Parking Can Get You There Faster
Model Augmentation to Speed up Real-Time Model Checking

Oliver Möller
BRICS
University of Aarhus, Denmark
omoeller@brics.dk
Timed Automata (UPPAAL Flavor)

clocks: x,y

guards: y==10, x>LARGE, x<10

invariants: y<=10

urgency: location S
Timed Automata (UPPAAL Flavor)

clocks:  x,y
guards:  y==10, x>LARGE, x<10
invariants:  y<=10
urgency:  location S

network of timed automata
hand-shake synchronization
discrete data types
...

\[
\begin{align*}
x &> \text{LARGE} \\
y &\leq 0 \\
x &< 10 \\
\text{QUICK} \\
y &\leq 10
\end{align*}
\]

\[
\begin{align*}
y &:= 0 \\
x &\leq \text{LARGE} \\
y &\leq 10 \\
\text{S}
\end{align*}
\]
Symbolic Forward Reachability

\[ x > 3 \]

\[ y := 0 \]
Symbolic Forward Reachability

\[ x > 3 \]

\[ y := 0 \]

\[ 1 \leq x \leq 4 \]
\[ 1 \leq y \leq 2 \]
Symbolic Forward Reachability

\[ n \]
\[ x > 3 \]
\[ y := 0 \]
\[ m \]

\[ 1 \leq x \leq 4 \]
\[ 1 \leq y \leq 2 \]

\[ -2 \leq x-y \leq 3 \]

Delays to

\[ y \]

\[ 1 \leq x \leq 4 \]
\[ 1 \leq y \leq 2 \]
Symbolic Forward Reachability

\[ x > 3 \]
\[ y := 0 \]

1 \leq x \leq 4
1 \leq y \leq 2

intersects to

3 < x
1 \leq y

-2 \leq x-y \leq 3
Symbolic Forward Reachability

Symbolic Forward Reachability

\[ x > 3 \]
\[ y := 0 \]

\[ 1 \leq x \leq 4 \]
\[ 1 \leq y \leq 2 \]

delays to

\[ 1 \leq y \]
\[ -2 \leq x-y \leq 3 \]

intersects to

\[ 3 < x \]
\[ 1 \leq y \]

projects to

\[ 3 < x \]
\[ y = 0 \]
Forward State Space Exploration

Algorithm: \textit{Reachability}

\begin{enumerate}
\item \textbf{input: } \textit{Goal} : \((\vec{l}_g; v_g)\)
\item \textit{Passed} := \{\}; \textit{Waiting} := \{(\vec{l}_0; v_0\uparrow\vec{l}_0)\}
\end{enumerate}

\textbf{REPEAT}

\textbf{FORALL} \((\vec{l}; v) \in \textit{Waiting}\)

\textbf{IF} \(\forall(\vec{l}; v') \in \textit{Passed}. v \not\subseteq v'\) \textbf{THEN}

\(\textit{Passed} := \textit{Passed} \cup (\vec{l}; v)\)

\textbf{FORALL} \((\vec{l}'; v')\) with \(\vec{l} \xrightarrow{g,r} \vec{l}'\)

\(v' := r(v \cap g)\)

\(v' \neq \emptyset\)

\(\textit{Waiting} := \textit{Waiting} \cup \{(\vec{l}'; v'\uparrow\vec{l}')\}\)

\textbf{UNTIL} \(\textit{Waiting} = \emptyset \vee \exists(\vec{l}, v) \in \textit{Passed}. \vec{l}_g \subseteq \vec{l} \land v_g \cap v \neq \emptyset\)
Problem: Repetitions in the State-Space

T is visited repeatedly

state space at control point T
Outline

1 Model Augmentation Technique
2 Application to RCX Bricks Sorter Model
3 Extension to Universal Path Properties
Idea: Subsume Many Small Steps by a Big One

\[
\begin{align*}
\text{S} & \quad \text{T} \\
\text{y} \leq 0 & \quad \text{x} \leq \text{LARGE} \\
x < 10 & \quad \text{y} = 10 \\
\text{QUICK} & \quad \text{NEW WAY TO REACH} \\
\text{y} \leq 10 & \quad \text{STATE SPACE AT CONTROL POINT} \\
x < 10 & \quad \text{NEW WAY TO REACH} \\
\text{LARGE} & \quad \text{NEW WAY TO REACH}
\end{align*}
\]
Idea: Subsume Many Small Steps by a Big One

\[ S \leq 0 \]
\[ y \leq 0 \]
\[ x > \text{LARGE} \]
\[ x < 10 \]
\[ y = 10 \]
\[ y = 0 \]
\[ x \leq \text{LARGE} \]
\[ x \leq \text{LARGE} \]
\[ y = 10 \]

new way to reach T state space at control point TPTS’01 7 April 2002 Oliver Möller: Parking Can Get You There Faster
Idea: Subsume Many Small Steps by a Big One

new way to reach $T$

state space at control point $T$
Model Checking: **QUICK** not reachable
Soundness for Safety

Crucial Observation:

every trace that was originally possible
is also possible after the modification

Therefore:

if a safety property $\text{A}[] \varphi$
can be established for the augmented model $\text{Aug}_A(M)$,
then it also holds for $M$. 
Challenges for Beneficial Augmentation

**Prerequisites**
- repetitions at one control point
- all processes can “park”
- return to the original control structure

**What to do?**
- find promising augmentation points
- identify suitable delays
- construct return conditions
Bricks Sorter Model

Processes of Sorter:

RCX_model
- Scheduler
- RCX0_main_task
- RCX0_kick_off_task

Environment
- black_brick
- black_brick2
- kick_off_arm
- Hurry_Dummy

Objective: Kick off all black bricks, but no red ones
From RCX to **UPPAAL**

**RCX program**

```plaintext
*** Var 0 = v
*** Var 1 = DELAY
*** Var 2 = LIGHT_LEVEL

*** Task 0 = main
000 Set var[1], 25 14 01 02 19 00
005 Set var[2], 42 14 02 02 2a 00
010 InType 0, Light 32 00 03
013 InMode 0, Percent 42 00 80
016 InType 2, Switch 32 02 01
019 InMode 2, Boolean 42 02 20
022 OutDir A, Fwd e1 81
024 OutMode A, On 21 81
026 OutPwr A, 1 13 01 02 01
030 Display 1 33 02 01 00
034 StartTask 1 71 01
036 Test Input(0) <= var[2], 47 95 09 00 00 00 02 05 00
044 Jump 36 72 89 00
047 ClearTimer 1 a1 01
050 PlaySound 1 51 01
051 Test Input(0) <= var[2], 51 95 09 00 00 00 02 fa ff
059 Test Timer(1) <= var[1], 70 95 01 00 01 00 01 05 00
067 Jump 78 72 0a 00
070 Test Input(0) <= var[2], 95 96 49 00 00 00 02 ef ff
078 Test Timer(1) <= var[1], 1a 95 01 00 01 00 01 0a 00
086 Set var[0], 1 14 00 02 01 00
091 Jump 36 72 b8 00
094 Test Input(0) >= var[2], 114 95 49 00 00 00 02 0e 00
102 ClearTimer 2 a1 02
104 PlaySound 1 51 01
106 Test Input(0) <= var[2], 106 95 09 00 00 00 02 fa ff
114 Test Timer(1) <= var[1], 114 95 01 00 01 00 01 fa ff
122 Set var[0], 1 14 00 02 01 00
127 Test Timer(2) <= var[1], 127 95 01 00 02 00 01 fa ff
136 Set var[0], 1 14 00 02 01 00
140 Jump 36 72 e9 00

*** Task 1 = skub_af
000 Set var[0], 0 14 00 02 00 00
005 Test 0 <= var[0], 48 95 42 00 00 00 00 25 00
013 Set var[0], 0 14 00 02 00 00
018 OutDir C, Rev e1 04
020 OutMode C, On 21 84
022 OutPwr C, 1 13 04 02 01
026 Delay 6 43 02 06 00
030 OutDir C, Fwd e1 84
032 OutMode C, On 21 84
034 OutPwr C, 1 13 04 02 01
038 Test 1 <= Input(2), 38 95 82 09 01 00 02 fa ff
046 OutMode C, Off 21 44
048 Jump 5 72 ac 00
```

(automatic) translation to **UPPAAL**
Augmentation of Wait Loops

*** Task 0 = main

...  
031 InType 2, Switch  
034 InMode 2, Boolean  
037 OutDir A, Fwd  
039 OutMode A, On  
041 OutPwr A, 1  
045 OutDir B, Fwd  
047 OutMode B, On  
049 OutPwr B, 6  
053 Display 1  
057 StartTask 1  
059 Test Input(0) <= var[4], 70  
067 Jump 59  
070 ...

waiting for an input  
'jumping' to this input
Augmenting the Scheduler

Original Scheduler Process

Augmented Scheduler Process
Augmentations in Total

- 9 model augmentations: 6 input / 3 time condition
- 16 new locations
- 34 new transitions

