

# Coverage-biased random exploration of large models



Marie-Claude Gaudel

*LRI, Université de Paris-Sud*

And the RASTA group: Alain Denise, Johan Oudinet, Sandrine-Dominique Gouraud, Richard Lasseigne, Sylvain Peyronnet

*For those who read the ETAPS programme*

*please note: I left Nancy in 1973, I left INRIA in 1982!*

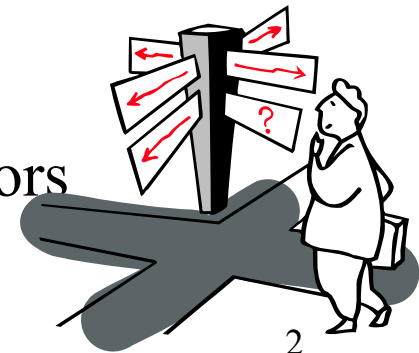
March 30, 2008

MBT'08



# Classical Random Walks

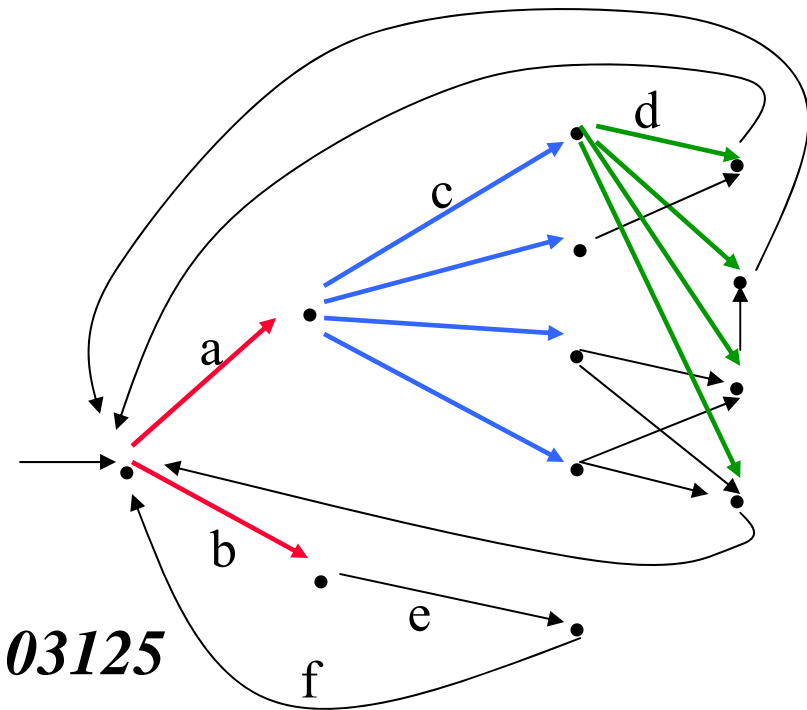
- ◆ A random walk in the state space of a model (in a control graph, etc) is
  - a sequence of states  $s_0, s_1, \dots, s_n$  such that  $s_i$  is chosen uniformly at random among the successors of the state  $s_{i-1}$ ,
- ◆ It is easy to implement and it only requires local knowledge of the graph.
- ◆ Numerous applications in
  - *Testing* (protocols [Mihail et Papadimitriou 1994], etc)
  - *Simulation* (Monte-Carlo, etc)
  - *Model-checking* ([Grosu 2004])
- ◆ In some variants, the choice among the successors may be biased.





# Drawback of classical random walks

The resulting coverage is dependent on the topology...



Classical random walks, length 3:

$$Pr(a; c; d) = 0.5 \times 0.25 \times 0.25 = 0.03125$$

$$Pr(b; e; f) = 0.5$$

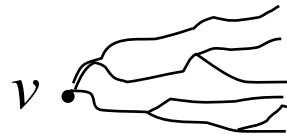
Uniform random sampling of traces, length 3:

$$Pr(a; c; d) = Pr(b; e; f) = 0.1$$

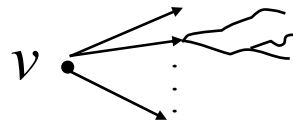


# Uniform generation of paths of a given length in a graph

Given any vertex  $v$ , let  $l_v(k)$  be *the number of paths of length  $k$  that start from  $v$*



- ◆ we are on vertex  $v$  with  $m$  successors  $v_1, v_2, \dots, v_m$



- ◆ condition for path uniformity: choose  $v_i$  with probability  $l_{v_i}(k-1)/l_v(k)$



# Counting labelled combinatorial structures

- ◆ Decomposable combinatorial structures
  - Atoms
  - Constructions : +,  $\times$ , SEQ, PSET, MSET, CYC
  - Cardinality constraints:  $\text{SEQ}_{\leq 3}$ , etc
- ◆ Let a combinatorial class  $C$ , which is decomposable, Explicit formula for  $|C_n| \Rightarrow$  Uniform drawing in  $C_n$ 
  - Enumeration based on generating functions
- ◆ Theorem [Mainly, Flajolet & al., 1994] :  
*The counts  $\{C_j | j= 0, \dots, n\}$  can be computed in  $O(n^{1+\varepsilon})$  arithmetic operations.*  
*Drawing UAR an element of size  $n$  is  $O(n \cdot \log n)$  in the worst case (and  $O(n)$  in our simple case)*



# Uniform generation of paths of length $n$

Given  $n$

- $l_v(0) = 1$

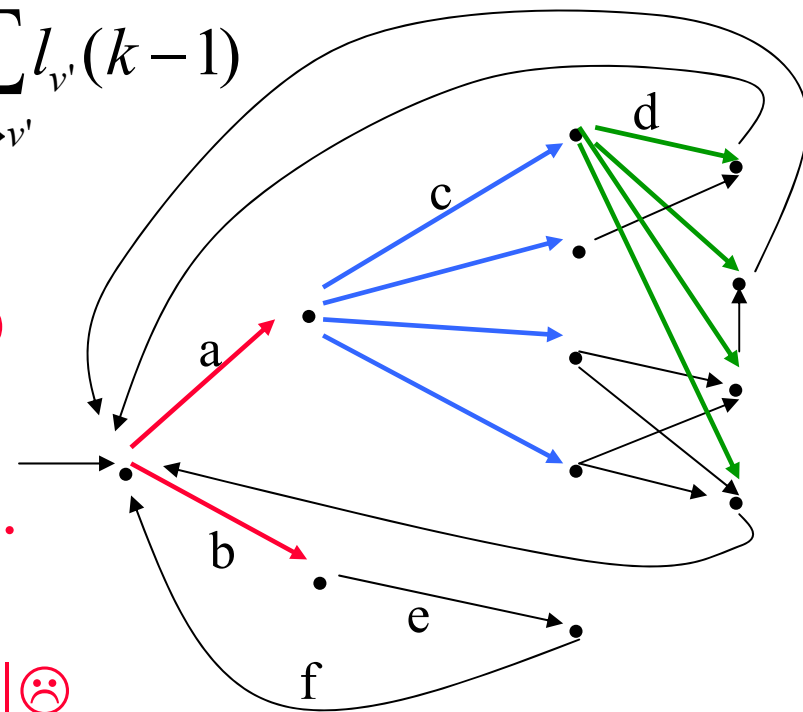
- for  $k$  s.t.  $n \geq k > 0$ ,  $l_v(k) = \sum_{v \rightarrow v'} l_{v'}(k-1)$

