Smart Reduction

Frédéric Lang

INRIA and LIG / VASY

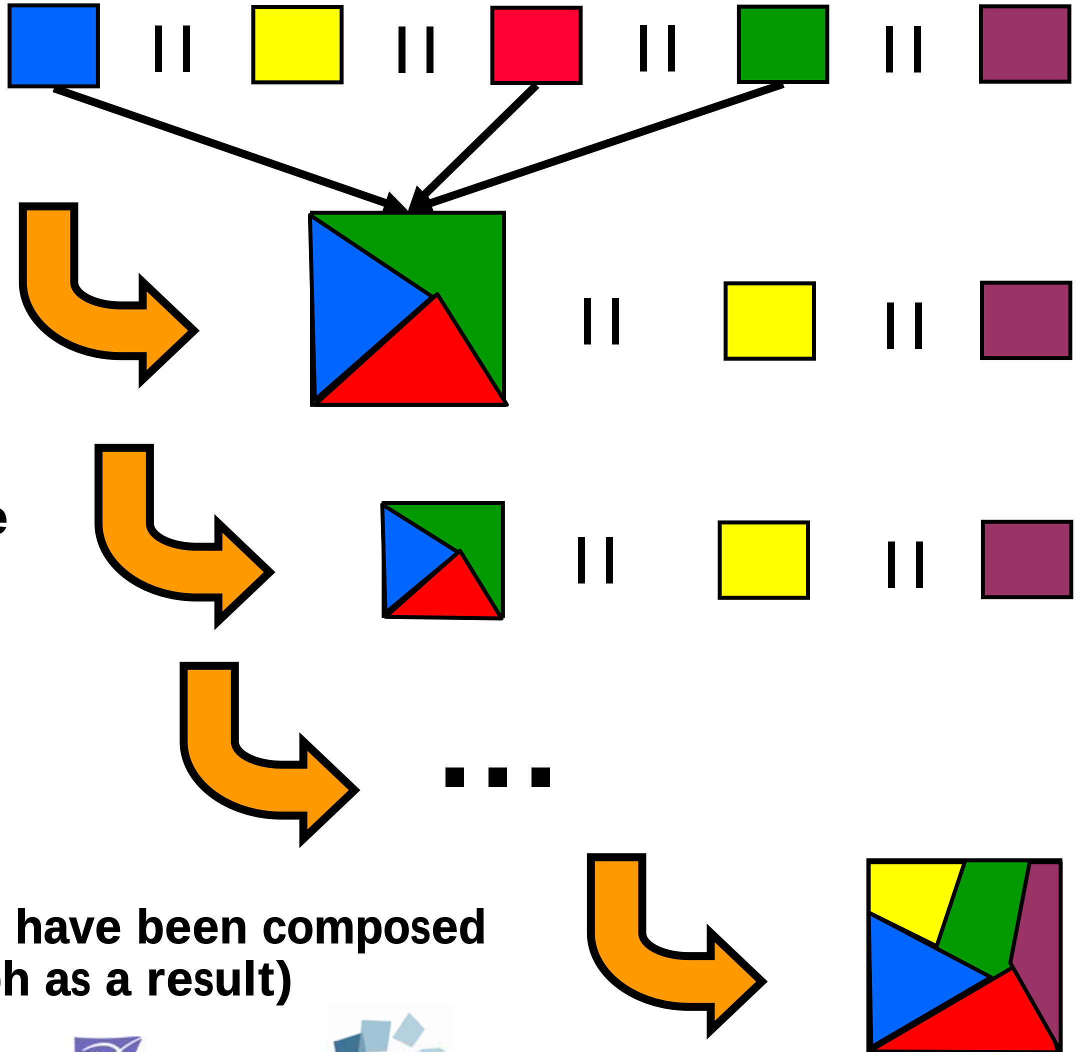
http://vasy.inria.fr

joint work with
Pepijn Crouzen (Saarland University)

Context

- **Explicit-state verification of concurrent systems**
  - Parallel composition of **asynchronous processes**
  - **Synchronisation** and/or **interleaving** of actions
  - Systematic exploration of the **behaviour graph**
- **Compositional verification**
  - Technique to palliate state explosion
  - Apply **property preserving abstractions** to the graphs of the composed processes (incremental)
  - In our case : **reductions modulo graph equivalences** (strong bisimulation, branching bisimulation, etc.)

