A SMoLCS Based Kit for Defining High-Level Algebraic Petri Nets *

Mohamed Bettaz[1] and Gianna Reggio[2]

[1] Institut D'Informatique Universite De Constantine – Algerie
[2] Dipartimento di Informatica e Scienze dell'Informazione
Università di Genova – Italy

Introduction

Petri nets are a description method for concurrent systems supported by graphic representations; their combination with abstract data types (shortly adt's) appear to be motivated by the need to explicitly specify complex data structures inherent to real systems and concrete applications.

Nowadays there is a large amount of proposals for combining Petri nets with adt's, see e.g. [5, 6, 8, 10, 12, 13, 1, 16, 17], concerning either variants of classical net or nets with new more complex concurrent features (as true concurrent Petri nets). The two formalisms may be linked in various ways. The following proposals are often found in the literature; an "algebraic net" is:

- a net schema (defined as usual) plus marking and labelling functions defined over concrete algebras;
- a net schema (defined as usual) plus marking and labelling functions defined over algebras abstractly denoted by algebraic specifications;
- a combination of a net schema and of the data aspects defined using a pure algebraic approach.

Here we use the words "high-level algebraic Petri net" (shortly HLA net) in the third case, i.e. for those nets using concepts from the field of algebraic specifications for defining labellings, markings and the net behaviour.

Our aim is to develop a "kit", i.e. a set of coordinated theoretical tools for formally defining varieties of HLA nets together with their *dynamic behaviour* (the allowed sequences of transition firings), called in the following *token game*, for a large class of varieties. Clearly also non-high-level and/or non-algebraic varieties of nets could be defined, since obviously black/coloured tokens and inscriptions consisting in natural numbers could be algebraically specified.

Precisely, using the kit, we characterize a particular variety of nets, say XYZ, by appropriate parameters (e.g. defining the form of the transitions, how to choose the transitions to fire, ...), and a particular net of such variety with an initial marking, say N, by other parameters (e.g. defining its tokens, places, transitions, ...); then we build a parametric specification that instantiated with such parameters returns the token game of N. All that is graphically represented in Fig. 1. Moreover the kit can

* This work has been supported by "Progetto Finalizzato Sistemi Informatici e Calcolo Parallelo" of C.N.R. (Italy) and by Esprit-BRA W.G. COMPASS n. 6112.

be extended to cope also with semantics over nets; unfortunately for lack of room we cannot present here this part, see [11].

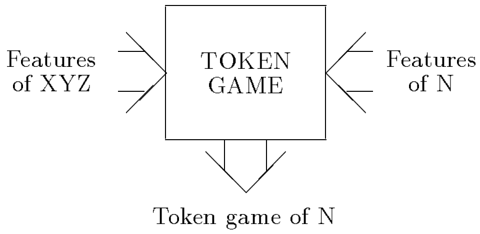

Fig. 1. A schematic view of the kit

The kit is based on the SMoLCS methodology for the specification of concurrent systems ([3, 4]). SMoLCS has been proved very suitable to this aim since it is an integrate method for specifying processes and adt's and offers the possibility of giving modular and parametric process specifications. Moreover SMoLCS is not based either on a particular kind of concurrency (e.g. interleaving plus binary handshaking communications) or on a particular semantics for the process (e.g. strong bisimulation), but it allows to directly specify how the process components of a system interact among them and which processes should be considered semantically equivalent.

We expect that using this kit it is possible:

- To define a large class of varieties of HLA nets, also the one with non-standard concurrent features, as those of [10]. Since, it is easy to see classical (set based) nets as HLA ones, the kit could be used also for defining classical nets with new concurrent features.
- To relate and compare in a unique setting different varieties of HLA nets.
- To associate with existing varieties of Petri nets a corresponding HLA version; and so, perhaps, to define new varieties of interesting Petri nets.
- To use the software tools, developed for the SMoLCS specifications (see [2]), for the rapid prototyping new (under development) HLA varieties; thus helping also the task of tuning the definition of such new varieties.

This work of defining Petri nets using SMoLCS brings together two different models for concurrency: the Petri nets and the structured concurrent labelled transition systems used by SMoLCS, and so it could also be a starting point for studying the relationships between them; for example [9] presents a specification methodology based on this work which allows to refine in a canonical way a system specified by a Petri net into a distributed realization described using SMoLCS.

Also Meseguer in [15] presents an overall algebraic definition of a particular variety of (coloured) Petri nets using rewriting logic; but there the emphasis is different, he is more interested in showing the generality of his specification formalism than in defining algebraically relevant varieties of nets.

Sect. 1 is a short overview on (HLA) nets, while in Sect. 2 we briefly present the SMoLCS methodology and in Sect. 3 the kit. For lack of room here we cannot show all interesting applications of the kit, in Sect. 4 we just show how to define using the kit an existing variety of HLA nets the CATNets of [10] and a new variety of HLA nets with priorities.

1 Summary of High-Level Algebraic Petri Nets

Petri nets have been developed during the years 1960-62 by C.A. Petri for modelling notions relative to synchronized actions. In the beginning, mainly Place/Transition nets using indistinguishable tokens ("black dots") have been studied.

From a formal point of view a Place/Transition net is an oriented bipartite graph. The set of vertices of such graph is given by $P \cup T$, where P is a set of objects named *places* and T a set of objects named *transitions*. The set of arcs is contained in $(P \times T) \cup (T \times P)$. Each place is annotated by a value representing the number of tokens associated with it and called its *marking*. An arc joining a place with a transition is annotated by a value representing the (input) place minimal marking necessary for enabling the transition. This value also defines (implicitly) the number of tokens that have to be removed if the transition is actually fired. An arc joining a transition with a place is annotated by a value representing the number of tokens that have to be added to the (output) place when the transition is fired.

Given a Place/Transition net with an initial marking, the sequences of firings of of the various enabled groups of transitions is said the *token game* of the net.

Although Place/Transition nets have sufficient power for modelling the control structures of most of the practical systems they are not adequate for modelling systems handling elaborated data structures. This led to developing a number of hybrid net/data models, associating a set of variables with the net and/or attributes with the tokens, which may be modified by transitions firings. The main drawback of hybrid models is that extensions are not conform to the spirit of Petri nets, and that analysis techniques and tools developed for simple nets are no longer applicable. This is probably behind the ideas which have led to developing the so called *high-level* Petri nets in which tokens are data items and arcs and transitions are inscribed with symbolic expressions. The earliest of these are Predicate/Transition nets and Coloured nets. In a coloured net transition firing depends not only on the amount of tokens present in input places but also on their colours. This restriction concerns also removed tokens as well as added ones. A Predicate/Transition net may be seen as a schema of Coloured nets, in the sense that inscriptions are defined at a more abstract level, like for instance defining colour "filtering" by equations associated with transitions.

The main reason for the success of high-level nets is probably that they make it possible to achieve more compact specifications while keeping the possibility of using a wide variety of analysis techniques.

The advent of HLA nets may be seen as a new approach combining the strengths of Petri nets with those of abstract data types. However many of the concepts devised during several years to cope with data structuring in Petri nets, turn out to be representable by well-established concepts from the field of algebraic specifications.

This section is not devoted to recalling the various varieties of HLA nets (see e.g.: [10, 16, 5, 13, 17, 12]). The reader interested in a more detailed study of such nets may, for instance, consult [1].

Let us also remark that, like in Predicate/Transition nets, in HLA nets not only places and arcs are annotated but also transitions. In [12], for instance transitions are annotated by boolean expressions named transition conditions, while in [13] transitions are annotated by equations. Moreover places, transitions and arcs may be annotated not only by single values or expressions but also by sets of inscriptions. In [10] for instance, an input arc is annotated by two distinct inscriptions defining respectively the tokens enabling a transition and those removed by the transition firing in an explicit way. In several varieties of nets, sets of inscriptions are associated with places in order to define their markings, their capacities and even capacities w.r.t. types of markings. According to these extensions a transition becomes enabled when some condition (determined by the inscriptions) on the input places marking is satisfied, the condition associated with the transition is true, and the capacity of the output places is not exceeded by transition firing.

Another feature is the level of abstraction used in syntactic notations. In certain varieties of nets, the marking and the various labelling functions are defined over abstract data types (isomorphim classes of algebras), usually given by an algebraic specification, thus leading to specifications of classes of systems. In other nets the mentioned functions are defined over concrete algebras rather suitable for defining concrete systems.

The several variants of HLA nets may also be distinguished by the way in which their dynamic behaviour is defined. It is worth mentioning that, depending on the level of abstraction used in the syntactic notation, this behaviour is usually defined in one of the following three ways; by defining firing rules similar to those of "low-level" (usual) Petri nets, by defining proper firing rules, or via interpretation into semantically oriented models.

Thus, summarizing, in general, we assume that for defining a HLA net of a certain variety we need to give:

- a conditional algebraic specification (admiting an initial model) of the tokens;
- a net schema, a graph whose nodes are the places and the transitions, decorated by various inscriptions defined using algebraic "ingredients" (as terms and atoms). Fig. 2 presents the general form for the part of a schema connected to a transition. There $P_1, \ldots, P_j$ $(j \geq 0)$ are the places appearing only in the preconditions, $P_{j+1}, \ldots, P_q$ $(q \geq j)$ those appearing only in the postconditions and $P_{q+1}, \ldots, P_r$ $(r \geq q)$ those appearing both in the pre and in the postconditions; for all i $cond_i$ is a term of sort $mset(token)$ representing the tokens that place P_i must contain for allowing the transition to fire, $destr_i$ is a term of sort $mset(token)$ representing the tokens eliminated by place P_i during the transition firing, $taken_i$ is a term of sort $mset(token)$ representing the tokens received by place P_i during the transition firing, and $trans\_cond$ is a conjuction of atoms representing the condition under which the transition may fire.
- the admissible place markings, since in general a place cannot contain whatever multiset of tokens (e.g. in some case there is a limit on the number of the contained tokens, while in other only tokens of a certain sort may mark a place;

– which groups of enabled transitions are allowed to fire simultaneously (the net dynamic behaviour).

Using the kit such ingredients may be defined in an overall algebraic setting.

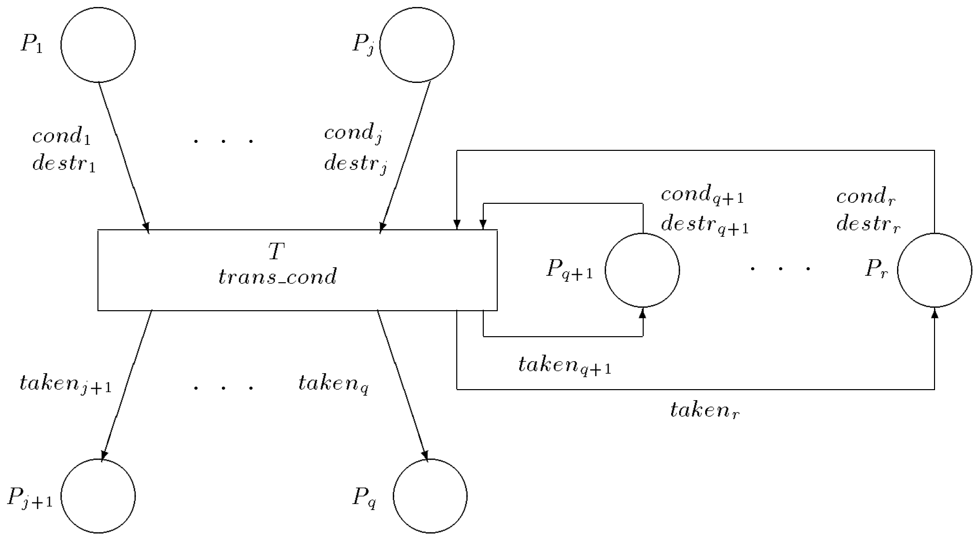

Fig. 2. A generic transition of a net

2 A Short Introduction to SMoLCS

The core of the SMoLCS approach has been developed, mainly by E. Astesiano and G. Reggio at the University of Genova, with a significant contribution by M. Wirsing of the University of Passau, since 1983. While the core of the method has been unchanged, significant improvements and additions have been made since, especially for what concerns semantics, tools and specifications at higher-level of abstraction.

SMoLCS has been applied to significant case-studies, the most important being the specification of the underlying concurrent model in the formal definition of full Ada and two case-studies proposed by ENEL (Italian National Electricity Board).

Here we briefly report the main points of SMoLCS relevant to its use for the definition of the kit, for a full view see, e.g., [3, 4].

Processes are modelled by *labelled transition systems* (shortly *lts*'s). An lts is a triple $(STATE, LAB, \rightarrow)$, where $STATE$ and LAB are two sets and $\rightarrow \subseteq STATE \times LAB \times STATE$; the elements of $STATE$ are the states (intermediate situations of interest) of the processes, $\rightarrow$ represents the possible moves of the processes and the elements of LAB are used to label the moves. Notice that $\rightarrow$ represents move capabilities, i.e. if $(s, l, s') \in \rightarrow$ (written $s \xrightarrow{l} s'$), then a process in the state (situation) s has the capability of passing to state s' performing a move whose interaction with the external (w.r.t. the process) world is represented by l; thus l represents both conditions on the external world for the move to become effective and the transformation of the external world due to such move.

Given an lts we can associate with each process the so called *transition tree*; precisely, a labelled tree whose nodes are decorated by states, whose arcs are decorated by labels, where the order of the branches is not considered, two identically decorated subtrees with the same root are considered as a unique one and there is an arc decorated by l between two nodes decorated respectively by s and s' iff $s \xrightarrow{l} s'$.

Processes are specified by algebraic specifications of lts's and are themselves data as any other; thus they can be manipulated by functions and processes.

SMoLCS supports the user-defined specification of any kind of concurrent structure, communication mechanism (from message passing to shared data) and execution modes (from interleaving to priorities). This support is provided by modularization, hierarchization and parameterization mechanisms for defining and combining specifications of parts of a system, with possibly reusable components of any kind (data, actions, communication and execution mode schemas).

Formally a process is specified by a dynamic specification (i.e. an algebraic specification of an lts), which is as follows.

- A *dynamic signature* $D\Sigma$ is a pair (Σ, DS) where:
  * $\Sigma = (S, OP, PR)$ is a predicate signature,
  * $DS \subseteq S$ (the elements in DS are the *dynamic sorts*, i.e. the sorts corresponding to states of lts's),
  * for all $st \in DS$ there exist a sort $l\text{-}st \in S - DS$ (the sort of the labels) and a predicate $\_ \xrightarrow{\_} \_ : st \times l\text{-}st \times st \in PR$ (the transition predicate).
- A *dynamic algebra* on $D\Sigma$ (shortly $D\Sigma$-algebra) is just a Σ-algebra.
- A pair $(D\Sigma, AX)$, s.t. $D\Sigma$ is a dynamic signature and AX a set of formulae of the form $\wedge_{i=1,\dots,n}\alpha_i \supset \alpha_{n+1}$ (conditional axioms), where α_i are atoms, i.e., have the form either $t = t'$ or $Pr(t_1, \dots, t_n)$ with Pr predicate symbol, is called a *(conditional) dynamic specification.*

  The axioms may refer both to static aspects (e.g., values, states of the system) and to the dynamic aspects, i.e. concerning the transitions predicates.

  A conditional dynamic specification has always an initial model which defines the associated lts.

Notice that by algebra we mean usually a total many-sorted algebra with predicates, see [14]; however there are no problems to use, for example, partial or order-sorted algebras.

A support to modular specification of concurrent systems is then given accordingly to the following schema, where we outline the methodological aspects, leaving apart the algebraic formalism.

A *concurrent system* (i.e. a structured process where several active components [other processes] interact among them) is algebraically specified by an lts as follows. The states are a multiset of states of the active components (process), written $p_1 \mid \dots \mid p_n$.

The moves are specified in several steps, where at each step some partial moves are defined using those defined at the previous step; at the first step the partial moves are defined starting from the moves of process components. It is shown that any specification of a concurrent system may be reduced to a canonical form consisting of the composition of three particular steps, respectively for defining *synchronous moves*, *composing* such moves and *monitoring*, i.e. for deciding which compositions of synchronous moves become moves of the system.

Appropriate algebraic parameterized schemas are given for expressing the three steps; ultimately the specification of a system is a dynamic specification which is an instantiation of a parameterized specification, where the parameters refer to various user defined aspects of the systems concerning dynamics and data. The axioms defining the transition relations for the three steps have particular forms.

By associating with processes an lts we give an operational semantics: two processes are operationally equivalent whenever the associated transition trees are the same. However in most cases such semantics is too fine, since it takes into account all details of the process activity. It may happen that two processes which we consider semantically equivalent have associated different trees (e.g., when the internal moves are not relevant). Thus SMoLCS allows also to define various semantics on the specified systems using a general way to give an observational semantics to a conditional dynamic specification; this approach is very well-suited since it generalizes the Milner-Park's bisimulation technique.

3 A Kit for Defining High-Level Algebraic Petri Nets

Here we present the "kit", i.e. a set of coordinated theoretical tools for formally defining varieties of HLA nets together with their *dynamic behaviour* (the allowed sequences of transition firings), called in the following *token game*, for a large class of varieties. Until now, all varieties of HLA nets that we known (see the various cited papers) could be defined using this kit; also non-high-level and/or non-algebraic nets could be defined, since obviously black/coloured tokens and inscriptions consisting in natural numbers could be algebraically specified.

Precisely, using the kit, we characterize a particular variety of nets, say XYZ, by appropriate parameters (e.g. defining which groups of transitions may fire simultaneously, ...), and a particular net of such variety with an initial marking, say N, by other parameters (e.g. defining the net tokens, places, transitions, ...); then we build a parametric specification that instantiated with such parameters returns the token game of N. All of that is schematically represented in Fig. 1.

Following the SMoLCS methodology we see the token game of a net as the behaviour of a concurrent system and specify such system by a conditional dynamic specification TOKEN_GAME, whose initial model is an algebraic version of an lts (a dynamic algebra) formalizing the net behaviour for all possible initial markings.

TOKEN_GAME as a parameterized specification, whose parameters (schematically represented in Fig. 3) formalize the features of the variety of nets (those on the left) and of the particular net (those on the right) that we are considering.

For specifying a concurrent system, we need to determine its active components and to describe their interactions following the three SMoLCS steps. In this case the active components are the places of the net, their dynamic activity consists in changing their marking by taking part in the firing of the transitions and their mutual interactions are determined by the net transitions.

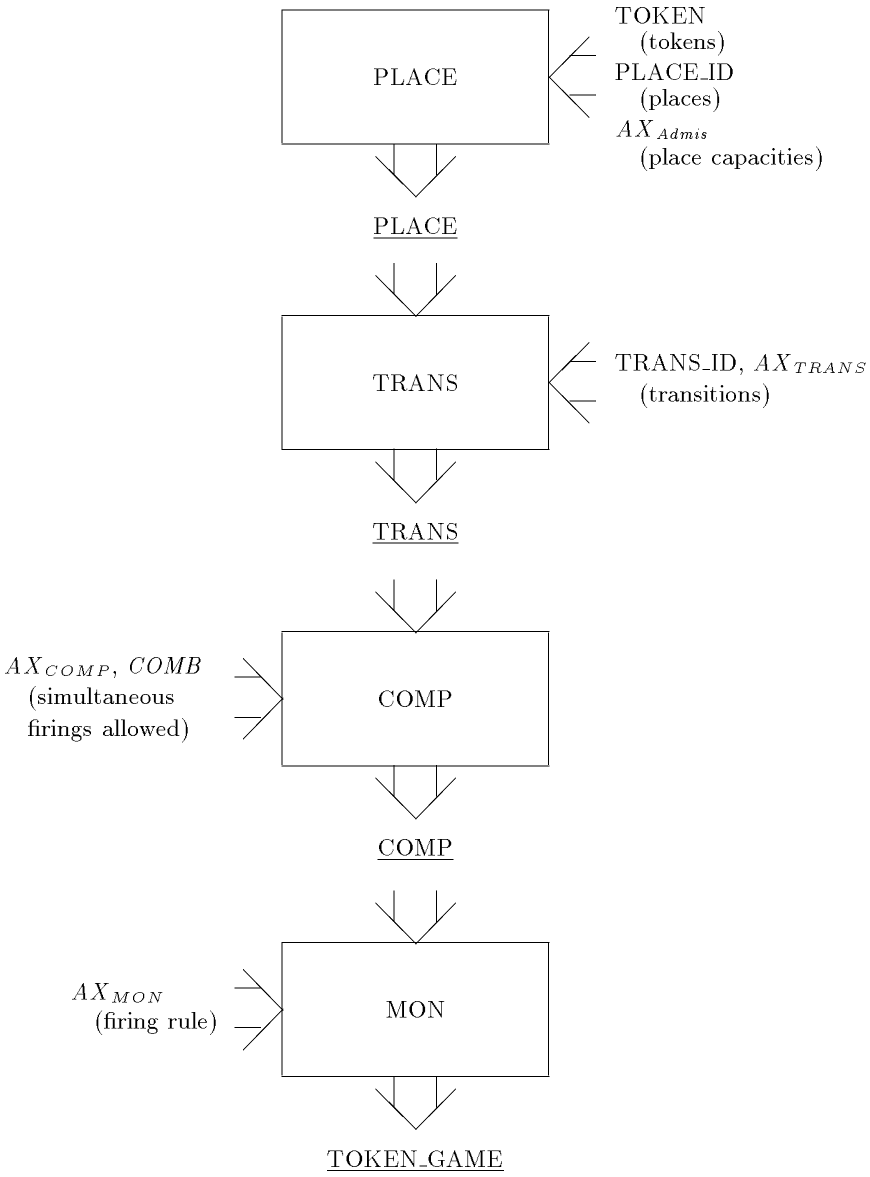

Fig. 3. A schematic view of TOKEN_GAME and of its parameters

The Active Components of TOKEN_GAME The places of the net correspond to the active components of the concurrent system defining the token games and are defined by the parametric dynamic specification PLACE, whose parameters are: an algebraic specification of the tokens of the net TOKEN (with a sort *token*); an algebraic specification PLACE_ID (with a sort *place-id*) stating which are the places of the net and a set of axioms AX_{Admis} defining a predicate "$Admis\colon place\text{-}id \times mset(token)$" stating which are the admissible markings of each place.

The move capabilities of a place are characterized by:

- being tested for seeing whether it contains some tokens (when it appears in the preconditions of a fired transition) and/or
- having some tokens destroyed (when it appears in the preconditions of a fired transition) and/or
- taking some other tokens (when it appears in the postconditions of a fired transition);

thus each place move is characterized by three multisets of tokens: those which should contain for performing the move, those which are destroyed during the move and those which are added during the move. Notice that some of these multisets may be empty (e.g. the third one for the moves of a place taking part only in the preconditions of a transition).

Here and in the following we use some simple constructs, inspired by ASL see [18], to give algebraic specifications in a structured way as: $SP_1 + SP_2$ denoting the specification having the sorts operations, predicates and axioms of both SP_1 and SP_2; **enrich** SP **by** ... denoting the specification obtained by adding to SP some sorts, operations, predicates and axioms. Moreover, **MSET** denotes the parametric algebraic specification of multisets (not reported here) with the operations $\emptyset$ (empty multiset), $\_ \mid \_$ (union), $\_ - \_$ (difference), $\{\_\}$ (singleton) and the predicate $\_ \subseteq \_$; usually $\{x_1\} \mid \ldots \mid \{x_n\}$ is simply written $x_1 \mid \ldots \mid x_n$.

spec PLACE(TOKEN, PLACE_ID, AX_{Admis}) =
  enrich MSET(TOKEN) + PLACE_ID **by**
  dsorts *place* -- states of the lts describing the places
  sorts *l-place* -- labels of the place moves
  opns
    $\_ : \_ : \mathit{place\text{-}id} \times \mathit{mset}(\mathit{token}) \to \mathit{place}$
  -- a state of a place is determined by its identification and by the actual
  -- marking (the contained tokens)
    $\_ \ \mathit{HAS} \ \_ \ \mathit{GIVES} \ \_ \ \mathit{TAKES} \ \_$:
      $\mathit{place\text{-}id} \times \mathit{mset}(\mathit{token}) \times \mathit{mset}(\mathit{token}) \times \mathit{mset}(\mathit{token}) \to \mathit{l\text{-}place}$
  preds
    $\mathit{Admis}: \mathit{place\text{-}id} \times \mathit{mset}(\mathit{token})$
    $\_ \xrightarrow{\_} \_ : \mathit{place} \times \mathit{l\text{-}place} \times \mathit{place}$
  -- transition relation describing the place moves
  axioms
    AX_{Admis}
    $\mathit{cond} \subseteq \mathit{mt} \ \wedge \ \mathit{Admis}(\mathit{pi}, (\mathit{mt} - \mathit{destr})|\mathit{taken}) \ \supset$
      $\mathit{pi}: \mathit{mt} \xrightarrow{\mathit{pi \ HAS \ cond \ GIVES \ destr \ TAKES \ taken}} \mathit{pi}: (\mathit{mt} - \mathit{destr})|\mathit{taken}$

The Synchronous Actions of TOKEN_GAME The synchronous actions (minimal groups of moves of the active components) of the concurrent system describing the token game of a net correspond to the firings of single transitions. Such synchronous actions are formalized by an lts specified by the parametric dynamic specification TRANS (the name recalls that correspond to the net transitions) given below, whose parameters are: an algebraic specification TRANS_ID (with a sort *trans-id*) stating which are the net transitions and a set of axioms AX_{TRANS}, one for each transition, describing the arcs connecting it to the places with the relative inscriptions (thus the net schema).

The axiom corresponding to a transition T of the general form of Fig. 2 is:

$$\begin{aligned}
&\wedge_{\,1\leq i\leq j} \quad p_i \xrightarrow{P_i\ HAS\ cond_i\ GIVES\ destr_i\ TAKES\ \emptyset} p'_i \ \wedge \\
&\wedge_{\,j+1\leq i\leq q} \quad p_i \xrightarrow{Pi\ HAS\ \emptyset\ GIVES\ \emptyset\ TAKES\ taken_i} p'_i \ \wedge \\
&\wedge_{\,q+1\leq i\leq r} \quad p_i \xrightarrow{P_i\ HAS\ cond_i\ GIVES\ destr_i\ TAKES\ taken_i} p'_i \ \wedge\ trans_cond \ \supset \\
&\qquad p_1 \mid \ldots \mid p_j \mid p_{j+1} \mid \ldots \mid p_q \mid p_{q+1} \mid \ldots \mid p_r \xrightarrow{T} \\
&\qquad\qquad p'_1 \mid \ldots \mid p'_j \mid p'_{j+1} \mid \ldots \mid p'_q \mid p'_{q+1} \mid \ldots \mid p'_r
\end{aligned}$$

spec TRANS($\underline{\text{PLACE}}$, AX_{TRANS}, TRANS_ID) =
  enrich MSET($\underline{\text{PLACE}}$) + TRANS_ID **by**
  dsorts $mset(place)$
  preds
    $\_ \xrightarrow{\_} \_ : mset(place) \times trans\text{-}id \times mset(place)$
  - - transition relation describing the synchronous actions
  axioms
    AX_{TRANS}

where $\underline{\text{PLACE}}$ is an instantiation of the parametric specification PLACE given in the previous subsection.

Synchronous Actions Composition of TOKEN_GAME The composition step of the concurrent system corresponding to the token game of a net describes which groups of enabled transitions may be fired together and which is the effect of their simultaneous firings. Also this step is formalized by an lts, whose moves correspond to the allowed synchronous action compositions, specified by the parametric dynamic specification COMP given below.

The various way of composing actions (i.e. synchronous actions and in turn compositions of synchronous actions) represented by means of the corresponding labels are given by some operations $Comb_1, \ldots, Comb_k$ (for simplicity here we assume that they are all binary operations but we can also consider combinators with different arities). How to compose synchronous actions is given by a set of axioms AX_{COMP} of the form either

$cond(t,t') \supset Comb_j(t,t') = t''$, for some j, $1 \leq j \leq k$

defining the action combinators, or

$$\wedge_{\,i=1,\ldots,n}\ mp_i \xrightarrow{t_i} mp'_i \ \wedge\ cond(t_1,\ldots,t_h) \ \supset$$
$$mp_{r_1} \mid \ldots \mid mp_{r_h} \mid mp \xrightarrow{t} mp'_{s_1} \mid \ldots \mid mp'_{s_k} \mid mp$$

defining the effect of the compound actions, where $cond(t,t')$, $cond(t_1,\ldots,t_h)$ are conjuctions of atoms, $\{r_1,\ldots,r_h\} \subseteq \{1,\ldots,n\}$, $\{s_1,\ldots,s_k\} \subseteq \{1,\ldots,n\}$ and t is a term obtained by combining the terms $t_1, \ldots, t_h$ using only the operations $Comb_1$, $\ldots$, $Comb_k$. where t_i's and t are terms, $cond$ a formula, mp_i, mp'_i's variables not appearing in $cond$, $\{r_1,\ldots,r_h\} \subseteq \{1,\ldots,n\}$ and $\{s_1,\ldots,s_k\} \subseteq \{1,\ldots,n\}$.

The above form is very general, but it include various useful and interesting ways of composing synchronous actions, as:

– parallel composition: the axioms have the form

$\wedge_{i=1,\dots,n}\ mp_i \xrightarrow{t_i} mp'_i \wedge cond \supset mp_1 \mid \dots \mid mp_n \mid mp \xrightarrow{t} mp'_1 \mid \dots \mid mp'_n \mid mp$;

– sequential composition: the axioms have the form
$mp \xrightarrow{t} mp' \ \wedge\ mp' \xrightarrow{t'} mp'' \ \wedge\ cond \supset mp \xrightarrow{t''} mp''$;

– sequential composition of serializable actions: the axioms have the form
$mp \xrightarrow{t_1} mp'_1 \ \wedge\ mp \xrightarrow{t_2} mp'_2 \ \wedge$
$mp'_1 \xrightarrow{t_2} mp' \ \wedge\ mp'_2 \xrightarrow{t_1} mp' \ \wedge\ cond \supset mp \xrightarrow{t} mp'$.

This very general form of composition is needed for example to define either the true concurrent nets of [10] or nets where groups of enabled disjoint transitions (i.e. involving disjoint sets of places) are allowed to fire simultaneously.

spec COMP($\underline{\text{TRANS}}$, $\{Comb_1, \dots, Comb_k\}$, AX_{COMP}) =
  enrich $\underline{\text{TRANS}}$ **by**
  opns
    $Comb_1, \dots, Comb_k$: *trans-id* $\times$ *trans-id* $\to$ *trans-id*
  axioms
    AX_{COMP}

where $\underline{\text{TRANS}}$ is an instantiation of the parameterized specification TRANS given in the previous subsection.

The Monitoring of TOKEN_GAME The final moves of the concurrent system describing the token game of a net consist of the execution of one of the compositions of synchronous actions (composition of net transitions) described in the previous step; so at this step we can decide which of such compositions are allowed to fire. Also this step is formalized by an lts, specified by the parametric dynamic specification MON given below.

spec MON($\underline{\text{COMP}}$, AX_{MON}) =
  enrich $\underline{\text{COMP}}$ **by**
  preds
    _ $\overset{-}{\sim\!\sim\!\sim\!>}$ _: *mset(place)* $\times$ *trans-id* $\times$ *mset(place)*
  axioms
    AX_{MON}

where $\underline{\text{COMP}}$ is an instantiation of the specification COMP given in the previous subsection and AX_{MON} is a set of axiom of the form
$mp \xrightarrow{t} mp' \ \wedge\ cond(t, mp \mid mp_1) \ \supset\ mp \mid mp_1 \overset{t}{\sim\!\sim\!\sim\!>} mp' \mid mp_1$
with $cond(t, mp \mid mp_1)$ conjiuction of atoms. The places in mp_1 are those not involved in the firing of the (group of) transitions.

The Whole Definition of TOKEN_GAME Using the parametric specifications defined in the previous subsection, now we are able to give the parametric specification TOKEN_GAME:

TOKEN_GAME(TOKEN, PLACE_ID, AX_{Admis},
  TRANS_ID, AX_{TRANS},
  $COMB$, AX_{COMP},
  AX_{MON}) =
MON($\underline{\text{COMP}}$, AX_{MON})

where $\underline{\text{COMP}} = \text{COMP}(\underline{\text{TRANS}}, \mathit{COMB}, AX_{COMP})$,
$\underline{\text{TRANS}} = \text{TRANS}(\underline{\text{PLACE}}, \text{TRANS\_ID}, AX_{TRANS})$ and
$\underline{\text{PLACE}} = \text{PLACE}(\text{TOKEN}, \text{PLACE\_ID}, AX_{Admis})$.

The initial model I of an appropriate instantiation of TOKEN_GAME (that exists, since it is a conditional specifications, see [18]) is a (dynamic) algebra determining an lts, with states $I_{mset(place)}$ (corresponding to the various markings of the net), labels $I_{trans\text{-}id}$ corresponding to possible groups of transitions which may fire together and transition relation $\rightsquigarrow^{I}$ describing the net firing sequences.

Composing Nets The kit offers also a way to define a complex net putting together the specifications associated with its subparts using the well-known operations for modularly defining algebraic specifications, as sum, rename and instantiations of parametric schemas.

