Toward the Revision of CTL Models through Kripke Modal Transition Systems

Paulo T. Guerra[1], Aline Andrade[2], and Renata Wassermann[1]

[1] University of São Paulo
{paulotgo,renata}@ime.usp.br
[2] Federal University of Bahia
aline@ufba.br

Abstract. In this paper we consider the problem of automatic repair of models in the context of system partial specification. This problem is a challenge involving theoretical and practical issues and the theory of belief revision is an alternative to give theoretical support to its solution. A Kripke structure is widely used to model systems, but it does not express partial information explicitly and a set of these structures might be required to represent several possibilities of behavior. A more general structure is the Kripke Modal Transition System (KMTS) which can specify systems with partial information and can be interpreted as a set of Kripke models. In this paper, we propose a framework for the repair of KMTS based on belief revision combined with model checking as an approach to revise sets of Kripke structures. We demonstrate the advantages of our approach, even with the existing restrictions in representing general sets of CTL models over the KMTS formalism.

1 Introduction

In the preliminary phases of system development it can be necessary to deal with incomplete information because generally not all requirements are already known. To specify an undetermined system it is desirable that models can represent partial information, such as possible behaviors. When a model does not explicitly express partial information, an alternative is to take several models as possible candidates for the system behavior. In both cases the models should be able to be formally verified and when a desired property is not satisfied the models must be repaired, ideally automatically.

We consider in this work the technique of model checking [1] for the verification of systems, particularly model checking over Kripke structures as CTL (Computation Tree Logic) models. A CTL model checker solves the decision problem: given a Kripke structure K, an initial state s_0 and a CTL formula φ, does K satisfy ϕ from s_0? ($K, s_0 \models \phi$?). When the property is not satisfied, the model checker shows a counter-example that can guide the repair of the model.

A CTL model does not express partial information explicitly. A set of these structures might be required to represent several possibilities of behavior. A more general Kripke structure is the Kripke Modal Transition System (KMTS) which is adequate for the specification of systems with partial information [2] and can

J. Iyoda and L. de Moura (Eds.): SBMF 2013, LNCS 8195, pp. 115–130, 2013.

be expanded in a set of CTL models. KMTS is interpreted over a 3-valued logic and can represent behavior that must or may occur. Model checking over KMTS [2], besides *true* and *false* values, can return *indefinite* meaning both values may be consistent.

The automatic repair of models is not straightforward and presents several challenges. The theory of belief revision [3] can be applied to this problem by considering models as beliefs [4,5]. In [4], a revision operation is defined to repair a set of CTL models when they are inconsistent with a desired property.

In this paper, we define the revision of a set of CTL models through the revision of a KMTS model when the KMTS model checking returns *false* or *indefinite*. We compare it with the revision of a set of CTL models as proposed in [5] and show the correspondence between these two approaches. Although there are some restrictions in representing a general set of Kripke models, we argue that the compact representation of KMTSs has advantages during the revision process. We show how revision can be implemented, using model checking through 3-valued model checking game as proposed in [6].

To the best of our knowledge this is the first work on revision of a set of CTL models through KMTS. In [7] the authors propose an algorithm to repair KMTS models based on primitive changes defined in [8]. Unlike our proposal this work is not based on belief revision and it does not make reference to any other theory of change and its context is abstract model checking, where a KMTS model represents an abstraction of a concrete Kripke structure as proposed by [6].

This paper is organized as follows. In Section 2, we briefly introduce CTL and the model revision approach. In Section 3 we introduce KMTS and how it is expanded into Kripke structures. In Section 4 we define revision of KMTS, its operations, the minimality criterion and proofs of its correctness. We describe how to implement KMTS revision in Section 5 based on a model checking game. Finally in Sections 6 and 7 we discuss this approach and conclude the paper.

2 Preliminaries

2.1 Computation Tree Logic

The computation tree logic (CTL) [9,10] is a temporal logic where the future is represented by a time-branching structure. CTL is suitable for example to describe properties over computer program and its different execution paths. The CTL syntax is given by the following Backus-Naur form:

$$\begin{aligned}\phi ::= &\top \mid \bot \mid p \mid (\neg\phi) \mid (\phi \vee \phi) \mid (\phi \wedge \phi) \mid (\phi \rightarrow \phi) \mid EX\phi \mid \\ &AX\phi \mid EF\phi \mid AF\phi \mid EG\phi \mid AG\phi \mid E[\phi U \phi] \mid A[\phi U \phi]\end{aligned}$$

where its temporal operators comprise: path quantifiers (E, “there is a path”, or A, “for all paths”); and state operators (X, “neXt state”, U, “Until”, G, “Globally in states” or F, “some Future state”).

The semantics for CTL is defined over a labelled transition system called Kripke structure. These structures are described by Definition 1.

Definition 1. *A Kripke structure is a tuple* $M = (AP, S, S_0, R, L)$ *where* AP *is a set of atomic propositions;* S *is a finite set of states,* $S_0 \subseteq S$ *is the set of initial states,* $R \subseteq S \times S$ *is transition relation over* S*, and* $L : S \to 2^{AP}$ *is a labelling function of truth assignment over states.*[1]

For convenience, we frequently refer to Kripke structures as CTL models.

2.2 CTL Model Revision

Guerra and Wassermann [4,5] propose a model repair framework using principles of belief revision theory [3]. Belief revision deals with how to rationally adapt dynamic beliefs set in order to incorporate new information, even if it is inconsistent with what is believed. This rationality principle usually involve a minimal change assumption, that is also intended to the model repair: the solution should preserve as much information as possible from the original model.

The authors define a model revision operator $\circ_c$ based on a set of basic model change operations, as proposed by [8]. These change operations represent all primitive structural changes over a CTL model:

PU1: Adding one pair to the relation R
PU2: Removing one pair from the relation R
PU3: Changing the labelling function on one state
PU4: Adding one state to S
PU5: Removing one isolated state of S

Let M and M' be two CTL models, we denote by $\mathit{Diff}_{PU_i}(M, M')$ the structural difference between M and M' produced by applications of PU_i, for example, $\mathit{Diff}_{PU_1}(M, M')$ denotes the transitions added to M in order to achieve M'.

