Abstract Data Nets

combining Petri nets and abstract data types

for high level specifications

of distributed systems

B.Berthomieu*, N.Choquet**,
C.Colin***, B.Loyer***,
JM.Martin*, A.Mauboussin**.

* LAAS-CNRS, 7 Avenue du Colonel Roche, 31077 Toulouse-Cédex, France.
** LdM-CGE, Route de Nozay, 91460 Marcoussis, France.
*** Alcatel, Route de Perros-Guirec, BP.344, 22304 Lannion-Cédex, France.

This work was supported by DAII, under grant n°84.35.087 including Alcatel, Laboratoires de Marcoussis and Laboratoire d'Automatique et d'Analyse des Systèmes.

I. Introduction

The work presented here is part of a broader project involving the Alcatel company, the CGE and the LAAS research laboratories. The goals of the project were to investigate the respective power and applicability of Petri nets and Algebraic abstract data types for modelling complex switching systems. A particular subtask of the project was aimed at designing and evaluating a composite model, combining Petri nets and Abstract data types; this paper summarizes the work done in this particular subtask.

Switching exchanges; which are usually very large systems, are characterized by a very high complexity. While most of the mechanism involved are related to real time process control, they also involve large amounts of data. Petri nets, when used alone, has proven to be unsufficient to account for all the relevant aspects of these systems; on the other hand, preliminary studies have shown that Abstract data types were suitable for representing most aspects of data handling, but lacked some capabibities for expressing comfortably some synchronization and real time constraints. This led to the idea of a composite model.

Let us recall first some capabilities and limits of nets and abstract data types in our context:

Petri nets are well suited for expressing and analyzing properties of operational nature in concurrent systems; for instance, they can clearly represent deadlocks, or parallelism. However, some mechanisms, even if they are pure synchronisation, require a level of abstraction which cannot be given by basic Petri nets (in which everything is expressed in terms of places and transitions). This lack of flexibility of Petri nets may lead to models which are complicated in spite of the fact that they give a correct representation of the problem; a typical example could be that of a waiting queue, for instance.

Considering these limits, several authors proposed what can be called "composite models", which generally combine two distinct formalisms, devoted to the control and data handling aspects, respectively. Among them, let us mention:

- the Keller model /Keller/, which combines a control structure expressed by a Petri net, with a programming language and data sets for representing data,

- Petri nets schemes from Valette /Valette/ or the UCLA Graph Model of Behavior /Cerf/ . Both models share the same goal as Keller's model. They both use nets (or UCLA graphs for the latter model) for expressing the control aspects, plus some formalism for data handling. In addition, both try to provide the data aspect with some known advantages of Petri nets (e.g. visual aids, ...).

On the other hand, algebraic specifications /Goguen/ allow describing abstractly the operations performed on data. However, the specifications does not generally describe the way actions are scheduled. Indeed, the constraints on operation scheduling could be expressed in this formalism too, but doing this would generally obscure the specifications; in any case, operational properties such as liveness or deadlock freeness are difficult to express and prove.

A strong advantage of algebraic specifications over Petri nets is their capability of manipulating infinite sets of data. Another advantage is that algebraic data types allow a modular and hierarchical design of specifications. A specification is built on a set of lower level specifications, and may be, in turn, used as component of a more complex specification. Complex specifications can be constructed from elementary specifications using libraries of specifications, through some facilities for combining specifications together, such as parameterisation or renaming.

A preliminary study showed several ways of combining Petri nets and Abstract data types /Berthomieu/; one of the models investigated, called "Abstract Data Nets", was selected for further investigations and is presented in the following section.

The proposed model can be seen as a graphical representation of abstract data types with a net part which is an extension of Predicate/Transition nets.

This model has some obvious relationships with two recently proposed composite models, also combining Petri nets and Abstract data types. The first of these is the Semi Graphical Specification language (SEGRAS) from Krämer /Krämer/, the second is much closer to our model thought developed independantly /Vautherin/. Both these models combine a control structure derived from Petri nets with the formal expression and the structuring facilities obtained from the algebraic specifications.

Abstract Data nets are presented in the following section, Section 3

is devoted to some examples of use while Section 4 concludes on the applicability of the model.

II. Presentation

II.1. The model

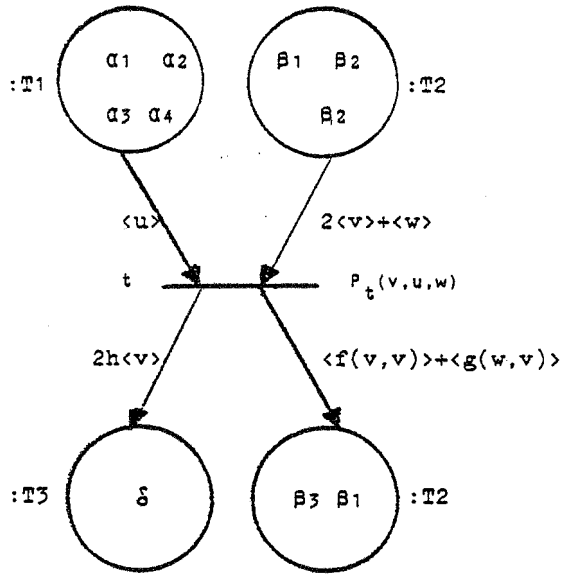
We use Predicate/Transition nets /Genrich/ as a starting point. But the conditions which guarantee their equivalence with classical Petri nets have been removed: tokens in the nets do not necessarily belong to finite sets. Furthermore, there is no restriction on the structure of tokens.

