Software Engineering Group
Department of Computer Science
Nanjing University
http://seg.nju.edu.cn

Technical Report No. NJU-SEG-2002-IC-001

Partial Order Path Technique for Checking Parallel Timed Automata

Jianhua Zhao, He Xu, Xuandong Li, Tao Zheng and Guoliang Zheng

Postprint Version. Originally Published in: Formal Techniques in Real-Time and Fault-Tolerant Systems 2002, Pages 417-432

Partial Order Path Technique for Checking Parallel Timed Automata*

Jianhua Zhao, He Xu, Xuandong Li, Tao Zheng, and Guoliang Zheng

State Key Laboratory of Novel Software Technology
Dept. of Computer Sci. and Tech. Nanjing University
Nanjing, Jiangsu, P.R.China 210093
zhaojh@nju.edu.cn

Abstract. In a parallel composition of timed automata, some transitions are independent to others. Generally the basic method generates one successors for each of the legal permutations of the transitions. These successors may be combined into one bigger symbolic state. In other words, the basic algorithm slices one big symbolic state into pieces. The number of these pieces can be up to $n!$, where n is the number of independent transitions.
In this paper, we introduce a concept, 'partial order path', to avoid treating the permutations one by one. A partial order path includes a set of transitions and a partial order on this set. Our algorithm generates one symbolic successor w.r.t. each partial order path. This big symbolic successor is just the combination of the successors w.r.t. all the global paths which are consistent to this partial order path. It is shown by some case studies that this method may result in significant space reduction.

1 Introduction

Model checking is a formal technique for validating whether a system model holds for a specific property. The basic method of model checking is exhaustive exploration of the system state space. However, the state space increases explosively when the scale of the model increases. Many techniques have been introduced to attack this problem. Partial order technique is one of the most efficient ones.

Partial order technique is first introduced for temporal model checking, and get many successful results[8], [2]. However, the progress of applying such technique into timed automata is slow. The main reason is that the clocks in different automaton increase in same rate. Different transition orders will result in different successors. A partial order technique for checking real time systems has been proposed in [3]. In that paper, the authors let the time of each component

* This work is supported by the National Natural Science Foundation of China (No.60073031 and No.69703009), the National 863 High-Tech Programme of China (No.2001AA113203), Jiangsu Province Research Project (No.BK2001033), and by International Institute for Software Technology, United Nations University (UNU/IIST).

W. Damm and E.-R. Olderog (Eds.): FTRTFT 2002, LNCS 2469, pp. 417–432, 2002.

automaton evolves independently. The authors proved that there are equivalence relations which can divide the infinite number of unsynchronized states into finite number of equivalence classes. However, the proof is not constructive.

Some other partial order techniques for checking timed petri net can be found in [9][10][11]. These techniques remove orderings in the zone on sets of independent transitions, reducing the representation size of the timed state space, and also reducing the number of generated states. They yield significant reduction in the number of symbolic states at each un-timed state.

In [5], F.Panagi also presented a partial order technique for checking real-time system models. In that paper, she studied the dependence relation between transitions, and the cases when partial order reduction can be applied.

In this paper, we propose a so-called 'partial order path' technique for real-time reachablity analysis. Given a global symbolic state (l, D), this technique calculates the successors of (l, D) w.r.t. a set of global paths, instead of one global transition, starting from the location.

2 Background

2.1 Timed Automaton Networks

In this subsection, we will give an informal description of timed automata[1] and timed automaton network. The definition of global paths and executions are different from, but essentially equivalent to, the ones in the literate.

Give a clock set C, A *clock assignment* u over C of clocks is a map from C to real R. For $d \in R$, we use $u + d$ to denote the clock assignment which maps each clock x in C to the value $u(x) + d$ and for $r \subseteq C$, $[r \mapsto 0]u$ to denote the assignment for C which maps each clock in r to the value 0 and agrees with u on $C - r$. For a subset $C'$of C, we use $u \rhd C'$ to denote the clock assignment over C' satisfying $\forall x \in C' \bullet u(x) = u'(x)$.

We use $\mathcal{B}(C)$ ranged over by $D, D_1, D_2, \ldots$ and $g, g_1, g_2, \ldots$, to stand for the set of conjunctions of atomic formula of the form $x - y \sim n$ for $x, y \in C \cup \{0\}$, $\sim \in \{\leq, <, >, \geq\}$ and n being an integer. Notice that, a formula like $x \sim n$ can be expressed as $x - 0 \sim n$. Elements of $\mathcal{B}(C)$ are called clock constraints over C. We use $u \models D$ to denote that the clock assignment $u \in R^C$ satisfies the clock constraint $D \in \mathcal{B}(C)$.

A set $\mathcal{A}$ of actions includes finite number of labels such that if a is in $\mathcal{A}$, $\overline{a}$ is also in $\mathcal{A}$. For convenience, we let $\overline{\overline{a}} = a$.

A timed automaton A is a tuple $\ll N, l^0, \mathcal{A}, C, E, I \gg$, where N is a finite set of locations, $l^0 \in N$ is the start location; $\mathcal{A}$ is a set of actions; C is a finite set of clocks; $E \subseteq N \times (\mathcal{A} \cup \{\perp\}) \times \mathcal{B}(C) \times 2^C \times N$ is a set of transitions; I assigns each location in N a location invariant in $\mathcal{B}(C)$.

A *timed automaton network* is a parallel composition of finite set of timed automata $A_1, A_2, \ldots, A_n$. These automata share a same set $\mathcal{A}$ of actions.

A concrete *state* of timed automaton A is a tuple (l, u), where $l \in N$ and u is a clock assignment over C. A state can be viewed as a snapshot of A on a certain time point when A is evolving. A *state* of a network $A = A_1 \mid \ldots \mid A_n$ is

a pair (l, u) where l, called *global location*, is a vector of control locations of each automaton and u is a clock assignment over $C = C_1 \cup \ldots \cup C_n$. A state of the network is essentially a combination of local states of the component automata.

