From Basic to Timed Net Models of Occam: an Application to Program Placement

Oliver Botti and Fiorella De Cindio

Dipartimento di Scienze dell'Informazione
Università degli Studi di Milano
Via Comelico, 39 - 20135 Milano (ITALY)

Abstract

Starting from the Petri net model of Occam, which is illustrated in [1] and uses 1-safe PT nets, the paper develops, on the top of it, a timed net model, using Generalized Stochastic Petri Nets (GSPN), to allow a performance analysis of Occam programs. As an example, we carry out the comparison of different placements of an Occam program over a given set of distributed processors.

1 Introduction

The development of a Petri net model of Occam programs [1] opens a wide range of possible applications, sketched in [2], based on the qualitative analysis techniques typical of net models. To extend the analysis of Occam programs to performance evaluation, in the following (section 2) we develop, on the top of it, a timed net model, using, in particular, the Generalized Stochastic Petri Nets (GSPN) as defined in [3]. We then present (section 3) the application of the GSPN model for comparing different program placements, i.e., for selecting the best solution among the possible assignments of the processes of an Occam program to a given set of distributed processors. Crucial for Occam, such field is today widely studied by means of different methodologies and tools [4,5].

First of all, let us therefore briefly sketch the general features characterizing the 1-safe PT net model of Occam. It has been developed starting from an abstract syntax which includes a suitable subset of Occam2 [6] (atomic actions; SEQ, IF, ALT, WHILE, PAR, PLACED PAR, PRI ALT, PRI PAR compositions; a subset of data and channels descriptions) together with some extentions to enhance the significancy of the model in the more general framework of (a wider class of) concurrent programming languages. The net model expresses the semantics of programs recognized by such an abstract syntax.

Since the very beginning, we aimed at developing a Petri net model of Occam, satisfying the modularity (the possibility to autonomously represent program modules within the correspondent model modules) and compositionality (the possibility to suitably compose simple modules to build more complex ones) requirements. Since the process is the structural basic computational component of Occam programs, we consider the modularity of a program in terms of its component processes. In the net model each process has its own corresponding *process-box*. A process-box [1,7] consists of: a labelled net which describes the control flow of an Occam process; two sets of places and two sets of transitions (called *interfaces*) which connect the process to its external environment. The *entry* and *exit* interfaces are two sets of places which are the pre-set, respectively the post-set, of the net which models the process. The *input* and *output* interfaces are two sets of transitions which represent the communication activities of input, respectively of output, of the process with its environment.

Given an Occam program, the compositional construction of the control flow model takes place through subsequent steps: first, a process-box for each atomic action is obtained; then these process-boxes are composed, step by step, by using suitable semantic operators defined (one for each syntactic operator of the language) to compose process-boxes as the Occam operators compose processes. Finally the suitable initial marking is given. In such a way, the process-box corresponding to the most external process, i.e. to the whole program, is produced (the details of its construction are here omitted for space reasons; see [1,2,8,9], where also the handling of data and priorities, which goes beyond the goal of this paper, is considered).

2 A GSPN model of Occam

To develop in §2.2, on the top of the 1-safe PT net model of Occam, the GSPN model, first (§2.1) we briefly recall some notions of the GSPN approach.

2.1 GSPN and preselection policies

A GSPN system (or GSPN for short) is a 7-tuple $\Sigma=(S,T,W,\Pi,H,\Lambda,Mo)$, where the underlying net system $N=(S,T,W,\Pi,H,Mo)$ is a PT system with priorities ($\Pi:T\rightarrow\mathbb{N}^+$) and inhibitor arcs ($H:S\times T\rightarrow\mathbb{N}$), with the property to be confusion-free at priority level greater than zero. We will consider only two priority levels, corresponding respectively to *timed* transitions (with priority zero) and to *immediate* transitions (with priority

TH0386-3/91/0000/0216/$01.00 © 1991 IEEE

greater than zero). The function $\Lambda:T\rightarrow\mathbb{R}^+$ associates with each timed transition its probabilistic exponentially distributed delay, and with each immediate transition (without delay) the suitable firing probability. Markings which enables timed transitions only are said to be *tangible*, while markings which enables *immediate* transitions are said to be vanishing. When several transitions have concession under a marking M, the priority level determines which transition(s) are actually enabled to fire. Therefore, in a vanishing marking M, only immediate transitions may be enabled, since they have priority over timed transitions. In particular, only one immediate transition t_i within each ECS (Extended Conflict Set [3]) is selected, according to a probability distribution determined by the associated weights. Let us remark that there is *true concurrency* among immediate transitions belonging to different ECS.

Timed transitions are actually enabled only in tangible markings. When several timed transitions are enabled in the same tangible marking M, firing rates are used to probabilistically select one transition to fire. The semantics of the competition between simultaneously enabled timed transitions is called *race policy* and implies that all transitions are assumed to compete with each other: the one that actually fires is the one for which the firing delay is minimum. This semantics sequentializes the firing of simultaneously enabled timed transitions; however, thanks to the absence of memory typical of the exponential distribution on which the delays are sampled [3], we can think of the actions associated with these transitions as actually going on concurrently, while their completions, represented by the (istantaneous) transition firings, are in sequence. This allows some degree of concurrency between timed transitions in a GSPN. This concurrency is exploited, in our case, to properly model the PAR composition.

In order to model the ALT composition we need a different policy to select simultaneously enabled (and conflicting) timed transitions, a policy based on additional specifications which are independent from the associated delays. One method is to define a probability distribution function over the set of enabled transitions in a given marking and to use it to (pre)select the next transition to fire. The implied semantics now is that transitions which are enabled but not preselected cannot fire: this behaviour properly models situations in which only one between several possible alternatives has to be choosen on the basis of information independent from time.

In the following we use a *local preselection policy* [10] by coupling a restricted use of the preselection policy with the race policy. In particular we apply the preselection at a local level, i.e. restricted to suitable sets of transitions, called *Local Preselection Sets* (LPS). In a given marking which enables transitions belonging to these sets, the next transition to fire is identified by first preselecting, according to the probability distribution locally defined, one enabled transition (if it exists) within each LPS, and then choosing, among the preselected transitions, the one for which the associated delay is minimum. Let us notice that this policy allows us: (1) to solve the conflicts within the net on the basis of time-indepentent information, thanks to the local preselection; and, (2) to represent in the model some degree of concurrency between the preselected transitions, thanks to the race policy.

The local preselection policy among conflicting transitions within an ECS is represented on the net by using *probabilistic arcs*. We define a function $P:SxT\rightarrow\mathbb{R}^+$ which associates with an (input) arc of each transition belonging to the ECS, a weight representing the preselection probabilitiy of that arc in the resolution of a conflict (the weights associated with the remaining arcs of the net are obviously to be intended equal to 1). The definition of the suitable preselection policy consists of the identification of the ECS to be used as LPS. In our case this is presented in the next section, and then illustrated by the Example 1.

2.2 The GSPN model construction and validation

The GSPN model for Occam consists of two main components. The *structural* component is built up according to the construction sketched in Section 1 which yelds a 1-safe PT system without priorities and inhibitor arcs (therefore in the following we will denote a GSPN system modelling an Occam program simply by $\Sigma=(S,T,W,\Lambda,Mo)$. The *stochastic* component consists of the function $\Lambda:T\rightarrow\mathbb{R}^+$, which associates with each transition a parameter representing, in the case of timed transitions, the rate of the exponential distribution of the associated delay, and in the case of immediate transitions, the weight used to determine the firing probability. The idea is to associate with each timed transition t corresponding to an atomic action of Occam, an exponentially distributed delay time with rate λ, defined as follows:

- λ_{skip} and λ_{ass} represent respectively the execution time of the SKIP and the assigment processes;
- λ_{stop}: the value λ_{stop}, representing the execution time of the STOP process, does not affect the semantics espressed by the GSPN model since the net model proposed for the STOP action [1] represents its semantics without involving any proper transition;
- transitions corresponding to matched communications are associated with a parameter λ_{comm} which represents the *phisical communication time*, i.e. the time necessary to the actual data transfer, while the syncronization (rendezvous) delays between input and output actions are instead captured by the causal structure of the net.

