Testing Distributed Systems through Symbolic Model Checking

G. Kalyon T. Massart C. Meuter L. Van Begin

Université Libre de Bruxelles
Département d'Informatique

FORTE'07

June 28, 2007

Need for Validation

Distributed control systems

- concurrent processes
- running on physically distributed hardware
- hard to design in nature
- critical systems (e.g. plant control system, ...)

Need for Validation

Distributed control systems

- concurrent processes
- running on physically distributed hardware
- hard to design in nature
- critical systems (e.g. plant control system, ...)

Validation Techniques

Model-Checking:

- exhaustive verification
- but: not (yet) scalable to real-sized systems
- works on a model of the system

Testing and Monitoring:

- non-exhaustive verification
- scalable for real-sized systems
- widely used in industry
- works on the real system

Testing

The System

- distributed and asynchronous
- instrumented to emit relevant events
  (e.g. variable assignments, message transfer)
- execution traces are collected

Testing

The System

- distributed and asynchronous
- instrumented to emit relevant events
  (e.g. variable assignments, message transfer)
- execution traces are collected

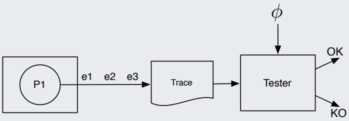
The Tester

- analyses the collected traces
- checks whether a certain property holds

Centralized v.s. Distributed Systems

Centralized System

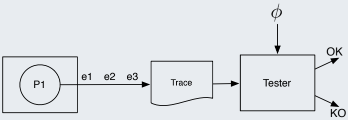
- only one process
- events are totally ordered



Centralized v.s. Distributed Systems

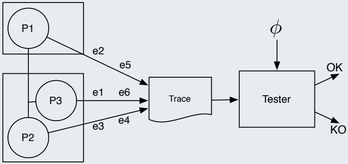
Centralized System

- only one process
- events are totally ordered



Distributed System

- multiple processes
- events are not totally ordered
- but a partial order can be obtained using vector clocks [Lam78, Mat89]



Partial Order Traces

Model

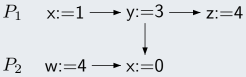
- partially ordered set of events
- events are labelled with assignments
- two restrictions:
  1. two events of the same process must be ordered
  2. two events assigning the same variable must be ordered

$$\begin{tikzcd} P_1 & x{:=}1 \arrow[r] & y{:=}3 \arrow[r] \arrow[d] & z{:=}4 \\ P_2 & w{:=}4 \arrow[r] & x{:=}0 & \end{tikzcd}$$

Partial Order Traces

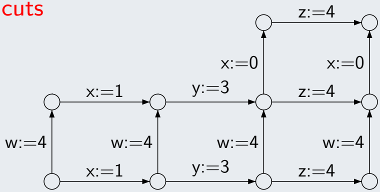
Model

- partially ordered set of events
- events are labelled with assignments
- two restrictions:
  1. two events of the same process must be ordered
  2. two events assigning the same variable must be ordered



Semantics

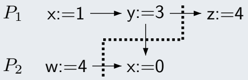
- a lattice of configurations called cuts



Partial Order Traces

Model

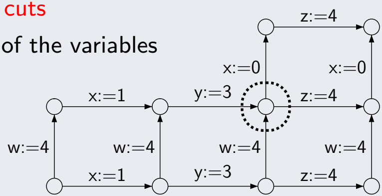
- partially ordered set of events
- events are labelled with assignments
- two restrictions:
  1. two events of the same process must be ordered
  2. two events assigning the same variable must be ordered



Semantics

- a lattice of configurations called cuts
- each cut has a unique valuation of the variables

$$v : \begin{cases} w & \mapsto & 4 \\ x & \mapsto & 1 \\ y & \mapsto & 3 \\ z & \mapsto & 0 \end{cases}$$



Global Predicate Detection

So far...

