Reducing CTL-live Model Checking to Semantic Entailment in First-Order Logic (Version 1)

Amirhossein Vakili and Nancy A. Day
Cheriton School of Computer Science
University of Waterloo
Waterloo, Ontario, Canada, N2L 3G1
{avakili,nday}@uwaterloo.ca

Technical Report: CS-2014-05
March, 2014

Abstract—The core of temporal logic model checking is the reachability problem, which is not expressible in first-order logic (FOL). Most model checking algorithms, both for finite and infinite Kripke structures, contain a loop that iterates to reach a fixed-point. As a result, reasoners with input languages no more expressive than FOL have been used iteratively for model checking rather than having the reasoner solve the problem completely by itself. In this article, we present a method for reducing model checking of finite and infinite Kripke structures that are expressed in FOL to entailment checking in FOL for a fragment of computational tree logic (CTL), which we call *CTL-live*. CTL-live includes all the CTL connectives that are expressible in the mu-calculus using the *least* fixed-point operator. These connectives are traditionally used to express liveness properties. This reduction allows us to consider model checking of CTL-live as a FOL theorem proving problem, and to use directly FOL reasoning techniques for model checking without the need of fixed-point operators, transitive-closure, or induction. We prove that CTL-live is maximal in the sense that model checking of CTL connectives that are not included in CTL-live is not reducible to semantic entailment in FOL.

1 Introduction

Model checking is the problem of checking whether a Kripke structure satisfies a temporal logic formula [1]. Model checking methods that use first-order reasoners (tools with input languages no more expressive than first-order logic (FOL), such SMT solvers [2]), can be divided into two major categories: 1) bounded model checking (e.g., [3], [4]) and 2) unbounded model checking (e.g., [5], [6]). Bounded methods check whether a property holds for a certain length of execution path by creating a formula consisting of the transition relation iterated to the desired bound. Since the bound is finite, the problem can be expressed in FOL, therefore, FOL reasoners can be used to solve the entire bounded (and therefore incomplete) model checking problem at one time. Unbounded methods call a FOL reasoner multiple times iteratively to check whether a fixed-point has been reached. These methods are mostly used for safety properties; for infinite systems, termination (without approximation) is guarantied only in the case where the property is violated. FOL reasoners have not been used to solve an entire unbounded model checking problem in one call because model checking is a reachability query about a graph (in this case a Kripke structure), which is not expressible in FOL [7].

Our first contribution in this article is to show that model checking of a fragment of computational tree logic (CTL), which we call *CTL-live*, is reducible to semantic entailment checking in FOL; in other words, model checking CTL-live properties of a Kripke structure can be done completely using deduction techniques of FOL. Thus, some reachability queries can be answered using an FOL reasoner even though they are traditionally solved by transitive-closure, which is not expressible in FOL. CTL-live includes all the CTL connectives that are expressible in the mu-calculus using the least fixed-point operator [1]. These connectives are traditionally used to express liveness properties. Our result holds for any Kripke structure expressible in FOL. Since FOL entailment checking is recursively enumerable [8], [9], our reduction can be used to generate automatically a proof if a CTL-live property is satisfied by a Kripke structure. This is the opposite of iterative unbounded methods, such as [6], which guarantee termination only if the property is not satisfied.

Model checking a CTL formula φ requires checking whether the set of initial states of a Kripke structure is included in the set of states that satisfy φ. Semantic entailment in FOL implicitly uses a universal quantifier over interpretations, which is not a first-order quantifier. The key insight in our approach is to use this implicit non-first-order quantifier to express a set of states that includes every state that satisfies φ and possibly more; this set is sufficient to solve the model checking problem for a CTL-live formula.

Our second contribution is that we show CTL-live is *maximal* in the sense that model checking of CTL connectives that are not included in CTL-live is not reducible to semantic entailment in FOL. For the CTL connectives not included in CTL-live, non-FOL reasoning techniques are required.

The rest of this article is organized as follows: Section 2 gives the definitions and notation used in this article; Section 3 describes how we represent Kripke structures in FOL. CTL-live and the reduction of its model checking to semantic entailment in FOL is presented in Section 4. Section 5 shows the maximality of CTL-live. Section 6 presents the related work, and Section 7 concludes the article.

2 Background

This section provides a standard summary of first-order logic (FOL) and computational tree logic (CTL) useful for understanding the remainder of our article. In this article, we use superscripts for FOL elements and subscripts for Kripke structures. Thus, the notation $X^{\mathcal{K}}$ denotes the value of X under interpretation $\mathcal{K}$, and $X_{\mathcal{K}}$ is a set of states in the Kripke structure $\mathcal{K}$.

2.1 First-Order Logic

Formulae in FOL are built from logical connectives, functional symbols, variables, and relational symbols [7]. The set of logical connectives and their semantics in FOL is fixed. The following is a standard set of logical connectives for FOL: true ($\top$), negation ($\neg$), disjunction ($\vee$), conjunction ($\wedge$), implication ($\rightarrow$), iff ($\leftrightarrow$), existential quantifier ($\exists$), universal quantifier ($\forall$), and equality ($=$). The set of functional and relational symbols and their semantics depends on the context and the constraints that they must satisfy. Since for different problems different sets of functional and relational symbols are used, we have the following definition:

Definition 1: **(Base)** A base for FOL is a pair of sets, $B = \langle F, R\rangle$, where F and R are sets of functional and relational symbols respectively.

Every functional and relational symbol has a corresponding arity, which is the number of arguments that is required by that symbol. Constants are considered to be functional symbols with arity 0. A symbol X with arity n, where n is nonzero, is denoted by X/n. The arity of a relational symbol is nonzero. The set of FOL formulae over base $B = \langle F, R\rangle$ is defined by the following grammar:

$$\begin{aligned}\Phi \quad &::= \quad r(t_1,\ldots,t_n) \mid t_1 = t_2 \quad \text{where } r/n \in R,\\ &::= \quad \neg\Phi \mid \Phi_1 \vee \Phi_2 \mid \Phi_1 \wedge \Phi_2 \mid \Phi_1 \rightarrow \Phi_2 \mid \Phi_1 \leftrightarrow \Phi_2\\ &::= \quad \exists v : \Phi \mid \forall v : \Phi \text{ where } v \text{ is a variable.} \qquad (1)\end{aligned}$$

$$\begin{aligned}t \quad &::= \quad v \quad \text{where } v \text{ is a variable,}\\ &::= \quad f(t_1,\ldots,t_n) \text{ where } f \in F \text{ and it is } n\text{-ary.} \qquad (2)\end{aligned}$$

Equations 1 and 2 are the rules for constructing formulae and terms respectively.

The semantics of a FOL formulae is defined using *interpretations*. An interpretation defines the meaning of a base by assigning values to variables, functional and relational symbols. Using these values along with the fixed semantics of FOL logical connectives, a formula evaluates to *true* or *false*. Given a base $B = \langle F, R\rangle$, an interpretation is a pair $\mathcal{I} = \langle \mathcal{D}, .^{\mathcal{I}}\rangle$, where $\mathcal{D}$ is a non-empty set, the *domain* of $\mathcal{I}$, and $.^{\mathcal{I}}$ is a mapping that assigns:

1) to every variable v an element in $\mathcal{D}$, $v^{\mathcal{I}} \in \mathcal{D}$,
2) to every 0-ary $c \in F$ an element in $\mathcal{D}$, $c^{\mathcal{I}} \in \mathcal{D}$,
3) to every functional symbol $f \in F$ of arity $n \geq 1$ a total function from $\mathcal{D}^n$ to $\mathcal{D}$, $f^{\mathcal{I}} : \mathcal{D}^n \mapsto \mathcal{D}$,
4) to every relational symbol $r/n \in R$ a subset of $\mathcal{D}^n$, $r^{\mathcal{I}} \subseteq \mathcal{D}^n$.

Definition 2: **(Semantics of FOL)** Let $B = \langle F, R\rangle$ be a base for FOL and $\mathcal{I} = \langle \mathcal{D}, .^{\mathcal{I}}\rangle$ an interpretation for B. The satisfiability relation over formulae and interpretations, $\Vdash$, is defined by using structural induction on Φ and t. In the following, $\mathcal{I}^{x:=d}$ is an interpretation over B that is same as $\mathcal{I}$ except that it maps the variable x to d.

$$\begin{array}{lcl}\mathcal{I} \Vdash r(t_1,\ldots,t_n) & \Longleftrightarrow & r^{\mathcal{I}}(t_1^{\mathcal{I}},\ldots,t_n^{\mathcal{I}}) \text{ holds,}\\ \mathcal{I} \Vdash t_1 = t_2 & \Longleftrightarrow & t_1^{\mathcal{I}} \text{ is equal to } t_2^{\mathcal{I}},\\ \mathcal{I} \Vdash \neg\Phi & \Longleftrightarrow & \mathcal{I} \Vdash \Phi \text{ does \textit{not} hold,}\\ \mathcal{I} \Vdash \Phi_1 \vee \Phi_2 & \Longleftrightarrow & \mathcal{I} \Vdash \Phi_1 \text{ or } \mathcal{I} \Vdash \Phi_2,\\ \mathcal{I} \Vdash \exists x : \Phi & \Longleftrightarrow & \text{there exists a } d \in \mathcal{D}\\ & & \text{such that } \mathcal{I}^{x:=d} \Vdash \Phi.\\ (f(t_1,\ldots,t_n))^{\mathcal{I}} & = & f^{\mathcal{I}}(t_1^{\mathcal{I}},\ldots,t_n^{\mathcal{I}})\end{array}$$

The FOL connectives that are mentioned in Definition 2 form a complete fragment of FOL: the other connectives can be written in terms of $\neg$, $\vee$, and $\exists$; e.g., $\forall x : \Phi$ is equivalent to $\neg(\exists x : \neg\Phi)$.

