Journal of Applied Sciences

ISSN 1812-5654

science alert

ANSInet
an open access publisher
http://ansinet.com

Journal of Applied Sciences 8 (12): 2286-2292, 2008
ISSN 1812-5654
© 2008 Asian Network for Scientific Information

A Semantics for the Control Part of Lotos

R. Mekki and B. Messabih
Université des Sciences et de la Technologie Mohamed Boudiaf,
B.P. 1505, El Mnaouer, Oran, Algeria

Abstract: In this study it is proposed a formal semantics for Basic LOTOS (Language Of Temporal Ordering Specification). The subset of LOTOS, where processes interact with each other by pure synchronizations, without exchanging values. In basic LOTOS the expressiveness of all the LOTOS process constructors (operators) can be appreciated without being distracted by interprocess value communication. LOTOS is an FDT generally applicable to distributed, concurrent information processing systems. During the last decade, a lot of works have been devoted to compilation and verification of LOTOS specifications. While using extended Petri nets as tool for compile a subset of LOTOS has already been pointed out. In this research it is proposed to extensively make use of a specific kind of high level Petri nets: the M-nets. Such nets, allowing for compositionality, appear particularly well-suited to give a formal semantics for basic LOTOS.

Key words: Specification, Petri nets, M-nets, compositionality, semantics, LOTOS

INTRODUCTION

LOTOS (Garavel, 1990) is one of the Formal Description Techniques FDT developed within ISO for the formal specification of open distributed systems. In LOTOS a distributed, concurrent system is seen as a process, possibly consisting of several sub-processes. A LOTOS specification describes a system via a hierarchy of process definitions. A process is an entity able to perform internal, unobservable actions and to interact with other processes, which form its environment. Basic LOTOS is a simplified version of the language employing a finite alphabet of observable actions. This is so because observable actions in basic LOTOS are identified only by the name of the gate where they are offered and LOTOS specifications can only have a finite number of gates.

The M-nets calculus (Best *et al*., 1995, 1999) defined as the high level or coloured version of the Petri Box Calculus (PBC), is widely accepted to give semantics to concurrent systems and programming languages like B(PN)^2 and SDL, cf. (Lilius and Pelz, 1996; Fleischhack and Grahlman, 1997; Klaudel and Reamann, 1995; Best *et al*., 1999). The most original aspect of M-nets is their full compositionality thanks to their interfaces and a set of various net operations defined semantics for them. In fact, M-nets constitute like PBC a net algebra. Their interest is augmented by the ability to use in practice an associated tool PEP (Programming Environment based on Petri nets) (Grahlman, 1997), which also offers various implemented verification and analysis methods. Whenever the system to be modeled consists of a lot of distinct conceptual parts, which need to be combined or coordinated in a non trivial way, a very modular and compositional proceeding is necessary to be able to control the correctness of modeling (BuiThanh and Klaudel, 2003, 2004). M-nets just offer these features. We have chosen to translate basic LOTOS specifications into the algebra of modular high-level Petri nets: the M-nets (Best *et al*., 1995, 1999; Klaudel, 1995). Due to the rich set of composition and communication operators we are able to define a semantic operator on M-nets for each syntactic operator of basic LOTOS. Using our approach a property of a basic LOTOS specification is checked as follows:

- The M-net semantics of the basic LOTOS specification is calculated
- The M-net is unfolded into a Petri box (a special low-level net)
- The basic LOTOS property is transformed into a net property
- The net property is checked against the Petri box
- The result is transformed back to the basic LOTOS level

BASIC LOTOS SPECIFICATION

Systems and their components are represented in LOTOS by processes. A specification is the process that represents the whole system being specified. A process displays an observable behaviour to its environment in term of permitted sequences of observable actions (Garavel, 1990). A process appears as a black box to its

Corresponding Author: Rachida Mekki, B.P. 1505, El Mnaouer, Oran, Algeria Tel: 213 7 72 93 32 52

environment since the environment has no information about its internal structure and mechanisms. Processes may have a local definitions (after the keyword where) in which other processes can be defined. LOTOS has scoping rules similar to block-structured programming language. The structure of basic LOTOS specification is:

```
specification   spec-name [gate list]: functionality
                behaviour expression
where
                process definitions
endspec
```

where gate list is a finite alphabet of gate names (or observable actions). A behaviour expression is built by applying an operator (e.g., '[]') to other behaviour expressions. A behaviour expression may also include instantiations of other processes whose definitions are provided in the where clause following the expression. The complete list of basic LOTOS behaviour expressions is given in Table 1, which includes all basic LOTOS operators Symbols B, B_1, B_2 stand any behaviour expression; G, G_1, G2,...,G_n are elements of a finite alphabet gate list.