In the rest part of this paper, for a transitions $e = (l_1, -, -, -, l_2)$, we use $\overleftarrow{e}$ to denote the source location l_1 of e and $\overrightarrow{e}$ to denote the target location l_2 of e.

A *global transition* e can be a single transition if $e = (-, \perp, -, -, -)$, or a pair of transition $e_1|e_2$, where $e_1 = (-, a, -, -, -)$ and $e_2 = (-, \overline{a}, -, -, -)$ are transitions of different automata. Given a global transition e, the projection of e on A_i, denoted e/A_i, is defined as follows.

$$e/A_i = \begin{cases} e & e \text{ is a single transition of } A_i \\ e_1 & e \text{ is } e_1|e_2 \text{ or } e_2|e_1, \text{ and } e_1 \text{ is a transition of } A_i \\ \delta & \text{else} \end{cases}$$

A *local path* of A starting from a location l is a sequence of transitions $e_1 \to e_2 \to \ldots \to e_n$ satisfying that $\overleftarrow{e_1} = l$ and $\overrightarrow{e_i} = \overleftarrow{e_{i+1}}$ $(1 \le i \le n-1)$.

A *global path* of the network starting from a global location l is a sequence of transitions $e_1 \to e_2 \to \ldots \to e_n$ such that for each component automaton A_i, the sequence $e_1/A_i \to e_2/A_i \to \ldots \to e_n/A_i$ is a local path of A_i starting from $l[i]$ if all δ are ignored.

Suppose that on a specific time point, the state of $A_i (1 \le i \le n)$ is (l_i, u_i). A_i can stay on l_i as long as its local clock value satisfies $I_i(l_i)$, which is the location invariant of l_i. It can also change to a state (l'_i, u'_i) if there is a transition $(l_i, \perp, g, r, l'_i)$ such that $u_i \models g$, $u'_i = [r \to 0]u_i$ and $u'_i \models I'_i(l'_i)$. If two timed automata, A_i and A_j, respectively stay on (l_i, u_i) and (l_j, u_j), they can change to new state (l'_i, u'_i) and (l'_j, u'_j) simultaneously if there are two transitions (l_i, a, g_i, r_i, l'_i), $(l_j, \overline{a}, g_j, r_j, l'_j)$ such that $u_i \models g_i$, $u_j \models g_j$, $u'_i = [r_i \mapsto 0]u_i$, $u'_j = [r_j \mapsto 0]u_j$, $u'_i \models I_i(l'_i)$ and $u'_j \models I_j(l'_j)$.

We can use executions to record the evolution progress of the component automata and the network.

A *local execution* of A_i starting from a state (l, u) is a time stamped local path $(e_1, t_1) \to (e_2, t_2) \to \ldots \to (e_k, t_k) \to (\delta, t_{k+1})$, where t_i is a real number. Suppose A_i is on (l, u), if we don't consider the synchronization between transitions, A_i can evolve as follows. It stays on l for t_1 time, then e_1 take place, then stays on $\overrightarrow{e_1}$ for $t_2 - t_1$ time, then e_2 takes place, ..., then e_k takes place, stays on $\overrightarrow{e_k}$ for $t_{k+1} - t_k$ time. Because the synchronization is ignored here, a local execution is possibly illegal in the global environment.

A *global execution* of the network starting from a global state (l, u) is a time stamped global path $(e_1, t_1) \to (e_2, t_2) \to \ldots \to (e_k, t_k) \to (\delta, t_{k+1})$ such that the time stamped transition sequence $(e_1/A_i, t_1) \to (e_2/A_i, t_2) \to \ldots \to (e_k/A_i, t_k) \to (\delta, t_{k+1})$ is a local execution of A_i if all the time stamped δs, except (δ, t_{k+1}), are ignored.

2.2 Symbolic States and Their Reachablity Relationship

Because the value of clocks are real numbers, the state space of a timed automaton network is infinite. However, we can use *symbolic states* of the form (l, D),

where $D \in \mathcal{B}(C)$, to express a set of concrete states. Intuitively, (l, D) represents the set of concrete states (l, u) such that $u \models D$. There is an equivalence relation to divide all the symbolic states into finite number of equivalence classes. We write $(l, u) \models (l', D)$ if $l = l'$ and $u \models D$.

We will use the following four operators on time constraints in this paper. Let C be a set of clocks, r and C' be subsets of C. For two time constraints $D \in \mathcal{B}(C)$ and $D' \in \mathcal{B}(C')$, the operators $D^{\uparrow}$, $r(D)$, $D \rhd C'$ and $D' \lhd C$ are defined as follows.

- for all $d \in R$, $u + d \models D^{\uparrow}$ iff $u \models D$.
- $u \models r(D)$ iff $\exists u' \bullet (u' \models D \wedge u = [r \mapsto 0]u')$.
- $D \rhd C' \in \mathcal{B}(C')$ and $u' \models D \rhd C'$ iff $\exists u \bullet (u \models D \wedge u' = u \rhd C')$.
- $D' \lhd C \in \mathcal{B}(C)$ and $u \models D' \lhd C$ iff $\exists u' \bullet (u' \models D' \wedge u' = u \rhd C')$.

It can be shown that $D^{\uparrow}$, $r(D)$ and $D' \lhd C$ are in $\mathcal{B}(C)$ and $D \rhd C'$ is in $\mathcal{B}(C')$.

We now define a strongest post-condition operator sp over the global state set of the three types of global evolutions.

- For global delay, $sp(\delta)(l, D) \stackrel{def}{=} (l, D^{\uparrow} \wedge I(l))$
- For a single transition $e = (l[i], \perp, g, r, l'_i)$ of A_i, $sp(e)(l, D) \stackrel{def}{=} (l', (r(g \wedge D)) \wedge I(l'))$, where $l' = l[l'_i/i]$.
- For a pair transitions $e = (l[i], a, g_i, r_i, l'_i) \in E_i$, and $e' = (l[j], \overline{a}, g_j, r_j, l'_j) \in E_j$, $sp(e|e')(l, D) \stackrel{def}{=} (l', ((r_i \cup r_j)(g_i \wedge g_j \wedge D)) \wedge I(l'))$, where $l' = l[l'_i/i][l'_j/j]$.

