IJCSI International Journal of Computer Science Issues, Vol. 7, Issue 3, No 1, May 2010
ISSN (Online): 1694-0784
ISSN (Print): 1694-0814

Distributed Maximality based CTL Model Checking

Djamel Eddine Saidouni[1] ,Zine EL Abidine Bouneb[2]

**[1] Department of Computer Science , University Mentouri
Constantine, 25000, Algeria**

**[2] Department of Computer Science , University El Arbi Ben Mehidi
Oum El Bouaghi, 04000, Algeria**

Abstract

In this paper we investigate an approach to perform a distributed CTL Model checker algorithm on a network of workstations using Kleen three value logic, the state spaces is partitioned among the network nodes, We represent the incomplete state spaces as a Maximality labeled Transition System MLTS which are able to express true concurrency. we execute in parallel the same algorithm in each node, for a certain property f on an incomplete MLTS , this last compute the set of states which satisfy f or which if they fail f are assigned the value $\perp$ .The third value $\perp$ mean unknown whether true or false because the partial state space lacks sufficient information needed for a precise answer concerning the complete state space .To solve this problem each node exchange the information needed to conclude the result about the complete state space. The experimental version of the algorithm is currently being implemented using the functional programming language Erlang.

Keywords: *Author Guide, Article, True concurrency semantics; State space explosion problem; Distributed model checking; three value logic.*

1. Introduction

Model checking is powerful technique for verifying reactive systems able to find subtle errors in real commercial designs, it is gaining wide industrial acceptance. Compared to other formal verification (e.g theorem proving) Model checking is largely automatic[1][2].In our approach the application to be verified is firstly specified by means of the formal description technique LOTOS[3][4] . This specification is translated, using the maximality based operational semantics, to a graph called Maximality-based Labeled Transition System (MLTS)[5] . This graph is used for the properties verification.

The main limiting factor of Model checking technique is the so called explosion problem where translation from the specification of the application to a state transition graph usually involves an exponential blow-up. State space does not fit into memory or state space fits in memory, but is too large for being explored entirely (e.g., access to hash table becomes slower as the number of states grows).

Three approaches has been proposed in the literature for tackling this problem, the first one uses some equivalence relation to reduce the number of states and transitions in the model (bisimulation relations, alpha reduction relation, partial order based relations, ...)[6][7][8]. The second approach consists of coding the model in an efficient representation like binary decision diagram (BDD) [9][10][11].

To overcome hardware limitations, a third approach is deeply investigated currently. This approach consists of using a cluster or a network of workstations. This last technique has showed its efficiency since it can preserve the result of the first and second approach with increasing performance[12][13] [14].

In this paper we continue our work for the parallelization of the model checking based on the maximality semantics the first step for the parallelization of the construction of the state space, which is modeled as Maximality Labeled Transition System has achieved with success, for more information we refer the reader to [14], in this paper we present the second step which is the parallelization of the Model checking verification algorithm discussed in [1].

First the state graph is partitioned among the network nodes, i.e. each network node owns a subset of the state space. Each node executes an instance of the parallel generation algorithm which computes partial MLTS[14].

Secondly we execute in parallel the same CTL Model checker algorithm in each node on these incomplete structure, this last use three value logic of Kleen [15][16] and return $\perp$ only when the partial state space lacks information needed for a defined answer about the complete state space. The algorithm exchange information about Border States which is not present in

the node to conclude the result about the complete state space; if an arbitrary node has new information he need to make a re-computation.To the best of our knowledge our Algorithm of verification is the first fix point algorithm of model checker which can be executed in parallel on Maximality Labeled Transition System.

2. Maximality Semantics

We assume that the reader is familiar with behavioural part of LOTOS and its interleaving semantics.

2.1 Maximality based Labeled Transition System

M being a countable set of events names, a maximality-based labeled transition system of support M is a quintuplet $(\Omega, A, \mu, \xi, \psi)$ with :

$\Omega = (S, T, \alpha, \beta)$ is a transition system such that :

S : the countable set of states in which the system can be.

T : the countable set of transitions indicating the change of system states.

α and β are two functions from T to S such that : for any transition $t \in T$; $\alpha(t)$ denotes the origin of the transition and $\beta(t)$ its goal.

(Ω, A) is a transition system labeled by an alphabet A.

$\psi : S \to 2^m$: is a function which associates to every state a finite set of maximal event names present at this state.

$\mu : T \to 2^m$:is a function which associates to every transition a finite set of event names corresponding to actions that have start their execution such that their terminations allow the start of this transition.

$\xi : T \to M$: is a function which associates to its transition an event name identifying its occurrence. Such that for any transition $t \in T$, $\mu(t) \in \psi(\alpha(t))$, $\xi(t) \notin \psi(\alpha(t)) - \mu(t)$ and $\psi(\beta(t)) = (\psi(\alpha(t)) - \mu(t)) \cup \{\xi(t)\}$.

2.2 The intuition behind the Maximality semantics

The semantics of a concurrent system can be characterized by the set of states of the system and transitions by which the system passes a state to another. In the approach based on the maximality, transitions are events that only represent the beginning of the execution of actions. Consequently, the concurrent execution of several actions becomes possible; hence we can distinguish sequential executions and parallel executions of actions. Being given that several actions have the same name can be executed in parallel (auto concurrency), we associate, to distinguish the executions of each action, an identifier to every beginning of the execution of that action. In a state, an event is said maximal if it corresponds to the beginning of the execution of an action that can be possibly always executing in this state. In order to illustrate this semantics let us consider the following example :

F = a; b ; stop [] b; a ;stop E = a ; stop ||| b ; stop

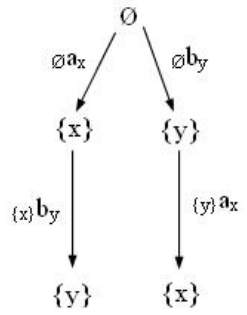

Fig.1 F = a; b ; stop [] b; a ;stop.

