A Markovian Concurrency Measure

Dominique Geniet (i) - René Schott (iii) - Loïs Thimonier (i)(ii)

(i) I.R.I., Bât. 490, Université Paris XI, 91405 ORSAY CEDEX, FRANCE
(ii) Université de Picardie, 33 rue saint Leu, 80039 AMIENS CEDEX, FRANCE
(iii) C.R.I.N., Université Nancy I, 54506 VANDOEUVRE LES NANCY CEDEX, FRANCE

Abstract

The aim of this work is to define a useful concurrency measure, easy to implement and whose computation complexity allows the study of real examples. We extend the measure introduced in [BT87] to a probabilistic one, by means of a natural translation of the synchronized automata of Arnold-Nivat's model to Markov chains: the computation of the measure uses the concept of average time before absorption. Some examples including the mutual exclusion are detailed.

Introduction

The concept of concurrency measure has been studied by several authors until now, particularly Françon [F86], Beauquier, Bérand and Thimonier [BT87]. Françon's measure enumerates the correct behaviors of a family of processes sharing resources: it is computed as the convergence radius of a generating series.

Beauquier, Bérand and Thimonier's concurrency measure is defined as the average of waiting time in the behaviors of the processes: the computation uses the generating function concept too.

In these two examples, the exact value of the measure is computed by means of an analytic approach. Although this technique is theoretically powerful, it is rather expensive in running-time. For instance, [G89] shows that Beauquier, Bérand and Thimonier's measure (we denote this measure BBT in the present work) can't be effectively computed if the automaton modeling the system of processes contains more than 100 states. For Françon's, no implementation has been proposed until now.

So, no real system may be studied by means of the previous measures. The first aim of this work is the improvement of the computation of BBT.

This measure is based on Arnold-Nivat's model, using regular languages: a finite automaton A represents a process p, and every word accepted by A a possible behavior of p. So, a collection of processes \{P_1, P_2, ..., P_n\} is modeled by the homogeneous product of the associated automata: each transition is labelled by an n-uple \(\{x_1, x_2, ..., x_n\}\) of letters, the ith component of \(x_i\) being the action performed by \(P_i\). However, accesses to the different critical sections of such a system (writing on a memory block, resource access, etc...) must be controlled to be performed correctly. This is the aim of synchronization, which is performed as the suppression of some transitions in the multi-process model automaton.

The concurrency measure enumerates the possible behaviors of the processes, without any probabilistic law (all the behaviors of the processes are counted with the same multiplicity). For instance, all the behaviors of a looping program are counted in the same way, which is very inconvenient. Indeed, in the computation of the roots of a function by means of Newton's algorithm, it is clear that the number of iterations is more likely to be 10 than 10^{10}...

This leads us to build a new concurrency measure, which extends BBT: the model described in [BT87] is improved by the association of an absorbing Markov chain to the synchronized automaton. This one takes into account the properties of the studied algorithms: for instance the termination speed.

Let a (resp. b) be the average waiting time (resp. the average number of performed atomic actions) before the former absorption. Our measure is defined as the ratio \(\frac{a}{n \cdot b}\) which gives the efficiency of each process.

In the first part, we present Arnold-Nivat's model and we give a method to build the Markovian chain from the synchronized automaton. The second part is devoted to a probabilistic extension of BBT's measure, whereas in the third part, the automatic computation of this new measure is studied. Finally, in the fourth part, we present the mutual exclusion example.

I) Fundamental notions

1) Arnold-Nivat's model

The reader is supposed to be familiar with the basic theory of finite automata and regular languages [H78]. The classical operations of cartesian product and intersection of such languages are used in Arnold-Nivat's model [AN82].
Here, a process is defined by the set of its finite behaviors. We are interested here in rational processes, which are represented by means of finite automata: each word accepted by the automaton is a possible behavior of the modeled process. The alphabet of the language is the set of the possible atomic actions for the process. We call regular language associated to the process the regular language accepted by the automaton. For a process \( p \), we denote \( L(p) \) the language associated to \( p \), \( A(p) \) the associated alphabet and \( A(p) \) the associated automaton. One implicit option of this model is to consider the modeled systems as synchronous: each process must perform exactly one action at each time unit, each atomic action being supposed to endure exactly one time unit. This last constraint doesn't restrain the model, because it can be used to model more general systems by using the gcd of the atomic actions duration as the time unit. So, each action is supposed to endure a multiple of the time unit. For instance, if \( a \) endures 3.5ms, \( b \) endures 4.2ms and \( c \) endures 8.4ms, the gcd is 0.7ms, each occurrence of \( a \) is replaced by \( a^5 \), each occurrence of \( b \) by \( b^6 \) and each occurrence of \( c \) by \( c^{12} \).

The concurrent functioning of a family \( (p_i)_{i \leq n} \) of processes is modeled by the cartesian product \( \prod_{i=1}^{n} A(p_i) \).

Each transition of this automaton is labelled by a \( n \)-uple \( (x_i)_{i \leq n} \) of letters: the \( i \)th component \( x_i \) of the vector represents the action of the \( i \)th process [AN82]. When the processes have to communicate or to synchronize themselves, some configurations (that are \( n \)-uple of atomic actions) are forbidden: the set of synchronized concurrent behaviors is a subset of the set of the concurrent behaviors of the system. The synchronization constraints are modeled by a centralized control, that is an automaton which restrains the possible \( n \)-uple of actions at each time unit. So, we consider a set \( S \subseteq \prod_{i=1}^{n} \Sigma_i \). Let \( A_S \) be an automaton accepting \( \left( \prod_{i=1}^{n} \Sigma_i \right) \setminus S \). The intersection \( A_S \cap \left( \prod_{i=1}^{n} \left( A(p_i) \right) \right) \) accepts the set of the synchronized concurrent behaviors. So, we insert in each \( A(p_i) \) a new letter, \# , which models the waiting of the processes. This action is called active waiting. For each process, we consider the language \( L_{\#}(p_i) \) whose words are the sequential behaviors of \( p_i \) shuffled with \( \#^* \) (for the notion of shuffle, see [FGT88]). This language is regular.

**Power of this model**

The basic program description operations accepted by the described model correspond to the basic automata constructors: the atomic action is represented by the finite part concept, the choice statement by the union concept and the looping statement by the star. Most of the programs built by means of classical languages (Pascal, C, Fortran, etc...) only use these three concepts (it is shown in [PMS88] that recursion may always be avoided in a program by using a correct memory management). So, this simple model allows the representation of a large class of real programs.

**A technique to compact non-deterministic finite automata [G89]**

In order to improve the computations time, we present in [G89],[G89a] a technique for compacting automata, using words instead of letters for the labels of the transitions. In the present work, we use this compacted form: the word \( uvw \) labeling a transition of such a compacted automaton will be denoted \( u:v:w \), the character ":" is only a virtual separator.

As an example, the automaton given at the left is compacted at the right: