# CTL Model Update: Semantics, Computations and Implementation

**Yulin Ding** and **Yan Zhang**[1]

**Abstract.** Minimal change is a fundamental principle for modeling system dynamics. In this paper, we study the issue of minimal change for Computational Tree Logic (CTL) model update. We first propose five primitive operations which capture the basic update of the CTL model, and then define the minimal change criteria for CTL model update based on these primitive operations. We provide essential semantic and computational characterizations for our CTL model update approach. We develop a formal algorithm to implement this update that employs the underlying minimal change principle. We also present a CTL model update example using the well known microwave oven scenario.

## 1 Introduction

Over the past decade, automated formal verification tools, such as model checkers, have shown their ability to provide a thorough automatic error diagnosis in complex designs, e.g. [10]. The current state of the art model checkers, such as SMV [4], NuSMV [3] and Cadence SMV [9], employ SMV specification language for both CTL and Lineal Temporal Logic (LTL) model checking. Other model checkers such as SPIN [7] use Promela specification language for on the fly LTL model checking. Also, the MCK [5] model checker was developed by integrating a knowledge operator to verify the knowledge related properties in security protocols.

Along with model checking, error repair has begun to employ a formal methods approach. Buccafurri et al. [2] used abductive model revision techniques to repair errors in concurrent programs. They aimed at using techniques and concepts from model checking by combining them with AI principles. In the paper of Harris and Ryan [6], model checking is formalized with a belief updating operator to satisfy classical proposition knowledge update KM postulates $U1$-$U8$. Baral and Zhang [1] presented a formal approach of knowledge update based on single agent S5 Kripke structures. As they argued, their approach of knowledge update could be integrated with model checking technology towards a more general automatic system modification. In this paper, we consider the problem of CTL model update from both theoretical and implementational perspectives. We first consider five primitive CTL model update operations, and based on these operations, we define a minimal change principle for CTL model update. We then investigate the essential semantic and computational properties of CTL model update. Based on these findings, we develop an algorithm to perform CTL model update. By presenting a case study, we also show how our system prototype is applied for system modification.

[1] Intelligent Systems Laboratory, School of Computing and Mathematics, University of Western Sydney, Australia. Email: {yding,yan}@scm.uws.edu.au. The corresponding author: Yan Zhang.

## 2 CTL Syntax and Semantics: An Overview

To begin with, we briefly review the syntax and semantics of CTL. Readers are referred to [4] and [8] for details.

**Definition 1** *[4] Let $AP$ be a set of atomic propositions. A Kripke model $M$ over AP is a three tuple $M = (S, R, L)$ where*

1. *$S$ is a finite set of states,*
2. *$R \subseteq S \times S$ is a transition relation,*
3. *$L : S \to 2^{AP}$ is a function that assigns each state with a set of atomic propositions.*

**Definition 2** *[8] Computation tree logic (CTL) has the following syntax given in Backus naur form:*

$$\phi ::= \top |\ \bot\ |p|(\neg\phi)|(\phi \wedge \psi)|(\phi \vee \psi)|\phi \supset \psi|AX\phi|EX\phi$$
$$|AG\phi|EG\phi|AF\phi|EF\phi|A[\phi \cup \psi]|E[\phi \cup \psi]$$

*where $p$ is any propositional atom.*

A CTL formula is evaluated on a Kripke model $M$. A path in $M$ from a state $s$ is an infinite sequence of states $\pi \overset{def}{=} [s_0, s_1, \cdots, s_{i-1}, s_i, s_{i+1}, \cdots]$ such that $s_0 = s$ and $(s_i, s_{i+1}) \in R$ holds for all $i \geq 0$. We write $(s_i, s_{i+1}) \subseteq \pi$ and $s_i \in \pi$. If we express a path as $\pi = [s_0, s_1, \cdots, s_i, \cdots, s_j, \cdots]$ and $i < j$, we say that $s_i$ is a state *earlier* than $s_j$ in $\pi$ as $s_i < s_j$. For simplicity, we may use $succ(s)$ to denote state $s'$ if there is a relation $(s, s')$ in $R$.

**Definition 3** *[8] Let $M = (S, R, L)$ be a Kripke model for CTL. Given any $s$ in $S$, we define whether a CTL formula $\phi$ holds in state $s$. We denote this by $(M, s) \models \phi$. The satisfaction relation $\models$ is defined by structural induction on all CTL formulas:*

