $\mathcal{T}^q$ with the tableau rule obtained from the quasi-evidence condition for T-formulas:

$$\mathcal{R}_T^q\ \frac{T|\rho|,\ \Box\rho sx}{\Box\rho sy}\ \rho xy \in \tilde{\Gamma}$$

The notation $|\rho|$ stands for r if $\rho = r$ or $\rho = r^{\check{}}$. $\mathcal{R}_T^q$ replaces $\mathcal{R}_T$, which is superfluous. As before, $\mathcal{T}^q$ is verification-sound.

As it comes to termination, neither $\mathcal{R}_T$ nor $\mathcal{R}_T^q$ are critical since they are not generative.

We now come to the pattern-based system. The critical point here is that pattern-evidence must imply quasi-evidence (Propositions 11.2 and 11.3). If we define pattern-evidence with the quasi-evidence condition for T-formulas, everything works out as before.

Finally, we consider the chain-based system. The critical point is that chain-evidence must imply quasi-evidence (Lemmas 12.4 and 12.5). If we define chain-evidence with the quasi-evidence condition for T-formulas, everything works out as before.

15 Conclusion

Let us summarize the contributions of this paper. We start with modal logic without equality.

- *PLM.* Our work is based on classical predicate logic with a touch of simple type theory. This basis provides for ordinary and modal quantification and easily accommodates features such as transitivity and equality.
- *Quasi-evidence.* Quasi-evidence provides for transparent model existence proofs for both the pattern-based and the chain-based system. Quasi-evidence explains how edges can be safely added after the tableau system has done its work. Before, this important technique was buried in monolithic model existence proofs.
- *Pattern-based blocking.* For the converse-free case, pattern-based blocking is an elegant alternative to the established chain-based blocking. It has great promise for efficient implementation since in total at most $|\mathcal{P}(\mathrm{Mod}\,\Gamma_0)|$ diamond expansions are needed, where Γ_0 is the initial clause. In contrast, chain-based blocking achieves the same bound only per ancestor chain.

Now we come to modal logic with equality, which is the real issue of this paper.

- *Abstract congruence closure.* We work with an abstract congruence closure, that is, do not commit to a particular representation of the closure in our tableau systems. This approach yields a transparent treatment of equality, providing a basis for routinely generalizing techniques from the equation-free case. The scalability was demonstrated with chain-based blocking and transitivity.
- *Difference Modalities.* This paper presents the first terminating tableau systems for the notorious difference modality. Given our approach, the difference modality is not a big deal. However, it were difficulties with the difference modality that finally led us to work with abstract congruence closures.

Bolander and Blackburn [7] start from basic hybrid logic and then successively add universal and converse modalities. Their strongest system uses chain-based blocking as does our system $\mathcal{T}^c$. For their converse-free system, they use a scheme that blocks diamond-expansion at a nominal x if there is an older nominal y such that $\mathcal{L}x \subseteq \mathcal{L}y$. For their basic system, they don't employ loop-checks but rather restrict the propagation of equational variants to initial nodes. Both termination disciplines can also be obtained with our abstract congruence closure.

We expect that our results extend to symmetric relations and to what is called role hierarchies in description logics [18]. Role hierarchies can be expressed with inclusion formulas $r \subseteq r'$ saying that rxy implies $r'xy$. For inclusion formulas, the notions of safe edge and quasi-evidence need to be adapted.

Another interesting question is whether $\mathcal{T}$ and $\mathcal{T}^q$ are refutation-complete. We expect that this is the case and can be shown using consistency properties [11].

References

[1] Carlos Areces and Balder ten Cate. Hybrid logics. In Blackburn et al. [6], pages 821–868.

[2] Philippe Balbiani and Stéphane Demri. Prefixed tableaux systems for modal logics with enriched languages. In Anca L. Ralescu and James G. Shanahan, editors, *Proc. 15th International Joint Conference on Artificial Intelligence (IJCAI'97)*, pages 190–195. Morgan Kaufmann, 1997.

[3] Christoph E. Benzmüller and Lawrence C. Paulson. Exploring properties of normal multimodal logics in simple type theory with LEO-II. In Christoph E. Benzmüller, Chad E. Brown, Jörg Siekmann, and Richard Statman, editors, *Festschrift in Honor of Peter B. Andrews on His 70th Birthday*, Studies in Logic and the Foundations of Mathematics. IFCoLog. To appear.

[4] Evert W. Beth. Semantic entailment and formal derivability. *Mededelingen der Koninklijke Nederlandse Akademie van Wetenschappen, Afdeling Letterkunde*, 18(13):309–342, 1955.

[5] Patrick Blackburn, Maarten de Rijke, and Yde Venema. *Modal Logic*. Cambridge University Press, 2001.

[6] Patrick Blackburn, Johan van Benthem, and Frank Wolter, editors. *Handbook of Modal Logic*, volume 3 of *Studies in Logic and Practical Reasoning*. Elsevier, 2006.

[7] Thomas Bolander and Patrick Blackburn. Termination for hybrid tableaus. *J. Log. Comput.*, 17(3):517–554, 2007.

[8] Thomas Bolander and Torben Braüner. Tableau-based decision procedures for hybrid logic. *J. Log. Comput.*, 16(6):737–763, 2006.

[9] Egon Börger, Erich Grädel, and Yuri Gurevich. *The Classical Decision Problem*. Springer, 1997.

[10] Maarten de Rijke. The modal logic of inequality. *J. Symb. Log.*, 57(2):566–584, June 1992.

[11] Melvin Fitting. *First-Order Logic and Automated Theorem Proving*. Graduate Texts in Computer Science. Springer, second edition, 1996.

[12] Melvin Fitting. Modal proof theory. In Blackburn et al. [6], pages 85–138.

[13] George Gargov and Valentin Goranko. Modal logic with names. *J. Phil. Log.*, 22:607–636, 1993.

[14] Joseph Y. Halpern and Yoram Moses. A guide to completeness and complexity for modal logics of knowledge and belief. *Artif. Intell.*, 54:319–379, 1992.

[15] Moritz Hardt and Gert Smolka. Higher-order syntax and saturation algorithms for hybrid logic. *Electr. Notes Theor. Comput. Sci.*, 174(6):15–27, 2007.

[16] K. Jaakko J. Hintikka. Form and content in quantification theory. Two papers on symbolic logic. *Acta Philosophica Fennica*, 8:7–55, 1955.

[17] Ian Horrocks, Ullrich Hustadt, Ulrike Sattler, and Renate Schmidt. Computational modal logic. In Blackburn et al. [6], pages 181–245.

[18] Ian Horrocks and Ulrike Sattler. A description logic with transitive and inverse roles and role hierarchies. *J. Log. Comput.*, 9(3):385–410, 1999.

[19] George E. Hughes and Maxwell J. Cresswell. *An Introduction to Modal Logic*. Methuen, 1968.

[20] Mark Kaminski and Gert Smolka. Hybrid tableaux for the difference modality. In *5th Workshop on Methods for Modalities*, 2007.

[21] Mark Kaminski and Gert Smolka. A straightforward saturation-based decision procedure for hybrid logic. In *International Workshop on Hybrid Logic (HyLo 2007)*, 2007.

[22] Mark Kaminski and Gert Smolka. Terminating tableaux for hybrid logic with the difference modality and converse. Technical report, Saarland University, 2008. Submitted.

[23] Saul A. Kripke. A completeness theorem in modal logic. *J. Symb. Log.*, 24(1):1–14, 1959.

[24] Saul A. Kripke. Semantical analysis of modal logic I: Normal modal propositional calculi. *Zeitschrift für Mathematische Logik und Grundlagen der Mathematik*, 9:67–96, 1963.