Methods for Modalities 3

Carlos Areces

INRIA Lorraine. 615, rue du Jardin Botanique, 54602 Villers les Nancy Cedex, France

---

On September 2003 LORIA, the "Laboratoire Lorrain de Recherche en Informatique et ses Applications" organized the third instance of the Methods for Modalities Workshop (M4M-3) in Nancy, France. As in the previous instances of the workshop, the focus of the meeting was on reasoning methods, decision methods and proof tools for modal and modal-like languages and, also as in previous instances, the event was a great place to interchange ideas and obtain an up-to-date picture of the field.

In addition to about 15 paper presentation, M4M-3 hosted 6 tutorials: Stephane Demri on *(Modal) Logics for Semistructured Data (bis)*; Claude Kirchner on *An Introduction to Deduction Modulo*; Carsten Lutz on *Expressivity and Complexity of Description Logics with Concrete Domains*; Maarten Marx on *Variable Free Reasoning on Finite Trees*; Stephan Merz on *The Automata-theoretic Framework for Model Checking Revisited*; and Ralf Möller on *The Ins and Outs of RACER*. And all that in only 2 days, which made for a very full, but also very interesting agenda.

Given the quality of the submissions received, the M4M Program Committee decided that a journal special issue was in order, and I took the editorial responsibilities. I decided to have an open call for papers to which all M4M-3 submissions were invited to contribute, but which also allowed the addition of new articles. The call was extremely successful, with a total of over 25 papers submitted, and the selection of the ones to be included in the present issue was not an easy matter.

All papers were carefully reviewed and refereed and the result of about one year of work is the following selection:

"A general method for proving decidability of intuitionistic modal logics" by Alechina and Shkatov. Alechina and Shkatov generalize in this paper a result of Ganzinger, Meyer and Veanes on the decidability of the two variable monadic guarded fragment of FOL with constraints on the guard relations expressible in monadic SOL to also allow constraints involving several relations. In this way, the framework fits the semantic definitions of many well known intuitionistic modal logics

Email address: areces@loria.fr (Carlos Areces).

Preprint submitted to Elsevier Science *1 June 2005*

where the accessibility relations governing the modalities usually interacts with the order relation used in the definition of the intuitionistic operators. The result: straightforwards new proofs of decidability for many intuitionistic modal logics.

"Intuitionistic hybrid logic" by Braüner and de Paiva. Continuing with the topic of intuitionistic modal logics, in this paper the authors investigate the effects of extending a number of such logics with hybrid operators (nominals and the 'at' operator). Sound and complete natural deduction calculi for a wide class of constructive hybrid logics are provided, and a general normalization theorem and a version of the subformula property are proved. In this way, the paper shows that constructive hybrid logics are a viable enterprise opening the way for future applications, and that the process of 'hybridization' does not require a classical basis.

"On modal μ-calculus with explicit interpolants" by D'Agostino and Lenzi. In this paper, D'Agostino and Lenzi discuss in detail the issue of how the existential bisimulation quantifier can be used to construct uniform interpolants for the μ-calculus. They provide an explicit form for the uniform interpolant of a given disjunctive μ-formula, and show that it belongs to the same level of the fix-point alternation hierarchy. Moreover, the paper proves that the result cannot be extended to the whole logic: while the first two levels of the hierarchy are closed under the existential bisimulation quantifier, the closure of the third level has already the expressivity of the whole μ-calculus. The paper finishes with a complete axiomatization of the μ-logic extended with the existential bisimulation quantifier.

"Model checking hybrid logics (with and application to semistructured data)" by Franceschet and de Rijke. Even though the satisfiability problem for many different hybrid logics has been, by now, well investigated, the model checking problem has received considerably less attention. The paper of Franceschet and de Rijke closes this gap, providing a detailed complexity analysis of this reasoning problem together with model checking algorithms for a wide spectrum of hybrid languages. In addition, the paper shows how model checking for hybrid languages can be used in two different applications (query and constraint evaluation) concerning semistructured data.

"A general tableau method for propositional interval temporal logics: theory and implementation" by Goranko, Montanari, Sala and Sciavicco. Temporal logics interpreted over point-based structures (either linear or branching) are widely used (e.g., in the field of verification) and their theoretical properties are well known. In contrast, *interval* based temporal logics, providing a natural framework for representing and reasoning in several areas of computer science, are much less understood. Goranko, Montanari, Sala and Sciavicco start this paper with an overview of the current state-of-the-art in the field of interval temporal logics. The main result of the paper is a general tableaux method for a very expressive interval based temporal logic which can be easily tailored to most propositional interval temporal logics proposed in the literature. A first implementation of the tableaux has been

developed and preliminary testing is discussed.

"Deduction chains for common knowledge" by Kretz and Studer. The Logic for Common Knowledge includes the modal operator C with the intended meaning that $\mathsf{C}\alpha$ is true if all agents know α, and all agents know that all agents know α, ... A proof system capturing this intended meaning needs to cope with the infinite nature of the common knowledge operator. Kretz and Studer show in this paper a very simple and elegant Tait-style proof system which is sound and complete for the Logic of Common Knowledge.

I close this short editorial with my thanks to the many referees which helped me screen and select the papers for this special issue, and to the authors of the selected papers for doing their best at incorporating the many detailed comments from the referees in their final versions.

A general method for proving decidability of intuitionistic modal logics

Natasha Alechina

School of Computer Science and IT, University of Nottingham

Dmitry Shkatov

School of Computer Science and IT, University of Nottingham

Abstract

We generalise the result of Ganzinger, Meyer and Veanes [14] on decidability of the two variable monadic guarded fragment of first order logic with constraints on the guard relations expressible in monadic second order logic. In [14], such constraints apply to one relation at a time. We modify their proof to obtain decidability for constraints involving several relations. Now we can use this result to prove decidability of multi-modal modal logics where conditions on accessibility relations involve more than one relation. Our main application is intuitionistic modal logic, where the intuitionistic and modal accessibility relations usually interact in a non-trivial way.

Key words: Intuitionistic Modal Logics, Decidibility

1 Introduction

In this paper, we present a new general way of proving decidability of multi-modal modal logics. This method relies on the result of Ganzinger, Meyer and Veanes [14], that a monadic two-variable guarded fragment GF^2_{mon} of classical first-order logic, where guard relations satisfy conditions that can be expressed as monadic second-order definable closure constraints, is decidable. Our contribution is a slight generalisation of this result to account for conditions which involve more than one

Email addresses: nza@cs.nott.ac.uk (Natasha Alechina), dxc@cs.nott.ac.uk (Dmitry Shkatov).

[1] This research was supported by the EPSRC grant GR/M98050/01

guard relation. We believe that this method is particularly promising for intuitionistic modal logic, where there exists a variety of systems, most of them semantically defined, with various conditions connecting the intuitionistic and modal accessibility relations. General results on decidability and finite modal property of intuitionistic modal logic have been proved in [34, 33, 35] using an embedding of intuitionistic modal logics with n modalities in classical modal logics with $n + 1$ modalities. However, their results can only be used to prove decidability of those intuitionistic modal logics, for which the corresponding classical logic is known to be decidable.

The decidability proof presented in this paper does not give a good decision procedure, since it proceeds by reduction to satisfiability of formulas of SkS (monadic second-order theory of trees with constant branching factor k, [28]) which is non-elementary [2] . It does however provide a rather simple way to establish decidability, before looking for a decision procedure tailored for a particular logic.

2 Two-variable monadic guarded fragment

We start by defining GF^2_{mon} as introduced in [14]. In the following definitions, $FV(\varphi)$ stands for the set of free variables of φ, and $\overline{x}$ stands for a sequence of variables. We assume a first order language which contains predicate letters of arbitrary arity, including equality $=$, and no constants or functional symbols.

Definition 1 *The guarded fragment GF of first-order logic is the smallest set that contains all first-order atoms and is closed under boolean connectives and the following rule: if ρ is an atom, $\varphi \in GF$, and $\overline{x} \subseteq FV(\varphi) \subseteq FV(\rho)$, then $\exists\overline{x}(\rho \wedge \varphi)$ and $\forall\overline{x}(\rho \rightarrow \varphi) \in GF$ (in such a case ρ is called a guard).*

The monadic two-variable guarded fragment GF^2_{mon} is the subset of GF containing formulas φ such that (i) φ has no more than two variables (free or bound), and (ii) all non-unary predicate letters of φ occur in guards.

3 Closure conditions

In this section we define the form of conditions on guards in GF^2_{mon} which yield decidable fragments. We generalise the notion of mso-definable (monadic second

[2] Better complexity bounds for the guarded fragment with transitive guards were obtained in [17] and [31], however their results apply only to transitivity, and it is not clear whether they could be extended to arbitrary closure conditions, which we need for intuitionistic modal logics.

order definable) closure conditions from [14] so that they can apply to more than one relation.

Definition 2 *Let W be a non-empty set. A unary function C on W is a* simple closure operator *if, for all $\mathcal{P}, \mathcal{P}' \subseteq W$,*

(1) $\mathcal{P} \subseteq C(\mathcal{P})$ (C is increasing),
(2) $\mathcal{P} \subseteq \mathcal{P}'$ implies $C(\mathcal{P}) \subseteq C(\mathcal{P}')$ (C is monotone)
(3) $C(\mathcal{P}) = C(C(\mathcal{P}))$ (C is idempotent).

An $n+1$-ary function C on the powerset of W is a parametrised closure operator *if $C(\mathcal{P}_1, \ldots, \mathcal{P}_n, -)$ for any $\mathcal{P}_1, \ldots, \mathcal{P}_n \subseteq W$ is a simple closure operator. We use thenotation $C^{\mathcal{P}_1,\ldots,\mathcal{P}_n}$ for a closure operator parametrised by $\mathcal{P}_1, \ldots, \mathcal{P}_n$.*

Example 3 *A reflexive, transitive closure operator for binary relations $TC(\mathcal{P})$ is a simple closure operator.*

Example 4 *A function $Incl^{\mathcal{P}'}(\mathcal{P}) = \mathcal{P}' \cup \mathcal{P}$ is a closure operator parametrised by $\mathcal{P}'$.*

Definition 5 *A condition on relation $\mathcal{P}$ is a* simple closure condition *if it can be expressed in the form $C(\mathcal{P}) = \mathcal{P}$, where C is a simple closure operator.*

A condition on relation $\mathcal{P}$ is a parametrised closure condition *if it can be expressed in the form $C^{\mathcal{P}_1,\ldots,\mathcal{P}_n}(\mathcal{P}) = \mathcal{P}$, where $C^{\mathcal{P}_1,\ldots,\mathcal{P}_n}$ is a parametrised closure operator.*

Example 6 *Reflexivity-and-transitivity is a simple closure condition, since it can be expressed in the form $TC(\mathcal{P}) = \mathcal{P}$.*

Example 7 *Condition $\mathcal{P}' \subseteq \mathcal{P}$ is a closure condition on $\mathcal{P}$ parametrised by $\mathcal{P}'$, since it can be stated as $Incl^{\mathcal{P}'}(\mathcal{P}) = \mathcal{P}$.*

Given a set of closure conditions on a set of relations S, we want to preclude circularity while closing off relations in S.

Definition 8 *Let S be a finite set of relations, $\mathbf{C}$ a set of closure conditions on those relations, and $\mathbf{C}(\mathcal{P})$ be all the closure conditions on the relation $\mathcal{P}$ from $\mathbf{C}$. $\mathbf{C}$ is* acyclic *if there is an ordering $\mathcal{P}_1, \ldots, \mathcal{P}_n$ of S such that all parameters in $\mathbf{C}(\mathcal{P}_{i+1})$ come from $\mathcal{P}_1, \ldots, \mathcal{P}_i$.*

Furthermore, we are not interested in arbitrary closure operators, but only in those definable in monadic second-order logic. Let $\|\varphi(x_1, \ldots, x_n)\|^{\mathcal{M}}$ stand for the set of n-tuples satisfying φ in model $\mathcal{M}$.

Definition 9 *A closure operator $C^{\mathcal{P}_1,\ldots,\mathcal{P}_m}$ on n-ary relations is mso (-definable), if there exists a monadic second-order formula $\overline{C_P^{P_1,\ldots,P_m}}$ with predicate parameters*

$P_1, \ldots, P_m$ *and* P*, such that, for any model* $\mathcal{M}$ *and any* n*-ary formula* φ*,*

$$C^{\mathcal{P}_1,\ldots,\mathcal{P}_m}(\|\varphi\|^{\mathcal{M}}) = \|\overline{C_P^{P_1,\ldots,P_m}}(\varphi/P))\|^{\mathcal{M}}.$$

Example 10 *The closure operator* TC *is definable by the mso formula*

$$\overline{TC_P}(z_1, z_2) = \forall X(X(z_1) \wedge \forall x, y(X(x) \wedge P(x, y) \rightarrow X(y)) \rightarrow X(z_2))$$

To see that $\overline{TC_P}$ defines the reflexive, transitive closure of P, assume that there is a P-chain $a_1 \stackrel{P}{\longrightarrow} a_2 \ldots a_{n-1} \stackrel{P}{\longrightarrow} a_n$, connecting a_1 and a_n, and that $X(a_1)$ and $\forall x, y(X(x) \wedge P(x, y) \rightarrow X(y))$ hold. Then $X(a_1)$ implies $X(a_2)$, $X(a_2)$ implies $X(a_3)$, etc., $X(a_n)$ is true, so $\overline{TC_P}(a_1, a_n)$ is true. Conversely, suppose there is no P-chain connecting a_1 and a_n. We can assign to X the set containing a_1 and all the elements P-reachable from a_1, which makes $X(a_n)$ and $\overline{TC_P}(a_1, a_n)$ false.

Example 11 *The closure operator* $Incl^{\mathcal{P}'}$ *is definable by the mso (in fact, first-order) formula*

$$\overline{Incl_P^{P'}}(z_1, z_2) = P'(z_1, z_2) \vee P(z_1, z_2)$$

Theorem 12 *Let* $\phi \in GF^2_{mon}$ *and* **C** *be an acyclic set of mso closure conditions on relations in* ϕ *so that at most one closure condition is associated with each relation. It is decidable whether* ϕ *is satisfiable in a model satisfying* **C***.*

PROOF. The proof is very similar to the proof given in [14] for non-parametrised closure conditions. In fact, it is slightly simpler, because in the original proof all relations are assumed to be closed under equivalence (to show decidability of the fragment with equality). However, closure under equivalence is a special case of a parametrised closure condition, so we do not need to treat it separately.

Let $\phi \in GF^2_{mon}$ and let **C** be an acyclic set of mso closure conditions on relations in ϕ. ϕ is satisfiable in a model satisfying **C** iff N, the Skolemised form of ϕ, is satisfiable in a Herbrand model in which all conditions from **C** hold. The idea of the decidability proof is to reduce the latter problem to satisfiability of formulas of SkS (the mso theory of trees with constant branching factor k), where k is the number of Skolem function symbols in N. We construct an mso formula MSO_N, in the vocabulary of SkS (an mso formula containing only unary relation variables, unary functions and equality), such that MSO_N is satisfiable in a tree model iff N has a Herbrand model satisfying closure conditions from **C**. The construction proceeds in three stages: defining counterparts for predicate letters, for clauses in N and finally for N itself.

Stage 1. For each predicate P in N, construct a formula φ_P in the vocabulary of SkS.

Let $P(\overline{t_1}), \ldots, P(\overline{t_m})$ be all positive literals of N containing P. Note that since $\phi \in GF^2_{mon}$, each P is either a unary or a binary predicate; each positive literal will contain at most one free variable. For each $P(\overline{t_i})$ above, a new unary second-order variable $X_{P(\overline{t_i})}$ is introduced. Let $\bar{t}[z]$ be the result of substituting a variable z for the free variable of $\bar{t}$. Then, if P is a unary predicate,

$$\varphi_P(z_1) = \bigvee_{i=1}^{m} \exists z(X_{P(t_i)}(z) \wedge z_1 = t_i[z])$$

and if P is a binary predicate,

$$\varphi_P(z_1, z_2) = \bigvee_{i=1}^{m} \exists z(X_{P(t_{i1},t_{i2})}(z) \wedge z_1 = t_{i1}[z] \wedge z_2 = t_{i2}[z])$$

Intuitively, the relation defined by φ_P is the minimal extension of P.

Next, for each predicate that has a closure condition imposed on it, we define the closure ψ_P of φ_P with respect to the closure condition on P. For each such P we have a single closure condition C_P, which may be parametrised by other predicates. For simplicity, assume that C_P is parametrised by a single predicate P' that, in its own turn, has a simple closure condition $C_{P'}$. We know, then, that $C_{P'}$ is definable by an MSO formula $\overline{C_{P'}}(z_1, z_2)$ containing P', and C_P is definable by an MSO formula $\overline{C_P^{P'}}(z_1, z_2)$, containing P' and P. First, we define the closure of P' with respect to its simple closure condition:

$$\psi_{P'}(z_1, z_2) = \overline{C_{P'}}(z_1, z_2)[\varphi_{P'}/P']$$

that is, we replace every occurrence of P' in $\overline{C_{P'}}(z_1, z_2)$ with $\varphi_{P'}$.

Next, we define the closure of P with respect to its parametrised condition:

$$\psi_P(z_1, z_2) = \overline{C_P^{P'}}(z_1, z_2)[\psi_{P'}/P', \varphi_P/P]$$

In general, for any acyclic set $\mathbf{C}$ of conditions on the collection of relations S, we first define the simple closures, then the closures parametrised by relations with simple closure conditions, etc. The acyclicity of $\mathbf{C}$ ensures that this procedure can be carried out.

Stage 2. For each clause $\chi = \{\rho_1, \ldots, \rho_l\}$ in N, construct a formula MSO_χ in the vocabulary of SkS.

For every literal ρ in χ, a formula MSO_ρ is defined according to the following rule:

$$MSO_\rho = \begin{cases} X_\rho(x), & \text{if } \rho \text{ is a non-ground atom containing } x \\ \exists z X_\rho(z), & \text{if } \rho \text{ is a ground atom} \\ \neg\psi_P(\bar{t}), & \text{if } \rho \text{ is } \neg P(\bar{t}) \end{cases}$$

where ψ_P is the formula constructed at stage 1. Now MSO_χ is defined as $MSO_\chi = \bigvee_{\rho\in\chi} MSO_\rho$.

Stage 3. Finally, $MSO_N = \exists\overline{X}\forall\overline{x}\bigwedge_{\chi\in N} MSO_\chi$, where $\overline{X}$ are all the free second order variables and $\overline{x}$ are all the first order variables in $\bigwedge_{\chi\in N} MSO_\chi$.

It remains to show that N has a Herbrand model satisfying the closure conditions in $\mathbf{C}$ iff MSO_N is satisfiable in a tree. Let $\mathcal{T}$ be the tree corresponding to the term algebra of the Herbrand universe of N.

($\Leftarrow$) Assume that N has a Herbrand model $\mathcal{A}$ satisfying the closure conditions in $\mathbf{C}$. We want to show that $\mathcal{T}$ satisfies MSO_N. Fix witnesses for second-order variables X_ρ of MSO_N as follows:

(i) If $\overline{t_i}$ is non-ground, then $X_{P(\overline{t_i})} = \{a : \mathcal{A} \models P(\overline{t_i}[a])\}$.

(ii) If $\overline{t_i}$ is ground, then $X_{P(\overline{t_i})}$ is a non-empty set.

We know that for each clause χ of N, and each tuple $\overline{a}$, $\mathcal{A} \models \chi(\overline{a})$. This means that for each $\overline{a}$, there is a literal ρ in χ such that $\mathcal{A} \models \rho(\overline{a})$. We show that for any $\overline{a}$ and ρ, if $\mathcal{A} \models \rho(\overline{a})$, then $\mathcal{T} \models MSO_\rho(\overline{a})$. Hence $\mathcal{A} \models \chi(\overline{a})$ implies $\mathcal{T} \models MSO_\chi(\overline{a})$.

There are three cases to consider, depending on the form of ρ. The first two (non-ground atom $P(\overline{t_i})$ and ground atom) are exactly the same as in [14]. If ρ is a negative literal $\neg P(\overline{t_i})$, we need to show that $\mathcal{T} \models \neg\psi_P(\bar{t})(\overline{a})$. It suffices to show that $\|\psi_P\|^{\mathcal{A}} \subseteq P^{\mathcal{A}}$. Indeed, this, together with our assumption that $\mathcal{A} \models \neg P(\bar{t})[\overline{a}]$, implies $\mathcal{T} \models \neg\psi_P(\overline{a})$. First, the definition of $\mathcal{T}$ guarantees that $\|\varphi_P\|^{\mathcal{A}} \subseteq P^{\mathcal{A}}$. Hence, by monotonicity of closure operators, $C_P^{P_1^{\mathcal{A}}}(\|\varphi_P\|^{\mathcal{A}}) \subseteq C_P^{P_1^{\mathcal{A}}}(P^{\mathcal{A}})$. By definition of ψ_P, $C_P^{P_1^{\mathcal{A}}}(\|\varphi_P\|^{\mathcal{A}}) = \|\psi_P\|^{\mathcal{A}}$; furthermore, since $\mathcal{A}$ satisfies conditions in $\mathbf{C}$, $C_P^{P_1^{\mathcal{A}}}(P^{\mathcal{A}}) = P^{\mathcal{A}}$; hence, $\|\psi_P\|^{\mathcal{A}} \subseteq P^{\mathcal{A}}$.

($\Rightarrow$) Assume that MSO_N is true in $\mathcal{T}$. Define a Herbrand model $\mathcal{A}$ as follows. The universe of $\mathcal{A}$ is the set of nodes of $\mathcal{T}$, and $P^{\mathcal{A}} = \|\psi_P\|$. First, we prove that $\mathcal{A}$ satisfies closure conditions $\mathbf{C}$. To this end, we have to show that $C_P^{\mathcal{P}_1}(P^{\mathcal{A}}) = P^{\mathcal{A}}$. Indeed, $C_P^{P_1^{\mathcal{A}}}(P^{\mathcal{A}}) = C_P^{P_1^{\mathcal{A}}}(\|\psi_P\|) = C_P^{\|\psi_{P_1}\|}(C_P^{\|\psi_{P_1}\|}(\|\ \varphi_P\|)) = C_P^{\|\psi_{P_1}\|}(\|\varphi_P\|) = \|\psi_P\| = P^{\mathcal{A}}$.

Finally, we need to show that $\mathcal{A}$ satisfies all clauses in N. This part of the proof is exactly the same as in [14].

4 Intuitionistic modal logics

One of the most promising applications of the result above is propositional intuitionistic modal logic. Intuitionistic modal logic is simply a modal logic with intuitionistic, rather than classical, base. The work on intuitionistic modal logic has several motivations: mathematical interest; preference for intuitionistic rather than classical logic; desire to give intuitionistic account of the notions studied in modal logic; and suitability of intuitionistic modal logic for modelling certain computational phenomena. There exists an extensive literature on intuitionistic modal logics, for example [13, 5, 6, 7, 27, 20, 22, 23, 15, 12, 26, 10, 32, 34, 33, 35]. A comprehensive survey can be found in [29]; for later references, see [36] and [24].

One of the motivations for intuitionistic modal logic is modelling computational phenomena. A considerable strand of work in this area is based on the work by Moggi [21] who extended a typed λ-calculus style semantics for functional programming languages with an additional construct - a monad - to model effects in functional programming languages (such as the raising of exceptions etc.). The correspondence between simply-typed λ-calculus and intuitionistic propositional logic is well known; it turns out that monads correspond to S4-type modalities. This created a considerable interest in intuitionistic S4 modal logic, its proof theory and categorical and Kripke semantics [4, 3, 16, 18, 25, 1, 8, 9, 24]. Other applications of intuitionistic modal logic to modelling computational phenomena included modelling incomplete information [32], communicating systems [30], hardware verification [19, 11], etc.

Intuitionistic modal languages are obtained by adding either or both of the unary connectives $\Box$ (necessity) and $\Diamond$ (possibility) to the language of propositional intuitionistic logic, which contains a set of propositional parameters $\mathtt{Par} = \{p_1, p_2, \ldots\}$, a unary connective $\sim$, and binary connectives $\wedge$, $\vee$, and $\Rightarrow$. Analogously to $\forall$ and $\exists$, in intuitionistic logic $\Box$ and $\Diamond$ are not required to be dual. It is also to be expected that for example $\Box(\varphi \vee {\sim}\varphi)$ is not valid. In some intuitionistic modal logics, $\Diamond(\varphi \vee \psi) \equiv (\Diamond\varphi \vee \Diamond\psi)$ is not valid either; see for example [32].

Kripke semantics of intuitionistic modal logics extends Kripke semantics for intuitionistic propositional logic. An intuitionistic Kripke model is a structure $\mathcal{M} = (W, \mathcal{R}, V)$ such that (i) $W \neq \emptyset$, (ii) $\mathcal{R}$ is a reflexive and transitive binary relation on W, and (iii) V is a function from $\mathtt{Par}$ into the powerset of W such that, for all $w \in W$ and $p \in \mathtt{Par}$, if $w \in V(p)$ and $w\mathcal{R}v$, then $v \in V(p)$ (condition we will refer to as upward persistence for propositional variables). Elements of W are called nodes. Truth at a node is defined as follows ($\rightarrow$ and $\neg$ stand for classical

implication and negation, respectively):

$$\begin{aligned}
&\mathcal{M}, w \models p && \text{iff } \ w \in V(p);\\
&\mathcal{M}, w \models {\sim}\varphi && \text{iff } \ \forall v(\mathcal{R}(w,v) \to \neg(\mathcal{M}, v \models \varphi));\\
&\mathcal{M}, w \models \varphi \wedge \psi && \text{iff } \ \mathcal{M}, w \models \varphi \text{ and } \mathcal{M}, w \models \psi;\\
&\mathcal{M}, w \models \varphi \vee \psi && \text{iff } \ \mathcal{M}, w \models \varphi \text{ or } \mathcal{M}, w \models \psi;\\
&\mathcal{M}, w \models \varphi \Rightarrow \psi && \text{iff } \ \forall v(\mathcal{R}(w,v) \to (\neg(\mathcal{M}, v \models \varphi) \text{ or } \mathcal{M}, v \models \psi);
\end{aligned}$$

To accommodate formulas of the form $\Box\varphi$ and $\Diamond\varphi$, intuitionistic Kripke models are augmented with binary relations $\mathcal{R}_\Box$ and $\mathcal{R}_\Diamond$. There is no single accepted way of defining the meaning of $\Box$ and $\Diamond$ in intuitionistic logic. The following clauses are encountered in the literature (see chapter 3 of [29] for a comprehensive survey):

$$\begin{aligned}
&(\Box_1)\ \mathcal{M}, w \models \Box\varphi \ \text{ iff } \ \forall v(w\mathcal{R}_\Box v \to \mathcal{M}, v \models \varphi)\\
&(\Box_2)\ \mathcal{M}, w \models \Box\varphi \ \text{ iff } \ \forall v(w\mathcal{R}v \to \forall u(v\mathcal{R}_\Box u \to \mathcal{M}, u \models \varphi))\\
&(\Diamond_1)\ \mathcal{M}, w \models \Diamond\varphi \ \text{ iff } \ \exists v(w\mathcal{R}_\Diamond v \wedge \mathcal{M}, v \models \varphi)\\
&(\Diamond_2)\ \mathcal{M}, w \models \Diamond\varphi \ \text{ iff } \ \forall v(w\mathcal{R}v \to \exists u(v\mathcal{R}_\Diamond u \wedge \mathcal{M}, u \models \varphi))
\end{aligned}$$

Observe that definition $(\Diamond_2)$ gives rise to a modality which does not distribute over disjunction (hence to a non-normal modal logic).

On top of the requirement that $\mathcal{R}$ is reflexive and transitive, some additional conditions are usually imposed on $\mathcal{R}$, $\mathcal{R}_\Box$, and $\mathcal{R}_\Diamond$. As a rule, these conditions specify the way $\mathcal{R}$, $\mathcal{R}_\Box$, and $\mathcal{R}_\Diamond$ interact. For example, the following conditions usually accompany truth clauses $(\Box_1)$ and $(\Diamond_1)$ (see [34]):

$$\mathcal{R} \circ \mathcal{R}_\Box \circ \mathcal{R} = \mathcal{R}_\Box \tag{1}$$

$$\mathcal{R} \circ \mathcal{R}_\Diamond^{-1} \circ \mathcal{R} = \mathcal{R}_\Diamond^{-1} \tag{2}$$

In the conditions above, $\circ$ stands for relational composition:

$$\mathcal{R} \circ \mathcal{R}' = \{\langle a, b\rangle : \exists c\, (\langle a, c\rangle \in \mathcal{R}\ \&\ \langle c, b\rangle \in \mathcal{R}')\}$$

Another condition occurring in the literature (see for example [11]) stipulates that

$$\mathcal{R}_\Diamond \subseteq \mathcal{R} \tag{3}$$

It turns out that many of the conditions on $\mathcal{R}$, $\mathcal{R}_\Box$ and $\mathcal{R}_\Diamond$, including conditions (1)

- (3) above, are mso-definable closure conditions as introduced in Section 3. For condition (3), see Examples 4 and 7. Below is a proof for (1) and (2).

Proposition 13 *Condition of the form $\mathcal{P} = \mathcal{P}' \circ \mathcal{P} \circ \mathcal{P}'$ is an mso-definable closure condition, provided that $\mathcal{P}'$ is reflexive and transitive.*

PROOF. Consider a function $Comp^{\mathcal{P}'}(\mathcal{P}) = \mathcal{P}' \circ \mathcal{P} \circ \mathcal{P}'$. If $\mathcal{P}'$ is reflexive and transitive, then $\mathcal{P} \subseteq \mathcal{P}' \circ \mathcal{P} \circ \mathcal{P}'$ by the reflexivity of $\mathcal{P}'$. $\mathcal{P}' \circ \mathcal{P} \circ \mathcal{P}'$ is obviously monotone in $\mathcal{P}$; and $Comp^{\mathcal{P}'}$ is idempotent because of the transitivity of $\mathcal{P}'$. This proves that $Comp^{\mathcal{P}'}$ is a closure operator provided that $\mathcal{P}'$ is reflexive and transitive. Conditions of the form $\mathcal{P}' \circ \mathcal{P} \circ \mathcal{P}' = \mathcal{P}$ can be expressed as closure conditions: $Comp^{\mathcal{P}'}(\mathcal{P}) = \mathcal{P}$. This condition is mso-definable; in fact, it is definable by a first order formula:

$$\overline{Comp^{\mathcal{P}'}_{\mathcal{P}}}(z_1, z_2) = \exists x \exists y (P'(z_1, x) \wedge P(x, y) \wedge P'(y, z_2))$$

5 Embedding into the two-variable monadic fragment

In this section, we show that every intuitionistic modal logic L defined semantically with any of the truth clauses $(\Box_1) - (\Diamond_2)$ can be translated into GF^2_{mon}.

We define, by mutual recursion, two translations, τ_x and τ_y, so that a first-order formula $\tau_v(\varphi)$ $(v \in \{x, y\})$ contains a sole free variable v, which intuitively stands for the world at which φ is being evaluated in the Kripke model. τ_x is defined by

$$\tau_x(p) := P(x)$$
$$\tau_x(\sim\varphi) := \forall y(R(x, y) \rightarrow \neg\tau_y(\varphi))$$
$$\tau_x(\varphi \wedge \psi) := \tau_x(\varphi) \wedge \tau_x(\psi)$$
$$\tau_x(\varphi \vee \psi) := \tau_x(\varphi) \vee \tau_x(\psi)$$
$$\tau_x(\varphi \Rightarrow \psi) := \forall y(R(x, y) \rightarrow (\neg\tau_y(\varphi) \vee \tau_y(\psi)))$$
$$\tau_x(\Box\varphi) := \forall y(R(x, y) \rightarrow \forall x(R_\Box(y, x) \rightarrow \tau_x(\varphi)))$$
$$\tau_x(\Diamond\varphi) := \forall y(R(x, y) \rightarrow \exists x(R_\Diamond(y, x) \wedge \tau_x(\varphi)))$$

τ_y is defined analogously, switching the roles of x and y. This translation assumes modal truth clauses $(\Box_2)$ and $(\Diamond_2)$. Clauses for $(\Box_1)$ and $(\Diamond_1)$ are even simpler (and familiar from classical modal logic):

$$\tau'_x(\Box\varphi) := \forall y(R_\Box(x, y) \rightarrow \tau'_y(\varphi))$$
$$\tau'_x(\Diamond\varphi) := \exists y(R_\Diamond(x, y) \wedge \tau'_y(\varphi))$$

Not surprisingly, since τ_x is a natural generalisation of the standard translation of modal logic into classical predicate logic, the following theorem holds:

Theorem 14 *Let ϕ be an intuitionistic modal formula and M be a class of models of intuitionistic modal logic. Let $\mathcal{M} \in$ M. Then, $\mathcal{M}, w \models \phi$ iff $\mathcal{M} \models \tau_x(\phi)[w]$ (where $\mathcal{M}$ is taken as a model of first order logic with $\mathcal{R}, \mathcal{R}_\Box, \mathcal{R}_\Diamond$ interpreting $R, R_\Box, R_\Diamond$).*

From the theorem it follows that if the satisfiability problem of GF^2_{mon} over M is decidable, then the satisfiability problem of intuitionistic modal logic over M is decidable.

It is well known that the guarded fragment is decidable over the class of all first order models [2]. Decidability of GF^2_{mon} over models with reflexive, transitive guards is proved in [14]. From this and from the fact that upward persistence for propositional variables occurring in ϕ is expressible in GF^2_{mon} it follows immediately that basic intuitionistic modal logic (with no conditions connecting $\mathcal{R}$, $\mathcal{R}_\Box$, and $\mathcal{R}_\Diamond$) is decidable. The purpose of this paper is to generalise the result of [14] to include classes of models defined using conditions involving interaction between $\mathcal{R}$, $\mathcal{R}_\Box$ and $\mathcal{R}_\Diamond$.

Theorems 14 and 12 give us our main theorem:

Theorem 15 *Let M be a class of intuitionistic modal models defined by an acyclic set of mso closure conditions on $\mathcal{R}$, $\mathcal{R}_\Box$, and $\mathcal{R}_\Diamond$ so that at most one closure condition is associated with each relation, and let ϕ be an intuitionistic modal formula. Then, it is decidable whether ϕ is satisfiable in M.*

6 Examples

In this section, we state several decidability results just to illustrate our approach.

The first example is by no means a surprise, although we doubt if anyone has proved this for all possible combinations of truth definitions for modalities. Essentially this is decidability of several flavours of basic intuitionistic modal logic (no conditions on the modal accessibility relation).

Proposition 16 *An intuitionistic modal logic L with two modalities $\Box$ and $\Diamond$, defined by a class of models where*

$$\mathcal{R} \circ \mathcal{R}_\Diamond^{-1} \circ \mathcal{R} = \mathcal{R}_\Diamond^{-1}$$
$$\mathcal{R} \circ \mathcal{R}_\Box \circ \mathcal{R} = \mathcal{R}_\Box$$

and employing any of the truth definitions for modalities $(\Box_1), (\Box_2), (\Diamond_1), (\Diamond_2)$ (in

any combination, e.g. $(\Box_1)$ *with* $(\Diamond_2)$*; possibly with more modalities, provided that all truth definitions can be translated in* GF^2_{mon}*), is decidable.*

PROOF. The class of models of L is defined by the following closure conditions on $\mathcal{R}_\Box$, $\mathcal{R}_\Diamond$ and $\mathcal{R}$:

(1) $\mathcal{R}$ is reflexive and transitive;
(2) $\mathcal{R} \circ \mathcal{R}_\Diamond^{-1} \circ \mathcal{R} = \mathcal{R}_\Diamond^{-1}$
(3) $\mathcal{R} \circ \mathcal{R}_\Box \circ \mathcal{R} = \mathcal{R}_\Box$

There is clearly at most one condition for each of the relations $\mathcal{R}$, $\mathcal{R}_\Diamond$ and $\mathcal{R}_\Box$, and the set of conditions is acyclic. We have shown in Examples 3 and 6 that the condition on $\mathcal{R}$ is a closure condition and in Example 10 that it is mso-definable. By Proposition 13, conditions on $\mathcal{R}_\Box$ and $\mathcal{R}_\Diamond$ are also mso-definable closure conditions.

We have shown that the class of models of L conforms to the conditions of Theorem 15 which proves that L is decidable.

The next example is related to a known result (decidability of PLL [11]), but for a slightly different logic (without fallible worlds):

Proposition 17 *An intuitionistic modal logic* L *with one modality* $\Diamond$*, defined by a class of models where*

$\mathcal{R}_\Diamond$ *is reflexive and transitive;*
$\mathcal{R}_\Diamond \subseteq \mathcal{R}$

and employing the truth definition $(\Diamond_2)$ *for the modality, is decidable.*

PROOF. The class of models of L is defined by the following closure conditions:

(1) $TC(\mathcal{R}_\Diamond) = \mathcal{R}_\Diamond$;
(2) $TC(\mathcal{R}) = \mathcal{R}$;
(3) $Incl^{\mathcal{R}_\Diamond}(\mathcal{R}) = \mathcal{R}$ (see Examples 4 and 7).

This set of conditions is acyclic and each condition is mso definable. However there are two constraints associated with $\mathcal{R}$: it is required to be closed both with respect to TC and to $Incl^{\mathcal{R}_\Diamond}$. To satisfy the conditions of Theorem 15 we need to combine them into one mso definable closure condition. Observe that $TC \circ Incl^{\mathcal{P}'}$ is a closure operator with the property that for any relation $\mathcal{P}$,

$$TC(Incl^{\mathcal{P}'}(\mathcal{P})) = \mathcal{P} \Leftrightarrow TC(\mathcal{P}) = \mathcal{P} \text{ and } Incl^{P'}(\mathcal{P}) = \mathcal{P}.$$

First of all, $TC \circ Incl^{\mathcal{P}'}$ is monotone and increasing, since both TC and $Incl^{\mathcal{P}'}$ are. It is also idempotent, because the result of applying $TC \circ Incl^{\mathcal{P}'}$ to any relation $\mathcal{P}$ is a transitive relation containing $\mathcal{P}'$, and any subsequent applications of $TC \circ Incl^{\mathcal{P}'}$ are not going to change it. So, $TC \circ Incl^{\mathcal{P}'}$ is a closure operator. To prove that closure with respect to this operator is equivalent to closure with respect to TC and $Incl^{\mathcal{P}'}$ separately, observe that one direction is immediate: if $\mathcal{P}$ is closed with respect to TC and $Incl^{\mathcal{P}'}$, then it is closed with respect to $TC \circ Incl^{\mathcal{P}'}$. For the other direction, assume first that

$$TC(Incl^{\mathcal{P}'}(\mathcal{P})) = \mathcal{P}$$

but $\mathcal{P}$ is not closed with respect to $Incl^{\mathcal{P}'}$, that is, it is a proper subset of $Incl^{\mathcal{P}'}(\mathcal{P})$. But since TC is increasing, $\mathcal{P}$ is then a proper subset of $TC(Incl^{\mathcal{P}'}(\mathcal{P}))$, which contradicts the assumption. Now assume that $\mathcal{P}$ is not closed with respect to TC, so that it is a proper subset of $TC(\mathcal{P})$. However, since $\mathcal{P} \subseteq Incl^{\mathcal{P}'}(P)$, we have

$$TC(\mathcal{P}) \subseteq TC(Incl^{\mathcal{P}'}(P))$$

so $\mathcal{P}$ is a proper subset of $TC(Incl^{\mathcal{P}'}(P))$, which again contradicts the assumption. This means that the conditions can be reformulated as

(1) $TC(\mathcal{R}_\Diamond) = \mathcal{R}_\Diamond$;
(2) $TC(Incl^{\mathcal{R}_\Diamond}(\mathcal{R})) = \mathcal{R}$;

and it is straightforward to show that the second condition is mso definable.

Finally, two non-examples. We failed to reformulate the condition $\mathcal{R}_\Box \circ \mathcal{R} \subseteq \mathcal{R} \circ \mathcal{R}_\Box$ defining an intuitionistic modal logic in [1] as a closure condition. We also could not apply our method to the logic IS4 defined in [29], since the truth conditions for IS4 formulas are defined on pairs (w, d) (where w is a possible world and d an element from its domain), so the image of IS4 under the standard translation is not in GF^2_{mon}.

7 Conclusions

We have described a general method for proving decidability of an intuitionistic modal logic by translating it into monadic GF^2 and showing that conditions on the intuitionistic and modal accessibility relations can be expressed using mso definable closure operators. We illustrate this method by showing that it works for various truth definitions for modalities and various conditions on the intuitionistic and modal accessibility occurring in the literature. Most of the decidability results for particular logics obtained as illustrations of our proof are already known, but we believe that our method can easily yield new results, especially for logics with

non-normal modalities defined using the truth definition ($\Diamond_2$) which are less well studied. Obviously, the same method works for intuitionistic logic with more than two modalities, provided all truth definitions can be translated in GF^2_{mon}.

References

[1] N. Alechina, M. Mendler, V. de Paiva, and E. Ritter. Categorical and Kripke semantics for constructive modal logics. In Laurent Fribourg, editor, *Proceedings of the 15th International Workshop Computer Science Logic, CSL 2001*, volume 2142 of *Lecture Notes in Computer Science*, pages 292–307. Springer, 2001.

[2] H. Andréka, J. van Benthem, and I. Németi. Modal languages and bounded fragments of predicate logic. *Journal of Philosophical Logic*, 27:217–274, 1998.

[3] N. Benton, G. Bierman, and V. de Paiva. Computational types from a logical perspective. *Journal of Functional Programming*, 8(2):177–193, 1998.

[4] G. M. Bierman and V. de Paiva. On an intuitionistic modal logic. *Studia Logica*, 65(3):383–416, 2000.

[5] R. A. Bull. A modal extension of intuitionistic modal logic. *Notre Dame Journal of Formal Logic*, VI(2):142–146, 1965.

[6] R. A. Bull. Some modal calculi based on IC. In *Formal Systems and Recursive Functions*, pages 3–7. North Holland, 1965.

[7] R. A. Bull. MIPC as the formalisation of an intuitionistic concept of modality. *Journal of Symbolic Logic*, 31(4):609–616, 1966.

[8] R. Davies and F. Pfenning. A modal analysis of staged computation. In Guy Steele, Jr., editor, *Proc. of 23rd POPL*, pages 258–270. ACM Press, 1996.

[9] R. Davies and F. Pfenning. A modal analysis of staged computation. *Journal of the ACM*, 48(3):555–604, 2001.

[10] K. Došen. Models for stronger normal intuitionistic modal logics. *Studia Logica*, 44:39–70, 1985.

[11] M. Fairtlough and M. Mendler. Propositional lax logic. *Information and Computation*, 137(1):1 – 33, 1997.

[12] G. Fisher Servi. On modal logics with intuitionistic base. *Studia Logica*, 27:533–546, 1986.

[13] F. B. Fitch. Intuitionistic modal logic with quantifiers. *Portugaliae Mathematicae*, 7:113–118, 1948.

[14] H. Ganzinger, C. Meyer, and M. Veanes. The two-variable guarded fragment with transitive relations. In *Proc. 14th IEEE Symposium on Logic in Computer Science*, pages 24–34. IEEE Computer Society Press, 1999.

[15] R. Goldblatt. Metamathematics of modal logic. *Reports on mathematical Logic*, 6,7:31 – 42, 21 – 52, 1976.

[16] J. Goubault-Larrecq. Logical foundations of eval/quote mechanisms, and the modal logic S4. Manuscript, 1996.

[17] E. Kieronski. The two-variable guarded fragment with transitive guards is 2exptime-hard. In Andrew D. Gordon, editor, *Foundations of Software Science and Computational Structures, 6th International Conference, FOSSACS 2003*, volume 2620 of *Lecture Notes in Computer Science*, pages 299–312, Warsaw, Poland, 2003. Springer.

[18] S. Kobayashi. Monad as modality. *Theoretical Computer Science*, 175:29 – 74, 1997.

[19] M. Mendler. Constrained proofs: a logic for dealing with behavioural constrains in formal hardware verification. In G. Jones and M. Sheeran, editors, *Proceedings of Workshop on Designing Correct Circuits, Oxford 1990*. Springer-Verlag, 1991.

[20] G. Mints. Some calculi of modal logic. *Trudy Matematicheskogo Instituta imeni V.A.Steklova*, 98:88–111, 1968.

[21] E. Moggi. Notions of computation and monads. *Information and Computation*, 93(1):55–92, July 1991.

[22] H. Ono. On some intuitionistic modal logics. *Publications of the Research Institute for Mathematical Science, Kyoto University*, 13:55–67, 1977.

[23] H. Ono and N.-Y. Suzuki. Relations between intuitionistic modal logics and intermediate predicate logics. *Reports on Mathematical Logic*, 22:65–87, 1988.

[24] F. Pfenning and R. Davies. A judgmental reconstruction of modal logic. *Mathematical Structures in Computer Science*, 11(4):511–540, 2001.

[25] A.M. Pitts. Evaluation logic. In G. Birtwistle, editor, *IVth Higher Order Workshop*, pages 162–189. Springer-Verlag, Banff, 1990.

[26] G. D. Plotkin and C. P. Stirling. A framework for intuitionistic modal logic. In J.Y. Halper, editor, *Theoretical Aspects of Reasoning about Knowledge*, pages 399–406, 1986.

[27] D. Prawitz. *Natural Deduction: A Proof-Theoretic Study*. Almqvist and Wiksell, 1965.

[28] M. Rabin. Decidability of second-order theories and automata on infinite trees. *Transactions of the American Mathematical Society*, 141:1–35, 1969.

[29] A. K. Simpson. *The Proof Theory and Semantics of Intuitionistic Modal Logic*. PhD thesis, University of Edinburgh, 1994.

[30] C. P. Stirling. Modal logics for communicating systems. *Theoretical Computer Science*, 49:311–347, 1987.

[31] W. Szwast and L. Tendera. On the decision problem for the guarded fragment with transitivity. In *Proceedings of the 16th Annual IEEE Symposium on Logic in Computer Science, LICS 2001*, pages 147–156, Boston, Massachusetts, USA, 2001. IEEE Computer Society.

[32] D. Wijesekera. Constructive modal logic I. *Annals of Pure and Applied Logic*, 50:271–301, 1990.

[33] F. Wolter and M. Zakharyaschev. On the relation between intuitionistic and classical modal logics. *Algebra and Logic*, 36:121–155, 1997.

[34] F. Wolter and M. Zakharyaschev. Intuitionistic modal logics. In A. Cantini, E. Casari, and P. Minari, editors, *Logic and Foundations of Mathematics*,

pages 227–238. Kluwer Academic Publishers, 1999.

[35] F. Wolter and M. Zakharyaschev. Intuitionistic modal logics as fragments of classical bimodal logics. In E. Orlowska, editor, *Logic at Work*, pages 168–186. Springer-Verlag, 1999.

[36] M. Zakharyaschev, F. Wolter, and A. Chagrov. Advanced modal logic. In Dov Gabbay, editor, *Handbook of philosophical logic*, volume 3, pages 83–266. Kluwer Academic Publishers, 2001.

Intuitionistic Hybrid Logic

Torben Braüner [a,1] Valeria de Paiva [b,2]

[a] *Department of Computer Science, Roskilde University, P.O. Box 260, DK-4000 Roskilde, Denmark*

[b] *PARC 3333 Coyote Hill Road, Palo Alto, CA 94304, USA*

Abstract

Hybrid logics are a principled generalization of both modal logics and description logics, a standard formalism for knowledge representation. In this paper we give the first constructive version of hybrid logic, thereby showing that it is possible to hybridize constructive modal logics. Alternative systems are discussed, but we fix on a reasonable and well-motivated version of intuitionistic hybrid logic and prove essential proof-theoretical results for a natural deduction formulation of it. Our natural deduction system is also extended with additional inference rules corresponding to conditions on the accessibility relations expressed by so-called geometric theories. Thus, we give natural deduction systems in a uniform way for a wide class of constructive hybrid logics. This shows that constructive hybrid logics are a viable enterprise and opens up the way for future applications.

Key words: Hybrid logic, modal logic, intuitionistic logic, natural deduction

1 Introduction

Classical hybrid logic is obtained by adding to ordinary classical modal logic further expressive power in the form of a second sort of propositional symbols called nominals, and moreover, by adding so-called satisfaction operators. A nominal is assumed to be true at exactly one world, so a nominal can be considered the name of a world. Thus, in hybrid logic a name is a particular sort of propositional symbol

Email addresses: torben@ruc.dk, Valeria.dePaiva@parc.com (Valeria de Paiva).

[1] Partially funded by the Danish Natural Science Research Council (the HyLoMOL project). Also partially funded by The IT University of Copenhagen.

[2] Partially funded by the Advanced Research and Development Activity NIMD Program (MDA904-03-C-0404).

whereas in first-order logic it is an argument to a predicate. If a is a nominal and A is an arbitrary formula, then a new formula $a : A$ called a satisfaction statement can be formed. The part $a :$ of $a : A$ is called a satisfaction operator. The satisfaction statement $a : A$ expresses that the formula A is true at one particular world, namely the world at which the nominal a is true.

The present paper concerns *constructive* hybrid logics, that is, hybrid logics where the classical logic basis has been replaced by a constructive logic basis. A question we can ask is of course *why* should one worry about constructive hybrid logics.

A first, philosophical answer might be that since we "believe" in constructive logics as well as in hybrid logics, we would like to combine them in one logical system.

A second, more mathematical answer may be simply that we should be able to define "constructive hybrid logics" since we presume that the main concerns of hybrid logic are orthogonal to whether the underline logic is constructive or not. We note that this supposition of orthogonality is justified by a distinction between the way of reasoning and what the reasoning is about. If we do define basic constructive hybrid logics and prove for them the kinds of results that we usually prove for constructive logics (normalization, subformula property, cut-elimination), we learn more about extant hybrid logics and we provide more evidence that hybrid logics are important in their own right [3] .

A third, pragmatic answer, or perhaps one geared to applications, is that if one needs to construct a logic of contexts with certain characteristics, perhaps a constructive basic hybrid logic might be the right foundation for this kind of application. The basic intuition here is that the satisfaction operators of hybrid logic might be just the syntactic tool required to construct logics that pay attention to contexts with certain desirable features. This hypothesis can only be checked, once we have defined and investigated one or more basic constructive hybrid logics, at least to some extent. Moreover, we reckon that a modal type theory based on hybrid logic could prove itself useful [4] , as other constructive modal type theories and as other classical hybrid logics have already proven themselves.

Of course, even if the supposition above, of orthogonality between hybridness and constructivity is valid, we still do not know exactly *how* to define constructive hybrid logics. We do not have a recipe for defining constructive hybrid logics: Several open possibilities are discussed, but we fix on a reasonable and well-motivated natural deduction system and prove some essential proof-theoretical results for it. Moreover, we show how to extend the system with additional inference rules corre-

[3] This would also provide evidence that these proof-theoretical properties are worth investigating for any logical system. But justifying proof-theory is not the issue here.

[4] Actually a constructive modal type theory, based on a constructive and hybrid logical version of S5, has independently of the present work been proposed by Jia and Walker [11] in 2004.

sponding to first-order conditions on the accessibility relations. The conditions we consider are expressed by so-called geometric theories. Different geometric theories give rise to different constructive hybrid logics, so natural deduction systems for new constructive hybrid logics can be obtained in a uniform way simply by adding inference rules as appropriate.

One of the hallmarks of contemporary modal logic is the view that modal logics are languages for talking about relational structures, that is, models in the sense of model-theory. Thus, modal logics are alternatives to (classical) first-order logics. This view is well justified, but we believe that proof-theory adds another important perspective to modal logics, emphasizing the notion of proof and the fundamental differences between various formal systems for representing proofs, see [14]. The perspective of this paper is proof-theoretic, so we shall focus on formal proof systems for constructive hybrid logic.

This paper is structured as follows. In the second section of the paper we make some preliminary considerations, in the third section we give a Kripke semantics, and in the fourth section we introduce a natural deduction system, and moreover, we prove a normalization theorem. The natural deduction system is an intuitionistic version of a natural deduction system for classical hybrid logic originally given in the paper [5]. In the fifth section we prove soundness and completeness with respect to Kripke semantics. In the final section we draw some conclusions and discuss future work. This paper is an extended version of [7]. While the previous version only discussed pure hybrid systems, the extended version here discusses additional inference rules corresponding to first-order conditions on the accessibility relations.

2 Preliminaries

Having decided to investigate *constructive* hybrid logics, we must describe, what their main components are. We start with a short description of constructive modal logic. This is followed by another short description of what constitutes classical hybrid logic. Finally we describe ways of putting these components together.

2.1 Constructive modal logic

'Constructive' is an umbrella term for logics that worry about deciding which disjunct is true and/or about providing witnesses for existential statements. Here we mean, much more restrictively, that we take our basis to be intuitionistic logic, henceforth IL for the propositional fragment and IFOL for intuitionistic first-order logic.

While the syntax (in axiomatic form) and the semantics (in usual Kripke style) of traditional, classical modal logics are relatively well-known, it is fair to say that even axioms for *constructive* modal logics, but especially natural deduction formalizations for these are subject to much debate.

We start by describing which basic axioms we require for a minimal K constructive modal logic. First we define our formulas. Given a basic set of propositional symbols ranged over by the metavariables p, q, r, ..., the well-formed formulas of our constructive modal logic are built by conjunction, disjunction, and implication, together with falsum and the necessity and possibility modalities. Thus, the formulas are defined by the grammar

$$S ::= p \mid S \wedge S \mid S \vee S \mid S \to S \mid \bot \mid \Box S \mid \Diamond S$$

where p is a propositional symbol. In what follows, the metavariables $A, B, C, \ldots$ range over formulas. Negation, as usual in constructive logic, is defined by the convention that $\neg A$ is an abbreviation for $A \to \bot$. Also, the nullary conjunction $\top$ is an abbreviation for $\neg\bot$ and the bi-implication $A \leftrightarrow B$ is an abbreviation for $(A \to B) \wedge (B \to A)$.

Axioms for this system consist of any basic axiomatization of intuitionistic propositional logic (IL) together with Simpson's axioms for the modalitites. These are:

$(\mathbf{taut})$ $\vdash A$ for all *intuitionistic* tautologies A

(K) $\vdash \Box(A \to B) \to (\Box A \to \Box B)$

$(\Diamond_\mathbf{1})$ $\vdash \Box(A \to B) \to (\Diamond A \to \Diamond B)$

$(\Diamond_\mathbf{2})$ $\vdash (\Diamond A \to \Box B) \to \Box(A \to B)$

$(\mathbf{dist_0})$ $\vdash \neg\Diamond\bot$

$(\mathbf{dist_1})$ $\vdash \Diamond(A \vee B) \to (\Diamond A \vee \Diamond B)$

together with the proof rules of Necessitation (Nec) and Modus Ponens (MP)

$$(Nec)\ \frac{\vdash A}{\vdash \Box A} \qquad (MP)\ \frac{\vdash A \to B \qquad \vdash A}{\vdash B}$$

We start by defining the possible-worlds semantics for the intuitionistic *modal* logic that we shall be using. We remark that this notion of a model for intuitionistic modal

logic was originally introduced in a tense-logical version by Ewald in [8] and it has also been used by Simpson in [17], p. 88.

Definition 1 *A* model *for basic intuitionistic modal logic is a tuple*

$$\mathfrak{M} = (W, \leq, \{D_w\}_{w \in W}, \{R_w\}_{w \in W}, \{V_w\}_{w \in W})$$

where

(1) W *is a non-empty set partially ordered by* $\leq$*;*
(2) for each w*,* D_w *is a non-empty set such that* $w \leq v$ *implies* $D_w \subseteq D_v$*;*
(3) for each w*,* R_w *is a binary relation on* D_w *such that* $w \leq v$ *implies* $R_w \subseteq R_v$*; and*
(4) for each w*,* V_w *is a function that to each propositional symbol* p *assigns a subset of* D_w *such that* $w \leq v$ *implies* $V_w(p) \subseteq V_v(p)$*.*

Given a model $\mathfrak{M}$ as defined above, the relation $\mathfrak{M}, w, d \models A$ is defined by induction, where w is an element of W, d is an element of D_w, and A is a formula.

$$\begin{aligned}
\mathfrak{M}, w, d &\models p \text{ iff } d \in V_w(p) \\
\mathfrak{M}, w, d &\models A \wedge B \text{ iff } \mathfrak{M}, w, d \models A \text{ and } \mathfrak{M}, w, d \models B \\
\mathfrak{M}, w, d &\models A \vee B \text{ iff } \mathfrak{M}, w, d \models A \text{ or } \mathfrak{M}, w, d \models B \\
\mathfrak{M}, w, d &\models A \rightarrow B \text{ iff for all } v \geq w, \mathfrak{M}, v, d \models A \text{ implies } \mathfrak{M}, v, d \models B \\
\mathfrak{M}, w, d &\models \bot \text{ iff falsum} \\
\mathfrak{M}, w, d &\models \Box A \text{ iff for all } v \geq w, \text{ for all } e \in D_v, dR_ve \text{ implies } \mathfrak{M}, v, e \models A \\
\mathfrak{M}, w, d &\models \Diamond A \text{ iff for some } e \in D_w, dR_we \text{ and } \mathfrak{M}, w, e \models A
\end{aligned}$$

This notion of model is very much a notion of a model of intuitionistic *quantificational* logic used simply for propositional modal logic. The introduction of D's might look an excessive complication.

Simpson's natural deduction system consists of labelled versions of the usual rules

for the propositional connectives plus the following rules for the modalities:

$$\frac{y : A \quad xRy}{x : \Diamond A}\,(\Diamond I) \qquad \frac{\begin{array}{c}[y : A][xRy]\\ \vdots \\ x : \Diamond A \qquad z : B\end{array}}{z : B}\,(\Diamond E)$$

$$\frac{\begin{array}{c}[xRy]\\ \vdots \\ y : A\end{array}}{x : \Box A}\,(\Box I) \qquad \frac{x : \Box A \quad xRy}{y : A}\,(\Box E)$$

The rules $(\Diamond E)$ and $(\Box I)$ are equipped with appropriate side-conditions in connection with the label y, that is, y does not occur in $x : \Diamond A$, in $z : B$, or in any undischarged assumptions other than the specified occurrences of $y : A$ and xRy. Moreover, y does not occur in $x : \Box A$ or in any undischarged assumptions other than the specified occurrences of xRy. Simpson's natural deduction system is different from, say natural deduction in Prawitz's book, in two aspects: First it has all of its formulas labelled; they have the form $x : A$ where x is a label, that is, a variable, and A is a formula. Intuitively, the variable x denotes a world in a modal model and the metalinguistic expression $x : A$ is to be read as 'formula A holds at world x.' Secondly, Simpson's system has relational premises on the form xRy, and similarly, relational assumptions. The metalinguistic expression xRy is to be read as 'world x sees world y.'

There are several versions of natural deduction systems for (mostly classical) modal logics of this form in the literature [1,12,16,9]. Their basic idea is always to give introduction and elimination rules for necessitiy ($\Box$) and possibility ($\Diamond$) which capture their possible worlds interpretation [5] . Recall that Simpson's particular version of natural deduction presentation satisfies normalization, as well as being "extendable" to a large collection of other logics given by so-called geometric theories. Hence, this system is proof-theoretically well-behaved, at least as far as the criteria of normalization and subformula property are concerned. A philosophical objection to this kind of system is that it builds-in the (desired) semantics into the given syntax, but the trade-off is that this way one obtains a uniform framework for a large class of modal logics.

[5] But the systems themselves and their models vary widely.

2.2 Hybrid logic

Hybrid logics are extensions of modal logics where to the usual stock of syntactic and semantic structures, we add a new kind of propositional symbols, the so-called *nominals*, which reference individual possible worlds, that is, a nominal is true at exactly one world. It is assumed that a set of ordinary propositional symbols and a countably infinite set of nominals are given. The sets are assumed to be disjoint. The metavariables p, q, r, ... range over ordinary propositional symbols and a, b, c, ... range over nominals. We also add a new kind of operators called the *satisfaction operators*. The formulas of hybrid modal logic are defined by the grammar

$$S ::= p \mid a \mid S \wedge S \mid S \vee S \mid S \to S \mid \bot \mid \Box S \mid \Diamond S \mid a : S$$

where p is an ordinary propositional symbol and a is a nominal. Formulas of the form $a : C$ are called *satisfaction statements*, cf. a similar notion in [3]. In this paper we will define a system of intuitionistic hybrid logic, using the same formulas of classical hybrid logic, which will be denoted by IHL.

There are many kinds of hybrid logics in the literature, one of the parameters being which extra operators one considers. Here we restrict our attention to only satisfaction operators, a system that in the classical case has been called $\mathsf{HL}(@)$ where $@$ is one notation for satisfaction operators.

2.3 Which Constructive Hybrid Logic?

Now there are at least two different approaches that one can take when trying to put together constructivity, modality and hybridness of logic systems.

First, one might think that since hybrid logic is obtained from classical modal logic by adding nominals and satisfaction operators, one should obtain constructive hybrid logic by adding to constructive modal logic nominals and satisfaction operators. This is sensible, and will be our approach to defining IHL, but of course it depends on the constructive modal logic chosen. Our goal is to reason constructively within constructive modal logic and we want to be able to "jump" to other modal worlds using some version of constructive satisfaction operators and modalities.

Second, one might think of "hybridizing" intuitionistic logic to enhance the expressive power of intuitionistic logic, considered as a language for talking about intuitionistic Kripke structures. Choosing this option puts an excessive emphasis on Kripke semantics as a guiding principle.

It is not clear whether these two lines of research will lead to different logical systems or not. But one suspects that the results will be different. In particular, con-

sidering only constructive modal logics, one of the reasons for the multiplicity of systems is whether one believes that only the propositional basis of the logic should be constructive or whether the modalities themselves should have a constructive component of their own. This can be ascertained by asking whether adding, say the excluded middle axiom, gives you back the classical modal system you first thought of, or not. Choosing Simpson's axiom system we're choosing that the addition of the excluded middle gives you back the classical modal logic system you started from, in our case modal K. Similarly, if the natural deduction system $\mathbf{N}_{\mathsf{IHL}}$ for intuitionistic hybrid logic given later in this paper is extended with the natural deduction rule corresponding to the excluded middle (technically, we just modify the rule for $\bot$ as appropriate), then we will get back the classical hybrid logic of [5], that is, the modal operator $\Diamond$ becomes definable in terms of $\Box$ ($\Diamond$ becomes equivalent to $\neg\Box\neg$) and we also have that $\wedge$ and $\vee$ become definable in terms of $\rightarrow$ and $\bot$.