In this definition, $sp(\delta)(l, D)$ expresses all the set of state reachable from a state in (l, D) by only time advancement. $sp(e)(l, D)$ expresses the set of state reachable from a state in (l, D) through a single transition $e \in E_i$. $sp(e|e')(l, D)$ expresses the set of state reachable from a state in (l, D) through a pair of matched transitions.

Now we extend the operator sp to global paths. Let $p = e_1 \to e_2 \to \ldots \to e_n$ be a global path,

$$sp(p)(l, D) = \begin{cases} sp(\delta)(l, D) & \text{if } p \text{ is a empty path} \\ sp(\delta)(sp(e_n)(sp(p')(l, D))) & p' = e_1 \to e_2 \to \ldots \to e_{n-1} \end{cases}$$

Let p be a global path from l to l'. From the definition of sp, we have that, for any concrete state (l', u'), $(l', u') \models sp(p)(l, D)$ iff there is a state (l, u) such that $(l, u) \models (l, D)$ and there is a global execution from (l, u) to (l', u') corresponding to p.

For each automaton, the local strongest post-operator sp_l is similar to sp and defined as follows.

1. For time delay, $sp_l(\delta)(l, D) \stackrel{def}{=} (l, D^{\uparrow} \wedge I(l))$
2. For a transition $e = (l, -, g, r, l')$, $sp_l(e)(l, D) \stackrel{def}{=} (l', r(g \wedge D))$

This local operator does not consider the synchronizations. We can also similarly extend sp_l to local paths. For a local path p, for any concrete state (l', u'), $(l', u') \models sp_l(p)(l, D)$ iff there is a state (l, u) such that $(l, u) \models (l, D)$ and there is a local execution from (l, u) to (l', u') corresponding to p.

Definition 1. Symbolic reachablity *Given two global symbolic state* (l, D) *and* (l', D')*, we say that* (l', D') *is reachable from* (l, D) *iff there is a global path* p *such that* $sp(p)(l, D) = (l', D'')$ *for some* D'' *and* $D' \wedge D'' \neq FALSE$.

Using the operator *sp*, we can get a basic algorithm to check whether a global symbolic state (l', D') is reachable from another state (l, D). This algorithm, or its variants, is wildly used in different model checking tools[6][4][7]. The basic algorithm depicted in Fig 1 checks whether a global symbolic state (l_{tgt}, D_{tgt}) is reachable from (l_0, D_0). In the figure, for a global location l, we use $enable(l)$ to denote the set $\{(l[i], \perp, -, -)\} \cup \{(l[i], a, -, -)|(l[j], \overline{a}, -, -) \mid i \neq j\}$. This algorithm, or its variants, is wildly used in different model checking tools[6][4][7]. In this paper, we will present a method to improve this algorithm.

```
PASSED  := {}
WAITING := {(l0, D0)}
repeat    begin
          get an auxiliary state (l, D) from WAITING
          WAITING = WAITING - {(l, D)}.
          if D ⊈ D' for all (l, D') ∈PASSED then
            begin
                add (l, D) to PASSED.
                for each transition e in enabled(l) do
                begin
                    let(l1, D1) = sp(δ)(sp(e)((l, D)))
                    if l1 = l_tgt and D1 ∩ D_tgt ≠ φ
                      return YES
                    else begin
                      if (l1, D1) is not contained by a state in WAITING∪PASSED
                        WAITING:= WAITING ∪ {(l1, D1)}
                    end
                end
            end
          end
until WAITING = {}
return NO.
```

Fig. 1. The basic reachability analysis algorithm

3 Basic Idea of Partial Order Path Technique

Let's take the system depicted in Fig 2 as an example to show our basic idea of real-time partial-order-path model-checking. Suppose the system stays on a symbolic state $(< l_{11}, l_{21}, l_{31} >, x_1 \geq 0 \wedge x_2 \geq 0 \wedge x_3 \geq 0)$. It evolves in the following way: A_1 evolves to l_{12} through e_{11}, A_2 evolves to l_{22} through e_{21}, and A_3 evolves to l_{32} through e_{31}. The basic algorithm calculates the successors in one-transition step-wise. So it will generally get many successors according to different global paths: $e_{11} \rightarrow e_{21} \rightarrow e_{31}$, $e_{11} \rightarrow e_{31} \rightarrow e_{21}$ and so on. If e_{11},

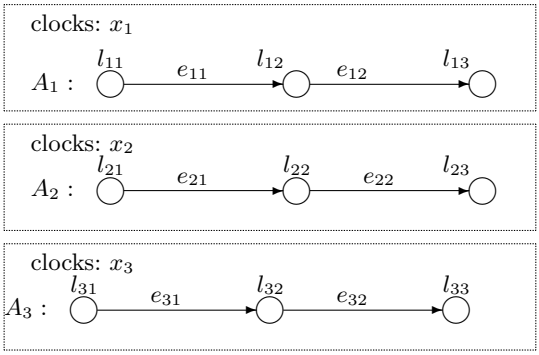

Fig. 2. Demo of partial order path model-checking

e_{21} and e_{31} respectively reset the clocks x_1, x_2 and x_3, the time constraints $x_3 - x_2 \leq 0$ and $x_2 - x_3 \leq 0$ will respectively appeared in the successors w.r.t. the first two paths. If e_{11}, e_{21}, e_{31} are not synchronizing transitions, the order of e_{11}, e_{21}, e_{31} in the paths is inessential. However, because the basic algorithm calculates the successors in a one-transition step, it generates a successor for each permutation of the transitions. So 6 successors will be generated in this cases. However, we know that, these symbolic states can be combined into one states $(< l_{12}, l_{22}, l_{32} >, x_1 \geq 0 \wedge x_2 \geq 0 \wedge x_3 \geq 0)$. Generally, if there are n independent transitions, the basic algorithm will generate $n!$ successors. In this paper, we will present a way to generate the combination of these successors as one symbolic state. So, the number of states generated by our algorithms is smaller.