- *Preprocessing:*

Computation of the  $l_v(k)$ ,  
for all  $k \leq n$  is **linear in  $n$**  ☺

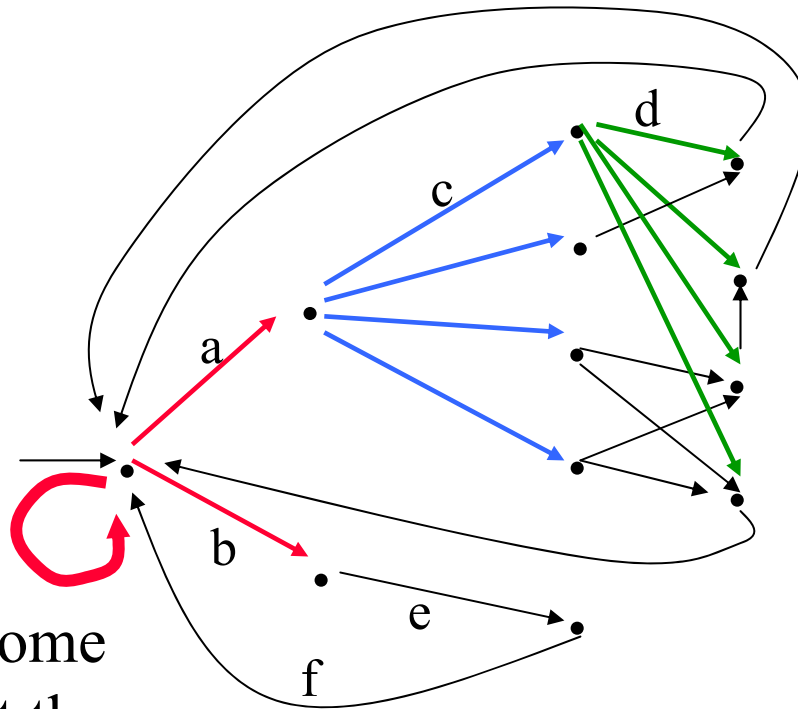
- *Drawing is linear in  $n$  too.*

- *Space requirement:  $n \times |G|$*  ☹





# Uniform generation of paths of length $\leq n$



Addition of some  
fictive loop at the  
initial state



# Main results of the talk

- ◆ **Experimental assessment of this method and of its limits (up to  $\pm 8000$  states)**
- ◆ **Uniform generation of paths in very large models:**
  - Given a model described as a set of concurrent LTS/FSM/IOTS it is possible to get (a very good approximation of) **global uniform random walks**,
    - without building the global model
    - using **local uniform drawing** of paths in the component LTS
- ◆ **Combination of weaker coverage criteria (transitions, states, ...) and random walks**
  - Coverage-biased random walks, “randomised coverage satisfaction”



# Experimentation

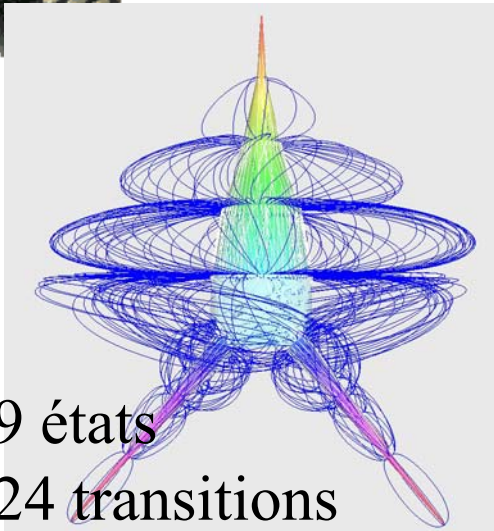
- ◆ Models are from the VLTS benchmark suite: <http://tinyurl.com/yuroxx>
- ◆ Intel Xeon 2.80GHz processor with 1GB memory, only :-)
- ◆ Elapsed time for generating 100 paths:

<i>Lengths/ #states</i>	<i>200</i>	<i>1000</i>	<i>2000</i>	<i>3000</i>	<i>5000</i>	<i>8000</i>
289	0.0s	0.9s	2.9s	6.3s	15.9s	40.1s
1183	0.1s	1.0s	3.2s	6.7s	18.2s	💣
5486	0.0s	0.9s	2.4s	5.2s	💣	💣
8879	0.2s	0.8s	2.4s	💣	💣	💣
$10^4$	0.0s	1.3s	💣	💣	💣	💣
$10^7$	💣	💣	💣	💣	💣	💣

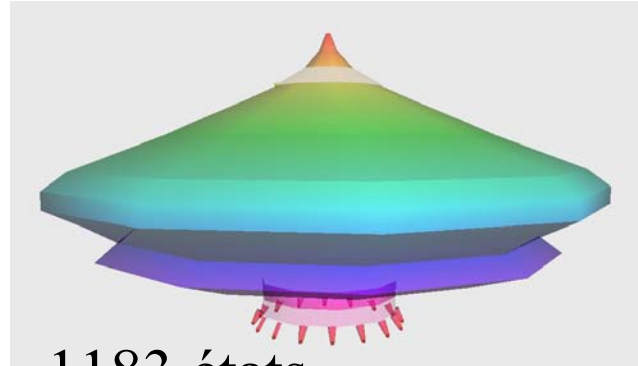
**Memory overflow**



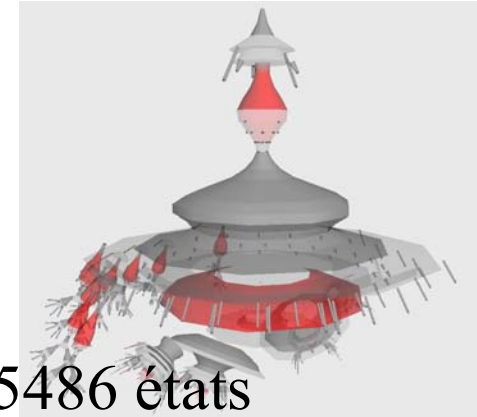
## VLTS : « Very large transition Systems »