3 Kripke Semantics

In this section we give the possible-worlds semantics for intuitionistic hybrid logic, IHL. This semantics is an extension of a possible-worlds semantics for intuitionistic modal logic which was originally introduced in a tense-logical version by Ewald [8].

The main intuition is that since we want to consider a constructive reading of hybrid logic where a distinction is made between the way of reasoning and what the reasoning is about, we need to separate the intuitionistic partial order from the interpretation of the nominals as well as the binary relation corresponding to the two modal operators.

Definition 2 *A* model *for intuitionistic hybrid logic is a tuple*

$$(W, \leq, \{D_w\}_{w\in W}, \{\sim_w\}_{w\in W}, \{R_w\}_{w\in W}, \{V_w\}_{w\in W})$$

where

(1) W *is a non-empty set partially ordered by* $\leq$*;*
(2) for each w*,* D_w *is a non-empty set such that* $w \leq v$ *implies* $D_w \subseteq D_v$*;*
(3) for each w*,* $\sim_w$ *is an equivalence relation on* D_w *such that* $w \leq v$ *implies* $\sim_w \subseteq \sim_v$*;*
(4) for each w*,* R_w *is a binary relation on* D_w *such that* $w \leq v$ *implies* $R_w \subseteq R_v$*; and*
(5) for each w*,* V_w *is a function that to each ordinary propositional symbol* p *assigns a subset of* D_w *such that* $w \leq v$ *implies* $V_w(p) \subseteq V_v(p)$*.*

It is assumed that if $d \sim_w d'$*,* $e \sim_w e'$*, and* dR_we*, then* $d'R_we'$*, and similarly, if*

$d \sim_w d'$ *and* $d \in V_w(p)$*, then* $d' \in V_w(p)$.

Intuitively, the elements of the set W are "states of knowledge" and for any such state w, the set D_w is the set of possible worlds known in the state of knowledge w, the relation $\sim_w$ corresponds to the known identities between possible worlds, and the relation R_w is the known relationship between possible worlds. Note that the definition requires that the partial order $\leq$ on the states of knowledge, which we call the "epistemic" partial order, preserves knowledge, that is, if an advance to a greater state of knowledge is made, then what is known is preserved.

Given a model $\mathfrak{M}$ and an element w of W, a *w-assignment* is a function that to each nominal assigns an element of D_w. The relation $\mathfrak{M}, g, w, d \models A$ is defined by induction, where w is an element of W, g is a w-assignment, d is an element of D_w, and A is a formula.

$$\begin{aligned}
\mathfrak{M}, g, w, d &\models p \text{ iff } d \in V_w(p) \\
\mathfrak{M}, g, w, d &\models a \text{ iff } d \sim_w g(a) \\
\mathfrak{M}, g, w, d &\models A \wedge B \text{ iff } \mathfrak{M}, g, w, d \models A \text{ and } \mathfrak{M}, g, w, d \models B \\
\mathfrak{M}, g, w, d &\models A \vee B \text{ iff } \mathfrak{M}, g, w, d \models A \text{ or } \mathfrak{M}, g, w, d \models B \\
\mathfrak{M}, g, w, d &\models A \to B \text{ iff for all } v \geq w, \mathfrak{M}, g, v, d \models A \text{ implies } \mathfrak{M}, g, v, d \models B \\
\mathfrak{M}, g, w, d &\models \bot \text{ iff falsum} \\
\mathfrak{M}, g, w, d &\models \Box A \text{ iff for all } v \geq w, \text{ for all } e \in D_v, dR_v e \text{ implies } \mathfrak{M}, g, v, e \models A \\
\mathfrak{M}, g, w, d &\models \Diamond A \text{ iff for some } e \in D_w, dR_w e \text{ and } \mathfrak{M}, g, w, e \models A \\
\mathfrak{M}, g, w, d &\models a : A \text{ iff } \mathfrak{M}, g, w, g(a) \models A
\end{aligned}$$

By convention $\mathfrak{M}, g, w \models A$ means $\mathfrak{M}, g, w, d \models A$ for every element d of D_w and $\mathfrak{M} \models A$ means $\mathfrak{M}, g, w \models A$ for every element w of W and every w-assignment g. A formula A is *valid* if and only if $\mathfrak{M} \models A$ for every model $\mathfrak{M}$. Note the difference in the interpretations of the two modal operators: The interpretation of the $\Box$ operator involves quantification over states of knowledge whereas the interpretation of $\Diamond$ does not. This is because the modal operators correspond to quantifiers in intuitionistic first-order logic where the interpretation of the $\forall$ uses the accessibility relation whereas the interpretation of $\exists$ does not.

An example of a formula valid in classical hybrid logic but not valid in the constructive semantics given here is $a : b \vee a : \neg b$, where $a : b$ is interpreted as the possible worlds $g(a)$ and $g(b)$ being related by $\sim_w$ but $a : \neg b$ is interpreted as $g(a)$ and $g(b)$ not being related by $\sim_v$ for any $v \geq w$. This formula corresponds to the formula $a = b \vee \neg a = b$ in the first-order correspondence language we introduce below. Since in intuitionistic first-order logic we do not have a general excluded middle, a constructivist thinks that this formula should only be valid, if the equality

predicate for nominals is a *decidable* predicate. (Incidentally, this formula *would* be valid if for any w, the relation $\sim_w$ is taken to be the identity on the set D_w. Thus, the relation $\sim_w$ is needed.)

Another example is the formula $a : A \leftrightarrow \neg a : \neg A$. This formula should not be valid constructively, but it should be valid classically and it is actually taken as an axiom for classical hybrid logic by some authors.

The semantics satisfies the following important proposition.

Proposition 3 *(Monotonicity) If* $\mathfrak{M}, g, w, d \models A$ *and* $w \leq v$*, then* $\mathfrak{M}, g, v, d \models A$.

PROOF. Induction in the structure of A. $\Box$

A model for intuitionistic hybrid logic can be considered a model for intuitionistic first-order logic with equality and conversely, a model for intuitionistic first-order logic with equality can be considered a model for intuitonistic hybrid logic (see [17] for the simpler correspondence between modal-logical models and intuitionistic first-order models without equality and see also [18] which gives a definition of intuitionistic first-order models with equality). The first-order language under consideration here has a 1-place predicate symbol corresponding to each ordinary propositional symbol of modal logic, a 2-place predicate symbol corresponding to the modalities, and a 2-place predicate symbol corresponding to equality. The language does not have constant or function symbols. It is assumed that a countably infinite set of first-order variables is given. The metavariables a, b, c, ... range over first-order variables. So the formulas of the first-order language we consider are defined by the grammar

$$S \ ::= \ p(a) \mid R(a,b) \mid a = b \mid S \wedge S \mid S \vee S \mid S \rightarrow S \mid \bot \mid \forall a S \mid \exists a S$$

where p is an ordinary propositional symbol of hybrid logic, and a and b are first-order variables. The connectives $\neg$, $\top$, and $\leftrightarrow$ are defined as in intuitionistic hybrid logic. Moreover, if nominals of hybrid logic are identified with first-order variables, then a w-assignment in the sense of intuitionistic hybrid logic can be considered as a w-assignment in the sense of intuitionistic first-order logic and vice versa.

Given a model $\mathfrak{M} = (W, \leq, \{D_w\}_{w \in W}, \{\sim_w\}_{w \in W}, \{R_w\}_{w \in W}, \{V_w\}_{w \in W})$ for intuitionistic hybrid logic, considered as a model for intuitionistic first-order logic, the relation $\mathfrak{M}, w \models A[g]$ is defined by induction, where w is an element of W, g is a

w-assignment, and A is a first-order formula.

$$
\begin{aligned}
\mathfrak{M}, w \models p(a)[g] &\text{ iff } g(a) \in V_w(p) \\
\mathfrak{M}, w \models a = b[g] &\text{ iff } g(a) \sim_w g(b) \\
\mathfrak{M}, w \models R(a,b)[g] &\text{ iff } g(a) R_w g(b) \\
\mathfrak{M}, w \models A \wedge B[g] &\text{ iff } \mathfrak{M}, w \models A[g] \text{ and } \mathfrak{M}, w \models B[g] \\
\mathfrak{M}, w \models A \vee B[g] &\text{ iff } \mathfrak{M}, w \models A[g] \text{ or } \mathfrak{M}, w \models B[g] \\
\mathfrak{M}, w \models A \to B[g] &\text{ iff for all } v \geq w, \mathfrak{M}, v \models A[g] \text{ implies } \mathfrak{M}, v \models B[g] \\
\mathfrak{M}, w \models \bot[g] &\text{ iff falsum} \\
\mathfrak{M}, w \models \forall a A[g] &\text{ iff for all } v \geq w, \text{ for all } g' \overset{a}{\sim} g, g'(a) \in D_v \text{ implies} \\
&\qquad \mathfrak{M}, v \models A[g'] \\
\mathfrak{M}, w \models \exists a A[g] &\text{ iff for some } g' \overset{a}{\sim} g, g'(a) \in D_w \text{ and } \mathfrak{M}, w \models A[g']
\end{aligned}
$$

By convention $\mathfrak{M} \models A$ means $\mathfrak{M}, w \models A[g]$ for every element w of W and every w-assignment g. In Section 5 we shall make use of the first-order semantics above in connection with the interpretation of geometric theories.

4 The Natural Deduction System

In this section a natural deduction system for intuitionistic hybrid logic is given and it is shown how to extend the system with additional rules corresponding to conditions on the accessibility relations. Moreover, a normalization theorem is proved. The natural deduction system is an intuitionistic version of a natural deduction system for classical hybrid logic originally given in [5]. An axiomatic formulation for intuitionistic hybrid logic can be found in the paper [4]. See the books [13] and [19] for the basics of natural deduction. See also the book [10] for an introduction to natural deduction with a slant towards intuitionistic logic.

We make use of the following conventions. The metavariables π, τ, ... range over derivations. The metavariables Γ, Δ, ... range over sets of formulas. A derivation π is a *derivation of* A if the end-formula of π is an occurrence of A and π is a *derivation from* Γ if each undischarged assumption in π is an occurrence of a formula in Γ (note that numbers annotating undischarged assumptions are ignored). If there exists a derivation of A from $\emptyset$, then we simply say that A is *derivable*. Moreover, $B[c/a]$ is the formula B where the nominal c has been substituted for all occurrences of the nominal a and $\pi[c/a]$ is the derivation π where each formula occurrence B has been replaced by $B[c/a]$. The *degree* of a formula is the number

$$\frac{a:A \qquad a:B}{a:(A\wedge B)}\,(\wedge I) \qquad \frac{a:(A\wedge B)}{a:A}\,(\wedge E1)\ \frac{a:(A\wedge B)}{a:B}\,(\wedge E2)$$

$$\frac{a:A}{a:(A\vee B)}\,(\vee I1)\ \frac{a:B}{a:(A\vee B)}\,(\vee I2) \qquad \frac{\begin{array}{ccc} & [a:A] & [a:B] \\ a:(A\vee B) & \vdots & \vdots \\ & C & C \end{array}}{C}\,(\vee E)$$

$$\frac{\begin{array}{c}[a:A]\\ \vdots \\ a:B\end{array}}{a:(A\to B)}\,(\to I) \qquad \frac{a:(A\to B) \qquad a:A}{a:B}\,(\to E)$$

$$\frac{a:\bot}{C}\,(\bot E)$$

$$\frac{a:A}{c:a:A}\,(:I) \qquad \frac{c:a:A}{a:A}\,(:E)$$

$$\frac{e:A \qquad a:\Diamond e}{a:\Diamond A}\,(\Diamond I) \qquad \frac{\begin{array}{cc} & [c:A]\ [a:\Diamond c] \\ a:\Diamond A & \vdots \\ & C \end{array}}{C}\,(\Diamond E)^{*}$$

$$\frac{\begin{array}{c}[a:\Diamond c]\\ \vdots \\ c:A\end{array}}{a:\Box A}\,(\Box I)^{\star} \qquad \frac{a:\Box A \qquad a:\Diamond e}{e:A}\,(\Box E)$$

$*$ c does not occur in $a:\Diamond A$, in C, or in any undischarged assumptions other than the specified occurrences of $c:A$ and $a:\Diamond c$.
$\star$ c does not occur in $a:\Box A$ or in any undischarged assumptions other than the specified occurrences of $a:\Diamond c$.

Fig. 1. Natural deduction rules for connectives

$$\frac{}{a : a}\,(Ref) \qquad \frac{a : c \quad a : A}{c : A}\,(Nom1)^* \qquad \frac{a : c \quad a : \Diamond b}{c : \Diamond b}\,(Nom2)$$

$*$ A is a propositional symbol (ordinary or a nominal).

Fig. 2. Natural deduction rules for nominals

$$\frac{\begin{array}{ccccc} & & [s_{11}]\ldots[s_{1n_1}] & & [s_{m1}]\ldots[s_{mn_m}] \\ & & \vdots & & \vdots \\ s_1 \ \ldots \ s_n & & C & \ldots & C \end{array}}{C}\,(R_G)^*$$

$*$ None of the nominals in $\bar{c}$ occur in C or in any of the undischarged assumptions other than the specified occurrences of s_{jk}. (Recall that nominals are identified with first-order variables and that $\bar{c}$ are the first-order variables existentially quantified over in the formula G.)

Fig. 3. Natural deduction rules for geometric theories

of occurrences of non-nullary connectives in it.

Natural deduction inference rules for intuitionistic hybrid logic are given in Figure 1 and Figure 2. All formulas in the rules are satisfaction statements. Note that the modal rules for our system are like Simpson's modal rules, except that we replaced first-order relational formulas like $R(a, c)$ by the hybrid formula $a : \Diamond c$, thereby internalizing the accessibility relation. Note that this internalization process can be accomplished for hybrid logic, but for modal logic, these relational formulas belong to the metalevel. The nominal natural deduction rules are similar to the usual equality rules for first order logic. The system thus obtained will be denoted $\mathbf{N}_{\mathsf{IHL}}$.

4.1 Extensions to Geometric Theories

In what follows we shall consider natural deduction systems obtained by extending $\mathbf{N}_{\mathsf{IHL}}$ with additional inference rules corresponding to first-order conditions on the accessibility relations. The conditions we consider are expressed by so-called geometric theories. A first-order formula is *geometric* if it is built out of atomic formulas of the form $R(a, c)$ and $a = c$ using only the connectives $\bot$, $\wedge$, $\vee$, and $\exists$. In what follows, the metavariables S_k and S_{jk} range over atomic formulas of the mentioned forms. Atomic formulas of the mentioned forms can be translated into first-order hybrid logic in a truth preserving way as follows.

$$HT(R(a, c)) = a : \Diamond c$$
$$HT(a = c) = a : c$$

See [20] for an introduction to geometric logic.

Now, a *geometric theory* is a finite set of closed first-order formulas each having the form $\forall\overline{a}(A \to B)$ where the formulas A and B are geometric, $\overline{a}$ is a list $a_1, \ldots, a_l$ of first-order variables, and $\forall\overline{a}$ is an abbreviation for $\forall a_1 \ldots \forall a_l$. It can be proved, cf. [17], that any geometric theory is intuitionistically equivalent to a *basic geometric theory* which is a geometric theory in which each formula has the form

$$(*) \qquad \forall\overline{a}((S_1 \wedge \ldots \wedge S_n) \to \exists\overline{c} \bigvee_{j=1}^{m} (S_{j1} \wedge \ldots \wedge S_{jn_j}))$$

where $n, m \geq 0$ and $n_1, \ldots, n_m \geq 1$. For simplicity, we assume that the variables in the list $\overline{a}$ are pairwise distinct, that the variables in $\overline{c}$ are pairwise distinct, and that no variable occurs in both $\overline{a}$ and $\overline{c}$. Note that a formula of the form displayed above is a Horn clause if $\overline{c}$ is empty, $m = 1$, and $n_m = 1$.

We now give hybrid natural deduction rules corresponding to a basic geometric theory. The metavariables s_k and s_{jk} range over hybrid-logical formulas of the forms $a : \Diamond c$ and $a : c$. With a first-order formula G of the form $(*)$ displayed above, we associate the natural deduction inference rule (R_G) given in Figure 3 where s_k is of the form $HT(S_k)$ and s_{jk} is of the form $HT(S_{jk})$. For example, if G is the formula

$$\forall a \forall c((R(a, c) \wedge R(c, a)) \to a = c)$$

then (R_G) is the natural deduction rule

$$\frac{\begin{array}{ccc} & & [a : c] \\ & & \vdots \\ a : \Diamond c & c : \Diamond a & C \end{array}}{C} (R_G)$$

The formula, and hence the inference rule, corresponds to the accessibility relation R being antisymmetric. Now, let $\mathbf{T}$ be any basic geometric theory. The natural deduction system obtained by extending $\mathbf{N}_{\mathsf{IHL}}$ with the set of rules $\{(R_G) \mid G \in \mathbf{T}\}$ will be denoted $\mathbf{N}_{\mathsf{IHL}} + \mathbf{T}$. We shall assume that we are working with a fixed basic geometric theory $\mathbf{T}$ unless otherwise specified.

It is straightforward to check that if a formula in a basic geometric theory is a Horn clause, then the rule (R_G) given in Figure 3 can be replaced by the following simpler rule (which we have also called (R_G)).

$$\frac{s_1 \quad \ldots \quad s_n}{s_{11}} (R_G)$$

For example, if G is the formula corresponding to the accessibility relation being antisymmetric, cf. above, then the following rule will do.

$$\frac{a : \Diamond c \qquad c : \Diamond a}{a : c}\,(R_G)$$

Natural deduction rules corresponding to Horn clauses were discussed already in [14].

4.2 An eliminable rule

Below we state a small proposition regarding an eliminable rule.

Proposition 4 *The rule*

$$\frac{a : c \quad a : A}{c : A}\,(Nom)$$

is eliminable.

PROOF. A straightforward extension of a proof in [5]. □

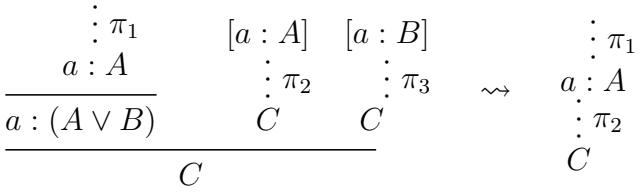
Note in the proposition above that A can be any formula; not just a propositional symbol. Thus, the rule (Nom) generalises $(Nom1)$ (and the rule $(Nom2)$ as well). The side-condition on the rule $(Nom1)$ enables us to prove a normalization theorem such that normal derivations satisfy a version of the subformula property called the quasi-subformula property. We shall return to this issue later.

4.3 Normalization

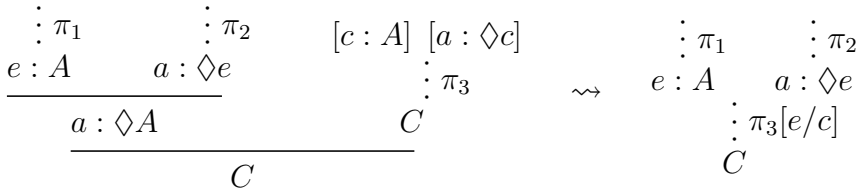
In what follows we give reduction rules for the natural deduction system and we prove a normalization theorem. First some conventions. If a premise of a rule has the form $a : c$ or $a : \Diamond c$, then it is called a *relational premise*, and similarly, if the conclusion of a rule has the form $a : c$ or $a : \Diamond c$, then it is called a *relational conclusion*. Moreover, if an assumption discharged by a rule has the form $a : \Diamond c$, then it is called a *relationally discharged assumption*. The premise of the form $a : A$ in the rule $(\rightarrow E)$ is called *minor* and the premises of the form C in the rules $(\vee E)$, $(\Diamond E)$, and (R_G) are called *parametric premises*. A premise of an elimination rule that is neither minor, relational, or parametric is called *major*.

A *maximum formula* in a derivation is a formula occurrence that is both the conclusion of an introduction rule and the major premise of an elimination rule. Maximum formulas can be removed by applying *proper reductions*. The rules for proper reductions are as follows. We have omitted the reduction rules involving the connectives $\wedge$, $\rightarrow$, :, and $\Box$ which can be found in [5].

$(\vee I1)$ followed by $(\vee E)$ (analogously in the case of $(\vee I2)$)

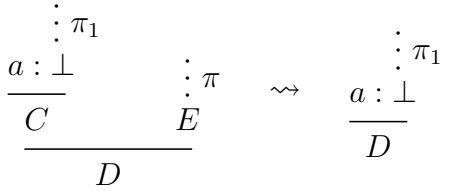


$(\Diamond I)$ followed by $(\Diamond E)$

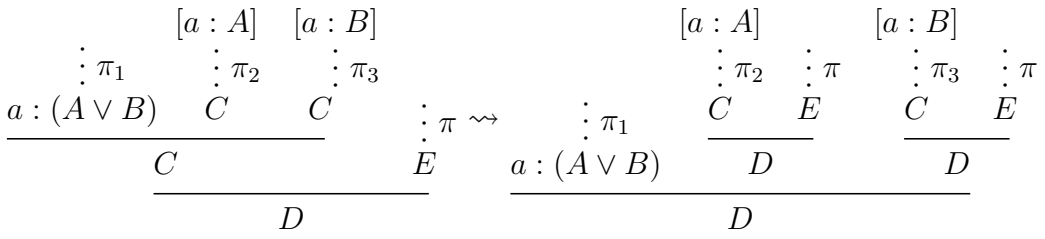


It turns out that we need further reduction rules in connection with the inference rules $(\bot E)$, $(\vee E)$, $(\Diamond E)$, and (R_G). A *permutable formula* in a derivation is a formula occurrence that is both the conclusion of $(\bot E)$, $(\vee E)$, $(\Diamond E)$, or (R_G) and the major premise of an elimination rule. Permutable formulas in a derivation can be removed by applying *permutative reductions*. The rules for permutative reductions are as follows in the case where the elimination rule has two premises. We have omitted the reduction rule where (R_G) is followed by an elimination which can be found in [5].

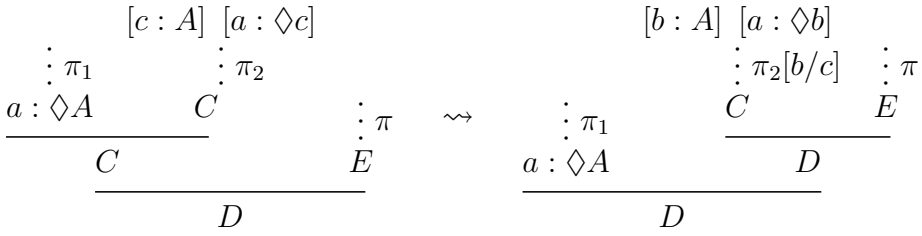
$(\bot E)$ followed by a two-premise elimination



$(\vee E)$ followed by a two-premise elimination



$(\Diamond E)$ followed by a two-premise elimination



The cases where the elimination rule has one or three premises are obtained by deleting or adding derivations as appropriate.

A derivation is *normal* if it contains no maximum or permutable formula. In what follows we shall prove a normalization theorem which says that any derivation can be rewritten to a normal derivation by repeated applications of reductions. To this end we need a number of definitions and lemmas.

Definition 5 *The* $\Diamond$-graph *of a derivation* π *is the binary relation on the set of formula occurrences in* π *of the form* $a : \Diamond c$ *which is defined as follows. A pair of formula occurrences* (A, B) *is an element of the* $\Diamond$*-graph of* π *if and only if it satisfies one of the following conditions.*

(1) A *is the relational premise of an instance of* $(\Diamond I)$ *which has* B *as the conclusion.*
(2) A *is the major premise of an instance of* $(\Diamond E)$ *at which* B *is relationally discharged.*
(3) A *is a parametric premise of an instance of* $(\vee E)$, $(\Diamond E)$, *or* (R_G) *which has* B *as the conclusion.*

Note that the $\Diamond$-graph of π is a relation on the set of formula occurrences of π; not the set of formulas occurring in π. Also, note that every formula occurrence in a $\Diamond$-graph is of the form $a : \Diamond c$.

Lemma 6 *The* $\Diamond$*-graph of a derivation* π *does not contain cycles.*

PROOF. Induction on the structure of π. $\Box$

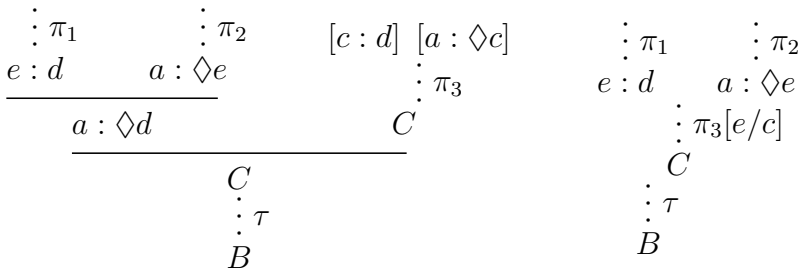
Definition 7 *The* potential *of a chain in the $\Diamond$-graph of π is the number of formula occurrences in the chain which are major premises of instances of $(\Diamond E)$. A* stubborn formula *in a derivation π is a maximum or permutable formula of the form $a : \Diamond c$ and the* stubbornness *of a stubborn formula in π is the maximal potential of a chain in the $\Diamond$-graph of π that contains the stubborn formula.*

Note that the notion of potential of a chain in the definition above is well-defined since Lemma 6 implies that the number of formula occurrences of a chain in a $\Diamond$-graph is bounded.

Lemma 8 *Let π be a derivation where all stubborn maximum formulas have stubbornness less than or equal to d and all stubborn permutable formulas have stubbornness less than d. Assume that A is a stubborn maximum formula with stubbornness d such that no formula occurrence above A is a stubborn maximum formula with stubbornness d. Let π' be the derivation obtained by applying the reduction such that A is removed.*

Then all stubborn maximum formulas in π' have stubbornness less than or equal to d and all stubborn permutable formulas in π' have stubbornness less than d, and moreover, the number of stubborn maximum formulas with stubbornness d in π' is less than the number of stubborn maximum formulas with stubbornness d in π.

PROOF. The derivations π and π' have the forms below.



Note that any formula occurrence in π' except the indicated occurrences of $e : d$, $a : \Diamond e$, and C in an obvious way can be mapped to a formula occurrence in π. Let f be the map thus defined (note that f need not be injective as the instance of $(\Diamond E)$ in π might discharge more than one occurrence of $a : \Diamond c$). Using the map f, a map from the $\Diamond$-graph of π' to the $\Diamond$-graph of π is defined as follows. There are a number of cases to consider. Case 1: An element (F, G) of the $\Diamond$-graph of π' where the formula occurrences F and G both are in the domain of f is mapped to $(f(F), f(G))$ which straightforwardly can be shown to be an element of the $\Diamond$-graph of π (observe that no assumption in π_1 or π_2 is discharged at a rule-instance in $\pi_3[e/c]$). Case 2: An element (F, G) where G is one of the indicated occurrences of $a : \Diamond e$ (and F therefore is in the domain of f) is mapped to $(f(F), G')$ where G' is the relational premise of the instance of $(\Diamond I)$. Case 3: An element (F, G)

where F is the indicated occurrence of C (and G therefore is in the domain of f) is mapped to $(F', f(G))$ where F' is the conclusion of the instance of $(\Diamond E)$. Case 4: An element (F, G) where F is one of the indicated occurrences of $a : \Diamond e$, F is different from the indicated occurrence of C, and G is in the domain of f is mapped to $(F', f(G))$ where F' is the assumption in π_3 discharged by the instance of $(\Diamond E)$ corresponding to the occurrence of $a : \Diamond e$ in question. Case 5: An element (F, G) where G is the indicated occurrence of C, G is different from each of the indicated occurrences of $a : \Diamond e$, and F is in the domain of f is mapped to $(f(F), G')$ where G' is the parametric premise of the instance of $(\Diamond I)$. Case 6: An element (F, G) where F is one of the indicated occurrences of $a : \Diamond e$ and G is the indicated occurrence of C is mapped to (F', G') where F' is the assumption in π_3 discharged by the instance of $(\Diamond E)$ corresponding to the occurrence of $a : \Diamond e$ in question and G' is the parametric premise of the instance of $(\Diamond E)$. By using the map from the $\Diamond$-graph of π' to the $\Diamond$-graph of π, any chain in the $\Diamond$-graph of π' that does not contain any of the indicated occurrences of $a : \Diamond e$ can in an obvious way be mapped to a chain in the $\Diamond$-graph of π with the same potential which does not contain the indicated occurrences of $a : \Diamond e$, $a : \Diamond d$, and $a : \Diamond c$, and similarly, any chain in the $\Diamond$-graph of π' that contains one of the indicated occurrences of $a : \Diamond e$ can in an obvious way be mapped to a chain in the $\Diamond$-graph of π with greater potential which contain the mentioned formula occurrences. The conclusions of the lemma follow straightforwardly. □

Definition 9 *A* segment *in a derivation π is a non-empty list $A_1, \ldots, A_n$ of formula occurrences in π with the following properties.*

(1) A_1 is not the conclusion of an instance of $(\vee E)$, an instance of $(\Diamond E)$, or an instance of (R_G) with more than zero parametric premises.
(2) For each $i < n$, A_i is a parametric premise of an instance of $(\vee E)$, $(\Diamond E)$, or (R_G) which has A_{i+1} as the conclusion.
(3) A_n is not a parametric premise of an instance of $(\vee E)$, $(\Diamond E)$, or (R_G).

The length *of a segment is the number of formula occurrences in the segment. A segment σ_1* stands above *a segment σ_2 if and only if the last formula occurrence in σ_1 stands above the first formula occurrence in σ_2. A* maximum segment *(*permutable segment*) is a segment in which the last formula occurrence is a maximum formula (permutable formula). A* stubborn segment *is a maximum or permutable segment where the formula that occurs in the segment is of the form $a : \Diamond c$. The* degree *of a segment is the degree of the formula that occurs in the segment.*

The following lemma is along the lines of a similar result for ordinary intuitionistic first-order logic given in [13].

Lemma 10 *Any derivation π can be rewritten to a derivation π' that does not contain permutable formulas or non-stubborn maximum formulas, by repeated applications of permutative reductions applied to permutable formulas and proper re-*

ductions applied to non-stubborn maximum formulas.

PROOF. To any derivation π we assign the pair (d, k) of non-negative integers where d is the maximal degree of a permutable or non-stubborn maximum segment in π or 0 if there is no such segment and k is the sum of the lengths of permutable and non-stubborn maximum segments in π of degree d (note that a list of formula occurrences with only one element is a segment if the one and only formula occurrence in the list is a maximum formula). The proof is by induction on such pairs equipped with the lexicographic order. Let π be a derivation to which a pair (d, k) is assigned such that $d > 0$. It is straightforward to check that there exists a permutable or non-stubborn maximum segment σ of degree d in π such that there is i) no permutable or non-stubborn maximum segment with degree d that stands above σ and ii) no permutable or non-stubborn maximum segment with degree d that stands above or contains a minor, relational, or parametric premise of the rule instance of which the last formula occurrence in σ is the major premise. Let π' be the derivation obtained by applying the appropriate reduction rule such that the last formula occurrence in σ is removed. Then it is straightforward to check that the pair (d', k') assigned to π' is less than (d, k) in the lexicographic order $\square$

We are now ready to prove the normalization theorem.

Theorem 11 (Normalization) *Any derivation can be rewritten to a normal derivation by repeated applications of proper and permutative reductions.*

PROOF. By Lemma 10 we just need to consider derivations that do not contain permutable formulas or non-stubborn maximum formulas. To any such derivation π we assign the non-negative integer d where d is the maximal stubbornness of a stubborn maximum formula in π or 0 if there is no stubborn maximum formula. Let π be a derivation to which an integer d is assigned such that $d > 0$. It is straightforward that there exists a stubborn maximum formula A with stubbornness d such that no formula occurrence above A is a stubborn maximum formula with stubbornness d. Let π' be the derivation obtained by applying the reduction such that A is removed. Then by inspecting the involved reduction rule it is trivial to check that all maximum or permutable formulas in π' are stubborn, and moreover, by Lemma 8 all stubborn maximum formulas in π' have stubbornness less than or equal to d and all stubborn permutable formulas in π' have stubbornness less than d, and furthermore, the number of stubborn maximum formulas with stubbornness d in π' is less than the number of stubborn maximum formulas with stubbornness d in π. By repeated applications of this procedure a derivation is obtained in which all maximum or permutable formulas are stubborn with stubbornness less than d. By application of Lemma 10 a derivation π'' is obtained that does not contain permutable formulas or non-stubborn maximum formulas. If all maximum or permutable formulas in a

derivation τ are stubborn with stubbornness less than d, then it is trivial to check by inspecting the involved reduction rules that all maximum or permutable formulas in the derivation τ' obtained by applying a permutative reduction are stubborn, and moreover, it can be proved in a way similar to the way in which Lemma 8 is proved, that all stubborn formulas in τ' have stubbornness less than d. Thus, all maximum formulas in π'' are stubborn with stubbornness less than d. We are therefore done by induction. $\Box$

4.4 The form of normal derivations

Below we adapt an important definition from [13] to intuitionistic hybrid logic.

Definition 12 *A* path *in a derivation π is a non-empty list $A_1, \ldots, A_n$ of formula occurrences in π with the following properties.*

(1) *A_1 is a relational conclusion, or the conclusion of a (R_G) rule with zero parametric premises, or an assumption that is not non-relationally discharged by an instance of $(\vee E)$ or $(\Diamond E)$.*
(2) *For each $i < n$, A_i is not a minor or relational premise and either*
  (a) *A_i is not the major premise of an instance of $(\vee E)$ or $(\Diamond E)$ and A_i stands immediately above A_{i+1}, or*
  (b) *A_i is the major premise of an instance r of $(\vee E)$ or $(\Diamond E)$ and A_{i+1} is an assumption non-relationally discharged by r.*
(3) *A_n is either the end-formula of π, or a minor or relational premise, or the major premise of an instance of $(\vee E)$ or $(\Diamond E)$ that does not non-relationally discharge any assumptions.*

Note that A_1 in the definition above might be a discharged assumption.

Lemma 13 *Any formula occurrence in a derivation π belongs to some path in π.*

PROOF. Induction on the structure of π. $\Box$

The definition of a path leads us to the lemma below. The lemma says that a path in a normal derivation can be split up into three parts: An analytical part in which formulas are broken down in their components by successive applications of the elimination rules, a minimum part in which an instance of the rule $(\perp E)$ may occur, and a synthetical part in which formulas are put together by by successive applications of the introduction rules. See [14].

Lemma 14 *Let $\beta = A_1, \ldots, A_n$ be a path in a normal derivation. Then there exists a formula occurrence A_i in β, called the* minimum formula *in β, such that*

(1) for each $j < i$, A_j is a major or parametric premise or the non-relational premise of an instance of $(Nom1)$;
(2) if $i \neq n$, then A_i is a non-relational premise of an introduction rule or the premise of an instance of $(\bot E)$; and
(3) for each j, where $i < j < n$, A_j is a non-relational premise of an introduction rule, a parametric premise, or the non-relational premise of an instance of $(Nom1)$.

PROOF. Let A_i be the first formula occurrence in β which is not the non-relational premise of an instance of $(Nom1)$, and is not a parametric premise, and is not the major premise of an elimination rule save possibly the major premise of an instance of $(\vee E)$ or $(\Diamond E)$ that does not non-relationally discharge any assumptions (such a formula occurrence exists in β as A_n satisfies the mentioned criteria). We are done if $i = n$. Otherwise A_i is a non-relational premise of an introduction rule or the premise of an instance of $(\bot E)$ (by inspection of the rules and the definition of a path). If A_i is the premise of an instance of $(\bot E)$, then each A_j, where $i < j < n$, is a non-relational premise of an introduction rule, or the non-relational premise of an instance of $(Nom1)$, or a parametric premise (by inspection of the rules, the definition of a branch, and normality of π). Similarly, if A_i is a non-relational premise of an introduction rule, then each A_j, where $i < j < n$, is a non-relational premise of an introduction rule or a parametric premise. □

In what follows we shall consider the form of normal derivations. To this end we give the following definition.

Definition 15 *The notion of a* subformula *is defined by the conventions that*

- *A is a subformula of A;*
- *if $B \wedge C$, $B \vee C$, or $B \to C$ is a subformula of A, then so are B and C; and*
- *if $a : B$, $\Diamond B$, or $\Box B$ is a subformula of A, then so is B.*

A formula $a : A$ is a quasi-subformula *of a formula $c : B$ if and only if A is a subformula of B.*

Now we state the theorem which says that normal derivations satisfy a version of the subformula property.

Theorem 16 *(Quasi-subformula property) Let π be a normal derivation of A from a set of satisfaction statements Γ. Any formula occurrence C in π is a quasi-subformula of A, or of some satisfaction statement in Γ, or of some relational premise, or of some relational conclusion, or of some relationally discharged assumption.*

PROOF. First a convention: The *order* of a path in π is the number of formula occurrences in π which stand below the last formula occurrence of the path. Now consider a path $\beta = A_1, \ldots, A_n$ in π of order p. By induction we can assume that the theorem holds for all formula occurrences in paths of order less than p. Note that it follows from Lemma 14 that any formula occurrence A_j such that $j \leq i$, where A_i minimum formula in β, is a quasi-subformula of A_1, and similarly, any A_j such that $j > i$ is a quasi-subformula of A_n.

We first consider A_n. We are done if A_n is the end-formula A or a relational premise. If A_n is the minor premise of an instance of $(\rightarrow E)$, then we are done by induction as the major premise belongs to a path of order less than p. If A_n is the major premise of an instance of $(\vee E)$ or $(\Diamond E)$ that does not non-relationally discharge any assumptions, then A_n is the minimum formula and hence a quasi-subformula of A_1. Now, we are done if A_1 is a relational conclusion, or an undischarged assumption, or a relationally discharged assumption. Otherwise A_1 is discharged by an instance of $(\rightarrow I)$ with a conclusion that belongs to some branch of order less than p (note that due to normality of π, A_1 is not the conclusion of a (R_G) rule with zero parametric premises).

We now consider A_1. We are done if A_1 is a relational conclusion, or an undischarged assumption, or a relationally discharged assumption. If A_1 is the conclusion of a (R_G) rule with zero parametric premises, then A_1 has the same form as the minimum formula which is a quasi-subformula of A_n. Otherwise A_1 is discharged by an instance of $(\rightarrow I)$ with a conclusion that belongs to β or to some path of order less than p. $\Box$

Note that it is a consequence of the theorem that C is a quasi-subformula of A, or of some formula in Γ, or of a formula of the form $a : c$ or $a : \Diamond c$ (since relational premises, relational conclusions, and relationally discharged assumptions are of the form $a : c$ or $a : \Diamond c$).

5 Soundness and completeness

Having given the Kripke semantics and the natural deduction system, we are now ready to prove soundness and completeness. Recall that we are working with a fixed basic geometric theory $\mathbf{T}$. A model $\mathfrak{M}$ is called a $\mathbf{T}$-*model* if and only if $\mathfrak{M} \models C$ for every formula C in $\mathbf{T}$ (note that the model $\mathfrak{M}$ in this definition is considered a first-order model as C is a first-order formula).

Theorem 17 *(Soundness) Let B be a satisfaction statement and let Γ be a set of satisfaction statements. The first claim below implies the second claim.*

(1) B *is derivable from* Γ *in* $\mathbf{N}_{\mathsf{IHL}} + \mathbf{T}$.
(2) For any $\mathbf{T}$*-model* $\mathfrak{M}$*, any element* w *of* W*, and any* w*-assignment* g*, if, for any formula* $C \in \Gamma$*,* $\mathfrak{M}, g, w \models C$*, then* $\mathfrak{M}, g, w \models B$.

PROOF. Induction on the structure of the derivation of B where we make use of Proposition 3. □

In what follows, we shall give a Henkin-type proof of completeness. In the interest of simplicity, we shall often omit reference to the basic geometric theory $\mathbf{T}$ and to the natural deduction system $\mathbf{N}_{\mathsf{IHL}} + \mathbf{T}$.

Definition 18 *A set of satisfaction statements* Γ *is* inconsistent *if and only if* $a : \bot$ *is derivable from* Γ *for some nominal* a *and* Γ *is* consistent *if and only if* Γ *is not inconsistent.*

Let $\mathbf{C}$ be any countably infinite set of nominals and let $\mathtt{L}(\mathbf{C})$ denote the set of hybrid-logical formulas built using the nominals in $\mathbf{C}$. Moreover, let $\mathbf{C}_0$ denote the set of nominals in the language defined earlier in this paper, thus, $\mathtt{L}(\mathbf{C}_0)$ denotes the language we have considered hitherto.

Definition 19 *Let* $\mathbf{C}$ *and* $\mathbf{E}$ *be disjoint countably infinite sets of nominals. A set of satisfaction statements* Γ *in the language* $\mathtt{L}(\mathbf{C} \cup \mathbf{E})$ *is* $\mathbf{E}$*-saturated if and only if*

(1) Γ *is consistent;*
(2) if A *is derivable from* Γ*, then* $A \in \Gamma$*;*
(3) if $a : (A \vee B) \in \Gamma$*, then* $a : A \in \Gamma$ *or* $a : B \in \Gamma$*;*
(4) if $a : \Diamond A \in \Gamma$*, then for some nominal* e *in* $\mathbf{E}$*,* $e : A \in \Gamma$ *and* $a : \Diamond e \in \Gamma$*; and*
(5) if $e : (s_1 \wedge \ldots \wedge s_n)[\overline{d}/\overline{a}] \in \Gamma$ *for some formula* $\forall \overline{a}((S_1 \wedge \ldots \wedge S_n) \rightarrow \exists \overline{c} \vee_{j=1}^{m} (S_{j1} \wedge \ldots \wedge S_{jn_j}))$ *in* $\mathbf{T}$ *where* $m \geq 1$*, then for some list* $\overline{b}$ *of nominals in* $\mathbf{E}$*,* $e : \vee_{j=1}^{m}(s_{j1} \wedge \ldots \wedge s_{jn_j})[\overline{d}, \overline{b}/\overline{a}, \overline{c}] \in \Gamma$.

We are now ready for a saturation lemma.

Lemma 20 *(Saturation lemma) Let* $\mathbf{C}$ *and* $\mathbf{E}$ *be disjoint countably infinite sets of nominals and let* A_1*,* A_2*,* A_3*, . . . be an enumeration of all satisfaction statements in* $\mathtt{L}(\mathbf{C} \cup \mathbf{E})$*. Let* Γ *be a set of satisfaction statements in* $\mathtt{L}(\mathbf{C})$ *and let* B *be a satisfaction statement in* $\mathtt{L}(\mathbf{C})$ *such that* B *is not derivable from* Γ*. An* $\mathbf{E}$*-saturated set of satisfaction statements* $\Gamma^* \supseteq \Gamma$ *from which* B *is not derivable is defined as follows. Firstly,* Γ^0 *is defined to be* Γ*. Secondly,* Γ^{n+1} *is defined by induction. If* B *is derivable from* $\Gamma^n \cup \{A_{n+1}\}$*, then* Γ^{n+1} *is defined to be* Γ^n*. Otherwise* Γ^{n+1} *is defined to be*

(1) $\Gamma^n \cup \{A_{n+1}, a : C\}$ *if* A_{n+1} *is of the form* $a : (C \vee E)$ *and* B *is not derivable from* $\Gamma^n \cup \{A_{n+1}, a : C\}$*;*

(2) $\Gamma^n \cup \{A_{n+1}, a : E\}$ *if* A_{n+1} *is of the form* $a : (C \vee E)$ *and the first clause does not apply;*
(3) $\Gamma^n \cup \{A_{n+1}, e : C, a : \Diamond e\}$ *if* A_{n+1} *is of the form* $a : \Diamond C$*;*
(4) $\Gamma^n \cup \{A_{n+1}, e : \vee_{j=1}^m (s_{j1} \wedge \ldots \wedge s_{jn_j})[\overline{d}, \overline{b}/\overline{a}, \overline{c}]\}$ *if there exists a formula in* $\mathbf{T}$ *of the form* $\forall \overline{a}((S_1 \wedge \ldots \wedge S_n) \rightarrow \exists \overline{c} \vee_{j=1}^m (S_{j1} \wedge \ldots \wedge S_{jn_j}))$ *such that* $m \geq 1$ *and* $A_{n+1} = e : (s_1 \wedge \ldots \wedge s_n)[\overline{d}/\overline{a}]$ *for some nominal* e *and some list* $\overline{d}$ *of nominals; and*
(5) $\Gamma^n \cup \{A_{n+1}\}$ *if none of the first four clauses apply.*

In clause 3, e *is a nominal in* $\mathbf{E}$ *that does not occur in* Γ^n *or* A_{n+1}*, and similarly, in clause 4,* $\overline{b}$ *is a list of nominals in* $\mathbf{E}$ *such that none of the nominals in* b *occur in* Γ^n *or* A_{n+1}*. Finally,* Γ^* *is defined to be* $\cup_{n \geq 0} \Gamma^n$*.*

PROOF. Firstly, B is not derivable from Γ^0 by definition. Secondly, to check that the non-derivability of B from Γ^n implies the non-derivability of B from Γ^{n+1}, we need to check each of the clauses in the definition of Γ^{n+1}. The first and fifth clauses are trivial. For the second clause, the derivability of B from $\Gamma^n \cup \{A_{n+1}, a : E\}$ implies the derivability of B from $\Gamma^n \cup \{A_{n+1}, a : C\}$ since the first clause does not apply, therefore B is derivable from $\Gamma^n \cup \{A_{n+1}\}$ by the rule $(\vee E)$. For the third clause, the derivability of B from $\Gamma^n \cup \{A_{n+1}, e : C, a : \Diamond e\}$ implies the derivability of B from $\Gamma^n \cup \{A_{n+1}\}$ by the rule $(\Diamond E)$. The fourth clause is analogous to the third clause. We conclude that B is not derivable from Γ^*. It is straightforward to check that Γ^* is $\mathbf{E}$-saturated. $\Box$

The canonical model given below is similar to a canonical model for first-order intuitionistic logic given in [18].

Definition 21 *(Canonical model) Let* $\mathbf{C}_1$*,* $\mathbf{C}_2$*,* $\mathbf{C}_3$*, ... be pairwise disjoint countably infinite sets of nominals disjoint from* $\mathbf{C}_0$ *and let* $\mathbf{C}_n^* = \cup_{1 \leq i \leq n} \mathbf{C}_i$ *where* $n \geq 1$*. Let* Γ *be a consistent set of satisfaction statements in the language* $\mathtt{L}(\mathbf{C}_0)$*. A model*

$$\mathfrak{M}^\Gamma = (W^\Gamma, \subseteq, \{D_w^\Gamma\}_{w \in W^\Gamma}, \{\sim_w^\Gamma\}_{w \in W^\Gamma}, \{R_w^\Gamma\}_{w \in W^\Gamma}, \{V_w^\Gamma\}_{w \in W^\Gamma})$$

and for each $w \in W^\Gamma$*, a* w*-assignment* g_w^Γ*, are defined as follows.*

- $W^\Gamma = \{\Delta \supseteq \Gamma \mid$ *for some* n*,* $\Delta \subseteq \mathtt{L}(\mathbf{C}_0 \cup \mathbf{C}_n^*)$ *and* Δ *is* $\mathbf{C}_n^*$*-saturated*$\}$*.*
- $D_\Delta^\Gamma = \mathbf{C}_0 \cup \mathbf{C}_n^*$ *where* Δ *is* $\mathbf{C}_n^*$*-saturated.*
- $a \sim_\Delta^\Gamma c$ *if and only if* $a : c \in \Delta$*.*
- $aR_\Delta^\Gamma c$ *if and only if* $a : \Diamond c \in \Delta$*.*
- $V_\Delta^\Gamma(p) = \{a \mid a : p \in \Delta\}$*.*
- $g_\Delta^\Gamma(a) = a$ *where* $a \in D_\Delta^\Gamma$*.*

Note that it follows from Lemma 20 that W^Γ is non-empty. It is straightforward to check the other requirements $\mathfrak{M}^\Gamma$ has to satisfy to be a model for intuitionistic hybrid logic. Given the saturation lemma and the definition of a canonical model, we are ready to prove a truth lemma.

Lemma 22 *(Truth lemma) For any $\Delta \in W^\Gamma$ and any satisfaction statement $a : A$ in $\mathtt{L}(D^\Gamma_\Delta)$, $a : A \in \Delta$ if and only if $\mathfrak{M}^\Gamma, g^\Gamma_\Delta, \Delta, a \models A$.*

PROOF. Induction on the degree of A. We only consider the case where A is of the form $\Box C$; the other cases are simpler.

Assume that $a : \Box C \in \Delta$. Let $\Lambda \supseteq \Delta$ and $aR^\Gamma_\Lambda e$, that is, $a : \Diamond e \in \Lambda$. Then $e : C \in \Lambda$ by the rule $(\Box E)$ which by the induction hypothesis implies that $\mathfrak{M}^\Gamma, g^\Gamma_\Lambda, \Lambda, e \models C$. It follows that $\mathfrak{M}^\Gamma, g^\Gamma_\Delta, \Delta, a \models \Box C$.

Assume that $a : \Box C \notin \Delta$. Assume that Δ is $\mathbf{C}^*_n$-saturated and let $e \in \mathbf{C}^*_{n+1}$. Then $e : C$ is not derivable from $\Delta \cup \{a : \Diamond e\}$ for otherwise we could derive $a : \Box C$ from Δ by the rule $(\Box I)$. According to Lemma 20, there exists a $\mathbf{C}^*_{n+2}$-saturated extension Λ of $\Delta \cup \{a : \Diamond e\}$ such that $e : C$ is not derivable from Λ. It follows by the induction hypothesis that $\mathfrak{M}^\Gamma, g^\Gamma_\Lambda, \Lambda, e \models C$ is not the case. This contradicts $\mathfrak{M}^\Gamma, g^\Gamma_\Delta, \Delta, a \models \Box C$ since $a : \Diamond e \in \Lambda$ implies that $aR^\Gamma_\Lambda e$. $\Box$

We only need one more lemma before we can prove completeness.

Lemma 23 *Let Γ be a consistent set of satisfaction statements. Then the canonical model $\mathfrak{M}^\Gamma$ is a $\mathbf{T}$-model.*

PROOF. Let $G \in \mathbf{T}$. Then G has the form $\forall \overline{a}((S_1 \wedge \ldots \wedge S_n) \rightarrow \exists \overline{c} \vee^m_{j=1} (S_{j1} \wedge \ldots \wedge S_{jn_j}))$ where $\overline{a} = a_1, \ldots, a_l$. Assume that Δ is an element of W^Γ and g is a Δ-assignment for $\mathfrak{M}^\Gamma$ such that $\mathfrak{M}^\Gamma, \Delta \models S_1[g]$, ..., $\mathfrak{M}^\Gamma, \Delta \models S_n[g]$. (Note that $\mathfrak{M}^\Gamma$ is considered a model for intuitionistic first-order logic.) So $g(a_1) = d_1$, ..., $g(a_l) = d_l$ for some nominals d_1, ..., d_l in D^Γ_Δ. Then $s_1[\overline{d}/\overline{a}], \ldots, s_n[\overline{d}/\overline{a}] \in \Delta$ by the definition of a canonical model. If $m \geq 1$, then it follows from Δ being saturated that there exists a list of nominals $\overline{b}$ such that $e : \vee^m_{j=1}(s_{j1} \wedge \ldots \wedge s_{jn_j})[\overline{d}, \overline{b}/\overline{a}, \overline{c}] \in \Delta$ where e is an arbitrary nominal. Therefore $e : (s_{j1} \wedge \ldots \wedge s_{jn_j})[\overline{d}, \overline{b}/\overline{a}, \overline{c}] \in \Delta$ and hence $s_{j1}[\overline{d}, \overline{b}/\overline{a}, \overline{c}], \ldots, s_{jn_j}[\overline{d}, \overline{b}/\overline{a}, \overline{c}] \in \Delta$ for some j where $1 \leq j \leq m$. But then it follows from the definition of a canonical model that $\mathfrak{M}^\Gamma, \Delta \models \exists \overline{c} \vee^m_{j=1} (S_{j1} \wedge \ldots \wedge S_{jn_j})[g]$. On the other hand, if $m = 0$, then $e : \bot \in \Delta$ by the rule (R_G) which contradicts the consistency of Δ. $\Box$

Now the completeness theorem.

Theorem 24 *(Completeness) Let B be a satisfaction statement and let Γ be a set of satisfaction statements. The second claim below implies the first claim.*

(1) B is derivable from Γ in $\mathbf{N}_{\mathsf{IHL}} + \mathbf{T}$.
(2) For any $\mathbf{T}$-model $\mathfrak{M}$, any element w of W, and any w-assignment g, if, for any formula $C \in \Gamma$, $\mathfrak{M}, g, w \models C$, then $\mathfrak{M}, g, w \models B$.

PROOF. Assume that B is not derivable from Γ. Consider the canonical model $\mathfrak{M}^\Gamma$ and let Λ be a $\mathbf{C}_1^*$ saturated extension of Γ from which B is not derivable, cf. Lemma 20. It follows from Lemma 22 that $\mathfrak{M}^\Gamma, g^\Gamma_\Lambda, \Lambda \models B$ is not the case but it also follows from Lemma 22 that for any $C \in \Gamma$, $\mathfrak{M}^\Gamma, g^\Gamma_\Lambda, \Lambda \models C$ is the case. But this contradicts the second statement in the theorem since $\mathfrak{M}^\Gamma$ is a $\mathbf{T}$-model by Lemma 23 $\Box$

6 Conclusions and Further Work

We have shown that constructivity and hybridness of logics are orthogonal concerns, at least to the extent that there is a logic, which we called IHL that is both hybrid and constructive. We also provided a natural deduction formulation $\mathbf{N}_{\mathsf{IHL}}$ for this logic. The system $\mathbf{N}_{\mathsf{IHL}}$ is based on Simpson's natural deduction formulation of constructive modal logic and we have shown that our hybrid version shares with it several good properties: The system is normalizing and satisfies a version of the subformula property. Hybridizing Simpson's system is not trivial, see below. Moreover, our system can be extended with inference rules corresponding to geometric first-order conditions on the accessibility relation, as can Simpson's. The possible-worlds semantics (based on Ewald's work) is sound and complete for $\mathbf{N}_{\mathsf{IHL}}$.

In the case of ordinary first-order logic, applying a reduction to a maximum formula only generates new maximum formulas having a lower degree than the original one. In fact, the technique used in the standard normalization proof for first-order logic is based on this property. The natural deduction system given in the present paper does not have this property since the reduction rules for $\Box$ and $\Diamond$ might generate new maximum formulas of the form $a : \Diamond e$, that is, maximum formulas that do not necessarily have a lower degree than the original one (here we ignore permutable formulas). Thus, the standard technique used in the normalization proof does not work directly for our case. In this paper the problem is solved by using the so-called $\Diamond$-graph of a derivation which keeps track of such maximum formulas. The notion of a $\Diamond$-graph is similar to a notion introduced in [5] to solve the same kind of problem for classical hybrid logic. We have not seen such a notion elsewhere.

We reiterate that, although the proofs are reasonably straightforward, the results are important as they confirm the presupposition in hybrid logic that it is the process of

hybridization that is central. Our work confirms that the process is at least to some extent independent of the basic logic considered. Thus our system provides some evidence of the naturalness of the hybridization process.

Concerning further work we would like to investigate whether we can adapt the "non-situated" version of natural deduction for hybrid logics, based on Seligman's work [15,6], to a constructive formulation. We would also like to consider an alternative version of basic modal logic [2] as a basis for a constructive hybrid logic. Finally we need to investigate the suitability of (all?) the logics just mentioned to the problem of modelling contexts in knowledge representation formalisms devised to deal with natural language semantics.

References

[1] D. Basin, S. Matthews, and L. Viganò. Labelled propositional modal logics: Theory and practice. *Journal of Logic and Computation*, 7:685–717, 1997.

[2] G. Bellin, V. de Paiva, and E. Ritter. Extended curry-howard correspondence for a basic constructive modal logic. In C. Areces and M. de Rijke, editors, *Workshop Proceedings of Methods for Modalities 2*. ILLC Amsterdam, 2001.

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

[4] T. Braüner. Axioms for classical and constructive hybrid logic. 2004. Manuscript. Hard-copies available from the author.

[5] T. Braüner. Natural deduction for hybrid logic. *Journal of Logic and Computation*, 14:329–353, 2004. Revised and extended version of paper in *Workshop Proceedings of Methods for Modalities 2*.

[6] T. Braüner. Two natural deduction systems for hybrid logic: A comparison. *Journal of Logic, Language and Information*, 13:1–23, 2004.

[7] T. Braüner and V. de Paiva. Towards constructive hybrid logic (extended abstract). In C. Areces and P. Blackburn, editors, *Workshop Proceedings of Methods for Modalities 3*, 2003. 15 pages.

[8] W. B. Ewald. Intuitionistic tense and modal logic. *Journal of Symbolic Logic*, 51:166–179, 1986.

[9] D. Gabbay and R.J.G.B. de Queiroz. The functional interpretation of modal necessity. In M. de Rijke, editor, *Advances in Intensional Logic*, Applied Logic Series, pages 59–91. Kluwer, 1997.

[10] J.-Y. Girard, Y. Lafont, and P. Taylor. *Proofs and Types*, volume 7 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, 1989.

[11] L. Jia and D. Walker. Modal proofs as distributed programs (extended abstract). In D. Schmidt, editor, *European Symposium on Programming*, volume 2986 of *Lecture Notes in Computer Science*, pages 219–233. Springer-Verlag, 2004.

[12] A. Masini and S. Martini. A computational interpretation of modal proofs. In H. Wansing, editor, *Proof Theory of Modal Logic*, volume 2 of *Applied Logic Series*, pages 213–241. Kluwer, 1996.

[13] D. Prawitz. *Natural Deduction. A Proof-Theoretical Study*. Almqvist and Wiksell, Stockholm, 1965.

[14] D. Prawitz. Ideas and results in proof theory. In J. E. Fenstad, editor, *Proceedings of the Second Scandinavian Logic Symposium*, volume 63 of *Studies in Logic and The Foundations of Mathematics*, pages 235–307. North-Holland, 1971.

[15] J. Seligman. The logic of correct description. In M. de Rijke, editor, *Advances in Intensional Logic*, Applied Logic Series, pages 107 – 135. Kluwer, 1997.

[16] L. Serafini and F. Giunchiglia. Ml systems: A proof theory for contexts. *Journal of Logic Language and Information*. To appear. ITC-IRST Technical Report 0006-01.

[17] A. Simpson. *The Proof Theory and Semantics of Intuitionistic Modal logic*. PhD thesis, University of Edinburgh, 1994.

[18] A. Troelstra and D. van Dalen. *Constructivism in Mathematics: An Introduction*. North-Holland, 1988.

[19] A.S. Troelstra and H. Schwichtenberg. *Basic Proof Theory*, volume 43 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, 1996.

[20] S. Vickers. *Topology via Logic*, volume 5 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, 1988.

On Modal μ-Calculus with Explicit Interpolants

G. D'Agostino [a] G. Lenzi [b]

[a] *Department of Mathematics and Computer Science, University of Udine, Viale delle Scienze 206, 33100 Udine, Italy*

[b] *Department of Mathematics, University of Pisa, Via Buonarroti 2, 56127 Pisa, Italy*

Abstract

This paper deals with the extension of Kozen's μ-calculus with the so-called "existential bisimulation quantifier". By using this quantifier one can express the uniform interpolant of any formula of the μ-calculus. In this work we provide an explicit form for the uniform interpolant of a disjunctive formula and see that it belongs to the same level of the fix-point alternation hierarchy of the μ-calculus than the original formula. We show that this result cannot be generalized to the whole logic, because the closure of the third level of the hierarchy under the existential bisimulation quantifier is the whole μ-calculus. However, we prove that the first two levels of the hierarchy are closed. We also provide the μ-logic extended with the bisimulation quantifier with a complete calculus.

Key words: μ-calculus, bisimulation quantifier, uniform interpolation.

1 Introduction

Bisimulation quantifiers were first considered in [4] and [12] as a tool for proving uniform interpolation for modal logic, and in [2] to show uniform interpolation for the modal μ-calculus. Given a formula ϕ, the language of ϕ is the set of all propositional constants appearing in the formula. The uniform interpolant of a formula ϕ with respect to a sublanguage L' of the language of ϕ is a formula ψ in the language L' which *behaves* like ϕ when L' is concerned, in the sense that ψ has the same L'-logical consequences as ϕ. A logic that contains all uniform interpolants of its formulas is said to enjoy uniform interpolation. Since uniform interpolation implies Craig interpolation, which in turn implies properties as the Beth one, a logic having uniform interpolation has a good interplay between syntactical and semantical

Email addresses: dagostin@dimi.uniud.it (G. D'Agostino), lenzi@mail.dm.unipi.it (G. Lenzi).

behaviour. Moreover, uniform interpolation allows *modularization*: if we are interested only in L'-consequences of ϕ then we may consider the (hopefully simpler) uniform interpolant ψ instead of ϕ, and *derive* all its L'-consequences (which are the same as the L'-consequences of ϕ) in an appropriate calculus: this would be like a module for this subtask. Notice that modularization is not possible if only Craig interpolation is present.

The relation between uniform interpolation and the existential bisimulation quantifier (which we denote by $\widetilde{\exists}P$) works as follows. The semantics of $\widetilde{\exists}P$ tells us that $\widetilde{\exists}P\phi(P)$ is true in a model if we can find a subset P satisfying $\phi(P)$ not only within the model, but also in any other model which is bisimilar to the given one. One can prove that any logic which is invariant under bisimulation and closed under the existential bisimulation quantifier (that is: given ϕ in the logic, there exists a formula ψ in the logic with the same semantics as $\widetilde{\exists}P\phi$) enjoys uniform interpolation: the uniform interpolant of a formula $\phi(P)$ with respect to the language $L(\phi) \setminus \{P\}$ is simply $\widetilde{\exists}P\phi(P)$. This is the way uniform interpolation is proved for the μ-calculus in [2]. However, this result does not give an explicit form for uniform interpolants in the original logic: given a μ-formula ϕ and a sublanguage L' of $L(\phi)$, we know that a uniform interpolant of ϕ with respect to L' exists in the μ-calculus, but we don't know how to construct it from ϕ. Hence, we cannot *use* the uniform interpolant to derive all L'-consequences of ϕ.

In this paper we study the relations between bisimulation quantifiers and the μ-calculus more closely. First of all we restrict our attention to disjunctive μ-formulas. The disjunctive formulas (see e.g. [7, 8]) form an important subset of the μ-calculus, because any μ-formula is equivalent to a disjunctive one, with the advantage that disjunctive formulas behave more nicely than in the general case: e.g. the problem of satisfiability for disjunctive formulas is linear. We shall see that this docility of the disjunctive formulas is confirmed when the existential bisimulation quantifier (or uniform interpolants) is concerned: if ϕ is disjunctive then $\widetilde{\exists}P\phi$ is equivalent to the formula obtained from ϕ by the simultaneous substitution of P and $\neg P$ by $\top$. This result allows us to calculate the uniform interpolant of a disjunctive formula explicitly, and to show that the uniform interpolant is not more complicate than the original formula: a good measure of the complexity of a μ-calculus formula is given by the fixpoint alternation hierarchy of the μ-calculus (proved to be strict in [1]) and from the result above it is clear that the uniform interpolant of a disjunctive formula ϕ belongs to the same level as ϕ.

We will then consider the general problem: is it true that all levels of the μ-calculus are closed under the existential bisimulation quantifier, or, equivalently, is it true that the uniform interpolant of a formula in a certain level of the hierarchy belongs to the same level? This is not a mere curiosity, because the best model checking algorithm known so far for μ-calculus formulas depends on the fixpoint alternation level of the formula: the lower the level, the easier it is to check whether the formula is true in a finite model (see e.g. [9]). For this reason it is sometime preferable to

consider not the whole μ-calculus but only formulas up to a certain level (in practice, all temporal logics used in applications can be embedded into the low levels of the hierarchy). Then the question of whether the levels of the hierarchy are closed or not under the existential bisimulation quantifier becomes relevant, because an affirmative answer would give a certification of the possibility of modularizations for the level under consideration. In particular, it would be good to know whether the low levels are closed. This closure property is already known for the 0-level of the fixpoint hierarchy, that is, it is already known that modal logic is closed under the existential bisimulation quantifier [4, 12]. In this paper we generalize this property to the levels 1 and 2 of the hierarchy. On the other hand, we see that level 3 is not closed, and more than this, that any μ-formula is obtained by considering the uniform interpolant of a formula of the third level.

Since the third level is not closed under the existential bisimulation quantifier it follows immediately that no simple rule such as $\widetilde{\exists} P\phi \leftrightarrow \phi[P/\top, \neg P/\top]$, which is valid for disjunctive formulas, can possibly hold for the whole μ-calculus. However, although we are not able to simplify so easily the existential bisimulation quantifier in the general case, we can still try to understand more precisely how this quantifier behaves w.r.t. the connectives and the operators of the μ-calculus. One way to do so is to enrich the original μ-language with an existential bisimulation quantifier $\widetilde{\exists} P$ with the appropriate semantics and provide this extended logic with a complete calculus. We shall see that to derive all validities in the extended logic we need some standard principles allowing introduction and elimination of the bisimulation quantifier, plus some natural principles of commutativity between the existential bisimulation quantifier and the operators of the μ-calculus.

Notice that, at least in principle, modularization for the μ-calculus could be obtained from modularization of the extended logic: to derive all L'-logical consequences of a μ-sentence ϕ we may go to the extended logic, write the uniform interpolant using bisimulation quantifiers and use the extended calculus to derive all consequences of it. The new logic, denoted by $\widetilde{\mu}$, is not more powerful then the original one, but in the extended language we gain the possibility to express uniform interpolants explicitly, and the complete calculus allows also to *work* with them.

The paper is organized as follows. In Section 2 we introduce the μ-calculus, give the definition of bisimulation quantifiers, and summarize the results already known about these quantifiers in the μ-calculus context. In Section 3 we calculate the uniform interpolant for disjunctive formulas. In Section 4 we prove the results concerning the fixpoint alternation hierarchy and the bisimulation quantifier. In the last section we find a complete calculus for the μ-logic extended with the existential bisimulation quantifier.

2 Notation and Preliminaries

2.1. *The μ-calculus*

First of all, we recall the definition of the extension of modal logic known as the modal μ-calculus.

Definition 2.1 *The* μ-calculus *is defined as the least set which contains a set of propositional constants $Prop$, a set of variables Var, and satisfies: if $\phi, \psi \in \mu$ then $\neg\phi, \phi \vee \psi, \Diamond\phi$ belong to μ; if $X \in Var$ and X occurs just positively in ϕ (that is: under an even number of negations) then $\mu X\phi$ belongs to μ.*

Remark 2.2 *The μ-calculus is usually defined as the extension of* multi-*modal logic, where a set of actions A and a corresponding set of operators $\Diamond_a$ are considered. For semplicity we restrict ourselves to the case in which only one action is present, although our results can easily be extended to the general case (by generalizing the covers-syntax as it is done in [7]).*

The derived operators $\phi \wedge \psi, \phi \rightarrow \psi$, $\phi \leftrightarrow \psi, \Box\phi$, and $\nu X.\phi$ are defined as usual. The variable X is said to be bound in $\mu X.\phi$, $\nu X.\phi$. Free variables in a formula and sentences are defined as usual. If ϕ is a formula, then $L(\phi)$ is defined as the set of propositional constants and free variables occurring in ϕ. We call ϕ a *modal formula* if it is constructed without using the fixpoint operators.

A μ-calculus formula is interpreted in structures for the language $\{r, R\} \cup Prop$; these are Kripke-structures, i.e. tuples of the form

$$M = (D^M, r^M, R^M, P_1^M, \ldots),$$

where the non-empty set D^M is the domain of M, r^M is an element of this domain, R^M is a binary relation on D^M, and P^M is a subset of D^M, for any $P \in Prop$. Given a structure M and a valuation $V : Var \rightarrow \wp(D^M)$, a μ-formula is interpreted in M as a subset $||\phi||_V$ of D^M, defined as follows:

$$\begin{aligned}
||P||_V &:= P^M \\
||X||_V &:= V(X) \\
||\neg\phi||_V &:= D^M \setminus ||\phi||_V \\
||\phi \vee \psi||_V &:= ||\phi||_V \cup ||\psi||_V \\
||\Diamond\phi||_V &:= \{s \in D^M \mid ||\phi||_V \cap \{t : sR^M t\} \neq \emptyset\} \\
||\mu X.\phi||_V &:= \bigcap\{S \subseteq D^M \mid ||\phi||_{V[X:=S]} \subseteq S\}
\end{aligned}$$

where $V[X := S]$ is equal to the valuation function V except that S is assigned to X. Note that $||\mu X.\phi||_V$ is the least fixpoint of the monotone operator $S \mapsto ||\phi||_{V[X:=S]}$.

In the following, we denote $s \in ||\phi||_V$ by $(M, s, V) \models \phi$ and we may leave out the valuation, if ϕ is a sentence. $(M, V) \models \phi$ is used to denote $(M, r^M, V) \models \phi$.

$\Gamma \models \phi$ denotes logical consequence: if $(M, V) \models \Gamma$ then $(M, V) \models \phi$ for every model M and for every valuation V.

An alternative syntax for the μ-calculus is obtained by substituting the $\Diamond$ operator with a set of *cover operators*, one for each natural n. For $n \geq 1$ these operators are defined as follows: if $\phi_1, \ldots, \phi_n$ are formulas, then

$$Cover(\phi_1, \ldots, \phi_n)$$

is a formula. We also allow the constant operator $Cover(\emptyset)$. The cover operators are interpreted in a Kripke structure M as follows: $Cover(\emptyset)$ is true in M if and only if the root of M does not have any successor, while $Cover(\phi_1, \ldots, \phi_n)$ is true in M if and only if the successors of the root are *covered* by $\phi_1, \ldots, \phi_n$. More formally, $(M, s, V) \models Cover(\phi_1, \ldots, \phi_n)$ if and only if:

(1) for every $i = 1, \ldots, n$ there exists t with $(s, t) \in R^M$ and $(M, t, V) \models \phi_i$;
(2) for every t with $(s, t) \in R^M$ there exists $i \in \{1, .., n\}$ with $(M, t, V) \models \phi_i$.

We call this syntax the *covers-syntax* to distinguish it form the original $\Diamond$-syntax.

Since $Cover(\phi_1, \ldots, \phi_n)$ is equivalent to

$$\Diamond(\phi_1) \wedge \ldots \wedge \Diamond(\phi_n) \wedge \Box(\phi_1 \vee \ldots \vee \phi_n),$$

cover operators are definable in the $\Diamond$ syntax. Conversely,

$$\Diamond\phi \Leftrightarrow Cover(\phi, \top).$$

Hence, the μ-calculus obtained from the covers-syntax is equivalent to the familiar μ-calculus constructed using the $\Diamond$-syntax. In this paper we use the covers-syntax because, as we shall see, cover operators behave nicely with respect to the existential bisimulation quantifier.

We now introduce an important class of μ-formulas.

Definition 2.3 *The class of* disjunctive μ-formulas *is the least class containing* $\top, \bot$*, and non-contradictory conjunction of literals which is closed under:*

(1) disjunctions;

(2) special conjunctions: if $\phi_1, \ldots, \phi_n$ are formulas in the class and σ is a non-contradictory conjunction of literals, then $\sigma \wedge Cover(\phi_1, \ldots, \phi_n)$ is in the class;
(3) fixpoint operators: if ϕ is disjunctive, ϕ does not contain $X \wedge \gamma$ as a subformula for any formula γ, and X is positive in ϕ, then $\mu X.\phi, \nu X.\phi$ are in the class.

The disjunctive formulas are representative of the whole μ-calculus:

Theorem 2.4 (In [7]) *Any μ-calculus formula is equivalent to a disjunctive one.*

The same is true (but the proof is easier) for the class of modal formulas with respect to disjunctive modal formulas (which are defined as in Definition 2.3 but without closing for fixpoint operators).

2.2. *Bisimulation Quantifiers and Uniform Interpolation*

To introduce bisimulation quantifiers we first need the notion of *bisimulation*.

Definition 2.5 *Let M, N be structures with D^M, D^N as respective domains. Let $Prop' \subseteq Prop$. A relation $Z \subseteq D^M \times D^N$ is a $Prop'$-*bisimulation *between M and N if:*

(1) $r^M Z r^N$;
(2) if wZv then $w \in P^M$ iff $v \in P^N$, for every $P \in Prop'$;
(3) if wZv and $wR^M w'$, then there exists a v' such that $vR^N v'$ and $w'Zv'$;
(4) if wZv and $vR^N v'$, then there exists a w' such that $wR^M w'$ and $w'Zv'$.

*Two structures M, N are $Prop'$-*bisimilar *(notation: $M \sim_{Prop'} N$) if there exists a $Prop'$-bisimulation between them (but we write $M \sim N$ if $Prop' = Prop$).*

If $Var' \subseteq Var$ and V_1, V_2 are valuations of the variables in Var' in the structures M, N, respectively, a bisimulation Z between (M, V_1) and (N, V_2) is a bisimulation between M and N such that if wZv then $w \in V_1(X)$ iff $v \in V_2(X)$, for every $X \in Var'$. We use the notation $(M, V_1) \sim (N, V_2)$ accordingly.

An *existential bisimulation quantifier* is defined as a classical monadic second order quantifier, except that we look for a subset satisfying a certain property not only within the model, but also in any other model which is bisimilar to the given one:

Definition 2.6 (Existential Bisimulation Quantifier)
We enrich the μ-grammar with a propositional quantifier $\widetilde{\exists} P\phi$, for any $P \in Prop$, whose semantics is defined as follows. If M is a structure and V is a valuation of the free variables of ϕ, then (M, V) satisfies the formula $\widetilde{\exists} P\phi$ iff:

(1) there exists a structure N and a valuation V' of the set $Free(\phi)$ of the free variables of ϕ with $(M, V) \sim_{Prop \setminus \{P\}} (N, V')$;

(2) there exists a subset $P \subseteq D^N$ such that $(N, P, V') \models \phi$, where (N, P, V') stands for the structure which is like (N, V') except for the propositional constant P that receives now a new interpretation (denoted again by P).

The set of formulas obtained using the extended grammar is denoted by $\widetilde{\mu}$.

Example 1. The sentence $\widetilde{\exists} P(\Diamond P \wedge \Diamond \neg P)$ is true in M iff the root has at least one successor. In fact, in a model, there exists a P such that $\Diamond P \wedge \Diamond \neg P$ holds if and only if the root has at least two successors (just put one successor of the root in P and one out of P); and a model M is bisimilar to one where the root has two successors if and only if, in M, the root has at least one successor.

Example 2. As we shall see (Theorem 2.9) the logic $\widetilde{\mu}$ has the same expressive power as the μ-calculus. However, the bisimulation quantifier can help to express a certain property in a way that is closer to natural language than the formulation of the property in the μ-calculus. For example, suppose we are interested in the existence of an infinite path starting from the root where P holds infinitely often. If we think of X as the path we are looking for, it is not difficult to see that this property is expressed by the monadic second order formula $\exists X F(X, P)$, where $\exists$ is the standard monadic second order existential quantifier,

$$F(X, P) = X \wedge \Box^*(X \rightarrow \Diamond^+(X \wedge P)),$$

the operator $\Box^*\phi$ is a shorthand for the μ-calculus formula $\nu X(\phi \wedge \Box X)$ and means that ϕ is true in every descendant of the current point, and $\Diamond^+\phi$ is a shorthand for $\mu X \Diamond(\phi \vee X)$ and means that ϕ is true in some proper descendant of the current point.

Since the existence of such a path is a property which is invariant under bisimulation, one can easily see the same property can be expressed in $\widetilde{\mu}$ by $\widetilde{\exists} X F(X, P)$. Compare now this formula with the μ sentence expressing the same property:

$$\nu X \mu Y((P \wedge \Diamond X) \vee \Diamond Y).$$

The μ formula is shorter, but it is not so easy for a non-specialist to find it or even just to check that the formula works.

Next, we consider the notion of uniform interpolant.

Definition 2.7 *Given a μ-sentence ϕ and a language $L' \subseteq L(\phi)$, the* uniform interpolant *of ϕ with respect to L' is a μ-sentence θ such that:*

(1) $\phi \models \theta$;
(2) whenever $\phi \models \psi$ and $L(\phi) \cap L(\psi) \subseteq L'$ then $\theta \models \psi$;
(3) $L(\theta) \subseteq L'$.

In [2] it is proved that the μ-calculus enjoys uniform interpolation, in the sense that for any μ-sentence ϕ and $L' \subseteq L(\phi)$ there exists a μ-sentence θ satisfying the above properties. Notice that uniform interpolation is stronger than Craig interpolation, which states that for any two formulas ϕ, ψ with $\phi \models \psi$ there exists a formula θ in the common language (called the Craig interpolant of ϕ, ψ) with $\phi \models \theta$ and $\theta \models \psi$: if the logic enjoys uniform interpolation then the Craig interpolant of ϕ, ψ does not depend on ψ but only on the common language: it is simply the uniform interpolant of ϕ relative to $L' = L(\phi) \cap L(\psi)$. This explains why we call this formula a *uniform* interpolant: no information is needed about the formula ψ except which non-logical symbols it has in common with ϕ.

The proof of uniform interpolation for the μ-calculus is obtained by considering first the case of $L' = L(\phi) \setminus \{P\}$ and then iterating the construction. In the case of $L' = L(\phi) \setminus \{P\}$ it can be proved that any μ-formula which is equivalent (in the $\widetilde{\mu}$-semantics) to $\widetilde{\exists} P\phi$ is a uniform interpolant of ϕ with respect to the language L':

Theorem 2.8 (In [2]) *If ϕ is a μ-formula and θ is such that*

$$\models \theta \leftrightarrow \widetilde{\exists} P\phi,$$

then θ is a uniform interpolant of ϕ w.r.t. $L(\phi) \setminus \{P\}$.

Then, uniform interpolation for the μ-calculus follows if one proves that the μ-calculus is closed under the existential bisimulation quantifier:

Theorem 2.9 (In [2]) *If ϕ is a μ-formula, there exists a μ-formula θ with $L(\theta) \subseteq L(\phi) \setminus \{P\}$ such that*

$$\models \theta \leftrightarrow \widetilde{\exists} P\phi.$$

3 Explicit Uniform Interpolants for Disjunctive Formulas

In this section we prove our first result concerning the existential bisimulation quantifier and the μ-calculus. We restrict our attention to the class of disjunctive formulas and prove that if $\phi(P, \neg P)$ belongs to this class, then its uniform interpolant $\widetilde{\exists} P\phi$ is equivalent to $\phi[P/\top, \neg P/\top]$. This can be proved using the correspondence between disjunctive formulas and nondeterministic automata as defined in [7]. We first recall the definition of this kind of automata.

Definition 3.1 *A nondeterministic parity modal automaton is a tuple*

$$A = \langle Q_A, \Sigma_A, q_{0,A}, \delta_A, \Omega_A \rangle,$$

such that:

- *Q_A is a finite set of states;*

- Σ_A *is a finite alphabet (the powerset of a finite subset* $Prop'$ *of* $Prop$*);*
- $q_{0,A} \in Q_A$ *is the initial state;*
- Ω_A *is a function from* Q_A *to the natural numbers;*
- δ_A *is a function which associates to every* $q \in Q_A$ *and* $\sigma \in \Sigma_A$ *a set* $\{D_1, \ldots, D_n\}$ *of subsets of* Q_A*.*

We omit the A-subscript when possible.

The acceptance condition of nondeterministic modal automata is usually defined by a game, but if we restrict to ω-expanded trees we can also describe acceptance in terms of labelings (a proof of this result can be found in [5], Lemma 3.4.3 and Lemma 3.4.4). Here we adopt the labeling condition as a definition of acceptance directly.

Recall that *an* ω*-expanded tree* is a tree T in which for every $s, s' \in T$ if $s' \in Succ(s)$ then there are at least ω distinct successors of s which are bisimilar to s'. A Σ-valued tree is a tree in which nodes are labeled by elements in Σ (we denote the label of s by $T(s)$).

Definition 3.2 *Let* $A = \langle Q, \Sigma, q_0, \delta, \Omega \rangle$ *be a nondeterministic modal automaton and* T *a* Σ*-valued tree. A total function* $l : T \to Q$ *is an* A-labeling *(also called an* accepting run *for* A *over* T*) if:*

(1) $l(r^T) = q_0$*;*
(2) If $l(s) = q$ *then* $\{q' \in Q : \exists t \in Succ(s), l(t) = q'\}$ *belongs to* $\delta(q, T(s))$*;*
(3) For any infinite T*-path* $s_0 = r^M, s_1, \ldots$*:*

$$min\{\Omega(q) \mid \text{there are infinitely many } s_i \text{ in the path s.t. } l(s_i) = q\}$$

is even.

Definition 3.3 *Given a nondeterministic modal automaton* A*, we say that an* ω*-expanded tree* T *is* accepted *by* A *iff it has an* A*-labeling.*

Nondeterministic modal automata are automata-theoretic counterparts of disjunctive μ-formulas, in the sense that to any such formula it corresponds a nondeterministic automaton accepting the same ω-expanded trees, and vice versa. To describe the explicit form of the uniform interpolant of a disjunctive formula we shall need a direct translation from automata to disjunctive formulas, as described in [6]. Let us describe how this translation is achieved. Given an automaton, we first put it in *tree normal form*:

Definition 3.4 *The automaton* $A = \langle Q, \Sigma, q_0, \delta, \Omega \rangle$ *is said to be in tree normal form if there is an order relation* $\leq_A$ *between the states of* A *which satisfies the following properties:*

(1) q_0 *is the minimum state of* Q *with respect to* $\leq_A$*;*

(2) if the states q_1, q_2, q_3 are such that $q_2, q_3 \leq_A q_1$, then q_2, q_3 are $\leq_A$-comparable;
(3) if $q_2 \in D \in \delta(q_1, \sigma)$, then either q_2 is an immediate $\leq_A$ successor of q_1 or $q_2 \leq_A q_1$ and $\Omega(q_2) \leq \Omega(q)$ for all q such that $q_2 \leq_A q \leq_A q_1$.

One can prove that for any automaton A there exists an equivalent automaton in tree normal form (see [6]). If A is in tree normal form and q is an A-state, we define the disjunctive formula $\phi_{A,q}$ by induction on the tree like form of A, from the leaves towards the root:

$$\phi_{A,q} = (\sigma_q X_q) \bigvee_{\sigma \in \Sigma_A} \bigvee_{D \in \delta(q,\sigma)} \hat{\sigma} \wedge Cover(\{\beta_{q,q'} : q' \in D\}),$$

where:

(1) $\hat{\sigma} = \bigwedge_{P \in \sigma} P \wedge \bigwedge_{P \notin \sigma} \neg P$;
(2) $\sigma_q = \nu$ if $\Omega(q)$ is even, $\sigma_q = \mu$ if $\Omega(q)$ is odd;
(3) $\beta_{q,q'} = X_{q'}$ if $q' \leq_A q$, $\beta_{q,q'} = \phi_{A,q'}$, otherwise.

Let $\phi_A = \phi_{A,q_0}$: then it is possible to prove (see [6]) that ϕ_A is equivalent to A.

Regarding bisimulation quantifiers, in [2] it is proved that they correspond to projections in the automata settings. If P is a proposition and $A = \langle Q, \Sigma, q_0, \delta, \Omega \rangle$ is an automaton, we define the automaton $\widetilde{\exists} PA = \langle Q, \Sigma', q_0, \delta', \Omega \rangle$ as follows:

(1) $\Sigma' = \{\sigma' : \sigma' \in \Sigma, P \notin \sigma'\}$;
(2) $\delta'(q, \sigma') = \delta(q, \sigma') \cup \delta(q, \sigma' \cup \{P\})$.

Then:

Theorem 3.5 (In [2]) *An ω-expanded tree T is* accepted *by $\widetilde{\exists} PA$ if and only if there exists an ω-expanded tree S which is accepted by A and is bisimilar to T with respect to $Prop \setminus \{P\}$.*

We can now prove:

Theorem 3.6 *The uniform interpolant $\widetilde{\exists} P\phi$ of a disjunctive μ-formula ϕ is equivalent to the μ-formula $\phi[P/\top, \neg P/\top]$, where $\phi[P/\top, \neg P/\top]$ is defined from ϕ by simultaneously substituting the literals P and $\neg P$ with $\top$.*

PROOF. Let A be a nondeterministic automaton in tree normal form which is equivalent to ϕ. From Theorem 3.5 it follows that the uniform interpolant of ϕ is equivalent to the automaton $\widetilde{\exists} PA$; hence, we are just left to verify that $\phi_{\widetilde{\exists} PA}$ is equivalent to $\phi[P/\top, \neg P/\top]$. We prove by induction on the tree structure of the set of states of A that

$$\phi_{(A',q)} \text{ is equivalent to } \phi_{A,q}[P/\top, \neg P/\top], \quad \text{for all } q \in Q$$

and when $q = q_0$ we obtain the desired result. We have:

$$\phi_{A',q} = (\sigma_q X_q) \bigvee_{\sigma' \in \Sigma'} \bigvee_{D' \in \delta(q,\sigma') \cup \delta(q,\sigma' \cup \{P\})} \hat{\sigma'} \wedge Cover(\{\beta'_{q,q'} : q' \in D'\},$$

where

$$\beta'_{q,q'} = \begin{cases} X_{q'} & \text{if } q' \leq_A q; \\ \phi_{A',q'} & \text{otherwise.} \end{cases}$$

On the other hand, the formula $\phi_{A,q}[P/\top, \neg P/\top]$ is

$$(\sigma_q X_q) \bigvee_{\sigma \in \Sigma} \bigvee_{D \in \delta(q,\sigma)} \hat{\sigma}[P/\top, \neg P/\top] \wedge Cover(\{\beta_{q,q'}[P/\top, \neg P/\top] : q' \in D\},$$

where

$$\beta_{q,q'} = \begin{cases} X_{q'} & \text{if } q' \leq_A q; \\ \phi_{A,q'} & \text{otherwise.} \end{cases}$$

Using induction the equivalence between $\phi_{(A',q)}$ and $\phi_{A,q}[P/\top, \neg P/\top]$ easily follows. □

From Theorem 3.6 it is intuitively clear that the uniform interpolant of a disjunctive formula is not more *complex* than the original formula. Before stating the result, however, we need a precise notion of complexity. One possibility is given by the syntactical hierarchy, whose definition is recalled in the next section.

4 The Existential Bisimulation Quantifiers and the Fixed Point Hierarchy

We consider now the behaviour of the existential bisimulation quantifier with respect to the fixpoint alternation levels of the μ-calculus. As an easy corollary of Theorem 3.6 we have that the uniform interpolant of a disjunctive formula ϕ belongs to the same level as ϕ. As we shall see, this is not true in general: in this section we prove that levels $0, 1$ and 2 are closed under the existential bisimulation quantifier, while the third level is not: the closure of this level is the whole μ-calculus.

In this section we consider the μ-formulas as constructed from a set of propositional constants $Prop$, their negations $\{\neg P : P \in Prop\}$, a set of variables Var, using the following operators: if $\phi_1, \ldots, \phi_n \in \mu$ and X in Var then $\phi_1 \vee \phi_2, \phi_1 \wedge \phi_2, Cover(\phi_1, \ldots, \phi_n), \mu X \phi_1$, and $\nu X \phi_1$ belong to μ.

Since $\Box(\phi)$ is semantically the same as $Cover(\emptyset) \vee Cover(\phi)$, this definition is equivalent to the one adopted in the previous sections, but it avoids the use of explicit negation.

Definition 4.1 *The* fixpoint alternation-depth hierarchy *of the μ-calculus is the sequence $N_0 = M_0, N_1, M_1, \ldots$ of sets of μ-formulas defined inductively as follows.*

(1) $N_0 = M_0$ is defined as the set of all modal fixpoint free formulas over the covers-signature.
(2) N_{k+1} is the closure of $N_k \cup M_k$ under the operations described in $(a), (b)$ below.
  (a) (Positive Substitution) If $\phi(P_1, \ldots P_n), \phi_1, \ldots, \phi_n$ are in N_{k+1}, then $\phi(\phi_1 \ldots \phi_n)$ is in N_{k+1}, provided $P_1, \ldots, P_n$ are positive in ϕ and no occurrence of a variable which was free in one of the ϕ_i becomes bound in $\phi(\phi_1 \ldots \phi_n)$.
  (b) If ϕ is in N_{k+1}, then $\nu X.\phi \in N_{k+1}$.
(3) Likewise, M_{k+1} is the closure of $N_k \cup M_k$ under positive substitution and the μ-operator.

From this definition and Theorem 3.6 it follows easily:

Corollary 4.2 *If ϕ is a disjunctive formula which belongs to the level N_k of the fixpoint alternation-depth hierarchy and $L' \subseteq L(\phi)$, then the uniform interpolant of ϕ w.r.t. L' belongs to the same level N_k.*

To generalize this result from disjunctive formulas to arbitrary μ-formulas we need to restrict ourselves to the levels N_0, N_1, N_2: we shall prove that for $i = 0, 1, 2$ any $\phi \in N_i$ is equivalent to a disjunctive formula in N_i, and then use Theorem 3.6. The fact that any μ-formula of these levels is equivalent to a disjunctive one of the same level is well known but we found no reference in the literature except for the zero level. For the sake of completeness we sketch here a proof which is an adaptation of the proof in [7, 6] that any μ-formula is equivalent to a disjunctive one. This result is proved by using tableaux for μ-formulas. We first recall some definitions.

A μ-formula γ is *guarded* if every occurrence of the bound variable X in every subformula $\tau X.\alpha$ (for $\tau \in \{\mu, \nu\}$) is under the scope of a cover, and it is *well-named* if, for every bound variable X, there exists only one subformula of type $\tau X.\alpha$ in γ. It is possible to prove that any μ-formula is equivalent to a positive, well-named, and guarded formula [10, 8], and that this formula can be found in linear time [11]. In this section we will only deal with this kind of formulas (henceforth simply called *formulas*). A bound variable X of such a formula γ is a μ-variable (ν-variable) if $\mu X.\alpha$ is a subformula of γ ($\nu X.\alpha$, respectively). We define a partial order $\preceq_\gamma$ on the set of the bound variables of a formula γ as the least partial order such that if $\tau X.\alpha$ is a γ-subformula, $\tau' Y.\beta$ is an α-subformula, and X occurs in β then $X \preceq_\gamma Y$. In other words, the bound variable Y is an immediate successor of the bound variable X if the scope of X contains Y and X occurs free in the scope of Y.

Remark 4.3 *It follows easily from Definition 4.1 that a formula $\gamma \in N_1$ only contains ν-variables, while if a formula γ belongs to N_2 then there is no pair of variables (X, Y) appearing in γ where X is a μ-variable, Y is a ν-variable, and*

$X \preceq_\gamma Y$.

Definition 4.4 *A* tableau $\mathbf{T} = (T, L)$ *for a* μ*-formula* γ *is a tree* T *(which we think as growing upwards) with a labeling function* L *such that the root is labeled by the set* $\{\gamma\}$ *and the sons of every node are created and labeled with sets of formulas according to the following rules:*

$$\frac{\{\alpha\} \cup \Gamma \quad \{\beta\} \cup \Gamma}{\{\alpha \vee \beta\} \cup \Gamma}; \quad (\textit{ or })$$

$$\frac{\{\alpha, \beta\} \cup \Gamma}{\{\alpha \wedge \beta\} \cup \Gamma}; \quad (\textit{ and })$$

$$\frac{\{\alpha\} \cup \Gamma}{\{\tau X.\alpha\} \cup \Gamma}; \quad (\textit{ fixed points })$$

$$\frac{\{\alpha\} \cup \Gamma}{\{X\} \cup \Gamma} \textit{ where } \tau X.\alpha \textit{ is a subformula of } \gamma; \quad (\textit{ reg })$$

if $\Gamma = \{Cover(\mathcal{F}_1), \ldots, Cover(\mathcal{F}_n)\} \cup \Delta$*, where* Δ *contains only propositional constants or negated propositional constants, then*

$$\frac{\ldots \{\alpha\} \cup \{\vee \mathcal{F}_j : j \neq i\} \ldots}{\Gamma} \quad (\textit{ mod })$$

where we have a successor labelled $\{\alpha\} \cup \{\vee \mathcal{F}_j : j \neq i\}$ *for each* $i \in \{1, \ldots, n\}$ *and* $\alpha \in \mathcal{F}_i$*.*

For example, an instance of the last rule is:

$$\frac{\{\alpha_1, \alpha_3\} \quad \{\alpha_2, \alpha_3\} \quad \{\alpha_1 \vee \alpha_2, \alpha_3\}}{\{Cover(\alpha_1, \alpha_2), Cover(\alpha_3), P, \neg Q\};}$$

if a node n in the tableau is labelled by $\{Cover(\alpha_1, \alpha_2), Cover(\alpha_3), P, \neg Q\}$ it will have 3 sons labelled respectively by $\{\alpha_1, \alpha_3\}$ $\{\alpha_2, \alpha_3\}, \{\alpha_1 \vee \alpha_2, \alpha_3\}$.

We say that a variable X is *regenerated* in a node n if the regeneration rule *(reg)* is applied to n.

Definition 4.5 *A* trace *on an infinite path* $\mathcal{P}$ *of a tableau* $\mathbf{T} = (T, L)$ *is a function* F *taking value on nodes* n *on an initial path of* $\mathcal{P}$ *such that* $F(n) \in L(n)$ *and whenever* F *is defined on* n *and* m *is the son of* n *in* $\mathcal{P}$ *then:*

(1) if $F(n)$ *is not reduced from* n *to* m *then* $F(m) = F(n)$*;*
(2) if $F(n)$ *is reduced from* n *to* m *then* $F(n)$ *is one of the result of this reduction where e.g:*
if the rule (or) *is applied to* n*,* $L(n) = \{\alpha \vee \beta\} \cup \Gamma$*,* $F(n) = \alpha \vee \beta$ *and* $L(m)$ *is* $\{\alpha\} \cup \Gamma$*, then* $F(m) = \alpha$*;*

if $L(n) = \{Cover(\mathcal{F}_1), \ldots, Cover(\mathcal{F}_n)\} \cup \Delta$ *where* Δ *contains only propositional constants or negated propositional constants,* $F(n) = Cover(\mathcal{F}_i)$*, and* $L(m)$ *is* $\{\alpha\} \cup \{\vee \mathcal{F}_j : j \neq i\}$ *for* $\alpha \in \mathcal{F}_i$*, then* $F(m) = \alpha$*;*
if $L(n) = \{Cover(\mathcal{F}_1), \ldots, Cover(\mathcal{F}_n)\} \cup \Delta$*, where* Δ *contains only propositional constants or negated propositional constants,* $F(n) = Cover(\mathcal{F}_h)$ *and* $L(m)$ *is* $\{\alpha\} \cup \{\vee \mathcal{F}_j : j \neq i\}$ *for an* $\alpha \in \mathcal{F}_i$ *and* $h \neq i$*, then* $F(m) = \vee \mathcal{F}_h$*.*

A trace is called a μ*-trace (*ν*-trace) if it is an infinite trace in which the least variable (with respect to the order of dependence* $\preceq_\gamma$*) which is regenerated infinitely often is a* μ*-variable (a* ν*-variable, respectively).*

It is then possible to prove that every infinite trace is either a μ- or a ν-trace, because along every trace there is always a least variable which is regenerated infinitely often.

Tableaux can be used to show that two formulas are equivalent: one can define a notion of tableau equivalence in such a way that if two tableaux are equivalent then the corresponding formulas are equivalent (see [7] for the definition).

Let us now summarize the main steps of the proof which allows to transform a μ-formula γ into an equivalent disjunctive formula $\hat{\gamma}$. What we will do in addition to the proof given in [7] is just to check that this transformation will not leave N_2 if the μ-formula is in N_2 (we leave the easier cases of levels N_0, N_1 to the reader). Let (T, L) be a tableau for the formula γ. The first step is to build a finite tree with "back edges" (T', L') (that is, a graph obtained from a finite tree by adding edges from some nodes to their ancestors), such that:

(1) (T', L') unwinds to (T, L);
(2) every node of (T', L') to which a back edge points (a "back node") is colored magenta or navy in such a way that for any infinite path from the unwinding of (T', L') we have: there exists a μ-trace on the path if and only if the highest node of (T', L') which appears infinitely often in the path is colored magenta;
(3) (only in the N_2-case) for no pair (m, n) of nodes in T' it holds: m is a back node colored magenta, n is a back node colored navy, and n lies on the path from m to a node k from which the back edge leading to m starts.

(T', L') can be constructed as follows: since γ is in N_2, a trace in T is a ν-trace if and only if it contains an infinite number of regenerations of ν-variables. Then one can build a deterministic Buchi automaton A on infinite words reading (the labels of the) infinite paths in T, and accepting only those paths having only ν-traces on them. The main idea for the construction of A is that the automaton must stay in a state of priority 1 until all traces have reached a new regeneration of a ν-variable, and when this happens, it will go to a state of 0-priority. Then it will start again, waiting until all traces have reached a new regeneration of a ν-variable and so on. A complete description of the automaton is given in the appendix. Suppose we have

such an automaton A, and let Ω be its parity function. If we run A on the infinite paths of the tableau, we may associate to each node n of the tableau a state $S(n)$ of the automaton in such a way that:

- if n_0 is the root of T, then $S(n_0)$ is the initial state of A;
- If m is a son of n then $(S(n), L(m), S(m))$ is a transition of the automata.

Then a path in T contains a μ-trace if and only if the least priority of the states appearing infinitely often on the path is odd. The set of nodes of (T', L') is then defined as the least subset of the set $\{(n, S(n)) : n \in T\}$ such that:

(1) $(n_0, S(n_0))$ belongs to T', where n_0 is the root of T;
(2) if $(n, S(n)) \in T'$, m is a son of n in T, and there exists a node $(m', S(m'))$ such that:
– $L(m') = L(m)$, $S(m') = S(m)$;
– m' is an ancestor of m in T and for all nodes n'' on the path between m' and m we have $\Omega(m') \leq \Omega(n'')$;
then we forget the node $(m, S(m))$ and build a back edge from $(n, S(n))$ to $(m', S(m'))$. If the preceding conditions are not fulfilled, then we add $(m, S(m))$ in T'.

We let $L'(m, S(m)) = L(m)$. We then color every back node $(m, S(m))$ magenta if $\Omega(S(m)) = 1$, or navy, if $\Omega(S(m)) = 0$, and doing so we see that the first two properties we required on (T', L') are fulfilled. As for the third property, suppose there exist a back node m colored magenta and a back node n colored navy which lies on the path from m to a node k from which the back edge leading to m starts. But this is impossible because, since there is a back edge from k to m, we should have $\Omega(S(m)) \leq \Omega(S(n))$, while $\Omega(S(m)) = 1$ and $\Omega(S(n)) = 0$. The second step in the proof is to construct the disjunctive formula $\hat{\gamma}$ from (T', L'): the construction starts from the leaves of the tree to the root; to all leaves from which a back edge starts leading to a node n we assign the variable X_n; this variable is then closed with a fixed point when we reach n, and the type of fixed point depends on the color of n: it will be a least fixed point if n is colored magenta, a greatest fixed point if n is colored navy. At the end of the construction we reach the root and the formula corresponding to the root will be the disjunctive formula $\hat{\gamma}$. Then one can prove that $\hat{\gamma}$ has a tableau which is equivalent to the tableau of γ, and hence γ is equivalent to $\hat{\gamma}$.

We will not enter in the details of this construction here, but we check that level N_2 of the syntactical hierarchy is preserved from γ to $\hat{\gamma}$. This is a consequence of the third property of (T', L'): $\hat{\gamma}$ will be in N_2 unless there exists a back node m colored magenta and a back node n colored navy which lies on the path from m to a node k from which the back edge leading to m starts, and we know that there are no such m, n. From the above discussion it follows:

Lemma 4.6 *A formula in N_k is equivalent to a disjunctive formula in N_k, for $k =$*

$0, 1, 2$.

4.2. *Closure under Bisimulation Quantifiers*

In [2] it is proved that the μ-calculus is closed under existential bisimulation quantifiers: if ϕ is a sentence of the μ-calculus, there exists a μ-sentence ψ which behaves like $\widetilde{\exists} P\phi$, that is:

$$M \models \psi \Leftrightarrow \ \exists N, \exists P \subseteq D^N \text{ with } N \sim_{Prop\setminus\{P\}} M \text{ and } (N, P) \models \phi.$$

The same result is known for modal logic, i.e. for level N_0 of the μ-calculus. Here we prove that the same holds for levels N_1 and N_2 of the μ-calculus hierarchy.

Theorem 4.7 *N_1 and N_2 are closed under existential bisimulation quantifiers on arbitrary models.*

PROOF. Fix $k \in \{1, 2\}$ and $\phi \in N_k$. By Lemma 4.6 we know that ϕ is equivalent to a disjunctive formula ψ in N_k. By Theorem 3.6 we know that $\widetilde{\exists} P\psi$ is equivalent to $\psi[P/\top, \neg P/\top]$ which is still a formula in N_k. □

Corollary 4.8 *The uniform interpolant of a μ-formula ϕ in N_1 or N_2 belongs to the same level as ϕ.*

4.3. *The Power of Two Alternations*

In the previous section we proved that N_1 and N_2 are closed under existential bisimulation quantifiers. Our next task is to show that this is not true after level N_2, because the whole μ-calculus is contained in the closure of level M_2. To prove this we shall use again the correspondence between the μ-calculus and nondeterministic automata introduced in Section 3. Since any μ-formula is equivalent to a disjunctive formula and disjunctive formulas correspond to nondeterministic automata we have:

Theorem 4.9 (In [7]) *For any μ-sentence ϕ there exists a nondeterministic automaton A such that an ω-expanded tree satisfies ϕ if and only if it is accepted by A. Conversely, any nondeterministic automaton is equivalent to a μ-sentence.*

We now prove that any μ-sentence can be obtained from a sentence in M_2 by using a certain number of existential bisimulation quantifiers. We shall do this in two steps: in Lemma 4.10 we prove the analogous result over the class of ω-expanded trees using monadic second order existential quantifiers instead of existential bisimulation quantifiers. Then in Corollary 4.13 we go from ω-expanded trees to arbitrary models by considering bisimulation quantifiers instead of monadic quantifiers.

Lemma 4.10 *For any μ-sentence $\phi(P_1, \ldots, P_m)$ there exists a μ-sentence $\theta(P_1, \ldots, P_m, Q_1, \ldots, Q_n)$ with $\theta \in M_2$ such that for any ω-expanded tree T it holds*

$$T \models \phi \leftrightarrow \exists Q_1 \ldots \exists Q_n \theta.$$

PROOF. By Theorem 4.9 for any μ-calculus formula ϕ there exists a nondeterministic automaton A which is equivalent to ϕ over ω-expanded trees. We now prove that the existence of an A-labeling over an ω-expanded tree can be expressed by using a finite number of monadic existential quantifiers $\exists Q_1 \ldots \exists Q_n$ over a formula θ in M_2.

By definition 3.3 a nondeterministic automaton accepts an ω-expanded tree T iff T has an A-labeling. This labeling defines subsets $Q_0, \ldots, Q_n$ of the tree T (where Q_i corresponds to the set of points labelled by the state q_i) having the following properties:

(1) the sets $Q_0, \ldots, Q_n$ form a partition of T and the root of the tree belongs to Q_0;
(2) If $l(s) = q$ then $\{q' \in Q : \exists t \in Succ(s), l(t) = q'\}$ belongs to $\delta(Q, T(s))$;
(3) if $s_0, s_1, \ldots$ is an infinite path starting from the root and for every j the index i_j is such that $s_j \in Q_{i_j}$, then the least number appearing infinitely often in the sequence

$$\Omega(q_{i_0}), \Omega(q_{i_1}), \ldots,$$

is even.

Conversely, the existence of subsets $Q_0, \ldots, Q_n$ satisfying the above properties clearly allows us to construct an A-labeling of T.

It follows that A accepts T iff $T \models \exists Q_1 \ldots \exists Q_n(\phi_1 \wedge \phi_2 \wedge \phi_3)$, where ϕ_1, ϕ_2, ϕ_3 are formulas expressing the above points. We now show that ϕ_1, ϕ_2, ϕ_3 can be chosen to be in N_1, N_1, M_2, respectively. This is obvious for ϕ_1, which is equivalent to

$$Q_0 \wedge \nu X(\bigwedge_{i \neq j} \neg(Q_i \wedge Q_j) \wedge \Box X)) \wedge \nu X(\bigvee_i Q_i) \wedge \Box X)).$$

As for ϕ_2, for any q_i, σ we consider the modal formula

$$f_{q_i,\sigma} = \bigvee_{\{Q_1, \ldots Q_n\} \in \delta(q_i,\sigma)} \Diamond(Q_1) \wedge \ldots \Diamond(Q_n) \wedge \Box(Q_1 \vee \ldots \vee Q_n).$$

We can then define ϕ_2 as the N_1-formula

$$\nu X((\bigwedge_{i,\sigma}(Q_i \wedge \hat{\sigma} \rightarrow f_{q_i,\sigma})) \wedge \Box X),$$

where

$$\hat{\sigma} = \bigwedge_{P \in \sigma} P \wedge \bigwedge_{P \notin \sigma} \neg P.$$

To express ϕ_3 with a formula in M_2 we proceed as follows. First of all, notice that the existence of an infinite chain starting from a node w in which P appears infinitely often and Q appears in any point can be described by a μ-formula $\psi(P, Q)$ of the second level N_2:

$$\psi(P, Q) := \nu X \mu Y (P \wedge Q \wedge \Diamond(X)) \vee (Q \wedge \Diamond(Y)).$$

Fix an index k and substitute Q_k for P and the conjunction of

$$\{\neg Q_i : \Omega(q_i) < \Omega(q_k)\}$$

for Q in the formula $\psi(P, Q)$; we obtain a formula $\psi_k \in N_2$ which is true in w iff from w starts a chain in which infinitely many points are in Q_k, and no point is in any of the Q_i, for $\Omega(q_i) < \Omega(q_k)$. Finally, notice that point (3) above is expressed by the M_2-formula

$$\bigwedge_{\Omega(q_k) odd} \nu X (\neg \psi_k \wedge \Box X).$$

This proves that any μ-formula ϕ is equivalent over ω-expanded trees to a formula of type $\exists Q_1, \ldots, \exists Q_n \theta$, with $\theta \in M_2$. □

To prove our next step, we show in Lemma 4.12 that bisimulation quantifiers applied to bisimulation invariant formulas behave like MSO-quantifiers on the class of ω-expanded trees. But first we remark:

Lemma 4.11 *If the trees T, T' are ω-expanded and bisimilar, then they satisfy the same MSO-sentences.*

PROOF. This holds because on ω-expanded trees any MSO-sentence is equivalent to a μ-calculus sentence [8], and bisimilar structures satisfy the same μ-sentences. □

Lemma 4.12 *If T is an ω-expanded tree and ϕ is a μ-sentence, then*

$$T \models \exists P \phi \leftrightarrow \widetilde{\exists} P \phi.$$

PROOF. The implication from left to right is trivial and does not require that T is ω-expanded. Conversely, suppose $T \models \widetilde{\exists} P \phi$, i.e. that there exists N with

$N \sim_{Prop\setminus\{P\}} T$ and $P \subseteq N$ such that $(N, P) \models \phi$. If N^ω is the ω-expansion of N we still have: there exists a $P \subseteq N^\omega$ with $(N^\omega, P) \models \phi$, that is: $N^\omega \models \exists P\phi$. But $\exists P\phi$ can be expressed as a second order property (in the language $Prop \setminus \{P\}$) of the structure N^ω, which is bisimilar to T w.r.t. this language. Hence, from Lemma 4.11 it holds: $T \models \exists P\phi$. □

Corollary 4.13 *For any μ-sentence $\phi(P_1, \ldots, P_m)$ there exists a μ-sentence $\theta(P_1, \ldots, P_m, Q_1, \ldots, Q_n)$ with $\theta \in M_2$ such that*

$$\models \phi \leftrightarrow \widetilde{\exists} Q_1, \ldots, \widetilde{\exists} Q_n \theta.$$

PROOF. Lemma 4.10 implies there exists $\theta \in M_2$ such that ϕ and $\exists Q_1, \ldots, \exists Q_n\theta$ are equivalent over ω-expanded models. Then Lemma 4.12 allows us to conclude that ϕ is equivalent to $\widetilde{\exists} Q_1, \ldots, \widetilde{\exists} Q_n\theta$ over arbitrary models. □

5 A Complete System for the μ-Calculus with Explicit Uniform Interpolants

Although by Corollary 4.13 there cannot be a simple rule for uniform interpolation of arbitrary formulas, we can still try to understand better how the existential bisimulation quantifier behaves w.r.t. the connectives and the operators of the μ-calculus.

To do so, we extend the original μ-language with the quantifier $\widetilde{\exists} P$ with the appropriate semantics (see Definition 2.6) and provide this extended logic $\widetilde{\mu}$ with a complete calculus. We shall see that to derive all validities in $\widetilde{\mu}$ we only need some standard principles allowing introduction and elimination of the bisimulation quantifier, plus some natural principles of commutativity between the existential bisimulation quantifier and the operators of the μ-calculus.

By inspecting the semantics of this quantifier we recognize easily that it enjoys at least the standard properties regarding substitutions and free variables. As usual, we say that the substitution of ψ for P in ϕ is *admissible* if no free variable of ψ becomes bound after the substitution for P in ϕ. If this is the case, we denote by $\phi[P/\psi]$ the formula obtained after the substitution.

The axiom and the rule for the existential bisimulation quantifiers are:

Ax1: $\phi[P/\psi] \rightarrow \widetilde{\exists} P\phi$ is provable, provided the substitution of ψ for P in ϕ is admissible;
R1: if $\phi \rightarrow \psi$ is provable, then $\widetilde{\exists} P\phi \rightarrow \psi$ is provable, provided P is not free in ψ.

The proof of the soundness of the above axiom and rule is left to the reader.

One could think that adding **Ax1** and **R1** to a Hilbert system which is complete for the μ-calculus (such as the Kozen system, proved to be complete in [13]) would give us the complete calculus for the extended logic, but this is not the case. Consider for example a valid principle as

$$\phi = \Diamond\top \rightarrow \widetilde{\exists}P(\Diamond(P) \wedge \Diamond(\neg P)).$$

It is easy to see that the system μ **+ Ax1 + R1** cannot prove ϕ: this is because all axioms and rules of μ**+ Ax1 + R1** are valid when we interpret the bisimulation quantifier $\widetilde{\exists}$ as a standard second order quantifier, while ϕ is not valid under this interpretation. Hence we need to add some more principles to μ**+ Ax1 + R1** in order to obtain a complete system. In this section we show that it is enough to add some simple *commutativity axioms*, relating the bisimulation quantifier $\widetilde{\exists}$ to disjunction, cover operators, and fixpoint operators.

First of all we prove that the existential bisimulation quantifier $\widetilde{\exists}P$ commutes with disjunctions and special conjunctions. In the next lemma, we denote $\sigma[P/\top, \neg P/\top]$ the formula obtained from a conjunction σ of literals by replacing every occurrence of P or $\neg P$ (if any) with $\top$.

Lemma 5.1 *If σ is a conjunction of a set of literals not containing both P and $\neg P$, and $\phi_1, \ldots, \phi_n$ are μ-formulas, then the following are valid formulas.*

$$\widetilde{\exists}P\sigma \leftrightarrow \sigma[P/\top, \neg P/\top], \qquad \widetilde{\exists}P(\phi_1 \vee \phi_2) \leftrightarrow \widetilde{\exists}P\phi_1 \vee \widetilde{\exists}P\phi_2$$

$$\widetilde{\exists}P(\sigma \wedge Cover(\phi_1, \ldots, \phi_n)) \leftrightarrow \widetilde{\exists}P\sigma \wedge Cover(\widetilde{\exists}P\phi_1, \ldots, \widetilde{\exists}P\phi_n),$$

PROOF. To prove the validity of $\sigma[P/\top, \neg P/\top] \rightarrow \widetilde{\exists}P\sigma$, suppose a model M satisfies $\sigma[P/\top, \neg P/\top]$; then the model M' which is like M except that P is interpreted as $\{r^M\}$, if P belongs to the conjunction σ, and as $\emptyset$ otherwise, is bisimilar to M if we do not consider P in the language, and satisfies σ.

On the other hand, if a model M satisfies $\widetilde{\exists}P\sigma$, then σ is true in a model M' which is bisimilar to M w.r.t. the language of M minus P; then any propositional constant which is different from P and is true in M' must be also true in M and M satisfies $\sigma[P/\top, \neg P/\top]$.

The verification of commutativity of $\widetilde{\exists}$ with disjunction is left to the reader. To prove that

$$\widetilde{\exists}P\sigma \wedge Cover(\widetilde{\exists}P\phi_1, \ldots, \widetilde{\exists}P\phi_n) \rightarrow \widetilde{\exists}P\sigma \wedge Cover(\phi_1, \ldots, \phi_n),$$

suppose $\widetilde{\exists}P\sigma \wedge Cover(\widetilde{\exists}P\phi_1, \ldots, \widetilde{\exists}P\phi_n)$ holds in a model (M, V), where V is a valuation of the free variables in $\sigma, \phi_1, \ldots, \phi_n$. Fix a successor v of the root r^M of M and consider all formulas of type $\widetilde{\exists}P\phi_i$ it satisfies: for any such formula there exists a model $N_{v,i}$ and a valuation $V_{v,i}$ of the free variables of ϕ_i such that

$(N_{v,i}, V_{v,i})$ is $Prop \setminus \{P\}$-bisimilar to (M, V) and $(N_{v,i}, V_{v,i}) \models \phi_i$. Consider a new model M' with a new root satisfying the same propositional constants as r^M and connected to all these $N_{v,i}$, when v varies in the successors of r^M. Define a valuation V' over a variable X as

$$V'(X) = \bigcup_{v,i} V_{v,i}(X), \text{ if } r^M \notin V(X),$$

and

$$V'(X) = \{r^M\} \cup \bigcup_{v,i} V_{v,i}(X), \text{ otherwise.}$$

Then (M', V') is $Prop \setminus \{P\}$-bisimilar to (M, V) and verifies the formula $\sigma \wedge Cover(\phi_1, \ldots, \phi_n)$.

The verification of the validity of the reverse arrow is left to the reader. □

As a first step towards a complete calculus for $\widetilde{\mu}$, let us prove that the principles discovered so far are complete if we do not consider fixpoint operators, that is, if we only consider modal logic K extended with the existential bisimulation quantifier. Let us denote the extended logic by $\widetilde{K}$.

Theorem 5.2 *Consider the Hilbert calculus for $\widetilde{K}$ consisting of the following axioms and rules:*

(1) a complete Hilbert system K of axioms and rules for modal logic;
(2) the axiom **Ax1** *and the rule* **R1***;*
(3) if σ is a non contradictory conjunction of literals and $\phi_1, \ldots, \phi_n$ are formulas, the axiom

$$\widetilde{\exists} P(\sigma \wedge Cover(\phi_1, \ldots, \phi_n)) \leftrightarrow \sigma[P/\top, \neg P/\top] \wedge Cover(\widetilde{\exists} P\phi_1, \ldots, \widetilde{\exists} P\phi_n).$$

Then this calculus is sound and complete for modal logic extended with the existential bisimulation quantifier.

PROOF. To prove that $\widetilde{K}$ is complete it is enough to show that for any formula ψ of the logic there exists a modal formula ψ^- such that $\psi \leftrightarrow \psi^-$ is provable in $\widetilde{K}$: if this is true, to derive a valid formula ψ in $\widetilde{K}$ we can derive the (provably equivalent and) valid modal formula ψ^- instead; but $\widetilde{K}$ proves ψ^- because K proves it (being a complete calculus for modal formulas) and $\widetilde{K}$ is an extension of K.

We find the formula ψ^- by induction on the structural complexity of ψ, the only interesting case being $\psi = \widetilde{\exists} P\phi$. By induction, we suppose that ϕ is provably equivalent to a modal formula ϕ^-. Now, any modal formula is semantically equivalent to a disjunctive formula ϕ_d^-, and $\widetilde{K}$ can prove this equivalence since our system

contains the complete system K. By the existential rule and axiom we have

$$\widetilde{\exists} P\phi \leftrightarrow \widetilde{\exists} P\phi_d^-,$$

and we only have to prove that $\widetilde{\exists} P\phi_d^-$ is provably equivalent to a modal formula. By Theorem 3.6, we know that this formula is the modal formula $\phi_d^-[P/\top, \neg P/\top]$. Moreover, if fixpoint operators are not present the (semantical) equivalence between $\widetilde{\exists} P\phi_d^-$ and $\phi_d^-[P/\top, \neg P/\top]$ can be easily proved inside $\widetilde{K}$ by using induction on the structural complexity of the disjunctive formula ϕ_d^-, the cover axioms, **Ax1**, and **R1**.

So we let $\psi^- = \phi_d^-[P/\top, \neg P/\top]$. □

We now go back to the μ-calculus. The strategy to find a complete calculus for this logic is the same as for $\widetilde{K}$, that is: we use the explicit form (in the original μ-language) of uniform interpolants of disjunctive μ-formulas (see Theorem 3.6). First we prove that the existential bisimulation quantifier $\widetilde{\exists} P$ commutes with the fixpoint operators $\mu X, \nu X$, provided the *contexts* $\mu X.\phi$, $\nu X.\phi$ are disjunctive.

Corollary 5.3 *If $\mu X.\phi$ and $\nu X.\phi$ are disjunctive formulas then*

$$\models \widetilde{\exists} P\mu X.\phi \leftrightarrow \mu X.\widetilde{\exists} P\phi \qquad\qquad \models \widetilde{\exists} P\nu X.\phi \leftrightarrow \nu X.\widetilde{\exists} P\phi.$$

PROOF. By Theorem 3.6, the formula $\widetilde{\exists} P\mu X.\phi$ is equivalent to $(\mu X.\phi)[P/\top, \neg P/\top]$, which is the same as $\mu X.(\phi[P/\top, \neg P/\top])$, which is equivalent to $\mu X.\widetilde{\exists} P\phi$. The proof for the operator ν is similar. □

Notice that these equivalences are not true without the disjunctivity hypothesis. Consider for example the formula $\phi = P \wedge \Diamond(\neg P) \wedge \Box X$. We have $\widetilde{\exists} P\phi = \Diamond\top \wedge \Box X$, hence $\nu X.\widetilde{\exists} P\phi = \nu X.\Diamond\top \wedge \Box X$, which is true in a model M iff all nodes accessible from the root satisfy $\Diamond\top$. Thus, the formula $\nu X.\widetilde{\exists} P\phi$ is satisfiable. On the other hand, the formula $\nu X.\phi$ is equivalent to $\bot$, and so is $\widetilde{\exists} P\nu X.\phi$.

In the next theorem we show that adding commutativity between $\widetilde{\exists}$ and fixpoint operators in a disjunctive context to the principles presented in Theorem 5.2 is enough to obtain a complete system for $\widetilde{\mu}$.

Theorem 5.4 *Consider the Hilbert calculus $\widetilde{\mu}$ consisting of the following axioms and rules:*

(1) axioms and rules of the system $\widetilde{K}$ (see Theorem 5.2);
(2) a complete system of Hilbert axioms and rules for the μ-calculus (e.g. the Kozen system, proved to be a complete system in [13]);
(3) if $\mu X.\phi$ is disjunctive, the axiom $\widetilde{\exists} P\mu x.\phi \leftrightarrow \mu x.\widetilde{\exists} P\phi$;

(4) if $\nu X.\phi$ is disjunctive, the axiom $\widetilde{\exists} P \nu x.\phi \leftrightarrow \nu x.\widetilde{\exists} P\phi$.

Then $\widetilde{\mu}$ is sound and complete for the μ-calculus extended with the existential bisimulation quantifier, that is: a formula ϕ of this logic is valid if and only if it is derivable within the system $\widetilde{\mu}$.

PROOF. To prove that $\widetilde{\mu}$ is complete, it is enough to show that for any formula ψ of the logic there exists a μ-formula ψ^- such that $\psi \leftrightarrow \psi^-$ is provable in $\widetilde{\mu}$. This can be achieved by induction on the structural complexity of the formula, the only interesting case being $\psi = \widetilde{\exists} P\phi$. By induction, we suppose that ϕ is provably equivalent to a μ-formula ϕ^-, which is in turn provably equivalent to a disjunctive μ-formula ϕ_d^- since our system contains the complete system μ. By the existential rule and axiom we have

$$\widetilde{\exists} P\phi \leftrightarrow \widetilde{\exists} P\phi_d^- .$$

By induction on the structure of the disjunctive formula ϕ_d^- we prove using the axioms and the rules above that $\widetilde{\exists} P\phi_d^- \leftrightarrow \phi_d^-[P/\top, \neg P/\top]$. Hence $\widetilde{\exists} P\phi$ is provably equivalent to the μ-formula $\phi_d^-[P/\top, \neg P/\top]$.

So we let $\psi^- = \phi_d^-[P/\top, \neg P/\top]$. □

6 Conclusions and Related Work

In this paper we gave a simple rule for the uniform interpolant of a disjunctive formula, and studied the behaviour of the existential bisimulation quantifiers w.r.t. the alternation-depth hierarchy of the μ-calculus. We also gave an axiomatixation of the μ-calculus extended with the existential bisimulation quantifier, allowing in this way the possibility of computing with a logic as powerful as the μ-calculus in which we have a way to denote uniform interpolants explicitly. A related question is the axiomatization of BQL [5] which is defined as the bisimulation quantifier closure of Propositional Dynamic Logic PDL. The logic BQL is semantically equivalent to the μ-calculus, but has the advantage of replacing all fixpoint operators (which are difficult to read, especially when nesting of two or more operators occur) by the Kleene star and the existential bisimulation quantifier. The structure of formulas expressing a certain property are closer to natural language in BQL than in the μ-calculus (see Example 2 in Subsection 2.2). A complete system of axioms and rules for BQL is presented in [3].

References

[1] J. Bradfield, The modal μ-calculus alternation hierarchy is strict, *Theoret. Comput. Sci.* **195** (1998), 133–153.

[2] G. D'Agostino and M. Hollenberg, Logical questions concerning the μ-calculus: interpolation, Lyndon and Los-Tarski, *J. Symbolic Logic* **65** (2000), 310–332.

[3] G. D'Agostino and G. Lenzi, An axiomatization of bisimulation quantifiers via the mu-calculus, Accepted for publication in *Theoretical Computer Science* .

[4] S. Ghilardi and M. Zawadowski, A sheaf representation and duality for finitely presented Heyting algebras, *J. Symbolic Logic* **60** (1995), 911-939.

[5] M. Hollenberg, *Logic and Bisimulation*, PhD Thesis, University of Utrecht, Vol XXIV, Zeno Institute of Philosophy, 1998.

[6] D. Janin , *Propriété logique du non determinisme et μ-calcul modal*, Thèse du doctorat, Université Bordeaux I

[7] D. Janin, and I. Walukiewicz, Automata for the modal μ-calculus and related results, in *Proceedings of the conference MFCS'95*, Lecture Notes in Computer Science **969**, Ed. Springer, 1995, 552–562.

[8] D. Janin, and I. Walukiewicz, On the expressive completeness of the propositional μ-calculus w.r.t. monadic second-order logic, in *Proceedings of the conference CONCUR '96*, Lecture Notes in Computer Science **1119**, Ed. Springer, 1996, 263–277.

[9] M. Jurdziński, Small progress measures for solving parity games, in *Proceedings of the conference STACS 2000*, Lecture Notes in Computer Science **1770**, Ed. Springer, 2000, 290–301.

[10] D. Kozen, Results on the propositional μ-calculus, *Theoret. Comput. Sci.* **27** (1983), 333–354.

[11] O. Kupferman, M. Vardi and P. Wolper, An automata-theoretic approach to branching time model checking, *Journal of the ACM*, **47**, no. 2, (2000), 312-360.

[12] A. Visser, *Bisimulations, model descriptions and propositional quantifiers*, Logic Group Preprint Series 161, Department of Philosophy, Utrecht University, 1996.

[13] I. Walukiewicz, Completeness of Kozen's axiomatization of the propositional μ-calculus, *Inform. and Comput.* **157** (2000), 142–182.

