Technical Report

UCAM-CL-TR-148
ISSN 1476-2986

Number 148

Extending coloured petri nets

Jonathan Billington

September 1988

ISSN 1476-2986

Abstract

Jensen's Coloured Petri Nets (CP-nets) are taken as a starting point for the development of a specification technique for complex concurrent systems. To increase its expressive power CP-nets are extended by including capacity and inhibitor functions. A class of extended CP-nets, known as P-nets, is defined that includes the capacity function and the *threshold* inhibitor extension. The inhibitor extension is defined in a totally symmetrical way to that of the usual pre place map (or incidence function). Thus the inhibitor and pre place maps may be equated allowing a marking to be purged by a single transition occurrence, useful when specifying the abortion of various procedures. A chapter is devoted to developing the theory and notation for the purging of a place's marking or part of its marking.

Two transformations from P-nets to CP-nets are presented and it is proved that they preserve interleaving behaviour . These are based on the notion of complementary places defined for PT-nets and involve the definition and proof of a new extended complementary place invariant for CP-nets.

The graphical form of P-nets, known as a P-Graph, is presented formally and draws upon the theories developed for algebraic specification. Arc inscriptions are multisets of tuples of terms generated by a many-sorted signature. Transition conditions are Boolean expressions derived from the same signature. An interpretation of the P-Graph is given in terms of a corresponding P-net. The work is similar to that of Vautherin, but includes the inhibitor and capacity extensions and a number of significant differences. In the P-Graph concrete sets are associated with places, rather than sorts and likewise there are concrete initial marking and capacity functions. Vautherin associates equations with transitions rather than the more general Boolean expressions. P-Graphs are useful for specification at a concrete level. Classes of the P-Graph, known as Many-sorted Algebraic Nets and Many-sorted Predicate/Transition Nets, are defined and illustrated by a number of examples. An extended place capacity notation is developed to allow for the convenient representation of resource bounds in the graphical form.

Some communications-oriented examples are presented including queues and the Demon Game of international standards fame.

The report concludes with a discussion of future work. In particular, an abstract P-Graph is defined that is very similar to Vautherin's Petri net-like schema, but including the capacity and inhibitor extensions and associating Boolean expressions with transitions. This will be useful for more abstract specifications (eg classes of communication protocols) and for their analysis.

It is believed that this is the first coherent and formal presentation of these extensions in the literature.

Contents

List of Figures

Chapter 1

Introduction

The limitations of Petri Nets for specifying complex systems have been well known for the last 15 years. During the latter part of the 1970's and particularly during the 1980's *high-level* nets [9,11,17,8,10,16] have been developed in an attempt to overcome this problem. It is also recognised that High-level nets also have their limitations in expressive power and this report tackles this issue.

The report defines a Petri net technique that is suitable for modelling or specifying the dynamic behaviour of complex concurrent systems such as communications protocols and services. The technique is based on Jensen's Coloured Petri nets (CP-nets) [11,10]. We extend the modelling convenience by introducing capacity and inhibitor functions and show how these can be mapped back to CP-nets, with certain restrictions for the inhibitor function.

The motivation for the introduction of the inhibitor function is to provide for more compact descriptions and to allow for the manipulation of markings by a single transition occurrence. For example, the (partial) *purging* of places, ie when we wish to empty a place of all (or a subclass) of its tokens. The second last chapter is devoted to this topic.

A graphical form of P-nets, the P-Graph, is defined where arcs and transitions are annotated by terms built from a many-sorted algebra. The notation is very similar to that of Predicate/Transition nets (PrT nets) [9,8], and the approach is similar to that of [22].

Many-sorted Algebraic Nets (MAN) and many-sorted PrT nets (MPrT nets) are derived as subclasses of the P-Graph. These are compared with the Algebraic Nets of [16] and PrT nets [8]. It is shown how the difficulties of partial functions and loose typing of variables that are encountered with Algebraic nets and PrT nets are overcome with their many-sorted versions.

Many examples are included to illustrate the technique, including the Demon Game of [1], which is intended as a test example for specification techniques for communication protocols. The Demon Game can be specified on a single page using an MPrT net and it provides a focus for the debate on interleaving versus truly concurrent specification techniques.

The ideas formalised here stem from the author's association with the development of Numerical Petri Nets (NPNs) [18,19,4,23,5] for the specification of communication protocols and services. A brief comparison of Numerical Petri Nets and P-nets is provided in Appendix B.

The name **P-net** has been coined for the net defined in this report, to reflect its genesis in the specification of protocols (P-net is an abbreviation for Protocol-net), but also because it is sufficiently different from other *high-level* nets to warrant a new name. Although P-nets have been specifically tailored for the protocol domain, it is believed that they are generally applicable for the modelling of distributed systems including the functional specification of telecommunication systems and services.

The report assumes a familiarity with nets [17,15,2] and also makes extensive use of multisets. Multisets may be considered as a special class of vectors, sometimes called *weighted-sets*. Definitions of sets together with multiset and vector notation and operations are gathered together in Appendix A. The definition of many-sorted signatures and algebras and their associated terminology has been included in the development of the P-Graph to keep the report self-contained.

1.1 P-net Design

An aim of this report is the development of a formal technique to be used in systems engineering and in particular for protocol engineering. This section briefly discusses some of the requirements of such a technique.

The development of P-nets has been guided by the following conflicting requirements:

- Expressive ability
- Analytic power
- Simplicity

There are two parts to expressive ability: modelling power and modelling convenience. Modelling power is the ability of a technique to model a class of systems (cf Chomsky hierarchy), whereas modelling convenience refers to the elegance or conciseness of expression of a property. It is well known [15] that there is a trade off between modelling power and analytic power. (We increase modelling power at the expense of our ability to analyse the model; more questions become undecidable.) In general we would like there to be just enough modelling power for our requirements (so that analytic power can be maximised).

It is essential that the technique be able to express all of the properties that we wish to express about protocols and their services. P-nets include the inhibitor extension which raises its modelling power to that of a Turing Machine. This allows us to model any system that is implementable. It turns out that most protocols do not require such modelling power. The technique allows for a whole range of modelling power, from state machines to Turing Machines, dependent on net structure and the use of inhibitors. The appropriate modelling power can then be chosen to suit the application.

We would also like to increase the modelling convenience of the technique, so that important elements in the application can be modelled relatively easily. We can add constructs (notation) to do this which can be defined in terms of the basic elements of the technique. The addition of too many constructs makes the language more complicated and difficult to learn and we need to strive for a few powerful constructs.

P-nets have been developed with these goals in mind. When considering a language for protocol engineering, we need to consider the perspectives of the specifier, analyser and implementer of protocols. It is hoped that P-nets provide a reasonable compromise.

1.2 The Nature of High-Level Nets

In applications we often find that we wish to model records (vectors) of information. For example, in protocol specification we wish to model messages (Service Data Units (SDUs) and Protocol Data Units (PDUs)) and compound state information (major states, housekeeping variables, etc). Thus information is structured. It is useful to be able to express this structure in our specification language.

In Place/Transition Nets, this structure can only be expressed by complex labelling of places by values. One place is required for each value of the domain of a data structure and there is no means of grouping places associated with the same data structure. The structure is lost in an amorphous sea of net elements. This also leads to an explosion of PT-net elements for even moderately complex applications, rendering the graphical form useless.

High-level nets [9,8,11,10] have overcome this problem by providing a mechanism for grouping sets of (PT-net) places (transitions) together and considering them as entities in their own right, but still usually referred to as (high-level) places (transitions). The places in the high-level net are now typed (implicitly or explicitly) by the domain of the data structure and tokens residing in the place take on values from the domain. Tokens may now have a structure and because they represent a value are no longer anonymous and are often referred to as *individual* tokens. The grouping of transitions allows sets (schemes) of similar actions to be referred to by the one transition. The inscriptions on arcs are no longer integers but involve multisets of terms, which when evaluated are multisets over the domain associated with the place. The price paid for structuring the net is the increased complexity of net inscriptions. The considerable advantage is being able to model systems in a much more compact and natural way. For example PDUs and SDUs, compound states and queues can all be represented by tokens in appropriately typed places.

Coloured Petri nets (CP-nets) [11,10] have been chosen as the basis for the development of P-nets because of their generality (arbitrary grouping of places and transitions, cf restricted arity of predicates in PrT nets); their transparent relationship to PT-nets; the possibility of adding an abstract data type within the same framework [22]; and the increasing range of analysis possibilities [21,14].

Chapter 2

Coloured Petri Nets

This chapter introduces the basic features of Coloured Petri Nets, including the colouring of places and transitions, their pre and post maps, the net marking and transition rule and culminates in a formal definition. It also relates the definition provided here to the one used by Jensen in [10].

2.1 Colouring Places and Transitions

Consider a set of (high-level) places, S, and transitions, T. We wish to associate a set with each place. Let there be a set $\mathcal{D}$ comprising the sets associated with each place. This set determines a structure on the underlying Place/Transition net. (We allow the sets associated with each place to be complex sets, such as unions of product sets eg if $D \in \mathcal{D}$ then we may have $D = (D_1 \times D_2) \cup D_3$). We define a *place grouping function*, GP, which associates a set in $\mathcal{D}$ with a place in S.

$$GP : S \longrightarrow \mathcal{D}$$

We also associate a set of *occurrence modes* with each transition and in a similar way, define the *transition grouping function*

$$GT : T \longrightarrow \mathcal{O}$$

where $\mathcal{O}$ is the set of all the occurrence mode sets. (In most applications, we can derive $\mathcal{O}$ from $\mathcal{D}$)

For economy of definition, we can use a single grouping function, called the *Colour Function* by Jensen [10].

$$C : S \cup T \longrightarrow \mathcal{C}$$

where $\mathcal{C} = \mathcal{D} \cup \mathcal{O}$ and $C(s) = GP(s)$ and $C(t) = GT(t)$. Jensen uses the term *occurrence-colours* for *occurrence modes* and the term *token-colours* to describe the members of a set in $\mathcal{D}$. In this report the terms *transition modes* and *occurrence modes* will be used interchangeably for *occurrence-colours*. We shall refer to the sets in $\mathcal{C}$ as *colour sets*.

2.2 Structure of Colour Sets

It should be emphasised that the colour sets may be quite complex and contain highly structured elements or multisets over other sets.

As an example, we may structure places by letting $C(s)$ be an arbitrary product set. For all $s \in S$ let

$$C(s) = D_1(s) \times D_2(s) \times \ldots \times D_{n_s}(s)$$

This may be compared with Predicate/Transition nets, where only a single arity (n_s) is allowed for a particular place. Note that our approach is more general. (We could allow $C(s)$ to be the union of a set of product sets, for example.) Places are **not** restricted to a single arity (as we have here) as arbitrary grouping of the underlying places and transitions is allowed. In many applications, a single arity for a place is all that is needed, however for protocols, there are times when we may like to have tuples of differing lengths associated with the same place. For example, it is convenient to allow a place to represent a set of queues in an underlying service. Items in the queue will be Service Data Units which will comprise messages which in general will need to be represented by tuples of differing lengths.

2.3 Pre and Post Maps

We follow the approach taken in [24] defining pre and post mappings without the intermediate step of defining a set of arcs.

We define two sets:

$$TRANS = \{(t, m) \mid m \in C(t), t \in T\}$$

$$PLACE = \{(s, g) \mid g \in C(s), s \in S\}$$

$TRANS$ is the set of transitions in the *unfolded* PT-net and likewise $PLACE$ is the unfolded set of places.

At times it will be useful to consider the above sets partitioned with respect to the set of transitions and places. Thus we define

$$TRANS_t = \{(t, m) \mid m \in C(t)\}$$

$$PLACE_s = \{(s, g) \mid g \in C(s)\}$$

where

$$TRANS = \bigcup_{t \in T} TRANS_t$$

$$PLACE = \bigcup_{s \in S} PLACE_s$$

We form the set of multisets over $TRANS$ and $PLACE$ and denote them $\mu TRANS$ and $\mu PLACE$ respectively.

We may now state the relationship between places and transitions by two multirelations:

$$Pre, Post : TRANS \longrightarrow \mu PLACE$$

- T is a finite set of transitions disjoint from S ($S \cap T = \emptyset$)
- C is a finite set of non-empty colour sets
- $C : S \cup T \longrightarrow C$ is the colour function used to structure places and transitions
- $Pre, Post : TRANS \longrightarrow \mu PLACE$ are the pre and post mappings with

$$TRANS = \{(t, m) \mid m \in C(t), t \in T\}$$

$$PLACE = \{(s, g) \mid g \in C(s), s \in S\}$$

- $M_0 \in \mu PLACE$ is a multiset known as the initial marking

Marking

A **Marking** is a multiset, $M \in \mu PLACE$.

Enabling

A finite multiset of transitions $T_\mu \in \mu TRANS$ is *enabled* at a marking M *iff*

$$Pre'(T_\mu) \subseteq M$$

Thus a multiset of transitions is enabled if there are enough tokens on the input places to satisfy the pre map.

Transition Rule

Given that a multiset of transitions T_μ is enabled at a marking M, then a *step* may occur resulting in a new marking M' given by

$$M' = M - Pre'(T_\mu) + Post'(T_\mu).$$

This is often denoted by $M[T_\mu > M'$.

Set of Reachable Markings

The set of reachable markings, $[M_0>$, of CP is obtained inductively as follows.

- $M_0 \in [M_0>$; and
- if $M_1 \in [M_0>$ and $M_1[tr>M_2$ for $tr \in TRANS$, then $M_2 \in [M_0>$.

2.4 Net Marking and Transition Rule

A marking multiset can now be defined for all places by

$$M \in \mu PLACE$$

and we shall denote the initial marking by M_0.

Sometimes it will be useful to consider the marking of a particular place, s. This can be achieved by partitioning the marking multiset according to places. We define, for $s \in S$ and for all $g \in C(s)$,

$$M_s \in \mu\{(s,g) \mid (s,g) \in M\}$$

such that $mult((s,g), M_s) = mult((s,g), M)$. ($mult(x, A)$ is the multiplicity of x in the multiset A - see Appendix A.) We have $\sum_{s \in S} M_s = M$.

We can consider places to be *marked* by a multiset, $M(s) \in \mu C(s)$, where $\forall g \in C(s)$, $mult(g, M(s)) = mult((s,g), M)$. $M(s)$ is often considered as a multiset of *tokens* which mark place s at marking M. Thus for any $s \in S$, a token is a member of $C(s)$ (cf *token-colour* used by Jensen).

The transition rule follows immediately from [24]. We consider a finite multiset of transitions, $T_\mu \in \mu TRANS$. By a linear extension of the above multirelations, we have

$$Pre', Post' : \mu TRANS \longrightarrow \mu PLACE$$

where

$$P'(T_\mu) = \sum_{tr \in TRANS} mult(tr, T_\mu) P(tr)$$

with $P = Pre$ or $Post$, $\sum$ refers to multiset addition and the multiplication can be viewed as scalar multiplication of a vector (see Appendix A).

We may now define a *step* (the simultaneous occurrence of a finite multiset of transition modes) as follows:

$$M[T_\mu > M' \text{ iff } Pre'(T_\mu) \subseteq M$$

and

$$M' = M - Pre'(T_\mu) + Post'(T_\mu)$$

where '$-$' and '$+$' are interpreted as multiset subtraction and addition respectively.

2.5 Definition of CP-nets

We are now in a position to provide a definition of CP-nets as a summary of the discussion above.

Definition

A **CP-net** is a structure $CP = (S, T, C; C, Pre, Post, M_0)$ where

- S is a finite set of places

2.6 Relationship to Jensen's CP-nets

CP-nets as defined above are very closely related to the way Jensen defines his CP-Matrix [10]. The main differences are:

1. The *structuring* set of colour sets, C, is included in the structure CP.
2. The *Pre* and *Post* mappings are defined in general for the whole net, rather than a set of functions, one for each arc.
3. The empty net ($S \cup T = \emptyset$) and isolated elements are allowed.

We shall now relate the pre and post mappings to Jensen's *positive* and *negative incidence* functions, $I_-(s,t)$ and $I_+(s,t)$.

We define the following functions

$$Pre(s,t), Post(s,t) : C(t) \longrightarrow \mu C(s)$$

and

$$Pre'(s,t), Post'(s,t) : \mu C(t) \longrightarrow \mu C(s)$$

so that $\forall m \in C(t), t \in T$ and $\forall g \in C(s), s \in S$

- $mult(g, Pre(s,t;m)) = mult((s,g), Pre(t,m))$ and
- $mult(g, Post(s,t;m)) = mult((s,g), Post(t,m))$

and similarly for the multiset extensions which are identical to Jensen's *positive* and *negative incidence* functions.

- $Pre'(s,t) = I_-(s,t)$
- $Post'(s,t) = I_+(s,t)$

As the multiset extension of a function includes the function itself, it is only necessary to use the multiset extension function. Jensen adopts this approach. I prefer to retain the original function and explicitly use it when the multiset extension is not required. This is the case when we wish to define the pre and post maps for example. It is hoped that this adds to the clarity of the presentation.

Chapter 3

Extensions to CP-nets

3.1 Place Capacity

In the definition of CP-nets it is assumed that places all have infinite capacity. We can generalise the notion of place capacity for PT systems [2] (and PrT nets [9]) quite easily. We denote the capacity by K, representing a multiset of tokens for each place

$$K \in \mu^{+}_{\infty}PLACE.$$

This capacity cannot be exceeded by the marking: $M \subseteq K$. Specifically, the initial Marking, $M_0 \in \mu PLACE$ satisfies $M_0 \subseteq K$. Note that the capacity may contain elements with infinite multiplicities but zero multiplicities are not allowed.

The enabling condition now becomes

$$M[T_\mu > M' \textit{ iff } Pre'(T_\mu) \subseteq M \subseteq K - Post'(T_\mu)$$

and the transition rule is unchanged. The subtraction used above is vector subtraction and 'inclusion' is vector inclusion (see Appendix A).

We can consider a partition of K according to places in the same way as we did for markings. We define for $s \in S$ and for all $g \in C(s)$,

$$K_s \in \mu\{(s,g) \mid (s,g) \in K\}$$

such that $mult((s,g), K_s) = mult((s,g), K)$ and $\sum_{s \in S} K_s = K$.

It will be useful to consider the capacity of a particular place, s, by defining

$$K(s) \in \mu^{+}_{\infty}C(s)$$

as the multiset of tokens such that $\forall g \in C(s)$

$$mult(g, K(s)) = mult((s,g), K)$$

3.2 Inhibitor Arc Extension

3.2.1 Zero-Testing Inhibitor

There may be times when we would like to have the ability to test places for a null marking. This corresponds to the well known *inhibitor arc* extension to Petri nets which increases its modelling power to that of a Turing machine [15].

We shall denote the power set of a set A by $\mathcal{P}(A)$. We can generalise the notion of an inhibitor arc for high-level nets, by introducing a function

$$I_0 : TRANS \longrightarrow \mathcal{P}(PLACE)$$

which associates with each transition a subset of places that will be used for zero testing.

To obtain a suitable inhibitor condition for a multiset of transitions we define the following function:

$$I_0' : \mu TRANS \longrightarrow \mathcal{P}(PLACE)$$

where $\forall tr \in TRANS$ and $T1_\mu, T2_\mu \in \mu TRANS$ we have

- $I_0'(\emptyset) = \emptyset$
- $I_0'(tr) = I_0(tr)$
- $I_0'(T1_\mu + T2_\mu) = I_0'(T1_\mu) \cup I_0'(T2_\mu)$

Thus for example for $n, m \in N^+$ and $tr, tr1, tr2 \in TRANS$

- $I_0'(ntr) = I_0'(tr) = I_0(tr)$
- $I_0'(ntr1 + mtr2) = I_0'(tr1) \cup I_0'(tr2) = I_0(tr1) \cup I_0(tr2)$

The enabling condition is then formulated as the conjunction of two predicates: the CP-net enabling predicate (section 2.5) and the inhibitor predicate, $(M \cap I_0'(T_\mu) = \emptyset)$ as follows.

A finite multiset of transitions, $T_\mu \in \mu TRANS$ is enabled by a marking M *iff*

$$(Pre'(T_\mu) \subseteq M) \wedge (M \cap I_0'(T_\mu) = \emptyset)$$

Thus a multiset of transitions is enabled if there are enough tokens on the input places to satisfy the pre maps, and no tokens reside on the inhibitor places.

The transition rule remains unchanged.

3.2.2 Threshold Inhibitor

For modelling convenience, we would like to introduce a *threshold* inhibitor condition, which instead of requiring that certain tokens must be absent from the marking as above, requires that certain tokens must not exceed a preset multiplicity, known as the *threshold.* We do this by generalising the above inhibitor function to associate a general multiset of place colours (the thresholds) with each transition mode.

$$I : TRANS \longrightarrow \mu_\infty PLACE$$

This implies that the multiplicity of tokens not having a threshold will be infinite.

We can again extend this function to multisets of transitions by defining

$$I' : \mu TRANS \longrightarrow \mu_{\infty} PLACE$$

where $\forall tr \in TRANS$, and $T1_\mu, T2_\mu \in \mu TRANS$ we have

- $I'(\emptyset) = \{(p, \infty) \mid p \in PLACE\}$
- $I'(tr) = I(tr)$
- $I'(T1_\mu + T2_\mu) = I'(T1_\mu) \cap I'(T2_\mu)$

The first item ensures that $I'(T1_\mu + \emptyset) = I'(T1_\mu)$.

As above, the enabling condition is then formulated as the conjunction of two predicates. A finite multiset of transitions, $T_\mu \in \mu TRANS$, is enabled by a marking M *iff*

$$Pre'(T_\mu) \subseteq M \subseteq I'(T_\mu)$$

Thus a multiset of transitions is enabled if there are enough tokens on the input places to satisfy the pre maps and the thresholds are not exceeded for the inhibitor places. Again the transition rule remains unchanged.

