

Formalising the translation of CTL into L_μ

Hasan Amjad

University of Cambridge Computer Laboratory, William Gates Building, 15 JJ Thomson Avenue, Cambridge CB3 0FD, UK (e-mail: Hasan.Amjad@cl.cam.ac.uk)

Abstract. The translation of the temporal logic CTL [2] into the modal μ -calculus L_μ [10] is formalised in the HOL theorem prover [8].

1 Introduction

Theorem proving and model checking are two complementary approaches to formal verification. Model checking is based on exhaustive exploration of the state space of the system under consideration. Verification is fully automatic and can provide counter-examples for debugging but suffers from the state explosion problem when dealing with complex systems. Theorem proving is based on exploring the space of correctness proofs for the system. It can handle complex formalisms but requires skilled manual guidance for verification and human insight for debugging.

HOL is based on the LCF proof assistant [7] and is written in Moscow ML. Terms are values of type `term` and can be freely constructed. Theorems are represented as values of type `thm` and can be constructed using axioms and inference rules only i.e. by proof. This reliance on minimising trusted code is often called the “fully-expansive” approach and gives a high assurance of the correctness of the results, provided the core trusted code and the underlying system (operating system, hardware etc.) are sound.

In [1] the feasibility of partially embedding a symbolic model checker for L_μ in HOL is demonstrated. This approach allows results returned from the model checker to be treated as theorems in HOL while retaining the advantages of the fully-expansive nature of HOL, without an unacceptable performance penalty.

One use of this approach would be to do property checking using logics that can be embedded into L_μ . Since L_μ is very expressive, most popular temporal logics can be embedded in it without loss of efficiency [6, 4]. A theorem returned by the model checker for an L_μ property can be translated into the corresponding theorem for whatever logic we are interested in, as long as the translation is correct. If the translation is done via a semantics-based embedding in HOL, we can use the theorem prover to prove it correct. Thus we can achieve a tight integration with established technology with little loss of efficiency, without having to write a verification tool for our new logic from scratch and with a high assurance of the correctness of our implementation. This paper provides proof-of-concept of this approach using as an example the standard embedding of the popular temporal logic CTL into L_μ . All definitions, propositions, lemmas and theorems presented here have been mechanised in HOL.

2 CTL

The temporal logic CTL is widely used to specify properties of a system in terms of the finite set AP of atomic propositions relevant to the system. We need to make the notion of “system” precise.

Definition 1. The tuple (S_M, S_{0M}, R_M, L_M) represents a Kripke structure M over AP where

- S_M is a finite set of states. A state is a boolean vector enumerating AP.
- $S_{0M} \subseteq S_M$ is the set of initial states.
- R_M is a binary transition relation on states such that $R_M(s, s')$ iff there is transition in M from s to s' . R_M is total i.e. $\forall s \in S. \exists s' \in S. R(s, s')$.
- $L_M : S_M \rightarrow 2^{AP}$ labels each state with the set of atomic propositions true in that state.

We elide the subscript whenever the owning Kripke structure is clear from the context.

CTL allows us to describe properties of the states and paths of a *computation tree*. A computation tree is formed by unwinding (infinitely) the transitions of M , starting with the initial states. For a path π , we use π_i to denote the i^{th} state along the path, and ${}^i\pi$ to denote the prefix of π upto but not including the state π_i , and π^i to denote the suffix of π starting from the state π_i .

Definition 2. A computation path π starting at some state s in a Kripke structure M is well formed, $PATHM\pi s$, if and only if $\pi_0 = s$, $\forall n. \pi_n \in S$ and $\forall n. R(\pi_n, \pi_{n+1})$.

Intuitively, a path starting with the state s is an infinite sequence of states s_0, s_1, s_2, \dots such that $s_0 = s$ and $\forall i. s_i \in S$. Paths are forced to be infinite by the totality of R .