A APPENDIX

Given a μ-formula γ with a tableau T, we construct a deterministic Buchi automaton A on infinite words which recognises exactly the infinite paths in the tableau having only ν-traces on them. We suppose that the tableau's nodes (except the root) are labelled by a pair in which the first component gives the formula which has been reduced in the father of the node, and the second component gives the label of the node in the tableau. E.g. if we are in node n which was created because of the rule and applied to the father m to the formula $\alpha = \beta \wedge \gamma$ then the

label of n will be $(\alpha, \{\beta, \gamma, ...\})$ The automaton A is defined as follows. The set of states is given by all sets of the form $\{(\alpha_1, u_1), ..., (\alpha_n, u_n)\}$ where u_i is either s (for "search") or f (for found) and $\{\alpha_1, ..., \alpha_n\}$ is the second component of a label of the tableau, where all propositional constants have been removed. This implies in particular that all α_j are different. When defining the transitions of the automaton we identify pairs $(\alpha, s), (\alpha, f)$ with the single element (α, s).The initial position is $\{(\gamma, s)\}$. The states having all element with second component equal to f are of priority 0, while the other states have priority 1.The automaton reads paths in T (but it skips the label of the root). We will define A in such a way that when A reads a path $(L(root))L(n_1).....L(n_i)...$ of the tableau, it will be in a state $\{(\alpha_1, u_1), ..., (\alpha_n, u_n)\}$ just after reading a label $\{\alpha_1, ..., \alpha_n\}$. If A is in state $\{(\alpha_1, u_1), ..., (\alpha_n, u_n)\}$, where not all u_i are equal to f, it reads a label (θ, L), and there exists i such that $\theta = \alpha_i$, then:

(1) if $\alpha_i = \beta \wedge \delta$ and $L = (\{\alpha_1, ..., \alpha_n\} \setminus \{\alpha_i\}) \cup \{\beta, \delta\}$ then A goes to the state

$$\{(\alpha_1, u_1), ..., (\alpha_n, u_n)\} \setminus \{(\alpha_i, u_i)\}) \cup \{(\beta, u_i), (\delta, u_i)\};$$

(2) if $\alpha_i = \beta \vee \delta$ and $L = (\{\alpha_1, ..., \alpha_n\} \setminus \{\alpha_i\}) \cup \{\epsilon\}$,(where ϵ is either β or δ), then A goes to the state

$$\{(\alpha_1, u_1), ..., (\alpha_n, u_n)\} \setminus \{(\alpha_i, u_i)\}) \cup \{(\epsilon, u_i\};$$

(3) if $\alpha_i = \sigma X \alpha$ where σ is ν or μ and $L = (\{\alpha_1, ..., \alpha_n\} \setminus \{\alpha_i\}) \cup \{\alpha\}$, then A goes to the state

$$\{(\alpha_1, u_1), ..., (\alpha_n, u_n)\} \setminus \{(\alpha_i, u_i)\}) \cup \{(\alpha, u_i)\}$$

(4) if $\alpha_i = X$, the binding of X is α, and $L = (\{\alpha_1, ..., \alpha_n\} \setminus \{X\}) \cup \{\alpha\}$, then A goes to the state

$$\{(\alpha_1, u_1), ..., (\alpha_n, u_n)\} \setminus \{(\alpha_i, u_i)\}) \cup \{(\alpha, s)\},$$

if X is a ν-variable, and to the state

$$\{(\alpha_1, u_1), ..., (\alpha_n, u_n)\} \setminus \{(\alpha_i, u_i)\}) \cup \{(\alpha, u_i)\},$$

if X is a μ-variable;

(5) if $\alpha_1 = Cover(\mathcal{F}_1), ..., \alpha_n = Cover(\mathcal{F}_n)$ and there exists a formula $\alpha \in \mathcal{F}_i$ such that $L = \{\alpha\} \cup \{\vee \mathcal{F}_i : j \neq i\}$ then A goes to the state

$$\{(\alpha, u_i)\} \cup \{(\vee \mathcal{F}_j, u_j) : j \neq i\}.$$

If A is in state $\{(\alpha_1, u_1), ..., (\alpha_n, u_n)\}$, where all u_i are equal to f, we just consider the previous transitions, but as if we where starting from the state $\{(\alpha_1, s), ..., (\alpha_n, s)\}$ instead of $\{(\alpha_1, u_1), ..., (\alpha_n, u_n)\}$. If none of the above condition is fulfilled, the automaton stops.

Model Checking Hybrid Logics (With an Application to Semistructured Data)

Massimo Franceschet [a,b,1] Maarten de Rijke [b,2]

[a] *Department of Sciences, University of Chieti-Pescara, Italy*

[b] *Informatics Institute, University of Amsterdam, The Netherlands*

Abstract

We investigate the complexity of the model checking problem for hybrid logics. We provide model checker algorithms for various hybrid fragments and we prove PSPACE-completeness for hybrid fragments including binders. We complement and motivate our complexity results with an application of model checking in hybrid logic to the problems of query and constraint evaluation for semistructured data.

Key words: Model Checking, Modal Logics, Hybrid Logics, Semistructured data

1 Introduction

In *model checking* [19] we are given a formal model and a property and we have to check whether the model satisfies the property. The model is a labelled graph, sometimes called Kripke structure, and the property is a formula in some logical language. We search the graph in order to *check* whether the formula is true in the *model*. As a technique, model checking has very strong links to (at least) two areas in computer science: verification and databases. In the first half of this paper, we focus on model checking algorithms for so-called hybrid logics; in the second half, we go on to show their relevance for reasoning about semistructured data.

Email addresses: francesc@science.uva.nl (Massimo Franceschet), mdr@science.uva.nl (Maarten de Rijke).

[1] Supported by a grant from the Netherlands Organization for Scientific Research (NWO) under project number 612.000.207.

[2] Supported by grants from the Netherlands Organization for Scientific Research (NWO) under project numbers 365-20-005, 612.069.006, 612.000.106, 220-80-001, 612.000.207, and 612.066.302.

Modal and temporal logics have been successfully used as specification languages in the model checking task [20]; they are algorithmically well-behaved and mathematically natural fragments of classical logics. However, something crucial is missing in propositional modal and temporal logics: they lack mechanisms for naming states, for accessing states by names, and for dynamically creating new names for states. In particular, traditional modal and temporal logics are able to express properties that satisfy the *tree model property*, that is, properties that are satisfiable if, and only if, they are satisfiable in a tree-like model. Are there extensions of modal and temporal logics violating the tree model property that are still computationally tractable? This is where *hybrid logics* come in. They allow us to refer to states in a truly modal framework, mixing features from first-order logic and modal logic, whence the name *hybrid* logic [15]. In addition to ordinary propositional variables, hybrid languages provide a type of atomic formulas called *nominals*. Syntactically, nominals behave like propositional variables, but they have an important semantic property: nominals are true at exactly one state in any model.

Nominals are only the first ingredient that sets hybrid languages apart from traditional modal-like languages. Hybrid languages may also contain the *at operator* $@_i$ which gives direct access to the unique state named by i: $@_i p$ holds if, and only if, p holds at the state named by i. Moreover, hybrid languages may be extended with the *downarrow binder* $\downarrow x$ that assigns the variable name x to the current state of evaluation. The operator $@$ combines naturally with $\downarrow$: $\downarrow$ stores the current state of evaluation and $@$ enables us to *retrieve* the information stored by shifting the point of evaluation in the model. While $\downarrow x$ stores the current state in x, the binder $\Downarrow x$ stores the 'label' of the current state in x, that is, the set of propositions holding at the current state. Finally, the *existential binder* $\exists x$ binds the variable name x to some state in the model.

Model checking for hybrid languages has hardly been explored so far. In this paper we address this gap. Our approach is incremental: on top of well-known model checking results for Propositional Temporal Logic and Converse Propositional Dynamic Logic, we investigate the model checking problem for these languages extended with the hybrid machinery as well as with the universal modality $\mathbf{A}$. It turns out that the addition of nominals, the @ operator, and the universal modality $\mathbf{A}$ does not increase the complexity of the model checker. In contrast, an arbitrary use of hybrid binders in formulas is computationally dangerous. The model checking problem for any hybrid logic that freely mixes the hybrid binder $\downarrow$ or $\Downarrow$ with temporal operators is PSPACE-complete, while $\exists$ is hard even without temporal operators. However, the model checker runs in exponential time with respect to the *nesting degree* of the binders in the formula. This means that we can still check in polynomial time long formulas, as long as the nesting degree of the hybrid binders on the formula is bound. We summarize our complexity results in Table 1, where k is the length of the formula, n and m are the number of nodes and the number of edges of the graph structure, respectively, and r is the nesting degree of hybrid binders. Moreover, $\mathbf{F}$ is the Future temporal operator, $\mathbf{P}$ is Past, $\mathbf{U}$ is Until and $\mathbf{S}$

Language	Complexity	Language	Complexity
$\mathrm{HL}(@, \mathbf{F}, \mathbf{P}, \mathbf{A})$	$\mathbf{O}(k \cdot (n+m))$	$\mathrm{HL}_r(\downarrow, @, \mathbf{F}, \mathbf{A})$	$\mathbf{O}(k \cdot (n+m) \cdot n^r)$
$\mathrm{HL}(@, \mathbf{U}, \mathbf{S}, \mathbf{A})$	$\mathbf{O}(k \cdot n \cdot m)$	$\mathrm{HDL}_r(\downarrow, @, \mathbf{A})$	$\mathbf{O}(k \cdot (n+m) \cdot n^r)$
$\mathrm{HDL}(@, \mathbf{A})$	$\mathbf{O}(k \cdot (n+m))$	$\mathrm{HL}(\exists)$	PSPACE-complete
$\mathrm{HL}(\downarrow, @)$	$\mathbf{O}(k \cdot n)$	$\mathrm{HL}(\exists, @, \mathbf{F}, \mathbf{A})$	PSPACE-complete
$\mathrm{HL}(\downarrow, \mathbf{F})$	PSPACE-complete	$\mathrm{HL}_r(\exists, @, \mathbf{F}, \mathbf{A})$	$\mathbf{O}(k \cdot (n+m) \cdot n^{r+1})$
$\mathrm{HL}(\downarrow, \mathbf{A})$	PSPACE-complete	$\mathrm{HDL}_r(\exists, @, \mathbf{A})$	$\mathbf{O}(k \cdot (n+m) \cdot n^{r+1})$
$\mathrm{HL}(\downarrow, @, \mathbf{F}, \mathbf{A})$	PSPACE-complete		

Table 1
Complexity of model checking for hybrid logics.

is Since. Finally, languages of the form $\mathrm{HL}(\cdot)$ are hybrid extensions of Propositional Temporal Logic, while languages of the form $\mathrm{HDL}(\cdot)$ are hybrid extensions of Converse Propositional Dynamic Logic. Notice that PSPACE-complete problems are hard with respect to expression (or formula, or query) complexity, which is the complexity of model checking if we only consider the length of the formula as a parameter. If data complexity (the complexity of model checking if we only consider the size of the model as a parameter) is taken into account, all the model checking problems summarized in the table can be solved in polynomial time.

In the second part of the paper we illustrate a general methodological point: since hybrid languages provide very natural modeling facilities, understanding the computational and algorithmic properties of hybrid languages is of great potential value. The complexity-theoretic results obtained in this paper are valuable results about hybrid logic in their own right, but we believe they get additional value because of the fact that hybrid languages provide such natural modeling facilities, which makes the formal results of this paper applicable in a fairly direct way. To back up these claims we apply model checking for hybrid logics to the problems of query and constraint evaluation for *semistructured data*: data with some structure but without a regular schema. We discuss a hierarchy of query languages for semistructured data corresponding to fragments of the language Lorel [3], the query language in the Lore system [32], which was designed for managing semistructured data. The languages that we discuss offer regular expressions to navigate the query graph at arbitrary depths, as well as the possibility of comparing object identities and object values. We embed those query languages into fragments of hybrid logics with different expressivity and establish a close relationship between the query processing problem for semistructured data and the global model checking problem for hybrid logics. Moreover, we describe languages to specify path constraints for semistructured data, including inclusion, inverse and functional path constraints. Path constraints generalize relational integrity constraints for semistructured databases and they are useful to provide a loose schema to the otherwise unstructured database. Once again, we provide an embedding into hybrid logic, this time of the constraint

language, and we underline the close connection between path constraint evaluation for semistructured data and model checking for hybrid logics.

The paper is organized as follows. In Section 2 we discuss related work. Section 3 introduces hybrid logic. In Section 4 we provide model checkers for different hybrid languages, analyze their computational complexity, and prove PSPACE-completeness of the model checking problem for hybrid fragments allowing binders. Readers mostly interested in the relation between hybrid logic and semistructured data, can skip Section 4 and return to it for details of the results used in Section 5, where we describe the application of model checking hybrid logic to semistructured data. We conclude the paper and outline future work in Section 6.

2 Related Work

Hybrid logic was invented by Arthur Prior, the inventor of tense logic. The germs of the idea seem to have emerged in the 1950s, but the first detailed account is [36]. Prior called nominals world propositions and worked with rich hybrid languages including quantifiers $\forall$ and $\exists$. The next big step was taken by Robert Bull, Prior's student, in [16]. Bull introduced a three-sorted hybrid language (propositional variables, state nominals and *path nominals*) and proved a completeness result for this logic. Path nominals name branches in tree-like models of time by being true at all and only the points of the branch. There were no further papers on the subject till the 1980s, when hybrid logic was reinvented by a group of Bulgarian logicians (Passy, Tinchev, Gargov, and Goranko). The locus classicus of this work is Passy and Tinchev's [35]; they initiated the study of binder-free systems. During the 1990s, the emphasis has been on understanding the hybrid hierarchy in more detail. Goranko introduced the $\downarrow$ binder [26], Blackburn and Seligman examined the interrelationships between a number of binders [14]. Characterizations with respect to first-order correspondence theory, interpolation properties, and computational complexity (of the satisfiability problem) for hybrid modal logics have been studied in [11]; recent contributions completing the picture are in [38]. The complexity of the satisfiability problem for hybrid temporal logics with respect to different classes of frames has been investigated in [9,10,25]. As for implementations, a resolution-based theorem prover for hybrid logic with $@$ and $\downarrow$ has been implemented [12]. For a comprehensive entry point to the field see the hybrid logic home page [30].

Since the mid-1990s there has been a lot of work on the interface of computational logic and semistructured data, making use of a wide variety of logical tools and techniques. E.g., [17] use simulations and morphisms, [4] concentrate on regular expressions, and [18] make the connection with description logic. The relation between model checking and query processing has been extensively explored for *structured* data; see, e.g., [28]. The relation between model checking and query

processing for *semistructured* data goes back at least to [6], where it was formulated in terms of suitable modal-like logics. Quintarelli [37] embeds a fragment of the graphical query language G-Log into CTL, and she sketches a mapping for subsets of other semistructured query languages, like Lorel, GraphLog and UnQL. It is worth noticing that the fragments considered in [37] do not allow queries with joins. De Alfaro [8] proposes the use of model checking for detecting errors in the structure and connectivity of web pages. Miklau and Suciu [33] and Gottlob et al. [27] sketch an embedding of the forward looking fragment of XPath into CTL. Finally, Marx [31] used PDL-like logics in order to extend the XPath core language to a language that is expressively complete with respect to first-order logic on finite trees.

As for the relation between model checking and path constraints evaluation for semistructured data, Alechina et al. [7] embed forward and backward path constraints into Converse Propositional Dynamic Logic. Calvanese et al. [18] use description logics, and Afanasiev et al. [5] turn to CTL and provide experimental results of the "query evaluation as model checking" perspective.

3 Hybrid Logics

Temporal logics [23] (TL, for short) may be viewed as fragments of classical logics [13]. They extend propositional logic by adding the well-known temporal operators future $\mathbf{F}$, past $\mathbf{P}$, until $\mathbf{U}$ and since $\mathbf{S}$. Let $\mathsf{PROP} = \{p, q, \ldots\}$ be a set of propositional variables. The syntax of temporal logic is as follows:

$$\phi := \top \mid p \mid \neg\phi \mid \phi \wedge \phi \mid \mathbf{F}\phi \mid \mathbf{P}\phi \mid \phi\mathbf{U}\phi \mid \phi\mathbf{S}\phi.$$

We adopt the usual Boolean shorthands. The dual of $\mathbf{P}$ is $\mathbf{H}\alpha = \neg\mathbf{P}\neg\alpha$, and the dual of $\mathbf{F}$ is $\mathbf{G}\alpha = \neg\mathbf{F}\neg\alpha$.

Temporal logic is interpreted over *Kripke structures* of the form $\langle M, R, V\rangle$, where M is a set of states (or worlds, points, nodes), R is a binary relation on M called the accessibility (or reachability) relation, and V is a valuation function from PROP to the powerset of M. We assume no specific structure of time (linear, branching, ...). Let $\mathcal{M}$ be a Kripke structure and $m \in M$. The semantics of temporal logic is given in Figure 1

Sometimes, TL also includes the transitive closure operators $\mathbf{F}^+$, $\mathbf{P}^+$, $\mathbf{U}^+$ and $\mathbf{S}^+$, interpreted over the transitive closure R^+ of the accessibility relation, as well as the universal modality $\mathbf{A}$. The semantics of the universal modality is as follows: $\mathcal{M}, m \models \mathbf{A}\phi$ iff for all m it holds that $\mathcal{M}, m \models \phi$. The dual of the universal modality is the existential modality $\mathbf{E}\alpha = \neg\mathbf{A}\neg\alpha$.

Hybrid logic (HL, for short) extends temporal logic with devices for naming states

$$
\begin{aligned}
\mathcal{M}, m &\models \top \\
\mathcal{M}, m &\models p \quad \text{iff} \quad m \in V(p), p \in \mathsf{PROP} \\
\mathcal{M}, m &\models \neg\phi \quad \text{iff} \quad \mathcal{M}, m \not\models \phi \\
\mathcal{M}, m &\models \phi \wedge \psi \quad \text{iff} \quad \mathcal{M}, m \models \phi \text{ and } \mathcal{M}, m \models \psi \\
\mathcal{M}, m &\models \mathbf{F}\phi \quad \text{iff} \quad \exists m' \, (Rmm' \wedge \mathcal{M}, m' \models \phi) \\
\mathcal{M}, m &\models \mathbf{P}\phi \quad \text{iff} \quad \exists m' \, (Rm'm \wedge \mathcal{M}, m' \models \phi) \\
\mathcal{M}, m &\models \psi\mathbf{U}\phi \quad \text{iff} \quad \exists m' \, (Rmm' \wedge \mathcal{M}, m' \models \phi \wedge \\
&\qquad\qquad \forall m'' \, (Rmm'' \wedge Rm''m' \rightarrow \mathcal{M}, m'' \models \psi)) \\
\mathcal{M}, m &\models \psi\mathbf{S}\phi \quad \text{iff} \quad \exists m' \, (Rm'm \wedge \mathcal{M}, m' \models \phi \wedge \\
&\qquad\qquad \forall m'' \, (Rm'm'' \wedge Rm''m \rightarrow \mathcal{M}, m'' \models \psi))
\end{aligned}
$$

Fig. 1. Semantics for temporal logic.

and accessing states by names. Let $\mathsf{NOM} = \{i, j, \ldots\}$ and $\mathsf{WVAR} = \{x, y, \ldots\}$ be sets of nominals and state variables, respectively. HL's syntax is:

$$\phi := \mathrm{TL} \mid i \mid x \mid @_t\phi \mid \downarrow x.\phi \mid \exists x.\phi,$$

with $i \in \mathsf{NOM}$, $x \in \mathsf{WVAR}$, $t \in \mathsf{NOM} \cup \mathsf{WVAR}$. The operators in $\{@, \downarrow, \exists\}$ are called *hybrid operators*. We call $\mathsf{WSYM} = \mathsf{NOM} \cup \mathsf{WVAR}$ the set of *state symbols*, $\mathsf{ALET} = \mathsf{PROP} \cup \mathsf{NOM}$ the set of *atomic letters*, and $\mathsf{ATOM} = \mathsf{PROP} \cup \mathsf{NOM} \cup \mathsf{WVAR}$ the set of *atoms*. We use $x = y$ for $@_x y$ and $x \neq y$ for $@_x \neg y$. For simplicity, we omit parenthesis after the $\downarrow$. For instance, in $\downarrow x.p \wedge @_x q$, the variable x used in $@_x q$ is bound by $\downarrow x$. Hence, it should be read as $\downarrow x.(p \wedge @_x q)$.

Hybrid logic is interpreted over *hybrid Kripke structures*, i.e., Kripke structures $\langle M, R, V\rangle$ where the valuation function V assigns singleton subsets of M to nominals $i \in \mathsf{NOM}$. To give meaning to the formulas, we also need the notion of *assignment*. An assignment g is a mapping $g : \mathsf{WVAR} \rightarrow M$. Given an assignment g, we define g^x_m by $g^x_m(x) = m$ and $g^x_m(y) = g(y)$ for $x \neq y$. For any atom a, let $[V, g](a) = \{g(a)\}$ if a is a state variable, and $V(a)$ otherwise.

The semantics of hybrid logic is given in Figure 2, where $\mathcal{M} = \langle M, R, V\rangle$ is a hybrid Kripke structure, $m \in M$, and g is an assignment; the semantics for Boolean and temporal operators is as for temporal logic. In words, the at operator $@_t$ shifts evaluation to the state named by t, where t is a nominal or a variable. The downarrow binder $\downarrow x$ binds the state variable x to the *current* state (where evaluation is being performed), while the existential binder $\exists x$ binds the state variable x to *some* state in the model; $\downarrow$ and $\exists$ do not shift evaluation away from the current state. We use $\mathrm{HL}(O_1, \ldots, O_n)$ to denote the hybrid language with hybrid and temporal operators $O_1, \ldots, O_n$.

$$\begin{aligned}
\mathcal{M}, g, m \models a \quad &\text{iff} \quad m \in [V, g](a),\ a \in \mathsf{ATOM} \\
\mathcal{M}, g, m \models @_t \phi \quad &\text{iff} \quad \mathcal{M}, g, m' \models \phi, \text{ where } [V, g](t) = \{m'\},\ t \in \mathsf{WSYM} \\
\mathcal{M}, g, m \models \downarrow x.\phi \quad &\text{iff} \quad \mathcal{M}, g^x_m, m \models \phi \\
\mathcal{M}, g, m \models \exists x.\phi \quad &\text{iff} \quad \text{there is } m' \in M \text{ such that } \mathcal{M}, g^x_{m'}, m \models \phi
\end{aligned}$$

Fig. 2. Semantics for hybrid logic.

The until operator $\mathbf{U}$ can be written in hybrid logic as follows: $\alpha \mathbf{U} \beta = \downarrow x.\mathbf{F}(\beta \wedge \mathbf{H}(\mathbf{P}x \rightarrow \alpha))$, and analoguously for the since operator $\mathbf{S}$. Moreover, the past operator is $\mathbf{P}\alpha = \downarrow x.\exists y.@_y(\mathbf{F}x \wedge \alpha)$. Finally, the downarrow binder $\downarrow$ is a particular case of the existential binder: $\downarrow x.\alpha = \exists x.(x \wedge \alpha)$, while $\exists$ can be simulated by $\downarrow$ and $\mathbf{E}$ as follows: $\exists x.\alpha = \downarrow y.\mathbf{E}\downarrow x.\mathbf{E}(y \wedge \alpha)$.

In addition to the hybrid languages listed so far, we consider hybrid *dynamic* languages; these will prove to be especially useful in Section 5. We consider a hybridization of converse propositional dynamic logic (CPDL) [29]. CPDL adds two operators to propositional logic, namely $\langle e \rangle \alpha$ and $\langle e \rangle^{-1} \alpha$, where e is a regular expression on a set of labels Σ and α is a CPDL formula. CPDL is interpreted over *Labelled Transitions Systems* (LTSs), Kripke structures in which both the nodes and the edges are labelled. The edges are labelled with symbols in Σ. Each regular expression e on the set of edge labels Σ identifies a binary relation R_e on the set of states. The relation R_e is recursively defined in terms of the structure of e. More precisely, R_l contains all the edges labelled with l, $R_{e_1.e_2} = R_{e_1} \circ R_{e_2}$, $R_{e_1+e_2} = R_{e_1} \cup R_{e_2}$, and $R_{e^*} = (R_e)^*$. A state s is *reachable* from a state r through the regular expression e if $(r, s) \in R_e$. Given an LTS $\mathcal{M}$ and a state s in $\mathcal{M}$, we have that $\langle e \rangle \alpha$ is true in $\mathcal{M}$ at s if there exists a state s' reachable from s trough e such that α is true in $\mathcal{M}$ at s'. Moreover, $\langle e \rangle^{-1} \alpha$ is true in $\mathcal{M}$ at s if there exists a state s' such that s is reachable from s' through e and α is true in $\mathcal{M}$ at s'.

Hybrid dynamic logic is the *hybridization* of CPDL. We write $\mathrm{HDL}(O_1, \ldots, O_n)$ to denote the extension of CPDL with nominals, hybrid operators and possibly the universal modality in $O_1, \ldots, O_n$. For instance, $\mathrm{HDL}(@, \downarrow, \mathbf{A})$ is CPDL with nominals, @, $\downarrow$ and $\mathbf{A}$ operators.

We now introduce a new hybrid binder $\Downarrow$. We have separated the introduction of this binder because it is usually not included in hybrid languages. However, $\Downarrow$ will come in handy in Section 5. Intuitively, while $\downarrow$ stores the current state into a variable, $\Downarrow$ stores the *label* (or *value*) of the current state, that is, the set of propositions that hold at the current state. Let $\mathsf{WVAR}' = \{v, w, \ldots\}$ be a set of variables such that WVAR and WVAR' are disjoint sets. The new variables in WVAR' serve as containers for sets of propositions. We add to the hybrid language the formulas $\Downarrow v.\alpha$, v, and $v = w$, and the new abreviation $v \neq w$ for $\neg(v = w)$, where $v, w \in \mathsf{WVAR}'$, with the following meaning. Let $\mathcal{M} = \langle M, R, V \rangle$ be a hybrid Kripke structure, $m \in M$, and g an assignment extended to WVAR'; that is, g is a mapping that associates

each variable in WVAR to a state in M and each variable in WVAR$'$ to a subset of PROP. We denote by V^{-1} the function from M to the powerset of PROP such that $p \in V^{-1}(m)$ iff $m \in V(p)$. Then, $\mathcal{M}, g, m \models \Downarrow v.\alpha$ iff $\mathcal{M}, g^v_{V^{-1}(m)}, m \models \alpha$. Moreover, $\mathcal{M}, g, m \models v$ iff $g(v) = V^{-1}(m)$ and $\mathcal{M}, g, m \models v = w$ iff $g(v) = g(w)$. In words, the binder $\Downarrow v$ binds the variable v to the label of the current state, while $v = w$ compares the labels stored in v and w.

To put our results on model checking in perspective we briefly recall the complexity/decidability results for satisfiability for the logics we consider. The basic hybrid logic with just nominals and the @ operator is PSPACE-complete, not harder than modal logic. However, as soon as either the past operator, or the until operator, or the universal modality is added, the satisfiability problem becomes EXPTIME-complete. Finally, if the $\downarrow$ binder is added, decidability of the satisfiability problem is lost.

4 Model Checking for Hybrid Logics

We now investigate the complexity of the global model checking problem for various hybrid languages. A hybrid Kripke structure $\mathcal{M} = \langle M, R, V \rangle$ is *finite* if M is finite. The *global model checking problem* for hybrid logic is: Given a finite hybrid Kripke structure $\mathcal{M}$, an assignment g, and a hybrid formula ϕ, is there a state $m \in M$ such that $\mathcal{M}, g, m \models \phi$? We distinguish between *expression complexity*, i.e., the complexity of the model checking problem when the complexity parameter is the length of the formula only, and *data complexity*, i.e., the complexity of the model checking problem when the complexity parameter is the size of the model only.

4.1 Model Checkers

We provide global model checkers for different hybrid logics and we analyze their worst-case behavior. We start by describing a model checker called `MCLITE` for the language $\mathrm{HL}(@, \mathbf{F}, \mathbf{P}, \mathbf{U}, \mathbf{S}, \mathbf{A})$. It receives a hybrid model $\mathcal{M} = \langle M, R, V \rangle$, an assignment g, and a hybrid formula ϕ in the considered language and, after termination, every state in the model is labelled with the subformulas of ϕ that hold at that state. The algorithm uses a *bottom-up strategy*: it examines the subformulas of ϕ in increasing order of length, until ϕ itself has been checked.

We need some auxiliary notation. Let R be an accessibility relation; then R^- is the inverse of R: R^-vu if, and only if, Ruv. For $n \geq 1$, let $R^n(w)$ be the set of states that are reachable from w in n R-steps, and $R^{-n}(w)$ be the set of states that are reachable from w in n R^--steps. The states belonging to $R^1(w)$ are *successors* of

Procedure $\texttt{MC}_{\mathbf{F}}(\mathcal{M}, g, \alpha)$
for $w \in L(\alpha)$ **do**
  for $v \in R^{-1}(w)$ **do**
    $L(\mathbf{F}\alpha, w) \leftarrow 1$
  end for
end for
Procedure $\texttt{MC}_{\mathbf{A}}(\mathcal{M}, g, \alpha)$
if $L(\alpha) = M$ **then**
  for $v \in M$ **do**
    $L(\mathbf{A}\alpha, v) \leftarrow 1$
  end for
end if
Procedure $\texttt{MC}_{@}(\mathcal{M}, g, t, \alpha)$
let $\{w\} = [V, g](t)$
if $L(\alpha, w) = 1$ **then**
  for $v \in M$ **do**
    $L(@_t\alpha, v) \leftarrow 1$
  end for
end if

Procedure $\texttt{MC}_{\mathbf{U}}(\mathcal{M}, g, \alpha, \beta)$
for $w \in M$ **do**
  $\texttt{unmark}(w)$
end for
for $w \in L(\beta)$ **do**
  for $v \in R^{-1}(w)$ **do**
    if $L(\alpha, w) = 0$ **then**
      for $u \in R^{-1}(v)$ **do**
        $\texttt{mark}(u)$
      end for
    end if
  end for
  for $v \in R^{-1}(w) \cup R^{-2}(w)$ **do**
    if $\texttt{marked}(v)$ **then**
      $\texttt{unmark}(v)$
    else
      $L(\alpha\mathbf{U}\beta, v) \leftarrow 1$
    end if
  end for
end for

Fig. 3. MCLITE subprocedures.

w, while those belonging to $R^{-1}(w)$ are *predecessors* of w. Given a model $\mathcal{M} = \langle M, R, V\rangle$, we denote by $\mathcal{M}^-$ the model $\langle M, R^-, V\rangle$. The *length* of a formula ϕ, denoted by $|\phi|$, is the number of operators (Boolean, temporal and hybrid) of ϕ plus the number of atoms (propositions, nominals and variables) of ϕ. Let $sub(\phi)$ be the set of subformulas of ϕ. Notice that $|sub(\phi)| = O(|\phi|)$.

The model checker MCLITE updates a table L of size $|\phi| \times |M|$ whose elements are bits. Initially, $L(\alpha, w) = 1$ if, and only if, α is an atomic letter in $sub(\phi)$ such that $w \in V(\alpha)$. When MCLITE terminates, $L(\alpha, w) = 1$ if, and only if, $\mathcal{M}, g, w \models \alpha$ for every $\alpha \in sub(\phi)$. Given $\alpha \in sub(\phi)$ and $w \in M$, we denote by $L(\alpha)$ the set of states $v \in M$ such that $L(\alpha, v) = 1$ and by $L(w)$ the set of formulas $\beta \in sub(\phi)$ such that $L(\beta, w) = 1$. MCLITE uses subroutines $\texttt{MC}_{\mathbf{F}}$, $\texttt{MC}_{\mathbf{U}}$, $\texttt{MC}_{\mathbf{A}}$, and $\texttt{MC}_{@}$ in order to check subformulas of the form $\mathbf{F}\alpha$, $\alpha\mathbf{U}\beta$, $\mathbf{A}\alpha$, and $@_t\alpha$, respectively. The pseudocode for these procedures is in Figure 3.

Proposition 4.1 (Correctness of subroutines) *Let $\mathcal{M} = \langle M, R, V\rangle$ be a hybrid model, g an assignment, α and β hybrid formulas, and t a state symbol. Let L be a table such that, for every $w \in M$, $L(\alpha, w) = 1$ (respectively, $L(\beta, w) = 1$) if, and only if, $\mathcal{M}, w \models \alpha$ (respectively, $\mathcal{M}, w \models \beta$). Then,*

(1) after termination of $\texttt{MC}_{\mathbf{F}}(\mathcal{M}, g, \alpha)$, for every state $w \in M$, $L(\mathbf{F}\alpha, w) = 1$ if, and only if, $\mathcal{M}, g, w \models \mathbf{F}\alpha$;
(2) after termination of $\texttt{MC}_{\mathbf{A}}(\mathcal{M}, g, \alpha)$, for every state $w \in M$, $L(\mathbf{A}\alpha, w) = 1$ if, and only if, $\mathcal{M}, g, w \models \mathbf{A}\alpha$;

(3) after termination of $\mathtt{MC}_{@}(\mathcal{M}, g, t, \alpha)$*, for every* $w \in M$*,* $L(@_t\alpha, w) = 1$ *if, and only if,* $\mathcal{M}, g, w \models @_t\alpha$*; and*

(4) after termination of $\mathtt{MC}_{\mathbf{U}}(\mathcal{M}, g, \alpha, \beta)$*, for every* $w \in M$*,* $L(\alpha\mathbf{U}\beta, w) = 1$ *if, and only if,* $\mathcal{M}, g, w \models \alpha\mathbf{U}\beta$*.*

Proof. The proofs for cases 1, 2, and 3 are easy. We only show the more involved case 4. The procedure $\mathtt{MC}_{\mathbf{U}}$ works as follows. First, all nodes are set to unmarked. Then, for each w that is labelled with β, the first inner for loop marks all the nodes u such that there exists a node v which is *not* labelled with α and Ruv and Rvw. The second inner for loop labels with $\alpha\mathbf{U}\beta$ the remaining unmarked nodes. The set of states labelled with $\alpha\mathbf{U}\beta$ grows monotonically as computation proceeds. Indeed, a node that is not labelled with $\alpha\mathbf{U}\beta$ during the iteration for some w may be labelled with $\alpha\mathbf{U}\beta$ during a later iteration for some w'. Let $w \in L(\beta)$. We claim that:

Claim 1 *After termination of the main for loop dedicated to* w *in the procedure* $\mathtt{MC}_{\mathbf{U}}$*, for every* $v \in R^{-1}(w) \cup R^{-2}(w)$*, we have that* v *is labelled with* $\alpha\mathbf{U}\beta$ *if, and only if, every successor of* v *that is a predecessor of* w *is labelled with* α*.*

It follows that, after termination of the procedure $\mathtt{MC}_{\mathbf{U}}$, for every $v \in M$, v is labelled with $\alpha\mathbf{U}\beta$ if, and only if, every successor of v that is a predecessor of *some* $w \in L(\beta)$ is labelled with α, which means that $\mathcal{M}, v \models \alpha\mathbf{U}\beta$. To prove the left to right direction of the claim, suppose that v is labelled with $\alpha\mathbf{U}\beta$. Then, v is unmarked before the second inner for loop. We infer that every successor of v that is a predecessor of w is labelled with α. Indeed, suppose there exists a successor z of v such that z precedes w and z is not labelled with α. Because of the first inner for loop, the predecessors of z are marked. Since v is a predecessor of z, v is marked as well, which contradicts the fact that v is unmarked. For the right to left direction, suppose every successor of v that is a predecessor of w is labelled with α. Then, after the first inner for loop, v is unmarked. Indeed, suppose v is marked. Then, there exists some successor z of v that is a predecessor of w and is not labelled with α. This contradicts the fact that every successor of v that is a predecessor of w is labelled with α. Hence, v is unmarked before entering the second inner for loop and, hence, v is labelled with $\alpha\mathbf{U}\beta$ at the end of it. □

What is the complexity of the subroutines? Let $\mathcal{M} = \langle M, R, V\rangle$ be a finite hybrid model, with $n = |M|$ and $m = |R|$. The procedure $\mathtt{MC}_{\mathbf{F}}$ runs in $O(n+m)$, and both $\mathtt{MC}_{@}$ and $\mathtt{MC}_{\mathbf{A}}$ run in $O(n)$. As for $\mathtt{MC}_{\mathbf{U}}$, for every $w \in L(\beta)$, the procedure pays two backward visits to the states reachable in 2 steps. Each visit costs $O(m)$. As there are $O(n)$ states in $L(\beta)$, $\mathtt{MC}_{\mathbf{U}}$ runs in $O(n \cdot m)$ time.

A model checker `MCLITE` for $\mathrm{HL}(@, \mathbf{F}, \mathbf{P}, \mathbf{U}, \mathbf{S}, \mathbf{A})$ can easily be programmed by taking advantage of subroutines $\mathtt{MC}_{\mathbf{F}}$, $\mathtt{MC}_{\mathbf{A}}$, $\mathtt{MC}_{@}$, and $\mathtt{MC}_{\mathbf{U}}$. `MCLITE` works bottom-up checking all subformulas of the input formula in increasing length order. When a formula starting with one of $@$, $\mathbf{F}$, $\mathbf{U}$, $\mathbf{A}$ needs to be checked, the corresponding procedure is invoked. Past temporal operators $\mathbf{P}$ and $\mathbf{S}$ are handled by feeding

the subroutines $\texttt{MC}_{\mathbf{F}}$ and $\texttt{MC}_{\mathbf{U}}$, respectively, with the reversed model $\mathcal{M}^{-}$. Finally, Boolean connectives are treated as usual. The correctness of MCLITE follows from the correctness of its subroutines (Proposition 4.1) and the semantics of the operators $\mathbf{P}$ and $\mathbf{S}$.

Theorem 4.2 (Correctness of MCLITE) *Let $\mathcal{M} = \langle M, R, V\rangle$ be a hybrid model, g an assignment, and ϕ a formula in $\mathrm{HL}(@, \mathbf{F}, \mathbf{P}, \mathbf{U}, \mathbf{S}, \mathbf{A})$. Then, after termination of $\texttt{MCLITE}(\mathcal{M}, g, \phi)$, we have that, for every $w \in M$ and for every $\alpha \in sub(\phi)$, it holds that $L(\alpha, w) = 1$ if, and only if, $\mathcal{M}, g, w \models \alpha$.*

Theorem 4.3 (Complexity of MCLITE) *Let $\mathcal{M} = \langle M, R, V\rangle$ be a hybrid model such that $n = |M|$ and $m = |R|$. Let g be an assignment, and ϕ a hybrid formula of length k. Then,*

- *if ϕ belongs to $\mathrm{HL}(@, \mathbf{F}, \mathbf{P}, \mathbf{U}, \mathbf{S}, \mathbf{A})$, then the model checker $\texttt{MCLITE}(\mathcal{M}, g, \phi)$ terminates in time $O(k \cdot n \cdot m)$;*
- *if ϕ belongs to $\mathrm{HL}(@, \mathbf{F}, \mathbf{P}, \mathbf{A})$, then the model checker $\texttt{MCLITE}(\mathcal{M}, g, \phi)$ terminates in time $O(k \cdot (n + m))$; and*
- *if ϕ belongs to $\mathrm{HL}(@, \mathbf{A})$, then the model checker $\texttt{MCLITE}(\mathcal{M}, g, \phi)$ terminates in time $O(k \cdot n)$.*

Proof. There are $|sub(\phi)| = |\phi| = k$ subformulas to check. The complexity of each check depends on the form of the subformula ψ. If ψ is atomic, it is checked in constant time. If its main operator is Boolean, @, or $\mathbf{A}$, then it is checked in $O(n)$. If ψ's main operator is $\mathbf{F}$ or $\mathbf{P}$, then it is checked in $O(n + m)$. Finally, if its main operator is $\mathbf{U}$ or $\mathbf{S}$, then ψ is checked in $O(n \cdot m)$. □

We can extend MCLITE to cope with transitive closure operators $\mathbf{F}^{+}$ and $\mathbf{U}^{+}$, as well as with their past counterparts, without increasing the asymptotic complexity. The idea is to replace the visit to the predecessors of w with a backward depth-first visit of the nodes that can reach w. As an alternative, one may first compute the transitive closure of the accessibility relation, and then use the model checker MCLITE on the transitive model.

We now move to hybrid languages with binders. The efficient bottom-up strategy used in MCLITE does not work for such languages. Why? Consider the formula $\mathbf{G}{\downarrow}x.\mathbf{F}x$. It says that every successor of the current point is reflexive. If we try to check this formula in a bottom-up fashion, we initially have to check the subformula x. However, at this stage, we do not have enough information to check x, and hence we cannot label the states of the model with x. Instead, we can proceed as follows: we first check ${\downarrow}x.\mathbf{F}x$ with a procedure yet to be developed, then we check $\mathbf{G}{\downarrow}x.\mathbf{F}x$, that is $\neg\mathbf{F}\neg{\downarrow}x.\mathbf{F}x$, taking advantage of the subprocedure $\texttt{MC}_{\mathbf{F}}$ of MCLITE. In order to check ${\downarrow}x.\mathbf{F}x$, for each state w, we first assign w to x, and then check $\mathbf{F}x$ at w under the new assignment for x. The latter can again be done using $\texttt{MC}_{\mathbf{F}}$.

Procedure $\texttt{Check}_{\mathbf{F}}(\mathcal{M}, g, \alpha)$
$\texttt{MCFULL}(\mathcal{M}, g, \alpha)$
$\texttt{MC}_{\mathbf{F}}(\mathcal{M}, g, \alpha)$

Procedure $\texttt{Check}_{\downarrow}(\mathcal{M}, g, x, \alpha)$
for $w \in M$ **do**
  $g(x) \leftarrow w$
  $\texttt{MCFULL}(\mathcal{M}, g, \alpha)$
  if $w \in L(\alpha)$ **then**
    $L(\downarrow x.\alpha, w) \leftarrow 1$
  end if
  $\texttt{Clear}(L, x)$
end for

Procedure $\texttt{Check}_{\exists}(\mathcal{M}, g, x, \alpha)$
for $v \in M$ **do**
  $g(x) \leftarrow v$
  $\texttt{MCFULL}(\mathcal{M}, g, \alpha)$
  for $w \in M$ **do**
    if $w \in L(\alpha)$ **then**
      $L(\exists x.\alpha, w) \leftarrow 1$
    end if
  end for
  $\texttt{Clear}(L, x)$
end for

Fig. 4. MCFULL subprocedures.

The sketched strategy, which combines top-down and bottom-up reasoning, has been implemented in a recursive model checker MCFULL for the full hybrid language $\mathrm{HL}(\mathrm{HO} \cup \mathrm{TO})$, where $\mathrm{HO} = \{@, \downarrow, \exists\}$, and $\mathrm{TO} = \{\mathbf{F}, \mathbf{P}, \mathbf{U}, \mathbf{S}, \mathbf{A}\}$. The auxiliary procedures $\texttt{Check}_*$, with $* \in \mathrm{HO} \cup \mathrm{TO}$, handle the cases of subformulas with main operator $*$. Whenever possible, these procedures re-use the subprocedures $\texttt{MC}_*$, with $* \in \{@\} \cup \mathrm{TO}$, of MCLITE. They use the new subroutine $\texttt{Clear}(L, x)$ to reset all the values of $L(\alpha)$, for any α containing the variable x free (we assume that different binders use different variables). Boolean operators are treated as usual. In Figure 4 we show the pseudocode for $\texttt{Check}_{\mathbf{F}}$, $\texttt{Check}_{\downarrow}$, and $\texttt{Check}_{\exists}$. The procedures for the other operators are similar to $\texttt{Check}_{\mathbf{F}}$.

Theorem 4.4 (Correctness of MCFULL) *Let $\mathcal{M} = \langle M, R, V\rangle$ be a hybrid model, g an assignment, and ϕ a hybrid formula. Then, after termination of* $\texttt{MCFULL}(\mathcal{M}, g, \phi)$*, we have that:*

- *for every $w \in M$, $L(\phi, w) = 1$ if, and only if, $\mathcal{M}, g, w \models \phi$; and*
- *for every $w \in M$ and every sentence $\alpha \in sub(\phi)$, $L(\alpha, w) = 1$ if, and only if, $\mathcal{M}, g, w \models \alpha$.*

Theorem 4.5 (Complexity of MCFULL) *Let $\mathcal{M} = \langle M, R, V\rangle$ be a hybrid model such that $n = |M|$ and $m = |R|$, and g an assignment. Let ϕ be a hybrid formula in $\mathrm{HL}(\exists, \downarrow, S)$, and let $r_{\downarrow}$ and $r_{\exists}$ be the nesting degree of $\downarrow$ and $\exists$, respectively, in ϕ. Let $\mathcal{C}_S$ be the model checking complexity for $\mathrm{HL}(S)$. Then,* $\texttt{MCFULL}(\mathcal{M}, g, \phi)$ *terminates in time $O(\mathcal{C}_S \cdot n^{r_{\downarrow}+r_{\exists}+1})$ if $r_{\exists} > 0$, and in time $O(\mathcal{C}_S \cdot n^{r_{\downarrow}})$ if $r_{\exists} = 0$. Moreover, the procedure uses polynomial space.*

Proof. We have that (1) the procedure $\texttt{Check}_*$ runs in time $\mathcal{C}_\alpha + \mathcal{C}_*$, where $\mathcal{C}_*$ is the cost of $\texttt{MC}_*$ and $\mathcal{C}_\alpha$ is the cost to check α; (2) the procedure $\texttt{Check}_{\downarrow}$ runs in time $n \cdot \mathcal{C}_\alpha$; and (3) the procedure $\texttt{Check}_{\exists}$ runs in time $n \cdot (\mathcal{C}_\alpha + n)$. Thus, the overall worst-case time complexity is $O(\mathcal{C}_S \cdot n^{r_{\downarrow}+r_{\exists}+1})$ if $r_{\exists} > 0$, and it is $O(\mathcal{C}_S \cdot n^{r_{\downarrow}})$ if $r_{\exists} = 0$. Since the height of the recursion stack for MCFULL is at most $|\phi|$, we have

that MCFULL uses polynomial space. □

MCFULL can easily be extended to cope with formulas involving $\Downarrow$ and the comparison of state labels. A procedure similar to $\texttt{Check}_{\downarrow}$ can be used to check formulas of the form $\Downarrow v.\alpha$. The only difference is that we have to bind the variable v to the *label* of the current state, and not to the current state itself. The complexity remains the same. Moreover, the formula v, for $v \in \mathsf{WVAR}'$, can be verified by comparing the label of the current state and the label stored in v, while the formula $v = w$, for $v, w \in \mathsf{WVAR}'$, can be checked by comparing the labels stored in v and w.

Furthermore, MCFULL can be viewed as a general model checker for the hybridization of *any* temporal logic. Given a temporal logic **T**, the hybridization of **T** is the hybrid logic obtained from the language of **T** by adding the hybrid machinery. Suppose that, for each temporal operator **O** of arity k in **T**, we can exploit a procedure $\texttt{Check}_{\mathbf{O}}$ that, given a model and k formulas $\alpha_1, \ldots, \alpha_k$, labels each state m of the model with the formula $\mathbf{O}(\alpha_1, \ldots, \alpha_k)$ if, and only if, the formula is true at m. A model checker for the hybridization of **T** can be synthesized from these model checking procedures following the example of MCFULL. For instance, a model checker for $\mathrm{HDL}(@, \downarrow, \Downarrow, \mathbf{A})$ can be programmed by taking advantage of a CPDL model checker. Model checking for CPDL can be done in linear time with respect to the length of the formula and the size of the model [21]. Hence, we have:

Theorem 4.6 *Model checking for* $\mathrm{HDL}(@, \mathbf{A})$ *has linear time data and expression complexity, while model checking for* $\mathrm{HDL}(@, \downarrow, \Downarrow, \mathbf{A})$ *has exponential time expression complexity and polynomial time data complexity. In any case, the problem can be solved in polynomial space.*

4.2 Lower Bounds

Can we do better than the upper bounds given in Section 4.1? Model checking for first-order logic is PSPACE-complete. Since $\mathrm{HL}(\exists, @, \mathbf{F})$ is as expressive as first-order logic [15], model checking for $\mathrm{HL}(\exists, @, \mathbf{F})$ is PSPACE-complete as well. What about fragments of $\mathrm{HL}(\exists, @, \mathbf{F})$? The question is particulary interesting for the fragment $\mathrm{HL}(\downarrow, @, \mathbf{F})$, since it corresponds to the *bounded fragment* of the first-order correspondence language [9]. Unfortunately, the model checking problem for $\mathrm{HL}(\downarrow, @, \mathbf{F})$, and hence for the bounded fragment, is still PSPACE-hard, even without $@$, nominals and propositions. It is worth noticing that all the lower bounds in this section refer to *expression complexity*.

Theorem 4.7 *Model checking for the pure nominal-free fragment of* $\mathrm{HL}(\downarrow, \mathbf{F})$ *is* PSPACE*-complete.*

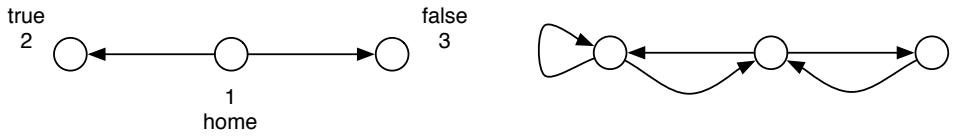


Fig. 5. Embedding QBF into model checking for hybrid logics.

$$\tau(x) = \mathbf{F}(x \wedge \downarrow y.\mathbf{F}y) \qquad \tau(\neg x) = \mathbf{F}(x \wedge \downarrow y.\mathbf{G}\neg y)$$
$$\tau(\alpha_1 \wedge \alpha_2) = \tau(\alpha_1) \wedge \tau(\alpha_2) \qquad \tau(\alpha_1 \vee \alpha_2) = \tau(\alpha_1) \vee \tau(\alpha_2)$$
$$\tau(\exists x.\alpha) = \mathbf{F}{\downarrow}x.\mathbf{F}(\downarrow y.\mathbf{G}\neg y \wedge \tau(\alpha)) \quad \tau(\forall x.\alpha) = \mathbf{G}{\downarrow}x.\mathbf{F}(\downarrow y.\mathbf{G}\neg y \wedge \tau(\alpha))$$

Fig. 6. From QBF to hybrid logic.

Proof. PSPACE-membership follows from Theorem 4.5. To prove PSPACE-hardness, we embed Quantified Boolean Formulas (QBF) [34] into the model checking problem for the pure nominal-free fragment of $\mathrm{HL}(\downarrow, \mathbf{F})$. We proceed in two steps. We first embed QBF into the model checking problem for $\mathrm{HL}(\downarrow, @, \mathbf{F})$. Then, we remove the $@$ operator and atomic letters.

Recall that an instance of QBF has the form $\Psi = Q_1 x_1 \ldots Q_n x_n.\alpha(x_1, \ldots, x_n)$, where $Q_i \in \{\exists, \forall\}$, and $\alpha(x_1, \ldots, x_n)$ is a Boolean formula using variables x_1, ..., x_n. Let $\mathsf{NOM} = \{\mathtt{true}, \mathtt{false}, \mathtt{home}\}$ and let $\mathcal{M}$ be the model depicted in Figure 5, left-hand side. Let ϕ_Ψ be the $\mathrm{HL}(\downarrow, @, \mathbf{F})$-formula obtained from Ψ by replacing every occurrence of $\exists x$ by $@_{\mathtt{home}}\mathbf{F}{\downarrow}x$, every occurrence of $\forall x$ by $@_{\mathtt{home}}\mathbf{G}{\downarrow}x$, every occurrence of x by $@_x\mathtt{true}$, and every occurrence of $\neg x$ by $@_x\mathtt{false}$. We have that Ψ is true if, and only if, $\mathcal{M}, 1 \models \phi_\Psi$. We now remove $@$ and atomic letters. To remove $@$, we have to find a way to "come back home" after $\mathbf{F}{\downarrow}x$ has fixed the variable x. We can add to the previous model two more edges, one from 2 to 1, and the other from 3 to 1, and use the $\mathbf{F}$ operator to come home. Since we don't have atomic letters, we have to distinguish in some structural way between state 2 (denoting true) and state 3 (denoting false). We can add, for instance, a reflexive edge leaving 2. The resulting frame is depicted in Figure 5 (right-hand side). Without loss of generality, we assume that negation in $\alpha(x_1, \ldots, x_n)$ is applied only to variables. Let τ be the translation given in Figure 6. We leave it to the reader to check that Ψ is true if and only if $\mathcal{M}, 1 \models \tau(\Psi)$. $\square$

Theorem 4.8 *Model checking for the pure nominal-free fragments of* $\mathrm{HL}(\downarrow, \mathbf{F}^+)$ *is* PSPACE*-complete.*

Proof. PSPACE-membership follows from Theorem 4.5. To prove hardness, we embed QBF into the model checking problem for the pure nominal-free fragment of $\mathrm{HL}(\downarrow, \mathbf{F}^+)$. Let $\Psi = Q_1 x_1 \ldots Q_n x_n.\alpha(x_1, \ldots, x_n)$, where $Q_i \in \{\exists, \forall\}$, and $\alpha(x_1, \ldots, x_n)$ is a Boolean formula using variables $x_1, \ldots, x_n$. Let $\mathcal{M}$ be the unlabelled model with frame $\langle\{1, 2\}, \{(1, 2), (2, 1)\}\rangle$. Let ϕ_Ψ be the $\mathrm{HL}(\downarrow, @, \mathbf{F})$-formula obtained from Ψ by replacing every occurrence of $\exists x$ by $\mathbf{F}^+{\downarrow}x$ and every occurrence

of $\forall x$ by $\mathbf{G}^+{\downarrow}x$. Then Ψ is true if, and only if, $\mathcal{M}, 1 \models \phi_\Psi$. $\square$

Theorem 4.9 *Model checking for the pure nominal-free fragment of* $\mathrm{HL}(\exists)$ *is* PSPACE-*complete.*

Proof. PSPACE-membership follows from Theorem 4.5. To prove hardness, we embed QBF into the model checking problem for the pure nominal-free fragments of $\mathrm{HL}(\exists)$. Let $\Psi = Q_1 x_1 \ldots Q_n x_n . \alpha(x_1, \ldots, x_n)$, where $Q_i \in \{\exists, \forall\}$, and $\alpha(x_1, \ldots, x_n)$ is a Boolean formula using variables $x_1, \ldots, x_n$. Ψ is a pure nominal-free formula in $\mathrm{HL}(\exists)$. Let $\mathcal{M}$ be the unlabelled model based on the frame $\langle \{1, 2\}, \emptyset \rangle$. Then Ψ is true if, and only if, $\mathcal{M}, 1 \models \Psi$. $\square$

Corollary 4.10 *Model checking for the pure nominal-free fragment of* $\mathrm{HL}(\downarrow, \mathbf{A})$ *is* PSPACE-*complete.*

Theorem 4.11 *Model checking for* $\mathrm{HL}(\Downarrow, \mathbf{F})$ *is* PSPACE-*complete.*

Proof. The PSPACE upper bound comes from Section 4.1. To prove hardness, we embed the model checking problem for $\mathrm{HL}(\downarrow, \mathbf{F})$ into the same problem for $\mathrm{HL}(\Downarrow, \mathbf{F})$. Consider a hybrid model $\mathcal{M} = \langle M, R, V \rangle$ and a formula α in $\mathrm{HL}(\downarrow, \mathbf{F})$. We construct a hybrid model $\mathcal{M}' = \langle M, R, V' \rangle$, where, for each $m \in M$, there exists a fresh nominal i_m with $V'(i_m) = \{m\}$. Moreover, let α' be the formula in $\mathrm{HL}(\Downarrow, \mathbf{F})$ that is obtained from α by replacing each instance of $\downarrow$ by $\Downarrow$ and each instance of x by v_x, where $x \in \mathsf{WVAR}$ and $v_x \in \mathsf{WVAR}'$. Then, we have that $\mathcal{M}, g, m \models \alpha$ if, and only if, $\mathcal{M}', g, m \models \alpha'$. $\square$

Theorem 4.12 *Model checking for the pure nominal-free fragment of* $\mathrm{HDL}(\downarrow)$ *is* PSPACE-*complete.*

5 Hybrid Logic in Action

On the previous pages we have discussed, and obtained, complexity results for model checking a variety of hybrid logics. What's the point? In this section we describe an application of hybrid logic model checking to the task of evaluation of queries and constraints on semistructured data. We will encounter a number of query and constraint formats as well as reasoning tasks, all of which correspond to model checking in some specific hybrid language.

5.1 Semistructured data

The growth of the World Wide Web has given us a vast, largely accessible database. The Extensible Markup Language (XML) [22] is a textual representation of information that was designed to represent the hierarchical content of documents. It has been proposed as a data model for semistructured databases since it is able to naturally represent missing or duplicated data as well as deeply nested information [1]. As an example, Figure 7 contains the XML representation of a simple bibliography file. A very natural representation for semistructured data is a labelled graph. In this paper, we will represent semistructured data as a graph in which a node corresponds to an object and an edge corresponds to an object attribute. Edges are labelled with attribute names, while leaf nodes are labelled with object values (internal nodes are not labelled). The data graph corresponding to the XML example on the left-hand side of Figure 7 is depicted on the right-hand side of Figure 7.

5.2 Query Processing via Hybrid Logic Model Checking

It is possible to perform significant query processing on semistructured data via model checking for hybrid languages. Quite a number of query languages for semistructured data have been proposed in the literature. However, many of them are similar in spirit and sometimes even in syntax [1]. For this reason, we have chosen one of them as a model: Lorel [3]. Lorel is the query language in the Lore system [32], which was designed for managing semistructured data. We discuss three fragments of Lorel, that differ in their expressive power; each will be embedded in a suitable hybrid logic. As a consequence, we are able to process a query in Lorel by taking advantage of a model checker for hybrid logics. Moreover, with the aid

```
<biblio>
  <book id = "o1">
    <author> Marx </author>
    <author> de Rijke </author>
    <title> Hybrid Logics </title>
  </book>
  <book id = "o2">
    <author> Franceschet </author>
    <title> Model Checking </title>
    <date> 2000 </date>
    <cite idref = "o1"/>
    <cite idref = "o3"/>
  </book>
  <paper id = "o3">
    <author> Afanasiev </author>
    <title> Model Checking </title>
    <cite idref = "o2"/>
  </paper>
</biblio>
```

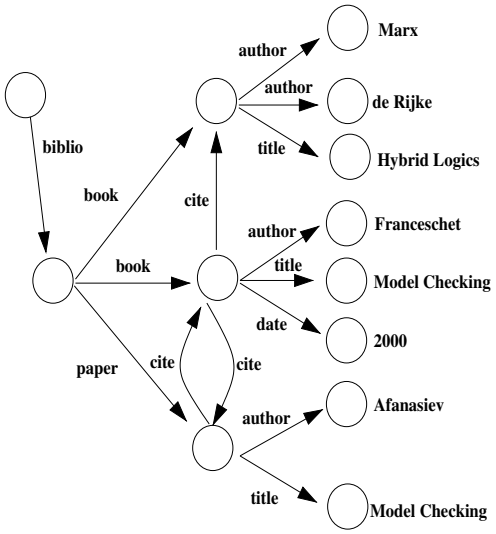


Fig. 7. (Left) An XML representation of a bibliography file. (Right) A graphical representation of the same file.

of the well-understood hybrid logic framework, it will be easy to investigate and compare the expressive power of various fragments of the Lorel query language.

All fragments that we consider allow regular expressions for navigating the data graph. Some offer the possibility of comparing object identities, which allows us to implement queries with joins. We will focus on *monadic queries* only: queries that return a set of objects of the database, or, equivalently, a set of nodes of the graph representation of the database. The reason for this restriction is that we want to embed the query processing problem into the global model checking problem, and the output of a global model checker is a set of nodes. We will consider 3 increasingly large fragments of Lorel: $\mathcal{L}^1_{\rm qry} \subset \mathcal{L}^2_{\rm qry} \subset \mathcal{L}^3_{\rm qry}$. The most expressive language, $\mathcal{L}^3_{\rm qry}$, captures a large subfragment of Lorel. Additional features of Lorel that are not expressible in $\mathcal{L}^3_{\rm qry}$ will be discussed towards the end of the section.

Fragment 1: $\mathcal{L}^1_{\rm qry}$ We start by defining the query language $\mathcal{L}^1_{\rm qry}$. A query in $\mathcal{L}^1_{\rm qry}$ has the following schema Q_1:

```
(Q1) select X
     from rexp X
     where X.fexp,
```

where `rexp` is a regular expression on edge labels (i.e., attribute names), and `fexp` is a so-called filter expression. Intuitively, Q_1 retrieves all nodes reachable from the root through `rexp` satisfying the filter `fexp`. The variable `X` is used as a container for these nodes. We call the variable `X` in the 'select' clause of the query the *focus* of the query, the condition `rexp X` in the 'from' clause of the query the *selection expression,* and the condition `X.fexp` in the 'where' clause of the query the *filter expression*. The syntax of filter expressions is:

`fexp` $=$ `true` | `rexp` | `rexp` :: `a` | `fexp and fexp` | `fexp or fexp` | `not fexp`,

where `a` is an object value. Given a set of nodes `X`, the filter `X.rexp` selects a node v in `X` if there exists at least one reachable node from v trough `rexp`. The filter `X.rexp::a` adds an additional constraint: it filters a node v in `X` if there exists at least one reachable node from v trough `rexp` that is labelled with a. E.g., with reference the example in Figure 7, consider the following:

```
select X
from biblio.book X
where X.(author::Franceschet and date::2000)
```

This query selects all books written in 2000 such that Franceschet is one of the authors. In our example, book $o2$ is retrieved. Moreover, the query

```
select X
```

```
from biblio._ X
where X.abstract
```

retrieves all entries for which an abstract has been provided: none.

The simple query language $\mathcal{L}^1_{\text{qry}}$ can be embedded into hybrid dynamic logic with only one nominal $root$ for the data graph root and no hybrid operators. The embedding τ_1 is as follows. Let ω be the obvious embedding from filter expressions to hybrid formulas: $\omega(\texttt{true}) = \top$, $\omega(\texttt{rexp}) = \langle \texttt{rexp} \rangle \top$, $\omega(\texttt{rexp :: a}) = \langle \texttt{rexp} \rangle a$, $\omega(\texttt{fexp}_1 \texttt{ and } \texttt{fexp}_2) = \omega(\texttt{fexp}_1) \wedge \omega(\texttt{fexp}_2)$, $\omega(\texttt{fexp}_1 \texttt{ or } \texttt{fexp}_2) = \omega(\texttt{fexp}_1) \vee \omega(\texttt{fexp}_2)$, and $\omega(\texttt{not fexp}) = \neg\omega(\texttt{fexp})$. To the query schema Q_1 we associate the hybrid formula $\tau_1(Q_1) = \langle \texttt{rexp} \rangle^{-1} root \wedge \omega(\texttt{fexp})$. Notice how the selection expression is translated 'backward,' while the filter expression is translated 'forward.'