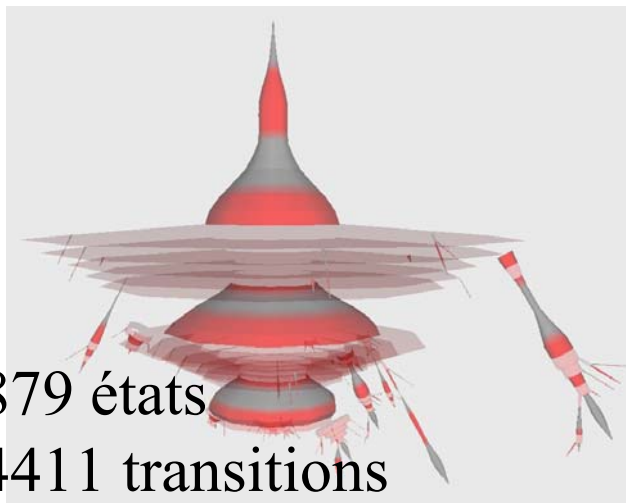
289 états  
1224 transitions



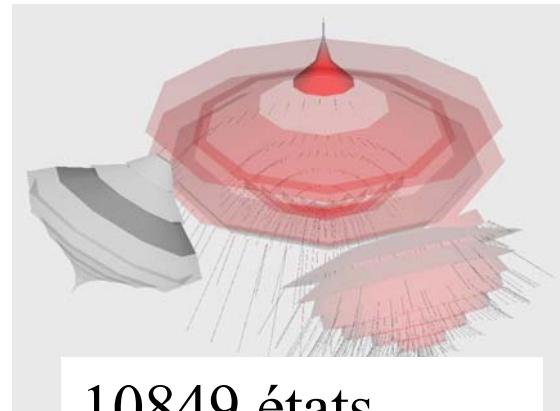
1183 états  
4464 transitions



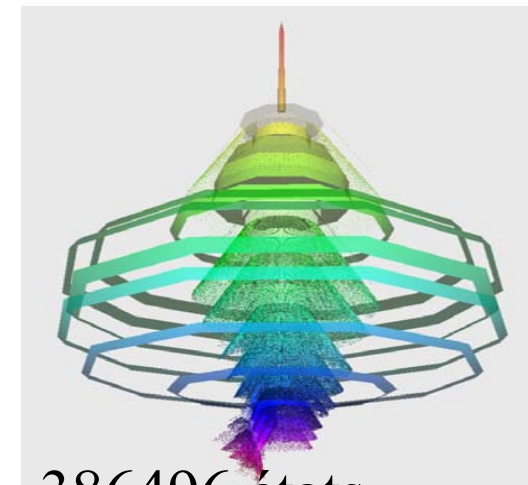
5486 états  
9676 transitions



8879 états  
24411 transitions



10849 états  
56156 transitions



386496 états<sub>10</sub>  
1172872 transitions

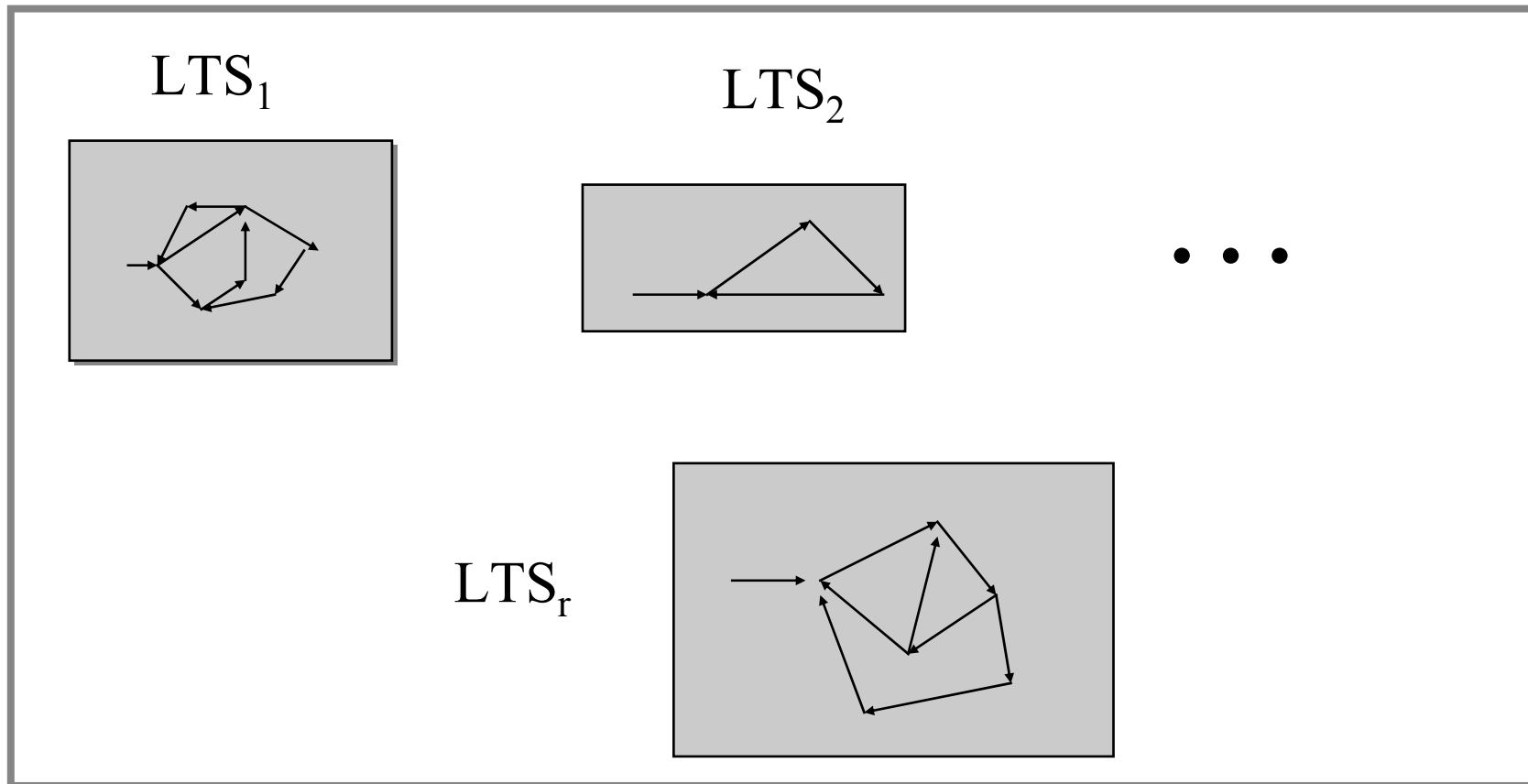


# What about huge models?

- ◆ Computing the  $l_v(k)$  for all vertex  $v$  and all  $k \leq n$  is no more possible
- ◆ Fortunately, huge models are obtained by *composition* of smaller ones
- ◆ Idea: use uniform random walks in the *components* to perform (approximately) uniform random walks in the *global system*
- ◆ Kind of considered composition: *concurrency*

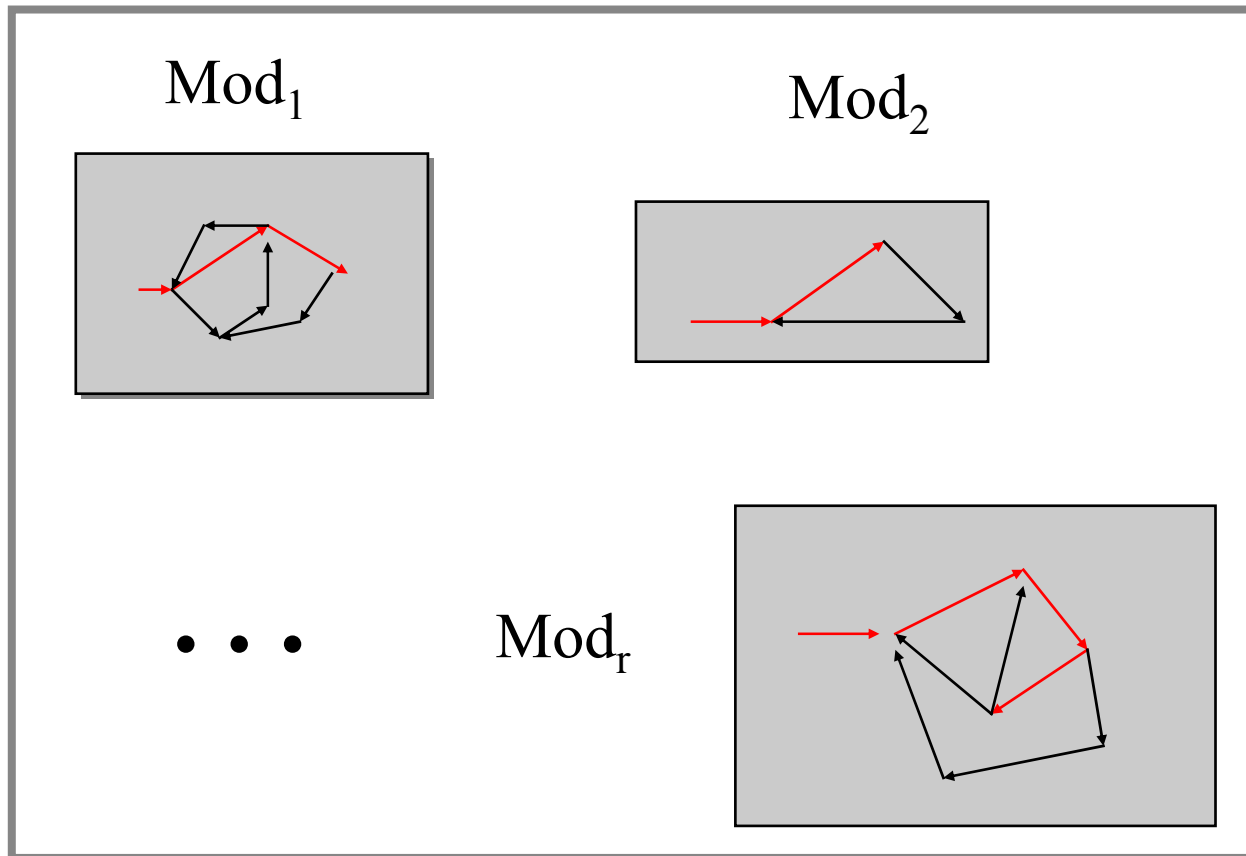


# 1- Composition of $r$ modules with no synchronisation





# Global path : interleaving of $r$ local paths



Local paths:

$$e_1^1 e_2^1 e_3^1$$

$$e_1^2 e_2^2$$

$$e_1^3 e_2^3 e_3^3 e_4^3$$

Possible  
interleavings:

$$e_1^1 e_1^2 e_1^3 e_2^1 e_2^2 e_2^3 e_3^1 e_3^3 e_4^3$$

$$e_1^2 e_2^2 e_1^3 e_1^1 e_2^1 e_2^3 e_3^3 e_3^1 e_4^3$$

$$e_1^3 e_1^1 e_2^3 e_2^1 e_1^2 e_3^3 e_2^2 e_4^3 e_3^1$$

...



