The Algorithm and a Case Study for CTL Model Update

Laura Florentina Cacovean, Cristina Ioana Brumar, Emil Marin Popa

***Abstract*—**In this paper is presented an update of the CTL model checker. The minimal modifications which appear represent the fundamental concept for model the dynamic system. In the paper used five primitive operations discompose from the operation of a CTL update used already by [1] which presented their approach of knowledge update the structures of single agent S5 Kripke. Then is willed defined the criteria of minimum change for the update of CTL model based on these primitive operations. The final in this section paper is willed present the algorithm of implement the CTL model updated and is will describe some details of algorithm implementation by applying the model update to the microwave oven scenario. The paper [10] is the base of results obtained.

***Keywords*—**CTL Kripke model, CTL model update, modeling systems dynamics, algorithm, atomic propositions, directed graph, implementation.

I. Introduction

THE verification tools to automated formal, such as model checkers, shows delivered a diagnosis to provide a thorough automatic error diagnosis in complex designs, examples in [5]. The current state of the art model checkers, as of example SMV [3], Cadence SMV [6], uses SMV as specification language for both CTL (Computational Tree Logic) and LTL (Lineal Temporal Logic) model checking. [2] used the abdicate model revision the techniques mended the errors in the concurrent programs. Progressing the update of the method of the model checkers, begun to employ a formal method for approximate for repair the error. In they work [4] the model checking is formalized offence with a updating operator satisfied the axioms U1-U8 what represent the classical proposition knowledge of updated KM. [1] are presented their approach of knowledge update the structures of single agent S5 Kripke.

The arguments using of these with approach their knowledge can be incorporate with the technology the model checkers with the aim generalized more the modification of the automatic system. In this paper, we considered the problem of the update of CTL model from both theories and the views of achievement.

In substance, as in the traditional knowledge is based the update [9], we consider an update of CTL model subdue a principle of minimum inferior change. More, this change the minimum be as well to is definite as a process based on of some operational process which so a concrete algorithm for the update of CTL model to can to be implemented.

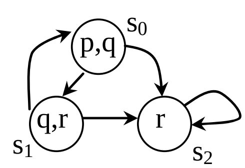
Is defined the principle of minimum change for the update of CTL model. Then research a necessary semantic and then calculating properties for an update the CTL model. Based on these ascertainment developed the algorithm for the execution update of CTL model. Presenting a study of case, we shown how the prototype of found system is applied for the system modified.

II. CTL model. Syntax and semantics

To begin with, we briefly review the syntax and semantics of CTL. Readers are referred to [3] and [8] for details.

Definition of a Kripke model [3] let *AP* is a set of atomic propositions. A Kripke model *M* over *AP* is a triple $M = (S,R, \mathcal{F}:S\rightarrow 2^{AP})$ where *S* is a finite set of states, $R\subseteq S\times S$ is a transition relation, $\mathcal{F}:S\rightarrow 2^{AP}$ is a function that assigns each state with a set of atomic proposition.

An example transition state graph is represented with form:



For more lightness for understand the methodology using CTL model checker we present an algebraic form presented in paper [7]. A model is defined [1] as a directed graph $M=\langle S, E, P:AP\rightarrow 2^{S}\rangle$ where *S* is a finite sets of states also called nodes, *E* is a finite sets of directed edges, and *P* represents proposition labelling function which labels each nodes with logical proposition. For each $s\in S$, use the notation $succ(s)=\{s'\in S \mid (s,s')\in E\}$. Each state in E must have at least one successor, that is $\forall s\in S$, $succ(s)\neq\emptyset$. A path in M is a infinite sequence of states $(s_0, s_1, s_2, \ldots)$ such that $\forall i$, $i\geq 0$, we have $(s_i, s_{i+1})\in E$. The labeling function *P* maps an atomic proposition in *AP* to the set of states in S on which sentences

Manuscript received February 29, 2008.
Laura Florentina Cacovean, is with the Department of Computer Science, Lucian Blaga University of Sibiu, Faculty of Sciences, Str. Dr. Ion Ratiu 5-7, 550012, Sibiu, ROMANIA, laura.cacovean@ulbsibiu.ro

Cristina Ioana Brumar, is with the Department of Computer Science, Lucian Blaga University of Sibiu, Faculty of Sciences, Str. Dr. Ion Ratiu 5-7, 550012, Sibiu, ROMANIA, cristina.brumar@ulbsibiu.ro

Emil Marin Popa, is with the Department of Computer Science, Lucian Blaga University of Sibiu, Faculty of Sciences, Str. Dr. Ion Ratiu 5-7, 550012, Sibiu, ROMANIA, emilm.popa@ulbsibiu.ro

is true. The Figure 1 exhibits a model [1] the behavior two processes competing in the entrance the critical section. The atomic propositions T_i, N_i, and C_i denote, respectively, process i, $1\leq i\leq 2$, try to enter into critical section, not to enter into critical section and to executed in the critical section.

