CTL Model Checking in Deduction Modulo

Kailiang Ji(✉)

INRIA and Paris Diderot, 23 Avenue d'Italie, CS 81321,
75214 Paris Cedex 13, France
kailiang.ji@inria.fr

Abstract. In this paper we give an overview of proof-search method for CTL model checking based on Deduction Modulo. Deduction Modulo is a reformulation of Predicate Logic where some axioms—possibly all—are replaced by rewrite rules. The focus of this paper is to give an encoding of temporal properties expressed in CTL, by translating the logical equivalence between temporal operators into rewrite rules. This way, the proof-search algorithms designed for Deduction Modulo, such as Resolution Modulo or Tableaux Modulo, can be used in verifying temporal properties of finite transition systems. An experimental evaluation using Resolution Modulo is presented.

Keywords: Model checking · Deduction modulo · Resolution modulo

1 Introduction

In this paper, we express Computation Tree Logic (CTL) [4] for a given finite transition system in Deduction Modulo [6,7]. This way, the proof-search algorithms designed for Deduction Modulo, such as Resolution Modulo [2] or Tableaux Modulo [5], can be used to build proofs in CTL. Deduction Modulo is a reformulation of Predicate Logic where some axioms—possibly all—are replaced by rewrite rules. For example, the axiom $P \Leftrightarrow (Q \vee R)$ can be replaced by the rewrite rule $P \hookrightarrow (Q \vee R)$, meaning that during the proof, P can be replaced by $Q \vee R$ at any time.

The idea of translating CTL to another framework, for instance (quantified) boolean formulae [1,14,16], higher-order logic [12], etc., is not new. But using rewrite rules permits to avoid the explosion of the size of formulae during translation, because rewrite rules can be used on demand to unfold defined symbols. So, one of the advantages of this method is that it can express complicated verification problems succinctly. Gilles Dowek and Ying Jiang had given a way to build an axiomatic theory for a given finite model [9]. In this theory, the formulae are provable if and only if they are valid in the model. In [8], they gave a slight extension of CTL, named SCTL, where the predicates may have arbitrary arities. And they defined a special sequent calculus to write proofs in SCTL. This

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

sequent calculus is special because it is tailored to each specific finite model M. In this way, a formula is provable in this sequent calculus if and only if it is valid in the model M. In our method, we characterize a finite model in the same way as [9], but instead of building a deduction system, the CTL formulae are taken as terms, and the logical equivalence between different CTL formulae are expressed by rewrite rules. This way, the existing automated theorem modulo provers, for instance iProver Modulo [3], can be used to do model checking directly. The experimental evaluation shows that the resolution based proof-search algorithms is feasible, and sometimes performs better than the existing solving techniques.

The rest of this paper is organized as follows. In Sect. 2 a new variant of Deduction Modulo for one-sided sequents is presented. In Sect. 3, the usual semantics of CTL is presented. Sections 4 and 5 present the new results of this paper: in Sect. 4, an alternative semantics for CTL on finite structures is given; in Sect. 5, the rewrite rules for each CTL operator are given and the soundness and completeness of this presentation of CTL is proved, using the semantics presented in the previous section. Finally in Sect. 6, experimental evaluation for the feasibility of rewrite rules using resolution modulo is presented.

2 Deduction Modulo

One-Sided Sequents. In this work, instead of using usual sequents of the form $A_1, \ldots, A_n \vdash B_1, \ldots, B_p$, we use one-sided sequents [13], where all the propositions are put on the right hand side of the sequent sign $\vdash$ and the sequent above is transformed into $\vdash \neg A_1, \ldots, \neg A_n, B_1, \ldots, B_p$. Moreover, implication is defined from disjunction and negation ($A \Rightarrow B$ is just an abbreviation for $\neg A \vee B$), and negation is pushed inside the propositions using De Morgan's laws. For each atomic proposition P we also have a dual atomic proposition $P^\perp$ corresponding to its negation, and the operator $\perp$ extends to all the propositions. So that the axiom rule can be formulated as

$$\frac{}{\vdash P, Q}\ \text{axiom, if } P \text{ and } Q \text{ are dual atomic propositions}$$

Deduction Modulo. A *rewrite system* is a set $\mathcal{R}$ of term and proposition rewrite rules. In this paper, only proposition rewrite rules are considered. A proposition rewrite rule is a pair of propositions $l \hookrightarrow r$, in which l is an atomic proposition and r an arbitrary proposition. For instance, $P \hookrightarrow Q \vee R$. Such a system defines a congruence $\hookrightarrow$ and the relation $\overset{*}{\hookrightarrow}$ is defined, as usual, as the reflexive-transitive closure of $\hookrightarrow$. Deduction Modulo [7] is an extension of first-order logic where axioms are replaced by rewrite rules and in a proof, a proposition can be reduced at any time. This possibility is taken into account in the formulation of *Sequent Calculus Modulo* in Fig. 1. For example, with the axiom $(Q \Rightarrow R) \Rightarrow P$ we can prove the sequent $R \vdash P$. This axiom is replaced by the rules $P \hookrightarrow Q^\perp$ and $P \hookrightarrow R$ and the sequent $R \vdash P$ is expressed as the one-sided sequent $\vdash R^\perp, P$. This sequent has the proof

$$\frac{}{\vdash R^\perp, P}\ \text{axiom}$$

as $P \overset{*}{\hookrightarrow} R$.

$$\begin{array}{ll}
\dfrac{}{\vdash_{\mathcal{R}} A, B}\ \text{axiom } A \overset{*}{\hookrightarrow} P, B \overset{*}{\hookrightarrow} P^{\perp} & \dfrac{\vdash_{\mathcal{R}} A, \Delta \qquad \vdash_{\mathcal{R}} B, \Delta}{\vdash_{\mathcal{R}} \Delta}\ \text{cut } A \overset{*}{\hookrightarrow} C, B \overset{*}{\hookrightarrow} C^{\perp} \\
\dfrac{\vdash_{\mathcal{R}} \Delta}{\vdash_{\mathcal{R}} A, \Delta}\ \text{weak} & \dfrac{\vdash_{\mathcal{R}} B, C, \Delta}{\vdash_{\mathcal{R}} A, \Delta}\ \text{contr } A \overset{*}{\hookrightarrow} B, A \overset{*}{\hookrightarrow} C \\
\dfrac{}{\vdash_{\mathcal{R}} A, \Delta}\ \top\ A \overset{*}{\hookrightarrow} \top & \dfrac{\vdash_{\mathcal{R}} B, \Delta \qquad \vdash_{\mathcal{R}} C, \Delta}{\vdash_{\mathcal{R}} A, \Delta}\ \wedge \text{ if } A \overset{*}{\hookrightarrow} B \wedge C \\
\dfrac{\vdash_{\mathcal{R}} B, \Delta}{\vdash_{\mathcal{R}} A, \Delta}\ \vee_1\ A \overset{*}{\hookrightarrow} B \vee C & \dfrac{\vdash_{\mathcal{R}} C, \Delta}{\vdash_{\mathcal{R}} A, \Delta}\ \vee_2\ A \overset{*}{\hookrightarrow} B \vee C \\
\dfrac{\vdash_{\mathcal{R}} C, \Delta}{\vdash_{\mathcal{R}} A, \Delta}\ \exists\ A \overset{*}{\hookrightarrow} \exists x B, (t/x)B \overset{*}{\hookrightarrow} C & \dfrac{\vdash_{\mathcal{R}} B, \Delta}{\vdash_{\mathcal{R}} A, \Delta}\ \forall\ A \overset{*}{\hookrightarrow} \forall x B, x \notin FV(\Delta)
\end{array}$$

Fig. 1. One-sided Sequent Calculus Modulo

Note that as our system is negation free, all occurrences of atomic propositions are positive. Thus, the rule $P \hookrightarrow A$ does not correspond to an equivalence $P \Leftrightarrow A$ but to an implication $A \Rightarrow P$. In other words, our one-sided presentation of Deduction Modulo is closer to Polarized Deduction Modulo [6] with positive rules only, than to the usual Deduction Modulo. The sequent $\vdash_{\mathcal{R}} \Delta$ has a cut-free proof is represented as $\vdash^{cf}_{\mathcal{R}} \Delta$ has a proof.

3 Computation Tree Logic

Properties of a transition system can be specified by temporal logic propositions. Computation tree logic is a propositional branching-time temporal logic introduced by Emerson and Clarke [4] for finite state systems. Let AP be a set of atomic propositions and p ranges over AP. The set of CTL propositions Φ over AP is defined as follows:

$$\begin{aligned}\Phi ::= p \mid \neg\Phi \mid \Phi \wedge \Phi \mid \Phi \vee \Phi \mid AX\Phi \mid EX\Phi \mid AF\Phi \mid EF\Phi \mid AG\Phi \mid EG\Phi \\ \mid A[\Phi U \Phi] \mid E[\Phi U \Phi] \mid A[\Phi R \Phi] \mid E[\Phi R \Phi]\end{aligned}$$

The semantics of CTL can be given using Kripke structure, which is used in model checking to represent the behavior of a system.

