Multi-Valued Symbolic LTL Model Checker

Wendy W. Liu

Department of Computer Science, University of Toronto,
Toronto, ON M5S 3G4, Canada.
www.cs.toronto.edu

Abstract. In the classical logic framework, studies of symbolic LTL [1] model checking has been done. This naturally leads to the extension of model checking to multi-valued temporal logics. In this paper, we will first review the approach used by classical symbolic LTL model checker and the basic syntax and semantics of multi-valued logics, then explore the multi-valued LTL model checking using similar approach. The resulting algorithm can be applied to any multi-valued logic whose truth values form a quasi-boolean lattice. We assume that readers are familiar with symbolic model checking.

1 Introduction

Multi-valued logics have started to receive attention in Software Engineering arena, especially in the field of Requirements Engineering [6]. It is can be used to model multiple view points in requirement elicitation. Recent papers have presented an implementation of CTL Model Checker for multi-valued temporal logics [2–4].

In the classical logic framework, various symbolic LTL model checking algorithms have been published [5]. We are interested in extending such an algorithm to multi-valued logics.

In Section 2, we will review a tableau based approach of classical symbolic LTL model checker. Next in Section 3, we will give the basic syntax and semantics of the multi-valued logics. At last, in Section 4 we will present the algorithm for multi-value LTL model checking using tableau based approach.

We will demonstrate the approach by using a coffee dispenser example.

2 Classical Symbolic LTL Model Checker

Given an LTL formula Af, where f is a *restricted path formula* in which the only state subformulas are atomic propositions, we want to determine whether all the states $s \in S$ in M entail Af, i.e., $M, s \models Af$, where $M = (S_M, R_M, L_M)$ is a Kripke structure [5]. Since $M, s \models Af$ iff $M, s \models \neg E \neg f$, it is sufficient to be able to check the truth value of formulas of the form Ef where f is a restricted path formula.

First, a sketch of the algorithm is given here. Given a formula Ef and a Kripke structure M, construct a *tableau* T for the path formula f. T is a Kripke structure which includes *every* path that satisfies f. Obtain a product Kripke structure of the set of paths

[1] Linear temporal logic

that appear in both T and M by composition. A state in M satisfies Ef iff it is the start of an infinite path in the product which satisfies f [5].

2.1 Tableau Construction

Let f be a restricted path formula. Let AP_f denote the set of atomic propositions of f. The tableau associated with f is a Kripke structure $T = (S_T, R_T, L_T)$ with AP_f as its set of atomic propositions. Each state in the tableau is a set of *elementary formulas* obtained from f.

Definition 1. *The set of* elementary formulas *of f, $el(f)$, is defined recursively as the following*

$$\begin{aligned} el(p) &= \{p\} \textit{ if } p \in AP_f. \\ el(\neg g) &= el(g). \\ el(g \vee h) &= el(g) \cup el(h). \\ el(Xg) &= \{Xg\} \cup el(g). \\ el(gUh) &= \{X(gUh)\} \cup el(g) \cup el(h). \end{aligned}$$

The state space S_T of the tableau is $\mathcal{P}(el(f))$ [2]. The labeling function L_T at each state is defined to be the set of atomic propositions contained in the state. According to this labeling scheme, the tableau may contain two different states of the same labeling, since one state may have Xg to be true, and the other has Xg to be false, where Xg is not part of atomic propositions.

Before we construct the transition relation R_T, we need to define a helper function sat which associates with each subformula g of f a set of states in S_T. Intuitively, $sat(g)$ is the set of states that satisfy g.

Definition 2. *The function* sat *associates with each subformula g of f a set of states in S_T.*

$$\begin{aligned} sat(g) &= \{s \mid g \in s\} \textit{ where } g \in el(f). \\ sat(\neg g) &= \{s \mid s \notin sat(g)\}. \\ sat(g \vee h) &= sat(g) \cup sat(h). \\ sat(gUh) &= sat(h) \cup (sat(g) \cap sat(X(gUh))). \end{aligned}$$

The transition relation must ensure that each elementary formula in a state is true in that state. If Xg is in some state s, i.e., Xg is true in s, then all of the successors of s should satisfy g. Since all LTL formulas are universally qualified, the formula must apply to all states. Thus we require that if Xg is not in some state y, i.e., Xg is false in y, then no successor of y should satisfy g.