1. *$(M, s) \models \top$ and $M, s \not\models \bot$ for all $s \in S$.*
2. *$(M, s) \models p$ iff $p \in L(s)$.*
3. *$(M, s) \models \neg\phi$ iff $(M, s) \not\models \phi$.*
4. *$(M, s) \models \phi_1 \wedge \phi_2$ iff $(M, s) \models \phi_1$ and $(M, s) \models \phi_2$.*
5. *$(M, s) \models \phi_1 \vee \phi_2$ iff $(M, s) \models \phi_1$ and $(M, s) \models \phi_2$.*
6. *$(M, s) \models \phi_1 \to \phi_2$ iff $(M, s) \not\models \phi_1$, or $(M, s) \models \phi_2$.*
7. *$(M, s) \models AX\phi$ iff for all $s_1$ such that $(s, s_1) \in R$, $(M, s_1) \models \phi$.*
8. *$(M, s) \models EX\phi$ iff for some $s_1$ such that $s \to s_1$, $(M, s_1) \models \phi$.*
9. *$(M, s) \models AG\phi$ holds iff for all paths $[s_0, s_1, s_2, \cdots]$, where $s_0 = s$, and all $s_i$ along the path, $(M, s_i) \models \phi$.*
10. *$(M, s) \models EG\phi$ holds iff there is a path $[s_0, s_1, s_2, \cdots]$, where $s_0 = s$, and for all $s_i$ along the path, $(M, s_i) \models \phi$.*
11. *$(M, s) \models AF\phi$ holds iff for all paths $[s_0, s_1, s_2, \cdots]$, where $s_0 = s$, there is some $s_i$ in the path such that $(M, s_i) \models \phi$.*
12. *$(M, s) \models EF\phi$ holds iff there is a path $[s_{0,1}, s_2, \cdots]$, where $s_0 = s$, and for some $s_i$ along the path, $(M, s_i) \models \phi$.*

*13.* $(M,s) \models A[\phi_1 \cup \phi_2]$ *holds iff for all paths* $[s_0, s_1, s_2, \cdots]$, *where* $s_0 = s$, *the path satisfies* $\phi_1 \cup \phi_2$, *i.e. there is some* $s_i$ *along the path, such that* $(M, s_i) \models \phi_2$, *and, for each* $j < i$, $(M, s_j) \models \phi_1$.

*14.* $(M,s) \models E[\phi_1 \cup \phi_2]$ *holds iff there is a path* $[s_0, s_1, s_2, \cdots]$, *where* $s_0 = s$, *the path satisfies* $\phi_1 \cup \phi_2$, *i.e. there is some* $s_i$ *along the path, such that* $(M, s_i) \models \phi_2$, *and, for each* $j < i$, $(M, s_j) \models \phi_1$.

In the rest of this paper, without explicit declaration, we will assume that all CTL formulas occurring in our context will be satisfable. For instance, when we consider to update a Kripke model with a CTL formula $\phi$, we already assume that $\phi$ is satisfiable.

## 3 Minimal Change for CTL Model Update

Given a CTL kripke model and a (satisfiable) CTL formula, we consider how this model can be updated in order to satisfy the given formula. We first give the following general definition on CTL model update.

**Definition 4** *(***CTL Model Update***) Given a CTL Kripke model* $M = (S, R, L)$ *and a CTL formula* $\phi$. *An* update *of* $\mathcal{M} = (M, s_0)$ *where* $s_0 \in S$ *with* $\phi$ *is a CTL Kripke model* $M' = (S', R', L')$ *such that* $\mathcal{M}' = (M', s_0') \models \phi$ *where* $s_0' \in S'$. *We use* $Update(\mathcal{M}, \phi)$ *to denote the result* $\mathcal{M}'$ *and* $Update(\mathcal{M}, \phi) = \mathcal{M}$ *if* $\mathcal{M} \models \phi$.

Definition 4 only presents an essential requirement for a CTL model update and does not tell how such update should be conducted. Basically, as in traditional knowledge base update [11], we would expect that a CTL model update obeys an underlying minimal change principle. Furthermore, this minimal change should be defined based on some operational process so that a concrete algorithm for CTL model update can be implemented. To this end, we first consider five primitive operations on the CTL model that provide a basis for all complex CTL model updates.

### 3.1 Primitive Operations

The operations to update the CTL model can be decomposed into five types identified as PU1, PU2, PU3, PU4 and PU5. These primitive updates are defined in their simplest forms as follows.

**PU1: Adding a relation only**
Given $M = (S, R, L)$, its updated model $M' = (S', R', L')$ is the result of $M$ having only added one new relation. That is $S' = S$, $L' = L$, and $R' = R \cup \{(s_{ar}, s_{ar2})\}$ where $(s_{ar}, s_{ar2}) \notin R$ for one pair of $s_{ar}, s_{ar2} \in S$.

**PU2: Removing a relation only**
Given $M = (S, R, L)$, its updated model $M' = (S', R', L')$ is the result of $M$ having only removed one existing relation. That is, $S' = S$; $L' = L$, and $R' = R - \{(s_{rr}, s_{rr2})\}$ where $(s_{rr}, s_{rr2}) \in R$ for one pair of $s_{rr}, s_{rr2} \in S$.

**PU3: Substituting a state and its associated relation(s) only**
Given $M = (S, R, L)$, its updated model $M' = (S', R', L')$ is the result of $M$ having only substituted one existing state and its associated relation(s). That is, $S' = S[s/s_{ss}]$ (i.e. $S'$ is the set of states where one state $s$ in $S$ is substituted by $s_{ss}$), $R' = R \cup \{(s_i, s_{ss}), (s_{ss}, s_j) \mid (s_i, s), (s, s_j) \in R\} - \{(s_i, s), (s, s_j) \mid (s_i, s), (s, s_j) \in R\}$, and $L'(s) = L(s)$ for all $s \in S \cap S'$ and $L'(s_{ss}) = \tau(s_{ss})$, where $\tau$ is a truth assignment on $s_{ss}$.