- NO locations/transitions removed
- NO new variables or clocks
Model Checking a (True) Safety Property

<table>
<thead>
<tr>
<th></th>
<th>#explored states</th>
<th>successors (average)</th>
<th>#deadlocks</th>
<th>time [sec]</th>
<th>memory [KB]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sorter</td>
<td>151·103</td>
<td>1.28</td>
<td>0</td>
<td>86.84</td>
<td>1´840</td>
</tr>
<tr>
<td>$Aug^*_2\alpha(Sorter)$</td>
<td>22·966</td>
<td>2.09</td>
<td>20</td>
<td>21.15</td>
<td>2´512</td>
</tr>
</tbody>
</table>

- both runs used *convex hull* over-approximation
- number of symbolic states changes significantly
- higher non-determinism
- deadlocks necessarily spurious
Universal Path Properties

\[ \zeta ::= A[]\zeta \mid A<>\zeta \mid \zeta \lor \zeta \mid \zeta \land \zeta \mid \varphi \]

\(\varphi\) is a local property, i.e., depends only on the current configuration.

A trace \(\sigma = (s_0, s_1, \ldots)\) satisfies \(\zeta\) at position \(i\) iff:

\[
\begin{align*}
(\sigma, i) \models A[]\zeta & \iff \forall j \geq i. (\sigma, j) \models \zeta \\
(\sigma, i) \models A<>\zeta & \iff \exists j \geq i. (\sigma, j) \models \zeta \\
(\sigma, i) \models \zeta_1 \lor \zeta_2 & \iff (\sigma, i) \models \zeta_1 \text{ or } (\sigma, i) \models \zeta_2 \\
(\sigma, i) \models \zeta_1 \land \zeta_2 & \iff (\sigma, i) \models \zeta_1 \text{ and } (\sigma, i) \models \zeta_2 \\
(\sigma, i) \models \varphi & \iff s_i \models \varphi
\end{align*}
\]

A timed automata model satisfies \(\zeta\), if all traces satisfy \(\zeta\) at position 0.
Problem: Augmentation Can Remove Deadlocks

Process $P$:

\[ x \leq 10 \]

\[ x < 5 \]

Formula: $A<> P.B$ (inevitably $B$)

not true: could get stuck at $A$
Problem: Augmentation Can Remove Deadlocks

Process $P$:

$x \leq 10$ (augmented)

Formula: $A \not< P.B$ (inevitably $B$)

not true: could get stuck at $A$

SUDDENLY HOLDS IN AUGMENTED MODEL
Problem: Augmentation Can Remove Deadlocks

Process $P$:

$x \leq 10$ (augmented)

Formula: $A<> P.B$ (inevitably B)

not true: could get stuck at A

SUDDENLY HOLDS IN AUGMENTED MODEL

Solution: change step semantics
Allow augmented transitions only if another (non-augmented) transition is enabled
Modified Step Semantics

\( M \): UPPAAL timed automata model

\( \mathfrak{A} \): \( \langle l_i \xrightarrow{g,a} l', L_{\mathfrak{A}}, T_{\mathfrak{A}}, \text{Type}_{\mathfrak{A}} \rangle \) a model augmentation of \( M \)

- define the weak traces of \( \text{Aug}_{\mathfrak{A}}(M) \) as the those where

  in a \((l, e, \nu)\) with \( l_i \in l\),

  the action transition \( l_i \xrightarrow{g,a} l' \) is only taken, if another action transition is enabled

- this yields \( T_{\mathfrak{A}}(\text{Aug}_{\mathfrak{A}}(M)) \subseteq T(\text{Aug}_{\mathfrak{A}}(M)) \)

- \( \text{Aug}_{\mathfrak{A}}(M) \models_{\mathfrak{A}} \zeta \) if and only if

  \( \forall \) traces \( \sigma = (s_0, s_1, \ldots) \in T_{\mathfrak{A}}(\text{Aug}_{\mathfrak{A}}(M)). \ (\sigma, 0) \models \zeta \)

**Theorem:**

\( \text{Aug}_{\mathfrak{A}}(M) \models_{\mathfrak{A}} \zeta \quad \Rightarrow \quad M \models \zeta \)
Conclusions

- over-approximation
  - *sound* but inherently *not complete*
- shifted-clock repetition seems to be specific to *real-time*
- technique has potential in special scenarios
- automation highly desirable
  - (and possible, but not done)