Definition 3. *The transition relation is defined to be*

$$R_T(s, s') = \bigwedge_{Xg \in el(f)} s \in sat(Xg) \Leftrightarrow s' \in sat(g).$$

[2] $\mathcal{P}$ denotes the power set.

In order to ensure that *eventuality* properties are fulfilled, we must add one more condition, a path π that starts from a state $s \in sat(f)$ satisfies f if and only if

– *For every subformula gUh of f and for every state s on π, if $s \in sat(gUh)$ then either $s \in sat(h)$ or there is a later state y on π such that $y \in sat(h)$.*

Let $\pi' = s'_0, s'_1, \ldots$ be a path in a Kripke structure M, then $label(\pi') = L(s'_0), L(s'_1), \ldots$. Let $k = k_0, k_1, \ldots$ be a sequence of subsets of the set AP and let $AP' \subseteq AP$. The *restriction* of k to AP', denoted by $k\ |_{AP'}$, is the sequence $m_0, m_1, \ldots$ where $m_i = k_i \cap AP'$ for every $i \geq 0$. The following theorem [5] presents a key property of the tableau.

Theorem 1. *Let T be the tableau for the path formula f. Then, for every Kripke structure M and every path π' of M, if $M, \pi' \models f$ then there is a path π in T that starts in a state in $sat(f)$, such that $label(\pi')\ |_{AP_f} = label(\pi)$.*

2.2 Product Construction

The product $P = (S_P, R_P, L_P)$ of the tableau $T = (S_T, R_T, L_T)$ and the Kripke structure $M = (S_M, R_M, L_M)$ is constructed as follows.

$S_P = \{(s, s') \mid s \in S_T, s' \in S_M \text{ and } L_M(s')\ |_{AP_f} = L_T(s)\}$ (state space)

$R_P((s, s'), (t, t'))$ iff $R_T(s, t)$ and $R_M(s', t')$ (transition relation)

$L_P((s, s')) = L_T(s)$ (label)

The transition relation of this product may fail to be total. If this happens, we remove from S all of those states that do not have successors and restrict the transition relation R to the remaining states.

We extent the function sat to be defined over the set of states S of P by $(s, s') \in sat(g)$ if and only if $s \in sat(g)$.

We next apply CTL model checking and find the set of all states V in P, $V \subseteq sat(f)$, that satisfy $EG\, true$ with the fairness constraints.

$$\{sat(\neg(gUh) \vee h) \ \mid \ gUh \text{ occurs in } f\}$$

Each of the states in V is in $sat(f)$. Moreover, it is the start of an infinite path that satisfies all of the fairness constraints. These paths have the property that no subformula gUh holds on the path while h remains false. If $V = \emptyset$, we can conclude that no state in M satisfies the formula Ef; thus every state satisfies $A(\neg f)$. Otherwise, a counter example is obtained.

The correctness of the construction of the product is summarized in the following theorem [5].

Theorem 2. *$M, s' \models E\, f$ if and only there is a state s in T such that $(s, s') \in sat(f)$ and $P, (s, s') \models EG\, true$ under fairness constraints $\{sat(\neg(gUh) \vee h) \mid gUh$ occurs in $f\}$.*

This completes the symbolic LTL model checking algorithm.

3 Multi-valued Logic

In [2–4], a useful family of multi-valued logics and the concept of the multi-valued sets and relations are defined. We will review only the required definitions below.
A *quasi-boolean logic* Λ has truth values that form a finite quasi-boolean lattice $(\mathcal{L}, \sqsubseteq)$.