3.1 Partial Order Path

However, the transitions of different automata are not always independent to each other. They interact with each other through some synchronization mechanisms. The orders of some transitions are essential. We use a partial order to express the essential orders and ignore the inessential ones.

Definition 2. Partial order path. *Given a set of global transitions $T = e_1, e_2, \ldots, e_m$ and a partial order $<$ over T. The tuple $(T, <)$ is a partial order path iff the following condition holds.*

1. *For any two transitions e_1, e_2 in T and a component automaton A_i, $e_1/A_i \neq \delta$ and $e_2/A_i \neq \delta$ implies that either $e_1 < e_2$ or $e_2 < e_1$.*
2. *For each permutation $p = e_{i_1} \rightarrow e_{i_2} \rightarrow \ldots \rightarrow e_{i_m}$ of T, p is a global path if for all k and l, $e_{i_k} < e_{i_l} \Rightarrow k < l$.*

We also say p is a global path consistent to $(T, <)$.

In the rest part of this paper, we will present a method to calculate the successor w.r.t. a partial order path, instead of a global transition.

From the definition of global paths, for a partial order path $(T, <)$ and a component automaton A_i, the local transition set $\{e/A_i \mid (e \in T) \wedge (e/A_i \neq \delta)\}$ should be able to form a local path of A_i. So a partial order path can be decomposed into a set of local paths.

Definition 3. Local path set. *Let l, l' be two global locations of a timed automaton network. Let $\mathcal{P} = \alpha_1, \alpha_2, \ldots, \alpha_n$ be a set of local paths. $\mathcal{P}$ is called a local path set from l to l' if for each i, $(1 \leq i \leq n)$, α_i is a local path of A_i from $l[i]$ to $l'[i]$.*

Example 1. $\{e_{11} \rightarrow e_{12}; e_{21} \rightarrow e_{22}; \phi\}$ is a local path set from $< l_{11}, l_{21}, l_{31} >$ to $< l_{13}, l_{23}, l_{31} >$

A local path set from l to l' describes a set of possible global paths from l to l'. The synchronizing transitions in different local paths can synchronize with each other in different ways.

Definition 4. Synchronization solution. *Let l, l' be two global locations. Let $\mathcal{P} = \alpha_1, \alpha_2, \ldots, \alpha_n$ be a local path set from l to l'. A synchronization solution $\mathcal{C}$ of $\mathcal{P}$ is a set of pairs of transitions satisfying the following conditions.*

1. *For each synchronizing transition $e = (-, a, -, -, -)$ in $\mathcal{P}$, there is one and only one pair of the form (e, e') or (e', e) in $\mathcal{C}$, where $e' = (-, \overline{a}, -, -, -)$.*
2. *Two transitions of each pair in $\mathcal{C}$ are of different component automata.*

A local path set $\mathcal{P}$ may have zero or many synchronization solutions.

For each synchronization solution $\mathcal{C}$, we can construct a global transition set T as $\{e \mid e$ is a non-synchronising transition in $\mathcal{P}\} \cup \{(e|e') \mid (e, e')$ is a pair in $\mathcal{C}\}$. We define a relation $<_{\mathcal{P},\mathcal{C}}$ over T as follows.

Definition 5. *For any two global transition e and e', $e <_{\mathcal{P},\mathcal{C}} e'$ if one of the following conditions holds.*

1. *There is an automaton A_i such that $e/A_i \neq \delta$, $e'/A_i \neq \delta$ and e/A_i takes place earlier than e'/A_i does in α_i.*
2. *There is a transition e'' in T such that $e <_{\mathcal{P},\mathcal{C}} e''$ and $e'' <_{\mathcal{P},\mathcal{C}} e'$.*

Proposition 1. *The tuple $(T, <_{\mathcal{P},\mathcal{C}})$ is a partial order path if $<_{\mathcal{P},\mathcal{C}}$ is a partial order relation.*

Proof. Let $T = \{e_1, e_2, \ldots, e_k\}$. Let $p = e_{i_1}, e_{i_2}, \ldots, e_{i_k}$ be an auxiliary permutation of T such that $e_{i_l} <_{\mathcal{P},\mathcal{C}} e_{i_m}$ implies $l < m$. From the definition of $<_{\mathcal{P},\mathcal{C}}$, we have that for any component automaton A_k, $e_{i_1}/A_k, e_{i_2}/A_k, \ldots, e_{i_k}/A_k$ is just α_k. That is, p is a global path. So $(T, <_{\mathcal{P},\mathcal{C}})$ is a partial order path.

Example 2. Let's suppose that, in the example in Fig 2, $e_{11} = (-, a, -, -, -)$, $e_{12} = (-, a, -, -, -)$, $e_{21} = (-, \overline{a}, -, -, -)$ and $e_{31} = (-, \overline{a}, -, -, -)$ be synchronizing transitions. Let e_{22} and e_{32} be local transitions. For the local path set

$\mathcal{P} = \{e_{11} \to e_{12}; e_{21} \to e_{22}; e_{31} \to e_{32}\}$, both $\mathcal{C}_1 = \{(e_{11}, e_{21}), (e_{12}, e_{31})\}$ and $\mathcal{C}_2 = \{(e_{11}, e_{31}), (e_{12}, e_{21})\}$ are synchronization solutions of $\mathcal{P}$.