The syntax of CTL is made out of *state formulas* which are true of states, and *path formulas* which are true of paths (where by path we just mean a sequence of states connected via transitions of the system). A well-formed CTL formula is always a state formula system. However, we need path formulas because the temporal operators talk about a state with respect to the path of the computation tree the state is on. Formally, formulas of CTL are constructed as follows:

Definition 3. Let AP be the set of atomic boolean propositions. Then CTL is the smallest set of all state formulas such that

- $p \in AP$ is a state formula.
- If f and g are state formulas then $\neg f$ and $f \wedge g$ are state formulas.
- If f and g are state formulas, then $\mathbf{X}f$, $\mathbf{F}f$, $\mathbf{G}f$, $f\mathbf{U}g$ and $f\mathbf{R}g$ are path formulas.
- If f is a path formula, then $\mathbf{A}f$ and $\mathbf{E}f$ are state formulas.

The ten compound operators thus formed can all be expressed in terms of the three operators \mathbf{EX} , \mathbf{EG} and \mathbf{EU} . We state the following without proof (see [4]) :

Proposition 4.

- $\mathbf{AX}f = \neg\mathbf{EX}(\neg f)$
- $\mathbf{EF}f = \mathbf{E}[True\mathbf{U}f]$
- $\mathbf{AG}f = \neg\mathbf{EF}(\neg f)$
- $\mathbf{AF}f = \neg\mathbf{EG}(\neg f)$
- $\mathbf{A}[f\mathbf{U}g] \equiv \neg\mathbf{E}[\neg g\mathbf{U}(\neg f \wedge \neg g)] \wedge \neg\mathbf{EG}(\neg g)$
- $\mathbf{A}[f\mathbf{R}g] \equiv \neg\mathbf{E}[\neg f\mathbf{U}\neg g]$
- $\mathbf{E}[f\mathbf{R}g] \equiv \neg\mathbf{A}[\neg f\mathbf{U}\neg g]$

Formally, the semantics $\llbracket f \rrbracket_M$ of a CTL formula f are defined with respect to the states of the Kripke structure M . They represent the set of states of M that satisfy f . Satisfaction of f by a state s of M is denoted by $M, s \models f$. So $M, s \models f \iff s \in \llbracket f \rrbracket_M$.

Definition 5. Given the CTL formulas f and g , atomic proposition p , a Kripke structure M and a state s of M ,

- $M, s \models p \iff p \in L(s)$
- $M, s \models f \iff M, s \not\models \neg f$
- $M, s \models f \wedge g \iff M, s \models f \wedge M, s \models g$
- $M, s \models \mathbf{EX}f \iff \exists \pi. PATHM\pi s \wedge M, \pi_1 \models f$
- $M, s \models \mathbf{EG}f \iff \exists \pi. PATHM\pi s \wedge \forall j. M, \pi_j \models f$
- $M, s \models \mathbf{E}[f\mathbf{U}g] \iff \exists \pi. PATHM\pi s \wedge \exists k. M, \pi_k \models g \wedge \forall j. j < k \Rightarrow M, \pi_j \models f$

The following lemmas will be useful later.

Lemma 6. For any Kripke structure M and CTL formula f ,

$$\llbracket \mathbf{EG}f \rrbracket_M = \llbracket f \wedge \mathbf{EX}(\mathbf{EG}f) \rrbracket_M$$

Proof

- \subseteq direction. For any $s \in S$,

$$\begin{aligned} & M, s \models \mathbf{EG}f \\ \iff & \exists \pi. \pi_0 = s \wedge \forall j. M, \pi_j \models f \quad \text{by 5} \\ \Rightarrow & M, s \models f \wedge \pi_1 \models \mathbf{EG}f \\ \iff & M, s \models f \wedge M, s \models \mathbf{EX}(\mathbf{EG}f) \quad \text{by 5} \\ \iff & M, s \models f \wedge \mathbf{EX}(\mathbf{EG}f) \quad \text{by 5} \end{aligned}$$

and we are done by extensionality.

