

Replace this file with `prentcsmacro.sty` for your meeting,
or with `entcsmacro.sty` for your meeting. Both can be
found at the [ENTCS Macro Home Page](#).

Coverage-biased Random Exploration of Models

Marie-Claude Gaudel^{1,2}

*LRI
Université Paris-Sud
91405 Orsay, France*

Alain Denise, Sandrine-Dominique Gouraud³

*LRI
Université Paris-Sud
91405 Orsay, France*

Richard Lassaigne⁴

*Equipe de Logique
Université Paris 7
75013 Paris, France*

Johan Oudinet, Sylvain Peyronnet⁵

*LRI
Université Paris-Sud
91405 Orsay, France*

Abstract

This paper describes a set of methods for randomly drawing traces in large models either uniformly among all traces, or with a coverage criterion as target. Classical random walk methods have some drawbacks. In case of irregular topology of the underlying graph, uniform choice of the next state is far from being optimal from a coverage point of view. Moreover, for the same reason, it is generally not practicable to get an estimation of the coverage obtained after one or several random walks: it would require some complex global analysis of the model topology. We present here some methods that give up the uniform choice of the next state. These methods bias this choice according to the number of traces, or states, or transitions, reachable via each successor.

Keywords: D.2.4 Software/Program Verification, D.2.5 Testing and Debugging, G.2.1 Combinatorics

¹ Marie-Claude Gaudel warmly thanks the MBT08 chairs for the invitation to the workshop and to the proceedings

² Email: mcg@lri.fr

³ Email: {denise, gouraud}@lri.fr

⁴ Email: lassaign@logique.jussieu.fr

⁵ Email: {Johan.Oudinet, syp}@lri.fr

1 Introduction

This paper describes a set of methods for randomly drawing traces in large models either uniformly among all traces, or with a coverage criterion as target.

Classical random walk methods have some drawbacks. In case of irregular topology of the underlying graph, uniform choice of the next state is far from being optimal from a coverage point of view. For instance, in Figure 1, when considering traces of length 3, trace *b.e.f* is followed with probability 0.5 while trace *a.c.d* has probability 0.03125 only. Moreover, for the same reason, it is generally not practicable to get an estimation of the coverage obtained after one or several random walks: it would require some complex global analysis of the model topology. We present

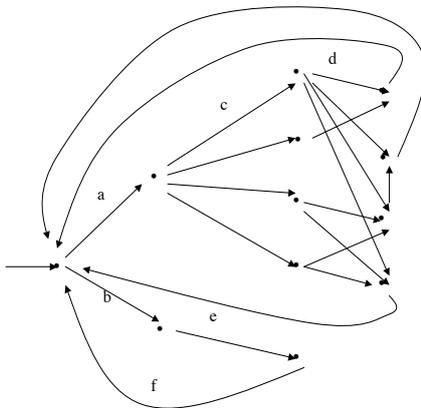


Fig. 1. Some example of irregular topology

here some methods that give up the uniform choice of the next state. They bias this choice according to the number of traces, or states, or transitions, reachable via each successor. The methods rely upon techniques for counting and drawing uniformly at random words in regular languages as defined by Flajolet et al. [5] and implemented in the CS package of the Mupad environment [12]. These techniques have, in the considered cases, a linear complexity in the size of the underlying automata, thus they allow dealing with rather large models.

Taking into account the number of traces starting from a state, it is shown in section 2 how it is possible to ensure a uniform probability on traces of a given length, or below a given length.

However, even linear complexity techniques cannot cope with very large models. But it is possible to exploit the fact that most of them are the result of the concurrent composition of several components, i.e a product, synchronised or not, of several models. Each component is considered as an automaton defining such a language. It is shown how it is possible to combine local uniform drawings of traces, and to obtain some global uniform random sampling, without constructing the global model. This is described in section 3

Considering coverage of other elements of the model than traces, such as states or transitions, is done by maximising the minimum probability to reach such an element, thus biasing random exploration toward classical coverage criteria such as state coverage or transition coverage, or less classical ones. Thus the probability of reaching a given coverage criterion after a certain number of drawings can be assessed. This is described in [section 4](#).