Definition 1 (Kripke Structure). *Let AP be a set of atomic propositions. A Kripke structure M over AP is a three tuple $M = (S, \mathsf{next}, \mathsf{L})$ where*

- *S is a finite (non-empty) set of states.*
- *$\mathsf{next} : S \to \mathcal{P}^{+}(S)$ is a function that gives each state a (non-empty) set of successors.*
- *$\mathsf{L} : S \to \mathcal{P}(AP)$ is a function that labels each state with a subset of AP.*

An infinite path is an infinite sequence of states $\pi = \pi_0\pi_1\cdots$ s.t. $\forall i \geq 0$, $\pi_{i+1} \in \mathsf{next}(\pi_i)$. Note that the sequence $\pi_i\pi_{i+1}\cdots\pi_j$ is denoted as π_i^j and the path π with $\pi_0 = s$ is denoted as $\pi(s)$.

Definition 2 (Semantics of CTL). *Let p be an atomic proposition. Let φ, φ_1, φ_2 be CTL propositions. The relation $M, s \models \varphi$ is defined as follows.*

$M, s \models p$	$\Leftrightarrow p \in \mathsf{L}(s)$
$M, s \models \neg\varphi_1$	$\Leftrightarrow M, s \not\models \varphi_1$
$M, s \models \varphi_1 \wedge \varphi_2$	$\Leftrightarrow M, s \models \varphi_1$ *and* $M, s \models \varphi_2$
$M, s \models \varphi_1 \vee \varphi_2$	$\Leftrightarrow M, s \models \varphi_1$ *or* $M, s \models \varphi_2$
$M, s \models AX\varphi_1$	$\Leftrightarrow \forall s' \in \mathsf{next}(s),\ M, s' \models \varphi_1$
$M, s \models EX\varphi_1$	$\Leftrightarrow \exists s' \in \mathsf{next}(s),\ M, s' \models \varphi_1$
$M, s \models AG\varphi_1$	$\Leftrightarrow \forall \pi(s),\ \forall i \geq 0,\ M, \pi_i \models \varphi_1$
$M, s \models EG\varphi_1$	$\Leftrightarrow \exists \pi(s)$ *s.t.* $\forall i \geq 0,\ M, \pi_i \models \varphi_1$
$M, s \models AF\varphi_1$	$\Leftrightarrow \forall \pi(s),\ \exists i \geq 0$ *s.t.* $M, \pi_i \models \varphi_1$
$M, s \models EF\varphi_1$	$\Leftrightarrow \exists \pi(s),\ \exists i \geq 0$ *s.t.* $M, \pi_i \models \varphi_1$
$M, s \models A[\varphi_1 U \varphi_2]$	$\Leftrightarrow \forall \pi(s),\ \exists j \geq 0$ *s.t.* $M, \pi_j \models \varphi_2$ *and* $\forall 0 \leq i < j,\ M, \pi_i \models \varphi_1$
$M, s \models E[\varphi_1 U \varphi_2]$	$\Leftrightarrow \exists \pi(s),\ \exists j \geq 0$ *s.t.* $M, \pi_j \models \varphi_2$ *and* $\forall 0 \leq i < j,\ M, \pi_i \models \varphi_1$
$M, s \models A[\varphi_1 R \varphi_2]$	$\Leftrightarrow \forall \pi(s),\ \forall j \geq 0,$ *either* $M, \pi_j \models \varphi_2$ *or* $\exists 0 \leq i < j$ *s.t.* $M, \pi_i \models \varphi_1$
$M, s \models E[\varphi_1 R \varphi_2]$	$\Leftrightarrow \exists \pi(s),\ \forall j \geq 0,$ *either* $M, \pi_j \models \varphi_2$ *or* $\exists 0 \leq i < j$ *s.t.* $M, \pi_i \models \varphi_1$

4 Alternative Semantics of CTL

In this part we present an alternative semantics of CTL using finite paths only.

Paths with the Last State Repeated (lr-Paths). A finite path is a *lr-path* if and only if the last state on the path occurs twice. For instance s_0, s_1, s_0 is a lr-path. Note that we use $\rho = \rho_0\rho_1 \ldots \rho_j$ to denote a lr-path. A lr-path ρ with $\rho_0 = s$ is denoted as $\rho(s)$, with $\rho_i = \rho_j$ is denoted as $\rho(i, j)$. The length of a path l is expressed by $\mathsf{len}(l)$ and the concatenation of two paths l_1, l_2 is $l_1 \hat{}\, l_2$.

Lemma 1. *Let M be a Kripke structure.*

1. *If π is an infinite path of M, then $\exists i \geq 0$ such that π_0^i is a lr-path.*
2. *If $\rho(i, j)$ is a lr-path of M, then $\rho_0^i \hat{}(\rho_{i+1}^j)^\omega$ is an infinite path.*

Proof. For Case 1, as M is finite, there exists at least one repeating state in π. If π_i is the first state which occurs twice, then π_0^i is a lr-path. Case 2 is trivial. □

Lemma 2. *Let M be a Kripke structure.*

1. *For the path $l = s_0, s_1, \ldots, s_k$, there exists a finite path $l' = s'_0, s'_1, \ldots, s'_i$ without repeating states s.t. $s'_0 = s_0$, $s'_i = s_k$, and $\forall 0 < j < i$, s'_j is on l.*
2. *If there is a path from s to s', then there exists a lr-path $\rho(s)$ s.t. s' is on ρ.*

Proof. For the first case, l' can be built by deleting the cycles from l. The second case is straightforward by the first case and Lemma 1. □

Definition 3 (Alternative Semantics of CTL). *Let p be an atomic proposition and $\varphi, \varphi_1, \varphi_2$ be CTL propositions. The relation $M, s \models_a \varphi$ is defined as follows.*

$M, s \models_a p$	$\Leftrightarrow$	$p \in L(s)$
$M, s \models_a \neg\varphi_1$	$\Leftrightarrow$	$M, s \not\models_a \varphi_1$
$M, s \models_a \varphi_1 \wedge \varphi_2$	$\Leftrightarrow$	$M, s \models_a \varphi_1$ *and* $M, s \models_a \varphi_2$
$M, s \models_a \varphi_1 \vee \varphi_2$	$\Leftrightarrow$	$M, s \models_a \varphi_1$ *or* $M, s \models_a \varphi_2$
$M, s \models_a AX\varphi_1$	$\Leftrightarrow$	$\forall s' \in \mathsf{next}(s),\ M, s' \models_a \varphi_1$
$M, s \models_a EX\varphi_1$	$\Leftrightarrow$	$\exists s' \in \mathsf{next}(s),\ M, s' \models_a \varphi_1$
$M, s \models_a AF\varphi_1$	$\Leftrightarrow$	$\forall \rho(s),\ \exists i < \mathsf{len}(\rho) - 1$ *s.t.* $M, \rho_i \models_a \varphi_1$
$M, s \models_a EF\varphi_1$	$\Leftrightarrow$	$\exists \rho(s),\ \exists i < \mathsf{len}(\rho) - 1$ *s.t.* $M, \rho_i \models_a \varphi_1$
$M, s \models_a AG\varphi_1$	$\Leftrightarrow$	$\forall \rho(s),\ \forall i < \mathsf{len}(\rho) - 1,\ M, \rho_i \models_a \varphi_1$
$M, s \models_a EG\varphi_1$	$\Leftrightarrow$	$\exists \rho(s),\ \forall i < \mathsf{len}(\rho) - 1,\ M, \rho_i \models_a \varphi_1$
$M, s \models_a A[\varphi_1 U \varphi_2]$	$\Leftrightarrow$	$\forall \rho(s),\ \exists i < \mathsf{len}(\rho) - 1$ *s.t.* $M, \rho_i \models_a \varphi_2$ *and* $\forall j < i,\ M, \rho_j \models_a \varphi_1$
$M, s \models_a E[\varphi_1 U \varphi_2]$	$\Leftrightarrow$	$\exists \rho(s),\ \exists i < \mathsf{len}(\rho) - 1$ *s.t.* $M, \rho_i \models_a \varphi_2$ *and* $\forall j < i,\ M, \rho_j \models_a \varphi_1$
$M, s \models_a A[\varphi_1 R \varphi_2]$	$\Leftrightarrow$	$\forall \rho(s),\ \forall i < \mathsf{len}(\rho) - 1,\ M, \rho_i \models_a \varphi_2$ *or* $\exists j < i$ *s.t.* $M, \rho_j \models_a \varphi_1$
$M, s \models_a E[\varphi_1 R \varphi_2]$	$\Leftrightarrow$	$\exists \rho(s),\ \forall i < \mathsf{len}(\rho) - 1,\ M, \rho_i \models_a \varphi_2$ *or* $\exists j < i$ *s.t.* $M, \rho_j \models_a \varphi_1$

We now prove the equivalence of the two semantics, that is, $M, s \models \varphi$ iff $M, s \models_a \varphi$. To simplify the proofs, we use a normal form of the CTL propositions, in which all the negations appear only in front of the atomic propositions.