An ADN (Abstract Data Net) specification is made up of two parts:

- the algebraic part stipulates:
  - the imported specifications defining the lower level data types,
  - the name of the new type or system under specification,
  - the profiles of the required operations and predicates (that are boolean operations),
  - a set of axioms.

- the net part expresses the control structure of the specified system. The tokens are typed terms whose types are defined in the imported algebraic specification:
  - each place of the net is typed: a place may only hold terms of its associated type, it may contain several copies of the same term;
  - multisets (or bags) of variables are associated with the incoming and outgoing arcs of the transitions. It is required that the variables used on the outcoming arcs also appear on the incoming arcs.
  - A predicate P_t is associated to each transition t. This predicate is built on operations defined in the imported specifications and can thus be algebraically specified. The free variables in P_t must belong to the variable bags associated to the incoming arcs of the transition.

In Fig. 1, we show an example of information associated to a transition t. The types T1, T2, T3 are defined in the imported specifications. α_i, β_i, δ_i are respectively terms of types T1, T2 and T3. u,v,w are typed variables and f,g,h are operations defined in the imported specifications. Let us notice that in Fig. 1, we give only a transition example and give neither the algebraic part of the specification corresponding to the new specified type, nor the imported specifications.



The profile of the operations f,g,h and P_t defined in the imported specifications are:

```
f : T2 x T2 ---> T2
g : T2 x T2 ---> T2
h : T2 ---> T3
Pt: T2 x T1 x T2 ---> boolean
Variables:
u: T1; v,w: T2
```

Figure 1: Example of a transition t in the ADN model

II.2 Evolution rules

The initial marking is an distribution of terms into the places of the net, in accordance with the type of each place.

Let p be one of the incoming places of t, and $\langle v_1\rangle+\langle v_2\rangle+\ldots+\langle v_n\rangle$ the multiset of variables associated with the arc (p,t) and $\langle u_1\rangle+\langle u_2\rangle+\ldots+\langle u_k\rangle$ the multiset associated with the arc (t,p);

Let $P_t(w_1,\ldots,w_n)$ be the predicate associated with the transition t, where w_i are incoming variables of transition t.

The transition t is firable from the marking M if and only if there exists a substitution S (a function which associates a term with each variable; and by extension a function on the set of terms) compatible with

the typing of each place such that:

$S(v_1)+S(v_2)+...+S(v_n) << M(p)$ is true for each incoming place p of the transition t;
and $P_t(S(w_1),...,S(w_n))$ is true;

where terms $S(v_i)$ and $S(w_i)$ are the results of substituting the variables v_i and w_i, and << denotes multiset inclusion.

For instance, in Fig. 2, a possible substitution would be: S(x)=3 and S(y)=4.

After the firing of the transition t, the new marking of place p will be

$M(p) -- S(v_1)+S(v_2)+...+S(v_n) ++ S(u_1)+S(u_2)+...+S(u_k)$

where $S(u_1)+S(u_2)+...+S(u_k)$ is the result of applying the substition S to the multiset associated with the arc (t,p), and -- (resp. ++) denote the multiset difference (resp. union).

Since the places are typed, the terms associated with the arcs adjacent to a place must possess the same type as the place. Thus a syntactic verification of the net is needed.

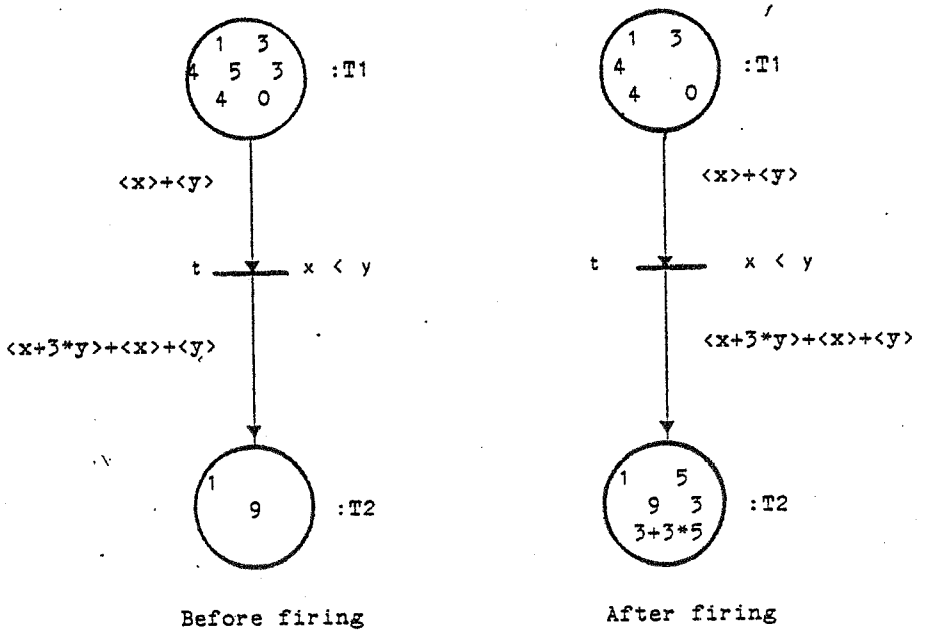


Figure 2: Example of firing

It must also be pointed out that the net expresses a transformation of terms and not the congruence on terms. The congruence is expressed, for each type, by the axioms of the algebraic specification. Thus, with respect to the example in Fig.2, it cannot be inferred from the net alone that the terms 3+3*5 and 18 are equivalent.

