Terminating Tableau Systems for Hybrid Logic with Difference and Converse

Mark Kaminski and Gert Smolka
Saarland University

October 2008

To appear in Journal of Logic, Language and Information, 2009.

This paper contributes to the principled construction of tableau-based decision procedures for hybrid logic with global, difference, and converse modalities. We also consider reflexive and transitive relations. For converse-free formulas we present a terminating control that does not rely on the usual chain-based blocking scheme. Our tableau systems are based on a new model existence theorem.

1 Introduction

This paper contributes to the principled construction of terminating tableau systems for modal logic with nominals and difference modalities [4, 1, 8, 11]. We also consider global and converse modalities and reflexive and transitive relations. Nominals and difference enrich modal logic with equational constraints, and handling these constraints in a terminating tableau system is the main challenge of this paper. We work with tableau-based decision methods since they may be realized with gracefully degrading performance even if the worst-case complexity of the decision problem is prohibitive. This is witnessed by the success of tableau-based decision procedures for description logics [2, 16], which are modal logics adapted to knowledge representation.

Modal logic with nominals is better known as hybrid logic [1, 4]. With nominals one can say that the current state equals some given state. The difference modalities [12, 8, 11, 4] are the modalities for the complement of the equality relation. With the existential difference modality one can say that there is a state

different from the current state that satisfies a given property. The difference modalities can express nominals and global modalities [11].

The construction of terminating tableau systems for hybrid logic is a recent activity. Bolander and Braüner [7] devise a terminating tableau system for hybrid logic with global modalities. Bolander and Blackburn [6] extend their work to converse modalities. Horrocks and Sattler [18] present a tableau decision procedure for the description logic SHOIQ, which subsumes hybrid logic with global and converse modalities. Balbiani and Demri [3] give a sound and complete tableau system for modal logic with difference. Although claimed, their system does not terminate on all inputs. Our previous work [20, 21], which is updated by the present paper, presents the first terminating tableau systems for hybrid logic with difference, first without [20] and then with [21] converse modalities.

To handle nominals and difference, equational constraints must be treated. We distinguish between the declarative and the procedural approach. The declarative approach used in [7, 6, 21] adds formulas but never deletes formulas (e.g., if $x{=}y$ is known and px is present, py is added). The procedural approach found in [18, 20] replaces formulas so that state variables can be eliminated (e.g., if $x{=}y$ is known and px is present, replace px with py). The procedural approach encompasses algorithmic decisions that are not present in the more abstract declarative approach. Given the high complexity of the correctness arguments, we will follow the simpler and more transparent declarative approach in this paper. A procedural system may then be obtained by refinement.

In our view, a tableau system should be based on a model existence theorem. The independent formulation of such a theorem provides an abstract base for insights and avoids preoccupation with algorithmic details. The closure conditions of the theorem yield the expansion rules of the tableau system. To obtain a decision procedure, a terminating control for the rules is needed. This introduces a design loop since a terminating control will only be possible if the closure conditions of the theorem do not require too much. So in the end, a tableau-based decision procedure is obtained with a suitable model existence theorem and a concomitant terminating control.

We base our tableau systems on a novel model existence theorem whose closure conditions suggest a simple “pattern-based” control that terminates for converse-free formulas. Only for formulas with converse modalities we have to resort to the usual chain-based blocking scheme [22, 17, 16]. The pattern-based control yields a smaller search space than chain-based blocking, and this shows in the performance of a first implementation. Pattern-based control first appeared in our paper [20].

In contrast to previous work [7, 6, 21], the closure conditions of our model existence theorem do not require the presence of equationally entailed formulas.

Instead, they refer to an equational closure whose representation is left open. The equational closure is also used by the tableau rules. This way we obtain simpler correctness proofs and avoid premature algorithmic commitments.

The paper is organized as follows. We first formalize the modal language we consider as a fragment of simple type theory. This way we have an expressive base for our model existence theorem, which we develop in three steps. We start with a syntactic characterization of satisfiability (Herbrand semantics), then obtain a first model existence theorem (evident sets), and then formulate the final model existence theorem (quasi-evident sets), which we prove by reduction to the first one. We then present the concomitant tableau system and show that it terminates for converse-free formulas if we prioritize box propagation. The next two sections present a revised system with chain-based blocking that decides the satisfiability of all formulas of our modal language. Finally, we discuss some design decisions and conclude.

2 Modal Logic in Simple Type Theory

Following [20, 21], we formalize modal logic as a fragment of simple type theory (see [9] to get started). This way we can make use of a rich syntactic and semantic framework and modal logic does not appear as an isolated formal system. We start with a quick review of type theory and then model the linguistic primitives of modal logic as defined constants.

2.1 Types and Terms

Types (σ, τ) are obtained from two base types B and I according to $\sigma ::= B \mid I \mid \sigma\sigma$. The elements of B are the two truth values, and the elements of I are called **individuals**. The elements of a functional type $\sigma\tau$ are the total functions from σ to τ. **Terms** (s, t, u) are obtained from **names** (x, y, z, p, q, r, b) according to $s ::= x \mid \lambda x.s \mid ss$. We assume a **typing relation** $s : \sigma$ satisfying the following properties:

1. For every term s there is at most one type σ such that $s : \sigma$.
2. For every type σ there are infinitely many names x such that $x : \sigma$.
3. For all x, s, σ, τ: $\lambda x.s : \sigma\tau \iff x : \sigma \land s : \tau$.
4. For all s, t, σ: $st : \sigma \iff \exists\tau\colon\ s : \tau\sigma \land t : \tau$.

A term σ is **well-typed** if there is a type σ such that $s : \sigma$. We only consider well-typed terms. We omit parentheses according to $\sigma\tau\rho \rightsquigarrow \sigma(\tau\rho)$ and $stu \rightsquigarrow (st)u$.

2.2 Formulas and Logical Constants

Terms of type B are called **formulas**. The **logical constants** are provided through the following names:

$\bot, \top$	$: B$	false, true
$\neg$	$: BB$	negation
$\wedge, \vee, \rightarrow$	$: BBB$	conjunction, disjunction, implication
$=_\sigma$	$: \sigma\sigma B$	identity
$\forall_\sigma, \exists_\sigma$	$: (\sigma B)B$	universal and existential quantification

We write $\forall x.s$ and $\exists x.s$ for $\forall_\sigma(\lambda x.s)$ and $\exists_\sigma(\lambda x.s)$. As usual, we employ infix notation for the binary logical constants. Parentheses are omitted according to the precedence order $=_\sigma$, $\neg$, $\wedge$, $\vee$, $\rightarrow$, where $=_\sigma$ binds strongest. Formulas of the form $s{=_\sigma}t$ are called **equations** and are usually written without the type index as $s{=}t$. It is well known that the logical constants can be defined with the identities. For instance:

$$\begin{aligned} \top &= ((\lambda x.x) =_B \lambda x.x) \\ \forall_\sigma &= \lambda f.\, f =_{\sigma B} \lambda x.\top \end{aligned}$$

2.3 Basic Modal Logic

The linguistic primitives of modal logic can be expressed with constants that can be defined with the classical logical constants. Figure 2.1 shows the definitions of the **modal constants** we are going to use. We start our explanation with the modal constants $\Box$ and $\Diamond$. They represent higher-order functions of the type $(IIB)(IB)IB$. Their first argument is a binary relation between individuals modeled as a function IIB. Their second argument is a property of individuals modeled as a function IB. Their third argument is an individual. Given these arguments, $\Box$ and $\Diamond$ return a truth value. Informally, the semantics of $\Box$ and $\Diamond$ can be stated as follows:

- $\Diamond rpx$ holds iff there exists an r-successor of x that satisfies p.
- $\Box rpx$ holds iff every r-successor of x satisfies p.

Formally, this is expressed with the equations defining $\Box$ and $\Diamond$ in Figure 2.1.

The expressions of modal logic describe properties of individuals (usually called worlds or states). Hence we represent modal expressions as terms of type IB. The modal constants $\dot{\bot}$, $\dot{\top}$, $\dot{\neg}$, $\dot{\wedge}$, $\dot{\vee}$, $\dot{\rightarrow}$ defined in Figure 2.1 provide lifted versions of the propositional constants. We employ infix notation for the binary modal constants and omit parentheses according to the precedence order

$$
\begin{array}{lll}
\Diamond & = & \lambda rpx.\,\exists y.\,rxy \wedge py \\
\Box & = & \lambda rpx.\,\forall y.\,rxy \rightarrow py \\
\dot{\bot} & = & \lambda x.\bot \\
\dot{\top} & = & \lambda x.\top \\
\dot{\neg} & = & \lambda px.\neg px \\
\dot{\wedge} & = & \lambda pqx.\,px \wedge qx \\
\dot{\vee} & = & \lambda pqx.\,px \vee qx \\
\dot{\rightarrow} & = & \lambda pqx.\,px \rightarrow qx \\
\{\} & = & \lambda bx.b \\
\mathrm{D} & = & \lambda px.\,\exists y.\,\neg x{=}y \wedge py \\
\bar{\mathrm{D}} & = & \lambda px.\,\forall y.\,x{=}y \vee py \\
\mathrm{R} & = & \lambda r.\,\forall x.rxx \\
\mathrm{T} & = & \lambda r.\,\forall xyz.\,rxy \wedge ryz \rightarrow rxz \\
^{-} & = & \lambda rxy.\,ryx
\end{array}
\qquad
\begin{array}{l}
\Diamond : (\mathit{IIB})(\mathit{IB})\mathit{IB} \\
\Box : (\mathit{IIB})(\mathit{IB})\mathit{IB} \\
\dot{\bot} : \mathit{IB} \\
\dot{\top} : \mathit{IB} \\
\dot{\neg} : (\mathit{IB})\mathit{IB} \\
\dot{\wedge} : (\mathit{IB})(\mathit{IB})\mathit{IB} \\
\dot{\vee} : (\mathit{IB})(\mathit{IB})\mathit{IB} \\
\dot{\rightarrow} : (\mathit{IB})(\mathit{IB})\mathit{IB} \\
\{\} : \mathit{BIB} \\
\mathrm{D} : (\mathit{IB})\mathit{IB} \\
\bar{\mathrm{D}} : (\mathit{IB})\mathit{IB} \\
\mathrm{R} : (\mathit{IIB})B \\
\mathrm{T} : (\mathit{IIB})B \\
^{-} : (\mathit{IIB})\mathit{IIB}
\end{array}
$$

Figure 2.1: Modal constants

$\dot{\neg}$, $\dot{\wedge}$, $\dot{\vee}$, $\dot{\rightarrow}$, where $\dot{\neg}$ binds strongest and all modal constants bind stronger than the classical constants. We can now write the equation

$$\Box r(p \mathbin{\dot{\wedge}} q) = \Box rp \mathbin{\dot{\wedge}} \Box rq$$

which happens to be valid. The expressions of basic multimodal logic can be obtained with the grammar

$$t ::= p \mid \dot{\bot} \mid \dot{\top} \mid \dot{\neg} t \mid t \mathbin{\dot{\wedge}} t \mid t \mathbin{\dot{\vee}} t \mid t \mathbin{\dot{\rightarrow}} t \mid \Diamond rt \mid \Box rt$$

where the syntactic variables p and r range over names of type IB and IIB, respectively.

