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 \( M = (S, R, E:S \rightarrow 2^{AP}) \) where \( S \) is a finite set of states, \( R \subseteq S \times S \) is a transition relation, \( E:S \rightarrow 2^{AP} \) 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 \( M=(S, E, P:AP \rightarrow 2^{S}) \) where \( S \) is a finite set of states also called nodes, \( E \) is a finite set of directed edges, and \( P \) represents proposition labelling function which labels each nodes with logical proposition. For each \( s \in S \), use the notation \( \text{succ}(s)=\{s'\in S \mid (s,s')\in E\} \). Each state in E must have at least one successor, that is \( \forall s \in S, \text{succ}(s)\neq \emptyset \). 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
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 $\neg f_1$, $f_1 \land f_2$, $f_1 \lor f_2$, $\exists x f_1$, $\forall x f_1$, $\text{EF } f_1$, $\text{AF } f_1$, $\text{AG } f_1$, and $\text{A}[f_1 \cup f_2]$.

The following definition represents semantics the CTL Kripke model.

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

$$f::= \top | \bot | \neg f_1 | f_1 \land f_2 | f_1 \lor f_2 | \Box f_1 | \Diamond f_1 | \text{AG } f_1 \text{ or } \text{EF } f_1 | A[f_1 \cup f_2] | E[f_1 \cup f_2]$$

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 $\pi = [s_0, s_1, ..., s_i, s_j, s_k, ...]$ such that $s_0 = s$, and $(s_i, s_{i+1}) \in R$ for all $i \geq 0$. We write $(s_0, s_1, ..., s_i, s_{i+1}, s_{i+2}, ...)$ 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 $\text{suc}(s)$ 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 $M = (S, R, \mathcal{F}: S \rightarrow 2^{\mathcal{AP}}$) 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 $p \in F(s)$.
3. $(M, s) \models \neg f$ iff $(M, s) \not\models f$.
4. $(M, s) \models f_1 \land f_2$ iff $(M, s) \models f_1$ and $(M, s) \models f_2$.
5. $(M, s) \models f_1 \lor f_2$ 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. $(M, s) \models \text{AX } f$ iff for all $s_j$ such that $(s, s_j) \in R$, $(M, s_j) \models f$.
8. $(M, s) \models \text{EX } f$ iff for some $s_1$ such that $s \rightarrow s_1$.
9. $(M, s) \models \text{AG } f$ holds iff for all paths $[s_0, s_1, s_2, ...]$, where $s_0 = s$, and all $s_j$ along the path, $(M, s_j) \models f$.
10. $(M, s) \models \text{EG } f$ holds iff there is a paths $[s_0, s_1, s_2, ...]$, where $s_0 = s$, and all $s_j$ along the path, $(M, s_j) \models f$.
11. $(M, s) \models \text{AF } f$ holds iff for all paths $[s_0, s_1, s_2, ...]$, where $s_0 = s$, there is some $s_i$ in the path such that $(M, s_i) \models f$.
12. $(M, s) \models \text{EF } f$ holds iff there is a paths $[s_0, s_1, s_2, ...]$, where $s_i = s$, and for some $s_j$ along the path $(M, s_j) \models f$.
13. $(M, s) \models \text{A } [f_1 \lor f_2]$ holds iff for all paths $[s_0, s_1, s_2, ...]$, where $s_0 = s$, the path satisfies $f_1 \lor f_2$. For example there is some $s_i$ along the path, such that $(M, s_i) \models f_2$ and for each $j < i$, $(M, s_j) \not\models f_1$.
14. $(M, s) \models \text{E } [f_1 \lor f_2]$ holds iff for all paths $[s_0, s_1, s_2, ...]$, where $s_0 = s$, the path satisfies $f_1 \lor f_2$. For example there is some $s_i$ along the path, such that $(M, s_i) \not\models f_2$ and for each $j < i$, $(M, s_j) \not\models f_1$.

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 give. 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 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 \( M = (S, R, F) \) and a CTL formula \( f \). An update of \( M' = (M, s_0) \), where \( s_0 \in S \) with \( f \) is a CTL Kripke model \( M' = (S', R', F') \) such that \( M' = (M', s_0) \),
\[(M', s_0) \models f \text{ where } s_0 \in S'. \]

We use \( \text{Update}(M, f) \) to denote the result \( M' \) and \( \text{Update}(M, f) = M \) if \( M \models f \).

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

III. PRIMITIVE OPERATIONS

P1. Add an only relation. Given \( M = (S, R, F) \), its updated model \( M' = (S', R', F') \) is the result of \( M \) having only added one new relation. That is \( S' = S, F' = F \), and \( R' = R \cup \{(s_{\text{addrel}}, s_{\text{addrel}2}) \mid (s_{\text{addrel}}, s_{\text{addrel}2}) \in R \} \) for one pair of \( s_{\text{addrel}}, s_{\text{addrel}2} \in S \).

P2. Remove an only relation. Given \( M = (S, R, F) \), its updated model \( M' = (S', R', F') \) is the result of \( M \) having only removed one existing relation. That is, \( S' = S, F' = F \), and \( R' = R \setminus \{(s_{\text{remrel}}, s_{\text{remrel}2}) \mid (s_{\text{remrel}}, s_{\text{remrel}2}) \in R \} \) for one pair of \( s_{\text{remrel}}, s_{\text{remrel}2} \in S \).

P3. Substitute a state and its associated with an only relations. Given \( M = (S, R, F) \), its updated model \( M' = (S', R', F') \) is the result of \( M \) having only substituted one existing state and its associated relations. That is, \( S' = S \setminus \{s_{\text{substate}}\} \). \( S' \) is the set of states where one state \( s \) in \( S \) is substituted by \( s_{\text{substate}} \), \( R' = R \setminus \{(s_{\text{remstate}}, s_{\text{remstate}}) \mid (s_{\text{remstate}}, s_{\text{remstate}}) \in R \} \) for some \( s_{\text{remstate}} \in S \), and \( F'(s) = F(s) \) for all \( s \in S' \) and \( F'(s_{\text{substate}}) = \tau(s_{\text{substate}}) \), where \( \tau \) is a truth assignment on \( s_{\text{substate}} \).

P4. Add a state and it’s associated with an only relations. Given \( M = (S, R, F) \), its updated model \( M' = (S', R', F') \) is the result of \( M \) having only added one new state and its associated relations. That is, \( S' = S \cup \{s_{\text{addstate}}\} \). \( S' \) is the set of states where one state \( s \) in \( S \) is added by \( s_{\text{addstate}} \), \( R' = R \cup \{(s_{\text{addrel}}, s_{\text{addrel}2}) \mid (s_{\text{addrel}}, s_{\text{addrel}2}) \in R \} \) for one pair of \( s_{\text{addrel}}, s_{\text{addrel}2} \in S \), and \( F'(s) = F(s) \) for all \( s \in S \cap S' \) and \( F'(s_{\text{addstate}}) = \tau(s_{\text{addstate}}) \), where \( \tau \) is a truth assignment on \( s_{\text{addstate}} \).

P5. Remove a state and its associated with an only relations. Given \( M = (S, R, F) \), its updated model \( M' = (S', R', F') \) is the result of \( M \) having only removed one existing state and its associated relations. That is, \( S' = S \setminus \{s_{\text{remstate}}\} \). \( S' \) is the set of states where one state \( s \) in \( S \) is removed by \( s_{\text{remstate}} \), \( R' = R \setminus \{(s_{\text{addrel}}, s_{\text{addrel}2}) \mid (s_{\text{addrel}}, s_{\text{addrel}2}) \in R \} \) for some \( s_{\text{remstate}} \in S \), and \( F'(s) = F(s) \) for all \( s \in S \setminus S' \).

We present here inbefore five operations atomic to all change on CTL model can to be in terms of with five these operation. Can to be argued that \( P1 \) can to be in terms with \( P4 \) and \( P5 \). 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 \( P3 \) directly more than \( P4 \) followed of \( P5 \). 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 \( \text{Diff}(X, Y) = (X - Y) \cup (Y - X) \). Given two CTL models, \( M = (S, R, \mathcal{F}) \), and \( M' = (S', R', \mathcal{F}') \) for each primitive operation \( P_i \) with \( i = 1, \ldots, 5 \), \( \text{Diff}_i(M, M') \) indicates the differences between one of two the CTL models where \( M' \) is a resulting model from \( M \), that make clear this difference between these operations the types may occur. Since \( P1 \) and \( P2 \) only changes, we define \( \text{Diff}_i(M, M') = (R - R') \cup (R' - R) \) where \( i = 1, 2 \). For the operations \( P3 \), \( P4 \), \( P5 \), then, we define \( \text{Diff}_i(M, M') = (S - S') \cup (S' - S) \) with \( i = 3, 4, 5 \). Although any state changes caused by \( P3 \), \( P4 \), \( P5 \) 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 \( P3 \), 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 \( \text{Diff}_i(M, M') = \text{Diff}_i(M, M), \text{Diff}_i(M, M'), \text{Diff}_i(M, M'), \text{Diff}_i(M, M'), \text{Diff}_i(M, M') \).

