

# Uniform Random Walks in Very Large Models

Johan Oudinet  
<oudinet@lri.fr>

Laboratoire de Recherche en Informatique (LRI)  
<http://www.lri.fr>  
Université de Paris-Sud & CNRS

Random Testing 2007 (RT'07)

November 6, 2007



# Outline

- ➊ Introduction
- ➋ Uniform Random Walks in a Single Model
- ➌ Uniform Drawing in Non-synchronized Systems without building the global system
- ➍ Uniform Drawing in Synchronized Systems without building the global system
- ➎ Conclusion



## ① Introduction

## ② Uniform Random Walks in a Single Model

## ③ Uniform Drawing in Non-synchronized Systems without building the global system

## ④ Uniform Drawing in Synchronized Systems without building the global system

## ⑤ Conclusion

# State Explosion Problem

In Model-Checking and Software Testing:

- have to face a combinatorial explosion of the state-space of real-world models,
- how to explore such systems?

Two types of solutions:

- Symbolic (e.g. BDD) or abstraction (e.g. State Fusion) methods that try to reduce the models,
- Random methods that pick some paths *at random*.



# State Explosion Problem

In Model-Checking and Software Testing:

- have to face a combinatorial explosion of the state-space of real-world models,
- how to explore such systems?

Two types of solutions:

- Symbolic (e.g. BDD) or abstraction (e.g. State Fusion) methods that try to reduce the models,
- **Random methods that pick some paths *at random*.**



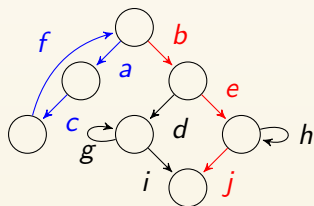
# Random Walks

Pros:

- need only local knowledge,
- can explore very large models.

Cons:

- unbalanced and unpredictable coverage of paths.



$$Pr(\{a, c, f\}) = 0.5$$

$$Pr(\{b, e, j\}) = 0.125$$

The best way to balance the probability of each path:  
uniformity of paths.



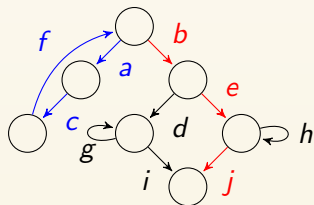
# Random Walks

Pros:

- need only local knowledge,
- can explore very large models.

Cons:

- unbalanced and unpredictable coverage of paths.



$$Pr(\{a, c, f\}) = 0.5$$

$$Pr(\{b, e, j\}) = 0.125$$

The best way to balance the probability of each path:  
uniformity of paths.



① Introduction

② **Uniform Random Walks in a Single Model**

③ Uniform Drawing in Non-synchronized Systems without building the global system

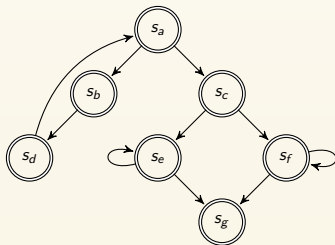
④ Uniform Drawing in Synchronized Systems without building the global system

⑤ Conclusion

# Uniform drawing of paths of length $n$

Principle:

- compute for each state the number of paths of length  $k$  ( $0 \leq k \leq n$ ) that start from this state (to be done once),
- run a guided random walk according to those values.



Building the table requires

$O(n \times |G|)$  operations.

Then, drawing one path needs

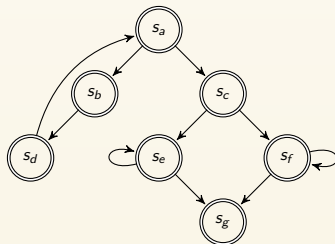
$O(n)$  arithmetic operations.



# Uniform drawing of paths of length $n$

Principle:

- compute for each state the number of paths of length  $k$  ( $0 \leq k \leq n$ ) that start from this state (to be done once),
- run a guided random walk according to those values.



#	$s_a$	$s_b$	$s_c$	$s_d$	$s_e$	$s_f$	$s_g$
0	1	1	1	1	1	1	1
1	2	1	2	1	2	2	0
2	3	1	4	2	2	2	0
3	5	2	4	3	2	2	0

Building the table requires

$O(n \times |G|)$  operations.

Then, drawing one path needs

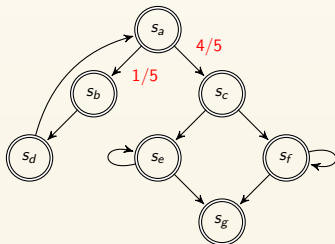
$O(n)$  arithmetic operations.



# Uniform drawing of paths of length $n$

Principle:

- compute for each state the number of paths of length  $k$  ( $0 \leq k \leq n$ ) that start from this state (to be done once),
- run a guided random walk according to those values.



#	$s_a$	$s_b$	$s_c$	$s_d$	$s_e$	$s_f$	$s_g$
0	1	1	1	1	1	1	1
1	2	1	2	1	2	2	0
2	3	1	4	2	2	2	0
3	5	2	4	3	2	2	0

Building the table requires

$O(n \times |G|)$  operations.

Then, drawing one path needs

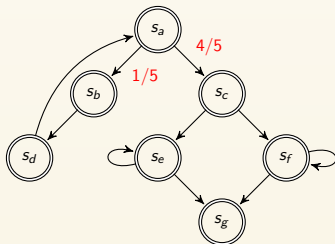
$O(n)$  arithmetic operations.



# Uniform drawing of paths of length $n$

Principle:

- compute for each state the number of paths of length  $k$  ( $0 \leq k \leq n$ ) that start from this state (to be done once),
- run a guided random walk according to those values.



Building the table requires

$O(n \times |G|)$  operations.

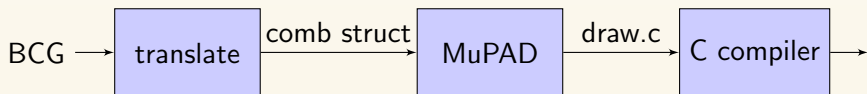
Then, drawing one path needs

$O(n)$  arithmetic operations.

#	$s_a$	$s_b$	$s_c$	$s_d$	$s_e$	$s_f$	$s_g$
0	1	1	1	1	1	1	1
1	2	1	2	1	2	2	0
2	3	1	4	2	2	2	0
3	5	2	4	3	2	2	0



# Implementation



`./draw n k` ← draw  $k$  paths of length  $n$ .

- Binary Coded Graph (BCG) from CADP toolbox [VASY/INRIA]
- MuPAD-Combinat: a library in MuPAD that can generate a C program to draw objects of length  $n$  from a combinatorial structure.



# Benchmarks for the uniform generation of paths

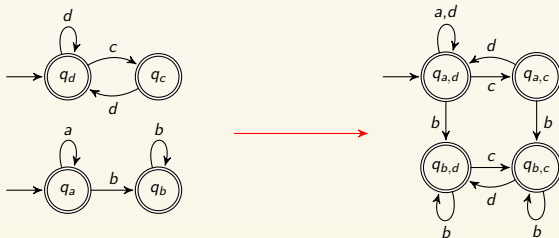
- Models are from the VLTS benchmark suite:  
<http://tinyurl.com/yuroxx>
- Benchmarks are performed by using an Intel Xeon 2.8GHz with 1GB memory during my master's thesis.
- **X** means there is not enough memory to build the table.

Elapsed time to generate 100 paths:

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



# A large model is composed of several smaller components



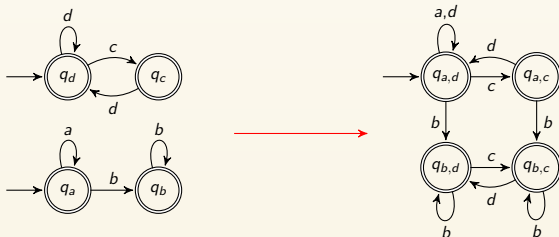
The composition of components is :

- *non-synchronized* if the components are independent,
- *synchronized* if some actions have to be simultaneously performed by several components.

If there is no synchronization, a path from the global model is an interleaving of paths from its components.



# A large model is composed of several smaller components



The composition of components is :

- *non-synchronized* if the components are independent,
- *synchronized* if some actions have to be simultaneously performed by several components.

If there is no synchronization, a path from the global model is an interleaving of paths from its components.