The CTL formulas are defined by the following rules [1]:

1. The logical constants true and false are CTL formulas.
2. Every atomic proposition, $ap \in AP$ is a CTL formula.
3. If f_1 and f_2 are CTL formulas, then so are $\neg f_1$, $f_1 \wedge f_2$, $f_1 \vee f_2$, $EX\, f_1$, $AX\, f_1$, $E[f_1 U f_2]$, and $A[f_1 U f_2]$.

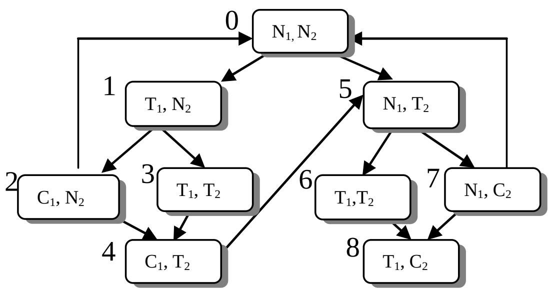


Fig. 1: Model example

Syntax definition of a CTL model checker [8] A CTL has the following syntax given in Backus near form:

f:: $\top$ |$\bot$ |p|($\neg$ f_1)| $f_1 \wedge f_2$| $f_1 \vee f_2$| $f_1 \supset f_2$| AX f_1| EX f_1| AG f_1| EG f_1| AF f_1| EF f_1|A[$f_1 \cup f_2$]| E[$f_1 \cup f_2$] where $\forall$ p$\in$AP.

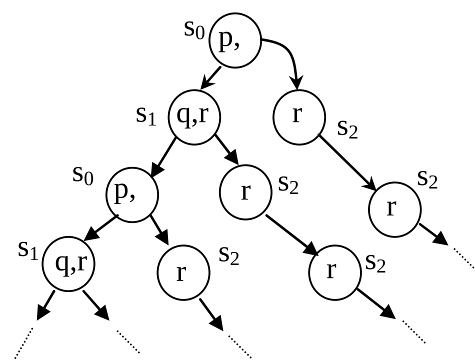
A CTL formula is evaluated on a Kripke model *M*. A path in *M* from a state s is an infinite sequence of states from definition $\pi = [s_0, s_1, \ldots, s_{i-1}, s_i, s_{i+1}, \ldots]$ such that s_0=s and $(s_i, s_{i+1}) \in R$ holds for all $i \geq 0$. We write $(s_i, s_{i+1}) \subseteq \pi$ and $s_i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

The following definition represents semantics the CTL Kripke model.

Semantics definition of a CTL model checker [8]. Let $M = (S, R, \mathcal{F}: S \rightarrow 2^{AP})$ be a Kripke model for CTL. Given any *s* in *S*, we define if a CTL formula *f* holds in state *s*. We denote this by $(M, s) \models f$. 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 $p \in \mathcal{F}(s)$.
3. $(M, s) \models \neg f$ iff $(M, s) \not\models f$.
4. $(M, s) \models f_1 \wedge f_2$ iff $(M, s) \models f_1$ and $(M, s) \models f_2$.
5. $(M, s) \models f_1 \vee f_2$ iff $(M, s) \models f_1$ or $(M, s) \models f_2$.
6. $(M, s) \models f_1 \rightarrow f_2$ iff $(M, s) \not\models f_1$ or $(M, s) \models f_2$.
7. $(M, s) \models AX\, f$ iff for all s_1 such that $(s,s_1) \in R$, $(M, s_1) \models f$.
8. $(M, s) \models EX\, f$ iff for some s_1 such that $s \rightarrow s_1$, $(M, s_1) \models f$.
9. $(M, s) \models AG\, f$ holds iff for all paths $[s_0, s_1, s_2, \ldots]$, where $s_0 = s$, and all s_i along the path, $(M, s_i) \models f$.
10. $(M, s) \models EG\, f$ holds iff there is a paths $[s_0, s_1, s_2, \ldots]$, where $s_0 = s$, and all s_i along the path, $(M, s_i) \models f$.
11. $(M, s) \models AF\, f$ holds iff for all paths $[s_0, s_1, s_2, \ldots]$, where $s_0 = s$, there is some s_i in the path such that $(M, s_i) \models f$.
12. $(M, s) \models EF\, f$ holds iff there is a paths $[s_0, s_1, s_2, \ldots]$, where $s_i = s$, and for some s_i along the path $(M, s_i) \models f$.
13. $(M, s) \models A\,[f_1 \cup f_2]$ holds iff for all paths $[s_0, s_1, s_2, \ldots]$, where $s_0 = s$, the path satisfies $f_1 \cup f_2$. For example there is some s_i along the path, such that $(M,s_i) \models f_2$ and for each $j < i$, $(M,s_j) \models f_1$.
14. $(M, s) \models E\,[f_1 \cup f_2]$ holds iff for all *paths* $[s_0, s_1, s_2, \ldots]$, where $s_0 = s$, the path satisfies $f_1 \cup f_2$. For example there is some s_i along the path, such that $(M,s_i) \models f_2$ and for each $j < i$, $(M,s_j) \models f_1$.

