Modeling and Analysis of Real-Time Systems with Mutex Components

APDCM’10

Guoqiang Li¹, Xiaojuan Cai¹, Shoji Yuen²

¹ BASICS, Shanghai Jiao Tong University
² Graduate School of Information Science, Nagoya University

19th, April. 2010
Backgrounds and Aims

- Formal models for complex real-timed systems (e.g. timed automata).

- A real-time system consists of several functionally independent components that interact with each other, e.g. processors, controllers, various chips, etc.
  
  - Synchronization is modeled by parallel composition of timed automata [RTSS’95]
  - Mutex ...

- In synthesis of a whole system, the “global” control of components is a key issue in design.

- Whether such a synthesis is decidable?
Timed Automata  [Alur & Dill TCS 94]
Parallel Composition  [Wang Yi et. al. RTSS’95]

- Actions are divided into two disjoint sets \( \Sigma = E \cup H \), for external and internal actions respectively.

- External actions \( E \) are partitioned to two disjoint sets \( E = E_o \cup E_i \), for triggering symbols, ranged over by \( a!, b!, \ldots \), and triggered symbols, ranged over by \( a?, b?, \ldots \).

As an example, consider Fig. 2. The figure shows the possible regions in each location of an automaton with two clocks \( /DC \) and \( /DD \). The largest number compared to \( /DC \) is 3, and the largest number compared to \( /DD \) is 2. In the figure, all corner points (intersections), line segments, and open areas are regions. Thus, the number of possible regions in each location of this example is 60.

A more efficient representation of the state-space for timed automata is based on the notion of zone and zone-graphs \([Dil89,HNSY92,YL93,YPD94,HNSY94]\). In a zone graph, instead of regions, zones are used to denote symbolic states. This in practice gives a coarser and thus more compact representation of the state-space. The basic operations and algorithms for zones to construct zone-graphs are described in Section 4.

As an example, a timed automaton and the corresponding zone graph (or reachability graph) is shown in Fig. 3. We note that for this automaton the zone graph has only 8 states. The region-graph for the same example has over 50 states.

![Diagram of a timed automaton](Image)
Why Need Controller Automata?

- Usually, mutex can be implemented by synchronization.
- However, in real-time system, time in an awaited component will elapse when it hangs up.
- There are three relations for two mutex components:
  - Competition e.g., Reading/Writing a shared buffer
  - Preemption and Resumption e.g., Interrupt
- Controller automata provide global controls among a group of timed automata.
Controller Automata

- Controller automata provide transitions for timed automata that represents different components.
- There are three kinds of transitions, push, pop and internal actions.
An Example: Reading/Writing with Priority

I. \( x_1 < 2 \)
   \( WTP, x_1 \geq 2 \land y_1 \leq 25, x_1 := 0 \)

II. \( x_2 < 3 \)
    \( RD_P, x_2 \geq 3 \land y_2 \leq 30, x_2 := 0 \)

Require \( W \) ?
Require \( R \) ?
Release \( W \) !
Release \( R \) !

\( \delta_{push} \) \quad \Rightarrow \quad \delta_{pop} \)
Time Lag in Timed Automata

- When a timed automaton is preempted by another one, the system will stop running current timed automaton, store the current status, and begin to run the latter timed automaton.

- A time lag adds a location and a fresh clock to wait a certain time when preempted by another timed automata.

\[
\begin{align*}
&x \leq 6, \\
&x \geq 6, y \leq 30, x := 0, \\
&x \geq 5, y \leq 25, x := 0 \\
&y > 25, y := 0 \\
&y > 30, \text{Go to } x := 0, y := 0
\end{align*}
\]
Time Lag in Timed Automata

- When a timed automaton is preempted by another one, the system will stop running current timed automaton, store the current status, and begin to run the latter timed automaton.

- A **time lag** adds a location and a fresh clock to wait a certain time when preempted by another timed automata.

\[
x \leq 6, y \leq 30, x := 0
\]

\[
x \geq 6, y \leq 25, x := 0, x_p := 0
\]
Running Controller Automata

\[ x = 0 \quad \text{if } x < 2 \]
\[ x = 0 \quad \text{if } x \geq 25 \]
\[ x = 0 \quad \text{if } 2 \leq x \leq 25 \]
\[ x = 0 \quad \text{if } 2 \leq x \leq 30 \]
\[ x = 0 \quad \text{if } x \leq 150 \]