The global transition set w.r.t. $\mathcal{P}$ and $\mathcal{C}_1$ is $T = \{(e_{11}|e_{21}), (e_{12}|e_{31}), e_{22}, e_{32}\}$. The relation $<_{\mathcal{P},\mathcal{C}_1}$ over T is $\{ (e_{11}|e_{21}) <_{\mathcal{P},\mathcal{C}_1} e_{22}, (e_{11}|e_{21}) <_{\mathcal{P},\mathcal{C}_1} (e_{12}|e_{31}), (e_{11}|e_{21}) <_{\mathcal{P},\mathcal{C}_1} e_{32}, (e_{12}|e_{31}) <_{\mathcal{P},\mathcal{C}_1} e_{32} \}$. We can check that $(T, <_{\mathcal{P},\mathcal{C}_1})$ is a partial order path. The global transition sequences $(e_{11}|e_{21}) \to e_{22} \to (e_{12}|e_{31}) \to e_{32}$; $(e_{11}|e_{21}) \to (e_{12}|e_{31}) \to e_{22} \to e_{32}$ and $(e_{11}|e_{21}) \to (e_{12}|e_{31}) \to e_{32} \to e_{22}$ are global paths consistent to $(T, <_{\mathcal{P},\mathcal{C}_1})$.

3.2 Local Symbolic Successors

In this section, let (l, D) be a global symbolic state of the network. Let l' be a global location of the network. Let $\mathcal{P} = \alpha_1, \alpha_2, \ldots, \alpha_n$ be a local path set from l to l'. Let $\mathcal{C}$ be a synchronization solution of $\mathcal{P}$. Let $(T, <_{\mathcal{P},\mathcal{C}})$, as described above, be a partial order path w.r.t. $\mathcal{P}$ and $\mathcal{C}$. We will explain informally how to calculate the global symbolic successor w.r.t. such a partial order path. We will first calculate the local successors w.r.t. α_i in $\mathcal{P}$, then compose the local successors into one global successor.

Let (l, u) be a concrete state. Let $Exe_i (1 \le i \le n)$ be executions from $(l[i], u \rhd C_i)$ to $(l'[i], u'_i)$ through α_i. Let u' be a clock assignment satisfying that for each i, $u' \rhd C_i = u'_i$. Then $Exe_i (1 \le i \le n)$ can be composed into a global execution from (l, u) to (l', u') through a path consistent to $(T, <_{\mathcal{P},\mathcal{C}})$ if the following conditions hold.

1. The executions $Exe_i (1 \le i \le n)$ have same time length.
2. For each transition pair (e, e') in $\mathcal{C}$, e and e' have same time stamp.
3. For any two transitions e and e' in T satisfying that $e <_{\mathcal{P},\mathcal{C}} e'$, the time stamp of e(or its component transition) must be non-greater than that of e'(or its component transition).

Notice that, for any global transition $e|e'$ in T, the time stamp of e and e' in their local executions are same, so we can construct the global execution as follows. For each transition e in T, the time stamp of e is that of e/A_i in α_i if $e/A_i \neq \delta$. From the definition of $<_{\mathcal{P},\mathcal{C}}$, the condition 3 holds if the condition 2 holds. So we get a global execution corresponding to a path consistent to $(T, <_{\mathcal{P},\mathcal{C}})$.

We must present a way to check the above conditions in a symbolic way because the algorithm operates on symbolic states. To do this, when calculating a local successor w.r.t a local path α_i, we introduce a set of auxiliary clocks for each component automata. The auxiliary clock set $\overline{C}$ includes the following clocks.

1. A clock t is introduced to record the time length of local execution. When the local execution starts, the initial value of t is 0 and no transition resets t.
2. For each clock c of the network, a shadow clock $\overline{c}$ is introduced. Each $\overline{c}$ has the same value as c at the start point, and it will never be reset. At the end of any local executions, the value of $\overline{c} - t$ is the initial value of c.

3. For each synchronizing transition e in α_i, a clock c_e is introduced to record at what time e takes place. This clock c_e will only be reset by e. So if e is in α_i, the ending value of $t - c_e$ is the time point at which e take places. (Two occurrence of one transition in α will be treated as two different ones.)

For each synchronizing transition $e = (-, -, r, -, -)$, it will also reset c_e when it takes place. So e is modified to $(-, -, r \cup \{c_e\}, -, -)$ when our algorithm calculates the local successors.

Definition 6. Extended local clock set. *Let $A = A_1 \mid A_2 \mid \ldots \mid A_n$ be a timed automaton network. The extended local clock set of A_i is $C_i \cup \overline{C}$, where C_i is the clock set of A_i and $\overline{C}$ is the auxiliary clock set as described above.*

In our algorithm, the local operator sp_l of A_i will operate on $N_i \times \mathcal{B}(C_i \cup \overline{C})$. The clocks in $\overline{C}$ is somehow 'global' because they appear in different extended local symbolic state. When we combine local successors into a global successor, it is required that the value of each auxiliary clock x in $\overline{C}$ are same in different local successors. However these clocks are never tested when we calculate the local successors.

Example 3. For the network depicted in Fig 2, the extended local clock set of A_1 is $\{x_1, x_2, t, \overline{x}_1, \overline{x}_2, \overline{x}_3, \overline{x}_4, c_{e_{11}}, c_{e_{12}}, c_{e_{21}}, c_{e_{32}}\}$.

To calculate the successor of a global symbolic state (l, D), the initial extended local symbolic state of A_i is the start state of the local executions. From the definition of auxiliary clocks, we should assign some time constraints to them as follows.

Definition 7. Shadow extended local symbolic state. *Let (l, D) be a global symbolic state. Let A_i be a component timed automaton. Let D' be a conjunction of following atomic formula.*

1. $t = 0$.
2. *For any clock constraint $x - y \sim n$ (or $x \sim n$) in D, $\overline{x} - \overline{y} \sim n$ (or $x \sim n$).*
3. *For each clock $x \in C_i$, $\overline{x} = x$.*

The symbolic state $(l[i], D')$ is called the shadow extended local symbolic state of (l, D) w.r.t. A_i.

From this definition, for each concrete state $(l[i], u')$ in $(l[i], D')$, we have that $u(t) = 0$, $\forall x \in C_i \bullet u'(x) = u'(\overline{x})$. For a clock assignment u over the clock set C of the network, we have that $u \models D$ if there is a $u' \models D'$ such that $\forall x \in C \bullet u(x) = u'(\overline{x})$.

Example 4. Let $(< l_{11}, l_{21}, l_{31} >, (x_1 - x_2 \leq 0) \wedge (x_3 = 0) \wedge (x_4 = 0))$ be a global symbolic state. Then the shadow extended local symbolic state of this state w.r.t. A_1 is $(l_{11}, \overline{x}_1 = x_1 \wedge \overline{x}_2 = x_2 \wedge t = 0 \wedge \overline{x}_1 - \overline{x}_2 \leq 0 \wedge \overline{x}_3 = 0 \wedge \overline{x}_4 = 0)$.

Notice that, the clock $\overline{x}$ will never be reset and the clock c_e for each synchronizing transitions will only be reset by e. The value t record the time length from the start point. From the definition of auxiliary clocks and sp_l, we have the following proposition.

Proposition 2. *Let $(l[i], D)$ be an extended local symbolic state of A_i. Let α be a local path from $l[i]$ to $l'[i]$. Let $(l'[i], D') = sp_l(\alpha)(l[i], D)$. Then for each concrete state $(l'[i], u'_i)$ such that $u'_i \models D'$, the following conditions hold.*

1. *The concrete state $(l'[i], u'_i)$ is reachable from a state $(l[i], u_i)$ in $(l[i], D)$ through α, and $u_i[x] = u_i[\overline{x}] = u'_i[\overline{x}] - u'_i[t]$, $u_i[t] = 0$, and*
2. *the value $u'_i[t]$ is the time length of the local execution from $(l[i], u_i)$ to $(l'[i], u'_i)$.*
3. *Of each synchronizing transition e in the local execution, the time stamp is $u'_i[t] - u'_i[c_e]$.*

From this proposition, for each concrete state in the local extended symbolic successor, we can decide the time length of the local execution, the time stamp of each synchronizing transition, and from which concrete state the execution starts. So we can compose these local successors into one global successor.

3.3 Composing the Extended Local Symbolic States

In the above subsection, we present a way to calculate local successors. Some information are incorporated into the local successors. Give a concrete state in a local successor, we can know from which concrete state this one evolves, the time length of the corresponding local execution, and when the synchronizing transitions take place in this execution.

Let (l, D) be a global symbolic state of $A = A_0 \mid A_1 \mid \ldots A_n$. Let $(l[i], D_i)$ be the shadow extended local symbolic states of (l, D) w.r.t. A_i. Let $\mathcal{P} = \alpha_1, \ldots, \alpha_n$ be a local path set from l to l'. Let $\mathcal{C}$ be a synchronization solution of $\mathcal{P}$. Let T be the global transition set w.r.t. $\mathcal{P}$ and $\mathcal{C}$. If $(T, <_{\mathcal{P},\mathcal{C}})$ is a partial order path, we can get the global successor of (l, D) w.r.t. $(T, <_{\mathcal{P},\mathcal{C}})$ by the operator defined below.

Definition 8. Global successors. *Using the above denotations, we can define a global successor operator $\overline{sp}$ as follows. Let $(l'[i], D'_i) = sp_l(\alpha_i)(l[i], D_i)$ where $1 \leq i \leq n$, and $D_{\mathcal{C}} = \wedge_{(e,e')\in\mathcal{C}}(c_e = c_{e'})$. Let $D' = D_{\mathcal{C}} \wedge (\wedge_{1\leq i\leq n}(D'_i \triangleleft (C \cup \overline{C})))$.*

$$\overline{sp}(T, <_{\mathcal{P},\mathcal{C}})(l, D) = (l', D' \triangleright C)$$

where C is the clock set of the network, and $\overline{C}$ is the set of all auxiliary clocks.

Notice that, if we use $\overline{sp}$ to compose the local successors, for any global transition $e|e'$ in T, it requires that $c_e = c'_e$. That is, $t - c_e = t - c'_e$, so the time stamps of e and e' in their local executions are equal. In each local execution Exe_i, if e_1 takes place earlier than e_2 does, the time stamp of e_1 is not greater than that of e_2. From the definition of $<_{\mathcal{P},\mathcal{C}}$, for any two global transitions e and e' in T, $e <_{\mathcal{P},\mathcal{C}} e'$ implies the time stamp of e is not greater than that of e'. In this case, we do not explicitly require that the time stamp e should not be greater than that of e' if $e <_{\mathcal{P},\mathcal{C}} e'$.

Lemma 1. *Using the above denotations, the symbolic state* $\overline{sp}(T, <_{\mathcal{P},\mathcal{C}})(l, D)$ *is just the set of concrete states reachable from a concrete state in* (l, D) *through a global path* p*, which is consistent to the partial order path* $(T, <_{\mathcal{P},\mathcal{C}})$.

Proof. From the definition of $\rhd$, for each concrete state (l', u') in $\overline{sp}(\mathcal{P})(\mathcal{C})(l, D)$, there is a concrete state (l', u'') in (l', D') such that $u' = u'' \rhd C$. From the definition of D', y we have that for each i, $(l', u'' \rhd (C_i \cup \overline{C}))$ is in $sp_l(\alpha_i)(l[i], D_i)$. From proposition 2, we have that there is a local execution $Exe_i (1 \leq i \leq n)$ from $(l[i], v_i)$ to $(l'[i], u'' \rhd (C_i \cup \overline{C}))$. For each clock x in C_i, $v_i(x) = v_i(\overline{x}) = u''(\overline{x}) - u''(t)$. The time length of Exe_i is $u''(t)$. And each synchronizing transition e in Exe_i takes place at the time point $u''[t] - u''[c_e]$. We can construct a sequence of time-stamped transitions based on the local executions as follows.

1. The global transition set of this execution is T.
2. The time stamp of a transition e in T is just that of e, or its projection, in its local execution.
3. The take-place order is as follows, e takes place previously to e' either if the stamp of e is less than that of e', or if the time stamp of e is equal to e' and $e <_{\mathcal{P},\mathcal{C}} e'$.

From the definition of $\overline{sp}$ and $<_{\mathcal{P},\mathcal{C}}$, the transition sequence is a path consistent to $(T, <_{\mathcal{P},\mathcal{C}})$. The time constraints of timed automata are satisfied because the time stamps of local executions satisfy the time constraints. So this sequence of time-stamped transitions is a global execution from (l, u) to (l', u'), where for each clock x in C, $u(x) = v_i(x) = u''(\overline{x}) - u''(t)$. So $(l, u \rhd C)$ is a concrete state in (l, D). Thus we prove that each concrete state in $\overline{sp}(T, <_{\mathcal{P},\mathcal{C}})(l, D)$ is reachable from a concrete state in (l, D).

Now we prove another direction of this lemma. Let (l', u') be a concrete state which is reachable from a state (l, u) in (l, D) through a global path consistent to $(T, <_{\mathcal{P},\mathcal{C}})$. So we have a global execution Exe from (l, u) to (l', u'). Let $Exe_i = Exe/A_i$, $1 \leq i \leq n$, be the projections of Exe on A_i. From the definitions of sp_l and $\overline{sp}$ and the auxiliary clock set, (l', u') is in $\overline{sp}(T, <_{\mathcal{P},\mathcal{C}})(l, D)$.

4 The Algorithms

In the previous sections, we present a method to generate the successors of a symbolic state w.r.t. a partial order path. However, the number of partial order paths leaving from a state is infinite. And we can not find an equivalence relation over the set of extended local symbolic states. So in our algorithm, we limit the number of local successors by limit the length local pathes. We count only the partial order path of which the local paths have at most M transitions. In the algorithm, for an un-timed state l, we use $\mathbf{P}_M(l)$ to denote the local path set, of which all local paths has no more than M transitions.

$$\mathbf{P}_M(l) = \{\mathcal{P} \mid (\text{the length of each local path in } \mathcal{P} \text{ is not greater than } M)\}.$$

We use $\mathcal{T}_M(l)$ to denote the set of partial order paths of which the length of local paths are no more than M.

$$\mathcal{T}_M(l) = \{(\mathcal{P}, <_{\mathcal{P},\mathcal{C}}) \mid \mathcal{P} \in \mathbf{P}_M(l) \wedge (<_{\mathcal{P},\mathcal{C}} \text{ is a partial order})\}$$

The improved algorithm is depicted in Fig 3. The improved one is same as the basic one except the part used to calculate successors.

```
Passed  := {}
Waiting := {(l0, D0)}
repeat  begin
        get an auxiliary state (l, D) from WAITING
        WAITING = WAITING - (l, D)
        if D ⊄ D' for all (l, D') ∈ PASSED then
          begin
              add (l, D) to PASSED.
              For each p in T_M(l) do
              begin
                  let (l1, D1) = sp(p)(l, D)
                  if l1 = l_tgt and D1 ∩ D_tgt ≠ φ
                    return YES
                  else begin
                    if (l1, D1) is not contained by a state in WAITING∪PASSED
                      WAITING:= WAITING ∪ {(l1, D1)}
                  end
              end
          end
        end
until WAITING = {}
return NO.
```

Fig. 3. The improved algorithm

5 Checking Systems Using Shared Variables

We have presented an algorithm to checking parallel timed automata using synchronizing labels. The technique presented in this paper can also be applied to parallel systems using shared variables.

Notice that, we can record the time point at which a transition takes place. So for each transition e which testing or setting shared variables, we can introduce an auxiliary clock c_e to record when e takes place. When we try to combine the local successors, we can give such a partial order over the local transitions that each transition sequence which is consistent to the partial order is a global path. For any two transitions e and e' such that e must take place later that e', we can add the constraint $c_e \leq c_{e'}$ in to the combined extended global successor.

6 Performance Analysis and Case Studies

Let (l, D) be a global symbolic state, and p_1, p_2 be two global paths from l to l'. In the basic algorithm, two successors(if exist) $sp(p_1)(l, D)$ and $sp(p_2)(l, D)$ will be generated. However, in the improved algorithm, these two successors will be included by a generated symbolic state if p_1 and p_2 are consistent to same partial order path. So the improved algorithm will generate less symbolic states when it checks a timed automaton network. The technique used in this algorithm improve the memory efficiency by generating bigger symbolic states.

We enumerate the partial order paths starting from l by enumerating the local path sets from l and the corresponding synchronization solutions. However, we don't have to calculate the local successors for each partial order path. Notice that a local path may be of many partial order paths, we can first calculate the local successors for local paths. The global successor $\overline{sp}(p)(l, D)$ is generated by just combining the local successors, which are calculated previously.

A handicap of this technique is that some auxiliary clocks are introduced to record extra information. More clocks means that more memory space are need for each symbolic state. More CPU time is also required for operations on these symbolic states. However, these auxiliary clocks are only used when the algorithm calculates $\overline{sp}(p)(l, D)$, where $p \in \mathcal{T}_M(l)$. The number of local symbolic successors are decide only by the number of the leaving local paths. Additional clocks will not increase the number of local successors. As soon as all successors of (l, D) w.r.t. the leaving partial order paths are generated, the local successors can be removed. The symbolic states in WAITING and PASSED are over the original clock set C. So the extra space requirement because of the auxiliary clock set is limited.

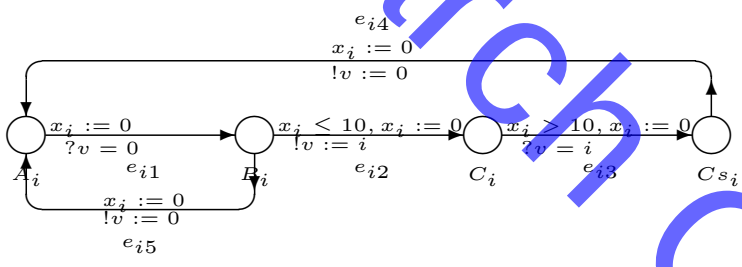

Fig. 4. The ith process of Fischer's exclusive protocol

We have incorporated this technique into our experimental tool. We applied this technique to several examples. The following tables are the performance data of our tool when it is used to check CSMA protocol and Fischer's protocol.

Fischer's protocol ensures mutex access to criteral regions using one shared variable and local clocks. Our tool checks the Fischer's protocol(as depicted in Fig 4) of 10 processes by exact and exhaustive exploration, using about 17 hours

Number of process	2	3	4	5	6	7	8	9	10
States generated	22	104	494	2392	11694	57220	278782	1291337	6347930
TIME	< 1 S.	< 1 S.	< 1 S.	3 S.	15 S.	157 S.	26.6 M.	2.2 H.	17.1 H.

Fig. 5. Space performance when checking Fischer's Protocol

Number of process	2	3	4	5	6	7	8
States generated(POP)	15	84	527	2877	13561	56836	218007
Time(POP)	< 1 S	< 1 S	3 S	21 S	5.0 M	31.6 M	3.7 H
States generated(w/o POP)	15	93	675	4794	34507	N/A	N/A

Fig. 6. Space performance when checking CSMA Protocol

in a Pentium4/1G memory computer. The total memory used by the algorithm is about 800M. As reported in the web page 'http://www.docs.uu.se/rtmv/ uppaal/benchmarks', the tool UPPAAL check's this protocol of 7 processes by exact exhaustive exploration. It can check this protocol of 11 processes using 'convex hull' approximation technique.

The CSMA/CD protocol is a protocol used in a broadcast network with a multi-access bus. This protocol is first checked by Kronos[12]. In Fig 6, we can see that when checking CSMA protocol, our technique also result in noticeable space reduction.

The above case studies show that this technique is effective in some cases. However, the technique may result in little space reduction in some other cases. The perform becomes worse because this technique needs extra space and CPU time for calculating partial order paths.

7 Conclusions and Future Works

In this paper, we proposed a so-called 'partial order path' technique for real-time reachablity analysis. Given a global symbolic state (l, D), this technique calculates the successors of (l, D) w.r.t. a set of global paths starting from the location. This set of global paths is expressed as a partial order path. Each successor can be expressed by one symbolic state. So the improved algorithm can have a better space efficiency.

This technique first suppose that each component timed automaton evolves independently to each others. The local successors are calculated without considering synchronization between different automata. After the local successors are calculated, this technique can combine these local successors into global successors w.r.t. the partial order over the transitions. However, combining these local successors needs some additional information. Some auxiliary clocks are introduced to record these information.

In [3], the authors let the time of each component automaton evolves independently. However, they did not give a constructive equivalence relation over the set of un-synchronized states. Our technique also calculate the local successors

independently. But these successors are combined into global states immediately. So we avoid checking the equivalence between two un-synchronized states.

This technique has been incorporated into an experimental tool. By applying these tool to several cases, we show that this technique may result in an improvement in time- and space- efficiency.

The primitive algorithm generates a reachability graph of the model being checked. The nodes of this graph is the generated symbolic states. Each edge is labeled by a global transitions. There is a edge labeled e from s to s' if and only if the successor of s w.r.t. e is included by s'. Our technique also generates a similar graph except that the edges are labeled by partial order paths. Notice that, a transition can be viewed as a special partial order path. Most properties which can be verified based on the basic graph can also be verified based on the graph generated by our algorithm.

Acknowledgement

The timed automaton model of Fischer's protocol and CSMA protocol used in this paper are downloaded from homepage of UPPAAL(http://www.docs.uu.se/rtmv/uppaal).

References

1. R. Alur and D. Dill. Automata for Modelling Real-Time Systems. In *Proc. of ICALP'90*, LNCS 443, 1990.
2. Rajeev Alur, Robert K. Brayton, Thomas A. Henzinger, Shaz Qadeer, and Sriram K. Rajamani. Partial-order reduction in symbolic state-space exploration. In *Proceedings of the Ninth International Conference on Computer-aided Verification (CAV 1997), LNCS 1254*, pages 340–351. Springer-Verlag, 1997.
3. Johan Bengtsson, Bengt Jonsson, Johan Lilius, and Wang Yi. Partial Order Reductions for Timed Systems. In *Proc. of the 9th International Conference on Concurrency Theory*, September 1998.
4. C.Daws, A.Olivero, S.Tripakis, and S.Yovine. The tool kronos. In *DIMACS Workshop on Verification and Control of Hybrid Systems*, LNCS 1066. Springer-Verlag, October 1995.
5. F.Pagani. Partial orders and verification of real-time systems. In B. Jonsson and J. Parrow, editors, *Proc. of Formal Techniques in Real-Time and Fault-Tolerant Systems*, LNCS 1135, pages 327–346. Springer-Verlag, 1996.
6. Kim G Larsen and Paul Pettersson Wang Yi. UPPAAL: Status & Developments. In Orna Grumberg, editor, *Proceedings of the 9th International Conference on Computer-Aided Verification. Haifa, Israel,*, LNCS 1254, pages 456–459. Springer-Verlag, June 1997.
7. T.A.Henzinger and P.-H. Ho. Hytech: The cornell hybrid technology tool. In *Proc. of Workshop on Tools and Algorithms for the Construction and Analysis of Systems*, 1995. BRICS report series NS-95-2.
8. Hans van der Schoot. Partial-order verification in spin can be more efficient, http://citeseer.nj.nec.com/vanderschoot97partialorder.html.

9. T.G.Rokicki. Representing and Modeling Circuits, 1993. PhD thesis, Standford University.
10. C.J. Myers, T.G.Rokicki, and T.H.Y.Meng. POSEET timing and its application to the synthesis and verification of gate-level timed circuits. *IEEE Transactions on Computer-Aided Design of Integrated Circuits* 18(6):769-786, June 1999.
11. W. Belluomini and C. J. Myers. Timed state space exploration using POSETs. *IEEE Transactions on Computer-Aided Design of Integrated Circuits*, 19(5), May 2000.
12. Sergio Yovine. Kronos: A verification Tool for Real-Time Systems. *Springer International Journal of Software Toolls for Technology Transfer* 1(1/2), Oct 1997.