The interpreting tree from the transition graph is represented in the below graph:



Without a detailed declaration we presupposes all the five formulae CTL presented in the contextually as the by-path satisfied. Toward example, we consider to update a model Kripke with CTL formulae, beginning from the fact as *f* is satisfied.

A CTL Kripke model give which satisfies the CTL formulae, we considered as a model what can be updated satisfied formulae gives. In the beginning is shall give a general definition of update the CTL model.

The next definition presents just a prerequisite for requirement the update of the CTL model and not tells how the update be as well to be directional. In substance, as in the traditional knowledge is based the update [9], we consider an update of CTL model supposed a minimal change principle.

More, this change the minimum be as well to is defined as a process based on some operational process so concrete algorithms for the update of CTL model to can to be implemented. To this end, we fall consider five the primitive operations on the CTL model which delivers a base for all updates of complex models CTL.

Definition CTL Model Update. Given a CTL Kripke model $M = (S, R, \mathcal{F})$ and a CTL formula f. An update of $\mathcal{M} = (M, s_0)$, where $s_0 \in S$ with f is a CTL Kripke model $M' = (S', R', \mathcal{F}')$ such that $\mathcal{M}' = (M', s_0')$,

$(M', s_0') \models f$ where $s_0' \in S'$. We use $Update(\mathcal{M}, f)$ to denote the result $\mathcal{M}'$ and $Update(\mathcal{M}, f) = \mathcal{M}$ if $\mathcal{M} \models f$.

The figure presented as has been stated above explanation this definition.

III. Primitive Operations

P_1. Add an only relation. Given M = (S, R, $\mathcal{F}$), its updated model M' = (S', R', $\mathcal{F}$') is the result of M having only added one new relation. That is S'= S, $\mathcal{F}' = \mathcal{F}$, and $R' = R \cup \{(s_{addrel}, s_{addrel2})\}$ where $(s_{addrel}, s_{addrel2}) \notin R$ for one pair of $s_{addrel}, s_{addrel2} \in S$.

P_2. Remove an only relation. Given M = (S, R, $\mathcal{F}$), its updated model M' = (S', R', $\mathcal{F}$') is the result of M having only removed one existing relation. That is, S'= S, $\mathcal{F}' = \mathcal{F}$, and $R' = R - \{(s_{remrel}, s_{remrel2})\}$ where $(s_{remrel}, s_{remrel2}) \in R$ for one pair of $s_{remrel}, s_{remrel2} \in S$.

P_3. Substitute a state and it's associated with an only relations. Given M = (S, R, $\mathcal{F}$), its updated model M' = (S', R', $\mathcal{F}$') is the result of M having only substituted one existing state and its associated relations. That is, $S' = S[s/s_{substate}]$. S' is the set of states where one state s in S is substituted by $s_{substate}$, $R' = R \cup \{(s_i, s_{substate}), (s_{substate}, s_j) \mid$ for some $s_i, s_j \in S\} - \{(s_i,s),(s,s_j) \mid (s_i,s),(s,s_j) \in R\}$ and $\mathcal{F}'(s) = \mathcal{F}(s)$ for all $s \in S \cap S'$ and $\mathcal{F}'(s_{substate}) = \tau(s_{substate})$, where τ is a truth assignment on $s_{substate}$.

P_4. Add a state and it's associated with an only relations. Given M = (S, R, $\mathcal{F}$), its updated model M' = (S', R', $\mathcal{F}$') is the result of M having only added one new state and its associated relations. That is, $S' = S \cup \{s_{addstate}\}$. S' is the set of states where one state s in S is added by $s_{addstate}$, $R' = R \cup \{(s_i, s_{addstate}), (s_{addstate}, s_j) \mid s_i, s_j \in S \cap S'\}$ and $\mathcal{F}'(s) = \mathcal{F}(s)$ for all $s \in S \cap S'$ and $\mathcal{F}'(s_{addstate}) = \tau(s_{addstate})$, where τ is a truth assignment on $s_{addstate}$.

P_5. Remove a state and its associated with an only relations. Given M = (S,R, $\mathcal{F}$), its updated model M' = (S',R', $\mathcal{F}$') is the result of M having only added one existing state and its associated relations. That is, $S' = S - \{s_{remstate} \mid s_{remstate} \in S\}$. S' is the set of states where one state s in S is removed by $s_{remstate}$, $R' = R - \{(s_i, s_{remstate}), (s_{remstate}, s_j) \mid$ for some $s_i, s_j \in S\}$ and $\mathcal{F}'(s) = \mathcal{F}(s)$ for all $s \in S \cap S'$.

We present hereinbefore five operations atomic to all change on CTL model can to be in terms of with five these operation. Can to be argued that P_3 can to be in terms with P_4 and P_5. Anyway, we treat state substitution differently from a combination of state addition and state removed. That is the context, whenever substitute it a state is needed, applied P_3 directly more than P_4 followed of P_5. This thing will simplify definition of minimal change of the CTL model.