Definition 3: **(Semantic entailment)** Suppose Γ is a set of FOL formulae and Φ is an FOL formula: Γ entails Φ, denoted by $\Gamma \models \Phi$, iff every interpretation that satisfies all the formulae in Γ also satisfies Φ:

$$\Gamma \models \Phi \quad \Longleftrightarrow \quad \forall \mathcal{I} : \big(\forall \Psi \in \Gamma : \mathcal{I} \Vdash \Psi\big) \Longrightarrow \mathcal{I} \Vdash \Phi$$

Semantic entailment checking for FOL is recursively enumerable [8], [9]. This means that semantic entailment checking for FOL is not computable, but there is procedure that given Γ and Φ produces a proof in the case where $\Gamma \models \Phi$.

2.2 Computational Tree Logic

Computational tree logic (CTL) is a temporal logic for specifying properties over time [10]. The semantics of CTL formulae is defined by using Kripke structures. A Kripke structure is a directed labelled graph. The set of vertexes and the set of edges of a Kripke structure are often called the *state space* and the *transition relation* respectively. The labels of each state show the local properties of the state: what holds and what does not hold in a state. In practice, different combinations of the values of the variables that are used to define a system represent the state space.

Definition 4: **(Kripke structure)** A Kripke structure is a four tuple, $\mathcal{K} = \langle S_{\mathcal{K}}, I_{\mathcal{K}}, N_{\mathcal{K}}, \mathbb{P}_{\mathcal{K}} \rangle$, where: $S_{\mathcal{K}}$ is a set of states; $I_{\mathcal{K}}$, the set of initial states, is a non-empty subset of $S_{\mathcal{K}}$; $N_{\mathcal{K}}$, the next-state relation, is a total binary relation over $S_{\mathcal{K}}$; $\mathbb{P}_{\mathcal{K}}$ is a finite set of unary predicates over states. Predicates represent the local properties of the states, and are called *labelling predicates*.

The syntax of CTL is defined for a given set of labelling predicates $\mathbb{P}$:

$$\begin{aligned} \varphi \quad &::= \quad P \mid \neg\varphi \mid \varphi_1 \vee \varphi_2 \mid \varphi_1 \wedge \varphi_2 \text{ where } P \in \mathbb{P} \\ &::= \quad EX\varphi \mid AX\varphi \mid EF\varphi \mid AF\varphi \mid EG\varphi \mid AG\varphi \\ &::= \quad \varphi_1 AU \varphi_2 \mid \varphi_1 EU \varphi_2 \end{aligned} \tag{3}$$

A Kripke structure defines a set of infinite computation paths, where each path represents a trace of execution. A computation path starting at state $s \in S_{\mathcal{K}}$ is a sequence of states, $s_0 \mapsto s_1 \mapsto \ldots$ such that $s = s_0$ and for every $i \geq 0$, $N_{\mathcal{K}}(s_i, s_{i+1})$. CTL is a branching-time temporal logic. A temporal connective of CTL consists of two parts: a path and a state quantifier. A path quantifier is either E, there exists a path, or A, for all paths. The state quantifiers are X (next state), F (eventually), G (globally), and U (strong until). The satisfiability relation for CTL, $\models_c$, is used to give meaning to CTL formulae. The notation $\mathcal{K}, s \models_c \varphi$ denotes that the state s of the Kripke structure $\mathcal{K}$ satisfies the CTL formula φ and $\mathcal{K}, s \not\models_c \varphi$ is used when $\mathcal{K}, s \models_c \varphi$ does not hold.

Definition 5: **(Semantics of CTL)** Let $\mathcal{K} = \langle S_{\mathcal{K}}, I_{\mathcal{K}}, N_{\mathcal{K}}, \mathbb{P}_{\mathcal{K}} \rangle$ be a Kripke structure and φ a CTL formula. The satisfiability relation for CTL, $\models_c$, is defined by structural induction on φ:

$$\begin{array}{lll} \mathcal{K}, s \models_c P & \iff & P(s) \text{ holds, where } P \in \mathbb{P}_{\mathcal{K}} \\ \mathcal{K}, s \models_c \neg\varphi & \iff & \mathcal{K}, s \not\models_c \varphi \\ \mathcal{K}, s \models_c \varphi_1 \vee \varphi_2 & \iff & \mathcal{K}, s \models_c \varphi_1 \ \vee \ \mathcal{K}, s \models_c \varphi_2 \\ \mathcal{K}, s \models_c EX\varphi & \iff & \exists s' \in S : N_{\mathcal{K}}(s, s') \wedge \mathcal{K}, s' \models_c \varphi \\ \mathcal{K}, s \models_c EG\varphi & \iff & \text{there exists a path } s_0 \mapsto s_1 \mapsto \ldots \\ & & \text{such that } s_0 = s \text{ and} \\ & & \text{for all } i\text{'s } \mathcal{K}, s_i \models_c \varphi. \\ \mathcal{K}, s \models_c \varphi_1 EU \varphi_2 & \iff & \text{there exists a } j \text{ and a path,} \\ & & s_0 \mapsto s_1 \mapsto \ldots, \text{ such that} \\ & & s = s_0,\ \mathcal{K}, s_j \models_c \varphi_2, \text{ and} \\ & & \text{for all } i < j\ \mathcal{K}, s_i \models_c \varphi_1. \end{array}$$

The connectives that are mentioned in Definition 5 form a complete fragment for CTL; e.g., $\varphi_1 AU \varphi_2$ is equivalent to $\neg(\neg\varphi_2 EU(\neg\varphi_1 \wedge \neg\varphi_2)) \wedge \neg(EG\neg\varphi_2)$, and $EF\varphi$ is equivalent to $\top EU \varphi$[1].

Symbol	Purpose
$S/1$	CP representing the set of states
$I/1$	CP representing the set of initial states
$N/2$	CP representing the next-state relation
$P/1$	CP's representing the labelling predicates

Fig. 1. Relational symbols required to specify Kripke structures (CP is "characteristic predicate")

The set of states of a Kripke structure $\mathcal{K}$ that satisfies a CTL formula φ is denoted by $[\varphi]_{\mathcal{K}}$:

$$[\varphi]_{\mathcal{K}} = \{ s \in S_{\mathcal{K}} \mid \quad \mathcal{K}, s \models_c \varphi \}$$

The Kripke structure $\mathcal{K}$ satisfies the CTL formula φ, denoted by $\mathcal{K} \models_c \varphi$, iff for all $s \in I_{\mathcal{K}}$ we have $\mathcal{K}, s \models_c \varphi$:

$$\mathcal{K} \models_c \varphi \iff I_{\mathcal{K}} \subseteq [\varphi]_{\mathcal{K}}$$

From the semantics of CTL, we can conclude that if two Kripke structures agree on every component except the labelling predicates, they have the same properties with respect to the CTL formulae that can be evaluated against both of them. A Kripke structure $\mathcal{K}_1 = \langle S_{\mathcal{K}_1}, I_{\mathcal{K}_1}, N_{\mathcal{K}_1}, \mathbb{P}_{\mathcal{K}_1} \rangle$ is a substructure of $\mathcal{K}_2 = \langle S_{\mathcal{K}_2}, I_{\mathcal{K}_2}, N_{\mathcal{K}_2}, \mathbb{P}_{\mathcal{K}_2} \rangle$, denoted by $\mathcal{K}_1 \sqsubseteq \mathcal{K}_2$, iff the following conditions hold:

$$S_{\mathcal{K}_1} = S_{\mathcal{K}_2}\ ,\ I_{\mathcal{K}_1} = I_{\mathcal{K}_2}\ ,\ N_{\mathcal{K}_1} = N_{\mathcal{K}_2}\ ,\ \mathbb{P}_{\mathcal{K}_1} \subseteq \mathbb{P}_{\mathcal{K}_2}$$

Theorem 1: Suppose $\mathcal{K}_1 \sqsubseteq \mathcal{K}_2$ and φ is a CTL formula over $\mathbb{P}_{\mathcal{K}_1}$; we have:

$$\mathcal{K}_1 \models_c \varphi \quad \Longrightarrow \quad \mathcal{K}_2 \models_c \varphi$$

Proof: By using the semantics of CTL. □

3 Kripke Structures in FOL

Modelling a Kripke structure in FOL requires a base that has at least the relational symbols of Figure 1. FOL formulae over such a base can be used to define the state space, the initial states, the next-state relation and the labelling predicates. Every satisfying interpretation of the FOL formulae represents a Kripke structure. The key observation here is that the relational symbols themselves do **not** represent a Kripke structure. A satisfying interpretation of the FOL formulae determines the content of these relational symbols, and as a result, represents a Kripke structure; the set of all the satisfying interpretations forms a class of Kripke structures. We call a set of formulae that represent a class of Kripke structures a *declarative model of a dynamic system* (for short, a declarative model) [11].