Remark: As we will see farther, the parallel operator '||' take an important part in the M-nets algebra; thus, in order to avoid confusion we reserve this notation for the composition of M-nets. In the sequel, for LOTOS, the notation for the operator of full synchronization will be $|[G^+]|$ instead of || where G^+ is the set of all gates which are defined in both behaviours or both processes.

M-nets: The M-nets algebra was introduced in (Best *et al*., 1995, 1999) as an abstract and flexible metalanguage for the definition of semantics of concurrent programming languages. These allow (as usually composition operators in process algebras do) the compositional construction of complex nets from simple ones, thereby satisfying various algebraic properties. the main difference between M-nets and coloured nets is that M-nets carry additional information on their places and transitions to support composition operations. Besides annotations on places (set of allowed tokens), arcs (multisets of variables and transitions (occurrence conditions), M-nets have, additionally, labels on places and labels on transitions. The labels on transitions denote communication capabilities similar to action symbols in a CCS term, respectively indicate their hierarchical nature. The labels on places denote their status as entry (part of the initial marking), exit (part of the final marking) or internal (otherwise). Thus, every place and transition of an M-net carries an inscription of the form inscription = (label | annotation).

Table 1: Syntax of behaviour expressions in basic LOTOS

Nom	Syntaxe
Inaction	Stop
Termination avec succés	Exit
Préfixage	
Action interne	i;B
Action observable	G;B
Choix	B_1[] B_2
Composition parallèle	
Cas général	B_1|[G_1, ... , G_n]|
Entrelacement	B_1|||B_2
Abstraction	hide G_1, ... , G_n in B
Composition	B_1>>B_2
Préemption	B_1[>B_2
Instanciation de processus	P[G_1, ... , G_n]

Auxiliary definitions: Let Val be a fixed and suitably large set of values, Var, resp. X sets of variables, resp. hierarchical variables, the latter ones being capital letters and Par a set of parameters. We assume the existence of a fixed but sufficiently large set A of parametrised action symbols. Each action A ϵ A is assumed to have an arity ar(A) which is a natural number describing the number of its parameters. The elements of A are grouped into pairwise conjugates by the bijections: A → A, called conjugation, satisfying: $\forall\ A \in A : \bar{A} \neq A, \bar{\bar{A}} = A$ and $ar(A) = ar(\bar{A})$

It is also assumed the existence of a fixed but sufficiently large set G of action gate symbols without conjugation. Each action gate symbol G ∈ G is assumed to have an arbitrary arity ar(G) An action term is, by definition, a construct $A(A(\tau_1,\ldots,\tau_{ar(a)}, G[G_1,\ldots,G_{ar(A)}]$ where $A \in A$ and $\tau_j \in Var \cup val$ for $1 \leq I \leq ar(A)$; $G \in G$ and $G_i \in$ gatelist for $1 \leq i \leq ar(G)$.

Definition of M-nets: An M-net is a triple (P, T, λ) such that P is a set of places, T a set of transitions with $P \cap T = \emptyset$, λ is an inscription function with domain $P \cup (P \times T) \cup (T \times P) \cup T$, such that:

- For every place $p \in P$ $\lambda(p)$ (α_p, β_p), where $\alpha(p) \in \{a,i,x\}$ is called the label or status of p and $\beta(p)$, the type of p, is a nonempty set of values from Val.
- For every transition $t \in T$ $\lambda(t)$ is a pair (α_t, β_t) where α_t, the label of t, is a finite multiset of action terms (t will then be called a communication transition), or a hierarchical action symbol, i.e., $\alpha(t) \in X$, (t will then be called a hierarchical transition); $\beta(t)$, the gard of t, is a multiset of terms over Val and Var.
- For every arc $(p,t) \in (P \times T)$, $\lambda(p,t)$ is a finite multiset of variables from Var and values respecting the type of the adjacent place p, (analogous for arcs (t, p) $\in (T \times P)$); the meaning of $\lambda(p,t) = \emptyset$ is that there is no arc leading from p to t (Benzaken *et al*., 1998).

