# A Logical Approach to CTL

Gilles Dowek, Ying Jiang

# A Logical Approach to CTL

Gilles Dowek[1] and Ying Jiang[2]

[1] INRIA
23 avenue d'Italie, CS 81321, 75214 Paris Cedex 13, France
gilles.dowek@inria.fr
[2] State Key Laboratory of Computer Science
Institute of Software
Chinese Academy of Sciences
P.O. Box 8718, 100190 Beijing, China
jy@ios.ac.cn

**Abstract.** We introduce a notion of proof for CTL with respect to a given finite model and show some advantages of using such a notion. This also suggests to define a slight extension of CTL, called SCTL, where predicates can have an arbitrary arity.

## 1 Introduction

Model-checking and proof-checking are ubiquitous in formal methods. They differ by the fact that proof-checking mostly uses undecidable theories, such as first-order logic, arithmetic, type theory, set theory, etc., while model-checking uses decidable theories, such as propositional modal logic, LTL, CTL, etc. When a theory is decidable, the notion of truth can be defined by an algorithm and the notion of proof is not needed.

However, even for decidable theories, a notion of proof may be useful. First, once the truth of a formula is established by a decision algorithm, a proof may be seen as a trace of the execution of this algorithm, and checking the correction of this proof requires less computation than checking the validity of the formula again, in the same way that checking some number is a divisor of a large natural number requires less computation than finding a divisor. Thus a proof may be a good way to remember or to communicate that a formula is valid. Defining such a notion of proof also permits to use proof-search algorithms instead of model-checking ones. Finally, a proof theoretical approach may also be useful to extend these logics to non decidable cases.

In this work we focus on CTL parametrized by a given finite model and introduce a notion of proof for this logic. While a provable proposition is usually valid in all models, we show here how to customize the notion of proof for a specific model.

This work has also suggested a slight extension of CTL, called SCTL, where predicates may have an arbitrary arity. Strangely, this extension of CTL is not more complex that CTL itself.

### 1.1 Kripke models

A possible notion of model for modal logic is that of a Kripke model. In general, a Kripke model for a language $\mathcal{L}$ is given by

- a set $S$ of *states* (or *worlds*), equipped with an *accessibility* relation $\longrightarrow$,
- for each state $s$, a domain of interpretation,
- for each state $s$ and symbol $P$ of the language $\mathcal{L}$, an interpretation $\hat{P}^s$ of the symbol $P$ in the state $s$.

In the particular case of propositional modal logics, the domains of interpretations are no longer needed and the definition boils down to

- a set $S$ of *states*, equipped with an *accessibility* relation $\longrightarrow$,
- for each state $s$ and proposition symbol $P$ of the language, an interpretation $\hat{P}^s \in \{0, 1\}$ of the symbol $P$ in the state $s$ or, equivalently, for each proposition symbol $P$ a set of states $\hat{P}$ (the set $\{s \mid \hat{P}^s = 1\}$).

Then, the validity of a formula $\phi$ in a given state $s$ of the model (written $s \Vdash \phi$) is defined by induction on the structure of the formula. Typically,

- $s \Vdash P$ if $s \in \hat{P}$ (that is $\hat{P}^s = 1$),
- $s \Vdash \phi_1 \wedge \phi_2$ if $s \Vdash \phi_1$ and $s \Vdash \phi_2$,
- $s \Vdash \diamond\phi$ if there exists an $s'$ such that $s \longrightarrow^* s'$ and $s' \Vdash \phi$, where $\longrightarrow^*$ is the transitive closure of $\longrightarrow$,
- etc.

### 1.2 From Kripke models to models of predicate logic

A proposition symbol $P$ is thus interpreted in a Kripke model as a set $\hat{P}$ of states, much like a unary predicate symbol would be interpreted in an ordinary model, whose domain would be the set $S$ of states of the Kripke model. Introducing a constant for each state, and considering $P$ not as a proposition symbol but as a unary predicate symbol, we can express the judgement $s \Vdash P$ as a formula $P(s)$ and the validity of this formula in a model would be defined as

- $\models P(s)$ if $s \in \hat{P}$,
- $\models \phi_1 \wedge \phi_2$ if $\models \phi_1$ and $\models \phi_2$,
- etc.

like in the case of predicate logic. The only specificity of such a model is that it is equipped with an accessibility relation that is used to define the interpretation of modalities. For instance, if $P$ is a unary predicate symbol, we can build the formula $(\diamond\ P)(s)$ and transform the clause above into

- $\models (\diamond\ P)(s)$ if there exists a $s'$ such that $s \longrightarrow^* s'$ and $\models P(s')$.

Note that replacing the judgement $s \Vdash P$ by the formula $P(s)$ permits to speak about specific states of the Kripke model in the language itself, which is not usual in modal logics. This transformation can be compared to the introduction of adverbial phrases in natural languages. For instance the sentence $P$ "The sky is blue" can be true or false at different moments. Introducing a tense such as the "The sky will be blue" corresponds to the introduction of a modality such as $\diamond P$ and the introduction of an adverbial phrase "The sky will be blue Monday January 1st" corresponds to the introduction of an explicit state $P(s)$.

A next step could be to introduce, in the language, a symbol for the transition relation [1], but this alone would not permit to express the inductive and co-inductive modalities. Thus, we prefer to keep this symbol implicit and introduce modalities such as $AX$, $EX$, $EF$, etc. as primitive symbols. For instance, the formula $EF(P)(s)$ (written $(\diamond P)(s)$ above) expresses the existence of a sequence $s_0, s_1, ...$ starting from $s$ and a natural number $n$ such that for all $i$, $s_i \longrightarrow s_{i+1}$ and, $P(s_n)$ holds. The formula $EG(P)(s)$ expresses the existence of a sequence $s_0, s_1, ...$ starting from $s$ such that for all $i$, $s_i \longrightarrow s_{i+1}$ and for all $i$, $P(s_i)$ holds, etc.

In modal logics, it is possible to apply a modality, such as $\diamond$, to an atomic formula, but also to a compound formula, for instance $\diamond(P \wedge Q)$. In the same way, in CTL, we can apply the modality $EF$ to a compound formula $EF(P \wedge Q)(s)$. When expressing CTL in predicate logic, $P$ and $Q$ are not proposition symbols anymore but unary predicate symbols and it does make sense to apply the conjunction $\wedge$ to unary predicate symbols. Thus it is natural to apply the modality $EF$ not to predicates symbols but to formulae: $EF(P(x))(s)$, $EF(P(x) \wedge Q(x))(s)$, etc. In this case, the symbol $EF$ must bind the variable $x$ in its first argument, like a quantifier, and we must write $EF_x(P(x))(s)$ and $EF_x(P(x) \wedge Q(x))(s)$ instead of $EF(P(x))(s)$ and $EF(P(x) \wedge Q(x))(s)$.