We only consider type-theoretic **interpretations** that satisfy the defining equations for the modal constants (see Figure 2.1). We also require that the logical constants are interpreted as usual and that functional types are interpreted as sets of total functions.

It should now be clear how we obtain modal logic as a fragment of simple type theory. No translation is necessary since modal expressions are directly obtained as terms built with higher-order modal constants. The definition of the modal constants in terms of the classical logical constants generalizes the Kripke semantics of modal logic. The well-known first-order translation of modal

expressions amounts to the fact that for every modal expression t there is a first-order formula s (expressed as a formula of simple type theory) such that the equation $t = \lambda x.s$ is valid. The term $\lambda x.s$ can be obtained by replacing the modal constants in t with their definitions and applying β-reduction, possibly followed by an η-expansion.

2.4 Variables and Nominals

We distinguish between constants and variables. The **constants** are exactly the names that we have introduced as classical logical constants or modal constants (see Figure 2.1). All other names are called **variables**. Variables of type I are called **nominals**.[1] We reserve the following letters for variables of the given types:

$$\begin{aligned} x, y, z &: I \\ p, q &: IB \\ r &: IIB \\ b &: B \end{aligned}$$

We use Nom to denote the set of all nominals.

2.5 Lifting and Global Quantification

We can use the formula $\forall t$ where t is a modal expression to say that all individuals satisfy the property t. To allow such global quantification within the modal syntax, we provide a **lifting operator** $\{\} : BIB$ and extend the syntax for modal expressions with the forms $\{\forall t\}$ and $\{\exists t\}$ (note that $\{s\}$ is notation for the application of $\{\}$ to s). The lifting operator is defined such that the formula $\{b\}x$ holds iff b is true.

2.6 Basic Hybrid Logic

The expressions of basic hybrid logic extend the expressions of basic modal logic with the forms $(=x)$ and $\{tx\}$. The term $(=x)$ represents the property that holds exactly for the individual x, and the term $\{tx\}$ represents the property that holds iff the individual x satisfies the property t. Syntactically, a term $(=x)$ is obtained as the application of the logical constant $=_I$ to a name x, and a term $\{tx\}$ is obtained as the application of the lifting operator $\{\}$ to a formula tx,

[1] Note that we deviate from the familiar terminology used in the introduction of the paper. There variables of type I are called state variables and the singleton predicates $(=x)$ introduced in § 2.6 are called nominals.

which is obtained as the application of a modal expression t to a name x. The usual hybrid logic notation for $\{tx\}$ is $@_x t$. Note that the equation

$$\{tx\} = \{\exists((=x) \mathbin{\dot\wedge} t)\}$$

is valid. Thus the form $\{tx\}$ doesn't add expressivity in a hybrid logic with global quantification.

2.7 Difference

The **difference modalities** are the modal constants D and $\bar{\mathrm{D}}$. Their type is $(IB)IB$ and their semantics can be stated as follows:

- $\mathrm{D}px$ holds iff there is an individual different from x that satisfies p.
- $\bar{\mathrm{D}}px$ holds iff all individuals different from x satisfy p.

Modal logic with difference has modal expressions of the forms $\mathrm{D}t$ and $\bar{\mathrm{D}}t$. The following equations are valid:

$$\begin{aligned}
\dot\neg \mathrm{D}p &= \bar{\mathrm{D}}\dot\neg p \\
\{\exists p\} &= p \mathbin{\dot\vee} \mathrm{D}p \\
\{\forall p\} &= p \mathbin{\dot\wedge} \bar{\mathrm{D}}p \\
(\forall xy.\ px \wedge py \to x{=}y) &= \forall(p \mathbin{\dot\to} \bar{\mathrm{D}}\dot\neg p)
\end{aligned}$$

The first equation says that D and $\bar{\mathrm{D}}$ are dual to each other. The second and third equation say that modal logic with difference can express global quantification. The fourth equation says that modal logic with difference can express that a property holds for at most one individual. Taken together, this means that modal logic with difference subsumes basic hybrid logic with global quantification.

2.8 Converse

The modal constant $^-$: $(IIB)IIB$ called **converse** yields the inverse of a relation. **Modal logic with converse** has modal expressions of the forms $\Diamond r^- t$ and $\Box r^- t$. The new forms can be characterized as follows:

- $\Diamond r^- px$ holds iff there exists an r-predecessor of x that satisfies p.
- $\Box r^- px$ holds iff every r-predecessor of x satisfies p.

2.9 Reflexivity and Transitivity

Reflexivity and transitivity of relations can be expressed with the modal constants R and T, which have the type $(IIB)B$:

- $\mathrm{R}r$ holds iff r is reflexive.
- $\mathrm{T}r$ holds iff r is transitive.

$$
\begin{array}{rcl}
\rho & ::= & r \mid r^{-} \\
t & ::= & p \mid \rho x \mid (=x) \mid \dot{\neg} t \mid t \mathbin{\dot{\wedge}} t \mid t \mathbin{\dot{\vee}} t \mid \{s\} \mid \Diamond \rho t \mid \Box \rho t \mid \mathrm{D} t \mid \bar{\mathrm{D}} t \\
s & ::= & tx \mid \exists t \mid \forall t \mid \mathrm{R} r \mid \mathrm{T} r
\end{array}
$$

x, p, and r range over variables of type I, IB, and IIB, respectively

Figure 3.1: Modal expressions (t) and modal formulas (s)

$$
\begin{array}{rcl@{\qquad}rcl}
\dot{\neg}\dot{\neg} p & = & p & & & \\
\dot{\neg}(p \mathbin{\dot{\wedge}} q) & = & \dot{\neg} p \mathbin{\dot{\vee}} \dot{\neg} q & \dot{\neg}(p \mathbin{\dot{\vee}} q) & = & \dot{\neg} p \mathbin{\dot{\wedge}} \dot{\neg} q \\
\dot{\neg}\Diamond r p & = & \Box r \dot{\neg} p & \dot{\neg}\Box r p & = & \Diamond r \dot{\neg} p \\
\dot{\neg}\mathrm{D} p & = & \bar{\mathrm{D}} \dot{\neg} p & \dot{\neg}\bar{\mathrm{D}} p & = & \mathrm{D} \dot{\neg} p \\
\dot{\neg}\{b\} & = & \{\neg b\} & & & \\
\neg p x & = & (\dot{\neg} p) x & & & \\
\neg\exists p & = & \forall \dot{\neg} p & \neg\forall p & = & \exists \dot{\neg} p \\
r x & = & \Diamond r^{-}(=x) & & & \\
r^{-} x & = & \Diamond r (=x) & & &
\end{array}
$$

Figure 3.2: Negation laws

3 Modal Expressions and Modal Formulas

The grammar in Figure 3.1 defines a class of **modal expressions** and a class of **modal formulas**. We will mainly be concerned with **normal** modal expressions and formulas, which are obtained by restricting the use of $\dot{\neg}$ to expressions of the forms p and $(=x)$. Figure 3.2 shows some valid equations we refer to as **negation laws**. Note the duality between $\dot{\wedge}$ and $\dot{\vee}$, $\Diamond$ and $\Box$, D and $\bar{\mathrm{D}}$, and $\exists$ and $\forall$. With the negation laws modal expressions using negation freely can be translated into normal modal expressions as long as there are no negative occurrences of the formulas $\mathrm{R} r$ and $\mathrm{T} r$. The last two equations are needed to translate modal terms of the form $\dot{\neg}\rho x$. For instance, we obtain $\dot{\neg}\Diamond r(rx) = \Box r(\Box r^{-}\dot{\neg}(=x))$ by applying the negation laws.

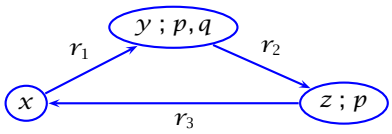
We say that a set of formulas is **satisfiable** if there exists a type-theoretic interpretation that satisfies every formula in the set. We will develop a tableau-based decision procedure that decides the satisfiability of finite sets of normal modal formulas. If the set is satisfiable, the procedure will construct a finite model satisfying it (finite meaning that I is interpreted as a finite set).

4 Herbrand Semantics

Our modal language receives its semantics through the interpretations of simple type theory. We will now define a second semantics, which we call Herbrand semantics since it has much in common with the respective notion for first-order logic. As it turns out, Herbrand semantics provides an excellent foundation for the development of tableau systems. Herbrand semantics also provides a direct connection with the Kripke semantics of modal logic. We will show that the general type-theoretic semantics and the special purpose Herbrand semantics yield the same notion of satisfiability for modal formulas.

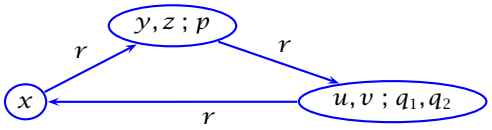
We start with some definitions. The letter A will always denote a set of modal formulas. We use $\mathcal{N}A$ to denote the set of all nominals that occur in at least one formula $s \in A$. A modal formula is **primitive** if it has one of the forms px, rxy, or $x{=}y$. An equation $x{=}y$ is trivial if the nominals x and y are identical. A set A is **straight** if every equation $s \in A$ is trivial. We write $x{\neq}y$ for a formula $(\dot{\neg}(=x))y$. This is justified since the equation $(\dot{\neg}(=x))y = (\neg\, x{=}y)$ is valid.

A **base** is a set H of primitive modal formulas such that $\mathcal{N}H$ is nonempty. A straight base can be seen as a transition system: The nominals in $\mathcal{N}H$ act as states, a formula $px \in H$ gives x the label p, and a formula rxy yields an r-transition from x to y. For instance, the base $\{r_1xy, r_2yz, r_3zx, py, qy, pz\}$ yields the following transition system:



Since straight bases can be seen as transition systems, they can also be seen as Kripke structures.