Operations on M-nets: The composition operators on M-nets as defined in (Best *et al*., 1995; Klaudel, 1995) are two different kinds: those concerning place interfaces and thus the flow control (which comprise sequential composition ;, choice [], parallel composition || and iteration *) and those concerning transition interfaces and thus capabilities for communication comprising synchronization and restriction.

The intuitive idea behind the synchronisation operation of an M-net N consists of a conglomeration of certain basic synchronisations over actions terms pairs (A(.,.) and Ā(.,.) and yield a new transitions in N (Best *et al*., 1999). The operation will be defined as follows: N sy A, where $N = (P,T,\lambda)$ is a given M-net and A is a given action symbol. The communication is performed by a most general unifier which renames the variables in the action terms appropriately (Klaudel, 1995). The restriction operation N rs A removes from N all transitions that mention either A or Ā and hence all synchronisation capabilities for A.

Scoping is a mixed operation, accepting an M-net and a set of communication action symbols; it is defined by the synchronisation followed by the restriction of the net over the set of action symbols: N' = (N sy A) rs A. This scoping mechanism is used for block structuring.

M-net model extensions: In order to adapt the M-net model for a basic LOTOS compositional semantics, the operations on M-nets must be extended to some other operators such that:

- The M-net refinement (Reimann *et al*., 2003; Devillers *et al*., 2003; Klaudel and Riemann, 1998): $N[X_i \leftarrow N_i | i \in I]$ allowing a hierarchical construction of M-net, means N where all X_i-labeled (hierarchical) | transitions are refined into (i.e., replaced by a copy of) N_i, for each i in the indexing set I. However, general refinement (Riemann *et al*., 2003) is not sufficient for our approach; in fact, knowing that a LOTOS system is described via a hierarchy of process definitions it must has a possibility to distinguish different instantiations of a process definitions and their behaviours. In order to satisfy this requirement, it is used the general parametrised refinement concept (Devillers *et al*., 2003). The parameter mechanism is the mean by which a refining net interact with the area of the refined transition.

  In this way, a formal parameter is add to hierarchical transition which deals with the concrete parameters of the call and return transitions labels that we see later.
- The synchronization-lotos (Mekki, 2000) sy_{lotos} is defined such that:

  Let two M-nets $N_1 = (P_1,T_1,\lambda_1)$ and $N_2 = (P_2,T_2,\lambda_2)$ and let G an action gate symbol.

  $(N_1 \| N_2) sy_{lotos} = (P,T\lambda)$ where:

  - $P = P_1 \cup P_2$

  - $T = T_1 \cup T_2 \cup T_{sy} \backslash (T_{G1} \cup T_{G2})$ where T_{sy} is the set of transitions t_{sy} arising through a basic synchronisation-lotos out of t_1 and t_2 over G such that:

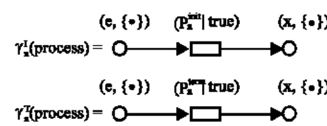
  $t_1 \in T_{G1}$, $t_2 \in T_{G2}$ and $T_{G1} = T_1\Gamma_G$, $T_{G2} = T_2\Gamma_G$

  - $\lambda = (\lambda_1 \cup \lambda_2)\Gamma_{P^\wedge T)}$ where $\alpha(t_{sy}) = \alpha(t_1) = \alpha(t_2)$
- The relabelling function (Mekki, 2000): $\prod^{p}{}_{gatelist}$ which is applied to an M-net N, is parametrised with a name of process definition (P) and a list of gate names (gate-list). It does the following substitution for each gate name $G_i \in$ gate-list in N.

Semantics of a basic LOTOS specification: A semantics of basic LOTOS specification will be defined associating one M-net to each process definition (avowed in the where clause) its behaviour expression.

