# ALGORITHM AND AN ELEVATOR CONTROL SYSTEM EXAMPLE FOR CTL MODEL UPDATE

Laura Florentina Cacovean
*Department of Computer Science, Lucian Blaga University of Sibiu, Faculty of Sciences*
*Str. Dr. Ion Ratiu 5-7, 550012, Sibiu, Romania*

Iulian Pah
*Department of Sociology, Babes-Bolyai University Cluj-Napoca*
*Bd. 21 decembrie 1989, no. 120-130, 400604, Cluj-Napoca, Romania*

Emil Marin Popa and Cristina Ioana Brumar
*Department of Computer Science, Lucian Blaga University of Sibiu, Faculty of Sciences*
*Str. Dr. Ion Ratiu 5-7, 550012, Sibiu, Romania*

Keywords: CTL Kripke model, model update, algorithm, directed graph, implementation.

Abstract: In this paper is presented an update of the Computational Tree Logic (CTL) model checker. The minimal modifications which appear represent the fundamental concept for model the dynamic system. In the paper we use five primitive operations discompose from the operation of a CTL update used already by (Baral, 2005) which presented their approach of knowledge updated on the structures of single agent S5 Kripke. Then we will define the criteria of minimal change for the CTL model update based on these primitive operations. In the final section of this paper are presented the steps of implement the CTL model updated and are described some details of algorithm implementation by applying the model update to the elevator control scenario. The paper (Ding, 2006) is base of results obtained.

## 1 INTRODUCTION

The verification tools o automated formal, such as model checkers, shows delivered diagnosis to provide through automatic error diagnosis in complex designs, examples in (Wing, 1995). The current state of the model checkers technique, as Symbolic Model Verification (SMV) example (Clarke, 2000), Cadence SMV (McMillan, 2002), uses SMV as specification language for both CTL (Computational Tree Logic) and LTL (Lineal Temporal Logic) model checking. Progressing update of the method of the model checkers, begun to employ a formal method for repair approximate error. Since model, checking can handle verification problems complex system and as it may, implemented via fast algorithms, it is quite natural to consider whether we can develop associated algorithms so that they can handle system modification as well. The idea of integrating model checking and automatic modification has been investigate in recent years. In work (Harris, 2003) the model checking is formalized often with an updating operator satisfied the axioms U1-U8 what represent the classical proposition knowledge of updated Katsuno-Mendelzon postulates for belief update (Baral, 2005). They discussed knowledge update and its minimal change, based on modal logic S5. Both the update of the knowledge base and the knowledge update are currently at the theoretical research stage. Their approach of knowledge update could integrate with model checking technology towards a more general automatic system modification. In this paper, we considered the problem of the update of CTL model from both theories. In substance, as the traditional knowledge



Florentina Cacovean L., Pah I., Marin Popa E. and Ioana Brumar C. (2008).
ALGORITHM AND AN ELEVATOR CONTROL SYSTEM EXAMPLE FOR CTL MODEL UPDATE.
In *Proceedings of the International Conference on e-Business*, pages 77-80
DOI: 10.5220/0001907800770080

based on the update (Winslett, 1990) consider an update of CTL model subdue a principle of minimum inferior change. More, this minimal change are defined be as well to is definite as a process based on of some operational processes which a concrete algorithm for the update of CTL model could be implemented. In the final section of this work, we present a study case where we shown how the system prototype (Ding, 2006) could be applied for the system modified.

## 2 SYNTAX AND SEMANTICS

CTL is a branching time temporal logic meaning that its formulas interpreted over all paths beginning in a given state of the Kripke structure. 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.

*Syntax definition of a CTL model checker* (Huth, 2000). A CTL has the following syntax given in Backus near form: $f ::\top \mid \bot \mid p \mid (\neg f_1) \mid f_1 \wedge f_2 \mid f_1 \vee f_2 \mid f_1 \subset f_2 \mid AX\, f_1 \mid EX\, f_1 \mid AG\, f_1 \mid EG\, f_1 \mid AF\, f_1 \mid EF\, f_1 \mid A[f_1 \cup f_2] \mid E[f_1 \cup f_2]$ where $\forall p \in AP$.