# Composition of $r$ modules (cont.)

- ◆ Product of the  $r$  underlying automata  $A_i$  with labels in  $X_i$ 
  - States: *cartesian product* of the states of the  $A_i$
  - Language (traces): *shuffles (interleavings)* of the words of the  $L_i \subseteq X_i^*$  (languages recognised by the  $A_i$ )
- ◆ **(1) Brute force method for uniform trace generation:**
  - Construction of the global automaton:  $O(\prod |A_i| \cdot \sum |X_i|)$
  - Preprocessing and generation of traces of length  $n$ :  
 $O(n \cdot \prod |A_i| \cdot \sum |X_i|)$
  - Intractable when  $r$  and/or some  $|A_i|$  or  $|X_i|$  are big (**NB:** *tractable on moderate sizes: it works with  $\pm 8000$  states*)



## (2) Uniform generation of global traces without global model

- ◆ *Better than brute force (we hope)*
- ◆  $n$  is the length of the global traces
- ◆ Choose some lengths  $(n_1, \dots, n_r)$  with  $\sum n_i = n$  with adequate probabilities (see next slide)
- ◆ For each  $L_i$ , draw uniformly at random some word  $w_i$  of length  $n_i$
- ◆ Shuffle the  $r$   $w_i$  (in a way that ensures uniformity of shuffles)



# Uniform generation of global traces without global model

1. Given  $n$ , **choose the lengths**  $(n_1, \dots, n_r)$  of  $r$  traces in  $L_i$  with

◆  $\sum n_i = n$  and

$$\Pr(n_1, \dots, n_r) = \frac{\binom{n}{n_1, \dots, n_r} l_1(n_1) \dots l_r(n_r)}{l(n)}$$

# shuffles of length  $n$  with  $n_1, \dots, n_r$

# traces of length  $n$

2. For each  $L_i$ , **draw uniformly at random some**  $w_i$  of length  $n_i$
  3. **Shuffle the**  $r$   $w_i$  (with a classical algorithm that ensures uniformity of shuffles, see next slide)
- ◆ **A key point:** no need to compute  $l(n)$  to get a very good approximation of uniformity ☺



# Uniform word shuffling

## Shuffling $r$ words

**Input:**  $r$  words  $w_1, \dots, w_r$ , of length  $n_1, \dots, n_r$

**Output:** word  $w$  of length  $n = \sum_i n_i$  drawn uniformly among the shuffles of  $w_1, \dots, w_r$

$w \leftarrow \varepsilon$ ;  $n \leftarrow \sum_i n_i$ ;

**while**  $n > 0$  **do**

- choose an integer  $i$  in  $[1, r]$  with probability  $n_i / n$
- add the first letter of  $w_i$  at the end of  $w$
- remove the first letter of  $w_i$
- $n_i \leftarrow n_i - 1$ ;  $n \leftarrow n - 1$

**done**



# Approximation of the $l_i(n_i)$ and $l(n)$

- ◆ Aperiodicity and strong connectivity of the automata  $\Rightarrow$

$$l_i(n_i) \sim C_i \cdot \omega_i^{n_i}$$

where

- $C_i$  and  $\omega_i$  can be computed in polynomial time w.r.t.  $|A_i|$
  - $C_i \cdot \omega_i^{n_i} / l_i(n_i)$  converges to 1 at an exponential rate
  - See [Flajolet et Sedgewick INRIA RR4103, 2001]
- ◆ Then  $l(n) \sim C_1 \dots C_r \cdot (\omega_1 + \dots + \omega_r)^n$ 
    - It is an easy, technical, computation (see paper)



# Application to the choice of the $n_i$

Now, we have 
$$\Pr(n_1, \dots, n_r) \sim \frac{\binom{n}{n_1, \dots, n_r} \omega_1^{n_1} \dots \omega_r^{n_r}}{(\omega_1 + \dots + \omega_r)^n}$$

It's better, ... and moreover, *there is no need to compute it:*

- Take the set of integers  $\{1 \dots r\}$
- Draw  $n$  numbers in this set with

$$\Pr(i) = \omega_i / (\omega_1 + \dots + \omega_r)$$

- $n_i$  is the number of occurrences of  $i$  in the result

*linear in  $n$*



# Summary of the complexity

- ◆ No need to build the product of automata
- ◆ Drawing the local lengths  $n_i$  is linear in  $n$
- ◆ *Preprocessing for the drawings above* : the computation of the  $\omega_i$  is polynomial /  $|A_i|$  and  $|X_i|$
- ◆ Drawing words  $w_i$  in the  $L_i$  is linear in  $n_i$
- ◆ The corresponding *preprocessings* deal with the  $A_i$ : linear in  $Card(Q_i)$  and  $Card(X_i)$
- ◆ Drawing in the shuffle of the  $w_i$  is  $\Theta(n)$



## Elapsed Time to generate 100 paths

with/without building the global system

length	200	1000	2000	3000	5000	8000
# states	200	1000	2000	3000	5000	8000
289	0.0s	0.9s	2.9s	6.3s	15.9s	40.1s
8879	0.2s	0.8s	2.4s	X	X	X
$10^4$	0.0s 0.4s	1.3s 0.9s	X 2.4s	X 4.3s	X 10.2s	X 23.5s
$10^9$	0.0s	1.1s	1.6s	2.8s	6.2s	13.8s
$10^{12}$	0.6s	0.2s	1.9s	1.8s	5.5s	12.3s
$10^{17}$	0.6s	0.0s	2.9s	2.7s	5.4s	10.7s
$10^{24}$	0.3s	0.3s	1.3s	2.8s	4.6s	9.7s

- The models  $10^4, \dots, 10^{24}$  are obtained by combining several copies of the model 289,
- drawing paths in models larger than  $10^{24}$  states is possible.



## Another benchmark with different components

With two components (289 and 1183 states):

- system size  $\sim$  3 millions,
- time to generate 100 paths:

200	1000	2000	3000	5000	8000
0.3s	1.0s	2.4s	4.1s	9.2s	X

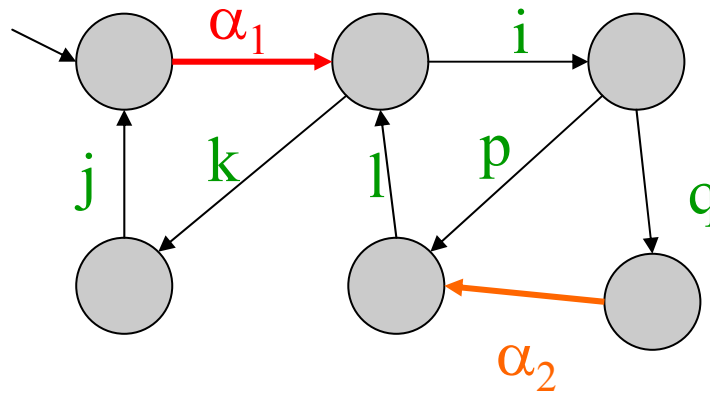
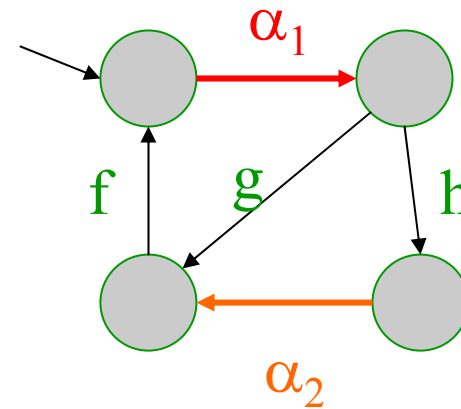
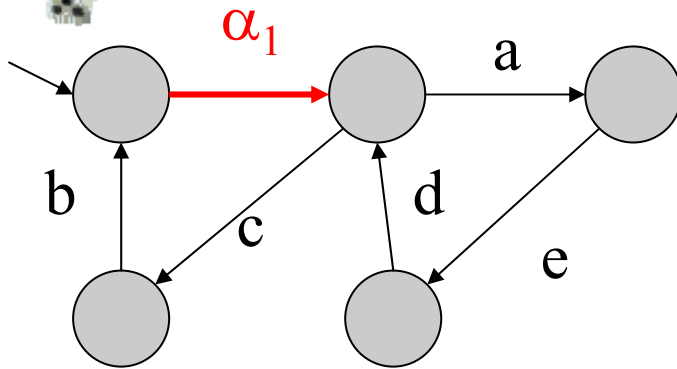
With three components (289, 1183, and 5486 states):

- system size  $\sim$  18 billions,
- time to generate 100 paths:

200	1000	2000	3000	5000	8000
0.2s	1.6s	2.3s	2.9s	X	X



# Model with synchronised components

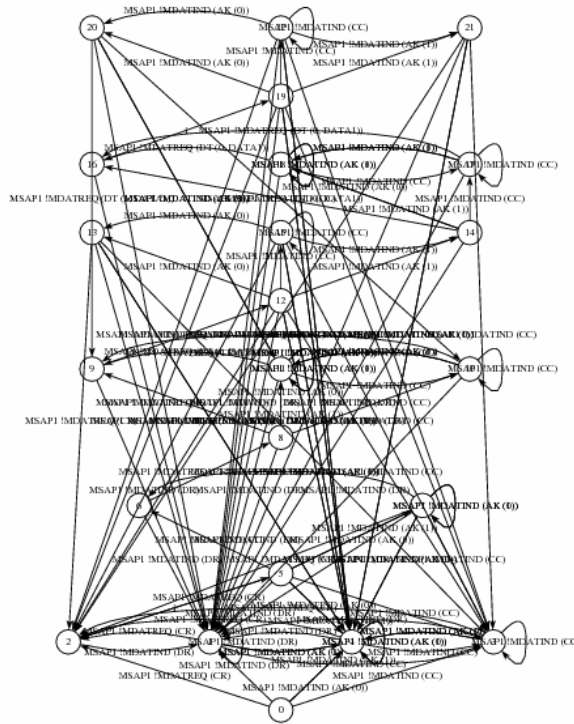


$\alpha_1 a e i h q \alpha_2 l d c k j f b \alpha_1 a h e \dots$

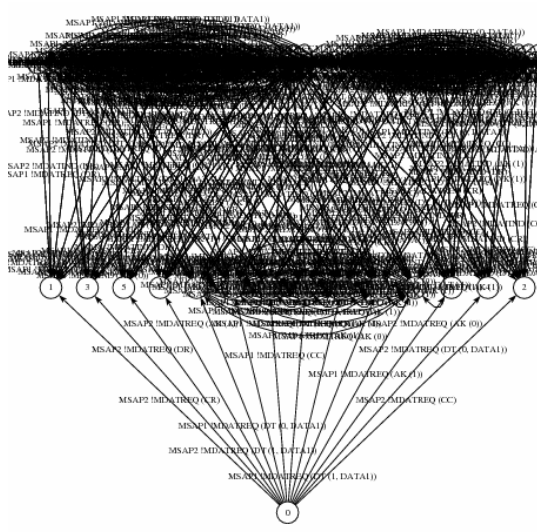
Note: many synchronised transitions  $\Rightarrow$  « smaller » global model



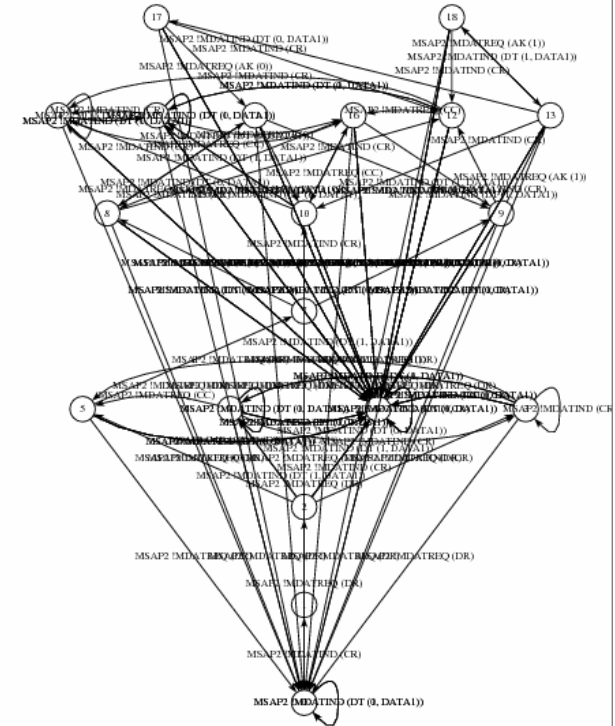
# INRES Communication Protocol



Initiator



Medium



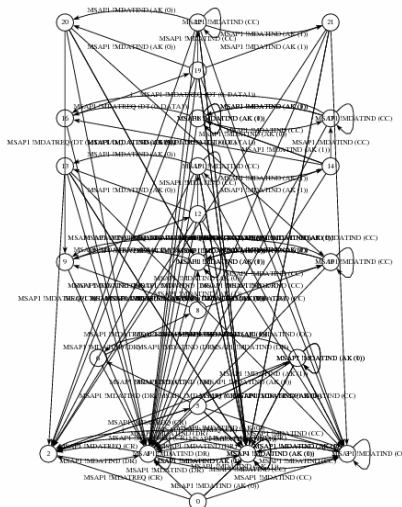
Responder

The INRES protocol implements a reliable, connection-oriented data transfer service between two users.

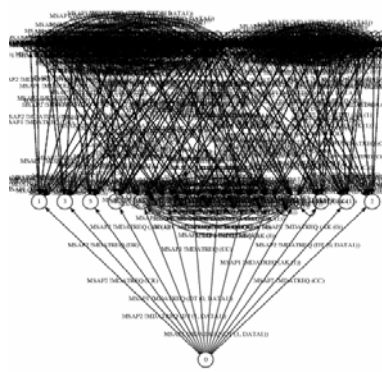
The protocol operates above a medium that offers an unreliable data transfer service.