Next, we clarify the relation between query processing for semistructured data and model checking for hybrid logic. Define the *answer set* of a monadic query q with respect to a semistructured database D as the set of nodes retrieved by q belonging to the data graph G_D associated to D. Each node in the answer set corresponds to an element in the corresponding XML representation of D. Moreover, given a hybrid formula α and a hybrid model $\mathcal{M}$, let the *truth set* of α with respect to $\mathcal{M}$ be the set of nodes of $\mathcal{M}$ at which α is true. The following result relates the query processing problem for semistructured data to the model checking problem for hybrid dynamic logic.

Proposition 5.1 *Let q be a query in $\mathcal{L}^1_{\text{qry}}$ and D be a semistructured database. Then, the answer set of q with respect to D corresponds to the truth set of $\tau_1(q)$ with respect to G_D.*

Moreover, since the translation τ_1 goes inside a fragment of $\text{HDL}(@, \mathbf{A})$, by virtue of Theorem 4.6, we have the following corollary.

Corollary 5.2 *Query processing for $\mathcal{L}^1_{\text{qry}}$ has linear time data and expression complexity.*

Fragment 2: $\mathcal{L}^2_{\text{qry}}$ The expressive power of $\mathcal{L}^1_{\text{qry}}$ can be extended by splitting the selection expression in more than one chunk. Consider the following query that selects the authors of all entries with title *Model Checking*:

```
select Y
from biblio._ X, X.author Y
where X.title::Model Checking
```

This query is not definable in $\mathcal{L}^1_{\text{qry}}$ but it can be embedded into the basic hybrid dynamic logic as $\langle \texttt{author} \rangle^{-1} (\langle \texttt{bibilo.\_} \rangle^{-1} root \wedge \langle \texttt{title} \rangle Model\ Checking)$. We

define the query schema Q_2 as follows:

(Q_2) `select` $\mathtt{X_i}$
`from` $\mathtt{rexp_1\ X_1,\ X_1.rexp_2\ X_2,\ \ldots,\ X_{n-1}.rexp_n\ X_n}$
`where` $\mathtt{fexp_1,\ \ldots,\ fexp_m}$

Each filter expression $\mathtt{fexp_j}$ has the form $\mathtt{X.fexp}$, where $\mathtt{X}$ is a variable and $\mathtt{fexp}$ in a filter expression as in Q_1. Intuitively, the schema Q_2 binds the variable $\mathtt{X_1}$ to the nodes reachable from the root through the regular expression $\mathtt{rexp_1}$, it binds the variable $\mathtt{X_2}$ to the nodes reachable from a node in $\mathtt{X_1}$ through the regular expression $\mathtt{rexp_2}$, and so on. The filter expression $\mathtt{X.fexp}$ filters the nodes placed in the variable $\mathtt{X}$ according to the Boolean filter $\mathtt{fexp}$. Finally, the nodes contained in the focus $\mathtt{X_i}$ are selected. A well-formed query q is defined as follows. Let X be the focus of q, S the selection sequence of q, F the filter sequence of q, and V the variables in S. The query q is *well-formed* if (i) $X \in V$, and (ii) all variables used in the filter sequence F are in V. Let $\mathcal{L}^2_{\mathrm{qry}}$ be the query language containing all the well-formed queries according to the schema Q_2. Notice that $\mathcal{L}^1_{\mathrm{qry}} \subset \mathcal{L}^2_{\mathrm{qry}}$.

We now develop an embedding τ_2 of queries in $\mathcal{L}^2_{\mathrm{qry}}$ into the basic hybrid dynamic logic with only one nominal *root* for the root of the data graph. Consider the query schema Q_2. For each variable $\mathtt{X_j}$ in Q_2, let $\mathtt{fexp_j}$ be the filter expression associated with the variable $\mathtt{X_j}$, or $\mathtt{fexp_j} = \mathtt{true}$ if no filter expression has been explicitly associated with $\mathtt{X_j}$. We use the following two auxiliary functions ν and ν^{-1} mapping a variable in Q_2 into a hybrid formula.

$$\nu(X_j) = \begin{cases} \omega(\mathtt{fexp_j}) \wedge \langle \mathtt{rexp_{j+1}} \rangle\, \nu(\mathtt{X_{j+1}}) & \text{if } j < n \\ \omega(\mathtt{fexp_j}) & \text{if } j = n \end{cases}$$

$$\nu^{-1}(X_j) = \begin{cases} \omega(\mathtt{fexp_j}) \wedge \langle \mathtt{rexp_j} \rangle^{-1}\, \nu^{-1}(\mathtt{X_{j-1}}) & \text{if } j > 1 \\ \omega(\mathtt{fexp_j}) \wedge \langle \mathtt{rexp_j} \rangle^{-1} root & \text{if } j = 1 \end{cases}$$

Let $\mathtt{X_i}$ be the focus of Q_2, that is, $\mathtt{X_i}$ is the variable in the select clause of Q_2. The translation τ_2 of Q_2 into hybrid logic is as follows:

$$\tau_2(Q) = \begin{cases} \nu(\mathtt{X_i}) \wedge \langle \mathtt{rexp_i} \rangle^{-1}\, \nu^{-1}(\mathtt{X_{i-1}}) & \text{if } i > 1 \\ \nu(\mathtt{X_i}) \wedge \langle \mathtt{rexp_i} \rangle^{-1} root & \text{if } i = 1 \end{cases}$$

Notice that $\tau_2(\mathcal{L}^1_{\mathrm{qry}}) = \tau_1(\mathcal{L}^1_{\mathrm{qry}})$.

Proposition 5.3 *Let q be a query in $\mathcal{L}^2_{\mathrm{qry}}$ and D be a semistructured database. Then, the answer set of q with respect to D corresponds to the truth set of $\tau_2(q)$ with respect to G_D.*

Since the codomain of the translation τ_2 is a fragment of $\mathrm{HDL}(@, \mathbf{A})$, by virtue of Theorem 4.6, we have the following corollary:

Corollary 5.4 *Query processing for $\mathcal{L}^2_{\mathrm{qry}}$ has linear time data and expression complexity.*

The results in Corollaries 5.2 and 5.4 do not really depend on previous results for model checking hybrid logics, and in fact they can be obtained directly from model checking converse PDL. Indeed, the root nominal can be simulated by a propositions holding at exactly the root node. The material below does use the expressive power of hybrid logic in an essential way.

Fragment 3: $\mathcal{L}^3_{\mathrm{qry}}$ So far, we have not used the full power of hybrid logic. In particular, the hybrid binder $\downarrow$ has not been exploited in the query translation. Consider the following query that selects papers with at least two authors:

```
select X
from biblio.paper X, X.author Y, X.author Z
where Y ≠ Z
```

It can be embedded in hybrid logic, but we need the binder $\downarrow$:

$$\downarrow x.\langle \texttt{biblio.paper}\rangle^{-1} root \wedge \langle \texttt{author}\rangle \downarrow y.@_x \langle \texttt{author}\rangle \downarrow z.\ y \neq z.$$

Moreover, the following query selects all the self-reference papers, that is, all papers whose authors cite themselves:

```
select X
from biblio.paper X, X.cite Y
where X = Y
```

It corresponds to the hybrid formula $\downarrow x.\langle \texttt{biblio.paper}\rangle^{-1} root \wedge \langle \texttt{cite}\rangle \downarrow y.x = y$. The following query retrieves all papers p such that there exists a paper q reachable from p trough a path of cite edges that cites back to p:

```
select X
from biblio.paper X, X.cite.cite* Y
where X = Y
```

This query maps to $\downarrow x.\langle \texttt{biblio.paper}\rangle^{-1} root \wedge \langle \texttt{cite.cite}^*\rangle \downarrow y.x = y$.

We define a query schema Q_3 that allows the above queries as follows:

```
(Q3) select Xi
  from sexp1, sexp2, ..., sexpn
  where fexp1, fexp2, ..., fexpm,
```

where each $\texttt{sexp}_\texttt{j}$ is a selection expression and each $\texttt{fexp}_\texttt{j}$ is a filter expression. A selection expression is either `rexp X` or `X.rexp Y`, where `rexp` is a regular expression and `X` and `Y` are variables. A filter expression is either `X.fexp`, or $\texttt{X} = \texttt{Y}$, or $\texttt{X} \neq \texttt{Y}$, where `X` and `Y` are variables and `fexp` is a filter expression as in Q_1. Not every selection sequence is legal. For instance, `X.a Y`, `X.b Y` is not legal for two reasons. First, it does not specify the content of `X`. Second, the content of `Y` is ambiguous. A well-formed query q is defined as follows. Let X be the focus of q, S the selection sequence of q, F the filter sequence of q, and V the variables in S. We construct the edge-labelled directed graph T_S with nodes in $V \cup \{/\}$. There is an edge $(/, \texttt{X})$ labelled with `rexp` if `rexp X` is in S, and there is an edge $(\texttt{X}, \texttt{Y})$ labelled with `rexp` if `X.rexp Y` is in S. We say that S is *legal* if T_S is a tree rooted at $/$. That is (i) each node in T_S is reachable from $/$, (ii) the root $/$ of T_S has no predecessor and all the other nodes in T_S have exactly one predecessor. The query q is *well-formed* if (i) $X \in V$, (ii) S is legal, and (iii) all the variables used in the filter sequence F are in V. Let $\mathcal{L}^3_{\text{qry}}$ be the query language containing all the well-formed queries according to the schema Q_3. Notice that $\mathcal{L}^2_{\text{qry}} \subset \mathcal{L}^3_{\text{qry}}$.

We now develop an embedding τ_3 of queries in $\mathcal{L}^3_{\text{qry}}$ into the hybrid dynamic logic with nominals, @ and $\downarrow$. For the sake of simplicity, we describe the embedding with an example. It is not difficult from this example to reconstruct the full query translation. Consider the following abstract query q:

```
select Z
from a X, X.b Y, X.c Z, Z.d W
where X.e,Y.f,Z.g,W.h,Y = W
```

Notice that q is well-formed. The corresponding hybrid formula is as follows:

$$\downarrow z.\langle g\rangle\top \wedge \langle d\rangle\downarrow w.\langle h\rangle\top \wedge$$
$$@_z\langle c\rangle^{-1}\downarrow x.\langle e\rangle\top \wedge \langle b\rangle\downarrow y.\langle f\rangle\top \wedge @_x\langle a\rangle^{-1}root \wedge$$
$$y = w$$

The formula has been deliberately divided into three lines. The first line constraints the focus of the query (node `Z`) and its subtree (node `W`). The second line predicates over the unique path from the parent of the focus to the root of the tree as well as over all the subtrees rooted at children of nodes on this path not belonging to the path (nodes `X`, `Y`, and `/`). The third line captures the identity checking constraints. This technique can be generalized to arbitrary trees. Notice that the use of the binder $\downarrow$ is necessary to encode the constraint that compares object identities. Indeed, if we remove the filter `Y = W` from the above query, the latter can be encoded without $\downarrow$ as follows:

$$\langle g\rangle\top \wedge \langle d\rangle\langle h\rangle\top \wedge$$
$$\langle c\rangle^{-1}(\langle e\rangle\top \wedge \langle b\rangle\langle f\rangle\top \wedge \langle a\rangle^{-1}root)$$

Proposition 5.5 *Let q be a query in $\mathcal{L}^3_{\text{qry}}$ and D be a semistructured database.*

Then, the answer set of q with respect to D corresponds to the truth set of $\tau_3(q)$ with respect to G_D.

Since the the translation τ_3 embeds into $\mathrm{HDL}(@, \downarrow, \Downarrow, \mathbf{A})$, by virtue of Theorem 4.6, we have the following corollary (we do not know whether it is optimal):

Corollary 5.6 *Query processing for $\mathcal{L}^3_{\mathrm{qry}}$ has exponential time expression complexity and polynomial time data complexity.*

Additional Features of Lorel Lorel includes additional queries that are not immediately expressible in our language. For instance, a query in Lorel may use regular expressions on object values, as in the following:

```
select X
from biblio.paper X, X.title Y
where matches("*.(D|d)ata.*",Y)
```

The query selects all papers with a title containing either `Data` or `data`. Regular expressions on object values can be incorporated into our model checking framework as follows. Given a query q featuring the regular expression r and a database D, the data graph representing D is preprocessed and all leaf nodes with a value matching r are labelled with a fresh symbol `reg_r`. Moreover, each instance of `matches("*.(D|d)ata.*",Y)` in q is replaced by `Y.`ϵ`::reg_r`. The query is then translated into hybrid logic and model checking is applied.

Lorel queries may also allow variables containing object values, instead of object identifiers. Consider the following query that retrieves all papers with two different authors with the same first name (we assume that the `author` element has a subelement `name` which in turn has an atomic subelement `first`):

```
select X
from biblio.paper X, X.author Y, Y.name.first N,
X.author Z, Z.name.first M
where Y ≠ Z, N = M
```

The constraint `Y` $\neq$ `Z` compares object identities, while `N = M` compares object values, since `Y` and `Z` contain internal nodes, while `N` and `M` contain leaf nodes. This query can be implemented in hybrid logic by using the hybrid binder $\Downarrow x$ that binds the current node value to the variable x:

$$\langle \texttt{biblio.paper} \rangle^{-1} root \wedge \downarrow x.\langle \texttt{author} \rangle \downarrow y.\langle \texttt{name.first} \rangle \Downarrow n.$$
$$@_x \langle \texttt{author} \rangle \downarrow z.\langle \texttt{name.first} \rangle \Downarrow m. y \neq z \wedge n = m$$

Finally, Lorel includes label and path variables as well. These variables can be used to combine schema and data information. These features are beyond the expressive

```
<author id = "a1">
  <name> Marx </author>
  <has_written idref = "p1"/>
  <has_written idref = "p2"/>
</author>
<author id = "a2">
  <name> de Rijke </author>
  <has_written idref = "p1"/>
</author>
<publication id = "p1">
  <title> Hybrid Logics </title>
  <code> MdeR03 </code>
  <year> 2003 </year>
  <written_by idref = "a1"/>
  <written_by idref = "a2"/>
</publication>
<publication id = "p2">
  <title> Computational Complexity </title>
  <code> M00 </code>
  <year> 2000 </year>
  <written_by idref = "a1"/>
</publication>
⋮
```

Fig. 8. Authors and publications in XML.

power of hybrid logic with nominals for states. *Path nominals*, i.e., nominals interpreted over paths, and corresponding path binders, have been introduced in hybrid logics [16]. However, at present model checking results for hybrid logics with path nominals and path binders are non-existent.

Constraint Evaluation via Hybrid Model Checking So far we have considered hybrid logic model checking as a mechanism for evaluating queries in (fragments of) Lorel. We now change tack and consider constraint evaluation for semistructured data. Recall that integrity constraints on structured data are conditions that restrict the possible populations of the database. They are important to maintain the integrity of data as well as for query optimization [2]. *Path constraints* are generalizations of integrity constraints in the context of semistructured data [1,24]. They are navigation conditions imposing conditions on nodes at arbitrary depths in the data graph. They include functional, inclusion and inverse path constraints [24].

Consider a semistructured database containing information about authors and publications. Each author has a list of publications and each publication has a corresponding list of authors. An example in XML is shown in Figure 8. Constraints such as the following are reasonable for this type of data:

(1) for each author a, all the publications in a's publications list should be contained in the database;
(2) for each publication p, all the authors in p's authors list should be contained in the database;
(3) for each author a and for each publication p in a's publications list, a should be contained in p's authors list; and
(4) for each publication p and for each author a in p's authors list, p should be contained in a's publications list.

The data in Figure 8 satisfies the above constraints. We could require that the attribute `code` is a *key* for the element publication: different publications should have different code values. Less restrictively, we could ask that the attribute `code` *functionally determines* the authors of a publication: any two different publications with the same code values should have the same authors.

More generally, let r, p and q be regular expressions on attribute names. Let τ be an attribute name and S a set of attribute names. A τ node is a node reachable through an edge labelled with τ. We consider the following constraints:

- *key constraints*, denoted $\tau[S] \rightarrow \tau$, saying that, for all τ nodes x and y, if x and y agree on the values of nodes reachable through attributes in S, then x and y are the same node;
- *path functional constraints*, denoted $\tau.p \rightarrow \tau.q$, saying that, for all τ nodes x and y, if x and y agree on the values of nodes reachable through p, then they should agree on the values of nodes reachable through q as well;
- *circular constraints*:
  - *forward* version, denoted $p \rightrightarrows q$, saying that all nodes reachable from the root through p are also reachable from the root through q;
  - *backward* version, denoted $p \leftrightarrows q$, saying that all nodes reachable from the root through p can reach back to the root through q;
- *lollipop constraints*:
  - *forward* version, denoted $r \rightarrow p \rightrightarrows q$, saying that, for each node x reachable from the root through r it holds that all nodes that are reachable from x through p are also reachable from x through q;
  - *backward* version, denoted $r \rightarrow p \leftrightarrows q$, saying that, for each node x reachable from the root through r it holds that all nodes that are reachable from x through p can reach back to x through q.

Forward circular constraints are special cases of forward lollipop constraints in which r is empty; similarly for backward constraints. Forward circular constraints are also called *inclusion path constraints*, and backward lollipop constraints are sometimes referred to as *inverse path constraints* [1,24]. Whenever S is a singleton $\{a\}$, the key constraint $\tau[\{a\}] \rightarrow \tau$ corresponds to the functional constraint $\tau.a \rightarrow \tau$.

Constraints (1) and (2) in our example above are forward circular constraints: (1) is a forward circular constraint in which $p =$ `*.publication.written_by` and $q =$ `*.author`, and (2) is a forward circular constraint with $p =$ `*.author.has_written` and $q =$ `*.publication`. Constraints (3) and (4) above are backward lollipop constraints: (3) is a backward lollipop constraint with $r =$ `*.publication`, $p =$ `written_by` and $q =$ `has_written`, and (4) is a backward lollipop constraint with $r =$ `*.author`, $p =$ `has_written` and $q =$ `written_by`. The key constraint `publication`[`code`] $\rightarrow$ `publication` states that the attribute code of the element publication is a key for the element publication. Finally, `publication.code` $\rightarrow$ `publication.written_by.name` is an example of a functional path constraint asking that the code of the element publication determines the set of authors of the publication.

To show how such constraints can be expressed in hybrid dynamic languages, we define a constraint language $\mathcal{L}_{\text{con}}$ containing any Boolean combination of constraints as introduced above. Formally, a constraint c in $\mathcal{L}_{\text{con}}$ has the following syntax:

- $c = atom \mid c \wedge c \mid c \vee c \mid \neg c$
- $atom = \tau[S] \rightarrow \tau \mid \tau.p \rightarrow \tau.q \mid p \rightrightarrows q \mid p \leftrightarrows q \mid r \rightarrow p \rightrightarrows q \mid r \rightarrow p \leftrightarrows q$,

where τ is an attribute name, S is a set of attribute names, and r, p, q are regular expressions on attribute names.

Next we show how to express the constraints in $\mathcal{L}_{\text{con}}$ within hybrid dynamic logic. We define a mapping σ from the constraint language $\mathcal{L}_{\text{con}}$ to $\text{HDL}(@, \downarrow, \Downarrow)$. Let $root$ be a nominal for the root of the data graph. The easiest constraints to encode are the circular ones. Their encodings do not require hybrid binders:

$$\sigma(p \rightrightarrows q) = @_{root}[p]\langle q\rangle^{-1} root$$
$$\sigma(p \leftrightarrows q) = @_{root}[p]\langle q\rangle root$$

We encode lollipop constraints. Their encodings require only one nesting of the hybrid binder $\downarrow$:

$$\sigma(r \rightarrow p \rightrightarrows q) = @_{root}[r]{\downarrow}x.[p]\langle q\rangle^{-1} x$$
$$\sigma(r \rightarrow p \leftrightarrows q) = @_{root}[r]{\downarrow}x.[p]\langle q\rangle x$$

Key and functional constraints are more involved, since they involve both the comparison of node identities and the comparison of node values. The translation of the key constraint $\tau[S] \rightarrow \tau$ states that, for all different τ nodes x and y, if x and y agree on the values of nodes reachable through attributes in S, then x and y are the same node:

$$\sigma(\tau[S] \rightarrow \tau) = \mathbf{A}{\downarrow}x.\mathbf{A}{\downarrow}y.(@_x\langle\tau\rangle^{-1}\top \wedge @_y\langle\tau\rangle^{-1} \wedge x \neq y) \rightarrow$$

$$(\bigwedge_{a\in S} @_x[a]\Downarrow v_1.@_y\langle a\rangle\Downarrow w_1.v_1 = w_1 \ \wedge \\ @_y[a]\Downarrow w_2.@_x\langle a\rangle\Downarrow v_2.v_2 = w_2) \ \rightarrow \ x = y$$

The translation of the path functional constraints $\tau.p \ \rightarrow \ \tau.q$ states that, for all different τ nodes x and y, if x and y agree on the values of nodes reachable through p, they should also agree on the values of nodes reachable through q:

$$\begin{aligned}\sigma(\tau.p \ \rightarrow \ \tau.q) = \mathbf{A}\downarrow x.\mathbf{A}\downarrow y.(@_x\langle\tau\rangle^{-1}\top \ \wedge \ @_y\langle\tau\rangle^{-1} \ \wedge \ x \neq y) \ \rightarrow \\ (@_x[p]\Downarrow v_1.@_y\langle p\rangle\Downarrow w_1.v_1 = w_1 \ \wedge \\ @_y[p]\Downarrow w_2.@_x\langle p\rangle\Downarrow v_2.v_2 = w_2) \ \rightarrow \\ (@_x[q]\Downarrow v_1.@_y\langle q\rangle\Downarrow w_1.v_1 = w_1 \ \wedge \\ @_y[q]\Downarrow w_2.@_x\langle q\rangle\Downarrow v_2.v_2 = w_2)\end{aligned}$$

We conclude the description of our translation by stipulating that σ distributes over the Boolean operators. In σ it is convenient to replace the universal modality $\mathbf{A}$ with $@_{root}[*]$. Notice that in this way each translated constraint is a formula starting with $@_{root}$. Given a semistructured database D and an integrity constraint $c \in \mathcal{L}_{\rm con}$, it is possible to check whether D satisfies c as follows. The database D is represented as a rooted graph G_D and the constraint c is translated into the hybrid formula $\sigma(c)$. Then, the formula $\sigma(c)$ is checked on G_D by using a hybrid model checker (in fact, it is sufficient to check the formula at the root of the graph). If the outcome of the model checker is the empty set of nodes, then D does not satisfy c, otherwise it does.

Proposition 5.7 *Let c be an integrity constraint in $\mathcal{L}_{\rm con}$ and D be a semistructured database. Then, D satisfies c if, and only if, the truth set of $\sigma(c)$ with respect to G_D is non-empty.*

The translation σ embeds (any Boolean combination of) circular constraints into $\mathrm{HDL}(@)$, lollipop constraints into the fragment of $\mathrm{HDL}(@,\downarrow)$ in which $\downarrow$ is nested only once, and key and functional constraints into the fragment of $\mathrm{HDL}(@,\downarrow,\Downarrow,\mathbf{A})$ in which the hybrid binders are nested a fixed number of times. As a consequence, by virtue of Theorem 4.6, we have the following.

Corollary 5.8

- *The constraint evaluation problem for circular constraints can be solved in linear time both in the length of the constraint (expression complexity) and in the size of the database (data complexity);*
- *The constraint evaluation problem for lollipop constraints can be solved in linear time in the length of the constraint (expression complexity) and in quadratic time in the size of the database (data complexity);*
- *The constraint evaluation problem for key and functional constraints can be solved in linear time in the length of the constraint (expression complexity) and*

in polynomial time in the size of the database (data complexity).

6 Conclusion and Work for the Future

We investigated the model checking problem for hybrid logics. We gave model checkers for a large number of fragments of hybrid and hybrid dynamic logic. We obtained lower bounds on the computational complexity of the model checking problem for hybrid logics with binders. We found that the addition of nominals and the @ operator does not increase the complexity of the model checking task. In contrast, whenever hybrid binders are present in the language, the running time of the resulting model checker is exponential in the nesting level of the binders. We cannot do better, since we proved that the model checking problem for hybrid logics with binders is PSPACE-complete.

We applied our findings to the problems of query and constraint evaluation for semistructured data. We identified significant fragments of well-known query and constraint languages for semistructured data that can be efficiently embedded into hybrid languages. These embeddings allowed us to solve query and constraint evaluation problems via model checking for hybrid logics.

An implementation of the model checkers `MCLITE` and `MCFULL` proposed in this paper is available at `http://www.luigidragone.com/hlmc`. The code is written in C available under the GNU General Public License. It can be freely used, modified and distributed in conformity with this license.

Acknowledgements

We are very grateful to Carlos Areces, Daniel Gorín, Maarten Marx and the anonymous referees for helpful comments and suggestions.

References

[1] S. Abiteboul, P. Buneman, and D. Suciu. *Data on the Web*. Morgan Kaufmann Publishers, Los Altos, CA 94022, USA, 2000.

[2] S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases*. Addison-Wesley, Reading, Mass., 1995.

[3] S. Abiteboul, D. Quass, J. McHugh, J. Widom, J. Wiener, and J. Widom. The Lorel Query Language for Semistructured Data. *International Journal on Digital Libraries*, 1(1):68–88, 1997.

[4] S. Abiteboul and V. Vianu. Regular path queries with constraints. *Journal of Computer and System Sciences*, 58(3):428–452, 1999.

[5] L. Afanasiev, M. Franceschet, M. Marx, and M. de Rijke. CTL Model Checking for Processing Simple XPath Queries. In *Proceedings TIME 2004*, pages 117–124, 2004.

[6] N. Alechina and M. de Rijke. Describing and querying semistructured data: Some expressiveness results. In S.M. Embury, N.J. Fiddian, W.A. Gray, and A.C. Jones, editors, *Advances in Databases*, LNCS. Springer, 1998.

[7] N. Alechina, S. Demri, and M. de Rijke. A modal perspective on path constraints. *Journal of Logic and Computation*, 13(6):939–956, 2003.

[8] L. De Alfaro. Model checking the World Wide Web. In *Proceedings CAV 2001*, volume 2102 of *LNCS*, pages 337–349, 2001.

[9] C. Areces, P. Blackburn, and M. Marx. A road-map on complexity for hybrid logics. In J. Flum and M. Rodriguez-Artalejo, editors, *Computer Science Logic*, volume 1683 of *LNCS*, pages 307–321. Springer, 1999.

[10] C. Areces, P. Blackburn, and M. Marx. The computational complexity of hybrid temporal logics. *Logic Journal of the IGPL*, 8(5):653–679, 2000.

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

[12] C. Areces and J. Heguiabehere. HyLoRes 1.0: Direct resolution for hybrid logics. In A. Voronkov, editor, *Automated Deduction – CADE-18*, volume 2392 of *LNCS*, pages 156–160. Springer-Verlag, July 27-30 2002.

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

[14] P. Blackburn and J. Seligman. Hybrid languages. *Journal of Logic, Language and Information*, 4:251–272, 1995.

[15] P. Blackburn and J. Seligman. What are hybrid languages? In M. Kracht, M. de Rijke, H. Wansing, and M. Zakharyaschev, editors, *Advances in Modal Logic, Volume 1*, pages 41–62. CSLI Publications, 1998.

[16] R. Bull. An approach to tense logic. *Theoria*, 36:282–300, 1970.

[17] P. Buneman, W. Fan, and S. Weinstein. Path constraints on semistructured and structured data. In *Proceedings PODS*, pages 129–138, 1998.

[18] D. Calvanese, G. De Giacomo, and M. Lenzerini. Representing and reasoning on XML documents: A description logic approach. *Journal of Logic and Computation*, 9(3):295–318, 1999.

[19] E. M. Clarke, O. Grumberg, and D. A. Peled. *Model Checking*. The MIT Press, Cambridge, Massachusetts, 1999.

[20] E. M. Clarke and H. Schlingloff. Model checking. In A. Robinson and A. Voronkov, editors, *Handbook of Automated Reasoning*, volume II, chapter 24, pages 1635–1790. Elsevier Science, 2001.

[21] R. Cleaveland and B. U. Steffen. A linear-time model checking algorithm for the alternation-free modal mu-calculus. *Formal Methods in System Design*, 2:121–147, 1993.

[22] World Wide Web Consortium. Extensible markup language (XML). Available at http://www.w3.org/XML, 1998.

[23] E.A. Emerson. Temporal and modal logic. In J. Van Leeuwen, editor, *Handbook of Theoretical Computer Science, Vol. B*, pages 995–1072. Elsevier Science Publishers B.V., 1990.

[24] W. Fan and J. Siméon. Integrity constraints for XML. *Journal of Computer and System Sciences*, 66(1):254–291, 2003.

[25] M. Franceschet, M. de Rijke, and B. H. Schlingloff. Hybrid logics on linear structures: expressivity and complexity. In *Proceedings TIME 2003*, pages 166–173. IEEE Computer Society Press, 2003.

[26] V. Goranko. Hierarchies of modal and temporal logics with reference pointers. *Journal of Logic, Language and Information*, 5(1):1–24, 1996.

[27] G. Gottlob and C. Koch. Monadic Queries over Tree-Structured Data. In *Logic in Computer Science*, pages 189–202, Los Alamitos, CA, USA, July 22–25 2002. IEEE Computer Society.

[28] J. Y. Halpern, R. Harper, N. Immerman, P. G. Kolaitis, M. Vardi, and V. Vianu. On the unusual effectiveness of logic in computer science. *The Bulletin of Symbolic Logic*, 7(2):213–236, 2001.

[29] D. Harel, J. Tiuryn, and D. Kozen. *Dynamic Logic*. MIT Press, 2000.

[30] HyLo: The Hybrid Logic home page. URL: `http://hylo.loria.fr`.

[31] M. Marx. Conditional XPath, the first order complete XPath dialect. In *Proceedings PODS*, pages 13–22, 2004.

[32] J. McHugh, S. Abiteboul, R. Goldman, D. Quass, and J. Widom. Lore: A database management system for semistructured data. *SIGMOD Record (ACM Special Interest Group on Management of Data)*, 26(3):54–66, 1997.

[33] G. Miklau and D. Suciu. Containment and equivalence for an XPath fragment. In *Proceedings PODS*, pages 65–76, 2002.

[34] C. Papadimitriou. *Computational Complexity*. Addison-Wesley, 1994.

[35] S. Passy and T. Tinchev. An essay in combinatory dynamic logic. *Information and Computation*, 93(2):263–332, 1991.

[36] A. Prior. *Past, Present and Future*. Clarendon, Oxford, 1967.

[37] E. Quintarelli. *Model-Checking Based Data Retrieval: an application to semistructured and temporal data*. PhD thesis, Dipartimento di Elettronica e Informazione, Politecnico di Milano, 2000.

[38] B. ten Cate. *Model theory for extended modal languages*. PhD thesis, University of Amsterdam, 2005. ILLC Dissertation Series DS-2005-01.

A General Tableau Method for Propositional Interval Temporal Logics: Theory and Implementation

V. Goranko [a] A. Montanari [b] P. Sala [b] G. Sciavicco [b]

[a] *Department of Mathematics, University of Johannesburg, South Africa,*
E-mail: vfg@rau.ac.za

[b] *Department of Mathematics and Computer Science, University of Udine, Italy,*
E-mail: {montana|sala|scia vicc}@dimi.uniud.it

Abstract

In this paper, we focus our attention on tableau methods for propositional interval temporal logics. These logics provide a natural framework for representing and reasoning about temporal properties in several areas of computer science. However, while various tableau methods have been developed for linear and branching time point-based temporal logics, not much work has been done on tableau methods for interval-based ones. We develop a general tableau method for Venema's CDT logic interpreted over partial orders (BCDT$^+$ for short). It combines features of the classical tableau method for first-order logic with those of explicit tableau methods for modal logics with constraint label management, and it can be easily tailored to most propositional interval temporal logics proposed in the literature. We prove its soundness and completeness, and we show how it has been implemented.

Key words: Interval Temporal Logics, Proof Systems, Tableau Methods

1 Introduction

In this paper, we focus our attention on tableau methods for propositional interval temporal logics. These logics provide a natural framework for representing and reasoning about temporal properties in several areas of computer science. However, while various tableau methods have been developed for linear and branching time point-based temporal logics, e.g., [5,9,18,29,33], not much work has been done on tableau methods for interval-based temporal logics. One reason for this disparity is

[1] This paper is an extended and revised version of [12].

that operators of interval temporal logics are in many respects more difficult to deal with. As an example, there exist straightforward inductive definitions of the main operators of point-based temporal logics, such as the *future* and the *until* operators, while inductive definitions of basic interval modalities turn out to be much more complex (consider, for instance, the one for the *chop* operator given in [4]).

Various propositional and first-order interval temporal logics have been proposed in the literature (see [13] for an up-to-date survey that analyzes the main contributions in the field). Propositional interval temporal logics include Halpern and Shoham's Modal Logic of Time Intervals (HS) [17], Venema's CDT logic [32], Moszkowski's Propositional Interval Temporal Logic (PITL) [22], and Goranko, Montanari, and Sciavicco's family of Propositional Neighborhood Logics ($\mathcal{PNL}$) [11], while the most interesting first-order versions are Moszkowski's Interval Temporal Logic (ITL) [22] and Zhou and Hansen's Neighborhood Logic (NL) [36]. Two different semantics have been given to interval logics, namely, a *non-strict* one, which includes intervals with coincident endpoints (point-intervals), and a *strict* one, which excludes them. We restrict our attention to the propositional setting, and we assume the non-strict semantics as the default (it is the most common and general).

In this paper, we develop a sound and complete general tableau method for Venema's CDT logic interpreted over partial orders, called (Non-Strict) Branching CDT (BCDT$^+$ for short). BCDT$^+$ features the same operators as CDT; however, since it is interpreted over partially ordered domains with linear intervals, it is expressive enough to include as subsystems or specializations all the above-mentioned propositional interval logics. While most existing tableau methods for modal and temporal logics are terminating methods for *decidable* logics, and thus they yield decision procedures, the proposed tableau method for BCDT$^+$ only provides a semi-decision procedure for unsatisfiability. In this respect, even though it shares some basic features with explicit tableaux for modal logics with constraint label management, it comes closer to the classical, possibly non-terminating tableau method for first-order logic [8]. Furthermore, it presents some similarities with the explicit tableau method developed for the *guarded fragment* of first-order logic [14]. Finally, it can be easily adapted to variations and subsystems of BCDT$^+$, thus providing a general tableau method for propositional interval logics.

The rest of the paper is organized as follows. In Section 2, we introduce the syntax and semantics of BCDT$^+$, and we compare its expressive power with that of the main propositional interval logics. In Section 3, we provide a survey of existing tableau methods for propositional temporal logics. In Section 4, we present our tableau method, and we prove its soundness and completeness. In Section 5, we describe the basic features of an efficient implementation of the method in an imperative language. Conclusions provide an assessment of the work and outline future research directions.

Fig. 1. A non-linear interval structure with/without the linear interval property.

2 CDT **over partial orders (**BCDT$^+$**)**

In this section, we give syntax and semantics of BCDT$^+$ and discuss its expressive power. To this end, we introduce some preliminary notions. Let $\mathbb{D} = \langle \mathrm{D}, < \rangle$ be a strict partial order. A (non-strict) **interval** on $\mathbb{D}$ is an ordered pair $[d_0, d_1]$ such that $d_0, d_1 \in \mathrm{D}$, and $d_0 \leq d_1$. When $d_0 < d_1$ we say that the interval is **proper** or **strict**; when $d_0 = d_1$ it is a **point-interval**. The set of all non-strict intervals on $\mathbb{D}$ will be denoted by $\mathbb{I}(\mathbb{D})^+$, while the set of all strict intervals will be denoted by $\mathbb{I}(\mathbb{D})^-$; by $\mathbb{I}(\mathbb{D})$ we will denote either of these. As in [17], we assume intervals to be *linear*, that is, for every interval $[d_0, d_1]$ and every pair of points d, d' belonging to it, namely, $d_0 \leq d \leq d_1$ and $d_0 \leq d' \leq d_1$, $d < d'$ or $d' < d$ or $d = d'$. Such an assumption keeps the temporal setting still very general, while making it fitting our intuition about the nature of time. In Figure 1 we give an example of a non-linear interval structure with the linear interval property (left) and an example of a non-linear interval structure that does not satisfy it (right). A pair $\langle \mathbb{D}, \mathbb{I}(\mathbb{D}) \rangle$ is called an **interval structure**. We can constrain an interval structure to be **linear**, **branching**, **discrete**, **dense**, **unbounded above** and/or **below**, **Dedekind complete**, and so on, by imposing suitable conditions on it. An element $d \in \mathrm{D}$ such that there are no elements $d' \in \mathrm{D}$ with $d < d'$ (resp., $d' < d$) is called **minimal** (resp., **maximal**) element.

BCDT$^+$ features the same operators as CDT, but it is interpreted over partially ordered domains with linear intervals. Its language consists of a set of propositional variables $\mathcal{AP}$, the logical connectives $\neg$ and $\wedge$, the modalities C, D, and T, and the modal constant π. The other logical connectives, as well as the logical constants $\top$ and $\bot$, can be defined in the usual way. BCDT$^+$ well-formed **formulas**, denoted by $\phi, \psi, \ldots$, are recursively defined as follows (where $p \in \mathcal{AP}$):

$$\phi = \pi \mid p \mid \neg\phi \mid \phi \wedge \psi \mid \phi C \psi \mid \phi D \psi \mid \phi T \psi.$$

The semantics of BCDT$^+$ is given in terms of **non-strict models** of the type $\mathbf{M}^+ = \langle \mathbb{D}, \mathbb{I}(\mathbb{D})^+, \mathcal{V} \rangle$, based on non-strict interval structures, where $\mathcal{V}$ is a **valuation function** for propositional variables. The valuation function is a mapping $\mathcal{V} : \mathbb{I}(\mathbb{D})^+ \mapsto 2^{\mathcal{AP}}$, where $\mathbb{I}(\mathbb{D})^+$ is the set of all intervals in $\mathbb{D}$, such that, for any $p \in \mathcal{AP}$, p holds over $[d_0, d_1]$ if and only if $p \in \mathcal{V}([d_0, d_1])$. **Truth** over an interval $[d_0, d_1]$ in a model

$\mathbf{M}^+$ is defined by induction on the structure of formulas:

- $\mathbf{M}^+, [d_0, d_1] \Vdash \pi$ iff $d_0 = d_1$;
- $\mathbf{M}^+, [d_0, d_1] \Vdash p$ iff $p \in \mathcal{V}([d_0, d_1])$, for all $p \in \mathcal{AP}$;
- $\mathbf{M}^+, [d_0, d_1] \Vdash \neg\psi$ iff it is not the case that $\mathbf{M}^+, [d_0, d_1] \Vdash \psi$;
- $\mathbf{M}^+, [d_0, d_1] \Vdash \phi \wedge \psi$ iff $\mathbf{M}^+, [d_0, d_1] \Vdash \phi$ and $\mathbf{M}^+, [d_0, d_1] \Vdash \psi$;
- $\mathbf{M}^+, [d_0, d_1] \Vdash \phi C \psi$ iff there exists $d_2 \in \mathrm{D}$ such that (i) $d_0 \leq d_2 \leq d_1$, and (ii) $\mathbf{M}^+, [d_0, d_2] \Vdash \phi$ and $\mathbf{M}^+, [d_2, d_1] \Vdash \psi$;
- $\mathbf{M}^+, [d_0, d_1] \Vdash \phi D \psi$ iff there exists $d_2 \in \mathrm{D}$ such that (i) $d_2 \leq d_0$, and (ii) $\mathbf{M}^+, [d_2, d_0] \Vdash \phi$ and $\mathbf{M}^+, [d_2, d_1] \Vdash \psi$;
- $\mathbf{M}^+, [d_0, d_1] \Vdash \phi T \psi$ iff there exists $d_2 \in \mathrm{D}$ such that (i) $d_1 \leq d_2$, and (ii) $\mathbf{M}^+, [d_1, d_2] \Vdash \phi$ and $\mathbf{M}^+, [d_0, d_2] \Vdash \psi$.

Satisfiability and **validity** of BCDT$^+$ formulas are defined in the usual way.

Let us compare the expressive power of BCDT$^+$ with that of the main propositional interval logics proposed in the literature. We say that a logic $\mathbf{L_1}$ is **at least as expressive** as a logic $\mathbf{L_2}$ if for every $\mathbf{L_2}$ formula there exists an equivalent $\mathbf{L_1}$ formula, and that $\mathbf{L_1}$ is (strictly) **more expressive** than $\mathbf{L_2}$ if and only if $\mathbf{L_1}$ is at least as expressive as $\mathbf{L_2}$, but not vice versa.

We preliminarily summarize the main characteristics of the considered interval temporal logics. HS features four basic operators: $\langle B \rangle$ (*begins*) and $\langle E \rangle$ (*ends*), and their transposes $\langle \overline{B} \rangle$ (*begun by*) and $\langle \overline{E} \rangle$ (*ended by*). Given a formula ϕ and an interval $[d_0, d_1]$, $\langle B \rangle \phi$ holds over $[d_0, d_1]$ if ϕ holds over $[d_0, d_2]$, for some $d_0 \leq d_2 < d_1$, and $\langle E \rangle \phi$ holds over $[d_0, d_1]$ if ϕ holds over $[d_2, d_1]$, for some $d_0 < d_2 \leq d_1$. It is possible to show that HS captures all Allen's relations [2]. In particular, it allows one to define the *strict after* operator $\langle A \rangle$ (and its transpose $\langle \overline{A} \rangle$) such that $\langle A \rangle \phi$ holds over $[d_0, d_1]$ if ϕ holds over $[d_1, d_2]$ for some $d_2 > d_1$, the *non-strict after* operator $\Diamond_r$ (and its transpose $\Diamond_l$) such that $\Diamond_r \phi$ holds over $[d_0, d_1]$ if ϕ holds over $[d_1, d_2]$ for some $d_2 \geq d_1$, and the *sub-interval* operator $\langle D \rangle$ such that $\langle D \rangle \phi$ holds over a given interval $[d_0, d_1]$ if ϕ holds over a proper sub-interval of $[d_0, d_1]$. In [17], Halpern and Shoham have shown the undecidability of HS over various classes of linear orders by a suitable encoding of the halting problem.
CDT has three binary operators C (*chop*), D, and T, which correspond to the ternary interval relations occurring when an extra point is added in one of the three possible distinct positions with respect to the two endpoints of the current interval (*between*, *before*, and *after*), plus a modal constant π which holds over a given interval if and only if it is a point-interval. Since HS can be embedded into CDT, undecidability results for the latter follow from those for the former.
PITL features the two modalities $\bigcirc$ (*next*) and C (the specialization of the *chop* operator for discrete structures). Intervals are defined as finite or infinite sequences of states. Given two formulas ϕ, ψ and a (finite) interval $d_0, \ldots, d_n$, $\bigcirc \phi$ holds over $d_0, \ldots, d_n$ if and only if ϕ holds over $d_1, \ldots, d_n$, while $\phi C \psi$ holds over $d_0, \ldots, d_n$ if and only if there exists i, with $0 \leq i \leq n$, such that ϕ holds over $d_0, \ldots, d_i$ and

ψ holds over $d_i, \ldots, d_n$. PITL has been proved to be undecidable by a reduction from the problem of testing the emptiness of the intersection of two grammars in Greibach form [22]. A decidable fragment of PITL extended with quantification over propositional variables (QPITL) has been obtained by imposing a suitable *locality* constraint which states that each propositional variable is true over an interval if and only if it is true at its first state [22]. By exploiting such a constraint, decidability of Local QPITL can be easily proved by embedding it into quantified propositional Linear Temporal Logic.
Finally, propositional neighborhood logics in $\mathcal{PNL}$ have two modalities for right and left interval neighborhoods, namely, $\langle A \rangle$ and $\langle \overline{A} \rangle$ in the strict semantics ($\mathcal{PNL}^-$ logics), and $\Diamond_r$ and $\Diamond_l$ in the non-strict semantics ($\mathcal{PNL}^+$ logics). While the undecidability of the first-order Neighborhood Logic NL can be easily proved by embedding HS in it, the decidability problem for its propositional fragments is still open.

We first note that both CDT and non-strict Propositional Neighborhood Logics ($\mathcal{PNL}^+$) are interpreted over linear structures, and that the operators of $\mathcal{PNL}^+$ logics can be expressed in CDT by means of the formulas $\Diamond_r \phi := \phi T \top$ and $\Diamond_l \phi := \phi D \top$. Moreover, it is well known that CDT does not semantically include HS in its full generality, since the latter allows the interval structure to be branching, while the former does not. On the other hand, HS is not more expressive than CDT, because it cannot express the *chop* operator [21].
BCDT$^+$ generalizes Venema's CDT (and thus all logics in $\mathcal{PNL}^+$) by allowing the interval structure to be non-linear, for as long as all intervals in it are linear (as in HS). Furthermore, it is strictly more expressive than HS and PITL. HS operators can be defined in BCDT$^+$ as follows: $\langle B \rangle \phi := \phi C \neg \pi$, $\langle \overline{B} \rangle \phi := \neg \pi T \phi$, $\langle E \rangle \phi := \neg \pi C \phi$, and $\langle \overline{E} \rangle \phi := \neg \pi D \phi$. Besides, the strict neighborhood operators $\langle A \rangle$ and $\langle \overline{A} \rangle$ can be defined in BCDT$^+$ by using π as follows: $\langle A \rangle \phi := (\phi \wedge \neg \pi) T \top$, and $\langle \overline{A} \rangle \phi := (\phi \wedge \neg \pi) D \top$. By exploiting such derived operators, all conditions on the interval structure mentioned in the preliminaries can be easily expressed in BCDT$^+$. In particular, linearity can be expressed in BCDT$^+$ by means of the following formula:

$$lin \triangleq (\langle A \rangle p \rightarrow [A](p \vee \langle B \rangle p \vee \langle \overline{B} \rangle p)) \wedge (\langle \overline{A} \rangle p \rightarrow [\overline{A}](p \vee \langle E \rangle p \vee \langle \overline{E} \rangle p)),$$

while discreteness of linear interval structures can be imposed by means of the formula:

$$disc \triangleq \pi \vee l1 \vee (\langle B \rangle l1 \wedge \langle E \rangle l1),$$

where $l1$ stands for $\langle B \rangle \top \wedge [B][B] \bot$.
As for the PITL operators, C is an operator of BCDT$^+$, while $\bigcirc$ can be defined over (linear) discrete structures as follows: $\bigcirc \phi := l1 C \phi$. On the other hand, BCDT$^+$ is strictly more expressive than PITL, since the latter is not able to access any interval which is not a sub-interval of the current interval.
The undecidability of BCDT$^+$ with respect to a number of interval structures immediately follows from results in [17], while finding meaningful decidable fragments

of BCDT$^+$ is an interesting open problem.

3 Tableau methods for temporal logics

In this section, we survey existing tableau methods for propositional point-based and interval-based temporal logics over linear and branching time. According to [8], tableau methods for (modal and) temporal logics can be classified as *explicit* or *implicit*. Explicit methods keep track of the accessibility relation by means of some sort of external device. One possibility is to maintain an auxiliary graph of named nodes $\mathbf{n_i}, \mathbf{n_j}, \ldots$, where each node contains a subformula, or a set of subformulas, of the formula to be checked, and the existence of an edge from $\mathbf{n_i}$ to $\mathbf{n_j}$ means that $\mathbf{n_j}$ is accessible from $\mathbf{n_i}$. Another possibility is to include structured labels into nodes to constrain the formula, or the set of formulas, associated with each node to hold only at the domain element(s) identified by the label. The resulting labeled tableau systems capture the accessibility relation by means of labeled formulas, and they provide suitable notions of closed branches and tableaux. In implicit methods [10,27], the accessibility relation is built-in into the structure of the tableau. As an example, in the case of linear and branching time point-based temporal logics the tableau represents a model of the satisfiable formulas (a timeline or a tree, respectively). The non-standard finite model property can then be exploited to show that the resulting tableau methods are actually decision procedures (they do not lead to infinite computations). In [18], implicit methods are further partitioned into *declarative* and *incremental* ones. Methods in the former class first generate all possible sets of subformulas of a given formula, and then they eliminate some (possibly all) of them, while those in the latter generate only 'meaningful' sets of subformulas.

3.1 Point-based linear and branching temporal logics

The problem of devising tableau systems for propositional Linear Temporal Logic (LTL), as well as for some extensions and fragments of it, has been extensively investigated in the literature. An exponential time declarative method to check LTL formulas has been developed by Wolper [33] and later extended by Lichtenstein and Pnueli to Past LTL (PLTL) [26], while an incremental method for PLTL has been proposed by Kesten et al. [18]. A labeled tableau system for the LTL-fragment LTL[F] has been proposed by Schmitt and Goubault-Larrecq [29] (an attempt to extend it to full LTL is reported in [30]). Finally, a tableau method for PLTL over bounded models has been developed by Cerrito and Cialdea-Mayer [5] (in [6], Cerrito et al. generalize the method to first-order PLTL). The satisfiability problem for LTL and PLTL is PSPACE-complete [31], while that for PLTL over bounded models of polynomial length and LTL[F] is NP-complete [5,31].

Wolper's tableau method is a natural extension to the one for propositional logic. The key idea is to take advantage of the so-called fix-point definition of temporal operators that allows one to split every temporal formula into a (possibly empty) part related to the current state and a part related to the next (resp. previous) state. As an example, the formula $\phi U \psi$ is analyzed as follows: either ψ holds now, or ϕ holds now and $\phi U \psi$ holds at the next state. Since only a finite set of distinct scenarios can be generated in this way, it is possible to devise a mechanism to control the repeated appearances of formulas, and to identify periodic situations in a finite time. The algorithm for checking the (un)satisfiability of a PLTL-formula ϕ behaves as follows: first (*construction*), it builds the tableau for ϕ; then (*elimination*), it removes unsuitable maximal strongly connected components (maximal strongly connected components which are not reachable from an initial node including ϕ or are not self fulfilling and have not outgoing edges [33]). It turns out that the formula ϕ is satisfiable if and only if the elimination phase does not end with an empty tableau. In [18] Kesten et al. provide an efficient incremental variant of Wolper's declarative procedure, which extends to PLTL the incremental method for LTL originally developed by Pnueli and Sherman [25]. Its basic ingredients are the same as Wolper's. However, instead of preliminarily generating the set of all nodes of the tableau and thus immediately paying the worst case exponential complexity price, it builds the tableau incrementally by introducing only those nodes which are reachable from an initial nodes including the formula to be checked. Even though in the worst case this procedure takes exponential time, one can expect that in many cases a much smaller number of nodes is explored.

Unlike the above-described implicit methods, the labeled tableau systems for LTL[F] and for PLTL over bounded models, respectively developed by Schmitt and Goubault-Larrecq [29] and by Cerrito and Cialdea-Mayer [5], employ a mechanism for labelling formulas with temporal constraints somewhat similar to ours.

The distinctive feature of Schmitt and Goubault-Larrecq's tableau system is that its termination can be established *locally*: it terminates when no further expansion rule can be applied, which is guaranteed to happen. The basic notions are those of signed clause, borrowed from [16], and temporal constraint. A *signed clause* is either a pair $[d_i, d_j]\Theta$ or a pair $|\infty|\Theta$, where Θ is a (multi)set of formulas (the *clause*) and $[d_i, d_j]$ is a *time interval*, where d_j can possibly be ∞. A signed clause $[d_i, d_j]\Theta$ evaluates to true in a given structure if for every $d \in [d_i, d_j]$, there exists $\psi \in \Theta$ such that ψ holds at d (disjunctive interpretation), while $|\infty|\Theta$ is true if for infinitely many time points d, every $\psi \in \Theta$ holds at d (conjunctive interpretation). A temporal constraint is an expression of the form $d_i \leq d_j$. A tableau T is a set of branches, where a *branch* is a pair (B, K) consisting of a set of signed clauses B and a set of temporal constraints K. The construction of the tableau is accomplished by applying two kinds of steps: *expansion* and *closing* steps. An expansion step is performed by choosing a branch (B, K) and an unused signed clause $\Theta \in B$, and by applying a matching logical tableau *rule*. As a result of this application, the premise clause is marked as used and the branch (B, K) is extended by adding

the signed clauses and the constraints, possibly involving new states, in the conclusions of the rule to B and K, respectively. Whenever the conclusions include various alternatives, the branch is split accordingly. A signed clause such that Θ includes only literals is called *atomic*. A branch whose unused signed clauses are all atomic is called atomic, and it cannot be further expanded. An atomic branch (B, K) is *satisfiable* if there exists an interpretation that satisfies all signed clauses in B and the set of constraints in K; otherwise, it is unsatisfiable. A closing step is applied to an atomic branch, and it marks the branch as *closed* if it is unsatisfiable. Termination is proved by introducing a suitable complexity measure and by showing that, for every rule, every conclusion is smaller than the premise with respect to such a measure. Critical formulas, such as $GF\phi$, which potentially lead to infinite computations, are dealt with by using the symbol ∞ in a suitable way.
PLTL interpreted over bounded models, that is, finite sequences $d_0, d_1, \ldots, d_k$ of states where k is known in advance, has been introduced to address planning problems in AI. A tableau method for it has been developed by Cerrito and Cialdea-Mayer [5]. The boundaries of the model are encoded by means of the special constant symbols $start$ and $finish$. A state d_i is encoded by means of an expression of the form $c+n$, where c is a constant and n is a natural number. A tableau is a set of branches to which *expansion* and *conflict resolution* rules are applied. Tableau nodes are either temporal constraints or labeled formulas. *Temporal constraints* are expressions of the form $d_i \leq d_j$, where d_i, d_j are states. An *initial constraint* $finish \leq start + k$, where k is the length of the model, is associated with every tableau. A *labeled formula* is a pair $([d_i, d_j], \psi)$, where $[d_i, d_j]$ is an interval and ψ is a PLTL-formula, which states that ψ is true at every state between d_i and d_j (conjunctive interpretation). Expansion rules are applied to pairs of nodes of the forms $([d_i, d_j], \psi)$ and $d_i \leq d_j$, and they cause the expansion, and possibly the splitting, of the branch. Conflict resolution rules force the two intervals over which contradictory literals hold (if any) to be disjoint. The closure of a branch B is established by checking the set K of constraints associated with it (which includes the initial constraint): B is closed if and only if K is unsatisfiable. Termination, soundness, and completeness of the method are proved by exploiting a suitable notion of *canonical tableau*.
The main differences between these tableau methods and ours are: (i) they are specifically designed to deal with natural/integer time structures (i.e., linear and discrete), while ours makes no assumptions; (ii) intervals only play a secondary role in them (e.g., in Cerrito and Cialdea-Mayer's system a formula is true on an interval if and only if it is true at every point in it), while in our system intervals are primary semantic objects on which the truth definitions are entirely based; (iii) the closedness of the tableau is defined in terms of unsatisfiability of the associated set of temporal constraints, while in our system it is entirely syntactic.

We conclude this section by considering the satisfiability problem for CTL, which is known to be EXPTIME-complete. An implicit tableau method to check the satisfiability of CTL formulas, that generalizes Wolper's method for LTL, has been pro-

posed by Emerson and Halpern in [9]. The algorithm for checking the (un)satisfiability of a CTL-formula ϕ basically behaves as Wolper's one: first, it builds the tableau for ϕ; then, it removes unsuitable maximal strongly connected components. The elimination phase encompasses both a local pruning process, that removes local inconsistencies, and another pruning process, that removes nodes including requests which are not fulfilled in the current tableau. As in the case of LTL, the formula ϕ is satisfiable if and only if the elimination phase does not end with an empty tableau.

3.2 Interval-based linear temporal logics and duration calculi

To the best of our knowledge, there exist very few tableau methods for interval temporal logics (and duration calculi) in the literature. A tableau-based decision procedure for an extension of Local PITL interpreted over finite state sequences (LPITL$_{proj}$), which pairs the operators $\bigcirc$ and C with a projection operator $proj$, has been proposed by Bowman and Thompson [4]. Such a procedure refines a previous tableau system for quantified LPITL$_{proj}$ developed by Kono [19]. It rests on a normal form for LPITL$_{proj}$ formulas that allows one to exploit a classical tableau method, devoid of any mechanism for constraint label management. In [7], Chetcuti-Serandio and Fariñas del Cerro isolate a fragment of Propositional Duration Calculus (PDC$_{pos}$), which only includes PDC formulas that satisfy suitable syntactic restrictions. PDC$_{pos}$ is expressive enough to capture Allen's relations [2] and decidable. The tableau construction for PDC$_{pos}$ combines the application of the rules of classical tableaux with that of a suitable constraint resolution algorithm and it essentially depends on the assumption of *bounded* variability of state expressions (they may have only a finite number of discontinuities on a bounded interval, thus being Riemann-integrable on all bounded intervals).
LPITL$_{proj}$ extends LPITL with the binary operator $proj$ which yields general repetitive behavior. For any given pair of formulas ϕ and ψ, $\phi\ proj\ \psi$ holds over an interval if such an interval can be partitioned into a series of sub-intervals each of which satisfies ϕ, while ψ (called the *projected formula*) holds over the new interval collecting the endpoints of these sub-intervals. LPITL$_{proj}$ formulas are interpreted over finite state sequences $d_0, d_1, \ldots, d_k$. The valuation function $\mathcal{V}$ maps each interval $[d_i, d_j]$ into the set of propositional variables that hold over it. The locality constraint imposes that, for any propositional variable p and interval $[d_i, d_j]$, $p \in \mathcal{V}([d_i, d_j])$ if and only if $p \in \mathcal{V}([d_i, d_i])$. The problem of satisfiability checking for LPITL$_{proj}$ is non-elementary [13]. The core of Bowman and Thompson's tableau method is the definition of suitable normal forms for all operators of the logic, which reflect the locality constraint and provide the operators with uniform inductive definitions. Taking advantage of them, Bowman and Thompson develop an implicit tableau-based decision procedure for satisfiability checking in the style of Wolper's one [33]. The normal form for LPITL$_{proj}$ formulas has the following

general format:

$$(\pi \wedge \phi_e) \vee \bigvee_i (\phi_i \wedge \bigcirc \phi'_i),$$

where π stands for the formula $\bigcirc\bot$ characterizing point-intervals, ϕ_e and ϕ_i are point formulas, and ϕ'_i is an arbitrary LPITL$_{proj}$ formula. The first disjunct states when a formula is satisfied over a point interval, while the second one states the possible ways in which a formula can be satisfied over a strict interval, namely, a point formula must hold at the initial point and then an arbitrary formula must hold over the remainder of the interval. This normal form embodies a recipe for evaluating LPITL$_{proj}$ formulas: the first disjunct is the base case, while the second disjunct is the inductive step. Bowman and Thompson show that any LPITL$_{proj}$ formula can be equivalently transformed into this normal form. As in the case of implicit methods for point-based temporal logics, the tableau construction splits the requirements imposed by any temporal formula into requirements about the present (the first state of an interval) and requirements about the remainder of the interval, and it generates a directed graph $G = (N, E)$, where each node corresponds to a state of the sequence and is labeled by a set of formulas. The construction of the graph G for a formula ϕ starts with the *initial* node $\mathbf{n_0}$ labeled with the set $\{\phi, \top C \pi\}$. The expansion rules for the Boolean connectives are the standard ones; formulas of the forms $\psi C \theta$ and $\psi \ proj \ \theta$, as well as $\neg(\psi C \theta)$ and $\neg(\psi \ proj \ \theta)$, are expanded by exploiting the normal forms of their subformulas; finally, as in Wolper's tableau method, formulas of the form $\bigcirc\psi$ are expanded into a new node, corresponding to a new state, labeled with ψ. Once the construction of the graph G has been completed, the procedure looks for unsatisfiable nodes in G and marks them. Unsatisfiable nodes are (i) nodes which contain a formula and its negation, (ii) nodes which contain both a formula $\bigcirc\psi$ and the formula π, and (iii) nodes whose successors are unsatisfiable. The formula ϕ is satisfiable if and only if the initial node is not marked. The proofs of termination, soundness, and completeness are similar to those for PLTL in their structure, but they are much more involved.

Chetcuti-Serandio and Fariñas del Cerro's tableau method operates on a decidable fragment of (propositional) Duration Calculus (DC). DC is a first-order interval temporal logic, interpreted over the set of reals, which is based on ITL [36,35]. The first-order language for DC extends the propositional one essentially the same way as in classical logic, but accounting for the fact that the first-order domain may change over time. Among the constants, there is a specific and important one, that is, the constant l, whose interpretation can vary over time, denoting the *length of the current interval*. It is combined with the structure of the additive group of reals as part of the temporal domain, which allows, for instance, to compute the length of concatenated intervals. A specific additional feature of the syntax of DC is the special category of terms called *state expressions* which are used to represent the duration for which a system stays in a particular state. Chetcuti-Serandio and Fariñas del Cerro provide a tableau method for PDC$_{pos}$ that presents many similarities with the one of Cerrito and Cialdea-Mayer. Tableau nodes are conjunctions of labeled formulas, labeled state expressions, and constraints (not all these components are necessarily included in any node). Labeled formulas (resp., state expressions) are

pairs $(\phi, [d_i, d_j])$ (resp., $(\sigma, [d_i, d_j])$), where ϕ (resp., σ) is a formula (resp., state expression) and $[d_i, d_j]$ is an interval. Constraints can be either *qualitative*, e.g., $d_i \leq d_j$, and *quantitative*, e.g., $d_j - d_i = k$ or $d_j - d_i > k$, where k is a constant. The construction of the tableau is fairly standard. It starts with an initial node including the pair $(\phi, [d_0, d_1])$, where ϕ is the formula to be checked and $[d_0, d_1]$ is a generic interval, and it proceeds by applying suitable expansion rules to labeled formulas or labeled state expressions in the leaf node of the considered branch. Closing rules detect contradictory formulas associated with the same interval or inconsistent sets of constraints in a leaf node. The proof of termination basically exploits a lemma showing that each expansion rule can be applied finitely often to any branch, while the soundness and completeness proof takes advantage of a lemma showing that expansion rules preserve (a suitable notion of) satisfiability. Complexity issues are not addressed.

3.3 Miscellany

We conclude the section by mentioning some additional tableau systems that present interesting connections to ours, such as the tableau methods for temporal logics of knowledge and belief, the free-variable tableau methods for modal logics, the tableau methods for first-order temporal logics, and the generic tableau provers, such as Paulson's [24]. Tableau methods for the propositional temporal Logics of knowledge and belief KL_n and BL_n are described in [23,34]. They are implicit methods, like those for PLTL, that introduce a specialized accessibility relation and specific rules for agent management. Free-variable semantic tableaux are a well-established technique for first-order theorem proving. In [3], Beckert and Goré show that they can be exploited to deal with propositional modal logics, providing a compact, efficient, and easily implementable technique. Tableau methods for decidable (monodic) fragments of first-order temporal logic over the natural numbers have been developed by Kontchakov et al. [20]. The decision procedure is obtained by separating the temporal and the first-order components of the formula to be checked and by dealing with the former using tableau methods for LTL, and with the latter using existing procedures for first-order decidable fragments. Finally, Abate and Goré [1] propose a tableau workbench that allows one to easily derive implicit tableau methods for various systems of modal logics by specifying the appropriate expansion rules. Furthermore, it allows one to express side conditions for rule firing and to maintain the history of the applied expansion steps.