**PU4: Adding a state and its associated relation(s) only**
Given $M = (S, R, L)$, its updated model $M' = (S', R', L')$ is the result of $M$ having only added one new state and its associated relation(s). That is, $S' = S \cup \{s_{as}\}$, $R' = R \cup \{(s_i, s_{as}), (s_{as}, s_j) \mid$ for some $s_i, s_j \in S \cap S'\}$, and $L'(s) = L(s)$ for all $s \in S \cap S'$ and $L'(s_{as}) = \tau(s_{as})$, where $\tau$ is the truth assignment on $s_{as}$.

**PU5: Removing a state and its associated relation(s) only**
Given $M = (S, R, L)$, its updated model $M' = (S', R', L')$ is the result of $M$ having only removed one existing state and its associated relation(s). That is, $S' = S - \{s_{rs} \mid s_{rs} \in S\}$, $R' = R - \{(s_i, s_{rs}), (s_{rs}, s_j) \mid$ for some $s_i, s_j \in S\}$, and $L'(s) = L(s)$ for all $s \in S \cap S'$ (note $S' \subset S$).

We call the above five operations *atomic* since all changes on a CTL model can be expressed by these five operations. It may be argued that PU3 can be expressed by PU4 and PU5. However, we will treat state substitution differently from a combination of state addition and state deletion. That is, in our context, whenever a state substitution is needed, we will apply PU3 directly rather than PU4 followed by PU5. This will simplify our definition on CTL model minimal change (see next).

### 3.2 Defining Minimal Change

In order to define the minimal change criteria for CTL model update, we need to consider changes on both states and relations for the underlying CTL models. We achieve this by specifying the differences of states and relations on CTL models through primitive operations. First we introduce some useful notions.

Given any two sets $X$ and $Y$, the *symmetric difference* between $X$ and $Y$ is denoted as $Diff(X, Y) = (X - Y) \cup (Y - X)$. Given two CTL models $M = (S, R, L)$ and $M' = (S', R', L')$, for each primitive operation $PUi$ $(i = 1, \cdots, 5)$, $Diff_{PUi}(M, M')$ denotes the differences between two CTL models where $M'$ is a resulting model from $M$, that make clear that several operations of this type may occur. Since PU1 and PU2 only change relations, we define $Diff_{PUi}(M, M') = (R - R') \cup (R' - R)$ $(i = 1, 2)$. For operations PU3, PU4 and PU5, on the other hand, we define $Diff_{PUi}(M, M') = (S - S') \cup (S' - S)$ $(i = 3, 4, 5)$. Although any changes on states caused by PU3-PU5 will also imply corresponding changes on relations, we only count changes on states and take state changes as the primitive factor to measure the difference between $M$ and $M'$[2]. For operation PU3, we should also consider the case that when a state is substituted by a new state, we require the difference between these two states to be minimal under the condition of satisfying the updated formula. Finally, we specify

$$Diff(M, M') = (Diff_{PU1}(M, M'), \cdots, Diff_{PU5}(M, M')).$$

Let $M$, $M_1$ and $M_2$ be three CTL models. We denote $Diff(M, M_1) \preceq Diff(M, M_2)$ iff (1) for each $i$ $(i = 1, \cdots, 5)$, $Diff_{PUi}(M, M_1) \subseteq Diff_{PUi}(M, M_2)$; or (2) $Diff_{PUi}(M, M_1) \subseteq Diff_{PUi}(M, M_2)$ for $i = 1, 2, 4, 5$, and $|Diff_{PU3}(M, M_1)| = |Diff_{PU3}(M, M_2)|$ implies for each state $s$ in $M$ substituted by $s_1$ and $s_2$ in $M_1$ and $M_2$ respectively, $Diff(s, s_1) \subseteq Diff(s, s_2)$.

**Definition 5** *(***Closeness Ordering***) Given three CTL Kripke models* $M$, $M_1$ *and* $M_2$, *where* $M_1$ *and* $M_2$ *are obtained from* $M$ *by applying PU1-PU5 operations. We say that* $M_1$ *is* closer *or* as close *to* $M$ as $M_2$, *denoted as* $M_1 \leq_M M_2$, *iff* $Diff(M, M_1) \preceq$

[2] In our full paper, we justify this principle by showing that as long as a state change is fixed, associated relation changes do not play a crucial role to influence the resulting CTL model.

*$Diff(M, M_2)$. We denote $M_1 <_M M_2$ if $M_1 \leq_M M_2$ and $M_2 \not\leq_M M_1$.*

**Definition 6** *(**Admissible Update**) Given a CTL Kripke model $M = (S, R, L)$, $\mathcal{M} = (M, s_0)$ where $s_0 \in S$, and a CTL formula $\phi$, $Update(\mathcal{M}, \phi)$ is called* admissible *if the following conditions hold: (1) $Update(\mathcal{M}, \phi) = (M', s_0') \models \phi$ where $M' = (S', R', L')$ and $s_0' \in S'$; and (2) there does not exist another resulting model $M'' = (S'', R'', L'')$ and $s_0'' \in S''$ such that $(M'', s_0'') \models \phi$ and $M'' <_M M'$.*

## 4 Semantic Characterizations

From Definition 6, we observe that for a given CTL Kripke model $M$ and a formula $\phi$, there may be many admissible updates to satisfy $\phi$, where some updates are simpler than others. In this section, we provide various semantic characterizations on CTL model update that present possible solutions to achieve admissible updates under certain conditions. In general, in order to achieve admissible update results, we may have to combine various primitive operations during an update process. Nevertheless, as will be shown in the following, for many situations, a single type primitive operation will be enough to achieve an admissible updated model. These characterizations also play an essential role to simplify CTL model update implementations.