Efficient techniques have been developed for several classes of predicate:

- Stable predicates [CL85] (such that $\phi \implies \mathrm{AG}\phi$)
- Disjunctive predicates (of the form $\mathrm{EF}(p_1 \vee p_2 \vee \cdots \vee p_n)$)
- Conjunctive predicates [GW94, GW96] (of the form $\mathrm{EF}(p_1 \wedge p_2 \wedge \cdots \wedge p_n)$)
- Observer independent predicates [CDF95] (such that $\mathrm{AF}\phi \iff \mathrm{EF}\phi$)
- Linear, semi-linear predicates [CG98]
- Regular predicates (RCTL logic) [GM01, SG03]

$$\phi ::= \top \mid p \mid \neg p \mid \phi \wedge \phi \mid \mathrm{EF}\phi \mid \mathrm{EG}\phi \mid AG\phi \mid \mathrm{EX}[i]\phi$$

Global Predicate Detection

So far...

Efficient techniques have been developed for several classes of predicate:

- Stable predicates [CL85] (such that $\phi \implies \mathrm{AG}\phi$)
- Disjunctive predicates (of the form $\mathrm{EF}(p_1 \vee p_2 \vee \cdots \vee p_n)$)
- Conjunctive predicates [GW94, GW96] (of the form $\mathrm{EF}(p_1 \wedge p_2 \wedge \cdots \wedge p_n)$)
- Observer independent predicates [CDF95] (such that $\mathrm{AF}\phi \iff \mathrm{EF}\phi$)
- Linear, semi-linear predicates [CG98]
- Regular predicates (RCTL logic) [GM01, SG03]

$$\phi ::= \top \mid p \mid \neg p \mid \phi \wedge \phi \mid \mathrm{EF}\phi \mid \mathrm{EG}\phi \mid AG\phi \mid \mathrm{EX}[i]\phi$$

Our goal

Provide an efficient technique for full CTL (Computational Tree Logic):

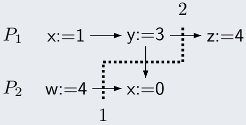
$$\phi ::= \top \mid p \mid \neg\phi \mid \phi \vee \phi \mid \phi \wedge \phi \mid \mathrm{EX}\phi \mid \mathrm{AX}\phi \mid \mathrm{EG}\phi \mid \mathrm{AG}\phi \mid \mathrm{E}[\phi \mathrm{U} \phi] \mid \mathrm{A}[\phi \mathrm{U} \phi]$$

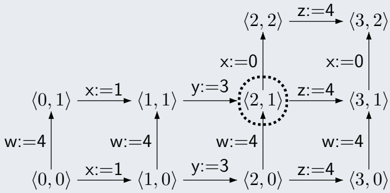
Symbolic Representation

Multirectangles

Cuts can be viewed as tuples of naturals:

- in a tuple $\langle x_1, \ldots, x_n \rangle$, $x_i =$ the number of events process i has executed





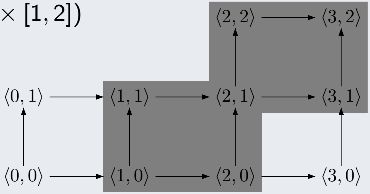
Symbolic Representation

Multirectangles

Cuts can be viewed as tuples of naturals:

- in a tuple $\langle x_1, \ldots, x_n \rangle$, $x_i =$ the number of events process i has executed
- sets of cut can be represented as a union of multirectangles

$$= ([1,2] \times [0,1]) \cup ([2,3] \times [1,2])$$



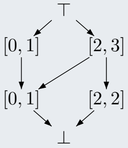
Symbolic Representation (cont'd)

Interval Sharing Tree (IST)

Symbolic data structure for representing sets of tuples:

- directed acyclic graph
- nodes are labelled with intervals
- each path from $\top$ to $\perp$ encodes a multirectangle

$$([0,1] \times [0,1]) \cup ([2,3] \times [0,1]) \cup ([2,3] \times [2,2])$$



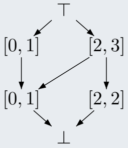
Symbolic Representation (cont'd)

Interval Sharing Tree (IST)

Symbolic data structure for representing sets of tuples:

- directed acyclic graph
- nodes are labelled with intervals
- each path from $\top$ to $\bot$ encodes a multirectangle

$$([0,1] \times [0,1]) \cup ([2,3] \times [0,1]) \cup ([2,3] \times [2,2])$$



Using IST for CTL

For each CTL formula ϕ, we build symbolically an IST $\mathcal{I}_\phi$ representing $[\![\phi]\!]$

Symbolic Computation

Tautology

If $\phi = \top$, we build $\mathcal{I}_\top$ iteratively:

$$
\begin{tikzcd}
\text{x:=1} \arrow[r] & \text{y:=3} \arrow[r] \arrow[d] & \text{z:=4} \\
\text{w:=4} \arrow[r] & \text{x:=0} &
\end{tikzcd}
$$

Symbolic Computation

Tautology

If $\phi = \top$, we build $\mathcal{I}_\top$ iteratively:

- start as if the trace had no communications

x:=1 $\longrightarrow$ y:=3 $\longrightarrow$ z:=4

w:=4 $\longrightarrow$ x:=0

$$\top \to [0,3] \to [0,2] \to \bot$$

$\mathcal{I}_0$

Symbolic Computation

Tautology

If $\phi = \top$, we build $\mathcal{I}_\top$ iteratively:

- start as if the trace had no communications
- treat communications one at a time

$$\begin{tikzcd}
x{:=}1 \arrow[r] & y{:=}3 \arrow[r] \arrow[d] & z{:=}4 \\
w{:=}4 \arrow[r] & x{:=}0 &
\end{tikzcd}$$

$$\underbrace{\begin{array}{c}\top\\ \downarrow\\ [0,3]\\ \downarrow\\ [0,2]\\ \downarrow\\ \bot\end{array}}_{\mathcal{I}_0} \setminus \left(\underbrace{\begin{array}{c}\top\\ \downarrow\\ [0,1]\\ \downarrow\\ [0,2]\\ \downarrow\\ \bot\end{array}}_{\mathcal{B}(\mathsf{y} := 3)} \bigcap \underbrace{\begin{array}{c}\top\\ \downarrow\\ [0,3]\\ \downarrow\\ [2,2]\\ \downarrow\\ \bot\end{array}}_{\mathcal{A}(\mathsf{x} := 0)} \right)$$

Symbolic Computation

Tautology

If $\phi = \top$, we build $\mathcal{I}_\top$ iteratively:

- start as if the trace had no communications
- treat communications one at a time
- stop when all communications have been treated

$$\begin{tikzcd} \text{x:=1} \arrow[r] & \text{y:=3} \arrow[r] \arrow[d] & \text{z:=4} \\ \text{w:=4} \arrow[r] & \text{x:=0} & \end{tikzcd}$$

$$\begin{tikzcd}[row sep=small] \top \arrow[d] \\ {[0,3]} \arrow[d] \\ {[0,2]} \arrow[d] \\ \bot \end{tikzcd} \setminus \left(\begin{tikzcd}[row sep=small] \top \arrow[d] \\ {[0,1]} \arrow[d] \\ {[0,2]} \arrow[d] \\ \bot \end{tikzcd} \cap \begin{tikzcd}[row sep=small] \top \arrow[d] \\ {[0,3]} \arrow[d] \\ {[2,2]} \arrow[d] \\ \bot \end{tikzcd} \right) = \begin{tikzcd}[row sep=small] & \top \arrow[dl] \arrow[dr] & \\ {[0,1]} \arrow[d] & & {[2,3]} \arrow[d] \arrow[dll] \\ {[0,1]} \arrow[dr] & & {[2,2]} \arrow[dl] \\ & \bot & \end{tikzcd}$$

$\mathcal{I}_0$ $\quad$ $\mathcal{B}(\text{y} := 3)$ $\quad$ $\mathcal{A}(\text{x} := 0)$ $\quad$ $\mathcal{I}_\top$

Symbolic Computation (cont'd)

Predicate

If $\phi = p \equiv (x \bullet c)$ with $\bullet \in \{=, \neq, <, \leq, >, \geq\}$, we build $\mathcal{I}_p$ as follows:

$$\begin{tikzcd}
\text{x:=1} \arrow[r] & \text{y:=3} \arrow[r] \arrow[d] & \text{z:=4} \\
\text{w:=4} \arrow[r] & \text{x:=0} &
\end{tikzcd}$$