Negation Normal Form. A CTL proposition is in negation normal form (NNF), if the negation $\neg$ is applied only to atomic propositions. Every CTL proposition can be transformed into an equivalent proposition of NNF using the following equivalences.

$\neg\neg\varphi \equiv \varphi$		
$\neg(\varphi \vee \psi) \equiv \neg\varphi \wedge \neg\psi$	$\neg AF\varphi \equiv EG\neg\varphi$	$\neg A[\varphi U \psi] \equiv E[\neg\varphi R \neg\psi]$
$\neg(\varphi \wedge \psi) \equiv \neg\varphi \vee \neg\psi$	$\neg EF\varphi \equiv AG\neg\varphi$	$\neg E[\varphi U \psi] \equiv A[\neg\varphi R \neg\psi]$
$\neg AX\varphi \equiv EX\neg\varphi$	$\neg AG\varphi \equiv EF\neg\varphi$	$\neg A[\varphi R \psi] \equiv E[\neg\varphi U \neg\psi]$
$\neg EX\varphi \equiv AX\neg\varphi$	$\neg EG\varphi \equiv AF\neg\varphi$	$\neg E[\varphi R \psi] \equiv A[\neg\varphi U \neg\psi]$

Lemma 3. *Let φ be a CTL proposition of NNF. If $M, s \models \varphi$, then $M, s \models_a \varphi$.*

Proof. By induction on the structure of φ. For brevity, we just prove the two cases where φ is $AF\varphi_1$ and $AG\varphi_1$. The full proof is in [10].

- Let $\varphi = AF\varphi_1$. We prove the contraposition. If there is a lr-path $\rho(s)(j, k)$ s.t. $\forall 0 \leq i < k,\ M, \rho_i \not\models \varphi_1$, then by Lemma 1, there exists an infinite path $\rho_0^j\hat{}(\rho_{j+1}^k)^\omega$, which is a counterexample of $M, s \models AF\varphi_1$. Thus for each lr-path $\rho(s)$, $\exists 0 \leq i < \mathsf{len}(\rho) - 1$ s.t. $M, \rho_i \models \varphi_1$ holds. Then by induction hypothesis (IH), for each lr-path $\rho(s)$, $\exists 0 \leq i < \mathsf{len}(\rho) - 1$ s.t. $M, \rho_i \models_a \varphi_1$ holds, and thus $M, s \models_a AF\varphi_1$ holds.
- Let $\varphi = AG\varphi_1$. We prove the contraposition. If there is a lr-path $\rho(s)(j, k)$ and $\exists 0 \leq i < k$ s.t. $M, \rho_i \not\models \varphi_1$, then by Lemma 1, there exists an infinite path $\rho_0^j\hat{}(\rho_{j+1}^k)^\omega$, which is a counterexample of $M, s \models AG\varphi_1$. Thus for each lr-path $\rho(s)(j, k)$ and $\forall 0 \leq i < k,\ M, \rho_i \models \varphi_1$ holds. Then by IH, for each lr-path $\rho(s)(j, k)$ and $\forall 0 \leq i < k,\ M, \rho_i \models_a \varphi_1$ holds, and thus $M, s \models_a AG\varphi_1$ holds.

$\square$

Lemma 4. *Let φ be a CTL proposition of NNF. If $M, s \models_a \varphi$, then $M, s \models \varphi$.*

Proof. By induction on the structure of φ. For brevity, we just prove the two cases where φ is $AF\varphi_1$ and $AG\varphi_1$. The full proof is in [10].

- Let $\varphi = AF\varphi_1$. If there is an infinite path $\pi(s)$ s.t. $\forall j \geq 0$, $M, \pi_j \not\models_a \varphi_1$, then by Lemma 1, there exists $k \geq 0$ s.t. π_0^k is a lr-path, which is a counterexample of $M, s \models_a AF\varphi_1$. Thus for each infinite path $\pi(s)$, $\exists j \geq 0$ s.t. $M, \pi_j \models_a \varphi_1$ holds. Then by IH, for each infinite path $\pi(s)$, $\exists j \geq 0$ s.t. $M, \pi_j \models \varphi_1$ holds and thus $M, s \models AF\varphi_1$ holds.
- Let $\varphi = AG\varphi_1$. Assume that there exists an infinite path $\pi(s)$ and $\exists i \geq 0$, $M, \pi_i \not\models_a \varphi_1$. By Lemma 2, there exists a lr-path $\rho(s)$ s.t. π_i is on ρ, which is a counterexample of $M, s \models_a AG\varphi_1$. Thus for each infinite path $\pi(s)$ and $\forall i \geq 0$, $M, \pi_i \models_a \varphi_1$ holds. Then by IH, for each infinite path $\pi(s)$ and $\forall i \geq 0$, $M, \pi_i \models \varphi_1$ holds and thus $M, s \models AG\varphi_1$ holds. □

Theorem 1. *Let φ be a CTL proposition. $M, s \models \varphi$ iff $M, s \models_a \varphi$.*

5 Rewrite Rules for CTL

The work in this section is to express CTL propositions in Deduction Modulo and prove that for a CTL proposition φ, the translation of $M, s \models_a \varphi$ is provable if and only if $M, s \models_a \varphi$ holds. So we fix such a model $M = (S, \mathsf{next}, \mathsf{L})$. As in [9], we consider a two sorted language $\mathcal{L}$, which contains

- constants $s_1, \ldots, s_n$ for each state of M.
- predicate symbols $\varepsilon_0, \varepsilon_{\sqcap_0}, \varepsilon_{\sqcup_0}, \varepsilon_1, \varepsilon_{\sqcap_1}, \varepsilon_{\sqcup_1}$, in which the binary predicates ε_0, $\varepsilon_{\sqcap_0}$ and $\varepsilon_{\sqcup_0}$ apply to all the CTL propositions, while the ternary predicates ε_1, $\varepsilon_{\sqcap_1}$ and $\varepsilon_{\sqcup_1}$ only apply to the CTL propositions starting with the temporal connectives AG, EG, AR and ER.
- binary predicate symbols *mem* for the membership, r for the next-notation.
- a constant nil and a binary function symbol con.

We use x, y, z to denote the variables of the state terms, X, Y, Z to denote the class variables. A class is in fact a set of states, here we use the *class theory*, rather than the *(monadic) second order logic*, is to emphasis that this formalism is a theory and not a logic.

CTL Term. To express CTL in Deduction Modulo, firstly, we translate the CTL proposition φ into a term $|\varphi|$ (*CTL term*). The translation rules are as follows:

$\lvert p\rvert = \overline{p}$, $p \in AP$	$\lvert EX\varphi\rvert = \mathsf{ex}(\lvert\varphi\rvert)$	$\lvert A[\varphi U\psi]\rvert = \mathsf{au}(\lvert\varphi\rvert, \lvert\psi\rvert)$
$\lvert\neg\varphi\rvert = \mathsf{not}(\lvert\varphi\rvert)$	$\lvert AF\varphi\rvert = \mathsf{af}(\lvert\varphi\rvert)$	$\lvert E[\varphi U\psi]\rvert = \mathsf{eu}(\lvert\varphi\rvert, \lvert\psi\rvert)$
$\lvert\varphi \wedge \psi\rvert = \mathsf{and}(\lvert\varphi\rvert, \lvert\psi\rvert)$	$\lvert EF\varphi\rvert = \mathsf{ef}(\lvert\varphi\rvert)$	$\lvert A[\varphi R\psi]\rvert = \mathsf{ar}(\lvert\varphi\rvert, \lvert\psi\rvert)$
$\lvert\varphi \vee \psi\rvert = \mathsf{or}(\lvert\varphi\rvert, \lvert\psi\rvert)$	$\lvert AG\varphi\rvert = \mathsf{ag}(\lvert\varphi\rvert)$	$\lvert E[\varphi R\psi]\rvert = \mathsf{er}(\lvert\varphi\rvert, \lvert\psi\rvert)$
$\lvert AX\varphi\rvert = \mathsf{ax}(\lvert\varphi\rvert)$	$\lvert EG\varphi\rvert = \mathsf{eg}(\lvert\varphi\rvert)$	

Note that we use Φ, Ψ to denote the variables of the CTL terms. Both finite sets and finite paths are represented with the symbols con and nil. For the set $S' = \{s_i, \ldots, s_j\}$, we use $[S']$ to denote its term form $\mathsf{con}(s_i, \mathsf{con}(\ldots, \mathsf{con}(s_j, \mathsf{nil})\ldots))$. For the path $s_i^j = s_i, \ldots, s_j$, its term form $\mathsf{con}(s_j, \mathsf{con}(\ldots, \mathsf{con}(s_i, \mathsf{nil})\ldots))$ is denoted by $[s_i^j]$.