Definition 4. *A finite lattice* $(\mathcal{L}, \sqsubseteq)$ *is* quasi-boolean *[1] if there exists a unary operator* $\neg$ *defined for it, with the following properties* (a, b *are elements of* $(\mathcal{L}, \sqsubseteq)$*):*

$$
\begin{array}{llll}
\neg(a \sqcap b) = \neg a \sqcup \neg b & \text{(De Morgan)} & \neg\neg a \;=\; a & (\neg \text{ involution}) \\
\neg(a \sqcup b) = \neg a \sqcap \neg b & & a \sqsubseteq b \Leftrightarrow \neg a \sqsupseteq \neg b & (\neg \text{ antimonotonic})
\end{array}
$$

Conjunction and disjunction of Λ are defined as $\sqcap$ and $\sqcup$ (meet and join) operations of $(\mathcal{L}, \sqsubseteq)$, respectively, and negation is defined as the $\neg$ operator of $(\mathcal{L}, \sqsubseteq)$. In this paper, we additionally assume that $(\mathcal{L}, \sqsubseteq)$ is distributive, i.e.,

$$
a \sqcup (b \sqcap c) = (a \sqcup b) \sqcap (a \sqcup c) \qquad a \sqcap (b \sqcup c) = (a \sqcap b) \sqcup (a \sqcap c) \quad \text{(distributivity)}
$$

Given a quasi-boolean lattice $\Lambda = (\mathcal{L}, \sqsubseteq)$, we define *multi-valued set* (referred to as *mv-set*) $S = \{x \mid x \in_\Lambda S\}_\Lambda$, where $x \in_\Lambda S$ gives the degree of membership of x in the mv-set S. The degree of membership is represented by a truth value in Λ. The domain of the membership function $\in_\Lambda$ is an explicit universe of discourse $\mathcal{U}$.

Definition 5. *Multi-valued set operations are defined to be the following:*

$$
\begin{array}{rll}
S \cap_\Lambda S' \triangleq & \{x \mid (x \in_\Lambda S) \sqcap (x \in_\Lambda S')\}_\Lambda & (\textit{Multiple-valued intersection}) \\
S \cup_\Lambda S' \triangleq & \{x \mid (x \in_\Lambda S) \;\sqcup\; (x \in_\Lambda S')\}_\Lambda & (\textit{Multiple-valued union}) \\
S = S' \triangleq & \forall x : ((x \in_\Lambda S) = (x \in_\Lambda S')) & (\textit{Extensional equality}) \\
\overline{S} \triangleq & \{x \mid \neg(x \in_\Lambda S)\}_\Lambda & (\textit{Multiple-valued complement})
\end{array}
$$

The concept of degrees of relations of two entities allows us to define multi-valued transition relations in a state machine model.

Definition 6. *A multiple-valued relation* $\mathcal{R}$ *on two sets* S *and* T *is an mv-subset of* $S \times T$. *The* forward image *of an mv-subset* Q *of* S *under* $\mathcal{R}$ *is*

$$
F(\mathcal{R}, Q) \triangleq \{t \mid \bigsqcup_{s \in S} (s \in_\Lambda Q) \sqcap ((s, t) \in_\Lambda \mathcal{R})\}_\Lambda
$$

and the backward image $B(\mathcal{R}, Q)$ *of an mv-subset* Q *of* T *is*

$$
B(\mathcal{R}, Q) \triangleq \{s \mid \bigsqcup_{t \in T} (t \in_\Lambda Q) \sqcap ((s, t) \in_\Lambda \mathcal{R})\}_\Lambda
$$

Given an mv-subset of S, its forward image under the relation $\mathcal{R}$ is an mv-subset of T; likewise, an mv-subset of T has an mv-subset of S as a backward image.

4 Multi-Valued Symbolic LTL Model Checking

We would like to undertake the same approach as presented in section 2 for many-valued model checking.

Here is the sketch of the algorithm. Given a formula Ef and a multi-valued Kripke structure M, construct a *tableau* T for the path formula f. T is a multi-valued Kripke structure which includes *every* path that satisfies f for every truth value in Λ. For each truth value $\ell \in \Lambda$, obtain an ℓ-product Kripke structure of the set of paths that appear in both T and M by composition with the transitions being *at least* the value ℓ. A state in M satisfies $Ef = \ell$ iff it is the start of an infinite path in the product which satisfies $f = \ell$.

