Partial Order Reduction for Verification of Real-Time Components

John Håkansson[1] and Paul Pettersson[1,2]

[1] Dept. of Information Technology, Uppsala University, P.O. Box 337, SE-751 05 Uppsala, Sweden
{johnh,paupet}@it.uu.se
[2] Dept. of Computer Science and Electronics, Mälardalen University, P.O. Box 883, SE-721 23, Västerås, Sweden
Paul.Pettersson@mdh.se

Abstract. We describe a partial order reduction technique for a real-time component model. Components are described as timed automata with data ports, which can be composed in static structures of unidirectional control and data flow. Compositions can be encapsulated as components and used in other compositions to form hierarchical models. The proposed partial order reduction technique uses a local time semantics for timed automata, in which time may progress independently in parallel automata which are resynchronized when needed. To increase the number of independent transitions and to reduce the problem of re-synchronizing parallel automata we propose, and show how, to use information derived from the composition structure of an analyzed model. Based on these ideas, we present a reachability analysis algorithm that uses an ample set construction to select which symbolic transitions to explore. The algorithm has been implemented as a prototype extension of the real-time model-checker Uppaal. We report from experiments with the tool that indicate that the technique can achieve substantial reduction in the time and memory needed to analyze a real-time system described in the studied component model.

1 Introduction

Component-based development has been successfully used for desktop and e-business applications, and it is currently being introduced in many embedded systems. The resource constrained nature of these systems has motivated the development of specific component models [1,14,16,21] and formal frameworks, e.g. [8,9,11].

In general, a component based system is a composition of components, where a component is an open system that accepts input from its environment and produces output. The internal behaviour of a component can be described by a composition, thereby forming a hierarchy of compositions. Components interact with their environment through ports, according to interfaces defined for the ports. Figure 1 shows three components A, B and C. The two components A

J.-F. Raskin and P.S. Thiagarajan (Eds.): FORMATS 2007, LNCS 4763, pp. 211–226, 2007.

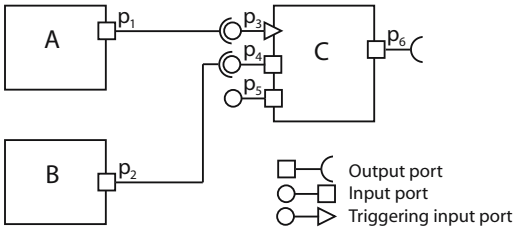

Fig. 1. An example of composition where components A, B and C are composed by connecting port p_1 to p_3, and p_2 to p_4

and B each have an output port, while component C has three input ports and an output port. Components can be composed into more complex functional units with well defined interfaces. A *horizontal composition* is a set of components with their ports connected, as in Fig. 1. The connections define how components can interact within a composition. A *vertical composition* is a component with its internal behaviour defined by a horizontal composition.

We use a model for components and composition based on the SAVE[1] component model [1,5], and designed for vehicle applications with analysability and safety in mind. The component model is similar to IEC 1131 [16] and Rubus [16]. In our model a component is either idle or executing, and data is transferred from a component when its execution has finished. Some input ports are called trigger ports, and are used to trigger the transition of a component's state from idle to executing. Control flow is specified by means of trigger ports: when one component becomes idle, it can trigger other components so that they become executing. As timeliness is an important property for many embedded systems, we model the execution of components as timed automata [2].

Model checking is an well-established and popular approach for analysis of models, although it is inherently complex and suffers from the so-called state-space explosion problem [13]. Partial order reduction [10,20,18,7] has been suggested as a technique to reduce the state-space explosion caused by parallelism. The idea is to explore representative traces — a property preserving subset of the full model based on independence of transitions. In this paper we present a partial order reduction technique for real-time systems, which is guided by the structure the component based system being analyzed. As in [3,17] we use local time semantics to increase independence. For timed automata the implicit synchronization of global time restricts independence of actions. For our component model we note that the separation of communication from internal computations makes internal transitions independent of actions in other components. We also note that we have extensive information on how components communicate, which is useful for our ample set construction. To increase independence further we relax the synchronization, so that we abstract from the exact time of

[1] SAVE is a project supported by Swedish Foundation for Strategic Research. See `http://www.mrtc.mdh.se/SAVE/` for more information.

non-triggering write operations while preserving the order of writes. We present an algorithm for partial order reduction that takes advantage of these ideas, and provide some experimental results from a prototype implementation. The experiments indicate that that our component model is well suited for partial order reduction, and they show good additional reductions when the further relaxed synchronization is used.

Related work includes partial order reduction techniques for timed systems, in particular the local time semantics for timed automata first introduced by Bengtsson et. al., [3]. They also apply the partial order reduction in reachability analysis of timed automata. This work is extended to Timed LTL model-checking by Minea in [17]. We adopt the local time semantics to develop a reachability analysis algorithm for a component model and study how the particular semantics and the static structure of the model can be used to improve previous results. A more recent approach to symbolic model checking of timed automata based on partial order semantics is presented by Lugiez et. al., in [15]. It relies on constraints over event occurrences, instead of clock constraints. In [19], Salah et. al., show that the union of zones reached by interleavings of the same set of transitions is convex. Concurrent semantics for networks of timed automata are investigated in [6,4], by a symbolic unfolding into petri nets with read arcs (to support urgency and invariants).

The rest of this paper is organized as follows: the component model is described in Section 2. In Section 3 we describes our approach to partial order reduction, and in Section 4 we give an algorithm for checking reachability and presents results from an experiment. Section 5 concludes the paper.

2 The Component Model

We introduce timed behaviours to model the execution of components as timed automata, and go on to define syntax and semantics for our component model.

Example 1 (Running Example). Figure 1 shows a horizontal composition of components A, B and C. Assume A is a timer, C a controller, and B a component generating setpoint for the controller. The timer A is connected to the input trigger port p_3 to periodically activate C. The port p_5 is used to read sensor input, which is compared to the setpoint when the controller computes its output to the actuator, port p_6.

2.1 Timed Behaviour

We define a timed behaviour as a timed automaton, extended with data variables and a final location such that no edges are leading out from this location. For a timed behaviour we have two sets of variables, the set V_{C} of clock variables, and V_{D} of data variables. The domain of variables in V_{C} is the non-negative real numbers $\mathbb{R}_{\geq 0}$, and for variables in V_{D} the domain is a bounded set of integers `INT`. We denote by $\mathcal{P}(V_{\mathrm{C}})$ the power-set of V_{C}, i.e. the set of all subsets of V_{C}.

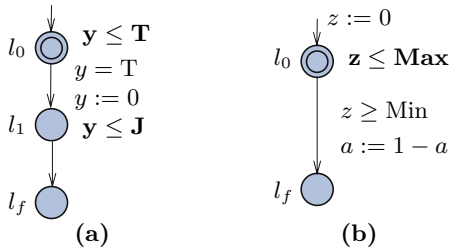

Fig. 2. Timed behaviours: **(a)** a timer with period T and jitter J, **(b)** a computation updating data variable a after between Min and Max time units

A term t is generated by the grammar $t ::= m \mid y \mid t_1 \otimes t_2$, where m is a natural number, $y \in V_\mathrm{D}$ is a data variable, $\otimes \in \{+, -, \times, /\}$, and t_1, t_2 are terms. Let U be the set of *variable updates*, each in the form $y := t$ for a data variable $y \in V_\mathrm{D}$ and a term t. An atomic clock constraint is of the form $y \sim m$, for $y \in V_\mathrm{C}$, m a natural number, and $\sim \in \{<, \leq, =, \geq, >\}$. Similarly, an atomic data constraint is of the form $t_1 \sim t_2$, with terms t_1 and t_2. We denote by $\mathrm{conj}(V_\mathrm{D}, V_\mathrm{C})$ the set of conjunctions of atomic constraints. For $g \in \mathrm{conj}(V_\mathrm{D}, V_\mathrm{C})$ we have $g_\mathrm{D} \in \mathrm{conj}(V_\mathrm{D})$ the atomic data constraints of g, and $g_\mathrm{C} \in \mathrm{conj}(V_\mathrm{C})$ the atomic clock constraints of g.

A *timed behaviour* is a timed automaton $\mathcal{B} = \langle N, l_0, l_f, V_\mathrm{D}, V_\mathrm{C}, r_0, r_f, E, I \rangle$, where N is a set of locations, l_0 is the initial location, l_f is the final location, V_D and V_C are sets of variables, $r_0 \subseteq V_\mathrm{C}$ and $r_f \subseteq V_\mathrm{C}$ are sets of clocks (initial and final resets), E is a set of edges so that $E \subseteq N \times \mathrm{conj}(V_\mathrm{D}, V_\mathrm{C}) \times U \times \mathcal{P}(V_\mathrm{C}) \times N$, and I maps each location l in $N \setminus \{l_f\}$ to its invariant $I(l)$, a conjunction of upper bounds on clocks ($y \leq m$ or $y < m$). We write $l \xrightarrow{g,e,r} l'$ iff $\langle l, g, e, r, l' \rangle \in E$ to denote an edge from location l to l' with a guard g, variable update e, and reset clocks $r \subseteq V_\mathrm{C}$.

2.2 Component

A component in our setting is defined by its interface and a timed behaviour. The interface of a component consists of data ports and trigger ports, where connected data ports define data flow between components, and connected trigger ports define control flow. The ports are either input or output. An input data port has an associated data variable holding the current data item for this port.

A component is initially *idle*, and it remains in this state until all input trigger ports have been activated, at which point it is *triggered* and switches to the executing state. The internal computation of a component starts with a *read* phase, where all the input data ports are stored internally. The internal copies of input data are used together with internal state variables during the *execute* phase, where the internal behaviour of the component is executed. When the execute phase is over the *write* phase writes output to the output data ports. Finally, the input trigger ports are reset and all outgoing trigger ports are activated, after which the component returns to the idle state.

Each component C is a tuple $\langle P_{\text{in}}, P_{\text{out}}, P_{\text{trig}}, \mathcal{B}, \pi \rangle$, where P_{in} is a set of input ports, P_{out} is a set of output ports, $P_{\text{trig}} \subseteq P_{\text{in}}$ are the trigger input ports, $\mathcal{B}$ is the timed behaviour, and $\pi : P \mapsto V_{\text{D}}$ is a mapping from ports to variables. We denote by P the set of ports $P_{\text{in}} \cup P_{\text{out}}$ of a component.

Example 2. Component A of Example 1 has no input ports so it is spontaneously triggered. Figure 2 (a) shows the timed behaviour $\mathcal{B}^{\text{A}}$ of A, with period T = 10 and jitter J = 1. When A is triggered the read phase leads to the initial location l_0. The automaton uses a clock y to ensure that l_f is reached every T time units, with a non-deterministic offset J. Reaching the final location l_f starts the write phase, after which A is spontaneously triggered again. Figure 2 (b) shows the timed behaviour $\mathcal{B}^{\text{B}}$ of component B, with response time between Min = 5 and Max = 20. The clock z is used to ensure that l_f is reached after between Min and Max time units. The setpoint value a is updated to $1 - a$. The port mapping π^{B} for B is such that $\pi^{\text{B}}(p_2) = a$, meaning that a is copied to p_2 in the write phase.