Definition 4 (Semantics). *Semantics of the propositions in $\mathcal{L}$ is as follows.*

$M \models \varepsilon_0(\vert\varphi\vert, s)$	$\Leftrightarrow M, s \models_a \varphi$
$M \models r(s, [S'])$	$\Leftrightarrow S' = \mathsf{next}(s)$
$M \models mem(s, [s_0^i])$	$\Leftrightarrow$ *s is on the path s_0^i*
$M \models \varepsilon_{\sqcap_0}(\vert\varphi\vert, [S'])$	$\Leftrightarrow \forall s \in S'$, $M \models \varepsilon_0(\vert\varphi\vert, s)$
$M \models \varepsilon_{\sqcup_0}(\vert\varphi\vert, [S'])$	$\Leftrightarrow \exists s \in S'$ *s.t.* $M \models \varepsilon_0(\vert\varphi\vert, s)$
$M \models \varepsilon_1(\mathsf{ag}(\vert\varphi_1\vert), s, [s_0^i])$	$\Leftrightarrow \forall$*lr-path* $s_0^i \,\hat{}\, s_{i+1}^k (s_{i+1} = s)$, $\forall i < j < k$, $M \models \varepsilon_0(\vert\varphi_1\vert, s_j)$
$M \models \varepsilon_1(\mathsf{eg}(\vert\varphi_1\vert), s, [s_0^i])$	$\Leftrightarrow \exists$*lr-path* $s_0^i \,\hat{}\, s_{i+1}^k (s_{i+1} = s)$, $\forall i < j < k$, $M \models \varepsilon_0(\vert\varphi_1\vert, s_j)$
$M \models \varepsilon_1(\mathsf{ar}(\vert\varphi_1\vert, \vert\varphi_2\vert), s, [s_0^i])$	$\Leftrightarrow \forall$*lr-path* $s_0^i \,\hat{}\, s_{i+1}^k (s_{i+1} = s)$, $\forall i < j < k$, *Either* $M \models \varepsilon_0(\vert\varphi_2\vert, s_j)$ *or* $\exists i < m < j$ *s.t.* $M \models \varepsilon_0(\vert\varphi_1\vert, s_m)$
$M \models \varepsilon_1(\mathsf{er}(\vert\varphi_1\vert, \vert\varphi_2\vert), s, [s_0^i])$	$\Leftrightarrow \exists$*lr-path* $s_0^i \,\hat{}\, s_{i+1}^k (s_{i+1} = s)$, $\forall i < j < k$, *either* $M \models \varepsilon_0(\vert\varphi_2\vert, s_j)$ *or* $\exists i < m < j$ *s.t.* $M \models \varepsilon_0(\vert\varphi_1\vert, s_m)$
$M \models \varepsilon_{\sqcap_1}(\mathsf{ag}(\vert\varphi_1\vert), [S'], [s_0^i])$	$\Leftrightarrow \forall s \in S'$, $M \models \varepsilon_1(\mathsf{ag}(\vert\varphi_1\vert), s, [s_0^i])$
$M \models \varepsilon_{\sqcup_1}(\mathsf{eg}(\vert\varphi_1\vert), [S'], [s_0^i])$	$\Leftrightarrow \exists s \in S'$ *s.t.* $M \models \varepsilon_1(\mathsf{eg}(\vert\varphi_1\vert), s, [s_0^i])$
$M \models \varepsilon_{\sqcap_1}(\mathsf{ar}(\vert\varphi_1\vert, \vert\varphi_2\vert), [S'], [s_0^i])$	$\Leftrightarrow \forall s \in S'$, $M \models \varepsilon_1(\mathsf{ar}(\vert\varphi_1\vert, \vert\varphi_2\vert), s, [s_0^i])$
$M \models \varepsilon_{\sqcup_1}(\mathsf{er}(\vert\varphi_1\vert, \vert\varphi_2\vert), [S'], [s_0^i])$	$\Leftrightarrow \exists s \in S'$ *s.t.* $M \models \varepsilon_1(\mathsf{er}(\vert\varphi_1\vert, \vert\varphi_2\vert), s, [s_0^i])$

Note that when a proposition $\varepsilon_1(|\varphi|, s, [s_i^j])$ is valid in M, for instance $M \models \varepsilon_1(\mathsf{eg}(|\varphi|), s, [s_i^j])$, $EG\varphi$ may not hold on the state s.

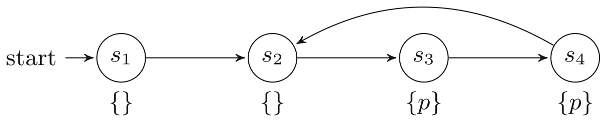

Fig. 2. Example of $\mathcal{L}$

Example 1. For the structure M in Fig. 2, $M \models \varepsilon_1(\mathsf{eg}(\overline{p}), s_3, \mathsf{con}(s_2, \mathsf{con}(s_1, \mathsf{nil})))$ holds because there exists a lr-path, for instance s_1, s_2, s_3, s_4, s_2 such that p holds on s_3 and s_4.

The Rewrite System $\mathcal{R}$. The rewrite system has three components

1. rules for the Kripke structure M (denoted as $\mathcal{R}_M$),
2. rules for the class variables (denoted as $\mathcal{R}_c$),
3. rules for the semantics encoding of the CTL operators (denoted as $\mathcal{R}_{CTL}$).

The Rules of $\mathcal{R}_M$ The rules of $\mathcal{R}_M$ are as follows:

– for each atomic proposition $p \in AP$ and each state $s \in S$, if $p \in \mathsf{L}(s)$, then $\varepsilon_0(\overline{p}, s) \hookrightarrow \top$ is in $\mathcal{R}_M$, otherwise $\varepsilon_0(\mathsf{not}(\overline{p}), s) \hookrightarrow \top$ is in of $\mathcal{R}_M$.
– for each state $s \in S$, take $r(s, [\mathsf{next}(s)]) \hookrightarrow \top$ as a rewrite rule of $\mathcal{R}_M$.

The Rules of $\mathcal{R}_c$ For the class variables, as the domain of the model is finite, there exists two axioms [9] $\forall x(x = x)$, and $\forall x \forall y \forall Z((x = y \vee mem(x, Z)) \Rightarrow mem(x, \mathsf{con}(y, Z)))$. The rewrite rules for these axioms are $x = x \hookrightarrow \top$ and $mem(x, \mathsf{con}(y, Z)) \hookrightarrow x = y \vee mem(x, Z)$. To avoid introducing the predicate "=", the rewrite rules are replaced by the rules ($\mathcal{R}_c$)

$$mem(x, \mathsf{con}(x, Z)) \hookrightarrow \top \text{ and } mem(x, \mathsf{con}(y, Z)) \hookrightarrow mem(x, Z).$$

The Rules of $\mathcal{R}_{CTL}$ The rewrite rules for the predicates carrying the semantic definition of the CTL propositions, are in Fig. 3.