2 Uniform generation of traces of a given length

First, let us consider traces of a given length n . Given any state s , let $\ell_s(k)$ be the number of traces of length k that start from s . If s has m successors s_1, s_2, \dots, s_m , a condition for trace uniformity is to choose s_i after s with probability $\ell_{s_i}(k-1)/\ell_s(k)$. The issue is to compute these probabilities in an efficient way. Fortunately, traces in a model can be expressed as combinatorial structures. The next subsection is a very short introduction to these structures and powerful methods for counting them.

2.1 A few words on counting and drawing combinatorial structures

Two major approaches have been developed for counting and drawing uniformly at random combinatorial structures : The Markov Chain Monte-Carlo approach (see e.g. the survey by Jerrum and Sinclair [8]) and the so-called recursive method, as described by Flajolet et al. in [5] and implemented in [12]. Although the former is more general in its applications, the latter is particularly efficient for dealing with a very large class of structures, namely those formed of *atoms* combined by the following constructions:

$$+, \times, \text{SEQ}, \text{PSET}, \text{MSET}, \text{CYC}$$

respectively corresponding to disjoint union, cartesian product, finite sequence, set, multiset, directed cycles. It is possible to state cardinality constraints via subscripts (for instance $\text{SEQ}_{\leq 3}$). These structures are called *decomposable structures*. The size of an object is the number of atoms it contains.

The enumeration of decomposable structures is based on generating functions. Let C_n be the number of objects of C of size n , the corresponding generating function is:

$$C(z) = \sum_{n \geq 0} C_n z^n$$

Decomposable structures can be translated into generating functions using classical results of combinatorial analysis. For instance, the specification $\mathcal{B} = \mathcal{Z} + (\mathcal{B} \times \mathcal{B})$ that defines binary trees, becomes $\mathcal{B} = z + \mathcal{B}^2$. A comprehensive dictionary is given in [5].

The main result on counting and uniform random generation of decomposable structures is:

Theorem 2.1 *Consider a combinatorial class C , which is decomposable. Then the counts $\{C_j | j = 0 \dots n\}$ can be computed in $O(n^{1+\epsilon})$ arithmetic operations. In addi-*

tion, it is possible to draw an element of size n uniformly at random in $O(n \log n)$ arithmetic operations in the worst case.

Remark 2.2 A first version of this theorem, with a computation of the counting sequence $\{C_j | j = 0 \dots n\}$ in $O(n^2)$ was given by Flajolet et al. in [5]. The improvement to $O(n^{1+\epsilon})$ is due to van der Hoeven [13].

This theory has led to powerful practical tools for random generation [12]. There is a preprocessing step for the construction of the $\{C_j | j = 0 \dots n\}$ tables. Then the drawing is performed following the decomposition pattern of C , taking into account the cardinalities of the involved sub-structures.

2.2 Drawing uniformly at random traces of length n

Traces can be considered as decomposable structures where atoms are transitions that are combined by products and disjoint unions. For such simple structures, uniform drawing of size n is in $O(n)$.

For a given length n , the number $\ell_s(k)$, $k = 0, \dots, n$ of traces of length k starting from state s is given by the following recurrence formulas:

$$\begin{aligned} \ell_s(0) &= 1 \\ \ell_s(k) &= \sum_{s \rightarrow s'} \ell_{s'}(k-1) \text{ for } k > 0 \end{aligned}$$

where $s \rightarrow s'$ means that there exists a transition from s to s' .

From the theorem above, it is possible to compute these values in linear time. However, the number of values to be stored is n times the number of states of the model. This is the bottleneck of the method.

2.3 Generalisation to traces of length $\leq n$

It is possible to generate traces of length $\leq n$ instead of exactly n , by performing a small change in the model. It is sufficient to add a new state s'_0 which becomes the new initial state, with a fictive transition from s'_0 to s_0 and a fictive loop transition from s'_0 to itself. Each trace of length $n+1$ from s'_0 to a state of this new model crosses k times the new loop transition for some k such that $0 \leq k \leq n$ and exactly once the one from s'_0 to s_0 . It is obvious to associate to such a trace a trace of length $n-k$ of the original model. It is straightforward to verify that any trace of length $\leq n$ can be generated in such a way, and the generation is uniform.

2.4 Some experimental results

We report some experiments performed on models of the VLTS (Very Large Transition Systems) benchmark suite that is maintained by INRIA and CWI and available at <http://tinyurl.com/yuroxx>. These first experiments have been performed on a Intel Xeon 2.80GHz processor with 1GB memory, only. The table below was first published in [10].

As said above, the main problem is memory. When there is enough memory to store the $\ell_s(k)$, $k = 0, \dots, n$, it is very efficient. However, one can see in Table 1

	length						
name	# states	200	1000	2000	3000	5000	8000
vasy_0_1	289	0.0s	0.9s	2.9s	6.3s	15.9s	40.1s
vasy_1_4	1183	0.1s	1.0s	3.2s	6.7s	18.2s	✗
vasy_5_9	5486	0.0s	0.9s	2.4s	5.2s	✗	✗
vasy_8_24	8879	0.2s	0.8s	2.4s	✗	✗	✗
vasy_10_56	10^4	0.0s	1.3s	✗	✗	✗	✗
vasy_12323_27667	10^7	✗	✗	✗	✗	✗	✗

Table 1
Elapsed time for uniform drawing of 100 traces of various lengths in models of various sizes. ✗ means there is not enough memory to build the table of the $\ell_s(i)$

that it was not possible to deal with systems of size more than 10^4 states. Even if the memory size of the processor was bigger, it is rather unlikely that it would be possible to gain an order of magnitude. This is the motivation of the next section, where we show that generating traces of composed models allows to handle up to 10^{27} states and more.

3 Uniform trace generation in products of models

Fortunately, huge models are rarely stated from scratch. They are obtained by composition of smaller ones, the main source of state number explosion being parallel compositions. We show here how to use uniform random walks in the components to perform (approximately) uniform random walks in the global system.

3.1 Parallel composition without synchronisation

When there is no synchronisation, the parallel composition of r models Mod_1, \dots, Mod_r is the product of the underlying automata [1]. The set of states is the cartesian product of the states of the Mod_i models. Transitions are labelled by the union of the labels of the Mod_i models. Traces are interleavings of local traces, namely of traces of the Mod_i models. A brute force method to uniformly drawing traces is to build the product and to use the methods above. Since it is possible for moderate sizes only (more or less 10^4 states) we have developed an alternative method that avoids the construction of the global model. This method is presented in detail in [3] and [10]. We sketch it below.

- Given n the length of the global trace to be drawn
- Choose some lengths n_1, \dots, n_r such that $\sum_{i=1, \dots, r} n_i = n$, with adequate probabilities (see below)
- For each Mod_i , draw uniformly at random some trace w_i of length n_i
- Interleave the r w_i in a randomised way that ensures uniformity among interleavings.

Let $\ell(n)$ be the number of global traces of length n , and $\ell_i(k), i = 1, \dots, r$ the number of traces of length k in Mod_i . The choice of the n_1, \dots, n_r should be done with the probability below:

$$(1) \quad \Pr(n_1, \dots, n_r) = \frac{\binom{n}{n_1, \dots, n_r} \ell_1(n_1) \dots \ell_r(n_r)}{\ell(n)}$$

where the numerator is the number of interleavings of length n that can be built with r local traces of lengths n_1, \dots, n_r . Since computing the exact value of $\ell(n)$ would require the construction of the global model and of the corresponding tables, we use the following approximation from [4]:

$$(2) \quad \ell(n) \sim C\omega^n.$$

where C and ω are two constants. A sufficient, but not necessary, condition for this approximation to hold is aperiodicity and strong connectivity of the automaton, which is satisfied by any LTS with a reset. Details of weaker conditions can be found in [4]. This approximation is precise enough even for small values of n since $C\omega^n/\ell(n)$ converges to 1 at exponential rate.