Incremental reduction



Select a subset of the individual processes

Compose the subset (hide internal labels)

Reduce modulo equivalence (congruence wrt parallel composition)

Continue until all processes have been composed (obtaining the minimal graph as a result)

Process subset selection

- A careful selection may avoid **intermediate graph explosion**
- Possibly many individual processes: **automated selection** is needed
- Use **metrics**
  - **Tai & Koppol 1993**
    - **C**omposition of *Finite State Machines*
    - Deterministic binary synchronisation (CCS-like)
  - **Crouzen & Hermanns 2010**
    - Refine the metrics, apply to performance models
    - Deterministic multiway synchronisation (CSP-like)

Process subset composition

- Easy with **binary associative** and **commutative parallel composition operators** (previous work):

$$I \subseteq J \Rightarrow ||_{j \in J} P_j = (||_{i \in I} P_i) \; || \; (||_{j \in J \setminus I} P_j)$$

- But… parallel composition operators are **not necessarily associative…**

Example (LOTOS): $P_1 \; |[a]| \; (P_2 \; ||| \; P_3) \neq (P_1 \; |[a]| \; P_2) \; ||| \; P_3$

- … and **not even binary**

Example (E-LOTOS): **par** *a*#2, *a*#3 in $P_1 \; || \; P_2 \; || \; P_3$ **end par**

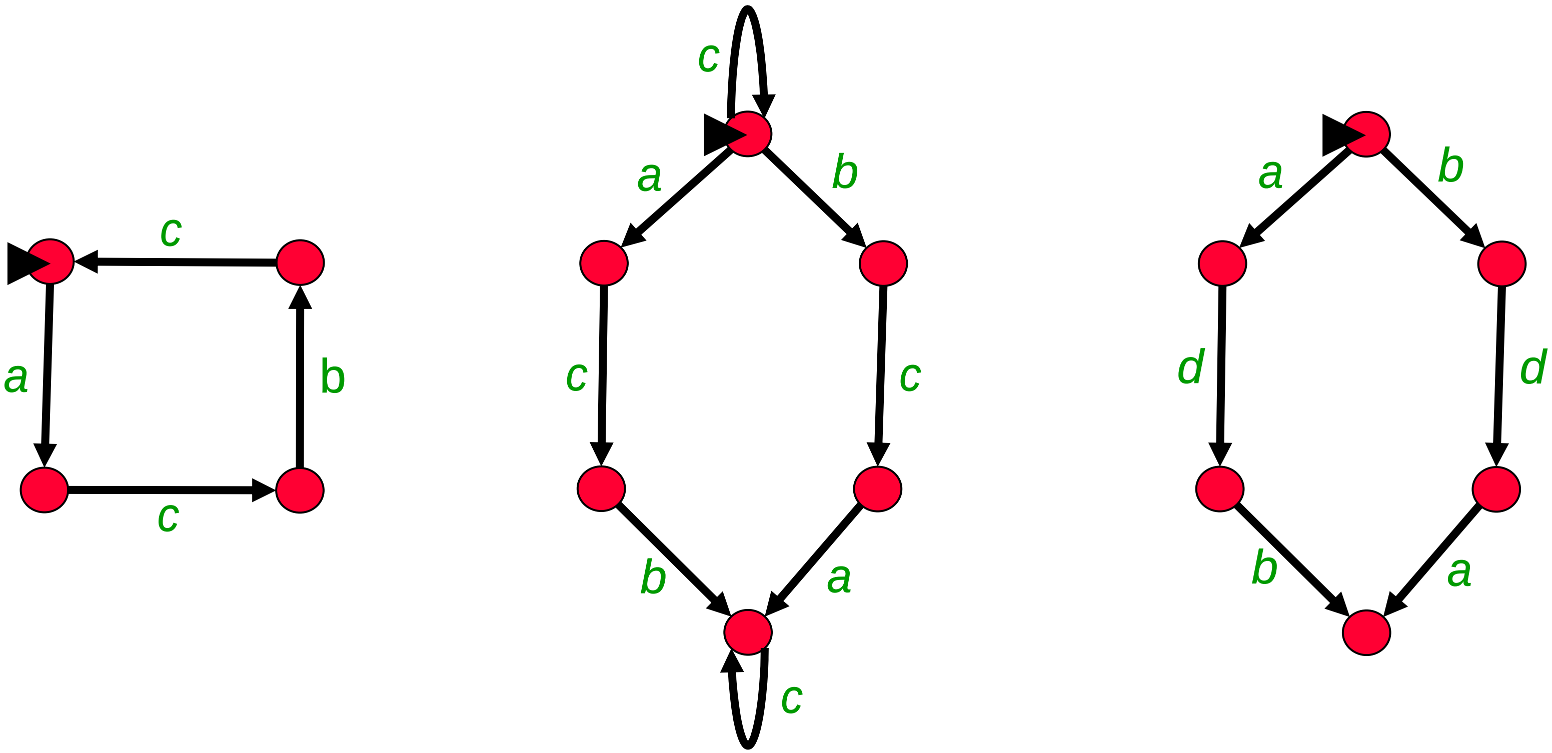
- Synchronization may be both **nondeterministic** and **multiway** (useful to write more concise models)

Contributions

- **Extension** of incremental reduction to *Networks of Labelled Transition Systems*, a composition model that subsumes most forms of synchronizations (including nondeterministic multiway synchronization)
- **Refinement** of the **selection metrics**
- **Implementation** in the CADP toolbox: the **smart reduction** operator
- **Experiments** on various case-studies

Individual processes

- Represented as LTS (*Labelled Transition Systems*)
- **Example**



P_1 P_2 P_3

Network of LTS

- **Inspired by MEC and FC2**
- **Subsume many composition models**

  CCS, CSP, LOTOS, E-LOTOS/LOTOS NT, mCRL, synch. vectors, …
- Of the form $((P_1, \ldots, P_n), V)$ where:
  - $P_1, \ldots, P_n$ are LTS (**individual processes**)
  - V is a set of **synchronization rules**
- Each rule has the form $(a_1, \ldots, a_n) \rightarrow b$ where:
  - a_i is either **a label** or **the special symbol •**
  - b is **a label**

Semantics of networks

- **Operational**: An LTS
  - **State:** vector $(s_1, \ldots, s_n)$ of local states s_i
  - **Transition:** $(s_1, \ldots, s_n) \xrightarrow{b} (s'_1, \ldots, s'_n)$ iff:
    - V has a synchronization rule $(a_1, \ldots, a_n) \rightarrow b$, and
    - $s_i \xrightarrow{a_i} s'_i$ (foreach $a_i \neq \bullet$), and
    - $s_i = s'_i$ (foreach $a_i = \bullet$)
- **Static**: **internal action** τ cannot be cut, renamed, or synchronised (easy syntactic check of synch. rules that contain τ)

Example (1/2)

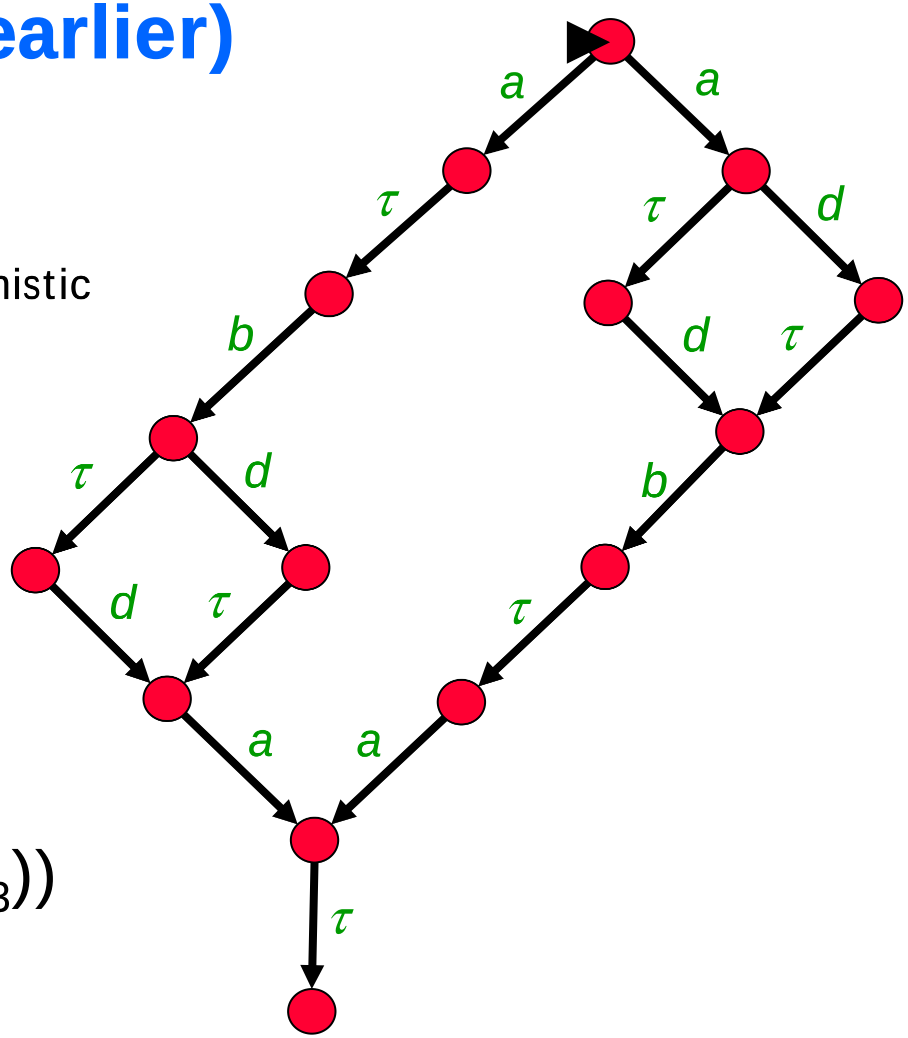
- **P_1, P_2, and P_3 (defined earlier) synchronised following:**

$(a, a, \bullet) \to a$ $(a, \bullet, a) \to a$	nondeterministic
$(b, b, b) \to b$	multiway
$(c, c, \bullet) \to \tau$	hidden
$(\bullet, \bullet, d) \to d$	interleaving

- **Same LTS as LOTOS**

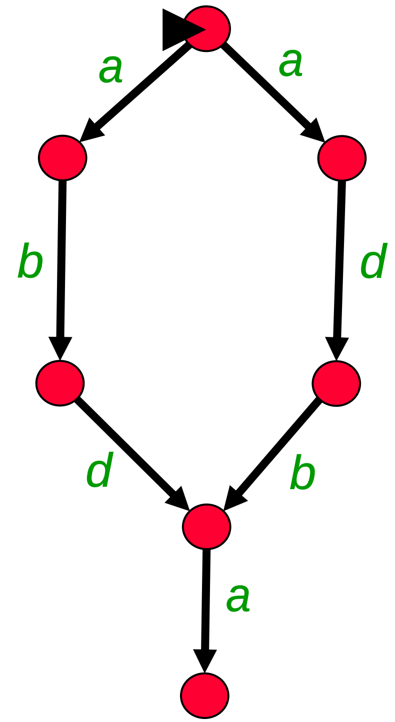
hide c **in**

$(P_1 \;|[a, b, c]|\; (P_2 \;|[b]|\; P_3))$



Example (2/2)