### 1.3 Predicate symbols of arbitrary arity

When expressing CTL this way, it is natural to introduce not only unary predicate symbols, such as $P$, but also predicates of arbitrary arity, for instance binary predicate symbols, while this would not make much sense in a Kripke model, where the central notion is the validity of a formula in a state ($s \Vdash \phi$). For instance, we may want to express the existence of a sequence $s_0, s_1, ...$ starting from $s$ such that for all $i$, $s_i \longrightarrow s_{i+1}$ and such that one can buy a left shoe at some state $s_n$ and then the right shoe *of the same pair* at a later state $s_p$. This requires to introduce a binary predicate symbol $P$, that relates these sates, and express it with the formula $EF_x(EF_y(P(x,y))(x))(s)$.

Introducing such predicate symbols of arbitrary arity extends the expressivity of CTL but not its complexity.

The first contribution of this paper is to introduce a system SCTL extending CTL with predicate symbols of arbitrary arity. In this system, modalities are applied to formulae and states, binding variables in these formulae. We will define a notion of model for this logic and focus on the case of finite models.

### 1.4 Models and proofs

The second contribution of this paper is to define a sequent calculus to write proofs in SCTL. Unlike the usual sequent calculus, where a formula is provable if and only if it is valid in all models, we design a sequent calculus tailored for each specific finite model $\mathcal{M}$: a formula is provable in this sequent calculus if and only it it is valid in the model $\mathcal{M}$.

Of course, such a sequent calculus can only be defined when the validity in the model $\mathcal{M}$ is semi-decidable. For instance there is no such sequent calculus for the standard model of arithmetic. But it can be defined for any finite model.

Such a proof-theoretic definition of truth is complementary from a model-theoretic one, although both are effective.

First, once the truth of a formula $EF_x(P(x))(s)$ is established, by finding a finite sequence $s_0, s_1, ..., s_n$ starting from $s$ such that for all $i < n$, $s_i \longrightarrow s_{i+1}$ and such that $P(s_n)$, a proof may be a good way to remember, or to communicate, this sequence.

Defining such a notion of proof also permits to use proof-search algorithms instead of model-checking ones, and, in some cases, proof-search may be more efficient than model-checking. Assume, for instance, that a state $s$ has immediate successors $s_1$, $s_2$, and $s_3$ and the predicate $P$ is verified by $s_1$ and $s_2$ but not by $s_3$. We want to prove the proposition $EX_x(\neg P(x))(s)$. A model checking-algorithm would enumerate all the successors of $s$, while a proof-search algorithm would introduce a variable $x$ for such a state and attempt to prove $\neg P(x)$, under the axioms $P(s_1)$, $P(s_2)$, and $\neg P(s_3)$, with the constraint that $x$ must be substituted with an element of $Next(s)$. The resolution method, for instance, would attempt to unifiy $P(x)$ with a $P(s_3)$ and directly find the substitution $x = s_3$ without enumerating $s_1$ and $s_2$. Of course, the generality of this example remains to be investigated.

Finally, a proof theoretical approach may also be useful to extend CTL for models that are infinite, but simpler than the standard model of arithmetic, such as pushdown systems.

Proof systems for modal logics have been defined. See, for instance, [2–6]. When designing such a proof-system for modal logic, one of the main issues is to handle co-inductive modalities, for instance asserting the existence of an infinite sequence whose elements all verify some property. It is tempting to reflect this infinite sequence as an infinite proof and then use the finiteness of the model to prune the search-tree in a proof-search method. Instead, we use the finiteness of the model to keep our proofs finite, like in the usual sequent calculus. This is the purpose of the *merge* rules.

## 2 Models and formulae

The models we consider in this work are Kripke models.

**Definition 1.** *A* Kripke model $\mathcal{M}$ *is given by*

– *a non empty set $S$, whose elements are called* states*,*
– *a binary relation $\longrightarrow$ defined on $S$, such that for each $s$ in $S$, there exists at least one $s'$ in $S$, such that $s \longrightarrow s'$,*
– *and a family of subsets of $S^n$, where $n$ is a natural number, called* relations*.*

We write $Next(s)$ for the set $\{s' \mid s \longrightarrow s'\}$. A *path* is an infinite sequence $s_0, s_1, ...$ of states such that for each $i$, $s_{i+1} \in Next(s_i)$.

Properties of such a model are expressed in a language, tailored for this model, that contains

– for each state $s$, a constant, also written $s$,
– for each relation $P$, a predicate symbol, also written $P$.

Formulae are built in the usual way with the connectors $\top$, $\bot$, $\wedge$, $\vee$ and $\neg$, to which we add modalities $AX$, $EX$, $AF$, $EG$, $AR$, and $EU$. If $\phi$ is a formula, and $t$ is either a constant or a variable, then $AX_x(\phi)(t)$, $EX_x(\phi)(t)$, $AF_x(\phi)(t)$, and $EG_x(\phi)(t)$ are formulae. Like quantifiers, modalities bind the variable $x$ in $\phi$. If $\phi_1$ and $\phi_2$ are formulae and $t$ is either a constant or a variable, then $AR_{x,y}(\phi_1, \phi_2)(t)$ and $EU_{x,y}(\phi_1, \phi_2)(t)$ are formulae. These modalities bind the variable $x$ in $\phi_1$ and $y$ in $\phi_2$.

We consider only formulae in negative normal form, that is formulae where negation is applied to atoms only. It is routine to check that each formula $\phi$ can be transformed into an equivalent formula $|\phi|$ in negative normal form.