Let us now consider bases with nontrivial equations. The nontrivial equations of the base yield an equivalence relation on the nominals occurring in the base. We now consider the transition system where the equivalence classes act as states and the formulas px and rxy yield labels and transitions for the respective states. For instance, the base $\{rxy, rzu, rvx, pz, q_1u, q_2v, y{=}z, u{=}v\}$ describes the following transition system:



Note that different non-straight bases may yield the same transition system. Bases that yield the same transition system will turn out to be semantically equivalent.

$$
\begin{aligned}
H \vDash px &\iff \exists x' \colon x' \sim_H x \ \wedge \ px' \in H \\
H \vDash rxy &\iff \exists x', y' \colon x' \sim_H x \ \wedge \ y' \sim_H y \ \wedge \ rx'y' \in H \\
H \vDash x{=}y &\iff x \sim_H y \\
H \vDash (\dot\neg t)x &\iff H \nvDash tx \\
H \vDash (t_1 \mathbin{\dot\wedge} t_2)x &\iff H \vDash t_1 x \ \wedge \ H \vDash t_2 x \\
H \vDash (t_1 \mathbin{\dot\vee} t_2)x &\iff H \vDash t_1 x \ \vee \ H \vDash t_2 x \\
H \vDash r^- xy &\iff H \vDash ryx \\
H \vDash \{s\}x &\iff H \vDash s \\
H \vDash \exists t &\iff \exists x \in \mathcal{N}H \colon H \vDash tx \\
H \vDash \forall t &\iff \forall x \in \mathcal{N}H \colon H \vDash tx \\
H \vDash \Diamond \rho t x &\iff \exists y \in \mathcal{N}H \colon H \vDash \rho xy \ \wedge \ H \vDash ty \\
H \vDash \Box \rho t x &\iff \forall y \in \mathcal{N}H \colon H \vDash \rho xy \implies H \vDash ty \\
H \vDash \mathrm{D}tx &\iff \exists y \in \mathcal{N}H \colon H \vDash x{\neq}y \ \wedge \ H \vDash ty \\
H \vDash \bar{\mathrm{D}}tx &\iff \forall y \in \mathcal{N}H \colon H \vDash x{=}y \ \vee \ H \vDash ty \\
H \vDash \mathrm{R}r &\iff \forall x \in \mathcal{N}H \colon H \vDash rxx \\
H \vDash \mathrm{T}r &\iff \forall x, y, z \in \mathcal{N}H \colon H \vDash rxy \wedge H \vDash ryz \implies H \vDash rxz
\end{aligned}
$$

Figure 4.1: Definition of $H \vDash s$

We use $\sim_A$ to denote the least equivalence relation on Nom such that $x \sim_A y$ for every equation $(x{=}y) \in A$. We refer to $\sim_A$ as the equivalence relation induced by the equations in A.

Given a base H, we can interpret every modal formula s as a statement about the transition system described by H, provided $\mathcal{N}\{s\} \subseteq \mathcal{N}H$. Figure 4.1 provides a recursive definition of the respective **satisfaction relation** $H \vDash s$. The definition is computational in that it yields a model checking algorithm that decides $H \vDash s$ for finite H. The reader familiar with modal logic will notice that for straight H and ordinary modal formulas s the definition of $H \vDash s$ agrees with the standard Kripke semantics.

It remains to make the connection with the type-theoretic semantics. A (type-theoretic) interpretation $\mathcal{I}$ **agrees with a base** H if the following holds:

1. $\mathcal{I}I = \{\, \mathcal{I}x \mid x \in \mathcal{N}H \,\}$.
2. $\mathcal{I} \vDash s \iff H \vDash s$ for every primitive modal formula s such that $\mathcal{N}\{s\} \subseteq \mathcal{N}H$.

Proposition 4.1 For every base there exists an interpretation that agrees with it.

Proof Let H be a base. For every $\sim_A$ equivalence class we choose a representative. We then choose an interpretation $\mathcal{I}$ such that $\mathcal{I}I$ is the set of all representatives that are in $\mathcal{N}H$ and $\mathcal{I}x$ yields the representative of x if $x \in \mathcal{N}H$. Moreover, we require that $\mathcal{I}$ interprets all variables p and r such that $\mathcal{I}$ agrees with H. ∎

Proposition 4.2 Let $\mathcal{I}$ be an interpretation and X be a set of nominals such that $\mathcal{I}I = \{\mathcal{I}x \mid x \in X\}$. Then the following holds for every term t of type IB:

1. $\mathcal{I} \vDash \exists t \iff \exists x \in X\colon\ \mathcal{I} \vDash tx$
2. $\mathcal{I} \vDash \forall t \iff \forall x \in X\colon\ \mathcal{I} \vDash tx$

Proposition 4.3 Let the interpretation $\mathcal{I}$ agree with the base H. Then $\mathcal{I} \vDash s \Leftrightarrow H \vDash s$ for all modal formulas s such that $\mathcal{N}\{s\} \subseteq \mathcal{N}H$.

Proof By induction on the size of s. We show the reasoning for diamond formulas. Let $s = \Diamond \rho t x$ such that $\mathcal{N}\{s\} \subseteq \mathcal{N}H$. Then:

	$\mathcal{I} \vDash \Diamond \rho t x$	
$\iff$	$\mathcal{I} \vDash \exists y.\, \rho x y \wedge t y$	definition of $\Diamond$, $y \notin \mathcal{N}t \cup \{x\}$
$\iff$	$\exists y \in \mathcal{N}H\colon\ \mathcal{I} \vDash (\lambda y.\, \rho x y \wedge t y)y$	Proposition 4.2
$\iff$	$\exists y \in \mathcal{N}H\colon\ \mathcal{I} \vDash \rho x y \wedge t y$	β-law
$\iff$	$\exists y \in \mathcal{N}H\colon\ \mathcal{I} \vDash \rho x y \ \wedge\ \mathcal{I} \vDash t y$	
$\iff$	$\exists y \in \mathcal{N}H\colon\ H \vDash \rho x y \ \wedge\ H \vDash t y$	induction hypothesis
$\iff$	$H \vDash \Diamond \rho t x$	definition of $H\vDash$

∎

Let H be a base and A be a set of modal formulas. We say that H is a **Herbrand model** of A if $\mathcal{N}A \subseteq \mathcal{N}H$ and H satisfies every formula $s \in A$. Moreover, we say that A is **(finitely) Herbrand satisfiable** if it has a (finite) Herbrand model.

Proposition 4.4 If a set of formulas is (finitely) Herbrand satisfiable, then it is (finitely) satisfiable.

Proof Follows with Propositions 4.1 and 4.3. ∎

We will eventually show that a finite set of normal modal formulas is satisfiable if it is finitely Herbrand satisfiable.

5 Evident Sets

The definition of $H \vDash s$ is such that the satisfaction of larger formulas depends on the satisfaction of smaller formulas. This suggests a completion process that

constructs a Herbrand model by recursively adding smaller formulas that are required for the satisfaction of larger formulas. The smaller formulas can be chosen such that satisfiable sets stay satisfiable and unsatisfiable sets stay unsatisfiable. If the initial set is satisfiable, the process will lead to a self-justifying set where larger formulas are justified by smaller formulas and the primitive formulas constitute a Herbrand model. Self-justifying sets where first explored by Jaakko Hintikka [15] for pure first-order logic, and it is common to call them Hintikka sets. We will refer to our basic version of self-justifying sets as evident sets.

We define the **base H_A of a set A** as follows:

$$H_A := \{ s \in A \mid s \text{ primitive} \} \cup \{ x{=}x \mid x \in \mathcal{N}A \}$$

The trivial equations are added so that $\mathcal{N}H_A = \mathcal{N}A$ holds. The definition of evidence will be such that the base of an evident set is a Herbrand model of the set.

The presence of equality (through $(=x)$ and the difference modalities) complicates the definition of evidence. We start by defining the **equational closure $\tilde{A}$** of a set A of modal formulas as follows:

$$\begin{aligned}\tilde{A} := A &\cup \{ tx \mid \exists x' \colon x' {\sim_A} x \wedge tx' \in A \} \\ &\cup \{ rxy \mid \exists x', y' \colon x' {\sim_A} x \wedge y' {\sim_A} y \wedge rx'y' \in A \}\end{aligned}$$

Note that the closure adds formulas that can be deduced with the equations in A. Also note that $\mathcal{N}A = \mathcal{N}\tilde{A}$ and that $\tilde{A}$ is finite if A is finite. Moreover, $\tilde{A} = A$ if A is straight. It is possible to define the closure smaller or larger than we do it here (cf. § 12). However, the following property must be maintained.

Proposition 5.1 Let s be a primitive formula of the form px or rxy and let A be a set of modal formulas. Then $H_A \vDash s \iff s \in \tilde{A}$.

We define the notation $|\rho xy|$ as follows: $|rxy| = rxy$ and $|r^{-}xy| = ryx$.

An **evident set** is a set A of normal modal formulas that contains at least one nominal and satisfies the **evidence conditions** listed in Figure 5.1. The evidence conditions are derived from the equivalences defining the satisfaction relation $H \vDash s$ (see Figure 4.1).

Theorem 5.2 If A is an evident set, then H_A is a Herbrand model of $\tilde{A}$.

Proof Let A be an evident set. We show by induction on the size of s that $H_A \vDash s$ for all $s \in \tilde{A}$. We show the reasoning for diamond formulas. Let $\Diamond \rho tx \in \tilde{A}$. Then

$$x {\sim_A} x' \text{ and } \Diamond \rho tx' \in A$$

$$
\begin{aligned}
(\dot\neg p)x \in A &\implies px \notin \tilde A\\
x \neq y \in A &\implies x \not\to_A y\\
(t_1 \mathbin{\dot\wedge} t_2)x \in A &\implies t_1 x \in \tilde A \ \wedge\ t_2 x \in \tilde A\\
(t_1 \mathbin{\dot\vee} t_2)x \in A &\implies t_1 x \in \tilde A \ \vee\ t_2 x \in \tilde A\\
r^- xy \in A &\implies ryx \in \tilde A\\
\{s\}x \in A &\implies s \in \tilde A\\
\exists t \in A &\implies \exists x \in \mathcal{N}A\colon\ tx \in \tilde A\\
\forall t \in A &\implies \forall x \in \mathcal{N}A\colon\ tx \in \tilde A\\
\Diamond \rho t x \in A &\implies \exists y \in \mathcal{N}A\colon\ |\rho xy| \in \tilde A \ \wedge\ ty \in \tilde A\\
\Box \rho t x \in A &\implies \forall y \in \mathcal{N}A\colon\ |\rho xy| \in \tilde A \implies ty \in \tilde A\\
\mathrm{D}tx \in A &\implies \exists y \in \mathcal{N}A\colon\ x \not\to_A y \ \wedge\ ty \in \tilde A\\
\bar{\mathrm{D}}tx \in A &\implies \forall y \in \mathcal{N}A\colon\ x \sim_A y \ \vee\ ty \in \tilde A\\
\mathrm{R}r \in A &\implies \forall x \in \mathcal{N}A\colon\ rxx \in \tilde A\\
\mathrm{T}r \in A &\implies \forall x,y,z \in \mathcal{N}A\colon\ rxy \in \tilde A \wedge ryz \in \tilde A \implies rxz \in \tilde A
\end{aligned}
$$