- The LTS of the network has **15 states and 17 transitions**
- The LTS reduced modulo branching bisimulation has **7 states and 7 transitions**



Incremental reduction of networks

- **Let N be a network**
- **Each reduction step requires four operations:**
  - **Select** a subset **I** of the individual LTS of N
  - **Extract** a new network $\mathbf{N_p(I)}$ modeling the **composition of the LTS inside I**
  - Compute the reduced LTS $\mathbf{P_p}$ of $\mathbf{N_p(I)}$
  - **Extract** a network $\mathbf{N_a(I)}$ modeling the **composition of** $\mathbf{P_p}$ **with the LTS outside I**

Extraction of $N_p(I)$

- **Projection of N on to the LTS inside I**
- Use **intermediate labels** (x_1, x_2, …) for rules that synchronize LTS both inside and outside I
- **Example**: I = {P_1, P_2}

extract			
	$(a, a) \to a$	from	$(a, a, \bullet) \to a$
	$(a, \bullet) \to x_1$	from	$(a, \bullet, a) \to a$
	$(b, b) \to x_2$	from	$(b, b, b) \to b$
	$(c, c) \to \tau$	from	$(c, c, \bullet) \to \tau$
	no rule	from	$(\bullet, \bullet, d) \to d$

Extraction of $N_a(I)$

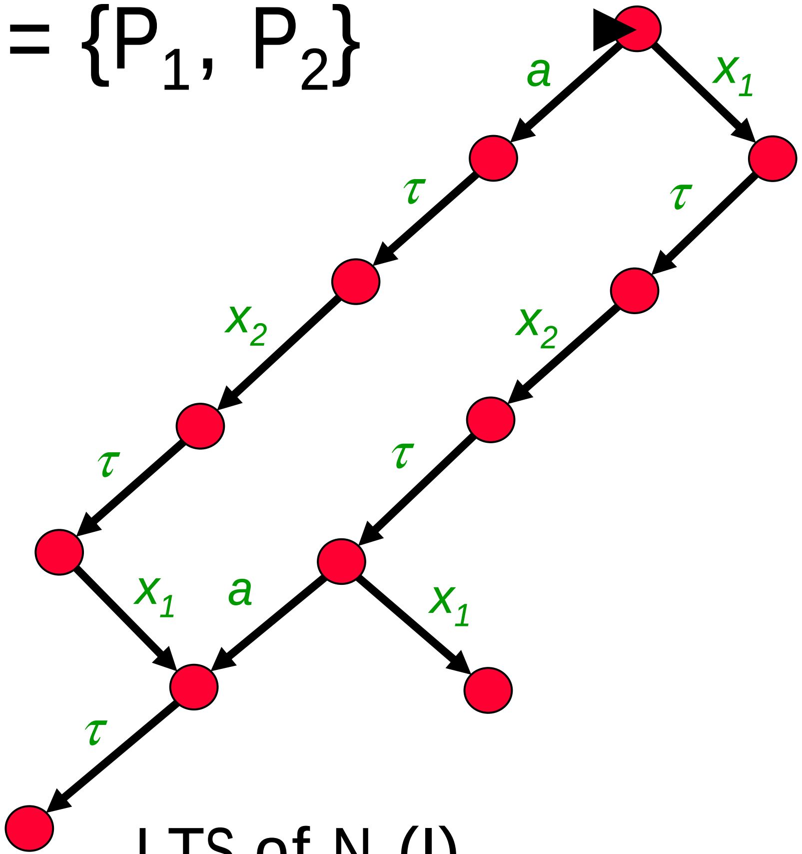
- **Composition of the (reduced) LTS of $N_p(I)$ with the LTS outside I**
- Use the same **intermediate labels** as before
- **Example**: $I = \{P_1, P_2\}$