– \supseteq direction. For any $s \in S$,

$$\begin{aligned}
& M, s \models f \wedge \mathbf{EX}(\mathbf{EG}f) \\
\iff & M, s \models f \wedge \exists \pi. \pi_0 = s \wedge M, \pi_1 \models (\mathbf{EG}f) \quad \text{by 5} \\
\iff & \exists \pi. \pi_0 = s \wedge \forall j. M, \pi_j \models f \quad \text{by 5} \\
\iff & M, s \models \mathbf{EG}f \quad \text{by 5}
\end{aligned}$$

and we are done by extensionality. \square

Effectively this is saying that $\mathbf{EG}f$ is a fixed point of the function $\tau(W) = f \wedge \mathbf{EX}(W)$. We have a similar result for $\mathbf{E}[f\mathbf{U}g]$.

Lemma 7. For any Kripke structure M and CTL formulas f and g ,

$$\llbracket \mathbf{E}[f\mathbf{U}g] \rrbracket_M = \llbracket g \vee (f \wedge \mathbf{EX}(\mathbf{E}[f\mathbf{U}g])) \rrbracket_M$$

Proof For any $s \in S$,

$$\begin{aligned}
& M, s \models \mathbf{E}[f\mathbf{U}g] \\
\iff & \exists \pi. \pi_0 = s \wedge \exists k. M, \pi_k \models g \wedge \forall j. j < k \Rightarrow M, \pi_j \models f \quad \text{by 5}
\end{aligned}$$

Now consider the cases $k = 0$ and $k \neq 0$. In the first case, we have $M, s \models g$. In the second case, we have $\forall j. j < k \Rightarrow M, \pi_j \models f$. But $k \neq 0$ so certainly $M, \pi_0 \models f$ i.e. $M, s \models f$. Further, $M, \pi_1 \models \mathbf{E}[f\mathbf{U}g]$ i.e. $M, s \models \mathbf{EX}(\mathbf{E}[f\mathbf{U}g])$, by 5. Since one of the two cases on k must hold, we have $M, s \models g \vee (M, s \models f \wedge M, s \models \mathbf{EX}(\mathbf{E}[f\mathbf{U}g]))$ and we have the required result by 5. \square

3 L_μ

Formulas of L_μ also describe properties of a system that can be represented as a state machine. As with CTL, the semantics of a formula is the set of states of the system for which the formula holds true.

The greater expressive power of L_μ requires a slightly modified version of the Kripke structure presented earlier. If AP is the set of atomic propositions relevant to our system, then a Kripke structure is defined as follows:

Definition 8. A Kripke structure M over AP is a tuple (S, S_0, T, L) where

- S is a finite set of states. A state is a boolean vector enumerating AP .
- $S_0 \subseteq S$ is the set of initial states.
- T is the set of actions such that for any action $a \in T$, $a \subseteq S \times S$.
- $L : S \rightarrow 2^{AP}$ labels each state with the set of atomic propositions true in that state.

Instead of a single transition relation, we now have a set of transition relations called actions. Note that the transition relations need not be total and therefore paths need not be infinite.

We now present the syntax and semantics of L_μ , essentially as given in [4]. The types of overloaded operators will be clear from the context.

Definition 9. Let $VAR = \{Q_i | Q_i \subseteq S\}$ be the set of relational variables. Then,

- True and False are formulas.
- If $p \in AP$, then p is a formula.
- A relational variable is a formula.
- If f and g are formulas, then $\neg f$, $f \wedge g$ and $f \vee g$ are formulas.
- If f is a formula and $a \in T$, then $[a]f$ and $\langle a \rangle f$ are formulas.
- If $Q \in VAR$ and f is a formula, then $\mu Q.f$ and $\nu Q.f$ are formulas, subject to the constraint that all occurrences of Q in the negated normal form of f are not negated.

We often use the term “variable” instead of “relational variable” or “propositional variable”; the meaning should be clear from the context. In the formulas $\mu Q.f$ and $\nu Q.f$, μ and ν are considered binders on Q , and thus there is the standard notion of bound and free variables. We use $f(Q_1, Q_2, \dots)$ to denote that Q_1, Q_2, \dots occur free in f .

Intuitively, the propositional fragment behaves as expected. In the modal fragment $[a]f$ holds of a state if f holds in all states reachable from that state by doing an a action, and $\langle a \rangle f$ holds of a state if it is possible to make an a action to a state in which f holds. We abbreviate $(s, s') \in a$ by $s \xrightarrow{a} s'$. It is often convenient to use a dot to denote an arbitrary action. In the recursive fragment, $\mu Q.f$ and $\nu Q.f$ represent the least and greatest fixpoints of the properties represented by f .

