RAIRO. INFORMATIQUE THÉORIQUE ET APPLICATIONS

DOMINIQUE GENIET
RENÉ SCHOTT
LOŸS THIMONIER

A Markovian concurrency measure

RAIRO. Informatique théorique et applications, tome 30, n° 4 (1996), p. 295-304

<http://www.numdam.org/item?id=ITA_1996__30_4_295_0>

Informatique théorique et Applications/Theoretical Informatics and Applications
(vol. 30, n° 4, 1996, pp. 295-304)

A MARKOVIAN CONCURRENCY MEASURE (*) (**)

by Dominique GENIET ([1]), René SCHOTT ([2]) and Loÿs THIMONIER ([3])

Communicated by A. ARNOLD

Abstract. – *We show how to modelize concurrency between several processors in terms of automata and Markov chains; then, we define a concurrency measure which reflects more faithfully the behaviour of the processes and is in addition easy to compute with a symbolic manipulator like* Maple *(this is an improvement over a previous measure [3] whose computation appears to be expensive.*

Résumé. – *Nous montrons comment modéliser la concurrence entre plusieurs processeurs en termes d'automates et de chaînes de Markov; ensuite nous définissons une mesure de concurrence qui reflète plus fidèlement le comportement des processus et qui de plus est facile à calculer avec un logiciel de calcul symbolique comme* Maple *(ce qui constitue une amélioration par rapport à une mesure antérieure dont le calcul était coûteux).*

1. INTRODUCTION

The notion of concurrency plays a central role in parallel processing, and has been studied rather extensively over the last decade mostly from algebraic and/or semantic point of view. This explains why the published literature on concurrency measures is scanty. J. Françon [10] has shown how to count explicitely the number of correct (*i.e.* without deadlock) behaviours of parallel systems under the mutual exclusion policy. His measure is in fact the inverse of the radius of convergence of some generating function and no implementation based on this approach has been realized until now.

(*) A preliminary version of this paper appeared in the Proceedings of CAAP'90, LNCS 437, pp 177-190, Springer-Verlag.

(**) Received January 1992.

([1]) LISI-ENSMA, Téléport 2, Site du Futuroscope, 86960 Futuroscope Cedex, France, dgeniet@diane.univ-poitiers.fr

([2]) CRIN, Université Nancy-I, 54506 Vandœuvre-lès-Nancy Cedex, France, schott@loria.fr

([3]) LIFIA, UFR Maths Informatique, Université de Picardie, 33, rue Saint-Leu, 80039 Amiens Cedex, France.

0988-3754/96/04/$ 7.00/

J. Beauquier, B. Bérard and L. Thimonier [3] consider also all behaviours of the concurrent systems and their measure, based on Arnold-Nivat's model, is the average waiting time of the processes. It has been shown [12] that the computation of this measure is often time and space consuming. B. Charron-Bost [5] introduces a coefficient which reflects the interaction between the different processors during the execution of a parallel computation. It is an interesting investigation of the problem we consider here.

Our approach uses also Arnold-Nivat's model [2], and we show that the behaviour of concurrent systems is modeled by absorbing Markov chains whose transition matrices contain all information necessary for easy computation of the concurrency measure. Classical properties of Markov chains lead to simple linear algebra problems, easy to solve with a symbolic manipulator like Maple.

The organization is as follows. Section 2 provides the set of necessary definitions and properties of automata and languages. Section 3 is devoted to Arnold-Nivat's model and its probabilistic extension. Details about the computation of our Markovian concurrency measure are given in Sections 4 and 5. Section 6 contains an exemple, and Section 7 concludes the paper.

2. BASIC NOTIONS

We assume that the reader is familiar with the basic aspects of language theory ([4], [6]). As usual, Σ^* is the free monoid over the alphabet Σ. Let w be a word of $L \subset \Sigma^*$. We denote by $|w|$ the length of w, $|w|_x$ the number of occurrences of x in w (x may be a letter or a subword...), and by $w^{(i)}$ the i-th letter of w.

Let $(L_i)_{i\in[1,n]}$ be a family of regular languages defined over the alphabets $(\Sigma_i)_{i\in[1,n]}$. Let $r_i : \prod_{j=1}^{j=n} \Sigma_j \to \Sigma_i$ be the i-th projection. We extend r_i to an homomorphism $r_i : \left(\prod_{j=1}^{j=n} \Sigma_j\right)^* \to (\Sigma_i)^*$. $\Omega_{j=1}^{j=n} L_j = \left\{w \in \left(\prod_{j=1}^{j=n} \Sigma_j\right)^* | \forall\, i \in [1, n], r_i(w) \in L_i\right\}$ is called *homogeneous product.* It follows immediately that $\forall\, i$, $|r_i(w)| = |w|$.

Note that the homogeneous product of regular languages is always regular.

Let $(\Sigma_i)_{i\in[1,n]}$ be a family of alphabets and Σ be the product of these alphabets (a letter of Σ is a vector whose i-th component is a letter of Σ_i). We call *projector on the i-th component* the morphism that associates with an element w of Σ its i-th component w_i.

Let us remember that:

DEFINITION 1: *A finite automaton is a* 5*-uple* $(\Sigma, Q, S, F, \delta)$, *where* Σ *is an alphabet,* $Q \subset \aleph$ *(set of states),* $S \subset Q$ *and* $F \subset Q$ *(sets of initial and terminal states),* δ *being a subset of* $Q \times \Sigma \times Q$ *(set of transitions).*

Remark: The product of n automata (defined as follows: set of states = product of the sets of states, set of transitions = product of the sets of transitions, terminal state = product of the terminal states) accepts the homogeneous product of the languages ${}^{L}A_i$ [1].

Notations: For each $t = (i, w, j)$ of δ, we denote t_- the state i, the t_+ the state j and w_t the label w.

For each i of Q, we denote i_- the set of transitions t of δ such that $t = (x, w, i)$ and i_+ the set of transitions t of δ such that $t = (i, w, x)$.

DEFINITION 2: *A finite probabilistic automaton is a* 5*-uple* $(\Sigma, Q, S, F, \delta)$, *where* Σ *is an alphabet,* $Q \subset \aleph$ *(set of states),* $S \subset Q$ *(set of initial states) and* $F \subset Q \times [0,1]$ *(set of terminal states and the corresponding absorption probabilities),* $\delta \subset Q \times \Sigma \times [0,1] \times Q$ *the set of transitions and, if* $p(t)$ *(resp.* $p(i)$*) is the probability associated with the transition* t *(resp. the terminal state* i*), we get* $i \notin F \Rightarrow \sum_{t=i_+} p(t) = 1$, *and* $i \in F \Rightarrow \sum_{t=i_+} p(t) = 1 - p(i)$.

Remark: i is an absorbing state if $p_{i,i} = 1$, where $p_{i,i}$ is the probability to remain in the state i.

3. REPRESENTATION OF CONCURRENT SYSTEMS

First, we show how to modelize concurrent processes.

3.1. Arnold-Nivat's model

In this model, a system of processes is represented by a synchronized product, *i.e.* an homogeneous product, where the transitions labelled by some forbidden vectors have been removed (*see* [1]).

3.2. Construction of a synchronized system

The fact that a process is blocked is modeled by the symbol #, and ${}^{L}A_i$ is now replaced by ${}^{L}A_i \text{ш} \#^*$ (ш is the classical shuffle product), and our concurrency measure will correspond to the average number of #'s over all words accepted by the automaton.

3.3. Probabilistic automaton associated with Arnold-Nivat's model

The basic idea is to transform the previous automaton into a probabilistic one: the transitions are labelled with probabilities whose values are obtained through experiments on the program. Our assumptions are these of the synchronized product (*see* [1]).

3.4. Computation of the transition probabilities

The probabilities associated with the transitions of the product automaton are computed from the probabilities of the transitions of the automata associated with the sequential processes. In the following, we consider a transition t of the product automaton to be issued from a state s of the product automaton. t is the vector $(t_i)_{i\in[1,n]}$, each t_i being a transition of the automaton A_i. We denote $S_t = \{i \in [1,n] | w_t^{(i)} = \#\}$, and $\Pi(t) = \prod_{i\in[1,n]\setminus S_t} p(t_i)$. Let $\Phi = \sum_{t\in i_+} \Pi(t)$ be the output probabilistic flow of state i. The probability of each t issued from i is the output flow the value $\Pi(t)$ represents in the set i_+ : $p(t) = \frac{\Pi(t)}{\Phi}$.