The rationale for building the product this way is that we may have more than two truth values in our logic. In order to find the truth value of the formula Ef, we must check against each of the truth values starting from the highest rank. This is due to the fact that we use partial ordering to define the transition relation in the tableau. Only when the product with the higher ranked truth value fails to be verified, we move on to the next lowest ranked truth value, and build the corresponding product on top of the previous one. When a non-total ordered lattice is used in the logic, we may have multiple values having the same rank. In such case, we will choose any truth value. This ranking approach turns out to be efficient. We start with a product of minimal set of transition relations, and progressively add more relations of lower ranked value only when it is necessary. The state space of all the products do not change as more transition relations are added.

To see how this should work, we will use a coffee dispenser model as our example, and verify property of this model using multi-valued logic. This example is a modified version from that of [2].

We further assume the multi-valued logic $\Lambda = (\mathcal{L}, \sqsubseteq)$ is given in the following sections.

4.1 Coffee Dispenser Example

In our coffee dispenser model, we are using a 3-valued logic lattice with the truth values $\top, S, \perp$ (Figure 1(a) uses T, S, F correspondingly) [3]. We will use these truth values to distinguish between behaviours that are true, possibly true or false.

In the model shown in Figure 1(b), each variable is assigned one of the three truth values, and each transition between states is labeled with one of the three truth values. Note that the $\perp$-valued variables are shown for clarity, but the $\perp$-valued transitions are omitted to avoid clutter.

Finally, we want to verify the property that the machine can make foam. This property is expressed in CTL as $EF(milk) = S$, and in LTL as $AG(\neg milk) = S$ [4]. To verify this property, we need to show that only $EF(milk) = S$ is satisfiable.

[3] Note that all the figures will use T, S, F instead of $\top, S, \perp$ due to limitations of drawing tool.
[4] Note that $EF(milk) = \neg AG(\neg milk)$, and $\neg S = S$.

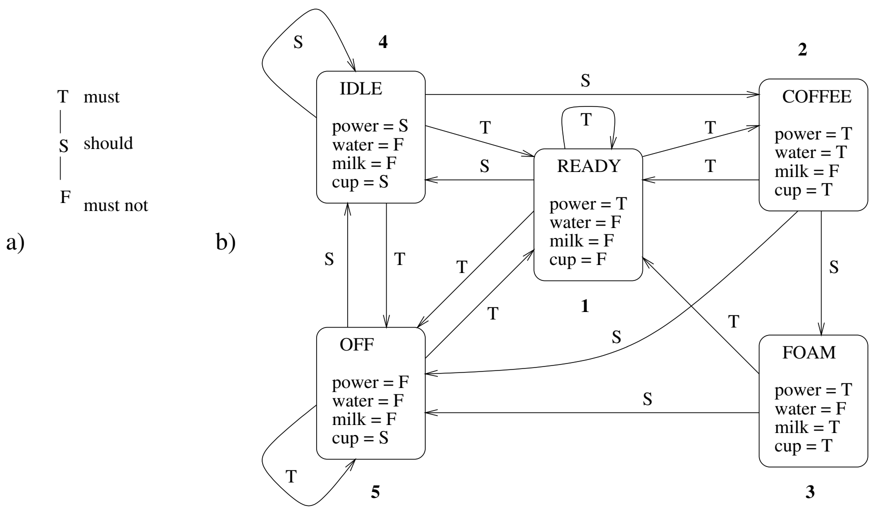

Fig. 1. (a) A lattice of truth values; (b) The coffee dispenser model that uses it.

4.2 Tableau Construction

We will first define the procedures to construct the tableau, then apply the procedures to the coffee dispenser example.