**Theorem 1** *Let $M = (S, R, L)$ be a Kripke model and $s_0$ be an initial state in $S$ and $\mathcal{M} = (M, s_0) \not\models EX\phi$, where $\phi$ is a propositional formula. Then an admissible updated model $\mathcal{M}' = Update(\mathcal{M}, EX\phi)$ can be obtained by doing one of the following operations:*

1. *PU3 is applied to any $succ(s_0)$ once to substitute it with a new state $s^* \models \phi$ and $Diff(succ(s_0), s^*)$ to be minimal, or PU4 is applied one time after adding a new state $s^* \models \phi$ and a new relation $(s_0, s^*)$;*
2. *if there exists some $s_i \in S$ such that $s_i \models \phi$ and $s_i \neq succ(s_0)$, PU1 is applied one time to add a new relation $(s_0, s_i)$.*

Theorem 1 provides two cases where admissible CTL model update results can be achieved for formula $EX\phi$. The first case says that we can either select one of $s_0$'s successor states and substitute it with a new state satisfying $\phi$ (i.e. applying PU3 one time), or simply add a new state that satisfies $\phi$ as a successor of $s_0$ (i.e. applying PU4 one time). The second case indicates that if there is some state $s_i$ in $S$ that already satisfies $\phi$, then it is enough to simply add a new relation $(s_0, s_i)$ to make it as a successor of $s_0$. It is easy to see that both cases will yield new CTL models that satisfy $EX\phi$. The theorem shows that such new models are also minimal with respect to the original CTL model.

**Theorem 2** *Let $M = (S, R, L)$ be a Kripke model and $\mathcal{M} = (M, s_0) \not\models AG\phi$, where $s_0 \in S$ and $\phi$ is a propositional formula. Then an admissible updated model $\mathcal{M}' = Update(\mathcal{M}, AG\phi)$ can be obtained by the following: for each path starting from $s_0$: $\pi = [s_0, \cdots, s_i, \cdots]$:*

1. *if for all $s < s_i$ in $\pi$, $s \models \phi$ but $s_i \not\models \phi$, PU2 is applied to remove relation $(s_{i-1}, s_i)$, or PU5 is applied to remove $s_i$ and its associated relations;*
2. *PU3 is applied to all states $s$ in $\pi$ not satisfying $\phi$ to substitute $s$ with $s^* \models \phi$ and $Diff(s, s^*)$ to be minimal.*

In Theorem 2, Case 1 considers a special form of path $\pi$ where the first $i$ states starting from $s_0$ already satisfy formula $\phi$. Under this situation, we can simply cut off the path (i.e. applying PU2 or PU5 one time) to disconnect all other states not satisfying $\phi$. Case 2 is straightforward.

**Theorem 3** *Let $M = (S, R, L)$ be a Kripke model, $\mathcal{M} = (M, s_0) \not\models EG\phi$, where $s_0 \in S$ and $\phi$ is a propositional formula. Then an admissible updated model $\mathcal{M}' = Update(\mathcal{M}, EG\phi)$ can be obtained by the following: Select a path $\pi = [s_0, s_1, \cdots]$ from $M$ which contains minimal number of states not satisfying $\phi$, and then*

1. *if for all $s' \in \pi$ such that $s' \not\models \phi$, there exist $s_i, s_j \in \pi$ satisfying $s_i < s' < s_j$ and $s_i \models \phi$ and $s_j \models \phi$, then PU1 is applied to add a relation $(s_i, s_j)$, or PU4 is applied to add a state $s^* \models \phi$ and new relations $(s_i, s^*)$ and $(s^*, s_j)$;*
2. *if there exists some $s_i \in \pi$ ($i > 1$) such that for all $s' < s_i$, $s' \models \phi$ and $s_i \not\models \phi$, then PU2 is applied to remove relation $(s_{i-1}, s_i)$, or PU5 is applied to remove state $s_i$ and its associated relations;*
3. *for all $s' \in \pi$, $s' \not\models \phi$, then PU3 is applied to substitute all $s'$ with new state $s^* \models \phi$ and $Diff(s, s^*)$ to be minimal.*