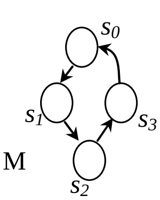
For defined the criteria of minimal change of update CTL model, we need to consider the changes for both states and relations for the underlying CTL models. We achieve these specifying the differences among states and relations on the models CTL using the primitive operations. Given any two sets X and Y, symmetrical difference among X and Y be denoted as $Diff(X, Y) = (X - Y) \cup (Y - X)$. Given two CTL models, $M = (S, R, \mathcal{F})$, and $M' = (S', R', \mathcal{F}')$ for each primitive operation P_i with i = 1,…, 5, $Diff\ P_i(M,M')$ indicates the differences between one of two the CTL models where M' is a resulting model from M, that make clear this difference between this operations the types may occur. Since P_1 and P_2 only changes relations, we define $DiffP_i(M,M') = (R - R') \cup (R'-R)$ where $i = 1, 2$. For the operations P_3, P_4 and P_5, but then, we define $DiffP_i(M,M') = (S-S') \cup (S'-S)$ with i = 3, 4, 5. Although any state changes caused by P_3, P_4, P_5 will imply also correspondence changes on relations, we only count the modifications states and take the state change as the primitive factor in order to measure difference between and M'. For the operations P_3, we should consider the case which a state is substituted with a new state. For this is necessary difference between these two states to be minimal before the condition of formulated update. In the next place is specified $Diff(M,M') = (DiffP_1(M,M'), DiffP_2(M,M'), DiffP_3(M,M'), DiffP_4(M,M'), DiffP_5(M,M'))$.

Let M, M_1, M_2 be three CTL models. We denote $Diff(M,M_1) \preccurlyeq Diff(M,M_2)$ iff for each i with $i = 1,\ldots, 5$, $DiffP_i(M,M_1) \subseteq DiffP_i(M,M_2)$; or we denote $Diff(M,M_1) \preccurlyeq Diff(M,M_2)$ iff $DiffP_i(M,M_1) \subseteq DiffP_i(M,M_2)$ for $i = 1, 2, 4, 5$, and $|DiffP_3(M,M_1)| = |DiffP_3(M,M_2)|$ implies for each state s in M substituted by s_1 and s_2 in M_1 and M_2 respectively, $Diff(s, s_1) \subseteq Diff(s, s_2)$.

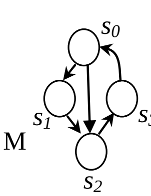
The Definition of Admissible Update is give by assertion: Given a CTL Kripke model $M = (S, R, \mathcal{F})$, $\mathcal{M} = (M, s_0)$, where $s_0 \in S$, and a CTL formula f, $Update(\mathcal{M}, f)$ is called admissible if the following conditions hold:

(1) $Update(\mathcal{M}, f) = (M', s_0') \models f$ where $M' = (S', R', \mathcal{F}')$ and $s_0' \in S'$;

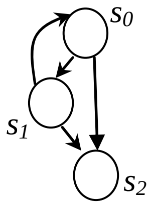
(2) There does not exist another resulting model $M'' = (S'', R'', \mathcal{F}'')$ and $s_0'' \in S''$; such that $(M', s_0'') \models f$ and $M'' <_M M'$.



M



M



M_2

In example as has been stated above is presented an illustration of minimal change rules.

We denote $M_1 <_M M_2$ if $M_1 \leq_M M_2$ and $M_2 \nleq_M M_1$. Given tree CTL Kripke models M, M_1, M_2 where M_1 and M_2 are obtained from M by applying P_1, P_2, P_3, P_4, P_5 operations. M_1 is closer to M as M_2, denoted as $M_1 \leq_M M_2$, iff $Diff(M,M_1) \preccurlyeq Diff(M,M_2)$.

IV. Characterizations of Semantic

From definition as has been stated above which enunciate the admissible update given for the CTL model, observed as for the CTL model Kripke give M and a formula *f*, there we can be many admissible updates satisfy *f*, waves some updates are simpler than others. In this part, are shall present a variety of characterizations semantic the CTL model updated that present the solution possible achieved the admissible updates under certain conditions. At large in order realization 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 model characterizations also gamble an essential role for simplified the implementation of update CTL model.