Figure 5.1: Evidence conditions

for some x'. Since A is evident, we have

$$y \in \mathcal{N}A \text{ and } |\rho x' y| \in \tilde A \text{ and } ty \in \tilde A$$

for some y. Thus $|\rho xy| \in \tilde A$. By the induction hypothesis we obtain

$$H_A \vDash |\rho xy| \text{ and } H_A \vDash ty$$

Since $\mathcal{N}H_A = \mathcal{N}A$, we have $H_A \vDash \Diamond \rho t x$. ∎

6 Quasi-Evident Sets

The terminating tableau systems we are aiming at require a stronger model existence theorem than the one we just established for evident sets. This theorem will assert the satisfiability of a class of quasi-evident sets, which is obtained by weakening the closure conditions for diamond formulas. Moreover, the closure condition for formulas $\mathrm{T}r$ is modified. Quasi-evident sets can always be made evident by adding so-called safe edges. **Edges** are formulas of the form rxy. To get the idea behind safe edges, consider the evident set $\{\Box rpx, py\}$. We can add the edge rxy without destroying evidence. The notions of quasi-evidence and safe edges will be such that

1. a quasi-evident set remains quasi-evident if a safe edge is added.
2. a quasi-evident set that contains all safe edges is evident.

Let us first explain quasi-evidence for sets that contain neither converse modalities nor T-formulas (i.e., $\mathrm{T}r$). In this case we call an edge rxy safe in A if either $rxy \in \tilde{A}$ or the names r, x, y occur in A and $ty \in \tilde{A}$ for all $\Box rtx \in \tilde{A}$. Quasi-evidence is defined like evidence, except that the condition for diamond-formulas is weakened to

$$\Diamond rtx \in A \implies \exists y \in \mathcal{N}A\colon\ ty \in \tilde{A} \ \wedge\ rxy \text{ safe in } A$$

It is now easy to verify that a quasi-evident set remains quasi-evident if safe edges are added (recall that we assume the absence of converse modalities and T-formulas), and that a quasi-evident set that contains all its safe edges is evident.

It is easy to extend the definition of safe edges to converse modalities. What is more difficult is the extension to transitive relations. The evidence condition for T-formulas requires that certain edges are present, which conflicts with the addition of safe edges. For instance, consider $A = \{\Box rpx,\ py,\ ryz,\ \mathrm{T}r\}$. According to what we said so far, this set is quasi-evident and rxy is safe. However, quasi-evidence is lost once we add rxy since the evidence condition for $\mathrm{T}r$ requires the presence of rxz, which is neither present nor safe.

We solve the problem by modifying the evidence condition for T-formulas so that it requires the presence of box formulas rather than edges. The idea is that once the box formulas are present the edges needed for transitivity can be added as safe edges. The validity of the following formulas justifies the addition of the necessary box formulas:

- $\mathrm{T}r \wedge \Box rpx \wedge rxy \to \Box rpy$
- $\mathrm{T}r \wedge \Box r^- px \wedge ryx \to \Box rpy$

We are now ready for the final definition of safe edges. An edge rxy **is safe in** A if either $rxy \in \tilde{A}$ or the following holds:

1. The names r, x, y occur in A.
2. For all $\Box rtx \in \tilde{A}$: $\ ty \in \tilde{A} \ \wedge\ (\mathrm{T}r \in A \implies \Box rty \in \tilde{A})$.
3. For all $\Box r^- ty \in \tilde{A}$: $\ tx \in \tilde{A} \ \wedge\ (\mathrm{T}r \in A \implies \Box r^- tx \in \tilde{A})$.

A **quasi-evident set** is a set A of normal modal formulas such that:

1. $\mathcal{N}A$ is nonempty.
2. A satisfies the **quasi-evidence conditions** in Figure 6.1.
3. A satisfies the evidence conditions in Figure 5.1 except those that apply to formulas of the forms $\Diamond \rho tx$ and $\mathrm{T}r$.

An evident set not containing T-formulas is always quasi-evident. However, evident sets with T-formulas may fail to be quasi-evident since they may lack box

$$\Diamond \rho t x \in A \implies \exists y\colon\ t y \in \tilde{A} \ \wedge\ |\rho x y| \text{ safe in } A$$
$$\mathrm{T} r \in A \implies \forall x, y, t\colon\ r x y \in \tilde{A} \ \wedge\ \Box r t x \in \tilde{A} \implies \Box r t y \in \tilde{A}$$
$$\mathrm{T} r \in A \implies \forall x, y, t\colon\ r x y \in \tilde{A} \ \wedge\ \Box r^{-} t y \in \tilde{A} \implies \Box r^{-} t x \in \tilde{A}$$

Figure 6.1: Quasi-evidence conditions

formulas that are required by the quasi-evidence conditions for T-formulas.

Example 6.1 Let $A = \{r x y,\ p y,\ \Box r p x,\ \mathrm{T} r\}$. Then A is evident but not quasi-evident. Note that the edges $r y y$ and $r y x$ are safe in A, and that the edge $r x x$ is not safe in A. □

Example 6.2 The set $\{\Diamond r p x,\ p y,\ \Diamond r q y,\ q z,\ \mathrm{T} r\}$ is quasi-evident. Since there are no box formulas, all r-edges between x, y, and z are safe. □

Proposition 6.3 Let A be a quasi-evident set such that $\{\mathrm{T} r,\ r x y,\ r y z\} \subseteq \tilde{A}$. Then $r x z$ is safe in A.

Lemma 6.4 (Safe Edges) Let A be a quasi-evident set and E be the set of all edges that are safe in A. Then:

1. $r x y$ safe in $A \iff r x y \in A \cup E \iff r x y$ safe in $A \cup E$
2. $A \cup E$ is quasi-evident.
3. $A \cup E$ is evident.

Proof Claim (1) is easy to verify. For (2) we have to show that $A \cup E$ satisfies the evidence conditions for box formulas and the quasi-evidence conditions for $\Diamond$- and T-formulas. For $\Box$- and T-formulas this follows from the definition of safe edges and the quasi-evidence of A. For $\Diamond$-formulas the claim follows from the quasi-evidence of A since we know by (1) that edges that are safe in A are safe in $A \cup E$.

For (3) we have to show that $A \cup E$ satisfies the evidence conditions for $\Diamond$- and T-formulas. This suffices since by (2) we know that $A \cup E$ is quasi-evident. Since $A \cup E$ satisfies the quasi-evidence conditions for $\Diamond$-formulas and we know by (1) that $A \cup E$ contains all edges that are safe in $A \cup E$, $A \cup E$ satisfies the evidence conditions for $\Diamond$-formulas. That $A \cup E$ satisfies the evidence condition for T-formulas follows by Proposition 6.3 and (1). ■

Theorem 6.5 (Model Existence) Every (finite) quasi-evident set is (finitely) Herbrand satisfiable.

Proof Let A be a quasi-evident set. By Lemma 6.4 we obtain an evident set A' such that $A \subseteq A'$. Now the claim follows with Theorem 5.2. For the finiteness claim it suffices to show that a finite A has only finitely many safe edges. This is the case since safe edges contain only names that occur in A. ∎

7 A Tableau System Based on Quasi-Evidence

The notion of quasi-evidence yields a tableau system. The main intuition is that the tableau rules add formulas to a satisfiable set so that it eventually becomes quasi-evident. We start with some general definitions to put the notion of a tableau system on a firm ground.

A **clause** is a finite set A of normal modal formulas such that $\mathcal{N}A \neq \emptyset$. Think of a clause as the set of formulas that are on a branch of a tableau. A **proof step** is a tuple $\langle A_1, \dots, A_n \rangle$ of clauses such that $n \geq 1$ and $\tilde{A}_1 \subsetneq \tilde{A}_i$ for all $i \in [2, n]$; we call A_1 the **head** and $A_2, \dots, A_n$ the **alternatives** of the proof step. A proof step is **sound** if the head is satisfiable if and only if one of the alternatives is satisfiable. A proof step is **refuting** if it has no alternatives ($n = 1$) and **branching** if it has at least two alternatives ($n \geq 3$). Note that the head of a refuting proof step is unsatisfiable.

A **tableau system** is a set of sound proof steps. The **expansion relation** of a tableau system T is defined as follows:

$$A \to_T A' \;:\Longleftrightarrow\; \langle A \rangle \notin T \;\wedge\; \exists \langle A_1, \dots, A_n \rangle \in T\colon\; A = A_1 \;\wedge\; \exists i \in [2, n]\colon\; A' = A_i$$

Note that $A \subsetneq A'$ and $\tilde{A} \subsetneq \tilde{A}'$ if $A \to_T A'$. A clause A is **refuted in** T if $\langle A \rangle \in T$. A refuted clause corresponds to a closed branch. A clause A is **terminal in** T if there is no clause A' such that $A \to_T A'$. A clause is **verified in** T if it is terminal and not refuted in T. Note that a clause that is terminal in T is either refuted or verified in T. A clause A is **verifiable in** T if there exists a clause A' such that $A \to_T^* A'$ and A' is verified in T. Let T be a tableau system and S be a set of clauses. We say that

- T is **verification sound for** S if every clause $A \in S$ that is verified in T is satisfiable.
- T is **verification complete for** S if every satisfiable clause $A \in S$ is verifiable in T.
- T **respects** S if $A' \in S$ for all $A \in S$ and all A' such that $A \to_T A'$.
- T **terminates on** S if the expansion relation of T terminates on every clause $A \in S$.