For the graphical form it will be convenient to define an inhibitor map for each place-transition pair as follows: for $s \in S$ and $t \in T$,

$$I(s,t) : C(t) \longrightarrow \mu C(s)$$

where $\forall m \in C(t), t \in T$ and $\forall g \in C(s), s \in S$

$$mult(g, I(s,t;m)) = mult((s,g), I(t,m))$$

3.3 P-nets

We are now in a position to define *Protocol Nets*, abbreviated to **P-nets**. P-nets are CP-nets extended by the capacity function and the threshold inhibitor map. As suggested in [12] we could use CP_{KI}-nets as the name to avoid a proliferation of names of high-level nets. After considering a number of options it was felt that the advantage of brevity; that P-net can be considered an abbreviation for CP_{KI}-net; and the mild link back to the application domain of protocols where some of the ideas for the extensions arose; were sufficient reasons to retain the name P-nets. One may also consider CP_{KI}-nets to be a broader class where the inhibitor extension may be defined differently, whereas with P-nets the inhibitor extension is the threshold inhibitor as defined above.

3.3.1 Definition

A **P-net**, is the structure

$$P = (CP, I, K)$$

where

- CP is a CP-net as defined in section 2.5 with initial marking restricted to comply with the capacity multiset: $M_0 \subseteq K$.
- $I : TRANS \longrightarrow \mu_\infty PLACE$ is the threshold inhibitor map; and
- $K \in \mu_\infty^+ PLACE$ is a multiset known as the place capacity.

3.3.2 Marking

A **Marking** is a multiset, $M \in \mu PLACE$, the same as for CP-nets.

3.3.3 Enabling

A finite multiset of transitions $T_\mu \in \mu TRANS$ is *enabled* at a marking M *iff*

$$(Pre'(T_\mu) \subseteq M \subseteq K - Post'(T_\mu)) \wedge (M \subseteq I'(T_\mu))$$

Thus a multiset of transitions is enabled if there are enough tokens on the input places to satisfy the pre map, there is enough capacity left in the output places, and the inhibitor thresholds are not exceeded.

3.3.4 Transition Rule

The transition rule is the same as for CP-nets and is given in section 2.5.

$$M' = M - Pre'(T_\mu) + Post'(T_\mu)$$

3.3.5 Reachable Markings

This is again defined in exactly the same way as for CP-nets (see section 2.5).

3.3.6 Special Cases

There are three obvious special cases of P-nets.

1. Capacity CP-nets (CP_K-nets)

   When $\forall tr \in TRANS, \forall p \in PLACE, mult(p, I(tr)) \geq mult(p, K - Post'(T_\mu))$, then $M \subseteq I(T_\mu)$ is guaranteed by $M \subseteq K - Post'(T_\mu)$, and the threshold inhibitor is redundant. In this case $P = (CP, K)$.

2. Inhibitor CP-nets (CP_I-nets)

   When the capacities of the places are infinite ($\forall p \in PLACE, K(p) = \infty$), the P-net becomes an inhibitor CP-net, $P = (CP, I)$.

3. CP-nets

   When the capacities of the places and the thresholds are infinite, ie

   $(\forall p \in PLACE, K(p) = \infty) \wedge (\forall tr \in TRANS, \forall p \in PLACE, mult(p, I(tr)) = \infty)$,

   the P-net reduces to a CP-net, $P = CP$.

Chapter 4

Transforming P-nets to CP-nets

To allow the analysis techniques that have been and are being developed for CP-nets to be directly applied to P-nets it is important to be able to transform P-nets to CP-nets and to know precisely under which circumstances these transformations are applicable. This chapter sets out two transformations under which the interleaving behaviours of the P-net and CP-net are equivalent in the sense that there single-step reachability trees are isomorphic.

To motivate the transformations, it is first necessary to establish an *extended complementary place invariant* for CP-nets, not previously published in the literature.

4.1 Extended Complementary Place Invariant

In this section we consider a class of CP-nets in which the set of places is partitioned into two sets of the same cardinality such that for each place in one set there is a corresponding place in the other. We also relate the pre and post maps restricted to one set of places to the pre and post maps restricted to the other set, in such a way that an invariant exists between the markings of the two sets of places.

This development has been inspired by the idea of complementation for PT nets [17]. We shall therefore call the corresponding set of places, *complementary* places. When complementing a PT net the pre map (post map) on the complementary places is set equal the post map (pre map) on the original set of places. This guarantees an invariant on each pair of complementary places. If p is an original place and $\hat{p}$ its complement, then for every reachable marking M, $M(p) + M(\hat{p}) = K(p)$ where $K(p) = M_0(p) + M_0(\hat{p})$ is a constant, the capacity of place p.

The following generalises this idea in two ways. Firstly we relax the relationship between the pre and post maps (we are only concerned with the equality of the differences in the pre and post maps) and secondly we raise these notions to the level of CP-nets.

4.1.1 Definitions

Let **CP1** $= (S1, T, C; C, Pre1, Post1, M1_0)$ be a coloured net where its components are defined in figure 4.1. The hat notation is used to indicate complementary places or sets or

$S1 = S \cup \hat{S}$
where $\hat{S} = \{\hat{s} \mid s \in S\}$
$\forall s \in S, C(\hat{s}) = C(s)$
$PLACE1 = PLACE \cup \widehat{PLACE}$
$PLACE = \{(s,g) \mid g \in C(s), s \in S\}$
$\widehat{PLACE} = \{(\hat{s},g) \mid g \in C(s), s \in S\}$
The pre and post maps are factored: $\forall tr \in TRANS$
$Pre1(tr) = Pre(tr) + \widehat{Pre}(tr)$
$Post1(tr) = Post(tr) + \widehat{Post}(tr)$
$Pre, Post : TRANS \longrightarrow \mu PLACE$
$\widehat{Pre}, \widehat{Post} : TRANS \longrightarrow \mu\widehat{PLACE}$
$M1_0 = M_0 + \widehat{M}_0$
where $M_0 \in \mu PLACE$ and $\widehat{M}_0 \in \mu\widehat{PLACE}$

Figure 4.1: **CP1**: Basic Definition

functions associated with complementary places. An overbar notation is defined in figure 4.2. It is used to complement the marking or pre and post maps associated with the original places.

Let $p = (s,g) \in PLACE$ and $\hat{p} = (\hat{s},g) \in \widehat{PLACE}$
If $X \in \mu PLACE$ then $\overline{X} \in \mu\widehat{PLACE}$ such that
$\forall p \in PLACE,\ \forall \hat{p} \in \widehat{PLACE},\ \overline{X}(\hat{p}) = X(p)$

Figure 4.2: Overbar Notation

We now state the restriction on the pre and post maps which guarantees that the complementary place invariant holds. $\forall tr \in TRANS$

$$\overline{Pre}(tr) - \overline{Post}(tr) = \widehat{Post}(tr) - \widehat{Pre}(tr) \tag{4.1}$$

4.1.2 Complementary Place Invariant

Let $M1 \in \mu PLACE1$, a reachable marking of **CP1**, be factored so that $M1 = M + \widehat{M}$ where $M \in \mu PLACE$ and $\widehat{M} \in \mu\widehat{PLACE}$. For convenience, let $\overline{M_0} + \widehat{M}_0 = \overline{K}$ where $\overline{K} \in \mu\widehat{PLACE}$, then the invariant is given in the following proposition.

Proposition 4.1 *For* **CP1** *above, satisfying equation 4.1,*

$$\forall M1 \in [M1_0\rangle, \overline{M} + \widehat{M} = \overline{K}$$

Proof

The proof is by induction over the reachable markings. The proposition is true by definition for the initial marking. Given any reachable marking $M1 = M + \widehat{M}$, we assume that

$$\overline{M} + \overline{\widehat{M}} = \overline{K} \tag{4.2}$$

and then prove it is true for any follower marking.

$\forall tr \in TRANS$ such that $Pre1(tr) \subseteq M1$, the follower marking, $M1'$, is given by the transition rule:

$$\begin{aligned}
M1' &= M1 + Pre1(tr) - Post1(tr) \\
&= M + \widehat{M} + Pre(tr) + \widehat{Pre}(tr) - Post(tr) - \widehat{Post}(tr) \\
&= M + Pre(tr) - Post(tr) + \widehat{M} + \widehat{Pre}(tr) - \widehat{Post}(tr) \\
&= M' + \widehat{M}'
\end{aligned}$$

where

$$M' = M + Pre(tr) - Post(tr) \tag{4.3}$$
$$\widehat{M}' = \widehat{M} + \widehat{Pre}(tr) - \widehat{Post}(tr) \tag{4.4}$$

Thus we need to prove that given equation 4.2

$$\overline{M'} + \overline{\widehat{M}'} = \overline{K} \tag{4.5}$$

Substituting for M' and $\widehat{M}'$ using equations 4.3 and 4.4, rearranging and using equation 4.1 gives the required result.

$$\begin{aligned}
\overline{M'} + \overline{\widehat{M}'} &= \overline{M} + \overline{Pre}(tr) - \overline{Post}(tr) + \overline{\widehat{M}} + \overline{\widehat{Pre}}(tr) - \overline{\widehat{Post}}(tr) \\
&= \overline{M} + \overline{\widehat{M}} + \overline{Pre}(tr) - \overline{Post}(tr) + \overline{\widehat{Pre}}(tr) - \overline{\widehat{Post}}(tr) \\
&= \overline{M} + \overline{\widehat{M}} \\
&= \overline{K}
\end{aligned}$$

□

4.2 Interleaving Equivalence of P-nets and CP-nets

In this section we give transformations from P-nets to CP-nets which preserve their interleaving behaviour and show that there is an isomorphism between the single step (interleaving) reachability trees of P-nets and CP-nets, under weak assumptions.

The transformations are important as they allow the theory developed for the analysis of CP-nets (high-level reachability trees and invariants analysis [10]) to be applied to P-nets in most practical situations.

We firstly consider the most straightforward case where the capacities of all places in the P-net are finite and then relax this condition when there is no inhibitor condition associated with an underlying place.

4.2.1 Complete Complementation Transformation

A P-net, $P = (S, T, C; C, Pre, I, Post, K, M_0)$, with the restriction that

$$\forall tr \in TRANS, \forall p \in PLACE, mult(p, I(tr)) < \infty \Rightarrow mult(p, K) < \infty \tag{4.6}$$

can be transformed into CP-net $CP = (\underline{S}, \underline{T}, \underline{C}; \underline{C}, \underline{Pre}, \underline{Post}, \underline{M_0})$, where we have used the underline to denote the CP-net elements.

Firstly we define a combined inhibitor-capacity function. For a P-net, the enabling condition for $tr \in TRANS$, at marking M, comprises

1. the *precondition* $Pre(tr) \subseteq M$
2. the *capacity condition* $M \subseteq K - Post(tr)$ and
3. the *inhibitor condition* $M \subseteq I(tr)$

We identify two cases, for $\forall p \in PLACE$ and $\forall tr \in TRANS$:

- C1: $mult(p, I(tr)) \geq mult(p, K - Post(tr))$
- C2: $mult(p, I(tr)) < mult(p, K - Post(tr))$

It will be convenient to combine enabling conditions 2 and 3 above and we therefore define the following inhibitor-capacity function:

$$IK : TRANS \longrightarrow \mu PLACE$$

where $\forall tr \in TRANS, IK(tr) \subseteq I(tr)$ such that

$$mult(p, IK(tr)) = \begin{cases} mult(p, K - Post(tr)) & \text{if C1} \\ mult(p, I(tr)) & \text{if C2} \end{cases}$$

The enabling condition becomes

1. $Pre(tr) \subseteq M$
2. $M \subseteq IK(tr)$

We shall replace the second enabling condition by an equivalent pre map on a set of complementary places in the CP-net. The construction is as follows.

Firstly we impose the restriction that all places have finite capacity

$$\forall p \in PLACE, mult(p, K) < \infty$$

For each $s \in S$, we create a complementary place $\hat{s}$ and gather them together into a set of complementary places, $\hat{S} = \{\hat{s} \mid s \in S\}$. The set of places of the CP-net is then $\underline{S} = S \cup \hat{S}$.

We denote the set of underlying complementary places by $\widehat{PLACE}$, given by

$$\widehat{PLACE} = \{(\hat{s}, g) \mid g \in C(s), s \in S\}$$

$$\boxed{\begin{array}{l}
\underline{S} = S \cup \hat{S} \\
\underline{T} = T \\
\underline{C} = C \\
\underline{C} : S \cup \hat{S} \cup T \longrightarrow C \\
\text{where } \forall s \in S, \forall t \in T, \underline{C}(s) = C(s), \underline{C}(\hat{s}) = C(s), \underline{C}(t) = C(t) \\
\text{The pre and post maps: } \forall tr \in TRANS \\
\underline{Pre}(tr) = Pre(tr) + \overline{K} - \overline{IK}(tr) \\
\underline{Post}(tr) = Post(tr) + \overline{Pre}(tr) - \overline{Post}(tr) + \overline{K} - \overline{IK}(tr) \\
\underline{M_0} = M_0 + \overline{K} - \overline{M_0}
\end{array}}$$

Figure 4.3: P-net to CP-net Transformation

and use the notation defined in figure 4.2 for complementing multisets associated with P-net places.

Let $\widehat{M}$ denote the marking of the complementary places $\hat{S}$, so that $\widehat{M} \in \mu\widehat{PLACE}$. The initial marking and pre and post maps will be chosen so that the following invariant holds

$$\overline{M} + \widehat{M} = \overline{K} \tag{4.7}$$

The $\underline{Pre}$ and $\underline{Post}$ maps are factored with respect to the original set of places and the set of complementary places and thus $\forall tr \in TRANS$

$$\begin{align} \underline{Pre}(tr) &= Pre(tr) + \widehat{Pre}(tr) \tag{4.8} \\ \underline{Post}(tr) &= Post(tr) + \widehat{Post}(tr) \tag{4.9} \end{align}$$

where $\widehat{Pre}, \widehat{Post} : TRANS \longrightarrow \mu\widehat{PLACE}$. From proposition 4.1, the above invariant holds if

$$\forall tr \in TRANS, \overline{Pre}(tr) - \overline{Post}(tr) = \widehat{Post}(tr) - \widehat{Pre}(tr) \tag{4.10}$$

If we complement the second enabling condition and substitute for $\overline{M}$ using equation 4.7, we obtain the equivalent precondition on the complementary places. Thus $\forall tr \in TRANS$

$$\begin{align} M \subseteq IK(tr) &\Leftrightarrow \overline{M} \subseteq \overline{IK}(tr) \\ &\Leftrightarrow \overline{K} - \widehat{M} \subseteq \overline{IK}(tr) \\ &\Leftrightarrow \overline{K} - \overline{IK}(tr) \subseteq \widehat{M} \tag{4.11} \end{align}$$

Thus $\forall tr \in TRANS, \widehat{Pre}(tr) = \overline{K} - \overline{IK}(tr)$ which then gives us $\underline{Pre}$(tr) from equation 4.8. Finally, knowing $\widehat{Pre}(tr)$, the post map is derived from equations 4.10 and 4.9.

The transformation is summarized in figure 4.3.

4.2.2 Proof of Interleaving Equivalence

The interleaving behaviours of the P-net and corresponding CP-net defined above are equivalent in the sense that their reachability trees (transition systems) are isomorphic. This can be stated as

Theorem 4.1 *1. For each reachable marking, $M \in [M_0\rangle$ of P, there is a one-to-one correspondence with a reachable marking $\underline{M} \in [\underline{M_0}\rangle$ of CP. That is there is a bijection:*

$$\rho : [M_0 > \longrightarrow [\underline{M_0} >$$

where $\underline{M} = \rho(M) = M + \overline{K} - \overline{M}$; and

2. The single step occurrences of transition modes in P and CP are in one-to-one correspondence:

$$M \xrightarrow{tr} M' \text{ iff } \rho(M) \xrightarrow{tr} \rho(M')$$

Proof:

The proof is by induction over the reachable markings. Point 1 is true for the initial marking by definition: $\underline{M_0} = M_0 + \overline{K} - \overline{M_0}$ (see figure 4.3).

Assume

$$\underline{M} = M + \overline{K} - \overline{M} \tag{4.12}$$

We firstly need to prove that if a transition mode, tr, is enabled at M (in P), then it is also enabled at $\underline{M}$ (in CP) and vice versa. This is formally stated in the following lemma.

Lemma 4.1 Enabling Lemma

$$Pre(tr) \subseteq M \subseteq IK(tr) \text{ iff } \underline{Pre}(tr) \subseteq \underline{M}$$

Proof:

Starting with the CP-net, using equation 4.12 and substituting for the definition of $\underline{Pre}(tr)$ reveals

$$\begin{array}{ll} & \underline{Pre}(tr) \subseteq \underline{M} \\ \Leftrightarrow & \underline{Pre}(tr) \subseteq M + \overline{K} - \overline{M} \\ \Leftrightarrow & (Pre(tr) + \overline{K} - \overline{IK}(tr)) \subseteq (M + \overline{K} - \overline{M}) \\ \Leftrightarrow & Pre(tr) \subseteq M \text{ and } \overline{M} \subseteq \overline{IK}(tr) \\ \Leftrightarrow & Pre(tr) \subseteq M \subseteq IK(tr) \end{array}$$

which has proved the enabling lemma. □

We now prove 1 and 2 together in two parts. Firstly we prove that ρ is an injection and the implication on the transition systems.

The enabling lemma tells us that if tr of P is enabled at M, then tr of CP is enabled at $\underline{M} = M + \overline{K} - \overline{M}$. Consider the successor markings

- $M \xrightarrow{tr} M'$; and
- $\underline{M} \xrightarrow{tr} \underline{M'}$

Assuming equation 4.12, we wish to prove that $\forall tr \in TRANS$

$$M \xrightarrow{tr} M' \Rightarrow M + \overline{K} - \overline{M} \xrightarrow{tr} M' + \overline{K} - \overline{M}'$$

From the transition rule and the definitions of the pre and post maps for the CP-net, we have

$$\begin{aligned}
\underline{M}' &= M + \overline{K} - \overline{M} - \underline{Pre}(tr) + \underline{Post}(tr) \\
&= M + \overline{K} - \overline{M} - Pre(tr) + Post(tr) + \overline{Pre}(tr) - \overline{Post}(tr) \\
&= M - Pre(tr) + Post(tr) + \overline{K} - \overline{M} + \overline{Pre}(tr) - \overline{Post}(tr) \\
&= M' + \overline{K} - \overline{M}'
\end{aligned}$$

Hence we have proved the one-way implication and also that ρ is an injection. We now prove the reverse implication of 2 and that ρ is surjective. The proof has exactly the same form as the previous proof.

Let R be the marking we get when a transition occurs in the P-net for mode tr at Marking M. We need to prove $\forall tr \in TRANS$

$$M + \overline{K} - \overline{M} \xrightarrow{tr} M' + \overline{K} - \overline{M}' \Rightarrow M \xrightarrow{tr} M'$$

and hence that $R = M'$.

From the transition rule, the definitions of the pre and post maps for the CP-net and equation 4.12, we have

$$\begin{aligned}
R &= M - Pre(tr) + Post(tr) \\
&= \underline{M} - \overline{K} + \overline{M} - Pre(tr) + Post(tr) \\
&= \underline{M} - \underline{Pre}(tr) + \underline{Post}(tr) - \overline{K} + \overline{M} - \overline{Pre}(tr) + \overline{Post}(tr) \\
&= \underline{M}' - \overline{K} + \overline{M}' \\
&= M'
\end{aligned}$$

Thus for each successor marking $\underline{M}' = M' + \overline{K} - \overline{M}'$ in CP we have a corresponding marking M' in P, which completes the proof. □

4.2.3 Less Restrictive Transformation

We now remove the restriction that all places must have finite capacity. If case C1 above applies for a $p \in PLACE$, then the inhibitor condition is already satisfied by the capacity condition. In particular, if for some $p \in PLACE$, $mult(p, K) = \infty$, then $\forall tr \in TRANS$, $mult(p, I(tr)) = \infty$, to obey the initial restriction (equation 4.6). In this case only an identity transformation is required.

We shall only create complementary places in the underlying PT net to eliminate the capacity condition, when finite, and the inhibitor condition, when C2 applies. The definition of $\widehat{PLACE}$ is modified to exclude an underlying place corresponding to p when $K(p) = \infty$.

$$\widehat{PLACE}' = \{\hat{p} \mid p \in PLACE \wedge K(p) \neq \infty\}$$

The set of complementary places now becomes

$$\hat{S}' = \{\hat{s} \mid \exists(\hat{s}, g) \in \widehat{PLACE}', s \in S, g \in C(s)\}$$

$\underline{S} = S \cup \hat{S}'$
where $\hat{S}' = \{\hat{s} \mid \exists(\hat{s}, g) \in \widehat{PLACE}', s \in S, g \in C(s)\}$
and $\widehat{PLACE}' = \{\hat{p} \mid p \in PLACE \wedge K(p) \neq \infty\}$
$\underline{T} = T$
$\underline{C} = C \cup \hat{C}$
where $\hat{C} = \{\underline{C}(\hat{s}) \mid \hat{s} \in \hat{S}'\}$
$\underline{C} : S \cup \hat{S}' \cup T \longrightarrow \underline{C}$
where $\forall s \in S, \forall t \in T$,
$\underline{C}(s) = C(s)$
$\underline{C}(\hat{s}) = \{g \mid (\hat{s}, g) \in \widehat{PLACE}'\}$
$\underline{C}(t) = C(t)$
The pre and post maps: $\forall tr \in TRANS$
$\underline{Pre}(tr) = Pre(tr) + \overline{K} - \overline{IK}(tr)$
$\underline{Post}(tr) = Post(tr) + \overline{Pre}(tr) - \overline{Post}(tr) + \overline{K} - \overline{IK}(tr)$
$\underline{M_0} = M_0 + \overline{K} - \overline{M_0}$
where the overline bar is defined in section 4.2.3

Figure 4.4: Less Restrictive P-net to CP-net Transformation

where $\hat{p} = (\hat{s}, g)$. The corresponding colour sets are $\forall s \in S, \underline{C}(s) = C(s)$ and

$$\forall \hat{s} \in \hat{S}', \underline{C}(\hat{s}) = \{g \mid (\hat{s}, g) \in \widehat{PLACE}'\}$$

so that the set of colour sets is $\underline{C} = C \cup \hat{C}$ where $\hat{C} = \{\underline{C}(\hat{s}) \mid \hat{s} \in \hat{S}'\}$.

The overbar notation is changed accordingly so that if $X \in \mu PLACE$ then $\overline{X} \in \mu\widehat{PLACE}'$ such that for $p \in PLACE$, $\forall \hat{p} \in \widehat{PLACE}', \overline{X}(\hat{p}) = X(p)$. Note that when $K(p) = \infty$, there is no corresponding element in $\overline{X}(\overline{X}(\hat{p}) = 0)$.