The initialization and clearing of each process definition The M-net of the last two kinds have to be composed sequentially with each other to enable later scoping over (created and terminated) processes. Finally, all these M-nets are put in parallel and the synchronization and restriction ovet all action names in the process definitions will insure the scoping. Thus, we get above global semantic formula:

M−net(bloc) = [M−net(process definitions) $\|(\Gamma^I$ (processes); M−net(behaviour expression; Γ^T(processes))] sy Γ(processes) rs Γ (process)



With

$$\Gamma^I(\text{processes}) = \|_{n=1}^{k} \gamma_n^I(\text{process}) \text{ and}$$

$$\Gamma^T(\text{processes}) = \|_{n=1}^{k} \gamma_n^T(\text{process}) \text{ and}$$

where, $\gamma_n{}^I$ (processes) and $\gamma_n{}^T$ (processes) are the auxiliary M-nets for initialization and termination of scoping.

In the sequel, the other M-nets appearing in the above global semantic formula will be precisely defined as well as the considered sets of action symbols.

Semantics of process definition: A process definition is similar to a procedure declaration in a programming language like Pascal. Let us first examine a single

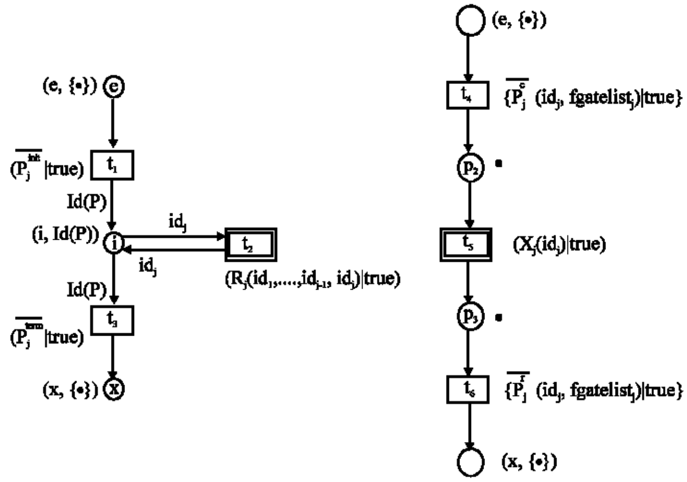


Fig. 1: Semantic components of process definition P_j

instantiation of a process. It is easy that it consists of a call transition, a return transition and a body. The call transition is labeled with $\overline{P}_c$ while the return transition is then correspondingly labeled with $\overline{P}_r$. The control flow in the body is represented by the black to which moves through the body representing the different actions that the process executes. The call of the process is equivalent to the synchronization with the P_c of the M-net for instantiation (which will be defined later, cf. Fig. 4). However if several instantiations of the process are active at the same time, this simple scheme does not work. The reason for this is, that since the control tokens have no identity, we may not distinguish between them and so the causal relations between the control and the process instantiations may be broken. We propose to solve the problem by using colored tokens (id) as indexes to distinguish between the instantiations, allowing to fold the different instantiations of the process into one M-net. Thus in the M-net construction of process definition semantics (Fig. 1), we allow Id (p) (the instantiation set) as type for internal places and id as concrete parameter for actions of hierarchic transition t_2.

In the sequel, it is supposed that the set $ID(P) \subseteq Val$ of instantiation identifiers is given. As mentioned in the introduction, generally a specification LOTOS is a hierarchy of processes. Therefore, it may has overlapping process definitions. Hence, in order to deal with process definition formally, we will denote by P_j a process definition identifier at the jth level of overlap (with $1 \leq \leq m$ where m is the maximal depth of overlap).

The box M-net (procdef) of a process definition declared at the level of overlap depth j is obtained by successive refinements in the operator M-net.

