



UPPAAL

Verification Engine, Options & Patterns


Alexandre David
1.2.05





BRICS
Basic Research
in Computer Science






Outline

- UPPAAL
 - Modelling Language
 - Specification Language
- UPPAAL Verification Engine
 - Symbolic exploration algorithm
 - Zones & DBMs
- Verification Options
- Modelling Patterns

12-02-2008 Alexandre David, TOV'08 2


Modelling Language



Modeling Language


- Network of TA = instances of templates
 - argument *const type expression*
 - argument *type & name*
- Types
 - built-in types: *int*, *int[min,max]*, *bool*, arrays
 - *typedef struct { ... } name*
 - *typedef built-in-type name*
- Functions
 - C-style syntax, no pointer but references OK.
- Select
 - *name : type*

12-02-2008 Alexandre David, TOV'08 4



Un-timed Example: Jugs

Jugs

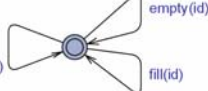


2 5

Actions:

- fill
- empty
- pour


`Jug(const id_t id)`



Goal: obtain 1 unit.

- Scalable, compact, & readable model.
 - `const int N = 2; typedef int[0,N-1] id_t;`
 - Jugs have their own *id*.
 - Actions = functions.
 - Pour: from *id* to another *k* different from *id*.

12-02-2008 Alexandre David, TOV'08 5



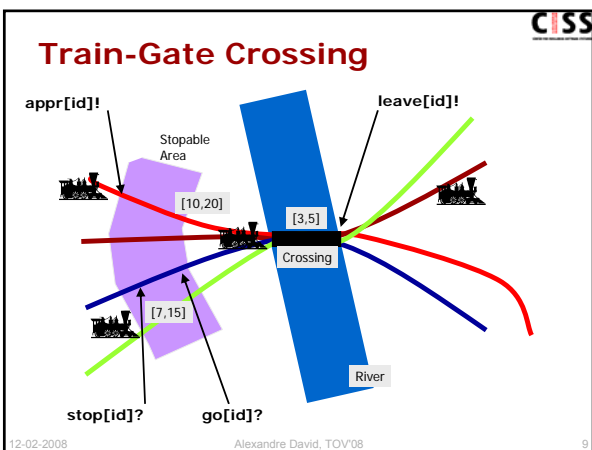
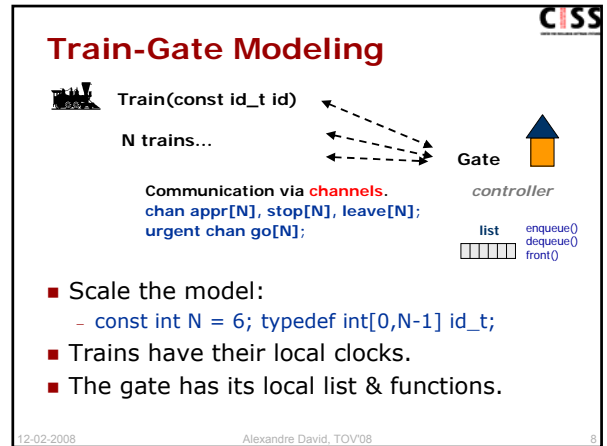
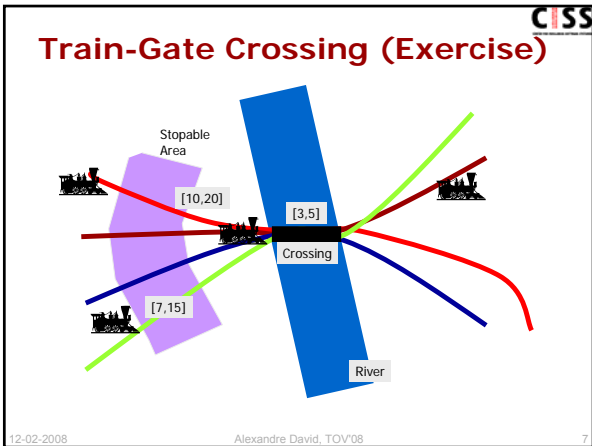
Jugs cont.

- Jug levels & capacities:


```
int level[N];
const int capa[N] = {2,5};
```
- `void empty(id_t i) { level[i]=0; }`
- `void fill(id_t i) { level[i] = capa[i]; }`
- `void pour(id_t i, id_t j)`

```
{
  int max = capa[j] - level[j];
  int poured = level[i] <? max;
  level[i] -= poured;
  level[j] += poured;
}
```
- Auto-instantiation: `system Jug;`

12-02-2008 Alexandre David, TOV'08 6



Specification Language

Logical Specifications

- Validation Properties**
 - Possibly: $E \ll P$
- Safety Properties**
 - Invariant: $A[] P$
 - Pos. Inv.: $E[] P$
- Liveness Properties**
 - Eventually: $A \ll P$
 - Leadsto: $P \rightarrow Q$
- Bounded Liveness**
 - Leads to within: $P \rightarrow_{\leq t} Q$

The expressions P and Q must be type safe, side effect free, and evaluate to a boolean.

Only references to integer variables, constants, clocks, and locations are allowed (and arrays of these).

12-02-2008 Alexandre David, TOV'08 11

Logical Specifications

- Validation Properties**
 - Possibly: $E \ll P$
- Safety Properties**
 - Invariant: $A[] P$
 - Pos. Inv.: $E[] P$
- Liveness Properties**
 - Eventually: $A \ll P$
 - Leadsto: $P \rightarrow Q$
- Bounded Liveness**
 - Leads to within: $P \rightarrow_{\leq t} Q$

12-02-2008 Alexandre David, TOV'08 12

CISS

Logical Specifications

- Validation Properties
 - Possibly: $E \langle \rangle P$
- Safety Properties
 - Invariant: $A [] P$
 - Pos. Inv.: $E [] P$
- Liveness Properties
 - Eventually: $A \langle \rangle P$
 - Leadsto: $P \rightarrow Q$
- Bounded Liveness
 - Leads to within: $P \rightarrow_{\leq t} Q$

12-02-2008 Alexandre David, TOV'08 13

CISS

Logical Specifications

- Validation Properties
 - Possibly: $E \langle \rangle P$
- Safety Properties
 - Invariant: $A [] P$
 - Pos. Inv.: $E [] P$
- Liveness Properties
 - Eventually: $A \langle \rangle P$
 - Leadsto: $P \rightarrow Q$
- Bounded Liveness
 - Leads to within: $P \rightarrow_{\leq t} Q$

12-02-2008 Alexandre David, TOV'08 14

CISS

Logical Specifications

- Validation Properties
 - Possibly: $E \langle \rangle P$
- Safety Properties
 - Invariant: $A [] P$
 - Pos. Inv.: $E [] P$
- Liveness Properties
 - Eventually: $A \langle \rangle P$
 - Leadsto: $P \rightarrow Q$
- Bounded Liveness
 - Leads to within: $P \rightarrow_{\leq t} Q$

12-02-2008 Alexandre David, TOV'08 15

CISS

Jug Example

- Safety: Never overflow.
 - $A [] \text{forall}(i: \text{id_t}) \text{level}[i] \leq \text{capa}[i]$
- Validation/Reachability: How to get 1 unit.
 - $E \langle \rangle \text{exists}(i: \text{id_t}) \text{level}[i] == 1$

