Timed Automata & Model Checking
Using UPPAAL: $x \in \{1,2,3,4\}$

Kim Guldstrand Larsen
BRICS@Aalborg & FMT@Twente

Validation & Verification
Construction of UPPAAL models

...and Beyond
Synthesis of Control Program

Timed Automata review

Alur & Dill 1990

Clocks: $x, y$

Guard:

$x \leq 5 \land y > 3$

$x := 0$

Transitions:

$(a, x=2.4, y=3.1415) \xrightarrow{0.1} (a, x=3.5, y=4.1515)$

$(a, x=2.4, y=3.1415) \xrightarrow{3.2} (a, x=3.5, y=4.1515)$

Location Invariants:

$g_1 = x \leq 5 \land y > 3$

$g_2 = x = 0$

$g_3 = x \leq 2$

$g_4 = x > 0$

Constraints

Definition:

Let $X$ be a set of clock variables. The set $B(X)$ of clock constraints $\phi$ is given by the grammar:

$\phi ::= x \leq c \mid c \leq x \mid x < c \mid c < x \mid \phi_1 \land \phi_2$

where $c \in \mathbb{N}$ (or $\mathbb{Q}$).
Clock Valuations and Notation

Definition
The set of clock valuations, $\mathbb{R}^C$, is the set of functions $C \rightarrow \mathbb{R}_{\geq 0}$ ranged over by $u,v,w,\ldots$

Notation
Let $u \in \mathbb{R}^C$, $r \subseteq C$, $d \in \mathbb{R}_{\geq 0}$, and $g \in B(X)$ then:
- $u \rightarrow d \in \mathbb{R}^C$ is defined by $(u + d)(x) = u(x) + d$ for any clock $x$
- $u[r](x) \in \mathbb{R}^C$ is defined by $u[r](x) = 0$ when $x \in r$ and $u[r](x) = u(x)$ for $x \notin r$.
- $u \models g$ denotes that $g$ is satisfied by $u$.

Timed Automata

Definition
A timed automaton $A$ over clocks $C$ and actions $Act$ is a tuple $(A_0, A_0, E, L)$, where:
- $L$ is a finite set of locations
- $A_0 \subseteq L$ is the initial location
- $E \subseteq L \times B(X) \times Act \times \mathcal{P}(C) \times L$ is the set of edges
- $I : L \rightarrow B(X)$ assigns to each location an Invariant

Semantics

Definition
The semantics of a timed automaton $A$ is a labelled transition system with state space $L \times \mathbb{R}^C$ with initial state $(A_0, w_0)^*$ and with the following transitions:
- $(i, u) \stackrel{\text{guard}}{\rightarrow} (i, u + d)$ iff $u \models \ell(i)$ and $u + d \models \ell(i)$.
- $(i, u) \stackrel{\text{reset-set}}{\rightarrow} (i', u')$ iff there exists $(i, g, a, r, l') \in E$ such that
  - $u \models g$,
  - $u' = u[r]$, and
  - $u' \models \ell(l')$.
- $uw(x) = 0$ for all $x \in C$.

Timed Automata: Example

Timed Automata: Example
**Timed Automata: Example**

![Timed Automata Diagram]

**Fundamental Results**

- **Reachability**
  - Alur, Dill
- **Trace-inclusion**
  - Alur, Dill
  - Alur, Dill
- **Bisimulation**
  - Timed; Untimed
- **Model-checking**
  - TCTL, Tmu, Lnu...

**Updatable Timed Automata**

<table>
<thead>
<tr>
<th>W Diagonals</th>
<th>Diagonal-free</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x \leq 0$, $y \leq 1$</td>
<td>Porse complete</td>
</tr>
<tr>
<td>$x = 0$, $y = 1$</td>
<td>Unstable</td>
</tr>
<tr>
<td>$x \geq 2$, $y \leq 0$</td>
<td>Porse complete</td>
</tr>
<tr>
<td>$z = 3$, $y = 4$</td>
<td>Unstable</td>
</tr>
</tbody>
</table>

**Other Extensions**

- Ordinary clocks: $x \cdot rate(1)
- Integer variables: $x \cdot rate(0)
- Stopwatches: $x \cdot rate(0)$ or $x \cdot rate(1)$ (loc.deps)
- Cost: $c \cdot rate(n)$ where $n$ is in Nat, however $c$ cannot be guarded
- Const. slope clocks: $x \cdot rate(n)$ where $n$ is in Nat
- Parameters: $x \cdot rate(0)$ (and NOT assignable)
- Multirate clocks: Linear Hybrid Automata
- Linear guards & linear assignment

**Parallel Composition (a'la CCS)**

![Parallel Composition Diagram]

**The UPPAAL Model**

- Networks of Timed Automata + Integer Var + Array Var + ....