– $|\phi| = \phi$, if $\phi$ is atomic,
– $|\top| = \top$,
– $|\bot| = \bot$,
– $|\phi_1 \wedge \phi_2| = |\phi_1| \wedge |\phi_2|$,
– $|\phi_1 \vee \phi_2| = |\phi_1| \vee |\phi_2|$,
– $|AX_x(\phi_1)(t)| = AX_x(|\phi_1|)(t)$,
– $|EX_x(\phi_1)(t)| = EX_x(|\phi_1|)(t)$,
– $|AF_x(\phi_1)(t)| = AF_x(|\phi_1|)(t)$,
– $|EG_x(\phi_1)(t)| = EG_x(|\phi_1|)(t)$,
– $|AR_x(\phi_1, \phi_2)(t)| = AR_x(|\phi_1|, |\phi_2|)(t)$,
– $|EU_x(\phi_1, \phi_2)(t)| = EU_x(|\phi_1|, |\phi_2|)(t)$,
– $|\neg\phi_1| = |\phi_1|^\bot$,

where $\phi^\bot$ is defined by

– $\phi^\bot = \neg\phi$, if $\phi$ is atomic,
– $(\neg\phi_1)^\bot = \phi_1$, if $\phi_1$ is atomic,
– $\top^\bot = \bot$,
– $\bot^\bot = \top$,
– $(\phi_1 \wedge \phi_2)^\bot = \phi_1^\bot \vee \phi_2^\bot$,
– $(\phi_1 \vee \phi_2)^\bot = \phi_1^\bot \wedge \phi_2^\bot$,
– $(AX_x(\phi_1)(t))^\bot = EX_x(\phi_1^\bot)(t)$,
– $(EX_x(\phi_1)(t))^\bot = AX_x(\phi_1^\bot)(t)$,
– $(AF_x(\phi_1)(t))^\bot = EG_x(\phi_1^\bot)(t)$,

- $(EG_x(\phi_1)(t))^\perp = AF_x(\phi_1^\perp)(t)$,
- $(AR_x(\phi_1,\phi_2)(t))^\perp = EU_x(\phi_1^\perp,\phi_2^\perp)(t)$,
- $(EU_x(\phi_1,\phi_2)(t))^\perp = AR_x(\phi_1^\perp,\phi_2^\perp)(t)$.

Finally, we use the following abbreviations

- $\phi_1 \Rightarrow \phi_2 = \phi_1^\perp \vee \phi_2$,
- $EF_x(\phi)(t) = EU_{z,x}(\top,\phi)(t)$,
- $ER_{x,y}(\phi_1,\phi_2)(t) = (EU_{y,z}(\phi_2,((z/x)\phi_1 \wedge (z/y)\phi_2))(t) \vee EG_y(\phi_2)(t))$, where $z$ is a variable that occurs neither in $\phi_1$ nor in $\phi_2$,
- $AG_x(\phi)(t) = (EF_x(\phi^\perp)(t))^\perp$,
- $AU_{x,y}(\phi_1,\phi_2)(t) = (ER_{x,y}(\phi_1^\perp,\phi_2^\perp)(t))^\perp$.

**Definition 2.** *Let $\mathcal{M}$ be a model and $\phi$ be a closed formula, the set of valid formulae $\models \phi$ in the model $\mathcal{M}$ is defined by induction on $\phi$ as follows*

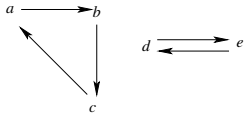
- $\models P(s_1,...,s_n)$, *if* $\langle s_1,...,s_n\rangle \in P$,
- $\models \neg P(s_1,...,s_n)$, *if* $\langle s_1,...,s_n\rangle \notin P$,
- $\models \top$,
- $\models \perp$ *is never the case,*
- $\models \phi_1 \wedge \phi_2$ *if* $\models \phi_1$ *and* $\models \phi_2$,
- $\models \phi_1 \vee \phi_2$ *if* $\models \phi_1$ *or* $\models \phi_2$,
- $\models AX_x(\phi_1)(s)$, *if for each state* $s'$ *in* $Next(s)$, $\models (s'/x)\phi_1$,
- $\models EX_x(\phi_1)(s)$, *if there exists a state* $s'$ *in* $Next(s)$ *such that* $\models (s'/x)\phi_1$,
- $\models AF_x(\phi_1)(s)$, *if for all paths* $s_0, s_1, ...$ *such that* $s_0 = s$ *there exists a natural number* $i$, *such that* $\models (s_i/x)\phi_1$,
- $\models EG_x(\phi_1)(s)$, *if there exists a path* $s_0, s_1, ...$ *such that* $s_0 = s$, *and for all natural numbers* $i$, $\models (s_i/x)\phi_1$,
- $\models AR_{x,y}(\phi_1,\phi_2)(s)$, *if for all paths* $s_0, s_1, ...$ *such that* $s_0 = s$, *for all* $j$, *either* $\models (s_j/y)\phi_2$ *or there exists an* $i < j$ *such that* $\models (s_i/x)\phi_1$,
- $\models EU_{x,y}(\phi_1,\phi_2)(s)$, *if there exists a path* $s_0, s_1, ...$ *such that* $s_0 = s$, *there exists a natural number* $j$ *such that* $\models (s_j/y)\phi_2$, *and for all* $i < j$, $\models (s_i/x)\phi_1$.

*Remark 1.* From the definitions above, we obtain $\models EF_x(\phi)(s)$, if there exists a path $s_0, s_1, ...$ such that $s_0 = s$, and a natural number $j$ such that $\models (s_j/x)\phi$, etc.

*Example 1.* The formula

$$EF_x(EF_y(P(x,y))(x))(a)$$

expresses the existence of a path starting from $a$, that contains two states related by $P$. This formula is valid in a model formed with the relation depicted below



and the set $P = \{\langle a, c\rangle\}$, but not in that formed with the same relation and the set $P = \{\langle a, d\rangle\}$ instead.

*Remark 2.* An alternative definition of $\models AF_x(\phi_1)(s)$, $\models AR_{x,y}(\phi_1, \phi_2)(s)$, and $\models EU_{x,y}(\phi_1, \phi_2)(s)$ is the following.

- $\models AF_x(\phi_1)(s)$, if there exists a finite tree $T$ such that $T$ has root $s$; for each internal node $s'$, the children of this node are labeled by the elements of $Next(s')$; and for each leaf $s'$, $\models (s'/x)\phi_1$,
- $\models AR_{x,y}(\phi_1, \phi_2)(s)$, if there exists an infinite tree $T$ such that $T$ has root $s$; for each internal node $s'$, the children of this node are labeled by the elements of $Next(s')$; for each node $s'$, $\models (s'/y)\phi_2$; and for each leaf $s'$, $\models (s'/x)\phi_1$,
- $\models EU_{x,y}(\phi_1, \phi_2)(s)$, if there exists a finite sequence $s_0, ..., s_n$ such that $s_0 = s$; $\models (s_n/y)\phi_2$; and for all $i$ between 0 and $n-1$, $\models (s_i/x)\phi_1$.

## 3 Proofs

We now consider a fixed finite model $\mathcal{M}$. As the model is finite, the sets $Next(s)$ are always finite.