Define the Construction Procedures Let f be a restricted path formula (see definition in Section 2. AP_f denotes the set of atomic propositions of f. The tableau associated with f is a structure $T = (S_T, R_T, L_T)$ with AP_f as its set of atomic propositions. Each state in the tableau is a permutation with respect to the membership degrees of the set of *elementary formulas* obtained from f. The definition of the set of elementary formulas of f denoted by $el(f)$ remains the same as in section 2.1. Recall the recursive definition of $el(f)$:

- $el(p) = \{p\}$ if $p \in AP_f$.
- $el(\neg g) = el(g)$.
- $el(g \vee h) = el(g) \cup el(h)$.
- $el(Xg) = \{Xg\} \cup el(g)$.
- $el(gUh) = \{X(gUh)\} \cup el(g) \cup el(h)$.

Let A be a mv-set containing all of the elementary formulas of f. The state space of the tableau S_T is the set of all permutations of A with respect to the degree of the elements. The labeling function L_T at each state is defined to be the tuples of atomic propositions and their degrees in the state. Eg., $L_T = \{(milk, \top), \ldots\}$.

In order to construct the transition relation R_T, we need to redefine the function sat using multi-valued logic Λ.

Definition 7. *The function* sat *associates with each subformula g of f a set of states in S_T for each truth value. Let ℓ be a truth value in Λ.*

$$\begin{aligned} sat(g = \ell) &\triangleq \{s \mid g \in_L s = \ell\} \text{ where } g \in el(f).\\ sat(\neg g = \ell) &\triangleq \{s \mid g \in_L s = \neg\ell)\}.\\ sat(g \sqcup h = \ell) &\triangleq \{s \mid (g \in_L s) \sqcup (h \in_L s) = \ell\}.\\ sat(g \sqcap h = \ell) &\triangleq \{s \mid (g \in_L s) \sqcap (h \in_L s) = \ell\}.\\ sat(gUh = \ell) &\triangleq sat(h = \ell) \cup (sat(g = \ell) \cap sat(X(gUh) = \ell)). \end{aligned}$$

$$sat(g \sqsupseteq \ell) \triangleq \bigcup_{\forall \ell_1 \sqsupseteq \ell} sat(g = \ell_1).$$

$$sat(g \sqsubseteq \ell) \triangleq \bigcup_{\forall \ell_1 \sqsubseteq \ell} sat(g = \ell_1).$$

Immediately, we can deduce the following

$$\begin{aligned} sat(g \sqcup h = \ell) &= sat(g = \ell) \cup sat(h = \ell) \backslash (sat(g \sqsupset \ell) \cup sat(h \sqsupset \ell)).\\ sat(\neg(g \sqcup h) = \ell) &= sat(\neg g \sqcap \neg h = \ell). \end{aligned}$$

Since each state in the tableau is a mv-set, the transition relation between two states is an mv-relation. Recall that if $B(\mathcal{R}, \mathcal{Q}) = \mathcal{P}$, $\mathcal{R}$ defines the transition relation from P to Q. We define the tableau transition relation R_T as follows.

Definition 8. *Let two states s and y, and truth value $\ell \in \mathcal{L}$ be given. For each element Xg in $el(f)$, the transition relation from s to y, $r = (s, y)$, must satisfy the following*

- $(r \in_\Lambda R_T) \sqcap (g \in_\Lambda y) = (Xg \in_\Lambda s)$.
- $\forall s : B(r, y) \cdot (Xg \in_\Lambda s = \ell) \Rightarrow (r \in_\Lambda R_T \sqsupseteq \ell) \wedge (g \in_\Lambda y \sqsupseteq \ell)$

Using the second condition, we identified the following sat sets, and using the first condition, we identified the transition relations amongst these sat sets. Using the 3-valued logic shown in Figure 1(a), we obtain the following

1. $sat(Xg = \top) \xmapsto{\top} sat(g = \top)$
2. $sat(Xg = S) \xmapsto{S,\top} sat(g = S)$ $\qquad sat(Xg = S) \xmapsto{S} sat(g \sqsupseteq S)$
3. $sat(Xg = \bot) \xmapsto{\bot,S,\top} sat(g = \bot)$ $\qquad sat(Xg = \bot) \xmapsto{\bot} sat(g \sqsupseteq \bot)$

Where $LHS \xmapsto{\top} RHS$ is parsed to be the transition relation from LHS to RHS has value $\top$.