For beginner we shall return to definition of CTL Model Update. 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($\mathcal{M}$,*f*)
/* $\mathcal{M} = (M, s_0) \nvDash f$. Update $\mathcal{M}$ to satisfy *f*. */
Input: $M = (S, R, \mathcal{F})$, $\mathcal{M} = (M, s_0)$, where $s_0 \in S$ and $\mathcal{M} \nvDash f$;
Output: $M' = (S', R', \mathcal{F}')$, $\mathcal{M}' = (M', s_0')$, $s_0' \in S'$, $\mathcal{M}' \vDash f'$;
{ *case*
f is $\perp$ *: return* {M};
f is atomic p : return {$Update_p(\mathcal{M}, p)$};
f is $\neg f_1$ *: return* {$Update_{\neg}(\mathcal{M}, f_1)$};
f is $f_1 \vee f_2$ *:return*{*CTLUpdate*($\mathcal{M}, f_1$) *or* *CTLUpdate*($\mathcal{M}, f_2$)};
f is $f_1 \wedge f_2$*: return* {$Update_{\wedge}(\mathcal{M}, f_1, f_2)$};
f is EXf_1*: return* {$Update_{EX}(\mathcal{M}, f_1)$};
f is AXf_1*: return* {$Update_{AX}(\mathcal{M}, f_1)$};
f is EFf_1*: return* {$Update_{EF}(\mathcal{M}, f_1)$};
f is AFf_1*: return* {$Update_{AF}(\mathcal{M}, f_1)$};
f is EGf_1*: return* {$Update_{EG}(\mathcal{M}, f_1)$};
f is AGf1: return {$Update_{AG}(\mathcal{M}, f_1)$};
f is $E(f_1 \cup f_2)$*: return* {$Update_{EU}(\mathcal{M}, f_1, f_2)$};
f is $A(f_1 \cup f_2)$*: return* {$Update_{AU}(\mathcal{M}, f_1, f_2)$};
}

function $Update_p(\mathcal{M}, p)$;
{ /* $\mathcal{M} \nvDash p$. Update s_0 to satisfy p */
P_3 *is applied:*
1. $s_0' := s_0 \cup \{p\}$;
2. $S' := S - \{s_0\} \cup \{s_0'\}$;
3. $R' := R - \{(s_0,s_i) \mid$ *for any* $s_i = succ(s_0)\} \cup \{(s_0',s_i) \mid$ *for any* $s_i = succ(s_0)\} - \{(s_j,s_0) \mid$ *for any* $s_j = pre(s_0)\} \cup \{(s_j,s_0') \mid$ *for any* s_j $pre(s_0)\}$;
4. $\mathcal{F}' : S' \rightarrow 2^{AP}$, *where for any* $s \in S'$, *if* $s \in S$ *then* $\mathcal{F}'(s) = \mathcal{F}(s)$; *else* $s = s_0'$, *and* $\mathcal{F}(s_0') := \tau(s_0')$ *is the truth assignment related to* s_0';
5. $\mathcal{M}' := (M', s_0')$, *where* $M' = (S', R', \mathcal{F}')$ *and* $s_0' \in S'$;
6. *return* { $\mathcal{M}'$};
}

From definition of Admissible Update as has been stated above which enunciate the admissible update given for the CTL model, observed as for the CTL model Kripke give *M* and a formula *f*, there we can be many admissible updates satisfy *f*, waves some updates are simpler than others. In this part, are shall present a variety of characterizations semantic the CTL model updated that present the solution possible achieved the admissible updates under certain conditions. At large in order realization 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 model characterizations also gamble an essential role for simplified the implementation of update CTL model.

We enounce the first theorem which provides two cases where admissible CTL model update results can be achieved for formula EX *f*. Let $M = (S, R, \mathcal{F})$, be a Kripke model and s_0 be an initial state in *S* and $\mathcal{M} = (M, s_0) \nvDash EX f$, where *f* is a propositional formula. Then an admissible updated model $\mathcal{M}' = Update(\mathcal{M}, EX f)$ can be obtained by doing one of the following operations:

1. P_3 is applied to any $succ(s_0)$ once to substitute it with a new state $s^* \vDash f$ and $Diff(succ(s_0), s^*)$ to be minimal, or P_4 is applied one time after adding a new state $s^* \vDash f$ and a new relation (s_0, s^*);
2. If there exists some $s_i \in S$ such that $s_i \vDash f$ and $s_i \neq succ(s_0)$, P_1 is applied one time to add a new relation (s_0, s_i).

Function $Update_{EX}(\mathcal{M}, f)$
/* $\mathcal{M} \nvDash EX f$. Update $\mathcal{M}$ to satisfy *EX f* */

{
1. *select state s_1=succ(s_0)such that $\mathcal{M}_1$=(M,s_1) $\not\models$ f;*
2. *Update the state s_1 with minimal change rule:*
(1) *Applying P_3: return {CTLUpdate($\mathcal{M}_1$,f)};*
(2) *Applying P_4:*
then S':=$S\cup s_1$, where $s\in S'$ and $s\notin S$;
R':=$R\cup\{(pre(s_1),s_i),(s_1,succ(s_1))\}$, where prec($s_1$),succ($s_1$)$\in S\cap S'$;
$\mathcal{F}'$:$S'\rightarrow 2^{AP}$, where $\forall s\in S'$, if $s\in S$, then $\mathcal{F}'(s)$:= $\mathcal{F}(s)$, else $\mathcal{F}'(s_1)$:= $\tau(s_1)$, where τ is a truth assignment on s_1.
return { $\mathcal{M}'$};
}

In the first case is defined as can select a state s_0 and which is a successor and substitute it with a new what state satisfies *f*. For example we could apply one time the primitive operation P_3, or adding a new which state satisfies *f* as successors of s_0. For example we could apply one time the primitive operation P_4. The second case indicates that if there am some states and from *S* which already satisfies *f*, then it is sufficient in order to simplified the 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 shall carry the show, to new the CTL models which satisfy *Ex f*. Theorem show thus presents the new what models the by-paths also minimum keeping the original CTL model.