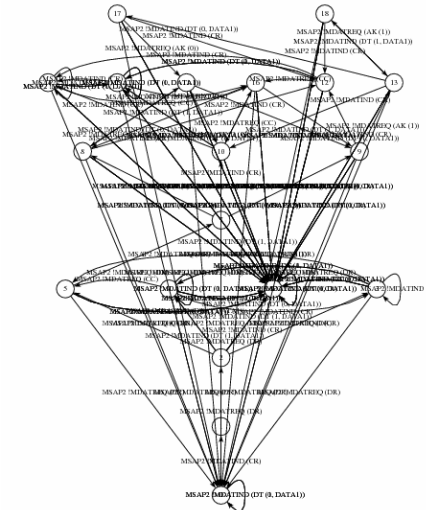
# Example of a model with many synchronised transitions: the INRES protocol



Initiator



Medium



Responder

Model	#states	#synchro. trans.	#non synchro. trans.
Initiator	34	111	4
Medium	65	294	0
Responder	26	81	2
<b>Global system</b>	<b>981</b>	<b>2290</b>	<b>262</b>



# Experiments with the INRES protocol

(brute force method)

length	200	500	1000	2000	4000	8000
	0.21s	0.74s	1.76s	4.53s	12.40s	<b>✗</b>

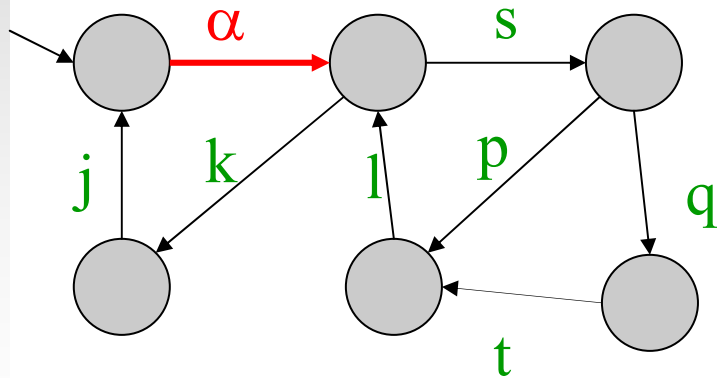
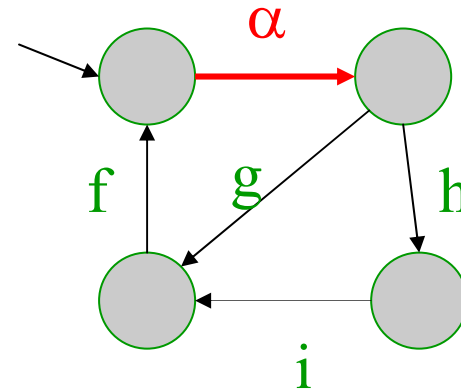
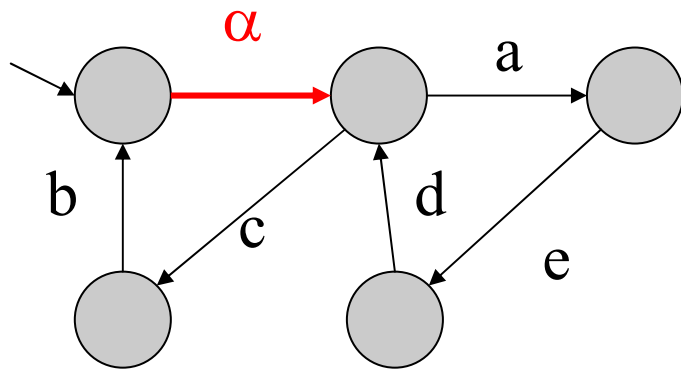
Table 11: Preprocessing: time to build the counting table from the INRES system according to the path lengths. **✗** means there is not enough memory to build the counting table.

length	200	500	1000	2000	4000	8000
	0.08s (0.00)	0.23s (0.00)	0.86s (0.00)	2.54s (0.00)	8.54s (0.00)	<b>✗</b>

Table 12: Generation: average time (and standard deviation) to generate 100 paths in the INRES system. **✗** means there is not enough memory to build the counting table.



# The case with many big models and 1 synchronised transition

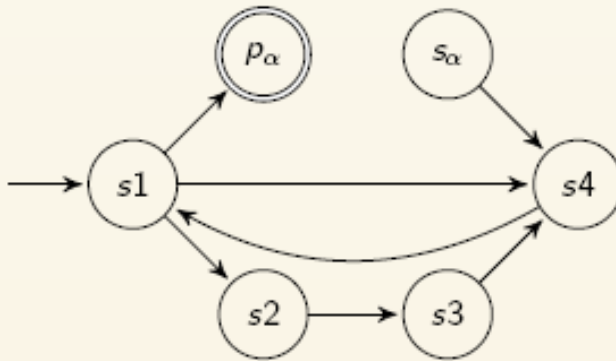
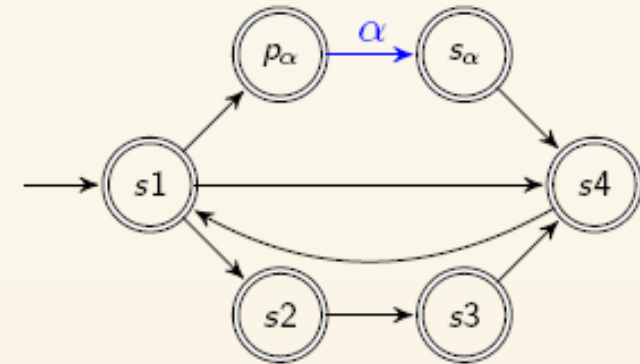


$\alpha$  aeshpildkjfcb  $\alpha$  ckgjfb  $\alpha$ ...



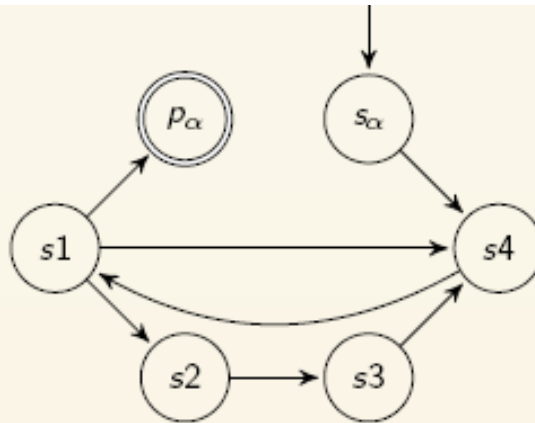
# Drawing in pruned local models

- If  $W$  is a path of length  $n$  with  $m$  synchronizations,
- then  $W = W_0 \alpha W_1 \alpha \dots \alpha W_m$   
 $m + 1$  shuffles of  $r$  sub-paths with no  $\alpha$ ,
- 4 sub-languages per component.



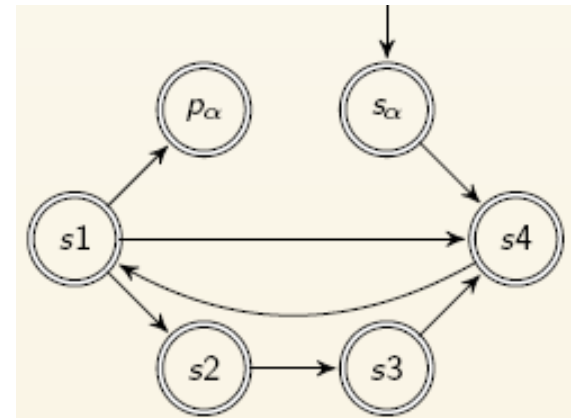
$W_0$

March 30, 2008



$W_1, W_2, \dots, W_{m-1}$

MBT'08



$W_m$

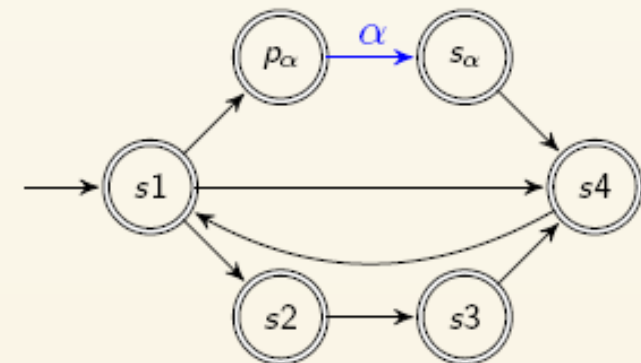
28

# A unique synchronization per component [RT'06]

- If  $W$  is a path of length  $n$  with  $m$  synchronizations,
- then  $W = W_0\alpha W_1\alpha \dots \alpha W_m$   
 $m + 1$  shuffles of  $r$  sub-paths with no  $\alpha$ ,
- 4 sub-languages per component.