4 A Tableau Method for BCDT^+

In this section we devise a tableau method for BCDT^+. The method can be adapted to its strict version BCDT^-, and can be accordingly restricted to CDT, HS, PITL,

and some $\mathcal{PNL}$ logics. We first introduce some basic terminology. A **finite tree** is a finite directed connected graph in which every node, apart from one (the **root**), has exactly one incoming edge. A **successor** of a node $\mathbf{n}$ is a node $\mathbf{n}'$ such that there is an edge from $\mathbf{n}$ to $\mathbf{n}'$. A **leaf** is a node with no successors; a **path** is a sequence of nodes $\mathbf{n_0}, \ldots, \mathbf{n_k}$ such that, for all $i = 0, \ldots, k-1$, $\mathbf{n_{i+1}}$ is a successor of $\mathbf{n_i}$; a **branch** is a path from the root to a leaf. The **height** of a node $\mathbf{n}$ is the maximum length (number of edges) of a path from $\mathbf{n}$ to a leaf. If $\mathbf{n}, \mathbf{n}'$ belong to the same branch and the height of $\mathbf{n}$ is less than (resp. less than or equal to) the height of $\mathbf{n}'$, we write $\mathbf{n} \prec \mathbf{n}'$ (resp. $\mathbf{n} \preceq \mathbf{n}'$).

Definition 1 *If $\mathbb{C} = \langle C, < \rangle$ is a finite partial order, a **labeled formula**, with label in $\mathbb{C}$, is a pair $(\phi, [c_i, c_j])$, where $\phi \in$ BCDT$^+$ and $[c_i, c_j] \in \mathbb{I}(\mathbb{C})^+$. For a node $\mathbf{n}$ in a tree $\mathcal{T}$, the **decoration** $\nu(\mathbf{n})$ is a triple $((\phi, [c_i, c_j]), \mathbb{C}, u_{\mathbf{n}})$, where $(\phi, [c_i, c_j])$ is a labeled formula, with label in $\mathbb{C}$, and $u_{\mathbf{n}}$ is a **local flag function** which associates the values 0 or 1 with every branch B in $\mathcal{T}$ containing $\mathbf{n}$.*

Intuitively, the value 0 for a node $\mathbf{n}$ with respect to a branch B means that $\mathbf{n}$ can be expanded on B. For the sake of simplicity, we will often assume the interval $[c_i, c_j]$ to consist of the elements $c_i < c_{i+1} < \cdots < c_{j-1} < c_j$, and sometimes, with a little abuse of notation, we will write $\mathbb{C} = \{c_i < c_k, c_m < c_j, \ldots\}$.

Definition 2 *A **decorated tree** is a tree in which every node has a decoration $\nu(\mathbf{n})$.*

For every decorated tree, we also use a **global flag function** u acting on pairs *(node, branch through that node)*, and defined as $u(\mathbf{n}, B) \triangleq u_{\mathbf{n}}(B)$. Sometimes, for convenience, we will include in the decoration of the nodes the global flag function instead of the local ones. For any branch B in a decorated tree, we denote by $\mathbb{C}_B$ the (partially) ordered set in the decoration of the leaf of B, and for any node $\mathbf{n}$ in a decorated tree, we denote by $\Phi(\mathbf{n})$ the formula in its decoration. If B is a branch, then $B \cdot \mathbf{n}$ denotes the result of the expansion of B with the node $\mathbf{n}$ (addition of an edge connecting the leaf of B to $\mathbf{n}$). Similarly, $B \cdot \mathbf{n_1} \mid \ldots \mid \mathbf{n_k}$ denotes the result of the expansion of B with k immediate successor nodes $\mathbf{n_1}$, $\ldots$, $\mathbf{n_k}$ (which produces k branches extending B). A tableau for BCDT$^+$ will be defined as a special decorated tree. We note again that $\mathbb{C}$ remains finite throughout the construction of the tableau.

Definition 3 *Given a decorated tree $\mathcal{T}$, a branch B in $\mathcal{T}$, and a node $\mathbf{n} \in B$ such that $\nu(\mathbf{n}) = ((\phi, [c_i, c_j]), \mathbb{C}, u)$, with $u(\mathbf{n}, \mathbf{B}) = \mathbf{0}$, the **branch-expansion rule** for B and $\mathbf{n}$ is defined as follows (in all the considered cases, $u(\mathbf{n}', \mathbf{B}') = \mathbf{0}$ for all new pairs $(\mathbf{n}', \mathbf{B}')$ of nodes and branches):*

- *If $\phi = \neg\neg\psi$, then expand the branch to $B \cdot \mathbf{n_0}$, with $\nu(\mathbf{n_0}) = ((\psi, [c_i, c_j]), \mathbb{C}_B, u)$;*
- *If $\phi = \psi_0 \wedge \psi_1$, then expand the branch to $B \cdot \mathbf{n_0} \cdot \mathbf{n_1}$, with $\nu(\mathbf{n_0}) = ((\psi_0, [c_i, c_j]), \mathbb{C}_B, u)$ and $\nu(\mathbf{n_1}) = ((\psi_1, [c_i, c_j]), \mathbb{C}_B, u)$;*

- *If $\phi = \neg(\psi_0 \wedge \psi_1)$, then expand the branch to $B \cdot \mathbf{n_0}|\mathbf{n_1}$, with $\nu(\mathbf{n_0}) = ((\neg\psi_0, [c_i, c_j]), \mathbb{C}_B, u)$ and $\nu(\mathbf{n_1}) = ((\neg\psi_1, [c_i, c_j]), \mathbb{C}_B, u)$;*
- *If $\phi = \neg(\psi_0 C \psi_1)$ and there exists $c_k \in \mathbb{C}_B$, with $c_k \in [c_i, c_j]$, which has not been used yet to expand the node $\mathbf{n}$ on B, then take the least such c_k and expand the branch to $B \cdot \mathbf{n_0}|\mathbf{n_1}$, with $\nu(\mathbf{n_0}) = ((\neg\psi_0, [c_i, c_k]), \mathbb{C}_B, u)$ and $\nu(\mathbf{n_1}) = ((\neg\psi_1, [c_k, c_j]), \mathbb{C}_B, u)$;*
- *If $\phi = \neg(\psi_0 D \psi_1)$, c is a minimal element of $\mathbb{C}_B$ such that $c \leq c_i$, and there exists $c' \in [c, c_i]$, which has not been used yet to expand the node $\mathbf{n}$ on B, then take the least such c' and expand the branch to $B \cdot \mathbf{n_0}|\mathbf{n_1}$, with $\nu(\mathbf{n_0}) = ((\neg\psi_0, [c', c_i]), \mathbb{C}_B, u)$ and $\nu(\mathbf{n_1}) = ((\neg\psi_1, [c', c_j]), \mathbb{C}_B, u)$;*
- *If $\phi = \neg(\psi_0 T \psi_1)$, c is a maximal element of $\mathbb{C}_B$ such that $c_j \leq c$, and there exists $c' \in [c_j, c]$ which has not been used yet to expand the node $\mathbf{n}$ on B, then take the greatest such c' and expand the branch to $B \cdot \mathbf{n_0}|\mathbf{n_1}$, with $\nu(\mathbf{n_0}) = ((\neg\psi_0, [c_j, c']), \mathbb{C}_B, u)$ and $\nu(\mathbf{n_1}) = ((\neg\psi_1, [c_i, c']), \mathbb{C}_B, u)$;*
- *If $\phi = (\psi_0 C \psi_1)$, then expand the branch to $B \cdot (\mathbf{n_i} \cdot \mathbf{m_i})| \ldots |(\mathbf{n_j} \cdot \mathbf{m_j})|(\mathbf{n'_i} \cdot \mathbf{m'_i})| \ldots |(\mathbf{n'_{j-1}} \cdot \mathbf{m'_{j-1}})$, where:*

(1) for all $c_k \in [c_i, c_j]$, $\nu(\mathbf{n_k}) = ((\psi_0, [c_i, c_k]), \mathbb{C}_B, u)$ and $\nu(\mathbf{m_k}) = ((\psi_1, [c_k, c_j]), \mathbb{C}_B, u)$;

(2) for all $i \leq k \leq j-1$, let $\mathbb{C}_k$ be the partial ordering obtained by inserting a new element c between c_k and c_{k+1} in $[c_i, c_j]$, $\nu(\mathbf{n'_k}) = ((\psi_0, [c_i, c]), \mathbb{C}_k, u)$, and $\nu(\mathbf{m'_k}) = ((\psi_1, [c, c_j]), \mathbb{C}_k, u)$;

- *If $\phi = (\psi_0 D \psi_1)$, then repeatedly expand the current branch, once for each minimal element c (where $[c, c_i] = \{c = c_0 < c_1 < \cdots c_i\}$), by adding the decorated subtree $(\mathbf{n_0} \cdot \mathbf{m_0})| \ldots |(\mathbf{n_i} \cdot \mathbf{m_i})|(\mathbf{n'_1} \cdot \mathbf{m'_1})| \ldots |(\mathbf{n'_i} \cdot \mathbf{m'_i})|(\mathbf{n''_0} \cdot \mathbf{m''_0})| \ldots |(\mathbf{n''_i} \cdot \mathbf{m''_i})$ to its leaf, where:*

(1) for all $c_k \in [c_0, c_i]$, $\nu(\mathbf{n_k}) = ((\psi_0, [c_k, c_i]), \mathbb{C}_B, u)$ and $\nu(\mathbf{m_k}) = ((\psi_1, [c_k, c_j]), \mathbb{C}_B, u)$;

(2) for all $1 \leq k \leq i$, let $\mathbb{C}_k$ be the partial ordering obtained by inserting a new element c' between c_{k-1} and c_k in $[c_0, c_i]$, $\nu(\mathbf{n'_k}) = ((\psi_0, [c', c_i]), \mathbb{C}_k, u)$, and $\nu(\mathbf{m'_k}) = ((\psi_1, [c', c_j]), \mathbb{C}_k, u)$;

(3) for all $0 \leq k \leq i$, let $\mathbb{C}_k$ be the partial ordering obtained by inserting a new element c' in $\mathbb{C}_B$, with $c' < c_k$, which is incomparable with all existing predecessors of c_k, $\nu(\mathbf{n''_k}) = ((\psi_0, [c', c_i]), \mathbb{C}_k, u)$, and $\nu(\mathbf{m''_k}) = ((\psi_1, [c', c_j]), \mathbb{C}_k, u)$;

- *If $\phi = (\psi_0 T \psi_1)$, then repeatedly expand the current branch, once for each maximal element c (where $[c_j, c] = \{c_j < c_{j+1} < \cdots c_n = c\}$), by adding the decorated subtree $(\mathbf{n_j} \cdot \mathbf{m_j})| \ldots |(\mathbf{n_n} \cdot \mathbf{m_n})|(\mathbf{n'_j} \cdot \mathbf{m'_j})| \ldots |(\mathbf{n'_{n-1}} \cdot \mathbf{m'_{n-1}})|(\mathbf{n''_j} \cdot \mathbf{m''_j})| \ldots |(\mathbf{n''_n} \cdot \mathbf{m''_n})$ to its leaf, where:*

(1) for all $c_k \in [c_j, c]$, $\nu(\mathbf{n_k}) = ((\psi_0, [c_j, c_k]), \mathbb{C}_B, u)$ and $\nu(\mathbf{m_k}) = ((\psi_1, [c_i, c_k]), \mathbb{C}_B, u)$;

(2) for all $j \leq k \leq n-1$, let $\mathbb{C}_k$ be the partial ordering obtained by inserting a new element c' between c_k and c_{k+1} in $[c_j, c_n]$, $\nu(\mathbf{n}'_{\mathbf{k}}) = ((\psi_0, [c_j, c']), \mathbb{C}_k, u)$, and $\nu(\mathbf{m}'_{\mathbf{k}}) = ((\psi_1, [c_i, c']), \mathbb{C}_k, u)$;

(3) for all $j \leq k \leq n$, let $\mathbb{C}_k$ be the partial ordering obtained by inserting a new element c' in $\mathbb{C}_B$, with $c_k < c'$, which is incomparable with all existing successors of c_k, $\nu(\mathbf{n}''_{\mathbf{k}}) = ((\psi_0, [c_j, c']), \mathbb{C}_k, u)$, and $\nu(\mathbf{m}''_{\mathbf{k}}) = ((\psi_1, [c_i, c']), \mathbb{C}_k, u)$.

Finally, for any node $\mathbf{m}$ ($\neq \mathbf{n}$) in B and any branch B' extending B, let $u(\mathbf{m}, B') = u(\mathbf{m}, B)$, and for any branch B' extending B, $u(\mathbf{n}, B') = 1$, unless $\phi = \neg(\psi_0 C \psi_1)$, $\phi = \neg(\psi_0 D \psi_1)$, or $\phi = \neg(\psi_0 T \psi_1)$ (in such cases $u(\mathbf{n}, B') = 0$).

Let us briefly explain the expansion rules for $\psi_0 C \psi_1$ and $\neg(\psi_0 C \psi_1)$ (similar considerations hold for the other temporal operators). The rule for the existential formula $\psi_0 C \psi_1$ deals with the two possible cases: either there exists $c_k \in \mathbb{C}_B$ such that $c_i \leq c_k \leq c_j$, ψ_0 holds over $[c_i, c_k]$, and ψ_1 holds over $[c_k, c_j]$ ($j - i + 1$ cases) or such an element c_k must be added to $\mathbb{C}_B$ ($j - i$ cases). The universal formula $\neg(\psi_0 C \psi_1)$ states that, for all $c_i \leq c_k \leq c_j$, ψ_0 does not hold over $[c_j, c_k]$ or ψ_1 does not hold over $[c_k, c_j]$ ($j - i + 1$ cases). As a matter of fact, the expansion rule imposes such a condition for a single element c_k in $\mathbb{C}_B$ (the least $c_i \leq c_k \leq c_j$ which has not been used yet), and it does not change the flag (which remains equal to 0). In this way, all elements will be eventually taken into consideration, including those elements in between c_i and c_j that will be added to $\mathbb{C}_B$ in some subsequent steps of the tableau construction.

Let us define now the notions of open and closed branch. We say that a node $\mathbf{n}$ in a decorated tree $\mathcal{T}$ is **available on a branch** B it belongs to if and only if $u(\mathbf{n}, \mathbf{B}) = \mathbf{0}$. The branch-expansion rule is **applicable** to a node $\mathbf{n}$ on a branch B if the node is available on B and the application of the rule generates at least one successor node with a new labeled formula. This second condition is needed to avoid looping of the application of the rule on formulas $\neg(\psi_0 C \psi_1)$, $\neg(\psi_0 D \psi_1)$, and $\neg(\psi_0 T \psi_1)$.

Definition 4 *A branch B is **closed** if some of the following conditions holds:*

(1) there are two nodes $\mathbf{n}, \mathbf{n}' \in B$ such that $\nu(\mathbf{n}) = ((\psi, [c_i, c_j]), \mathbb{C}, u)$ and $\nu(\mathbf{n}') = ((\neg\psi, [c_i, c_j]), \mathbb{C}', u)$ for some formula ψ and $c_i, c_j \in C \cap C'$;

(2) there is a node $\mathbf{n}$ such that $\nu(\mathbf{n}) = ((\pi, [c_i, c_j]), \mathbb{C}, u)$ and $c_i \neq c_j$;

(3) there is a node $\mathbf{n}$ such that $\nu(\mathbf{n}) = ((\neg\pi, [c_i, c_j]), \mathbb{C}, u)$ and $c_i = c_j$.

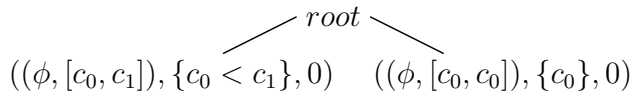
*If none of the above conditions hold, the branch is **open**.*

Definition 5 *The **branch-expansion strategy** for a branch B in a decorated tree $\mathcal{T}$ is defined as follows:*

(1) apply the branch-expansion rule to a branch B only if it is open;

(2) if B is open, apply the branch-expansion rule to the first available node one encounters moving from the root to the leaf of B to which the branch-expansion rule is applicable (if any).

Definition 6 *An* **initial tableau** *for a given formula $\phi \in \text{BCDT}^+$ is the following finite decorated tree $\mathcal{T}$:*



*where the value of u is 0. A **tableau** for a given formula $\phi \in \text{BCDT}^+$ is any finite decorated tree isomorphic to a finite decorated tree $\mathcal{T}$ obtained by expanding the initial tableau for ϕ through successive applications of the branch-expansion strategy to the existing branches.*

It is easy to show that if $\phi \in \text{BCDT}^+$, $\mathcal{T}$ is a tableau for ϕ, $\mathbf{n} \in \mathcal{T}$, and $\mathbb{C}$ is the ordered set in the decoration of $\mathbf{n}$, then $\mathbb{C}$ has the linear interval property.

Definition 7 *A tableau for* BCDT^+ *is **closed** if and only if every branch in it is closed, otherwise it is **open**.*

We conclude the section by giving some examples of the application of the proposed method (for the sake of readability, we omit some minor details in the figures). As a first example, let ϕ be the unsatisfiable formula $pT\neg(\top Cp)$. A closed tableau for ϕ is given in Figure 2. As a second example, let ϕ be the formula $\neg\pi \wedge \neg(\neg((p \wedge \neg\pi)T\neg p)T\neg((p \wedge \neg\pi)T\neg p))$, which is satisfiable, but it only admits infinite models. Hence, all its tableaux include at least one open branch. A tableau (to be further expanded) for ϕ is given in Figure 3. Finally, let ϕ be the formula $p \leftrightarrow pT\pi$, which is one of the CDT axioms given in [32]. Such an axiomatic system is (claimed to be) sound and complete for the class of all linear (non-strict) interval structures. From this, it follows that the negation of the formula $(p \rightarrow pT\pi) \wedge lin$ (cf. Section 3) should be unsatisfiable in the class of all (non-strict) interval structures. However, the open tableau for $\neg(p \rightarrow pT\pi)$ shown in Figure 4 proves that this is not the case (as matter of fact, to remedy this it suffices to substitute $pT\pi \rightarrow p$ for $p \leftrightarrow pT\pi$).

4.1 Soundness and Completeness

Definition 8 *Given a set S of labeled formulas with labels in $\mathbb{C}$, we say that S is **satisfiable over** $\mathbb{C}$ if there exists a non-strict model $\mathbf{M}^+ = \langle \mathbb{D}, \mathbb{I}(\mathbb{D})^+, V \rangle$ such that $\mathbb{D}$ is an extension of $\mathbb{C}$ and $\mathbf{M}^+, [c_i, c_j] \Vdash \psi$ for all $(\psi, [c_i, c_j]) \in S$.*

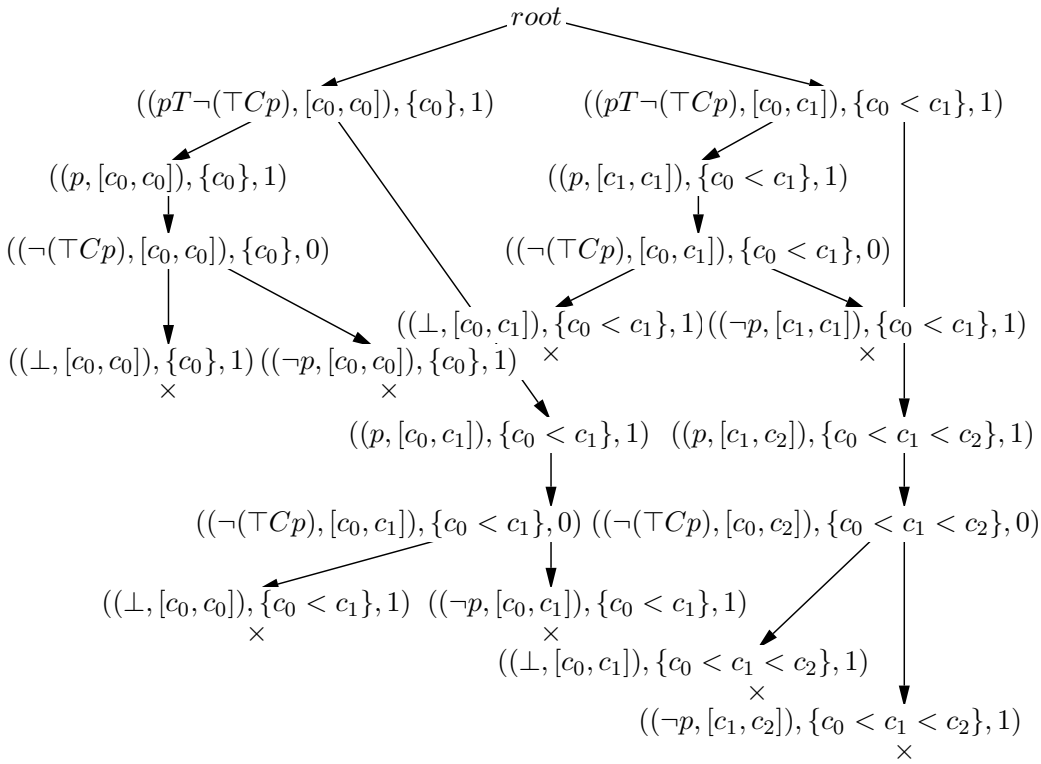


Fig. 2. A closed tableau for the formula $pT\neg(\top Cp)$.

If S contains only one labeled formula, the notion of satisfiability of a (labeled) formula over $\mathbb{C}$ is equivalent to the notion of satisfiability given in Section 2.

Theorem 1 (Soundness) *If $\phi \in$ BCDT$^+$ and a tableau $\mathcal{T}$ for ϕ is closed, then ϕ is not satisfiable.*

Proof. We will prove by induction on the height h of a node $\mathbf{n}$ in the tableau $\mathcal{T}$ the following claim: *if every branch including* $\mathbf{n}$ *is closed, then the set* $S(\mathbf{n})$ *of all labeled formulas in the decorations of the nodes between* $\mathbf{n}$ *and the root is not satisfiable over* $\mathbb{C}$*, where* $\mathbb{C}$ *is the interval structure in the decoration of* $\mathbf{n}$.

If $h = 0$, then $\mathbf{n}$ is a leaf and the unique branch B containing $\mathbf{n}$ is closed. Then, either $S(\mathbf{n})$ contains both the labeled formulas $(\psi, [c_k, c_l])$ and $(\neg\psi, [c_k, c_l])$ for some BCDT$^+$-formula ψ and $c_k, c_l \in \mathbb{C}$, or the labeled formula $(\pi, [c_k, c_l])$ and $c_k \neq c_l$, or the labeled formula $(\neg\pi, [c_k, c_l])$ and $c_k = c_l$. Take any model $\mathbf{M}^+ = \langle \mathbb{D}, \mathbb{I}(\mathbb{D})^+, V\rangle$ where $\mathbb{D}$ is an extension of $\mathbb{C}$. In the first case, clearly $\mathbf{M}^+, [c_k, c_l] \Vdash \psi$ if and only if $\mathbf{M}^+, [c_k, c_l] \nVdash \neg\psi$. In the second (resp., third) case, $\mathbf{M}^+, [c_k, c_l] \Vdash \pi$ (resp., $\neg\pi$) if and only if $c_k = c_l$ (resp., $c_k \neq c_l$). Hence, $S(\mathbf{n})$ is not satisfiable over $\mathbb{C}$. Otherwise, suppose $h > 0$. Then either $\mathbf{n}$ has been generated as one of the successors, but not the last one, when applying the branch-expansion rule in $\wedge$, C, D, T, $\neg C, \neg D$, or $\neg T$ cases, or the branch-expansion rule has been applied to some labeled formula $(\psi, [c_i, c_j]) \in S(\mathbf{n}) - \{\Phi(\mathbf{n})\}$ to extend the branch at $\mathbf{n}$. We deal with the latter case. The former can be dealt with in the same way. Let

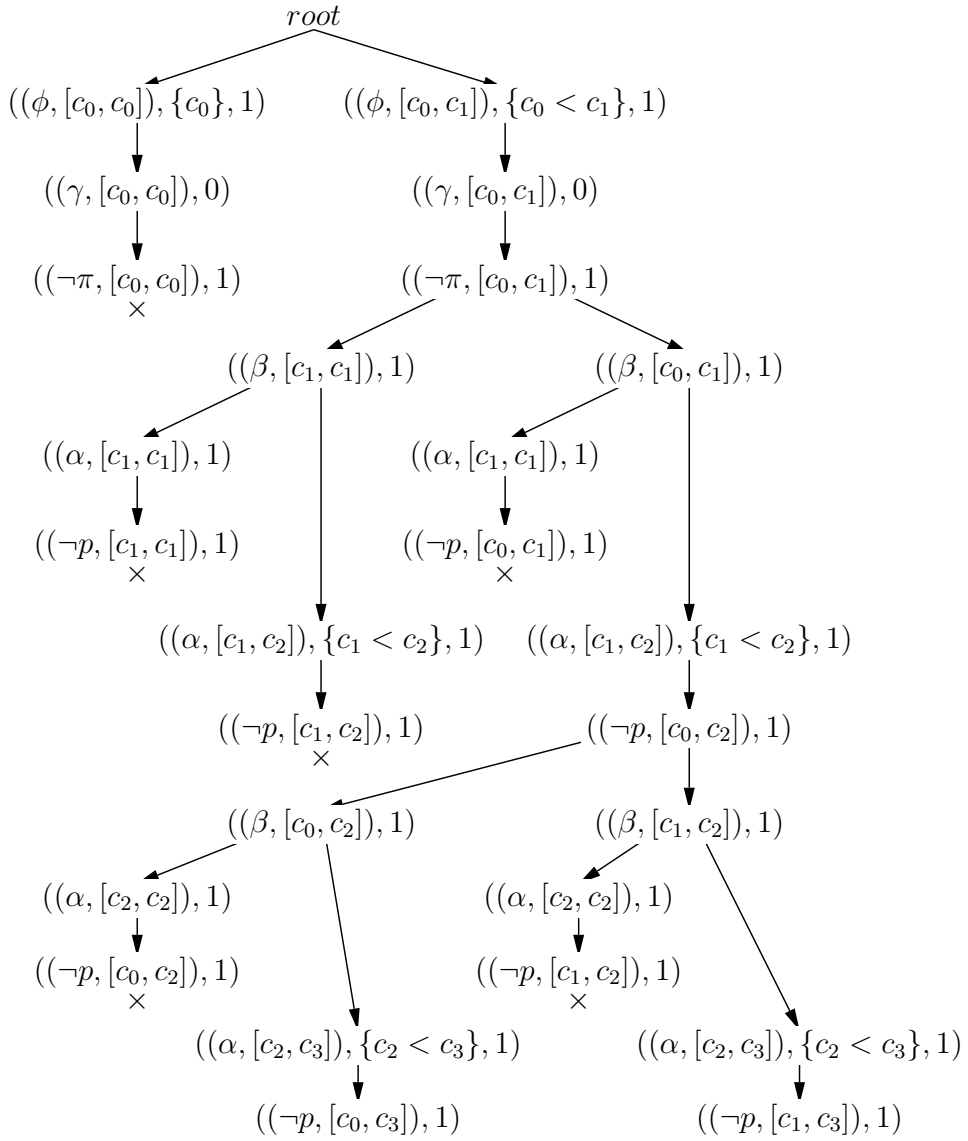


Fig. 3. A tableau (to be further expanded) for the formula $\neg\pi \wedge \neg(\neg((p \wedge \neg\pi)T\neg p)T\neg((p \wedge \neg\pi)T\neg p))$, where α, β, and γ stand for $p \wedge \neg\pi$, $\alpha T \neg p$, and $\neg(\neg\beta T\neg\beta)$, respectively.

$\mathbb{C} = \{c_1, \ldots, c_n\}$, be the interval structure from the decoration of $\mathbf{n}$. Notice that every branch passing through any successor of $\mathbf{n}$ must be closed, so the inductive hypothesis applies to all successors of $\mathbf{n}$. We consider the possible cases for the branch-expansion rule applied at $\mathbf{n}$:

- Let $\psi = \neg\neg\xi$. Then there exists $\mathbf{n_0}$ such that $\nu(\mathbf{n_0}) = ((\xi, [c_i, c_j]), \mathbb{C}, u)$ and $\mathbf{n_0}$ is a successor of $\mathbf{n}$. Since every branch containing $\mathbf{n}$ is closed, then every branch containing $\mathbf{n_0}$ is closed. By the inductive hypothesis, $S(\mathbf{n_0})$ is not satisfiable over $\mathbb{C}$ (since $\mathbf{n_0} \prec \mathbf{n}$). Since ξ_0 and $\neg\neg\xi_0$ are equivalent, $S(\mathbf{n})$ cannot be satisfiable over $\mathbb{C}$;
- Let $\psi = \xi_0 \wedge \xi_1$. Then there are two nodes $\mathbf{n_0} \in B$ and $\mathbf{n_1} \in B$ such that

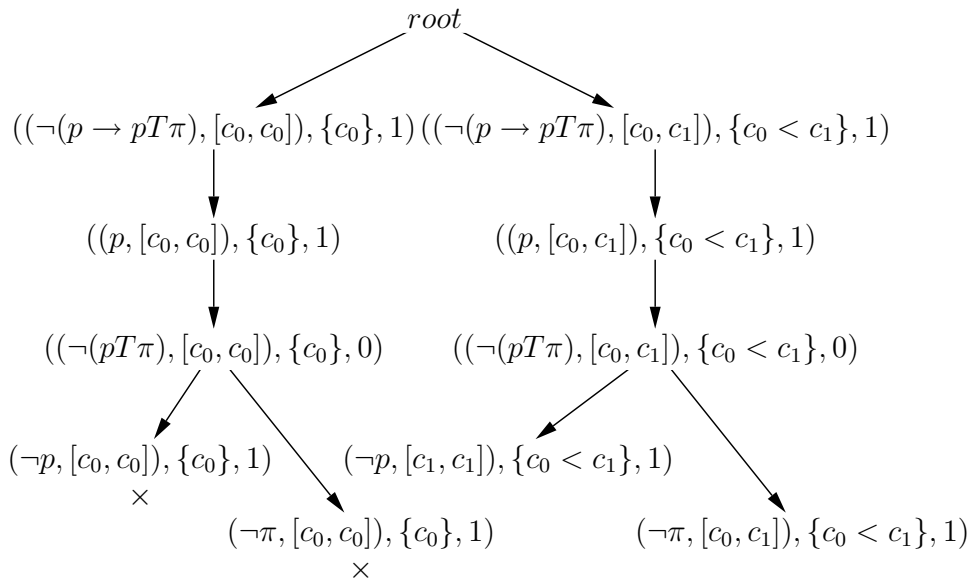


Fig. 4. An open tableau for the formula $\neg(p \rightarrow pT\pi)$.

$\nu(\mathbf{n_0}) = ((\xi_0, [c_i, c_j]), \mathbb{C}, u)$, $\nu(\mathbf{n_1}) = ((\xi_1, [c_i, c_j]), \mathbb{C}, u)$, and, without loss of generality, $\mathbf{n_0}$ is the successor of $\mathbf{n}$ and $\mathbf{n_1}$ is the successor of $\mathbf{n_0}$. Since every branch containing $\mathbf{n}$ is closed, then every branch containing $\mathbf{n_1}$ is closed. By the inductive hypothesis, $S(\mathbf{n_1})$ is not satisfiable over $\mathbb{C}$ since $\mathbf{n_1} \prec \mathbf{n}$. Since every model over $\mathbb{C}$ satisfying $S(\mathbf{n})$ must, in particular, satisfy $(\xi_0 \wedge \xi_1, [c_i, c_j])$, and hence $(\xi_0, [c_i, c_j])$ and $(\xi_1, [c_i, c_j])$, it follows that $S(\mathbf{n})$, $S(\mathbf{n_0})$, and $S(\mathbf{n_1})$ are equi-satisfiable over $\mathbb{C}$. Therefore, $S(\mathbf{n})$ is not satisfiable over $\mathbb{C}$;

- Let $\psi = \neg(\xi_1 \wedge \xi_2)$. Then there exist two successor nodes $\mathbf{n_0}$ and $\mathbf{n_1}$ of $\mathbf{n}$ such that $\nu(\mathbf{n_0}) = ((\xi_0, [c_i, c_j]), \mathbb{C}, u_0)$, $\nu(\mathbf{n_1}) = ((\xi_1, [c_i, c_j]), \mathbb{C}, u_1)$, $\mathbf{n_0}, \mathbf{n_1} \prec \mathbf{n}$. Since every branch containing $\mathbf{n}$ is closed, then every branch containing $\mathbf{n_0}$ and every branch containing $\mathbf{n_1}$ is closed. By the inductive hypothesis $S(\mathbf{n_0})$ and $S(\mathbf{n_1})$ are not satisfiable over $\mathbb{C}$. Since every model over $\mathbb{C}$ satisfying $S(\mathbf{n})$ must also satisfy $(\xi_0, [c_i, c_j])$ or $(\xi_1, [c_i, c_j])$, it follows that $S(\mathbf{n})$ cannot be satisfiable over $\mathbb{C}$;

- Let $\psi = \neg(\xi_0 C \xi_0)$. Suppose that $S(\mathbf{n})$ is satisfiable over $\mathbb{C}$. Since $(\neg(\xi_0 C \xi_1), [c_i, c_j]) \in S(\mathbf{n})$, there is a model $\mathbf{M}^+ = \langle \mathbb{D}, \mathbb{I}(\mathbb{D})^+, V \rangle$ such that $\mathbb{D}$ is an extension of $\mathbb{C}$ and $\mathbf{M}^+, [c_i, c_j] \Vdash \neg(\xi_0 C \xi_1)$. So, for every c_k such that $c_i \leq c_k \leq c_j$, we have that $\mathbf{M}^+, [c_i, c_k] \Vdash \neg\xi_0$ or $\mathbf{M}^+, [c_k, c_j] \Vdash \neg\xi_1$. By construction, the two immediate successors of $\mathbf{n}$ are $\mathbf{n_1}$ and $\mathbf{n_2}$ such that, for an element c_k with $c_i \leq c_k \leq c_j$, $(\neg\xi_0, [c_i, c_k])$ is in the decoration of $\mathbf{n_0}$ and $(\neg\xi_1, [c_k, c_j])$ is in the decoration of $\mathbf{n_1}$. By inductive hypothesis, since $\mathbf{n_1}, \mathbf{n_2} \prec \mathbf{n}$, $S(\mathbf{n_1})$ and $S(\mathbf{n_2})$ are not satisfiable over $\mathbb{C}$. Thus, such a model $\mathbf{M}^+$ cannot exist, and $S(\mathbf{n})$ is not satisfiable over $\mathbb{C}$;

- The cases $\psi = \neg(\xi_0 D \xi_1)$ and $\psi = \neg(\xi_0 T \xi_1)$ are analogous;

- Let $\psi = \xi_0 C \xi_1$. Assuming that $S(\mathbf{n})$ is satisfiable over $\mathbb{C}$, there is a model $\mathbf{M}^+ = \langle \mathbb{D}, \mathbb{I}(\mathbb{D})^+, V \rangle$, where $\mathbb{D}$ is an extension of $\mathbb{C}$, such that $\mathbf{M}^+, [c_i, c_j] \Vdash \theta$ for all $(\theta, [c_i, c_j]) \in S(\mathbf{n})$. In particular, $\mathbf{M}^+, [c_i, d] \Vdash \xi_0$ and $\mathbf{M}^+, [d, c_j] \Vdash \xi_1$ for some $c_i \leq d \leq c_j$. Consider two cases:

(1) If $d \in \mathbb{C}$, then $d = c_m$ for some $c_i \leq c_m \leq c_j$. But among the successors of $\mathbf{n}$ there are two nodes $\mathbf{n_m}, \mathbf{m_m}$ where $\nu(\mathbf{n_m}) = ((\xi_0, [c_i, c_m]), \mathbb{C}, u)$ and $\nu(\mathbf{m_m}) = ((\xi_1, [c_m, c_j]), \mathbb{C}, u)$, and since $\mathbf{n_m}, \mathbf{m_m} \prec \mathbf{n}$ (without loss of generality, suppose $\mathbf{n_m} \prec \mathbf{m_m}$), by the inductive hypothesis $S(\mathbf{n_m}) = S(\mathbf{n}) \cup \{(\xi_0, [c_i, c_m]), (\xi_1, [c_m, c_j])\}$ is not satisfiable over $\mathbb{C}$, which is a contradiction, and $S(\mathbf{n})$ is not satisfiable over $\mathbb{C}$;

(2) If $d \notin \mathbb{C}$, then there is an m such that $i \leq m \leq j-1$ and $c_m < d < c_{m+1}$. Hence, there are two successors $\mathbf{n'_m}, \mathbf{m'_m}$ of $\mathbf{n}$ such that $\nu(\mathbf{n'_m}) = ((\xi_0, [c_i, d]), \mathbb{C} \cup \{d\}, u)$, $\nu(\mathbf{m'_m}) = ((\xi_1, [d, c_j]), \mathbb{C} \cup \{d\}, u)$, and since $\mathbf{n'_m}, \mathbf{m'_m} \prec \mathbf{n}$ (without loss of generality, suppose $\mathbf{n'_m} \prec \mathbf{m'_m}$), by the inductive hypothesis $S(\mathbf{n'_m}) = S(\mathbf{n}) \cup \{(\xi_0, [c_i, d]), (\xi_1, [d, c_j])\}$ is not satisfiable over $\mathbb{C} \cup \{d\}$ which, again, is a contradiction, and $S(\mathbf{n})$ is not satisfiable over $\mathbb{C}$;

- Let $\psi = \xi_0 D \xi_1$. Assuming that $S(\mathbf{n})$ is satisfiable over $\mathbb{C}$, there is a model $\mathbf{M}^+ = \langle \mathbb{D}, \mathbb{I}(\mathbb{D})^+, V \rangle$, where $\mathbb{D}$ is an extension of $\mathbb{C}$, such that $\mathbf{M}^+, [c_i, c_j] \Vdash \theta$ for all $(\theta, [c_i, c_j]) \in S(\mathbf{n})$. In particular, $\mathbf{M}^+, [d, c_i] \Vdash \xi_0$ and $\mathbf{M}^+, [d, c_j] \Vdash \xi_1$ for some $d \leq c_i$. Consider 3 cases:

(1) If $d \in \mathbb{C}$, then $d = c_m$ for some $c_m \leq c_i$. But between the successors of $\mathbf{n}$ there are two nodes $\mathbf{n_m}, \mathbf{m_m}$ where $\nu(\mathbf{n_m}) = ((\xi_0, [c_m, c_i]), \mathbb{C}, u)$ and $\nu(\mathbf{m_m}) = ((\xi_1, [c_m, c_j]), \mathbb{C}, u)$, and since $\mathbf{n_m}, \mathbf{m_m} \prec \mathbf{n}$ (without loss of generality, suppose $\mathbf{n_m} \prec \mathbf{m_m}$), by the inductive hypothesis $S(\mathbf{n_m}) = S(\mathbf{n}) \cup \{(\xi_0, [c_m, c_i]), (\xi_1, [c_m, c_j])\}$ is not satisfiable over $\mathbb{C}$, which is a contradiction, and $S(\mathbf{n})$ is not satisfiable over $\mathbb{C}$;

(2) If $d \notin \mathbb{C}$ and there exist a minimal element $c \in \mathbb{C}$ and an index m such that $c_m, c_{m+1} \in [c, c_i]$ and $c_m < d < c_{m+1}$, then there are two successors $\mathbf{n'_m}, \mathbf{m'_m}$ of $\mathbf{n}$ such that $\nu(\mathbf{n'_m}) = ((\xi_0, [d, c_i]), \mathbb{C} \cup \{d\}, u)$ and $\nu(\mathbf{m'_m}) = ((\xi_1, [d, c_j]), \mathbb{C} \cup \{d\}, u)$, and since $\mathbf{n'_m}, \mathbf{m'_m} \prec \mathbf{n}$ (without loss of generality, suppose $\mathbf{n'_m} \prec \mathbf{m'_m}$), by the inductive hypothesis $S(\mathbf{n'_m}) = S(\mathbf{n}) \cup \{(\xi_0, [d, c_i]), (\xi_1, [d, c_j])\}$ is not satisfiable over $\mathbb{C} \cup \{d\}$ which, again, is a contradiction, and $S(\mathbf{n})$ is not satisfiable over $\mathbb{C}$;

(3) If $d \notin \mathbb{C}$ and there exist a minimal element $c \in \mathbb{C}$ and an index m such that $c_{m+1} \in [c, c_i]$, $d < c_{m+1}$, and d is not comparable with all predecessors of c_{m+1}, then, again, there are two successor nodes $\mathbf{n''_m}, \mathbf{m''_m}$ of $\mathbf{n}$ such that $\nu(\mathbf{n''_m}) = ((\xi_0, [d, c_i]), \mathbb{C} \cup \{d\}, u)$ and $\nu(\mathbf{m''_m}) = ((\xi_1, [d, c_j]), \mathbb{C} \cup \{d\}, u)$, and since $\mathbf{n''_m}, \mathbf{m''_m} \prec \mathbf{n}$ (without loss of generality, suppose $\mathbf{n''_m} \prec \mathbf{m''_m}$), by the inductive hypothesis $S(\mathbf{n''_m}) = S(\mathbf{n}) \cup \{(\xi_0, [d, c_i]), (\xi_1, [d, c_j])\}$ is not satisfiable over $\mathbb{C} \cup \{d\}$ which, again, is a contradiction, and $S(\mathbf{n})$ is not satisfiable over $\mathbb{C}$;

- The case of $\psi = \xi_0 T \xi_1$ is similar. $\Box$

Definition 9 *If $\mathcal{T}_0$ is the three-node tableau built up from a root with void decoration and two leaves decorated respectively by $((\phi, [c_b, c_e]), \{c_b < c_e\}, 0)$ and $((\phi, [c_b, c_b]), \{c_b\}, 0)$ for a given* BCDT$^+$*-formula ϕ, the **limit tableau** $\overline{\mathcal{T}}$ for ϕ is the (possibly infinite) decorated tree obtained as follows. First, for all i, $\mathcal{T}_{i+1}$ is the tableau obtained by the simultaneous application of the branch-expansion strategy to every branch in $\mathcal{T}_i$. Then, we ignore all flags from the decorations of the nodes in every $\mathcal{T}_i$. Thus, we obtain a chain by inclusion of decorated trees: $\mathcal{T}_1 \subseteq \mathcal{T}_2 \subseteq \ldots$, and we define $\overline{\mathcal{T}} = \bigcup_{i=0}^{\infty} \mathcal{T}_i$.*

Notice that the chain above may stabilize at some $\mathcal{T}_i$ if it closes, or if the branch-expansion rule is not applicable to any of its branches. If $\overline{\mathcal{T}}$ is a limit tableau, we associate with each branch B in $\overline{\mathcal{T}}$ the interval structure $\mathbb{C}_B = \bigcup_{i=0}^{\infty} \mathbb{C}_{B_i}$, where, for all i, $\mathbb{C}_{B_i}$ is the interval structure from the decoration of the leaf of the (sub-)branch B_i of B in $\mathcal{T}_i$. The definitions of closed and open branches readily apply to $\overline{\mathcal{T}}$.

Definition 10 *A branch in a (limit) tableau is **saturated** if there are no nodes on that branch to which the branch-expansion rule is applicable on the branch. A (limit) tableau is **saturated** if every open branch in it is saturated.*

Now we will show that the set of all labeled formulas on an open branch in a limit tableau has the saturation properties of a Hintikka set in first-order logic.

Lemma 2 *Every limit tableau is saturated.*

Proof. Given a node $\mathbf{n}$ in a limit tableau $\overline{\mathcal{T}}$, we denote by $d(\mathbf{n})$ the distance (number of edges) between $\mathbf{n}$ and the root of $\overline{\mathcal{T}}$. Now, given a branch B in $\overline{\mathcal{T}}$, we will prove by induction on $d(\mathbf{n})$ that after every step of the expansion of that branch at which the branch-expansion rule becomes applicable to $\mathbf{n}$ (because $\mathbf{n}$ has just been introduced, or because a new point has been introduced in the interval structure on B) that rule is subsequently applied on B to that node.

Suppose the inductive hypothesis holds for all nodes with distance to the root less than l. Let $d(\mathbf{n}) = l$ and the branch-expansion rule has become applicable to $\mathbf{n}$. If there are no nodes between the root (incl. the root) and $\mathbf{n}$ (excl. $\mathbf{n}$) to which the branch-expansion rule is applicable at that moment, the next application of the branch-expansion rule on B is to $\mathbf{n}$. Otherwise, consider the closest-to-$\mathbf{n}$ node $\mathbf{n}^*$ between the root and $\mathbf{n}$ to which the branch-expansion rule is applicable or will become applicable on B at least once thereafter. (Such a node exists because there are only finitely many nodes between $\mathbf{n}$ and the root.) Since $d(\mathbf{n}^*) < d(\mathbf{n})$, by the inductive hypothesis the branch-expansion rule has been subsequently applied to $\mathbf{n}^*$. Then the next application of the branch-expansion rule on B must have been to $\mathbf{n}$ and that completes the induction. Now, assuming that a branch in a limit tableau is not saturated, consider the closest-to-the-root node $\mathbf{n}$ on that branch B to which the branch-expansion rule is applicable on that branch. If $\Phi(\mathbf{n})$ is none of the cases

$\neg C, \neg D$, and $\neg T$, then the branch-expansion rule has become applicable to $\mathbf{n}$ at the step when $\mathbf{n}$ is introduced, and by the claim above, it has been subsequently applied, at which moment the node has become unavailable thereafter, which contradicts the assumption. Suppose that $\Phi(\mathbf{n}) = \neg(\psi_0 C \psi_1)$. Then an application of the rule on B would create two successors with labels $(\neg\psi_0, [c_i, c])$ and $(\neg\psi_1, [c, c_j])$, at least one of them new on B. But c_i, c_j, c have already been introduced at some (finite) step of the construction of B and at the first step when the three of them, as well as $\mathbf{n}$, have appeared on the branch, the branch-expansion rule has become applicable to $\mathbf{n}$, hence is has been subsequently applied on B and that application must have introduced the labels $(\psi_0, [c_i, c])$ and $(\psi_1, [c, c_j])$ on B, which again contradicts the assumption. The same holds if $\Phi(\mathbf{n}) = \neg(\psi_0 D \psi_1)$ or $\Phi(\mathbf{n}) = \neg(\psi_0 D \psi_1)$. □

Corollary 3 *Let ϕ be a* BCDT$^+$*-formula and $\overline{\mathcal{T}}$ be the limit tableau for ϕ. For every open branch B in $\overline{\mathcal{T}}$, the following closure properties hold:*

- *If there is a node $\mathbf{n} \in B$ such that $\nu(\mathbf{n}) = ((\neg\neg\psi, [c_i, c_j]), \mathbb{C}, u)$, then there is a node $\mathbf{n_0} \in B$ such that $\nu(\mathbf{n_0}) = ((\psi, [c_i, c_j]), \mathbb{C}, u_0)$;*
- *If there is a node $\mathbf{n} \in B$ such that $\nu(\mathbf{n}) = ((\psi_0 \wedge \psi_1, [c_i, c_j]), \mathbb{C}, u)$, then there is a node $\mathbf{n_0} \in B$ such that $\nu(\mathbf{n_0}) = ((\psi_0, [c_i, c_j]), \mathbb{C}, u_0)$ and a node $\mathbf{n_1} \in B$ such that $\nu(\mathbf{n_1}) = ((\psi_1, [c_i, c_j]), \mathbb{C}, u_1)$;*
- *If there is a node $\mathbf{n} \in B$ such that $\nu(\mathbf{n}) = ((\neg(\psi_0 \wedge \psi_1), [c_i, c_j]), \mathbb{C}, u)$, then there is a node $\mathbf{n_0} \in B$ such that $\nu(\mathbf{n_0}) = ((\neg\psi_0, [c_i, c_j]), \mathbb{C}, u_0)$ or a node $\mathbf{n_1} \in B$ such that $\nu(\mathbf{n_1}) = ((\neg\psi_1, [c_i, c_j]), \mathbb{C}, u_1)$;*
- *If there is a node $\mathbf{n} \in B$ such that $\nu(\mathbf{n}) = ((\psi_0 C \psi_1, [c_i, c_j]), \mathbb{C}, u)$, then, for some $c \in \mathbb{C}_B$ such that $c_i \leq c \leq c_j$ there are two nodes $\mathbf{n}', \mathbf{m}' \in B$ such that $\nu(\mathbf{n}') = ((\psi_0, [c_i, c]), \mathbb{C}', u')$ and $\nu(\mathbf{m}') = ((\psi_1, [c, c_j]), \mathbb{C}', u')$;*
- *Similarly for every node $\mathbf{n}$ with $\Phi(\mathbf{n}) = \psi_0 D \psi_1$ or $\Phi(\mathbf{n}) = \psi_0 T \psi_1$;*
- *If there is a node $\mathbf{n} \in B$ such that $\nu(\mathbf{n}) = ((\neg(\psi_0 C \psi_1), [c_i, c_j]), \mathbb{C}, u)$, then for all $c \in \mathbb{C}_B$ such that $c_i \leq c \leq c_j$, there is a node $\mathbf{n}' \in B$ such that $\nu(\mathbf{n}') = ((\neg\psi_0, [c_i, c]), \mathbb{C}', u')$ or a node $\mathbf{m}' \in B$ such that $\nu(\mathbf{m}') = ((\neg\psi_1, [c, c_j]), \mathbb{C}', u')$;*
- *Similarly for every node $\mathbf{n}$ with $\Phi(\mathbf{n}) = \neg(\psi_0 D \psi_1)$ or $\Phi(\mathbf{m}) = \neg(\psi_0 T \psi_1)$.*

Lemma 4 *If the limit tableau for some formula $\phi \in$*BCDT$^+$ *is closed, then some finite tableau for ϕ is closed.*

Proof. Suppose the limit tableau for ϕ is closed. Then every branch closes at some finite step of the construction and then remains finite. Since the branch-expansion rule always produces finitely many successors, every finite tableau is finitely branching, and hence so is the limit tableau. Then, by König's lemma, the limit tableau, being a finitely branching tree with no infinite branches, must be finite, hence its construction stabilizes at some finite stage. At that stage a closed tableau for ϕ is constructed. □

Theorem 5 (Completeness) *Let $\phi \in$BCDT$^+$ be a valid formula. Then there is a closed tableau for $\neg\phi$.*

Proof. We will show that the limit tableau $\overline{\mathcal{T}}$ for $\neg\phi$ is closed, whence the claim follows by the previous lemma.

By contraposition, suppose that $\overline{\mathcal{T}}$ has an open branch B. Let $\mathbb{C}_B$ be the interval structure associated with B and $S(B)$ be the set of all labeled formulas on B. Consider the model $\mathbf{M}^+ = \langle \mathbb{C}_B, V \rangle$ where, for every $[c_i, c_j] \in \mathbb{I}(\mathbb{C}_B)^+$ and $p \in \mathcal{AP}$, $p \in V([c_i, c_j])$ iff $(p, [c_i, c_j]) \in \Phi(B)$. We show by induction on ψ that, for every $(\psi, [c_i, c_j]) \in S(B)$, $\mathbf{M}^+, [c_i, c_j] \Vdash \psi$.

We reason by induction on the complexity of ψ:

- Let $\psi = \pi$ (resp., $\psi = \neg\pi$). Since $(\pi, [c_i, c_j]) \in S(B)$ (resp., $(\neg\pi, [c_i, c_j]) \in S(B)$) and B is open, then $c_i \neq c_j$ (resp., $c_i = c_j$). Hence $\mathbf{M}^+, [c_i, c_j] \Vdash \pi$ (resp., $\mathbf{M}^+, [c_i, c_j] \Vdash \neg\pi$);
- Let $\psi = p$ or $\psi = \neg p$ where $p \in \mathcal{AP}$. Then the claim follows by definition, because if $(\neg p, [c_i, c_j]) \in S(B)$ then $(p, [c_i, c_j]) \notin S(B)$ since B is open;
- Let $\psi = \neg\neg\xi$. Then by Corollary 3, $(\xi, [c_i, c_j]) \in S(B)$, and by inductive hypothesis $\mathbf{M}^+, [c_i, c_j] \Vdash \xi$. So $\mathbf{M}^+, [c_i, c_j] \Vdash \psi$;
- Let $\psi = \xi_0 \wedge \xi_1$. Then by Corollary 3, $(\xi_0, [c_i, c_j]) \in S(B)$ and $(\xi_1, [c_i, c_j]) \in S(B)$. By inductive hypothesis, $\mathbf{M}^+, [c_i, c_j] \Vdash \xi_0$ and $\mathbf{M}^+, [c_i, c_j] \Vdash \xi_1$, so $\mathbf{M}^+, [c_i, c_j] \Vdash \psi$;
- Let $\psi = \neg(\xi_0 \wedge \xi_1)$. Then by Corollary 3, $(\neg\xi_0, [c_i, c_j]) \in S(B)$ or $(\neg\xi_1, [c_i, c_j]) \in S(B)$. By inductive hypothesis $\mathbf{M}^+, [c_i, c_j] \Vdash \neg\xi_0$ or $\mathbf{M}^+, [c_i, c_j] \Vdash \neg\xi_1$, so $\mathbf{M}^+, [c_i, c_j] \Vdash \psi$;
- Let $\psi = \xi_0 C \xi_1$. Then by Corollary 3, $(\xi_0, [c_i, c]) \in S(B)$ and $(\xi_1, [c, c_i]) \in S(B)$ for some $c \in \mathbb{C}_B$ such that $c_i \leq c \leq c_j$. Thus, by inductive hypothesis, $\mathbf{M}^+, [c_i, c] \Vdash \xi_0$ and $\mathbf{M}^+, [c, c_j] \Vdash \xi_1$, and thus $\mathbf{M}^+, [c_i, c_j] \Vdash \psi$;
- Similarly for $\psi = \xi_0 D \xi_1$ and $\psi = \xi_0 T \xi_1$;
- Let $\psi = \neg(\xi_0 C \xi_1)$. Then by Corollary 3, for all $c \in \mathbb{C}_B$ such that $c_i \leq c \leq c_j$, $(\neg\xi_0, [c_i, c]) \in S(B)$ and $(\neg\xi_1, [c, c_j]) \in S(B)$. Hence, by the inductive hypothesis, $\mathbf{M}^+, [c_i, c] \Vdash \neg\xi_0$ and $\mathbf{M}^+, [c, c_j] \Vdash \neg\xi_1$, for all $c_i \leq c \leq c_j$. Thus, $\mathbf{M}^+, [c_i, c_j] \Vdash \psi$;
- Similarly for $\psi = \neg(\xi_0 D \xi_1)$ and $\psi = \neg(\xi_0 T \xi_1)$.

This completes the induction. In particular, we obtain that $\neg\phi$ is satisfied in $\mathbf{M}^+$, which is in contradiction with the assumption that ϕ is valid. $\Box$

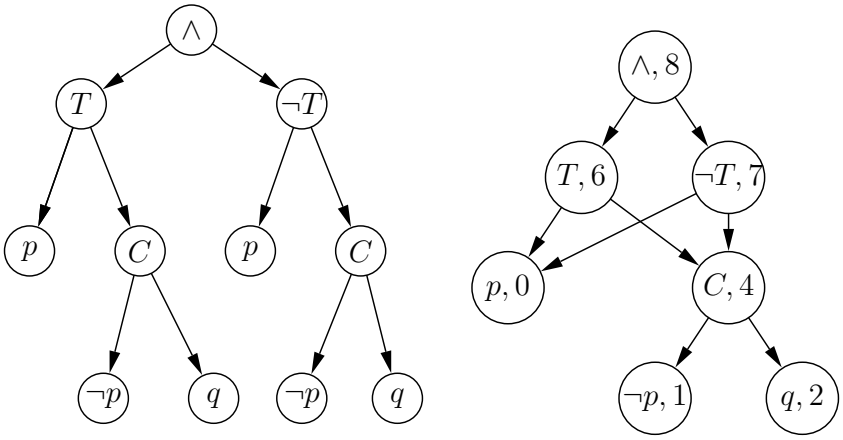


Fig. 5. The tree for the formula $\phi = (pT(\neg pCq)) \wedge \neg(pT(\neg pCq))$.

5 An implementation of the tableau method

In this section, we describe the main features of our implementation of the proposed tableau method, which has been successfully used for testing a number of BCDT$^+$-formulas [28].

Input formula. The input BCDT$^+$-formula is arranged in a tree structure whose internal nodes contain a Boolean connective or a temporal operator and whose leaves contain a literal. As an example, the formula $\phi = (pT(\neg pCq)) \wedge \neg(pT(\neg pCq))$ generates the tree depicted in Figure 5, left. The subtrees of the tree for a formula ϕ identify the subformulas of ϕ. Obviously, identical subformulas give raise to identical subtrees. To obtain a more compact representation, we collapse identical subtrees, thus turning the tree into a direct acyclic graph (DAG). To this end, we define a recursive procedure that enumerates the distinct subformulas and generates a DAG whose nodes contain a subformula labeled by a natural number. We constrain such a labeling procedure to associate even (resp. odd) numbers with positive (resp. negative) subformulas; moreover, if the positive formula ψ is labeled by $n-1$, the corresponding negative subformula $\neg\psi$ (if present) is labeled by n. This allows us to check whether two subformulas are identical, or whether one of them is the negation of the other, by comparing two natural numbers. The application of this enumeration procedure to the above formula ϕ produces the DAG of Figure 5, right. Furthermore, if a formula ϕ has h distinct subformulas, we can use an array of at most $2 \cdot h$ bits to compactly represent any subset of its subformulas: we set the n-th bit of the array to 1 if and only if the subformula labeled by $n-1$ belongs to the subset.

Branch management. Any single branch of the tableau for an input formula ϕ is managed according to a depth-first strategy. A branch is (recursively) expanded until it is closed (in such a case, the procedure backtracks and it starts again with the

next branch, if any) or it is open, but saturated (in such a case, the method returns a model for the input formula). Obviously it may happen that the expansion goes forever with the branch neither closed nor (open and) saturated. A single branch contains: (i) a list L of pairs $(\psi, [c_i, c_j])$, where ψ is a subformula of ϕ and $[c_i, c_j]$ is an interval; (ii) a (possibly empty) sublist L' of L that contains only those pairs $(\psi, [c_i, c_j])$, where ψ is a formula of the form $\neg C$, $\neg D$, or $\neg T$ (universal formula); (iii) a graph G that represents the current interval structure associated with (the leaf of) the branch. Furthermore, a current position is defined for the list L. All elements of L to the left of the current position identify the expansion steps applied to the branch so far, while those to the right (including that in the current position) correspond to expansion steps that have never been executed yet. L' includes the universal subformulas of ϕ that can possibly be reused during some subsequent expansion step that extends the interval structure. The nodes of G correspond to the endpoints of the intervals belonging to the current interval structure (such a structure must be updated whenever an expansion step introduces a new endpoint). Every node of G is labeled by a unique identifier c_i and provided with two lists of pointers to its immediate successors and predecessors (cf. Figure 6).
The problem of keeping track of the subformulas of ϕ associated with (true over) the various intervals of the current interval structure is dealt with as follows. For every node (labeled by) c_i, let c_j, with $i \leq j$, be the label with maximum index for which there exists a subformula which is true over $[c_i, c_j]$, or over $[c_j, c_i]$ (if any). We associate an array $[c_i, c_{i+1}, \ldots, c_{j-1}, c_j]$ with the node c_i. Moreover, for every c_k belonging to this array, if there exists a subformula which is true over $[c_i, c_k]$ (or $[c_k, c_i]$), we provide c_k with an array of $2 \cdot h$ bits, where h is the number of distinct subformulas of ϕ. For every $1 \leq n \leq 2 \cdot h$, the array has value value 1 in the n-th position if (and only if) the subformula of ϕ labeled by $n - 1$ is true over $[c_i, c_k]$ (or $[c_k, c_i]$). Such an encoding allows one to keep track of the truth of a subformula over a given interval and to possibly detect a contradiction, as well as to withdraw the truth of a subformula over an interval during backtracking, in constant time. Finally, to efficiently deal with the labeled universal subformulas $(\psi, [c_i, c_j])$ in L', we maintain an array of successors of c_j (resp. predecessors of c_i, elements between c_i and c_j), that can be easily updated whenever a new endpoint is added to the interval structure. Taking advantage of these arrays, one can easily determine the universal subformulas associated with the branch which are activated by the addition of the endpoint.