A CTL formula is evaluate 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 $\pi$ as $s_i < s_j$.

*Semantics definition of a CTL model checker (Huth, 2000)*. 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 $\models$ define by structural induction on all fourteen CTL formulas (Ding, 2006). We assume all the five formulas CTL presented in the contextually as the paths are satisfied. Be a CTL Kripke model which satisfies the CTL formulas and we considered as a model that can be updated satisfying given formulas. The minimal change should define, based on some operational process, a concrete algorithm for CTL model update that can be implemented.

The CTL update definition: Be a CTL Kripke model $M=(S,R,\mathcal{F})$ and a CTL formula $f$. An *update* of $M=(M,s_0)$, where $s_0 \in S$ with $f$ is a CTL Kripke model $M' = (S',R',\mathcal{F}')$ such that $M' = (M',s_0')$, $(M',s_0') \models f$ where $s_0' \in S'$. We use $Upd(M, f)$ to denote the result $M'$ and $Upd(M,f) = M$ if $M \models f$.

## 3 PRIMITIVE OPERATORS

$P_1$. 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$\cup\{(s_{add}, s_{add2})\}$ where $(s_{add}, s_{add2}) \notin R$ for one pair of $s_{add}, s_{add2} \in S$.

$P_2$. 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-$\{(s_{rem}, s_{rem2})\}$ where $(s_{rem}, s_{rem2}) \in R$ for one pair of $s_{rem}, s_{rem2} \in S$.

$P_3$. Substitute 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 substituted one existing state and its associated relations. That is, S' = S[s/$s_{subst}$], R' = R$\cup\{(s_i, s_{subst}), (s_{subst}, s_j) |$for some $s_i, s_j \in S\}$-$(s_i,s),(s,s_j) | (s_i,s),(s,s_j) \in R\}$ and $\mathcal{F}'(s) = \mathcal{F}(s)$ for all $s \in S \cap S'$ and $\mathcal{F}'(s_{subst}) = \tau(s_{subst})$, where $\tau$ is a truth assignment on $s_{subst}$.

$P_4$. Add a state and it 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 it associated relations. That is, S' = S$\cup\{s_{addst}\}$, R'=R$\cup\{(s_i, s_{addst}),(s_{addst}, s_j) | s_i, s_j \in S \cap S'\}$ and $\mathcal{F}'(s)=\mathcal{F}(s)$ for all $s \in S \cap S'$ and $\mathcal{F}'(s_{addst})=\tau(s_{addst})$, where $\tau$ is a truth assignment on $s_{addst}$.

$P_5$. Remove a state and it 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-$\{s_{remst} | s_{remst} \in S\}$, R'=R-$\{(s_i, s_{remst}),(s_{remst}, s_j)$ | for some $s_i, s_j \in S\}$ and $\mathcal{F}'(s) = \mathcal{F}(s)$ for all $s \in S \cap S'$.

All the changes on CTL model can be in terms of all five operations. It can be arguing $P_3$ can be defined in terms of $P_4$ and $P_5$. Anyway, we treat state substitution differently from a combination of state addition and state removed. That is the context, whenever it substitutes a state 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, it needs 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. Be any two sets X and Y, symmetrical difference among X and Y be denoted as $Diff(X, Y) = (X - Y) \cup (Y - X)$. Be 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 $P_5$, 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 substitute with a new state. For this is necessary difference between these two states to be minimal before the condition of formulated update. A formal algorithm for the proposed CTL model update approach is described in (Ding, 2006) and (Cacovean, 2007).

## 4 ELEVATOR EXAMPLE

In this section we present a study of case where it is illustrated the features of CTL model updated approaches.

As example, we shall present a scenario for an elevator control system. The designer analyzes the state-transition diagram for the only control transformation, Elevator Controller (*EC*), finds eight locked-state events (Gomma, 1993). These locked-state events occur because the *EC*, in most instances, takes one action and then awaits a response before moving on a new state. In fact, have only two event flow, *Up* and *Down Request*, when we denote with *Move* state when the request exist and is not a locked-state event. This event flows is not qualify because each of them can arrive any time a client presses a floor button or when the scheduler schedules an elevator. The remaining events can only arrive when the *EC* is expecting them.