The product of a family of probabilistic automata is a probabilistic automaton: its algebraic structure is the homogeneous product of the algebraic structures of the component probabilistic automata; the probabilities associated to its transitions fulfil the conditions of Definition 2: $p(i) + \sum_{t\in i_+} p(t) = \sum_{t\in i_+} \frac{\Pi(t)}{\Phi} = \frac{1}{\Phi} \sum_{t\in i_+} \Pi(t) = \frac{\Phi}{\Phi} = 1$ (one can see that this property holds for terminal states, too).

The effective determination of the probabilities is very simple for imperative statements ($p(t) = 1$), but must be detailed for test or loop statements as well as for absorption.

3.4.1. *Tests and absorption*

In the flowchart automaton A, a test statement *If...Then...Else* or *Case...Of...End* generates a state s with many output transitions $(t_i)_{i\in[1,p]}$: $|s_+| > 1$ (see *Fig.* 1). We determine the corresponding probabilities in the following way: the program is executed N times, and we count the number

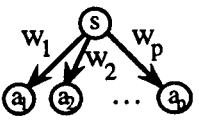

Figure 1.

of times the transition t_i is performed. Then, we take $p(t_i) = \frac{N_i}{\sum_{j=1}^{j=p} N_j}$. Absorption is treated in the same way: we count the number of times a terminal state is reached.

3.4.2. *Loops*

A loop can be represented by the diagram 

. Statistics on the average number of looping steps performed by real programs lead us to consider that the random variable X (equal to the number of steps required for leaving the loop) has a geometric distribution $p(X = n) = q^{n-1}(1 - q)$ (with $q \in [0, 1]$), hence $E(X) = \frac{1}{1-q}$.

Remark: When the implemented algorithm is well-known, the study of its speed of convergence gives an upper-bound for the maximal number of steps. From this, we deduce an upper bound for $E(X)$, and therefore for q.

4. THE CONCURRENCY MEASURE

The probabilistic extension of Arnold-Nivat's model leads to an absorbing Markov chain model. Evaluating the average length of a word accepted by the synchronized automaton is equivalent to evaluating the length of the paths before absorption in the corresponding Markov chain.

Let $\overline{n}$ (resp. $\overline{a}$) be the average number of steps (resp. #'s) until absorption, and p the number of processes.

DEFINITION 3: *The concurrency measure of the system is the ratio* $\frac{\overline{a}}{p\overline{n}}$.

5. COMPUTATION OF THE CONCURRENCY MEASURE

The concurrency measure, as defined above, involves two mean values whose computation is easy thanks to classical results of Markov chain theory ([7], [14]).

DEFINITION 4: *A Markov chain is a triple* $\langle E, s_0, M \rangle$, *where* E *is the set of states,* $s_0 \in E$ *the initial state, and* $M = ((m_{i,j}))$ *a* $E \times E$ *matrix such that* $\forall i \in E$, $\sum_{j=1}^{j=|E|} m_{i,j} = 1$.

A state $i \in E$ is called *transient* if the Markov chain comes back only a finite number of times to i (otherwise, i is called *recurrent*).

Our approach is based on properties of the fundamental matrix of an absorbing Markov chain. Definition and main properties are summerized below (*see* [7] and [14] for details).

Consider a Markov chain with s non absorbing states, and $|E|-s$ absorbing states. Then, the corresponding matrix $M = (m_{i,j})$ can be rewritten in the form $M = \left(\frac{I_s \mid 0}{R \mid W}\right)$, where W is the submatrix of transitions between the transient states, I_s is the $s \times s$ identity matrix (in our case, I_s is reduced to 1).

PROPOSITION 1: *The matrix $I_s - W$ has an inverse $N = (I_s - W)^{-1}$.*

N is called **fundamental** matrix of the Markov chain.

PROPOSITION 2: *Let $N = (n_{i,j})$ be the fundamental matrix of an absorbing Markov chain. Then.*

1. *$n_{i,j}$ is the average number of times the chain reaches the state j, if the starting state is i, and $m_i = \sum_{j=1}^{j=|E|-s} n_{i,j}$ is the average number of steps before absorption.*

2. *Let $a_{i,j}$ be the probability that starting from the state i, the absorption takes places in state k and denote by A the matrix $(a_{i,j})$, then $A = N.R$* (i.e. $A = R + W.A$).