12-02-2008 Alexandre David, TOV'08 16

CISS

Train-Gate Crossing

- Safety: One train crossing.
 - $A [] \text{forall}(i: \text{id_t}) \text{forall}(j: \text{id_t}) \text{Train}(i).\text{Cross} \ \&\& \ \text{Train}(j).\text{Cross} \ \text{imply} \ i == j$
- Liveness: Approaching trains eventually cross.
 - $\text{Train}(0).\text{Appr} \ \text{-->} \ \text{Train}(0).\text{Cross}$
 - $\text{Train}(1).\text{Appr} \ \text{-->} \ \text{Train}(1).\text{Cross}$
 - ...
- No deadlock.
 - $A [] \text{not deadlock}$

12-02-2008 Alexandre David, TOV'08 17

UPPAAL Verification Engine

CISS

Overview

- Zones and DBMs
- Reachability algorithm revisited
- Minimal Constraint Form
- Clock Difference Diagrams

- Distributed UPPAAL [CAV2000, STTT2004]
- Unification & Sharing [FTRTFT2002, SPIN2003]
- Acceleration [FORMATS2002]
- Static Guard Analysis [TACAS2003, TACAS2004]
- Storage-Strategies [CAV2003]

12-02-2008 Alexandre David, TOV'08 19

CISS

Zones

From infinite to finite

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

Symbolic state (set)
(n, 1·x<4, 1·y<3)

Zone:
conjunction of
 $x-y < n$,
 $x < n$,
 $x > = n$

12-02-2008 Alexandre David, TOV'08 20

CISS

Symbolic Transitions

Thus $(n, 1 \leq x \leq 4, 1 \leq y \leq 3) \xrightarrow{a} (m, 3 < x, y = 0)$

12-02-2008 Alexandre David, TOV'08 21

CISS

Symbolic Exploration

Reachable?

12-02-2008 Alexandre David, TOV'08 22

CISS

Symbolic Exploration

Reachable?

12-02-2008 Alexandre David, TOV'08 23

CISS

Symbolic Exploration

Reachable?

12-02-2008 Alexandre David, TOV'08 24

Symbolic Exploration

12-02-2008 Alexandre David, TOV'08 25

Symbolic Exploration

12-02-2008 Alexandre David, TOV'08 26

Symbolic Exploration

12-02-2008 Alexandre David, TOV'08 27

Symbolic Exploration

12-02-2008 Alexandre David, TOV'08 28

Symbolic Exploration

12-02-2008 Alexandre David, TOV'08 29

Symbolic Exploration

12-02-2008 Alexandre David, TOV'08 30

CISS

Forward Reachability Algorithm

Init -> Final ?

PW

Waiting

Final

Init

Passed

INITIAL Passed := \emptyset ;
Waiting := $\{(n_0, Z_0)\}$

REPEAT

UNTIL Waiting = \emptyset
return false

12-02-2008 Alexandre David, TOV'08 32

CISS

Forward Reachability Algorithm

Init -> Final ?

PW

Waiting

Final

Init

Passed

INITIAL Passed := \emptyset ;
Waiting := $\{(n_0, Z_0)\}$

REPEAT
pick (n,Z) in Waiting

UNTIL Waiting = \emptyset
return false

12-02-2008 Alexandre David, TOV'08 33

CISS

Forward Reachability Algorithm

Init -> Final ?

PW

Waiting

Final?

Init

Passed

INITIAL Passed := \emptyset ;
Waiting := $\{(n_0, Z_0)\}$

REPEAT
pick (n,Z) in Waiting
if (n,Z) = Final return true

UNTIL Waiting = \emptyset
return false

12-02-2008 Alexandre David, TOV'08 34

CISS

Forward Reachability Algorithm

Init -> Final ?

PW

Waiting

Final

Init

Passed

INITIAL Passed := \emptyset ;
Waiting := $\{(n_0, Z_0)\}$

REPEAT
pick (n,Z) in Waiting
if (n,Z) = Final return true
for all (n,Z) \rightarrow (n',Z'):
if for some (n',Z') $Z' \subseteq Z''$ continue

UNTIL Waiting = \emptyset
return false

12-02-2008 Alexandre David, TOV'08 35

CISS

Forward Reachability Algorithm

Init -> Final ?