Using the same approximations for the $\ell_i(n_i), i = 1, \dots, r$, we get (see [3]):

$$(3) \quad \ell(n) \sim C_1 \dots C_r (\omega_1 + \dots + \omega_r)^n$$

and then

$$(4) \quad \Pr(n_1, \dots, n_r) \sim \frac{\binom{n}{n_1, \dots, n_r} \omega_1^{n_1} \omega_2^{n_2} \dots \omega_r^{n_r}}{(\omega_1 + \omega_2 + \dots + \omega_r)^n}.$$

This avoids the computation of $\ell(n)$, and the constants $\omega_i, i = 1, \dots, r$ are computable in polynomial time with respect to the size of the Mod_i . It means that the complexity is dependent on the size of the components only, and not of the size of the global model.

In [3], we provide an algorithm for drawing n_1, \dots, n_r with this probability without computing it: draw a random sequence of n integers in $\{1, \dots, r\}$, with the probability to choose i equal to $\Pr(i) = \frac{\omega_i}{\omega_1 + \omega_2 + \dots + \omega_r}$; then take as n_i the number of occurrences of i in this sequence.

This concludes the issue of the choice of the n_1, \dots, n_r . Below, we recall a classical randomised way of interleaving r w_i of lengths n_i that ensures uniformity.

Algorithm 1 $w \leftarrow \varepsilon; n \leftarrow \sum_{i=1 \dots r} n_i;$

while $n > 0$ *do* {
 choose i *in* $\{1, \dots, r\}$ *with probability* n_i/n ;
 put the first symbol of w_i *at the end of* w ;
 remove the first symbol of w_i ;
 $n_i \leftarrow n_i - 1; n \leftarrow n - 1$ }

done

To summarise, we have presented a way of drawing traces of length n approximately uniformly at random in a product of r models Mod_i , without constructing the product. The complexity of the method is:

- polynomial (cubic) in the sizes of the Mod_i models for the computations of the constants ω_i , that are done once only for each $i, i = 1, \dots, r$, as pre-processing

the drawings of the n_i ,

- linear in n for the drawing of the n_i ,
- linear, in the sizes of the Mod_i models and in n , for pre-processing the uniform local drawings of the w_i , to be done once only for each $i, i = 1, \dots, r$,
- linear in n_i for the drawing of the local w_i ,
- linear in n for the interleaving of the w_i .

Table 2 gives the average times for drawing 100 traces of lengths 200 to 8000 in compositions of the *vasy_0_1* model (see Table 1) with itself⁶. The *vasy_0_1* model has 289 states.

length	200	500	1000	2000	4000	8000
# components						
2	0.58s	0.91s	1.83s	4.71s	14.93s	37.39s
4	0.91s	1.25s	2.42s	5.09s	14.57s	36.05s
6	1.37s	1.73s	3.00s	6.39s	18.31s	42.21s
8	1.78s	2.20s	3.70s	7.91s	22.61s	49.88s
10	2.16s	2.76s	4.57s	9.30s	26.82s	58.61s
12	2.65s	3.41s	5.31s	11.23s	31.36s	68.73s

Table 2
Average times for drawing 100 traces in composed models without synchronisation (*vasy_0_1* is composed with itself).

We observe that 289^{12} is of the same order as 10^{27} . It means that this method is applicable to very large models. Other experiments are reported in [10].

3.2 Parallel composition with synchronisation

We present here the generalisation of the method above to synchronised products when there is one synchronised transition in every model. Let note α its label, $s_i^{\rightarrow\alpha}$ its origin state in Mod_i , and $s_i^{\alpha\rightarrow}$ its target state in Mod_i model.