Application to the Coffee Dispenser Example In the coffee dispenser model, we want to check the property $EF(milk)$. Let $g = \top\, U\, milk$ [5] be a specification. Figure 2 gives the transition relation R_T for the tableau of the formula g. $el(f) = \{milk, Xg\}$. Each permutation of the membership degrees of set $el(g)$ is a state. When labeling the states in Figure 2, we use m as an abbreviation for $milk$. The truth value of each transition

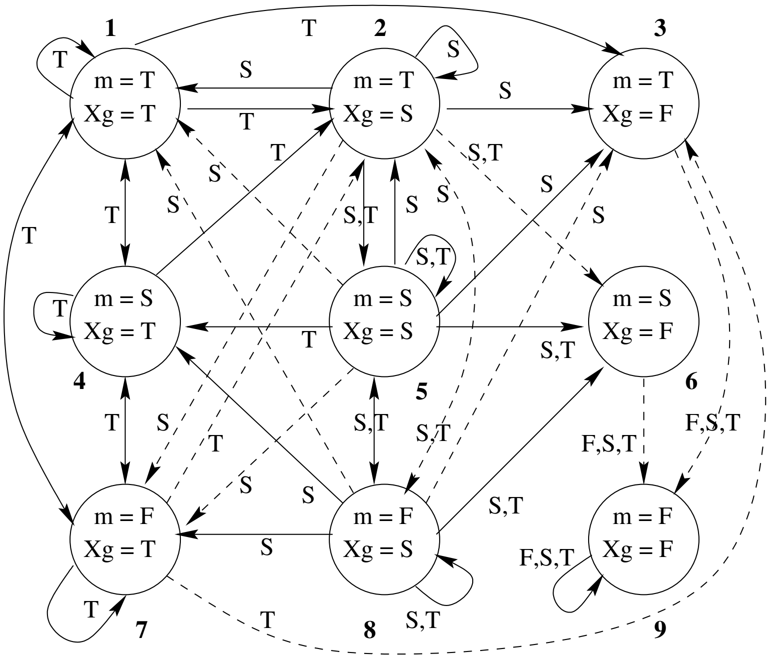

Fig. 2. Tableau for $g = \top$ U $milk$.

relation is labeled with an arrow edge directing from the originating state to the next state. For clarity we eliminated transition relations that have value $\perp$.

The required sat sets are (each number refers to a state in Figure 2):

$$sat(Xg = \top) = \{1, 4, 7\} \qquad sat(g = \top) = \{1, 2, 3, 4, 7\}$$
$$sat(Xg = S) = \{2, 5, 8\} \qquad sat(g = S) = \{5, 6, 8\}$$
$$sat(Xg = \perp) = \{3, 6, 9\} \qquad sat(g = \perp) = \{9\}$$

To build the tableau with all transitions more systematically, we rely on the sat sets. Each $sat(g = \ell)$ set represents the states in which g holds the value ℓ. The transitions between sat sets are illustrated in Figure 3.

Unfortunately, the definition of R_T does not guarantee that *eventuality* properties are fulfilled. We can see this behavior in Figure 2. Although state 7 belongs to $sat(g = \top)$, the path that loops forever in state 7 does not satisfy the formula g since $milk$ never has the truth value $\top$ along that path, similarly to states 4 and 8. Consequently, an additional condition is necessary in order to identify those paths along which f has a truth value that is the minimum of what is being verified. A path π that starts from a state $s \in sat(f = \ell)$, where $\ell \in \mathcal{L}$ satisfies $f = \ell$ if and only if

- For every subformula gUh of f and for every state s on π, given a truth value $\ell \in \mathcal{L}$, if $s \in sat(gUh = \ell)$ then either $s \in sat(h = \ell)$ or there is a later state t on π such that $t \in sat(h = \ell)$.