$$\varepsilon_0(\mathsf{or}(\Phi, \Psi), x) \hookrightarrow \varepsilon_0(\Phi, x) \vee \varepsilon_0(\Psi, x) \qquad \varepsilon_0(\mathsf{and}(\Phi, \Psi), x) \hookrightarrow \varepsilon_0(\Phi, x) \wedge \varepsilon_0(\Psi, x)$$
$$\varepsilon_0(\mathsf{ax}(\Phi), x) \hookrightarrow \exists X(r(x, X) \wedge \varepsilon_{\sqcap_0}(\Phi, X)) \quad \varepsilon_0(\mathsf{ex}(\Phi), x) \hookrightarrow \exists X(r(x, X) \wedge \varepsilon_{\sqcup_0}(\Phi, X))$$
$$\varepsilon_0(\mathsf{af}(\Phi), x)) \hookrightarrow \varepsilon_0(\Phi, x) \vee \exists X(r(x, X) \wedge \varepsilon_{\sqcap_0}(\mathsf{af}(\Phi), X))$$
$$\varepsilon_0(\mathsf{ef}(\Phi), x)) \hookrightarrow \varepsilon_0(\Phi, x) \vee \exists X(r(x, X) \wedge \varepsilon_{\sqcup_0}(\mathsf{ef}(\Phi), X))$$
$$\varepsilon_0(\mathsf{ag}(\Phi), x) \hookrightarrow \varepsilon_1(\mathsf{ag}(\Phi), x, \mathsf{nil}) \qquad \varepsilon_0(\mathsf{eg}(\Phi), x) \hookrightarrow \varepsilon_1(\mathsf{eg}(\Phi), x, \mathsf{nil})$$
$$\varepsilon_0(\mathsf{au}(\Phi, \Psi), x) \hookrightarrow \varepsilon_0(\Psi, x) \vee (\varepsilon_0(\Phi, x) \wedge \exists X(r(x, X) \wedge \varepsilon_{\sqcap_0}(\mathsf{au}(\Phi, \Psi), X)))$$
$$\varepsilon_0(\mathsf{eu}(\Phi, \Psi), x) \hookrightarrow \varepsilon_0(\Psi, x) \vee (\varepsilon_0(\Phi, x) \wedge \exists X(r(x, X) \wedge \varepsilon_{\sqcup_0}(\mathsf{eu}(\Phi, \Psi), X)))$$
$$\varepsilon_0(\mathsf{ar}(\Phi, \Psi), x) \hookrightarrow \varepsilon_1(\mathsf{ar}(\Phi, \Psi), x, \mathsf{nil}) \qquad \varepsilon_0(\mathsf{er}(\Phi, \Psi), x) \hookrightarrow \varepsilon_1(\mathsf{er}(\Phi, \Psi), x, \mathsf{nil})$$
$$\varepsilon_{\sqcap_0}(\Phi, \mathsf{con}(x, X)) \hookrightarrow \varepsilon_0(\Phi, x) \wedge \varepsilon_{\sqcap_0}(\Phi, X) \ \varepsilon_{\sqcap_0}(\Phi, \mathsf{nil}) \hookrightarrow \top$$
$$\varepsilon_{\sqcup_0}(\Phi, \mathsf{con}(x, X)) \hookrightarrow \varepsilon_0(\Phi, x) \vee \varepsilon_{\sqcup_0}(\Phi, X)$$
$$\varepsilon_1(\mathsf{ag}(\Phi), x, Y) \hookrightarrow mem(x, Y) \vee (\varepsilon_0(\Phi, x) \wedge \exists X(r(x, X) \wedge \varepsilon_{\sqcap_1}(\mathsf{ag}(\Phi), X, \mathsf{con}(x, Y))))$$
$$\varepsilon_1(\mathsf{eg}(\Phi), x, Y) \hookrightarrow mem(x, Y) \vee (\varepsilon_0(\Phi, x) \wedge \exists X(r(x, X) \wedge \varepsilon_{\sqcup_1}(\mathsf{eg}(\Phi), X, \mathsf{con}(x, Y))))$$
$$\varepsilon_1(\mathsf{ar}(\Phi, \Psi), x, Y) \hookrightarrow$$
$$mem(x, Y) \vee (\varepsilon_0(\Psi, x) \wedge (\varepsilon_0(\Phi, x) \vee \exists X(r(x, X) \wedge \varepsilon_{\sqcap_1}(\mathsf{ar}(\Phi, \Psi), X, \mathsf{con}(x, Y)))))$$
$$\varepsilon_1(\mathsf{er}(\Phi, \Psi), x, Y) \hookrightarrow$$
$$mem(x, Y) \vee (\varepsilon_0(\Psi, x) \wedge (\varepsilon_0(\Phi, x) \vee \exists X(r(x, X) \wedge \varepsilon_{\sqcup_1}(\mathsf{er}(\Phi, \Psi), X, \mathsf{con}(x, Y)))))$$
$$\varepsilon_{\sqcap_1}(\Phi, \mathsf{con}(x, X), Y) \hookrightarrow \varepsilon_1(\Phi, x, Y) \wedge \varepsilon_{\sqcap_1}(\Phi, X, Y)$$
$$\varepsilon_{\sqcap_1}(\Phi, \mathsf{nil}, Y) \hookrightarrow \top$$
$$\varepsilon_{\sqcup_1}(\Phi, \mathsf{con}(x, X), Y) \hookrightarrow \varepsilon_1(\Phi, x, Y) \vee \varepsilon_{\sqcup_1}(\Phi, X, Y)$$

Fig. 3. Rewrite Rules for CTL Connectives ($\mathcal{R}_{CTL}$)

Now we are ready to prove the main theorem. Our goal is to prove that $M \models \varepsilon_0(|\varphi|, s)$ holds if and only if $\varepsilon_0(|\varphi|, s)$ is provable in Deduction Modulo.

Lemma 5 (Soundness). *For a CTL formula φ of NNF, if the sequent $\vdash^{cf}_{\mathcal{R}}$ $\varepsilon_0(|\varphi|, s)$ has a proof, then $M \models \varepsilon_0(|\varphi|, s)$.*

Proof. More generally, we prove that for any CTL proposition φ of NNF,

- if $\vdash^{cf}_{\mathcal{R}} \varepsilon_0(|\varphi|, s)$ has a proof, then $M \models \varepsilon_0(|\varphi|, s)$.
- if $\vdash^{cf}_{\mathcal{R}} \varepsilon_{\sqcap_0}(|\varphi|, [S'])$ has a proof, then $M \models \varepsilon_{\sqcap_0}(|\varphi|, [S'])$.
- if $\vdash^{cf}_{\mathcal{R}} \varepsilon_{\sqcup_0}(|\varphi|, [S'])$ has a proof, then $M \models \varepsilon_{\sqcup_0}(|\varphi|, [S'])$.
- if $\vdash^{cf}_{\mathcal{R}} \varepsilon_1(|\varphi|, s, [s^j_i])$ has a proof, in which φ is either of the form $AG\varphi_1$, $EG\varphi_1$, $A[\varphi_1 R \varphi_2]$, $E[\varphi_1 R \varphi_2]$, then $M \models \varepsilon_1(|\varphi|, s, [s^j_i])$.
- if $\vdash^{cf}_{\mathcal{R}} \varepsilon_{\sqcap_1}(|\varphi|, [S'], [s^j_i])$ has a proof, in which φ is either of the form $AG\varphi_1$, $A[\varphi_1 R \varphi_2]$, then $M \models \varepsilon_{\sqcap_1}(|\varphi|, [S'], [s^j_i])$.
- if $\vdash^{cf}_{\mathcal{R}} \varepsilon_{\sqcup_1}(|\varphi|, [S'], [s^j_i])$ has a proof, in which φ is either of the form $EG\varphi_1$, $E[\varphi_1 R \varphi_2]$, then $M \models \varepsilon_{\sqcup_1}(|\varphi|, [S'], [s^j_i])$.

By induction on the size of the proof. Consider the different case for φ, we have 18 cases (2 cases for the atomic proposition and its negation, 2 cases for and and or, 10 cases for the temporal connectives ax, ex, af, ef, ag, eg, au, eu, ar, er, 4 cases for the predicate symbols $\varepsilon_{\sqcap_0}$, $\varepsilon_{\sqcup_0}$, $\varepsilon_{\sqcap_1}$, $\varepsilon_{\sqcup_0}$), but each case is easy. For brevity, we just prove two cases. The full proof is in [10].

- Suppose the sequent $\vdash^{cf}_{\mathcal{R}} \varepsilon_0(\mathsf{af}(|\varphi|), s)$ has a proof. As $\varepsilon_0(\mathsf{af}(|\varphi|), s) \hookrightarrow \varepsilon_0(|\varphi|, s) \vee \exists X(r(s, X) \wedge \varepsilon_{\sqcap_0}(\mathsf{af}(|\varphi|), X))$, the last rule in the proof is $\vee_1$ or $\vee_2$. For $\vee_1$, $M \models \varepsilon_0(|\varphi|, s)$ holds by IH, then $M \models \varepsilon_0(\mathsf{af}(|\varphi|), s)$ holds by its semantic definition. For $\vee_2$, $M \models \exists X(r(s, X) \wedge \varepsilon_{\sqcap_0}(\mathsf{af}(|\varphi|), X))$ holds by IH, thus there exists S' s.t. $M \models r(s, [S'])$ and $M \models \varepsilon_{\sqcap_0}(\mathsf{af}(|\varphi|), [S'])$ holds. Then we get $S' = \mathsf{next}(s)$ and for each state s' in S', $M \models \varepsilon_0(\mathsf{af}(|\varphi|), s')$ holds. Now assume $M \not\models \varepsilon_0(\mathsf{af}(|\varphi|), s)$, then there exists a lr-path $\rho(s)(j, k)$ s.t. $\forall 0 \le i < k$, $M \not\models \varepsilon_0(|\varphi|, \rho_i)$. For the path $\rho(s)(j, k)$,
  - if $j \neq 0$, then ρ^k_1 is a lr-path, which is a counterexample of $M \models \varepsilon_0(\mathsf{af}(|\varphi|), \rho_1)$.
  - if $j = 0$, then $\rho^k_1 \hat{\ } \rho_1$ is a lr-path, which is a counterexample of $M \models \varepsilon_0(\mathsf{af}(|\varphi|), \rho_1)$.