We assume that we have an elevator system control which including in first case, a process for normal moving of lift cabin and in second case, for a faulty process. In first case for the normal moving the elevator cabin process don't appear with errors, so the door is closed and the passenger going up or down when the button is pressed. For the second process, the faulty process appears when the lift cabin isn't moving when the button is pressed for start the moving. The aim of the model is where the faulty process appears. The objective of model updating, on other word, is to correct the original model, which contains the faulty process. Starting from the original CTL structure for our propose *EC* system presented in the figure *1* with eight states denoted with $s_1, s_2, \ldots, s_7$, and $s_d$ state we added for checking if the elevator is required of another passenger.

The Kripke model has eight states and the propositional variables are from the set {*Start*, *Close*, *Move*, *Error*}. *Start (St)* represented the *start button* for start moving up or down the elevator, *Close (Cl)* represent the *close door* to the lift cabin, *Move (Mv)* is *moving up or down* the elevator and *Error (Er)* means occur some *error*.

The formal definition of the Kripke structure of *EC* is given by $M=(S,R,\mathcal{F})$, where $S=\{s_1,s_2,\ldots, s_7\}$, $R=\{(s_1,s_2), (s_2,s_3), (s_3,s_2), (s_3,s_4), (s_4,s_3), (s_4,s_5), (s_5,s_6), (s_6,s_7), (s_7,s_7), (s_7,s_4), (s_4,s_1), (s_1,s_d), (s_d,s_4), (s_d,s_1)\}$, $AP=\{St, Cl, Mv, Er\}$. The $\mathcal{F}$ assigns state $s_1$ in $M$ with *not start, not close, not move* and *not error,* write this as $\{\neg St, \neg Cl, \neg Mv, \neg Er\}$. State $s_2=\{St, \neg Cl, \neg Mv, Er\}$, $s_3=\{St, Cl, \neg Mv, Er\}$, $s_4=\{\neg St, Cl, \neg Mv, \neg Er\}$, $s_5=\{St, Cl, \neg Mv, \neg Er\}$, $s_6=\{St, Cl, Mv, \neg Er\}$ and $s_7=\{\neg St, Cl, Mv, \neg Er\}$.

The model shown hereinbefore:

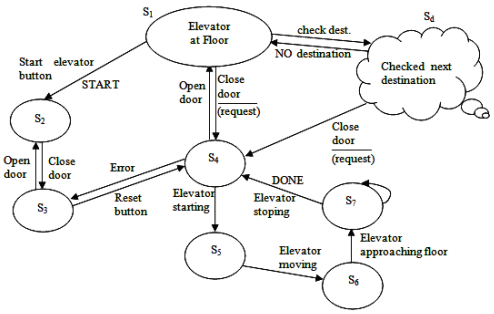


Figure 1: The CTL structure of Elevator Controller.

In figure *1 START* represented the *start elevator*, *Open and Close* represent the *open door* and *close the door*, *RESET* is for a new initialization and

*DONE* represents the done moving of elevator.

The faulty process from this graph is the path [$s_1$, $s_2$, $s_3$, $s_4$]. The interpretation is: start elevator {$s_1$,$s_2$}. In the state $s_2$ we observed that have *not close*, that is the door and it isn't close, and the moving is out of order and it pointed some error. Passed from the state $s_2$ in the state $s_3$ where the door elevator shall be close. In the state $s_3$ has error and the movement of elevator don't start so it shall push the reset button for the reestablishment. That is, from $s_3$ passed to the state $s_4$. Observed that the process with normal move in the case view from the original CTL Kripke structure through [$s_1$, $s_4$, $s_5$, $s_6$, $s_7$]. Noticed that this model do not satisfies the property $f = \neg EF(St \wedge EG \neg Mv)$ (Harris, 2003). The CTL model updated brings a minimum modification of the Kripke model which satisfies the property *f*. Firstly, it should analyze *f* in $AG(\neg(St \wedge EG \neg Mv))$ for remove the symbol ¬. The translation is doing with the function $Upd\neg$. Then is necessary to check each state whether it satisfies $\neg(St \wedge EG \neg Mv)$. This string shall be parsing before it is checked. Selecting the $EG \neg Mv$ to elevator through the model checking function for *EG*.