extract

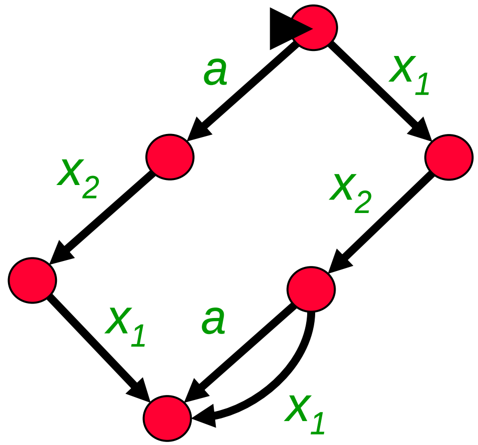
$(a, \bullet) \to a$	from	$(a, a, \bullet) \to a$
$(x_1, a) \to a$	from	$(a, \bullet, a) \to a$
$(x_2, b) \to b$	from	$(b, b, b) \to b$
$(\tau, \bullet) \to \tau$	from	$(c, c, \bullet) \to \tau$
$(\bullet, d) \to d$	from	$(\bullet, \bullet, d) \to d$

Example (1/2)

$I = \{P_1, P_2\}$



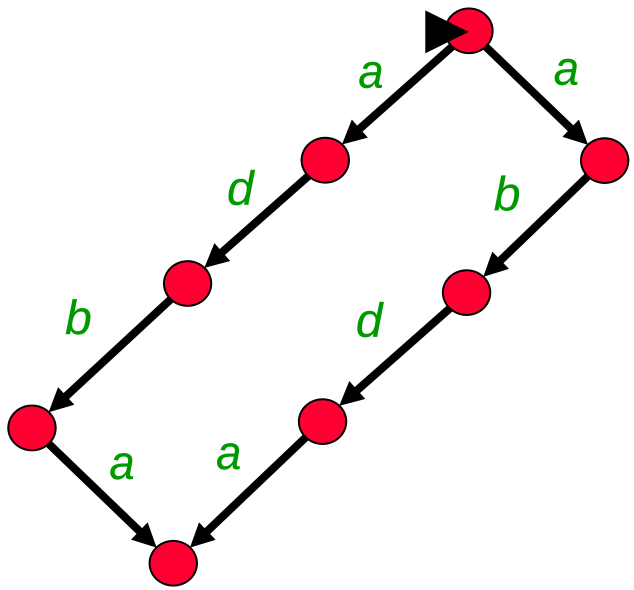
LTS of $N_p(I)$
(composition of P1 and P2)
12 states, 12 transitions



Reduced modulo branching
6 states, 7 transitions

Example (2/2)

$I = \{P_1, P_2\}$



LTS corresponding to $N_a(I)$
(composition of reduced LTS in previous slide with P_3)
8 states, 8 transitions

Largest intermediate LTS now: **12 states, 12 transitions** instead of **15 states, 17 transitions**

Remarks (1/2)

- **LTS selection** is crucial to reduce the largest intermediate LTS size

I	Largest intermediate LTS
$\{P_1, P_2\}$	**12 states, 12 transitions**
$\{P_1, P_3\}$	**19 states, 23 transition**
$\{P_2, P_3\}$	**18 states, 34 transitions**
$\{P_1, P_2, P_3\}$	**15 states, 17 transitions**

- In this example $\{P_1, P_2\}$ is the **optimal selection**

Remarks (2/2)

- **Composition of P_1 and P_2 (optimal selection) is not expressible in LOTOS**
  - No subterm in **(**P_1 |[*a*, *b*, *c*]| (P_2 |[*b*]| P_3)) with P_1 and P_2 only
  - Associating terms otherwise changes semantics, e.g., **((**P_1 |[*a*, *b*, *c*]| P_2) |[*b*]| P_3)
- **Networks enable any subset of LTS to be composed**

LTS selection

- **Apply a metric on I to:**
  - **Maximize the amount of transitions** in $N_p(I)$ that are **hidden** (likely to disappear by reduction)
  - **Minimize the amount of transitions** in the individual LTS of $N_p(I)$ that **interleave**
- **Use a worst case estimate of transition numbers**
  - count all transitions, **reachable** and **unreachable**
  - Doing better requires **reachability analysis**: the problem we are trying to solve