II.3. Relationships between the algebraic part and the net part.

An ADN specification can be seen as an algebraic specification in the following way:

- A new type can be associated with the composite model, which has the structure a product type; a constant of the type is associated with the initial marking of the composite model; constructors are associated with the transitions of the model (a constructor may handle one or more transitions), and selectors (or observers) are associated with the places of the model. Intuitively, the values of the type associated with the model are the markings of the net;

- The constructor associated with transition t returns the 'global' marking of the model obtained by firing the transition t from the current marking when it enables the transition; the transformation on marking performed by each constructor can be expressed by a conditional equation. The selector associated with place p returns the marking of place p for a given 'global' state of the model;

Thus, from a composite specification, it is possible to build (mechanically) an algebraic specification which can be used in a higher level specification. The net part of an ADN specification can also be seen as a graphical representation of the type being defined.

II.4. Simplified notation

Prior to formulating our examples with this formalism, it seems useful to introduce a more intuitive notation for the net.

In most cases, the predicate associated with a transition can be expressed by pattern-matching constraints expressed directly on the incoming arcs of the transitions. Arcs carry: the information relative to the number of terms which must either be extracted from the incoming places or added to the outcoming places and the information (which may be partial) on the contents (or structure) of these terms. The example in Fig. 3, which places have tokens of the "element" or "list of elements" type,

illustrates this notation.

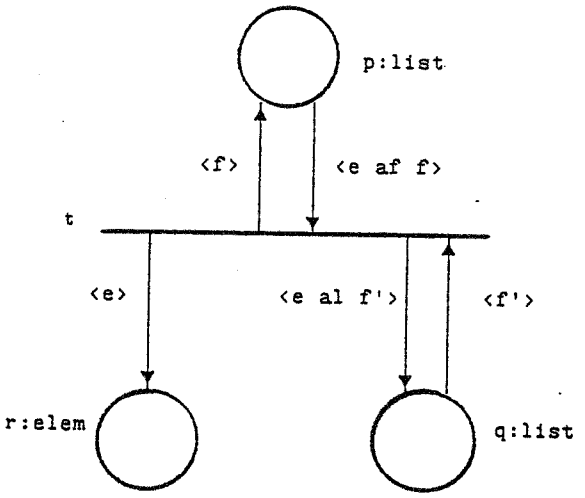


Figure 3: Use of the simplified notation

Let "elem" be some data type and let "list" be the algebraic specification of a list of these elements, in which the operations empty (for the empty list), al (add a last element) and af (add a first element) are defined. The profiles of these operations are:

```
empty : ---> list
_al_  : elem x list ---> list
_af_  : elem x list ---> list
```

Let $e_1,\ldots,e_n$, be elements of the elem type. A term of the list type can be described specifically as a composition sequence of either the _al_ operator:

e_1 al (e_2 al (... (e_n al empty)...))

or the _af_ operator:

e_1 af (e_2 af (... (e_n af empty)...))

on the empty operator.

It is always possible to use one of these notations to describe the terms of the list type and to change from one notation to the other one.

Let us assume that we have to describe a transition t (cf.Fig.3) which takes a non-empty list f1 from the place p, extracts the first element e from this list, introduces it in the place r and add it to the end of some list in the place q.

The non-empty condition on f1 means that there must exist a list f and an element e such that f1 = <e af f>. To simplify the notation, this condition is written directly on the arc with the label <e af f>. Similarly, the outgoing arc to q will be labeled <e al f'>.

III. Examples

III.1. Preliminary remarks

Two applications examples are presented in this paper, i.e. the alternating bit protocol and a telephone-based example. In accordance with the model defined in chap.II, each specification comprises a net part and an algebraic part.

For each example, specifications are written using the ADN model, when the control aspects are essential to the specification. When such aspects are not relevant, the specification is directly built with algebraic data types.

III.2. The Alternating bit protocol

The first example we specify using the ADN model, is the alternating bit protocol. This protocol deals with the exchange of data between two distant entities called "sender" and "receiver". These entities are connected through an unreliable transmission support called "medium", a message is either correctly transmitted or lost. Each message sent must be positively acknowledged. It is assumed that the loss of messages is signaled by the medium.

The addition of a label to a message allows detecting the duplicates when several copies of the same messages have been sent. In this protocol a one bit label is enough since only one message can be in transit at a time.

We wont detail further the behavior of the protocol since it is very well known; the reader who wants more details can find them in /Bartlett/.