The semantics of a formula f is written $\llbracket f \rrbracket_{Me}$, where M is a Kripke structure and the environment $e : VAR \rightarrow 2^S$ holds the state sets corresponding to the free variables of f . By $e[Q \leftarrow W]$ we mean the environment that has $e[Q \leftarrow W]Q = W$ but is the same as e otherwise. We use \perp to denote the empty environment. We now define $\llbracket f \rrbracket_{Me}$.

Definition 10. The semantics of L_μ are defined recursively as follows

- $\llbracket True \rrbracket_{Me} = S$ and $\llbracket False \rrbracket_{Me} = \emptyset$
- $\llbracket p \rrbracket_{Me} = \{s | p \in L(s)\}$
- $\llbracket Q \rrbracket_{Me} = e(Q)$
- $\llbracket \neg f \rrbracket_{Me} = S \setminus \llbracket f \rrbracket_{Me}$
- $\llbracket f \wedge g \rrbracket_{Me} = \llbracket f \rrbracket_{Me} \cap \llbracket g \rrbracket_{Me}$
- $\llbracket f \vee g \rrbracket_{Me} = \llbracket f \rrbracket_{Me} \cup \llbracket g \rrbracket_{Me}$
- $\llbracket \langle a \rangle f \rrbracket_{Me} = \{s | \exists t. s \xrightarrow{a} t \wedge t \in \llbracket f \rrbracket_{Me}\}$
- $\llbracket [a]f \rrbracket_{Me} = \{s | \forall t. s \xrightarrow{a} t \Rightarrow t \in \llbracket f \rrbracket_{Me}\}$
- $\llbracket \mu Q.f \rrbracket_{Me}$ is the least fixpoint of the predicate transformer $\tau : 2^S \rightarrow 2^S$ given by $\tau(W) = \llbracket f \rrbracket_{Me[Q \leftarrow W]}$

- $\llbracket \nu Q.f \rrbracket_{Me}$ is the greatest fixpoint of the predicate transformer $\tau : 2^S \rightarrow 2^S$ given by $\tau(W) = \llbracket f \rrbracket_{Me}[Q \leftarrow W]$

Environnements can be given a partial ordering \subseteq under componentwise subset inclusion. Now the semantics evaluate monotonically over environments [10],

Proposition 11. *For any Kripke structure M , environments e and e' , relational variable Q and well-formed L_μ formula f , and $W \subseteq S$ and $W' \subseteq S$, we have*

$$e \subseteq e' \wedge W \subseteq W' \Rightarrow \llbracket f(Q) \rrbracket_{Me}[Q \leftarrow W] \subseteq \llbracket f(Q) \rrbracket_{Me'}[Q \leftarrow W']$$

so by Tarski's fixpoint theorem in [14], the existence of fixpoints is guaranteed. In fact, since S is finite, monotonicity implies continuity [14], which gives

Proposition 12.

$$\llbracket \mu Q.f \rrbracket_{Me} = \bigcup_i \tau^i(\emptyset) \text{ and } \llbracket \nu Q.f \rrbracket_{Me} = \bigcap_i \tau^i(M.S)$$

where $\tau^i(Q)$ is defined by $\tau^0(Q) = Q$ and $\tau^{i+1} = \tau(\tau^i(Q))$. So we can compute the fixpoints by repeatedly applying τ to the result of the previous iteration, starting with $\llbracket False \rrbracket_{Me}$ for least fixpoints and $\llbracket True \rrbracket_{Me}$ for greatest fixpoints. Since S is finite, the computation stops at some $k \leq |S|$, so that the least fixpoint is given by $\tau^k(False)$ and the greatest fixpoint by $\tau^k(True)$.

4 The Translation

The semantics for both CTL and L_μ are in terms of sets of states. This allows a purely syntactic translation scheme [4].