The desired transformation is given in figure 4.4.

Because the transformation is of the same form as before the proofs carry through to the new transformation. Some care is needed with the enabling lemma in the implication proof when 'unbarring' where we need to note that for $p \in PLACE$ and $\forall tr \in TRANS$

$$mult(p, IK(tr)) = \begin{cases} mult(\hat{p}, \overline{IK}(tr)) & \text{if } \hat{p} \in \widehat{PLACE}' \\ \infty & \text{otherwise} \end{cases}$$

5.1 Informal Introduction

5.1.1 General

The graphical form comprises two parts: a *Graph* which represents the net elements graphically and carries textual inscriptions; and a *Declaration*, defining all the sets, variables, constants and functions that will be used to annotate the Graph part. The declaration may also include the initial marking, the capacity and the colour function if these cannot be inscribed on the graph part due to lack of space.

5.1.2 Initial Marking

The initial Marking is specified for each place by $M_0(s) \in \mu C(s)$, such that for all $g \in C(s)$ and $s \in S$

$$mult(g, M_0(s)) = mult((s, g), M_0).$$

5.1.3 Places

In the usual way we shall represent places by circles (or ellipses). A place s may carry four inscriptions.

- the place name;
- the colour set associated with the place, $C(s)$;
- the place capacity, $K(s)$; and
- the initial marking, $M_0(s)$.

The first three would be inscribed close to the place, whereas the initial marking would be inscribed inside the circle representing the place. $C(s)$, $K(s)$ and $M_0(s)$ can be defined in the Declaration if there is insufficient space in the Graph part. We shall adopt the convention that places not annotated by a capacity multiset will have infinite capacity for all tokens in $C(s)$.

A useful notation for $K(s)$ is given later in section 5.7.

5.1.4 Transitions

Transitions are represented by rectangles, annotated by a name and may be inscribed by a boolean expression, known as the *Transition Condition*. The Transition Condition may only involve the variables of the inscriptions of its surrounding arcs.

5.1.5 Arcs

We shall indicate that a place is related to a transition by the pre map, Pre, by an arrow drawn from the place to the transition and for the post map, $Post$, by an arrow drawn from the transition to the place. If a place is related to a transition by both mappings and they are identical, then this may be shown by an arc with an arrow head at each end. (In this case only a single inscription is required.) No arrow is drawn from place s to transition t if and only if for $t \in T$ and $s \in S$, $\forall g \in C(s), \forall m \in C(t), mult(g, Pre(s,t;m)) = 0$ and similarly for the post map.

The usual convention will be adopted for the representation of the inhibitor map, I. If a place is related to a transition by an inhibitor map, in which the multiplicities of at least one token is not infinite, then this is represented by an edge from the place to the transition with a small circle instead of an arrow head at its destination. Equivalently, no inhibitor arc is drawn from place s to transition t if and only if for $t \in T$ and $s \in S$, $\forall g \in C(s), \forall m \in C(t), mult(g, I(s,t;m)) = \infty$.

The arcs will be annotated with multisets of tuples of terms of appropriate type (determined by the colour function). The terms, which include variables, are built from (the signature of) a many sorted algebra (see section 5.2).

5.1.6 Markings and Tokens

A token is a member of $\bigcup_{s \in S} C(s)$. A Marking of the net may be shown graphically by annotating a place with its multiset of tokens $M(s)$. We therefore need a convenient representation for multisets. We use the symbolic sum or vector representation described in Appendix A. The convention is adopted that all tokens are enclosed in angular brackets. Thus if $g \in M(s)$, $< g >$ would appear written in the circle representing place s. We use the natural numbers greater than one, to represent the multiplicity of the token in $M(s)$. Thus if $mult(g, M(s)) = m_g$ we would represent this by juxtaposition: $m_g < g >$ and this would be written inside the circle representing s. If $m_g = 1$, it would be omitted from the inscription. If g is an n-tuple (for example $g = (a,b,c)$), then we adopt the convention of dropping the parentheses (eg (a,b,c) would be represented by $< a,b,c >$ and not $< (a,b,c) >$.)

5.2 Mathematical Preliminaries

In the graphical form, the P-net maps are represented by inscribing arcs with multisets of tuples of terms involving variables, and transitions with Boolean expressions. Many-sorted algebras provide an appropriate mathematical framework for this representation. Signatures provide a convenient way to characterise many-sorted algebras at a syntactic level. This section introduces the concepts of signatures, terms and many-sorted algebras that will be required for the definition of the graphical form of P-nets. The ideas of this section are not new and similar work may be found in [7,13].

5.2.1 Signatures

A *many-sorted* (or *R-sorted*) signature, Σ, is a pair:

$$\Sigma = (R, \Omega)$$

where

- R is a set of sorts (the **names** of sets, eg *Int* for the integers); and
- Ω is a set of operators (the **names** of functions) together with their *arity* in R which specifies the names of the domain and co-domain of each of the operators.

The arity is a function from Ω to $R^* \times R$, where R^* is the set of finite sequences, including the empty string, ε, over R. Thus every operator in Ω is indexed by a pair (σ, r), $\sigma \in R^*$ and $r \in R$ denoted by $w_{(\sigma,r)}$. $\sigma \in R^*$ is known as the *input* sorts, and r as the output sort of operator w. (The sequence of input sorts will define a cartesian product as the domain of the function corresponding to the operator and the output sort will define its co-domain - but this is jumping ahead to the many-sorted algebra.)

For example, if $R = \{Int, Bool\}$, then $w_{(Int.Int,Bool)}$ would represent a binary predicate symbol such as *equality* (=) or *less than* (<). Using a standard convention, the type of a constant may be declared by letting $\sigma = \varepsilon$. For example an integer constant would be denoted by $cons_{(\varepsilon,Int)}$ or simply $cons_{Int}$.

Types of variables may also be declared in the same way. This leads to the consideration of signatures with variables.

5.2.2 Signatures with Variables

A many-sorted signature with variables is the triple:

$$\Sigma = (R, \Omega, V)$$

where R is a set of sorts, Ω a set of operators with associated arity as before and V is a set of typed variables, known as an *R-sorted set of variables*. It is assumed that R, Ω and V are disjoint. The type of the variable is defined by the arity function: $V \to \{\varepsilon\} \times R$. A variable in V of sort $r \in R$ would be denoted by $v_{(\varepsilon,r)}$ or more simply by v_r. For example, if $Int \in R$, then an integer variable would be $v_{(\varepsilon,Int)}$ or v_{Int}.

V may be partitioned according to sorts, where V_r denotes the set of variables of type (sort) r (ie $v_a \in V_r$ *iff* $a = r$).

Including the variables in the signature is a convenient way of ensuring that they are appropriately typed.

5.2.3 Natural and Boolean Signatures

The term *Boolean Signature* is used to mean a many-sorted signature where one of the sorts is Boolean. Similarly, the term *Natural Signature* is used when one of the sorts corresponds to the Naturals ($\mathcal{N}$).

5.2.4 Terms of a Signature with Variables

Terms of type $r \in R$ may be built from a signature $\Sigma = (R, \Omega, V)$ in the normal way. We denote a term, e, of type r by $e : r$ and generate them inductively as follows. For $r, r_1, \ldots, r_n \in R$ $(n > 0)$

1. A variable $v_r \in V$ is a term of type r;
2. A constant $w_{(\varepsilon,r)} \in \Omega$ is a term of type r; and
3. If $e_1 : r_1 \ldots e_n : r_n$ are terms, then $w_{(r_1 \ldots r_n, r)}(e_1, \ldots, e_n) \in \Omega$ is a term of type r.

Thus if Int is a sort, integer constants and variables, and operators of output sort Int are terms of type Int.

We denote the set of all terms of a signature with variables by $TERM(\Omega \cup V)$, the set of all closed terms (those not containing variables) by $TERM(\Omega)$, and the set of terms of sort $r \in R$, by $TERM(X)_r$, where $X = \Omega$ or $\Omega \cup V$.

5.2.5 Tuples of Terms

We can now denote tuples of terms of type $\sigma \in R^* \setminus \{\varepsilon\}$ by

$$< e_1, \ldots, e_n >: r_1 \ldots r_n \textit{ iff } e_1 : r_1, \ldots, e_n : r_n.$$

We denote the set of all tuples of terms of a signature with variables by $TU(\Omega \cup V)$, the set of all tuples of closed terms by $TU(\Omega)$, and the set of tuples of type σ, by $TU(X)_\sigma$, where $X = \Omega$ or $\Omega \cup V$.

5.2.6 Multisets of Tuples

Multisets or *bags* of tuples can also be built inductively from the signature if we assume that we have a Natural signature. We define multisets of tuples this way to allow the multiplicities to be terms of sort Nat, rather than just the Naturals themselves.

Let $BTU(\Omega \cup V)$ be the set of multisets, then it is derived inductively as follows.

- $TU(\Omega \cup V) \subset BTU(\Omega \cup V)$;
- if $b1, b2 \in BTU(\Omega \cup V)$, then $(b1 + b2) \in BTU(\Omega \cup V)$; and
- if $i \in TERM(\Omega \cup V)_{Nat}$ and $b \in BTU(\Omega \cup V)$, then $ib \in BTU(\Omega \cup V)$ where juxtaposition represents scalar multiplication.

This may be extended to the set of bags with infinite multiplicities, $B_\infty TU(\Omega \cup V)$, as follows

- $BTU(\Omega \cup V) \subset B_\infty TU(\Omega \cup V)$; and
- if $b \in BTU(\Omega \cup V)$, then $\infty b \in B_\infty TU(\Omega \cup V)$.

5.2.7 Many-sorted Algebras

A many-sorted algebra, (or Σ-Algebra), $\mathcal{A}$, provides an interpretation (meaning) for the signature Σ. For every sort, $r \in R$, there is a corresponding set, $\mathcal{A}_r$, known as a *carrier* and for every operator $w_{(r_1 \ldots r_n, r)} \in \Omega$, there is a corresponding function

$$w_{\mathcal{A}} : \mathcal{A}_{r_1} \times \ldots \times \mathcal{A}_{r_n} \rightarrow \mathcal{A}_r.$$

For example, if $\Sigma = (\{Int, Bool\}, \{<_{(Int.Int,Bool)}\})$ then a corresponding many-sorted algebra would be

$$\mathcal{A} = (Z, Boolean; lessthan)$$

where Z is the set of integers: $\{\ldots, -1, 0, 1, \ldots\}$
$Boolean = \{true, false\}$
and $lessthan : Z \times Z \rightarrow Boolean$ is the usual integer comparison function.

It could also be

$$\mathcal{B} = (N, Boolean; lessthan)$$

where N is the set of non-negative integers: $\{0, 1, \ldots\}$
$Boolean = \{true, false\}$
and $lessthan : N \times N \rightarrow Boolean$.

(The power of the signature is that it allows a class of algebras to be categorised.)

For signatures with variables, the type of a variable is defined by a sort. In the algebra, the variable is typed by the carrier corresponding to the sort.

Some useful notation for product sets is now defined. For $\sigma \in R^*$ and $\sigma = r_1 r_2 \ldots r_n$ then the corresponding product carrier $\mathcal{A}_{r_1} \times \ldots \times \mathcal{A}_{r_n}$ is denoted by $\mathcal{A}_\sigma$.

Assignment and Evaluation

Given an R-sorted algebra, H, with variables in V, an *assignment*[1] for H and V is a set of functions α, comprising an assignment function for each sort $r \in R$,

$$\alpha_r : V_r \rightarrow H_r.$$

This function may be extended to terms by considering the family of functions

$$\alpha_r : TERM(\Omega \cup V)_r \rightarrow H_r$$

for each sort $r \in R$. The values are determined inductively as follows. For $\sigma \in R^* \setminus \varepsilon$, $\sigma = r_1 r_2 \ldots r_n$, with $r, r1, \ldots, r_n \in R$ and $e, e_1, \ldots, e_n \in TERM(\Omega \cup V)$,

[1]The terms *binding* and *valuation* are also used in this context.

- If e is a variable, then $\alpha(e)$ is given by the assignment function.
- If e is a constant, w_r, then $\alpha(w_r) = w_H \in H_r$.
- If e is an operator, $w_{(\sigma,r)}$, then $\alpha(w_{(\sigma,r)}(e_1,\ldots,e_n)) = w_H(\alpha(e_1),\ldots,\alpha(e_n)) \in H_r$, where $e_1 : r_1 \ldots e_n : r_n$.

Tuples may now be evaluated, for a particular assignment, α. Let $\tau \in TU(\Omega \cup V)$ and $\sigma = r_1 r_2 \ldots r_n$, with other symbols as defined previously, then the value of $\tau =< e_1,\ldots,e_n >\in TU(\Omega \cup V)_\sigma$ in H for α is given by

$$Val_H(\tau) =< \alpha(e_1),\ldots,\alpha(e_n) >\in H_\sigma.$$

Knowing the values of tuples and terms we can determine the value of multisets of tuples by expanding the multiset into a sum of scaled tuples and evaluating each scalar and tuple for a particular assignment to variables. This is defined inductively as follows for $b, b1, b2 \in BTU(\Omega \cup V)$ and $i \in TERM(\Omega \cup V)_{Nat}$

- $Val_H(b1 + b2) = Val_H(b1) + Val_H(b2)$
- $Val_H(ib) = Val_H(i)Val_H(b)$

5.3 Graphical Form of P-nets

In this section a definition of the basic graphical form of P-nets is given by defining a **P-Graph**. It consists of an inhibitor net where the arcs are annotated by multisets of tuples of terms. The multiplicities of the multisets are non-negative integer terms. Transitions are annotated by Boolean terms. The terms are built from a Natural-Boolean signature which has an associated many-sorted algebra. The colour function restricted to places is included. It associates with a place non-empty sets comprising unions of products of carriers of the many-sorted algebra. The capacity and initial marking are multisets over the place colour set as usual.

5.3.1 Definition

A **P-Graph** is a structure

$$\mathbf{PG} = (IN, \mathcal{D}, \Sigma, C, AN, K, M_0)$$

where

- $IN = (S, T; F, IF)$ is an inhibitor net, with
  - S a finite set of places;
  - T a finite set of transitions disjoint from S;
  - $F \subseteq (S \times T) \cup (T \times S)$ a set of arcs; and
  - $IF \subseteq S \times T$ a set of inhibitor arcs.

- $\mathcal{D}$ is a finite set of non-empty colour sets;
- $\Sigma = (R, \Omega, V)$ is a Natural-Boolean signature with variables. It has a corresponding Σ-Algebra, H.
- $C : S \rightarrow \mathcal{D}$ is the colour function restricted to places. For all $s \in S$, the colour set is chosen so that it is in general a union of product sets of carriers of H: $C(s) = \bigcup\{H_{\sigma_i} \mid \text{for } i = 1, \ldots, n, \sigma_i \in R^* \setminus \varepsilon\}$.
- $AN = (A, IA, TC)$ is a triple of net annotations.
  - $A : F \rightarrow BTU(\Omega \cup V)$ such that for $s \in S$, $(x, y) \in F$, and $x = s$ or $y = s$, then $Val_H(A(x, y)) \in \mu C(s)$. It is a function that annotates arcs with a multiset of tuples which when evaluated is a multiset over $C(s)$.
  - $IA : IF \rightarrow B_\infty TU(\Omega \cup V)$ such that for every $(s, t) \in IF$, $Val_H(IA(s, t)) \in \mu_\infty C(s)$. It is a function that annotates inhibitor arcs with a multiset of tuples which on evaluation must be in $\mu_\infty C(s)$.
  - $TC : T \rightarrow TERM(\Omega \cup V)_{Bool}$ where for all $t \in T, TC(t) \in TERM(\Omega \cup V(t))_{Bool}$ and $V(t)$ is the set of variables occurring in the arc inscriptions associated with t. TC is a function which annotates transitions with Boolean expressions.
- $K : S \rightarrow \bigcup_{s \in S} \mu^+_\infty C(s)$ where $K(s) \in \mu^+_\infty C(s)$ is the capacity function.
- $M_0 : S \rightarrow \bigcup_{s \in S} \mu C(s)$ such that $\forall s \in S$, $M_0(s) \subseteq K(s)$, is the initial marking.

5.3.2 Discussion

Arc Annotations

When generating multisets of tuples for the arc inscriptions, we allow the multiplicities to be natural number terms, so that the value can depend on the values of variables and operators of other types. In particular this includes the *generalised Kronecker delta* extension to PrT nets [8].

Strong Typing vs Weak Typing

The inclusion of the colour function, C, and associated colour sets, $\mathcal{D}$, may be considered unnecessary. This is because the co-domain of the capacity function and initial marking function could be represented as the set of multisets of tuples in $TU(\Omega)$ evaluated in the Σ-Algebra, H. The colour set of a place would be determined by the types of the tuples in the annotations of the surrounding arcs (evaluated in H) and the capacity and initial marking functions.

The inclusion of the colour function has a number of advantages. Firstly it encourages good design, as the typing of places needs to be considered early in the specification of a system. Secondly, it ensures that the initial marking, capacity function and arc annotations are all consistently typed. This can be used to great advantage for type checking specifications with automated tools. Finally, it allows a straightforward interpretation in terms of a P-net.

We shall use the term *strongly-typed* for P-Graphs in which the colour function is included and *weakly-typed* when it is not included.

Alternative Graphical Forms

Another graphical form would be to just consider an annotated net (rather than an inhibitor net). The definition would be as before, except that the inhibitor net IN would be replaced by a net N and annotations of the input and output arcs would be separated. The ouput arcs would be annotated as before, but the input arcs would carry a pair as an inscription. The first element of the pair would refer to the pre map and the second to the inhibitor map. This may prove to be a more convenient graphical representation as less arcs are involved and it would tend to de-emphasize the rôle of the inhibitor. This is desirable when the inhibitor is acting as a way of increasing modelling convenience rather than modelling power, for example when purging places with finite capacities.

A slightly less syntactic approach would be to replace the signature with the many-sorted algebra, a set of variables, and a typing function associating a variable with a particular carrier of the algebra. This would be closer to the approach in [16] for Algebraic Nets.

There are a number of alternative graphical forms and the choice of the most suitable form will depend on further experience in particular application domains. Present experience indicates that the above definition is at least a useful one.

5.4 Interpretation of the P-Graph as a P-net

The P-Graph may be given an interpretation as a P-net in the following way.

1. Places: S is the set of places in the P-net.
2. Transitions: T is the set of transitions in the P-net.
3. Colour Sets: $\mathcal{D}$ is the subset of $\mathcal{C}$ to be associated with places. The colour set for a transition is determined by the types of the variables occurring in the surrounding arc annotations restricted by its transition condition.

   Let there be n_t free variables associated with the arcs surrounding a transition $t \in T$. Let these have names $v_{r_1}(t), \ldots, v_{r_{n_t}}(t) \in V$. In the Σ-Algebra, H, for all $i \in \{1, 2, \ldots, n_t\}$, let the carrier corresponding to r_i, H_{r_i}, be denoted by G_i with typed variables $v_i(t) : G_i$. Following [10], let $g_i \in G_i$, then

   $$C(t) = \{(g_1, \ldots, g_{n_t}) \mid (\lambda(v_1(t), \ldots, v_{n_t}(t)).TC(t))(g_1, \ldots, g_{n_t})\}$$

   (The λ-expression provides a means for formally substituting values for the variables in the Transition Condition. Tuples which satisfy $TC(t)$ are included in $C(t)$.)

   The set of colour sets for transitions is therefore $\mathcal{O} = \{C(t) \mid t \in T\}$. Thus $\mathcal{C} = \mathcal{D} \cup \mathcal{O}$.
4. The Colour Function: The colour function restricted to places is defined in the P-Graph and $C(t)$ is given above.

5. Pre and Post Maps.

   The pre and post maps are given, for all $(s,t),(t,s) \in F$, by the following mappings from $C(t)$ into $\mu C(s)$

   $$Pre(s,t) : \lambda(v_1(t), \ldots, v_{n_t}(t)).A(s,t)$$

   $$Post(s,t) : \lambda(v_1(t), \ldots, v_{n_t}(t)).A(t,s)$$

   For $(s,t) \notin F$ and $\forall m \in C(t)$, $Pre(s,t;m) = \emptyset$ and for $(t,s) \notin F$ and $\forall m \in C(t)$, $Post(s,t;m) = \emptyset$.

6. Inhibitor Map

   The inhibitor map is a function from $C(t)$ into $\mu_\infty C(s)$ where for all $(s,t) \in IF$

   $$I(s,t) : \lambda(v_1(t), \ldots, v_{n_t}(t)).IA(s,t)$$

   and for $(s,t) \notin IF, \forall g \in C(s), m \in C(t), mult(g, I(s,t;m)) = \infty$.

7. Capacity Function.

   $K(s)$ is as defined in the P-Graph.

8. Initial Marking.

   $M_0(s)$ is as defined in the P-Graph.

5.5 Simple Examples

5.5.1 Consume any token

A simple P-Graph and its corresponding P-Net are shown in figure 5.1. It illustrates the use of PrT net notation. For each value of $x \in A$, there is an occurrence mode of $t1$. Consider the (multi)set $T_\mu = \{(t1, x) \mid \forall x \in A\}$. Then

$$Pre'(T_\mu) = \{(p1, x) \mid \forall x \in A\} = M_0$$

Thus all modes of the transition are enabled and any subset could occur simultaneously.

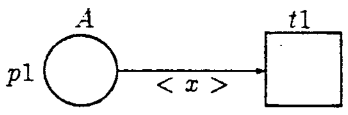
This P-Net represents a set of $|A|$ independent underlying input place/transition pairs, which are concurrently enabled.

This example illustrates a number of conventions that are adopted in the graphical form.

- Inhibitors: It is quite often the case that inhibitor arcs are not present so that $IF = \emptyset$ and hence IA is also empty. The net is a CP-net and also a (many-sorted) PrT net.
- Omission of a capacity annotation or declaration indicates infinite capacity.
- Quite often it is not necessary to state the signature explicitly and we can operate at the level of the algebra. Thus we can just declare the sets and operators. In this case there are no operators.