**Proof:** We prove case 1 here while proofs for the other two cases are presented in our full paper. Without loss of generality, we assume for the selected path $\pi$, there exists one state $s'$ that does not satisfy $\phi$, and all other states in $\pi$ satisfy $\phi$. We also assume that such $s'$ is in the *middle* of path $\pi$. Therefore, there are two other states $s_i, s_j$ in $\pi$ such that $s_i < s' < s_j$. That is, $\pi = [s_0, \cdots, s_{i-1}, s_i, \cdots, s', \cdots, s_j, s_{j+1}, \cdots]$. We first consider to apply PU1. It is clear that by applying PU1 to add a new relation $(s_i, s_j)$, a new path is formed: $\pi' = [s_0, \cdots, s_{i-1}, s_i, s_j, s_{j+1}, \cdots]$. Note that each state in $\pi'$ is also in path $\pi$ and $s' \notin \pi'$. According, we know $EG\phi$ is held in the new model $M' = (S, R \cup \{(s_i, s_j\}, L)$ at state $s_0$. On the other hand, we consider $Diff(M, M')$. Clearly, $Diff(M, M') = (\{(s_i, s_j)\}, \emptyset, \emptyset, \emptyset, \emptyset)$, which implies $M'$ must be a minimal model with respect to $\leq_M$ that satisfies $EG\phi$.

Now we consider to apply PU4. In this case, we will have a new model $M' = (S \cup \{s^*\}, R \cup \{(s_i, s^*), (s^*, s_j)\}, L')$ where $L'$ is an extension of $L$ on new state $s^*$ that satisfies $\phi$. We can see that $\pi' = [s_0, \cdots, s_i, s^*, s_j, \cdots]$ is a path in $M'$ which shares all states with path $\pi$ except the state $s^*$ in $\pi'$ and those states between $s_{i+1}$ and $s_{j-1}$ including $s'$ in $\pi$. So we also have $(M', s_0) \models EG\phi$. On the other hand, we have $Diff(M, M') = (\emptyset, \emptyset, \emptyset, \{s^*\}, \emptyset)$. Obviously, $M'$ is a minimal model with respect to $\leq_M$ that satisfies $EG\phi$. □

Theorem 3 characterizes three typical situations for the update with formula $EG\phi$. Basically, this theorem says that in order to make formula $EG\phi$ be true, we first select a path, then we can either make a new path based on this path so that all states in the new path satisfy $\phi$ (i.e. Case 1), trim the path from the state where all previous states satisfy $\phi$ (i.e. Case 2); or simply substitute all states not satisfying $\phi$ in the path with new states satisfying $\phi$ (i.e. Case 3). Our proof shows that resulting models from these operations are admissible.

In our full paper, we also provide semantic characterizations for updates with other CTL formulas such as $EF\phi$, $AX\phi$, $AF\phi$, $E[\phi \cup \psi]$ and $A[\phi \cup \psi]$. All these results have been proved to be useful in simplifying the implementation and improving the efficiency of the CTL model update performance.

## 5 Computational Properties

In this section, we study the computational complexity of our approach for CTL model update. We first present the following general result.

**Theorem 4** *Given two CTL Kripke models $M = (S, R, L)$ and $M' = (S', R', L')$, where $s_0 \in S$ and $s'_0 \in S'$, and a CTL formula $\phi$. Deciding whether $(M', s'_0)$ is an admissible result of $Update((M, s_0), \phi)$ is co-NP-complete. The hardness holds even if $\phi$ is of the form $EX\psi$ where $\psi$ is a propositional formula.*

**Proof:** Membership proof. First from [4], we know that checking whether $(M', s'_0) \models \phi$ can be done in time $\mathcal{O}(|\phi| \cdot (|S| + |R|))$. In order to check whether $(M', s'_0)$ is an admissible update result, we need to check whether $M'$ is minimal with respect to ordering $\leq_M$. To do so, we consider the complement of the problem. That is to check whether $M'$ is *not* a minimal model. Therefore, we do two things: (1) guessing a CTL model $M'' = (S'', R'', L'')$ satisfying $\phi$ for some $s'' \in S''$; and (2) testing whether $M'' <_M M'$. Step (1) can be done in polynomial time. To check $M'' <_M M'$, we first compute $Diff(S, S')$, $Diff(S, S'')$, $Diff(R, R')$ and $Diff(R, R'')$. All these can be computed in polynomial time. Then according to these sets, we identify, through PU1 to PU5, $Diff_{PUi}(M, M')$ and $Diff_{PUi}(M, M'')$ $(i = 1, \cdots, 5)$. Again, these can also be done in polynomial time. Finally, by checking $Diff_{PUi}(M, M'') \subseteq Diff_{PUi}(M, M')$ $(i = 1, \cdots, 5)$ we can decide whether $M'' <_M M'$. So both steps (1) and (2) can be achieved in polynomial time with a non-deterministic Turing machine.