Definition 13. *The translation \mathcal{T} from CTL to L_μ is defined by recursion over CTL formulas as follows*

- $\mathcal{T}(p \in AP) = p$
- $\mathcal{T}(\neg f) = \neg \mathcal{T}(f)$
- $\mathcal{T}(f \wedge g) = \mathcal{T}(f) \wedge \mathcal{T}(g)$
- $\mathcal{T}(\mathbf{EX}f) = \langle \cdot \rangle \mathcal{T}(f)$
- $\mathcal{T}(\mathbf{EG}f) = \nu Q.f \wedge \langle \cdot \rangle Q$
- $\mathcal{T}(\mathbf{E}[f\mathbf{U}g]) = \mu Q.g \vee (f \wedge \langle \cdot \rangle Q)$

We need to prove this translation correct with respect to the semantics. Since the underlying models for CTL and L_μ are slightly different, we need to be able to translate a CTL model into an L_μ model. We overload \mathcal{T} for this purpose.

Definition 14. *If M is a Kripke structure as given in Definition 1, $\mathcal{T}M$ is the Kripke structure*

$$(S_M, S_{0M}, \lambda a.R_M, L_M)$$

Theorem 15.

$$\forall M f. \llbracket f \rrbracket_M = \llbracket \mathcal{T}(f) \rrbracket_{\mathcal{T}M\perp}$$

Proof By induction on the definition of f .

– $f \equiv p$.

$$\begin{aligned} & \llbracket p \rrbracket_M \\ &= \{s \mid p \in L_M(s)\} \quad \text{by 5} \\ &= \{s \mid \mathcal{T}(p) \in L_{\mathcal{T}M}(s)\} \quad \text{by 13, 14} \\ &= \llbracket \mathcal{T}(p) \rrbracket_{\mathcal{T}M\perp} \quad \text{by 10} \end{aligned}$$

– $f \equiv \neg f'$.

$$\begin{aligned} & \llbracket \neg f' \rrbracket_M \\ &= S \setminus \llbracket f' \rrbracket_M \quad \text{by 5 and set theory} \\ &= S \setminus \llbracket \mathcal{T}(f') \rrbracket_{\mathcal{T}M\perp} \quad \text{by the IH} \\ &= \llbracket \mathcal{T}(\neg f') \rrbracket_{\mathcal{T}M\perp} \quad \text{by 10, 13} \end{aligned}$$

– $f \equiv f' \wedge f''$.

$$\begin{aligned} & \llbracket f' \wedge f'' \rrbracket_M \\ &= \llbracket f' \rrbracket_M \cap \llbracket f'' \rrbracket_M \quad \text{by 5 and set theory} \\ &= \llbracket \mathcal{T}(f') \rrbracket_{\mathcal{T}M\perp} \cap \llbracket \mathcal{T}(f'') \rrbracket_{\mathcal{T}M\perp} \quad \text{by the IH} \\ &= \llbracket \mathcal{T}(f' \wedge f'') \rrbracket_{\mathcal{T}M\perp} \quad \text{by 10, 13} \end{aligned}$$

– $f \equiv \mathbf{EX} f'$. For any state $s \in S_M$,

$$\begin{aligned} & s \in \llbracket \mathbf{EX} f' \rrbracket_M \\ \iff & M, s \models \mathbf{EX} f' \quad \text{by definition of } \llbracket - \rrbracket_M \\ \iff & \exists \pi. \text{PATHM} \pi s \wedge M, \pi_1 \models f' \quad \text{by 5} \\ \iff & \exists \pi. \text{PATHM} \pi s \wedge \pi_1 \in \llbracket f' \rrbracket_M \quad \text{by definition of } \llbracket - \rrbracket_M \\ \\ \iff & \exists \pi. \text{PATHM} \pi s \wedge \pi_1 \in \llbracket \mathcal{T}(f') \rrbracket_{\mathcal{T}M\perp} \quad \text{by the IH} \\ \iff & \exists \pi. R_M(s, \pi_1) \wedge \pi_1 \in \llbracket \mathcal{T}(f') \rrbracket_{\mathcal{T}M\perp} \quad \text{by 2} \\ \iff & \exists \pi. s \dot{\rightarrow} \pi_1 \wedge \pi_1 \in \llbracket \mathcal{T}(f') \rrbracket_{\mathcal{T}M\perp} \quad \text{by definition of } \dot{\rightarrow} \end{aligned}$$