Algorithm :

- 1 choose  $m$  with probability :  $l(n, m)/l(n)$ ,
- 2 choose the lengths  $(i_0, \dots, i_m)$  of the sub-paths,
- 3 draw  $m + 1$  times  $r$  paths in the suitable sub-language of each component.





## Some details about the algorithm...

Computing  $l(n, m)$ , the number of paths of length  $n$  with  $m$

synchronizations: 
$$l(n, m) = \sum_{i_0 + i_m = n - m} l(n, m, i_0, i_m)$$

$$l(n, m, i_0, i_m) \sim \binom{n - 2 - i_0 - i_m}{m - 2} (C_{b,1} \dots C_{b,r}) (C_{c,1} \dots C_{c,r})^{m-1} (C_{e,1} \dots C_{e,r}) \\ (\omega_{b,1} + \dots + \omega_{b,r})^{i_0} (\omega_{c,1} + \dots + \omega_{c,r})^{n-m-i_0-i_m} (\omega_{e,1} + \dots + \omega_{e,r})^{i_m}$$

Choosing the lengths  $(i_0, \dots, i_m)$ :

- $P(i_0, i_m) = l(n, m, i_0, i_m) / l(n, m)$
- $P(i_1, \dots, i_{m-1}) \sim 1 / \binom{n - 2 - i_0 - i_m}{m - 2}$



## ◆ Complexity :

- $n$  is the length of the generated paths,  $r$  is the number of components,  $|A_{max}|$  is the size of the biggest component
- preprocessing :
  - space:  $O(n^3)$  integers,
  - # operations:  $O(n^3)$ , linear in  $r$
- drawing :
  - #operations,  $O(n^2)$  and linear in  $r$  and  $|A_{max}|$ .

## Elapsed time to generate 100 paths

Each component - the 289 model - contains a unique transition labeled  $\alpha$ .

The size corresponds to the number of states of the global system.

**X** means there is not enough memory to compute all  $l(n, m, i_0, i_m)$

length	100	200	300	400	500	600
# states						
$10^4$	1.9s	2.1s	4.3s	4.7s	4.4s	X
$10^7$	0.9s	1.2s	1.2s	2.6s	3.8s	X
$10^9$	1.5s	1.5s	0.9s	3.2s	2.5s	X
$10^{12}$	3.0s	0.4s	0.9s	3.1s	2.5s	X
$10^{14}$	4.1s	0.7s	0.9s	3.8s	2.5s	X
$10^{17}$	3.7s	2.9s	1.9s	1.6s	2.4s	X
$10^{24}$	4.6s	0.6s	0.8s	1.5s	2.2s	X



### **Pro :**

- Good complexity w.r.t. the size of the global model
- Generalisable to several (not many) synchronised transitions

### **Cons :**

- There is the assumption that all the involved languages (even small) satisfy the asymptotic formula
- Tables with  $O(n^3)$  big integers.
- Quadratic w.r.t. the number of synchronised transitions.

**On-going :** combining brute-force and local-drawing methods depending on the architecture; avoiding the tables...



# Main results of the talk

- ◆ Experimental assessment of this method and of its limits (up to  $\pm 8000$  states)
- ◆ Uniform generation of paths in very large models:
  - Given a model described as a set of concurrent LTS/FSM/IOTS it is possible to get (a very good approximation of) **global uniform random walks**,
    - without building the global model
    - using **local uniform drawing** of paths in the component LTS
- ◆ **Combination of weaker coverage criteria (transitions, states, ...) and random walks**
  - Coverage-biased random walks, “randomised coverage satisfaction”

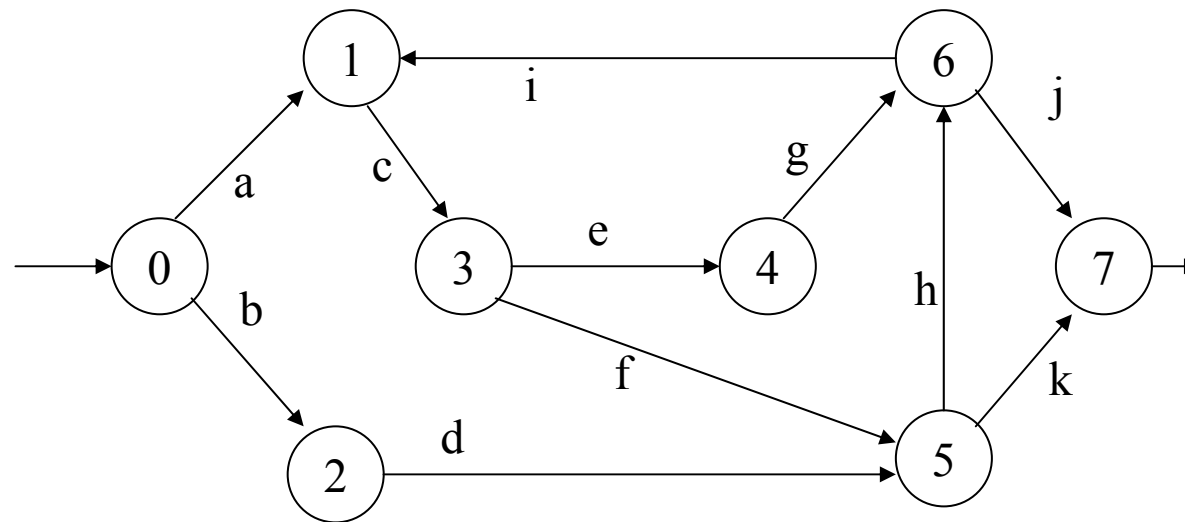


# Other coverage criteria

- ◆ All vertices
- ◆ All transitions
- ◆ MC/DC, etc
  
- ◆ How a randomised selection of paths can ensure a good coverage of such elements of a graph?
- ◆ What does it mean to “satisfy” a coverage criteria in a randomised framework?



# An example (control graph, transition system, ...)



We just saw that it is possible to generate paths that ensure uniform coverage of paths.

Generating paths that ensure uniform coverage of vertices or transitions is generally not possible for topological

Ma reasons...



# Quality of a random distribution on paths w.r.t a coverage criterion

- ◆ Given some graph  $G$  and a distribution on its paths,
- ◆ Given a coverage criterion  $C$  that requires a set  $S$  of elements of  $G$   $\{e_1, \dots, e_m\}$  to be covered,
- ◆ Let  $p_i$  the probability to reach  $e_i$  when drawing a path
- ◆  $p_{min} = \min(p_1, \dots, p_m)$  is the minimal probability of any element to be covered to be reached by a path
- ◆ *Idea:* Find a distribution that maximise  $p_{min}$



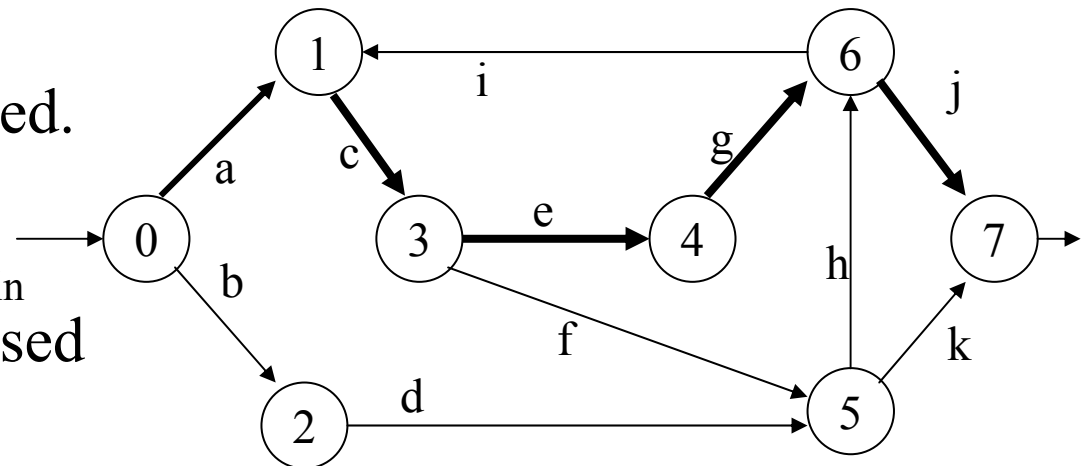
# Yes, but...