PW

Waiting

Final

Init

Passed

INITIAL Passed := \emptyset ;
Waiting := $\{(n_0, Z_0)\}$

REPEAT
pick (n,Z) in Waiting
if (n,Z) = Final return true
for all (n,Z) \rightarrow (n',Z'):
if for some (n',Z') $Z' \subseteq Z''$ continue
else add (n',Z') to Waiting

UNTIL Waiting = \emptyset
return false

12-02-2008 Alexandre David, TOV'08 36

CISS

Forward Reachability Algorithm

Init -> Final ?

PW

Waiting

Final

Init

Passed

INITIAL Passed := \emptyset ;
Waiting := $\{(n_0, Z_0)\}$

REPEAT
pick (n,Z) in Waiting
if (n,Z) = Final return true
for all (n,Z) \rightarrow (n',Z'):
if for some (n',Z') $Z' \subseteq Z''$ continue
else add (n',Z') to Waiting
move (n,Z) to Passed

UNTIL Waiting = \emptyset
return false

12-02-2008 Alexandre David, TOV'08 37

CISS

Forward Reachability Algorithm

Init -> Final ?

PW

Waiting

Init

Passed

Final

INITIAL Passed := 0;
Waiting := {(n₀, Z₀)}

REPEAT
pick (n, Z) in Waiting
if (n, Z) = Final return true
for all (n', Z') ∈ Z ⊆ Z':
if for some (n', Z') Z' ⊆ Z' continue
else add (n', Z') to Waiting
move (n, Z) to Passed

UNTIL Waiting = 0
return false

12-02-2008 Alexandre David, TOV'08 38

CISS

Difference Bound Matrices

$x_0 - x_0 \leq 0$	$x_0 - x_1 \leq -2$	$x_0 - x_2 \leq -1$
$x_1 - x_0 \leq 6$	$x_1 - x_1 \leq 0$	$x_1 - x_2 \leq 3$
$x_2 - x_0 \leq 5$	$x_2 - x_1 \leq 1$	$x_2 - x_2 \leq 0$

$x_i - x_j \leq C_{ij}$

Zone

12-02-2008 Alexandre David, TOV'08 39

CISS

Difference Bound Matrices

$x_0 - x_0 \leq 0$	$x_0 - x_1 \leq -2$	$x_0 - x_2 \leq -1$
$x_1 - x_0 \leq 6$	$x_1 - x_1 \leq 0$	$x_1 - x_2 \leq 3$
$x_2 - x_0 \leq 5$	$x_2 - x_1 \leq 3$	$x_2 - x_2 \leq 0$

$x_i - x_j \leq C_{ij}$

Canonical representation:
All constraints as tight as possible.
Needed for inclusion checking.
→ Unique DBM to represent a zone.

12-02-2008 Alexandre David, TOV'08 40

CISS

How to Make DBMs Canonical?

Bellman 1958, Dill 1989

How to check for inclusion?

D1

$x <= 1$
 $y - x <= 2$
 $z - y <= 2$
 $z <= 9$

Graph

? ⊆ ?

D2

$x <= 2$
 $y - x <= 3$
 $y <= 3$
 $z - y <= 3$
 $z <= 7$

Graph

12-02-2008 Alexandre David, TOV'08 41

CISS

How to Make DBMs Canonical?

How to check for inclusion?

D1

$x <= 1$
 $y - x <= 2$
 $z - y <= 2$
 $z <= 9$

Graph

Shortest Path Closure

? ⊆ ?

D2

$x <= 2$
 $y - x <= 3$
 $y <= 3$
 $z - y <= 3$
 $z <= 7$

Graph

Shortest Path Closure

12-02-2008 Alexandre David, TOV'08 42

CISS

Empty Zones/DBMs?

After the "closure" algorithm: one <0 in the diagonal.