A model change is said to be *admissible* if it produces a model M' from M such that M' satisfies the desired property and there is no model M'' obtained from M such that $\mathit{Diff}_{PU_i}(M, M'') \subseteq \mathit{Diff}_{PU_i}(M, M')$, $i = 1, ..., 5$ and $\mathit{Diff}_{PU_i}(M, M'') \subset \mathit{Diff}_{PU_i}(M, M')$, for some $i = 1, ..., 5$. Guerra and Wassermann define a minimality criterion over admissible changes in order to select minimal changes according to belief revision principles, therefore defining the following revision operator:

$$Mod(\psi \circ_c \phi) = Min_{Mod(\psi)}(Mod(\phi)),$$

where ψ, ϕ are CTL formulas that represent the initial beliefs and the new information, respectively, $Mod(\alpha)$ all CTL models of a formula α and $Min_{\mathcal{B}}(\mathcal{A})$ the set of all minimal models of $\mathcal{A}$ according to any admissible modification on any model of $\mathcal{B}$. The authors show that $\circ_c$ satisfies the rationality postulates for belief revision as presented in [11].

Guerra and Wasserman [4] also proposed an algorithm for CTL model revision. The algorithm receives as input a CTL formula ϕ and a set of CTL models

[1] Usually the transition relation is defined as total. Although it makes simple the definition of many temporal logic semantics, this requirement is not needed.

that do not satisfy ϕ, then by repairing each model individually and filtering these repaired models according to their belief revision ordering criterion, the algorithm returns as result a set of revised models representing possible corrections to the original models relative to the formula ϕ.

3 Kripke Model Transition System as Sets of CTL Models

KMTS are expressive models to represent undetermined or sub-specified systems. They have two types of transitions, transitions that *must* occur and transitions that *may* occur, which represent necessary and possible behavior, respectively.

Specification over KMTS are written in the $\mu - calculus$ and in this work we use this language in its negation normal form.

Definition 2. *($\mu - calculus$). Let AP be a set of atomic propositions and V a set of propositional variables. The set of literals over AP is defined as $Lit = AP \cup \{\neg p \mid p \in AP\}$. The $\mu - calculus$ in its negation normal form over AP is defined by $\varphi ::= \mathtt{l} \mid Z \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid AX\varphi \mid EX\varphi \mid \mu Z.\varphi \mid \upsilon Z.\varphi$ where $\mathtt{l} \in Lit$ and $Z \in V$. AX means for all successors and EX means there exists a successor. μ denote the least fixpoint and υ denote the greatest fixpoint. A formula φ is closed if all its variables Z are bounded by a fixpoint operator μ or υ.*

CTL formulas can be specified in $\mu - calculus$ by the following translation: $EF\phi \equiv \mu Z.\phi \vee EXZ; AF\phi \equiv \mu Z.\phi \vee AXZ; EG\phi \equiv \upsilon Z.\phi \wedge EXZ; AG\phi \equiv \upsilon Z.\phi \wedge AXZ; E[\phi U \phi] \equiv \mu Z.\phi \vee (\phi \wedge EXZ)$; and $A[\phi U \phi] \equiv \mu Z.\phi \vee (\phi \wedge AXZ)$.

Definition 3. *A Kripke modal transition system (KMTS) is a tuple $M = \langle AP, S, S_0, R^+, R^-, L\rangle$, where S is a set of finite sates, $S_0 \subseteq S$ is the set of initail states, $R^+ \subseteq S \times S$ and $R^- \subseteq S \times S$ are transition relations such that $R^+ \subseteq R^-$, and $L : S \to 2^{Lit}$ is a label function, such that for all state s and $p \in AP$, at most one between p and $\neg p$ occur. The transitions R^+ e R^- correspond to the transitions must and may respectively.*

The semantics defined below is presented in [6]. A complete semantics of $\mu - calculus$ is presented in [12].

Definition 4. *The semantics of three values $\|\varphi\|_3^M$ of a closed formula φ with respect to a KMTS M is a map from S to $\{T, F, \perp\}$. The interesting cases are defined below.*

$$\|\mathtt{l}\|_3^M(s) = T \text{ if } \mathtt{l} \in L(s), F \text{ if } \neg\mathtt{l} \in L(s), \perp \text{ otherwise.}$$

$$\|AX\varphi\|_3^M(s) = \begin{cases} T, & \text{if } \forall t \in S, \text{if } R^-(s,t) \text{ then } \|\varphi\|_3^M(t) = T \\ F, & \text{if } \exists t \in S \text{ such that } R^+(s,t) \text{ and } \|\varphi\|_3^M(t) = F \\ \perp, & \text{otherwise.} \end{cases}$$

And dually for $EX\varphi$ exchanging $\mathtt{F}$ and $\mathtt{T}$.

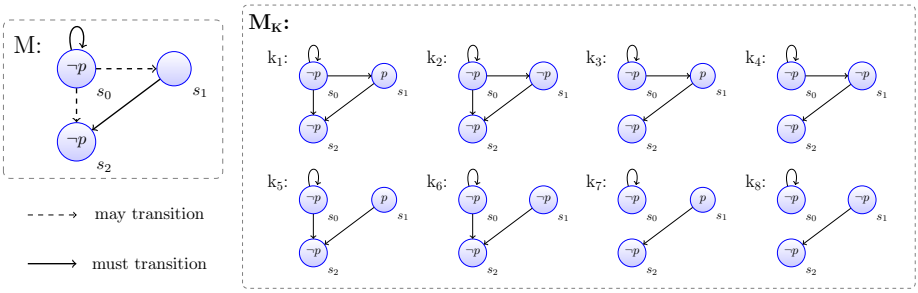


Fig. 1. (a) Example of a KMTS M (b) Expansion M_K of M

3.1 Expanding KMTS into CTL Models

In this section we formally define a KMTS expansion into a set of Kripke structures showing its capacity to compactly represent CTL models and some limitations of this representation.

Definition 5. *Let $M = \langle AP, S, S_0, R^+, R^-, L\rangle$ be a KMTS, the KMTS expansion of M, denoted by M_K, is the set of all Kripke models $K' = \langle AP', S', S'_0, R', L'\rangle$ such that $AP' = AP$, $S' = S$, $S'_0 = S_0$, $R^+ \subseteq R' \subseteq R^-$ and $L(s) \subseteq L'(s)$, for all $s \in S$.*

The KMTS expansion may lead to an exponential set of Kripke models, as stated in Proposition 1. On the other hand, it shows the capacity of this formalism to compactly represent a huge set of CTL models in one single structure. It is important to note that KMTS may not be expressive enough to represent all possible sets of CTL models, as shown in Proposition 2.

Proposition 1. *Let $M = \langle AP, S, S_0, R^+, R^-, L\rangle$ be a KMTS with $m = |R^- \backslash R^+|$ genuine (strictly) may transitions and $n = |\{s \in S \mid p \in AP \text{ and } p, \neg p \notin L(s)\}|$ state indeterminations. M can be expanded into 2^{m+n} Kripke structures.*

Proof. It follows straight from the number of possible combinations of each KMTS indetermination that can be realized or not in the Kripke structures.

Proposition 2. *Let $K = \{k_1, ..., k_n\}$ any set of kripke structures $k_i = \langle AP, S, S_0, R_i, L_i\rangle$. Not necessarily exists a KMTS $M = \langle AP, S, S_0, R^+, R^-, L\rangle$ that can be expanded into K.*

Proof. Take for example $K = \{k_3, k_5\}$ of Figure 1(b). No KMTS $M = \langle \{p\}, \{s_0, s_1, s_2\}, \{s_0\}, \{(s_0, s_0), (s_1, s_2)\}, R_-, L\}\rangle$ can be expanded in this set. This is because the KMTS formalism does not provide any way of expressing interdependency between indeterminations. In this example, we could not express in M that the transitions (s_0, s_1) and (s_0, s_2) should not occur at the same time.

To represent any set of Kripke structures we have two alternatives: (1) to associate a selection function to a KMTS that selects the desired Kripke models among its expanded models; (2) to consider a set of KMTS models that represent the set of Kripke models. In the second alternative, in the worse case, each KMTS will be a Kripke model.

Proposition 3. *Let M be a KMTS and $K = \{k_1, ..., k_n\}$ the Kripke structures expanded from M. Consider s_0 the initial state of M. For all closed formula φ of $\mu - calculus$, if the semantic value of $\|\varphi\|_3^M(s_0)$ is equal to*

1. *$\bot$, then $\exists k_i, k_j \in K, i \neq j$ such that $(\|\varphi\|^{k_i}(s_0) = T$ and $\|\varphi\|^{k_j}(s_0) = F$*
2. *T, then $\forall k_i \in K, \|\varphi\|^{k_i}(s_0) = T$*
3. *F, then $\forall k_i \in K, \|\varphi\|^{k_i}(s_0) = F$*

Proof. It follows straight from the semantics of KMTS and the expansion of it.

4 Revision of KMTS Models

In this section we define the KMTS model revision operation, through the specification of minimal change criterion over KMTS models and showing its correspondence to the minimal changes over sets of KMTS expanded Kripke models. This minimality criterion is similar to that proposed by [5], but now considering a different set of primitive operations which represent possibilities of changes in KMTS models, as shown below.

P1: Removing one pair from the relation R^-
P2: Removing one pair from the relation R^+
P3: Transforming one pair (s_i, s_j) of R^- to (s_i, s_j) of R^+
P4: Changing a defined literal on one state label
P5: Assigning a literal to a a state label if it is undefined in it

For the definitions below we consider some notation. X_{Pn} denotes a set of changes relative to operation $P_n, 1 \leq n \leq 5$. Each change in X_{Pn} is represented as (s_i, s_j) or (s_i, l), where l is a literal, depending on whether the change is relative to transitions or to state labels, respectively. A change X is represented as $X = (X_{P1}, ..., X_{P5})$, where X_{Pn} can be an empty set if no change of type P_n occurs. We say that $X = (X_{P1}, ..., X_{P5}) \subset Y = (Y_{P1}, ..., Y_{P5})$ if for each $X_{Pn} \subseteq Y_{Pn}$ and at least one $X_{Pi} \subset Y_{Pi}$. The application of X to a model A results in another model denoted by $A(X)$. We refer to $M, s_0 \models \varphi$ is $True$, $False$ or $\bot$ to indicate the result of model checking φ in M from s_0.

Our definition of minimal change over KMTS is based on the operations P2 and P4, the operations P1, P3 and P5 are disregarded. This makes sense because among the Kripke models expanded of the KMTS there are models without the transitions of P1, which already have the transitions of case P3, and those where the state label already has the literal of P5 assigned. In this sense, these modifications should not be considered for all models. We then define minimal changes considering a reduced change $X/$ of a change X as defined below.

Definition 6. *Let $X = (X_{P1}, ..., X_{P5})$, the reduced change $X/$ of X is defined as $X/ = (X_{P2}, X_{P4})$.*

A reduced change $X/ = (X_{P2}, X_{P4})$ over a KMTS M induces changes in $K \in M_K$: all $(s_i, s_j) \in X_{P2}$ induces a change (s_i, s_j) of type $PU2$ in K and all $(s_i, l) \in X_{P4}$ induces a change (s_i, l) of type $PU3$ in K. So, we also refer X_{P2} and X_{P4} as changes over K meaning its corresponding induced changes.