**Example transitions**

- $(l_1, m_1, n_1, x \geq 2, y \geq 3, i \geq 2, \ldots)$
- $(l_2, m_2, n_2, x \geq 3, y \geq 4, \ldots)$
- $(l_3, m_3, n_3, x \geq 2, y \geq 5, \ldots)$
- $(l_4, m_4, n_4, x \geq 3, y \geq 6, \ldots)$

**Two-way synchronization on complementary actions.**

**Closed Systems!**
LEGO Mindstorms/RCX

- Sensors: temperature, light, rotation, pressure.
- Actuators: motors, lamps.
- Virtual machine: 10 tasks, 4 timers, 16 integers.
- Several Programming Languages: NotQuiteC, Mindstorm, Robotics, legOS, etc.

Exercise: Design Controller so that only black boxes are being pushed out.

NQC programs

```c
int active;
int DELAY;
int LIGHT_LEVEL;

task MAIN{
    DELAY=75;
    LIGHT_LEVEL=35;
    active=0;
    Sensor(IN_1, IN_LIGHT);
    Fwd(OUT_A,1);
    Display(1);
    start PUSH;
    while(true){
        wait(IN_1<=LIGHT_LEVEL);
        ClearTimer(1);
        active=1;
        PlaySound(1);
        wait(IN_1>LIGHT_LEVEL);
    }
}
```

UPPAAL Demo

From RCX to UPPAAL

- Model includes Round-Robin Scheduler.
- Compilation of RCX tasks into TA models.
- Presented at ECRTS 2000
Case-Studies: Controllers

- Gearbox Controller [TACAS'98]
- Bang & Olufsen Power Controller [RTSS'99, FTRTFT'2k]
- SIDMAR Steel Production Plant [RTCSA'99, DSV'2k]
- Real-Time RCX Control-Programs [ECRTS'2k]
- Experimental Batch Plant (2000)
- RCX Production Cell (2000)
- Terma, Memory Management for Radar (2001)

Case Studies: Protocols

- Philips Audio Protocol [HS'95, CAV'95, RTSS'95, CAV'96]
- Collision-Avoidance Protocol [SPIN'95]
- Bounded Retransmission Protocol [TACAS'97]
- Bang & Olufsen Audio/Video Protocol [RTSS'97]
- TDMA Protocol [PRFTS'97]
- Lip-Synchronization Protocol [FMICS'97]
- Multimedia Streams [DSVIS'98]
- ATM ABR Protocol [CAV'99]
- ABB Fieldbus Protocol [ECRTS'2k]

UPPAAL 3.2 (and 3.3, 3.4)

Released October 01

- Graphical User Interface
  - XML based file format
  - Better syntax-error indication
  - Drop-and-drag for transitions
  - Changed menu
- Verification Engine
  - Restructured (increased flexibility)
  - Normalization-bug fixed
  - More freedom in combining optimization options
  - Deadlock checking
  - Support for more general properties (E[p] ∨ A<>p, p→q)

Communication via channels and shared variable.
Zones
From infinite to finite

State
\( (n, x=3.2, y=2.5) \)

Symbolic state (set)
\( (n, 1 \leq x \leq 4, 1 \leq y \leq 3) \)

Zone: conjunction of
\( x \leq n, x \geq n \)

Symbolic Transitions

\( x > 3 \)

\( y := 0 \)

Symbolic Transitions

\( 1 \leq x \leq 4, 1 \leq y \leq 3 \)

\( 1 \leq x, 1 \leq y \)

\(-2 \leq x - y \leq 3 \)

\( 3 < x, y = 0 \)

Thus \( (n, 1 \leq x \leq 4, 1 \leq y \leq 3) = a \Rightarrow (m, 3 < x, y = 0) \)

Fischer's Protocol
analysis using zones

Init:
\( V = 1 \)

Critical Section

Fischers cont.

Fischers cont.

Untimed case

Taking time into account

Fischers cont.

Fischers cont.

Untimed case

Taking time into account
Fischers cont.

Untimed case

\[ A_1, A_2, v = 1 \]
\[ A_1, B_2, v = 2 \]
\[ A_1, CS_2, v = 2 \]
\[ B_1, CS_2, v = 1 \]
\[ CS_1, CS_2, v = 1 \]

Taking time into account

\[ X \leq 10 \]
\[ X > 10 \]
\[ Y \leq 10 \]
\[ Y > 10 \]
\[ X < 10 \]

Forward Rechability

INITIAL Passed := \( \emptyset \);
Waiting := \( ((n0, Z0)) \)

REPEAT
- pick \( (n, Z) \) in Waiting
- if for some \( Z' \geq Z \)
  \( (n, Z') \) in Passed then STOP
- else \( \text{explore} \) add \( \{ (m, U) : (n, Z) \Rightarrow (m, U) \} \) to Waiting;