Definition 6: **(Declarative model)** A declarative model $\mathfrak{D}$ is a pair $\mathfrak{D} = \langle B, \Gamma \rangle$: B is a base that includes the

1. $\top$ is equivalent to $P \vee \neg P$ for any labelling predicate P.

relational symbols in Figure 1; Γ is a set of FOL formulae over B that includes the well-formedness constraints in Definition 4, such as "the set of initial states is not empty" and "the next-state relation is total". The **C**lass of **K**ripke structures represented by a declarative model $\mathfrak{D}$ is denoted by $CK(\mathfrak{D})$:

$$CK(\mathfrak{D}) = \{\mathcal{K} \mid \forall \Phi \in \Gamma : \ \mathcal{K} \Vdash \Phi\} \tag{4}$$

Inclusion of the well-formedness formulae in Γ insures that every member of $CK(\mathfrak{D})$, according to Definition 4, is a valid Kripke structure.

There are many reasons for a set of FOL formulae to have more than one satisfying interpretation: the use of uninterpreted functions (relations) can result in more than one satisfying interpretation; moreover, under-constraining a model makes it possible to have non-isomorphic Kripke structures that are satisfying interpretations.

Other work on modelling Kripke structures in a logic no more expressive than FOL (e.g., [3], [6], [12]) has taken the approach of directly modelling the constraints on variables such as v and v', where v' represents the value of v in the next state, rather than modelling the next-state relation explicitly. In these approaches, the satisfying interpretations of the constraints are the possible transitions of a unique next-state relation (and therefore, unique Kripke structure). In our work, we take a more general approach by modelling the next-state relation explicitly; therefore, the set of satisfying interpretations can include multiple Kripke structures.

In Definition 6, where $CK(\mathfrak{D})$ is a class of Kripke structures, two model checking questions can be studied: 1) *do all the Kripke structures* in $CK(\mathfrak{D})$ satisfy the property? 2) *is there a Kripke structure* in $CK(\mathfrak{D})$ that satisfies the property? We define two model checking problems for a class of Kripke structures that correspond to these questions [11]:

Definition 7: **(Universal model checking)** The universal model checking of a declarative model $\mathfrak{D}$ and a CTL formula φ, denoted by $\mathfrak{D} \models_\forall \varphi$, is defined as checking whether all the Kripke structures in $CK(\mathfrak{D})$ satisfy φ:

$$\mathfrak{D} \models_\forall \varphi \iff \forall\ \mathcal{K} \in CK(\mathfrak{D}) : \mathcal{K} \models_c \varphi$$

Definition 8: **(Existential model checking)** The existential model checking of the declarative model $\mathfrak{D}$ and a CTL formula φ, $\mathfrak{D} \models_\exists \varphi$, is defined as checking whether there exists a Kripke structure in $CK(\mathfrak{D})$ that satisfies φ:

$$\mathfrak{D} \models_\exists \varphi \iff \exists\ \mathcal{K} \in CK(\mathfrak{D}) : \mathcal{K} \models_c \varphi$$

Figure 2 is a summary of the satisfiability notations used in this article.

Symbol			Meaning	
$\mathcal{I}$	$\Vdash$	Φ	FOL satisfiability	Definition 2
Γ	$\models$	Φ	FOL entailment	Definition 3
$\mathcal{K}$	$\models_c$	φ	CTL model checking	Definition 5
$\mathfrak{D}$	$\models_\forall$	φ	Universal model checking	Definition 7
$\mathfrak{D}$	$\models_\exists$	φ	Existential model checking	Definition 8

Fig. 2. Summary of satisfiability notations

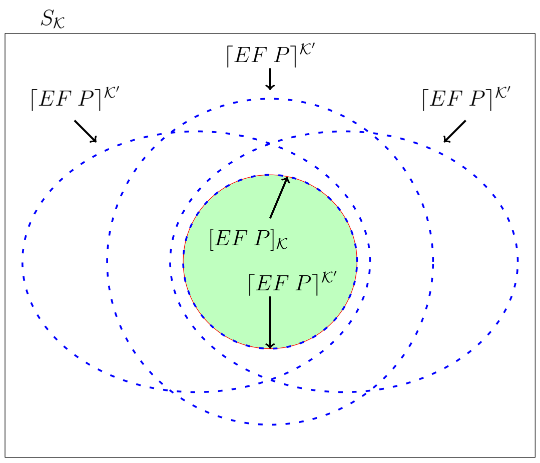
$S_{\mathcal{K}}$
$\lceil EF\ P\rceil^{\mathcal{K}'}$
$\lceil EF\ P\rceil^{\mathcal{K}'}$
$\lceil EF\ P\rceil^{\mathcal{K}'}$
$[EF\ P]_{\mathcal{K}}$
$\lceil EF\ P\rceil^{\mathcal{K}'}$

Fig. 3. $[EF\ P]_{\mathcal{K}} = \bigcap_{\mathcal{K}' \in CK(\mathfrak{A})} \lceil EF\ P\rceil^{\mathcal{K}'}$

4 CTL Model Checking in FOL

In this section, we present the first contribution of our work: identifying a fragment of CTL whose model checking can be done directly using a FOL reasoner. First, we give the intuition behind our approach, then, we focus on universal model checking. We also show how this approach can be used for existential model checking by studying the relation between these two model checking questions.

4.1 Intuition

Suppose $\mathcal{K}$ is a Kripke structure, $P \in \mathbb{P}_{\mathcal{K}}$ is a labelling predicate, and we are interested in checking whether $\mathcal{K}$ satisfies $EF\ P$. From the encoding of CTL in the mu-calculus, we know that the set of states that satisfy $EF\ P$, $[EF\ P]_{\mathcal{K}}$, is the **smallest** set such that its characteristic predicate, $\lceil EF\ P\rceil$, satisfies the following two FOL formulae:

$$\begin{aligned} &1)\quad \forall s : P(s) \rightarrow \lceil EF\ P\rceil(s)\\ &2)\quad \forall s : \big(\exists s' : N_{\mathcal{K}}(s,s') \wedge \lceil EF\ P\rceil(s')\big) \rightarrow \lceil EF\ P\rceil(s) \end{aligned} \tag{5}$$

Intuitively, the first constraint states that every state that satisfies P, also satisfies $EF\ P$. The second constraint states that if a state s has a next state that satisfies $EF\ P$, then s also satisfies $EF\ P$.

Adding $\lceil EF\ P\rceil$ as a new labelling predicate to $\mathcal{K}$ along with the two formulae in Equation 5 results in a declarative model $\mathfrak{A}$ that represents a class of Kripke structures $CK(\mathfrak{A})$. This class has the following property depicted in Figure 3:

$$[EF\ P]_{\mathcal{K}} = \bigcap_{\mathcal{K}' \in CK(\mathfrak{A})} \lceil EF\ P\rceil^{\mathcal{K}'}$$

Temporal part		
φ	$::=$	$\pi \mid EX\varphi \mid AX\varphi$
	$::=$	$\varphi_1 \vee \varphi_2 \mid \varphi_1 \wedge \varphi_2$
	$::=$	$\varphi_1 EU \varphi_2 \mid \varphi_1 AU \varphi_2$
Propositional part		
π	$::=$	$P \mid \neg\pi \mid \pi_1 \vee \pi_2$
	where P is a labelling predicate.	

Fig. 4. CTL-live

The Kripke structure $\mathcal{K}$ satisfies $EF\ P$ iff the set of initial states ($I_{\mathcal{K}}$) is a subset of the states that satisfy $EF\ P$. Since $[EF\ P]_{\mathcal{K}}$ is the *smallest* among the $\lceil EF\ P\rceil^{\mathcal{K}'}$ for **every** $\mathcal{K}' \in CK(\mathfrak{A})$, checking whether $I_{\mathcal{K}}$ is a subset of $[EF\ P]_{\mathcal{K}}$ is equivalent to checking whether $I_{\mathcal{K}}$ is a subset of $\lceil EF\ P\rceil^{\mathcal{K}'}$ for **every** $\mathcal{K}' \in CK(\mathfrak{A})$:

$$I_{\mathcal{K}} \subseteq [EF\ P]_{\mathcal{K}} \iff \forall \mathcal{K}' \in CK(\mathfrak{A}) : I_{\mathcal{K}} \subseteq \lceil EF\ P\rceil^{\mathcal{K}'},$$

and since the formulae of Equation 5 do not have any effect on the set of initial states, $I_{\mathcal{K}} = I_{\mathcal{K}'}$ for every $\mathcal{K}' \in CK(\mathfrak{A})$; therefore, we have:

$$I_{\mathcal{K}} \subseteq [EF\ P]_{\mathcal{K}} \iff \forall \mathcal{K}' \in CK(\mathfrak{A}) : I_{\mathcal{K}'} \subseteq \lceil EF\ P\rceil^{\mathcal{K}'} \quad (6)$$

The universal quantifier in Equation 6 is over interpretations, which is not available in FOL, but, it is implicitly used in the definition of semantic entailment. Recall that $\Gamma \models \Phi$ iff every satisfying interpretation of Γ satisfies Φ. Since $CK(\mathfrak{A})$ is the set of all the satisfying interpretations of $\mathfrak{A}$, we can conclude the following:

$$\begin{aligned} I_{\mathcal{K}} \subseteq [EF\ P]_{\mathcal{K}} &\iff \forall \mathcal{K}' \in CK(\mathfrak{A}) : I_{\mathcal{K}'} \subseteq \lceil EF\ P\rceil^{\mathcal{K}'} \\ &\iff \mathfrak{A} \models \forall s : I(s) \to \lceil EF\ P\rceil(s) \end{aligned}$$

Therefore, we reduced model checking of EF to semantic entailment in FOL.