Proposition 7.1 Let S be a set of clauses and T be a tableau system. If T respects S, terminates on S, and is verification sound for S, then T is verification

$$\mathcal{R}_{\dot\neg}\ \frac{(\dot\neg p)x}{\emptyset}\ \ px \in \tilde{A} \qquad \mathcal{R}_{\dot\neg}\ \frac{x \neq y}{\emptyset}\ \ x \sim_A y$$

$$\mathcal{R}_{\dot\wedge}\ \frac{(t_1 \mathbin{\dot\wedge} t_2)x}{t_1 x\,,\ t_2 x} \qquad \mathcal{R}_{\dot\vee}\ \frac{(t_1 \mathbin{\dot\vee} t_2)x}{t_1 x \mid t_2 x}$$

$$\mathcal{R}_{\breve{}}\ \frac{r^{\breve{}} xy}{ryx} \qquad \mathcal{R}_{\{\}}\ \frac{\{s\}x}{s}$$

$$\mathcal{R}_{\exists}\ \frac{\exists t}{tx}\ \ x \notin \mathcal{N}A \text{ and } \exists t \text{ not evident in } A \qquad \mathcal{R}_{\forall}\ \frac{\forall t}{tx}\ \ x \in \mathcal{N}A$$

$$\mathcal{R}_{\Diamond}\ \frac{\Diamond \rho t x}{|\rho x y|\,,\ t y}\ \ y \notin \mathcal{N}A \text{ and } \Diamond \rho t x \text{ not quasi-evident in } A \qquad \mathcal{R}_{\Box}\ \frac{\Box \rho t x}{t y}\ \ |\rho x y| \in \tilde{A}$$

$$\mathcal{R}_{\mathbf{D}}\ \frac{\mathrm{D}tx}{x \neq y\,,\ ty}\ \ y \notin \mathcal{N}A \text{ and } \mathrm{D}tx \text{ not evident in } A \qquad \mathcal{R}_{\bar{\mathbf{D}}}\ \frac{\bar{\mathrm{D}}tx}{x = y \mid ty}\ \ y \in \mathcal{N}A$$

$$\mathcal{R}_{\mathbf{R}}\ \frac{\mathrm{R}r}{rxx}\ \ x \in \mathcal{N}A$$

$$\mathcal{R}_{\mathbf{T}}\ \frac{\mathrm{T}r}{\Box r t y}\ \ rxy \in \tilde{A} \text{ and } \Box r t x \in \tilde{A} \qquad \mathcal{R}_{\mathbf{T}}\ \frac{\mathrm{T}r}{\Box r^{\breve{}} t x}\ \ rxy \in \tilde{A} \text{ and } \Box r^{\breve{}} t y \in \tilde{A}$$

Figure 7.1: Tableau system $\mathcal{T}$

complete for S and yields a decision procedure for the satisfiability of the clauses in S.

We are now ready to define the quasi-evidence-based tableau system $\mathcal{T}$. The proof steps of the system are described by the rules in Figure 7.1. The formula above the horizontal line of a rule must appear in the head of the proof step, and the formulas below the line are added to the head to obtain the alternatives. The letter A in the side conditions always refers to the head of the proof step. Refuting rules have an "$\emptyset$" below the rule, and branching rules separate their alternatives with "$|$". We say that a rule **applies** to a clause A if the rule yields a proof step whose head is A.

Note that the tableau rules correspond closely to the conditions defining quasi-evidence. We call the rules $\mathcal{R}_{\exists}$, $\mathcal{R}_{\Diamond}$, and $\mathcal{R}_{\mathrm{D}}$ **generative** since they introduce fresh nominals. Besides a **freshness constraint** (e.g., $x \notin \mathcal{N}A$) needed

for soundness, the generative rules come with a **blocking constraint** (e.g., $\exists t$ not evident in A). The blocking constraint prevents unnecessary applications since it requires that the formula the rule is applied to doesn't already satisfy the respective evidence or quasi-evidence condition. For the other non-refuting rules this holds without an explicit blocking constraint since every proof step must produce alternatives whose closure is larger than the closure of the head.

Let us be precise about the meaning of the formulations "s is not evident in A" or "s is not quasi-evident in A" in the blocking constraints of the generative rules. The formulations refer to the evidence condition that applies to the formula s. For instance, "$\exists t$ evident in A" means that there is a nominal $x \in \mathcal{N}A$ such that $tx \in \tilde{A}$. Moreover, "$\Diamond\rho tx$ quasi-evident in A" means that there is a nominal $y \in \mathcal{N}A$ such that $ty \in \tilde{A}$ and $|\rho xy|$ is safe in A. Note that a diamond formula that is evident in A is also quasi-evident in A.

Proposition 7.2 The proof steps obtained with the rules in Figure 7.1 are sound.

Proposition 7.3 A clause is verified in $\mathcal{T}$ iff it is quasi-evident.

Proposition 7.4 Every clause that is verifiable in $\mathcal{T}$ is finitely Herbrand satisfiable.

Proof Follows with Proposition 7.3 and Theorem 6.5. ∎

Example 7.5 The blocking constraint of the rule $\mathcal{R}_\Diamond$ prevents superfluous applications. Here is an example where a verified clause is reached after one expansion step.

$\forall(\Diamond rp),\ px$	initial clause
$\Diamond rpx$	$\mathcal{R}_\forall$

Note that $\mathcal{R}_\Diamond$ does not apply since rxx is safe in the final clause. If the blocking constraint of $\mathcal{R}_\Diamond$ would be changed from quasi-evident to evident, the modified rule would be applicable and the clause would not be verifiable in the modified tableau system. □

Example 7.6 Here is a closed tableau that refutes an unsatisfiable clause.

$\mathrm{T}r,\ \Box r(p \dot\wedge q)x,\ \Diamond r(\Diamond r \dot\neg p \dot\vee \Diamond r \dot\neg q)y,\ x{=}y$			initial clause
$ryz,\ (\Diamond r \dot\neg p \dot\vee \Diamond r \dot\neg q)z$			$\mathcal{R}_\Diamond$
$\Box r(p \dot\wedge q)z$			$\mathcal{R}_\mathrm{T}$
$\Diamond r(\dot\neg p)z$	$\mathcal{R}_{\dot\vee}$	$\Diamond r(\dot\neg q)z$	$\mathcal{R}_{\dot\vee}$
$rzu,\ (\dot\neg p)u$	$\mathcal{R}_\Diamond$	$rzu,\ (\dot\neg q)u$	$\mathcal{R}_\Diamond$
$(p \dot\wedge q)u$	$\mathcal{R}_\Box$	$(p \dot\wedge q)u$	$\mathcal{R}_\Box$
$pu,\ qu$	$\mathcal{R}_{\dot\wedge}$	$pu,\ qu$	$\mathcal{R}_{\dot\wedge}$

□

Example 7.7 The following derivation exhibits a satisfiable and converse-free clause for which $\mathcal{T}$ does not terminate.

$\forall(\Diamond rp),\ \forall(\Box rq),\ x{=}x$	initial clause
$\Diamond rpx,\ \Box rqx$	$\mathcal{R}_\forall$
$rxy,\ py$	$\mathcal{R}_\Diamond$
$\Diamond rpy,\ \Box rqy$	$\mathcal{R}_\forall$
$ryz,\ pz$	$\mathcal{R}_\Diamond$
$\ldots$	

The problem is that $\mathcal{R}_\Box$ is not applied. Once $\mathcal{R}_\Box$ is applied, there is a safe edge from the newest nominal z to y or z, and $\mathcal{R}_\Diamond$ does not apply anymore due to the blocking constraint. □

Example 7.8 The following derivation exhibits a satisfiable clause for which $\mathcal{T}$ does not terminate even if $\mathcal{R}_\Box$ is prioritized over $\mathcal{R}_\Diamond$. The non-termination is due to the presence of the converse box expression $\Box r^- p$.

$\forall(\Diamond r(\Box r^- p)),\ x{=}x$	initial clause
$\Diamond r(\Box r^- p)x$	$\mathcal{R}_\forall$
$rxy,\ \Box r^- py$	$\mathcal{R}_\Diamond$
px	$\mathcal{R}_\Box$
$\Diamond r(\Box r^- p)y$	$\mathcal{R}_\forall$
$ryz,\ \Box r^- pz$	$\mathcal{R}_\Diamond$
py	$\mathcal{R}_\Box$
$\ldots$	

The problem is that the expression $\Box r^- p$ renders edges from the most recently introduced nominal z to x and y unsafe since pz is missing.

Note that once py is added, the edge ryy becomes safe. This means that the edge ryz is no longer needed for the quasi-evidence of $\Diamond r(\Box r^- p)y$. In fact, if we delete the formulas that contain z, we obtain a quasi-evident clause that contains the initial clause. Thus the tableau system does construct a quasi-evident clause that contains the initial clause. □

Example 7.9 The following derivation illustrates the application of $\mathcal{R}_{\bar{\mathrm{D}}}$. The derivation constructs an evident set that describes a one-node Herbrand model.

$\Diamond rpx,\ \bar{\mathrm{D}}(\dot{\neg}p)x$	initial clause
$rxy,\ py$	$\mathcal{R}_\Diamond$
$x{=}y$	$\mathcal{R}_{\bar{\mathrm{D}}}$

□

Remark 7.10 (Rule $\mathcal{R}_{\bar{\mathrm{D}}}$) Given that $\bar{\mathrm{D}}tx$ is a universal quantification, it is somewhat surprising that $\mathcal{R}_{\bar{\mathrm{D}}}$ is a branching rule ($\mathcal{R}_\forall$ and $\mathcal{R}_\Box$ are not). It turns out that a non-branching version of $\mathcal{R}_{\bar{\mathrm{D}}}$ is not possible. The natural candidate for a non-branching rule for $\bar{\mathrm{D}}$ would be

$$\frac{\bar{\mathrm{D}}tx}{ty} \;\; (x \neq y) \in \tilde{A}$$

The proof steps obtained with this rule are sound but the resulting tableau system is not verification sound, as the unsatisfiable clause $\{px,\ (\dot{\neg}p)y,\ \bar{\mathrm{D}}px\}$ shows. □

8 Termination for Diamond-Free Clauses

Our tableau system $\mathcal{T}$ terminates for clauses that don't contain modal expressions of the form $\Diamond\rho t$. To show this fact, we look carefully at the type of formulas that are added by the tableau rules. We start with some definitions.

We say that a term **occurs** in a clause A if it occurs as subterm in at least one formula $s \in A$. For instance, the terms occurring in the clause $\{rxy\}$ are the following: r, x, y, rx, rxy. We say that a clause **contains** a term if the term occurs in the clause. A clause is **diamond-free** if it doesn't contain the constant $\Diamond$.

The **height** of a clause is the size of the largest formula that is an element of the clause. The **breadth** of a clause is the number of formulas the clause contains as elements. The **vocabulary** of a clause is the set of all variables x, p, and r that occur in the clause. The **universe** of a clause A is the set of all modal formulas s such that the size of s is at most the height of A and s contains only variables in the vocabulary of A. Note that the vocabulary and the universe of a clause are finite (since clauses are finite). The **slack** of a clause A is the number of formulas s such that s is an element of the universe of A but not of A.