P-Graph

A: Non-empty set
$M_0(p1) = A$



Linear P-Net

$S = \{p1\}$
$T = \{t1\}$
$C = \{A\}$
$C(p1) = C(t1) = A$
$\forall a \in A, Pre(t1, a) = \{(p1, a)\}$
$\forall a \in A, Post(t1, a) = \emptyset$
the capacity of $p1$ is infinite
$M_0 = \{(p1, a) \mid a \in A\}$

Figure 5.1: Folding Input Place/Transition Pairs

- Implicit typing of variables. When the colour set of a place is a simple product of carriers (or a union of products of different degree), then the type of a variable in an arc annotation is determined from its position in the tuple, the degree of the tuple and the colour set definition. (If the variable occurs in the argument of a function, then it is typed by the domain of the function.) In this example, $x : A$.

  If the variable is used in a number of arc insciptions, then it is possible for mistakes to be made with implicit typing, so that the typing of a specific variable is inconsistent. Considerable care is required with implicit typing and ambiguity will be avoided if all variables are declared in the Declaration.

- Default Transition Condition. If for $t \in T, TC(t) = true$, t is left blank rather than annotating it with the constant $true$. This is the convention adopted for $t1$ in this example.

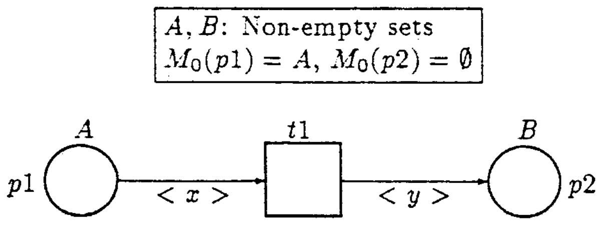
Remark: Choosing $C(t) = A$ is demanded by the above transformation (section 5.4) but this is not necessary. We could have chosen $C(t) = B$ with $|B| = |A|$ and defined Pre as a bijection

$$Pre : \{(t1, b) \mid b \in B\} \rightarrow \{(p1, a) \mid a \in A\}$$

Thus there is an isomorphism. We chose $C(t) = A$ as it provides the simplest way of defining the rule for Pre. Choosing $C(t) = B$ would be equivalent to renaming the transitions in the underlying PT-net.

5.5.2 Consume any token and create any token

P-Graph



Linear P-Net

$S = \{p1, p2\}$
$T = \{t1\}$
$C = \{A, B\}$
$C(p1) = A$
$C(p2) = B$
$C(t1) = A \times B$
$\forall a \in A, \forall b \in B, Pre(t1, a, b) = \{(p1, a)\}$
$\forall a \in A, \forall b \in B, Post(t1, a, b) = \{(p2, b)\}$
the capacities of $p1$ and $p2$ are infinite
$M_0 = \{(p1, a) \mid \forall a \in A\}$

Figure 5.2: More Complex Folding

The P-Net of figure 5.2 shows an example of more complex folding, where each underlying place $\{(p1, a) \mid a \in A\}$ is an input to $|B|$ transitions each of which has a different place chosen from $\{(p2, b) \mid b \in B\}$ as an output place. The variables are implicitly typed: $x : A$ and $y : B$.

5.5.3 Information Flow

By replacing y by x in the above example, we can see how information can flow around a net. To ensure that the resulting net is a P-Graph we must have that $A \subseteq B$ and $x : A$. In this case, including the type of x in the declaration is mandatory as implicit typing is ambiguous (is $x : A$ or $x : B$?). The corresponding P-Net is shown in figure 5.3.

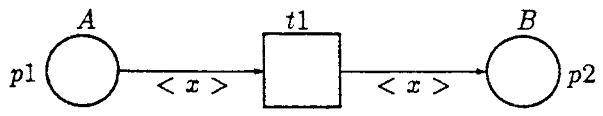
The underlying PT net is a set of $|A|$ identical input place, transition, output place subnets. There are also $|B \setminus A|$ isolated places. (In an application, there would not be any isolated places and the above would be a subnet. Another part of the total net would ensure that no underlying places were isolated.)

5.5.4 Transition Condition

Consider the example of section 5.5.2 with the added constraint that $x < y$ is attached as a condition to transition $t1$. The P-Graph is given in figure 5.4. The comparison operator, $<$, must be defined in the Declaration. Infix notation is used when it is customary. The

P-Graph

A, B: Non-empty sets
$A \subseteq B$; $x : A$
$M_0(p1) = A$, $M_0(p2) = \emptyset$



Linear P-Net

$S = \{p1, p2\}$
$T = \{t1\}$
$C = \{A, B\}$
$C(p1) = A$
$C(p2) = B$
$C(t1) = A$
$\forall a \in A, Pre(t1, a) = \{(p1, a)\}$
$\forall a \in A, Post(t1, a) = \{(p2, a)\}$
the capacities of $p1$ and $p2$ are infinite
$M_0 = \{(p1, a) \mid \forall a \in A\}$

Figure 5.3: Information Flow

A, B: Non-empty sets
$<: A \times B \rightarrow Boolean$
$M_0(p1) = A$, $M_0(p2) = \emptyset$

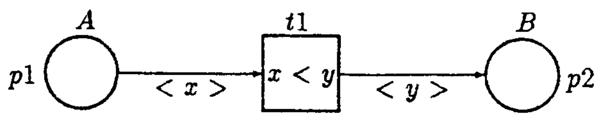


Figure 5.4: P-Graph with Transition Condition

corresponding linear form is the same as that of figure 5.5.2 except that the occurrence modes are limited by the condition $x < y$ so that for $a \in A, b \in B$,

$$C(t1) = \{(a,b) \mid a < b\}.$$

5.6 P-Graph Subclasses

Subclasses of the P-Graph can be formed by restricting its structure. In particular we can derive many-sorted versions of recent formulations of two High-level nets: Algebraic Nets [16] and Predicate/Transition Nets [8].

5.6.1 Many-sorted Algebraic Nets

Algebraic nets were proposed as a reformulation of PrT nets with an improved invariants calculus in [16], where a *partial* algebra over a single carrier was employed. The many-sorted nature of applications was captured by allowing the carrier to be the union of a number of sets. This then lead to the definition of partial functions and their associated operators to be used in the multiset of terms for arc inscriptions.

A colour function is not included in [16] and the net is therefore *weakly-typed*. We shall consider two many-sorted algebraic nets: one weakly-typed and the other strongly-typed.

Weakly-typed many-sorted algebraic nets

A weakly-typed many-sorted algebraic net, AL_w, is one of the simplest special cases of a P-Graph, where the inhibitor arcs and annotations, the Transition Condition, and the colour and capacity functions are removed (ie $IF = \emptyset$; $(\forall t \in T)TC(t) = \mathit{true}$; all places have infinite capacity; and the colour function is not included). The arc annotations are also restricted to multisets where the multiplicities are constants rather than natural number terms.

$$AL_w = (N, \Sigma, A, M_0)$$

where

- $N = (S, T; F)$ is a net.
- $\Sigma = (R, \Omega, V)$ is a an R-sorted signature with variables. It has a corresponding R-sorted algebra, D.
- $A : F \rightarrow \mu TU(\Omega \cup V)$ is the arc annotation function.
- $M_0 : S \rightarrow \mu\{Val_D(\tau) \mid \tau \in TU(\Omega)\}$ is the initial marking.

I believe that this net captures the spirit of Algebraic nets in terms of a specification language and it has the following advantages:

$D = A \cup B$
$A = \{a_1, \ldots, a_n\}, n \in N^+$
$B = \{b_1, \ldots, b_n\}$
$f : D \rightarrow D$ where
$f(a_i) = b_i$ for $i = 1, \ldots, n$
$f(b_i)$ is undefined for $i = 1, \ldots, n$
$M_0(p1) = B$, $M_0(p2) = \emptyset$

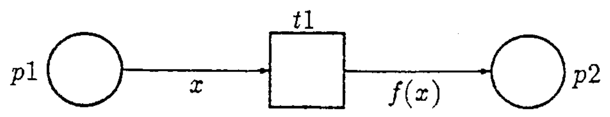


Figure 5.5: Algebraic Net with an undefined follower marking

1. Functions are total.

   Because functions are partial in [16], it is possible to annotate arcs with terms that are not defined in the algebra. This leads to difficulties in interpreting the behaviour of such nets. An example of an Algebraic net illustrating the difficulty is shown in figure 5.5.

   Firstly, consider the situation when $M_0(p1) = A$. Using the terminology of Algebraic nets, a *valuation* (assignment) for x, $\beta(x) = a_1$ for example, will enable $t1$ in mode β. When $t1$ occurs in mode β, a_1 is removed from place $p1$ and $f(a_1) = b_1$ is added to $p2$, ie $M(p1) = A \setminus \{a_1\}$ and $M(p2) = \{b_1\}$. A similar situation occurs for any valuation, $\beta(x) \in A$. Any valuation, $\beta(x) \in B$, will not enable $t1$, due to the initial marking of $p1$.

   Now consider when $M_0(p1) = B$, (a perfectly legal initial marking as $M_0 : S \rightarrow \mu D$, where $D = A \cup B$). A valuation, $\beta(x) \in B$, will now enable $t1$. When $t1$ occurs in mode b_1, the follower marking for $p1$ is clear, $M_0(p1) = B \setminus \{b_1\}$, **but** the follower marking for $p2$ is undefined as the value $f(b_1)$ is not defined.

   This problem does not occur with many-sorted algebraic nets as defined above because functions are total. The intention of the designer of the above algebraic net is unclear. A possible interpretation would be that transition, $t1$, is only enabled when x is bound to an element of A. This interpretation is easily handled with a many-sorted algebraic net (MAN) (see figure 5.6).

   The MAN has essentially the same graphical form. The net part and initial marking are identical, and the annotations very similar (ie the only difference is that angular brackets are used to enclose each inscription). The main difference is that a signature with variables is explicitly included. The sorts $R = \{r1, r2\}$ have corresponding carriers $D_{r1} = A$ and $D_{r2} = B$. The set of operators includes a unary operator $f = w_{(r1,r2)}$ and enough constants of type $A \cup B$ to define the initial marking. The set of variables, V, is a singleton $x = v_{r1}$ and thus $x : A$. The function corresponding to the operator f is a bijection $f_D : A \rightarrow B$, where for $A = \{a_1, \ldots, a_n\}$ and

Sorts: $R = \{r1, r2\}$
Carriers: $D_{r1} = A$ and $D_{r2} = B$
Operators: All constants from A and B and unary operator $f = w_{(r1,r2)}$
Variable: $x = v_{r1}$ thus $x : A$
$A = \{a_1, \ldots, a_n\}, n \in N^+$
$B = \{b_1, \ldots, b_n\}$
$f_D : A \rightarrow B$ where
$f_D(a_i) = b_i$ for $i = 1, \ldots, n$
$M_0(p1) = A \cup B$, $M_0(p2) = \emptyset$

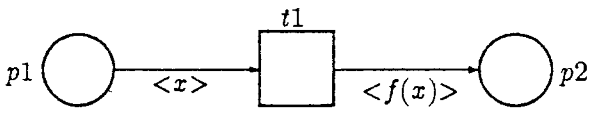


Figure 5.6: Weakly-typed MAN interpretation of above Algebraic Net

$B = \{b_1, \ldots, b_n\}$, $f_D(a_i) = b_i$ for $i = 1, \ldots, n$. To make the example more interesting we have set the initial marking to $M_0(p1) = A \cup B$ and $M_0(p2) = \emptyset$.

Transition, $t1$, is enabled in all modes, $m \in A$, and once all the $a's$ in $p1$ have been transformed into $b's$ in $p2$, $t1$ is dead. There is no possibility of binding x to an element of B, as it is of type $x : A$ as defined in the signature. Thus there are no difficulties of interpretation.

2. Sets can be simple.

   The sets of the many-sorted algebra are simple (as opposed to complex unions of other component sets) and correspond to the sets of the physical world that is being modelled. This contrasts with Algebraic nets where there is only one carrier which needs to contain the union of all the simple sets. This is more than an aesthetic problem when developing automated tools, as valuations for each variable will be over the rather large set D, instead of a much smaller domain corresponding to a carrier of the many-sorted algebra.

3. Tuples.

   Tuples are built indirectly in Algebraic nets from operators and product sets which must be contained in D. It is a much more straightforward task to build tuples in the many-sorted algebraic net.

The implications for analysis, however, are still to be determined.

Strongly-typed many-sorted algebraic nets

A strongly-typed many-sorted algebraic net, AL_s, includes a colour function and is given by

$$AL_s = (N, \mathcal{D}, C, \Sigma, A, M_0)$$

where

- $N = (S, T; F)$ is a net.
- $\mathcal{D}$ is a finite set of non-empty colour sets.
- $C : S \rightarrow \mathcal{D}$ is the colour function restricted to places.
- $\Sigma = (R, \Omega, V)$ is a an R-sorted signature with variables. It has a corresponding R-sorted algebra, D.
- $A : F \rightarrow \mu TU(\Omega \cup V)$ is the arc annotation function. It is restricted so that for all $s \in S, (x, y) \in F$, and $x = s$ or $y = s$, then for all $a \in A(x, y)$, a is of type $\sigma \in R^* \setminus \varepsilon$ where $D_\sigma \subseteq C(s)$.
- $M_0 : S \rightarrow \bigcup_{s \in S} \mu C(s)$ such that $\forall s \in S$, $M_0(s) \in \mu C(s)$ is the initial marking.

The strongly-typed many-sorted algebraic net has the advantage that static type checking can be done to eliminate errors as discussed before. In the above example, it may have been that place $p1$ should never be marked with tokens from B and that the initial marking was just a mistake. In this case it would be appropriate to set $C(p1) = A$ and, depending on the application, $C(p2) = B$. In this case, setting $M_0(p1) = B$, would violate the typing rules and be detected in a static check. This would not be the case in a weakly-typed MAN, where the error would be detected at run time when an attempt to execute the net would reveal that $t1$ was dead.

The P-nets of figures 5.1, 5.2 and 5.3 are examples of strongly-typed MANs, but figure 5.4 is not a strongly-typed MAN as it has a transition condition different from *true*.

5.6.2 Many-sorted PrT Nets

In the latest definition of Predicate/Transition Nets [8], Genrich mentions the use of many-sorted structures and a 'formalism for abstract data types' (many-sorted algebras) but does not pursue these ideas. PrT nets do not include the inhibitor extension nor the colour function and in [8] a capacity function is not defined. This section presents a subclass of the P-Graph which may be considered to be a many-sorted PrT net (MPrT).

Definition

A many-sorted PrT net is the structure

$$\mathbf{MPrT} = (N, \Sigma, AN, M_0)$$

where

- $N = (S, T; F)$ is a net.
- $\Sigma = (R, \Omega, V)$ is a Natural-Boolean signature with variables. It has a corresponding Σ-Algebra, H.
- $AN = (A, TC)$ is a pair of net annotations.

- $A : F \to BTU(\Omega \cup V)$ such that for $s \in S$,$(x, y) \in F$, and $x = s$ or $y = s$, then $A(x, y) \in BTU(\Omega \cup V)_{\sigma_s}$, where $\sigma_s \in R^* \setminus \{\varepsilon\}$. It is a function that annotates arcs leading into or out of place s with a multiset of tuples of type σ_s.
- $TC : T \to TERM(\Omega \cup V)_{Bool}$ where for all $t \in T, TC(t) \in TERM(\Omega \cup V(t))_{Bool}$ and $V(t)$ is the set of variables occurring in the arc inscriptions associated with t. TC is a function which annotates transitions with Boolean expressions.

- $M_0 : S \to \mu\{Val_H(\tau) \mid \tau \in TU(\Omega)\}$ such that for all $s \in S$, $M_0(s) \in \mu\{Val_H(\tau) \mid \tau \in TU(\Omega)_{\sigma_s}\}$, is the initial marking. This is equivalent to $M_0(s) \in \mu H_{\sigma_s}$.

The colour function is easily derived from the type of the initial marking which is the same as the type of the tuples annotating the associated arcs. For all $s \in S$

$$C(s) = H_{\sigma_s}$$

and thus $C(s)$ is in general a product set of carriers. Hence, many-sorted PrT nets are implicitly strongly-typed.

The advantages of many-sorted PrT nets over the PrT nets of [8] are the same as points 1 and 2 of those for many-sorted algebraic nets compared with Algebraic nets (see section 5.6.1). This is because of the single-sorted definition of PrT nets and that applications are naturally many-sorted. To capture this, the domain of the relational structure used to define PrT nets must comprise a union of sets. This leads to the definition of partial functions and loose typing of variables, as variables take on as a default the whole of the domain as their type. PrT nets do allow variables to be typed by using a transition condition, but as this is an option of the specifier, mistakes can easily arise.

Simple Examples

The four examples of the previous section (5.5) are all MPrT-nets, where the colour function ($C(s) = H_{\sigma_s}$) has been used to annotate places explicitly.

Train Example

In [8], Genrich describes the operation of two trains travelling in the same direction on a circular track of seven sections. For safe operation, the trains must never be on the same section or even on adjacent sections. A MPrT-net is given in figure 5.7 where any number of sections greater than 4 is allowed.

The model is a little different from that in [8]. Apart from the minor difference of generalising the number of track sections, the marking of place $p2$ represents which track sections are vacant. In the original model, the same place represented the predicate that sections i and i$\oplus$1 were vacant. As a minor modelling point, the simpler meaning for a place is preferred. The less intuitive predicate also necessitates the definition of two functions, the modulo 7 successor and predecessor functions, whereas only one (modulo n addition) is required in the MPrT-net. There is also no need for the transition condition and extra variables.

The drawback of the PrT net is that the successor functions are partial, whereas the variables all range over IUT. Thus there are legal substitutions for the variables for which the transition condition is undefined. This situation does not arise with the MPrT-net.

Declarations

Set of Trains:T = {a,b}
Set of track sections:I = $\{0, 1, \ldots, n-1 \mid n > 4\}$
n: number of sections
Variables x:T ; i:I
Function ⊖:I×I→I is modulo n addition
Place $p1$: Sections occupied by trains
Place $p2$: Vacant sections
$M_0(p1) = \{<0,a>,<2,b>\}$
$M_0(p2) = \{1, 3, \ldots, n-1\}$

Graph

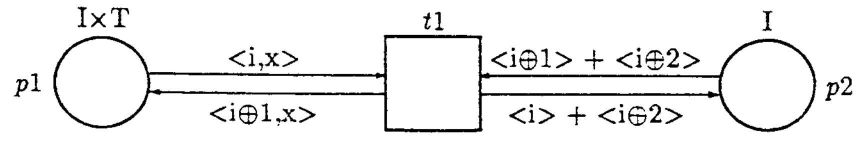


Figure 5.7: MPrT-Net of Safe Train Operation

It can be seen that this net is also a strongly-typed MAN.

Conditionals in arc expressions

In this example we use a variant of the readers/writers problem to illustrate the use of conditionals in arc expressions. It is essentially the same as the resource management scheme example of [8], but the model is considerably simplified by removing unnecessary states and colours. The identities of the agents wishing to access the common resource have been retained, but the access 'tickets' are not distinguished.

A number (N) of agents (processes) wish to access a shared resource (such as a file). Access can be in one of two modes: shared (s), where up to L agents may have access at the same time (eg reading) and exclusive (e), where only one agent may have access (eg writing). No assumptions are made regarding scheduling. An MPrT-net model is given in figure 5.8.

It has been assumed that the initial state is when all the agents are idle or waiting to gain access to the shared resource (with no queueing discipline assumed). Place *Wait* is marked with all agents; *Access* is empty and the *Control* place contains L ordinary tokens. An agent can obtain access in one of two modes: if shared (m=s), then a single token is removed from *Control* (as m=e is false) when *enter* occurs in a single mode; if exclusive (m=e), then all L tokens are removed preventing further access until the resource is released (transition *Leave*). Shared access is limited to a maximum of L agents as transition *enter* is disabled when *Control* is empty.

Following [8] outfix notation has been used for the function *Bool* $\rightarrow \{0, 1\}$ and this will be used as a standard convention. It is assumed that integer addition and subtraction and the equality predicate are primitive and do not need to be defined in the Declaration.

Declarations

Set of Agents:A = $\{a_1, \ldots, a_N\}$
Set of Access Modes:M = {s,e}
Control: C = $\{\bullet\}$
Positive integer constants: N,L
Variables x:A ; m:M
Function [ ]:$Bool \rightarrow \{0,1\}$ where
$[true] = 1$ and $[false] = 0$
M_0(Wait) =A
M_0(Control) =L<•>
M_0(Accessing) = $\emptyset$

Graph

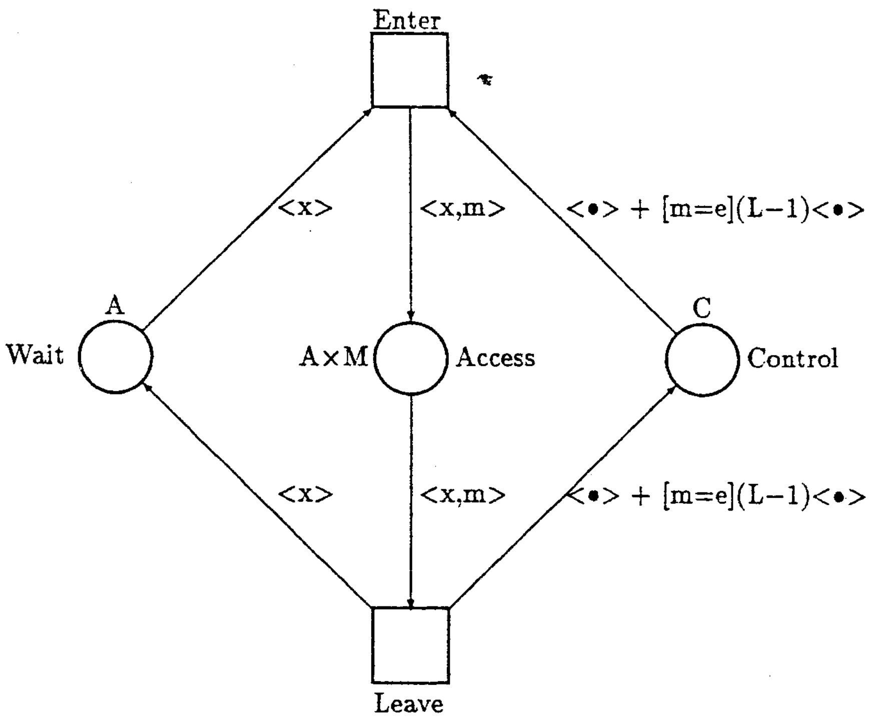


Figure 5.8: MPrT-Net of Resource Management

Remark: The net of figure 5.8 is very like a PrT net. If the places were annotated by predicates rather than colour sets and the domain was formed as the union D = A∪M∪C, with variables x and m of type D, it would be a PrT net. The indices of the predicates annotating *Wait* and *Control* would each be one, and that of *Access*, two. It is important to note that the behaviour of the two nets is not the same. In the PrT net m can be bound to any element of D. Hence an agent could gain access to the resource in a meaningless mode • or a_i for example. The meaning of this is unclear and contrary to the intention of the specification.