Hardness proof. It is well known that the validity problem for a propositional formula is co-NP-complete. Given a propositional formula $\phi$, we construct in polynomial time a transformation from the problem of deciding $\phi$'s validity to a CTL model update. Let $X$ be the set of all variables occurring in $\phi$, and $a, b$ two new variables not occurring in $X$. We denote $\neg X = \bigwedge_{x_i \in X} \neg x_i$. We specify a CTL Kripke model based on the variable set $X \cup \{a, b\}$: $M = (\{s_0, s_1\}, \{(s_0, s_1), (s_1, s_1)\}, L)$, where $L(s_0) = \emptyset$ (i.e. all variables are assigned false), $L(s_1) = X$ (i.e. variables in $X$ are assigned true, while $a, b$ are assigned false). Now we define a new formula $\mu = EX(((\phi \supset a) \wedge (\neg X \wedge b)) \vee (\neg\phi \wedge a))$. Clearly, formula $((\phi \supset a) \wedge (\neg X \wedge b)) \vee (\neg\phi \wedge a)$ is satisfiable and $s_1 \not\models ((\phi \supset a) \wedge (\neg X \wedge b)) \vee (\neg\phi \wedge a)$. So $(M, s_0) \not\models \mu$. Consider the update $Update((M, s_0), \mu)$. We define $M' = (\{s'_0, s'_1\}, \{(s'_0, s'_1), (s'_1, s'_1)\}, L')$, where $L'(s'_0) = L(s_0)$ and $L'(s'_1) = \{a, b\}$. Now we show that $\phi$ is valid iff $(M', s'_0)$ is an admissible update result from $Update((M, s_0), \mu)$.

<u>Case 1</u>. We show that if $\phi$ is valid, then $(M', s'_0)$ is an admissible update result from $Update((M, s_0), \mu)$. Since $\phi$ is valid, we have $\neg X \models \phi$. So we have $s'_1 \models (\phi \supset a) \wedge (\neg X \supset b))$. This follows $(M', s'_0) \models \mu$. Also note that $M'$ is obtained by applying PU3 to substitute $s_1$ with $s'_1$. $Diff(s_1, s'_1) = X \cup \{a, b\}$ which presents a minimal change from $s_1$ in order to satisfy $(\phi \supset a) \wedge (\neg X \wedge b)$.

<u>Case 2</u>. Suppose that $\phi$ is not valid. Then there exists $X_1 \subseteq X$ such that $X_1 \models \neg\phi$. We construct $M'' = (\{s''_0, s''_1\}, \{(s''_0, s''_1), (s''_1, s''_1)\}, L'')$, where $L''(s''_0) = L(s_0)$ and $L''(s''_1) = X_1 \cup \{a\}$. It is easy to see that $s''_1 \models (\neg\phi \wedge a)$, and hence $(M'', s''_0) \models \mu$. Now we show that $(M', s'_0) \models \mu$ implies $M'' <_M M'$. Suppose $(M', s'_0) \models \mu$. Clearly, both $M'$ and $M''$ are obtained from $M$ by applying PU3 one time on $s_1$ respectively. But we have $Diff(s_1, s''_1) = (X - X_1) \cup \{a\} \subset X \cup \{a, b\} = Diff(s, s'_1)$. This concludes that $(M', s'_0)$ is not an admissible update model. □

Theorem 4 implies that it is probably not feasible to develop a polynomial time algorithm to implement our CTL model update. Indeed, our algorithm described in the next section, generally runs in exponential time. Nevertheless, we can identify certain typical update cases that may be achieved in polynomial time.

**Theorem 5** *Let $M = (S, R, L)$ be a CTL Kripke model and $\phi$ a CTL formula. If an admissible update result $Update((M, s_0), \phi)$ $(s_0 \in S)$ can be obtained by only applying PU1, PU2, PU4 and PU5 without path selection in $M$ (i.e. $\phi$ does not involve $EG\psi$, $EF\psi$ or $E[\psi_1 \cup \psi_2]$), then this result can be computed in polynomial time.*

## 6 The Algorithm and a Case Study

A prototype for the CTL model update addressed in previous sections has been implemented recently. The algorithm is designed following a similar style of CTL model checking algorithm SAT [8], where an updated formula is parsed through its structure and recursive calls to proper functions are made to its sub-formulas.

```
CTLUpdate($\mathcal{M}$,$\phi$) / $\mathcal{M} = (M, s_0) \not\models \phi$. Update $\mathcal{M}$ to satisfy $\phi$. /
  Input: $M = (S, R, L)$, $\mathcal{M}= (M, s_0)$, where $s_0 \in S$ and $\mathcal{M}\not\models \phi$;
  Output: $M' = (S', R', L')$, $\mathcal{M}'= (M', s'_0)$, $s'_0 \in S'$, $\mathcal{M}'\models \phi'$;
  { case
      $\phi$ is $\bot$ : return $\{M\}$;
      $\phi$ is atomic $p$ : return {Update$_p$($\mathcal{M}$, $p$)};
      $\phi$ is $\neg\phi_1$ : return {Update$_\neg$($\mathcal{M}$, $\phi_1$)};
      $\phi$ is $\phi_1 \vee \phi_2$ :
        return{CTLUpdate($\mathcal{M}$, $\phi_1$) or CTLUpdate($\mathcal{M}$, $\phi_2$)};
      $\phi$ is $\phi_1 \wedge \phi_2$: return {Update$_\wedge$($\mathcal{M}$,$\phi_1$, $\phi_2$)};
      $\phi$ is $EX\phi_1$: return {Update$_{EX}$($\mathcal{M}$, $\phi_1$)};
      $\phi$ is $AX\phi_1$: return {Update$_{AX}$($\mathcal{M}$, $\phi_1$)};
      $\phi$ is $EF\phi_1$: return {Update$_{EF}$($\mathcal{M}$, $\phi_1$)};
      $\phi$ is $AF\phi_1$: return {Update$_{AF}$($\mathcal{M}$, $\phi_1$)};
      $\phi$ is $EG\phi_1$: return {Update$_{EG}$($\mathcal{M}$, $\phi_1$)};
      $\phi$ is $AG\phi_1$: return {Update$_{AG}$($\mathcal{M}$, $\phi_1$)};
      $\phi$ is $E(\phi_1 \cup \phi_2)$: return {Update$_{EU}$($\mathcal{M}$, $\phi_1$, $\phi_2$)};
      $\phi$ is $A(\phi_1 \cup \phi_2)$: return {Update$_{AU}$($\mathcal{M}$, $\phi_1$, $\phi_2$)};
  }
```