Semantics. In order to define a state of a component we first introduce clock and data valuations. For a set of clocks V_{C} a clock valuation is a map $u : V_{\text{C}} \mapsto \mathbb{R}_{\geq 0}$. Similarly, for a set of data variables V_{D} and ports P a data valuation is a map $v : (V_{\text{D}} \cup P) \mapsto \texttt{INT}$. Operations on valuations are:

$$
\begin{array}{rll}
u' = [r := 0]u & \text{iff} & u'(y) = 0 \text{ for clocks } y \in r, \text{ and} \\
 & & u'(y') = u(y') \text{ for } y' \notin r. \\
v' = [y := t]v & \text{iff} & v'(y) = v(t) \text{ for } y, \text{ and} \\
 & & v'(y') = v(y') \text{ for } y' \neq y. \\
u' = u \oplus \delta & \text{iff} & \delta \in \mathbb{R}_{\geq 0} \text{ and } u'(y) = u(y) + \delta \text{ for any clock } y.
\end{array}
$$

We introduce the *idle* location $l_\perp \notin N$, and denote by $N_\perp$ the set $N \cup \{l_\perp\}$. A *state* of a component is a tuple $\langle l, v, u \rangle$, where l is a location in $N_\perp$, v is a data valuation, and u is a clock valuation. We introduce values active and inactive for trigger ports, and define a component as *triggered* for a data valuation v, triggered(v), iff for each $p \in P_{\text{trig}}$ we have $v(p) = \text{active}$. A function input(v) is used to copy values from input ports to corresponding internal variables, similarly output(v) copies internal variables to output ports, and idle(v) inactivates trigger inputs:

$$
\begin{array}{ll}
\text{input}(v) & = [y := p \,|\, p \in P_{\text{in}},\ y = \pi(p)]v \\
\text{output}(v) & = [p := y \,|\, p \in P_{\text{out}},\ y = \pi(p)]v \\
\text{idle}(v) & = [p := \text{inactive} \,|\, p \in P_{\text{trig}}]v
\end{array}
$$

The transition rules for a component C are:

- *delay transition*: $\langle l, v, u \rangle \xrightarrow{\delta} \langle l, v, u \oplus \delta \rangle$ if $\delta \in \mathbb{R}_{\geq 0}$, $u \oplus \delta \models I(l)$, $l \neq l_f$, and if $l = l_\perp$ then $\neg\text{triggered}(v)$.
- *internal transition*: $\langle l, v, u \rangle \xrightarrow{\tau} \langle l', v', u' \rangle$ along an edge $l \xrightarrow{g,e,r} l'$ with e in the form $y := t$ if $v \models g_{\text{D}}, u \models g_{\text{C}}$, $u' \models I(l')$, $v' = [y := t]v$, and $u' = [r := 0]u$.

- *read transition*: $\langle l_\perp, v, u\rangle \xrightarrow{r} \langle l_0, \text{input}(v), [r_0 := 0]u\rangle$ if triggered(v).
- *write transition*: $\langle l_f, v, u\rangle \xrightarrow{w} \langle l_\perp, \text{idle}(\text{output}(v)), [r_f := 0]u\rangle$.

We use the following restriction on the internal behaviour of components to avoid spurious local time traces:

Definition 1 (Time Divergence). *We require for a timed behaviour that time diverges, i.e. that there is no time-stop due to invariants, and within any finite time bound only a finite number of transitions can be taken (non-zenoness).*

2.3 Composition

A composition is a set of interconnected components, also known as a horizontal composition. We define a composition as a tuple $\langle P^{\text{x}}_{\text{in}}, P^{\text{x}}_{\text{out}}, \mathcal{C}, \mathcal{X}\rangle$, where P^{x}_{in} and $P^{\text{x}}_{\text{out}}$ are external ports connecting the composition to its environment, $\mathcal{C}$ is a set of components and $\mathcal{X}$ is a set of connections. A connection $x = \langle p, p'\rangle$ connects port $p \in (P^i_{\text{out}} \cup P^{\text{x}}_{\text{in}})$ to port $p' \in (P^j_{\text{in}} \cup P^{\text{x}}_{\text{out}})$ for C^i and C^j in $\mathcal{C}$. We do not allow conflicting connections, i.e. connecting output ports of the same component with the same port.

Example 3. The composition of Fig. 1 has an external input port p_5 for sensor input, an external output port p_6 for actuation, three components A, B and C, and two connections $\langle p_1, p_3\rangle$ and $\langle p_2, p_4\rangle$.

Semantics. A state of a composition is a triple $\langle l, v, u\rangle$, where l is a location vector, v is a data valuation and u is a clock valuation. For a state s we denote by $s[i]$ the state $\langle l[i], v[i], u[i]\rangle$ of a component $C^i \in \mathcal{C}$. The local valuations $v[i]$ and $u[i]$ for a component C^i are such that $v[i](y) = v(y)$ for $y \in (P^i \cup V^i_{\text{D}})$, and $u[i](y) = u(y)$ for $y \in V^i_{\text{C}}$. In addition to the local valuations, the data valuation v also maps external ports P^{x} to their values. The transfer of data and triggering introduced by writing to ports Q:

$$\begin{aligned}\text{writedata}(Q, v) &= [p' := p \mid \langle p, p'\rangle \in \mathcal{X}, p \in Q, p' \in P^{\text{x}}_{\text{out}} \cup P^j_{\text{in}} \setminus P^j_{\text{trig}}]v\\ \text{writetrig}(Q, v) &= [p' := \text{active} \mid \langle p, p'\rangle \in \mathcal{X}, p \in Q, p' \in P^j_{\text{trig}}]v\end{aligned}$$

The transition rules for a composition are then:

- *delay transition*: $s \xrightarrow{\delta} s'$ if $s[i] \xrightarrow{\delta} s'[i]$ for each component $C^i \in \mathcal{C}$.
- *internal transition*: $s \xrightarrow{\tau^i} s'$ in the behaviour of C^i if $s[i] \xrightarrow{\tau} s'[i]$, and $s[j] = s'[j]$ for $j \neq i$.
- *read transition*: $s \xrightarrow{r^i} s'$ if $s[i] \xrightarrow{r} s'[i]$ and $s'[j] = s[j]$ for $j \neq i$.
- *write transition*: $s \xrightarrow{w^i} s'$ where either $C^i \in \mathcal{C}$ for internal component C^i writing to ports $Q = P^i_{\text{out}}$ or $i = Q$ for external write to ports $Q \subseteq P^{\text{x}}_{\text{in}}$ if
  - internal state of writer is updated: $s[i] \xrightarrow{w} s_1[i]$ if $C^i \in \mathcal{C}$, $s_1[j] = s[j]$ for $j \neq i$ (for external writes $s_1 = s$), and
  - data or triggering is transferred from ports Q: $s' = \langle l_1, v', u_1\rangle$ with $v' = \text{writetrig}(Q, \text{writedata}(Q, v_1))$.