  Thus $M \models \varepsilon_0(\mathsf{af}(|\varphi|), s)$ holds by its semantic definition.
- Suppose that $\vdash^{cf}_{\mathcal{R}} \varepsilon_1(\mathsf{ag}(|\varphi|), s, [s^j_i])$ has a proof. As $\varepsilon_1(\mathsf{ag}(|\varphi|), s, [s^j_i]) \hookrightarrow mem(s, [s^j_i]) \vee (\varepsilon_0(|\varphi|, s) \wedge \exists X(r(s, X) \wedge \varepsilon_{\sqcap_1}(\mathsf{ag}(|\varphi|), X, \mathsf{con}(s, [s^j_i]))))$, the last rule in the proof is $\vee_1$ or $\vee_2$. For $\vee_1$, $M \models mem(s, [s^j_i])$ holds by IH, thus $s^j_i \hat{\ } s$ is a lr-path and $M \models \varepsilon_1(\mathsf{ag}(|\varphi|), s, [s^j_i])$ holds by its semantic definition. For $\vee_2$, $M \models \varepsilon_0(|\varphi|, s)$ and $M \models \exists X(r(s, X) \wedge \varepsilon_{\sqcap_1}(\mathsf{ag}(|\varphi|), X, \mathsf{con}(s, [s]^j_i)))$ holds by IH. Thus there exists S' s.t. $M \models r(s, [S']) \wedge \varepsilon_{\sqcap_1}(\mathsf{ag}(|\varphi|), [S'], \mathsf{con}(s, [s^j_i]))$ holds. Then by the semantic definition, $S' = \mathsf{next}(s)$ and for each state $s' \in S'$, $M \models \varepsilon_1(\mathsf{ag}(|\varphi|), s', \mathsf{con}(s, [s^j_i]))$ holds. Thus $M \models \varepsilon_1(\mathsf{ag}(|\varphi|), s, [s^j_i])$ holds by its semantic definition.

Lemma 6 (Completeness). *For a CTL formula φ of NNF, if $M \models \varepsilon_0(|\varphi|, s)$, then the sequent $\vdash^{cf}_{\mathcal{R}} \varepsilon_0(|\varphi|, s)$ has a proof.*

Proof. By induction on the structure of φ. For brevity, here we just prove some of the cases. The full proof is in [10].

– Suppose $M \models \varepsilon_0(\mathsf{af}(|\varphi_1|), s)$ holds. By the semantics of $\mathcal{L}$, there exists a state s' on each lr-path starting from s s.t. $M \models \varepsilon_0(|\varphi_1|, s')$ holds. Thus there exists a finite tree T s.t.
  - T has root s;
  - for each internal node s' in T, the children of s' are labelled by the elements of $\mathsf{next}(s')$;
  - for each leaf s', s' is the first node in the branch starting from s s.t. $M \models \varepsilon_0(|\varphi_1|, s')$ holds.

  By IH, for each leaf s', there exists a proof $\Pi_{(\varphi_1,s')}$ for the sequent $\vdash^{cf}_{\mathcal{R}} \varepsilon_0(|\varphi_1|, s')$. Then, to each subtree T' of T, we associate a proof $|T'|$ of the sequent $\vdash^{cf}_{\mathcal{R}} \varepsilon_0(\mathsf{af}(|\varphi_1|), s')$ where s' is the root of T', by induction, as follows,
  - if T' contains a single node s', then the proof $|T'|$ is as follows:

$$\dfrac{\Pi_{(\varphi_1,s')}}{\vdash^{cf}_{\mathcal{R}} \varepsilon_0(\mathsf{af}(|\varphi_1|), s')}\ \vee_1$$

  - if $T' = s'(T_1, \ldots, T_n)$, then the proof $|T'|$ is as follows:

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\vdash^{cf}_{\mathcal{R}} r(s', [\mathsf{next}(s')])}\ \top \qquad \dfrac{|T_1| \quad \ldots \quad |T_n|}{\vdash^{cf}_{\mathcal{R}} \varepsilon_{\sqcap_0}(\mathsf{af}(|\varphi_1|), [\mathsf{next}(s')])}\ \wedge^n}{\vdash^{cf}_{\mathcal{R}} r(s', [\mathsf{next}(s')]) \wedge \varepsilon_{\sqcap_0}(\mathsf{af}(|\varphi_1|), [\mathsf{next}(s')])}\ \wedge}{\vdash^{cf}_{\mathcal{R}} \exists X(r(s', X) \wedge \varepsilon_{\sqcap_0}(\mathsf{af}(|\varphi_1|), X))}\ \exists}{\vdash^{cf}_{\mathcal{R}} \varepsilon_0(\mathsf{af}(|\varphi_1|), s')}\ \vee_2$$

  This way, $|T|$ is a proof of the sequent $\vdash^{cf}_{\mathcal{R}} \varepsilon_0(\mathsf{af}(|\varphi_1|), s)$.

– Suppose $M \models \varepsilon_0(\mathsf{ag}(|\varphi_1|), s)$ holds. By the semantics of $\mathcal{L}$, for each state s' on each lr-path starting from s, $M \models \varepsilon_0(|\varphi_1|, s')$ holds. Thus there exists a finite tree T s.t.
  - T has root s;
  - for each internal node s' in T, the children of s' are labelled by the elements of $\mathsf{next}(s')$;
  - the branch starting from s to each leaf is a lr-path;
  - for each internal node s' in T, $M \models \varepsilon_0(|\varphi_1|, s')$ holds and by IH, there exists a proof $\Pi_{(\varphi_1,s')}$ for the sequent $\vdash^{cf}_{\mathcal{R}} \varepsilon_0(|\varphi_1|, s')$.

  Then, to each subtree T' of T, we associate a proof $|T'|$ of the sequent $\vdash^{cf}_{\mathcal{R}} \varepsilon_1(\mathsf{ag}(|\varphi_1|), s', [s_0'^{k-1}])$ where s' is the root of T' and $s_0'^{k}(s'_k = s')$ is the branch from s to s', by induction, as follows,
  - if T' contains a single node s', then $s_0'^{k}$ is a lr-path and the proof is as follows:

$$\dfrac{\dfrac{}{\vdash^{cf}_{\mathcal{R}} \mathit{mem}(s', [s_0'^{k-1}])}\ \top}{\vdash^{cf}_{\mathcal{R}} \varepsilon_1(\mathsf{ag}(|\varphi_1|), s', [s_0'^{k-1}])}\ \vee_2$$

- if $T' = s'(T_1, \ldots, T_n)$, the proof is as follows:

$$\dfrac{\dfrac{\Pi_{s'}}{\vdash^{cf}_{\mathcal{R}} \varepsilon_0(|\varphi_1|, s')} \qquad \dfrac{\dfrac{\dfrac{}{\vdash^{cf}_{\mathcal{R}} r(s', [\mathsf{next}(s')])}\,\top \qquad \dfrac{|T_1| \qquad \ldots \qquad |T_n|}{\vdash^{cf}_{\mathcal{R}} \varepsilon_{\sqcap_1}(\mathsf{ag}(|\varphi_1|), [\mathsf{next}(s')], [s'^k_0])}\,\wedge^n}{\vdash^{cf}_{\mathcal{R}} r(s', [\mathsf{next}(s')]) \wedge \varepsilon_{\sqcap_1}(\mathsf{ag}(|\varphi_1|), [\mathsf{next}(s')], [s'^k_0])}\,\wedge}{\vdash^{cf}_{\mathcal{R}} \exists X(r(s', X) \wedge \varepsilon_{\sqcap_1}(\mathsf{ag}(|\varphi_1|), X, [s'^k_0]))}\,\exists}{\dfrac{\vdash^{cf}_{\mathcal{R}} \varepsilon_0(|\varphi_1|, s') \wedge \exists X(r(s', X) \wedge \varepsilon_{\sqcap_1}(\mathsf{ag}(|\varphi_1|), X, [s'^k_0]))}{\vdash^{cf}_{\mathcal{R}} \varepsilon_1(\mathsf{ag}(|\varphi_1|), s', [s'^{k-1}_0])}\,\vee_1}\,\wedge$$

This way, as $\varepsilon_0(\mathsf{ag}(|\varphi_1|), s)$ can be rewritten into $\varepsilon_1(\mathsf{ag}(|\varphi_1|), s, \mathsf{nil})$, $|T|$ is a proof for the sequent $\vdash^{cf}_{\mathcal{R}} \varepsilon_0(\mathsf{ag}(|\varphi_1|), s)$. □

Theorem 2 (Soundness and Completeness). *For a CTL proposition φ of NNF, the sequent $\vdash^{cf}_{\mathcal{R}} \varepsilon_0(|\varphi|, s)$ has a proof iff $M \models \varepsilon_0(|\varphi|, s)$ holds.*

6 Applications

6.1 Polarized Resolution Modulo

In Polarized Resolution Modulo, the polarized rewrite rules are taken as one-way clauses [6]. For example, the rewrite rule

$$\varepsilon_1(\mathsf{eg}(\Phi), x, Y) \hookrightarrow_+ mem(x, Y) \vee (\varepsilon_0(\Phi, x) \wedge \exists X(r(x, X) \wedge \varepsilon_{\sqcup_1}(\mathsf{eg}(\Phi), X, \mathsf{con}(x, Y))))$$

is translated into one-way clauses $\underline{\varepsilon_1(\mathsf{eg}(\Phi), x, Y)} \vee mem(x, Y)^{\perp}$ and $\underline{\varepsilon_1(\mathsf{eg}(\Phi), x, Y)} \vee \varepsilon_0(\Phi, x)^{\perp} \vee r(x, X)^{\perp} \vee \varepsilon_{\sqcup_1}(\mathsf{eg}(\Phi), X, \mathsf{con}(x, Y))^{\perp}$, in which the underlined literals have the priority to do resolution.