Let us first make precise the primitive data types needed for describing the protocol. Only the names of the defined sorts and the name and profile of the required functions are given:

```
sort msg  % describes a message: no more details are necessary here except
          that there exists a particular constant "nil" which denotes the
          absence of message %

sort bit
functions
     0,1 : ---> bit
     ~   : bit ---> bit    % to complement a bit %

sort pkt  % a packet is made up of a message and a bit %
functions
     <_ _>: msg x bit ---> pkt

% Each sort m_list , b_list, p_list is a particular instantiation of the
"list of elem" sort where elem is respectively msg, bit, pkt %
sort list of elem
functions
     empty : ---> list
     _af_  : elem x list ---> list % f1 = e af f0: e first elem of f1 %
     _al_  : elem x list ---> list % f1 = e al f0: e last elem of f1%
```

Specific data are required for the sender (respectively the receiver). They correspond to their states:

```
sort s_state = {readysm, wait}
sort r_state = {readyrm, readysa}
```

Then we specify the sender and the receiver. Afterwards we built the alternating bit specification combining these two specifications and describing the interactions between the sender and the receiver (medium).

a) Sender

Constructors are:

- **send**: associated with both the sending and the retransmission of a message after the reception of a negative acknowledgement. In the net part of the specification, two transitions are associated with this constructor.

- rack : represents the reception of an acknowledgement from the receiver. In the net part of the specification, two transitions are associated with rack. One corresponds to a positive acknowledgement, the other one to a negative acknowledgement.

Observers are:

- to_send: represents the list of messages to send.
- last_p: memorizes the last sent packet (last messages and last bit sent).
- sstate: gives the state of the sender which is either ready to send a new message (readysm) or waiting for a positive acknowledgement (wait). In the net part of the specification, two control places are associated with the state of the sender.

b) Receiver

Constructors are:

- rec: represents the reception of a packet. Depending on the sequence bit of its packet, the message is either accepted, stored into the list of received messages or rejected. In the net part of the specification two transitions are associated with this constructor.
- sack: represents the sending of a positive or negative acknowledgement.

Observers are:

- received: represents the list of messages transmitted correctly.
- last_a: gives the value of the last sent acknowledgement.
- rstate: gives the state of the receiver, which is either ready to receive a packet (readyrm) or ready to send an acknowledgement (readysa). In the net part of the specification, two control places are associated with the state of the receiver.

c) Medium

The medium describes the interconnection of the sender and the receiver. It may be split into two parts: sender ---> receiver and receiver ---> sender, as can be seen on the net part of the specification (Fig 5.).

The medium is characterized by the list of packets in transit from the sender to the receiver and by the list of acknowledgements in transit from the receiver to the sender. These are associated with the observers

p_transit and a_transit, respectively.

Furthermore as transmission is assumed unreliable, packets and acknowledgements may be lost.

There are thus two specific constructors of the medium: loss_p and loss_a.

Combining those parts and adding an initialisation constructor, we get the list of the functions used in the alternating bit protocol specification (Fig.4).

The init constructor represents the initialisation of the global system. In the net part no transition is associated with it. But the initial marking of the net corresponds to the initial values of each observer of the system. These values are given in the axiom item of the algebraic part.

In the alternating bit example, the control aspect is more complex than the data management aspect. Therefore the net part of this example is important and is not very different from the corresponding specification to those written in Predicate/Transition nets. However the ADN model enables some writing simplification by allocating a type to places. In the same way, places of type message list, packet list or bit list can be used.

```
specification AB
Algebraic Part
import MSG, BIT, PKT, M_LIST, B_LIST, P_LIST, S_STATE, R_STATE
sort system
functions
     init      : m_list ---> system
     send      : system x bit ---> system
     rack      : system x pkt ---> system
     rec       : system x pkt ---> system
     sack      : system x bit ---> system

     loss_p    : system ---> system
     loss_a    : system ---> system

     to_send   : system ---> m_list      % list of messages to send %
     last_p    : system ---> pkt         % last sent packet %
     sstate    : system ---> s_state     % state of the sender %

     received  : system ---> m_list     % list of received messages  %
     last_a    : system ---> pkt        % last sent acknowledgement %
     rstate    : system ---> r_state    % state of the receiver %

     p_transit: system ---> p_list
     a_transit: system ---> b_list

variables                              % also used in the net part %
     b: bit; m,x: msg; f: m_list; l: p_list; l': b_list;

axioms
     to_send (init(f)) = f
     last_p (init(f)) = <nil 0>;
     sstate (init(f)) = readysm;
     received (init(f)) = empty;
     last_a (init(f)) = 0;
     rstate (init(f)) = readyrm;
     p_transit (init(f)) = empty;
     a_transit (init(f)) = empty;
```

% In this example there is no other axiom in the algebraic part of the specification because those needed for an algebraic data type specification can be deduced from the net part of the specification. %