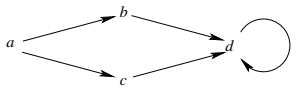
Consider, for instance, the formula $AF_x(P(x))(s)$. This formula is valid if there exists a finite tree $T$ whose root is labeled by $s$, such that the children of an internal node labeled by a state $a$ are labeled by the elements of $Next(a)$, and such that all the leaves are in $P$. Such a tree can be called a *proof* of the formula $AF_x(P(x))(s)$.

Consider now the formula $AF_x(AF_y(P(x,y))(x))(s)$ that contains nested modalities. To justify the validity of this formula, one needs to provide a tree, where at each leaf $a$, the formula $AF_y(P(a,y))(a)$ is valid. And to justify the validity of the formulae $AF_y(P(a,y))(a)$, one needs to provide other trees. These hierarchical trees can be formalized with the sequent calculus rules

$$\frac{\vdash (s/x)\phi}{\vdash AF_x(\phi)(s)}\ \mathbf{AF}\text{-right}$$

$$\frac{\vdash AF_x(\phi)(s_1)\ ...\ \vdash AF_x(\phi)(s_n)}{\vdash AF_x(\phi)(s)}\ \begin{array}{l}\mathbf{AF}\text{-right}\\ \{s_1...s_n\} = Next(s)\end{array}$$

*Example 2.* Consider the model formed with the relation




and the set $P = \{b, c\}$. A proof of the formula $AF_x(P(x))(a)$ is

$$\dfrac{\dfrac{\dfrac{}{\vdash P(b)}\ \text{atom-right}}{\vdash AF_x(P(x))(b)}\ \mathbf{AF}\text{-right} \quad \dfrac{\dfrac{}{\vdash P(c)}\ \text{atom-right}}{\vdash AF_x(P(x))(c)}\ \mathbf{AF}\text{-right}}{\vdash AF_x(P(x))(a)}\ \mathbf{AF}\text{-right}$$

where besides the rules **AF**-right, we use the rule

$$\frac{}{\vdash P(s_1, ..., s_n)}\ \text{atom-right}\ \langle s_1, ..., s_n\rangle \in P$$

The above discussion leads to the sequent calculus depicted in Figure 1.

Note that in the usual sequent calculus, the right rule of disjunction can either be formulated in a multiplicative way

$$\frac{\Gamma \vdash \phi_1, \phi_2, \Delta}{\Gamma \vdash \phi_2 \vee \phi_2, \Delta}$$

or in an additive way

$$\frac{\Gamma \vdash \phi_1, \Delta}{\Gamma \vdash \phi_1 \vee \phi_2, \Delta}$$

$$\frac{\Gamma \vdash \phi_2, \Delta}{\Gamma \vdash \phi_1 \vee \phi_2, \Delta}$$

These two formulations are equivalent in presence of structural rules. For instance the proof of the sequent $\vdash P \Rightarrow P$, that is $\vdash \neg P \vee P$, in the multiplicative system

$$\dfrac{\dfrac{\dfrac{}{P \vdash P}\ \text{axiom}}{\vdash \neg P, P}\ \neg\text{-right}}{\vdash \neg P \vee P}\ \vee\text{-right}$$

can be rewritten in the additive one

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{P \vdash P}\ \text{axiom}}{\vdash \neg P, P}\ \neg\text{-right}}{\vdash \neg P, \neg P \vee P}\ \vee\text{-right}}{\vdash \neg P \vee P, \neg P \vee P}\ \vee\text{-right}}{\vdash \neg P \vee P}\ \text{contraction-right}$$

The contraction rule, or the multiplicative rule, is needed to build the sequent $\vdash \neg P, P$ that is provable even if none of its weakenings $\vdash \neg P$ and $\vdash P$ is, because we do not know whether the formula $P$ is true or false.

Our sequent calculus needs neither contraction rules nor multiplicative rules, because for each atomic formula $P$, either $P$ is provable or $\neg P$ is. Therefore the sequent $\vdash \neg P \vee P$ is proved by proving either the sequent $\vdash \neg P$ or the sequent $\vdash P$.

As we have neither multiplicative rules nor structural rules, if we start with a sequent $\vdash \phi$, then each sequent in the proof has one formula on the right and none on the left. Thus, the lists $\Gamma$ and $\Delta$ are empty and the rules can be formulated as

$$\frac{\vdash \phi_1}{\vdash \phi_1 \vee \phi_2}$$

$$\frac{}{\vdash P(s_1,...,s_n)}\ \text{atom-right}\ \langle s_1,...,s_n\rangle \in P$$

$$\frac{}{\vdash \neg P(s_1,...,s_n)}\ \neg\text{-right}\ \langle s_1,...,s_n\rangle \notin P$$

$$\frac{}{\vdash \top}\ \top\text{-right}$$

$$\frac{\vdash \phi_1 \qquad \vdash \phi_2}{\vdash \phi_1 \wedge \phi_2}\ \wedge\text{-right}$$

$$\frac{\vdash \phi_1}{\vdash \phi_1 \vee \phi_2}\ \vee\text{-right}$$

$$\frac{\vdash \phi_2}{\vdash \phi_1 \vee \phi_2}\ \vee\text{-right}$$

$$\frac{\vdash (s_1/x)\phi \quad ... \quad \vdash (s_n/x)\phi}{\vdash AX_x(\phi)(s)}\ \mathbf{AX}\text{-right}\ \{s_1...s_n\} = Next(s)$$

$$\frac{\vdash (s'/x)\phi}{\vdash EX_x(\phi)(s)}\ \mathbf{EX}\text{-right}\ s' \in Next(s)$$

$$\frac{\vdash (s/x)\phi}{\vdash AF_x(\phi)(s)}\ \mathbf{AF}\text{-right}$$

$$\frac{\vdash AF_x(\phi)(s_1)\ ...\ \vdash AF_x(\phi)(s_n)}{\vdash AF_x(\phi)(s)}\ \mathbf{AF}\text{-right}\ \{s_1...s_n\} = Next(s)$$

$$\frac{\vdash (s/x)\phi \qquad \Gamma, EG_x(\phi)(s) \vdash EG_x(\phi)(s')}{\Gamma \vdash EG_x(\phi)(s)}\ \mathbf{EG}\text{-right}\ s' \in Next(s)$$

$$\frac{}{\Gamma \vdash EG_x(\phi)(s)}\ \mathbf{EG}\text{-merge}\ EG_x(\phi)(s) \in \Gamma$$