N_I N_{I_j} (id_1:Id(P),...,idj_{-1}:Id(P)) of semantic components of P_j where:

N_{I_j} (id_1:Id(P),...,id_{j-1}:Id(P)) in charge for the management of instanciations P_j;

$N_{F_j}(\overline{id} : ID(P))$ in charge for the initialization (call) and the termination (return) of each instanciation of P_j;

In the box $N_{F_j}(\overline{id} : ID(P))$, the transition labeled χ_j is a hierarchical transition which is refined by the representative net of process definition body: $NBP_j(id_j)$. The event of the transition t_4 carried out by synchronization with the transition labeled by its complementary action belonging to the instanciation M-net (see further) of the process according to the behaviour expression of the block; thus the list of the formal gates fgatelist in the net.

$$N_{F_j}(\overline{id : Id(P))[\chi_j(id_j)} \leftarrow N_{BP_j}(id_j)]$$

must be renamed in list of effective gates belonging instanciation Pcall M-net.

This renaming is obtained using the function $Pi^{p}_{gate-list}$. Finally, after refinement in $N_{F_j}(\overline{id}{:}Id(P))$, we obtains:

$$N_{R_j}(\overline{id}{:}Id(P)) = N_{F_j}(\overline{id}{:}Id(P))[\chi_j(id_j) \leftarrow N_{BP_j}(id_j)]\Pi^{P}_{actgatelist_j}$$

M-net(procdef} is obtained from the following expression:

$$M-net(procdef\ P_j[gatelist] = N_{I_j}(id{:}Id(P),...,id_{j-1} : Id(P))$$
$$[R_j \leftarrow N_{R_j}(\overline{id} : Id(P))]$$

In a basic LOTOS specification or process, the declaration semantics of many process definitions with same overlap level is given by the following formula:

$$\text{M-net(process definitions)} = \prod\nolimits^{k}_{n=1} P_n$$

and the action symbols for scoping are:

$$\Gamma(\text{process}) = \bigcup_{n=1}^{k} \gamma_n(\text{process})$$

where:

$$\gamma_n(\text{process}) = \{P_n^{\,c}, P_n^{\,r}, P_n^{\,init}, P_n^{\,term}\}$$

with

$$ar(P_n^{\,c}) = ar(P_n^{\,r}) = 1$$

and

$$ar(P_n^{\,init} = ar(P_n^{\,term}) = 0$$

Behaviour expression M-net: An essential component of a specification is its behaviour expression where the observable and intern behaviours are defined. It is the specification behaviour obtained by combination of the behaviour operators (sequential composition, parallel composition, ...

In the sequel, the M-nets semantics of each operator (Table 1) is given.

Inaction specified by stop, models a situation in which a process is unable to interact with its environment. Inaction can be used to describe deadlock (Fig. 2).

Successful termination: In LOTOS exit is a nullary operator. It is equivalent to the following behaviour: δ; stop, this sequential composition is an interaction with the special successful termination gate δ, which ever appears explicitly in LOTOS program, followed by a stop operator Fig. 2.

Action prefix:: The semantics of action prefix behaviour expression B is:

M-net (G ; B) = M-net(G); M-net(B) where M-net(G) is given in Fig. 2.

Choice: M-net(B_1[]B_2 = M-net (B_1) M-net(B_2)

Parallel composition: LOTOS has three kinds of parallel operators (Table 1):

- General case $B_1|[\{G_1,...G_2\} \cup \{\delta\}]|B_2$ where $\{G_1,...,G_m\}$ a set of user-defined gates called synchronization gates of B_1 and B_2
- Pure interleaving $B_1|[G]|B_2$ when the set of synchronization gates is only δ.
- Full synchronization $B_1|[G]|B_2$ when the set of synchronization gates is the set of all gates appearing in both behaviour expressions including δ. The M-nets semantics of parallel operators is carried out in two stages: Concurrency is expressed applying the parallel operator of M-nets algebra (||).