To determine the parameters values to be associated with each atomic action, it is useful to recall that the mean delay time of a transition k in a GSPN is $d_k=1/\lambda_k$.

The choice of the parameter λ_k may hence account for the mean execution time of the corresponding Occam action in some implementation. By the definition of the Λ function, the GSPN model is completed.

It is easy to show that, for each Occam composition, the GSPN model preserves the qualitative behaviour of the untimed one, and is therefore consistent with the Occam semantics. The proof (cf.[2]) is straightforward in the case of SEQ, IF, PAR and WHILE compositions, while it is worth discussing the ALT.

The *alternative composition* (ALT), is characterized by a non-deterministic choice between the possible alternatives: in the 1-safe PT model the entry transitions of each process-box corresponding to each component process are in conflict with each other. According to the standard interpretation of the original GSPN, the conflict between two or more transitions is solved in favour of the one with the shortest associated delay; this behaviour obviously does not correspond to the non-deterministic semantics of the ALT composition. The previously discussed local preselection policy plays its role here. We choose, as LPS, the actions contained in each ALT composition, therefore called in the following ALTS, standing for ALT Sets. This means to define a probability distribution function over each set of transitions in structural conflict. According to the local preselection policy, one enabled transition is selected within each ALTS, and then the race policy competition applies to these preselected transitions and to the other transitions of the model possibly enabled. It is worth noticing that the preselection applies *only* to the resolution of conflicts which actually occur: therefore, if two (or more) transitions are in structural conflict, but some of them is not enabled, the preselection acts only over the enabled transitions. This behaviour correctly models the Occam ALT composition in which the non-deterministic choice between two or more alternatives takes places *only* among those which may at the moment be executed.

Let us notice that the assignment of suitable values to the weights associated with probabilistic arcs allows the construction of the model either according to the 'theoretic' semantics of Occam - by assigning equal probabilities to each conflicting transition - or reflecting the peculiarities of an implementation of the language in some running environment - by assigning particular probability values - when a greater correspondence of the GSPN model to the running environment is requested. Each implementation in fact solves the non-determinism of the ALT composition by a choice between the possible alternatives and the model can in this way account for it.

Let us also point out that, in general, the obtained GSPN consists of timed transitions only. Therefore the simpler SPN model [3] could be used. The choice of GSPN allows us to use the same kind of timed model even when considering a variant of the basic model, called τ-model (cf. [2,8]), which makes use of unobservable immediate τ-transitions.

3 Program placement analysis

In this section we introduce a net-based notion of program placement, where by *placement* we intend the different possible assignments of the processes of a program to several available processors, disregarding the topology of the underlying phisical net of processors. Within Occam, the placement is determined by the use of the PLACED PAR composition that allows the distribution of a program over several processors assigning each simple or composed process to a particular processor.

3.1 Extending the net model to capture placement

According to [11], the placement of a program modelled by a 1-safe PT net can be defined as follows. Given a 1-safe PT system N=(S,T,W,Mo), a *processor assignment* (or configuration) of N is a pair C=(P,f), where P is a finite set of places, called the processors, satisfying $S \cap P=\emptyset$ and $f:T \rightarrow P$ is a function yielding, for each transition $t \in T$, the place $p=f(t) \in P$ corresponding to the processor which executes the activities represented by t.

Given a placed Occam program Γ and its corresponding GSPN model $\Sigma(\Gamma)=(S,T,W,\Lambda,Mo)$, the placement is represented by a processor assignment C=(P,f) defined as follows:
- the set P contains a place for each processor named within a PLACED PAR;
- the function f associates with each process (transition) contained within any arm of a PLACED PAR the corresponding processor (place).