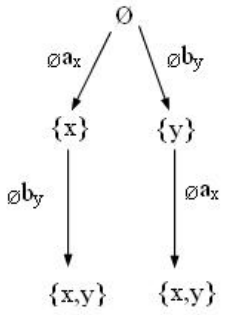

Fig.2 E = a ; stop ||| b ; stop

Fig.1 represents the MLTS of the LOTOS behavioral expression F and Fig.2 represents the MLTS of the LOTOS behavioral expression E. It is clear that in states 2 and 4 of Fig.2 actions a and b are currently executed in parallel this fact is represented by the presence of the two event names x and y in each states . However, in states 2 and 4 of Fig.1, only one action may be in execution, this fact is captured by the presence of one event name in each state. A detailed presentation of the maximality semantics can be found in .

The maximality based operational semantics of LOTOS is defined on configurations associated to behavior expressions. For illustration, let us reconsider the

behavioral expressions E and F . In the initial state, no action has been executed again, therefore the sets of maximal event names associated to the initial states are empty, hence the initial configurations associated to the behavior expressions E and F are ${}_{\varnothing}[E]$ and ${}_{\varnothing}[F]$. So *a configuration represents a state.* When applying the maximality base operational semantics, the following derivations are possible :

$${}_{\varnothing}[E] \overset{\varnothing\, a_x}{\bullet}_m \qquad {}_x[stop] \,|||_{\varnothing}\, [b;stop] \qquad \overset{\varnothing\, b_y}{\bullet}_m \qquad {}_x[stop] \,|||_y\, [stop] \quad (1)$$

x (respectively y) being the name of the event identifying the beginning of the action "a" (respectively "b""). Note that nothing can be concluded on the termination of the two actions a and b in the configuration :

$${}_x[stop] \,|||_y\, [stop] \quad (2)$$

x and y are said maximal in this configuration. Let's note that x is also maximal in the intermediate state represented by the configuration:

$${}_x[stop] \,|||_{\varnothing}\, [b;stop] \quad (3)$$

For the implementation we can implement events as integer.

Definition 1. A Kripke structure M is a tuple (S, L, R, I) , where S is a finite set of states, $L : S \times AP \to \{true, false\}$ is an interpretation that associates a truth value in $\{true, false\}$ with each atomic proposition $P \in AP$ the set of all atomic proposition, for each state in S , $R \subseteq S \times S$ is a transition relation on S , and $I \subseteq S$ is a set of initial states.

3. Maximality Labeled Transition System as Kripke Structure

let M=($\Omega, A, \mu, \xi, \psi$) be an MLTS such that $\Omega = (S, T, \alpha, \beta)$ and let K= (S, L, R, I) be a kripke structure, it is clear to see that if we take from M the maximal events as atomic proposition we can consider M as a kripke structure defined by : (S, ψ, T, I).

Example 1. : we take the example of the MLTS in Fig.2 this MLTS can be seen as kripke structure like this :

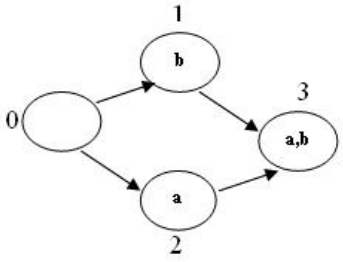

Fig .3 The MLTS of E = a ; stop ||| b ; stop as Kripke structure

The atomic propositions are based on the content of states, since we can define each state as a function:

This function answer the question: "is the arbitrary action "a" currently executed in the state i?" to make an idea let consider the MLTS of Fig 2:

$$state_2(a) = True; state_2(b) = True$$

Whereas:

$$state_1(a) = True; state_1(b) = False$$

for example if "a" = A-InCriticalSection and "b" = B-InCriticalSection we can see that the state 2 violate the principle of mutual exclusion (safety property : some thing bad never happen) . We make the remark that the model of MLTS is very rich of information and also can be used for the scheduling in multiprocessor platform, since it represent also the dependence between action. The main advantage is that this graph can be generated automatically by a compiler. Our use of MLTS here as logical Model for verification is very simple than scheduling since we don't need all the information contained in it.

Definition 2. Let $M = (S, L, R, I)$ a Kripke structure, the set of border states in M is $border(M) = \{s \in S \mid \neg \exists s'.(s, s') \in R\}$.[19]

Definition 3. Let $M = (S, L, R, I)$ be a Kripke structure, $T \subseteq S$. We define the partial kirpke structure $\Phi_M(T) = (S_T, L_T, R_T, I_T)$ as follows:

$$S_T = \{s \mid s \in T or \exists s' \in T : (s', s) \in R\}$$
$$R_T = \{(s_1, s_2) \in R \mid s_1 \in T, s_2 \in S_T\}$$
$$I_T = \{s \in S_T \mid s \in I\}$$
$$L_T : S_T \times P \to \{false, \perp, true\}$$

We call the partial Kripke structure $\Phi_M(T)$ a *fragment* of M, the states in the set T are all present in the Node and $S_T - T$ is the set of border states of $\Phi_M(T)$ i.e $border(\Phi_M(T))$. From the definition of R_T we can see that the fragment $\Phi_M(T)$ know all the *immediate successors* of the states present in the node i.e in T [19]. The truth function L_T for the fragment of M and a CTL formula φ is a total function

$L_T : S \times \varphi \to \{True, \perp, False\}$.