\((S_0, 0)\)
Running Controller Automata

\[
\begin{align*}
\text{pat}, x &= 0 \\
\text{pat}, x < 2 &
\end{align*}
\]

\[
\begin{align*}
\text{trigger}_p, x \geq 2, x &= 0 \\
x > 25 &
\end{align*}
\]

\[
\begin{align*}
\text{pat}, x &= 0 \\
\text{pat}, x < 2 &
\end{align*}
\]

\[
\begin{align*}
\text{trigger}_q!, 2 \leq x \leq 25, x &= 0 \\
\text{trigger}_q!, 2 \leq x \leq 30, x &= 0 \\
x \leq 50 &
\end{align*}
\]

\[
\begin{align*}
\text{turn}?, T, \emptyset &
\end{align*}
\]

\[
\begin{align*}
\text{b}_{\text{pop}}, T, \emptyset &
\end{align*}
\]

\[
\begin{align*}
\text{c}_{\text{pop}}, \top, \emptyset &
\end{align*}
\]

\[
\begin{align*}
\text{a}_{\text{pop}}, T, \emptyset &
\end{align*}
\]

\[
\begin{align*}
\text{pat}, ?, \top, \emptyset &
\end{align*}
\]

\[
\begin{align*}
\text{turn}?, T, \emptyset &
\end{align*}
\]

\[
\begin{align*}
\text{run} \leq 150 \\
\text{run} \leq 50 &
\end{align*}
\]

\[
\begin{align*}
\delta_{\text{push}} &
\end{align*}
\]

\[
\begin{align*}
\delta_{\text{pop}} &
\end{align*}
\]

\[
(S_0, 0)
\]
Running Controller Automata

\[
\begin{align*}
&\text{pat}, x := 0 \\
&\text{pat}, x < 2 \\
&\text{trigger}_p, x \geq 2, x := 0 \\
&\text{pat}, x < 2 \\
&\text{trigger}_q, 2 \leq x \leq 25, x := 0 \\
&\text{pat}, x := 0 \\
&\text{pat}, x < 2 \\
&\text{turn}, \top, \emptyset \\
&\text{a}_{pop}, \top, \emptyset \\
&\text{pat}, \top, \emptyset \\
&\text{turn}, \top, \emptyset \\
&\text{c}_{pop}, \top, \emptyset \\
&\delta_{push} \\
&\delta_{pop} \\
&(S_0, 0)
\end{align*}
\]
Running Controller Automata

\[
\begin{align*}
pat?, \; x &= 0 \\
pat?, \; x < 2 \\
\text{trigger}_{p!}, \; x \geq 2, \; x &= 0 \\
pat?, \; x < 2 \\
\text{trigger}_{q!}, \; 2 \leq x \leq 25, \; x &= 0 \\
pat?, \; x < 2 \\
\text{trigger}_{q!}, \; 2 \leq x \leq 30, \; x &= 0 \\
\text{trigger}_{q!}, \; 2 \leq x \leq 25, \; x &= 0 \\
\text{trigger}_{q!}, \; 2 \leq x \leq 30, \; x &= 0 \\
x_{\text{run}} \leq 150 \\
x_{\text{run}} \leq 50 \\
turn?, \; \top, \; \emptyset \\
apop, \; \top, \; \emptyset \\
bpop, \; \top, \; \emptyset \\
cpop, \; \top, \; \emptyset \\
\delta_{\text{push}}, \; x_{\text{run}} \leq 150 \\
\delta_{\text{pop}}, \; x_{\text{run}} \leq 150 \\
(S_0, 0)
\end{align*}
\]
Running Controller Automata
Running Controller Automata

\[
\begin{align*}
\text{pat?, } x &= 0 \\
\text{pat?, } x < 2 \\
\text{trigger, } x \geq 2, x &= 0 \\
\text{pat?, } x < 2 \\
\text{trigger, } x \leq 2, x &= 0 \\
\text{pat?, } x > 25 \\
\text{pat?, } x < 2 \\
\text{trigger, } 2 \leq x \leq 30, x &= 0 \\
\text{pat?, } x \geq 0 \\
x_{\text{run}} \leq 150
\end{align*}
\]
Decidability Problems of Controller Automata

- Some comments...
  - controller automata are not beyond timed (pushdown) automata...
  - controller automata are stopwatch pushdown automata...

- Controller automata are less expressive than stopwatch automata
  - Fact. the frozen clocks are kept zero in CA.

- The decidability problems (e.g. reachability problem) of controller automata are in general undecidable.
  - Infinite insertion of fresh clocks and control locations.

- With a strict partial order on the state, an ordered controller automaton can be translated to a timed automaton.
Conclusion

- **Controller automata** are introduced, to perform global control on complex real-time systems.

- Analysis techniques (e.g. reachability) of controller automata are investigated.

- Future work:
  - **Theoretical approaches**: to investigate the languages category recognized by controller automata.
  - **Practical approaches**: to verify properties for complex real-time systems, e.g. liveness
  - **Implementation work**: translate an OCA to a timed automaton recognized by UPPAAL.
Thank You!

li.g@sjtu.edu.cn