The GSPN $\Sigma_c(\Gamma)=(S^*,T^*,W^*,\Lambda^*,Mo^*)$ including the placement C=(P,f) is defined extending $\Sigma(\Gamma)$ in the following way: $S^*=S \cup P$; $T^*=T$; $W^*(s,t)=W^*(t,s)=1$ for each $s \in P$, $t \in T$ such that $f(t)=s$; $W^*(s,t)=W(s,t)$ and $W^*(t,s)=W(t,s)$ for each $s \in S$, $t \in T$; $\Lambda^*=\Lambda$; $Mo^*=Mo \cup P$.

Let us notice that a placement increases the number of conflicts in the net to model the competition of processes for processors. These further conflicts require the use of a suitable preselection policy assigning equal probabilities to the probabilistic arcs outgoing from processor places to represent the equal sharing of a processor among several processes (unless otherwise specified). Such policy represents the concurrency reduction due to processor sharing, and then is the basis for distinguishing among the various program placements from the performance evaluation perspective. It is meaningful to consider the set of possible placements $\Omega=\{C_i=(P_i,f_i)\}$ of a system $\Sigma(\Gamma)$ and compare the augmented systems $\Sigma_{c_i}(\Gamma)$ with respect to their execution times, since all of them generate the same set of reachable markings but differ in the degree of

concurrency allowed among their transitions.

3.2 Placement comparison by analysis of the GSPN model

In general several performance indices [3] can be associated with a GSPN; when applied to our case, they yield different performance indices of Occam programs. For instance, the *probability of an event* can be used to determine the probability of execution of a program module, while the *firing frequency* of a transition gives the average number of utilizations of a module; the *average delay in traversing a net* allows an estimation of the average execution time of a program or a module.

Let us consider the *Average Delay in Traversing a Net* in steady-state (ADTN for short), for evaluating the time taken to reach the marking of the exit interface places of the (GSPN) process-box of a program from the marking of its entry interface places. We define a probabilistic average optimality criterion to compare different placements in the following way: let Σ be a GSPN, $q \geq 1$ a natural number (of processors) and $C_1=(P_1,f_1)$ e $C_2=(P_2,f_2)$ two placements of Σ, with $|P_1|=|P_2|=q$, then C_1 is said to be *less time consuming* than C_2 ($C_1 < C_2$) iff $ADTN\Sigma_{C1} < ADTN\Sigma_{C2}$.

To compare the ADTNs of GSPNs, we must recall that GSPN are equivalent to continuous-time Markov Chains (MC) defined over a set of states isomorphic to the set of tangible markings of the net [3]. The equilibrium solutions of this MC provide the probability distributions on the markings of the GSPN leading to the performance indices related to the net behaviour. During the analysis of a GSPN we perform a 'closure' of the net to make its Reachability Graph (RG) strongly connected; this is in fact a sufficient condition to the existence of the steady-state of the equivalent MC. Let us summarize the analysis steps of a GSPN to compute its ADTN:

1) 'closure' of the GSPN by the introduction of a fictitious transition t', whose parameter λ' is known (e.g., $\lambda'=1$), which connects the exit interface places of the net with the entry ones;
2) construction of the RG of the net according to the local preselection policies specified in the model;
3) construction of the *infinitesimal generator* matrix, denoted with Q;
4) computation of the steady-state solution of the model by solving the system of linear equations:
   $\Pi \bullet Q=0$
   $\Sigma \pi_i=1$
   where π_i denotes the steady-state probability of marking M_i and where $\Pi=(\pi_1,\ldots,\pi_n)$ is the equilibrium probability mass function (pmf) over the n reachable markings;
5) computation of the $ADTN=1/\pi_{ex} \cdot \frac{1}{t'}=1/\pi_{ex}-1$, of the net, where π_{ex} is the probability associated with the exit interface of the net and $\frac{1}{t'}=1$ is the average execution time of the activity associated with the introduced transition t'.