2.4 Composite Component

In our component model [1,5] we introduce composite components to support hierarchical composition, by allowing the behaviour of a component to be described by a composition. As any other component, a composite component is defined by its interface (ports) and its timed behaviour. Unlike other components the behaviour of a composite component is described as an internal composition. This is sometimes referred to as vertical composition. Composite components can be constructed from compositions that are time divergent (Definition 1). We also require that internal components have at least one input trigger port, to avoid spontaneous triggering.

For a composite component C, an internal transition is either an internal, read or write transition of some internal component. The read operation of C correspond to a write to the external input ports of the internal composition. Component C can write when all internal components are idle. The port values are already updated by internal writes, so the write operation only need to inactivate input trigger ports.

3 Partial Order Reduction

The idea of partial order reduction is to explore representative sequences of independent transitions, instead of examining all possible sequences. However, the implicit synchronization of global time restricts independence for transitions of timed automata. As in [3,17] we use local time semantics to increase independence. It essentially allow us to analyse components of a composition in isolation, and then synchronize the components to a shared state whenever one writes to the others. To increase independence further than [3,17] we relax the synchronization, so that we abstract from the exact time of non-triggering write operations.

3.1 Representatives and Local Time Traces

To describe the concept of representative traces, we first need a notion of independent transitions. Two transitions are considered independent if they can be reordered within a trace without affecting the final state of the trace, or the validity of the trace. We denote by enabled(σ) the set of transitions that can immediately follow a finite trace σ, and define independent transitions as in [17]:

Definition 2. *Two transitions α_1 and α_2 are independent iff for any trace σ such that $\alpha_1, \alpha_2 \in \text{enabled}(\sigma)$:*

- Enabledness: $\alpha_2 \in \text{enabled}(\sigma\alpha_1)$ *and* $\alpha_1 \in \text{enabled}(\sigma\alpha_2)$.
- Commutativity: *Any state reachable by the trace $\sigma\alpha_1\alpha_2$ can also be reached by the trace $\sigma\alpha_2\alpha_1$.*

Independence is a sufficient condition for reordering transitions within a trace so that the same state is reached, however it is not a sufficient condition for

reordering of transitions during reachability analysis. There could for example be a transition α_3 in enabled($\sigma\alpha_1$) which is not in enabled($\sigma\alpha_2$), which we would miss if we only considered the trace $\sigma\alpha_2\alpha_1$. For analysis of a composition we need a strategy to make sure that some representative trace is explored for each possible trace of the full state graph. We examine conditions for reordering further in Sect. 4.1.

Independence can be concluded from the structure of a composition, using a set active(α) of components that participate in a transition α such that active(α^i) = $\{C^i\}$ for $\alpha \in \{\tau, \delta, r\}$ and active(w^i) = $\{C^i\} \cup \{C^j \mid \langle p, p' \rangle \in \mathcal{X}, p \in P^i_{\text{out}}, p' \in P^j_{\text{in}}\}$. We then restate a theorem found in e.g. [3,17], i.e. that two local time actions are independent if no automata participate in both actions:

Theorem 1. $\text{active}(\alpha_1) \cap \text{active}(\alpha_2) = \emptyset \;\Rightarrow\; \text{independent}(\alpha_1, \alpha_2)$

Proof. See [17]. □

We reduce the independence relation further (for internal transitions) in Sect. 3.2, where we introduce local time semantics. We say that two transitions α_1 and α_2 are dependent, and write dependent(α_1, α_2), whenever independent(α_1, α_2) cannot be concluded from the structure of a composition. The dependency relation is thus a safe approximation of transitions that are not independent.

We define a local time trace to be a representative of some timed trace. A trace σ is a representative of a trace σ' iff independent transitions of σ can be reordered to construct σ'. A timed trace (as defined in [2]) is a pair $\langle \sigma, t \rangle$ such that $\sigma(i)$ is the i*th* transition of the trace, $t(i)$ is the time of this transition, and the timepoints $t(i)$ are monotonic, i.e. $i \leq j$ implies $t(i) \leq t(j)$. Local time traces are then defined as:

Definition 3. *A* local time trace *is a trace $\langle \sigma, t \rangle$ such that dependent transitions are monotonic, i.e. for any $\sigma(i)$ and $\sigma(j)$ that are dependent we have $i \leq j$ implies $t(i) \leq t(j)$.*

3.2 Local Time Semantics

To keep track of the local time within components we introduce a reference clock $c^i \in V^i_{\text{C}}$ for each component $C^i \in \mathcal{C}$. We define a local delay transition for component C^i, where other components do not need to delay correspondingly.

We relax the synchronization of our local time semantics compared to [3] by completing the internal computation of a component before synchronizing with other components. This can be done since values written to input ports of a component are not used during its internal computation. We run internal computations ahead implicitly in the rule for a write operation w^i of C^i by requiring that any C^j such that dependent(w^i, w^j) is either idle ($l[j] = l^j_\perp$) or finished ($l[j] = l^j_f$).

Time is synchronized so that the local time in all components dependent on a write operation is ensured to be later than the local time of the writer, i.e.

$c^i \leq c^j$ for a writer C^i and any C^j such that dependent(w^i, w^j). This is to make traces of the local time semantics be *local time traces*, according to Definition 3. The set of components that are triggered by a write, given a location vector l and data valuation v, is:

$$\text{trig}(l, v) \;=\; \{C^j \mid l[j] = l^j_\perp \wedge \text{triggered}^j(v[j])\}$$

The clock synchronization constraint $\text{sync}^i(l, v)$ also preserves the exact time of triggering by requiring that $c^i = c^j$ for triggered components C^j:

$$\text{sync}^i(l, v) \quad = \quad \left(\bigwedge_{C^j \in \text{trig}(l,v)} c^i = c^j \right) \wedge \left(\bigwedge_{\text{dependent}(w^i, w^j)} c^i \leq c^j \right)$$

We use compositions to describe component based systems. We define local time transitions $s \longrightarrow_t s'$ for a composition using the corresponding rules for transitions $s \longrightarrow s'$:

- *delay transition*: $s \xrightarrow{\delta^i}_t s'$ if $s[i] \xrightarrow{\delta} s'[i]$ and $s'[j] = s[j]$ for $j \neq i$.
- *internal transition*: $s \xrightarrow{\tau^i}_t s'$ if $s \xrightarrow{\tau^i} s'$.
- *read transition*: $s \xrightarrow{r^i}_t s'$ if $s \xrightarrow{r^i} s'$.
- *write transition*: $s \xrightarrow{w^i}_t s'$ for $s = \langle l, v, u\rangle$ if $s \xrightarrow{w^i} s'$, $u \models \text{sync}^i(l, v)$, and either $l[j] = l^j_\perp$ or $l[j] = l^j_f$ for C^j such that dependent(w^i, w^j).

Lemma 1 shows that internal transitions in the local time semantics are independent of write transitions. This independence makes our model suited for partial order reduction, and enables the weak synchronization (otherwise we would need $c^i = c^j$ also for non-triggering participants, instead we use $c^i \leq c^j$ to preserve order of write operations).

Lemma 1. *For the local time semantics we have:*

- independent(α^i_1, α^j_2) *if* $i \neq j$ *and* α_1, α_2 *in* $\{\tau, \delta\}$
- independent(w^i, α^j) *if* $i \neq j$ *and* α *in* $\{\tau, \delta\}$
- independent(w^i, r^j) *if* $C^j \notin \text{active}(w^i)$
- independent(w^i, w^j) *if* $\text{active}(w^i) \cap \text{active}(w^j) = \emptyset$

Proof. by Theorem 1, and for independent(w^i, τ^j) we note that w^i is not enabled if τ^j is enabled for $C^j \in \text{active}(w^i)$, similarly for δ^j. □

Theorem 2 (Correctness of Local Time Semantics). *Assume local time states* s_0 *and* $s_f = \langle l_f, v_f, u_f\rangle$, *global time states* s'_0 *and* $s'_f = \langle l'_f, v'_f, u'_f\rangle$, *and a component* C^k. *Let* $s_0 = s'_0$ *except for reference clocks which are zero in* s_0 *but not included in* s'_0.

- *(*Soundness*) whenever* $s_0 \longrightarrow^*_t s_f$ *then* $s'_0 \longrightarrow^* s'_f$ *so that* $l_f[k] = l'_f[k]$, $v_f[k](y) = v'_f[k](y)$ *for* $y \in V^k_{\text{D}}$ *(i.e. not for* $y \in P^k$*), and* $u_f[k] = u'_f[k]$.
- *(*Completeness*) whenever* $s'_0 \longrightarrow^* s'_f$ *then* $s_0 \longrightarrow^*_t s_f$ *so that* $l_f[k] = l'_f[k]$, $v_f[k](y) = v'_f[k](y)$ *for* $y \in V^k_{\text{D}}$ *(i.e. not for* $y \in P^k$*), and* $u_f[k] = u'_f[k]$.

Proof. Soundness is shown by induction over a local time trace, and completeness by construction of the corresponding local time trace (see [12]). □

3.3 Symbolic Local Time Semantics

We define a zone Z as a set of clock valuations, and use symbolic states $\langle l, v, Z\rangle$ to represent all states $\langle l, v, u\rangle$ such that $u \in Z$. For a zone Z and a clock constraint g we define *conjunction* $Z \wedge g$ as the set of valuations $u \in Z$ such that $u \models g$, *reset* $r(Z)$ as u' such that $u' = [r := 0]u$ for $u \in Z$, and *symbolic local delay* $Z^{\uparrow i}$ as u' such that for $u \in Z$, $\delta \in \mathbb{R}_{\geq 0}$ we have $u'[i] = u[i] \oplus \delta$ and $u'[j] = u[j]$ for $i \neq j$.

The initial symbolic state is $\langle l_0, v_0, Z_0\rangle$, where $Z_0 = (\{u_0\})^{\uparrow} \wedge I(l_0)$ incorporates an initial delay in all components from a zone with a single solution u_0. To improve the presentation we incorporate the semantics of a component into the transition rules for a composition:

- *internal transition*: $\langle l, v, Z\rangle \stackrel{\tau^i}{\Longrightarrow}_t \langle [l^{i\prime}/l^i]l, v', Z'\rangle$ along an edge $l^i \stackrel{g,e,r}{\longrightarrow} l^{i\prime}$ with e in the form $y := t$ if $v \models g_{\mathrm{D}}$, $v' = [y := t]v$, and
  if $l^{i\prime} = l^i_f$ then $Z' = r(Z \wedge g_{\mathrm{C}})$, otherwise $Z' = (r(Z \wedge g_{\mathrm{C}}))^{\uparrow i} \wedge I^i(l^{i\prime})$.
- *read transition*: $\langle l, v, Z\rangle \stackrel{r^i}{\Longrightarrow}_t \langle [l^i_0/l^i_{\perp}]l, v', Z'\rangle$ if $l[i] = l^i_{\perp}$ and $\mathrm{triggered}^i(v[i])$, with $v'[i] = \mathrm{input}(v[i])$, $v'[j] = v[j]$ for $j \neq i$, and
  if $l^i_0 = l^i_f$ then $Z' = r^i_0(Z)$, otherwise $Z' = (r^i_0(Z))^{\uparrow i} \wedge I^i(l^i_0)$.
- *write transition*: $\langle l, v, Z\rangle \stackrel{w^i}{\Longrightarrow}_t \langle [l^i_{\perp}/l^i_f]l, v', Z'\rangle$ where either $C^i \in \mathcal{C}$ writing to ports $Q = P^i_{\mathrm{out}}$, or $i = Q$ for external write to $Q \subseteq P^{\mathrm{x}}_{\mathrm{in}}$, if $l[i] = l^i_f$ and:
  - $l[j] = l^j_{\perp}$ or $l[j] = l^j_f$ for C^j such that $\mathrm{dependent}(w^i, w^j)$,
  - $v_1[i] = \mathrm{idle}(\mathrm{output}(v[i]))$ if $C^i \in \mathcal{C}$, $v_1[j] = v[j]$ for $j \neq i$,
  - $v' = \mathrm{writetrig}(Q, \mathrm{writedata}(Q, v_1))$, and
  - if $\mathrm{triggered}^i(v'[i])$ then $Z' = r^i_f(Z \wedge \mathrm{sync}^i(l, v))$, otherwise $Z' = (r^i_f(Z \wedge \mathrm{sync}^i(l, v)))^{\uparrow i}$.

For global time semantics a zone can be represented as a conjunction of clock difference constraints. Constraints on two clocks are preserved by global time delay because both clocks grow equally, but for local time we need a different zone representation that is preserved by local time delay. Local time zones can be efficiently represented [3,17] as difference constraints on reference clocks c^i and timestamps t_y for the latest reset of a clock y.

Example 4. We explore a symbolic trace of the composition in Fig. 1. In the initial state $\langle [l^{\mathrm{A}}_{\perp}, l^{\mathrm{B}}_{\perp}, l^{\mathrm{C}}_{\perp}], v_0, Z_0\rangle$ both A and B can read, since they have no trigger input ports (and so all their triggers are trivially active). If A reads first (r^{A}) we get to a state $\langle [l^{\mathrm{A}}_0, l^{\mathrm{B}}_{\perp}, l^{\mathrm{C}}_{\perp}], v_0, Z_1\rangle$ with $Z_1 = Z_0^{\uparrow \mathrm{A}} \wedge (y \leq 10)$. We continue the trace with r^{B} to $\langle [l^{\mathrm{A}}_0, l^{\mathrm{B}}_0, l^{\mathrm{C}}_{\perp}], v_0, Z_2\rangle$ with $Z_2 = r_z(Z_1)^{\uparrow \mathrm{B}} \wedge (z \leq 20)$ for $r_z = \{z\}$. From this state internal transitions τ_A and τ_B are enabled. By τ^{B} we get to a state $\langle [l^{\mathrm{A}}_0, l^{\mathrm{B}}_f, l^{\mathrm{C}}_{\perp}], v_3, Z_3\rangle$ with $v_3 = [a := 1]v_0$ and $Z_3 = Z_2 \wedge (z \geq 5)$. Component B cannot write from this state, because w^{B} depends on w^{A} and A is neither idle or in its final location, so we take τ^A to the state $\langle [l^{\mathrm{A}}_1, l^B_f, l^C_{\perp}], v_3, Z_4\rangle$ with $Z_4 = r_y(Z_3 \wedge y = 10)^{\uparrow \mathrm{A}} \wedge y \leq 1$ for $r_y = \{y\}$. Another τ^{A} leads to $\langle [l^{\mathrm{A}}_f, l^{\mathrm{B}}_f, l^{\mathrm{C}}_{\perp}], v_3, Z_4\rangle$.

From this state both A and B can write, A when $10 \leq c^{\mathrm{A}} \leq 11$ and B when $5 \leq c^{\mathrm{B}} \leq 20$, but it cannot be determined if C will get data from B before being triggered by A.

Theorem 3 (Correctness of Symbolic Local Time Semantics). *Assume location vectors l_0, l_f, variable valuations v_0, v_f, clock valuations u_0, u_f, and local time zones Z_0, Z_f.*

- *(*Soundness*) whenever $\langle l_0, v_0, Z_0\rangle \Longrightarrow_t^* \langle l_f, v_f, Z_f\rangle$ then for any $u_f \in Z_f$ $\langle l_0, v_0, u_0\rangle \longrightarrow_t^* \langle l_f, v_f, u_f\rangle$.*
- *(*Completeness*) whenever $\langle l_0, v_0, u_0\rangle \longrightarrow_t^* \langle l_f, v_f, u_f\rangle$ then $\langle l_0, v_0, Z_0\rangle \Longrightarrow_t^* \langle l_f, v_f, Z_f\rangle$ so that $u_f \in Z_f$.*

Proof. Symbolic transition rules are constructed from local time semantics and definitions of zone operations. Preservation of zone representation is shown in [17]. See [12]. □

4 Reachability Analysis

We perform reachability analysis by exploring a subset of enabled transitions from each explored state, in order to reach a target location denoted $l_{\odot}^{k}$ for a component C^k. In the analysis we use the symbolic local time semantics, to get the independence introduced in our local time semantics and to get a finite state space.

4.1 The Ample Set Method

An ample set [18] is a subset of the enabled transitions that is sufficient to explore when model checking. The ample set method reduces a state graph G to a subgraph R such that correctness of model checking is preserved, i.e. checking R gives the same result as checking G. In general we need to select an ample set so that the checked property is preserved, i.e. a property holds in a representative trace σ in R iff the same property holds in all traces σ' in G represented by σ, and so that all traces in G have a representative in R.

For local reachability we note first that we only need to consider traces of G that can actually reach the target location. We also note that local reachability in a component C^k is preserved by representative traces, since the reordering of independent transitions does not affect which local states are reachable. The following describes how we can construct an ample set:

Definition 4 (Ample Set Construction). *From the static structure of a composition we compute $\mathcal{P}^k$ the enabled transitions that can give progress in C^k, i.e. transitions of C^k or of some C^j producing data or triggering to C^k (possibly via other components). From $\mathcal{P}^k$ we define the ample set as follows:*

- *The ample set is empty iff $\mathcal{P}^k$ is empty, otherwise some $\alpha_0 \in \mathcal{P}^k$ is in the ample set. This must hold for each valuation of a zone, as discussed in [17].*

- *If α is in the ample set, α' is enabled, and* dependent(α, α') *then α' must also be in the ample set.*
- *If α is in the ample set, α' is not enabled,* dependent(α, α')*, and there is a transition α'' that can lead to $\alpha' \in \mathcal{P}^k$ before α is taken, then α'' must also be in the ample set. This can be determined from the static structure of a composition.*
- *For a local cycle, at least one transition in $\mathcal{P}^k$ and outside of the cycle is in the ample set.*

Theorem 4. *For any trace σ in G reaching the target location, the subgraph R induced by the ample set construction in Definition 4 contains a representative of an extension $\sigma\rho$ in G.*

Proof. By a construction similar to that in [7], with simplifications for local reachability, and cycle closing due partly to construction of $\mathcal{P}^k$ (see [12]). □

As mentioned in [18] constructing an optimal ample set with respect to state space reduction is NP hard, so we suggest a heuristic. The construction in Definition 4 starts from a transition α_0 in $\mathcal{P}^k$, according to the first rule. Once a transition has been selected the other rules are used to find the least fixpoint, which is an ample set. To reduce the size of the ample set we select α_0 as an internal transition τ^i, if possible. Otherwise select α^i with minimal upper bound on the reference clock of component C^i. This reduces the possibility for other components to interfere with the execution of α^i. We also prefer read operations over writes when selecting α_0.

4.2 Model Checking Algorithm

An algorithm for symbolic reachability analysis based on the symbolic local time semantics and ample set construction is shown in Fig. 3. It is a standard reachability algorithm, with the exception that the *ample* transitions are explored instead of all enabled transitions. The normalisation $\max(Z)$ is used to ensure termination, as the symbolic semantics is not finite. In [3] it is shown that there is a finite partitioning of the state space, and [17] suggests a method for constructing $\max(Z)$. Efficient representations of local time zones are also discussed in [3,17].

We expect the algorithm to perform well: the partial order reduction is a subset of the symbolic local time semantics, by exploring only the ample set: a subset of enabled transitions. The local time semantics is sound (Theorem 2) with respect to the global time semantics. The partitioning of the state space induced by $\max(Z)$ is however incomparable with the normalisation of global time zones, so we cannot conclude any strict improvements. For timed automata with local time semantics [3,17] soundness is shown only for synchronized states, and in general for a local time state there might not exist a corresponding synchronized state. This means that some local time traces lead outside the global time semantics, giving a larger state graph to search. The components of our model are time divergent, can always accept input, and never require input to

PASSED := $\emptyset$
WAITING := $\{\langle l_0, v_0, Z_0\rangle\}$
repeat for some $\langle l, v, Z\rangle \in$ WAITING
    WAITING := WAITING $\setminus \{\langle l, v, Z\rangle\}$
    if $l[k] = l^k_\odot$ **then return "REACHABLE"**
    else if $\max(Z) \not\subseteq Z'$ **for all** $\langle l, v, Z'\rangle \in$ PASSED **then**
        PASSED := PASSED $\cup \langle l, v, \max(Z)\rangle$
        SUCC := $\{\langle l', v', Z'\rangle \mid \langle l, v, Z\rangle \stackrel{\alpha}{\Longrightarrow}_t \langle l', v', Z'\rangle,\ \alpha \in \text{ample}(l, v, Z)\}$
        WAITING := WAITING $\cup$ SUCC
until WAITING = $\emptyset$
return "NOT REACHABLE"

Fig. 3. An algorithm for symbolic reachability analysis, exploring a selected subset of enabled transitions from each state until component C^k reaches $l^k_\odot$

be available. Because of this components can always catch up, which is why we can show soundness for local states.

Example 5. A symbolic trace is described in Example 4. The local traces for A and B are $r^{\mathrm{A}}\tau^{\mathrm{A}}\tau^{\mathrm{A}}$ and $r^{\mathrm{B}}\tau^{\mathrm{B}}$, respectively. For global time semantics there are six possible interleavings, although probably more interesting nine states are passed. When searching for a location in C using the ample set construction in Definition 4 we select either r^{A} and r^{B} first. The selected component will reach its final location before the other components read operation is selected. Even though we have not specified which transition to select only one trace is explored, because r^{A} and r^{B} are independent. The trace explored using the ample set construction passes through six states, instead of the nine for global time semantics.

4.3 Experimental Results

We have developed a prototype implementation of our method as an extension of the UPPAAL tool[2]. Figure 4 illustrates the synthetic benchmarks we use to evaluate our implementation (similar to the benchmarks of Salah et.al. [19]). A synthetic benchmark NxM is a grid of components in N columns and M rows. Each component $C_{n,m}$ is connected to $C_{n,m+1}$ and to $C_{n+1,m+1}$. The components in the first row are triggered once, and the timed behaviour of each component is a delay by at least 4 time units. As target component C^k for our ample set construction we use $C_{N,M}$ (i.e. $C_{5,3}$ for Fig. 4). Table 1 shows the computation time and number of explored states for *global* time semantics (Section 2), local time with *strict* synchronization (using constraints $c^i = c^j$ also for w^i, w^j dependent) and *local* time semantics (Section 3). The prototype does not implement the normalisation step $\max(Z)$. Normalisation is not required for the benchmark models, and is turned off in the global time implementation. We note that the algorithms with partial order reduction performs much better than

[2] See the web site `www.uppaal.com` for more information.

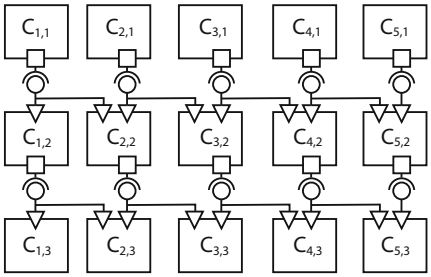

Fig. 4. The synthetic benchmark 5x3 (with N=5 columns, M=3 rows)

Table 1. Benchmark results: we use $\perp$ to denote that the experiment did not terminate in 30 minutes

Method	3x2	3x3	3x4	4x3	4x4
global	264/0.16s	499/0.22s	838/0.31s	2553/0.65s	4778/1.6s
strict	19/0.14s	105/0.19s	443/0.47s	105/0.26s	990/1.7s
local	19/0.14s	65/0.18s	275/0.43s	93/0.25s	336/1.0s
	4x5	5x4	5x5	6x5	6x6
global	8146/3.8s	26108/12s	48096/36s	481318/11m16s	$\perp$
strict	7549/21s	990/3.5s	15892/33s	15892/2m59s	$\perp$
local	2380/10s	598/2.8s	2156/20s	4316/1m08s	15101/7m36s

the algorithm without partial order reduction, and that weak synchronization performs better than the strict. We also note that *global* cover more states per second, this is because of the overheads when synchronizing components and constructing ample sets.

5 Conclusion

In this paper, we have applied and improved existing symbolic partial order reduction techniques for timed automata to develop an efficient model-checking technique for real-time components. The behavior of components are internally described as timed automata that can be hierarchical in the sense that a component can be described as a composition of components. To compose components explicit data and control flow is modeled, a property that is exploited in order to increase the independence between components, and thus to reduce the growth of the state-space caused by interleavings. We give a concrete and symbolic local time semantics for the component model, as well as a symbolic reachability analysis algorithm that uses an ample set construction to select symbolic transitions to explore. We also describe a heuristics that can be used for accelerating the analysis of local reachability properties (e.g., reachability of a location in a single component).

Our experiments suggest that our technique can achieve substantial reduction in the time and space needed to analyze a real-time system described in the studied component model. As future work we plan to further evaluate the reduction in a case study for the component model. We will also complete our implementation of the proposed reachability analysis and evaluate the achieved reduction with respect to existing techniques, such as the *event zones* of [15]. We also plan to further enrich the component model with more complex interaction structures, and support for modeling of other non-functional properties than real-time.

References

1. Åkerholm, M., Carlson, J., Fredriksson, J., Hansson, H., Håkansson, J., Möller, A., Pettersson, P., Tivoli, M.: The SAVE approach to component-based development of vehicular systems. Journal of Systems and Software (2006)
2. Alur, R., Dill, D.L.: A theory of timed automata. Theoretical Computer Science 126(2), 183–235 (1994)
3. Bengtsson, J., Jonsson, B., Lilius, J., Yi, W.: Partial order reductions for timed systems. In: Sangiorgi, D., de Simone, R. (eds.) CONCUR 1998. LNCS, vol. 1466, pp. 485–500. Springer, Heidelberg (1998)
4. Bouyer, P., Haddad, S., Reynier, P.-A.: Timed unfoldings for networks of timed automata. In: Graf, S., Zhang, W. (eds.) ATVA 2006. LNCS, vol. 4218, pp. 292–306. Springer, Heidelberg (2006)
5. Carlson, J., Håkansson, J., Pettersson, P.: SaveCCM: An analysable component model for real-time systems. In: Proceedings of the 2nd Workshop on Formal Aspects of Components Software (FACS 2005). Electronic Notes in Theoretical Computer Science, Elsevier, Amsterdam (2005)
6. Cassez, F., Chatain, T., Jard, C.: Symbolic unfoldings for networks of timed automata. In: Graf, S., Zhang, W. (eds.) ATVA 2006. LNCS, vol. 4218, pp. 307–321. Springer, Heidelberg (2006)
7. Clarke, E.M., Grumberg, O., Minea, M., Peled, D.: State space reduction using partial order techniques. International Journal on Software Tools for Technology Transfer 2(3), 279–287 (1999)
8. de Alfaro, L., Henzinger, T.A.: Interface automata. In: Proceedings of the Ninth Annual Symposium on Foundations of Software Engineering, pp. 109–120. ACM Press, New York (2001)
9. de Alfaro, L., Henzinger, T.A., Stoelinga, M.: Timed interfaces. In: Proceedings of the Second International. Workshop on Embedded Software. LNCS, Springer, Heidelberg (2002)
10. Godefroid, P., Wolper, P.: Using partial orders for the efficient verification of deadlock freedom and safety properties. In: Larsen, K.G., Skou, A. (eds.) CAV 1991. LNCS, vol. 575, pp. 332–342. Springer, Heidelberg (1992)
11. Gössler, G., Sifakis, J.: Composition for component-based modelling. Science of Computer Programming 55(1-3), 161–183 (2005)
12. Håkansson, J., Pettersson, P.: Partial order reduction for verification of real-time components. Technical report, Department of Information Technology, Uppsala University (July 2007)
13. Holzmann, G.: The Design and Validation of Computer Protocols. Prentice Hall, Englewood Cliffs (1991)

14. IEC. International standard IEC 1131: Programmable controllers (1992)
15. Lugiez, D., Niebert, P., Zennou, S.: A partial order semantics approach to the clock explosion problem of timed automata. Theoretical Computer Science 345(1), 27–59 (2005)
16. Lundbäck, K.-L., Lundbäck, J., Lindberg, M.: Development of dependable real-time applications. Arcticus Systems (December 2004)
17. Minea, M.: Partial order reduction for model checking of timed automata. In: Baeten, J.C.M., Mauw, S. (eds.) CONCUR 1999. LNCS, vol. 1664, pp. 431–446. Springer, Heidelberg (1999)
18. Peled, D.: All from one, one for all: on model checking using representatives. In: Courcoubetis, C. (ed.) CAV 1993. LNCS, vol. 697, pp. 409–423. Springer, Heidelberg (1993)
19. Salah, R.B., Bozga, M., Maler, O.: On interleaving in timed automata. In: Baier, C., Hermanns, H. (eds.) CONCUR 2006. LNCS, vol. 4137, pp. 465–476. Springer, Heidelberg (2006)
20. Valmari, A.: A stubborn attack on state explosion. In: Rozenberg, G. (ed.) Advances in Petri Nets 1990. LNCS, vol. 483, Springer, Heidelberg (1991)
21. van Ommering, R., van der Linden, F., Kramer, J., Magee, J.: The Koala component model for consumer electronics software. IEEE Computer, pp. 78–85 (2000)