To illustrate the management of the interval structure, in Figure 6 we describe the effects of the application of some expansion steps to a branch of a tableau for the formula $\phi = ((pTq)Tp)C\neg(pTq)$. We assume the subformulas of ϕ to be labeled as follows: $label(p) = 0, label(\neg p) = 1, label(q) = 2, label(\neg q) = 3, label(pTq) = 4, label(\neg(pTq)) = 5, label((pTq)Tp) = 6, label(\neg((pTq)Tp)) = 7, label(\phi) = 8$, and $label(\neg\phi) = 9$. At the first step, the interval structure consists of two nodes (labeled by) c_0 and c_1, with $c_0 < c_1$. The arrays $[c_0, c_1]$ and $[c_1]$ are associated with c_0 and c_1, respectively. Since $0 < 1$, to constrain the formula ϕ, with $label(\phi) = 8$, to hold over $[c_0, c_1]$, we provide the entry c_1 of the c_0-labeled node with a 10-bits

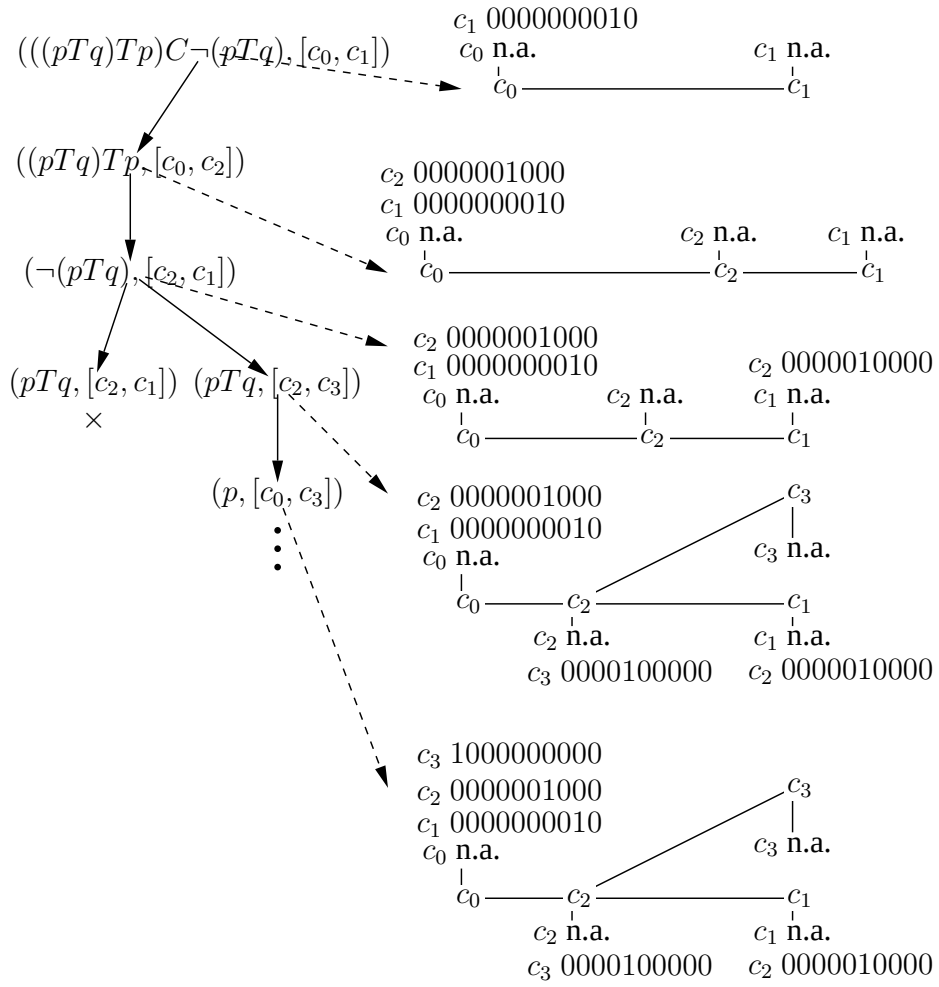


Fig. 6. An example of interval structure.

array whose 9-th bit is set to 1. Since there are not formulas associated with the intervals $[c_0, c_0]$ and $[c_1, c_1]$, there are not 10-bits arrays corresponding to the entry c_0 for the c_0-labeled node and to the entry c_1 for the c_1-labeled node. In Figure 6, we denote this situation by the expression n.a., which stands for not allocated. At the second step, we extend the branch by applying the expansion rule for C to the formula ϕ. Accordingly, we add a node c_2, with $c_0 < c_2 < c_1$, to the current interval structure. Since $(pTq)Tp$ (resp. $\neg(pTq)$) holds over $[c_0, c_2]$ (resp. $[c_2, c_1]$), and $0 < 2$ (resp., $1 < 2$), we provide the entry c_2 of the c_0-labeled (resp. c_1-labeled) node with a 10-bits array whose 7-th (resp. 6-th) bit is set to 1. The next step shows how a branch can be closed. Since c_1 is a successor of c_2, the application of the expansion rule for T to the formula $(pTq)Tp$ may result in the request for pTq (resp. p) to hold over $[c_2, c_1]$ (resp. $[c_0, c_1]$). Since $1 < 2$, the 10-bits array associated with the entry c_2 of the c_1-labeled node must be updated by setting its 5-th bit to 1. This means that we require both pTq and its negation to hold over $[c_2, c_1]$ (contradiction). Such a contradiction can be immediately detected: in the

10-bits array associated with the entry c_2 of the c_1-labeled node there exist two consecutive bits, the first one being in an odd position (in the example, the 5-th and the 6-th one), both set to 1. Once the contradiction has been detected, the procedure backtracks, and it explores alternative expansions of the branch (if any). In the example, it chooses to add a new node c_3, with $c_2 < c_3$, incomparable with c_1.

Experimental results. We developed a C-implementation of the proposed tableau method for BCDT$^+$ and we carried out some tests on an Intel x86 machine, with a 2GHz Pentium 4 CPU, 40 Gb Hard Drive serial-ATA 150, and 1Gb of DDR SDRAM. We also compared its performances with those of the well-known first-order theorem prover Spass [15] (installed on the same machine), which takes advantage of a special form of syntactic unification. To make the comparison possible, BCDT$^+$ formulas have been mapped into their first-order counterparts (in a language with a binary relation symbol $<$ which has been constrained to be a partial ordering). A comparison of the performances of the two systems on some meaningful unsatisfiable/satisfiable BCDT$^+$ formulas (execution time is measured in msec) is given in the following table.

$BCDT^+$-formula	our implementation	Spass
$\neg(\neg(\phi T\psi)C\phi \rightarrow \neg\psi)$	< 1	30
$\neg(\neg(\phi T\psi)D\psi \rightarrow \neg\phi)$	10	41
$\neg(\pi C\phi \leftrightarrow \phi)$	9	29
$\neg(\pi T\phi \leftrightarrow \phi)$	11	32
$((pTq)Tp)C\neg(pTq)$	10	26
$(p \rightarrow (pT\pi))$	10	23

6 Conclusions

In this paper, we described a general tableau method for CDT logic, interpreted over partial orders, which combines features of the classical tableau method for first-order logic with those of explicit tableau methods for modal logics with constraint label management. The method can be easily tailored to most existing propositional interval temporal logics. We proved its soundness and completeness, and we provided it with an efficient implementation in C. We are currently looking for meaningful syntactic (fragments of the logic) and/or semantic (classes of interval structures) fragments where the tableau terminates, thus providing a decision procedure.

Acknowledgments. The authors would like to thank the Italian Ministero degli Affari Esteri and the National Research Foundation of South Africa for the research grant, under the Joint Italy/South Africa Science and Technology Agreement, that they received for the project: “Theory and applications of temporal logics to computer science and artificial intelligence”.

References

[1] P. Abate, R. Gor´e, System description: The tableaux workbench, in: Proc. of the International Conference TABLEAUX 2003, Vol. 2796 of LNAI, Springer, 2003, pp. 230–236.

[2] J. Allen, Maintaining knowledge about temporal intervals, Communications of the ACM 26 (11) (1983) 832–843.

[3] B. Beckert, R. Gor´e, Free-variable tableaux for propositional modal logics, Studia Logica 69 (1) (2001) 59–96.

[4] H. Bowman, S. Thompson, A decision procedure and complete axiomatization of finite interval temporal logic with projection, Journal of Logic and Computation 13 (2) (2003) 195–239.

[5] S. Cerrito, M. Cialdea-Mayer, Bounded model search in linear temporal logic and its application to planning, in: Proc. of the International Conference TABLEAUX 1998, Vol. 1397 of LNAI, Springer, 1998, pp. 124–140.

[6] S. Cerrito, M. Cialdea-Mayer, S. Praud, First-order linear temporal logic over finite time structures, in: H. Ganzinger, D. McAllester, A. Voronkov (Eds.), Proc. of the 6th International Conference on Logic for Programming, Artificial Intelligence and Reasoning, Vol. 1705 of LNAI, Springer, 1999, pp. 62–76.

[7] N. Chetcuti-Serandio, L. F. del Cerro, A mixed decision method for duration calculus, Journal of Logic and Computation 10 (2000) 877–895.

[8] M. D’Agostino, D. Gabbay, R. Hähnle, J. Posegga (Eds.), Handbook of Tableau Methods, Kluwer Academic Press, 1999.

[9] E. Emerson, J. Halpern, Decision procedures and expressiveness in the temporal logic of branching time, Journal of Computer and System Sciences 30 (1) (1985) 1–24.

[10] M. Fitting, Proof Methods for Modal and Intuitionistic Logics, Vol. 169, D. Reidel, Dordrecht, Holland, 1983.

[11] V. Goranko, A. Montanari, G. Sciavicco, Propositional interval neighborhood temporal logics, Journal of Universal Computer Science 9 (9) (2003) 1137–1167.

[12] V. Goranko, A. Montanari, G. Sciavicco, A general tableau method for propositional interval temporal logics, in: Proc. of the International Conference TABLEAUX 2003, Vol. 2796 of LNAI, Springer, 2003, pp. 102–116.

[13] V. Goranko, A. Montanari, G. Sciavicco, A road map of interval temporal logics and duration calculi, Journal of Applied Non-Classical Logics 14 (1-2) (2004) 9–54.

[14] E. Grädel, C. Hirsch, M. Otto, Back and forth between guarded and modal logics, ACM Transactions on Computational Logics 3 (3) (2002) 418–463.

[15] U. Hustadt and R.A. Schmidt, MSPASS: Modal Reasoning by Translation and First-Order Resolution, in: Proc. of the International Conference TABLEAUX 2000, Vol. 1847 of LNAI, Springer, 2000, pp. 67–71.

[16] R. Hähnle, O. Ibens, Improving temporal logic tableaux using integer constraints, in: Proc. of the 1st International Conference on Temporal Logic, Vol. 827 of LNCS, Springer, 1994, pp. 535–539.

[17] J. Halpern, Y. Shoham, A propositional modal logic of time intervals, Journal of the ACM 38 (4) (1991) 935–962.

[18] Y. Kesten, Z. Manna, H. McGuire, A. Pnueli, A decision algorithm for full propositional temporal logic, in: Proc. of the 5th International Conference on Computer Aided Verification, 1993, pp. 97–109.

[19] S. Kono, A combination of clausal and non-clausal temporal logic programs, in: M. Fisher, R. Owens (Eds.), Executable Modal and Temporal Logics, Vol. 897 of LNCS, Springer, 1995, pp. 40–57.

[20] R. Kontchakov, C. Lutz, F. Wolter, M. Zakharyaschev, Temporalizing tableaux, Studia Logica 76 (1) (2004) 91–134.

[21] M. Marx, Y. Venema, Multi-Dimensional Modal Logics, Kluwer Academic Press, 1997.

[22] B. Moszkowski, Reasoning about digital circuits, Ph.D. thesis, Department of Computer Science, Stanford University, Technical Report STAN-CS-83-970, Stanford, CA, 1983.

[23] L. Nguyen, Analytic tableau systems for propositional bimodal logics of knowledge and belief, in: Proc. of the International Conference TABLEAUX 2002, Vol. 2381 of LNAI, Springer, 2002, pp. 206–220.

[24] L. Paulson, A generic tableau prover and its integration with Isabelle, Journal of Universal Computer Science 5 (3) (1999) 73–87.

[25] A. Pnueli, R. Sherman, Semantic tableau for temporal logic, Technical Report CS81–82, The Weizmann Institute, 1981.

[26] O. Lichtenstein, A. Pnueli, Propositional temporal logic: Decidability and completeness, Logic Journal of the IGPL 8 (1) (2000) 55–85.

[27] W. Rautenberg, Modal tableau calculi and interpolation, Journal of Philosophical Logics 12 (1983) 403–423.

[28] P. Sala, Tableau Systems for Interval Temporal Logics (in Italian), Tesi di Laurea in Informatica, Università di Udine, 2003.

[29] P. Schmitt, J. Goubault-Larrecq, A tableau system for linear-time temporal logic, in: E. Brinksma (Ed.), Proc. of the 3rd Workshop on Tools and Algorithms for the Construction and Analysis of Systems, Vol. 1217 of LNCS, Springer, 1997, pp. 130–144.

[30] P. Schmitt, J. Goubault-Larrecq, A tableau system for full linear temporal logic. Draft, available at: *http://www.dyade.fr/fr/actions/vip/jgl/ltl2.ps.gz*, Technical Report, 1997.

[31] A. Sistla, E. Clarke, The complexity of propositional linear time temporal logics, Journal of the ACM 32 (3) (1985) 733–749.

[32] Y. Venema, A modal logic for chopping intervals, Journal of Logic and Computation 1 (4) (1991) 453–476.

[33] P. Wolper, The tableau method for temporal logic: An overview, Logique et Analyse 28 (1985) 119–136.

[34] M. Wooldridge, C. Dixon, M. Fisher, A tableau-based proof method for temporal logics of knowledge and belief, Journal of Applied Non-Classical Logics 8 (3) (1998) 225–258.

[35] C. Zhou, C. Hoare, A. P. Ravn, A calculus of durations, Information Processing Letters 40 (5) (1991) 269–276.

[36] C. Zhou, M. R. Hansen, An adequate first order interval logic, in: W. de Roever, H. Langmaak, A. Pnueli (Eds.), Compositionality: the Significant Difference, Vol. 1536 of LNCS, Springer, 1998, pp. 584–608.

Deduction Chains for Common Knowledge

Mathis Kretz [a,1] Thomas Studer [a]

[a] *Institut für Informatik und angewandte Mathematik, Universität Bern, Neubrückstrasse 10, CH-3012 Bern, Switzerland*

Abstract

Deduction chains represent a syntactic and in a certain sense constructive method for proving completeness of a formal system. Given a formula ϕ, the deduction chains of ϕ are built up by systematically decomposing ϕ into its subformulae. In the case where ϕ is a valid formula, the decomposition yields a (usually cut-free) proof of ϕ. If ϕ is not valid, the decomposition produces a countermodel for ϕ. In the current paper, we extend this technique to a semiformal system for the Logic of Common Knowledge. The presence of fixed point constructs in this logic leads to potentially infinite-length deduction chains of a non-valid formula, in which case fairness of decomposition requires special attention. An adequate order of decomposition also plays an important role in the reconstruction of the proof of a valid formula from the set of its deduction chains.

Key words: Deduction Chains, Logics of Common Knowledge, Tait-style Calculi

1 Introduction

Modal logic may be employed to reason about knowledge. A necessity for this arises for example when modeling systems of distributed agents, say computers connected over a network. In this setting, an agent knowing some proposition ϕ in state s is usually understood as ϕ holding in all states reachable from s in one step and thus each agent's knowledge may be modeled using a respective box operator. Furthermore, through arbitrary nesting of boxes epistemic situations of considerable complexity become expressible. However, it is well known that any formula of modal logic can only talk about a finite portion of a model and that this is not sufficient to express certain epistemic situations of particular interest. One such example often encountered in problems of coordination and agreement is common

Email addresses: kretz@iam.unibe.ch (Mathis Kretz), tstuder@iam.unibe.ch (Thomas Studer).

[1] Research supported by the Swiss National Science Foundation

knowledge of a proposition ϕ, which can roughly be viewed as the infinitary conjunction "all agents know ϕ and all agents know that all agents know ϕ and ...". In order to express common knowledge in the setting of modal logic, a fixed point extension is required, yielding the so called Logic of Common Knowledge which was introduced in [5] and studied extensively from a model-theoretic point of view in [3]. A more proof-theoretic study of this logic is given in [1] and [2].

In the current study we aim to deepen the proof-theoretic understanding of Logic of Common Knowledge by giving an alternative completeness proof for an infinitary proof system for this logic using the method of deduction chains. Deduction chains represent a syntactic and in a certain sense constructive method for proving completeness of a formal system. Given a formula ϕ, the deduction chains of ϕ are built up by systematically decomposing ϕ into its subformulae. In the case where ϕ is a valid formula, the decomposition yields a (usually cut-free) proof of ϕ. If ϕ is not valid, the decomposition produces a countermodel for ϕ. The method of deduction chains was first introduced by Schütte in [9,11] and has been used mainly in the proof-theory of systems of first and second order arithmetic. See [6,8] for applications of the method in this field. In [10] Schütte extends deduction chains to modal logic and we extend this approach again to accommodate fixed-point constructs. The main additional difficulty is that the presence of fixed-points requires a fully deterministic procedure for the decomposition of a given formula in order to guarantee fairness in the case of an infinite deduction chain.

We begin our account by giving an introduction to the syntax and semantics of Logic of Common Knowledge. In particular we will state the infinitary proof system $\mathsf{T}^{\omega}_{\mathsf{KC}_n}$, the completeness of which will be the main goal. In Section 3 we introduce the concept of deduction chains for formulae of Logic of Common Knowledge and prove some crucial properties required for the subsequent argument, chiefly fairness and saturation. We then proceed to prove the so called *principal semantic lemma*, which represents one half of the deduction chain argument. The principal semantic lemma secures the construction of a countermodel in case of an infinite deduction chain. Section 5 takes care of the other half of the argument, the so called *principal syntactic lemma* which yields the construction of a proof from the set of all deduction chains of a formula, if all of these chains are finite. Completeness is then obtained as a corollary to the two principal lemmata. In the concluding section we give a short overview of the main completeness argument.

2 Syntax and semantics

The language $\mathcal{L}^n_{\mathsf{C}}$ for Logic of Common Knowledge comprises a set of *atomic propositions* $\mathsf{p}, \mathsf{q}, \ldots$, the *propositional connectives* $\wedge$ and $\vee$, the *epistemic operators* $\mathsf{K}_1, \mathsf{K}_2, \ldots, \mathsf{K}_n$ and the *common knowledge operator* C. Additionally, we assume there is an auxiliary symbol $\sim$ to form complements of atomic propositions

and dual epistemic operators. The formulae $\alpha, \beta, \gamma, \ldots$ (possibly with subscripts) of $\mathcal{L}_{\mathsf{C}}^n$ are defined inductively as follows.

(1) All atomic propositions p and their complements $\sim\mathsf{p}$ are $\mathcal{L}_{\mathsf{C}}^n$ formulae.
(2) If α and β are $\mathcal{L}_{\mathsf{C}}^n$ formulae, so are $(\alpha \vee \beta)$ and $(\alpha \wedge \beta)$.
(3) If α is an $\mathcal{L}_{\mathsf{C}}^n$ formula, so are $\mathsf{K}_i\alpha$ and $\sim\mathsf{K}_i\alpha$.
(4) If α is an $\mathcal{L}_{\mathsf{C}}^n$ formula, so are $\mathsf{C}\alpha$ and $\sim\mathsf{C}\alpha$.

Often we omit parentheses if there is no possible confusion. We can define the *negation* $\neg\alpha$ of general $\mathcal{L}_{\mathsf{C}}^n$ formulae α by making use of de Morgan's laws and the law of double negation.

(1) If α is the atomic proposition p, then $\neg\alpha$ is $\sim\alpha$; if α is the formula $\sim\mathsf{p}$, then $\neg\alpha$ is p.
(2) If α is the formula $(\beta \vee \gamma)$, then $\neg\alpha$ is $(\neg\beta \wedge \neg\gamma)$; if α is the formula $(\beta \wedge \gamma)$, then $\neg\alpha$ is $(\neg\beta \vee \neg\gamma)$.
(3) If α is the formula $\mathsf{K}_i\beta$, then $\neg\alpha$ is $\sim\mathsf{K}_i(\neg\alpha)$; if α is the formula $\sim\mathsf{K}_i\beta$, then $\neg\alpha$ is $\mathsf{K}_i(\neg\alpha)$;
(4) If α is the formula $\mathsf{C}\beta$, then $\neg\alpha$ is $\sim\mathsf{C}(\neg\alpha)$; if α is the formula $\sim\mathsf{C}\beta$, then $\neg\alpha$ is $\mathsf{C}(\neg\alpha)$;

We set

$$\mathsf{E}\alpha := \mathsf{K}_1\alpha \wedge \cdots \wedge \mathsf{K}_n\alpha.$$

The formula $\mathsf{K}_i\alpha$ can be interpreted as "agent i knows that α". Thus $\mathsf{E}\alpha$ means "everybody knows that α". We will also need iterations $\mathsf{E}^m\alpha$ for all natural numbers m, formally defined by

$$\mathsf{E}^0\alpha := \top,\ \mathsf{E}^1\alpha := \mathsf{E}\alpha \text{ and } \mathsf{E}^{m+1}\alpha := \mathsf{E}\mathsf{E}^m\alpha,$$

where $\top$ is taken to refer to some trivially valid formula as for example $\mathsf{p} \vee \sim \mathsf{p}$ where p is an atomic proposition.

The semantics for logics of common knowledge is given by *Kripke structures*

$$\mathcal{M} = (S, \mathcal{K}_1, \ldots, \mathcal{K}_n, \pi)$$

where S is a non-empty set of *worlds*, $\mathcal{K}_1, \ldots, \mathcal{K}_n$ are binary relations on S and π is a *valuation function* assigning to each atomic proposition a subset of S. We say w is a world of $\mathcal{M} = (S, \mathcal{K}_1, \ldots, \mathcal{K}_n, \pi)$, expressed by $w \in \mathcal{M}$, if w is an element of S. The truth set $\|\alpha\|^{\mathcal{M}}$ of an $\mathcal{L}_{\mathsf{C}}^n$ formula α with respect to the Kripke structure $\mathcal{M} = (S, \mathcal{K}_1, \ldots, \mathcal{K}_n, \pi)$ is defined by induction on the complexity of α:

$$
\begin{aligned}
\|\mathsf{p}\|^{\mathcal{M}} &:= \pi(\mathsf{p}) \\
\| \sim\mathsf{p}\|^{\mathcal{M}} &:= S \setminus \|\mathsf{p}\|^{\mathcal{M}}, \\
\|\alpha \vee \beta\|^{\mathcal{M}} &:= \|\alpha\|^{\mathcal{M}} \cup \|\beta\|^{\mathcal{M}}, \\
\|\alpha \wedge \beta\|^{\mathcal{M}} &:= \|\alpha\|^{\mathcal{M}} \cap \|\beta\|^{\mathcal{M}}, \\
\|\mathsf{K}_i\alpha\|^{\mathcal{M}} &:= \{v \in S : w \in \|\alpha\|^{\mathcal{M}} \text{ for all } w \text{ with } (v,w) \in \mathcal{K}_i\}, \\
\| \sim\mathsf{K}_i\alpha\|^{\mathcal{M}} &:= S \setminus \|\mathsf{K}_i\neg\alpha\|^{\mathcal{M}}, \\
\|\mathsf{C}\alpha\|^{\mathcal{M}} &:= \bigcap\{\|\mathsf{E}^m\alpha\|^{\mathcal{M}} : m \geq 1\}, \\
\| \sim\mathsf{C}\alpha\|^{\mathcal{M}} &:= S \setminus \|\mathsf{C}\neg\alpha\|^{\mathcal{M}}.
\end{aligned}
$$

Using these truth sets, we can express that a formula α is *valid* in a world w of a Kripke structure $\mathcal{M}$. This is the case if $w \in \|\alpha\|^{\mathcal{M}}$. We will employ the following notation:

$$\mathcal{M}, w \models \alpha \ :\Longleftrightarrow\ w \in \|\alpha\|^{\mathcal{M}}.$$

Next, we are going to present the semiformal Tait-style calculus $\mathsf{T}^{\omega}_{\mathsf{KC}_n}$ for common knowledge. Tait-style calculi [12,14] are one-sided Gentzen calculi which derive finite sets of formulae. This kind of calculi is particularly well-suited for the study of cut-elimination and meta-mathematical investigations. $\mathsf{T}^{\omega}_{\mathsf{KC}_n}$ has been introduced by Alberucci and Jäger [1,2]. It incorporates an analogue of the ω rule which permits the derivation of the formula $\mathsf{C}\alpha$ from the infinitely many premises

$$\mathsf{E}^1\alpha, \mathsf{E}^2\alpha, \ldots, \mathsf{E}^m\alpha, \ldots$$

for all natural numbers $m \geq 1$. The system $\mathsf{T}^{\omega}_{\mathsf{KC}_n}$ is called *semiformal* since, as opposed to formal systems, it has basic inferences with infinitely many premises [11].

The system $\mathsf{T}^{\omega}_{\mathsf{KC}_n}$ derives finite sets of $\mathcal{L}^n_{\mathsf{C}}$ formulae which are denoted by $\Gamma, \Delta, \Sigma, \Pi, \ldots$ (possibly with subscripts). Usually we will write for example $\alpha, \beta, \Delta, \Gamma$ for the union $\{\alpha, \beta\} \cup \Delta \cup \Gamma$. Moreover, if Γ is the set $\{\alpha_1, \ldots, \alpha_m\}$, then we use the following abbreviations:

$$
\begin{aligned}
\bigvee\Gamma &:= \alpha_1 \vee \cdots \vee \alpha_m, \\
\neg\Gamma &:= \{\neg\alpha_1, \ldots, \neg\alpha_m\}, \\
\neg\mathsf{K}_i\Gamma &:= \{\neg\mathsf{K}_i\alpha_1, \ldots, \neg\mathsf{K}_i\alpha_m\}, \\
\neg\mathsf{C}\Gamma &:= \{\neg\mathsf{C}\alpha_1, \ldots, \neg\mathsf{C}\alpha_m\}.
\end{aligned}
$$

The axioms and rules of $\mathsf{T}^{\omega}_{\mathsf{KC}_n}$ consist of the usual propositional axioms and rules of Tait calculi, rules for the epistemic operators K_i with additional side formulae $\neg\mathsf{C}\Delta$ plus rules dealing with common knowledge. Note that $\mathsf{T}^{\omega}_{\mathsf{KC}_n}$ includes neither an induction rule nor a cut rule.

Definition 2.1 *The infinitary Tait-style calculus $\mathsf{T}^{\omega}_{\mathsf{KC}_n}$ over the language $\mathcal{L}^n_{\mathsf{C}}$ is defined by the following axioms and inference rules:*

$$\Gamma, \mathsf{p}, \neg\mathsf{p} \quad (\mathit{ID})$$

$$\frac{\Gamma, \alpha, \beta}{\Gamma, \alpha \vee \beta} \quad (\vee) \qquad \frac{\Gamma, \alpha \quad \Gamma, \beta}{\Gamma, \alpha \wedge \beta} \quad (\wedge)$$

$$\frac{\neg\mathsf{C}\Delta, \neg\Gamma, \alpha}{\neg\mathsf{C}\Delta, \neg\mathsf{K}_i\Gamma, \mathsf{K}_i\alpha, \Sigma} \quad (\mathsf{K}_i)$$

$$\frac{\Gamma, \neg\mathsf{E}\alpha}{\Gamma, \neg\mathsf{C}\alpha} \quad (\neg\mathsf{C}) \qquad \frac{\Gamma, \mathsf{E}^k\alpha \textit{ for all } k \in \omega}{\Gamma, \mathsf{C}\alpha} \quad (\mathsf{C}^\omega)$$

The infinitary system $\mathsf{T}^\omega_{\mathsf{K}^\mathsf{C}_n}$ is formulated over the finitary language $\mathcal{L}^n_\mathsf{C}$ and derives finite sets of formulae. It is infinitary only because of the rule (C^ω) for introducing common knowledge. This rule has infinitely many premises and thus may give rise to infinite proof trees. For arbitrary ordinals α and finite sets Γ of $\mathcal{L}^n_\mathsf{C}$ formulae we define the derivability relation $\mathsf{T}^\omega_{\mathsf{K}^\mathsf{C}_n} \vdash_\alpha \Gamma$ as usual by induction on α.

(1) If Γ is an axiom of $\mathsf{T}^\omega_{\mathsf{K}^\mathsf{C}_n}$, then we have $\mathsf{T}^\omega_{\mathsf{K}^\mathsf{C}_n} \vdash_\alpha \Gamma$ for all ordinals α.
(2) If $\mathsf{T}^\omega_{\mathsf{K}^\mathsf{C}_n} \vdash_{\alpha'_i} \Gamma_i$ and $\alpha'_i < \alpha$ for all premises of a rule of $\mathsf{T}^\omega_{\mathsf{K}^\mathsf{C}_n}$, then we have $\mathsf{T}^\omega_{\mathsf{K}^\mathsf{C}_n} \vdash_\alpha \Gamma$ for the conclusion Γ of this rule.

We will write $\mathsf{T}^\omega_{\mathsf{K}^\mathsf{C}_n} \vdash \Gamma$ if $\mathsf{T}^\omega_{\mathsf{K}^\mathsf{C}_n} \vdash_\alpha \Gamma$ for some ordinal α.

Now we have to mention some structural properties of $\mathsf{T}^\omega_{\mathsf{K}^\mathsf{C}_n}$ which will be important in the sequel. The first two, weakening and inversion, are easily shown by induction on the length of the involved derivations.

Lemma 2.2 (Weakening) *If* $\mathsf{T}^\omega_{\mathsf{K}^\mathsf{C}_n} \vdash_\alpha \Gamma$ *and* $\Gamma \subset \Gamma'$*, then also* $\mathsf{T}^\omega_{\mathsf{K}^\mathsf{C}_n} \vdash_\alpha \Gamma'$.

Lemma 2.3 (Inversion)

(1) If $\mathsf{T}^\omega_{\mathsf{K}^\mathsf{C}_n} \vdash_\alpha \Gamma, \phi_1 \wedge \phi_2$*, then* $\mathsf{T}^\omega_{\mathsf{K}^\mathsf{C}_n} \vdash_\alpha \Gamma, \phi_1$ *and* $\mathsf{T}^\omega_{\mathsf{K}^\mathsf{C}_n} \vdash_\alpha \Gamma, \phi_2$.
(2) If $\mathsf{T}^\omega_{\mathsf{K}^\mathsf{C}_n} \vdash_\alpha \Gamma, \phi_1 \vee \phi_2$*, then* $\mathsf{T}^\omega_{\mathsf{K}^\mathsf{C}_n} \vdash_\alpha \Gamma, \phi_1, \phi_2$.
(3) If $\mathsf{T}^\omega_{\mathsf{K}^\mathsf{C}_n} \vdash_\alpha \Gamma, \mathsf{C}\phi$*, then* $\mathsf{T}^\omega_{\mathsf{K}^\mathsf{C}_n} \vdash_\alpha \Gamma, \mathsf{E}^k\phi$ *for every* $k \in \omega$.

Lemma 2.4 *If* $\mathsf{T}^\omega_{\mathsf{K}^\mathsf{C}_n} \vdash_\alpha \Gamma, \neg\mathsf{E}^k\phi$ *for some* $k \in \omega$*, then* $\mathsf{T}^\omega_{\mathsf{K}^\mathsf{C}_n} \vdash_{\alpha+1} \Gamma, \neg\mathsf{C}\phi$.

PROOF. We proceed by induction on k. The base case of $k = 1$ holds directly by the rule $(\neg\mathsf{C})$. We thus assume $\mathsf{T}^\omega_{\mathsf{K}^\mathsf{C}_n} \vdash_\alpha \Gamma, \neg\mathsf{E}^{k+1}\phi$, which by iteration of Lemma 2.3 means

$$\mathsf{T}^\omega_{\mathsf{K}^\mathsf{C}_n} \vdash_\alpha \Gamma, \neg\mathsf{K}_1\mathsf{E}^k\phi, \ldots, \neg\mathsf{K}_n\mathsf{E}^k\phi \tag{1}$$

and show $\mathsf{T}^{\omega}_{\mathsf{KC}_n} \vdash\!\!\!\!-_{\alpha+1} \Gamma, \neg\mathsf{C}\phi$ by induction on length α of the proof. The case of $\alpha = 0$ is trivial, thus assume that the claim holds for all $\alpha' < \alpha$. We make a case distinction as to the last rule applied to derive (1).

Case 1) The last rule was (K_i) for some $1 \leq i \leq n$: Then there is a formula $\mathsf{K}_i\xi \in \Gamma$ such that $\mathsf{T}^{\omega}_{\mathsf{KC}_n} \vdash\!\!\!\!-_{\alpha} \neg\mathsf{C}\Delta_1, \neg\mathsf{K}_i\Delta_2, \mathsf{K}_i\xi, \Sigma$ and $\neg\mathsf{K}_j\mathsf{E}^k\phi \in \Sigma$ for all $j \neq i$. If we also have $\neg\mathsf{K}_i\mathsf{E}^k\phi \in \Sigma$, then the claim is trivial. Otherwise we must have $\neg\mathsf{K}_i\mathsf{E}^k\phi \in \neg\mathsf{K}_i\Delta_2$ and by the premise of (K_i)

$$\mathsf{T}^{\omega}_{\mathsf{KC}_n} \vdash\!\!\!\!-_{\alpha'} \neg\mathsf{C}\Delta_1, \neg\Delta_2, \xi,$$

where $\alpha' < \alpha$ and $\neg\mathsf{E}^k\phi \in \neg\Delta_2$. By the hypothesis of the outer induction $\mathsf{T}^{\omega}_{\mathsf{KC}_n} \vdash\!\!\!\!-_{\alpha} \neg\mathsf{C}\Delta_1, \neg\mathsf{C}\phi, \neg\Delta_2', \xi$, where $\neg\Delta_2' = \neg\Delta_2 \setminus \{\neg\mathsf{E}^k\phi\}$. Therefore, applying (K_i) yields $\mathsf{T}^{\omega}_{\mathsf{KC}_n} \vdash\!\!\!\!-_{\alpha+1} \neg\mathsf{C}\Delta_1, \neg\mathsf{C}\phi, \neg\mathsf{K}_i\Delta_2', \mathsf{K}_i\xi, \Sigma$, meaning $\mathsf{T}^{\omega}_{\mathsf{KC}_n} \vdash\!\!\!\!-_{\alpha+1} \Gamma, \neg\mathsf{C}\phi$.

Case 2) The last rule was not (K_i) for any $1 \leq i \leq n$: In this case the claim follows directly by applying the hypothesis of the inner induction to the premise of the respective rule. □

Transfinite induction on the length of derivations yields the correctness of $\mathsf{T}^{\omega}_{\mathsf{KC}_n}$ with respect to the semantics for logics of common knowledge. That is we have the following theorem.

Theorem 2.5 *For all finite sets Γ of $\mathcal{L}^n_{\mathsf{C}}$ formulae, all Kripke structures $\mathcal{M}$ and all worlds $w \in \mathcal{M}$ we have that*

$$\mathsf{T}^{\omega}_{\mathsf{KC}_n} \vdash \Gamma \implies \mathcal{M}, w \models \bigvee \Gamma.$$

3 Deduction chains

In this section we are going to define the notion of deduction chain in the context of $\mathsf{T}^{\omega}_{\mathsf{KC}_n}$. Schütte [9] originally introduced deduction chains for classical logic. Later, he showed in [10] how to extend this technique to the case of intuitionistic and modal logics. We adapt his method and apply it to show completeness of our infinitary fixed point logic.

In the sequel we will make use of the following notation for projections. If a is a tuple (x, y), then $a_1 := x$ and $a_2 := y$.

We start by defining labeled index trees. Such trees will provide the frame on which the countermodel of a non-valid formula ψ is based. The set of worlds will consist of all nodes of the labeled index trees of a deduction chain for ψ. The accessibility relation for agent i will be given the successor relation σ_i.

Definition 3.1 *A labeled index tree is a set I of pairs (k, α), where k is in $\{0, \ldots, n\}$ and α is a sequence of natural numbers such that I has the following properties*

(1) $(0, (0)) \in I$
(2) For every $m \in \omega$ we have that

$$(k, (\alpha, m+1)) \in I \text{ for some } k \in \{1, \ldots, n\}$$

implies

$$(l, (\alpha, m)) \in I \text{ for some } l \in \{1, \ldots, n\}.$$

(3) If there exists a $k \in \{1, \ldots, n\}$ with $(k, (\alpha, 0)) \in I$, then there exists an $l \in \{1, \ldots, n\}$ such that $(l, \alpha) \in I$
(4) If $(k, \alpha) \in I$ and $(l, \alpha) \in I$, then $k = l$.

Definition 3.2 *Let I be a labeled index tree and $a, b \in I$. We define the following binary relations on I:*

$$\begin{aligned}
a = b &:\Leftrightarrow a_2 = b_2 \\
a \sigma_i b &:\Leftrightarrow a = (j, \alpha) \text{ and } b = (i, (\alpha, l)) \\
&\qquad \text{for some sequence } \alpha,\ j \in \{1, \ldots, n\} \text{ and } l \in \omega \\
a \prec b &:\Leftrightarrow a_2 \text{ is a prefix of } b_2 \\
a \preccurlyeq b &:\Leftrightarrow a = b \text{ or } a \prec b \\
a \sqsubset b &:\Leftrightarrow (a \prec b) \text{ or} \\
&\qquad (a_2 = (\alpha, l) \text{ and } b_2 = (\alpha, k) \text{ and } l < k)
\end{aligned}$$

Definition 3.3 *A literal is a formula of the form p or $\sim \mathsf{p}$ where p is an atomic formula. A formula ϕ is reducible if it is not a literal.*

A deduction chain for a formula ϕ is built by decomposing ϕ. It is crucial for our argument that this decomposition satisfies certain fairness conditions. In particular, formulae of the form $\sim \mathsf{C}\alpha$ need special care. When we treat such a formula for the first time, we create a new formula $\neg \mathsf{E}^1 \alpha$. When we deal with it for the second time, then we create $\neg \mathsf{E}^2 \alpha$ and so on. Moreover, if there is another formula $\sim \mathsf{C}\beta$, we have to pay attention that we consider $\sim \mathsf{C}\alpha$ and $\sim \mathsf{C}\beta$ in alternation. In order to guarantee this, we need some bookkeeping which is achieved using so-called iteration histories.

Definition 3.4 *Let $\mathcal{L}^n_{\mathsf{C}}|_{\neg \mathsf{C}}$ denote the set of all formulae of the language $\mathcal{L}^n_{\mathsf{C}}$ which have the form $\sim \mathsf{C}\beta$ for some $\beta \in \mathcal{L}^n_{\mathsf{C}}$. An iteration history is a finite set $E \subset \mathcal{L}^n_{\mathsf{C}}|_{\neg \mathsf{C}} \times \omega \times \omega$ such that for any $e, f \in E$, we have $e = f$ if $e_1 = f_1$.*

Definition 3.5 *Given an iteration history E, we define*

$$dom_E := \{\alpha \in \mathcal{L}^n_{\mathsf{C}}|_{\neg \mathsf{C}}; \exists e \in E \text{ such that } e_1 = \alpha\}$$

Furthermore, for all $\alpha \in dom_E$ and $k \in \omega$ we define the following functions:

$$add_E(\sim \mathsf{C}\beta, k) = \begin{cases} E \cup \{(\sim \mathsf{C}\beta, k, 0)\} & \text{if } \sim \mathsf{C}\beta \notin dom_E \\ E & \text{otherwise} \end{cases}$$

$$lookup_E(\alpha) = (k, l) \text{ where } (\alpha, k, l) \in E$$

$$ord_E(\alpha) = (lookup_E(\alpha))_1$$

$$deg_E(\alpha) = (lookup_E(\alpha))_2$$

$$max_E = \begin{cases} \max\{ord_E(\beta); \beta \in dom_E\} & \text{if } dom_E \neq \emptyset \\ 0 & \text{otherwise} \end{cases}$$

$$min_E = \begin{cases} \min\{ord_E(\beta); \beta \in dom_E\} & \text{if } dom_E \neq \emptyset \\ 0 & \text{otherwise} \end{cases}$$

Definition 3.6 *A formula sequence S is an $n+2$-tuple $(\Gamma, \Delta_1, \ldots, \Delta_n, E)$, where Γ is a finite sequence of formulae of $\mathcal{L}^n_{\mathsf{C}}$, Δ_i are finite sequences of formulae of the form $\neg\alpha$, where $\alpha \in \mathcal{L}^n_{\mathsf{C}}$ and E is an iteration history. We will use ϵ to denote the empty sequence. The distinguished formula of S is the rightmost reducible formula appearing in Γ, if such a formula exists. For any finite sequence of formulae Λ, we denote by set(Λ) the set of all formulae appearing in Λ. We define set$(S) :=$ set$(\Gamma) \cup dom_E$,*

$$\textit{set}^+(S) := \textit{set}(S) \cup \{\sim \mathsf{K}_i\beta; \neg\beta \in \textit{set}(\Delta_1)\} \cup \ldots \cup \{\sim \mathsf{K}_n\beta; \neg\beta \in \textit{set}(\Delta_n)\},$$

$max_S := max_E$, $min_S := min_E$ and $dom_S := dom_E$. Further, for all formulae $\beta \in dom_S$ we set $ord_S(\beta) := ord_E(\beta)$. Let FS be the set of all formula sequences.

A sequence tree is a labeled index tree of formula sequences. That is we annotate each node of the index tree with a formula sequence. In the construction of a countermodel for a non-valid formula, the sequence at a node will be the basis for defining the valuation function π at that node. In particular, π will be defined such that if a formula ψ belongs to the annotation of a node, then ψ will not hold at that node.

Definition 3.7 *Let I be a labeled index tree. A sequence tree over I is a function*

$$\mathbf{R} : I \longrightarrow FS$$

We use the notation $\mathbf{R}_a$ for $\mathbf{R}(a)$, where $a \in I$ and define $max(\mathbf{R})$ as $max\{max_{\mathbf{R}_a}; a \in I\}$. Furthermore, given a formula α and an iteration history E we define the operation

$$it(\mathbf{R}, \alpha, E) = \begin{cases} (E \setminus \{(\alpha, k, l)\}) \cup \{(\alpha, max(\mathbf{R}) + 1, l + 1)\} & \text{if } \alpha \in dom_E \\ E & \text{otherwise} \end{cases}$$

Definition 3.8 *Let $\mathbf{R}$ be a sequence tree over I. Further, let J be the set $\{a \in I;$*

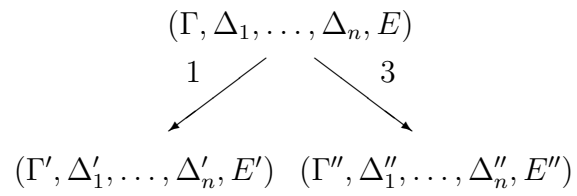


Fig. 1. A sequence tree

$dom_{\mathbf{R}_a} \neq \emptyset\}$. *We define the relation* $\sqsubset^*$ *for all* $a, b \in J$ *as follows:*

$$a \sqsubset^* b \quad :\Leftrightarrow \quad min_{\mathbf{R}_a} < min_{\mathbf{R}_b} \text{ or } [min_{\mathbf{R}_a} = min_{\mathbf{R}_b} \text{ and } a \sqsubset b]$$

The redex of a sequence tree is the formula that will be decomposed next. It is basically found as follows. The rightmost reducible formula of the main sequence of a node a of $\mathbf{R}$ is called distinguished formula of $\mathbf{R}$ at a (see Definition 3.6). The redex of $\mathbf{R}$ is defined as the topmost distinguished formula if such a formula exists; otherwise as the formula of the form $\sim \mathsf{C}\alpha$ (if such a formula exists) which has to be treated next according to information given by the iteration histories. If neither of these two conditions apply, then $\mathbf{R}$ has no redex.

Definition 3.9 *Let* $\mathbf{R}$ *be a sequence tree over* I *and* $a \in I$*. A formula* ϕ *is called redex of* $\mathbf{R}$ *at* a *if one of the following two conditions holds:*

(1) ϕ *is the distinguished formula of* $\mathbf{R}_a$ *and* a *is* $\sqsubset$*-minimal among all* $b \in I$*.*
(2) there are no distinguished formulae in $\mathbf{R}$*,* $\phi \in dom_{\mathbf{R}_a}$*,* $ord_{\mathbf{R}_a}(\phi) = min_{\mathbf{R}_a}$ *and* a *is* $\sqsubset^*$*-minimal in* $\mathbf{R}$*.*

Note that for a sequence tree $\mathbf{R}$ over I there is at most one $a \in I$ and one formula ϕ such that ϕ is the redex of $\mathbf{R}$ at a.

Definition 3.10 *Let* α *be a formula,* $S = (\Gamma, \Delta_1, \ldots, \Delta_n, E)$ *a formula sequence in a sequence tree* $\mathbf{R}$ *and* Γ' *the sequence* α, Γ*. Define the operation*

$$\alpha \circ S = \begin{cases} S \text{ if } \alpha \text{ is already in } \Gamma, \\ (\Gamma', \Delta_1, \ldots, \Delta_n, E) \text{ if } \alpha \text{ not in } \Gamma \text{ and not of the form } \sim\mathsf{C}\beta, \\ (\Gamma, \Delta_1, \ldots, \Delta_n, add_E(\alpha, max(\mathbf{R}) + 1)) \text{ if } \alpha \text{ not in } \Gamma \text{ and} \\ \qquad\qquad\qquad\qquad\qquad\qquad \text{of the form } \sim\mathsf{C}\beta \end{cases}$$

Given a finite sequence $\Lambda = (\alpha_1, \alpha_2, \ldots, \alpha_n)$ *of formulae and a formula sequence* S*, we write* $\Lambda \circ S$ *for* $\alpha_1 \circ (\alpha 2 \circ (\ldots \circ (\alpha_n \circ S)))$

Definition 3.11 *A sequence tree* $\mathbf{R}$ *over* I *is called reducible, if* $\mathbf{R}$ *has a redex.* $\mathbf{R}$ *is called axiomatic if there exists an* $a \in I$ *and an atomic proposition* p*, such that* $\mathbf{R}_a = (\Gamma, \Delta_1, \ldots, \Delta_n, E)$ *and both* p *and* $\sim\mathsf{p}$ *appear in* Γ*. Generally, we say that a formula* α *appears in* $\mathbf{R}$ *at some* $a \in I$ *if* $\alpha \in set(\mathbf{R}_a)$*.*

A deduction chain is a sequence $\Theta_0, \Theta_1, \Theta_2, \ldots$ of sequence trees. If Θ_i is ax-

iomatic, then Θ_i is the last element of the deduction chain. Θ_i is also the last element of the deduction chain if it does not contain a redex. If Θ_i is not axiomatic and has a redex ψ at a, then ψ will be decomposed and a new sequence tree Θ_{i+1} is added to the deduction chain. Θ_{i+1} is obtained from Θ_i by removing ψ and adding

(1) ψ_1, ψ_2 at a if $\psi = \psi_1 \vee \psi_2$,

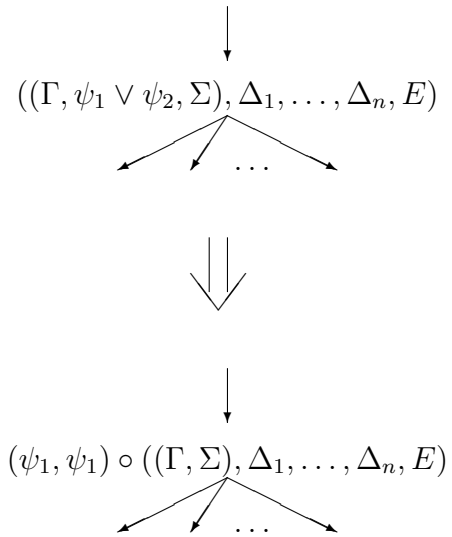


Fig. 2. Type 1 reduction

(2) ψ_1 or ψ_2 at a if $\psi = \psi_1 \wedge \psi_2$,

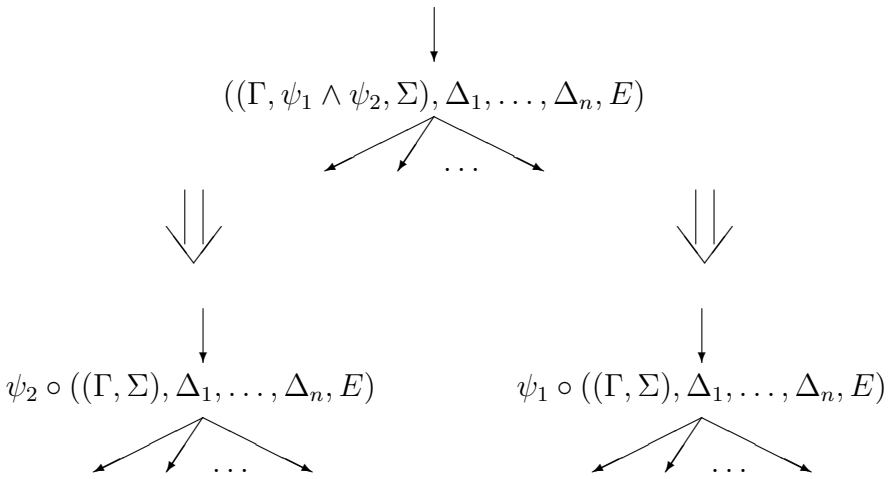


Fig. 3. Type 2 reduction

(3) $\neg\psi_1$ at every successor of a (and remembering $\neg\psi_1$ at a) if $\psi = {\sim}\mathsf{K}_i\psi_1$,

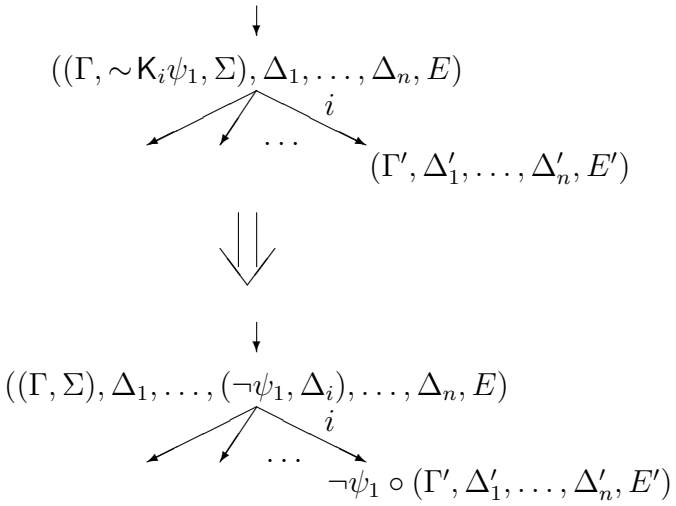


Fig. 4. Type 3 reduction

(4) a new successor of a initialized with ψ_1 (plus anything remembered at a) if $\psi = \mathsf{K}_i\psi_1$,

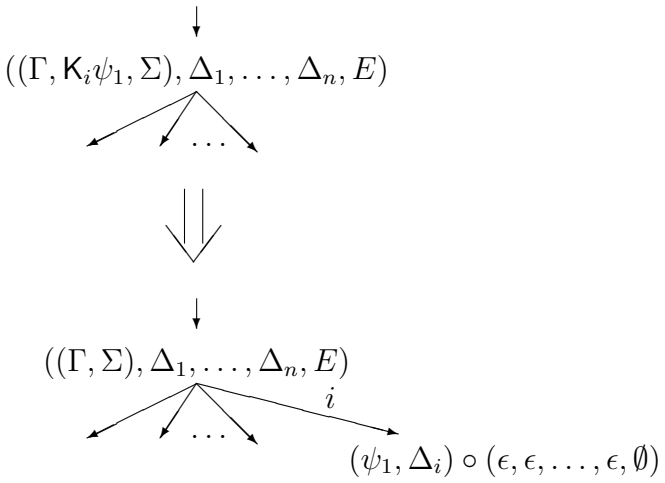


Fig. 5. Type 4 reduction

(5) $\mathsf{E}^k\psi_1$ at a for some k if $\psi = \mathsf{C}\psi_1$,

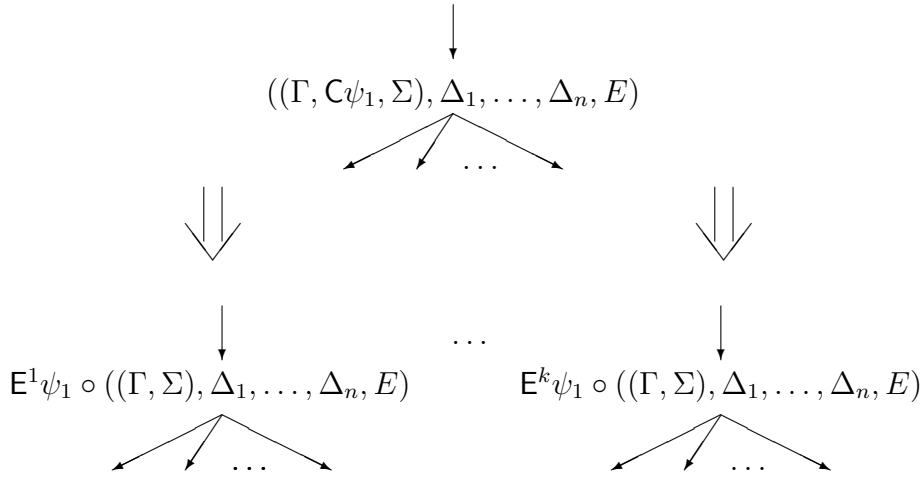
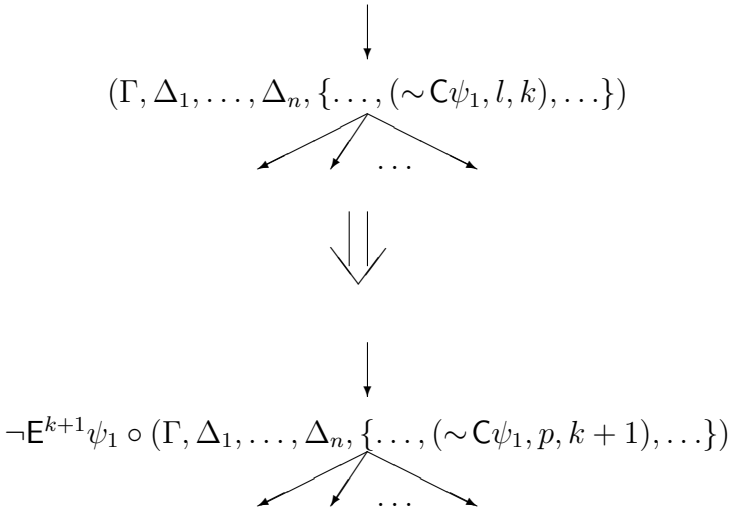


Fig. 6. Type 5 reduction

(6) $\neg\mathsf{E}^{k+1}\psi_1$ at a where k is the maximum number of iterations tried at a if $\psi = \sim \mathsf{C}\psi_1$.



where p is maximal for the whole sequence tree.

Fig. 7. Type 6 reduction

These six cases will be made precise in the next definition.

Definition 3.12 *Let* $\mathbf{R}$ *be a sequence tree. A deduction chain of* $\mathbf{R}$ *is a finite or infinite sequence*

$$\Theta_0, \Theta_1, \Theta_2, \ldots$$

of sequence trees with the following properties:

(1) $\Theta_0 = \mathbf{R}$
(2) If Θ_m *is axiomatic or not reducible, then* Θ_m *is the last element of the sequence.*

(3) If Θ_m is not axiomatic and reducible, then Θ_{m+1} is derived from Θ_m in the following manner:
Let Θ_m be the sequence tree $\mathbf{S}$ over index tree I and let ϕ be the redex of $\mathbf{S}$ at $a \in I$. If $\phi \notin \mathcal{L}^n_{\mathsf{C}}|_{\neg\mathsf{C}}$, then $\mathbf{S}_a = (\Gamma, \Delta_1, \ldots, \Delta_n, E)$ and $\Gamma = \Omega, \phi, \Omega'$, where Ω' is a sequence of non-reducible formulae.
Case 1: $\phi = \psi_1 \vee \psi_2$
Then Θ_{m+1} is the sequence tree $\mathbf{T}$ over I, where

$$\begin{aligned}\Gamma' &= \Omega, \Omega' \\ \mathbf{T}_a &= (\psi_1, \psi_2) \circ (\Gamma', \Delta_1, \ldots, \Delta_n, E) \\ \mathbf{T}_b &= \mathbf{S}_b \text{ for all other } b \in I\end{aligned}$$

In this case we say that Θ_m has type 1 successor Θ_{m+1}.
Case 2: $\phi = \psi_1 \wedge \psi_2$
Then Θ_{m+1} is the sequence tree $\mathbf{T}$ over I, where

$$\begin{aligned}\Gamma' &= \Omega, \Omega' \\ \mathbf{T}_a &= \psi_1 \circ (\Gamma', \Delta_1, \ldots, \Delta_n, E) \text{ or} \\ \mathbf{T}_a &= \psi_2 \circ (\Gamma', \Delta_1, \ldots, \Delta_n, E) \\ \mathbf{T}_b &= \mathbf{S}_b \text{ for all other } b \in I\end{aligned}$$

In this case we say that Θ_m has type 2 successor Θ_{m+1}.
Case 3: $\phi = {\sim}\mathsf{K}_i\psi$
Then Θ_{m+1} is the sequence tree $\mathbf{T}$ over I, where

$$\begin{aligned}\Gamma' &= \Omega, \Omega' \\ \Delta'_i &= \neg\psi, \Delta_i \\ \mathbf{T}_a &= (\Gamma', \Delta_1, \ldots, \Delta'_i, \ldots, \Delta_n, E)\end{aligned}$$

and for all $b \in I$ such that $a\sigma_i b$

$$\mathbf{T}_b = \neg\psi \circ \mathbf{S}_b$$

and $\mathbf{T}_c := \mathbf{S}_c$ for all other $c \in I$.
In this case we say that Θ_m has type 3 successor Θ_{m+1}.
Case 4: $\phi = \mathsf{K}_i\psi$
Let $a = (l, \alpha)$ and k be the smallest number such that $(j, (\alpha, k)) \notin I$ for any number j. Then Θ_{m+1} is the sequence tree $\mathbf{T}$ over $I \cup \{b\}$, where $b = (i, (\alpha, k))$ and

$$\begin{aligned}\Gamma' &= \Omega, \Omega' \\ \mathbf{T}_a &= (\Gamma', \Delta_1, \ldots, \Delta_n, E) \\ \mathbf{T}_b &= (\psi, \Delta_i) \circ (\epsilon, \epsilon, \ldots, \epsilon, \emptyset) \\ \mathbf{T}_c &= \mathbf{S}_c \text{ for all other } c \in I\end{aligned}$$

In this case we say that Θ_m has type 4 successor Θ_{m+1}.

Case 5: $\phi = \mathsf{C}\psi$

Then Θ_{m+1} *is the sequence tree* $\mathbf{T}$ *over* I*, where*

$$\begin{aligned}\Gamma' &= \Omega, \Omega' \\ \mathbf{T}_a &= \mathsf{E}^i\psi \circ (\Gamma', \Delta_1, \ldots, \Delta_n, E) \text{ for some } i \in \omega \\ \mathbf{T}_b &= \mathbf{S}_b \text{ for all other } b \in I\end{aligned}$$

In this case we say that Θ_m *has type 5 successor* Θ_{m+1}.

If $\phi \in \mathcal{L}^n_{\mathsf{C}}|_{\neg\mathsf{C}}$*, then we proceed as follows:*

Case 6: $\phi = \sim\mathsf{C}\psi$ *Then* Θ_{m+1} *is the sequence tree* $\mathbf{T}$ *over* I*, where*

$$\begin{aligned}\mathbf{T}_a &= \neg\mathsf{E}^k\psi \circ (\Gamma, \Delta_1, \ldots, \Delta_n, it(\Theta_m, \sim\mathsf{C}\psi, E)) \\ &\qquad\qquad \text{where } k = deg_E(\sim\mathsf{C}\psi) + 1 \\ \mathbf{T}_b &= \mathbf{S}_b \text{ for all other } b \in I\end{aligned}$$

In this case we say that Θ_m *has type 6 successor* Θ_{m+1}.

Definition 3.13 *Let* ϕ *be an* $\mathcal{L}^n_{\mathsf{C}}$ *formula. A deduction chain of* ϕ *is a deduction chain of the sequence tree* $\mathbf{R}$ *which is given by the function mapping the index tree* $\{(0,(0)\}$ *to the formula sequence* $\phi \circ (\epsilon, \epsilon, \ldots, \epsilon, \emptyset)$.

4 Principal semantic lemma

The principal semantic lemma states that if there exists a deduction chain of a formula ψ which is infinite or ends in a non-axiomatic sequence tree, then there exists a countermodel for ψ. For this section we assume $\Theta_0, \Theta_1, \Theta_2, \ldots$ is such a deduction chain and we let $I_0, I_1, I_2, \ldots$ be the respective labeled index trees.

The Kripke structure K_Θ that will serve as countermodel is (roughly) constructed as $\Theta_0 \cup \Theta_1 \cup \Theta_2 \cup \ldots$ where $\pi(p) = \{a; \neg p \text{ appers at node } a\}$. Fairness in the construction of the deduction chain ensures that if $\phi \in a$, then $\mathsf{K}_\Theta, a \not\models \phi$. Finally we observe that ψ is an element of the root of K_Θ.

The following three lemmata follow directly from the definition of deduction chain.

Lemma 4.1 *If a literal* α *appears in* Θ_i *at* $a \in I_i$*, then* α *also appears in every* Θ_j *at* $a \in I_j$ *for* $j \geq i$.

Lemma 4.2 *For every* Θ_i *we have: There does not exist an* $a \in I_i$ *such that for some atomic formula* p *both* p *and* $\sim\mathsf{p}$ *appear in* Θ_i *at* a.

Lemma 4.3 *For each* Θ_k *there exists an* $l \geq k$*, such that* Θ_l *has no distinguished formulae.*

Lemma 4.4 *If $\mathbf{R} = \Theta_k$, $\sim \mathsf{C}\beta$ appears in $\mathbf{R}$ at a and $ord_{\mathbf{R}_a}(\sim \mathsf{C}\beta)$ is minimal in $\mathbf{R}$, then there exists an $l \geq k$, such that $\sim \mathsf{C}\beta$ is the redex of Θ_l at a.*

PROOF. By definition of deduction chains and the operations it and $\circ$ there can only be one formula and one $a \in I_k$, such that $ord_{\mathbf{R}_a}(\sim \mathsf{C}\beta)$ is minimal in Θ_k. By Lemma 4.3 there exists an $l \geq k$, such that Θ_l has no distinguished formulae. Then a is $\sqsubset^*$-minimal in Θ_l and so $\sim \mathsf{C}\beta$ is the redex of Θ_l at a. □

Lemma 4.5 *For every Θ_k and $m \geq 0$ there exists an $l \geq k$, such that the (finite) set*

$$d_{\Theta_l}(m) := \{(\sim \mathsf{C}\beta, a); \sim \mathsf{C}\beta \textit{ appears in } \Theta_l \textit{ at } a \textit{ and } ord_{(\Theta_l)_a}(\sim \mathsf{C}\beta) \leq m\}$$

is empty.

PROOF. The claim is trivial if Θ_k does not contain any formulae of the form $\sim \mathsf{C}\alpha$. We thus assume otherwise and prove the claim by induction on m.

$\underline{m = 0}$: The set $d_{\Theta_k}(0)$ can only contain a pair $(\sim \mathsf{C}\beta, a)$, where we have $ord_{(\Theta_k)_a}(\sim \mathsf{C}\beta) = 0$. Since $ord_{(\Theta_k)_a}(\sim \mathsf{C}\beta)$ must be minimal in Θ_k by Lemma 4.4 there exists an $l \geq k$, such that $\sim \mathsf{C}\beta$ at a is redex of Θ_l. Then by the definition of deduction chains $d_{\Theta_{l+1}}(0) = \emptyset$.