Definition 7. *Given two changes $X1 = (X1_{P1}, ..., X1_{P5})$ and $X2 = (X2_{P1}, ..., X2_{P5})$, $X1 \leq X2$ iff for all n, $X1/_{Pn} \subseteq X2/_{Pn}$. $X1 < X2$ iff $X1 \leq X2$ and there is at least one n, such that $X1/_{Pn} \subset X2/_{Pn}$. If there is no $X2$ such that $X2 < X1$, $X1$ is said to be minimal.*

Propositions 4, 5 and 6 show that the defined minimality criterion for KMTS correspond to the minimality criterion (presented in section 2.2) for the set of Kripke models expanded of the KMTS, i.e. the revision of a set of Kripke models can be achieved by the revision of a KMTS that represents them. The next proposition specifies that any change in a Kripke model, that belongs to M_K, can be achieved through a change in M.

Proposition 4. *Let M be a KMTS, M_K its corresponding expansion, $K1$ a model in M_K and $Y = (Y_{P2}, Y_{P4})$ a change in $K1$. Then there is a change X in M such that $M(X)_K$ contains the model $K1(Y)$.*

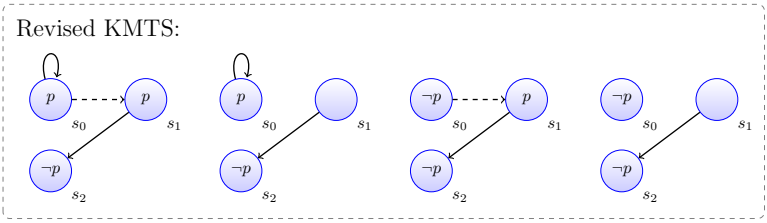
Proof. Construct $X = (X_{P1}, X_{P2}, X_{P3}, X_{P4}, X_{P5})$: X_{P1} contains all (s_i, s_j) may transitions such that (s_i, s_j) are not transitions of $K1$; X_{P2} contains $(s_i, s_j) \in Y_{P2}$ if (s_i, s_j) is a must transition in M, otherwise (s_i, s_j) is included in X_{P1}; X_{P3} contains all (s_i, s_j) may transitions which correspond to (s_i, s_j) transitions of $K1$; X_{P4} contains $(s_i, l) \in Y_{P4}$ if l or $\neg l \in label(s_i)$ in M, otherwise include it in X_{P5}. Take a model $K2$ from M_K which differs from $K1$ in two ways: 1) for all $(s_i, l) \in X_{P5}, l \in label(s_i)$ in $K2$; 2) $K2$ does not have the transitions (s_i, s_j) of Y_{P2} if they are may transitions in M ($K2$ exists because the expansion of M generates all Kripke models resulting from all the possibilities of transforming indetermination in M in determinations in Kripke models, in the best case $K2 = K1$ and $X/ = Y$). Therefore $M(X)_K$ contains the model $K2(X/)$ which is equal to $K1(Y)$.

Proposition 5. *Let M be a KMTS and $X = (X_{P1}, X_{P2}, X_{P3}, X_{P4}, X_{P5})$ be a minimal change in M. Then $X/ = (X_{P2}, X_{P4})$ is a minimal change in M_K.*

Proof. Suppose $X/$ is not minimal in M_K, so there is a change $Y = (Y_{P2}, Y_{P4})$ in M_K such that $Y < X/$. The transitions of Y_{P2} are not may transitions in M and all literals of the states in Y_{P4} are defined literals in the respective states in M. By the proposition 4 there is a change Z in M constructed from Y such that $Z/ = Y$. So, $Z/ < X/$ which implies that $Z < X$, a contradiction.

Proposition 6. *Let M be a KMTS such that $M, s_0 \models \varphi$ is False, $X = (X_{P1}, X_{P2}, X_{P3}, X_{P4}, X_{P5})$ a minimal change in M such that $M(X), s_0 \models \varphi$ is True. So, there is a model K in M_K such that $K(X/), s_0 \models \varphi$ is True.*

Proof. Take a model K of M_K such that K does not have the transitions of X_{P1}, has the transitions of X_{P3} if $X_{P3} \neq \emptyset$ and for all (s_i, l) in X_{P5}, $l \in$ labels of s_i of K. For all K_i of $M(X)_K$, $K_i, s_0 \models \varphi$ is True because $M(X), s_0 \models \varphi$ is True (see proposition 3). The model $K(X/)$ is one of the K_i models.



Minimal changes over M

$X1 = (\{(s_0, s_2)\}, \emptyset, \emptyset, \{(s_0, p)\}, \{(s_1, p)\})$
$X2 = (\{(s_0, s_1), (s_0, s_2)\}, \emptyset, \emptyset, \{(s_0, p)\}, \emptyset)$
$X3 = (\{(s_0, s_2)\}, \{(s_0, s_0)\}, \emptyset, \emptyset, \{(s_1, p)\})$
$X4 = (\{(s_0, s_1), (s_0, s_2)\}, \{(s_0, s_0)\}, \emptyset, \emptyset, \emptyset)$

Reduced changes

$X1/ = (\emptyset, \{(s_0, p)\})$
$X2/ = (\emptyset, \{(s_0, p)\})$
$X3/ = (\{(s_0, s_0)\}, \emptyset)$
$X4/ = (\{(s_0, s_0)\}, \emptyset)$

Fig. 2. Revision by AXp of the KMTS M (Fig. 1(a))

Figures 2 and 3 show an example of the relation between minimal changes of a KMTS M and minimal changes of the set M_K. They present the minimal possible changes (with the operations P_1 to P_5 above) in M to satisfy the property AXp from s_0 and their respective reduced changes which correspond to the minimal changes of M_K. Consider the change $X = (\emptyset, \emptyset, \emptyset, \{(s_0, p), (s_2, p)\}, \{(s_1, p)\})$, it is not minimal because $X1 < X$. To exemplify the Proposition 4 consider the model K_4 (Fig. 1(b)) and the change $Y = (\emptyset, \{(s_0, p), (s_1, p)\})$, the model $K_4(Y) = K_3(X_2/)$. As an example of Proposition 6, consider the model K_3 (Fig. 1(b)) and the change X_1 of M (Fig. 2), $M(X_1), s_0 \models AXp$ is $True$ and $K_3(X_1/), s_0 \models AXp$ is $True$.