The following theorem in first case considers a special form of path π where the first *i* states starting from s_0 already satisfy formula *f*. Under this situation, we can simply cut off the path. For example we can apply one time P_2 or P_5 to disconnect all other states not satisfying *f*.

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

1. if for all $s < s_i$ in π, $s \models f$ but $s_i \not\models f$, P_2 is applied to remove relation (s_{i-1}, s_i), or P_5 is applied to remove s_i and its associated relations;
2. P_3 is applied to all states *s* in π not satisfying *f* to substitute s with $s^* \models f$ and $Diff(s, s^*)$ to be minimal.

Function Update$_{AG}$($\mathcal{M}$, f)
/ $\mathcal{M}\not\models$ AG f. Update $\mathcal{M}$ to satisfy AG f */*
{
if M_0 = (M,s_0) $\not\models$ f, then P_3 is applied to s_0 such that
$\mathcal{M}'$ =CTLUpdate($\mathcal{M}_0$, f);
else
{ 1. *select a path π = [s_0 ,s_1,…], where $\exists s_i\in\pi$ such that $\mathcal{M}_i$ =(M,s_i)$\not\models$ f;*
2. *select a state $s_i\in\pi$ such that $\nexists$ s_j<s_i with (M,s_j) $\not\models$ f then*
(1) *Applying P_2 to remove relation (pre(s_i),s_i), then*
S':=S;
R':=R-{(pre(s_i),s_i)};
$\mathcal{F}'$= $\mathcal{F}$, since is removed a only relation;
or
(2) *Applying P_5 to remove state s_i, its associated relation, then*
S':=S-{s_i}; R':=R-{(pre(s_i),s_i),(s_i,succ(s_i))} where if associated relations of s_i ;
$\mathcal{F}'$:$S'\rightarrow 2^{AP}$, since $S'\subseteq S$, $\forall s\in S'$,such that $\mathcal{F}'(s)$:= $\mathcal{F}(s)$, is removed a only relation;
or
(3) *Applying P_3 $\mathcal{M}'$=CTLUpdate($\mathcal{M}_i$,f);*
}
if $\mathcal{M}'\models$ AG f then return $\mathcal{M}'$;
else return { Update$_{AG}$($\mathcal{M}'$,f)};
}

The implementation used hereinbefore can be formulated in a short form as follows:

Function Update$_{AG}$($\mathcal{M}$, f)
/ $\mathcal{M}\not\models$ AG f. Update $\mathcal{M}$ to satisfy AG f */*
{
1. *select a path π = [s_0 ,s_1,…], where $s_i\in\pi$ and $\mathcal{M}_i$ =(M,s_i)$\not\models$ f;*
2. *$\mathcal{M}'$=CTLUpdate($\mathcal{M}_i$,f);*
3. *if $\mathcal{M}'\models$ AG f then return $\mathcal{M}'$;*
else return Update$_{AG}$($\mathcal{M}'$,f);
}

Next theorem characterizes three the typical situations for the update with formulate *EG f*. In substance, this theorem says as formulated marked *EG f* so that is true, we in the beginning am due to select a path, then we can do a new path based on this path as the all states from the new his path satisfies *f* (Case1), arrange the path from the state whence all previous his state satisfies *f* (Case 2) or simply to don't replaces all states satisfying *f* in the new property what path satisfy *f* (Case 3). Proof of this theorems as the resulting the model is presented the in the work [10] whereat resulted the models from these what operations admissible by-paths.

This theorem is enounced here below.

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

1. if for all $s' \in \pi$ such that $s'\not\models f$, there exist s_i, $s_j \in \pi$ satisfying $s_i < s' < s_j$ and $s_i \models f$ and $s_j \models f$, then P_1 is applied to add a relation (s_i, s_j), or P_4 is applied to add a state $s^* \models f$ and new relations (s_i, s^*) and (s^*, s_j);
2. if there exists some $s_i \in \pi$ with $i > 1$ such that for all

$s' < s_i$, $s' \models f$ and $s_i \not\models f$, then P_2 is applied to remove relation (s_{i-1}, s_i), or P_5 is applied to remove state s_i and its associated relations;

3. for all $s' \in \pi$, $s' \not\models f$, then P_3 is applied to substitute all s' with new state $s^* \models f$ and $Diff(s, s^*)$ to be minimal.