Figure 4: Alternating bit specification

Net Part

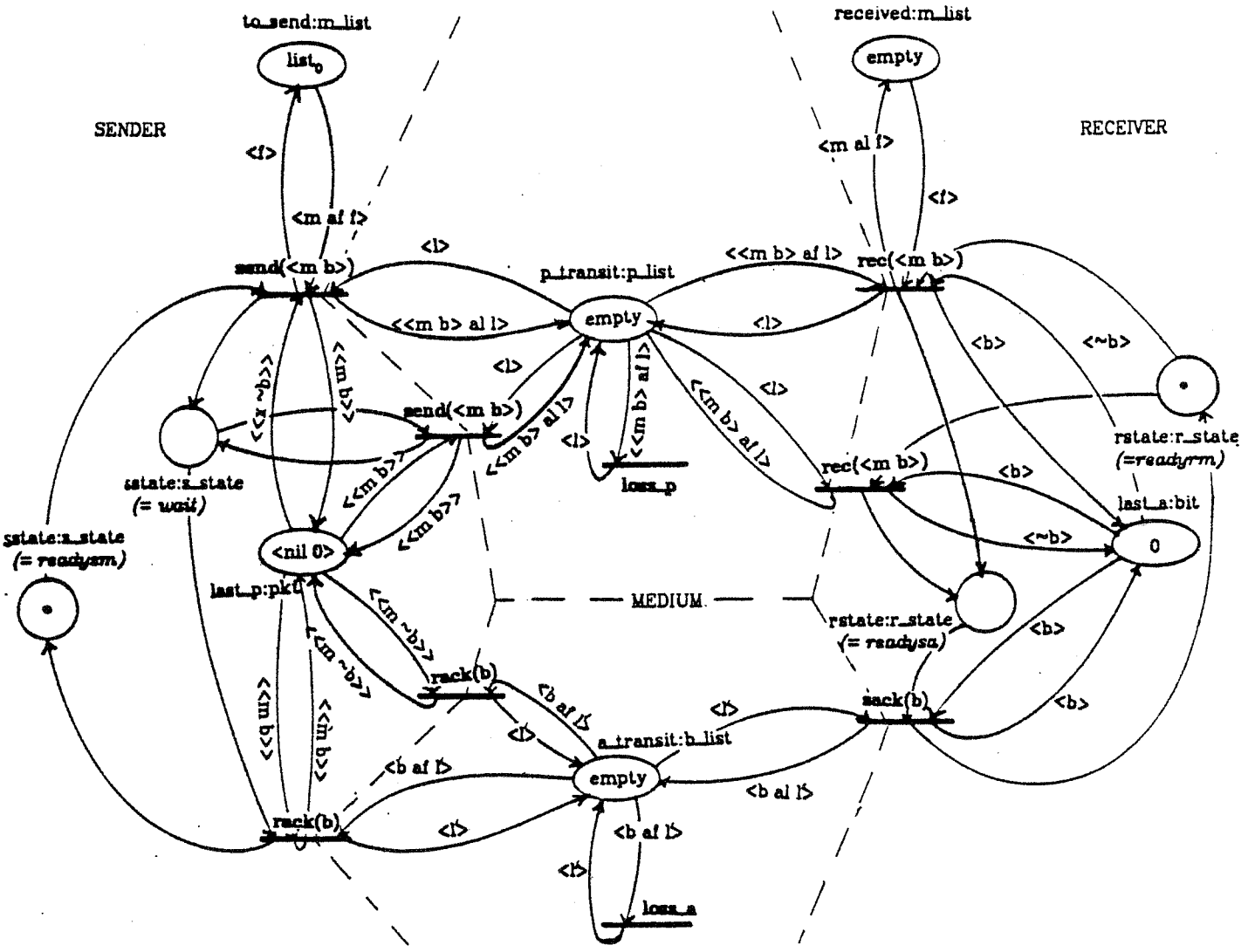


Figure 5: Alternating bit specification (continuation)

III.3. A telecommunication protocol: the signalling link management.

This section presents a significant example taken from the Signalling System No. 7 /CCITT/: The signalling link management level 3 of the Message Transfer Part (MTP).

III.3.1. Background

The Signalling System No. 7 is an internationally standardized general

purpose common channel signalling system. It provides a reliable transport system for information transfer between switching exchanges. It is divided into a common MTP on the one hand and separate User Parts as Telephone User Part (TUP) on the other hand.

The functions of the MTP are separated into 3 levels: the signalling data link functions (level 1), the signalling link functions (level 2), which are related to the transfer of signalling messages over one individual signalling data link without duplication or loss, and the signalling network functions (level 3) which are transport functions common to several signalling links. Two of these functions are the signalling traffic management, which controls message routing, and the signalling link management (SLM), which controls the locally connected link sets.

III.3.2. The specification

The SLM provides means for establishing and maintaining a predetermined capability of a link set. In our example, p links among n must be active even if some of them become unavailable.

The ADN model of SLM imports two algebraic specifications: the LINK specification that represents the individual signalling link and the LINKSETSTATE specification.

```
Specif LINK
import LINKNUM,  % link number%
       LINKSTATE
sort link
functions

              % constructors %
     linkinit : linknum ---> link       % initialisation %
     act      : link ---> link          % link activation %
     desact   : link ---> link          % link desactivation %
     in_serv  : link ---> link          % link activated %

              % observers %
     st       : link ---> linkstate     % link state %
     ln       : link ---> linknum       % link number %


Variables
     x : linknum
     l : link
```

```
axioms
     st (linkinit(x)) = inactive;
     st (act(l)) = init;
     st (in_serv(l)) = active;
     st (desact(l)) = inactive;
  selective_obs (ln, linkinit);
     ln (linkinit(x)) = x;

end LINK
```

Note: "Selective_obs" is a metaconstruction /Biebow/ allowing the writing of the significant axioms only. The other ones, which do not change the value of the observers, are automatically deduced.