For example let N_1 and N_2 be two nets of the same variety and TK_1, TK_2 be the specifications of their token games given using the kit. Now the specification of token game of the net N_3, which is the composition of N_1 and of N_2 (i.e. the net having the tokens, the places and the transitions of both N_1 and of N_2, where the common ones are taken once) is simply $\text{TK}_3 = \text{TK}_1 + \text{TK}_2$.
Recall that "+" is the ASL-like sum of specifications, thus in TK_3 the sorts *token*, *place-id*, *trans-id*, ... of TK_1 and of TK_2 are identified and have both the elements specified in TK_1 and those TK_2.

4 Using the Kit

The CATNets Defined Using the Kit We use our kit to define the CATNets of [10]; they are HLA nets following the schema of Sect. 1, whose most distinctive feature is that at each step groups of enabled transitions which are "truly concurrent" (i.e. that after firing one of them, the others may still fire) are fired simultaneously.

The parameters for instantiating TOKEN_GAME in this case are as follows.

The tokens

spec $\underline{\text{TOKEN}}$ =
  enrich TOK **by**
  sorts *token*
  opns
    $\{\ S_{tok}\colon tok \to token \mid tok \text{ sort of TOK } \}$

The places

spec $\underline{\text{PLACE\_ID}}$ =
  sorts *place-id*
  opns
    $P_1, \ldots, P_k \colon \to place\text{-}id$ -- $P_1, \ldots, P_k$ are the the places of the net

The admissible place markings They are defined by the following axioms:
$mt \subseteq mt' \ \supset\ Admis(P, mt)$;
for each place P whose capacity is represented by the ground term mt';
$\{Admis(P, S_{tok}(t_1) \mid \ldots \mid S_{tok}(t_r)) \mid r \geq 0\}$
for each place P with infinite capacity whose associate sort is tok.

The transitions

> **spec** TRANS_ID =
> **sorts** *trans-id*
> **opns**
> $T_1, \ldots, T_h : \rightarrow$ *trans-id* - - $T_1, \ldots, T_h$ are the net transitions

The axioms defining the transitions are given as explained in Sect. 3.

The transition composition There are two transition compositors: _ // _ (parallel composition) and _ ; _ (serializable sequential composition) and the set of axiom defining the transition compositions is

$t;t' = t';t \quad t;(t';t'') = (t;t');t''$

$t//t' = t'//t \quad t//(t'//t'') = (t//t')//t''$

- - ";" and "//" are commutative and associative

$mp \xrightarrow{t} mp' \supset mp \mid mp_1 \xrightarrow{t} mp' \mid mp_1$

$mp_1 \xrightarrow{t_1} mp'_1 \wedge mp_2 \xrightarrow{t_2} mp'_2 \supset mp_1 \mid mp_2 \xrightarrow{t_1//t_2} mp'_1 \mid mp'_2$

- - if two disjoint transitions are both enabled, then they may fire simultaneously

$mp \xrightarrow{t_1} mp'_1 \wedge mp \xrightarrow{t_2} mp'_2 \wedge mp'_1 \xrightarrow{t_2} mp' \wedge mp''_2 \xrightarrow{t_1} mp' \supset$

- - if t_1 and t_2 are enabled and t_2 is enabled after firing t_1 and t_1 is enabled after
- - firing t_2 then

$mp \xrightarrow{t_1;t_2} mp'$

- - t_1 and t_2 may be sequentially composed

The transitions which are actually fired

$mp \xrightarrow{t} mp' \supset mp \mid mp_1 \overset{t}{\rightsquigarrow} mp' \mid mp_1$

In [11] it is shown that this definition of CATNets coincides with the one of [10].

High-Level Algebraic Nets with Priorities We use our kit to define a new variety of HLA nets following the general schema of Sect. 1, where the transitions are totally ordered w.r.t. a degree of urgency $<$ (that we call priority order) to be intended as follows: whatever in a net with a certain marking two transitions are enabled, say T and T', s.t. $T < T'$ in the priority order, then it cannot happen that T is fired while T' is not. These HLA nets have been inspired by the classical nets with priorities of [7].

For what concerns the firing of transitions we assume in this case that at each step only one transition may fire (thus there is no way to compose transitions) and that in the case of several enabled transitions the maximum one w.r.t. the priority order is chosen; however using the kit we may consider also the case where several enabled transitions fire together respecting the priorities.

In Fig. 4 we present an example of these nets describing a buffer, where reading takes priority over writing; with such marking first READ will fire, then WRITE will fire twice.

The parameters for instantiating TOKEN_GAME are as follows.

The tokens and the places They are defined as in the previous subsection.

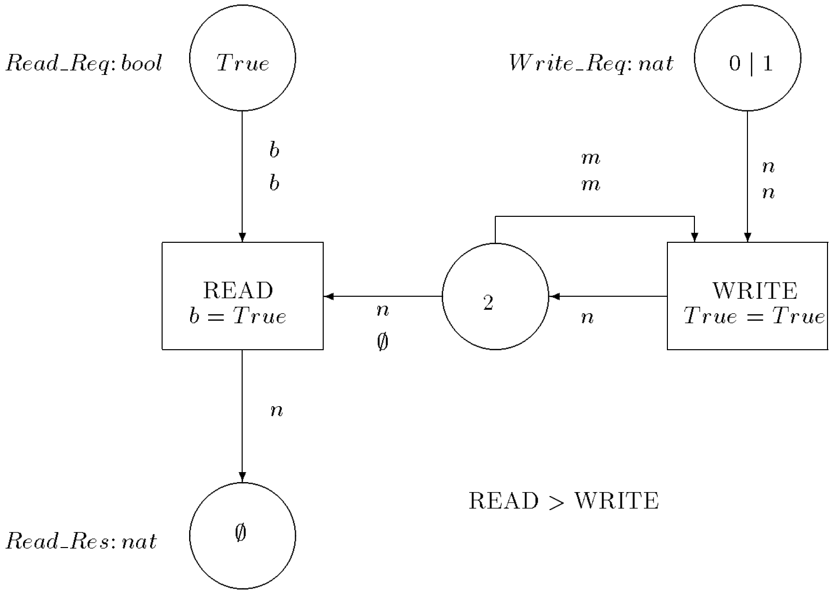

Fig. 4. A buffer specified using a HLA net with priorities

The admissible place markings They are defined by the following axioms:

$\{Admis(P, S_{tok}(t_1) \mid \ldots \mid S_{tok}(t_r)) \mid r \geq 0\}$

for each place P whose associate sort is tok.

The transitions Assume that the priority order of the net is $T_1 < \ldots < T_h$.

spec <u>TRANS_ID</u> =
  sorts *trans-id*
  opns
    $T_1, \ldots, T_n \colon \to$ *trans-id*
  preds
    $\_ < \_ \colon$ *trans-id* $\times$ *trans-id*
  axioms
    $T_1 < T_2 \quad \ldots \quad T_{h-1} < T_h$

The axioms defining the transitions are given as explained in Sect. 3; e.g. the axiom associated with the transition WRITE of the net in Fig. 4 is

$$pb \xRightarrow{Cont\ HAS\ m\ GIVES\ m\ TAKES\ n} pb' \wedge$$
$$pw \xRightarrow{Write\_Req\ HAS\ n\ GIVES\ n\ TAKES\ \emptyset} pw' \supset pb \mid pw \xrightarrow{\text{WRITE}} pb' \mid pw'$$

The transition composition The set of combinators and the set of axioms defining the transition composition are empty, since in this case no transitions can be fired simultaneously.

The transitions which are actually fired

$$mp \xrightarrow{t} mp' \wedge (\nexists\, t', \underline{mp}, \underline{mp'} \,.\, t' > t \wedge \underline{mp} \subseteq mp \mid mp_1 \wedge \underline{mp} \xrightarrow{t'} \underline{mp'}) \supset$$
$$mp \mid mp_1 \overset{t}{\rightsquigarrow} mp' \mid mp_1$$

(an enabled transition may fire if there is no another enabled transition with higher priority)
Notice this is not a conditional axiom, however it is possible to replace it with a conditional formula using auxiliary predicates and operations.

References

1. ——. *High-Level Petri Nets.* Springer Verlag, 1991.
2. E. Astesiano, A. Giovini, F. Morando, and G. Reggio. Algebraic specification at work. In *Proc. AMAST'91.* Springer Verlag, 1992.
3. E. Astesiano and G. Reggio. SMoLCS-driven concurrent calculi. In *Proc. TAPSOFT'87, Vol. 1*, number 249 in L.N.C.S. Springer Verlag, 1987.
4. E. Astesiano and G. Reggio. A structural approach to the formal modelization and specification of concurrent systems. Technical Report PDISI-92-01, Dipartimento di Informatica e Scienze dell'Informazione, Università di Genova, Italy, 1992.
5. E. Battiston, F. De Cindio, and G. Mauri. OBJSA nets: a class of high-level nets having objects as domains. In *Advances in Petri Nets*, number 340 in L.N.C.S. Springer Verlag, 1988.
6. L. Berardinello and F. De Cindio. A survey of basic net models and modular net classes. In *Advances in Petri Nets*, L.N.C.S. Springer Verlag, 1992. To appear.
7. E. Best and M. Koutny. Petri net semantics of priority systems. *T.C.S.*, 96, 1992.
8. M. Bettaz. An association of algebraic term nets and abstract data types for specifying real communication protocols. In *Recent Trends in Data Type Specification*, number 534 in L.N.C.S. Springer Verlag, 1991.
9. M. Bettaz, A. Choutri, and G. Reggio. A life-cycle for parallel and distributed systems based on two models of concurrency. In *Proc. of Second Euromicro Workshop for Parallel and Distributed Processing.* IEEE Computer Society, 1993. To appear.
10. M. Bettaz and M. Maouche. How to specify nondeterminism and true concurrency with algebraic term nets. In *Recent Trends in Data Type Specification*, number 655 in L.N.C.S. Springer-Verlag, 1992.
11. M. Bettaz and G. Reggio. A SmoLCS based kit for defining the high-level algebraic Petri nets and their semantics. In Preparation, 1993.
12. J. Billington. Many-sorted high level nets. In *Proceedings of the Third International Workshop on Petri Nets and Performance Models*, Japan, 1989.
13. C. Dimitrovici and U. Hummert. Composition of algebraic high-level nets. In *Recent Trends in Data Type Specification*, number 534 in L.N.C.S. Springer Verlag, 1991.
14. J. Goguen and J. Meseguer. Models and equality for logic programming. In *Proc. TAPSOFT'87, Vol. 2*, number 250 in L.N.C.S. Springer Verlag, 1987.
15. J. Meseguer. Rewriting as a unified model of concurrency. *TCS*, 96, 1992.
16. W. Reisig. Petri nets and algebraic specifications. *TCS*, 80, 1991.
17. J. Vautherin. Parallel system specifications with coloured Petri nets and algebraic data types. In *Advances in Petri Nets*, number 266 in L.N.C.S. Springer Verlag, 1987.
18. M. Wirsing. Algebraic specifications. In *Handbook of Theoret. Comput. Sci.*, volume B. Elsevier, 1990.

This article was processed using the LaTeX macro package with LLNCS style