Adaptive Task Automata with Earliest-Deadline-First Scheduling

Leo Hatvani, Alexandre David, Cristina Seceleanu, and Paul Pettersson
Presented at AVoCS 2014
25 September 2014
Outline

- Introduction
- Approach
- Task Automata
- Task Adaptivity
- Adaptive Task Automata
- Static vs Dynamic Priorities
- Example
- Conclusion
Introduction (1)

- Widespread use
- Safety (or mission) critical applications
- Increasing autonomy
- Rapidly changing environments

- A solution: verification?
Introduction (2)

- Robustness
  - Cost
  - Technology
- Quality of service
- Emergency procedures

- A solution: adaptivity?
Approach

- Focus on the task level
  - Hard deadlines
- Adapt to the extraordinary
- Verify everything
- Realistic (precise) system model
Task Automata

- 20+ years of timed automata research (counting from Alur & Dill in 1990's)
  - States defined as pairs:
    - (current location, values of clocks)
- Timed automata + tasks = task automata (2002)
  - Each state defined as a triple:
    - (current location, values of clocks, ready queue)
- Tasks are release upon entering certain locations in the task automaton
- Easy to model standard release pattern: periodic tasks, periodic tasks with jitter, etc.
Task Automata

- Possible to model many environments and task release patterns
- Tasks are added to a queue which is handled by a model of CPU and scheduler
A task automaton

\[ x \leq T \]

\[ x \geq T \]

\[ x = 0 \]

\[ t_1 \ (C = 3, D = 5) \]
A task automaton

Start

$t_1$ ($C = 3, D = 5$)

$\begin{align*}
x & \leq T \\
x & \geq T \\
x & = 0
\end{align*}$

(Start, x=0, [])
(Start, x=0, [t_1(3,5)])
A task automaton

Start

$x \leq T$

$\Rightarrow t_1 (C = 3, D = 5)$

(Start, x=0, [])

(Start, x=0, [t_1(3,5)])

(Start, x=3, [])

$x \geq T$

$x = 0$

Idle

0 3 T T+3 2T 2T+3
A task automaton

\[ \text{Start} \]
\[ x \leq T \]
\[ t_1 (C = 3, D = 5) \]
\[ x \geq T \]
\[ x = 0 \]

(Start, x=0, [])
(Start, x=0, [t_1(3,5)])
(Start, x=3, [])
(Start, x=T, [])
(Start, x=0, [t_1(3,5)])

Idle

P

0 3 T T+3 2T 2T+3
A task automaton

\begin{align*}
& \text{Start } x \leq T \\
& \quad t_1 (C = 3, D = 5) \\
& \text{Start } x \geq T \\
& \quad \text{Start, } x = 0, [\quad] \\
& \quad \text{Start, } x = 3, [\quad] \\
& \quad \text{Start, } x = T, [\quad] \\
& \quad \text{Start, } x = 0, [t_1(3,5)] \\
& \quad \text{Start, } x = 3, [\quad] \\
\end{align*}
A task automaton

$t_1 (C = 3, D = 5)$

(Start, $x=0$, [])
(Start, $x=0$, $[t_1(3, 5)]$)
(Start, $x=3$, [])
(Start, $x=T$, [])
(Start, $x=0$, $[t_1(3, 5)]$)
(Start, $x=3$, [])
...

Idle

P

0  3  T  T+3  2T  2T+3
Task Automata: The Key

Schedulability as Reachability
Task Adaptivity

Utilization

Normal operation
Task Adaptivity

Utilization

Normal operation

Occasional task
Task Adaptivity

Utilization

Normal operation
Task Adaptivity

Utilization

Normal operation
Task Adaptivity

Utilization

Normal operation

Occasional task

\[ t_1, t_2, t'_3, t_x \]
Task Adaptivity

Utilization

$t_1$
$t_2$
$t'_3$
$t_x$

Normal operation
Task Adaptivity

Utilization

Normal operation
Adaptive Task Automata (ATA)

- Feedback at the point of task release
- Maintains the schedulability verification
ATA: Predicates

\( x \leq 3 \) \hspace{1cm} \( x \geq 3 \)

\[
\begin{array}{c|c|c}
\text{ } & C & D \\
\hline
 t_1 & 6 & 8 \\
 t_2 & 3 & 3 \\
 t'_2 & 2 & 2 \\
\end{array}
\]

(a) (b)
ATA: Predicates

- Available predicates:
  - `sched(task1)`
  - `sched(task1, task2)`
  - `inqueue(task1)`
- Used in conjunction with guards
ATA: The Encoding

\[ r_1 = C_1 \]

\[ r_{n-1} = r_1 + C_{n-1} \]

- \( r_1, r_{n-1} \) - response times
- \( c_1, c_{n-1} \) - clocks tracking used up computation time
- \( C_1, C_{n-1} \) - worst case execution times
ATA: The Encoding

Task $\text{task}_1$ is schedulable as long as

$$r_{n-1} - c_{n-1} \leq D_{n-1} - d_{n-1}$$

$D_{n-1}$ - relative deadline

$d_{n-1}$ - deadline clock

$D_{n-1} - d_{n-1}$ - time left until the deadline
ATA: The Encoding

Finite state-space partitioning implies Reachability implies Schedulability Verification
ATA: Static vs. Dynamic Policy (1)

Priority level n
Priority level n-1
...
Priority level 2
Priority level 1

Dispatcher
ATA: Static vs. Dynamic Policy (1)

Dispatcher

Priority level n
Priority level n-1
…
Priority level 2
Priority level 1
ATA: Static vs. Dynamic Policy (1)

Dispatcher

Priority level n
Priority level n-1
...
Priority level 2
Priority level 1
Dispatcher

Priority level 1
ATA: Static vs. Dynamic Policy (2)

Dispatcher

Priority level 1
ATA: Static vs. Dynamic Policy (2)

Dispatcher

Priority level 2

Priority level 1
ATA: Static vs. Dynamic Policy (3)

- Copy the computed response time
- Timed automata with updates
- Keeps the number of clocks bounded
ATA: The Encoding

- A quick example: task3 -> task1 -> task2
ATA: The Encoding

- A quick example: task3 -> task1 -> task2
ATA: The Encoding

- A quick example: task3 -> task1 -> task2

\[ \text{task}_1 \quad \text{c}_1 \]

\[ \text{task}_3 \quad \text{c}_3 \]
ATA: The Encoding

- A quick example: task3 -> task1 -> task2

```
task_1
  c_1

C_3

- C_3
```
ATA: The Encoding

- A quick example: task3 -> task1 -> task2

```
    task_1  c_1

    task_3  c_3
```
ATA: The Encoding

- A quick example: task3 -> task1 -> task2
ATA: The Encoding

- A quick example: task3 -> task1 -> task2

\[
\begin{align*}
\text{task}_1 & \quad \boxed{C_1} \\
\text{task}_2 \\
\text{task}_3 & \quad \boxed{C_3}
\end{align*}
\]
ATA: The Encoding

- A quick example: task3 -> task1 -> task2
ATA: The Encoding

- A quick example: task3 -> task1 -> task2
ATA: The Encoding

- A quick example: task3 -> task1 -> task2
Conclusion

- Verification of systems that adapt to the properties of the task set.
- Extension of ATA to dynamic scheduling policies.

Future work

- Tool support.
- Increased flexibility.
Image credits

- *NASA Mars Rover* image by Maas Digital LLC for Cornell University and NASA/JPL
- Blue Marble by NASA
- *Smartphone* image by tuexperto_com from Flickr (CC-BY-SA) via [fotopedia.com](http://fotopedia.com)