3.3 Some application examples

The construction and analysis of the GSPN models of Occam programs are illustrated by the two following examples. The first one mainly shows how the use of the local preselection policy, associated with the possibility of suitable choosing the probability parameters, allows us to adapt the model either to the formal semantics of Occam or to a particular implementation, producing faithful performance eveluations. The second one applies the placement comparison method presented in §3.2 and sketches how the modularity of the net model can be exploited to jump from simple programs to larger ones (this point will be reconsidered in the conclusions).

Example 1

Given the Occam program:

```
PAR
  ALT
    P1
    P2
  P3
```

the corresponding GSPN model is given in Fig.1, where, e.g., we assign to the timed transitions the following rates: $\lambda1=1$, $\lambda2=3$, $\lambda3=2$, $\lambda'=1$ (t' realizes the 'closure' of the net).

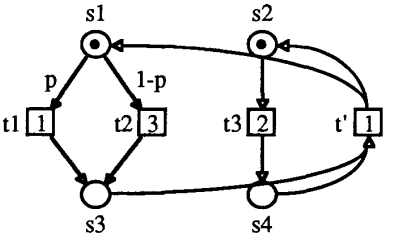

Fig.1

Through the construction of the reachability graph and of its corresponding infinitesimal generator matrix Q, we obtain the steady-state solution and then the ADTN of the net: $ADTN=1/\pi_{ex}-1=(48p+57)/90$. By assuming $p=1/2$ (which corresponds to equiprobable alternatives, according to the formal Occam semantics) we obtain ADTN=0,9; otherwise, according to different implementations of the language, let p assume different values (e.g. p=1; p=0; p=2/3) yielding different ADTN (resp.: 1.16; 0,63; 0,98).

Example 2

Given the Occam program:

```
PAR
  P1
  P2
  P3
```

and two processors PR1 and PR2 available, let us consider the two following placements C1 and C2:

C1:	C2:
PLACED PAR	PLACED PAR
PROCESSOR PR1	PROCESSOR PR1
PAR	P1
P1	PROCESSOR PR2
P2	PAR
PROCESSOR PR2	P2
P3	P3

The two GSPN models, corresponding respectively to C1 and C2, are given in Fig.2. Processes P1, P2 and P3 are respectively modelled by the transitions t1, t2 and t3 associated with the parameters $\lambda 1=1$, $\lambda 2=10/9$ (i.e. we assume that P2 is the Occam process studied in Example 1) and $\lambda 3=3$. In both cases the parameter associated with the closure transition t' is assumed to be $\lambda'=1$. The analysis of the two nets, according to the procedure presented above, yields respectively the values ADTN1=2,14 and ADTN2=1,93, i.e. placement C2 is less time consuming then C1.

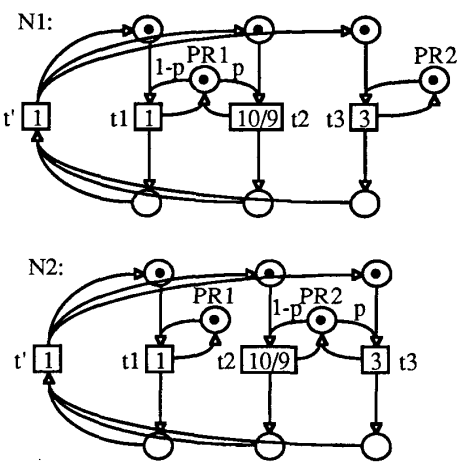

Fig.2

4 Concluding remarks and future work

Starting from the 1-safe net model of Occam, we have developed a GSPN model which supports program placement analysis, i.e. the identification of the best placement among the possible placements of the processes of an Occam program to a given set of processors. Let us notice that to fully achieve this goal, the model should be extended to deal with the topological characteristics of the available net of processors. Future (and automated) applications could be extended, e.g., to the optimization of the level of parallelism for an Occam program, as discussed in [2].

