
Kai Lampka, Simon Perathoner, Lothar Thiele

Artist-Design Cluster Meeting (Hardware Platforms)

Haus der Wissenschaft, Braunschweig, Germany, 26 June 2009
Performance Analysis of Embedded Real-Time Systems

Analytic Real-Time Analysis

Solution of closed form expressions
Examples: RTC, SymTA/S, MAST, ...

+ Good scalability
+ Fast analysis

− Limited to few specific measures (e.g. delays, buffer sizes)
− Systems restricted to specific models
− Overly conservative results

State-based Real-Time Analysis

Model checking of properties
Examples: Timed Automata (TA), FSM, ...

− Poor scalability
− Slow verification

+ Verification of functional and non-functional properties
+ Modeling power
+ Exact results

\[ r_i = C_i + \sum_{j \in \text{ne}(i)} \frac{r_j}{T_j} C_j \]

\[ d_{\text{max}} \leq \sup_{\lambda \geq 0} \{ \inf_{\tau \geq 0} \{ \alpha^\lambda(\lambda) \leq \tau(\lambda + \tau) \} \} \]
New Compositional Framework for Hybrid Analysis

Interfaces

Analytic

State-based

Analytic

Analytic

Analytic

Analytic
Motivation for Hybrid Approach

1. The obtained performance metrics are not destructively over-approximated
   (Tighter analysis results compared to purely analytical abstraction)

2. The problem of state space explosion is limited to the level of isolated components
   (Faster verification compared to purely state-based models)
Interfacing Real-Time Calculus and Timed Automata

\[ \alpha' = f(\alpha, \beta) \]
Contributions

• Pattern for conversion of abstract event stream models (such as PJD or arrival curves) to a network of cooperating TA

• Proof of correctness and completeness

• Pattern for derivation of abstract event stream models from a TA-based system model

• Implementation and Case Study
Related work

• Event Count Automata


• CATS Tool


• Efficient Model-Checking for Real-Time Task Networks

Real-Time Calculus (RTC)

Compositional abstraction
Timed Automata (TA)

System Declarations
channel on, off;
clock x, y;
enum phases
   \{g = 1, r, y\} s;
Interface RTC → TA

How to represent arrival curves as TA?
Principle

1. Decompose arrival curves to set of simpler curve components → Set of linear staircase functions

2. Represent each curve component as TA (Leaky Bucket pattern) → Set of simple TA

3. Synchronize all TA such to obtain same event stream model as described by arrival curve → Network of synchronized TA
Linear arrival curves

Upper arrival curve

\[ \alpha^u(\Delta) = N^u + \left\lfloor \frac{\Delta}{\delta^u} \right\rfloor \]

Max fill level: \( N^u \)

Fill rate: \( 1/\delta^u \)

Event emission allowed if fill level > 0

Automaton for linear upper arrival curve (UTA)
Linear arrival curves

Lower arrival curve

$$\alpha^l(\Delta) = \max \left\{ 0, N^l + \left\lfloor \frac{\Delta}{\delta^l} \right\rfloor \right\}$$

Max fill level: $$|N^l|$$

Fill rate: $$1/\delta^l$$

Event emission enforced if maximum fill level reached

Automaton for linear lower arrival curve (LTA)
Linear arrival curves

Combination of lower and upper arrival curves

UTA

\[ x = 0, \]
\[ b = \min(b+1, \text{BMAX}) \]

\[ x \leq \Delta r \]

if \( b = \text{BMAX} \) then
\[ x = 0, \]
\[ b = b - 1 \]

LTA

\[ x = 0, \]
\[ b = \text{BMAX} \]

\[ x \leq \Delta r \]

if \( b = 0 \) then
\[ x = 0, \]
\[ b = \max(b-1, 0) \]

\# events

\[ N^u \]

\[ N^l \]

\[ \delta^u \]

\[ \delta^l \]

\[ \Delta t \]

Synchronization
Convex and concave patterns

Composition of linear staircase functions

\[ \alpha^u = \min\{\alpha_1^u, \alpha_2^u, \alpha_3^u\} \]

\[ \alpha^l = \max\{0, \alpha_1^l, \alpha_2^l\} \]
Convex and concave patterns

- Event generation only if all UTA permit it (AND composition)
- Single LTA can enforce event generation (OR composition)
General arrival curves

How to represent non-convex/concave patterns?