$$\frac{\vdash (s/x)\phi_1 \qquad \vdash (s/y)\phi_2}{\Gamma \vdash AR_{x,y}(\phi_1,\phi_2)(s)}\ \mathbf{AR}\text{-right}$$

$$\frac{\vdash (s/y)\phi_2 \qquad \Gamma' \vdash AR_{x,y}(\phi_1,\phi_2)(s_1) \quad ... \quad \Gamma' \vdash AR_{x,y}(\phi_1,\phi_2)(s_n)}{\Gamma \vdash AR_{x,y}(\phi_1,\phi_2)(s)}\ \begin{array}{l}\mathbf{AR}\text{-right}\\ \{s_1,...,s_n\} = Next(s)\\ \Gamma' = \Gamma, AR_{x,y}(\phi_1,\phi_2)(s)\end{array}$$

$$\frac{}{\Gamma \vdash AR_{x,y}(\phi_1,\phi_2)(s)}\ \mathbf{AR}\text{-merge}\ AR_{x,y}(\phi_1,\phi_2)(s) \in \Gamma$$

$$\frac{\vdash (s/y)\phi_2}{\vdash EU_{x,y}(\phi_1,\phi_2)(s)}\ \mathbf{EU}\text{-right}$$

$$\frac{\vdash (s/x)\phi_1 \qquad \vdash EU_{x,y}(\phi_1,\phi_2)(s')}{\Gamma \vdash EU_{x,y}(\phi_1,\phi_2)(s)}\ \mathbf{EU}\text{-right}\ s' \in Next(s)$$

**Fig. 1.** SCTL

$$\frac{\vdash \phi_2}{\vdash \phi_1 \vee \phi_2}$$

Finally, as all sequents have the form $\vdash \phi$, the left rules and the axiom rule can be dropped as well. In other words, unlike the usual sequent calculus, our sequent

calculus, like Hilbert systems, is tailored for deduction, not for hypothetical deduction.

The case of co-inductive formulae, for instance $EG_x(P(x))(s)$, is more complex than that of the inductive formulae, such as $AF_x(P(x))(s)$. To justify its validity, one needs to provide an infinite sequence, that is an infinite tree with only one branch, such that the root of the tree is labeled by $s$, the child of a node labeled by a state $a$ is labeled by an element of $Next(a)$, and each node of the tree verifies $P$. But, as the model is finite, we can always restrict to regular trees and use a finite representation of such trees. This leads us to introduce a rule, called **EG**-*merge*, that permits to prove a sequent of the form $\vdash EG_x(P(x))(s)$, provided such a sequent already occurs lower in the proof. To make this rule local, we re-introduce hypotheses $\Gamma$ to record part of the history of the proof. The sequents have therefore the form $\Gamma \vdash \phi$, with a non empty $\Gamma$ in this particular case only, and the **EG**-merge rule is then just an instance of the axiom rule.

## 4 Soundness and completeness

Propositions 1 and 2 below permit to transform finite structures into infinite ones and will be used in the Soundness proof, while Propositions 3 and 4 permit to transform infinite structures into finite ones and will be used in the Completeness proof.

**Proposition 1 (Finite to infinite sequences).** *Let $s_0, ..., s_n$ be a finite sequence of states such that for all $i$ between $0$ and $n-1$, $s_i \longrightarrow s_{i+1}$, and $s_n = s_p$ for some $p$ between $0$ and $n-1$. Then there exists an infinite sequence of states $s'_0, s'_1, ...$ such that for all $i$, $s'_i \longrightarrow s'_{i+1}$, and all the $s'_j$ are among $s_0, ...., s_n$.*

*Proof.* Take the sequence $s_0, ..., s_{p-1}, s_p, ..., s_{n-1}, s_p, ..., s_{n-1}, ...$

**Proposition 2 (Finite to infinite trees).** *Let $\Phi$ be a set of states and $T$ be a finite tree labeled by states such that, for each internal node $s$, the immediate successors of $s$ are the elements of $Next(s)$ and each leaf is labeled with a state which is either in $\Phi$ or also a label of a node on the branch from the root of $T$ to this leaf. Then there exists an infinite tree $T'$ labeled by states such that for each internal node $s$ the successors of $s$ are the elements of $Next(s)$, all the leaves are labeled by elements of $\Phi$, and all the labels of $T'$ are labels of $T$.*

*Proof.* Consider for $T'$ the tree whose root is labeled by the root of $T$ and such that for each node $s$, if $s$ is in $\Phi$, then $s$ is a leaf of $T'$, otherwise the successors of $s$ are the elements of $Next(s)$. It is easy to check that all the nodes of $T'$ are labeled by labels of $T$.

**Proposition 3 (Infinite to finite sequences).** *Let $s_0, s_1, ...$ be an infinite sequence of states such that for all $i$, $s_i \longrightarrow s_{i+1}$. Then there exists a finite sequence of states $s'_0, ..., s'_n$ such that for all $i$ between $0$ and $n-1$, $s'_i \longrightarrow s'_{i+1}$, $s'_n = s'_p$ for some $p$ between $0$ and $n-1$, and all the $s'_j$ are among $s_0, s_1, ....$*

*Proof.* As the number of states is finite, there exists $p$ and $n$ such that $p < n$ and $s_p = s_n$. Take the sequence $s_0, ..., s_n$.

**Proposition 4 (Infinite to finite trees).** *Let $\Phi$ be a set of states and $T$ be an infinite tree labeled by states such that for each internal node $s$ the successors of $s$ are the elements of $Next(s)$ and each leaf is labeled by a state in $\Phi$. Then, there exists a finite tree labeled by states such that for each internal node $s$ the successors of $s$ are the elements of $Next(s)$ and each leaf is labeled with a state which is either in $\Phi$ or also a label of a node on the branch from the root of $T$ to this leaf.*

*Proof.* As the number of states is finite, on each infinite branch, there exists $p$ and $n$ such that $p < n$ and $s_p = s_n$. Prune the tree at node $s_n$. This tree is finitely branching and each branch is finite, hence, by Köning's lemma, it is finite.

**Definition 3.** *Let $s$ be a state and $T_1, ..., T_n$ be trees labeled by states. We write $s(T_1, ..., T_n)$ for the tree whose root is labeled by $s$ and whose immediate subtrees are $T_1, ..., T_n$.*

**Proposition 5 (Soundness).** *Let $\phi$ be a closed formula. If the sequent $\vdash \phi$ has a proof $\pi$, then $\models \phi$.*

*Proof.* By induction on the structure of the proof $\pi$.