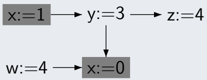
$$p(x \neq 0)$$

Symbolic Computation (cont'd)

Predicate

If $\phi = p \equiv (x \bullet c)$ with $\bullet \in \{=, \neq, <, \leq, >, \geq\}$, we build $\mathcal{I}_p$ as follows:

- we know that all events assigning x are totaly ordered



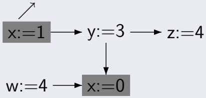
$p(x \neq 0)$

Symbolic Computation (cont'd)

Predicate

If $\phi = p \equiv (x \bullet c)$ with $\bullet \in \{=, \neq, <, \leq, >, \geq\}$, we build $\mathcal{I}_p$ as follows:

- we know that all events assigning x are totaly ordered
- some events sets p to true



$p(x \neq 0)$

$$\top \to [1,3] \to [0,2] \to \bot$$

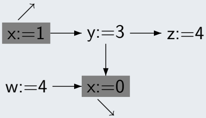
$\mathcal{A}(\mathrm{x} := 1)$

Symbolic Computation (cont'd)

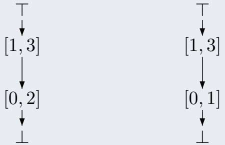
Predicate

If $\phi = p \equiv (x \bullet c)$ with $\bullet \in \{=, \neq, <, \leq, >, \geq\}$, we build $\mathcal{I}_p$ as follows:

- we know that all events assigning x are totaly ordered
- some events sets p to true
- some events sets p to false



$p \equiv (x \neq 0)$



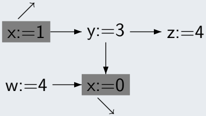
$\mathcal{A}(\text{x} := 1)$ $\mathcal{B}(\text{x} := 0)$

Symbolic Computation (cont'd)

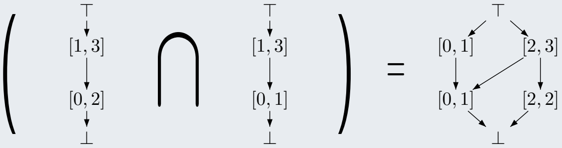
Predicate

If $\phi = p \equiv (x \bullet c)$ with $\bullet \in \{=, \neq, <, \leq, >, \geq\}$, we build $\mathcal{I}_p$ as follows:

- we know that all events assigning x are totaly ordered
- some events sets p to true
- some events sets p to false
- compute the union of all slices where:
  - p was set to true,
  - but not yet to false



$p \equiv (x \neq 0)$



$\mathcal{A}(x := 1)$ $\mathcal{B}(x := 0)$ $\mathcal{I}_p$

Symbolic Computation (cont'd)

Boolean Operators

We can use standard set operations on IST:

$$
\begin{aligned}
[\![\phi_1 \wedge \phi_2]\!] &= [\![\phi_1]\!] \cap [\![\phi_2]\!] \\
[\![\phi_1 \vee \phi_2]\!] &= [\![\phi_1]\!] \cup [\![\phi_2]\!] \\
[\![\neg\phi]\!] &= \overline{[\![\phi]\!]}
\end{aligned}
$$

ULB

Symbolic Computation (cont'd)

Boolean Operators

We can use standard set operations on IST:

$$\begin{array}{rcl} [\![\phi_1 \wedge \phi_2]\!] & = & [\![\phi_1]\!] \cap [\![\phi_2]\!] \\ [\![\phi_1 \vee \phi_2]\!] & = & [\![\phi_1]\!] \cup [\![\phi_2]\!] \\ [\![\neg\phi]\!] & = & \overline{[\![\phi]\!]} \end{array}$$

Existential Modalities

- symbolic computation for $\mathsf{EX}\phi$ using $[\![\mathsf{EX}\phi]\!] = \bigcup_{i\in[1,k]}[\![\mathsf{EX}[i]\phi]\!]$:

$$\mathcal{I}_{EX\phi} = \bigcup_{i\in[1,k]}(\mathcal{I}_{\phi}^{[x_i\leftarrow(x_i-1)]} \cap \mathcal{I}_{\top})$$