5.7 Notation for Capacity

The capacity of a particular place, s, is given by the function

$$K(s) : C(s) \longrightarrow N^+_\infty$$

It is convenient to use a shorthand notation for this function, particularly for annotating places of the P-Graph. Firstly, consider a place with a product colour set:

$$C(s) = G_1 \times \ldots \times G_n.$$

Let $g_i \in G_i$ be constants and $v_i : G_i$ be typed variables for $i \in In = \{1, \ldots, n\}$ and $n \in N^+$.

We shall use the following notation to annotate places in the graphical form, where the capacity of each token is the same.

Capacity Notation for place s	Meaning in terms of $K(s)$
$K(v_1, \ldots, v_n) = m$	$\forall i \in In\ \forall g_i \in G_i\ K(s; g_1, \ldots, g_n) = m$

With the capacity restricted in this way, it is the same as that defined for PrT nets [9]. We shall adopt the abbreviation K for $K(v_1, \ldots, v_n)$ as in [9], when there is no danger of ambiguity.

This notation can be extended to colour sets that are unions of product sets by the conjunction of the notation for the individual products. The meaning is then given by the conjunction of the meanings for the notation of the individual products.

5.8 Extended Capacity Notation

Although the P-Net capacity function and the above notation may be of use in some applications, it turns out that a much richer capacity notation is required that allows a limit to be placed on the cardinality of multisets over elements of partitions of a place's colour set. An example is the *total* capacity of a place (ie the sum of all tokens in the place) which represents a resource bound, eg a buffer capacity. Here we are not placing a direct limit on the multiplicity of each element of the colour set but a limit on the sum of multiplicities of elements and thus the capacity function is inadequate.

As a further illustration, consider the following example encountered while modelling the M-Access Service of the Cambridge Fast Ring [3].

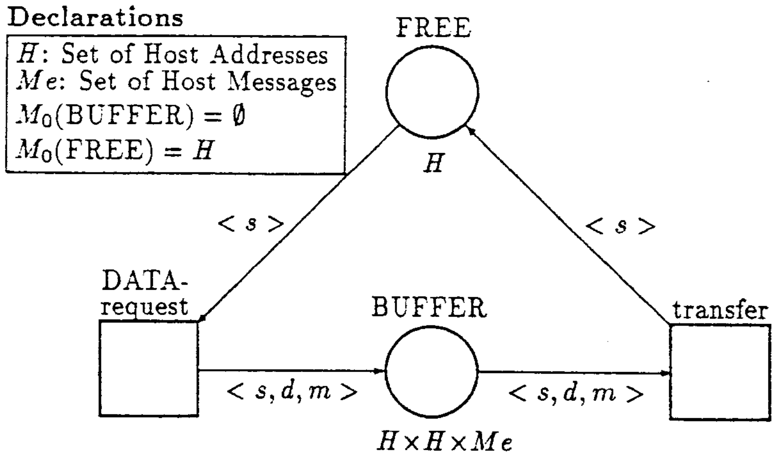


Figure 5.9: LAN Access Buffer

A local area network interconnects a set of hosts, with addresses, H, so that each host can send messages $m : Me$ to each other host. Hosts access the network via a single buffer so that only a single message is stored in the network per host before being delivered. The source ($s : H$) and destination ($d : H$) addresses are appended to the message and then submitted to the network where it is stored in the buffer. When resources are available the message is transferred into the network for routing and delivery.

A P-Graph of the access procedure is shown in figure 5.9. Place **BUFFER** represents the set of access buffers, one for each host. Place **FREE** indicates which buffers are available. If this place contains a token with host a's address, then host a's buffer is free and can be used for the next message host a wishes to submit to the network (transition **DATA-request** occurs). The buffer will not be free again until the network accepts the message (transition **transfer** occurs). Hence place FREE provides the control necessary to ensure a capacity limit of one buffer per host.

In the graphical form of P-nets it would be convenient to replace the capacity control for place BUFFER by an extended capacity inscription. This is shown in figure 5.10, where place BUFFER is inscribed by '$K(s,*,*) = 1$'. We may interpret this to mean that there is one buffer available for each host ie that the sum of tokens over $d : H$ and $m : Me$ in place BUFFER for a particular value of s is at most one ($\sum_{h \in H} \sum_{g \in Me} < s, h, g > \leq 1$). The $*$'s indicate sums over the variables they replace. This may be viewed as an extended capacity condition on the marking of the place concerned. For all hosts ($s \in H$), $\sum_{h \in H} \sum_{g \in Me} M(BUFFER; s, h, g) \leq K(s,*,*)$, for all markings of BUFFER.

More generally, a place, s, with $C(s) = G_1 \times \ldots \times G_n$, may be annotated by an inscription $K(a_1, \ldots, a_n) = k$ with $k \in N^+$ where the syntax of $a_i, i \in In$ is given by the production rule a_i ::=$< v_i >$ |$*$ where angular brackets denote non-terminals. (At present we shall leave open the choice of syntax for variables, but it would most likely be a finite string of alphanumeric characters.) As above $v_i : G_i$.

Declarations

H: Set of Host Addresses
Me: Set of Host Messages
$M_0(\text{BUFFER}) = \emptyset$

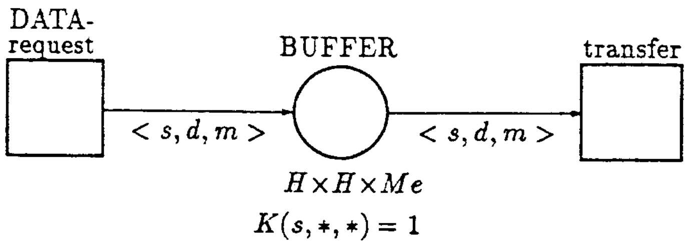


Figure 5.10: LAN Access Buffer illustrating extended capacity notation

We shall now give the meaning of this notation in terms of a P-Graph without it.

5.8.1 Interpretation of Extended Capacity Notation

When there are no stars present in the argument of $K(a_1, \ldots, a_n)$, then the meaning has already been given in the previous section. We now consider two cases:

- A. when there is at least one star but less than n stars
- B. when all arguments are stars.

For case A, for each place, s, inscribed by $K(a_1, \ldots, a_n) = k$, we remove the inscription and replace it with a *projected* complementary place, $\bar{s}$, and associated arcs in the following manner.

1. From the argument of K create a tuple consisting of only the variables by deleting the stars. This will be of the form $< v_i, \ldots, v_j >$ with $i \leq j \leq n$.
2. Create a place, $\bar{s}$, with colour set $C(\bar{s}) = G_i \times \ldots \times G_j$ derived from the types of the above tuple, where G_i is the type of the variable v_i and so forth.
3. Create an arc $(\bar{s}, t)$ for each arc (t, s), $t \in T$ and an arc $(t', \bar{s})$ for each arc (s, t'), $t' \in T$.
4. Annotate each arc by the tuple $< v_i, \ldots, v_j >$.
5. The initial Marking, $M_0(\bar{s})$ is related to $M_0(s)$ and the value of k in the following way. For every $g_i \in G_i \ldots g_j \in G_j$

$$mult((g_i, \ldots, g_j), M_0(\bar{s})) + \sum mult((g_1, \ldots, g_n), M_0(s)) = k$$

where the sum is over the domains of the variables that have been replaced by stars in the argunment of K.

For case B, for each place, s, inscribed by $K(*,\ldots,*) = k$, we remove the inscription and replace it with a *completely-projected* complementary place, $\overline{s}$, (a PT place) and associated arcs in the following manner.

1. Create a place, $\overline{s}$, with colour set $C(\overline{s}) = \{\bullet\}$.
2. Create an arc $(\overline{s},t)$ for each arc (t,s), $t \in T$ and an arc $(t',\overline{s})$ for each arc (s,t'), $t' \in T$.
3. Annotate each arc by the singleton $< \bullet >$.
4. The initial Marking, $M_0(\overline{s})$ is related to $M_0(s)$ and the value of k in the following way.
$$mult((\bullet), M_0(\overline{s})) + \sum mult((g_1,\ldots,g_n), M_0(s)) = k$$
where the sum is over the domains of all the variables.

Case B corresponds to a resource limit and the notation K^* will be adopted for it (ie $K^* = K(*,\ldots,*)$) as in Numerical Petri Nets [23].

In this section the colour sets have been restricted to a single product set. No attempt is made to generalise to unions of product sets as the complexity and infrequent usage do not justify it.

Chapter 6

Manipulation of Markings

When modelling applications it is convenient to be able to manipulate markings atomically (ie with the occurrence of a single mode of a transition). The purging of a queue [4] or aborting a broadcast [3] are relevant examples.

The manipulation of the marking of a place can be achieved with P-nets by setting the pre map equal to the inhibitor map and hence equal to the current marking when the transition is enabled. The post map then allows the desired manipulation of the marking as it can be defined in terms of the pre map (ie the current marking).

The idea of removing a place marking (purging) on the single occurrence of a transition has arisen in the context of PT-nets [20], where a class of nets known as self-modifying nets was defined. The work of this section generalises this idea in the context of high-level inhibitor nets, making it more expressive and suitable for complex applications.

The term *reset* will be used to connote the emptying of a place of all its tokens on the single occurrence of a transition. A *complete reset* implies the resetting of *all* places on the single occurrence of a transition.

Firstly we investigate P-nets with the *complete reset* property, the class of P-nets where the pre map and inhibitor map are the same. This class of P-net turns out not to be suitable for our purposes, but it is the simplest case conceptually and hence it is dealt with first. Having dispensed with this curiosity, we consider P-nets with the *reset* property where at least one place may be emptied by the single occurrence of a transition. The term, *purging* a place, is introduced, to describe the resetting of a place irrespective of its marking. A graphical representation of *purging* is given where a *reset arc* is defined as the superposition of the inhibitor and normal arcs. It is inscribed by a variable typed by the set of multisets over the colour set of the input place. The idea is generalised to purging submultisets (subbags) of markings and further to considering partitions and subsets of partitions of the input place's colour set.

6.1 Completely resetting P-nets - a curiosity

Definition: A P-net with the property of *completely resetting* is a P-net with $Pre = I$ and $\forall tr \in TRANS, pre(tr) \neq \emptyset$.

The possibility of having a transition connected to all places by zero-testing inhibitors is excluded as it leads to either the transition always being dead of if enabled by the empty marking then it is infinitely self-concurrently enabled. It appears that neither situation is of practical value and the restriction leads to a simple statement for the following proposition.

Proposition 6.1 *A completely resetting P-net is an interleaving model in that no two transition modes can be concurrently enabled.*

Proof: The proof is by contradiction. Assume that $tr1, tr2 \in TRANS$ are concurrently enabled at a reachable marking $M \in [M_0\rangle$, then the following must hold from the definition of enabling

$$Pre'(tr1 + tr2) \subseteq M \subseteq I'(tr1 + tr2) \tag{6.1}$$

From the definitions of Pre' and I'

$$\begin{aligned} Pre'(tr1 + tr2) &= Pre(tr1) + Pre(tr2) && (6.2) \\ I'(tr1 + tr2) &= I'(tr1) \cap I'(tr2) \\ &= I(tr1) \cap I(tr2) \\ &= Pre(tr1) \cap Pre(tr2) && (6.3) \end{aligned}$$

Inserting equations 6.2 and 6.3 into 6.1

$$Pre(tr1) + Pre(tr2) \subseteq M \subseteq Pre(tr1) \cap Pre(tr2) \tag{6.4}$$

From the definition of multiset intersection (see Appendix A), this inequality can only be satisfied if $Pre(tr1) = \emptyset$ and $Pre(tr2) = \emptyset$ which is forbidden by the above definition. Hence two transitions cannot be concurrently enabled. □

Proposition 6.2 *For $tr \in TRANS$ enabled at a marking M, the transition rule is given by*

$$M' = Post(tr)$$

Proof: From Proposition 6.1, only one transition mode can occur at any instant (interleaving) and the transition rule becomes for all $tr \in TRANS$

$$M' = M - Pre(tr) + Post(tr) \tag{6.5}$$

From the enabling condition and definition of *completely resetting* P-nets

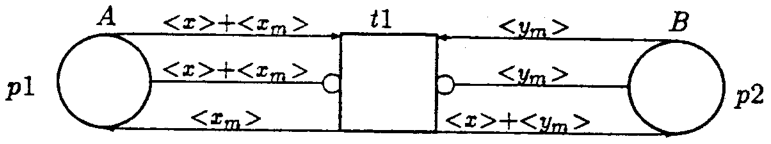
$$\begin{aligned} & Pre(tr) \subseteq M \subseteq I(tr) \\ \Rightarrow\ & Pre(tr) \subseteq M \subseteq Pre(tr) \\ \Rightarrow\ & M = Pre(tr) && (6.6) \end{aligned}$$

and thus from equation 6.5, $(\forall tr \in TRANS) M' = Post(tr)$. □

Thus the follower marking is independent of the initial marking and it is this property that gives rise to the term *reset*.

P-Graph

A, B: Non-empty sets
$A \subseteq B$
$x : A, x_m : \mu A, y_m : \mu B$
$M_0(p1) = A, M_0(p2) = \emptyset$



P-Net

$S = \{p1, p2\}$
$T = \{t1\}$
$C = \{A, B, A \times \mu A \times \mu B\}$
$C(p1) = A$
$C(p2) = B$
$C(t1) = A \times \mu A \times \mu B$
$Pre(p1, t1) = \pi_1 + \pi_2$
$I(p1, t1) = Pre(p1, t1)$
$Post(p1, t1) = \pi_2$
$Pre(p2, t1) = \pi_3$
$I(p2, t1) = Pre(p2, t1)$
$Post(p2, t1) = \pi_1$
where π_1, π_2, π_3 are projection functions:
$\pi_1 : C(t1) \rightarrow A$
$\pi_2 : C(t1) \rightarrow \mu A$
$\pi_3 : C(t1) \rightarrow \mu B$
the capacities of $p1$ and $p2$ are infinite
$M_0 = \{(p1, a) \mid a \in A\}$

Figure 6.1: Reset Net corresponding to Figure 5.3

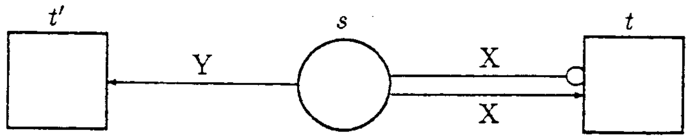


Figure 6.2: Transitions t and t' cannot occur concurrently

Although it is possible to model applications with completely resetting P-nets, it is far too unwieldy and often will not match with the intuition of the designer as every transition affects every place. A simple example is given in figure 6.1. It models the situation in figure 5.3, where an item from a store is transferred to another store. With the net in figure 5.3, as many items as exist in $p1$ may be transferred as all the modes of $t1$ are concurrently enabled. This is not the case in figure 6.1 as only one mode is enabled for any given marking. The net is much more complex and rather convoluted and counter-intuitive. For these reasons completely resetting P-nets will not be investigated further.

What we would like to do is to allow resetting only when it is necessary.

6.2 P-nets with the reset property

Instead of resetting the whole marking of a P-net, we now consider the resetting of the marking of individual places. Consider the P-Graph of figure 6.2 where X and Y are any legal arc insciptions. Transition t has the reset property over place s. We assert that transition t cannot occur concurrently with any other transition mode, (including its own), that has an enabling condition dependent on place s.

Proposition 6.3 *For a P-net with transitions $t, t' \in T$, and place $s \in S$ with $Pre(s,t) = I(s,t)$, a mode of t, $m \in C(t)$ where $Pre(s,t;m) \neq \emptyset$ and a mode of t', $m' \in C(t')$ with $Pre(s,t';m') \neq \emptyset$, can not occur concurrently.*

Proof: The proof is essentially the same as the proof of proposition 6.1. From the enabling condition restricted to place s and the reset property, it follows that for any $m \in C(t)$ and $m' \in C(t')$

$$Pre(s,t;m) + Pre(s,t';m') \subseteq M(s) \subseteq Pre(s,t;m) \cap I(s,t';m')$$

A necessary condition for this inequality to be satisfied is $Pre(s,t';m') = \emptyset$. Hence transitions t and t' cannot occur concurrently when $Pre(s,t';m') \neq \emptyset$. □

An immediate specialisation of this proposition is:

Proposition 6.4 *For a P-net with transition $t \in T$ and place $s \in S$ such that $Pre(s,t) = I(s,t)$ and $\forall m \in C(t), Pre(s,t;m) \neq \emptyset$, the modes of t can never be concurrently enabled.*

Proof: The proof follows from proposition 6.3 by setting $t' = t$. □

The reset condition ensures that the only marking that can satisfy the precondition is equality with the pre map. More precisely

Proposition 6.5 *For a P-net with transition $t \in T$ and place $s \in S$ such that $Pre(s,t) = I(s,t)$ and a mode, $m \in C(t)$, enabled at marking $M(s)$, then $Pre(s,t;m) = M(s)$.*

Proof: The proof follows immediately from the reset property and the enabling condition. For a mode $m \in C(t)$ enabled at $M(s)$,

$$\begin{aligned} & Pre(s,t;m) \subseteq M(s) \subseteq I(s,t;m) \\ \Rightarrow\quad & Pre(s,t;m) \subseteq M(s) \subseteq Pre(s,t;m) \\ \Rightarrow\quad & M(s) = Pre(s,t;m) \end{aligned} \tag{6.7}$$

□

Corollary 6.1 *A consequence of proposition 6.5 is that if transition, t, occurs in mode m, then the follower marking of place s is given by $M'(s) = Post(s,t;m)$.*

Proof: Follows immediately from the transition rule and equation 6.7. □

6.3 Purging

This section describes how to empty a place of its current marking. This will be referred to as *purging*.

If we wish to purge a place s with a single occurrence of a mode of transition t, we need a mode, $m \in C(t)$, for every marking of s, $M(s) \in \mu C(s)$. $C(t)$ will in general be a product. To guarantee a mode for each marking, let $\mu C(s)$ be one of the sets in the product. We also need the pre map, $Pre(s,t)$, to select the current marking. From proposition 6.5 this can be done by using a reset: $I(s,t) = Pre(s,t)$. $Pre(s,t)$ and $I(s,t)$ will then be projection functions that select out a marking in $\mu C(s)$ from $C(t)$.

Let $D_s = \mu C(s)$ and $C(t) = D_s \times D'$ where D' is in general a product set dependent upon the inscriptions of the other arcs connected to t. (If there is no requirement for a product, then $C(t) = D_s$ or equivalently D' contains a single nondescript element so that the product $D_s \times D'$ is isomorphic to D_s. We shall use the latter approach to include this special case in the general presentation.)

A projection function is defined as

$$\pi_s : C(t) \longrightarrow \mu C(s)$$

where $\pi_s(d_s, d') = d_s$ with $d_s \in D_s$ and $d' \in D'$.

Then $I(s,t) = Pre(s,t) = \pi_s$.

Corollary 6.1 ensures that whenever transition t occurs, place s will be emptied of its current marking, so long as $Post(s,t;m) = \emptyset$.

The following propositions summarise the above discussion.

Proposition 6.6 *Given a P-net with $t \in T$, $s \in S$, $C(t) = \mu C(s) \times D'$ and $Pre(s,t) = \pi_s$ then $\exists m \in C(t)$, such that $Pre(s,t;m) = M(s)$ for any $M(s) \in \mu C(s)$.*

Proof: Firstly $M(s) \in \mu C(s)$. From the definitions of $C(t)$ and π_s, noting that π_s is surjective, $Pre(s,t)$ ranges over $\mu C(s)$. Hence we can always choose an appropriate argument, m, of $Pre(s,t)$ such that $Pre(s,t;m) = M(s)$. □

The conditions under which this proposition holds ensure that a mode of transition t can always be chosen so that the pre map $Pre(s,t;m)$ is equal to the current marking.

Proposition 6.7 *Given a P-net as in the previous proposition with* $Pre(s,t) = I(s,t)$ *and* $\forall m \in C(t), Post(s,t;m) = \emptyset$, *then when* t *occurs, place* s *is purged, ie* $M'(s) = \emptyset$.

Proof: Follows immediately from corollary 6.1. □

6.3.1 Graphical Representation

In the P-Graph, $\mu C(s)$ is a carrier of the many-sorted algebra, H, (ie $H_r = \mu C(s)$) and a variable that ranges over multisets over $C(s)$ is included in the signature, for example $Y : \mu C(s)$. The arc (s,t) and inhibitor arc are annotated by a tuple consisting of this variable: $A(s,t) = IA(s,t) = Y$. Any variable name could be chosen and the declaration part would provide its type. However, a capital letter can be used to alert readers of the graphical part that this variable is of higher order type.

To save space in the P-Graph, the normal and inhibitor arcs are superimposed and only one inscription is used as $A(s,t) = IA(s,t)$. This may be referred to as a *reset arc*.

6.3.2 Example: Purging a place

An example of a simple net where all tokens are removed from a place, irrespective of its marking, is shown in figure 6.3.

Transition $t1$ is always enabled in one of its modes but never concurrently with itself, except when $M(p1) = \emptyset$. When $t1$ occurs, $p1$ is emptied ($M'(p1) = \emptyset$). When $M(p1) = \emptyset$, $t1$ is enabled by a mode in which $I(tr) = Pre(tr) = \emptyset$. An unusual phenomenon occurs where the transition is infinitely self-concurrently enabled and its occurrence has no effect on the current state. This possibility may be excluded by restricting the set of multisets over A to be non-empty.

The situation is the same as in figure 6.3 with the following modifications. In the P-Graph and in the P-net we replace μA by $\mu A \setminus \{\emptyset\}$ so that Pre and I become injections. Hence $Pre(tr) \neq \emptyset, \forall tr \in TRANS$. Thus $t1$ is not enabled when the marking is null, but is enabled for all other markings.