In the case that the KMTS model checking returns $\perp$, the KMTS revision selects among its expanded Kripke models those that are consistent with the

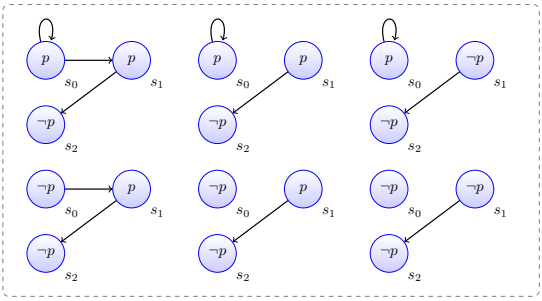


Fig. 3. Revision by AXp of the expansion M_K (Fig 1(b))

verified property. This result is aligned to the result produced by the operator $\circ_c$. When the KMTS model checking returns $false$, changes effectively modify the KMTS model. As stated before in this paper we consider only changes in state labels and removal of transitions. Although these changes seem relatively restrictive, the results presented in this paper are still relevant to domains where the KMTS completely defines the consistent information, and thus no other information such as new states can be added. Our approach is also a step forward in the definition of a general KMTS revision operator with all kinds of changes, which we intend to define afterwards.

5 Implementing Revision of KMTS Models

The revision of a KMTS model M might occur when $M, s_0 \models \varphi$ is $\perp$ or F. In case $\perp$ the revision consists of refining the KMTS to be expanded into only Kripke structures where the required property is satisfied. In case F, the KMTS should be repaired resulting in KMTSs where model checking results in T causing changes in the expanded Kripke structures.

In this section we present the 3-valued model checking game proposed by Grumberg in [12] and our proposal of an abstract algorithm over this game to refine a KMTS model.

5.1 The 3-valued Model Checking Game

In the $\mu - calculus$ 3-valued model checking game proposed in [12], Grumberg introduces the concept of non-losing strategy to identify the causes of $\perp$ in model checking besides the known concept of winning strategy. These games are defined between two players, $\exists$ and $\forall$, where the player $\exists$ tries to verify the formula and the $\forall$ tries to refute the formula.

The game for model checking a formula φ consists of a graph of configurations of type $s \vdash \psi$ where s is a state of the model and ψ is a subformula of φ. These configurations are determined from the decomposition of the formula φ in its subformulas according to the rules presented in Figure 4, considering the states and transitions of the KMTS model.

In Figure 5 we show an example of a graph of configurations of a 3-valued model checking game. A configuration is classified as a $\exists$ configuration when ψ is of the form of the antecedent of an $\exists$ rule and is represented as an ellipse in the game graph and is classified as a $\forall$ configuration if ψ is of the form of the antecedent of an $\forall$ rule and is represented as a rectangle in the game graph. Dotted edges correspond to KMTS genuine may transitions $(R_- \backslash R_+)$ and normal edges correspond to both KMTS must transitions and other moves generated from the rules that do not involve transitions of the model.

The players move from their configurations according to a strategy. A strategy of a player σ is a function between its configurations and all the configurations of the game graph. A winning strategy of player σ is such that it makes σ win a game independent of the strategy used by the other player. When neither players

win the game, both of them have a non-losing strategy and the game results $\perp$. For example, in Figure 5 the bold edges are part of non-losing strategies of the $\forall$ player.

Rules of player $\exists$:

$$\frac{s \vdash \psi_0 \vee \psi_1}{s \vdash \psi_i} : i \in \{0,1\} \qquad \frac{s \vdash EX\psi}{t \vdash \psi} : R^+(s,t) \text{ or } R^-(s,t)$$

$$\frac{s \vdash \eta Z.\psi}{s \vdash Z} : \eta \in \{\mu, \upsilon\} \qquad \frac{s \vdash Z}{s \vdash \psi} : \text{if } f_p(Z) = \eta Z.\psi, \eta \in \{\mu, \upsilon\}, \text{ and } f_p(Z) \text{ is the unique subformula identified by } Z$$

Rules of player $\forall$:

$$\frac{s \vdash \psi_0 \wedge \psi_1}{s \vdash \psi_i} : i \in \{0,1\} \qquad \frac{s \vdash AX\psi}{t \vdash \psi} : R^+(s,t) \text{ or } R^-(s,t)$$

Fig. 4. Rules of the model checking game

A play can be finite or infinite and it is defined as a sequence of configurations $C_0, C_1, ...$ such that there is an edge from C_i to C_{i+1}. Each configuration of the graph is colored depending on the result of all plays starting from this configuration: with T if the player $\exists$ wins, with F if the player $\forall$ wins, or $\perp$ if both players do not win (or do not lose). A necessary condition for a player to win a play is to obey the restriction that all of his/her movements in the configurations of the play are through normal edges, meaning that the player does not move between configurations that corresponds to genuine may transitions of the model. Moreover, there are other conditions to determine the winner of a play as presented below.

Conditions for the player $\exists$ win a play $C_0, C_1, ...$:

1. To exist a $n \in N$ such that $C_n = t \vdash l$ and the state t of the model is labelled with l or
2. To exist a $n \in N$ such that $C_n = t \vdash AX\psi$ and there does not exist $t' \in S$ such that (t, t') is a transition in the model or
3. the outermost variable that occurs infinitely often is of type υ

Conditions for the player $\forall$ win a play $C_0, C_1, ...$:

1. To exist a $n \in N$ such that $C_n = t \vdash l$ and the state t of the model is labelled with $\neg l$ or
2. to exist a $n \in N$ such that $C_n = t \vdash EX\psi$ and does not exits $t' \in S$ such that (t, t') is a transition in the model or
3. the outermost variable that occurs infinitely often is of type μ

If neither player wins a play, the result of it is $\perp$, meaning that both players have a non-losing strategy for that play. A player wins a game if he/she wins all the plays in the game from the initial configuration $(s_0 \vdash \varphi)$.