The short implementation is:

Function $Update_{EG}(\mathcal{M}, f)$
/* $\mathcal{M} \not\models EG\, f$. *Update* $\mathcal{M}$ *to satisfy* $EG\, f$ */
{
1. *select a path* $\pi = [s_0, s_1, \ldots]$, *in* $\mathcal{M}$;
2. *select a state* $s_i \in \pi$ *such that* $\mathcal{M}_i = (M, s_i) \not\models f$
3. $\mathcal{M}' = CTLUpdate(\mathcal{M}_i, f)$;
4. *if* $\mathcal{M}' \models EG\, f$ *then return* $\mathcal{M}'$;
   else return $Update_{EG}(\mathcal{M}', f)$;

}

V. MICROWAVE OVEN EXAMPLE

In this section we present a study of case wherewith is illustrated the features of CTL model updated approached.

As example we shall present a scenario for a microwave oven [3]. Presuppose that we have a microwave oven which including in first cases a process for normal heat and in second case for a faulty process. In first case for the normal heat process doesn’t shall appear the errors, so the oven is closed and the feed shall be heat. For the second process the faulty process, when the oven doesn’t shall warm the feed after oven is start. The aim of the model is where the faulty process is. The objective of model updating, on other word, is to correct the original model which contains the faulty process. Starting from the original CTL Kripke structure for the microwave oven presented in the figure 2 with seven states of the system denoted with $s_1, s_2, \ldots, s_7$.

The Kripke model has seven the states and the propositional variables are from the set {*Start*, *Close*, *Heat*, *Error*}. *Start* represented the *start oven*, *Close* represent the *close door* to oven, *Heat* is *heat* the feed or *warm up* and *Error* means occur some *error*.

The formal definition of the Kripke structure of the microwave oven is given by: $M = (S, R, \mathcal{F})$, where $S=\{s_1,s_2,\ldots,s_7\}$, $R=\{(s_1,s_2), (s_2,s_5), (s_5,s_2), (s_5,s_3), (s_3,s_1), (s_1,s_3), (s_3,s_6), (s_6,s_7), (s_7,s_4), (s_4,s_4), (s_4,s_3), (s_4,s_1)\}$, AP={*Start*, *Close*, *Heat*, *Error*}, $\mathcal{F}$ assigns state s_1 in M with *not start*, *not close*, *not heat* and *not error*, that is set {¬*Start*, ¬ *Close*, ¬ *Heat*, ¬ *Error*}. $\mathcal{F}$ assigns state s_2 in M with {*Start*, ¬*Close*, ¬ *Heat*, *Error*}, the state s_3 in M with {¬ *Start*, *Close*, ¬ *Heat*, ¬ *Error*}, the state s_4 in M with {¬ *Start*, *Close*, *Heat*, ¬ *Error*}, the state s_5 in M with {*Start*, *Close*, ¬ *Heat*, *Error*}, the state s_6 in M with {*Start*, *Close*, ¬ *Heat*, ¬ *Error*}and the state s_7 in M with {*Start*, *Close*, *Heat*, ¬ *Error*}.

The model is shown hereinbefore:

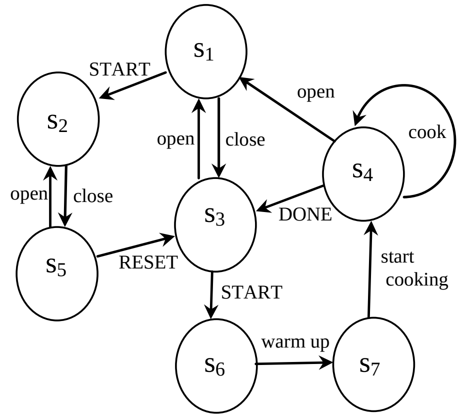


Fig. 2 The CTL Kripke structure of a microwave oven

In this figure *START* represented the *start oven*, *Open and Close* represent the *open door* and *close the door*, *RESET* is for a new initialization and *DONE* represented the done heat.

The faulty process from this graph is the path $s_1 \rightarrow s_2 \rightarrow s_5 \rightarrow s_3$. The interpretation is following. In first is start oven $\{s_1, s_2\}$. In the state s_2 we observed that have *not close* (that is the door don't is close) and the heating is out of order and it pointed some error. Is passed from the state s_2 in the state s_5 where the door oven shall be closed. In the state s_5 have error and the heating don’t started so is shall done reset for the reestablishment. That is, from s_5 is passed to the state s_3. Is can observed that the process with normal heat in the case view from the original CTL Kripke structure through $s_1 \rightarrow s_3 \rightarrow s_6 \rightarrow s_7 \rightarrow s_4$.

Noticed that this model don't satisfies the property $f = \neg EF(Start \wedge EG \neg Heat)$ [4]. The CTL model updated brings a minimum modification of the Kripke model which satisfies the property *f*. Firstly, is shall analyzed *f* in $AG(\neg (Start \wedge EG \neg Heat))$ for remove the symbol ¬. The translation is doing with the function *Update*¬. Then is necessary to check each state whether it satisfies $\neg (Start \wedge EG \neg Heat)$. This string shall be parsed before it is checked. Selecting the $EG\neg Heat$ to feed through the model checking function for *EG*.