① Introduction

② Uniform Random Walks in a Single Model

③ **Uniform Drawing in Non-synchronized Systems without building the global system**

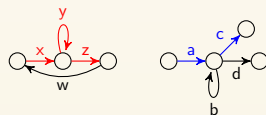
④ Uniform Drawing in Synchronized Systems without building the global system

⑤ Conclusion

# Principle [RT'06]

- 1 Select the path length  $n$ ,
- 2 choose the lengths  $(n_1, \dots, n_r)$  of paths to be drawn in each component according to a suitable distribution,
- 3 uniformly draw, at random, a path of length  $n_i$  in each component,
- 4 *suitably* shuffle those  $r$  paths.

$$n = 6$$



x y y z    a c

x y a y z c

Complexity:

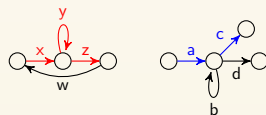
- pre-processing: cubic in the size of the largest component and linear to the length  $n$ ,
- generation:  $O(r \times n)$



# Principle [RT'06]

- 1 Select the path length  $n$ ,
- 2 choose the lengths  $(n_1, \dots, n_r)$  of paths to be drawn in each component according to a suitable distribution,
- 3 uniformly draw, at random, a path of length  $n_i$  in each component,
- 4 *suitably* shuffle those  $r$  paths.

$$n = 6$$



x y y z    a c

x y a y z c

Complexity:

- pre-processing: cubic in the size of the largest component and linear to the length  $n$ ,
- generation:  $O(r \times n)$



## How to choose the lengths $(n_1, \dots, n_r)$

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

### A well-known theorem [e.g. see Flajolet and Sedgewick, 01]

Let  $l_i(n_i)$  be the number of paths of length  $n_i$  in the component  $M_i$ ,  
If the component is aperiodic and strongly connected then:

$l_i(n_i) \sim C_i \omega_i^{n_i}$ , where  $C_i$  and  $\omega_i$  are computed from the generating function of the component.

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



## How to choose the lengths $(n_1, \dots, n_r)$

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

### A well-known theorem [e.g. see Flajolet and Sedgewick, 01]

Let  $l_i(n_i)$  be the number of paths of length  $n_i$  in the component  $M_i$ ,  
If the component is aperiodic and strongly connected then:

$l_i(n_i) \sim C_i \omega_i^{n_i}$ , where  $C_i$  and  $\omega_i$  are computed from the generating function of the component.

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



## How to choose the lengths $(n_1, \dots, n_r)$

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

### A well-known theorem [e.g. see Flajolet and Sedgewick, 01]

Let  $l_i(n_i)$  be the number of paths of length  $n_i$  in the component  $M_i$ ,  
If the component is aperiodic and strongly connected then:

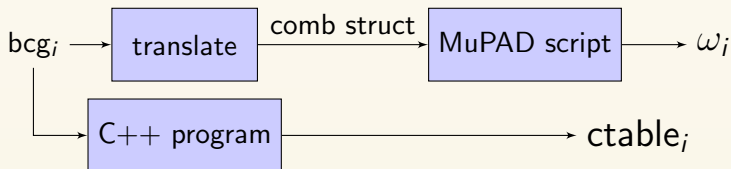
$l_i(n_i) \sim C_i \omega_i^{n_i}$ , where  $C_i$  and  $\omega_i$  are computed from the generating function of the component.

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

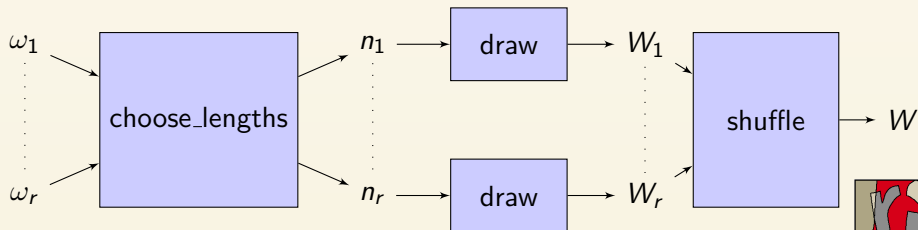


# Workflow

Pre-processing for each component:



The drawing of a path  $W$  of length  $n$ :



# Elapsed Time to generate 100 paths

with/without building the global system

length	200	1000	2000	3000	5000	8000
# states						
289	0.0s	0.9s	2.9s	6.3s	15.9s	40.1s
8879	0.2s	0.8s	2.4s	✗	✗	✗
$10^4$	0.0s 0.4s	1.3s 0.9s	✗ 2.4s	✗ 4.3s	✗ 10.2s	✗ 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	✗

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

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

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



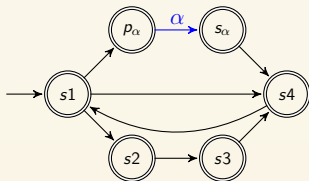
- ① Introduction
- ② Uniform Random Walks in a Single Model
- ③ Uniform Drawing in Non-synchronized Systems without building the global system
- ④ Uniform Drawing in Synchronized Systems without building the global system**
- ⑤ Conclusion

# 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.

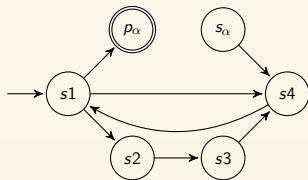


# A unique synchronization per component [RT'06]

- 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.

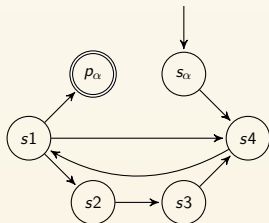


# A unique synchronization per component [RT'06]

- 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.

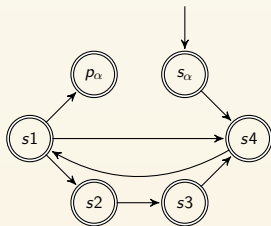


# A unique synchronization per component [RT'06]

- 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.

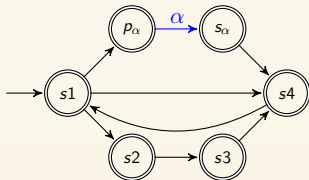


# A unique synchronization per component [RT'06]

- 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  $I(n, m)$ , the number of paths of length  $n$  with  $m$

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

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

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

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

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

Complexity:

- Pre-processing:  $O(r \times n^4)$  in time and  $O(n^3)$  in memory,
- generation:  $O(n^2)$ .



## Some details about the algorithm...

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

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

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

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

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

Complexity:

- Pre-processing:  $O(r \times n^4)$  in time and  $O(n^3)$  in memory,
- generation:  $O(n^2)$ .



## 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



① Introduction

② Uniform Random Walks in a Single Model

③ Uniform Drawing in Non-synchronized Systems without building the global system

④ Uniform Drawing in Synchronized Systems without building the global system

⑤ Conclusion

## What has been done in my master's thesis

- A method that both explores very large models and takes into account a coverage criterion (here, all paths of length  $\leq n$ ).
- Benchmarks that show the uniform generation of paths in one model is efficient for medium-sized models.
- Benchmarks for the uniform drawing of paths in models that are the product of several components.



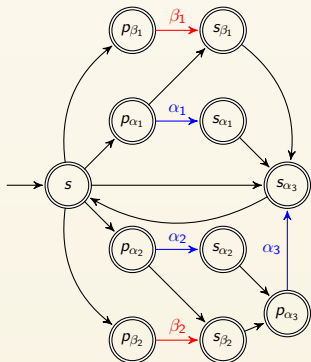
## Further work

- Other coverage criteria
- Benchmarks for the uniform generation in systems with several synchronizations
- Exploit the system architecture and topology of the models to decrease the number of synchronizations to consider.
- A free open-source library to uniformly draw paths at random and dedicated to Software Testing and Model-Checking



## Several synchronizations (not in the paper)

- $W = W_0 \alpha W_1 \beta \dots \alpha W_m$   
 $m + 1$  shuffles of  $r$  sub-paths without synchronization
- For  $k$  synchronized transitions,  
 $(k + 1)^2$  languages per component.
- The choice of the synchronized transition matrix:  $A_m \in [1..k]^{r \times m}$
- The same algorithm as for one synchronization, but choose  $A_m$  instead of  $m$ .  
 → need to compute  $k^{r \times n}$  values.



# Partial synchronizations

- Where does a component stop before a synchronization ?
- The case where there is only one label of synchronization:
  - (i) generate a first path from components that synchronize,
  - (ii) generate a second path from asynchronous components,
  - (iii) shuffle these two paths.
- The general case implies each state of a component that doesn't synchronize can be the begin or the end of a sub-path.  
→ exploitation of the architecture.