A modal expression is **auxiliary** if it has the form rx, $(=x)$, or $\dot{\neg}(=x)$. A modal formula is **auxiliary** if it has the form tx where t is a modal expression. The **stock** of a clause is the set of all modal expressions and all modal formulas that occur in the clause but are not auxiliary. For an example, consider the clause

$$\{\Diamond rpx,\ \Box r(=x)y,\ \Diamond r(ry)x,\ \exists q,\ y \neq z\}$$

The stock of this clause consists of the terms $\Diamond rp$, p, $\Box r(=x)$, $\Diamond r(ry)$, $\exists q$, and q. The modal expressions $(=x)$, ry, and $\dot{\neg}(=y)$ occur in the clause but are not in the stock. We exclude auxiliary terms from the stock since they may be added by the tableau rules.

We say that a clause A' is obtained from a clause A by **expansion** if $A \to_{\mathcal{T}} A'$.

Proposition 8.1

1. Expansion preserves the height of a clause.
2. Expansion preserves the stock of a clause.
3. Expansion increases the breadth of a clause.
4. Expansion with a non-generative rule
   - preserves the vocabulary of a clause.
   - preserves the universe of a clause.
   - decreases the slack of a clause.

A **infinite derivation** is an infinite sequence $A_1, A_2, \ldots$ of clauses such that $A_1 \to_{\mathcal{T}} A_2 \to_{\mathcal{T}} \cdots$.

Proposition 8.2 Every infinite derivation employs a generative rule.

Proof By contradiction. Suppose there is an infinite derivation that doesn't employ a generative rule. By Proposition 8.1 we know that every step of the derivation decreases the slack of the clause. Contradiction. ∎

Proposition 8.3

1. Expansion preserves evidence of formulas of the form $\exists t$.
2. Expansion with $\mathcal{R}_\exists$ increases the number of evident formulas of the form $\exists t$ in the stock.

To show termination of clauses with D, we need two definitions.

- Dt is **instantiated** in A if there exists a nominal x such that $tx \in \tilde{A}$.
- Dt is **doubly instantiated** in A if there exist nominals x, y such that $\{x{\neq}y, tx, ty\} \subseteq \tilde{A}$.

Proposition 8.4

1. Expansion preserves instantiation and double instantiation of modal expressions of the form Dt.
2. Expansion with $\mathcal{R}_\mathrm{D}$ increases either the number of instantiated or the number of doubly instantiated modal expressions of the form Dt in the stock.

Proposition 8.5 Let c be one of the constants $\dot\wedge$, $\dot\vee$, $\{\}$, $\Diamond$, $\Box$, D, $\bar{\mathrm{D}}$, $\exists$, $\forall$, R, and T. Then c occurs in a clause if and only if it occurs in the stock of the clause. Hence the presence or absence of these constants is preserved by expansion.

Proposition 8.6 Every infinite derivation employs $\mathcal{R}_\Diamond$.

Proof By contradiction. Suppose there is an infinite derivation that doesn't employ $\mathcal{R}_\Diamond$. Since expansion preserves the stock of a clause, we know by the propositions 8.3 and 8.4 that only finitely many steps of the derivation employ $\mathcal{R}_\exists$ or $\mathcal{R}_\mathrm{D}$. Hence there exists an infinite derivation that employs no generative rule. This contradicts Proposition 8.1. ∎

Theorem 8.7 $\mathcal{T}$ terminates on diamond-free clauses.

Proof By contradiction. Suppose there is an infinite derivation that issues from a diamond-free clause. Then we know by Proposition 8.5 that all clauses of the derivation are diamond-free. Hence the derivation doesn't employ $\mathcal{R}_\Diamond$. This contradicts Proposition 8.6. ∎

Corollary 8.8 $\mathcal{T}$ decides the satisfiability of diamond-free clauses.

Proof Follows with Theorem 8.7, Proposition 7.4, and Proposition 4.4. ∎

9 Termination for Converse-Free Clauses

A clause is **converse-free** if it doesn't contain the modal constant $^-$ that yields the inverse of a relation. Example 7.7 provides us with a converse-free clause for which $\mathcal{T}$ does not terminate. Nevertheless, it is easy to obtain termination for converse-free clauses. All we have to do is to prioritize the box-propagating rules $\mathcal{R}_\Box$ and $\mathcal{R}_\mathrm{T}$ over $\mathcal{R}_\Diamond$. This proviso introduces safe edges that prevent superfluous applications of $\mathcal{R}_\Diamond$.

A clause is **box-propagated** if it cannot be expanded with one of the rules $\mathcal{R}_\Box$ or $\mathcal{R}_\mathrm{T}$. We use $\mathcal{R}_\Diamond^p$ to denote the tableau rule that is obtained from $\mathcal{R}_\Diamond$ by imposing "A is box-propagated" as additional constraint, and $\mathcal{T}^p$ to denote the resulting tableau system.

Proposition 9.1 A clause is verified in $\mathcal{T}^p$ if and only if it is verified in $\mathcal{T}$.

A **pattern** is a set $\{\Diamond rt, \Box rt_1, \ldots, \Box rt_n\}$ of modal expressions such that $n \geq 0$. A pattern is **realized** in a clause A if there are nominals x and y such that $\{rxy, ty, \Box rt_1 x, \ldots, \Box rt_n x\} \subseteq \tilde{A}$.

Proposition 9.2 Let A be a box-propagated and converse-free clause. Then a formula $\Diamond rtx \in A$ is quasi-evident in A if the pattern $\{\Diamond rt\} \cup \{\, \Box ru \mid \Box rux \in \tilde{A}\,\}$ is realized in A.

Proof Let $\{\Diamond rt\} \cup \{\, \Box ru \mid \Box rux \in \tilde{A}\,\}$ be realized in A. Then there exist nominals y, z such that $\{ryz, tz\} \cup \{\, \Box ruy \mid \Box rux \in \tilde{A}\,\} \subseteq \tilde{A}$. Since A is box-propagated and converse-free, rxz is safe in A. Hence $\Diamond rtx$ is quasi-evident in A since $tz \in \tilde{A}$. ∎

Proposition 9.3

1. Expansion preserves realization of patterns.
2. Expansion of a converse-free clause with $\mathcal{R}_\Diamond^p$ increases the number of realized patterns that are subsets of the stock of the clause.

Proof The first claim is obvious. The second claim follows with Proposition 9.2. ∎

Theorem 9.4 $\mathcal{T}^p$ terminates on converse-free clauses.

Proof By contradiction. Suppose there is an infinite derivation in $\mathcal{T}^p$ that issues from a converse-free clause. Then we know by Proposition 8.5 that all clauses of the derivation are converse-free. By Proposition 9.3 we know that only finitely many steps of the derivation employ $\mathcal{R}_\Diamond^p$ (since the stock does not change). Hence there exists an infinite derivation in $\mathcal{T}^p$ that doesn't employ $\mathcal{R}_\Diamond^p$. Such a derivation is also a derivation in $\mathcal{T}$ that doesn't employ $\mathcal{R}_\Diamond$. This contradicts Proposition 8.6. ∎

Corollary 9.5 $\mathcal{T}^p$ decides the satisfiability of converse-free clauses.

Proof Follows with Theorem 9.4, Proposition 9.1, Proposition 7.4, and Proposition 4.4. ∎

10 Chain-Based Blocking

To obtain a tableau system that terminates for all clauses, we have to resort to an old idea we call **chain-based blocking**. Chain-based blocking first appears in Kripke [22] with a terminating tableau system deciding 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 [17] adapt chain-based blocking to a modal logic with converse, and Bolander and Blackburn [6] use it for a hybrid logic with converse (they refer to chain-based blocking as *loop-checks*).

One characteristic feature of tableau systems that employ chain-based blocking is *overgeneration*. Overgeneration means that the system may stop with a final clause that is neither refuted nor quasi-evident. In this case the final clause contains a quasi-evident clause that contains the initial clause. This suffices for the Herbrand satisfiability of the initial clause.

We use Example 7.8 to motivate chain-based blocking. The example gives a clause with converse for which $\mathcal{T}$ and $\mathcal{T}^p$ do not terminate, and it shows that after a few steps a clause is reached that contains a quasi-evident subclause that contains the initial clause.

Recall that the stock of a clause is finite and is preserved by expansion (see § 8). We use **Stk** A to denote the stock of a clause A. Based on the stock of a clause we define the following sets:

$$\begin{aligned} \mathrm{Lab}\, A &:= \{\, \Box\rho t \mid \Box\rho t \in \mathrm{Stk}\, A \,\} \cup \{\, t \mid \exists \rho\colon\ \Diamond\rho t \in \mathrm{Stk}\, A \vee \Box\rho t \in \mathrm{Stk}\, A \,\} \\ \mathcal{L}_A x &:= \{\, t \in \mathrm{Lab}\, A \mid tx \in \tilde{A} \,\} \end{aligned}$$

We refer to the modal expressions in $\mathrm{Lab}\, A$ as the **labels of** A and call $\mathcal{L}_A x$ the **label set of** x **in** A. For every clause A the set $\mathrm{Lab}\, A$ is finite and is preserved by expansion (since the stock is preserved). Moreover, if we have a derivation $A_1 \to_{\mathcal{T}} A_2 \to_{\mathcal{T}} \cdots$, all sets $\mathcal{L}_{A_i} x$ are subsets of the initial set $\mathrm{Lab}\, A_1$. We say that two nominals x, y are **modally equivalent** in a clause A if $\mathcal{L}_A x = \mathcal{L}_A y$.

Chain-based blocking records the ancestors of the nominals introduced with the diamond rule $\mathcal{R}_\Diamond$ through a relation $\prec$ such that $x \prec y$ holds if and only if the nominal y was introduced to expand a diamond formula $\Diamond\rho t x$. 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 the expansion of formulas $\Diamond\rho t x$ if the ancestor chain of x is repeating. Since there is only a finite supply of labels, the diamond rule can add nominals only up to a certain ancestor depth. This suffices for termination.

Formally, 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.

Note that the first two conditions require that the graph given by the set of all nominals and the ancestor relation is a forest. 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 ancestor chain of maximal length that ends at x. An ancestor chain is **repeating in** A if it contains two different nominals that are modally equivalent in A. A nominal x is **repeating in** A if its ancestor chain is repeating in A. A nominal x is **relevant in** A if there exists a nominal x' that is not repeating in A such that $x \sim_A x'$. We write $\mathrm{Rep}\, A$ for the set of all nominals that are repeating in A, and $\mathrm{Rel}\, A$ for the set of all nominals that are relevant in A. A formula $s \in A$ is **relevant in** A if it contains only relevant nominals. The **kernel of a clause** A

is the set of relevant formulas in A:

$$\operatorname{Ker} A \;:=\; \{\, s \in A \mid \mathcal{N}\{s\} \subseteq \operatorname{Rel} A \,\}$$

Our final tableau system $\mathcal{T}^c$ employs chain-based blocking and terminates on all clauses. It has the property that the kernel of a verified clause that is obtained from an initial clause is always quasi-evident. Since the kernel contains the initial clause (the formulas of the initial clause don't contain repeating nominals), the Herbrand model of the kernel also satisfies the initial clause. The development of $\mathcal{T}^c$ is considerably complicated by the presence of equations. Thus we recommend that the reader first understands the equation-free case. In the absence of nontrivial equations we have $\tilde{A} = A$ and a nominal is relevant iff it is non-repeating.