Use min/max operators locally on subsets of UTA/LTA
Complexity

Run-time of verification increases exponentially with number of clocks

→ Approximate arrival curves with few staircase functions

e.g.

\[
\begin{align*}
&d = 0 \lor d \leq p - j : \quad N^u = \left\lfloor \frac{i}{p} \right\rfloor + 1; \quad N^l = \left\lfloor \frac{i}{p} \right\rfloor; \quad \delta^u = \delta^l = p \\
&d > 0 \land d > p - j : \quad N_1^u = 1; \quad \delta_1^u = d; \quad N_2^u = \left\lfloor \frac{i}{p} \right\rfloor + 1 \\
&\quad N^l = \left\lfloor \frac{i}{p} \right\rfloor; \quad \delta_2^u = \delta^l = p
\end{align*}
\]
Interface TA → RTC

How to derive output arrival curves from a TA sub-system model?
Key parameters of curve (e.g. max burst) are determined by appropriate observer TA and binary search.
**Interface TA → RTC**

- Verify compliance of system output with a number of UTA \((N_i, \delta_i)\) and LTA \((N_i, \delta_i)\) (Search strategy: Fix one parameter and modify the other by binary search)

- Combine obtained linear staircase functions by min and max operators

→ Yields convex/concave approximation of system output

\[
x = 0, \\
b = \min(b+1, \text{BMAX})
\]

\[
x = \text{Delta}
\]

\[
b > 0 \\
b = 0 \quad \&\& \quad x < \text{Delta}
\]

\[
x < \text{Delta}
\]

\[
\text{event?} \\
\text{if } (b==\text{BMAX}) \quad x = 0, \\
b--
\]

\[
\text{Verify} \\
(A[] \text{ (not violation)})
\]
Case Study

CPU1: Load-dependent frequency adaptation

- Characterize output of T1
- Determine delays and required buffer sizes
Case Study

TA model for CPU1
# Case Study

## Results of performance analysis

<table>
<thead>
<tr>
<th></th>
<th>Max delay [ms]</th>
<th>Max buffer [events]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$T_1$</td>
<td>$T_2$</td>
</tr>
<tr>
<td>RTC</td>
<td>29</td>
<td>8</td>
</tr>
<tr>
<td>TA + RTC</td>
<td>25</td>
<td>5.5</td>
</tr>
<tr>
<td>TA</td>
<td>25</td>
<td>4.6</td>
</tr>
</tbody>
</table>

**Diagram:**

- **Event source A** connected to **CPU 1**
- **Event source B** connected to **CPU 1**
- **CPU 1** with tasks $T_1$, $T_2$, $T_3$
- **CPU 2** with tasks $T_1$, $T_2$, $T_3$
- **Event sink A** connected to **CPU 2**
- **Event sink B** connected to **CPU 2**
Case Study

Delay computation for T2

\[ \Delta t \]

\[ T1 \]

\[ T2 \]

\[ T3 \]

CPU 1

Event source A

Event source B

CPU 2

Event sink A

Event sink B

\[ \alpha_{1,TA}^{nu} \]

\[ \alpha_{1,RTC}^{nu} \]

\[ \alpha_{1,TA}^{nl} \]

\[ \alpha_{1,RTC}^{nl} \]

\[ \beta_{CPU2}^{ul} \]

\[ \text{del}_{2,RTC} = 8 \]

\[ \text{del}_{2,TA + RTC} = 5.5 \]
## Case Study

### Results of performance analysis

<table>
<thead>
<tr>
<th></th>
<th>Max delay [ms]</th>
<th>Max buffer [events]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$T_1$</td>
<td>$T_2$</td>
</tr>
<tr>
<td>RTC</td>
<td>29</td>
<td>8</td>
</tr>
<tr>
<td>TA + RTC</td>
<td>25</td>
<td>5.5</td>
</tr>
<tr>
<td>TA</td>
<td>25</td>
<td>4.6</td>
</tr>
</tbody>
</table>

### Run-times

<table>
<thead>
<tr>
<th></th>
<th>RTC</th>
<th>TA + RTC</th>
<th>TA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total run-time</td>
<td>&lt; 1s</td>
<td>11min</td>
<td>1h</td>
</tr>
</tbody>
</table>
Conclusions

• Hybrid and compositional analysis method that couples analytical approach (RTC) with state-based approach (TA)

• Permits to trade off analysis accuracy against verification time

• Key principle: Represent arrival curves by min/max of linear staircase functions