D

$x <= 1$
 $y >= 5$
 $y - x <= 3$

Graph

Negative Cycle iff empty solution set

Compact

12-02-2008 Alexandre David, TOV'08 43

Canonical Datastructures for Zones

Difference Bounded Matrices

Future

$1 \leq x \leq 4$
 $1 \leq y \leq 3$

0		
∞	0	
∞		0

$1 \leq x, 1 \leq y$
 $-2 \leq x-y \leq 3$

Shortest Path Closure

Remove upper bounds on clocks

12-02-2008 Alexandre David, TOV'08 44

Canonical Datastructures for Zones

Difference Bounded Matrices

Reset

$1 \leq x, 1 \leq y$
 $-2 \leq x-y \leq 3$

$y=0, 1 \leq x$

Remove all bounds involving y and set y to 0

12-02-2008 Alexandre David, TOV'08 45

Canonical Datastructures for Zones

Difference Bounded Matrices

$x1-x2 \leq 4$
 $x2-x1 \leq 10$
 $x3-x1 \leq 2$
 $x2-x3 \leq 2$
 $x0-x1 \leq 3$
 $x3-x0 \leq 5$

Shortest Path Closure $O(n^3)$

Space n^2

12-02-2008 Alexandre David, TOV'08 46

Canonical Datastructures for Zones

Minimal Constraint Form

RTSS 1997

$x1-x2 \leq 4$
 $x2-x1 \leq 10$
 $x3-x1 \leq 2$
 $x2-x3 \leq 2$
 $x0-x1 \leq 3$
 $x3-x0 \leq 5$

Shortest Path Closure $O(n^3)$

Space worst $O(n^2)$ practice $O(n)$

Large gain in space. Small price in time.

12-02-2008 Alexandre David, TOV'08 47

Earlier Termination

Init -> Final ?

INITIAL Passed := \emptyset ;
Waiting := $\{(n_0, Z_0)\}$

REPEAT
pick (n, Z) in Waiting
if $(n, Z) = \text{Final}$ return true
for all $(n', Z') \in \text{PW}$:
if $(n', Z') \in \text{PW}$ continue
else add (n', Z') to Waiting
move (n, Z) to Passed

UNTIL Waiting = \emptyset
return false

Consider $U_i(n', Z')$

12-02-2008 Alexandre David, TOV'08 50

Clock Difference Diagrams

= Binary Decision Diagrams + Difference Bounded Matrices

CAV99

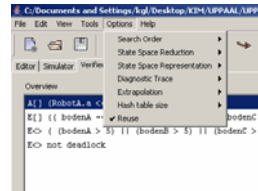
CDD-representations

- 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 Möller, Lichtenberg
- Past experiments showed gains in time & space.

12-02-2008 Alexandre David, TOV'08 51

Verification Options

Verification Options



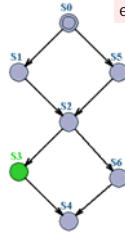
- Search Order
 - Depth First
 - Breadth First
- State Space Reduction
 - None
 - Conservative
 - Aggressive
- State Space Representation
 - DBM
 - Compact Form
 - Under Approximation
 - Over Approximation
- Diagnostic Trace
 - Some
 - Shortest
 - Fastest

12-02-2008

Alexandre David, TOV'08

55

State Space Reduction



However, **Passed** list useful for efficiency

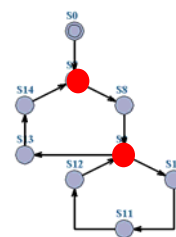
No Cycles: **Passed** list not needed for *termination*

12-02-2008

Alexandre David, TOV'08

56

State Space Reduction



Cycles:
Only symbolic states involving loop-entry points need to be saved on **Passed** list