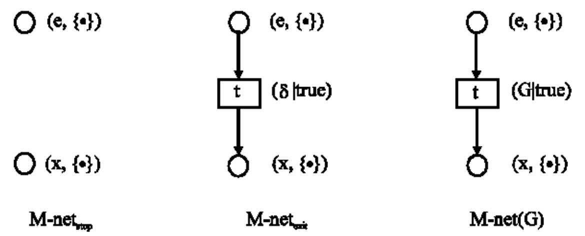


Fig. 2: M-net$_{stop}$, M-net$_{exit}$ and M-net of action over the gate G

Synchronization and communication LOTOS are expressed by coupling of all the committed transitions in a same event. Two transitions t_1 and t_2 belonging respectively to M-nets B_1 and B_2 must synchronize through sy_{lotos} if they have in their respective labels the same action gate symbol belonging to the set of synchronization gates of specified operator, either $[G_1,...,G_2]$ for the first case and [G] for the second. Thus, the semantics of this three operators is:

$$\begin{aligned} & M-net(B_1\,|[G_1,\ldots,G_m]|\,B_2) = (M-net(B_1)|| \\ & M-net(B_2))sy_{lotos}\{(G_1 \cup,\ldots,\cup G_m) \cup \delta\} \end{aligned}$$

$$M-net(B_1\,|||\,B_2) = (M-net(B_1)||M-net(B_2))sy_{lotos}\{\delta\}$$

$$\begin{aligned} & M-net(B_1\,|[G]|\,B_2) = (M-net(B_1)|| \\ & M-net(B_2))sy_{lotos}\{(G_1 \cup,\ldots,\cup G_n) \cup \delta\} \end{aligned}$$

with $[G_1,...,G_n]$ = [gatelist]

Sequential composition:

$$M-net(B_1 >> B_2) = M-net(B_1)\,;\,M-net(B_2)$$

Hiding: The operator hide ... in allows one to transform some observable actions of a process into unobservable ones. The M-nets semantics for the hiding operator consist of a renaming of all hidden gates by i (unobservable action) (Best *et al.*, 1999) over which no more interactions are possible.

M−net(hide $G_1,...,G_r$ in B) = M−net(B) where $G_1,...,G_r$ are renamed by i.

Disabling: $B_1[>B_2$. In almost any OSI connection oriented protocol or service.

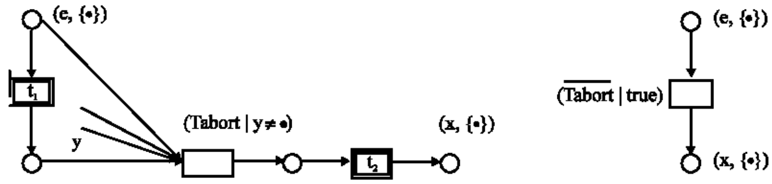


Fig. 3: M-net(TAbort) and M-net(Abort)

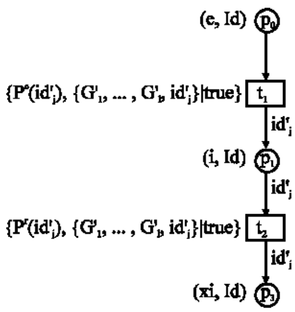


Fig. 4: Process instantiation: Pcall

It is the case that the normal course of action can be disrupted at any point in time by events signaling disconnection or abortion of a connection. This has led to the definition in basic LOTOS of an application generated operator namely the disabling operator. To illustrate such a disruption, we use a particularly M-net called M-net (TAbort) which contains a special kind of transition namely TAbort with an arbitrary (according to the whole of disrupted M-net places) number of entry-places and one exit-place. This transition is able to take into account this aborting by eliminating all token places of the disrupted behaviour net.

$M\langle \text{T Abort} \rangle M'$ with $\forall p \in {}^{\bullet}t$:
$M'(p) = 0$ and
$M(p) \geq 0$

The semantics of disabling is expressed by the following scoping:

$$\begin{aligned}&M-\text{net}(B_1[> B_2) = M-\text{net}((B'_1; \text{exit})\\&[> B_2) = M-\text{net}(\text{TAbort})[t_1 \leftarrow M-\text{net}(B'_1),\\&t_2 \leftarrow M-\text{net}(B_2)] \,||\, M-\text{net}(\text{Abort})\text{sy TAbort rs}\\&\text{TAbort}[]M-\text{net}(B'_1); M-\text{net}_{\text{exit}}\end{aligned}$$

In Fig. 3, t_1 and t_2 are some hierarchical transitions which may be refined by the M-nets of B'_1 and B_2 behaviours respectively such that $B_1 = B'_1$; exit.

According to the refinement definition and disabling operator definition, $y \neq \bullet$ means that the exit-places of the refining M-net of T_1 are not all marked (i.e., to the less one among them is empty).