Remark: All information necessary for computing the concurrency measure is therefore contained in M, W, and A. Only classical operations on matrices are necessary for its computation.

6. AN EXAMPLE

We consider the classical mutual-exclusion problem (see *Fig.* 2). The corresponding language is $L = (r^* d \# ^* c + s)^*$, where the actions are r (not critical section), d (asking for the resource), $\#$ (waiting when the resource is used by another process), c (critical section) and s (leaving the critical section).

For simplicity, we consider that all theses processes begin and terminate their execution simultaneously. L is accepted by the automaton given in Figure 2.

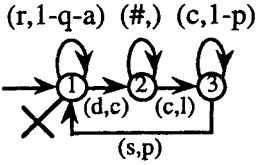

Figure 2.

6.1. Construction of the synchronize automaton

Let:

1. $\overline{\gamma}$ be the average time of each owning of the critical section.

$$\overline{\gamma} = 1 + E\left(\frac{\text{Number of transitions from the state 3 to 3}}{\text{The chain is in state 3}}\right)$$
$$= 1 + p\sum_{i=0}^{\infty} i(1-p)^i = \frac{1}{p}$$

$E\left(\frac{\cdots}{\cdots}\right)$ means the expected conditional value.

2. $\overline{\rho}$ be the average time spent in the critical section.

$$\overline{\rho} = 1 + E\left(\frac{\text{Number of transitions from the state 1 to 1}}{\text{The chain is in state 1}}\right)$$
$$= \sum_{i=0}^{\infty} i(1-q-a)^i(q+a) = \frac{1}{q+a} - 1$$

3. $\overline{\alpha}$ be the average number of request for the critical section.

$$\overline{\alpha} = 1 + E\left(\frac{\text{Number of transitions to state 2}}{\text{The chain is in state 1}}\right)$$
$$= \sum_{i=0}^{\infty} i.q^i(1-q) = \frac{1}{1-q} - 1 = \frac{q}{1-q}$$

Synchronization fobids simultaneous performing of the critical section by two different processes. Therefore, all transitions whose label contains more than one occurrence of the letter c must be delected. The synchronized automaton of two processes is drawn on Figure 3. The transitions labelled $(c, \#)$ and $(\#, c)$ from state (2.2) respectively to states (3,2) and (2,3) have the same probability: this is due to the fact that there is no priority for the processes.

The Markov chain associated with the synchronized automaton is given in Figure 4.

6.2. Use of the concurrency measure

The computing of the concurrency measure with different values for the indicators γ, ρ and α leads to different values for the efficiency of the concurrent system. If $\overline{n}$ is the average waiting time before absorption (*i.e.*

the average number of statements by session), these experimentations are summarized in the following array:

$\overline{\gamma}$	$\overline{\rho}$	$\overline{\alpha}$	$\frac{a}{p\overline{n}}$
5	10	5	0.2459
10	10	5	0.3798
10	5	5	0.3806
5	5	10	0.2509
5	10	10	0.2474

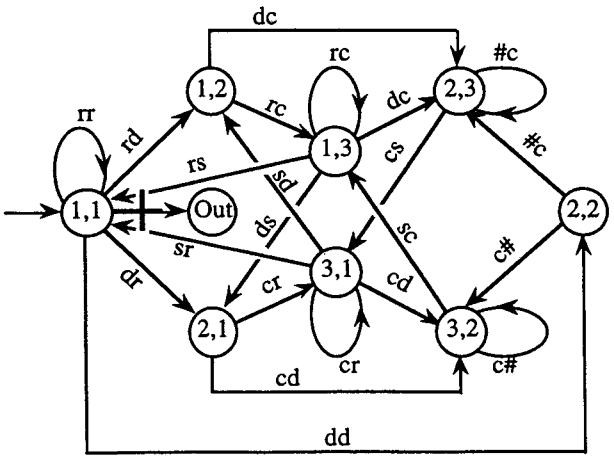

Figure 3.

Here, it is pointed out that the performances of the system are less efficient when processes stay a long time in the critical section. So, we conclude that a system of concurrent processes sharing a resource is more efficient when there are many short jobs using the resource than when there are few long jobs... This is a known result.