- ◆ Maximising  $p_{min}$  must not lead to give up the randomness of the method

## An extreme example:

$\{a, c, e, g, j\}$  must be covered.

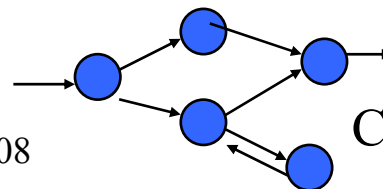
Giving probability 1 to path a.c.e.g.j maximises  $p_{min}$   
But it is no more a randomised test method...





# Biasing random walks toward coverage criteria

- ◆ It is a difficult multi-criteria problem that would require:
  - Maximising  $p_{\min}$ , i.e. the minimal probability to any element to be covered to be reached by a path
  - Maximising the minimal probability of any path traversing an element to be covered
- ◆ There are cases where these criteria are contradictory...



C is all-vertices



# A weakened version of the problem

- ◆  $p_{\min}$  must be maximum
- ◆ Any path traversing an element to be covered must have a non-null probability



## A weakened version (cont.)

- ◆ **A two-step approach:**

1. Draw non uniformly among the  $e_1, \dots, e_m$  to be covered, with probability  $p_1(e_i)$
2. Given this drawn  $e_k$ , draw uniformly a path of length  $\leq n$  among those traversing  $e_k$

- ◆ Let  $S$  be the set of elements to be covered (statements, branches, ...);  $\forall e \in S$ , the probability of  $e$  to be reached is

$$p(e) = p_1(e) + \sum_{e' \in S \setminus \{e\}} p_1(e') \frac{|C_n^{e,e'}|}{|C_n^{e'}|}$$

- ◆ where  $|C_n^e|$  is the nb of paths traversing  $e$  and  $|C_n^{e,e'}|$  is the nb of paths traversing both  $e$  and  $e'$



# Note on the power of combinatorial structures

- ◆ The use of such structures makes it possible to express, count, and draw uniformly from  $C_n^e$  and  $C_n^{e,e'}$
- ◆ Our *August* prototype builds automatically, from the text of C programs CS specifications for  $C_n$  and  $C_n^e$ . The  $C_n^{e,e'}$  are useful for optimisation purpose only
- ◆ *Generic, again: applicable to FSM and LTS with  $S$  the set of states, or of transitions, or of transition pairs...*



## How to **increase** $\min_{e \in S} p(e)$ ?

- ◆ By drawing **non uniformly** from S:
  - let  $S = \{e_1, \dots, e_m\}$ , and  $p(e_i)$  the probability of  $e_i$  to be covered

- let  $c_{i,j} = \frac{|C_n^{e_i, e_j}|}{|C_n^{e_j}|}$ , then  $p(e_i) = \sum_{1 \leq j \leq m} p_1(e_j) \times c_{i,j}$

- ◆ Given the  $c_{i,j}$ , the problem of finding values  $\{p_1(e_1), \dots, p_1(e_m)\}$  such that:
  - $\min\{p_1(e_i), i = 1, \dots, m\}$  is maximum
  - and  $p_1(e_1) + \dots + p_1(e_m) = 1$is a classical one in combinatorial optimisation



PB : Maximise  $p_{\min}$  under these constraints

$$\left\{ \begin{array}{l} p_{\min} \leq c_{1,1} \times p_1(e_1) + \dots + c_{1,|S|} \times p_1(e_{|S|}) \\ \dots \\ p_{\min} \leq c_{|S|,1} \times p_1(e_1) + \dots + c_{|S|,|S|} \times p_1(e_{|S|}) \\ 1 = p_1(e_1) + p_1(e_2) + \dots + p_1(e_{|S|}) \end{array} \right.$$

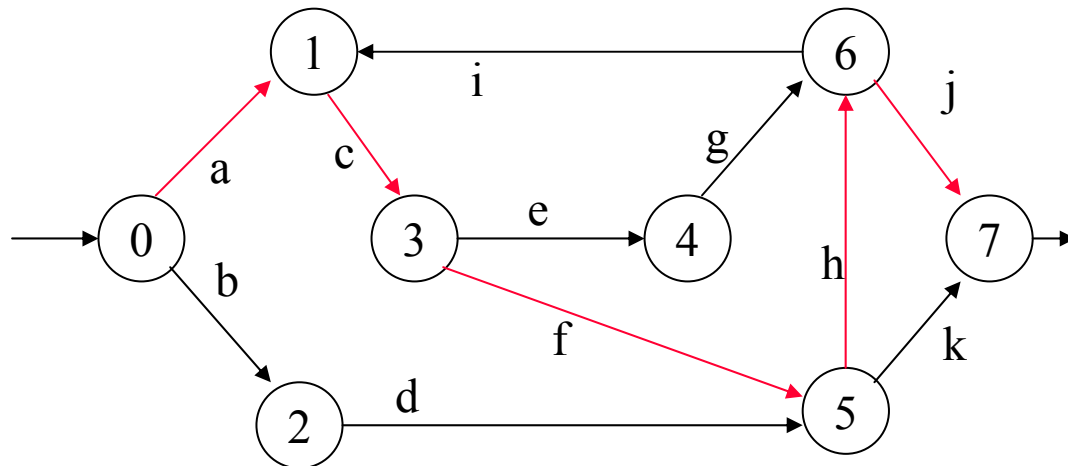
*This optimisation problem can be solved by a Simplex algorithm.*

*But, there may be paths traversing some  $e_i$  that get a null probability...*



# The example again

- ◆ The coverage criteria is “all-transitions”
- ◆ The result of the above method is in the table, BUT



March 30, 2008

MBT'08

$p_1(a)$	0
$p_1(b)$	0.3125
$p_1(c)$	0
$p_1(d)$	0
$p_1(e)$	0.375
$p_1(f)$	0
$p_1(g)$	0
$p_1(h)$	0
$p_1(i)$	0
$p_1(j)$	0
$p_1(k)$	0.3125
$\mathbf{P}_{\min}$	<b>0.5</b>

45



# Decreasing $p_{\min}$ to avoid impossible paths

$$\left\{ \begin{array}{l} p_{\min} \leq c_{1,1} \times p_1(e_1) + \dots + c_{1,|S|} \times p_1(e_{|S|}) \\ \dots \\ p_{\min} \leq c_{|S|,1} \times p_1(e_1) + \dots + c_{|S|,|S|} \times p_1(e_{|S|}) \\ 1 = p_1(e_1) + p_1(e_2) + \dots + p_1(e_{|S|}) \end{array} \right.$$

Addition of  $|S|$  new constraints :  $p_1(e_i) \geq \varepsilon$ , with  $\varepsilon = 0.0001$



# The final distribution

- ◆ The coverage criteria is “all-transitions”
- ◆ with  $\varepsilon = 0.0001$



March 30, 2008

MBT'08

$p_1(a)$	0.001
$p_1(b)$	0.001
$p_1(c)$	0.001
$p_1(d)$	0.301
$p_1(e)$	0.001
$p_1(f)$	0.001
$p_1(g)$	0.3708
$p_1(h)$	0.001
$p_1(i)$	0.001
$p_1(j)$	0.001
$p_1(k)$	0.311
<b><math>P_{\min}</math></b>	<b>0.4908</b>



# Conclusions and perspectives

- ◆ Combining random exploration (testing, model-checking) and coverage criteria is feasible
- ◆ Significant results for drawing paths UAR in concurrent models
  - work remains to be done on:
  - partial synchronisations
  - typology of concurrent architectures and synchronisation patterns
  - partial order reduction
- ◆ First results on “coverage satisfaction” assessment and improvement for randomised model-based testing