The LINK specification uses LINKNUM and LINKSTATE which describe the number of the link and its state, respectively.

sort linknum is an instantiation of a more general sort: the natural number interval sort with an upper bound equal to n, a lower bound equal to 0 and with the operations +1 and -1.

sort linkstate = (inactive, init, active). The link is inactive in the initial state. If SLM decides to initialise a link, it sends "Start" to the level 2. The link state is then init. The level 2 can answer to SLM by "Out of Service" or "In Service". In the first case, another "Start" is transmitted to level 2 and in the second one, the link becomes active. The link can therefore be desactived after the management request or if the number of active links become greater than p.

The linkstate LINKSETSTATE specification defines the linksetstate sort and corresponds to the two possible states of a link observed from a link set point of view: "LSACTIVE" when the link is "init" or "active" and "LSINACTIVE" when the link is "inactive". If no link belongs to LSACTIVE, the link set is inactive.
sort linksetstate = { LSACTIVE, LSINACTIVE }

The Fig. 6 and Fig. 7 represent respectively the algebraic part and net part of the ADN of SLM.

The constructor (event) names recall the various messages received from the management (MGMT) and the level 2 (L2) that trigger off the firing of the corresponding transition. They are:

MGMT?Actls : represents the reception of the "Active link set" message (Actls) sent by MGMT;

MGMT?Desls : represents the reception of the "Desactive link set" message (Desls) sent by MGMT;

MGMT?Desl : represents the reception of the "Desactive link" message (Desls) sent by MGMT;

L2↑RPO : represents the reception of the "Remote Processor Outrage" message sent by L2;

L2↑IS : represents the reception of the "In Service" message sent by L2;

L2↑OS : represents the reception of the "Out of Service" message sent by L2.

Observers are :

link? : used to observe the evolution of an individual link;

state : the state of the link observed by the link set: LSACTIVE, when its state is active or init, or LSINACTIVE otherwise.

ACTIVELINK : memorises the number of active links. So this place always contains one token.

The net part (Fig. 7) uses the following conventions:

- The initial marking is written accordingly to the initialization axioms of the algebraic part.

- Each transition (or event) is labeled with the received message which enables it and the messages that must be sent (messages are separated by ;).

For instance:

MGMT?Actls means that the transition cannot fire without having received the message "Actls" from MGMT ("?" means that MGMT is a module of level 3),

L2↑OS(x) means that the transition cannot fire without having received "OS(x)" from the level 2 and

L2↓Start(x) means that during the firing of the transition, SLM will send Start(x) to level 2.

Let us explain some of the transitions of Fig. 7.

The transition (1) is fired at the system activation. Then p "Start" signals are sent to the level 2 (indicated by L2↓Start) to activate the p signalling links, the p tokens, linkinit(0), ..., linkinit(p-1), are removed from the place "LSINACTIVE" and p tokens act(linkinit(0)), ...

act(linkinit(p-1)) are added to the place "LSACTIVE". This transformation is mentioned on the arc (2). The system is then waiting for the answer of the various links of the level 2 for each start request. If the answer is "Out of Service" (3) (L2↑OS), another link of the place "LSINACTIVE" is activated, and if it is "In Service" (4) (L2↑IS), the content of the place "ACTIVELINK" is incremented.

The model does not show the transfer of messages because it only represents one entity. However, if we want to have a global model of the SLM connected to level 2, we should introduce a medium (cf. II.2.C.)

Note:

The method used for building the net part of the SLM is a bit different from the one used for the alternating bit protocol specification. We said in II.2. that we would associate to each observer of the algebraic part a place or a set of places (when they correspond to control states) of the net part. In large applications, this method may complicate the net because of the great number of places with loops connecting them to the transitions. So, we choose to associate several observers to a place or a set of places when it is possible that is to say when the observers describe the same entities. In this example, we associate the observers link? and state in the two places **LSACTIVE:link** and **LSINACTIVE:link**. The modifications of the values of the observers **link?** and **state** are then mentioned respectively on the arcs between the transitions and these places, and by the name of the place.

Other applications of the ADN /Choquet1/ have been done: they concern the signalling traffic management level 3 of the MTP and some functions related to the call control signalling, as TUP and the subscribers management.

```
Specif SLM

Algebraic Part
import LINK, LINKSETSTATE
sort linkset

functions

                    % constructors %
     linksetinit: ---> linkset                          % initialisation %
     MGMT?Actls : linkset ---> linkset;                 % link set activation %
     MGMT?Desls : linkset ---> linkset                  % link set desactivation %
     MGMT?Desl  : linkset x linknum ---> linkset        % link desactivation %
     L2↑OS      : linkset x linknum ---> linkset        % out of service %
     L2↑IS      : linkset x linknum ---> linkset        % in service %
     L2↑RPO     : linkset x linknum ---> linkset        % remote processor outrage %


                    % observers %
     link?      : linkset x linknum ---> link
     state      : linkset x linknum ---> linksetstate
     activelink : linkset ---> linknum

Variables
     x,y:linknum
     l0, ..., lp-1, lk, l,l' : link

axioms
          % initialisation %
     link? (linksetinit, x) = linkinit (x)
     state ( linksetinit, x) = LSINACTIVE
     activelink (linksetinit) = 0
```