In respect of the applicability of the GSPN model of Occam programs, in the case of real-life examples the size of the (underlying) net model might affect the feasability of the ADTN computation. As a possibility to face with this problem we can exploit the modularity and compositionality features of the basic model. By exploiting the process-box to define suitable net morphisms, one can define not only models in which it exists a bijection among the atomic actions of the program and the net transitions, but also models in which a transition represents a program module, i.e., in Occam, a non-atomic process. The delay to be associated with such a, let's say, non-atomic transition, is the outcome of the computation of the ADTN on the net modelling the corresponding non-atomic process; these partial results may then be used to obtain a quantitative analysis of the whole program, as sketched in Example 2 above. The precise definition ot this technique is straightforward, but for two aspects we would deserve it a special attention: (i) the handling of the PRI PAR compositions, which has been recognized as intrinsecally non-compositional [2]; and (ii) the definition of the morphism when communication transitions are involved.

We finally notice that, when real-life examples are considered, both the qualitative analysis of the PT model and the quantitative analysis of the GSPN one, heavily relies on the availability of software tools [12] for the model construction and solution.

Acknowledgements

The authors wish to thank Eike Best, Jon Hall and Richard Hopkins with whom the PT net model has been developed, and Marco Ajmone Marsan for the fruitful discussions and his valuable suggestions on the GSPN model. This research has been carried on within the ESPRIT BRA project #3148 DEMON (DEsign Methods based On Nets) [13].

References

[1] O. Botti, J. Hall, R. Hopkins, A Petri Net Semantics of Occam2, ESPRIT BRA 3148, DEMON, Technical Report n.80 (1990).

[2] O. Botti, *Un modello in reti di Petri per Occam2*, Tesi di laurea, Dipartimento di Scienze dell'Informazione, Università di Milano, (A.A. 1990/91).

[3] M. Ajmone Marsan et al., *Generalized Stochastic Petri Nets: Definition at the Net Level*, (submitted for publication).

[4] S. Bokhari, *On the Mapping Problem*, in IEEE Transactions on Computers, Vol.30, n.3 (1981).

[5] O. Kraemer, H. Muehlenbein, *Mapping Strategies in Message Based Multiprocessor Systems*, in P. Treleaven, C. Nijman, J. De Bakker (eds.), Proc. of PARLE, LNCS Vol.258, Springer (1987).

[6] INMOS ltd., *Occam2 Reference Manual*, Prentice Hall (1988).

[7] E. Best, *The Petri Box Calculus for Concurrent Programs*, in E. Best, G. Rozenberg (eds.), '3rd Workshop on Concurrency and Compositionality', GMD Studien n.191, (1991).

[8] O. Botti, F. De Cindio, *Some Remarks about a Petri Net Model of Occam2 using t-transitons*, ESPRIT BRA 3148, DEMON, Technical Report n.81 (1990).

[9] E. Best, M. Koutny, *Partial Order Semantics of Priority Systems*, ESPRIT BRA 3148, DEMON, Technical Report n.82 (1990). Also to appear in TCS.

[10] M. Ajmone Marsan et al., *The Effect of Execution Policies on the Semantics and Analysis of Stochastic Petri Nets*, in IEEE TSE, Vol.15, n.7 (1989).

[11] E. Best, *Weighted Basic Modular Nets*, in F.H. Vogt (ed.) 'Concurrency 88', LNCS Vol.335, Springer Verlag (1988).

[12] G. Chiola, *A Graphical Petri Net Tool for Performance Analysis*, in Proc. of the 3rd Int. Workshop on Modelling Techniques and Performance Evaluation,AFCET, Paris, 1987.

[13] E. Best, *Design Methods Based on Nets*, in G. Rozenberg (ed.), 'Advances in Petri Nets 89', LNCS Vol.424, Springer Verlag (1989).