- If the last rule of $\pi$ is atom-right, then the proved sequent has the form $\vdash P(s_1, ..., s_n)$, hence $\models P(s_1, ..., s_n)$.
- If the last rule of $\pi$ is $\neg$-right, then the proved sequent has the form $\vdash \neg P(s_1, ..., s_n)$, hence $\models \neg P(s_1, ..., s_n)$.
- If the last rule of $\pi$ is $\top$-right, the proved sequent has the form $\vdash \top$ and $\models \top$.
- If the last rule of $\pi$ is $\wedge$-right, then the proved sequent has the form $\vdash \phi_1 \wedge \phi_2$. By induction hypothesis $\models \phi_1$ and $\models \phi_2$, hence $\models \phi_1 \wedge \phi_2$.
- If the last rule of $\pi$ is $\vee$-right, then the proved sequent has the form $\vdash \phi_1 \vee \phi_2$. By induction hypothesis $\models \phi_1$ or $\models \phi_2$, hence $\models \phi_1 \vee \phi_2$.
- If the last rule of $\pi$ is **AX**-right, then the proved sequent has the form $\vdash AX_x(\phi_1)(s)$. By induction hypothesis, for each $s'$ in $Next(s)$, $\models (s'/x)\phi_1$, hence $\models AX_x(\phi_1)(s)$.
- If the last rule of $\pi$ is **EX**-right, then the proved sequent has the form $\vdash EX_x(\phi_1)(s)$. By induction hypothesis, there exists a state $s'$ in $Next(s)$, such that $\models (s'/x)\phi_1$, hence $\models EX_x(\phi_1)(s)$.
- If the last rule of $\pi$ is **AF**-right, then the proved sequent has the form $\vdash AF_x(\phi_1)(s)$. We associate a finite tree $|\pi|$ to the proof $\pi$ by induction in the following way.
  - If the proof $\pi$ ends with the first **AF**-right rule with a subproof $\rho$ of the sequent $\vdash (s/x)\phi_1$, then the tree contains a single node $s$.

- If the proof $\pi$ ends with the second **AF**-right rule, with subproofs $\pi_1$, ..., $\pi_n$ of the sequents $\vdash AF_x(\phi_1)(s_1)$, ..., $\vdash AF_x(\phi_1)(s_n)$, respectively, then $|\pi|$ is the tree $s(|\pi_1|, ..., |\pi_n|)$.

The tree $|\pi|$ has root $s$; for each internal node $s'$, the children of this node are labeled by the elements of $Next(s')$; and for each leaf $s'$ the sequent $\vdash (s'/x)\phi_1$ has a proof smaller than $\pi$. By induction hypothesis, for each leaf $s'$ of $|\pi|$, $\models (s'/x)\phi_1$. Hence $\models AF_x(\phi_1)(s)$.

– If the last rule of $\pi$ is **EG**-right, then the proved sequent has the form $\vdash EG_x(\phi_1)(s)$. We associate a finite sequence $|\pi|$ to the proof $\pi$ by induction in the following way.
  - If the proof $\pi$ ends with the **EG**-merge rule, then the sequence contains a single element $s$.
  - If the proof $\pi$ ends with the **EG**-right rule, with subproofs $\rho$ and $\pi_1$ of the sequents $\vdash (s/x)\phi_1$ and $\Gamma, EG_x(\phi_1)(s) \vdash EG_x(\phi_1)(s')$, respectively, then $|\pi|$ is the sequence $s|\pi_1|$.

  The sequence $|\pi| = s_0, s_1, ..., s_n$ is such that $s_0 = s$; for all $i$ between 0 and $n-1$, $s_i \longrightarrow s_{i+1}$; for all $i$ between 0 and $n$, the sequent $\vdash (s_i/x)\phi_1$ has a proof smaller than $\pi$; and $s_n$ is equal to $s_p$ for some $p$ between 0 and $n-1$. By induction hypothesis, for all $i$, we have $\models (s_i/x)\phi_1$. Using Proposition 1, there exists an infinite sequence $s'_0, s'_1, ...$ such that for all $i$, we have $s'_i \longrightarrow s'_{i+1}$, and $\models (s'_i/x)\phi_1$. Hence, $\models EG_x(\phi_1)(s)$.

– If the last rule of $\pi$ is **AR**-right, then the proved sequent has the form $\vdash AR_{x,y}(\phi_1, \phi_2)(s)$. We associate a finite tree $|\pi|$ to the proof $\pi$ by induction in the following way.
  - If the proof $\pi$ ends with the first **AR**-right rule with subproofs $\rho_1$ and $\rho_2$ of the sequents $\vdash (s/x)\phi_1$ and $\vdash (s/y)\phi_2$, respectively, or with the **AR**-merge rule, then the tree contains a single node $s$.
  - If the proof $\pi$ ends with the second **AR**-right rule, with subproofs $\rho, \pi_1$, ..., $\pi_n$ of the sequents $\vdash (s/y)\phi_2, \Gamma, AR_{x,y}(\phi_1, \phi_2)(s) \vdash AR_{x,y}(\phi_1, \phi_2)(s_1)$, ..., $\Gamma, AR_{x,y}(\phi_1, \phi_2)(s) \vdash AR_{x,y}(\phi_1, \phi_2)(s_n)$, respectively, then $|\pi|$ is the tree $s(|\pi_1|, ..., |\pi_n|)$.

  The tree $|\pi|$ has root $s$; for each internal node $s'$, the children of this node are labeled by the elements of $Next(s')$; for each node $s'$ of $|\pi|$, the sequent $\vdash (s'/y)\phi_2$ has a proof smaller than $\pi$; and for each leaf $s'$, either the sequent $\vdash (s'/x)\phi_1$ has a proof smaller than $\pi$, or $s'$ is also a a label of a node on the branch from the root of $|\pi|$ to this leaf. By induction hypothesis, for each node $s'$ of this tree $\models (s'/y)\phi_2$ and for each leaf $s'$, either $\models (s'/x)\phi_1$ or $s'$ is also a label of a node on the branch from the root of $|\pi|$ to this leaf. Using Proposition 2, there exists an infinite tree $T'$ labeled by states such that for each internal node $s$ the successors of $s$ are the elements of $Next(s)$, for each node $s'$ of $T'$, $\models (s'/y)\phi_2$, and for each leaf $s'$ of $T'$, $\models (s'/x)\phi_1$. Thus, $\models AR_{x,y}(\phi_1, \phi_2)(s)$.