What we have shown here is that even though the constraints in Equation 5 do not precisely express the set of states that satisfy $EF\ P$, they can be used to express a set that **includes** every state that satisfies $EF\ P$ (and possibly more). Since in model checking, it is important to see whether the set of initial states is **included** in the set of states that satisfy $EF\ P$, those constraints along with the definition of entailment in FOL, which implicitly uses a universal quantifier over interpretations, can be used to express the model checking problem for the CTL connective EF.

The key idea behind this result is that the CTL connective EF can be expressed as the *smallest* set that satisfies some FOL formulae. We can generalize this result for other CTL connectives that have the same property: AF, EU, and AU. In the mu-calculus, the semantics of these connectives are expressed by means of the least fixed-point operator.

4.2 Universal Model Checking

In this subsection, we present the fragment of CTL that its model checking problem can be reduced to semantic entailment in FOL. Figure 4 presents the fragment of CTL that can be model checked directly using an FOL reasoner[2]. We call this fragment *CTL-live*, since it contains the CTL connectives that are usually used to express liveness properties. CTL-live's grammar has two parts: temporal and propositional. CTL-live disallows a temporal connective to be in the scope of negation ($\neg$); e.g., the CTL formula $\neg(P\ AU\ Q)$ is not allowed, but $((\neg P) AU\ Q)$ is allowed.

To model check a declarative model $\mathfrak{D}$, and a CTL-live formula φ, we use functions called `theory` and `axiom`, shown in Figure 5, to create a declarative model that is an enriched version of $\mathfrak{D}$. The function `theory` recurses through the structure of φ. For each logical connective of CTL-live, the constraints that are added to $\mathfrak{D}$ by `theory` are defined by the (non-recursive) function `axiom`. For every sub-formula φ' of φ, `axiom` creates one or two FOL formulae for each new labelling predicate, $\lceil\varphi'\rceil$, which are added to $\mathfrak{D}$. The complexity of `theory` is linear with respect to the size of φ.

Recall that a declarative model is a set of constraints that models Kripke structures and its satisfying interpretations form a class of Kripke structures (Definition 6 and Equation 4). For a declarative model $\mathfrak{D}$, every $\mathcal{K} \in CK(\mathfrak{D})$ is both a Kripke structure and an interpretation. In essence, $[\varphi]_{\mathcal{K}}$ and $\lceil\varphi\rceil^{\mathcal{K}}$ are both sets of states; the content of $[\varphi]_{\mathcal{K}}$ is determined by the semantics of CTL and considering $\mathcal{K}$ as a Kripke structure, whereas, the content of $\lceil\varphi\rceil^{\mathcal{K}}$ is determined by the semantics of FOL and considering $\mathcal{K}$ as a satisfying interpretation of $\mathfrak{D}$. In the following, we explore some properties of a declarative model generated by the function `theory`.

First, we study the relationship between the class of Kripke structures defined by the declarative model $\mathfrak{D}$ and `theory(`$\mathfrak{D},\varphi$`)`. The declarative model `theory(`$\mathfrak{D},\varphi$`)` contains a labelling predicate and some constraints for every sub-formula of φ; as a result, every Kripke structure in $CK($`theory(`$\mathfrak{D},\varphi$`)`$)$ can be converted to a Kripke structure in $CK(\mathfrak{D})$ by simply dropping the extra labelling predicates that the function `theory` adds to $\mathfrak{D}$. This property is formalized in Lemma 1:

Lemma 1: Let $\mathfrak{D}$ be a declarative model and φ a CTL-live formula; for every $\mathcal{K}$ in $CK($`theory(`$\mathfrak{D},\varphi$`)`$)$ there exists a $\mathcal{K}'$ in $CK(\mathfrak{D})$ that is a substructure of $\mathcal{K}$, i.e., $\mathcal{K}' \sqsubseteq \mathcal{K}$.

Proof: By the definition of `theory`. □

The second property that we investigate is the relationship between the set of states that satisfy a CTL-live formula φ, $[\varphi]$, and the set of states defined by the labelling predicate $\lceil\varphi\rceil$. If a CTL-live formula π is derived from only the propositional part of Figure 4, the sets $[\pi]_{\mathcal{K}}$ and $\lceil\pi\rceil^{\mathcal{K}}$ for every $\mathcal{K} \in CK($`theory(`$\mathfrak{D},\varphi$`)`$)$ are equal. This is due to the fact that the constraints that are defined by `axiom` for these connectives are necessary

2. Note that $(EF\varphi)$ and $(AF\varphi)$ are equivalent to $(\top EU\varphi)$ and $(\top AU\varphi)$ respectively.

```
theory(𝔇,$\varphi$):
  case $\varphi$ of
  1) $P$          ->  $\mathfrak{D}$
  2) $\Box\psi$   ->  let $\langle\langle F,R\rangle,\Gamma\rangle$=theory(𝔇,$\psi$) in $\langle\langle F,R\cup\{\lceil\varphi\rceil/1\}\rangle,\ \Gamma\ \cup$ axiom($\varphi$)$\rangle$
  3) $\psi_1\Diamond\psi_2$ ->  let $\langle\langle F,R_1\rangle,\Gamma_1\rangle$=theory(𝔇,$\psi_1$) and
                   $\langle\langle F,R_2\rangle,\Gamma_2\rangle$=theory(𝔇,$\psi_2$) in $\langle\langle F,R_1\cup R_2\cup\{\lceil\varphi\rceil/1\}\rangle,\ \Gamma_1\ \cup\ \Gamma_2\ \cup$ axiom($\varphi$)$\rangle$

axiom($\varphi$):
   case $\varphi$ of
 1) $P$                  ->  $\{\ \forall s: P(s)\leftrightarrow\lceil P\rceil(s)\ \}$            where $P$ is a labelling predicate
 2) $\neg\psi$           ->  $\{\ \forall s:\ \lceil\varphi\rceil(s)\leftrightarrow\neg\lceil\psi\rceil(s)\ \}$
 3) $\psi_1\vee\psi_2$   ->  $\{\ \forall s:\ \lceil\varphi\rceil(s)\leftrightarrow\lceil\psi_1\rceil(s)\vee\lceil\psi_2\rceil(s)\ \}$
 4) $\psi_1\wedge\psi_2$ ->  $\{\ \forall s:\ \lceil\varphi\rceil(s)\leftrightarrow\lceil\psi_1\rceil(s)\wedge\lceil\psi_2\rceil(s)\ \}$
 5) $EX\psi$             ->  $\{\ \forall s:\ \lceil\varphi\rceil(s)\leftrightarrow\big(\exists s':\ N(s,s')\wedge\lceil\psi\rceil(s')\big)\ \}$
 6) $AX\psi$             ->  $\{\ \forall s:\ \lceil\varphi\rceil(s)\leftrightarrow\big(\forall s':\ N(s,s')\rightarrow\lceil\psi\rceil(s')\big)\ \}$
 7) $\psi_1 EU\psi_2$    ->  $\{\ \forall s:\lceil\psi_2\rceil(s)\rightarrow\lceil\varphi\rceil(s)\ ,\ \ \forall s:\lceil\psi_1\rceil(s)\wedge\big(\exists s':N(s,s')\wedge\lceil\varphi\rceil(s')\big)\rightarrow\lceil\varphi\rceil(s)\ \}$
 8) $\psi_1 AU\psi_2$    ->  $\{\ \forall s:\lceil\psi_2\rceil(s)\rightarrow\lceil\varphi\rceil(s)\ ,\ \ \forall s:\lceil\psi_1\rceil(s)\wedge\big(\forall s':N(s,s')\rightarrow\lceil\varphi\rceil(s')\big)\rightarrow\lceil\varphi\rceil(s)\ \}$
```

Fig. 5. The definition of `theory` and `axiom`. $\Box\in\{\neg,EX,AX\}$, $\Diamond\in\{\vee,\wedge,EU,AU\}$ and φ is a CTL-live formula.

and *sufficient* to characterize the set of states that satisfy π:

Lemma 2: Let $\mathfrak{D}$ be a declarative model and π a CTL-live formula that is derived from only the propositional part of Figure 4; we have:

$$\forall\ \mathcal{K}\in CK\big(\texttt{theory}\,(\mathfrak{D},\varphi)\big):\ [\pi]_{\mathcal{K}}=\lceil\pi\rceil^{\mathcal{K}}$$

Proof: Proof by structural induction on π. In the following cases, we assume $\mathcal{K}\in CK\big(\texttt{theory}\,(\mathfrak{D},\varphi)\big)$:

- **Base case:** suppose $\pi=P$. By the semantics of CTL and the definition of `axiom` at Line 1, for every state s we have:

  $$s\in[P]_{\mathcal{K}}\iff P_{\mathcal{K}}(s)\iff s\in\lceil P\rceil^{\mathcal{K}}$$

  therefore, $[P]_{\mathcal{K}}=\lceil P\rceil^{\mathcal{K}}$.
- **Induction step:** according to the structure of π, two cases are distinguished having $[\pi_1]_{\mathcal{K}}=\lceil\pi_1\rceil^{\mathcal{K}}$ and $[\pi_2]_{\mathcal{K}}=\lceil\pi_2\rceil^{\mathcal{K}}$ as induction hypotheses:
  1) suppose $\pi=\neg\pi_1$. By the semantics of CTL, and the induction hypotheses, for every state s we have:

     $$s\in[\neg\pi_1]_{\mathcal{K}}\iff s\notin[\pi_1]_{\mathcal{K}}\iff s\notin\lceil\pi_1\rceil^{\mathcal{K}}$$

     and by the definition of `axiom` at Line 2,

     $$s\notin\lceil\pi_1\rceil^{\mathcal{K}}\iff s\in\lceil\neg\pi_1\rceil^{\mathcal{K}}$$

     therefore, $[\neg\pi_1]_{\mathcal{K}}=\lceil\neg\pi_1\rceil^{\mathcal{K}}$.
  2) suppose $\pi=\pi_1\vee\pi_2$. By the semantics of CTL, for every state s we have:

     $$s\in[\pi_1\vee\pi_2]_{\mathcal{K}}\iff s\in[\pi_1]_{\mathcal{K}}\vee s\in[\pi_2]_{\mathcal{K}}$$

     by the induction hypotheses,

     $$s\in[\pi_1]_{\mathcal{K}}\vee s\in[\pi_2]_{\mathcal{K}}\iff s\in\lceil\pi_1\rceil^{\mathcal{K}}\vee s\in\lceil\pi_2\rceil^{\mathcal{K}}$$

     and by the definition of `axiom` at Line 3,

     $$s\in\lceil\pi_1\rceil^{\mathcal{K}}\vee s\in\lceil\pi_2\rceil^{\mathcal{K}}\iff s\in\lceil\pi_1\vee\pi_2\rceil^{\mathcal{K}}$$

     therefore, $[\pi_1\vee\pi_2]_{\mathcal{K}}=\lceil\pi_1\vee\pi_2\rceil^{\mathcal{K}}$.

□

A similar result to Lemma 2 can be proven for the CTL-live formulae that are derived from the temporal part of Figure 4. The difference is that the set $[\varphi]_{\mathcal{K}}$ is a subset of $\lceil\varphi\rceil^{\mathcal{K}}$ rather than being equal to it. The reason is that the constraints that are added to $\mathfrak{D}$ by `theory` do not completely characterize $[\varphi]_{\mathcal{K}}$: these constraints are necessary but they are not sufficient; as a result, the set $\lceil\varphi\rceil^{\mathcal{K}}$ includes $[\varphi]_{\mathcal{K}}$ and possibly some other states.

Lemma 3: Let $\mathfrak{D}$ be a declarative model and φ a CTL-live. We have:

$$\forall\ \mathcal{K}\in CK\big(\texttt{theory}\,(\mathfrak{D},\varphi)\big):\ [\varphi]_{\mathcal{K}}\subseteq\lceil\varphi\rceil^{\mathcal{K}}$$

Proof: Proof by structural induction on φ. In the following cases, we assume $\mathcal{K}\in CK\big(\texttt{theory}\,(\mathfrak{D},P)\big)$:

- **Base case:** suppose $\varphi=\pi$, where π is derived from the propositional part of Figure 4. According to Lemma 2, $[\pi]_{\mathcal{K}}=\lceil\pi\rceil^{\mathcal{K}}$; as a result, $[\pi]_{\mathcal{K}}\subseteq\lceil\pi\rceil^{\mathcal{K}}$.
- **Induction step:** according to the structure of φ, six cases are distinguished having $[\psi_1]_{\mathcal{K}}\subseteq\lceil\psi_1\rceil^{\mathcal{K}}$ and $[\psi_2]_{\mathcal{K}}\subseteq\lceil\psi_2\rceil^{\mathcal{K}}$ as induction hypotheses:
  1) suppose $\varphi=\psi_1\vee\psi_2$. The proof of this case is similar to Part 2 in the induction step of Lemma 2.
  2) suppose $\varphi=\psi_1\wedge\psi_2$. The proof of this case is similar to Part 2 in the induction step of Lemma 2.
  3) suppose $\varphi=EX\psi$. By the semantics of CTL for every s we have:

     $$s\in[EX\psi]_{\mathcal{K}}\Longrightarrow\exists s':N_{\mathcal{K}}(s,s')\wedge s'\in[\psi]_{\mathcal{K}}$$

     by the induction hypotheses,
     $$\exists s':N_{\mathcal{K}}(s,s')\wedge s'\in[\psi]_{\mathcal{K}}\Longrightarrow\exists s':N_{\mathcal{K}}(s,s')\wedge s'\in\lceil\psi\rceil^{\mathcal{K}}$$
     and by the definition of `axiom` at Line 5,

     $$\exists s':N_{\mathcal{K}}(s,s')\wedge s'\in\lceil\psi\rceil^{\mathcal{K}}\Longrightarrow s\in\lceil EX\psi\rceil^{\mathcal{K}}$$

therefore, $[EX\psi]_\mathcal{K} \subseteq \lceil EX\psi \rceil^\mathcal{K}$.

4) suppose $\varphi = AX\psi$. The proof of this part is similar to Part 3.
5) suppose $\varphi = \psi_1 EU \psi_2$. If $s \in [\psi_1 EU \psi_2]_\mathcal{K}$, then, with respect to the semantics of CTL, there exists a j and a path, $s_0 \mapsto \dots s_j \mapsto \dots$, such that $s_0 = s$, $s_j \in [\psi_2]_\mathcal{K}$, and for all $i < j$: $s_i \in [\psi_1]_\mathcal{K}$. This means that the state s can reach another state s_j in j number of steps, where s_j satisfies ψ_2; we use induction on j, the least number of steps required to get to a state that satisfies ψ_2, to prove that $s \in \lceil \psi_1 EU \psi_2 \rceil^\mathcal{K}$ holds. Suppose $s \in [\psi_1 EU \psi_2]_\mathcal{K}$:
   - **Base case:** suppose $j = 0$; in this case, the state s itself satisfies ψ_2, which means $s \in [\psi_2]_\mathcal{K}$. Using the induction hypotheses from the outer induction, we have $s \in \lceil \psi_2 \rceil^\mathcal{K}$, and according to the first constraint in the definition of `axiom` at Line 7, we have $s \in \lceil \psi_1 EU \psi_2 \rceil^\mathcal{K}$; therefore, if $j = 0$ then $s \in \lceil \psi_1 EU \psi_2 \rceil^\mathcal{K}$
   - **Induction step:** suppose $j = m + 1$. The induction hypotheses for this inner induction is: if $s' \in [\psi_1 EU \psi_2]_\mathcal{K}$ and s' can reach a state satisfying ψ_2 in less than or equal to m steps, then $s' \in \lceil \psi_1 EU \psi_2 \rceil^\mathcal{K}$. Since $j \neq 0$ for s and j is the least number of steps required to get to a state that satisfies ψ_2, state s does not satisfy ψ_2 itself; as a result, there exists a state next to s, s', that satisfies $\psi_1 EU \psi_2$ and j for that state is less than or equal to m:

     $$\exists s' : N_\mathcal{K}(s, s') \wedge s' \in [\psi_1 EU \psi_2]_\mathcal{K}$$

     according to the induction hypotheses of the inner induction, we have:

     $$\exists s' : N_\mathcal{K}(s, s') \wedge s' \in \lceil \psi_1 EU \psi_2 \rceil^\mathcal{K}$$

     Since $s \in [\psi_1 EU \psi_2]_\mathcal{K}$, and $s \notin [\psi_2]$, from the semantics of CTL, we can conclude that $s \in [\psi_1]$. Using the induction hypotheses of the outer induction ($[\psi_1]_\mathcal{K} \subseteq \lceil \psi_1 \rceil^\mathcal{K}$), we derive $s \in \lceil \psi_1 \rceil^\mathcal{K}$. According to the second constraint in the definition of `axiom` at Line 7, and the above property, we have: $s \in \lceil \psi_1 EU \psi_2 \rceil^\mathcal{K}$; therefore if $j = m + 1$ then $s \in \lceil \psi_1 EU \psi_2 \rceil^\mathcal{K}$.

   By putting the base case and the induction step together, we have:

   $$[\psi_1 EU \psi_2]_\mathcal{K} \subseteq \lceil \psi_1 EU \psi_2 \rceil^\mathcal{K}$$

6) suppose $\varphi = \psi_1 AU \psi_2$. The proof of this part is similar to Part 5.

□

Another way of proving Parts 5 and 6 in the induction step of Lemma 3 is to consider that the set of states satisfying $\psi_1 EU \psi_2$ and $\psi_1 AU \psi_2$ is the smallest set that satisfies the constraints in Line 7 and 8 of `axiom` respectively.