The metric for networks

- Use a **combined metric** CM(I) =

$$\frac{HR(I) + 1 - IR(I)}{|I|}$$

size of I, to favor smaller compositions

- **Hiding Rate** HR(I)

$$\frac{\text{\# transitions of } N_p(I) \text{ hidden from LTS outside I}}{\text{\# transitions of } N_p(I)}$$

- **Interleaving Rate** IR(I)

$$\frac{\text{\# transitions of } N_p(I)}{\text{\# transitions in nonsynchronized product of LTS inside I}}$$

Accounting for interleaving

- **Full interleaving**

  A local transition that is **not synchronized**:
  $(a_1, \ldots, a_n) \rightarrow b$ where exactly one a_i is a label (the rest are $\bullet$)

- **Partial interleaving**

  Synchronization does not involve all LTS:
  $(a_1, \ldots, a_n) \rightarrow b$ where at least one a_i is $\bullet$

- **Previous metrics account for full interleaving only**
- **Refinement:** Our interleaving metric also accounts for partial interleaving

Example

I	$HR(I)\ /\ \|I\|$	$(1 - IR(I))\ /\ \|I\|$	$CM(I)$
$\{P_1, P_2\}$	0,211	0,357	0,568
$\{P_1, P_3\}$	0,000	0,227	0,227
$\{P_2, P_3\}$	0,000	0,124	0,124
$\{P_1, P_2, P_3\}$	0,129	0,249	0,378

$$CM(\{P_1, P_2\}) > CM(\{P_1, P_2, P_3\}) > CM(\{P_1, P_3\}) > CM(\{P_2, P_3\})$$

12 trans. < 17 trans. < 23 trans. < 34 trans.

Implementation in CADP (1/2)

- **CADP: A toolbox for analyzing asynchronous systems using formal methods**
  - Specification, Explicit-state verification, simulation, …
  - Contains more than **45 tools** and **22 code libraries**
- **Support for compositional verification**:
  - **Bcg_Min 2.0** tool for **LTS reduction**
  - **Exp.Open 2.0** tool for **networks of LTS** represented using numerous operators
  - **SVL** language and compiler for **scripting verification scenarios**
  - **…**

Implementation in CADP (2/2)

- **New operator smart reduction in SVL**

  Example

```
% DEFAULT_LOTOS_FILE="proc.lotos"
% DEFAULT_SMART_LIMIT=3
"p123.bcg" = smart branching reduction of
   hide c in (P1 |[a, b, c]| (P2 |[b]| P3));
```

- **SVL delegates work to EXP.OPEN 2.0**
  - Metric computation and process selection
  - Network extractions

Experimental results (1/2)

- **Smart branching reduction** was compared on 28 examples with
  - **node branching reduction**: compose LTS pairwise (syntactic order)
  - **root leaf branching reduction**: compose all LTS at once
- **Example**: Dynamic Fault Tree
  - **635,235 transitions** using **node reduction**
  - **117,772 transitions** using **root leaf reduction**
  - **346 transitions** using **smart reduction**

Experimental results (2/2)

- Of course, **smart reduction** does not always make the optimal LTS selection
  - There are **unreachable transitions** (how much?)
  - Branching reduction does not remove all internal transitions
  - Hiding and interleaving **rates may cancel each other out**
  - Compositional reduction may prevent **partial order reductions** otherwise possible using **root leaf reduction**
  - …
- **Combining the hiding and interleaving rates generally prevents bad selections**

Conclusions

- **A fully automated verification technique**
  - Appropriate to **models with many processes**
  - Saves time in the verification task
- **Refines previous work**
  - Network model **subsumes many composition models and elaborate forms of synchronisation**
  - Network model **enables any subset of processes to be selected**
  - Metric **accounts better for process interleaving**
- **Implementation available in CADP**

http://vasy.inria.fr/cadp

Want to know more about CADP?

Attend Hubert Garavel's

TACAS presentation

CADP 2010: a toolbox for the construction and analysis of distributed processes

Thursday, March 31 at 12:00