– If the last rule of $\pi$ is **EU**-right, then the proved sequent has the form $\vdash EU_{x,y}(\phi_1, \phi_2)(s)$. We associate a finite sequence $|\pi|$ to the proof $\pi$ by induction in the following way.

- If the proof $\pi$ ends with the first **EU**-right rule with a subproof $\rho$ of the sequent $\vdash (s/y)\phi_2$, then the sequence contains a single element $s$.
- If the proof $\pi$ ends with the second **EU**-right rule, with subproofs $\rho$ and $\pi_1$ of the sequents $\vdash (s/x)\phi_1$ and $\vdash EU_{x,y}(\phi_1, \phi_2)(s')$, respectively, then $|\pi|$ is the sequence $s|\pi_1|$.

The sequence $|\pi| = s_0, ..., s_n$ is such that $s_0 = s$; for each $i$ between 0 and $n-1$, $s_i \longrightarrow s_{i+1}$; for each $i$ between 0 and $n-1$, the sequent $\vdash (s_i/x)\phi_1$ has a proof smaller than $\pi$; and the sequent $\vdash (s_n/y)\phi_2$ has a proof smaller than $\pi$. By induction hypothesis, for each $i$ between 0 and $n-1$, $\models (s_i/x)\phi_1$ and $\models (s_n/y)\phi_2$. Hence, $\models EU_{x,y}(\phi_1, \phi_2)(s)$.

– The last rule cannot be a merge rule.

**Proposition 6 (Completeness).** *Let $\phi$ be a closed formula. If $\models \phi$ then the sequent $\vdash \phi$ is provable.*

*Proof.* By induction over the structure of $\phi$.

– If $\phi = P(s_1, ..., s_n)$, then as $\models P(s_1, ..., s_n)$, the sequent $\vdash P(s_1, ..., s_n)$ is provable with the rule atom-right.
– If $\phi = \neg P(s_1, ..., s_n)$, then as $\models \neg P(s_1, ..., s_n)$, the sequent $\vdash \neg P(s_1, ..., s_n)$ is provable with the rule $\neg$-right.
– If $\phi = \top$, then $\vdash \top$ is provable with the rule $\top$-right.
– If $\phi = \bot$, then it is not the case that $\models \bot$.
– If $\phi = \phi_1 \wedge \phi_2$, then as $\models \phi_1 \wedge \phi_2$, $\models \phi_1$ and $\models \phi_2$. By induction hypothesis, the sequents $\vdash \phi_1$ and $\vdash \phi_2$ are provable. Thus the sequent $\vdash \phi_1 \wedge \phi_2$ is provable with the $\wedge$-right rule.
– If $\phi = \phi_1 \vee \phi_2$, as $\models \phi_1 \vee \phi_2$, $\models \phi_1$ or $\models \phi_2$. By induction hypothesis, the sequent $\vdash \phi_1$ or $\vdash \phi_2$ is provable and the sequent $\vdash \phi_1 \vee \phi_2$ is provable with the $\vee$-right rule.
– If $\phi = AX_x(\phi_1)(s)$, as $\models AX_x(\phi_1)(s)$, for each state $s'$ in $Next(s)$, we have $\models (s'/x)\phi_1$. By induction hypothesis, for each $s'$ in $Next(s)$, the sequent $\vdash (s'/x)\phi_1$ is provable. Using these proofs and the **AX**-right rule, we build a proof of the sequent $\vdash AX_x(\phi_1)(s)$.
– If $\phi = EX_x(\phi_1)(s)$, as $\models EX_x(\phi_1)(s)$, there exists a state $s'$ in $Next(s)$ such that $\models (s'/x)\phi_1$. By induction hypothesis, the sequent $\vdash (s'/x)\phi_1$ is provable. With this proof and the **EX**-right rule, we build a proof of the sequent $\vdash EX_x(\phi_1)(s)$.
– If $\phi = AF_x(\phi_1)(s)$, as $\models AF_x(\phi_1)(s)$, there exists a finite tree $T$ such that $T$ has root $s$, for each internal node $s'$, the children of this node are labeled by the elements of $Next(s')$, and for each leaf $s'$, $\models (s'/x)\phi_1$. By induction hypothesis, for every leaf $s'$, the sequent $\vdash (s'/x)\phi_1$ is provable. Then, to each subtree $T'$ of $T$, we associate a proof $|T'|$ of the sequent $\vdash AF_x(\phi_1)(s')$ where $s'$ is the root of $T'$, by induction, as follows.
  - If $T'$ contains a single node $s'$, then the proof $|T|$ is built with the first **AF**-right rule from the proof of $\vdash (a/x)\phi_1$ given by the induction hypothesis.

- If $T' = s'(T_1, ..., T_n)$, then the proof $|T|$ is built with the second **AF**-right rule from the proofs $|T_1|$, ..., $|T_n|$ of the sequents $\vdash AF_x(\phi_1)(s_1)$, ..., $\vdash AF_x(\phi_1)(s_n)$, respectively, where $s_1, ..., s_n$ are the elements of $Next(s')$.

This way, the proof $|T|$ is a proof of the sequent $\vdash AF_x(\phi_1)(s)$.

– If $\phi = EG_x(\phi_1)(s)$, as $\models EG_x(\phi_1)(s)$, there exists a path $s_0, s_1, ...$ such that $s_0 = s$ and for all $i$, $\models (s_i/x)\phi_1$. By induction hypothesis, all the sequents $\vdash (s_i/x)\phi_1$ are provable. Using Proposition 3, there exists a finite sequence $T = s_0, ..., s_n$ such that for all $i$, $s_i \longrightarrow s_{i+1}$, the sequent $\vdash (s_i/x)\phi_1$ is provable and $s_n$ is some $s_p$ for $p < n$. We associate a proof $|s_i, ..., s_n|$ of the sequent $EG_x(\phi_1)(s_0), ..., EG_x(\phi_1)(s_{i-1}) \vdash EG_x(\phi_1)(s_i)$ to each suffix of $T$ by induction as follows.
  - The proof $|s_n|$ is built with the **EG**-merge rule.
  - If $i \leq n-1$, then the proof $|s_i, ..., s_n|$ is built with the **EG**-right rule from the proof of $\vdash (s_i/x)\phi_1$ given by the induction hypothesis and the proof $|s_{i+1}, ..., s_n|$ of the sequent $EG_x(\phi_1)(s_0), ..., EG_x(\phi_1)(s_i) \vdash EG_x(\phi_1)(s_{i+1})$.

  This way, the proof $|s_0, ..., s_n|$ is a proof of the sequent $\vdash EG_x(\phi_1)(s)$.