Figure 6: Algebraic part of the SLM specification

Net Part

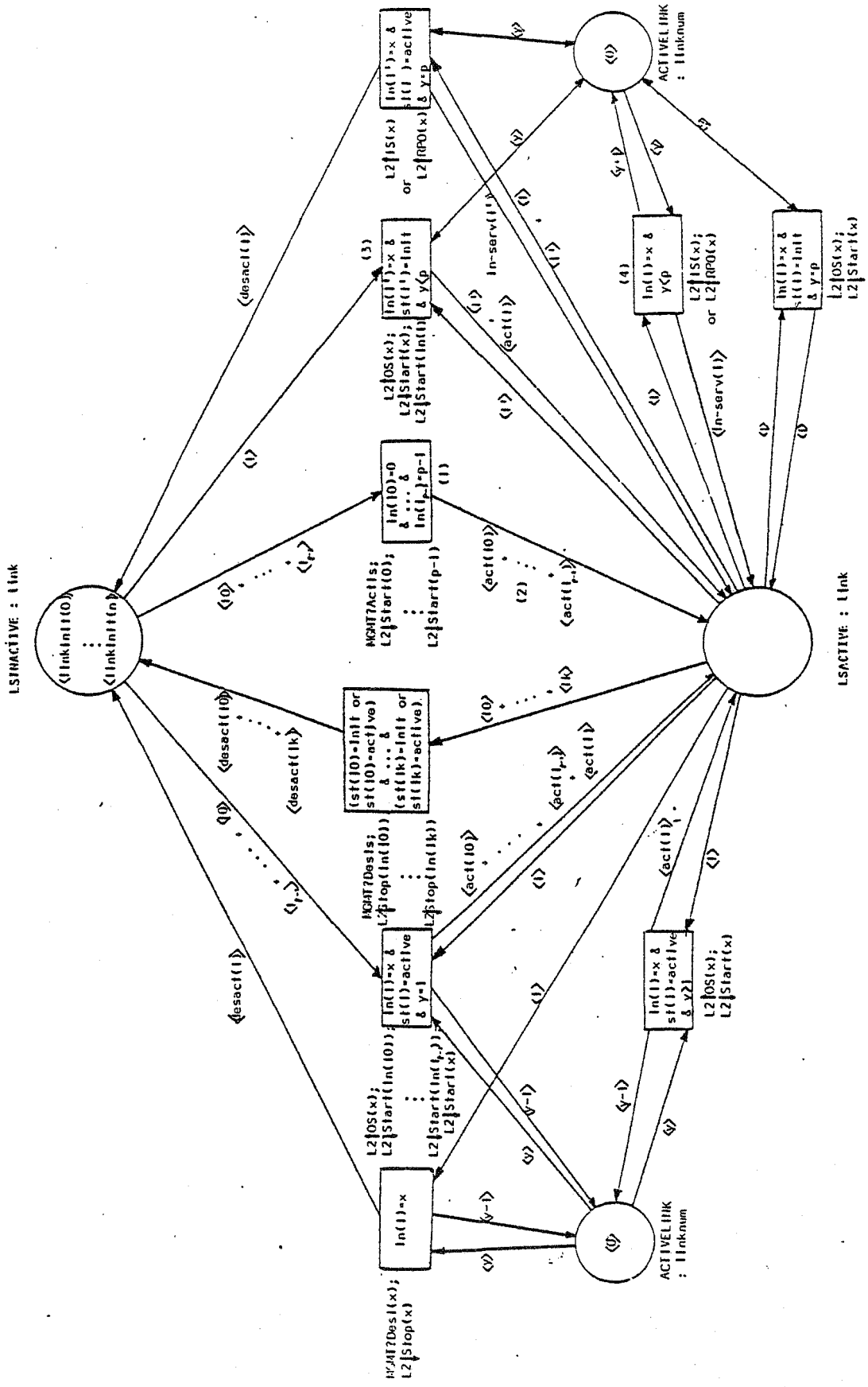


Figure 7: Net part of the SLM specification

IV. Model evaluation

IV.1. Expressiveness

With respect to Petri nets describing a control structure, ADNs have clearly the same expressiveness as Turing machines (the zero-testing problem can be easily encoded into an ADN); ADNs thus extend the expressiveness of nets. Moreover, the new structuration possibilities still increase the power of abstraction of such models; the tokens contained in the places are not only integers (as in Place/Transition nets) or constituted of some basic types (as in Predicate/Transition net) but are typed marks of arbitrary structure, for which a complete specification is written.

With respect to data handling, ADNs have the same capability as the abstract data type formalism.

IV.2. Structuration

The structuration possibilities of the ADN model originate from algebraic abstract data type ones, since it is always possible to shift from a ADN specification to an abstract data type specification.

Furthermore, it is not necessary to specify each level by using the ADN model. This formalism will be used when operational aspects must be described, elsewhere the algebraic data type formalism is sufficient. Thus, as can be seen in the example given in III.3. some specifications (as the link specification) can be directly written in the algebraic data type formalism.