A trace in the synchronised product [1] of the Mod_i models has the following shape:

$$W_0\alpha W_1\alpha W_2\alpha \dots \alpha W_m$$

where m is the number of occurrences of α and

- W_0 is some interleaving of r local traces $w_i^0, i = 1, \dots, r$, without α , that respectively start from the initial state of Mod_i and lead to $s_i^{\rightarrow\alpha}$,
- W_1 is some interleaving of r local traces $w_i^1, i = 1, \dots, r$, without α , that respectively start from $s_i^{\alpha\rightarrow}$ and lead to $s_i^{\rightarrow\alpha}$; W_2, \dots, W_{m-1} are defined in similar

⁶ This table is slightly different from the one presented during the workshop, and it is also the case for Table 3. New experiment campaigns have been performed with a new implementation where pre-processings and drawings are no more performed with MUPAD, but with a specific C++ library. Each figure in these new tables corresponds to the average time for ten drawings.

ways.

- W_m is some interleaving of r local traces $w_i^m, i = 1, \dots, r$, without α , that respectively start from $s_i^{\alpha \rightarrow}$ and lead to any state of Mod_i .

Moreover, there is a special case for $m = 0$, namely interleavings of r local traces $w_i, i = 1, \dots, r$, without α , that respectively start from the initial state of Mod_i and lead to any state of Mod_i .

The definition above leads to consider four sub-languages of traces without α for each Mod_i , i.e. four sub-models.

The generalisation of the method of [subsection 3.1](#) is sketched below:

- Given n the length of the global trace to be drawn,
- Choose some m with probability $\ell(n, m)/\ell(n)$ where $\ell(m, n)$ is the number of global traces of length n with m synchronisations α ,
- Choose some lengths i_0, \dots, i_m for the W_0, \dots, W_m such that $\sum_{k=0, \dots, m} i_k = n - m$, with adequate probabilities,
- For each W_k , draw uniformly at random some global trace of length i_k in the product of the r relevant sub-models using the method of [subsection 3.1](#),
- Build the concatenation of the $m + 1$ W_k interspersed with m occurrences of α .

Details on the computations of the suitable probabilities, using the same approximation as in [subsection 3.1](#) can be found in [3] and in [9]. The approximate computation of the values $\ell(n, m)/\ell(n)$ is in $O(n^3 + r)$ operations and needs to be done once only for a given n . Drawing is quadratic in n . The various pre-processings are polynomial in the size of the biggest Mod_i model.

We performed some experiments using a modified version of the *vasy_0.1* model (see [Table 1](#)), where we picked at random a transition and labelled it as synchronised, that we composed with itself several times (2 to 12). [Table 3](#) gives the average times for drawing 100 traces of lengths 100 to 600 in these compositions. The limitation to 600 is due, again, to memory overflow during the pre-processing.

length	100	200	300	400	500	600
# components						
2	0.57s	1.17s	2.05s	4.97s	7.16s	9.50s
4	1.09s	1.71s	3.96s	6.09s	13.37s	18.38s
6	1.53s	2.44s	5.87s	9.15s	13.99s	27.01s
8	1.93s	3.22s	5.64s	11.92s	18.05s	25.53s
10	2.38s	4.01s	6.85s	10.47s	23.04s	32.18s
12	3.01s	4.80s	8.17s	12.54s	27.29s	38.38s

Table 3
Average time for drawing 100 traces in composed systems with synchronisation (the modified version of *vasy_0.1* is composed with itself).

The generalisation to several synchronisations is sketched in [9] and is the topic of an on-going Ph.D. thesis. Preliminary results on complexity let think that it will be practicable in the case of a small number of synchronisations only. However, in presence of many synchronisations, the synchronised product is smaller and a brute-force method, where it is constructed and used for uniform drawings, may become feasible. Actually, practical solutions are probably combinations of these two approaches depending on the architecture of the global system. The classification of architecture patterns accordingly to such combinations is also part of this on-going work.

4 Combination of weaker coverage criteria and randomness

Until now, we have considered as coverage criterion the coverage of all paths of length $\leq n$. This criterion is very demanding and in practice other ones are used, such as all transitions or all states. In this section we discuss the following questions: How a randomised selection of paths can ensure a good coverage of such elements of a model? What does it mean to “satisfy” a coverage criteria in a randomised framework? Then we provide some method for drawing traces according to such coverage criteria.