Proposition 10.1 For every clause A:

1. $\operatorname{Stk}\tilde{A} = \operatorname{Stk} A$, $\operatorname{Lab}\tilde{A} = \operatorname{Lab} A$, and $\mathcal{L}_{\tilde{A}}x = \mathcal{L}_A x$.
2. For all $x, y \in \operatorname{Rel} A$: $x \sim_{\operatorname{Ker} A} y \iff x \sim_A y$.
3. $\widetilde{\operatorname{Ker} A} = \operatorname{Ker}\tilde{A}$.

A diamond formula $\Diamond\rho t x$ **is ancestor-evident in a clause** A if there exists a nominal y such that $x \prec y$ and $|\rho x y|, ty \in \tilde{A}$. A diamond formula $\Diamond\rho t x$ **is chain-evident in a clause** A if there exists a nominal $x' \sim_A x$ such that x' is non-repeating in A and $\Diamond\rho t x'$ is ancestor-evident in A. Note that a diamond formula is evident if it is ancestor-evident or chain-evident. The diamond rule of $\mathcal{T}^c$ will be defined such that it can only be applied to non-chain-evident diamond formulas, and such that it renders the diamond formula it is applied to chain-evident. The next lemma formulates the key property of chain-evidence.

Lemma 10.2 Let A be a box-propagated clause and $\Diamond\rho t x \in \operatorname{Ker} A$ be chain-evident in A. Then $\Diamond\rho t x$ is quasi-evident in $\operatorname{Ker} A$.

Proof Let x', y be nominals such that $x' \sim_A x$, x' is non-repeating in A, $x' \prec y$, and $|\rho x y|, ty \in \tilde{A}$. We show that $\Diamond\rho t x$ is quasi-evident in $\operatorname{Ker} A$. Case analysis.

Let y be non-repeating in A. Then $|\rho x y|, ty \in \operatorname{Ker}\tilde{A}$ and hence $|\rho x y|, ty \in \widetilde{\operatorname{Ker} A}$ by Proposition 10.1. Thus $\Diamond r t x$ is evident in $\operatorname{Ker} A$, which means that it is also quasi-evident in $\operatorname{Ker} A$.

Let y be repeating in A. Since $x' \prec y$ and x' is not repeating in A, there is an ancestor z of x' that is not repeating in A and satisfies $\mathcal{L}_A z = \mathcal{L}_A y$. Since $t \in \operatorname{Lab} A$ and $ty \in \tilde{A}$, we have $tz \in \tilde{A}$. Since tz is relevant, we have $tz \in \widetilde{\operatorname{Ker} A}$ by Proposition 10.1. It remains to show that $|\rho x z|$ is safe in $\operatorname{Ker} A$. By our assumptions $|\rho x z|$ contains only names that occur in $\operatorname{Ker} A$. We show the claim for $\rho = r$. The case $\rho = r^-$ follows analogously.

Let $\Box rux \in \widetilde{\operatorname{Ker} A}$. We show $uz \in \widetilde{\operatorname{Ker} A}$. Since A is box-propagated and $rxy \in \tilde{A}$, we have $uy \in \tilde{A}$. Since $\mathcal{L}_A z = \mathcal{L}_A y$, we have $uz \in \tilde{A}$. Since u and z are relevant, we have $uz \in \operatorname{Ker} \tilde{A}$. The claim follows with Proposition 10.1.

Let $\Box r^- uz \in \widetilde{\operatorname{Ker} A}$. We show $ux \in \widetilde{\operatorname{Ker} A}$. Since $\mathcal{L}_A z = \mathcal{L}_A y$, we have $\Box r^- uy \in \tilde{A}$. Since A is box-propagated and $rxy \in \tilde{A}$, we have $ux \in \tilde{A}$. Since u and x are relevant, we have $uz \in \operatorname{Ker} \tilde{A}$. The claim follows with Proposition 10.1.

If $\mathrm{T} r \in A$, the additional proof obligations can be shown with arguments analogous to the above. ∎

We now define the chain-based tableau system $\mathcal{T}^c$. The rules of $\mathcal{T}^c$ are obtained from the rules of $\mathcal{T}$ as follows:

- The rules $\mathcal{R}_\Box$ and $\mathcal{R}_\mathrm{T}$ remain unchanged.
- $\mathcal{R}_{\dot\neg}$, $\mathcal{R}_{\dot\wedge}$, $\mathcal{R}_{\dot\vee}$, $\mathcal{R}_-$, $\mathcal{R}_{\{\}}$ are constrained such that they apply only to relevant formulas.
- $\mathcal{R}_\exists$ is constrained such that it introduces fresh nominals that are initial.
- $\mathcal{R}_\forall$ and $\mathcal{R}_\mathrm{R}$ are constrained such that they add only relevant formulas.
- $\mathcal{R}_\mathrm{D}$ is constrained such that it applies only to relevant formulas and introduces fresh nominals that are initial.
- $\mathcal{R}_{\bar{\mathrm{D}}}$ is constrained such that it applies only to relevant formulas and adds only relevant formulas.
- For diamond formulas $\mathcal{T}^c$ has the following rule:

$$\mathcal{R}^c_\Diamond \quad \frac{\Diamond \rho t x}{|\rho x y|\,,\ t y} \quad x \sim_A x',\ x' \notin \operatorname{Rep} A,\ y \notin \mathcal{N} A,\ x' \prec y$$

provided $\Diamond \rho t x$ is not chain-evident in A and not quasi-evident in $\operatorname{Ker} A$

The blocking constraint "$\Diamond \rho t x$ not quasi-evident in $\operatorname{Ker} A$" of $\mathcal{R}^c_\Diamond$ reduces the search space but is not needed for termination. The blocking constraint "$\Diamond \rho t x$ not chain-evident in $\operatorname{Ker} A$" and the side condition "$x' \notin \operatorname{Rep} A$" are essential for termination.

Proposition 10.3 The proof steps of $\mathcal{T}^c$ are sound.

Proof Except for the proof steps obtained with $\mathcal{R}^c_\Diamond$, all proof steps of $\mathcal{T}^c$ are proof steps of $\mathcal{T}$ and hence are sound by Proposition 7.2. The soundness of the proof steps obtained with $\mathcal{R}^c_\Diamond$ is easy to verify. ∎

Proposition 10.4 Let A be a clause that is verified in $\mathcal{T}^c$ such that $\operatorname{Ker} A$ contains at least one nominal. Then $\operatorname{Ker} A$ is quasi-evident.

Proof Most of the evidence and quasi-evidence conditions follow with Proposition 10.1(3) from the non-applicability of the respective tableau rule. For $\mathcal{R}_\exists$ and $\mathcal{R}_\mathrm{D}$ it is essential that the introduced fresh nominals are initial. The argumentation for the quasi-evidence condition for diamond formulas is as follows.

Let $\Diamond\rho tx \in \operatorname{Ker} A$ and suppose $\Diamond\rho tx$ is not quasi-evident in $\operatorname{Ker} A$. Then there exist nominals x' and y such that $x \sim_A x'$, $x' \notin \operatorname{Rep} A$, $y \notin \mathcal{N}A$, and $x' \prec y$. Since A is verified in $\mathcal{T}^c$, $\mathcal{R}^c_\Diamond$ does not apply and hence $\Diamond\rho tx$ must be chain-evident in A. Since A is box-propagated, it follows with Lemma 10.2 that $\Diamond\rho tx$ is quasi-evident in $\operatorname{Ker} A$. Contradiction. ∎

The proof exploits that the box-propagating rules $\mathcal{R}_\Box$ and $\mathcal{R}_\mathrm{T}$ are not constrained to relevant formulas. This makes sure that the entire clause is eventually box-propagated, which is needed so that Lemma 10.2 can be used in the proof.

Example 10.5 Here is a verifying $\mathcal{T}^c$-derivation for the clause of Example 7.8:

$\forall(\Diamond r(\Box r^- p)),\ x{=}x$	initial clause
$\Diamond r(\Box r^- p)x$	$\mathcal{R}^c_\forall$
$rxy,\ \Box r^- py,\ px$	$\mathcal{R}^c_\Diamond,\ \mathcal{R}_\Box$
$\Diamond r(\Box r^- p)y$	$\mathcal{R}^c_\forall$
$ryz,\ \Box r^- pz,\ py$	$\mathcal{R}^c_\Diamond,\ \mathcal{R}_\Box$
$\Diamond r(\Box r^- p)z$	$\mathcal{R}^c_\forall$
$rzu,\ \Box r^- pu,\ pz$	$\mathcal{R}^c_\Diamond,\ \mathcal{R}_\Box$

The nominal x is initial and $x \prec y \prec z \prec u$ holds. The final clause A if verified since $\mathcal{L}_A y = \mathcal{L}_A z = \{\Box r^- p,\ \Diamond r(\Box r^- p),\ p\}$ and thus the nominals z and u are repeating. $\mathcal{R}^c_\forall$ does not add the instance $\Diamond r(\Box r^- p)u$ since it is not relevant. $\operatorname{Ker} A$ consists of all formulas that don't contain the repeating nominals z and u. □

11 Termination of the Chain-Based System

The ancestor relation organizes the nominals in a clause into a finite forest. Since new initial nominals are only introduced by $\mathcal{R}^c_\exists$ and $\mathcal{R}^c_\mathrm{D}$, we know from § 8 that the number of initial nominals in a derivation is bound by the initial clause. Since successors are only introduced by $\mathcal{R}^c_\Diamond$ and only for non-repeating nominals, the depth of the derived clause forests is bound by the initial clause. Moreover, we will show that the breadth of the derived clause forests (i.e., the maximal outdegree) is bound by the stock of the initial clause. Hence the number of nominals in a derived clause is bound by the initial clause, which means that the generative

rules can only be applied finitely often. Thus $\mathcal{T}^c$ terminates on all clauses that contain only initial nominals.

Let us argue more formally. We define the **depth of a nominal** as the length of its ancestor chain and write $\mathrm{depth}\, x$ for the depth of x.

Proposition 11.1 Let x be a nominal that is non-repeating in a clause A. Then $\mathrm{depth}\, x \leq |\mathcal{P}(\mathrm{Lab}\, A)|$.