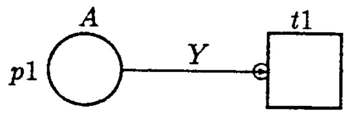
6.4 Transferring a Marking

There are situations when we would like to be able to transfer the contents of one place to another place [3]. This may be achieved by setting the post map for the receiving place to the pre map for the place that is being purged.

Let s be the place that will be purged and $s1$ the one that will receive its contents. We require that $C(s) \subseteq C(s1)$.

P-Graph

A: Non-empty set
$Y : \mu A$
$M_0(p1) = A$



P-Net

$S = \{p1\}$
$T = \{t1\}$
$C = \{A, \mu A\}$
$C(p1) = A$
$PLACE = \{(p1, a) \mid a \in A\}$
$C(t1) = \mu A$
$TRANS = \{(t1, m) \mid m \in \mu A\}$
Pre and I are bijections: $TRANS \rightarrow \mu PLACE$
where for all $m \in \mu A, I(t1, m) = Pre(t1, m) = (p1, m)$
For all $tr \in TRANS$, $Post(tr) = \emptyset$
$\forall a \in A, K(p1, a) = \infty$
$M_0 = \{(p1, a) \mid a \in A\}$

Figure 6.3: Purging a place

As above we have $I(s,t) = Pre(s,t)$ and we now set $Post(s1,t) = Pre(s,t)$.

In the graphical form, we shall have a reset arc from s to t annotated by Y as will be the arc from t to $s1$. ($A(s,t) = IA(s,t) = A(t,s1) = Y$.)

6.5 Purging subbags of Markings

Consider a transition, t, with an input place, s. Let G_s be a subset of $C(s)$, form the set of multisets over G_s, μG_s, and include it as part of the product set comprising $C(t)$ in a similar way to section 6.3. Let $D_s = \mu G_s$ and $C(t) = D_s \times D'$ as before and recalling the projection function, the pre and inhibitor maps are

$$Pre(s,t) = \pi_s$$

$$I(s,t;m) = \pi_s(m) + INF(s,t)$$

where $m \in C(t)$ and $INF(s,t)$ is an infinite multiset over $C(s) \setminus G_s$, where

$$mult(g, INF(s,t)) = \begin{cases} 0 & \text{if } g \in G_s \\ \infty & \text{otherwise} \end{cases}$$

Thus the pre and inhibitor maps are completely determined by the subset G_s.

We may now generalise the propositions in section 6.3.

Proposition 6.8 *Given a P-net with $t \in T$, $s \in S$, $C(t) = \mu G_s \times D'$ and $Pre(s,t) = \pi_s$ then $\exists m \in C(t)$, such that for any $M(s) \in \mu C(s)$, $g \in C(s)$*

$$mult(g, Pre(s,t;m)) = \begin{cases} mult(g, M(s)) & \textit{if } g \in G_s \\ 0 & \textit{otherwise} \end{cases}$$

Proof: Since the range of π_s and hence $Pre(s,t)$ is μG_s then for all $m \in C(t)$, and $g \notin G_s$, $mult(g, Pre(s,t;m)) = 0$. Now consider the subbag of $M(s)$, SUB, where the multiplicities of all elements not in G_s are set to zero and the others are as they were in $M(s)$. Then $SUB \in \mu G_s$. Hence we may set $m = (SUB, d')$ and hence $Pre(s,t;m) = SUB$. The proposition now follows immediately. □

Proposition 6.9 *Given a P-net as in the previous proposition with for all $m \in C(t)$, $I(s,t;m) = \pi_s(m) + INF(s,t)$ (as defined above) and $\forall m \in C(t), Post(s,t;m) = \emptyset$, then when t occurs, place s is purged of all tokens in G_s, ie*

$$mult(g, M'(s)) = \begin{cases} 0 & \textit{if } g \in G_s \\ mult(g, M(s)) & \textit{otherwise} \end{cases}$$

Proof: To prove this proposition we need the following lemma.

Lemma 6.1 *Given a P-net with $t \in T$, $s \in S$, $C(t) = \mu G_s \times D'$, $Pre(s,t) = \pi_s$ and for all $m \in C(t)$, $I(s,t;m) = \pi_s(m) + INF(s,t)$ such that a mode $m \in C(t)$ is enabled at M, then for $g \in C(s)$*

$$mult(g, Pre(s,t;m)) = \begin{cases} mult(g, M(s)) & \textit{if } g \in G_s \\ 0 & \textit{otherwise} \end{cases}$$

Proof: From above, $\forall g \notin G_s$, $mult(g, Pre(s,t;m)) = 0$. From the enabling condition on place s it follows that for $g \in G_s$

$$\begin{aligned}
& mult(g, Pre(s,t;m)) \le mult(g, M(s)) \le mult(g, I(s,t;m)) \\
\Rightarrow \quad & mult(g, Pre(s,t;m)) \le mult(g, M(s)) \le mult(g, Pre(s,t;m)) \\
\Rightarrow \quad & mult(g, M(s)) = mult(g, Pre(s,t;m)) \qquad (6.8)
\end{aligned}$$

which proves the lemma. □

From the firing rule, this lemma, and that for all $m \in C(t)$, $Post(s,t;m) = \emptyset$ we have for all $m \in C(t)$

$$\begin{aligned}
& M'(s) = M(s) - Pre(s,t;m) + Post(s,t;m) \\
\Rightarrow \; & M'(s) = M(s) - Pre(s,t;m) \\
\Rightarrow \; & mult(g, M'(s)) = \begin{cases} 0 & \text{if } g \in G_s \\ mult(g, M(s)) & \text{otherwise} \end{cases}
\end{aligned}$$

which proves the proposition. □

The condition that for $m \in C(t)$ and $g \notin G_s$, $mult(g, I(s,t;m)) = \infty$ implies that there is no enabling requirement on tokens that are not in the subbag G_s. Thus the enabling of t only depends on places other than s.

6.5.1 Graphical Representation

Let μG_s be a carrier of the many-sorted algebra and include a variable $Z : \mu G_s$ in the signature. If we wish to purge place, s, of a subbag of $M(s)$ by an occurrence of transition, t, then $A(s,t) = IA(s,t) = Z$. Note that we are using a convention that infinite multiplicities for inhibitor arcs are not represented explicitly. (If all multiplicities for the inhibitor map are infinite, then no inhibitor arc is drawn - it is just the natural extension of this).

6.5.2 Notation for Subsets of Product Sets

Let $G = G_1 \times \ldots \times G_n$, and represent an element of G by the tuple $(g_1, \ldots, g_n)$, where $g_i \in G_i$ are constants for $i \in In$ where $In = \{1, \ldots, n\}$.

We shall also use the notation $(g_1, \ldots, g_n)$ to represent the singleton set $\{(g_1, \ldots, g_n)\}$. The notation $*$ is introduced in position i to indicate a subset of G where all values of G_i are included. Thus $(g_1, \ldots, g_{i-1}, *, g_{i+1}, \ldots, g_n)$ represents the subset $\{g1\} \times \ldots \times \{g_{i-1}\} \times G_i \times \{g_{i+1}\} \times \ldots \times \{g_n\}$. This may be generalised to allow a '$*$' to replace any constant, g_k, in the tuple where $\{g_k\}$ is then replaced by G_k to obtain the subset. If all the constants are replaced by stars then the subset is the original set G.

In general, given a tuple consisting of constants and stars, let $G_s \subseteq G$, and $I_* \subseteq In$ be the set of positions in which stars occur in the tuple. Then the subset of G corresponding to the tuple notation is given by

$$G_s = \{(g_1, \ldots, g_n) \mid g_i \in G_i, i \in I_*\}.$$

For $C(s) = G_1 \times \ldots \times G_n$, it is convenient to use the above notation to indicate the subset G_s. We use the following annotation on the reset arc (s,t), to represent that a subbag of $M(s)$ is to be purged. For example

- $A(s,t) = IA(s,t) = < \#(g_1, \ldots, g_n) >$. In this case, the subset of $C(s)$ is the singleton $G_s = \{< g_1, \ldots, g_n >\}$.
- $A(s,t) = IA(s,t) = < \#(g_1, \ldots, g_{i-1}, *, g_{i+1}, \ldots, g_n) >$ Here $G_s = \{(g_1, \ldots, g_n) \mid g_i \in G_i\}$.

The notation on the arc just represents a variable the type of which is the set of multisets over G_s. It allows the subset to be identified on the graph part of the P-Graph without having to refer back to the declaration part.

6.6 Purging Partitions of Markings

Consider a partition of $C(s)$

$$C(s) = \bigcup \{G^1, \ldots, G^k\}$$

and let $D^j = \mu G^j$ for $j = 1, \ldots, k$.

Define the colour set for transition, t, as follows.

$$C(t) = (\bigcup_j D^j) \times D'$$

where D' is as before.

Recalling the projection function ($\pi_s : C(t) \to \mu C(s)$) with $C(t) = D_s \times D'$ and $D_s = \bigcup_j D^j$, the pre and inhibitor maps are given by

- $Pre(s,t) = \pi_s$
- For all $m \in C(t)$, $I(s,t;m) = \pi_s(m) + INF(s,t;m)$

where for $m = (d, d')$, $d \in D_s, d' \in D'$ and $g^j \in G^j$ for all $j = 1, 2, \ldots, k$

$$mult(g^j, INF(s,t;m)) = \begin{cases} 0 & \text{if } d \in D^j \\ \infty & \text{otherwise} \end{cases}$$

Similar propositions to those of the previous section may now be stated.

Proposition 6.10 *Given a P-net with $t \in T$, $s \in S$, $g \in C(s)$, $C(s) = \bigcup\{G^1, \ldots, G^k\}$, $D^j = \mu G^j$ for $j = 1, \ldots, k$, $C(t) = (\bigcup_j D^j) \times D'$ and $Pre(s,t) = \pi_s$ then $\exists m = (d^j, d') \in C(t)$, $d^j \in D^j$, such that for any $M(s) \in \mu C(s)$,*

$$mult(g, Pre(s,t;(d^j, d')) = \begin{cases} mult(g, M(s)) & \text{if } g \in G^j \\ 0 & \text{otherwise} \end{cases}$$

Proof: Essentially the same as that for proposition 6.8. We have $Pre(s,t;(d^j,d')) = \pi_s(d^j,d') = d^j$ where $d^j \in \mu G^j$ and hence $\forall d^j$, $Pre(s,t;(d^j,d')) \in \mu G^j$. Now if $g \notin G^j$ then $g \notin \mu G^j$ and thus $mult(g, Pre(s,t;(d^j,d')) = 0$. Further if $g \in G^j$, we may always choose a multiset $d^j \in \mu G^j$ such that $mult(g,d^j) = mult(g,M(s))$ as they both range over N. Hence we can always choose a mode $m = (d^j,d')$ of $t \in T$, such that $mult(g, Pre(s,t;(d^j,d')) = mult(g,M(s))$. □

We can now state that an occurrence of t will purge a member of a partition of $C(s)$.

Proposition 6.11 *Given a P-net with $t \in T$, $s \in S$, $g \in C(s)$, $C(s) = \bigcup\{G^1,\ldots,G^k\}$, $D^j = \mu G^j$ for $j = 1,\ldots,k$, $C(t) = (\bigcup_j D^j) \times D'$, $Pre(s,t) = \pi_s$ and $I(s,t;m) = \pi_s(m) + INF(s,t;m)$ (as defined above) and $\forall m \in C(t), Post(s,t;m) = \emptyset$, then when t occurs in mode $m = (d^j,d') \in C(t)$, $d^j \in D^j$, place s is purged of all tokens in G^j, ie*

$$mult(g,M'(s)) = \begin{cases} 0 & \textit{if } g \in G^j \\ mult(g,M(s)) & \textit{otherwise} \end{cases}$$

Proof: Follows directly from proposition 6.9.□

6.6.1 Graphical Representation

Let $D_s = \bigcup_j \mu G^j$ be a carrier of the many-sorted algebra and include a variable $y : D_s$ in the signature. If we wish to purge place s of all elements of any member of a partition of $C(s)$ by an occurrence of transition t, then $(s,t) \in F$ and $(s,t) \in IF$ and $A(s,t) = IA(s,t) = <y>$.

6.6.2 Notation for Partitions of Product Sets

For G as defined above (section 6.5.2), consider a tuple of typed variables $(v_1,\ldots,v_n)$ where $\forall i \in In$, $v_i : G_i$. This represents a partition of G in which each element of G is a singleton set member of the partition. If $Part(G)$ denotes a partition of G, then

$$Part(G) = \{\{(g_1,\ldots,g_n)\} \mid g_i \in G_i, i \in In\}.$$

The $*$ notation can now be used in the same way as in section 6.5.2 to allow sums over the variables replaced by stars. For example, if v_i is replaced by a star we obtain the notation $(v_1,\ldots,v_{i-1},*,v_{i+1},\ldots,v_n)$, and the resulting partition is

$$Part(G) = \{\{(g_1,\ldots,g_n) \mid g_i \in G_i\}, \mid g_j \in G_j, j \in In \setminus \{i\}\}.$$

This may be extended to allow stars in any of the tuple positions.

$$Part(G) = \{\{(g_1,\ldots,g_n) \mid g_l \in G_l, l \in I_*\}, \mid g_j \in G_j, j \in In \setminus I_*\}$$

with I_* as defined in 6.5.2. If stars are placed in every position the partition is equal to the original set.

For $C(s) = G_1 \times \ldots \times G_n$, we may replace the variable $y : D_s$ annotating the reset arc (s,t), by the above tuple notation preceeded by a # to indicate the partition in the graph part. Two examples are

- $A(s,t) = IA(s,t) =< \#(v_1,\ldots,v_n) >$ The part in parenthesis is just notation for identifying a partition as defined above. A generic member of the partition is $G^j = \{(g_1,\ldots,g_n)\}$.
- $A(s,t) = IA(s,t) =< \#(v_1,\ldots,v_{i-1},*,v_{i+1},\ldots,v_n) >$ The part in parenthesis now defines another partition determined by the notation defined above.

We may of course have a star in any of the positions of the tuple. (When there is no partitioning of $C(s)$, the corresponding notation is $< \#(*,*,\ldots,*) >$ which is equivalent to Y, where $Y : \mu C(s)$.)

6.6.3 Example: Aborting a Broadcast

In concurrent systems design, situations often arise in which we wish to send information to a list of destinations, for example broadcast protocols. Quite often there is a mechanism for aborting, part-way through the broadcast. An elegant way of specifying this behaviour is to have a list of possible destinations for each source and store them in a place. To allow aborts to occur at any stage, the broadcast is done one destination at a time (the destination being chosen arbitrarily from the list), with the destinations that have been serviced being transferred one by one to another place (destinations serviced). When the abort occurs from a particular source, all the destinations that have been serviced are removed from the serviced list and reinstated on the original list. A P-net specification of the management of the list is given in figure 6.4. ID1 and ID2 are identity functions.

6.6.4 Purging a Selected Partition

This section illustrates a notation for purging all elements of a member of a partition of a place's colour set, that occur in the place's marking, where the member of the partition is selected according to the marking of another place. This situation arises in aborting broadcasts for example [3].

An example is shown in figure 6.5. The partition of A×B that is to be purged is determined by the value of the variable x annotating the arc $(p1, t1)$. y is a variable which ranges over the sets of multisets over each of the partition members. Without the transition condition, the colour set associated with $t1$ would be $C(t1) = \mathrm{A} \times \bigcup_{\mathrm{a}\in\mathrm{A}} \mu\{(\mathrm{a},\mathrm{b}) \mid \mathrm{b} \in\mathrm{B}\}$. This would correspond to any partition member being purged independently of the value of x. To ensure that the partition selected does depend on the value of x, a transition condition is used which restricts the colour set to

$$C(t1) = \bigcup_{\mathrm{a}\in\mathrm{A}}\{\mathrm{a}\} \times \mu\{(\mathrm{a},\mathrm{b}) \mid \mathrm{b} \in\mathrm{B}\}.$$

Although this representation does allow a direct translation from the P-Graph to the P-net, according to section 5.4, it is rather complicated to understand. A more easily understood graphical representation is shown in figure 6.6, where we have used the tuple notation developed in the previous section.

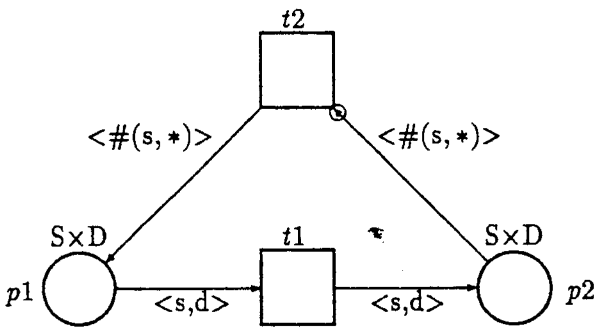
For a∈A, and x = a, we can read #(x,*) (associated with a reset arc) to mean: purge all tokens in place $p2$ with the value 'a' in the first position of the pair, when $t1$ occurs. The matching of the variable x in both arc inscriptions is used to imply that the transition condition of figure 6.5 applies.

P-Graph

Declarations

S: Finite set of source addresses
D: Finite set of destination addresses
Variables: s:S, d:D, #(s,*):$\bigcup_{s\in S}\mu\{(s,d) \mid d \in D\}$
$\forall s\in S, \forall d\in D$, $K(p1;s,d) = 1$
$\forall s\in S, \forall d\in D$, $K(p2;s,d) = 1$
$M_0(p1) = S\times D$
$M_0(p2) = \emptyset$

Graph



P-Net

$S = \{p1, p2\}$
$T = \{t1, t2\}$
$C = \{S\times D, \bigcup_{s\in S}\mu\{(s,d) \mid d \in D\}\}$
$C(p1) = C(p2) = S\times D$
$C(t1) = S\times D$
$C(t2) = \bigcup_{s\in S}\mu\{(s,d) \mid d \in D\}$
$Pre(p1,t1) = Post(p2,t1) = \mathrm{ID1}{:}S\times D \longrightarrow S\times D$
$Pre(p2,t2) = Post(p1,t2) = \mathrm{ID2}{:}C(t2) \longrightarrow \bigcup_{s\in S}\mu\{(s,d) \mid d \in D\}$
$\forall m1 \in C(t1), Pre(p2,t1;m1) = Post(p1,t1;m1) = \emptyset$
$\forall m2 \in C(t2), Pre(p1,t2;m2) = Post(p2,t2;m2) = \emptyset$
$\forall m2 \in C(t2), \forall (s,d) \in S\times D$
$I(p2,t2;m2) = \mathrm{ID2}(m2) + INF(p2,t2;m2)$ where
$$mult((s,d), INF(p2,t2;m2)) = \begin{cases} 0 & \text{if } m2 \in \mu\{(s,d) \mid d \in D\} \\ \infty & \text{otherwise} \end{cases}$$
$\forall m1 \in C(t1), \forall (s,d) \in S\times D$
$mult((s,d), I(p1,t1;m1)) = \infty$
$mult((s,d), I(p2,t1;m1)) = \infty$
$\forall m2 \in C(t2), \forall (s,d) \in S\times D$, $mult((s,d), I(p1,t2;m1)) = \infty$
The capacities and initial markings are given in the P-Graph.

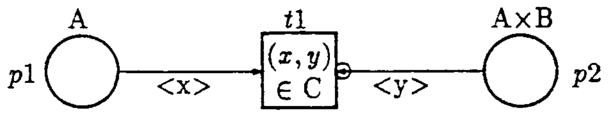
Figure 6.4: Aborts: Address List Management

P-Graph

Declarations

A: non-empty set
B: non-empty set
$C = \bigcup_{a \in A}\{a\} \times \mu\{(a,b) \mid b \in B\}$
Variables: x:A, y:$\bigcup_{a \in A} \mu\{(a,b) \mid b \in B\}$
$\forall a \in A, \forall b \in B,\ K(p2;a,b) = 1$
$M_0(p1) \subseteq A$
$M_0(p2) = A \times B$

Graph



P-Net

$S = \{p1, p2\}$
$T = \{t1\}$
$C = \{A, A\times B, \bigcup_{a \in A}\{a\} \times \mu\{(a,b) \mid b \in B\}\}$
$C(p1) = A$
$C(p2) = A \times B$
$C(t1) = \bigcup_{a \in A}\{a\} \times \mu\{(a,b) \mid b \in B\}$
$Pre(p1,t1) = \pi_1 : \bigcup_{a \in A}\{a\} \times \mu\{(a,b) \mid b \in B\} \rightarrow A$
$Pre(p2,t1) = \pi_2 : \bigcup_{a \in A}\{a\} \times \mu\{(a,b) \mid b \in B\} \rightarrow \bigcup_{a \in A} \mu\{(a,b) \mid b \in B\}$
$\forall m1 \in C(t1), Post(p1,t1;m1) = Post(p2,t1;m1) = \emptyset$
$\forall m1 = (c,d) \in C(t1), \forall (a,b) \in A\times B$
$I(p2,t1;m1) = \pi_2(m1) + INF(p2,t1;m1)$, where
$$mult((a,b), INF(p2,t1;(c,d))) = \begin{cases} 0 & \text{if } d \in \mu\{(a,b) \mid b \in B\} \\ \infty & \text{otherwise} \end{cases}$$
$\forall m1 \in C(t1), \forall a \in A,\ mult(a, I(p1,t1;m1)) = \infty$
$\forall a \in A,\ K(p1;a) = \infty$
$\forall a \in A, \forall b \in B,\ K(p2;a,b) = 1$
Initial markings are given in the P-Graph.