4.1 Quality of a random distribution on traces w.r.t. a coverage criterion

Let G be some graph, and Ω some distribution on its paths (bounded by a given length if there are loops). Given a coverage criterion C that requires a set $\{e_1, \dots, e_m\}$ of elements of G to be covered, Ω defines some probabilities p_i to reach $e_i, i = 1, \dots, m$ when drawing a path in G . Let $p_{min} = \min(p_1, \dots, p_m)$. Similarly to Thévenod-Fosse and Waeselinck in [11] we start with the idea that p_{min} characterises the quality of distribution Ω w.r.t. coverage criterion C . Thus it seems natural, for a given G and a given C to define the Ω distribution in a way such that p_{min} is maximum.

However, maximising p_{min} must not lead to give up the randomness of the method. It may happen, for instance, when there is a path in G going through all the elements e_1, \dots, e_m to be covered. In this case, maximising p_{min} leads to give probability 1 to this path and 0 to all the other ones, yielding a deterministic exploration strategy of the graph...

It turns out that biasing random walks toward coverage criteria in an optimal way is a somewhat difficult multi-criteria problem that would requires, in addition to the maximisation of p_{min} , a non null probability for any path traversing an element to be covered. Even more, to be coherent with the maximisation of p_{min} , the minimal probability on such paths should be maximised. But there are cases where these requirements are contradictory (see [7], [6]).

In the next sub-section, we present a method that slightly weakens these requirements. It makes it possible to draw paths at random in such a way that: any path traversing an element to be covered must have a non-null probability and the minimal probability of reaching by a path any element to be covered is maximum.

4.2 A drawing method in two steps

Given a graph G and a set $S = \{e_1, \dots, e_m\}$ of elements to be covered (vertices or arcs of G), drawing a path of length $\leq n$ is performed in the following way:

- (i) Draw non uniformly some e among the e_1, \dots, e_m with probabilities $\pi_i, i = 1, \dots, m$ (the determination of these probabilities is given below),
- (ii) Draw uniformly a path of length $\leq n$ among those traversing e .

Let α_i the number of paths of length $\leq n$ in G traversing e_i , and $\alpha_{i,j}$ the number of paths of length $\leq n$ in G traversing both e_i and e_j , the probability of an element $e_i, i = 1, \dots, m$ to be reached when drawing a path is:

$$p_i = \pi_i + \sum_{j \in [1..m] - \{i\}} \pi_j \frac{\alpha_{i,j}}{\alpha_j},$$

which simplifies to

$$(5) \quad p_i = \sum_{j=1}^m \pi_j \frac{\alpha_{i,j}}{\alpha_j}$$

since $\alpha_{i,i} = \alpha_i$. Note that using techniques for counting combinatorial structures as seen in [subsection 2.1](#), the α_i and $\alpha_{i,j}$ can be efficiently computed. More details can be found in [2].

We now explain how to compute the π_i values in order to fulfill the two requirements stated in the above sub-section. The values $\pi_i, i = 1, \dots, m$ must satisfy the following two first constraints:

$$\pi_1 + \dots + \pi_m = 1, \quad \forall i = 1, \dots, m, \pi_i \geq \varepsilon$$

The second constraint is sufficient to ensure that any path traversing an element to be covered has a non-null probability. The choice of the value of ε is a matter of balance between the two requirements: the bigger it is, the smaller is the maximum possible value of p_{min} . Moreover, p_{min} must satisfy the following constraints, that are linear in π_i :

$$\forall i \leq m, \quad p_{min} \leq p_i = \sum_{j=1}^m \pi_j \frac{\alpha_{i,j}}{\alpha_j}$$

Therefore, the maximisation of p_{min} is just a linear programming problem where the unknowns are the $\pi_i, i = 1, \dots, m$. Thus standard methods that are polynomial in m can be used.