Let \( M, M_1, M_2 \) be three CTL models. We denote \( \text{Diff}(M, M_1) \subseteq \text{Diff}(M, M_2) \) iff for each \( i \) with \( i = 1, \ldots, 5 \), \( \text{Diff}_i(P_1(M, M_1)) \subseteq \text{Diff}_i(P_1(M, M_2)) \); or we denote \( \text{Diff}(M, M_1) \subseteq \text{Diff}(M, M_2) \) iff \( \text{Diff}_i(P_1(M, M_1)) \subseteq \text{Diff}_i(P_1(M, M_2)) \) for \( i = 1, 2, 4, 5 \), and \( |\text{Diff}_3(M, M_1)| = |\text{Diff}_3(M, M_2)| \) implies for each state \( s \) in \( M \) substituted by \( s_1 \) and \( s_2 \) in \( M_1 \) and \( M_2 \) respectively, \( \text{Diff}(s, s_1) \subseteq \text{Diff}(s, s_2) \).

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

1. \( \text{Update}(M, f) = (M', s_0) \models f \) where \( M' = (S', R', F') \) and \( s_0 \in S' \);
2. There does not exist another resulting model \( M'' = (S'', R'', F'') \) and \( s_0'' \in S'' \); such that \( (M', s_0'') \models f \) and \( M'' \neq M' \).
In example as has been stated above is presented an illustration of minimal change rules.

We denote $M_1 \preceq M_2$ if $M_1 \subseteq M_2$ and $M_2 \not\subseteq M_1$. 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 $M_1 \preceq M_2$, iff $Diff(M, M_1) \ll Diff(M, M_2)$.

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.

```plaintext
function Updatef(M, p);
{ /* M ≠ p. Update s0 to satisfy p */
P3 is applied:
1. $s'_0 := s_0 \cup \{p\};$
2. $S' := S-\{s_0\} \cup \{s'_0\};$
3. $R' := R - \{(s_0, s_0')\}$ for any $s_i \in succ(s_0)$; $\forall s_j \in pre(s_0)$ for any $s_j \in succ(s_0)$; 
4. $F' \cdot S' \rightarrow 2^{SP}$, where for any $s \in S'$,
   if $s \in S$ then $F'(s) = F(s)$;
   else $s = s'_0$, and $F(s'_0) := \tau(s_0)$ is the truth assignment related to $s_0'$;
5. $M' = (M', s'_0)$, where $M' = (S', R', F')$
   and
6. return \{ $M'$ \};
}
```

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 $M = (S, R, F)$ be a Kripke model and $s_0$ be an initial state in $S$ and $M = (M, s_0) \not\models EX f$, where $f$ is a propositional formula. Then an admissible updated model $M'$ = $Update(M, EX f)$ 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 $s^* \not\models f$ and $Diff(succ(s_0), s^*)$ to be minimal, or
2. If there exists some $s_i \in S$ such that $s_i \models f$ and $s_i \not\in succ(s_0)$, $P_1$ is applied one time to add a new relation $(s_i, s_0)$.