Figure 6.5: Selecting a member of a partition for purging

P-Graph

Declarations

A: non-empty set
B: non-empty set
Variables: x:A, #(x,*):$\bigcup_{a\in A}\mu\{(a,b) \mid b \in B\}$
$\forall a\in A, \forall b\in B$, $K(p2;a,b) = 1$
$M_0(p1) \subseteq A$
$M_0(p2) = A\times B$

Graph

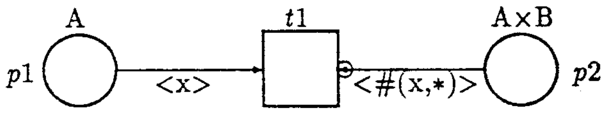


Figure 6.6: A more readable representation for purging a selected member of a partition

6.7 Purging Subsets of Partitions of Markings

The above two sections may be generalised to allow for the purging of a subset of a partition. Let

$$C(s) = \bigcup\{G^1, \ldots, G^k\}$$

as before and I_s be a set of indices that index members of the partition included in the subset, so that $I_s \subseteq \{1, 2, \ldots, k\}$. The set of occurrence modes for transition, t, is then defined as

$$C(t) = (\bigcup_{j\in I_s} D^j) \times D'$$

where $D^j = \mu G^j$ and the pre and inhibitor maps become

- $Pre(s,t) = \pi_s$
- For all $m \in C(t)$, $I(s,t;m) = \pi_s(m) + INF(s,t;m)$

where for $m = (d, d')$, $d \in \bigcup_{j\in I_s} D^j, d' \in D'$ and $g^j \in G^j$ for all $j \in I_s$

$$mult(g^j, INF(s,t;m)) = \begin{cases} 0 & \text{if } d \in D^j \\ \infty & \text{otherwise} \end{cases}$$

The propositions of section 6.6 apply when $j \in I_s$.

6.7.1 Graphical Representation

Let $D_s = \bigcup_{j\in I_s} \mu G^j$ be a carrier of the many-sorted algebra and include a variable $y : D_s$ in the signature. If we wish to purge place s of all elements of any member of a subset of a partition of $C(s)$ by an occurrence of transition t, then $(s,t) \in F$ and $(s,t) \in IF$ and $A(s,t) = IA(s,t) =<y>$.

6.7.2 Notation for Subsets of Partitions of Product Sets

We can combine the notation developed in sections 6.5.2 and 6.6.2 to obtain a tuple notation consisting of constants, variables and stars to obtain subsets of partitions of product sets. With G, g_i, v_i and $i \in In$ as previously defined we consider two situations.

The tuple $(g_1, \ldots, g_{i-1}, v_i, g_{i+1}, \ldots, g_n)$ represents a subset of the partition where all positions have variables. The subset is

$$\{\{(g_1, \ldots, g_n)\} \mid g_i \in G_i\}.$$

We can generalise this for variables in any number of positions. Let $I_v \subseteq In$ be the set of positions in which variables occur, then the subset is

$$\{\{(g_1, \ldots, g_n)\} \mid g_l \in G_l, l \in I_v\}.$$

When $I_v = In$ we have the notation described in section 6.6.2, where all positions of the tuple are occupied by variables.

A complete notation is now obtained by allowing stars to occur in any position. Let $I_* \subseteq In$ be the set of positions in which stars occur. The subset of partitions described above (see section 6.6.2) when using stars and variables is

$$\{\{(g_1, \ldots, g_n) \mid g_k \in G_k, k \in I_*\} \mid g_l \in G_l, l \in I_v\}$$

Of course, I_v and I_* must be disjoint.

For example, the tuple $(g_1, \ldots, g_{i-1}, v_i, g_{i+1}, \ldots, g_{k-1}, *, g_{k+1}, \ldots, g_n)$ represents the following subset of a partition defined above (see section 6.6.2) with all variables in the tuple except for a star at position k.

$$\{\{(g_1, \ldots, g_n) \mid g_k \in G_k\} \mid g_i \in G_i\}$$

As before, when $C(s)$ is a product set, we can use this notation, preceeded by a #, to replace the variable $y : D_s$ annotating a reset arc (s,t) to indicate the subset of the partition on the graph form. An example is

$$A(s,t) = IA(s,t) =< \#(3, b, *) >$$

where $C(s) = N \times Bool \times N_8$ with $b : Bool$ a Boolean variable and $N_8 = \{0, 1, \ldots, 7\}$.

Chapter 7

Communications Examples

7.1 Queues

Queues are important data structures in communications. For example they are often used when defining service specifications. We would therefore like to include a syntax for the manipulation of queues.

A first observation is that the colour set will consist of strings, A^*, over an *alphabet*, A. We may have for a place, s, representing a queue or set of queues, that $C(s) = A^*$, so that a token represents a queue. We shall allow the queue to consist of structured items, so that A may be the union of products of sets (eg $A = (B \times C) \cup D \cup (E \times F \times G)$).

We shall denote the empty string by ε and the set of strings over A of length no more than n by A^{n*}.

7.1.1 Functions

We define two functions that will be generally useful for the specification of queueing systems.

Concatenation: A binary function on strings which appends one string to the tail of another.

$$. : A^* \times A^* \longrightarrow A^*$$

where if $x = a_1a_2\ldots a_j$ and $y = b_1b_2\ldots b_k$, then $x.y = a_1a_2\ldots a_jb_1b_2\ldots b_k$, with $a_i \in A, i = 1\ldots j$ $b_i \in A, i = 1\ldots k$ and $x, y \in A^*$.

We use the convention that the first letter of the string is on the left and the last on the right.

Word Length: A unary function returning the length of a string.

$$WL : A^* \longrightarrow N$$

where for $x = a_1a_2\ldots a_n$, $WL(x) = n$. It will be convenient to use outfix notation $|x|$ for $WL(x)$.

7.1.2 Predicates

We need predicates for string lengths. The standard binary predicates *less than* $<$, *less than or equal to* $\leq$, *equals* $=$, *not equals* $\neq$, *greater than* $>$ and *greater than or equal to* $\geq$ are useful. For example $|x| < n$, where $x \in A^*, n \in N^+$, to restrict a queue size or $|x| = n$ to select strings of length n.

7.1.3 Examples

FIFO Queues

First-In-First-Out (FIFO) queues are important models for communications services and protocols as they are a convenient way of representing sequence preservation. Let us firstly consider an infinite (unbounded) FIFO queue.

A convenient representation is shown in the P-Graph of figure 7.1 along with its linear form. Transition AF (Add to FIFO) adds items of type A to the queue, whereas SF (Serve FIFO) removes the head of the queue. The concatenation operator has been defined above (section 7.1.1).

We now consider a bounded FIFO queue of length n as shown in figure 7.2. The addition of items to the queue is restricted by the typing of q to sequences of length less than n ($|q| < n$). Since $|x| = 1$, this ensures that the queue size never exceeds its bound, n. If $M(FIFO) = \{s\}$ and $|s| = n$, then AF is not enabled.

In an application, A may be the union of a number of product sets, which will allow items to have their own structure. We may also index queues by setting $C(FIFO) = J \times A^*$ where J is a set of indices, which could also be a product set. A later example illustrates these features.

The above representations have the problem that they require interleaving of enqueueing and dequeueing, whereas these actions are inherently concurrent (unless the queue is full or empty).

Where it is important that enqueueing and dequeueing are concurrent, the FIFO queue can be represented as a circular buffer with pointers to the head and tail of the queue. This representation is not as abstract and is at the level of an implementation. The costs are in the complexity of the representation (you need to keep track of the values of the pointers) and in the increase in the number of states. The situation is depicted in figure 7.3.

LIFO Queue

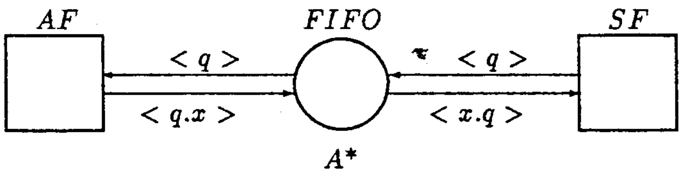
The Last-In-First-Out (LIFO) queue or *stack* is the same as the FIFO queue except that items are added to the head (instead of the tail) of the queue. Thus the pre map for adding is the same as the post map for serving and vice versa. A bounded LIFO queue is shown in figure 7.4. Transition AL (Add-to-LIFO) adds items to the queue and transition SL (Serve-LIFO) removes items from the queue.

With LIFO queues there is contention between adding items to the queue and removing them. Thus it is important that these activities are in conflict, which is the case in figure

P-Graph

Declarations

A: Set of Queue Items
A^*: Set of strings over A
ε: Empty string
$x : A, q : A^*$
'.' concatenation operator
$M_0(FIFO) = \{\varepsilon\}$



P-Net

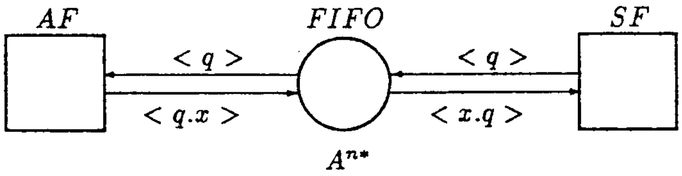
$$
\begin{array}{|l|}
\hline
S = \{FIFO\} \\
T = \{AF, SF\} \\
C = \{A^*, A \times A^*\} \\
C(FIFO) = A^* \\
C(AF) = C(SF) = A \times A^* \\
\forall x \in A, q \in A^*, Pre(FIFO, AF; (x, q)) = \{q\} \\
\forall x \in A, q \in A^*, Pre(FIFO, SF; (x, q)) = \{x.q\} \\
\forall x \in A, q \in A^*, Post(FIFO, AF; (x, q)) = \{q.x\} \\
\forall x \in A, q \in A^*, Post(FIFO, SF; (x, q)) = \{q\} \\
\forall q \in A^*, K(FIFO; q) = \infty \\
M_0(FIFO) = \{\varepsilon\} \\
\hline
\end{array}
$$

Figure 7.1: Infinite FIFO Queue

P-Graph

Declarations

A: Set of Queue Items
A^{n*}: Set of strings, length $\leq n$
ε: Empty string
$x : A, q : A^{(n-1)*}$
'.' concatenation operator
$M_0(FIFO) = \{\varepsilon\}$



P-Net

$S = \{FIFO\}$
$T = \{AF, SF\}$
$C = \{A^{n*}, A \times A^{(n-1)*}\}$
$C(FIFO) = A^{n*}$
$C(AF) = C(SF) = A \times A^{(n-1)*}$
$\forall x \in A, q \in A^{(n-1)*}, Pre(FIFO, AF; (x, q)) = \{q\}$
$\forall x \in A, q \in A^{(n-1)*}, Pre(FIFO, SF; (x, q)) = \{x.q\}$
$\forall x \in A, q \in A^{(n-1)*}, Post(FIFO, AF; (x, q)) = \{q.x\}$
$\forall x \in A, q \in A^{(n-1)*}, Post(FIFO, SF; (x, q)) = \{q\}$
$\forall s \in A^{n*}, K(FIFO; s) = \infty$
$M_0(FIFO) = \{\varepsilon\}$

Figure 7.2: Bounded FIFO Queue

Declarations

A: Set of Queue Items
$N_n = \{0, 1, \ldots, n-1\}, n \in N^+$
$i : N_n, x : A$
$\oplus : N_n \times N_n \rightarrow N_n$ modulo n addition
$M_0(FIFO) = \emptyset$
$M_0(PT) = M_0(PH) = \{0\}$

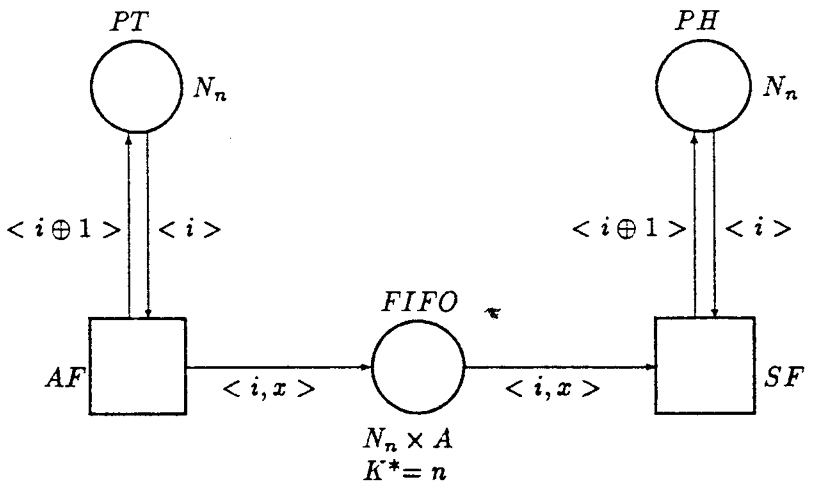


Figure 7.3: Concurrent Bounded FIFO Queue

Declarations

A: Set of Queue Items
A^{n*}: Set of strings, length $\leq n$
ε: Empty string
$x : A, q : A^{(n-1)*}$
'.' concatenation operator
$M_0(FIFO) = \{\varepsilon\}$

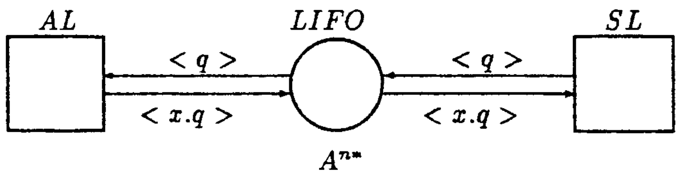


Figure 7.4: Bounded LIFO Queue

7.4. We could build a model of the LIFO that did not involve sequences of items as tokens, but single items as tokens. In this case we would need to have a pointer to the head of the queue, with mutually exclusive access to it for adding and removing. This has the disadvantage of having to store the value of the pointer and leads to a less abstract description.

7.2 Demon Game

In this section a small example is presented that is being used as a test case for formal methods being developed in ISO and CCITT with application to Open Systems Interconnection protocols and services. The example is called the Demon (Daemon) Game [1].

The following provides a description of the demon game which is slightly more abstract than the narrative description in [1] in that no assumption is made regarding communication. Thus there is no reference to the use of 'signals', as this is considered to be prejudicing an implementation. It is believed that the spirit of the game is still the same!

7.2.1 Narrative Description

Assume that there is a demon which generates *bumps*. The aim of the game is to guess when there has been an odd number of bumps generated. The demon informs a player of the outcome of the guess: either *win* or *lose* corresponding to there being an odd or even number of bumps respectively, at the time of the guess. The demon keeps a score which is initially zero. It is incremented for a successful guess and decremented if unsuccessful. A player can request his score at any time and the result will be returned by the demon.

The game can be played by several players. Before starting a game, a player must log-in. A unique identifier is allocated to a player on logging-in and deallocated on logging-out.

7.2.2 MPrT-net Specification

The Demon Game can be specified using an MPrT-net without transition conditions. It illustrates the use of simple many-sorted unary operators.

The game can be specified by 4 places and 5 transitions with their associated inscribed arcs and is given in figure 7.5. The top two transitions and associated arcs and places specify the behaviour of players logging-in and logging-out. The next two transitions specify how to play the game (guessing and requesting the cumulative score) and the bottom transition specifies the bumping of the demon.

The convention is used that if the annotation of the arcs associated with the same place and transition are the same ($A(s,t) = A(t,s)$), then both the arcs and the annotations are superimposed, producing a singly annotated arc with an arrowhead at both ends. As an example, see $f1$ = (Scores,Request) and $f2$ = (Request,Scores) in figure 7.5.

Information about players is represented as a ternary relation comprising: an identifier; the outcome of a guess (including initially the *null* outcome denoting that no guess has yet been made); and a score. This state information is stored as the marking of place *Players*.

Declarations

Set of Player Identifiers:I
Set of Game States:G = {win,lose,null}
State of Bumps:B = {even,odd}
Set of Integers:Z
Variables b:B;i:I;g:G;s,r:Z
Functions
Complement:B→B where
$\overline{\text{even}}$ =odd and $\overline{\text{odd}}$ = even
Score S:B→ $\{-1, 1\}$ where
S(even) = -1 and S(odd) = 1
Outcome O:B→G where
O(even) = lose and O(odd) = win
M_0(IDs) = I
M_0(Scores) = $\emptyset$
M_0(Players) = $\emptyset$
M_0(Bumps) = even

Graph

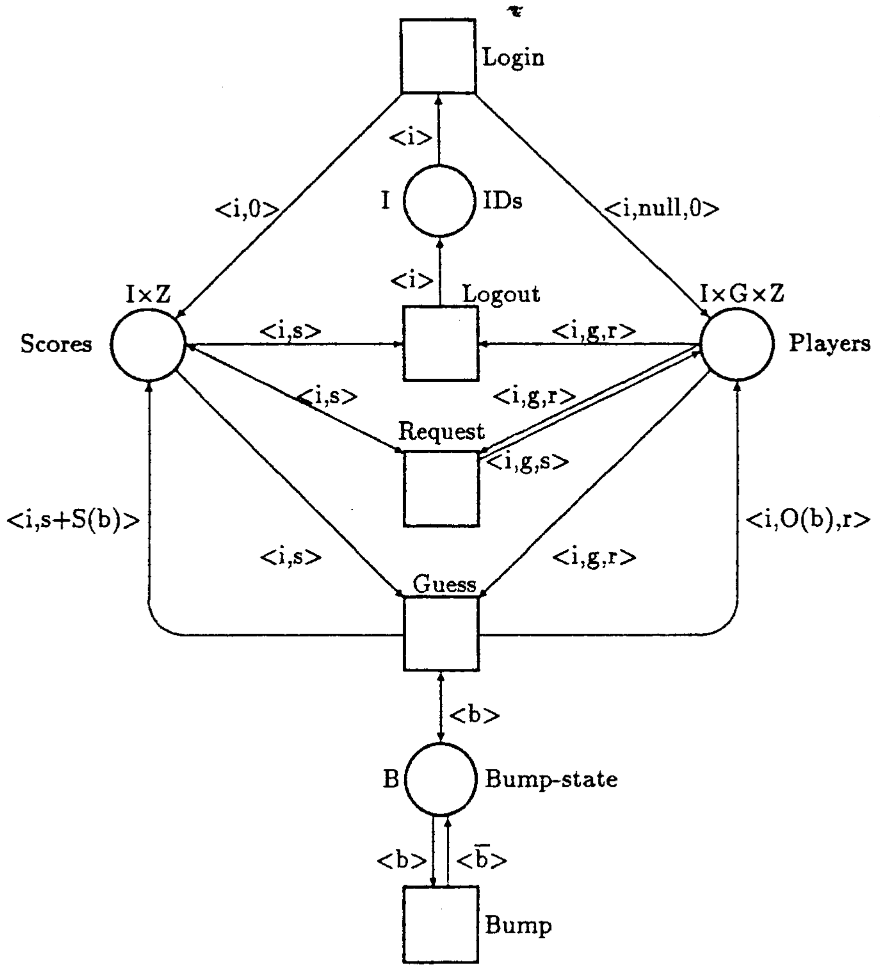


Figure 7.5: MPrT-Net of Demon Game

Unused identifiers are stored in place *IDs*; players' scores in *Scores*; and the state of the demon's bumps in *Bump-state*.

Initially, there are no players (place *Players* is empty); no scores (place *Scores* is empty); all identifiers are available (place *IDs* is marked with the complete set of identifiers I); and the demon has not begun to bump. As far as the game is concerned it is only important to model the state of the bumps as *even* or *odd*, there is no need to count the actual number of bumps. Thus initially there are an even (zero) number of bumps (place *Bumps* is marked with the token *even*).

On logging-in (transition *Login*), a player's state and score is initialised, and his identifier is removed from the unused identifier list. He may now make a guess (transition *Guess*) whereupon his score is updated and he is informed of the outcome. He may also request his score (transition *Request*) or logout (transition *Logout*) with his identifier being returned to the unused list and all information about him being destroyed. The demon bumps whenever it wishes.

7.2.3 Discussion of Concurrency, Conflict and Interleaving

The bumping is arbitrarily interleaved with players making guesses (a conflict). Similarly, after logging-in, a player may (non-deterministically) make a guess; request his score; or logout (another conflict). This interleaving behaviour is an essential part of the design. For example, it makes no sense to be able to logout and request the score simultaneously. In analogy with the Readers/Writers problem it also makes no sense to guess and bump at the same time or to guess and request the score simultaneously. These situations are naturally in conflict and require interleaving of these events.

On the other hand, for a particular player, the events of requesting a score or logging in or out are completely independent of the demon bumping. Hence transitions *Login* and *Bump*; *Logout* and *Bump*; and *Request* and *Bump* are concurrent.

We would also expect that all players would act independently of one another and this is mostly the case. Any number of players may login, logout or request their scores concurrently but are limited to interleaving when making guesses. Here we have made the assumption that 'read access' to the bump-state is exclusive. This is not essential and it is valuable to delay such decisions to the implementation phase.

This limitation may be overcome by making copies of the Bump-state, and removing all the old ones when the demon bumps (transition *Bump*). Let us assume that there can be n simultaneous accesses to the bump-state, where $n \in N^+$, then setting $A(\textit{Bump-state}, \textit{Bump}) = n < \mathrm{b} >$ and $A(\textit{Bump}, \textit{Bump-state}) = n < \overline{\mathrm{b}} >$ achieves the desired specification. *Bump* and *Guess* are still in conflict, but *Guess* may occur concurrently with itself limited by n and the number of players logged-on.

These more subtle parts of the design would be completely glossed over with a technique based on interleaving semantics. With an interleaving model, the implementer could be unaware of which parts of the specification were intentionally in conflict and which could be concurrent. Also, the need to specify the number of simultaneous accesses to a resource could be completely overlooked.

Chapter 8

Discussion of Future Work

This report embodies a set of investigations concerning how the expressive power of CP-nets may be extended while maintaining mappings back to CP-nets to allow for the analysis of P-nets using techniques applicable to CP-nets. These investigations are by no means complete and this section sets out some of the issues still needing further work. The future work on the P-Graph and extended capacity notation has been particularly stimulated by the comments of Jensen [12].

8.1 P-Graph

8.1.1 Tupling

In the P-Graph definition we have included explicit tupling in the signature, in a similar way to PrT-nets [8]. This has the advantage that only relatively simple sorts need be included in the signature. It has the disadvantage that it makes for a more complex definition. If complex sorts (eg those corresponding to products sets in the algebra) are allowed in the signature, then tupling can always be done with identity operators in the signature. Places may now be typed by a single sort and the relationship between sorts and colour sets becomes more transparent as places may now be R-sorted with the associated carrier (in the algebra) being the corresponding colour set. This is a more elegant approach and it is pursued further in the next section.

The use of angular brackets for enclosing tuples has also followed the practice of PrT nets. Its use is to distinguish between scalars and values (for example we would like to distinguish $2 < 1 >$ from 21). The use of angular brackets can be criticised in that often 3 characters are used when only one would suffice (eg $< x >$ instead of x). A notation which allows scalars and values to be distinguished when required would be preferable. The optional use of brackets would be one example (ie x, $2x$, and $2(1)$ would be valid syntax).