The next lemma relates every Kripke structure in $CK(\mathfrak{D})$ to a Kripke structure in $CK\big(\texttt{theory}(\mathfrak{D},\varphi)\big)$:

Lemma 4: Let $\mathfrak{D}$ be a declarative model and φ a CTL-live formula. For every $\mathcal{K} \in CK(\mathfrak{D})$ there exists $\mathcal{K}' \in CK\big(\texttt{theory}(\mathfrak{D},\varphi)\big)$ such that:

$$I_\mathcal{K} = I_{\mathcal{K}'} \wedge [\varphi]_\mathcal{K} = \lceil \varphi \rceil^{\mathcal{K}'}$$

Proof: Suppose $\mathcal{K} \in CK(\mathfrak{D})$. Let $\mathcal{K}'$ be an interpretation with the same domain as $\mathcal{K}$. For each symbol in the base of $\mathfrak{D}$, $\mathcal{K}'$ has the same value as $\mathcal{K}$, and for every sub-formula φ' of φ (including φ) $\mathcal{K}'$ assigns $[\varphi']_\mathcal{K}$ to $\lceil \varphi' \rceil$, thus $[\varphi']_\mathcal{K} = \lceil \varphi' \rceil^{\mathcal{K}'}$. According to the semantics of CTL, the constraints that are added for each sub-formula φ' of φ by `theory` to $\mathfrak{D}$, are necessary constraints that the sets $[\varphi']_\mathcal{K}$ satisfy; therefore, $\mathcal{K}'$ is a satisfying interpretation of $CK\big(\texttt{theory}(\mathfrak{D},\varphi)\big)$:

$$\mathcal{K}' \in CK\big(\texttt{theory}(\mathfrak{D},\varphi)\big)$$

□

In the next theorem, we present our first contribution by putting together all the properties we have proven: universal model checking of a CTL-live formula is reducible to semantic entailment checking in FOL:

Theorem 2: Let $\mathfrak{D}$ be a declarative model and φ a CTL-live formula; for universal model checking:

$$\mathfrak{D} \models_\forall \varphi \iff \texttt{theory}(\mathfrak{D},\varphi) \models \forall s : I(s) \rightarrow \lceil \varphi \rceil(s)$$

Proof: Recall that the set of satisfying interpretations of $\mathfrak{D}$ is $CK(\mathfrak{D})$ (Definition 6). We need to prove two statements: 1) $\mathfrak{D} \models_\forall \varphi$ implies $\texttt{theory}(\mathfrak{D},\varphi) \models \forall s : I(s) \rightarrow \lceil \varphi \rceil(s)$, and 2) $\mathfrak{D} \not\models_\forall \varphi$ implies $\texttt{theory}(\mathfrak{D},\varphi) \not\models \forall s : I(s) \rightarrow \lceil \varphi \rceil(s)$.

- **Case (1)** assume $\mathfrak{D} \models_\forall \varphi$. We need to show that every satisfying interpretation of $\texttt{theory}(\mathfrak{D},\varphi)$ also satisfies $\forall s : I(s) \rightarrow \lceil \varphi \rceil(s)$:

  $$\forall \mathcal{K} \in CK\big(\texttt{theory}(\mathfrak{D},\varphi)\big) : I_\mathcal{K} \subseteq \lceil \varphi \rceil^\mathcal{K}$$

  Using Lemma 1, for every $\mathcal{K} \in CK\big(\texttt{theory}(\mathfrak{D},\varphi)\big)$, there exists a $\mathcal{K}' \in CK(\mathfrak{D})$ such that it is a substructure of $\mathcal{K}$. Since $\mathcal{K}' \in CK(\mathfrak{D})$ and $\mathfrak{D} \models_\forall \varphi$, we have:

  $$\mathcal{K}' \models_c \varphi$$

  By Theorem 1,

  $$\mathcal{K}' \models_c \varphi \Longrightarrow \mathcal{K} \models_c \varphi$$

  and by the semantics of CTL,

  $$\mathcal{K} \models_c \varphi \Longrightarrow I_\mathcal{K} \subseteq [\varphi]_\mathcal{K}$$

  and by Lemma 3,

  $$I_\mathcal{K} \subseteq [\varphi]_\mathcal{K} \Longrightarrow I_\mathcal{K} \subseteq \lceil \varphi \rceil^\mathcal{K}$$

therefore,

$$\forall \mathcal{K} \in CK\big(\texttt{theory}\,(\mathfrak{D},\varphi)\big) : I_{\mathcal{K}} \subseteq \lceil\varphi\rceil^{\mathcal{K}}$$

- **Case (2)** assume $\mathfrak{D} \not\models_{\forall} \varphi$. We need to show that there exists an interpretation that satisfies $\texttt{theory}\,(\mathfrak{D},\varphi)$, but it does not satisfy $\forall s : I(s) \rightarrow \lceil\varphi\rceil(s)$:

  $$\exists \mathcal{K} \in CK\big(\texttt{theory}\,(\mathfrak{D},\varphi)\big) : I_{\mathcal{K}} \not\subseteq \lceil\varphi\rceil^{\mathcal{K}}$$

  Since $\mathfrak{D} \not\models_{\forall} \varphi$, there exists a Kripke structure $\mathcal{K} \in CK(\mathfrak{D})$ that does not satisfy φ:

  $$\mathcal{K} \in CK(\mathfrak{D}) \wedge I_{\mathcal{K}} \not\subseteq [\varphi]_{\mathcal{K}}$$

  Since $\mathcal{K} \in CK(\mathfrak{D})$, according to Lemma 4, there exists a Kripke structure $\mathcal{K}' \in CK\big(\texttt{theory}\,(\mathfrak{D},\varphi)\big)$ that has the following property:

  $$I_{\mathcal{K}} = I_{\mathcal{K}'} \wedge [\varphi]_{\mathcal{K}} = \lceil\varphi\rceil^{\mathcal{K}'}$$

  Using this property and $I_{\mathcal{K}} \not\subseteq [\varphi]_{\mathcal{K}}$, we have:

  $$I_{\mathcal{K}'} \not\subseteq \lceil\varphi\rceil^{\mathcal{K}'}$$

  therefore,

  $$\exists \mathcal{K} \in CK\big(\texttt{theory}\,(\mathfrak{D},\varphi)\big) : I_{\mathcal{K}} \not\subseteq \lceil\varphi\rceil^{\mathcal{K}}$$

□

4.3 Existential Model Checking

To solve the existential model checking problem using FOL reasoning, we study the relationship between existential and universal model checking, so that we can use our result from Theorem 2. If a Kripke structure $\mathcal{K}$ satisfies a CTL formula $\neg\varphi$, we can conclude that $\mathcal{K}$ does not satisfy φ. However, the opposite is not true unless the Kripke structure has only one initial state.

Lemma 5: Let $\mathcal{K}$ and φ be a Kripke structure and a CTL formula respectively; we have:

$$\mathcal{K} \models_c \neg\varphi \quad\Longrightarrow\quad \mathcal{K} \not\models_c \varphi$$

Proof: By the definition of satisfiability in CTL. □

Theorem 3: Let $\mathfrak{D}$ and φ be a declarative model and a CTL-formula respectively, we have:

$$\mathfrak{D} \models_{\exists} \neg\varphi \quad\Longrightarrow\quad \mathfrak{D} \not\models_{\forall} \varphi$$

Proof: By the semantics of CTL, Definitions 7 and 8, and Lemma 5. □

Lemma 5 is not an "iff" property since the semantics of CTL does not imply that a Kripke structure has to satisfy a CTL formula or its negation. Suppose φ is a CTL formula and $\mathcal{K}$ a Kripke structure that has two initial states s_0 and s_0' where s_0 satisfies φ and s_0' does not. It is easy to see that $\mathcal{K}$ neither satisfies φ nor $\neg\varphi$; as a result, in general, the other direction of Lemma 5 does not hold. By not allowing a Kripke structure to have more than one initial state, we can prove the other direction of Lemma 5:

Lemma 6: Let $\mathcal{K}$ be a Kripke structure that has only one initial state, $|I_{\mathcal{K}}| = 1$, and φ a CTL formula; we have:

$$\mathcal{K} \not\models_c \neg\varphi \quad\Longleftrightarrow\quad \mathcal{K} \models_c \varphi$$

Proof: By using the semantics of CTL. □

By using the result of Lemma 6, we can prove the following corollary:

Corollary 1: Let $\mathfrak{D}$ be a declarative model such that every Kripke structure $\mathcal{K} \in CK(\mathfrak{D})$ has exactly one initial state, $|I_{\mathcal{K}}| = 1$ and φ a CTL-live formula; we have:

$$\mathfrak{D} \models_{\exists} \neg\varphi \quad\Longleftrightarrow\quad \mathfrak{D} \not\models_{\forall} \varphi \quad\Longleftrightarrow\quad \texttt{theory}\,(\mathfrak{D},\varphi) \not\models \varphi$$

5 Maximality of CTL-live

Theorem 2 shows that model checking CTL-live is reducible to semantic entailment in FOL. The logical connectives that are not included in CTL-live are EG, AG, and $\neg$ over temporal connectives. To show model checking of these three connectives is not reducible to FOL entailment, we reduce the complement of the halting problem on an empty tape for a deterministic Turing machines (DTM) to universal model checking of EG and AG. The complement of the halting problem is not recursively enumerable but FOL semantic entailment is; therefore, universal model checking of EG and AG cannot be reduced to FOL entailment checking, otherwise, this problem would be recursively enumerable. We call this result the maximality of CTL-live.

In the following, we assume a DTM $\mathcal{M} = \langle \mathcal{Q}, \delta\rangle$ is a pair where $\mathcal{Q} = \{q_0, \ldots, q_n\}$ is a finite set of states, the tape alphabet is $\{b, 0\}$ and δ, the transition function, is a total function from $\mathcal{Q} \times \{b, 0\}$ to $\mathcal{Q} \times \{b, 0\} \times \{L, R\}$. For example, the transition $\delta(q_9, 0) = (q_2, b, L)$ means that if the DTM $\mathcal{M}$ is at state q_9 and the tape head is scanning 0, in the next step, $\mathcal{M}$ goes to the state q_2, writes b on the tape and moves the head to the Left.

A DTM $\mathcal{M} = \langle \mathcal{Q}, \delta\rangle$ starts in the state q_0. We consider $\mathcal{M}$ to have halted if it reaches state q_n. The tape is one way infinite. In the initial state, the head tape is on the left most square of the tape, and every square on the tape is blank (b).

The idea behind reducing the complement of the halting problem on an empty tape for a DTM to universal model checking of EG or AG is that the set of all the configurations of a DTM can be considered as the state space for a Kripke structure and the transition relation of this Kripke structure can be derived from the transition function of the DTM. Since the underlying DTM is deterministic, this Kripke structure has only one computation path, and therefore, satisfying EG and AG would be equivalent.The Globaly part of EG and AG can be used to state that no state along the path is a halting state.

Lemma 7: Let $\mathcal{M} = \langle \mathcal{Q}, \delta\rangle$ be a DTM; the complement of the halting problem on an empty tape for $\mathcal{M}$ is reducible to universal model checking of an EG formula.

Proof: To prove this lemma, we create a declarative model $\mathfrak{D}$ from $\mathcal{M}$ such that $\mathfrak{D}$ universally satisfies an

EG formula iff $\mathcal{M}$ does not halt on an empty tape. To encode $\mathcal{M}$ as a declarative model $\mathfrak{D} = \langle B, \Gamma \rangle$, we use the following base $B = \langle F, R \rangle$:

- $F = \{0,\ inc/1,\ dec/1,\ Q/1,\ H/1\}$,
- $R = \{b/2,\ I/1,\ N/2,\ halt/1\}$

The constant 0 represents number zero. The functional symbols $inc/1$ and $dec/1$ are used to model increment and decrement operations on natural numbers. We can refer to a certain natural number by applying inc to 0; e.g., number 2 is represented as $inc(inc(0))$, for short $inc^2(0)$. In this lemma and the following, natural numbers are short forms of their representations using this base; e.g., in the formula $Q(t) = 2$, the symbol 2 is a short form of $inc^2(0)$.

The natural numbers are used to represent configurations of $\mathcal{M}$: the position of the tape head, the current state of $\mathcal{M}$, and to point to different squares of the tape. The binary relation symbol $b(t, i)$ represents whether at step t the i^{th} square is blank or 0. The functional symbol $Q(t) = i$ represents that the state of $\mathcal{M}$ at step t is q_i, and the functional symbol $H(t) = i$ represents that the tape head of $\mathcal{M}$ at step t is on the i^{th} square. The relational symbols I and N are used to model the Kripke structures, and $halt$ is a relational symbol to represent whether a state is a halting state.

In the declarative model $\mathfrak{D} = \langle B, \Gamma \rangle$, the constraints in Γ consist of 5 parts:

1) Constraints for encoding an "acceptable" semantics for 0, inc, and dec:
   - $\forall i : inc(i) \neq 0$
   - $\forall i, i' : inc(i) = inc(i') \to i = i'$
   - $\forall i : i \neq 0 \to (\exists i' : inc(i') = i)$
   - $dec(0) = 0$
   - $\forall i : dec(inc(i)) = i$
   - $\forall i : i \neq 0 \to inc(dec(i)) = i$
2) Constraint stating that at each step of computation at most one position of the tape can be changed:
$$\forall t, i : H(t) \neq i \to \big(b(t, i) \leftrightarrow b(inc(t), i)\big)$$
3) Constraints for encoding the initial configuration of $\mathcal{M}$:
   - $Q(0) = 0$: at step 0, $\mathcal{M}$ is at state q_0,
   - $H(0) = 0$: at step 0, the tape head of $\mathcal{M}$ is at position 0,
   - $\forall i : b(0, i)$: at step 0, every position of the tape is blank.
4) Constraints for encoding the transition function δ: for every pair of $\mathcal{Q} \times \{b, 0\}$ we have a formula that mimics the computation of $\mathcal{M}$. For example, the formula that simulates $\delta(q_9, 0) = (q_2, b, L)$ is
   $\forall t, i : Q(t) = 9 \wedge \neg b(t, i) \wedge H(t) = i \to$
   $\quad Q(inc(t)) = 2 \wedge b(inc(t), i) \wedge H(inc(t)) = dec(i)$
   and the formula that simulates $\delta(q_6, b) = (q_7, 0, R)$ is
   $\forall t, i : Q(t) = 6 \wedge b(t, i) \wedge H(t) = i \to$
   $\quad Q(inc(t)) = 7 \wedge \neg b(inc(t), i) \wedge H(inc(t)) = inc(i)$
5) Constraints for the initial state, transition relation, and halting state of the corresponding Kripke structure. We use natural numbers as the state space of a Kripke structure. The configuration of $\mathcal{M}$ at state (step) t is represented by $Q(t)$, $H(t)$, and $b(t, .)$:
   - $\forall t : I(t) \leftrightarrow t = 0$: initial state,
   - $\forall t, t' : N(t, t') \leftrightarrow t' = inc(t)$: transition relation,
   - $\forall t : halt(t) \leftrightarrow Q(t) = n$: halting states.

We claim that the following holds:

$$\mathfrak{D} \models_{\forall} EG\neg halt \iff \mathcal{M} \text{ does not halt on an empty tape.}$$

We need to prove two statements: 1) $\mathfrak{D} \models_{\forall} EG\neg halt$ implies $\mathcal{M}$ does not halt on an empty tape, and 2) $\mathcal{M}$ does not halt on an empty tape implies $\mathfrak{D} \models_{\forall} EG\neg halt$.

- **Case (1)** $\mathfrak{D} \models_{\forall} EG\neg halt$ means that every Kripke structure in $CK(\mathfrak{D})$ satisfies $EG\neg halt$. The standard interpretation of natural numbers, which satisfies $\mathfrak{D}$ corresponds to the computation of $\mathcal{M}$. Since $EG\neg halt$ means there exists a path along which halt is never true, and the DTM $\mathcal{M}$ is deterministic, and therefore has only one path, we can conclude that $\mathcal{M}$ does not halt on an empty tape.
- **Case (2)** By induction on the number of steps, we can prove that if at step t, $\mathcal{M}$ is at state q_i, every Kripke structure in $CK(\mathfrak{D})$ satisfies $Q(t) = i$. Assuming $\mathcal{M}$ does not halt on an empty tape, we can conclude that every Kripke structure in $CK(\mathfrak{D})$ has an infinite path starting at 0:
$$0 \mapsto 1 \mapsto 2 \mapsto 3 \mapsto ..$$
where none of them is the halting state q_n:
$$Q(0) \neq n,\ Q(1) \neq n,\ Q(2) \neq n,\ Q(3) \neq n, ..$$
Therefore, every Kripke structure in $CK(\mathfrak{D})$ satisfies $EG\neg halt$:
$$\mathfrak{D} \models_{\forall} EG\neg halt$$

□

Lemma 8: Let $\mathcal{M} = \langle \mathcal{Q}, \delta \rangle$ be a DTM; the complement of the halting problem on an empty tape for $\mathcal{M}$ is reducible to universal model checking of an AG formula.

Proof: To prove this lemma, we create a declarative model $\mathfrak{D}$ from $\mathcal{M}$ such that $\mathfrak{D}$ universally satisfies an AG formula iff $\mathcal{M}$ does not halt on an empty tape. The declarative model that we need to build for this reduction is same as the one in Lemma 7. We claim that the following holds:

$$\mathfrak{D} \models_{\forall} AG\neg halt \iff \mathcal{M} \text{ does not halt on an empty tape.}$$

We need to prove two statements: 1) $\mathfrak{D} \models_{\forall} AG\neg halt$ implies $\mathcal{M}$ does not halt on an empty tape, and 2) $\mathfrak{D} \not\models_{\forall} AG\neg halt$ implies $\mathcal{M}$ halts on an empty tape.

- **Case (1)** similar to Case 1 of Lemma 7.
- **Case (2)** since $\mathfrak{D} \not\models_{\forall} AG\neg halt$, there exists a Kripke structure $K \in CK(\mathfrak{D})$ that does not satisfy $AG\neg halt$. This means that there is a finite path from an initial state of K that reaches a state satisfying $halt$. By

induction on the length of this path, we can show that this finite path corresponds to a finite sequence of configurations in $\mathcal{M}$ that results in a halting configuration; as a result, $\mathcal{M}$ halts on an empty tape. □

Theorem 4: **(Maximality of CTL-live)** The temporal part of CTL-live cannot be extended with EG, AG, or $\neg$ for universal model checking in FOL.

Proof: In Lemma 7 (8), we showed that the complement of the halting problem on an empty tape for a DTM is reducible to universal model checking of EG (AG). This problem is not recursively enumerable, as a result, it cannot be reduced to entailment checking in FOL, which is a recursively enumerable problem. We also know that $EG\varphi$ is equivalent to $\neg(\top AU \neg\varphi)$. Since AU is include in this fragment, $\neg$ cannot be added as well. □

6 Related Work

Immerman and Vardi show how the semantics of CTL and CTL* can be encoded in FOL plus the transitive-closure operator [13]. Based on their work, we showed how universal and existential model checking can be solved for a finite declarative model [11]. Since transitive-closure is not expressible in FOL, this encoding cannot be used with an FOL reasoner.

SAT and SMT solvers have been used for bounded model checking [3], [4]. These methods use a reasoner directly for model checking by expanding the transition relation for a finite number of steps.

K-induction is a technique for unbounded model checking of safety properties [5]. This technique extends bounded model checking by proving that bounded model checking for bound K is sufficient. The number K is dominated by the diameter of a Kripke structure. The diameter is computed iteratively using a SAT solver to check the equivalence of two formulae: the equivalence holds iff no new state can be reached by taking more than K steps. In [5], termination is guarantied due to finiteness of the Kripke structures under study.

Bultan, Gerber, and Pugh use Presburger formulae to represent infinite sets of states symbolically [6]. Their model checking approach requires a fix-point calculation, and termination is achieved by using conservative approximation. This approach allows false negatives.

The problem of model checking parametrized systems is related to our work. A parametrized system defines an infinite family of finite Kripke structures where each finite Kripke structure is determined by fixing a parameter. In most cases, this parameter is the number of processes that can execute. Model checking a parametrized system means checking whether all the Kripke structures in the family satisfy a temporal property. If a parametrized system is presented as a declarative model, then model checking a parametrized system is equivalent to universal model checking. In general, this problem is undecidable [14]. One approach to model checking parametrized systems is to achieve decidability by restricting the structure of a parametrized system; e.g., Emerson and Kahlon consider asynchronous systems consisting of an arbitrary number of homogeneous copies of a generic process template [15]. Some work restricts the topology of a network and communications between processes [16], [17]. Other approaches to model checking a parametrized system are based on abstraction, network invariants, and compositional model checking [18]–[21]. Our approach is more general in the sense that the set of Kripke structures have no restrictions on them as long as they are expressible in FOL, however, our results are only for a fragment of CTL, and FOL entailment checking is not decidable.

7 Conclusion

In this article, we presented a fragment of CTL, called CTL-live, whose model checking problem is reducible to semantic entailment in FOL. This reduction shows that FOL reasoning techniques are sufficient for model checking CTL-live formulae, without the need for transitive-closure, fixed-point operators, or induction. The key insight in our approach is to use the implicit higher-order quantifier in the definition of semantic entailment to require that all initial states of a Kripke structure are within all the sets of states that satisfy a representation of a CTL-live temporal operator, and thereby, reduce model checking to semantic entailment in FOL. Semantic entailment checking for FOL is recursively enumerable; as a result, this reduction allows one to generate automatically a proof in the case where a CTL-live formula is satisfied.

CTL-live has two parts: 1) propositional 2) temporal. The propositional part contains all propositional logic connectives. The temporal section includes all the connectives such that their encoding in the mu-calculus requires the *least* fixed-point operator. These connectives are usually used to express liveness properties. The temporal part also includes conjunction and disjunction since their corresponding set operators, intersection and union, are monotonic with respect to set inclusion relation.

The input to our model checking technique is a set of logical formulae, called a *declarative model*, where every satisfying interpretation is a Kripke structure; as a result, a declarative model can represent a class of Kripke structures. We studied two questions about such a class of Kripke structures: 1) *universal model checking*: do all the Kripke structures in the class satisfy a CTL formula? 2) *existential model checking*: is there at least one Kripke structure in the class that satisfies a CTL formula? We showed how our encoding of CTL-live in FOL can be used to solve universal model checking and how existential model checking can be reduced to universal model checking.

We proved that CTL-live is maximal in the sense that if any other CTL connective is added, non-FOL reasoning techniques would be required and the model checking problem becomes harder than a recursively enumerable problem. The connectives that cannot be model checked in FOL are the ones whose encoding in the mu-calculus requires the *greatest* fixed-point operator. An implicit result of our work is that some reachability queries can be answered by using an FOL reasoner even though reachability is not expressible in FOL.

The rapid improvements in the efficiency of fields such as SMT solvers, DL reasoners, FOL automated theorem proving, etc. have a direct effect on the applicability of our results. A practitioner who wants to use the theoretical result of this article must first check if the temporal property of interest is within CTL-live; then model checking can be accomplished using a FOL reasoner by itself. We are currently studying the use of SMT solvers and decidable fragments of FOL for model checking CTL-live formulae.

References

[1] E. Clarke, O. Grumberg, and D. A. Peled, *Model Checking*. MIT Press, 1999.

[2] C. Barrett, R. Sebastiani, S. Seshia, and C. Tinelli, *Satisfiability Modulo Theories*, ser. Frontiers in Artificial Intelligence and Applications. IOS Press, February 2009, vol. 185, ch. 26, pp. 825–885.

[3] A. Biere, A. Cimatti, E. Clarke, and Y. Zhu, “Symbolic Model Checking without BDDs,” in *TACAS*, ser. LNCS. Springer, 1999, vol. 1579, pp. 193–207.

[4] T. Schüle and K. Schneider, “Bounded model checking of infinite state systems,” *Formal Methods in System Design*, pp. 51–81, 2007.

[5] M. Sheeran, S. Singh, and G. Stålmarck, “Checking Safety Properties Using Induction and a SAT-Solver,” in *FMCAD*, ser. LNCS. Springer, 2000, vol. 1954, pp. 127–144.

[6] T. Bultan, R. Gerber, and W. Pugh, “Symbolic Model Checking of Infinite State Systems Using Presburger Arithmetic,” in *CAV*, ser. LNCS, O. Grumberg, Ed. Springer, 1997, vol. 1254, pp. 400–411.

[7] M. Huth and M. Ryan, *Logic in Computer Science, Modelling and Reasoning about Systems*, 2nd ed. Cambridge University Press, 2004.

[8] C. Gödel, “Über die Vollständigkeit des Logikkalküls,” Ph.D. dissertation, 1929, proof of completenss theorem for FOL.

[9] A. Church, “An Unsolvable Problem of Elementary Number Theory,” *American Journal of Mathematics*, pp. 345–363, 1936.

[10] E. M. Clarke, E. A. Emerson, and A. P. Sistla, “Automatic Verification of Finite-State Concurrent Systems Using Temporal Logic Specifications,” *ACM TOPLS*, pp. 244–263, 1986.

[11] A. Vakili and N. Day, “Temporal Logic Model Checking in Alloy,” in *ABZ*, ser. LNCS. Springer, 2012, vol. 7316, pp. 150–163.

[12] A. R. Bradley and Z. Manna, “Checking Safety by Inductive Generalization of Counterexamples to Induction,” in *FMCAD*, 2007, pp. 173–180.

[13] N. Immerman and M. Vardi, “Model Checking and Transitive-Closure Logic,” in *CAV*, ser. LNCS. Springer, 1997, vol. 1254, pp. 291–302.

[14] K. R. Apt and D. C. Kozen, “Limits for Automatic Verification of Finite-State Concurrent Systems,” *Information Processing Letters*, 1986.

[15] E. Emerson and V. Kahlon, “Reducing Model Checking of the Many to the Few,” in *Automated Deduction - CADE-17*, ser. LNCS. Springer, 2000, vol. 1831, pp. 236–254.

[16] S. M. German and A. P. Sistla, “Reasoning about systems with many processes,” *J. ACM*, vol. 39, no. 3, pp. 675–735, july 1992.

[17] E. A. Emerson and K. S. Namjoshi, “Reasoning about rings,” in *POPL*. ACM, 1995, pp. 85–94.

[18] E. M. Clarke, O. Grumberg, and M. C. Browne, “Reasoning about Networks with Many Identical Finite-State Processes,” ser. PODC ’86. ACM, 1986, pp. 240–248.

[19] R. P. Kurshan and K. McMillan, “A structural induction theorem for processes,” in *Proceedings of PODC*. ACM, 1989, pp. 239–247.

[20] P. Wolper and V. Lovinfosse, “Verifying properties of large sets of processes with network invariants,” in *Automatic Verification Methods for Finite State Systems*, ser. LNCS. Springer, 1990, vol. 407, pp. 68–80.

[21] E. Clarke, O. Grumberg, and S. Jha, “Verifying parameterized networks using abstraction and regular languages,” in *CONCUR: Concurrency Theory*, ser. LNCS. Springer, 1995, vol. 962, pp. 395–407.