Moreover, TAbort must have all the places of t_1 refined in its entrance. Then, in the M-net which illustrate B'_1 every entry or internal-place must have a dual status {x,e} or {x,i}, respectively.

Process instanciation: The standard form of a process instantiation is the synchronous instantiation. The following M-net is inserted for each process instantiation. The process is called by transition t_1 after which the caller waits in place p_1 until the process returns, at which point t_2 is fired. The M-nets semantics is expressed by: $M-\text{net}(P[G'_1,...,G'_l]) = \text{Pcall}$.

CONCLUSION

Through the M-nets Algebra, a high level metalanguage and among simplest, we have presented a compositional semantics of a basic LOTOS specification. We have extensively made use of the modularity features of the M-net calculus and established thus a fully compositional model.

In this study it has only treated a simplified version of the language called basic LOTOS, while full LOTOS or simply LOTOS includes data structures that are derived from abstract data type language ACT ONE. The M-net formalism is insufficient for data type specification and another class of high level Petri-Boxes: A-nets allow such a specification. Thus, the future work in this aim should be a formal semantics for full LOTOS with A-nets.

The tool of verification and simulation for a composed M-nets is the PEP (Programming Environment based on Petri-nets) system, it's an interactive system allowing a visualization of some complete, partial or step by step executions.

REFERENCES

Benzaken, V., N. Hugon, H. Klaudel, E. Pelz and R. Riemann, 1998. M-net calculus based semantics for triggers. Proceedings of the 19th International Conference on Application and Theory of Petri Nets, LNCS 1420. June 1998, Springer-Verlag London, UK., pp: 306-325.

Best, E., H. Fleischhack, W. Fraczak, R.P. Hopkins, H. Klaudel and E. Pelz, 1995. A Class of Composable High Level Nets. In: Application and Theory of Petri Nets, LNCS 935, De Michelis, G. and M. Diaz (Eds.). Springer, Torino, pp: 103-120.

Best, E., W. Fraczak, R.P. Hopkins, H. Klaudel and E. Pelz, 1999. M-nets: An algebra of high-level Petri nets with an application to the semantics of concurrent programming languages. Acta Informatica, 35: 813-857.

BuiThanh, C. and H. Klaudel, 2003. Encapsulation in an object-oriented notation based on modular Petri Nets. In: Simulation with Petri Nets, Workshop European Simulation and Modelling Conference, October 27-29 ESMC'2003. www.ibisc.Univ-evry.fr/pub/basilic/out/Publications/2003/Bk03/.

BuiThanh, C. and H. Klaudel, 2004. Object-Oriented Modeling with High Level Modular Peti Nets: Integrating Formal Methods. LNCS 2999, Springer, Berlin/Heidelberg, Germany, pp: 287-306.

Devillers, R., H. Klaudel and R.C. Riemann, 2003. General parametrised refinement and recursion for the M-net calculus. Theor. Comput. Sci., 300: 259-300.

Fleischhack, H. and B. Grahlmann, 1997. A compositional Petri net semantics for SDL. Technical report. Universität Hildesheim.

Garavel, H., 1990. Introduction to LOTOS. ftp://ftp.inrialpes.fr/pub/vasy/publications/cadp/Garavel-90-b.pdf.

Grahlman, B., 1997. The PEP Tool. In: Computer Aided Verification Lecture Notes in Computer Science, Grumberg, O., (Ed.). Springer-Verlag, UK., pp: 440-443.

International Programme on Chemical Safety, 1985. Environmental Health Criteria 46. Guidelines for the study of genetic effects in human population. WHO, Geneva, pp: 25-54.

Klaudel, H., 1995. Algebraic models, based on the Petri nets for the semantics of concurrent programming languages. Ph.D Thesis, University Paris-Sud, Orsay.

Klaudel, H. and R.C. Riemann, 1998. Application of general refinement to the semantics of a parallel programing language. Rapport de recherche, L.R.I Université de Paris-Sud.

Mekki, R.H., 2000. A compositional semantics for basic LOTOS via a class of high level Petri Nets. Magister Thesis, Computer Science Department, USTO.