Proof Follows from the fact that the label sets $\mathcal{L}_A x$ are subsets of $\mathrm{Lab}\, A$. ∎

We define the **breadth of a nominal x in a clause A** as follows:

$$\mathrm{breadth}_A\, x := |\{\, y \in \mathcal{N}A \mid x \prec y \,\}|$$

A clause A is **chain-admissible** if for every nominal $x \in \mathcal{N}A$ the following conditions are satisfied:

1. $\mathrm{depth}\, x \;\leq\; |\mathcal{P}(\mathrm{Lab}\, A)| + 1$
2. $\mathrm{breadth}_A\, x \;\leq\; |\{\, \Diamond\rho t \in \mathrm{Stk}\, A \mid \Diamond\rho t x \text{ is ancestor-evident in } A \,\}| \;\leq\; |\mathrm{Stk}\, A|$

Proposition 11.2 $\mathcal{T}^c$-expansion preserves ancestor-evidence of diamond formulas and chain-admissibility of clauses.

Proof The preservation of ancestor-evidence is obvious. Moreover, $\mathrm{Stk}\, A$ and $\mathrm{Lab}\, A$ are preserved by $\mathcal{T}^c$-expansion. Thus it suffices to consider $\mathcal{R}^c_\Diamond$ since this is the only rule that introduces non-initial nominals. Consider an application of $\mathcal{R}^c_\Diamond$ to $\Diamond\rho t x \in A$ and let A' be the clause obtained. Then there is some $x' \sim_A x$ such that x' is non-repeating in A and $x' \prec y$ for the new nominal y. The depth bound for y follows with Proposition 11.1 and the fact that x' is non-repeating in A. For the breadth bound of x' it suffices to show that $\Diamond\rho t x'$ is not ancestor-evident in A. This is the case since otherwise $\Diamond\rho t x$ would be chain-evident in A, which contradicts the assumption that $\mathcal{R}^c_\Diamond$ is applicable. ∎

Together with the results of § 8, our informal termination argument given at the beginning of this section now rests on firm ground. Thus we have:

Theorem 11.3 (Termination) $\mathcal{T}^c$ terminates on chain-admissible clauses.

Corollary 11.4 $\mathcal{T}^c$ decides the satisfiability of clauses and constructs finite Herbrand models for satisfiable clauses.

Proof Let a clause A be given. First we rename the nominals of A so that the renamed clause A' contains only initial nominals. Clearly, A is satisfiable iff A' is satisfiable and Herbrand models of A' can be renamed into Herbrand models

of A. The renamed clause A' is chain-admissible. If $\mathcal{T}^c$ does not produce a verified clause for A', termination (Theorem 11.3) and the soundness of the proof steps yield that A' is unsatisfiable. Otherwise, let $\mathcal{T}^c$ produce a verified clause A''. Since $A' \subseteq \operatorname{Ker} A''$, $\operatorname{Ker} A''$ contains at least one nominal and we know by Proposition 10.4 that $\operatorname{Ker} A''$ is quasi-evident. Hence we know by Theorem 6.5 that $\operatorname{Ker} A''$ is finitely Herbrand satisfiable. Thus A' and A are finitely Herbrand satisfiable. ∎

12 Remarks

Equational Closure

The way we have defined the equational closure $\tilde{A}$ is not the only possibility. We could also work with a smaller equational closure adding only primitive formulas:

$$\begin{aligned}\tilde{A} \;:=\; A \;&\cup\; \{\, px \mid \exists x' \colon\; x' \sim_A x \;\wedge\; px' \in A \,\} \\ &\cup\; \{\, rxy \mid \exists x', y' \colon\; x' \sim_A x \;\wedge\; y' \sim_A y \;\wedge\; rx'y' \in A \,\}\end{aligned}$$

Working with a smaller closure has the consequence that the tableau rules have to add more formulas. For instance, the quasi-evident set $\{(p \mathbin{\dot\vee} (q_1 \mathbin{\dot\vee} q_2))x,\, (q_1 \mathbin{\dot\vee} q_2)y,\, q_2 y,\;\; x{\approx}y\}$ would not be quasi-evident with the smaller closure.

We could also work with a larger closure that contains formulas that can be obtained by replacing nominals within non-auxiliary modal expressions. Consider the clause $\{\Diamond r(\Box r(=x))x,\, \Diamond r(\Box r(=y))x,\, \Box r(=y)x,\, x{\approx}y\}$. It is not quasi-evident with our definition of the closure, but with the larger closure that contains $\Box r(=x)x$ it would be.

We have chosen to not work with the larger closure since implementing the medium-sized closure seems easier (due to term indexing).

Difference

Fitting [10] gives tableau rules for the difference modalities that assume that all prefixes denote different states (we model prefixes as nominals). This way he can handle difference without primitive equational constraints. However, he overlooks that the assumption that all prefixes denote differently renders his diamond rule unsound (since it always introduces a new prefix). For instance, Fitting's rules can refute the satisfiable expression $\Diamond rp \mathbin{\dot\wedge} \bar{\mathrm{D}} \dot\neg p$ (cf. our Example 7.9). Soundness of Fitting's system can be recovered by a nondeterministic diamond rule that admits already present prefixes as witnesses. We don't follow this approach since a nondeterministic diamond rule would reduce refutation performance even if no difference modalities are present.

Safe Edges

A model existence theorem for a terminating tableau system for a modal logic that has transitive relations or global or difference modalities must add safe edges to the set of formulas obtained with the tableau rules. For transitive relations this is already done in Kripke [22]. Safe edges may introduce cycles into the forest structure obtained with the tableau rules. The idea to block individual diamond formulas that are justified by safe edges first appeared in our paper [20].

Transitive Relations

There are two ways one can obtain a tableau rule for transitive relations: Either one adds edges as required by the evidence condition or one adds box formulas as required by the quasi-evidence condition. While Kripke [22] adds edges, recent systems [13, 17, 10] add boxes. The advantage of adding boxes is that it combines well with safe edges and is more flexible as it comes to model construction.

Symmetric Relations

Our approach extends to formulas that assert the symmetry of a relation. If a relation is symmetric, ordinary modalities also express converse modalities. For symmetric relations the definition of safe edges must be adapted. Moreover, chain-based blocking is needed for termination.

Simple Type Theory

We have used simple type theory as logical base throughout the paper. This way we can use the notions of classical logic and the modal operators can be defined with the classical operators. The type-theoretic base has the advantage that the formulas used with a tableau system appear as formulas of the object language. This is not the case with basic modal logic, since it cannot express prefixed modal expressions and accessibility formulas. For difference and nominals it is natural to work with equations and disequations, which again are not available in basic modal logic. Hybrid logic does have enough syntax to express all tableau formulas (see the internalized tableau system in [6]), but the way prefixed expressions and equations have to be expressed is far from natural.

Search Space

We have tried to keep the search space for the diamond rule as small as possible. Our main workhorse is quasi-evidence, that is, diamond formulas that are quasi-evident (in the kernel) must not be expanded. The blocking condition based on chain-evidence avoids further diamond expansions. These search space prunings are not provided by the systems in [6, 21].

13 Conclusion

This paper updates and extends results of two preliminary papers [20, 21]. In our view, our main contributions are as follows.

- *Model existence theorem.* The model existence theorem for quasi-evident sets explains how edges can be safely added after the tableau system has done its work. Before, this important technique was buried in opaque model existence proofs for particular tableau systems. In our case, the model existence theorem precedes the tableau systems, not vice versa. Our basic system $\mathcal{T}$ corresponds directly to the model existence theorem for quasi-evident sets, and the refined systems $\mathcal{T}^p$ and $\mathcal{T}^c$ are obtained from $\mathcal{T}$ by adding blocking constraints.
- *Pattern-based control.* The basic tableau system terminates for converse-free formulas if box propagation is prioritized. We speak of pattern-based blocking to distinguish this new control from the established control obtained with chain-based blocking. Pattern-based blocking has great potential for efficient implementation since in total at most $|\mathcal{P}(\mathrm{Lab}\, A)|$ diamond expansions are needed, where A is the initial clause. In contrast, chain-based blocking achieves the same bound only per ancestor chain.
- *Treatment of equality.* The main technical difficulty we had to overcome for this paper is the transparent treatment of the equational constraints that come with nominals and difference. After exploring several possibilities [14, 20, 21], we finally arrived at the treatment employed in this paper. The construction is now such that the treatment of equational constraints comes as a transparent refinement of an underlying equality-free system. This is achieved with an equational closure whose representation is left open.
- *Difference.* We present the first terminating tableau systems for the difference modalities. Given the general approach of this paper, the treatment of difference is no big deal. But it were difficulties with the difference modalities that led us to the abstract treatment of equality with $\sim_A$ and $\tilde{A}$.

References

[1] Carlos Areces and Balder ten Cate. Hybrid logics. In Blackburn et al. [5], pages 821–868.

[2] Franz Baader and Carsten Lutz. Description logic. In Blackburn et al. [5], pages 757–820.

[3] 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 Intl. Joint Conf. on Artificial Intelligence (IJCAI'97)*, pages 190–195. Morgan Kaufmann, 1997.

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

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

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

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

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

[9] William M. Farmer. The seven virtues of simple type theory. *J. Appl. Log.*, 6(3):267–286, 2008.

[10] Melvin Fitting. Modal proof theory. In Blackburn et al. [5], pages 85–138.

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

[12] George Gargov, Solomon Passy, and Tinko Tinchev. Modal environment for Boolean speculations (preliminary report). In Dimiter G. Skordev, editor, *Mathematical Logic and Its Applications: Proceedings of an Advanced International Summer School and Conference in honor of the 80th anniversary of Kurt Gödel's birth*, pages 253–263. Plenum Press, 1987.

[13] 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.

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

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

[16] Ian Horrocks, Ullrich Hustadt, Ulrike Sattler, and Renate Schmidt. Computational modal logic. In Blackburn et al. [5], pages 181–245.

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

[18] Ian Horrocks and Ulrike Sattler. A tableau decision procedure for SHOIQ. *J. Autom. Reasoning*, 39(3):249–276, 2007.

[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. To appear in ENTCS 2009.

[21] Mark Kaminski and Gert Smolka. Terminating tableaux for hybrid logic with the difference modality and converse. In Alessandro Armando, Peter Baumgartner, and Gilles Dowek, editors, *IJCAR 2008*, volume 5195 of *LNCS (LNAI)*, pages 210–225. Springer, 2008.

[22] Saul A. Kripke. Semantical analysis of modal logic I: Normal modal propositional calculi. *Z. Math. Logik Grundlagen Math.*, 9:67–96, 1963.