$\underline{m \to m+1}$: By the induction hypothesis there exists an $l' \geq k$, such that the set

$$d_{\Theta_{l'}}(m) := \{(\sim \mathsf{C}\beta, a); \Theta_{l'} \text{ contains } \sim \mathsf{C}\beta \text{ at } a \text{ and } ord_{(\Theta_{l'})_a}(\sim \mathsf{C}\beta) \leq m\}$$

is empty. Thus the set $d_{\Theta_{l'}}(m+1)$ contains only the pair $(\sim \mathsf{C}\gamma, a)$ such that $ord_{(\Theta_{l'})_a}(\sim \mathsf{C}\gamma) = m+1$. Since $ord_{(\Theta_{l'})_a}(\sim \mathsf{C}\gamma)$ is minimal in $\Theta_{l'}$ by Lemma 4.4 there exists an $l'' \geq l'$ such that $\sim \mathsf{C}\gamma$ at a is the redex of $\Theta_{l''}$. Therefore, again by the definition of deduction chains $d_{\Theta_{l''+1}}(m+1)$ must be empty.

Thus we have shown the claim for all $m \geq 0$. □

Lemma 4.6 (Fairness) *If a reducible formula ϕ appears in Θ_k at $b \in I_k$, then there exists an $l \geq k$, such that ϕ is the redex of Θ_l at $b \in I_l$.*

PROOF. Due to the definition of redex, we must distinguish the following two cases:

Case 1) ϕ is not of the form $\sim \mathsf{C}\psi$: Then the claim follows by Lemma 4.3.
Case 2) ϕ is of the form $\sim \mathsf{C}\psi$: Then the claim follows by Lemma 4.5. □

Definition 4.7 *Define the Kripke structure $\mathsf{K}_\Theta = (S_\Theta, \mathcal{K}_1, \ldots, \mathcal{K}_n, \pi)$ as follows:*

(i) $S_\Theta := \bigcup I_i$
(ii) *for each* $a \in S_\Theta$ *define* $B_a := \bigcup set(\mathbf{R}^i{}_a)$, *where* $\mathbf{R}^i := \Theta_i$
(iii) $\pi(\mathsf{p}) := \{a \in S_\Theta; \neg\mathsf{p} \in B_a\}$, *for each atomic formula* p
(iv) $\mathcal{K}_i := \sigma_i$, *for each* $i \in \{1, \ldots, n\}$

We write $a \in \mathsf{K}_\Theta$ *for* $a \in S_\Theta$.

Lemma 4.8 (Saturation) *Let* $a \in \mathsf{K}_\Theta$.

(1) If $\phi \vee \psi \in B_a$, *then* $\phi \in B_a$ *and* $\psi \in B_a$
(2) If $\phi \wedge \psi \in B_a$, *then* $\phi \in B_a$ *or* $\psi \in B_a$
(3) If $\mathsf{K}_i\phi \in B_a$, *then there exists a node* $c \in \mathsf{K}_\Theta$, *such that* $a\mathcal{K}_i c$ *and* $\phi \in B_c$
(4) If $\sim\mathsf{K}_i\phi \in B_a$, *then* $\neg\phi \in B_c$ *for all* $c \in S_\Theta$ *such that* $a\mathcal{K}_i c$.
(5) If $\mathsf{E}^k\phi \in B_a$ *for some* $k \in \omega$, *then there exists a* $c \in S_\Theta$, *reachable in* k *steps from* a *such that* $\phi \in B_c$
(6) If $\neg\mathsf{E}^k\phi \in B_a$ *for some* $k \in \omega$, *then* $\neg\phi \in B_c$ *for all* $c \in S_\Theta$ *reachable in* k *steps from* a.
(7) If $\mathsf{C}\phi \in B_a$, *then* $\mathsf{E}^k\phi \in B_a$ *for some* $k \in \omega$
(8) If $\sim\mathsf{C}\phi \in B_a$, *then* $\neg\mathsf{E}^k\phi \in B_a$ *for all* $k \in \omega$

PROOF. All claims are consequences of Definition 3.12, Definition 4.7 and Lemma 4.6. □

Lemma 4.9 *For every formula* $\phi \in \mathcal{L}^n_{\mathsf{C}}$ *and every* $a \in S_\Theta$

(1) If $\phi \in B_a$, *then* $\mathsf{K}_\Theta, a \nvDash \phi$
(2) If $\neg\phi \in B_a$, *then* $\mathsf{K}_\Theta, a \vDash \phi$

PROOF. We prove the claims by induction on the structure of ϕ.

$\underline{\phi = \mathsf{p}}$:
(1): $\mathsf{p} \in B_a \overset{\text{Lemma 4.2}}{\Longrightarrow} \sim\mathsf{p} \notin B_a \implies a \notin \pi(\mathsf{p}) \implies \mathsf{K}_\Theta, a \nvDash \mathsf{p}$
(2): $\neg\mathsf{p} \in B_a \implies a \in \pi(\mathsf{p}) \implies \mathsf{K}_\Theta, a \vDash \mathsf{p}$
$\underline{\phi = \sim\mathsf{p}}$: Dually to the previous case.
$\underline{\phi = \psi_1 \wedge \psi_2}$:
(1): $\psi_1 \wedge \psi_2 \in B_a \overset{\text{Lemma 4.8}}{\Longrightarrow} \psi_1 \in B_a$ or $\psi_2 \in B_a$
$\overset{\text{ind. hyp.}}{\Longrightarrow} \mathsf{K}_\Theta, a \nvDash \psi_1$ or $\mathsf{K}_\Theta, a \nvDash \psi_2 \implies \mathsf{K}_\Theta, a \nvDash \psi_1 \wedge \psi_2$
(2): $\neg(\psi_1 \wedge \psi_2) \in B_a \overset{\text{Lemma 4.8}}{\Longrightarrow} \neg\psi_1 \in B_a$ and $\neg\psi_2 \in B_a$
$\overset{\text{ind. hyp.}}{\Longrightarrow} \mathsf{K}_\Theta, a \vDash \psi_1$ and $\mathsf{K}_\Theta, a \vDash \psi_2 \implies \mathsf{K}_\Theta, a \vDash \psi_1 \wedge \psi_2$
$\underline{\phi = \psi_1 \vee \psi_2}$: Dually to the previous case.
$\underline{\phi = \mathsf{K}_i\psi}$:
(1): If $\mathsf{K}_i\psi \in B_a$, then by Lemma 4.8 there exists a $c \in S_\Theta$ such that $a\mathcal{K}_i c$ and $\psi \in B_c$. Thus by induction hypothesis there exists a $c \in S_\Theta$ such that $a\mathcal{K}_i c$ and $\mathsf{K}_\Theta, c \nvDash \psi$. Therefore $\mathsf{K}_\Theta, a \nvDash \mathsf{K}_i\psi$

(2): If $\neg \mathsf{K}_i\psi \in B_a$, then by Lemma 4.8 $\sim\psi \in B_c$ for all $c \in S_\Theta$ such that $a\mathcal{K}_i c$. Thus by induction hypothesis $\mathsf{K}_\Theta, c \vDash \psi$ for all $c \in S_\Theta$ such that $a\mathcal{K}_i c$ and therefore $\mathsf{K}_\Theta, a \vDash \mathsf{K}_i\psi$.

$\underline{\phi = \sim\mathsf{K}_i\psi}$:

(1): If $\sim \mathsf{K}_i\psi \in B_a$, then by the previous case $\mathsf{K}_\Theta, a \vDash \mathsf{K}_i\psi$. Thus also $\mathsf{K}_\Theta, a \nvDash \neg\mathsf{K}_i\psi$.

(2): $\neg \sim\mathsf{K}_i\psi$ is the formula $\mathsf{K}_i\psi$. Thus if $\neg \sim\mathsf{K}_i\psi \in B_a$, then by the previous case $\mathsf{K}_\Theta, a \nvDash \psi$. Therefore $\mathsf{K}_\Theta, a \vDash \sim\mathsf{K}_i\psi$.

$\underline{\phi = \mathsf{C}\psi}$:

(1): If $\mathsf{C}\psi \in B_a$, then by Lemma 4.8 $\mathsf{E}^k\psi \in B_a$ for some $k \in \omega$. Then, again by Lemma 4.8 there exists a $c \in S_\Theta$ which is reachable from a in k steps and $\psi \in B_a$. Thus by induction hypothesis there exists a $c \in S_\Theta$ which is reachable from a in k steps and $\mathsf{K}_\Theta, c \nvDash \psi$. Therefore $\mathsf{K}_\Theta, a \nvDash \mathsf{E}^k\psi$ and thus also $\mathsf{K}_\Theta, a \nvDash \mathsf{C}\psi$.

(2): If $\sim \mathsf{C}\psi \in B_a$, then by Lemma 4.8 $\neg\mathsf{E}^k\psi \in B_a$ for all $k \in \omega$. Thus by induction hypothesis $\mathsf{K}_\Theta, a \vDash \mathsf{E}^k\psi$ for all $k \in \omega$ and therefore $\mathsf{K}_\Theta, a \vDash \mathsf{C}\psi$.

$\underline{\phi = \sim\mathsf{C}\psi}$:

(1): If $\sim \mathsf{C}\psi \in B_a$, then by the previous case $\mathsf{K}_\Theta, a \vDash \mathsf{C}\psi$, thus trivially $\mathsf{K}_\Theta, a \nvDash \neg\mathsf{C}\psi$.

(2): $\neg \sim\mathsf{C}\psi$ is the formula $\mathsf{C}\psi$. Thus by the previous case, if $\neg \sim\mathsf{C}\psi \in B_a$, then $\mathsf{K}_\Theta, a \nvDash \mathsf{C}\psi$. Therefore, trivially $\mathsf{K}_\Theta, a \vDash \sim\mathsf{C}\psi$

This concludes the proof of (1) and (2) for all cases and thus the claim is shown. □

An immediate consequence of the previous lemma is the principle semantic lemma stated as follows.

Lemma 4.10 (Principle semantic lemma) *Let ϕ be a formula of $\mathcal{L}^n_{\mathsf{C}}$. If there exists a deduction chain of ϕ which does not end with an axiomatic sequence, then we can find a Kripke structure $\mathcal{M}$ and a world w such that $\mathcal{M}, w \nvDash \phi$.*

5 Principal syntactic lemma

The principle syntactic lemma says that if all deduction chains for a formula ψ end in axiomatic sequence trees, then there exists a proof of ψ in $\mathsf{T}^\omega_{\mathsf{KC}_n}$. Hence, together with the principal semantic lemma we obtain either a proof or a countermodel for each formula ψ of $\mathcal{L}^n_{\mathsf{C}}$. This amounts to a (constructive) completeness result for $\mathsf{T}^\omega_{\mathsf{KC}_n}$.

The principle syntactic lemma is proven along the following lines.

(1) Code each sequence tree $\mathbf{R}$ in the deduction tree (consisting of all deduction chains) of ψ as a set of formulae $C^{\mathbf{R}}$.
(2) Show that $\mathsf{T}^{\omega}_{\mathsf{K}^{\mathsf{C}}_n} \vdash C^{\mathbf{L}}$ for each leaf $\mathbf{L}$ of the deduction tree.
(3) Show by induction along the Kleene-Brouwer ordering of the deduction tree that $\mathsf{T}^{\omega}_{\mathsf{K}^{\mathsf{C}}_n} \vdash C^{\mathbf{R}}$ if $\mathsf{T}^{\omega}_{\mathsf{K}^{\mathsf{C}}_n} \vdash C^{\mathbf{S_i}}$ for all successors $\mathbf{S_i}$ of $\mathbf{R}$.
(4) Finally, observe $C^{\mathbf{R}} = \psi$ for the root $\mathbf{R}$ of the deduction tree.

However, in order to prove step (3) of the above procedure, we need a series of lemmata. They state that (in certain cases) the rules of $\mathsf{T}^{\omega}_{\mathsf{K}^{\mathsf{C}}_n}$ may also be applied deep inside $\mathcal{L}^n_{\mathsf{C}}$ formulae. These lemmata are shown first.

Definition 5.1 *We extend the alphabet of the language $\mathcal{L}^n_{\mathsf{C}}$ by a propositional variable x. Let $\mathcal{L}^n_{\mathsf{C},\mathsf{x}}$ be the set of all formulae over this new alphabet. Let ϕ and ψ be formulae in $\mathcal{L}^n_{\mathsf{C},\mathsf{x}}$. $\phi[\psi]$ shall denote the formula which results from substituting all occurrences of x in ϕ with ψ. Furthermore, we define $\hat{\mathcal{L}}^n_{\mathsf{C}}$ to be the set of all formulae of $\mathcal{L}^n_{\mathsf{C}}$ which are of the form p, $\sim\mathsf{p}$, $\mathsf{K}_i\beta$, $\sim\mathsf{K}_i\beta$ or $\sim\mathsf{C}\beta$ for some β in $\mathcal{L}^n_{\mathsf{C}}$. Let $dis\hat{\mathcal{L}}^n_{\mathsf{C}}$ denote the set of disjunctions over elements of $\hat{\mathcal{L}}^n_{\mathsf{C}}$.*

Definition 5.2 *Let $\#$ denote the natural sum operation on ordinals. For all formulae $\alpha \in \mathcal{L}^n_{\mathsf{C}}$, we inductively define a complexity measure comp(α) as follows:*

1. *comp$(\alpha) = 1$ for all $\alpha \in \hat{\mathcal{L}}^n_{\mathsf{C}}$*
2. *comp$(\alpha \wedge \beta) = 1 \,\#$ comp$(\alpha) \,\#$ comp(β)*
3. *comp$(\alpha \vee \beta) = 1 \,\#$ comp$(\alpha) \,\#$ comp(β)*
4. *comp$(\mathsf{C}\alpha) = \omega^{comp(\alpha)}$*

Furthermore, given a finite set $\Gamma = \{\gamma_1, \ldots, \gamma_l\} \subset \mathcal{L}^n_{\mathsf{C}}$, we define

$$comp(\Gamma) = comp(\gamma_1) \,\#\, \ldots \,\#\, comp(\gamma_l).$$

Remark 5.3 *By Definition 5.2 we have comp$(\mathsf{E}^k\xi) <$ comp$(\mathsf{C}\xi)$ for any formula ξ of $\mathcal{L}^n_{\mathsf{C}}$ and any $k \in \omega$. Furthermore, for any finite $\Gamma \subset \mathcal{L}^n_{\mathsf{C}}$ we have comp$(\Gamma) \geq |\Gamma|$. In particular, we have comp$(\Gamma) = |\Gamma|$ if $\Gamma \subset \hat{\mathcal{L}}^n_{\mathsf{C}}$.*

Definition 5.4 *We inductively define the subsets $\mathcal{A}^k_{\mathsf{x}}$ of $\mathcal{L}^n_{\mathsf{C},\mathsf{x}}$ as follows:*

$$\begin{aligned}
\mathcal{A}^0_{\mathsf{x}} &:= \{\phi \in \mathcal{L}^n_{\mathsf{C},\mathsf{x}}; \phi = \psi \vee \mathsf{x} \text{ and } \psi \in \mathcal{L}^n_{\mathsf{C}}\} \\
\mathcal{A}^{k+1}_{\mathsf{x}} &:= \{\phi \in \mathcal{L}^n_{\mathsf{C},\mathsf{x}}; \phi = \psi \vee \mathsf{K}_i\delta[\mathsf{x}] \text{ where } \psi \in dis\hat{\mathcal{L}}^n_{\mathsf{C}} \text{ and } \delta[\mathsf{x}] \text{ is in } \mathcal{A}^k_{\mathsf{x}}\}
\end{aligned}$$

Furthermore we define $\mathcal{A}_{\mathsf{x}}$ as $\bigcup \mathcal{A}^k_{\mathsf{x}}$ and for $\phi \in \mathcal{A}_{\mathsf{x}}$ depth(ϕ) as the least k, such that $\phi \in \mathcal{A}^k_{\mathsf{x}}$.

Lemma 5.5 *Let A be a formula in $\mathcal{A}_{\mathsf{x}}$ and Γ be a finite subset of $\hat{\mathcal{L}}^n_{\mathsf{C}}$. The following implications hold:*

1. *If* $\mathsf{T}^{\omega}_{\mathsf{K}^{\mathsf{C}}_n} \vdash \Gamma, A[\mathsf{E}^k\phi]$ *for every* $k \in \omega$*, then* $\mathsf{T}^{\omega}_{\mathsf{K}^{\mathsf{C}}_n} \vdash \Gamma, A[\mathsf{C}\phi]$
2. *If* $\mathsf{T}^{\omega}_{\mathsf{K}^{\mathsf{C}}_n} \vdash \Gamma, A[\phi]$ *and* $\mathsf{T}^{\omega}_{\mathsf{K}^{\mathsf{C}}_n} \vdash \Gamma, A[\psi]$*, then* $\mathsf{T}^{\omega}_{\mathsf{K}^{\mathsf{C}}_n} \vdash \Gamma, A[\phi \wedge \psi]$
3. *If* $\mathsf{T}^{\omega}_{\mathsf{K}^{\mathsf{C}}_n} \vdash \Gamma, A[\sim\mathsf{C}\phi \vee \neg\mathsf{E}^k\phi]$ *for some* $k \in \omega$*, then* $\mathsf{T}^{\omega}_{\mathsf{K}^{\mathsf{C}}_n} \vdash \Gamma, A[\sim\mathsf{C}\phi]$

PROOF. All three clauses are shown by induction on $d := \mathrm{depth}(A)$.
Clause 1: The base case of $d = 0$ follows directly by Lemma 2.3 and the rule (C^{ω}). We thus consider the induction step and assume that

$$\mathsf{T}^{\omega}_{\mathsf{K}^{\mathsf{C}}_n} \mid\!\!\!\frac{}{\alpha_k} \Gamma, \psi \vee \mathsf{K}_i\delta[\mathsf{E}^k\phi]$$

for all $k \in \omega$ where $\mathrm{depth}(\delta) = d$. Therefore, by iterated applications of Lemma 2.3 and the fact that $\psi \in dis\hat{\mathcal{L}}^n_{\mathsf{C}}$ we have

$$\mathsf{T}^{\omega}_{\mathsf{K}^{\mathsf{C}}_n} \mid\!\!\!\frac{}{\alpha_k} \Gamma, \psi_1, \ldots, \psi_l, \mathsf{K}_i\delta[\mathsf{E}^k\phi] \tag{2}$$

for all $k \in \omega$ and suitable $\psi_1, \ldots, \psi_l$. We claim that

$$\mathsf{T}^{\omega}_{\mathsf{K}^{\mathsf{C}}_n} \vdash \Gamma, \psi_1, \ldots, \psi_l, \mathsf{K}_i\delta[\mathsf{C}\phi] \tag{3}$$

and distinguish two cases:

(i) For some $m \in \omega$ $\mathsf{K}_i\delta[\mathsf{E}^m\phi]$ was obtained by weakening in the derivation of (2), say after some $\beta_m \leq \alpha_m$.
(ii) For all $k \in \omega$ $\mathsf{K}_i\delta[\mathsf{E}^k\phi]$ was obtained by an application of the rule (K_i) in the derivation of (2), each one say after $\beta_k \leq \alpha_k$ respectively.

In case (i) we may instead conclude $\mathsf{K}_i\delta[\mathsf{C}\phi]$ after β_m and due to the fact that $\Gamma, \psi_1, \ldots, \psi_l \subset \hat{\mathcal{L}}^n_{\mathsf{C}}$ we may use the same inferences henceforth to conclude $\mathsf{T}^{\omega}_{\mathsf{K}^{\mathsf{C}}_n} \vdash \Gamma, \psi_1, \ldots, \psi_l, \mathsf{K}_i\delta[\mathsf{C}\phi]$.
In case (ii) by the premise of the rule (K_i) we have for each $k \in \omega$

$$\mathsf{T}^{\omega}_{\mathsf{K}^{\mathsf{C}}_n} \vdash \neg\mathsf{C}\Delta^k_1, \neg\Delta^k_2, \delta[\mathsf{E}^k\phi] \tag{4}$$

where $\neg\mathsf{C}\Delta^k_1 \subset \hat{\mathcal{L}}^n_{\mathsf{C}}$ and $\neg\Delta^k_2 \subset \mathcal{L}^n_{\mathsf{C}}$ are suitable finite sets of formulae. Now define $\Gamma' := \Gamma, \psi_1, \ldots, \psi_l$, $\Gamma'|_{\neg\mathsf{C}} := \{\sim\mathsf{C}\xi \in \Gamma'\}$ and $\Gamma'|_{\neg\mathsf{K}_i} := \{\xi; \sim\mathsf{K}_i\xi \in \Gamma'\}$. By the fact that $\Gamma' \subset \hat{\mathcal{L}}^n_{\mathsf{C}}$ the following two statements hold for every $k \in \omega$:

$$\neg\mathsf{C}\Delta^k_1 \subset \Gamma'|_{\neg\mathsf{C}} \tag{5}$$
$$\neg\Delta^k_2 \subset \neg\Gamma'|_{\neg\mathsf{K}_i} \tag{6}$$

Clearly, we also have

$$\Gamma'|_{\neg\mathsf{C}} \subset \Gamma' \tag{7}$$
$$\neg\mathsf{K}_i\Gamma'|_{\neg\mathsf{K}_i} \subset \Gamma' \tag{8}$$

By Lemma 2.2, (4), (5) and (6) we get

$$\mathsf{T}^{\omega}_{\mathsf{KC}_n} \vdash \Gamma'|_{\neg\mathsf{C}}, \neg\Gamma'|_{\neg\mathsf{K}_i}, \delta[\mathsf{E}^k\phi] \tag{9}$$

for every $k \in \omega$. We show that

$$\mathsf{T}^{\omega}_{\mathsf{KC}_n} \vdash \Gamma'|_{\neg\mathsf{C}}, \neg\Gamma'|_{\neg\mathsf{K}_i}, \delta[\mathsf{C}\phi] \tag{10}$$

by induction on $\gamma := \mathrm{comp}(\neg\Gamma'|_{\neg\mathsf{K}_i})$. As the base case we have $\gamma = |\neg\Gamma'|_{\neg\mathsf{K}_i}|$ by Remark 5.3. But in this case $\neg\Gamma'|_{\neg\mathsf{K}_i}$ is either empty or a subset of $\hat{\mathcal{L}}^n_{\mathsf{C}}$. Therefore, the claim follows by induction hypothesis of the outer induction. Now assume that the claim holds for all $\gamma' < \gamma$. Then there exists a set $\Sigma \subset \mathcal{L}^n_{\mathsf{C}}$ and formulae ξ_1, ξ_2, ξ such that one of the following three cases holds

(a) $\Sigma, \xi_1 \wedge \xi_2 = \neg\Gamma'|_{\neg\mathsf{K}_i}$ and $\mathrm{comp}(\xi_1), \mathrm{comp}(\xi_2) < \mathrm{comp}(\xi_1 \wedge \xi_2)$
(b) $\Sigma, \xi_1 \vee \xi_2 = \neg\Gamma'|_{\neg\mathsf{K}_i}$ and $\mathrm{comp}(\xi_1), \mathrm{comp}(\xi_2) < \mathrm{comp}(\xi_1 \vee \xi_2)$
(c) $\Sigma, \mathsf{C}\xi = \neg\Gamma'|_{\neg\mathsf{K}_i}$ and by Remark 5.3 $\mathrm{comp}(\mathsf{E}^k\xi) < \mathrm{comp}(\mathsf{C}\xi)$ for all $k \in \omega$.

Case (a): By (9) and Lemma 2.3 we have

$$\mathsf{T}^{\omega}_{\mathsf{KC}_n} \vdash \Gamma'|_{\neg\mathsf{C}}, \Sigma, \xi_1, \delta[\mathsf{E}^k\phi] \text{ and}$$
$$\mathsf{T}^{\omega}_{\mathsf{KC}_n} \vdash \Gamma'|_{\neg\mathsf{C}}, \Sigma, \xi_2, \delta[\mathsf{E}^k\phi]$$

for all $k \in \omega$. Thus by the induction hypothesis of the inner induction

$$\mathsf{T}^{\omega}_{\mathsf{KC}_n} \vdash \Gamma'|_{\neg\mathsf{C}}, \Sigma, \xi_1, \delta[\mathsf{C}\phi] \text{ and}$$
$$\mathsf{T}^{\omega}_{\mathsf{KC}_n} \vdash \Gamma'|_{\neg\mathsf{C}}, \Sigma, \xi_2, \delta[\mathsf{C}\phi]$$

and again by the rule $(\wedge)$ we obtain the claim.
Case (b) and case (c) are treated in analogous ways, using Lemma 2.3. From (10) using (K_i), we obtain $\mathsf{T}^{\omega}_{\mathsf{KC}_n} \vdash \Gamma'|_{\neg\mathsf{C}}, \neg\mathsf{K}_i\Gamma'|_{\neg\mathsf{K}_i}, \mathsf{K}_i\delta[\mathsf{C}\phi]$. With (7), (8) and Lemma 2.2 we conclude $\mathsf{T}^{\omega}_{\mathsf{KC}_n} \vdash \Gamma', \mathsf{K}_i\delta[\mathsf{C}\phi]$. Thus (3) holds in both cases (i) and (ii). Then, by an iterated application of $(\vee)$ $\mathsf{T}^{\omega}_{\mathsf{KC}_n} \vdash \Gamma, A[\mathsf{C}\phi]$ follows and this clause is shown.

Clause 2: The base case of $d = 0$ follows by Lemma 2.3, the rule $(\wedge)$ and finally an application of the rule $(\vee)$. The induction step is analogous to clause 1 only that in this case we are dealing with just two premises instead of infinitely many.

Clause 3: The base case of $d = 0$ follows by Lemmata 2.3 and 2.4. We therefore consider the induction step and assume that $\mathsf{T}^{\omega}_{\mathsf{KC}_n} \mid\!\!\!\frac{}{\alpha}\ \Gamma, \psi \vee \mathsf{K}_i\delta[\sim \mathsf{C}\phi \vee \neg\mathsf{E}^k\phi]$, where $\mathrm{depth}(\delta) = d$. Therefore, by iterated applications of Lemma 2.3 and the fact that $\psi \in dis\hat{\mathcal{L}}^n_{\mathsf{C}}$ we have

$$\mathsf{T}^{\omega}_{\mathsf{KC}_n} \mid\!\!\!\frac{}{\alpha}\ \Gamma, \psi_1, \ldots, \psi_l, \mathsf{K}_i\delta[\sim\mathsf{C}\phi \vee \neg\mathsf{E}^k\phi] \tag{11}$$

For suitable $\psi_1, \ldots, \psi_l$. We claim that

$$\mathsf{T}^{\omega}_{\mathsf{K}^{\mathsf{C}}_n} \vdash \Gamma, \psi_1, \ldots, \psi_l, \mathsf{K}_i\delta[\sim \mathsf{C}\phi] \tag{12}$$

and distinguish two cases:

(i) $\mathsf{K}_i\delta[\sim \mathsf{C}\phi \vee \neg \mathsf{E}^k\phi]$ was introduced by weakening in the derivation of (11), say after some $\beta < \alpha$.
(ii) $\mathsf{K}_i\delta[\sim \mathsf{C}\phi \vee \neg \mathsf{E}^k\phi]$ was obtained by the rule (K_i) in the derivation of (11).

In case (i) we may instead introduce $\mathsf{K}_i\delta[\sim \mathsf{C}\phi]$ with weakening after β and due to the fact that $\Gamma, \psi_1, \ldots, \psi_l \subset \hat{\mathcal{L}}^n_{\mathsf{C}}$ we may use the same inferences henceforth to conclude the claim. In case (ii) we have $\mathsf{T}^{\omega}_{\mathsf{K}^{\mathsf{C}}_n} \vdash \neg \mathsf{C}\Delta_1, \neg\Delta_2, \delta[\sim \mathsf{C}\phi \vee \neg \mathsf{E}^k\phi]$ for suitable sets Δ_1 and Δ_2. Then by induction hypothesis and an identical argument to the corresponding case in clause 1 we obtain

$$\mathsf{T}^{\omega}_{\mathsf{K}^{\mathsf{C}}_n} \vdash \neg \mathsf{C}\Delta_1, \neg\Delta_2, \delta[\sim \mathsf{C}\phi].$$

The rule (K_i) yields $\mathsf{T}^{\omega}_{\mathsf{K}^{\mathsf{C}}_n} \vdash \neg \mathsf{C}\Delta_1, \neg \mathsf{K}_i\Delta_2, \mathsf{K}_i\delta[\sim \mathsf{C}\phi]$. Then by the fact that $\Gamma, \psi_1, \ldots, \psi_l \subset \hat{\mathcal{L}}^n_{\mathsf{C}}$, we may use the same inferences again to arrive at the claim. Thus (12) holds in both cases (i) and (ii). Therefore, by an iteration of the rule $(\vee)$ we arrive at $\mathsf{T}^{\omega}_{\mathsf{K}^{\mathsf{C}}_n} \vdash \Gamma, A[\sim \mathsf{C}\phi]$ and the clause is shown. □

Definition 5.6 *Let $\psi_1, \ldots, \psi_l$ be formulae of $\mathcal{L}^n_{\mathsf{C}}$. We inductively define the subsets $\mathcal{B}^k_{\mathsf{x},\psi_1,\ldots,\psi_l}$ of $\mathcal{L}^n_{\mathsf{C},\mathsf{x}}$ as follows:*

$$\mathcal{B}^1_{\mathsf{x},\psi_1,\ldots,\psi_l} := \{\phi \in \mathcal{L}^n_{\mathsf{C},\mathsf{x}}; \phi = \psi \vee \neg \mathsf{K}_i\psi_1 \vee \ldots \vee \neg \mathsf{K}_i\psi_l \vee \mathsf{K}_i\mathsf{x} \text{ and } \psi \in \mathcal{L}^n_{\mathsf{C}}\}$$
$$\mathcal{B}^{k+1}_{\mathsf{x},\psi_1,\ldots,\psi_l} := \{\phi \in \mathcal{L}^n_{\mathsf{C},\mathsf{x}}; \phi = \psi \vee \mathsf{K}_i\delta[\mathsf{x}] \text{ where } \psi \in dis\hat{\mathcal{L}}^n_{\mathsf{C}} \text{ and } \delta[\mathsf{x}] \text{ is in } \mathcal{B}^k_{\mathsf{x},\psi_1,\ldots,\psi_l}\}$$

Furthermore we define $\mathcal{B}_{\mathsf{x},\psi_1,\ldots,\psi_l}$ as $\bigcup \mathcal{B}^k_{\mathsf{x},\psi_1,\ldots,\psi_l}$ and for $\phi \in \mathcal{B}_{\mathsf{x},\psi_1,\ldots,\psi_l}$ depth(ϕ) as the least k, such that $\phi \in \mathcal{B}^k_{\mathsf{x},\psi_1,\ldots,\psi_l}$.

Lemma 5.7 *Let B be a formula in $\mathcal{B}_{\mathsf{x},\psi_1,\ldots,\psi_l}$ and Γ be a finite subset of $\hat{\mathcal{L}}^n_{\mathsf{C}}$. If $\mathsf{T}^{\omega}_{\mathsf{K}^{\mathsf{C}}_n} \vdash \Gamma, B[\phi \vee \neg\psi_1 \vee \ldots \vee \neg\psi_l]$, then $\mathsf{T}^{\omega}_{\mathsf{K}^{\mathsf{C}}_n} \vdash \Gamma, B[\phi]$.*

PROOF. We prove this claim by induction on $d := \text{depth}(B)$.

$\underline{d = 1}$: We thus have $\mathsf{T}^{\omega}_{\mathsf{K}^{\mathsf{C}}_n} \vdash^{}_{\alpha} \Gamma, \psi \vee \neg \mathsf{K}_i\psi_1 \vee \ldots \vee \neg \mathsf{K}_i\psi_l \vee \mathsf{K}_i(\phi \vee \neg\psi_1 \vee \ldots \vee \neg\psi_l)$ and with iterated applications of Lemma 2.3

$$\mathsf{T}^{\omega}_{\mathsf{K}^{\mathsf{C}}_n} \vdash^{}_{\alpha} \Gamma, \psi, \neg \mathsf{K}_i\psi_1, \ldots, \neg \mathsf{K}_i\psi_l, \mathsf{K}_i(\phi \vee \neg\psi_1 \vee \ldots \vee \neg\psi_l). \tag{13}$$

We show that $\mathsf{T}^{\omega}_{\mathsf{K}^{\mathsf{C}}_n} \vdash \Gamma, \psi, \neg \mathsf{K}_i\psi_1, \ldots, \neg \mathsf{K}_i\psi_l, \mathsf{K}_i\phi$ by induction on α. The base case of $\alpha = 0$ is trivial. Therefore, we assume that the claim holds for all

$\alpha' < \alpha$ and distinguish cases, as to whether or not $\mathsf{K}_i(\phi \vee \neg\psi_1 \vee \ldots \vee \neg\psi_l)$ was the distinguished formula of the last inference used to derive (13). If it was the distinguished formula, then we have

$$\mathsf{T}^{\omega}_{\mathsf{K}^{\mathsf{C}}_n} \vdash \Delta, \neg\psi_1, \ldots, \neg\psi_l, \phi \vee \neg\psi_1 \vee \ldots \vee \neg\psi_l$$

for some suitable set Δ. Hence, with an iteration of Lemma 2.3 we obtain $\mathsf{T}^{\omega}_{\mathsf{K}^{\mathsf{C}}_n} \vdash \Delta, \neg\psi_1, \ldots, \neg\psi_l, \phi$ and thus applying (K_i) we arrive at the claim. If $\mathsf{K}_i(\phi \vee \neg\psi_1 \vee \ldots \vee \neg\psi_l)$ was not the distinguished formula, then we distinguish further cases for the last rule applied to obtain (13). In the cases of the rules $(\wedge)$, $(\vee)$, (C^{ω}) and $(\neg\mathsf{C})$ we simply use the induction hypothesis of the inner induction on the premise and apply the same rule again. In the case of rule (K_j) (for any $1 \leq j \leq n$) we see that $\mathsf{K}_i(\phi \vee \neg\psi_1 \vee \ldots \vee \neg\psi_l)$ can only have been obtained with weakening. Thus we may obtain $\mathsf{K}_i\phi$ instead in the same manner.

$\underline{d \longrightarrow d+1}$: Thus $\mathsf{T}^{\omega}_{\mathsf{K}^{\mathsf{C}}_n} \mid\!\!\!\frac{}{}_{\alpha} \Gamma, \psi \vee \mathsf{K}_i\delta[\phi \vee \neg\psi_1 \vee \ldots \vee \neg\psi_l]$ and by iteration of Lemma 2.3

$$\mathsf{T}^{\omega}_{\mathsf{K}^{\mathsf{C}}_n} \mid\!\!\!\frac{}{}_{\alpha} \Gamma, \psi_1, \ldots, \psi_l, \mathsf{K}_i\delta[\phi \vee \neg\psi_1 \vee \ldots \vee \neg\psi_l] \tag{14}$$

for suitable $\psi_1, \ldots, \psi_l$. We claim that $\mathsf{T}^{\omega}_{\mathsf{K}^{\mathsf{C}}_n} \vdash \Gamma, \psi_1, \ldots, \psi_l, \mathsf{K}_i\delta[\phi]$ and again distinguish two cases:

(i) $\mathsf{K}_i\delta[\phi \vee \neg\psi_1 \vee \ldots \vee \neg\psi_l]$ was obtained by weakening in the derivation of (14)

(ii) $\mathsf{K}_i\delta[\phi \vee \neg\psi_1 \vee \ldots \vee \neg\psi_l]$ was obtained by the rule (K_i) in the derivation of (14).

In both cases we may show the claim as before using the fact that $\Gamma, \psi_1, \ldots, \psi_l \subset \hat{\mathcal{L}}^n_{\mathsf{C}}$. Then by an iteration of the rule $(\vee)$, we arrive at $\mathsf{T}^{\omega}_{\mathsf{K}^{\mathsf{C}}_n} \vdash \Gamma, A[\phi]$ and the Lemma is shown.

□

Definition 5.8 *Let ψ_1 be a formula of $\mathcal{L}^n_{\mathsf{C}}$. We inductively define the subsets $\mathcal{C}^k_{\mathsf{x},\psi_1}$ of $\mathcal{L}^n_{\mathsf{C},\mathsf{x}}$ as follows:*

$$\begin{aligned}
\mathcal{C}^1_{\mathsf{x},\psi_1} &:= \{\phi \in \mathcal{L}^n_{\mathsf{C},\mathsf{x}}; \phi = \psi \vee \neg\mathsf{K}_i\psi_1 \vee \mathsf{K}_i(\mathsf{x} \vee \alpha_1) \vee \ldots \vee \mathsf{K}_i(\mathsf{x} \vee \alpha_p) \\
&\qquad \textit{and } \psi, \alpha_1, \ldots, \alpha_p \in \mathcal{L}^n_{\mathsf{C}}\} \\
\mathcal{C}^{k+1}_{\mathsf{x},\psi_1} &:= \{\phi \in \mathcal{L}^n_{\mathsf{C},\mathsf{x}}; \phi = \psi \vee \mathsf{K}_i\delta[\mathsf{x}] \textit{ where } \psi \in dis\hat{\mathcal{L}}^n_{\mathsf{C}} \textit{ and } \delta[\mathsf{x}] \textit{ is in } \mathcal{C}^k_{\mathsf{x},\psi_1}\}
\end{aligned}$$

Furthermore we define $\mathcal{C}_{\mathsf{x},\psi_1}$ as $\bigcup \mathcal{C}^k_{\mathsf{x},\psi_1}$ and for $\phi \in \mathcal{C}_{\mathsf{x},\psi_1}$ depth(ϕ) as the least k, such that $\phi \in \mathcal{C}^k_{\mathsf{x},\psi_1}$.

Lemma 5.9 *Let C be a formula in $\mathcal{C}_{\mathsf{x},\phi}$ and Γ be a finite subset of $\hat{\mathcal{L}}^n_{\mathsf{C}}$. $\overline{C}$ denotes the formula of $\mathcal{L}^n_{\mathsf{C}}$ which results from erasing every disjunct of the form x in C. If $\mathsf{T}^{\omega}_{\mathsf{K}^{\mathsf{C}}_n} \vdash \Gamma, C[\neg\phi]$, then $\mathsf{T}^{\omega}_{\mathsf{K}^{\mathsf{C}}_n} \vdash \Gamma, \overline{C}$*

PROOF. We prove this claim by induction on $d := \mathrm{depth}(C)$.

$\underline{d = 1}$: Thus $\mathsf{T}^{\omega}_{\mathsf{KC}_n} \vdash^{-}_{\alpha} \Gamma, \psi \vee \neg \mathsf{K}_i\phi \vee \mathsf{K}_i(\neg\phi \vee \alpha_1) \vee \ldots \vee \mathsf{K}_i(\neg\phi \vee \alpha_p)$ and by repeated applications of Lemma 2.3

$$\mathsf{T}^{\omega}_{\mathsf{KC}_n} \vdash^{-}_{\alpha} \Gamma, \psi, \neg \mathsf{K}_i\phi, \mathsf{K}_i(\neg\phi \vee \alpha_1), \ldots, \mathsf{K}_i(\neg\phi \vee \alpha_p) \tag{15}$$

We claim that $\mathsf{T}^{\omega}_{\mathsf{KC}_n} \vdash \Gamma, \psi, \neg \mathsf{K}_i\phi, \mathsf{K}_i\alpha_1, \ldots, \mathsf{K}_i\alpha_p$ by induction on α. The base case of $\alpha = 0$ is trivial. Thus we assume that the claim holds for all $\alpha' < \alpha$ and make a case distinction as to whether or not $\mathsf{K}_i(\neg\phi \vee \alpha_j)$ was the distinguished formula of the last inference used to derive (15) for any $1 \leq j \leq l$. In the first case we then have $\mathsf{T}^{\omega}_{\mathsf{KC}_n} \vdash \Delta, \neg\phi, \neg\phi \vee \alpha_j$ and thus with Lemma 2.3 $\mathsf{T}^{\omega}_{\mathsf{KC}_n} \vdash \Delta, \neg\phi, \alpha_j$. Therefore, using (K_i) we obtain the claim. If $\mathsf{K}_i(\neg\phi \vee \alpha_j)$ was not the distinguished formula for any $1 \leq j \leq l$, then we distinguish further cases for the last rule applied to obtain (15). In the cases of the rules $(\wedge)$, $(\vee)$, (C^{ω}) and $(\neg\mathsf{C})$ we simply use the induction hypothesis of the inner induction on the premise and apply the same rule again. In the case of rule (K_h) (for any $1 \leq h \leq n$) we see that for every $1 \leq j \leq l$ $\mathsf{K}_i(\neg\phi \vee \alpha_j)$ can only have been obtained with weakening. Thus we may obtain $\mathsf{K}_i\alpha_j$ for every $1 \leq j \leq l$ in the same manner.

$\underline{d \to d+1}$: This part of the induction is analogous to the corresponding part in the proof of Lemma 5.7.

□

Definition 5.10 *Let $\mathbf{R}$ be a sequence tree over I and $a = (l, \alpha) \in I$. We define the characteristic set $C^{\mathbf{R}}_a$ of $\mathbf{R}$ at a inductively as follows:*

(1) If a is a leaf of I, then $C^{\mathbf{R}}_a := \mathit{set}^{+}(\mathbf{R}_a)$
(2) If a has successors $b_1, \ldots, b_m \in I$ and

$$\begin{aligned} b_1 &= (p_1, (\alpha, q_1)) \\ &\vdots \quad \vdots \\ b_m &= (p_m, (\alpha, q_m)), \end{aligned}$$

then $C^{\mathbf{R}}_a := \mathit{set}^{+}(\mathbf{R}_a) \cup \{\mathsf{K}_{p_1} \bigvee C^{\mathbf{R}}_{b_1}\} \cup \ldots \cup \{\mathsf{K}_{p_m} \bigvee C^{\mathbf{R}}_{b_m}\}$.

Lemma 5.11 *If $\mathbf{R}$ is an axiomatic sequence tree over I, then $\mathsf{T}^{\omega}_{\mathsf{KC}_n} \vdash C^{\mathbf{R}}_{(0,(0))}$.*

PROOF. Since $\mathbf{R}$ is axiomatic, there exists a $c \in I$ and some atomic formula p, such that p and $\sim\mathsf{p}$ are both in $C^{\mathbf{R}}_c$. Thus using (ID) we obtain $\mathsf{T}^{\omega}_{\mathsf{KC}_n} \vdash C^{\mathbf{R}}_c$. We show that $\mathsf{T}^{\omega}_{\mathsf{KC}_n} \vdash C^{\mathbf{R}}_b$ for all $b \preccurlyeq c$ by induction inverse to the length of c.

$\underline{b = c}$: This case is already shown above.

$\underline{b \prec c}$: Let $b = (k, \beta)$. Then there exists a $d \preccurlyeq c$ such that $d = (i, (\beta, l))$ for some natural numbers i and l. By induction hypothesis $\mathsf{T}^{\omega}_{\mathsf{K}^{\mathsf{C}}_n} \vdash C^{\mathbf{R}}_d$, thus an iteration of applications of ($\vee$) yields $\mathsf{T}^{\omega}_{\mathsf{K}^{\mathsf{C}}_n} \vdash \bigvee C^{\mathbf{R}}_d$. Then, applying ($\mathsf{K}_i$), we obtain $\mathsf{T}^{\omega}_{\mathsf{K}^{\mathsf{C}}_n} \vdash C^{\mathbf{R}}_b$.

Thus the claim holds and since $(0, (0)) \preccurlyeq c$ the Lemma is shown. □

Lemma 5.12 *Let* $\mathbf{R}$ *be a sequence tree with redex* $\phi \vee \psi$ *and* $\mathbf{S}$ *be the type 1 successor of* $\mathbf{R}$*. If* $\mathsf{T}^{\omega}_{\mathsf{K}^{\mathsf{C}}_n} \vdash C^{\mathbf{S}}_{(0,(0))}$*, then* $\mathsf{T}^{\omega}_{\mathsf{K}^{\mathsf{C}}_n} \vdash C^{\mathbf{R}}_{(0,(0))}$.

PROOF. This claim trivially holds since $\mathsf{T}^{\omega}_{\mathsf{K}^{\mathsf{C}}_n} \vdash C^{\mathbf{S}}_{(0,(0))}$ and $\mathsf{T}^{\omega}_{\mathsf{K}^{\mathsf{C}}_n} \vdash C^{\mathbf{R}}_{(0,(0))}$ are the same set of formulae. □

Lemma 5.13 *Let* $\mathbf{R}$ *be a sequence tree with redex* $\phi \wedge \psi$ *and* $\mathbf{S}, \mathbf{T}$ *be the type 2 successors of* $\mathbf{R}$*. If* $\mathsf{T}^{\omega}_{\mathsf{K}^{\mathsf{C}}_n} \vdash C^{\mathbf{S}}_{(0,(0))}$ *and* $\mathsf{T}^{\omega}_{\mathsf{K}^{\mathsf{C}}_n} \vdash C^{\mathbf{T}}_{(0,(0))}$*, then* $\mathsf{T}^{\omega}_{\mathsf{K}^{\mathsf{C}}_n} \vdash C^{\mathbf{R}}_{(0,(0))}$.

PROOF. There exists a formula $A \in \mathcal{A}_{\mathsf{x}}$, such that $A[\phi] = \bigvee C^{\mathbf{S}}_{(0,(0))}$ and $A[\psi] = \bigvee C^{\mathbf{T}}_{(0,(0))}$ as well as $A[\phi \wedge \psi] = \bigvee C^{\mathbf{R}}_{(0,(0))}$. Therefore, the claim holds by clause 2 of Lemma 5.5 and iterations of Lemma 2.3. □

Lemma 5.14 *Let* $\mathbf{R}$ *be a sequence tree with redex* $\sim \mathsf{K}_i\phi$ *and* $\mathbf{S}$ *be the type 3 successor of* $\mathbf{R}$*. If* $\mathsf{T}^{\omega}_{\mathsf{K}^{\mathsf{C}}_n} \vdash C^{\mathbf{S}}_{(0,(0))}$*, then* $\mathsf{T}^{\omega}_{\mathsf{K}^{\mathsf{C}}_n} \vdash C^{\mathbf{R}}_{(0,(0))}$.

PROOF. Since $\mathbf{S}$ is the type 3 successor of a sequence tree with redex $\sim \mathsf{K}_i\phi$, there exists a formula $C \in \mathcal{C}_{\mathsf{x},\phi}$ such that $C[\neg\phi] = \bigvee C^{\mathbf{S}}_{(0,(0))}$ and $\overline{C} = \bigvee C^{\mathbf{R}}_{(0,(0))}$. Therefore, the claim holds by Lemma 5.9 and iterations of Lemma 2.3. □

Lemma 5.15 *Let* $\mathbf{R}$ *be a sequence tree with redex* $\mathsf{K}_i\phi$ *and* $\mathbf{S}$ *be the type 4 successor of* $\mathbf{R}$*. If* $\mathsf{T}^{\omega}_{\mathsf{K}^{\mathsf{C}}_n} \vdash C^{\mathbf{S}}_{(0,(0))}$*, then* $\mathsf{T}^{\omega}_{\mathsf{K}^{\mathsf{C}}_n} \vdash C^{\mathbf{R}}_{(0,(0))}$.

PROOF. Since $\mathbf{S}$ is the type 4 successor of a sequence tree with redex $\mathsf{K}_i\phi$, there exist formulae $\psi_1, \ldots, \psi_l$ and a formula $B \in \mathcal{B}_{\mathsf{x},\psi_1,\ldots,\psi_l}$ such that

$$B[\phi \vee \neg\psi_1 \vee \ldots \vee \neg\psi_l] = \bigvee C^{\mathbf{S}}_{(0,(0))}$$

and $B[\phi] = \bigvee C^{\mathbf{R}}_{(0,(0))}$. Therefore, the claim holds by Lemma 5.7 and iterations of Lemma 2.3. □

Lemma 5.16 *Let* $\mathbf{R}$ *be a sequence tree with redex* $\mathsf{C}\phi$ *and* $\mathbf{S}^i$ *where* $i \in \omega$ *be the type 5 successors of* $\mathbf{R}$*. If* $\mathsf{T}^{\omega}_{\mathsf{K}^{\mathsf{C}}_n} \vdash C^{\mathbf{S}^i}_{(0,(0))}$ *for all* $i \in \omega$*, then* $\mathsf{T}^{\omega}_{\mathsf{K}^{\mathsf{C}}_n} \vdash C^{\mathbf{R}}_{(0,(0))}$.

PROOF. There exists a formula $A \in \mathcal{A}_{\mathsf{x}}$, such that $A[\mathsf{E}^k\phi] = \bigvee C^{\mathbf{S}^k}_{(0,(0))}$ and $A[\mathsf{C}\phi] = \bigvee C^{\mathbf{R}}_{(0,(0))}$. Therefore, the claim holds by clause 1 of Lemma 5.5 and iterations of Lemma 2.3. □

Lemma 5.17 *Let* $\mathbf{R}$ *be a sequence tree with redex* $\sim \mathsf{C}\phi$ *and* $\mathbf{S}$ *be the type 6 successor of* $\mathbf{R}$*. If* $\mathsf{T}^{\omega}_{\mathsf{KC}_n} \vdash C^{\mathbf{S}}_{(0,(0))}$*, then* $\mathsf{T}^{\omega}_{\mathsf{KC}_n} \vdash C^{\mathbf{R}}_{(0,(0))}$.

PROOF. There exists a formula $A \in \mathcal{A}_{\mathsf{x}}$, such that

$$A[\sim \mathsf{C}\phi \vee \neg \mathsf{E}^k\phi] = \bigvee C^{\mathbf{S}}_{(0,(0))}$$

for some $k \in \omega$ and $A[\sim \mathsf{C}\phi] = \bigvee C^{\mathbf{R}}_{(0,(0))}$. Therefore, the claim holds by clause 3 of Lemma 5.5 and iterations of Lemma 2.3. □

Definition 5.18 *Let* $\mathbf{R}$ *be a sequence tree. The deduction tree of* $\mathbf{R}$ *denoted by* $\mathbb{DT}(\mathbf{R})$ *is the set of all deduction chains of* $\mathbf{R}$*, closed under initial segments. For* $\Theta, \Theta' \in \mathbb{DT}(\mathbf{R})$ *we say* $\Theta \lhd \Theta'$ *if and only if* Θ *is a proper initial segment of* Θ'*. For all finite* $\Theta \in \mathbb{DT}(\mathbf{R})$ *we define last*(Θ) *to be the last sequence tree in* Θ*.*

In order to establish the principal syntactic lemma we require the following consequence of a standard result about the Kleene-Brouwer ordering on a wellfounded tree. Proofs of this result may be found in [4] (Corollary 5.4.18) and [13] (Lemma V.1.3).

Lemma 5.19 *Let* $\mathbf{R}$ *be a sequence tree. If the deduction tree* $\mathbb{DT}(\mathbf{R})$ *contains only finite deduction chains, then there exists an ordinal* α *and a bijective function* $\mathsf{f} : \alpha + 1 \to \mathbb{DT}(\mathbf{R})$*, such that for all ordinals* $\beta, \gamma \leq \alpha$

$$\mathsf{f}(\beta) \lhd \mathsf{f}(\gamma) \implies \gamma < \beta.$$

Lemma 5.20 (Principle syntactic lemma) *If every deduction chain of* $\mathbf{R}$ *ends with an axiomatic sequence tree, then* $\mathsf{T}^{\omega}_{\mathsf{KC}_n} \vdash C^{\mathbf{R}}_{(0,(0))}$.

PROOF. By assumption the deduction tree $\mathbb{DT}(\mathbf{R})$ contains only finite deduction chains. Thus we may apply Lemma 5.19 to obtain a function f and an ordinal α with the described properties. It suffices to show

$$\mathsf{T}^{\omega}_{\mathsf{KC}_n} \vdash C^{\mathrm{last}(\mathsf{f}(\beta))}_{(0,(0))} \tag{16}$$

for all $\beta \leq \alpha$, since $\mathrm{last}(\mathsf{f}(\alpha)) = \mathbf{R}$. We prove (16) by transfinite induction on β.

$\underline{\beta = 0}$: By Lemma 5.19 we find that $\mathsf{f}(\beta)$ must be $\sqsubset$-maximal. Thus by assumption $\mathrm{last}(\mathsf{f}(\beta))$ is axiomatic and the claim follows by Lemma 5.11.

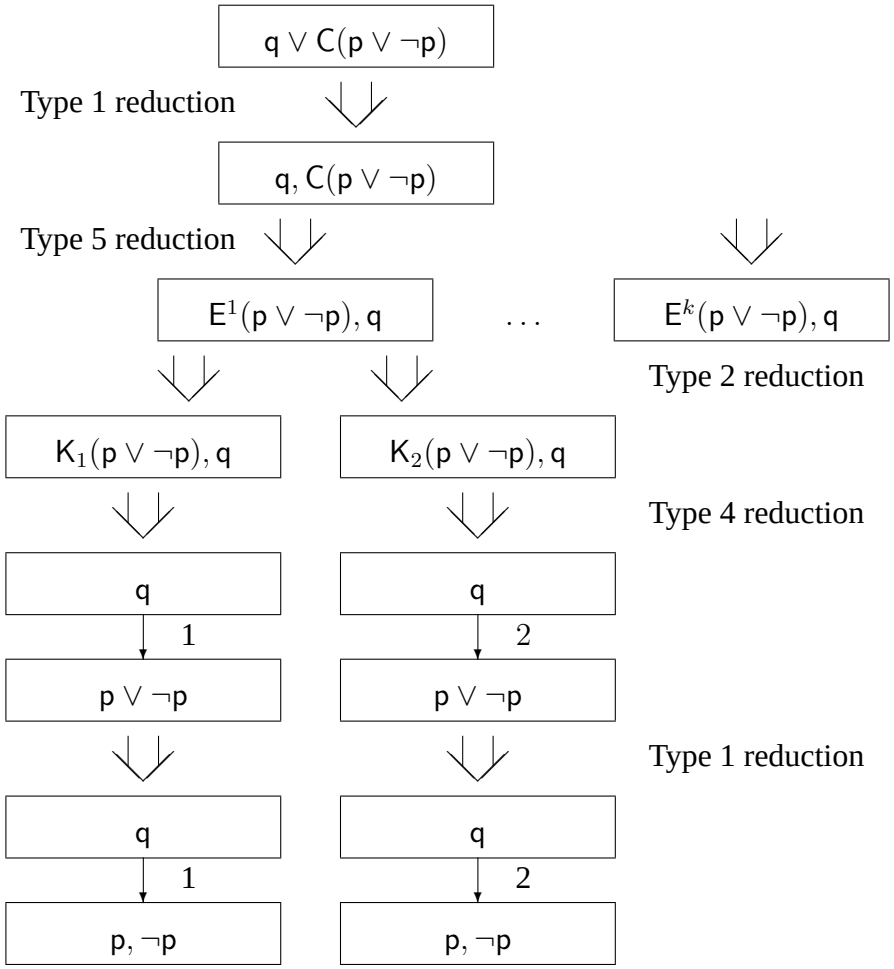


Fig. 8. Example deduction tree

(16) holds for all $\hat{\beta} < \beta$: If $\mathsf{last}(\mathsf{f}(\beta))$ is axiomatic, then the claim holds again by Lemma 5.11. Otherwise $\mathsf{last}(\mathsf{f}(\beta))$ has a redex ϕ. We distinguish between the different possibilities for ϕ and use Lemmata 5.12 – 5.17. The case of $\phi = \psi_1 \vee \psi_2$ is given as an example. In this case there exists an $\mathsf{f}(\gamma)$, such that $\mathsf{last}(\mathsf{f}(\gamma))$ is the type 1 successor of $\mathsf{last}(\mathsf{f}(\beta))$, thus $\mathsf{f}(\beta) \lhd \mathsf{f}(\gamma)$. By Lemma 5.19 we have $\gamma < \beta$ and by induction hypothesis $\mathsf{T}^{\omega}_{\mathsf{KC}_n} \vdash C^{\mathsf{last}(\mathsf{f}(\gamma))}_{(0,(0))}$. Therefore applying Lemma 5.12 yields $\mathsf{T}^{\omega}_{\mathsf{KC}_n} \vdash C^{\mathsf{last}(\mathsf{f}(\beta))}_{(0,(0))}$. The other cases are treated analogously using the induction hypothesis and applications of Lemmata 5.13 – 5.17.

Thus (16) holds for all $\beta \leq \alpha$ and the claim is shown. □

Combining the principle semantic lemma and the principle syntactic lemma yields completeness for $\mathsf{T}^{\omega}_{\mathsf{KC}_n}$.

Corollary 5.21 (Completeness) *Let ϕ be a formula of $\mathcal{L}^n_{\mathsf{C}}$. If for all Kripke struc-*

tures $\mathcal{M}$ and all worlds $w \in \mathcal{M}$ we have that $\mathcal{M}, w \models \phi$, then $\mathsf{T}^{\omega}_{\mathsf{KC}_n} \vdash \phi$.

PROOF. Assume we had $\mathcal{M}, w \models \phi$ for all Kripke structures $\mathcal{M}$ and all worlds $w \in \mathcal{M}$ and ϕ were not provable in $\mathsf{T}^{\omega}_{\mathsf{KC}_n}$. By contraposition of the principal syntactic lemma there would need to exist a deduction chain of ϕ which is infinite or ends non-axiomatically. But in this case the principal semantic lemma would supply us with a countermodel for ϕ, contradicting our assumption. Thus ϕ must be provable in $\mathsf{T}^{\omega}_{\mathsf{KC}_n}$ and indeed the principal syntactic lemma constructs such a proof. □

6 Conclusion

In the current study we have given a syntactic method for proving completeness of the infinitary system $\mathsf{T}^{\omega}_{\mathsf{KC}_n}$ as is stated more precisely in Corollary 5.21. In the case of a valid formula ϕ, a proof of ϕ in $\mathsf{T}^{\omega}_{\mathsf{KC}_n}$ may be reconstructed from the principal syntactic lemma along with Lemmata 5.11 to 5.17 and thus, in this sense, our method is constructive. However, our analysis does not yet provide us with any statements about the length of canonical proofs for valid formulae let alone about whether such proofs are optimal in length. On the semantic side our method also behaves constructively to the extent of providing canonical countermodels for non-valid formulae. This is guaranteed by the principal semantic lemma. It is known from [3] that Logic of Common Knowledge possesses a strong form of the finite model property where the size of a countermodel for a non-valid formula ϕ may be bounded exponentially in the length of ϕ. Currently this result is not reflected in the canonical countermodels constructed by our method, but further refinements should ultimatively lead to the construction of size-optimal countermodels. As mentioned before, the main contribution of this study is the extension of the deduction chain method to Logic of Common Knowledge. In a next step the method could be adapted to other more expressive modal logics with fixed points as well as the modal μ-calculus in its general form [7] and thus contribute to a better proof-theoretical understanding of the area in particular with respect to systematic proof-search and syntactic decision procedures.

An approach similar to the one presented here has recently been undertaken by Tanaka [15] in the framework of predicate common knowledge logic. Let us briefly compare the two studies. Tanaka investigates proof systems for CKL, the predicate common knowledge logic for Kripke frames with constant domain. He introduces an infinitary cut-free deductive system for CKL and proves a completeness theorem about it. Like in our system $\mathsf{T}^{\omega}_{\mathsf{KC}_n}$, Tanaka's rule for introducing the common knowledge operator has infinitely many premises. His deductive system is a kind of tree sequent calculus. That means his system does not derive (sets of) formulae but so-called tree sequents which are finite trees where each node is a sequent and

the edges are labeled by symbols for the agents. A formula ϕ is called derivable if the tree sequent which consists only of the root node $\vdash \phi$ is derivable.

There is a relation between Tanaka's approach and the method of deduction chains: the rules of his calculus correspond to the conditions we impose on deduction chains. Hence, a branch of a derivation in Tanaka's system corresponds to a deduction chain in our approach. In order to prove completeness, he only needs to show the analogue of our principal semantic lemma: given a non-derivable tree sequent, it is possible to construct a countermodel. Since we work in the Tait-style system $\mathsf{T}^{\omega}_{\mathsf{KC}_n}$ which derives sets of formulae and not tree sequents, we also need the principal syntactic lemma. This lemma states that if every deduction chain of a formula ϕ ends axiomatically, then it is provable in $\mathsf{T}^{\omega}_{\mathsf{KC}_n}$. That could be translated into something like if ϕ is derivable in Tanaka's system, then it is provable in $\mathsf{T}^{\omega}_{\mathsf{KC}_n}$.

Acknowledgements

We would like to thank the anonymous referee and Carlos Areces for many comments which helped to improve the presentation of our results.

References

[1] Luca Alberucci. *The Modal μ-Calculus and the Logics of Common Knowledge*. PhD thesis, Institut für Informatik und angewandte Mathematik, Universität Bern, 2002.

[2] Luca Alberucci and Gerhard Jäger. About cut elimination for logics of common knowledge. To appear in *Annals of Pure and Applied Logic*.

[3] Ronald Fagin, Joseph Y. Halpern, Yoram Moses, and Moshe Y. Vardi. *Reasoning about Knowledge*. MIT Press, 1995.

[4] Jean-Yves Girard. *Proof Theory and Logical Complexity*. Bibliopolis, 1987.

[5] Joseph Y. Halpern and Yoram Moses. Knowledge and common knowledge in a distributed environment. *Journal of the ACM*, 37(3):549–587, 1990.

[6] Gerhard Jäger and Thomas Strahm. Bar induction and omega model reflection. *Annals of Pure and Applied Logic*, 97:221–230, 1999.

[7] Dexter Kozen. Results on the propositional μ-calculus. *Theoretical Computer Science*, 27:333–354, 1983.

[8] Wolfram Pohlers. Subsystems of set theory and second order number theory. In Sam Buss, editor, *Handbook of Proof Theory*, pages 209–335. Elsevier, 1998.

[9] Kurt Schütte. *Beweistheorie*. Springer, 1960.

[10] Kurt Schütte. *Vollständige Systeme modaler und intuitionistischer Logik*. Springer, 1968.

[11] Kurt Schütte. *Proof Theory*. Springer, 1977.

[12] Helmut Schwichtenberg. Proof theory: Some applications of cut-elimination. In Jon Barwise, editor, *Handbook of Mathematical Logic*, pages 867–895. North-Holland, 1977.

[13] Stephen Simpson. *Subsystems of Second Order Arithmetic*. Springer, 1998.

[14] William Tait. Normal derivability in classical logic. In Jon Barwise, editor, *The Syntax and Semantics of Infinitary Languages*, pages 204–236. Springer, Berlin, 1968.

[15] Yoshihito Tanaka. Some proof systems for predicate common knowledge logic. *Reports on Mathematical Logic*, 37:79–100, 2003.