[5] Note that $\neg AG(\neg milk) = EF(milk) = E(\top\, U\, milk)$

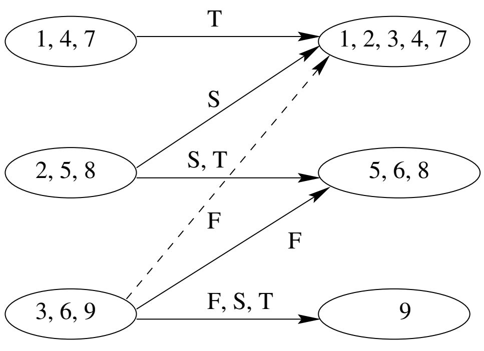

Fig. 3. Transition relations between sat sets.

4.3 Construct the Product

We now compute the product $P_\ell = (S_P, R_P, L_P)$ of the tableau $T = (S_T, R_T, L_T)$ and the Kripke structure $M = (S_M, R_M, L_M)$ for the truth value $\ell \in \mathcal{L}$.

- $S_P = \{(s, s') \mid s \in S_T, s' \in S_M$ and $L_M(s') \mid_{AP_f} = L_T(s)\}$.
- $R_P((s, s'), (t, t'))$ iff $R_T(s, t) \sqcap R_M(s', t')$.
- $L_P((s, s')) = L_T(s)$.

The transition relation of this product may fail to be total. If this happens, we remove from S_P all those states that do not have successors and restrict the transition relation R_P to the remaining states.

We extent the definition of the function sat to be over the set of states S_P of P by $(s, s') \in sat(g = \ell)$ if and only if $s \in sat(g = \ell)$.

We next apply CTL model checking and find the set of all states V in $P, V \subseteq sat(f \sqsupseteq \ell)$, that satisfy $EG\, true$ with the fairness constraints:

$$\{sat(\neg(gUh) \sqcup h \sqsupseteq \ell) \mid gUh \text{ occurs in } f\}$$

Each of the states in V is in $sat(f \sqsupseteq \ell)$. Moreover, it is the start of an infinite path that satisfies all of the fairness constraints. These paths have the property that no subformula gUh is at least as true as ℓ on the path while h remains strictly more false than ℓ.

To illustrate this construction, we check the formula $g = (\top\, U\, m)$. First, we must check that $Eg = \top$ is not satisfiable. The computed product $P_\top$ of the truth value $\top$ for the coffee dispenser Kripke structure M in Figure 1 and the tableau T in Figure 2 is shown in Figure 4. Each state in the product is marked by a pair of states (s, s') where $s \in T$ and $s' \in M$. We have omitted the states $(3, 3), (8, 5), (9, 1), (9, 2), (9, 4), (9, 5)$ from the product structure since they are not the beginning of an infinite path. We use the CTL model checking algorithm to find the set V of states in $sat(g \sqsupseteq \top)$ that satisfy the formula $EG\, true$ with the fairness constraint $sat(\neg(\top U milk) \sqcup milk \sqsupseteq \top)$. Because $sat(g \sqsupseteq \top) = \{(1, 3), (2, 3), (3, 3)\}$, but none of these states satisfies the fairness constraint, $V = \emptyset$. This shows that no state in M satisfies $EF(milk) = \top$. Thus we go on to check the next truth value S.

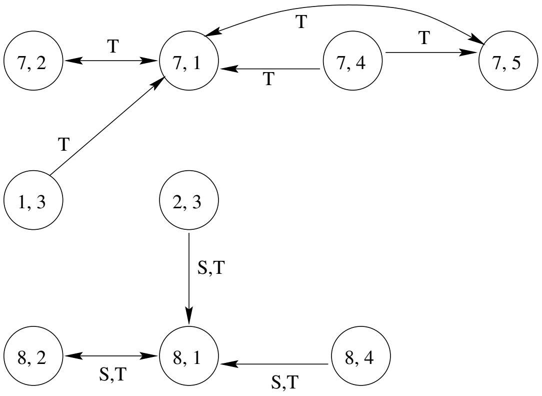

Fig. 4. The product $P_\top$ of the coffee dispenser M and the tableau T before applying the fairness constraint.