Due to a space limit, here we only outline the main idea of implementing functions $Update_\neg$ and $Update_{AG}$.

```
Update$_\neg$($\mathcal{M}$,$\phi$)  / $\mathcal{M} \not\models \phi$. Update $\mathcal{M}$ to satisfy $\phi$. /
  {  case
      $\phi$ is $\neg p$: return {Update$_\neg p$($\mathcal{M}$,$p$)};
      $\phi$ is $\neg(\phi_1 \wedge \phi_2) = \neg\phi_1 \vee \neg\phi_2$:
          return {Update$_\neg$($\mathcal{M}$,$\phi_1$)} or {Update$_\neg$($\mathcal{M}$,$\phi_2$)};
      $\phi$ is $\neg EF(\phi_1) = AG(\neg\phi_1)$: return {Update$_{AG}$($\mathcal{M}$,$\neg\phi_1$)};

   }
Update$_{AG}$($\mathcal{M}$,$\phi$)    /$\mathcal{M}\not\models AG\phi$. Update $\mathcal{M}$ to satisfy $AG\phi$./
{ if $\mathcal{M}_0 = (M, s_0) \not\models \phi$, then PU3 is applied to $s_0$
        such that $\mathcal{M}' = CTLUpdate(\mathcal{M}_0, \phi)$;
  else select a path $\pi = [s_0, s_1, \cdots]$, where $\exists s \in \pi$ such that
        $\mathcal{M}_i = (M, s) \not\models \phi$;
    select the earliest state $s_i \in \pi$ such that $(M, s_i) \not\models \phi$;
    perform one of the following three operations:
      (1) applying PU2 to remove relation $(s_{i-1}, s_i)$,
            obtain result $\mathcal{M}'$;
      (2) applying PU5 to remove state $s_i$ and
            its associated relations, obtain result $\mathcal{M}'$;
      (3) applying PU3: $\mathcal{M}' = CTLUpdate(\mathcal{M}_i, \phi)$;
    if $\mathcal{M}' \models AG\phi$, return $\mathcal{M}'$;
    else return $\{Update_{AG}(\mathcal{M}', \phi)\}$;
}
```

**Theorem 6** *Given a Kripke model $M = (S, R, L)$ and a satisfiable CTL formula $\phi$ such that $(M, s) \not\models \phi$ for some $s \in S$. Then $CTLUpdate((M, s), \phi))$ will return a resulting Kripke model $M' = (S', R', L')$ such that $(M', s') \models \phi$ for some $s' \in S'$ and it is admissible.*

In the rest of this section, we will describe some details of our algorithm implementation by applying the model modification to the well known microwave oven scenario [4]. The microwave oven model is described as in Figure 1.

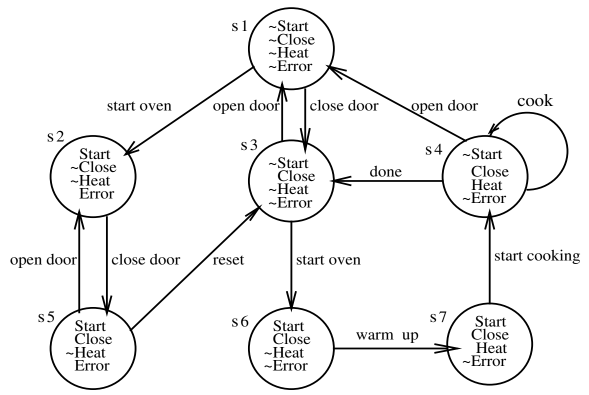

Figure 1. The CTL Kripke structure of a microwave oven.

The Kripke model has 7 states and propositional variables are from the set $\{Start, Close, Heat, Error\}$. It is observed that this model does not satisfy a desirable property $\phi = \neg EF(Start \wedge EG\neg Heat)$ [4]. What we would like to do is to apply our CTL model update program to minimally modify this kripke model to satisfy property $\phi$. First, we parse $\phi$ into $AG(\neg(Start \wedge EG\neg Heat))$ to remove the front $\neg$. The translation is done by function $Update_{\neg}$. Then we check for each state whether it satisfies $\neg(Start \wedge EG\neg Heat)$. This string is parsed before it is checked. We pick out $EG\neg Heat$ to feed through the model checking function for $EG$. In this model, any path that has any state with $\neg Heat$ is picked out. Here we find paths $[s_1, s_2, s_5, s_3, s_1, \cdots]$ and $[s_1, s_3, s_1, \cdots]$ which are strongly connected component loops [4] satisfying $EG\neg Heat$. Then we identify all states with $Start$, they are $\{s_2, s_5, s_6, s_7\}$. Now we select those states with both $Start$ and $\neg Heat$, they are $\{s_2, s_5\}$. Since formula $AG(\neg(Start \wedge EG\neg Heat))$ identifies that the model should not have any states with both $Start$ and $\neg Heat$, we should perform model update related to states $s_2$ and $s_5$.