Now define our existential witness π by

$$\begin{aligned} \pi_0 &= s \\ \pi(n+1) &= (n=0)? s' \mid \varepsilon r. R_M(\pi_n, r) \end{aligned}$$

where ε is Hilbert's selection operator. Then simplifying and continuing,

$$\begin{aligned}
&\iff \exists s'.s \dot{\rightarrow} s' \wedge s' \in \llbracket \mathcal{T}(f') \rrbracket_{TM\perp} \quad \text{by definition of } \pi \\
&\iff s \in \{s \mid \exists s'.s \dot{\rightarrow} s' \wedge s' \in \llbracket \mathcal{T}(f') \rrbracket_{TM\perp}\} \quad \text{by defn of } \in \\
&\iff s \in \llbracket \langle \cdot \rangle \mathcal{T}(f') \rrbracket_{TM\perp} \quad \text{by 10} \\
&\iff s \in \llbracket \mathcal{T}(\mathbf{EX}f') \rrbracket_{TM\perp} \quad \text{by 13}
\end{aligned}$$

and we have the required result by extensionality.

– $f \equiv \mathbf{EG}f'$. Define

$$\tau(W) = \llbracket \mathcal{T}(f) \wedge \langle \cdot \rangle Q \rrbracket_{TM\perp} \perp [Q \leftarrow W]$$

and we have

- \subseteq direction.

$$\begin{aligned}
&\llbracket \mathbf{EG}f \rrbracket_M \subseteq \llbracket \mathcal{T}(\mathbf{EG}f) \rrbracket_{TM\perp} \\
&\iff \llbracket \mathbf{EG}f \rrbracket_M \subseteq \bigcap_n \tau^n S_{TM} \quad \text{by 12,13,10,14} \\
&\iff \forall n. \llbracket \mathbf{EG}f \rrbracket_M \subseteq \tau^n S_{TM} \quad \text{by set theory}
\end{aligned}$$

Then induction on n gives

- * $n \equiv 0$. Immediate by 10,14,12.
- * $n \equiv n' + 1$. Consider the “outer” IH,

$$\begin{aligned}
&\llbracket \mathbf{EG}f \rrbracket_M \subseteq \tau^{n'} S_{TM} \\
&\Rightarrow \tau(\llbracket \mathbf{EG}f \rrbracket_M) \subseteq \tau(\tau^{n'} S_{TM}) \quad \text{by 11} \\
&\iff \llbracket \mathbf{EG}f \rrbracket_M \subseteq \tau^{n'+1} S_{TM} \quad \text{by 6,13,10}
\end{aligned}$$

which is the required result.

- \supseteq direction.

$$\begin{aligned}
&\llbracket \mathbf{EG}f \rrbracket_M \supseteq \llbracket \mathcal{T}(\mathbf{EG}f) \rrbracket_{TM\perp} \\
&\iff \llbracket \mathbf{EG}f \rrbracket_M \supseteq \bigcap_n \tau^n S_{TM} \quad \text{by 12,13,10,14}
\end{aligned}$$

Now consider some $s \in \bigcap_n \tau^n S_{TM}$. By 12,

$$s \in \tau\left(\bigcap_n \tau^n S_{TM}\right)$$

Suppose π is a path starting at s . Then by the definition of τ , $\pi_1 \in \bigcap_n \tau^n S_{TM}$. We use the ε operator to pick π_1 for us (this is needed

because totality of $R.M$ only tells us that π_1 exists). By repeatedly using ε to pick the appropriate next state on the path, we can construct π such that $\forall i. \pi_i \in \bigcap_n \tau^n S_{TM}$. But for any $s', s' \in \bigcap_n \tau^n S_{TM} \Rightarrow M, s' \models f$ by the definition of τ and the outer IH. Thus we have that $\forall i. M, \pi_i \models f$, and we have the required result by 5.