7. CONCLUSION

Evaluating the efficiency of concurrent processes is a very difficult problem. This paper is a contribution to the definition and computation of a simple and good concurrency measure. It combines the power of

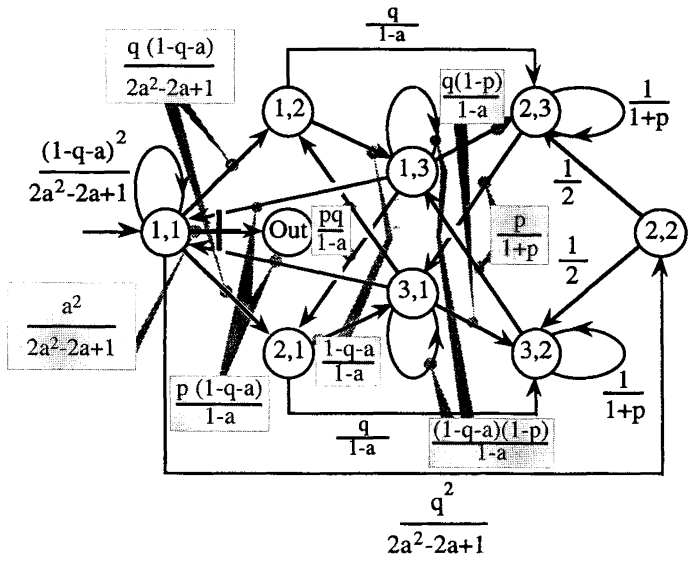

Figure 4.

Arnold/Nivat's model with the Markov chain theory, and avoids the computational drawbacks of previous measures ([3], [10], [11]).

We can also consider the concurrency measure as a stochastic process whose expected value has been investigated in this paper. Therefore, it would be interesting to know the variance and higher moments, the limiting distributions and the probability that at a time t, p processors among n are active. The techniques used for the analysis of dynamic data structures ([8], [9], [13]) seem to be helpful. These more theoretical aspects are the object of work in progress.

ACKNOWLEDGMENTS

The authors thank A. Arnold and the referees for pertinent comments.

REFERENCES

1. A. Arnold, *Finite transition systems*, Prentice Hall, 1994.
2. A. Arnold and M. Nivat, Comportements de processus L.I.T.P., Rapport No. 82-12, Univ. Paris, France.

3. J. BEAUQUIER, B. BÉRARD and L. THIMONIER, On a concurrency measure, LRI Orsay Techn. Report No. 288, 1986, and 2nd I.S.C.I.S. proc., Istanbul 1987, Turkey, pp. 211-225.
4. J. M. AUTEBERT, *Langages algébriques*, Masson, 1987.
5. B. CHARRON-BOST, Mesures de la concurrence et du parallélisme des calculs répartis, *Thèse*, Univ. Paris-VII, France, 1989.
6. S. EILENBERG, *Automata, languages and machines*, Academic Press, 1976.
7. W. FELLER, *An introduction to probability theory*, Addison-Wesley, 1968.
8. P. FLAJOLET, Analyse d'algorithmes de manipulation d'arbres et de fichiers, *B.U.R.O. Cahiers*, 34-35, 1981.
9. J. FRANÇON, B. RANDRIANARIMANANA and R. SCHOTT, Dynamic data structures with finite population: a combinatorial analysis, *F.C.T.'89 proc., L.N.C.S.* 380, pp. 162-174.
10. J. FRANÇON, A quantitative approach of mutual exclusion, *R.A.I.R.O. Theoretical Informatics and Applications*, No. 20, 1986, pp. 275-289.
11. D. GENIET and L. THIMONIER, Using generating functions to compute concurrency, *F.C.T.'89 proc., L.N.C.S.*, 380, pp. 185-196.
12. D. GENIET, Automaf: un système de construction d'automates synchronisés et de calcul de mesure du parallélisme, *Thesis,* Univ. Paris-XI, France, 1989.
13. G. LOUCHARD, B. RANDRIANARIMANANA and R. SCHOTT, Dynamic algorithms in D. E. Knuth's model: a probabilistic analysis, *I.C.A.L.P.'89 proc., L.N.C.S.*, 372, pp. 521-533, Stresa, Italy, 1989, full version in *T.C.S.*, 93, 1992, pp. 210-225.
14. C. PAIR, M. MOHR and R. SCHOTT, *Construire les algorithmes*, Dunod Informatique, 1988.