UNTIL Waiting = \( \emptyset \)
or Final is in Waiting

Init \( \Rightarrow \) Final ?
Forward Reachability

\[
\text{INITIAL} \quad \text{Passed} \leftarrow \emptyset; \quad \text{Waiting} := ((n, Z))
\]

\[
\text{REPEAT}
\]

\[
\text{- pick } (n, Z) \text{ in Waiting}
\]

\[
\text{- if for some } Z' \subseteq Z \text{ then STOP}
\]

\[
\text{- else (otherwise) add } \{(m, U) : (n, Z) \Rightarrow (m, U)\} \text{ to Waiting;}
\]

\[
\text{Add } (n, Z) \text{ to Passed}
\]

\[
\text{UNTIL } \text{Waiting} = \emptyset \text{ or } \text{Final is in } \text{Waiting}
\]

Canonical Datastructures for Zones

**Difference Bounded Matrices**

Bellman 1958, Dill 1989

**Inclusion**

\[
\begin{array}{c}
\text{D1} \\
\text{x} \leq 1 \\
y \geq 5 \\
y - x \leq 3
\end{array}
\]

\[
\begin{array}{c}
\text{D2} \\
x \leq 2 \\
y \geq 3 \\
y - x \leq 3
\end{array}
\]

**Emptiness**

\[
\begin{array}{c}
\text{D} \\
x \leq 1 \\
y \leq 3
\end{array}
\]

**Negative Cycle**

iff empty solution set

**Future**

\[
\begin{array}{c}
\text{D} \\
1 \leq x \leq 4 \\
1 \leq y \leq 3
\end{array}
\]

**Remove all bounds involving y and set y to 0**

\[
\begin{array}{c}
\text{D} \\
1 \leq x, 1 \leq y \\
2 \leq x - y \leq 3
\end{array}
\]

**Canonical Datastructures for Zones**

Bellman 1958, Dill 1989

**Difference Bounded Matrices**

**Inclusion**

\[
\begin{array}{c}
\text{D1} \\
x \leq 1 \\
y \leq 5 \\
y - x \leq 3
\end{array}
\]

\[
\begin{array}{c}
\text{D2} \\
x \leq 2 \\
y \leq 3 \\
y - x \leq 3
\end{array}
\]

**Emptiness**

\[
\begin{array}{c}
\text{D} \\
x \leq 1 \\
y \leq 3
\end{array}
\]

**Negative Cycle**

iff empty solution set

**Reset**

\[
\begin{array}{c}
\text{D} \\
x \leq 1 \\
y \leq 3
\end{array}
\]

**Remove all bounds involving y and set y to 0**

\[
\begin{array}{c}
\text{D} \\
x \leq 1 \\
y \leq 3
\end{array}
\]
**Canonical Datastructure for Zones**
*Difference Bounded Matrices*

Bellman'58, Dill'89

1 = x1 - x2 <= 4
2 = x2 - x1 <= 10
3 = x3 - x1 <= 2
4 = x2 - x3 <= 2
5 = x0 - x1 <= 3
6 = x3 - x0 <= 5

Shortest Path Closure O(n^3)

Bellman'58, Dill'89

**New Canonical Datastructure**

*Minimal collection of constraints*

Bellman'58, Dill'89

Shortest Path Closure O(n^3)

Space worst O(n^2)

**SPACE PERFORMANCE**

**TIME PERFORMANCE**

**Shortest Path Reduction**

1st attempt

Idea

An edge is REDUNDANT if there exists an alternative path of no greater weight. 

**Problem**

v and w are both redundant. Removal of one depends on presence of other.

**Observation**: If no zero- or negative cycles then SAFE to remove all redundancies.

**Solution**

G: weighted graph
Shortest Path Reduction
Solution

G: weighted graph

1. Equivalence classes based on 0-cycles.
2. Graph based on representatives.
   Safe to remove redundant edges
3. Shortest Path Reduction
   One cycle per class
   Removal of redundant edges between classes

Earlier Termination

INITIAL Passed := Ø;
Waiting := {(n0,Z0)}

REPEAT
- pick (n,Z) in Waiting
  - if for some Z' Z (n,Z') in Passed then STOP
  - else (explor(e) add
    (m,U) : (n,Z) => (m,U))
    to Waiting;
    Add (n,Z) to Passed
UNTIL Waiting = Ø or Final is in Waiting
Clock Difference Diagrams

= Binary Decision Diagrams + Difference Bounded Matrices

CAV99

- Nodes labeled with differences
- Maximal sharing of substructures (also across different CDDs)
- Maximal intervals
- Linear-time algorithms for set-theoretic operations.

- NDD’s Maler et. al
- DDD’s Møller, Lichtenberg

**SPACE PERFORMANCE**

- CDD
- Reduced CDD
- CDD+BDD

**TIME PERFORMANCE**

- CDD
- Reduced CDD
- CDD+BDD

**UPPAAL 1995 - 2001**

Every 9 months
10 times better performance!