– $f \equiv \mathbf{E}[f' \mathbf{U} f'']$. Define

$$\tau(W) = \llbracket \mathcal{T}(f'') \vee (\mathcal{T}(f') \wedge \langle \cdot \rangle Q) \rrbracket_{TM} \perp [Q \leftarrow W]$$

and we have

- \subseteq direction.

$$\begin{aligned} \llbracket \mathbf{E}[f' \mathbf{U} f''] \rrbracket_M &\subseteq \llbracket \mathcal{T}(\mathbf{E}[f' \mathbf{U} f'']) \rrbracket_{TM} \perp \\ \iff \llbracket \mathbf{E}[f' \mathbf{U} f''] \rrbracket_M &\subseteq \bigcup_n \tau^n \emptyset \quad \text{by 12,13,10,14} \end{aligned}$$

Now consider some $s \in \llbracket \mathbf{E}[f' \mathbf{U} f''] \rrbracket_M$. Then by 5 we have,

$$\exists \pi. \text{PATHM} \pi s \wedge \exists k. M, \pi_k \models f'' \wedge \forall j. j < k \Rightarrow M, \pi_j \models f'$$

We proceed by induction on the length $|^k \pi|$ of $^k \pi$.

* $|^k \pi| = 0$. Note that since π is infinite, $|^k \pi| = k$ by the definition of $^k \pi$. So $k = 0$. This implies $M, s \models f''$ by 2 i.e. $s \in \llbracket f'' \rrbracket_M$ and we are done by the definition of τ and the outer IH.

* $|^k \pi| = k' + 1$. We note again that $k = k' + 1$. Consider the path π^1 . Then,

$$\begin{aligned} M, \pi_k &\models f'' \\ \iff M, \pi_{k'}^1 &\models f'' \end{aligned}$$

and

$$\begin{aligned} \forall j. j < k &\Rightarrow M, \pi_j \models f' \\ \iff \forall j. j + 1 < k &\Rightarrow M, \pi_{j+1} \models f' \\ \iff \forall j. j < k' &\Rightarrow M, \pi_j^1 \models f' \end{aligned}$$

So by the IH,

$$\begin{aligned} \pi_0^1 &\in \bigcup_n \tau^n \emptyset \\ \iff \pi_1 &\in \bigcup_n \tau^n \emptyset \\ \iff s \in \tau \left(\bigcup_n \tau^n \emptyset \right) &\quad \because M, s \models f' \text{ and defn of } \tau \end{aligned}$$

and we are done by the outer IH.

- \supseteq direction.

$$\begin{aligned}
& \llbracket \mathbf{E}[f' \mathbf{U} f''] \rrbracket_M \supseteq \llbracket \mathcal{T}(\mathbf{E}[f' \mathbf{U} f'']) \rrbracket_{\mathcal{T}M} \perp \\
\iff & \llbracket \mathbf{E}[f' \mathbf{U} f''] \rrbracket_M \supseteq \bigcup_n \tau^n \emptyset \text{ by 12,13,10,14} \\
\iff & \forall n. \llbracket \mathbf{E}[f' \mathbf{U} f''] \rrbracket_M \supseteq \tau^n S_{\mathcal{T}M} \text{ by set theory}
\end{aligned}$$

Then induction on n gives

- * $n \equiv 0$. Immediate by 10,14,12.
- * $n \equiv n' + 1$. Consider the outer IH,

$$\begin{aligned}
& \llbracket \mathbf{E}[f' \mathbf{U} f''] \rrbracket_M \supseteq \tau^{n'} \emptyset \\
\Rightarrow & \tau(\llbracket \mathbf{E}[f' \mathbf{U} f''] \rrbracket_M) \supseteq \tau(\tau^{n'} \emptyset) \text{ by 11} \\
\iff & \llbracket \mathbf{E}[f' \mathbf{U} f''] \rrbracket_M \supseteq \tau^{n'+1} \emptyset \text{ by 7,13,10}
\end{aligned}$$

which is the required result.