IV.3. Verification

As each ADN specification can be transformed into an algebraic specification, the usual verification techniques for abstract data types can be applied. The verification of properties may use equational reasoning, confluence analysis (Knuth-Bendix algorithm), structural induction /Thompson/, /Paul/, or symbolic evaluation /Kaplan/ and specification execution /Choquet 2/.

But, as marks are not necessarily finite natural integers, few information can generally be inferred from the net structure. Checking liveness properties, for instance, will need general purpose methods such as Floyd's method.

The notion of complete specification is generally too powerful to be verified. Thus, Guttag /Guttag/ defines a sufficiently complete notion

which is suitable. It cannot be shown whether or not a set of axioms is sufficiently complete. Nevertheless, /Guttag/ gives a syntactic method to establish specifications, which ensures a sufficiently complete axiomatisation. Moreover, when there is no axiom expressing the relations between constructors, this method ensures the consistency of the specification.

V. Conclusion

The ADN model can be seen either as a formalism belonging to the Predicate/Transition net family in which tokens are terms of some algebraically defined abstract data type, or as a graphical representation of an algebraic specification.

For the expressiveness aspects, ADNs brings both the advantages of Petri nets for the operational aspects, and of abstract data types, for abstract data handling. On the verification side, it is not clear yet what dedicated verification methods for ADNs would be, nor if such methods are needed since one can transform ADNs into algebraic specifications, in any case, tools and methods derived for Petri nets seem to be of few help. These issues should be investigated further.

References

/Bartlett/ Bartlett. K.A., Scantleburry R.A., Wilkinson P.T.,
"A note on reliable full-duplex transmission over half-duplex links", Commun. Ass. Comput. Mach., vol.12 n°5, May 1969.

/Berthomieu/ Berthomieu B.,
"Perspectives sur quelques modèles mixtes à base de réseaux de Petri et de types abstraits algébriques", Contrat DAII/Alcatel/LdM-CGE/LAAS-CNRS, No.84.35.087, doc GTS.P.3.7, Toulouse, March 1985.

/Biebow/ Biebow B., Choquet N., Mauboussin A.
"Spécification par types abstraits algébriques de fonctions caractéristiques d'une Unité de Raccordement d'Abonnés", marché DAII/Alcatel/LdM-CGE/LAAS-CNRS, No. 84.35.087, doc GTS.P.3.12, Marcoussis, 1985.

/Cerf/ Cerf V.G.,
"Multiprocessors, Semaphores, and a graph Model of Computation", PhD Dissertation, UCLA Computer Science Dept., April 1972.

/CCITT/ "Specifications of signalling system No. 7", Recommandation Q.704, Red book, vol.VI, fasc. VI.7, October 1984.

/Choquet1/ Choquet N., Colin C., Martin J.-M., Mauboussin A., "Spécifications en modèles mixtes, réseaux de Petri et types abstraits algébriques, de fonctions caractéristiques d'une unité de raccordement d'abonnés", marché DAII/Alcatel/LdM-CGE/LAAS-CNRS, No.84.35.087, doc GTS.P.3.13, 1985.

/Choquet2/ Choquet N., Fribourg L., Mauboussin A., "Runnable Protocol Specifications Using the logical Interpreter SLOG", 5th IFIP Workshop on Protocol Specifications, Verification and Testing, Toulouse_Moissac, June 1985.

/Genrich/ Genrich H.J., Lautenbach K., Thiagarajan P.S.. "Predicate/Transition nets", Lecture notes in computer sciences" n° 84, Springer Verlag.

/Goguen/ Goguen J., Thatcher J., Wagner J. "An initial approch to the specification, correctness, and implementation of abstract data types", Current Trends in Programming Methodology, Vol 4 Prentice Hall 1978.

/Guttag/ Guttag J.V., Horning J.J., "The Algebraic Specification of Abstract Data Type", Acta Informatica, vol.10, n°1, pp27-52, 1978.

/Kaplan/ Kaplan S., "Conditional Rewrite Rules", Theoretical Computer Science 33, December 1984.

/Keller/ Keller R.M., "Formal Verification of Parallel Programs", Communications of the ACM, July 1976.

/Krämer/ Krämer B.. "Stepwise Construction of Non-Sequential Software Systems Using a Net-Based Specification Language". Advances in Petri Nets 1984, L.N.C.S. 188, G.Rozenberg Ed., p. 307-327. Springer-Verlag (1985).

/Paul/ Paul E., Manuel OASIS, Rapport Greco n°6.85, Fev. 1985.

/Thompson/ Thompson D.H., Sunshine C.A., Erickson R.W., Gerhart S.L., Schwabe D.,
"Specification and verification of Communication Protocols in AFFIRM using state transition models", IEEE Transaction on Software Engineering, SE-8.5, p.460-489, 1982,

/Valette/ Valette R.,
"Sur la description, l'analyse et la validation des systèmes de commande parallèles", Thèse d'état, UPS Toulouse, November 1976.

/Vautherin/ Vautherin J.,
"Un modèle algébrique basé sur les réseaux de Petri, pour l'étude des systèmes parallèles", Thèse de Doctorat d'Ingénieur, Université de Paris-Sud, 1985.