- for $\mathsf{EG}\phi$ and $\mathsf{E}[\phi_1 \mathsf{U} \phi_2]$, use classical fixed point:

$$\begin{array}{rcl} [\![\mathsf{EG}\phi]\!] & = & \textbf{gfp } \lambda\psi \cdot [\![\phi]\!] \cap [\![\mathsf{EX}\psi]\!] \\ [\![\mathsf{E}[\phi_1 \mathsf{U} \phi_2]]\!] & = & \textbf{lfp } \lambda\psi \cdot [\![\phi_2]\!] \cup ([\![\phi_1]\!] \cap [\![\mathsf{EX}\psi]\!]) \end{array}$$

- for $\mathsf{EF}\phi \stackrel{def}{=} \mathsf{E}[\top \mathsf{U} \phi]$, we can compute $\mathcal{I}_{\mathsf{EF}\phi} = \downarrow\mathcal{I}_{\phi} \cap \mathcal{I}_{\top}$

Symbolic Computation (cont'd)

Universal Modalities

- symbolic computation for AXϕ using $[\![\mathsf{AX}\phi]\!] = [\![\neg\mathsf{EX}\neg\phi]\!]$:

$$\mathcal{I}_{AX\phi} = \overline{\mathcal{I}_{\mathsf{EX}\neg\phi}}$$

- for AGϕ and A$[\phi_1 \mathsf{U} \phi_2]$, use classical fixed point:

$$\begin{array}{rcl} [\![\mathsf{AG}\phi]\!] & = & \mathbf{gfp}\ \lambda\psi \cdot [\![\phi]\!] \cap [\![\mathsf{AX}\psi]\!] \\ [\![\mathsf{A}[\phi_1 \mathsf{U} \phi_2]]\!] & = & \mathbf{lfp}\ \lambda\psi \cdot [\![\phi_2]\!] \cup ([\![\phi_1]\!] \cap [\![\mathsf{AX}\psi]\!]) \end{array}$$

- for AGϕ, since $[\![\mathsf{AG}\phi]\!] = [\![\neg\mathsf{EF}\neg\phi]\!]$, we can compute $\mathcal{I}_{\mathsf{AG}\phi} = \overline{\downarrow\mathcal{I}_{\neg\phi} \cap \mathcal{I}_{\top}}$

Experimental Results

Model	#proc	#events	IST (in sec.)	NuSMV (in sec.)
Peterson	2	2000	0.46	349.57
	2	5000	7.53	↑↑
	2	15000	189.65	↑↑
Peterson Generalized	2	2000	0.20	294.46
	2	5000	6.44	↑↑
	2	20000	390.90	↑↑
	5	1000	2.04	13.74
	5	1500	6.82	↑↑
	5	5000	176.62	↑↑
	10	1500	7.53	150.23
	10	2000	27.01	↑↑
	10	5000	147.89	↑↑

↑↑ indicates ($>$ 10 min.)

ULB

Experimental Results

Model	#proc	#events	IST (in sec.)	NuSMV (in sec.)
Alternating Bit Protocol	2	1000	13.60	297.28
	2	2000	27.56	↑↑
	2	5000	257.29	↑↑
Dining Philosophers	3	100	0.15	6.36
	3	200	1.11	↑↑
	3	2000	366.22	↑↑
	5	100	0.25	↑↑
	5	200	27.05	↑↑
	5	500	125.56	↑↑
	10	100	1.67	↑↑
	10	200	26.94	↑↑
	10	500	↑↑	↑↑

↑↑ indicates ($>$ 10 min.)

ULB

Conclusion

What has been done?

- we proposed a symbolic technique for testing full CTL properties on distributed systems
- has been implemented and works well in practice

Future Work

- integration of the methods in our tool TraX
- interface with our industrial design environment $_{\mathrm{d}}$SL [DMM03,DGMM05]
- investigate possible further improvements of our technique
- investigate similar models (MSC...)
- comparison with other symbolic data structures (IDD,...)

Conclusion

Questions?