To calculate the result of the game, one can color each configuration of the graph bottom up with T, F or $\perp$ depending on whether $\exists$ has a winning strategy, or $\forall$ has a winning strategy, or both players have a non-losing strategy,

respectively, in all plays starting from that configuration. Initially the deadend configurations are colored (a deadend configuration is one that does not reach another configuration), then the coloring proceeds to other configurations taking other plays until all the configurations are colored as explained in next section. The result of the game will be the color of the root node of the graph (configuration $s_0 \vdash \varphi$). Figure 5 presents a game graph with the colored configurations (represented by the symbols enclosed in parenthesis inside the node of the configuration) and with edges that belong to the non-losing strategies of player $\forall$, represented as bold edges.

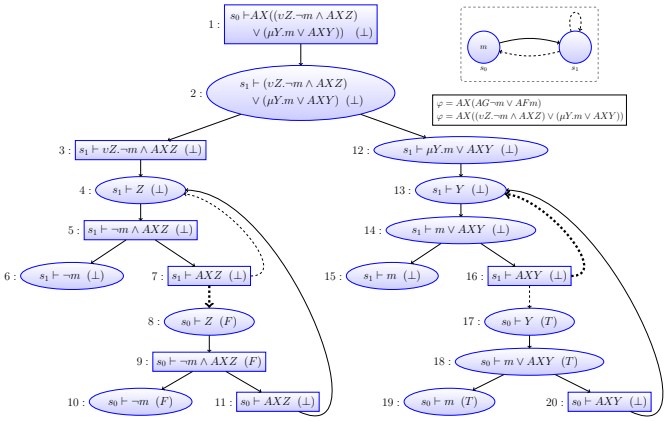


Fig. 5. Example of failure witnesses of non-losing strategy of player $\forall$

5.2 Implementing KMTS Repair

In this section we develop an algorithm to refine the KMTS (case $M, s_0 \models \varphi$ is $\bot$) based on the repair of the 3-valued model checking game. The algorithm considers non-losing strategies (that are not winning strategies) for both players $\forall$ and $\exists$ defined in [12] to determine the witnesses of the failure. Our algorithm consists of reducing the KMTS to represent only the Kripke structures that satisfy the property φ by eliminating genuine may transitions, or transforming genuine may transitions into must transitions, or changing the labels of undefined states. At the end of this section we present a quick overview of an algorithm to implement the repair when $M, s_0 \models \varphi$ is $False$.

From now on we will refer to the configurations of the game as nodes. Let ψ be a subformula of φ. We define a witness of a failure in case $M, s_0 \models \psi$ is $\bot$ one of the following transitions, which belongs to non-losing strategies of $\forall$ or $\exists$, found bottom up in the game graph: (1) a genuine may edge, coming from a node of type AX colored $\bot$, to a child node colored F or $\bot$, (2) a genuine may edge, coming from a node of type EX colored $\bot$, to a child node colored T or $\bot$, (3) a must edge, coming from a node of type EX colored $\bot$, to a child node colored $\bot$, (4) an edge coming from a node of type $s_i \vdash l \wedge \psi$ to a node $s_i \vdash l$

colored with $\perp$, (5) an edge from a node of type $s_i \vdash l \vee \psi$ where its child node $s_i \vdash l$ is colored with $\perp$ and the other child is colored $\perp$ or F. In Figure 5 the bold edges are examples of failure witnesses.

In order to obtain all the Kripke models that satisfy the property, all failure witnesses might be considered, resulting in different KMTSs. It is not necessary to consider all possible combinations of changes in order to generate all possible Kripke models because the KMTS expresses possibilities by the may transitions. For example if a node $s_0 \vdash EXp$ colored $\perp$ has two may children nodes $s_1 \vdash p, s_2 \vdash p$ both colored T, it is enough to change only one may edge at a time (to be a must edge), because the resultant KMTSs expresses the Kripke models with both transitions as must.

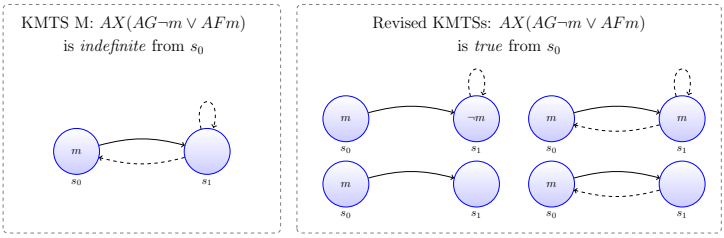


Fig. 6. Example of KMTS refinement

The algorithm Revision-game controls the refinements of the model M from the sequence of failure witnesses (Fwitness) identified by procedure Check-model, which determines them from a the 3-valued model checker game. One failure of Fwitness at a time is processed by Refine-game until no more failures exists in Fwitness. For the game of Figure 5, Fwitness will be initialized with the sequence of failure witnesses $((7,8),(16,13),(14,15))$, where a pair (m,n) represents the edge from the node m to the node n of the game. Other failure witnesses are considered by Refine-game to complement the change $X = (\{(s_1,s_0)\},\emptyset,\emptyset,\emptyset,\emptyset)$ such as $(5,6)$ and $(16,13)$. The algorithm returns 4 KMTSs (see Figure 6) that satisfy φ with the changes: (1) $X = (\{(s_1,s_0)\},\emptyset,\emptyset,\emptyset,\{(s_1,\neg m)\})$, (2) $X = (\{(s_1,s_0),(s_1,s_1)\},\emptyset,\emptyset,\emptyset,\emptyset)$, (3) $X = (\{(s_1,s_1)\},\emptyset,\emptyset,\emptyset,\emptyset)$, (4) $X = (\emptyset,\emptyset,\emptyset,\emptyset,\{(s_1,m)\})$.