This method was implemented and experimented on some C programs. Some experimental results are reported in [6] and [2].

4.3 Assessing coverage satisfaction

For deterministic testing methods, coverage satisfaction is a binary concept: either it is achieved by a test set, or not. Moreover, the minimal required number of tests can be determined. In the case of randomised testing methods, coverage satisfaction becomes a probabilistic concept: what is the probability of satisfying a given criterion after a certain number of tests? What number of tests are required to satisfy it with a given probability?

Given a random test method based on some distribution Ω on the paths of a graph G , and a coverage criterion C characterised by a set $S = \{e_1, \dots, e_m\}$ of elements to be covered, the probability to reach element e_i after N tests is greater than $1 - (1 - p_{min})^N$. This gives a minorant of the probability to satisfy C after N tests.

Conversely, the number N of tests to reach criterion C with a given probability p_C satisfies the inequality:

$$(6) \quad N \geq \frac{\log(1 - p_C)}{\log(1 - p_{min})}$$

We get the minimum required number of tests to satisfy C with a given probability.

5 Conclusion

In this paper, we have briefly presented how to combine random exploration of models and coverage criteria. We have reported some promising first results in this area. A common basis of these various pieces of work is that they are based upon powerful techniques on counting and drawing uniformly at random combinatorial structures. It is our belief that these techniques could bring much to simulation, model based testing, structural testing or model-checking.

References

- [1] Arnold A. , “Finite Transition Systems”, Prentice-Hall, 1994.
- [2] A. Denise, M.-C. Gaudel et S.-D. Gouraud. *A Generic Method for Statistical Testing*, 15th IEEE Int. Symp. on Software Reliability Engineering (ISSRE), 25-34, (2004).
- [3] Denise A., M.-C. Gaudel, S.-D. Gouraud, R. Lassaigne and S. Peyronnet. *Uniform Random Sampling of Traces in very Large Models*. 1st Int. ACM Workshop on Random Testing (2006), 10 -19, ACM Press.
- [4] Ph. Flajolet and R. Sedgewick. Analytic combinatorics: functional equations, rational, and algebraic functions, INRIA Research Report RR4103, January 2001, 98 pages. Part of the book project “Analytic Combinatorics”. URL: <http://algo.inria.fr/flajolet/Publications/books.html>.
- [5] Flajolet Ph., and P. Zimmermann, and B. Van Cutsem. *A Calculus for the Random Generation of Labelled Combinatorial Structures*, Theoretical Computer Science, **132** (1994), 1-35.
- [6] Gouraud S.-D. “Utilisation des Structures Combinatoires pour le Test Statistique”. PhD thesis, Université Paris-Sud 11, Orsay, 2004.
- [7] Gouraud S.-D., A. Denise, M.-C. Gaudel, and B. Marre. *A new way of automating statistical testing methods*. In IEEE Int. Conf. on Automated Software Engineering (ASE), 5-12, 2001.
- [8] Jerrum M., and A. Sinclair. The Markov chain Monte Carlo method: an approach to approximate counting and integration. in “Approximation Algorithms for NP-hard Problems”, D.S.Hochbaum ed., PWS Publishing, Boston, 1996.
- [9] Oudinet J. “Uniform random walks in concurrent models”. Master thesis, Université de Paris-Sud, 2007 URL: <http://www.lri.fr/oudinet/publis/07/mthesis.pdf>
- [10] Oudinet J. Uniform random walks in very large models. 2nd Int. ACM workshop on Random testing (2007) 26-29, ACM Press.
- [11] Thévenod-Fosse P. and H. Waeselync. *An investigation of software statistical testing*. Journal of Software Testing, Verification and Reliability, **1(2)**, (1991), 5-26.
- [12] Thiéry N. M. Mupad-combinat algebraic combinatorics package for MUPAD. URL: <http://mupad-combinat.sourceforge.net/>.
- [13] van der Hoeven J. *Relax, but dont be too lazy*, Journal of Symbolic Computation, **34(6)** (2002), 479-542.