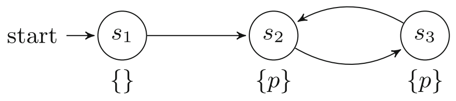

Fig. 4. Resolution Example

Example 2. For the structure M in Fig. 4, we prove that $M, s_1 \models_a EXEGp$. The one-way clauses of M are: $\underline{\varepsilon_0(\mathsf{not}(\overline{p}), s_1)}$, $\underline{\varepsilon_0(\overline{p}, s_2)}$, $\underline{\varepsilon_0(\overline{p}, s_3)}$, $\underline{r(s_1, \mathsf{con}(s_2, \mathsf{nil}))}$, $\underline{r(s_2, \mathsf{con}(s_3, \mathsf{nil}))}$, $\underline{r(s_3, \mathsf{con}(s_2, \mathsf{nil}))}$. The translation of $M, s_1 \models_a EXEGp$ is $\varepsilon_0(\mathsf{ex}(\mathsf{eg}(\overline{p})), s_1)$ and the resolution steps start from

$$\varepsilon_0(\mathsf{ex}(\mathsf{eg}(\overline{p})), s_1)^{\perp}.$$

First apply resolution with $\underline{\varepsilon_0(\mathsf{ex}(\Phi), x)} \vee r(x, X)^{\perp} \vee \varepsilon_{\sqcup_0}(\Phi, X)^{\perp}$, with $x = s_1$ and $\Phi = \mathsf{eg}(\overline{p})$, this yields

$$r(s_1, X)^{\perp} \vee \varepsilon_{\sqcup_0}(\mathsf{eg}(\overline{p}), X)^{\perp}.$$

Then apply resolution with $\underline{r(s_1, \mathsf{con}(s_2, \mathsf{nil}))}$, with $X = \mathsf{con}(s_2, \mathsf{nil})$, this yields

$$\varepsilon_{\sqcup_0}(\mathsf{eg}(\overline{p}), \mathsf{con}(s_2, \mathsf{nil}))^{\perp}.$$

Then apply resolution with $\underline{\varepsilon_{\sqcup_0}(\Phi, \mathsf{con}(x, X))} \vee \varepsilon_0(\Phi, x)^{\perp}$, with $x = s_2$, $X = \mathsf{nil}$ and $\Phi = \mathsf{eg}(\overline{p})$, this yields

$$\varepsilon_0(\mathsf{eg}(\overline{p}), s_2)^{\perp}.$$

Then apply resolution with one-way clause $\underline{\varepsilon_0(\mathsf{eg}(\Phi), x)} \vee \varepsilon_1(\mathsf{eg}(\Phi), x, \mathsf{nil})^{\perp}$, with $\Phi = \overline{p}$ and $x = s_2$, this yields

$$\varepsilon_1(\mathsf{eg}(\overline{p}), s_2, \mathsf{nil})^{\perp}.$$

Then apply resolution with (‡ [1]), with $\Phi = \overline{p}$, $x = s_2$ and $Y = \mathsf{nil}$, this yields

$$\varepsilon_0(\overline{p}, s_2)^{\perp} \vee r(s_2, X)^{\perp} \vee \varepsilon_{\sqcup_1}(\mathsf{eg}(\overline{p}), X, \mathsf{con}(s_2, \mathsf{nil}))^{\perp}.$$

Then apply resolution with $\underline{\varepsilon_0(\overline{p}, s_2)}$, this yields

$$r(s_2, X)^{\perp} \vee \varepsilon_{\sqcup_1}(\mathsf{eg}(\overline{p}), X, \mathsf{con}(s_2, \mathsf{nil}))^{\perp}.$$

Then apply resolution with $\underline{r(s_2, \mathsf{con}(s_3, \mathsf{nil}))}$, with $X = \mathsf{con}(s_3, \mathsf{nil})$, this yields

$$\varepsilon_{\sqcup_1}(\mathsf{eg}(\overline{p}), \mathsf{con}(s_3, \mathsf{nil}), \mathsf{con}(s_2, \mathsf{nil}))^{\perp}.$$

Then apply resolution with $\underline{\varepsilon_{\sqcup_1}(\Phi, \mathsf{con}(x, X), Y)} \vee \varepsilon_1(\Phi, x, Y)^{\perp}$, with $\Phi = \mathsf{eg}(\overline{p})$, $x = s_3$, $X = \mathsf{nil}$ and $Y = \mathsf{con}(s_2, \mathsf{nil})$, this yields

$$\varepsilon_1(\mathsf{eg}(\overline{p}), s_3, \mathsf{con}(s_2, \mathsf{nil}))^{\perp}.$$

Then apply resolution with (‡), with $\Phi = \overline{p}$, $x = s_3$, $Y = \mathsf{con}(s_2, \mathsf{nil})$, this yields

$$\varepsilon_0(\overline{p}, s_3)^{\perp} \vee r(s_3, X)^{\perp} \vee \varepsilon_{\sqcup_1}(\mathsf{eg}(\overline{p}), X, \mathsf{con}(s_3, \mathsf{con}(s_2, \mathsf{nil})))^{\perp}.$$

Then apply resolution with $\underline{\varepsilon_0(\overline{p}, s_3)}$, this yields

$$r(s_3, X)^{\perp} \vee \varepsilon_{\sqcup_1}(\mathsf{eg}(\overline{p}), X, \mathsf{con}(s_3, \mathsf{con}(s_2, \mathsf{nil})))^{\perp}.$$

Then apply resolution with $\underline{r(s_3, \mathsf{con}(s_2, \mathsf{nil}))}$, with $X = \mathsf{con}(s_2, \mathsf{nil})$, this yields

$$\varepsilon_{\sqcup_1}(\mathsf{eg}(\overline{p}), \mathsf{con}(s_2, \mathsf{nil}), \mathsf{con}(s_3, \mathsf{con}(s_2, \mathsf{nil})))^{\perp}.$$

Then apply resolution with $\underline{\varepsilon_{\sqcup_1}(\Phi, \mathsf{con}(x, X), Y)} \vee \varepsilon_1(\Phi, x, Y)^{\perp}$, with $\Phi = \mathsf{eg}(\overline{p})$, $x = s_3$, $X = \mathsf{nil}$ and $Y = \mathsf{con}(s_2, \mathsf{nil})$, this yields

$$\varepsilon_1(\mathsf{eg}(\overline{p}), s_2, \mathsf{con}(s_3, \mathsf{con}(s_2, \mathsf{nil})))^{\perp}.$$

Then apply resolution with $\underline{\varepsilon_1(\mathsf{eg}(\Phi), x, Y)} \vee mem(x, Y)^{\perp}$, with $x = s_2$ and $Y = \mathsf{con}(s_3, \mathsf{con}(s_2, \mathsf{nil}))$, this yields

$$mem(s_2, \mathsf{con}(s_3, \mathsf{con}(s_2, \mathsf{nil})))^{\perp}.$$

Then apply resolution with $\underline{mem(x, \mathsf{con}(y, Z))} \vee mem(x, Z)^{\perp}$, with $x = s_2$, $y = s_3$ and $Z = \mathsf{con}(s_2, \mathsf{nil})$, this yields

$$mem(s_2, \mathsf{con}(s_2, \mathsf{nil}))^{\perp}.$$

Then apply resolution with $\underline{mem(x, \mathsf{con}(x, Z))}$, with $x = s_2$ and $Z = \mathsf{nil}$, this yields the empty clause. Thus $M, s_1 \models_a EXEGp$ holds.

[1] ‡ is $\underline{\varepsilon_1(\mathsf{eg}(\Phi), x, Y)} \vee \varepsilon_0(\Phi, x)^{\perp} \vee r(x, X)^{\perp} \vee \varepsilon_{\sqcup_1}(\mathsf{eg}(\Phi), X, \mathsf{con}(x, Y))^{\perp}$.

6.2 Experimental Evaluation

In this Section, we give a comparison between Resolution-based and QBF-based verification, that are implemented in iProver Modulo and VERDS [15] respectively. iProver Modulo is a prover by embedding Polarized Resolution Modulo into iProver [11]. The comparison is by proving 24 CTL properties on two kinds of programs: *Programs with Concurrent Processes* and *Programs with Concurrent Sequential Processes*. The programs and CTL properties refer to [16].