□

5 Concluding Remarks

Closely related work includes the HOL-Voss system [13, 9] and the integration of an L_μ symbolic model checker with PVS [12, 11]. Voss has a lazy functional language FL with BDDs as a built-in datatype. In [9] Voss was interfaced to HOL and verification using a combination of deduction and symbolic trajectory evaluation (STE) was demonstrated. The interface does not allow a tight integration however: properties verified in Voss cannot be used as theorems in HOL without invoking oracles, which may compromise the assurance of soundness provided by the fully-expansive nature of HOL. In [12] this tight integration is achieved within the PVS framework. However, the proof step invoking the model checker is atomic and thus this approach does not extend readily to the fully-expansive approach used in HOL.

Since this result has been mechanised in HOL, we can convert a CTL property to L_μ , use the L_μ property checker, and convert the resulting theorem back to a CTL property. In general, we can leverage our existing property checker to verify specifications expressed in a new logic (embeddable in L_μ) without risking unsoundness caused by an incorrect translation. This may seem trivial for CTL but the translations of other popular logics such as LTL or CTL* into L_μ are considerably more involved [5, 3] and the chance of an incorrect implementation correspondingly higher. Our fully-expansive approach towards integrating model-checking and theorem-proving removes this possibility assuming only the soundness of the HOL kernel and the operating environment.

References

1. H. Amjad. Programming a symbolic model checker in a fully expansive theorem prover. In David A. Basin and Burkhart Wolff, editors, *Proceedings of the 16th International Conference on Theorem Proving in Higher Order Logics*, volume 2758 of *Lecture Notes in Computer Science*, pages 171–187. Springer-Verlag, 2003.
2. M. Ben-Ari, Z. Manna, and A. Pnueli. The temporal logic of branching time. *Acta Informatica*, 20:207–226, 1983.
3. E. M. Clarke, O. Grumberg, and K. Hamaguchi. Another look at LTL model checking. *Formal Methods in System Design*, 10(1):47–71, 1997.
4. E. M. Clarke, O. Grumberg, and D. Peled. *Model Checking*. The MIT Press, 1999.
5. M. Dam. CTL* and ECTL* as fragments of the modal mu-calculus. *Theoretical Computer Science*, 126(1):77–96, 1994.
6. E. A. Emerson and C-L. Lei. Efficient model checking in fragments of the propositional mu-calculus. In *1st Annual Symposium on Logic in Computer Science*, pages 267–278. IEEE Computer Society Press, 1986.
7. M. J. C. Gordon, A. R. G. Milner, and C. P. Wadsworth. Edinburgh LCF: A mechanised logic of computation. volume 78 of *LNCS*. Springer-Verlag, 1979.
8. The HOL-4 Proof Tool. Tool URL <http://hol.sf.net>, 2003.
9. Jeffrey J. Joyce and Carl-Johan H. Seger. The HOL-Voss system : Model checking inside a general-purpose theorem prover. In Jeffrey J. Joyce and Carl-Johan H. Seger, editors, *Higher Order Logic Theorem Proving and its Applications*, volume 780 of *LNCS*, pages 185–198. Springer, 1993.
10. D. Kozen. Results on the propositional mu-calculus. *Theoretical Computer Science*, 27:333–354, 1983.
11. S. Owre, S. Rajan, J. M. Rushby, N. Shankar, and M. K. Srivas. PVS: Combining specification, proof checking, and model checking. In Thomas A. Henzinger Rajeev Alur, editor, *CAV'96: 8th International Conference on Computer Aided Verification*, volume 1102 of *LNCS*, pages 411–414. Springer, July 1996.
12. S. Rajan, N. Shankar, and M. K. Srivas. An integration of model checking and automated proof checking. In Pierre Wolper, editor, *Proceedings of Computer Aided Verification*, volume 939 of *LNCS*, pages 84–97. Springer-Verlag, 1995.
13. C-J. H. Seger. Voss - a formal hardware verification system: User's guide. Technical Report UBC-TR-93-45, The University of British Columbia, December 1993.
14. A. Tarski. A lattice-theoretical fixpoint theorem and its applications. *Pacific Journal of Mathematics*, 5:285–309, 1955.