– If $\phi = AR_{x,y}(\phi_1, \phi_2)(s)$, as $\models AR_{x,y}(\phi_1, \phi_2)(s)$, there exists an infinite tree such that the root of this tree is $s$, for each internal node $s'$, the children of this node are labeled by the elements of $Next(s')$, for each node $s'$, $\models (s'/y)\phi_2$ and for each leaf $s'$, $\models (s'/x)\phi_1$. By induction hypothesis, for each node $s'$ of the tree, the sequent $\vdash (s'/y)\phi_2$ is provable and for each leaf $s'$ of the tree, the sequent $\vdash (s'/x)\phi_1$ is provable. Using Proposition 4, there exists a finite tree $T$ such that for each internal node $s'$ the successors of $s'$ are the elements of $Next(s')$, for each node $s'$, the sequent $\vdash (s'/y)\phi_2$ is provable, and for each leaf $s'$, either the sequent $\vdash (s'/x)\phi_1$ is provable or $s'$ is also a label of a node on the branch from the root of $T$ to this leaf. Then, to each subtree $T'$ of $T$, we associate a proof $|T'|$ of the sequent $AR_{x,y}(\phi_1, \phi_2)(s_1), ..., AR_{x,y}(\phi_1, \phi_2)(s_m) \vdash AR_{x,y}(\phi_1, \phi_2)(s')$ where $s'$ is the root of $T'$ and $s_1, ..., s_m$ is the sequence of nodes in $T$ from the root of $T$ to the root of $T'$.
  - If $T'$ contains a single node $s'$, and the sequent $\vdash (s'/x)\phi_1$ is provable then the proof $|T'|$ is built with the first **AR**-right rule from the proofs of $\vdash (s'/x)\phi_1$ and $\vdash (s'/y)\phi_2$ given by the induction hypothesis.
  - If $T'$ contains a single node $s'$, and $s'$ is among $s_1, ..., s_m$, then the proof $|T'|$ is built with the **AR**-merge rule.
  - If $T' = s'(T_1, ..., T_n)$, then the proof $|T'|$ is built with the second **AR**-right rule from the proofs $\vdash (s'/y)\phi_2$ given by the induction hypothesis and the proofs $|T_1|$, ..., $|T_n|$ of the sequents

    $AR_{x,y}(\phi_1, \phi_2)(s_1), ..., AR_{x,y}(\phi_1, \phi_2)(s_m), AR_{x,y}(\phi_1, \phi_2)(s') \vdash AR_{x,y}(\phi_1, \phi_2)(s'_1)$

    ...

    $AR_{x,y}(\phi_1, \phi_2)(s_1), ..., AR_{x,y}(\phi_1, \phi_2)(s_m), AR_{x,y}(\phi_1, \phi_2)(s') \vdash AR_{x,y}(\phi_1, \phi_2)(s'_n)$

    respectively, where $s'_1, ..., s'_n$ are the elements of $Next(s')$.

  This way, the proof $|T|$ is a proof of the sequent $\vdash AR_{x,y}(\phi_1, \phi_2)(s)$.

– If $\phi = EU_{x,y}(\phi_1, \phi_2)(s)$, as $\models EU_{x,y}(\phi_1, \phi_2)(s)$, there exists a finite sequence $T = s_0, ..., s_n$ such that $\models (s_n/y)\phi_2$ and for all $i$ between 0 and $n-1$, $\models (s_i/x)\phi_1$. By induction hypothesis, the sequent $\vdash (s_n/y)\phi_2$ is provable and for all $i$ between 0 and $n-1$, the sequent $\vdash (s_i/x)\phi_1$ is provable.

  We associate a proof $|s_i, ..., s_n|$ of the sequent $\vdash EU_{x,y}(\phi_1, \phi_2)(s_i)$ to each suffix of $T$ by induction as follows.

  - The proof $|s_n|$ is built with the first **EG**-right rule from the proof of $\vdash (s_n/y)\phi_2$ given by the induction hypothesis.
  - If $i \leq n-1$, then the proof $|s_i, ..., s_n|$ is built with the second **EG**-right rule from the proof of $\vdash (s_i/x)\phi_1$ given by the induction hypothesis and the proof $|s_{i+1}, ..., s_n|$ of the sequent $\vdash EU_{x,y}(\phi_1, \phi_2)(s_{i+1})$.

  This way, the proof $|s_0, ..., s_n|$ is a proof of the sequent $\vdash EU_{x,y}(\phi_1, \phi_2)(s)$.

## 5 Decidability

As they have a proof system, the logics CTL and SCTL are semi-decidable. As they are moreover complete, they are also is decidable and a simple enumeration always finds a proof of $\phi$ or of $\phi^{\perp}$.

But a more efficient algorithm is to search for a proof in the sequent calculus depicted in Figure 1: the premises of a rule always have a number of symbols that is smaller than or equal to the size of the conclusion, thus the search space for a formula is finite and a bottom-up search, avoiding redundancies, always terminate.

## 6 Conclusion and Future Work

We have shown that a natural notion of proof can be defined for CTL relative to a given finite model. This logic is sound, complete, and decidable. Searching for a proof bottom-up mimics an explicit model-checking algorithm, although some choices can be delayed by the introduction of a variable at existential nodes. Further investigations include the use of such a logic in automated theorem proving.

Also this work in only a first step in the direction of bridging the gap between proof-checking and model-checking. More proof-systems are needed for instance for infinite models and to analyze symbolic model-checking.

## Acknowledgment

This work is supported by the ANR-NSFC project LOCALI (NSFC 61161130530 and ANR 11 IS02 002 01)

## References


1. G. Dowek and Y. Jiang, Axiomatizing truth in a finite model, manuscript, 2013.
2. E.A. Emerson and J.Y. Halpern. Decision Procedures and Expressiveness in the Temporal Logic of Branching Time. Journal of Computer and System Sciences, 30(1), 1–24, 1985.
3. M.Fisher, C.Dixon, and M.Peim. Clausal temporal resolution. ACM Transactions on Computational Logic 2(1): 12-56, 2001.
4. O. Friedmann. A Proof System for CTL*mu. Diploma Thesis, University of Munich, 2008.
5. D.M. Gabbay and A. Pnueli. A Sound and Complete Deductive System for CTL* Verification. *Logic Journal of the IGPL* 16(6): 499–536, 2008.
6. M. Reynolds. An axiomatization of full Computational Tree Logic. *The Journal of Symbolic Logic* 66(3): 1011–1057, 2001.