Table 1. Experimental Results

iProver/Verds		Con. Processes			Con. Seq. Processes		
Prop	Num	True	False	>20m	True	False	>20m
p_{01}	40	-	40/40	-	23/-	5/4	12/36
p_{02}	40	40/40	-	-	40/40	-	-
p_{03}	40	2/-	37/37	1/3	-	25/15	15/25
p_{04}	40	17/-	-	23/40	-	-	40/40
p_{05}	40	25/34	6/5	9/1	24/24	8/2	8/14
p_{06}	40	31/40	-	9/-	36/31	-	4/9
p_{07}	40	40/40	-	-	40/40	-	-
p_{08}	40	40/40	-	-	40/40	-	-
p_{09}	40	32/32	8/8	-	35/29	5/1	-/10
p_{10}	40	40/40	-	-	40/40	-	-
p_{11}	40	10/10	30/30	-	27/23	8/4	5/13
p_{12}	40	40/40	-	-	40/35	-	-/5
p_{13}	40	-	40/40	-	-	40/40	-
p_{14}	40	3/3	37/37	-	3/3	37/33	-/4
p_{15}	40	5/-	33/33	2/7	-	23/15	17/25
p_{16}	40	19/-	-	21/40	-	-	40/40
p_{17}	40	28/37	3/2	9/1	25/26	5/1	10/13
p_{18}	40	32/40	-	8/-	37/31	-	3/9
p_{19}	40	5/5	35/35	-	6/6	34/34	-
p_{20}	40	15/17	21/21	4/2	12/11	18/22	10/7
p_{21}	40	3/3	37/37	-	3/3	37/37	-
p_{22}	40	3/3	37/37	-	3/3	37/37	-
p_{23}	40	-	40/40	-	-	40/40	-
p_{24}	40	20/25	12/10	8/5	8/8	23/22	9/10
Sum	960	450/449	416/412	94/99	442/393	345/307	173/260

For the *Con. Processes*, each testing case contains 12/24 variables and 3 processes. For the *Con. Seq. Processes*, each testing case contains 12/16 variables and 2 processes.

Table 2. Speed Comparisons

		Con. Processes			Con. Seq. Processes		
Prop	Num	adv/T	adv/F	O(iP/Ver)	adv/T	adv/F	O(iP/Ver)
p_{01}	40	-	0/40	-	-	0/3	25/1
p_{02}	40	40/40	-	-	40/40	-	-
p_{03}	40	-	1/37	2/-	-	11/15	10/-
p_{04}	40	-	-	17/-	-	-	-
p_{05}	40	0/25	3/5	1/9	6/20	2/2	10/4
p_{06}	40	0/31	-	-/9	10/28	-	8/3
p_{07}	40	33/40	-	-	37/40	-	-
p_{08}	40	35/40	-	-	38/40	-	-
p_{09}	40	19/32	0/8	-	22/29	0/1	10/-
p_{10}	40	19/40	-	-	18/40	-	-
p_{11}	40	0/10	0/30	-	9/23	3/4	8/-
p_{12}	40	3/40	-	-	7/35	-	5/-
p_{13}	40	-	38/40	-	-	40/40	-
p_{14}	40	2/3	0/37	-	3/3	23/33	4/-
p_{15}	40	-	0/33	5/-	-	10/14	9/1
p_{16}	40	-	-	19/-	-	-	-
p_{17}	40	0/28	1/2	1/9	8/22	1/1	7/4
p_{18}	40	0/32	-	-/8	11/29	-	8/2
p_{19}	40	2/5	9/35	-	6/6	12/34	-
p_{20}	40	1/15	7/20	1/3	6/11	9/17	2/5
p_{21}	40	2/3	18/37	-	3/3	23/37	-
p_{22}	40	2/3	19/37	-	2/3	22/37	-
p_{23}	40	-	17/40	-	-	25/40	-
p_{24}	40	0/20	1/10	2/5	1/7	4/21	3/2
Sum	960	158/407	114/411	48/43	227/379	185/299	109/22

All the cases are tested on Intel® Core ™ i5-2400 CPU @ 3.10 GHz × 4 with Linux and the testing time of each case is limited to 20 min. The experimental data is presented in Tables 1 and 2[2]. The comparison is based on two aspects: the number of testing cases that can be proved, and the time used if a problem can be proved in both. As can be seen in Table 1, among the 960 testing cases of the *Con. Processes*, 94 of them are timeout in iProver Modulo, while the number in VERDS is 99. For the *Con. Seq. Processes*, among the 960 testing cases, 173 of them are timeout in iProver Modulo, while in VERDS, the number is 260. Table 2 shows that, among the 818 testing cases of the *Con. Processes*, that are both proved in iProver Modulo and VERDS, iProver

[2] adv/T(F): has advantage in speed when both return T(F). O(iP/Ver): only solved by iProver/Verds.

Modulo performs better in 272 of them and among the 678 testing cases of the *Con. Seq. Processes*, 412 of them run faster in iProver Modulo.

In summary, for the 1920 testing cases, 1653 (86 %) of them are solved by iProver Modulo, while 1561 (81 %) are solved by VERDS. For all the 1496 testing cases that are both proved, 684 (45.8 %) testing cases run faster in iProver Modulo.

7 Conclusion and Future Work

In this paper, we defined an alternative semantics for CTL, which is bounded to lr-paths. Based on the alternative semantics, a way to embed model checking problems into Deduction Modulo has been presented. Thus this work has given a method to solve model checking problems in automated theorem provers. An experimental evaluation of this approach using resolution modulo has been presented. The comparison with the QBF-based verification showed that automated theorem proving modulo, which performed as well as QBF-based method, can be considered as a new way to quickly determine whether a property is violated in transition system models.

The proof-search method does not work well on proving some temporal propositions, such as the propositions start with AG. One of the reasons is during the search steps, it may visit the same state repeatedly. To design new rewrite rules for the encoding of temporal connectives or new elimination rules to avoid this problem remains as future work.

Acknowledgements. I am grateful to Gilles Dowek, for his careful reading and comments.

References

1. Biere, A., Cimatti, A., Clarke, E., Zhu, Y.: Symbolic model checking without BDDs. In: Cleaveland, W.R. (ed.) TACAS 1999. LNCS, vol. 1579, pp. 193–207. Springer, Heidelberg (1999)
2. Burel, G.: Embedding deduction modulo into a prover. In: Dawar, A., Veith, H. (eds.) CSL 2010. LNCS, vol. 6247, pp. 155–169. Springer, Heidelberg (2010)
3. Burel, G.: Experimenting with deduction modulo. In: Bjørner, N., Sofronie-Stokkermans, V. (eds.) CADE 2011. LNCS, vol. 6803, pp. 162–176. Springer, Heidelberg (2011)
4. Clarke Jr., E.M., Grumberg, O., Peled, D.A.: Model Checking. MIT Press, Cambridge, MA, USA (1999)
5. Delahaye, D., Doligez, D., Gilbert, F., Halmagrand, P., Hermant, O.: Zenon modulo: when Achilles outruns the tortoise using deduction modulo. In: McMillan, K., Middeldorp, A., Voronkov, A. (eds.) LPAR-19 2013. LNCS, vol. 8312, pp. 274–290. Springer, Heidelberg (2013)
6. Dowek, G.: Polarized resolution modulo. In: Calude, C.S., Sassone, V. (eds.) TCS 2010. IFIP AICT, vol. 323, pp. 182–196. Springer, Heidelberg (2010)
7. Dowek, G., Hardin, T., Kirchner, C.: Theorem proving modulo. J. Autom. reasoning **31**, 33–72 (2003)
8. Dowek, G., Jiang, Y.: A Logical Approach to CTL (2013). http://hal.inria.fr/docs/00/91/94/67/PDF/ctl.pdf (manuscript)

9. Dowek, G., Jiang, Y.: Axiomatizing Truth in a Finite Model (2013). https://who.rocq.inria.fr/Gilles.Dowek/Publi/classes.pdf (manuscript)
10. Ji, K.: CTL Model Checking in Deduction Modulo. In: Felty, A.P., Middeldorp, A. (eds.) CADE-25, 2015. LNCS, vol. 9195, pp. xx–yy (2015). https://drive.google.com/file/d/0B0CYADxmoWB5UGJsV2UzNnVqVHM/view?usp=sharing (fullpaper)
11. Korovin, K.: iProver – an instantiation-based theorem prover for first-order logic (system description). In: Armando, A., Baumgartner, P., Dowek, G. (eds.) IJCAR 2008. LNCS (LNAI), vol. 5195, pp. 292–298. Springer, Heidelberg (2008)
12. Rajan, S., Shankar, N., Srivas, M.: An Integration of Model Checking with Automated Proof Checking. In: Wolper, P. (ed.) CAV 1995. LNCS, vol. 939, pp. 84–97. Springer, Berlin Heidelberg (1995)
13. Troelstra, A.S., Schwichtenberg, H.: Basic Proof Theory. Cambridge University Press, New York (1996)
14. Zhang, W.: Bounded semantics of CTL and SAT-based verification. In: Breitman, K., Cavalcanti, A. (eds.) ICFEM 2009. LNCS, vol. 5885, pp. 286–305. Springer, Heidelberg (2009)
15. Zhang, W.: VERDS Modeling Language (2012). http://lcs.ios.ac.cn/~zwh/verds/index.html
16. Zhang, W.: QBF encoding of temporal properties and QBF-based verification. In: Demri, S., Kapur, D., Weidenbach, C. (eds.) IJCAR 2014. LNCS, vol. 8562, pp. 224–239. Springer, Heidelberg (2014)