12-02-2008

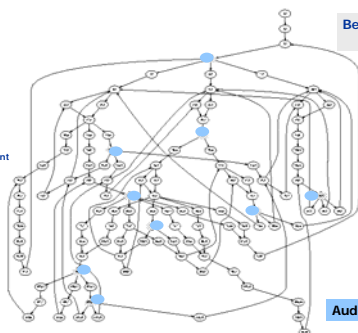
Alexandre David, TOV'08

57

To Store or Not To Store

117 states_{total}
81 states_{entrypoint}
9 states

Time OH less than 10%



Behrmann, Larsen, Pelanek 2003

Audio Protocol

12-02-2008

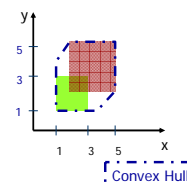
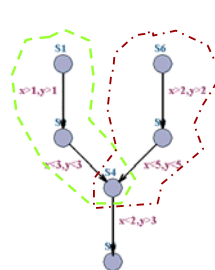
Alexandre David, TOV'08

58

Over-approximation

Convex Hull

TACAS04: An **EXACT** method performing as well as Convex Hull has been developed based on abstractions taking max constants into account.



12-02-2008

Alexandre David, TOV'08

60

Under-approximation Bitstate Hashing

12-02-2008 Alexandre David, TOV'08 61

Under-approximation Bitstate Hashing

12-02-2008 Alexandre David, TOV'08 62

Modelling Patterns

Variable Reduction

- Reduce size of state space by explicitly resetting variables when they are not used!
- Automatically performed for clock variables (active clock reduction)

```

// Remove the front element of the queue
void dequeue()
{
    int i = 0;
    len -= 1;
    while (i < len)
    {
        list[i] = list[i + 1];
        i++;
    }
    list[i] = 0;
}

```

12-02-2008 Alexandre David, TOV'08 64

Clock Reduction (Automatic)

x is only **active** in location S1

Definition
x is **inactive** at S if on all path from S, x is always reset before being tested.

12-02-2008 Alexandre David, TOV'08 65

Synchronous Value Passing

	Unconditional		Conditional	
One-way	$c!$ var := out	$c?$ in := var, var := 0	$c!$ var := out	$c?$ in := var, var := 0 $cond(in)$
Asymmetric two-way	$c!$ var := out	$c?$ in := var	$c!$ var := out $cond1(var)$	$c?$ in := var, var := out $cond2(in)$

12-02-2008 Alexandre David, TOV'08 66

Atomicity

- Loops & complex control structures: C-functions.
- To allow encoding of multicasting.
- Committed locations.

12-02-2008 Alexandre David, TOV'08 67

Bounded Liveness

- Leads to within: $\phi \rightarrow_{\leq t} \psi$
 - More efficient than leadsto : $\phi \text{ leadsto}_{\leq t} \psi$ reduced to $A \Box (b \Rightarrow z \leq t)$ with
 - bool b set to true and clock z reset when ϕ holds.
 - When ψ holds set b to false.

12-02-2008 Alexandre David, TOV'08 68

Bounded Liveness

- The truth value of b indicates whether or not ψ should hold in the future.

$A[] (b \text{ imply } z \leq t)$
 $E \leftrightarrow b$ (for meaningful check)

$b \text{ true, check } z \leq t$

12-02-2008 Alexandre David, TOV'08 69

Zenoness

- Problem:** UPPAAL does not check for zenoness directly.
 - A model has "zeno" behavior if it can take an infinite amount of actions in finite time.
 - That is usually not a desirable behavior in practice.
 - Zeno models may wrongly conclude that some properties hold though they logically should not.
 - Rarely taken into account.
- Solution:** Add an observer automata and check for non-zenoness, i.e., that time will always pass.

12-02-2008 Alexandre David, TOV'08 71

Zenoness

Detect by adding the observer:

Constant (10) can be anything ($\neq 0$), but choose it well w.r.t. your model for efficiency. Clocks 'x' are local.

and check the property
 $\text{ZenoCheck.A} \dashrightarrow \text{ZenoCheck.B}$

12-02-2008 Alexandre David, TOV'08 72