Now function $Update_{AG}$ starts to perform update as described in the above pseudo code and Theorem 2 (see section 3). It turns out that there are three possible minimal updates: (1) applying PU2 to remove relation $(s_1, s_2)$; (2) applying PU5 to remove state $s_2$ and its associated relations $(s_1, s_2)$, $(s_2, s_5)$ and $(s_5, s_2)$; and (3) applying PU3 on $s_2$ and $s_5$. To perform option (3), our program first converts $\neg(Start \wedge EG\neg Heat)$ to $\neg Start \vee \neg EG\neg Heat$, then $s_2$ and $s_5$ are updated with either $\neg Start$ or $\neg EG\neg Heat$ by the main function $CTLUpdate$ to deal with $\vee$ and function $Update_{\neg}$. In our program, updating $\neg Start$ is chosen since formula $\neg Start$ is simpler than $\neg EG\neg Heat$. After the update, we will obtain a resulting model $(M, s_1) \models \phi$. For instance, by choosing update (1) operation above, we will obtain a new Kripke model as shown in Figure 2, which simply states that no state transition from $s_1$ to $s_2$ is allowed.

Our algorithm will generate one of the three resulting models without specific indication, because under our criteria they are all minimally changed from the original model. However, in our prototype implementation, we have considered the computational cost of an update upon users' request, which may affect the system efficiency. For instance, in the above example, with three possible updates, if the user requires a low computational cost, our system will select the simplest operation (1) to achieve the goal.

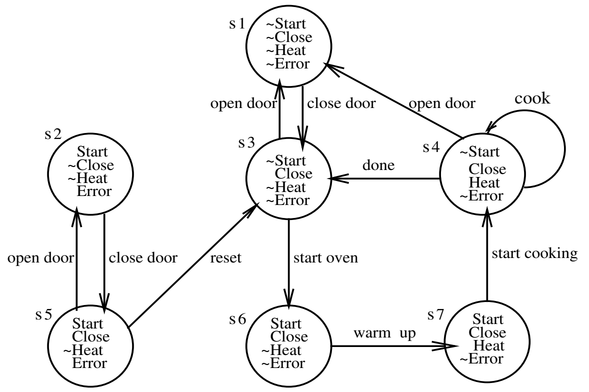

Figure 2. An updated CTL Kripke structure for the microwave oven.

## 7 Conclusion

In this paper, we presented a formal approach to update CTL models. By specifying five primitive operations on CTL Kripke models, we defined the minimal change criteria for CTL model update. We also investigated the semantic and computational properties of our approach in detail. Based on our formalization, we developed a CTL model update algorithm and implemented a system prototype to perform CTL model updating. Our case study showed an application of this work.

## REFERENCES

[1] C. Baral and Y. Zhang, 'Knowledge updates: semantics and complexity issues', *Artificial Intelligence*, **164**, 209–243, (2005).

[2] F. Buccafurri, T. Eiter, G. Gottlob, and N. Leone, 'Enhancing model checking in verification by ai techniques', *Artificial Intelligence*, **112**, 57–104, (1999).

[3] A. Cimatti and et al, 'Nusmv: A new symbolic model verifier', in *Proceedings of the 11th International Conference on Computer Aided Verification*, pp. 495–499, (1999).

[4] E.Jr. Clarke, O. Grumberg, and D.A. Peled, *Model Checking*, MIT Press, Cambridge, Massachusetts, 2000.

[5] P. Gammie and R. van der Meyden, 'Mck-model checking the logic of knowledge', in *the Proceedings of the 16th International Conference on Computer Aided Verification*, pp. 479–483, (2004).

[6] H. Harris and M. Ryan, 'Theoretical foundations of updating systems', in *the Prodeedings of the 18th IEEE International Conference on Automated Software Engineering*, pp. 291–298, (2003).

[7] C. Holzmann, *The SPIN Model Checker: Primer and Reference Manual*, Addison-Wesley Professional, 2003.

[8] M. Huth and M. Ryan, *Logic in Computer Science: Modelling and Reasoning about Systems*, Cambridge University Press, 2000.

[9] K. McMillan and N. Amla, 'Automatic abstraction without counterexamples', in *Cadence Berkeley Labs, Cadence Design Systems*, (2002).

[10] J. Wing and M. Vaziri-Farahani, 'A case study in model checking software', in *Proceedings of the 3rd ACM SIGSOFT Symposium on the Foundations of Software Engineering*, (1995).

[11] M. Winslett, *Updating Logical Databases*, Cambridge University Press, 1990.