$L_T(s,\varphi) = True$ iff $M, s \Box \varphi$ and $L_T(s,\varphi) = False$ iff $M, s \not\Box \varphi$ we use $L_T(s,\varphi) = \perp$ if we don't know the truth value at certain stage of computation of the truth function for example in the start of computation of the truth on border states.

The truth function on the complete Kripke structure M is a total function $L : S \times \varphi \to \{True, False\}$. since all the information for the computation needed are available we don't need the third value unknown represented by $\perp$.

Definitions .4 Let $M = (S, L, R, I)$ be a Kripke structure,and $\Phi_M(Z)$ a fragment of M we define :

$T(p) = \{s \in S \mid L(s,p) = true\}$
$U(p) = \{s \in S \mid L(s,p) = \perp\}$
$F(p) = \{s \in S \mid L(s,p) = false\}$

for $s \in T, inT(s) = (1,s)$
for $s \in U, inU(s) = (\perp, s)$
for $s \in F, inF(s) = (o,s)$

$T^+(p) = \{inT(s) \mid \forall s \in T(p)\}$
$U^+(p) = \{inU(s) \mid \forall s \in U(p)\}$
$F^+(p) = \{inF(s) \mid \forall s \in F(p)\}$
$S^+(p) = T^+(p) \cup U^+(p) \cup F^+(p)$
$\forall e_1, e_2 \in S^+$ we have $e_1 B e_2$ iff $snd(e_1) = snd(e_2)$

We Interpret the logical operators $\wedge$ and $\vee$ on partial kripke structure using Kleene's three value logic . An accurate and compatible interpretation of Kleene's connectives was given by Korner [15].Korner defined the notion of an inexact class of a given non-empty domain A generated by a partial definition D(P) of a property P of elements of A as a three-valued ``characteristic function'

$X_p : A \to \{-1, 0, 1\}$

$$X_p(a) = \begin{cases} -1 & when\ P(a)\ according\ to\ D(P)\ is\ false, \\ 0 & when\ P(a)\ is\ D(P) - undecidable, \\ 1 & when\ P(a)\ according\ to\ D(P)\ is\ true. \end{cases}$$

Any family of inexact classes of a given domain *A* is a de Morgan lattice, the algebraic operations $\cup, \cap$ and $-$:

$(X \cup Y)(a) = max\{X(a), Y(a)\}$
$(X \cap Y)(a) = min\{X(a), Y(a)\}$
$(-X)(a) = -X(a)$

being counterparts of the Kleene connectives[15]. We now consider our theory based on the theory mentioned above :

Let $B_\perp = \{false, \perp, true\}$.
Let $(B_\perp, <)$ be a total order such that :

$false < \perp < true$

$\forall e_1, e_2 \in S^+$ in the case when $e_1 B e_2$ we have :

$e_1 \wedge e_2 = (min(fst\ e_1, fst\ e_2), snd\ e_1)$
$e_1 \vee e_2 = (max(fst\ e_1, fst\ e_2), snd\ e_1)$

Let $S \subseteq S^+$ and $G \subseteq S^+$ we define :

$S \lceil G = \{e_1 \wedge e_2 \mid \forall e_1 \in S, \forall e_2 \in G\ such\ that\ e_1 B e_2\}$
$S \lfloor G = \{e_1 \mid \forall e_1 \in S\ and\ \nexists e_2 \in G\ such\ that\ e_1 B e_2\}$
$\cup \quad \{e_2 \mid \forall e_2 \in G\ and\ \nexists e_1 \in S\ such\ that\ e_1 B e_2\} \quad \cup$
$\{e_1 \vee e_2 \mid \forall e_1 \in S, \exists e_2 \in G\ such\ that\ e_1 B e_2\}$

Let $succ$ be a function defined as follow :

succ : $S^+ \to 2^{S^+}$
$succ(e) = \{e'^+ \mid (snd(e), snd(e')) \in R\}$

4. CTL model checking on fragments

Theorem.1 Given $M = (S, L, R, I)$ a fragment of Kripke structure and a CTL formula, the following recursive algorithm compute the set of states $H(f) \subseteq S$ which satisfy f or it may satisfy f and exclude all states which not satisfy f.

• $H(p) = T^+(p) \cup U^+(p)$ such that p is an atomic proposition

• $H(\neg f) = \{inT(s) \mid s \in (S - map(snd, H(f))\} \cup U^+(f)$

• $H(f \wedge g) = H(f) \sqcap H(g)$

• $H(f \vee g) = H(f) \sqcup H(g)$

•

$H(AXf) =$

$(\{InT(snd(e)) \mid \forall e \in S^+(f).succ(e) \subseteq T^+(f) \subseteq H(f)\}$

$\cup \{inU(snd(e)) \mid \forall e \in S^+(f).succ(e) \subseteq (T^+(f) \cup U^+(f))$

$and \ \exists e' \in succ(e) \ such \ that \ e'^+(f)\}) \sqcup$

$\{inU(s) \mid s \in border(M)\}$

•

$H(EXf) = (\{InT(snd(e)) \mid \forall e \in S^+(f).\exists e' \in succ(e) \ and \ e'^+(f)\}$

$\cup$

$\{inU(snd(e)) \mid \forall e \in S^+(f).succ(e) \subseteq U^+(f) \subseteq H(f)\})$

$\sqcup$

$\{inU(s) \mid s \in border(M)\}$

• $H(AGf) = \upsilon Z.(H(f) \sqcap AXZ)$

• $H(AFf) = \mu Z.(H(f) \sqcup AXZ)$

• $H(A(fUG)) = \mu Z.(H(g) \sqcup (H(f) \sqcap AXZ))$

After the application of the above recursive algorithm we have $\forall s \notin H(f) \Rightarrow L(s, f) = false$. The other operators like EG can be all deduced from the operators cited above, for example $H(EGf) = H(\neg(AF\neg f))$ for more information we refer the reader to [2].

proof

• For the atomic proposition we can see that $H(p)$ is the set of states where the formula hold or where the formula may hold i.e where we are not sure that the formula is false

• For the case of $H(AXf)$: The set $(\{InT(snd(e)) \mid \forall e \in S^+(f).succ(e) \subseteq T^+(f) \subseteq H(f)\}$ represent the set of states where all of there successors are states where the formula f holds, and the set $\{inU(snd(e)) \mid \forall e \in S^+(f).succ(e) \subseteq (T^+(f) \cup U^+(f)) \ and \ \exists e' \in succ(e) \ such \ that \ e'^+(f)\}$ represent the set of states where there successors may satisfy f, hence this set is the set of states where $H(AXf)$ may be satisfied. Furthermore because we don't know the successors of the border states we add this sates to the result of computation $\{inU(s) \mid s \in border(M)\}$ because this states may satisfy the formula $H(AXf)$.

• We will prove the fix point characterization of the operators AGf, the fix point characterization for the remaining CTL operators can be established in similar manner. The set 2^S of all subset of S form a lattice under the set inclusion ordering. Each element s' of the lattice can also be thought of as a predicate on S where the predicate is viewed as being true or may be true $\perp$ for exactly the states in s'. The least element in the lattice is the empty set, which we also refer to as False , and the greatest element in the lattice is the whole set S, which we sometimes write as True. A function that maps 2^S to 2^S will be called a predicate transformer. We follow the same manner as in [2] first we can see that $\tau(Z) = H(f) \sqcap AXZ$ is monotonic, and $\sqcap$-continuous by the theorem of Tarski and Knaster we can conclude that AGf is the great fix point of. $\tau(Z) = H(f) \sqcap AXZ$

proposition 1:The predicate transformer $\tau(Z) = H(f) \sqcap AXZ$ is monotonic

proof

let $P_1 \subseteq P_2$ To show that $\tau(P_1) \subseteq \tau(P_2)$, consider an arbitrary sate $s \in \tau(P_1)$.Then s satisfy f or it may satisfy f. $i.e(L(s, f) = true \ or \ L(s, f) = \perp)$ and for all states s' such that $(s, s') \in R$ and $s' \in P_1$. Because $P_1 \subseteq P_2$, $s' \in P_2$ as well thus $s' \in \tau(P_2)$

proposition 2:The predicate transformer $\tau(Z) = H(f) \sqcap AXZ$ is $\sqcap$-continuous

proof

We want to proof that $\tau(\sqcap_i P_i) = \sqcap_i \tau(P_i)$.

first we can see that $(P_1 \sqcap P_2) \subseteq P_1$ because τ is monotonic we have $\tau(P_1 \sqcap P_2) \subseteq \tau(P_1)$ the same for $\tau(P_1 \sqcap P_2) \subseteq \tau(P_2)$ which mean that $\tau(P_1 \sqcap P_2) \subseteq (\tau(P_1) \sqcap \tau(P_1))$ more generale we have $\tau(\sqcap_i P_i) \subseteq \sqcap_i \tau(P_i)$. Furthermore we have for an arbitrary state $s \in (\tau(P_1) \sqcap \tau(P_2) \sqcap ... \tau(P_n))$ this mean that : [illegible] and $s \in \tau(P_2) \ldots s \in \tau(P_n)$ this mean that: $(s \in \tau(P_1) \Leftrightarrow s \Box_{may} f$ and $\forall s'$ such that

$(s,s') \in R$ implies that $s' \in P_1)$ and ... $(s \in \tau(P_n) \Leftrightarrow s \Box_{may} f$ and $\forall s'$ such that $(s,s') \in R$ implies that $s' \in P_n)$.

Hence $s \in \tau(P_2) ... s \in \tau(P_n)$ implies $s \Box_{may} f$ and $s' \in (P_1 \cap P_2 ... \cap P_n)$ which mean that $s \in \tau(\cap_i P_i)$ i.e $\cap_i \tau(P_i) \subseteq \tau(\cap_i P_i)$ Ω

With an informal way we can see that at the first iteration we have all the state which satisfy f or it may satisfy f, lets said that this set is Z_1, at the second iteration we compute the set of states which is in Z_1 and there successor satisfied f or it may satisfied f i.e there successor is in Z_1, we forward the computation on the successors until we reach a fix point. Hence we understand that we have giving to the operators AGf the semantics, that we look for the states s, which has the property that all the states of the paths stemming from s satisfy f or it may satisfy f. Ω

Example 2 lets take the fragment of the Kripke Structure M shown on Figure 8 in node 1, and the property to be checked on this fragment is $AG(a \vee c)$:

$H(a) = T^+(a) \cup U^+(a)$, $T^+(a) = \{(1,2)\}$,
$U^+(a) = \{(\perp,3),(\perp,4)\}$
$H(a) = \{(1,2),(\perp,3),(\perp,4)\}$
$H(c) = \{(\perp,3),(\perp,4)\}$, $H(a \vee c) = H(a) \cap H(c)$
$H(a \vee c) = \{(1,2)\} \cup \{(\perp,3) \vee (\perp,3),(\perp,4) \vee (\perp,4)\}$
$H(a \vee c) = \{(1,2),(\perp,3),(\perp,4)\}$
$H(AG(a \vee c)) = \upsilon Z.(\{(1,2),(\perp,3),(\perp,4)\} \cap AXZ)$
$Z_0 = \{(1,1),(1,2),(1,3),(1,4)\}$
$AXZ_0 = \{(1,1),(1,2),(1,3),(1,4)\} \cap \{(\perp,3),(\perp,4)\}$
$AXZ_0 = \{(1,1),(1,2),(1,3),(1,4)\}$
$Z_1 = \{(1,2),(\perp,3),(\perp,4)\} \cap AXZ_0$
$Z_1 = \{(1,2),(\perp,3),(\perp,4)\}$
$AXZ_1 = \{(1,1),(\perp,2)\} \cap \{(\perp,3),(\perp,4)\}$
$AXZ_1 = \{(1,1),(\perp,2),(\perp,3),(\perp,4)\}$
$Z_2 = \{(1,2)(\perp,3),(\perp,4)\} \cap AXZ_1 = \{(\perp,2),(\perp,3),(\perp,4)\}$
$Z_3 = \{(\perp,2),(\perp,3),(\perp,4)\}$
$Z_3 = Z_2$

Fix point reached, the algorithm stop the computation, from the precedent theorem. we conclude that, $L(1, AG(a \vee c)) = false$ so the final result of computation on the fragment in node 1 is :

$\{(0,1),(\perp,2),(\perp,3),(\perp,4)\}$

5. Distributed CTL Model checking

The main idea of the distributed verification algorithm is that if we want to check some formula φ in some state s see figure 4, it is clear that the truth of formula depend on the truth of this formula in s' which is in node II. Hence we start the computation in node I with $L(s',\varphi) = \perp$, i.e we consider that the formula ϕ may hold in s'. when the node II finish the computation, if the formula hold in s', the node number I make a recomputation and found that the formula hold in s for example in the case of $\varphi \in \{EGf, AGf, AFf, EFf, A(fUg), E(fUg)\}$ and the result of the first computation in node I is $\perp$. If the formula don't hold in s' and the result of the first computation in node I is $\perp$ this mean that the formula don't hold in s. The main difference between the reasoning algorithm on fragments and the distributed Algorithm is that in the case of $\varphi \in \{EGf, AGf, AFf, EFf, A(fUg), E(fUg)\}$ we consider in the fragments algorithm that f may hold in border states, but in the distributed version we consider the whole formula φ not only f may hold in border states and the truth on border states is parameter passed to the predicate transformer as follow :

- $AFp = \lambda Y.\mu Z.(p \vee Y \vee AXZ)$
- $EFp = \lambda Y.\mu Z.(p \vee Y \vee EXZ)$
- $AGp = \lambda Y.\upsilon Z.(p \vee Y \wedge AXZ)$
- $EGp = \lambda Y.\upsilon Z.(p \vee Y \wedge EXZ)$

- $A(p \cup q) = \lambda Y.\mu Z.(q \vee Y \vee (p \wedge AXZ))$
- $E(p \cup q) = \lambda Y.\mu Z.(q \vee Y \vee (p \wedge EXZ))$

where

$Y = \{s \in border(M) \mid L(s,\varphi) = True \text{ or } L(s,\varphi) = \bot\}$

and φ is an arbitrary formula represented by one of the six operator described above respectively, here Y represent the missing part of information in border sates ,if some one give us the set of border sates where the formula to be verified is valid we can conclude the truth of the formula on the whole Kripke structure, this fact can be represented as the application of model checking function to the given information.

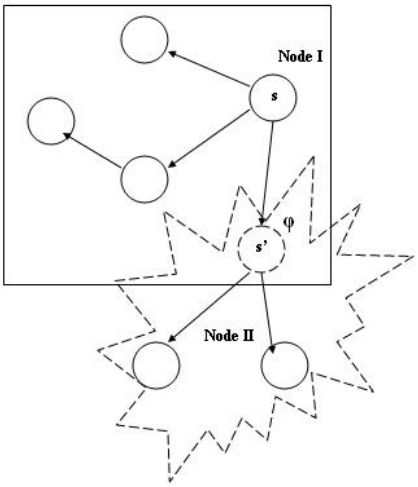

Fig .4

Theorem .2 Given $M = (S, L, R, I)$ a fragment of Kripke structure , a CTL formula $\varphi(f)$ and $Y = H_b\varphi(f)$ the set $s \in border(M)$ which satisfy $\varphi(f)$ or it may satisfy $\varphi(f)$, the following recursive algorithm compute the set of states $H(\varphi(f)) \subseteq S$ which satisfy $\varphi(f)$ or it may satisfy $\varphi(f)$ and exclude all states which not satisfy $\varphi(f)$.

- $H(p) = \lambda Y.T^+(p) \cup U^+(p)$ such that p is an atomic proposition
- $H(\neg f) = \lambda Y.\{inT(s) \mid s \in (S - map(snd, H(f))\} \cup U^+(f)$
- $H(f \wedge g) = \lambda Y.H(f) \cap H(g)$
- $H(f \vee g) = \lambda Y.H(f) \cup H(g)$
- $H(AXf) = \lambda Y.(\{InT(snd(e)) \mid \forall e \in S^+(f).succ(e) \subseteq T^+(f) \subseteq H(f)\} \cup \{inU(snd(e)) \mid \forall e \in S^+(f).succ(e) \subseteq (T^+(f) \cup U^+(f)) \text{ and } \exists e' \in succ(e) \text{ such that } e' \in^+(f)\}) \cup \{inU(s) \mid s \in border(M)\}$
- $H(EXf) = \lambda Y.(\{InT(snd(e)) \mid \forall e \in S^+(f).\exists e' \in succ(e) \text{ and } e' \in^+(f)\} \cup \{inU(snd(e)) \mid \forall e \in S^+(f).succ(e) \subseteq U^+(f) \subseteq H(f)\}) \cup \{inU(s) \mid s \in border(M)\}$
- $H(AGf) = \lambda Y.\upsilon Z.((H_p(f) \cup Y) \cap AXZ)$
- $H(AFf) = \lambda Y.\mu Z.((H_p(f) \cup Y) \cup AXZ)$
- $H(A(fUG)) = \lambda Y.\mu Z.((H_p(g) \cup Y) \cup (H_p(f) \cap AXZ))$

Note : $H(f) = H_p(f) \cup Y$ where $H_p(f)$ is the set of state $s \notin border(M)$ which satisfy f or it may satisfy f Ω

Lemma .1

The result of the above recursive algorithm can be influenced only by the truth value of formula to be verified on border states thus we need a recomputation only when the truth value on border states change.

proof

The proof is easy, we can see that the model checking algorithm is a function depend only on Y the truth value of the formula to be checked on border states. Ω

Lemma .2

The distributed termination is reached when no change of the information on all border states.

proof

using lemma.1 we can see that if there is no change in all border states, each instance of the distributed algorithm don't need to make a new computation , a hence the distributed algorithm reach a fix point and terminate. Ω

Lemma .3

When the termination is detected and still some value has undefined truth on some sates s i.e $L(s, f) = \bot$, example in the case of cycle, this implies that :

1. In the case of $f = AG\phi$, $L(s, f) = true$
2. In the case of $f = AF\phi$, $L(s, f) = false$

3. In the case of $f = A(\phi_1 U \phi_2)$, $L(s, f) = false$

proof

The proof is easy, we can see that the transition relation is a partial order, in the example of figure 5 we have $1 \prec 2 \prec 3 \prec 1$ i.e that the truth of the formula in state 1 depend on the truth of the formula in sate 2 and so on, since we know the truth of the immediate component of our formula (in the example is just atomic proposition P) in the state present in the node, which must be true for arriving to this situation, we conclude the result about the whole Kripke. Ω

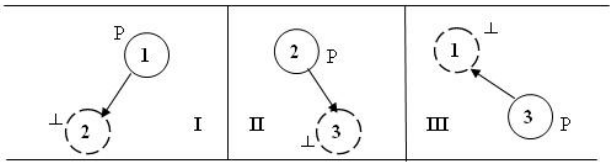

Fig .5 The Fragments of M distributed over Nodes

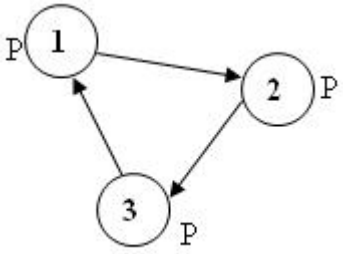

Fig .6 The Kripke structure M with cycle

Example 3.

Structure M shown in Figure 7, property to be checked on M is $AG(a \vee c)$:

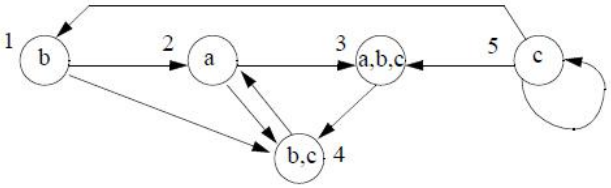

Fig .7The Kripke structure M

$S = \{1,2,3,4,5\}, AP = \{a,b,c\},$

$R = \{(1,2),(2,3),(3,5),(5,5),(5,1),(2,4),(4,2),(1,4),(3,4)\}$

, $L(1) = \{b\}, L(2) = \{a\}, L(3) = \{a,b,c\}, L(4) = \{b,c\}, L(5) = \{c\}$

The partitioning of the system on three network nodes using the following partition function h is shown in figure 8 :

$h : \{s_1, s_2, s_3, s_4, s_5\} \rightarrow \{node_1, node_2, node_3\}$

$h(1) = h(2) = 1, h(4) = 2, h(3) = h(5) = 3$

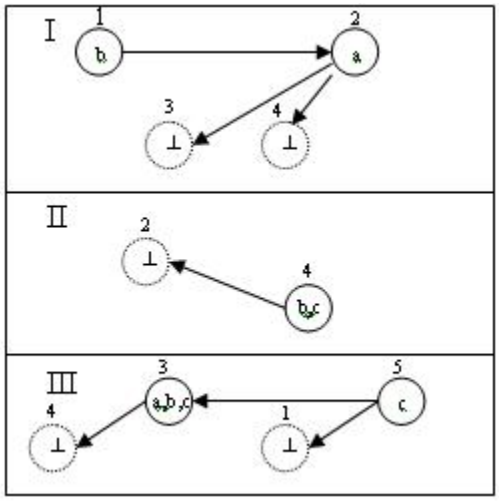

Fig .8 The Fragments of M distributed over Nodes

The application of the Algorithm on the complete M give the same result as in [2], $H(AG\, a \vee c) = \{2,3,4\}$, since all the information needed for the computation is available. We make the remark that in the case of the application of the algorithm in the complete Kripke, our Algorithm can be simplified to the Algorithm in [1], because $Y = \{\}$ and the operations $\sqcap, \sqcup$ will be $\cap, \cup$ respectively .

iteration 1:

Node I :

$H_p(a) = \{(1,2)\}$

$H_p(c) = \{\}$

$H_p(a \vee c) = \{(1,2)\}$

$Y = H_B(AG(a \vee c)) = \{(\perp,3),(\perp,4)\}$

$H(a \vee c) = Y \sqcup H_p(a \vee c) = \{(1,2),(\perp,3),(\perp,4)\}$

$H(AG(a \vee c)) = \upsilon Z.(\{(1,2),(\perp,3),(\perp,4)\} \sqcap AXZ)$

$Z_0 = \{(1,1),(1,2),(1,3),(1,4)\}$

$AXZ_0 = \{(1,1),(1,2),(1,3),(1,4)\} \sqcup \{(\perp,3),(\perp,4)\}$

$AXZ_0 = \{(1,1),(1,2),(1,3),(1,4)\}$

$Z_1 = \{(1,2),(\perp,3),(\perp,4)\} \sqcap AXZ_0$

$Z_1 = \{(1,2),(\perp,3),(\perp,4)\}$

$AXZ_1 = \{(1,1),(\perp,2)\} \sqcup \{(\perp,3),(\perp,4)\}$

$AXZ_1 = \{(1,1),(\perp,2),(\perp,3),(\perp,4)\}$

$Z_2 = \{(1,2),(\perp,3),(\perp,4)\} \sqcap AXZ_1 = \{(\perp,2),(\perp,3),(\perp,4)\}$

$Z_3 = \{(\perp,2),(\perp,3),(\perp,4)\}$

$Z_3 = Z_2$

Fix point reached, the algorithm stop the computation and wait new information, if possible, about his border states, from the theorem 2. we conclude that, $L(1, AG(a \vee c)) = false$ so the final result of computation on the iteration 1 in node 1 is :

$$\{(0,1),(\perp,2),(\perp,3),(\perp,4)\}$$

Node II :

$H_p(a \vee c) = \{(1,4)\}$

$Y = H_B(AG(a \vee c)) = \{(\perp,2)\}$

$H(a \vee c) = Y \cup H_p(a \vee c) = \{(\perp,2),(1,4)\}$

$H(AG(a \vee c)) = \upsilon Z.(\{(\perp,2),(1,4)\} \cap AXZ)$

$Z_0 = \{(1,2),(1,4)\}$

$AXZ_0 = \{(\perp,2),(\perp,4)\}$

$Z_1 = \{(\perp,2),(\perp,4)\}$

$AXZ_1 = \{(\perp,2),(\perp,4)\}$

$Z_2 = \{(\perp,2),(\perp,4)\}$

$Z_3 = Z_2$

Node III :

$H_p(a \vee c) = \{(1,3),(1,5)\}$

$Y = H_B(AG(a \vee c)) = \{(\perp,1),(\perp,4)\}$

$H(a \vee c) = Y \cup H_p(a \vee c) = \{(1,3),(1,5),(\perp,1),(\perp,4)\}$

$H(AG(a \vee c)) = \upsilon Z.(\{(1,3),(1,5),(\perp,1),(\perp,4)\} \cap AXZ)$

$Z_0 = \{(1,1),(1,3),(1,4),(1,5)\}$

$AXZ_0 = \{(1,1),(1,3),(1,4),(1,5)\}$

$Z_1 = \{(\perp,1),(1,3),(\perp,4),(\perp,5)\}$

$AXZ_1 = \{(\perp,3),(\perp,5)\} \cup \{(\perp,1),(\perp,4)\}$

$AXZ_1 = \{(\perp,1),(\perp,3),(\perp,4),(\perp,5)\}$

$Z_2 = \{(\perp,1),(\perp,3),(\perp,4),(\perp,5)\}$

$Z_3 = \{(\perp,1),(\perp,3),(\perp,4),(\perp,5)\}$

$Z_3 = Z_2$ Fix point reached .

iteration 2:

Node III : we can make recomputing only in node III since from Lemma 1. only in node 3 we have a change in the truth of border states, because the truth value in sate 1 is changed:

$H(a \vee c) = \{(1,3),(1,5),(\perp,4)\}$

$Z_0 = \{(1,1),(1,3),(1,4),(1,5)\}$

$AXZ_0 = \{(1,1),(1,3),(1,4),(1,5)\}$

$Z_1 = \{(1,3),(\perp,4),(1,5)\}$

$AXZ_1 = \{(\perp,3),(\perp,4),(\perp,1)\}$

$Z_2 = \{(\perp,3),(\perp,4)\}$

$Z_3 = Z_2 = \{(\perp,3),(\perp,4)\}$

from theorem 2. we conclude that

$L(5, AG(a \vee c)) = false$ so the final result is in node III : $\{(\perp,3),(\perp,4),(0,5)\}$

using Lemma 2. because no change will happen in border states, the computation terminate , and the distributed algorithm halt in the iteration number 2.

using Lemma 3 . we conclude that :

$L(2, AG(a \vee c)) = true$, $L(3, AG(a \vee c)) = true$, $L(4, AG(a \vee c)) = true$. The final result of the whole computation on the three node is : $\{(1,2),(1,3),(1,4)\}$

6. Conclusions and related work

We have developing a theory of reasoning on fragments of MLTS using a three value logic as a base for a parallel model checker and presenting a natural approach for distributed model checking on MLTS, to the best of our knowledge, our algorithm is the first algorithm that use fix point model checking with three value logic on maximality labeled transition system. Closest to our work is the work of [19].In fact, the main problem of the distributed verification discussed here has been treated in their work, using the notion of Assumption, which is not a natural and easy approach for treating the problem , since they present there idea using an imperative paradigm which make the proof difficult . Furthermore they don't show how to get fragments of the system to be verified, for that reason I think it is not easy to apply their result directly to the industry. Our approach has several advantages, First we have showing how to get the fragments from a standard language and with a semantic model which allow the design of systems by action refinement, second we have making a little change to the approach of verification, all this make our idea easy to apply it for industry. Another work similar to our work in the principle of using three value logic of Kleen on partial Kripke structure was introduced by [16] but our algorithm is different from their Algorithm since they use a two pass , the first one is optimistic which consider the $\perp$ as true, the second pessimistic which consider the $\perp$ as false, hence the result of the Algorithm have four results (false, false) <(true, false) < (false, true) < (true, true), for that reason we think that our approach is the best since it is easy to adapt it for distributed Model checking. another

interesting work is the work of [20] which define a multi valued model checking , which is more general than our work, this work miss an application, our work can be considered as an application with special case using three value logic.