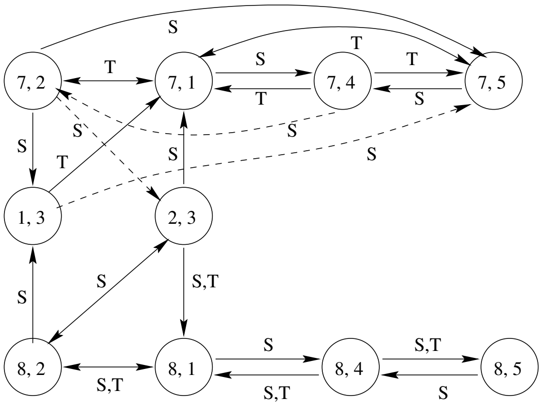

Fig. 5. The product P_S of the coffee dispenser M and the tableau T after applying the fairness constraint.

The product P_S of the truth value S for the coffee dispenser Kripke structure M in Figure 1 and the tableau T, is shown in Figure 5. We have omitted the states (3,3), (9,1), (9,2), (9,4), (9,5) from the product structure since they are not the beginning of an infinite path. Likewise, we use the CTL model checking algorithm to find the set V of states in $sat(g \sqsupseteq S)$ that satisfy the formula $EGtrue$ with the fairness constraint $sat(\neg(\top U milk) \sqcup milk \sqsupseteq S)$. Because $sat(g \sqsupseteq S) = \{(1,3),(2,3),(3,3)\}$ but $(3,3)$ is not the beginning of an infinite path, $V = \{(1,3),(2,3)\}$. We can therefore conclude that state 3 in M satisfies $EF(milk) = S$ and therefore all states satisfy $AG(\neg milk) = S$.

4.4 Correctness of the Algorithm

Let $\models_l$ denote the entailment that has at most the truth value of ℓ.

The key property of the tableau is summarized in the following theorem.

Theorem 3. *Let T be the tableau for the path formula f. Then, for every Kripke structure M and every path π' of M, if $M, \pi' \models_l f$ then there is a path π in T that starts in a state in $sat(f \sqsupseteq l)$, such that $label(\pi') \mid_{AP_f} = label(\pi)$.*

The correctness of the construction of the product P_ℓ with respect to $\ell \in \mathcal{L}$ is summarized in the following theorem.

Theorem 4. *$M, s' \models_l E\,f$ if and if only there is a state s in T such that $(s, s') \in sat(f \sqsupseteq \ell)$ and $P, (s, s') \models_l EG\,true$ under fairness constraints $\{sat(\neg(gUh) \sqcup h \sqsupseteq \ell) \mid gUh$ occurs in $f\}$, and l is the highest ranked such truth value.*

Proofs of the theorems are required, which will be given in our future work.

5 Conclusion and Future Work

In this paper, we used the tableau approach to explore the multi-valued symbolic LTL model checking. We have presented the construction details of the tableau and the product structures. Property of the coffee dispenser example is successfully checked using this algorithm. This has proved the feasibility of such approach to the multi-valued symbolic LTL model checking. Our future work will complete the algorithm by providing formal proofs to ensure the correctness.

References

1. L. Bolc and P. Borowik. *Many-Valued Logics*. Springer-Verlag, 1992.
2. M. Chechik, B. Devereux, and S. Easterbrook. “Implementing a Multi-Valued Symbolic Model-Checker”. In *Proceedings of TACAS'01*, April 2001.
3. M. Chechik, S. Easterbrook, and V. Petrovykh. “Model-Checking Over Multi-Valued Logics”. In *Proceedings of FME'01*, March 2001.
4. M. Chechik and B. Devereux and S. Easterbrook and A. Lai and V. Petrovykh. “Efficient Multiple-Valued Model-Checking Using Lattice Representations”. (submitted for publication), January 2001.
5. E. Clarke, O. Grumberg, and D. Peled. *Model Checking*. MIT Press, 1999.
6. S. Easterbrook and M. Chechik. “A Framework for Multi-Valued Reasoning over Inconsistent Viewpoints”. In *Proceedings of International Conference on Software Engineering (ICSE'01)*, May 2001.