In this model, any path has any state when $\neg Mv$ is selected. Here are searched the paths in the form $[s_1,s_2,s_3,s_4,s_1,\ldots]$ and $[s_1,s_4,s_1,\ldots]$ which represent the connected components loops satisfy $EG \neg Mv$. Then are identified all states with *St*, these are $\{s_2,s_3,s_5,s_6\}$. Then are selected the states with *St* and $\neg Mv$, these are $\{s_2,s_3\}$. Because the $AG(\neg(St \wedge EG \neg Mv))$ formula identifies the model don't have the both states *St* and $\neg Mv$, is necessary an execution with states $s_2$ and $s_3$ so it should apply the updated model. From execution of $Upd_{AG}$ function, we shown the case in which applying $P_3$ on the state $s_2$ and $s_3$. The first translate will be from $\neg(St \wedge EG \neg Mv)$ to $\neg St \wedge \neg EG \neg Mv$, therefore $s_2$ and $s_3$ are updated with any $\neg St$ or $\neg EG \neg Mv$ by the main function *CTLUpd* what is dealt with $\vee$ and with the $Upd_{\neg}$ function. In other words, the new states of $s_2$ and $s_3$ shall be denoting with $s_2'$ and $s_3'$. The $Upd_{AG}(M,\neg(St \wedge EG \neg Mv))$ function calls the main function $CTLUpd(M,\neg St)$ or $CTLUpd(M, \neg EG \neg Mv)$ for the case $f_1 \vee f_2$. We choose the $\neg St$ because this is simplest than $\neg EG \neg Mv$. In this case is necessary to update the *St* in states $s_2$ and $s_3$ of path $\pi$ with $\neg St$ instead, then no states on path $\pi$ have the specification $EF(St \wedge EG \neg Mv)$. $M' = (M', s_1) \models \neg EF(St \wedge EG \neg Mv)$. The state $s_2'$ is set $\{\neg St, \neg Cl, \neg Mv, Er\}$ and the state $s_3'$ is set $\{\neg St, Cl, \neg Mv, Er\}$.

The algorithm will generate one of the three resulting models without specific indication, because criteria used are satisfying all the minimally changes from the original model. We consider that our elevator model propose is a model much more simple for understandable and for implemented, because we used a steps method to illustrate this elevator controller. In our case we used the CTL model checker update, verifying all five properties mentioned above which are accomplished also in our case of study.

# 5 CONCLUSIONS

In this paper, we presented a formal approach for the update the CTL models. Specification of five primitives on the CTL Kripke models (Ding, 2006), define the minimal change criteria of the CTL model updated. Also in this paper are presented semantics and the computing property of approach that we used. The proposed case study is an update principle of minimal change with maximal reachable states, which can significantly improve the update results in modification scenarios of complex system.

# REFERENCES


Baral C. and Y. Zhang, 2005, "*Knowledge updates: semantics and complexity issues*", Artificial Intelligence, 164, 209-243.

Cacovean L., Popa E.M., Brumar C.I., 2007, *Implementation of CTL Model Checker Update*, in Proc. 11th WSEAS Int. Conf., COMPUTERS, Greece

Clarke E.Jr., O. Grumberg, and D.A. Peled, 2000, "*Model Checking*", MIT Press, Cambridge

Gomma H., 1993, "*Software Design Methods for Concurrent and Real-Time Systems"*, Addision-Wesley Publishing Company, Reading Massachusetts

Harris H. and M. Ryan, 2003, "*Theoretical foundations of updating systems*", in Proc. 18th IEEE, 291-298.

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

McMillan K. and N. Amla, 2002, "*Automatic abstraction without counterexamples*", in Cadence Berkeley Labs.

Wing J. and M. Vaziri-Farahani, 1995, "*A case study in model checking software*", in Proc. 3 ACM SIGSOFT.

Winslett M., 1990, "Updating Logical Databases", Cambridge University Press, 1990.

Ding Y., Yan Zhang, 2006, "*CTL Model Update: Semantics, Computations and Implementation*". ECAI, Italy.