Acknowledgments

I would like to thank all the fellows and staff at the International institute for software technology, for their indirect contributions and encouragements during the realization of this work. A special thanks to Dr Jeff W Sanders for his collaboration with Dr Djamel Eddine Saidouni and accepting me as fellow under his supervision in the UNU/IIST and training me on research. I deeply thank Dr Jeff W Sanders, whose help, advice and supervision was invaluable. I can said without his help this work can not be achieved.

References

[1] E.M. Clarke, E. A. Emerson, and A.P. Sistla. Automatic Verification of finite state concurrent systems using temporal logic Specifications. ACM transactions on Programming Languages And Systems, 8(2):244-263 (April1986).
[2] Edmund M Clarke,Orna Grumberg and al : Model check-ing MIT press ,2000 .
[3] T. Bolognesi and E. Brinksma, " Introduction to the ISO Specification Language LOTOS", volume 14, Computer Networks and ISDN Systems, 1987.
[4] ISO8807. LOTOS, a formal description technique based on the ordering of observation behaviour. ISO (November 1988).
[5] J. P. Courtiat and D. E. Saidouni. Relating maximality-based semantics to action refinement in process algebras. In "D. Hogrefe and S. Leue, Editors, IFIP TC/WG6.1, 7th Int. Cof of Formal Description Techniques(FORTE'94)", pages 293- 308. Chapman Hall (1995).
[6] Milner. "Communication and Concurrency". Prentice Hall (1989).
[7] D. E. Saidouni and A.Benamira Considiration of the cov-ering steps in The Maximality-based labled transitions systems in Proceedings of acit 2006.
[8] P. GodeFroid, " Using Partial Orders to Improve Auto-matic Verification Methods", in Proceedings of CAV'90, volume 3, pages321-340, ACM, DIMACS, 1990.
[9] R.E. Bryant, Graph-based algorithme for boolean func-tion Manipulation. IEEE Transactions on Computer Science, 37: 77-121, 1986.
[10] D.-E.Saidouni,O. Labbani Maximality-based symbolic model checking in ACS/IEEE International Conference July 2003.
[11]A. Layeb - D.E. Saidouni Quantum Differential Evolu-tion Algorithm for Variable Ordering Problem of Binary Decision Diagram in 13th International CSI Computer Conference 2008.
[12]F. Lerda and R. Sisto Distributed-Memory Model Check-ing with SPIN in Proceedings of the 5th and 6th Inter-national SPIN Workshops on Theoretical and Practical Aspects of SPIN Model Checking,1999.
[13] H. Garavel, R. Mateescu, and I. Smarandache. Parallel state space Construction for model-checking. In Proc. 8th Inter. SPIN Workshop, volume LNCS 2057, pages 217-234. Springer, 2001. Revised version available as INRIA Research Report RR-4241, Dec. 2001.
[14] Z. El Abidine Bouneb, D. E. Saidouni : Parallel state space construction for a model checking based on max-imality semantics - CISA 2009
[15] The Handbook of the History of Logic volume 8 : The Many valued and nonmonotonic turn in Logic ISBN: 978-0-444-51623-7, 2007.
[16] G.Bruns and P.Godefroid Model Checking Partial State Spaces with 3-valued Temporal logics in CAV Interna-tional Conference 1999.
[17] L. Lamport. What good is temporal logic ? Information processing, 83:657-668, 1983.
[18] ftp://ftp.rfc-editor.org/in-notes/rfc1321.txt
[19] Lubos Brim, Karen Yorav, Jitka Zidkova : Assumption-based distribution of CTL model checking, Springer-Verlag 2004
[20] Marsha Chechik, Steve M. Easterbrook, Benet Devereux: Model Checking with Multi-Valued Temporal Logics.

Zine El Abidine Bouneb. born in Algeria in 1976. He obtained his master degree from Constantine university, Algeria, in December 2003 on the field of computation and information. Zine El Abidine Bouneb is interested to the following topics : maximality semantics , formal methods , type theory , state explosion problem , game theory , models for concurrency , functional programming. Mr Bouneb Z.A currently is a lecturer at the university of Oum El Bouaghi and a PhD student in the university of Constantine under the supervision of Dr Saidouni Djamel Eddine working on symbolic verification.

Djamel Eddine Saidouni. was born in Algeria in 1968. He obtained his PhD degree from Paul Sabatier university, France, in 1996. Djamel Eddine is interested to the following topics : maximality semantics , formal methods , real time system , state explosion problem , models for concurrency , refinement. Dr Djamel Eddine is currently member of the RT-LOTOS project and the author of the true concurrency model for process algebra called Maximality Labeled Transition system. Dr Djamel Eddine has many publications in theoretical computer science and formal methods . Dr Djamel Eddine is currently an assistant professor in the department of computer science at the University of Constantine. He is also the head of the research group on formal methods.