In this model, any path has any state with ¬Heat is selected. Here are shall search the paths in the form $[s_1,s_2,s_5,s_3,s_1,\ldots]$ and $[s_1,s_3,s_1,\ldots]$ which represents the connected components loops satisfy $EG\neg Heat$. Then is identified all states with *Start*, these are $\{s_2,s_5,s_6,s_7\}$. Then is selected the states with *Start* and ¬*Heat*, these are $\{s_2,s_5\}$. Because the $AG(\neg(Start \wedge EG \neg Heat))$ formula identifies the which model has no both states *Start* and ¬*Heat*, is necessary an execution with states s_2 and s_5 so is shall apply the updated model. For the execution of $Update_{AG}$ function we can used three minimum updates: (case 1) applying P_2 where remove the connection (s_1, s_2); (case 2) applying the property P_5 where removed the state s_2 and the associate relations therewith s_1,s_2), (s_2,s_5) and (s_5,s_2); and (case 3) applying P_3 on the state s_2 and s_5. Taking on case three the first translate will be from $\neg(Start \wedge EG \neg Heat)$ to $\neg Start \wedge \neg EG \neg Heat$, therefore s_2 and s_5 are updated with any ¬*Start* or

$\neg EG\neg Heat$ by the main function *CTLUpdate* what is dealt with $\vee$ and with the $Update_{\neg}$ function.

In other words, the new states of s_2 and s_5 shall be denoting with s_2' and s_5'. The $Update_{AG}(\mathcal{M}, \neg(Start \wedge EG\neg Heat))$ function which calls the main function $CTLUpdate(\mathcal{M}, \neg\text{Start})$ or $CTLUpdate(\mathcal{M}, \neg EG\neg Heat)$ for the case $f_1 \vee f_2$. We choose the $\neg Start$ because this is simplest than $\neg EG\neg Heat$. In this case is necessary to update the atomic proposition *Start* in states s_2 and s_5 of path π with $\neg Start$ instead, then no states on path π have the specification EF $(Start \wedge EG\ \neg Heat)$. That is $\mathcal{M}' = (M', s_1) \models \neg EF\ (Start \wedge EG\ \neg Heat)$. The resulting model is presented in figure 3.

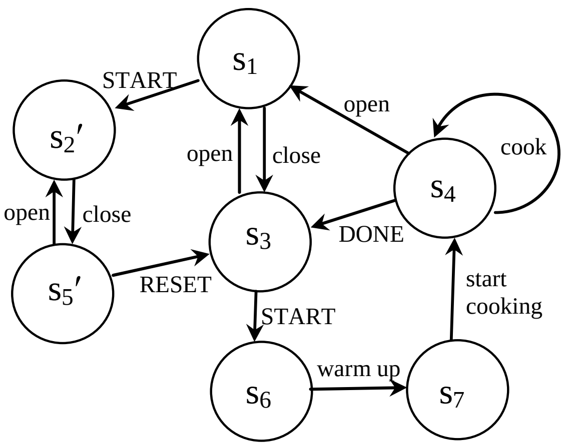


Fig. 3 The updated model for a microwave oven with primitive P3

Where s_2' is set $\{\neg Start, \neg\ Close, \neg\ Heat, Error\}$ and the state s_5' is set $\{\neg\ Start, Close, \neg\ Heat, Error\}$.

The algorithm will generate one of the three resulting models without specific indication, because criteria used they are all minimally changed from the original model.

VI. Conclusion

In this paper, we presented a formal approach for the update the CTL models. Specifying five one primitive on the CTL Kripke models [10], the definite minimal change criteria arrived at the CTL model updated. Also in this paper presented semantics and the calculating property of approach used. Base were developed a CTL model update algorithm and implemented a system prototype of system improved an update of CTL model.

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] E.Jr. Clarke, O. Grumberg, and D.A. Peled, “Model Checking”, MIT Press, Cambridge, 2000.

[4] H. Harris and M. Ryan, ”Theoretical foundations of updating systems”, in the Prodeedings 18th IEEE pp. 291.298, 2003.

[5] J. Wing and M. Vaziri-Farahani, ”A case study in model checking software”, in Proceedings 3rd ACM SIGSOFT, 1995.

[6] K. McMillan and N. Amla, ”Automatic abstraction without counterexamples”, in Cadence Berkeley Labs, Cadence Design Systems, 2002.

[7] L. Cacovean, E.M. Popa, C.I. Brumar, B.A. Brumar, ”Algebraic Algorithm of CTL Model Checker”, in Proceedings 8th WSEAS International Conference on MMACTEE ’06, București, România, 2006.

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

[9] M. Winslett, ”Updating Logical Databases”, Cambridge University Press, 1990.

[10] Yulin Ding, Yan Zhang, ”CTL Model Update: Semantics, Computations and Implementation”. ECAI 2006, Italy.