The algorithm Refine-game controls the possible refinements from a failure witness of Fwitness. Each change X is used to the modification and recoloring of the game graph by Recolor-game (which is supposed to call the 3-valued model cheker). A change is done relative to the model, i.e., if an edge (m,n) corresponds to the transition $(s_i,\ s_j)$ in the model, all edges (r,s) which correspond to $(s_i,\ s_j)$ should be removed of the game graph and if the subgraphs from nodes s are no more accessible from the root node they must be desconsidered by other search for failure witnesses. If the model checker results $\perp$, all the other failure witnesses in Nwitness (determined by Recolor-game) are considered (one at a time) to complement X by call Refine-game recursively. When the result of model checking is T the model $M(X)$ is returned, the game is restored to a

previous state in order to other failures witnesses from Nwitness be considered and achieve all possible complementations for the change X.

The algorithm Refine-play generates the change X according to Failure = (m, n), i.e., the change generated depends on the cause of the failure which is relative to the node m or n. Consider node m of type $(s_i \vdash \psi)$ and n of type $(s_j \vdash \chi)$. Node m can be: a AX node $(s_i \vdash AX\psi)$, a EX node $(s_i \vdash EX\psi)$, a disjunctive node of type $s_i \vdash \psi \vee l$, or a conjunctive node of type $s_i \vdash \psi \wedge l$. The conditions specified in the algorithm cover the cases described below that represent the possible changes required.

A node EX is true if it has a must child colored with true, it is false if all its may children are false and otherwise it is $\perp$. If node m is a EX node and n is colored T and (m, n) is a may transition, it should be transformed into a must transition. A may edge to a node of type $s_k \vdash V$ that represents loop in the graph and V is a greatest fixpoint variable (ν) (formulas of type EG) is also changed to must. A node AX is true if all its may (including must) children are colored true, it is false if it has a must child colored false, otherwise it is $\perp$. If node n is colored F or $\perp$ (not of type $s_j \vdash l$) the may transition (m, n) should be cut, if it is $\perp$ and of type $s_j \vdash l$ two alternatives exits: the label of s_j is changed to contain l or the transition is cut. To consider both alternatives when a failure witness of this type is found then this failure witness (m, n) is duplicated in the sequence of witnesses (Fwitness or Nwitness) and in the second one the node is represented as a negative number $(-n)$. A node $s_i \vdash \psi \wedge l$ is colored $\perp$ if it does not have a child colored F and has one or both children colored $\perp$. So, if node n of type $s_j \vdash l$ is colored $\perp$ the label of s_j should be changed. A node $s_i \vdash \psi \vee \chi$ is colored $\perp$ if it does not have a child colored T and has a child colored $\perp$. So, if node n of type $s_i \vdash l$ is colored $\perp$ its label should be changed.

Algorithm 1. Revision-game()

Input: KMTS M to revise, property φ /* X is declared as a global variable */
Output: KMTSs resultant of the changes with $s_0 \vdash \varphi$ colored T

```
1  Read(M, φ) ;
2  Check-model(M, φ, Fwitness); /* returns a 3-valued model checking game
   graph and the possible failure witnesses in case s0 ⊢ φ colored ⊥ */
3  if s0 ⊢ φ colored ⊥ then
4      repeat
5          X := () ;
6          Refine-game(Fwitness, X) ;
7          Restore-game(head(Fwitness)) ;
8          Fwitness := tail(Fwitness) ;
9      until Fwitness = nil;
10 end
```

Algorithm 2. Refine-game(Fwitness, X)

Input: Fwitness - sequence of pairs (m, n) determining the failure witness edges
Output: KMTSs resultant of the changes with $s_0 \vdash \varphi$ colored T

```
1  Failure := head(Fwitness);
2  Refine-play(Failure, X); /* X contains the changes to be done */
3  Recolor-game(φ, Nwitness, X); /* other failure witnesses are put in Nwitness if
   they exists (s0 ⊢ φ is colored ⊥) */
4  if s0 ⊢ φ is colored T then
5  |  Return M(X) ;
6  |  Restore-game(Failure) /* the game is restored by removing the change
   |  corresponding to Failure ;
7  else if s0 ⊢ φ is colored ⊥ then
8  |  while Nwtiness ≠ nil do
9  |  |  Refine-game(Nwitness, X) ;
10 |  |  Nwitness := tail(Nwitness) ;
11 |  end
12 end
```

Algorithm 3. Refine-play(Change, X)

Input: Failure = (m, n) such that node m is colored $\bot$ and is of type $s_i \vdash \psi$ and node n is of type $s_j \vdash \chi$
Output: Changes X

```
1  if node m is of type s_i ⊢ EXψ and n is colored T or (ψ = V and V is a
   variable of type ν and n is colored ⊥ and (m,n) represents a loop in the game
   graph then
2  |  X.P3 := X.P3 ∪ {(s_i, s_j)};
3  else if node m is of type s_i ⊢ AXψ then
4  |  if node n is colored F or n is not of type s_j ⊢ l and is colored ⊥ then
5  |  |  X.P1 := X.P1 ∪ {(s_i, s_j)} ;
6  |  else if node n is of type s_j ⊢ l and is colored ⊥ then
7  |  |  if n > 0 then X.P5 := X.P5 ∪ {(s_j, l)};
8  |  |  else X.P1 := X.P1 ∪ {(s_i, s_j)} ;
9  |  end
10 if node n is of type s_j ⊢ l and m is not of type s_i ⊢ AXψ then
11 |  X.P5 := X.P5 ∪ {(s_j, l)};
12 end
```

For the implementation of the repair of the KMTS when a property is inconsistent with the model, a similar algorithm used for the refinement can be developed. Winning strategies of player $\forall$ instead of non-losing strategies should now be considered to identify some failure witnesses, combined with other failures witnesses such as deadends nodes colored with F. The algorithm can proceed also from bottom up changing labels or eliminating transitions that are causes of the failure in the game.

6 Final Remarks

As addressed before, general sets of Kripke structures cannot be represented by a single KMTS (Proposition 2). A solution is to generalize KMTS revision to deal with a set of KMTS instead of a single model. This solution increases the revision complexity, but it is upperbounded by the complexity of CTL revision (in the worst case, each KMTS will be a Kripke structure). The set of KMTS still have on average a more compact representation, which allows the development of more efficient revision methods.

Revision over KMTS structures is significantly more efficient than Kripke revision. For example, the revision of the models of Figure 3 produces 32 repair candidates, which have to be compared to the 8 initial models in order to select the minimal ones, which involve approximately 256 comparisons. This computation can be even more complicated if it involves a fixpoint formula as EF or AG, in the sense that it increases the number of repair candidates greatly, as the number of redundant or useless change to achieve them. On the other hand, the KMTS revision is almost straightforward from M to the solution set of 4 KMTS, with almost no redundant or useless modifications. The algorithms used for the repair were specified over 3-valued model checking game which can be implemented as two $\mu - calculus$ 2-valued model checking game [12]. It is noteworthy that $\mu - calculus$ 2-valued model checking game for CTL is linear in time.

6.1 Related Work

Zhang and Ding [8] proposed the first approach on this line, improving model checking with belief update theory [11]. As shown in [4], the choice of a belief revision approach, rather then belief update, may avoid some unnecessary loss of information in static contexts. Zhang and Ding [8] do not deal with partial system information. Belief revision principles were adopted in [4], but with no focus on partial system information. Although their framework may deal with partial information by handling sets of models, its lack of a compact representation like KMTS that can make it difficult to be used in real applications. Grumberg [6] addresses KMTS representation, but the context is abstract model checking, where a KMTS model represents an abstraction of a concrete Kripke structure. Grumberg also proposes an algorithm based on 3-valued model checking to refine a KMTS with a different proposal which consists in expanding an abstract state of the KMTS (with some undefined literal) into concrete states (states of the concrete Kripke structure that was abstracted in the KMTS). Finally, in [7], the authors deal with KMTS and develop an algorithm, not using 3-valued model checking, to repair KMTS models. Two main differences distinguish their approach from ours: the focus is on abstract model checking and not on partial system specification; and their proposal does not refer to a known change theory.

7 Conclusion

In this paper we presented a new approach to the revision of a set of CTL models through the revision of a KMTS model. We considered the revision of

KMTS both when the satisfiability of a property is undefined or is inconsistent with the model. We defined a minimality criterion relative to KMTS repair and proved that it preserves the minimality criterion relative to the repair of its set of expanded CTL models as in [4]. We presented an algorithm to implement the revision in case the property is undefined. The design of an algorithm for the repair of the KMTS in case the property is false is our next goal.

The work presented here is a first step towards a framework for the automatic repair of partial specifications. In this version of this work we considered only changes of labels of states and the removal of transitions. We aim to propose a generalization of this solution from an extension of the approach presented here.

Acknowledgments. The first author was supported by the grant #2010/15392-3, São Paulo Research Foundation (FAPESP). The second author was supported by the grant #2012/16308-1, São Paulo Research Foundation (FAPESP). The third author was supported by Brazilian Research Council (CNPq) grant #304043/2010-9.

References

1. Clarke, E.M., Grumberg, O., Peled, D.A.: Model checking. Springer (1999)
2. Huth, M.: Model checking modal transition systems using kripke structures. In: Cortesi, A. (ed.) VMCAI 2002. LNCS, vol. 2294, pp. 302–316. Springer, Heidelberg (2002)
3. Alchourron, C., Gärdenfors, P., Makinson, D.: On the logic of theory change. J. Symb. Logic 50(2), 510–530 (1985)
4. Guerra, P.T., Wassermann, R.: Revision of CTL models. In: Kuri-Morales, A., Simari, G.R. (eds.) IBERAMIA 2010. LNCS, vol. 6433, pp. 153–162. Springer, Heidelberg (2010)
5. Guerra, P.T.: Revisão de modelos CTL. Master's thesis, Univ. São Paulo (2010)
6. Grumberg, O.: 2-valued and 3-valued abstraction-refinement in model checking. Information and Communication Security 25, 105–128 (2011)
7. Chatzieleftheriou, G., Bonakdarpour, B., Smolka, S.A., Katsaros, P.: Abstract model repair. In: Goodloe, A.E., Person, S. (eds.) NFM 2012. LNCS, vol. 7226, pp. 341–355. Springer, Heidelberg (2012)
8. Zhang, Y., Ding, Y.: CTL model update for system modifications. Journal of Artificial Intelligence Research 31(1), 113–155 (2008)
9. Clarke, E.M., Emerson, E.A.: Design and synthesis of synchronization skeletons using branching-time temporal logic. In: Kozen, D. (ed.) Logic of Programs, vol. 131, pp. 52–71. Springer, Heidelberg (1982)
10. Clarke, E.M., Emerson, E.A., Sistla, A.P.: Automatic verification of finite-state concurrent systems using temporal logic specifications. ACM Transactions on Programming Languages and Systems (TOPLAS) 8(2), 244–263 (1986)
11. Katsuno, H., Mendelzon, A.O.: On the difference between updating a knowledge base and revising it. In: Proc. of KR, pp. 387–395. Morgan Kaufmann (1991)
12. Grumberg, O., Lange, M., Leucker, M., Shoham, S.: When not losing is better than winning: Abstraction and refinement for the full μ-calculus. Inf. Comput. 205(8), 1130–1148 (2007)