Function $Update_{EX}(M, f)$

```plaintext
/* M ≠ EX f. Update M to satisfy EX f */
```
1. select state \( s_1 = \text{succ}(s_0) \) such that \( M = (M, s_1) \not\models f \);
2. Update the state \( s_1 \) with minimal change rule:
   (1) Applying \( P_2 \): return \( \text{CTLUpdate}(M, f) \);
   (2) Applying \( P_3 \):
   - then \( S' := S \setminus s_1 \), where \( s \in S' \) and \( s \not\in S \);
   - \( R' := R \setminus \{ (\text{pre}(s_1), s_1), (s_1, \text{succ}(s_1)) \} \), where \( \text{pre}(s_1), \text{succ}(s_1) \in S \setminus S' \);
   - \( F' := F \setminus \{ s \} \), where \( s \in S' \), if \( s \not\in S \), then \( F' := \emptyset \).
   return \( \{ M \} \);
s' < s_i s' \models f \text{ and } s_i \neq f, \text{ then } P_2 \text{ is applied to remove relation } (s_i, s_j), \text{ or } P_2 \text{ is applied to remove state } s_i \text{ and its associated relations;}

3. for all s' \in \pi, s' \neq f, \text{ then } P_3 \text{ is applied to substitute all } s'
   \text{ with new state } s^* \models f \text{ and } Diff(s, s^*) \text{ to be minimal.}

The short implementation is:

\begin{verbatim}
Function UpdateEG(M, f)
  /* M \models EG f. Update M to satisfy EG f */
  
  1. select a path \pi = [s_0, s_1, ...], in M;
  2. select a state s_i \in \pi such that M = (M, s_i) \models f
  3. M' = CTLUpdate(M, f);
  4. if M' \models EG f then return M';
  else return UpdateEG(M, f);

V. MICROWAVE OVEN EXAMPLE

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

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, ..., 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, ..., s_7}, R={(s_1, s_2), (s_2, s_3), (s_3, s_4), (s_4, s_5), (s_5, s_6), (s_6, s_7)}, \mathcal{F} assigns state s_1 in M with not start, not close, not heat and not error, that is set \{\neg Start, \neg Close, \neg Heat, \neg Error\}. \mathcal{F} assigns state s_2 in M with \{Start, \neg Close, \neg Heat, \neg Error\}, the state s_3 in M with \{\neg Start, Close, \neg Heat, \neg Error\}, the state s_4 in M with \{Start, Close, \neg Heat, \neg Error\}, the state s_5 in M with \{Start, Close, Heat, \neg Error\}, the state s_6 in M with \{Start, Close, \neg Heat, Error\} and the state s_7 in M with \{Start, Close, Heat, Error\}.

The model is shown hereinbefore:

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_3 \rightarrow s_4. 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_3 where the door oven shall be closed. In the state s_3 have error and the heating don’t started so shall be done reset for the reestablishment. That is, from s_3 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_2 \rightarrow s_3 \rightarrow s_7. The interpretation is following. In first is start oven \{s_1, s_2\}. The translation is doing with the function Update_. Then is necessary to check each state whether it satisfies \neg (Start \land 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 \neg Heat is selected. Here are shall search the paths in the form \{s_1,s_2,s_3,s_4,s_5,...\} and \{s_1,s_2,s_3,...\} 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 \neg Heat, these are \{s_2,s_3\}. Because the AG\neg (Start \land EG \neg Heat) formula identifies the which model has no both states Start and \neg Heat, is necessary an execution with states s_2 and s_3 so is shall apply the updated model. For the execution of UpdateEG 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_2 where removed the state s_2 and the associate relations therewith (s_1, s_2), (s_2, s_3) and (s_3, s_2); and (case 3) applying P_3 on the state s_2 and s_3. Taking on case three the first translate will be from \neg (Start \land EG \neg Heat) to \neg (Start \land \neg EG \neg Heat), therefore s_2 and s_3 are updated with any \neg Start or
¬EG¬Heat by the main function CTLUpdate what is dealt with ∨ and with the Update_ function.

In other words, the new states of s2 and s5 shall be denoting with s2′ and s5′. The Update_ function which calls the main function CTLUpdate(M, ¬(Start ∧ EG¬Heat)) or CTLUpdate(M, ¬EG¬Heat) for the case f1 ∨ f2. We choose the ¬Start because this is simplest than ¬EG¬Heat. In this case is necessary to update the atomic proposition Start in states s2 and s5 of path π with ¬Start instead, then no states on path π have the specification EF (Start ∧ EG¬Heat). That is M′ = (M′, s1) ≠ ¬EF (Start ∧ EG¬Heat). The resulting model is presented in figure 3.

![Fig. 3 The updated model for a microwave oven with primitive P3](image)

Where s2′ is set {¬Start, ¬Close, ¬Heat, Error} and the state s5′ is set {¬Start, Close, ¬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