8.1.2 Abstract P-Graphs

The P-Graph defined previously included concrete colour sets, markings and capacities. This is often the level at which telecommunication and other systems are specified. However it is very useful to have a more abstract specification that allows classes of systems to be specified. For example the range of sequence numbers or window sizes in protocols may be left open. The hope is that it will be possible to prove properties about systems for a whole range of parameter values by just considering the more abstract specification.

This is the approach adopted by Vautherin [22] where he defines a Petri net-like schema, Σ-schema, and provides an interpretation for it with a class of CP-nets. Vautherin does not include capacity or inhibitor functions, only allows equations to be associated with transitions and does not explicitly type variables in the signature. The following defines a schema addressing these points. The term *Abstract P-Graph* is used for the schema as suggested in [12].

Definition

A **Abstract P-Graph** or P-Graph Schema is a structure

$$\mathbf{APG} = (IN, \Sigma, \tau, AN, \mathcal{K}, \mathcal{M}_0)$$

where

- $IN = (S, T; F, IF)$ is an inhibitor net, with
  - S a finite set of places;
  - T a finite set of transitions disjoint from S;
  - $F \subseteq (S \times T) \cup (T \times S)$ a set of arcs; and
  - $IF \subseteq S \times T$ a set of inhibitor arcs.
- $\Sigma = (R, \Omega, V)$ is a Natural-Boolean signature with variables.
- $\tau : S \rightarrow R$ is a function that types places. (Places are R-sorted.)
- $AN = (A, IA, TC)$ is a triple of net annotations.
  - $A : F \rightarrow BTERM(\Omega \cup V)$ such that for $s \in S$, $(x, y) \in F$, $A(s, y), A(x, s) \in BTERM(\Omega \cup V)_{\tau(s)}$. Arcs are annotated with a multiset of terms which have the same type as the associated place.
  - $IA : IF \rightarrow B_\infty TERM(\Omega \cup V)$ such that for every $(s, t) \in IF$, $IA(s, t) \in B_\infty TERM(\Omega \cup V)_{\tau(s)}$. Inhibitor arcs are annotated with a multiset of terms which have the same type as the associated place.
  - $TC : T \rightarrow TERM(\Omega \cup V)_{Bool}$ where for all $t \in T$, $TC(t) \in TERM(\Omega \cup V(t))_{Bool}$ and $V(t)$ is the set of variables occurring in the arc inscriptions associated with t. TC is a function which annotates transitions with Boolean expressions.
- $\mathcal{K} : S \rightarrow \mu^+_\infty TERM(\Omega)$ where $\forall s \in S$, $\mathcal{K}(s) \in \mu^+_\infty TERM(\Omega)_{\tau(s)}$ is the capacity function associating a multiset of closed terms with each place.
- $\mathcal{M}_0 : S \rightarrow \mu TERM(\Omega)$ such that $\forall s \in S$, $\mathcal{M}_0(s) \subseteq \mathcal{K}(s)$, is the initial marking at a syntactic level which respects the capacity.

$BTERM(\Omega \cup V)$ and $B_\infty TERM(\Omega \cup V)$ are multisets of terms generated in the same way that multisets of tuples were generated in section 5.2.6.

Interpretation as a P-net

For a many-sorted algebra, H, satisfying the R-sorted signature, the interpretation of the abstract P-Graph as a P-net is given in section 5.4, with the following exceptions.

1. Colour Sets. For each place $s \in S$, $C(s) = H_{\tau(s)}$. $C(t)$ is the same as in section 5.4 and $C = \{C(x) \mid x \in S \cup T\}$.
2. Capacity Function. For all $s \in S$, $K(s) = Val_H(\mathcal{K}(s))$.
3. Initial Marking. For all $s \in S$, $M_0(s) = Val_H(\mathcal{M}_0(s))$.

8.2 Extended Capacity Notation

When extending the capacity notation care was taken to give an interpretation in terms of another P-Graph. This meant that the basic P-net definition of the enabling condition was retained.

A different and more general approach has been suggested by Jensen [12], where another *capacity* colour set is associated with each place. Hence we need a *capacity colour function.* If $C_K(s)$ is the capacity colour set associated with place $s \in S$, then we form the set $CAP = \{(s,c) \mid c \in C_K(s), s \in S\}$ and define the capacity to be $K \in \mu^+_\infty CAP$. We introduce a new function $\kappa : \mu PLACE \rightarrow \mu CAP$ and replace the capacity enabling condition by

$$\kappa(M + Post'(T_\mu)) \subseteq K$$

The usefulness of this generalisation and its effect on transformations back to CP-nets are yet to be studied.

8.3 P-net to CP-net Transformations

The transformations presented in this report preserve the interleaving behaviour of the P-net. Further work is envisaged in two directions. Firstly the restrictions imposed on the pre and inhibitor maps to ensure that the truly concurrent behaviour is preserved, need to be investigated. It is currently considered that this will be the case for many applications, particularly where inhibitors are used for purging. Secondly, other transformations using partial complementary places need to be studied. This is indicated by the MPrT net of safe train operation. Place $p2$ and associated arcs could be removed and replaced by an appropriately inscribed inhibitor arc from place $p1$ to $t1$, indicating that there were no trains on sections $i \oplus 1$ and $i \ominus 2$. The properties preserved by these transformations also need to be investigated.

8.4 Analysis

No attempt has been made in this report to present a set of analysis methods for P-nets. However, due to the possibilities of transforming from P-nets back to CP-nets, the analysis methods developed for CP-nets will be able to be applied. These analysis techniques include reachability, invariants and reductions [6], model checking and also analysis via the *skeleton* PT-net as discussed in [22]. It may also be the case that some of these techniques may be able to be applied directly on the P-net.

8.5 Applications

A number of examples have been included in this report for illustrative purposes, but they have all been rather small. The aim of the work is that P-nets will be able to be applied to large industrial applications and particularly within the telecommunication and information services sector. This will require ways of structuring and refining specifications. The development of hierarchical CP-nets [12] shows promise here. This area has not been addressed in this report and it will require a large amount of effort in the future.

Acknowledgements

I gratefully acknowledge the valuable discussions with my supervisor, Dr Glynn Winskel, during the course of this work. In particular he suggested the use of many-sorted algebras for P-Graphs, provided considerable help in formalising proofs and has made numerous comments on earlier drafts which have considerably improved the report. I am also very grateful for the detailed comments and suggestions of Dr Kurt Jensen on an earlier version of this report. The comments of Dr Wolfgang Reisig and my colleague Geoff Wheeler have also been taken into account.

The permission of the Director Research, Telecom Australia, to publish this report is hereby acknowledged.

References

[1] ISO/TC 97/SC 21/WG1. Revised guidelines for the application of formal description techniques to OSI. ISO/TC 97/SC 21 N1534, September 1986. Output of joint WG1/FDT-A and CCITT WP X/3.

[2] E. Best and C. Fernandez. *Notations and Terminology on Petri Net Theory.* Arbeitspapiere 195, GMD, January 1986.

[3] Jonathan Billington. *A High-Level Petri Net Specification of the Cambridge Fast Ring M-Access Service.* Technical Report 121, University of Cambridge Computer Laboratory, New Museums Site, Pembroke Street, Cambridge, England, December 1987.

[4] Jonathan Billington. Specification of the Transport Service using Numerical Petri Nets. In C. Sunshine, editor, *Protocol Specification, Testing and Verification*, pages 77–100, North Holland, Amsterdam, 1982.

[5] Jonathan Billington, Geoffrey Wheeler, and Michael Wilbur-Ham. PROTEAN: a high-level Petri net tool for the specification and verification of communication protocols. *IEEE Transactions on Software Engineering, Special Issue on Tools for Computer Communication Systems*, SE-14(3):301–316, March 1988.

[6] W. Brauer, W. Reisig, and G. Rozenberg, editors. *Petri Nets: Central Models and Their Properties.* Volume 254 of *Lecture Notes in Computer Science*, Springer-Verlag, Berlin, 1987. Advances in Petri Nets 1986, Part 1: Proceedings of an Advanced Course, Bad Honnef, September, 1986.

[7] H. Ehrig and B. Mahr. *Fundamentals of Algebraic Specification 1, Equations and Initial Semantics.* Volume 6 of *EATCS Monographs on Theoretical Computer Science*, Springer-Verlag, Berlin, 1985.

[8] Hartmann J. Genrich. Predicate/Transition Nets. In W. Brauer, W. Reisig, and G. Rozenberg, editors, *Petri Nets: Central Models and their Properties. Advances in Petri Nets 1986, Part 1: Proceedings of an Advanced Course, Bad Honnef, September 1986*, pages 207 - 247, Springer-Verlag, Berlin, February 1987. Lecture Notes in Computer Science, Volume 254.

[9] Hartmann J. Genrich and Kurt Lautenbach. System modelling with high-level Petri nets. *Theoretical Computer Science*, 13:109–136, 1981.

[10] Kurt Jensen. Coloured Petri Nets. In W. Brauer, W. Reisig, and G. Rozenberg, editors, *Petri Nets: Central Models and Their Properties. Advances in Petri Nets*

1986, Part 1: Proceedings of an Advanced Course, Bad Honnef, September 1986, pages 248 – 299, Springer-Verlag, Berlin, February 1987. Lecture Notes in Computer Science, Vol. 254.

[11] Kurt Jensen. Coloured Petri Nets and the invariant-method. *Theoretical Computer Science*, 14:317–336, 1981.

[12] Kurt Jensen. Private communication. July 1988.

[13] J. Loeckx. Algorithmic specifications: a constructive specification method for abstract data types. *ACM Transactions on Programming Languages and Systems*, 9(4):646 – 685, October 1987.

[14] G. Memmi. Analysing nets by the invariant method. In W. Brauer, W. Reisig, and G. Rozenberg, editors, *Petri Nets: Central Models and Their Properties*, pages 300 – 336, Springer-Verlag, Berlin, 1987. Lecture Notes in Computer Science, Vol. 254.

[15] James L. Peterson. *Petri Net Theory and the Modeling of Systems.* Prentice Hall, Englewood Cliffs, N.J., 1981.

[16] W. Reisig and J. Vautherin. An algebraic approach to high level Petri nets. In *Proceedings of the Eighth European Workshop on Application and Theory of Petri Nets*, pages 51–72, Zaragoza, Spain, 24-26 June 1987.

[17] Wolfgang Reisig. *Petri Nets, An Introduction.* Volume 4 of *EATCS Monographs on Theoretical Computer Science*, Springer-Verlag, Berlin, 1985.

[18] F. J. W. Symons. *Modelling and Analysis of Communication Protocols using Numerical Petri Nets.* PhD thesis, University of Essex, 1978. Dept. of Elec. Eng. Sci. Telecommunication Systems Group Report No. 152, May 1978.

[19] F.J.W. Symons. Introduction to Numerical Petri Nets, a general graphical model of concurrent processing systems. *Australian Telecommunication Research*, 14(1), 1980.

[20] R. Valk. Self-modifying nets, a natural extension of Petri nets. In *Automata, Languages and Programming, Udine (Lecture Notes in Computer Science 62)*, pages 464–476, Springer-Verlag, 1978.

[21] J. Vautherin. Calculation of semi-flows for Pr/T-systems. In M.K. Molloy and M.K. Vernon, editors, *Proceedings of the International Workshop on Petri Nets and Performance Models*, pages 174 – 183, IEEE Computer Society Press, Washington D.C., August 1987.

[22] J. Vautherin. Parallel systems specifications with Coloured Petri Nets and algebraic specifications. In G. Rozenberg, editor, *Advances in Petri Nets 1987*, pages 293 – 308, Springer-Verlag, Berlin, April 1987. Lecture Notes in Computer Science, Vol. 266.

[23] G. R. Wheeler. *Numerical Petri Nets - A Definition.* Research Laboratories Report 7780, Telecom Australia, May 1985.

[24] Glynn Winskel. Petri nets, algebras, morphisms, and compositionality. *Information and Computation*, 72(3):197 – 238, March 1987. Earlier version published in University of Cambridge Computer Laboratory Technical Report 79, September 1985.

Appendix A

Sets, Multisets and Vectors

A.1 Sets

We shall make use of the following sets:

- $N = \{0, 1, \ldots\}$ the natural numbers.
- $N_\infty = N \cup \{\infty\}$
- $N^+ = N \setminus \{0\}$, the positive integers
- $N^+_\infty = N^+ \cup \{\infty\}$
- $Z = \{\ldots, -1, 0, 1, \ldots\}$, the integers

A.2 Multisets

We define a multiset, B, (also known as a bag) over a *basis* set, A, to be the function

$$B : A \longrightarrow N$$

which associates a multiplicity, possibly zero, with each of the basis elements. The set of multisets over A is denoted by μA (ie $\mu A = [A \longrightarrow N]$). For a multiset $B \in \mu A$, to avoid confusion, we sometimes use the notation $mult(a, B) = B(a)$ where $a \in A$, for the multiplicity of a in B.

We may extend the definition to include the value ∞, and denote the set of multisets over A, that allows infinite multiplicities, by $\mu_\infty A = [A \longrightarrow N_\infty]$ and that which disallows multiplicities of zero by $\mu^+_\infty A = [A \longrightarrow N^+_\infty]$.

A.2.1 Vector or Sum representation

We may represent a multiset as a symbolic sum of basis elements scaled by their multiplicity.

$$B = \sum_{a \in A} B(a)a$$

A.2.2 Membership

Given a multiset, $B \in \mu_\infty A$, we say that $a \in A$ is a member of B, denoted $a \in B$, if $B(a) > 0$, and conversely if $B(a) = 0$, then $a \notin B$.

The empty multiset, $\emptyset$, has no members: $\forall a \in A$, $\emptyset(a) = 0$.

A.2.3 Cardinality

We define *multiset cardinality* in the following way. The cardinality $|B|$ of a multiset B, is the sum of the multiplicities of each of the members of the multiset.

$$|B| = \sum_{a \in A} B(a)$$

A.2.4 Equality and Inclusion

Two multisets, $B1, B2 \in \mu A$, are equal, $B1 = B2$, *iff* $\forall a \in A$, $B1(a) = B2(a)$, and $B1$ is contained in $B2$, $B1 \subseteq B2$ *iff* $\forall a \in A$, $B1(a) \leq B2(a)$.

A.2.5 Operations

We define the following four 'set-like' operations on multisets $B1, B2 \in \mu A$. For all $a \in A$ we have:

Union	$B = B1 \cup B2$	*iff*	$B(a) = max(B1(a), B2(a))$
Intersection	$B = B1 \cap B2$	*iff*	$B(a) = min(B1(a), B2(a))$
Addition	$B = B1 + B2$	*iff*	$B(a) = B1(a) + B2(a)$
Subtraction	$B = B1 - B2$	*iff*	$B(a) = B1(a) - B2(a)$ when $B1(a) \geq B2(a)$

We also define scalar multiplication of a multiset, $B1 \in \mu A$, by a natural number, $n \in N$, to be

$$B = nB1 \textit{ iff } \forall a \in A, B(a) = n \times B1(a)$$

A.3 Vectors

There are times when we wish to subtract one multiset from another when the above restriction on multiset subtraction does not apply. We then need to consider multisets as vectors. We define a vector, V, over a (basis) set, A, to be the function

$$V : A \longrightarrow Z$$

which associates a negative, zero or positive multiplicity, with each of the basis elements. The set of vectors over A is denoted by νA (ie $\nu A = [A \longrightarrow Z]$). For a vector, $V \in \nu A$, to avoid confusion, we sometimes use the notation $mult(a, V) = V(a)$ where $a \in A$, for the multiplicity of a in V.

Subtraction is a closed operation for vectors defined component-wise as follows. For $V1, V2 \in \nu A$

$$V = V1 - V2 \textit{ iff } \forall a \in A,\ V(a) = V1(a) - V2(a)$$

We can also define scalar multiplication of a vector, $V1 \in \nu A$, by an integer, $z \in Z$, to be

$$V = zV1 \textit{ iff } \forall a \in A, V(a) = z \times V1(a)$$

A.3.1 Equality and Inclusion

Two vectors, $V1, V2 \in \nu A$, are equal, $V1 = V2$, *iff* $\forall a \in A$, $V1(a) = V2(a)$, and $V1$ is less than $V2$, $V1 \subseteq V2$, *iff* $\forall a \in A$, $V1(a) \leq V2(a)$.

Appendix B

Some comments on Numerical Petri Nets

B.1 Background

Numerical Petri Nets (NPNs) were developed for the modelling and analysis of protocols with the original ideas coming from Symons [18,19]. NPNs evolved over several years, culminating in a definition by Wheeler [23].

It is instructive to compare P-nets with NPNs in the same algebraic framework. Firstly we note the major differences with the main stream of Net Theory.

B.2 Main NPN Distinguishing Features

What makes NPNs different from other high-level nets? The main differences concern the enabling condition and transition rule.

B.2.1 Enabling Condition

Numerical Petri Nets generalise the enabling condition of Petri Nets to allow for a set of binary predicates on the marking of a place. General logical formulae are then built up with the negation operator and usual logical connectives.

B.2.2 Transition Rule

The transition rule is in general not dependent on the enabling condition. The multiset of tokens to be removed from the input places when a transition occurs is defined separately. In the framework of CP-nets, this corresponds to the function $I_{-}(s,t)$ with the difference that this function need not be part of the enabling condition.

B.3 Definition

We can define NPNs in the same framework that we have used for the definition of P-nets. A Numerical Petri Net is a structure $NPN = (\mathrm{N}, Pre2)$, where N is a P-net, except that the capacity function is restricted as already discussed (see section 5.8). The function $Pre2$ is like Pre

$$Pre2 : TRANS \longrightarrow \mu PLACE$$

It defines a multiset over which a logical formula is built for the precondition. The I mapping is also used as a threshold in NPNs for the 'between X and Y' notation, where the image of I defines the multiset for 'Y'. These two maps define the 'Input Condition' of NPNs.

The images of the *Pre* and *Post* mappings define the 'Destroyed Tokens' and 'Created Tokens' of NPNs respectively.

The 'Transition Condition' is taken into account by the colour function. It just provides a restriction on $C(t)$ in a similar way to that described in section 5.4.

Numerical Petri Nets have a set of variables that can be associated with particular transitions, known as 'p-variables'. The value of a p-variable may be changed by a 'Transition Operation'. These variables are taken into account in the above definition. Each p-variable of type W can be represented as a token in a place of colour W, where the token is the current value of the variable. Any transition operations on the p-variable can be simulated by the *Pre* and *Post* mappings.

A semantics for the 'oldest notation' of NPNs is provided in the section on queues (section 7.1) for P-nets.

B.4 Enabling Condition

The multiset returned by $Pre2$ is used with a set of binary predicates involving the marking $M \in \mu PLACE$. Let $PRED$ be a set of binary predicate symbols, $tr \in TRANS$, $p \in PLACE$ and $Q \in PRED$, then $Q(Pre2(tr;p), M(p))$ is an atomic predicate.

Consider a partition of $PLACE$: $PART(PLACE) = \{P_i : 1 \leq i \leq n\}$.

Logical formulae are built inductively as follows:

- for $1 \leq i \leq n$, $Q \in PRED$, $(\forall p_i \in P_i)Q(Pre2(tr;p_i), M(p_i))$ is a formula
- If l_1 and l_2 are formulae, then
  1. $\neg l_1$
  2. $l_1 \vee l_2$
  3. $l_1 \wedge l_2$

  are formulae. (The connectives $\rightarrow, \leftrightarrow$ could be derived in the usual way, although these are not specifically included in NPNs as defined in [23].)

Let L_{tr} be the set of logical formulae so constructed with the restriction that a formula must be effectively quantified over $p \in PLACE$. That is, there must be exactly one occurrence of a formula of the form $(\forall p_i \in P_i)Q(Pre2(tr;p_i), M(p)_i)$ for every member of the partition of $PLACE$ as part of a formula in L_{tr}.

In NPNs three binary predicate symbols $(\leq, =, \geq)$ are defined. When the predicates are uniformly applied across the net, they can be replaced by predicates defined on multisets. These are inclusion (the usual predicate of nets), equality and superbag ($Pre2(tr) \subseteq M$, $Pre2(tr) = M$ and $Pre2(tr) \supseteq M$).

A transition, $tr \in TRANS$ is enabled at a marking M, *iff*

$$L_{tr} \wedge (M \subseteq K - Post(tr)) \wedge M \subseteq I(tr)$$

B.5 Transition Rule

Numerical Petri Nets are only defined with interleaving semantics, so we only need to consider the occurrence of one transition at a time. As usual if several transitions are enabled at the same time, one is chosen to occur arbitrarily.

If a transition tr is enabled at M, then it may occur resulting in a new marking M' given by

$$M' = (M - Pre(tr)) + Post(tr)$$

where '+' is multiset addition, but '−' is an extension of multiset subtraction that is closed, defined as follows. If $B, B1, B2$ are multisets over A and $B' = B1 \cap B2$, then $\forall a \in A$

$$B = B1 - B2 \Rightarrow B(a) = B1(a) - B'(a)$$

This transition rule is similar to the one for CP-nets, but it takes into account the fact that one cannot remove tokens from a place if they dont exist. This only happens because the enabling condition and transition rule have been divorced in NPNs. This feature provides difficulties in translating NPNs to P-nets in a straightforward way (see discussion below).

B.6 Discussion

The above formulation is slightly more general than that described in [23], where a specific syntax and semantics has been given for the 'Input Conditions'. However, it is believed that the above formulation does capture the spirit of NPNs.

NPNs gain expressive power by supporting a more complex enabling condition but with the following penalties:

- A more complex structure with only interleaving;
- A reliance on reachability analysis as there is no clear mapping back to other forms of nets which have richer analysis possibilities.

The main question to ask is whether the extra expressive power is worth the penalties. My experience to date suggests that mostly it is not worth it. Most of the functionality of protocols and services can be expressed quite easily with P-nets. The advantage with P-nets is that the structure is less complex (and hence easier to learn) and that there are well defined mappings back to CP-nets under weak assumptions. It is also clear when the inhibitor extension is required whereas with NPNs it may not be so clear.

Since both P-nets and NPNs have the modelling power of Turing Machines, it is always possible to simulate an NPN with a P-net. In most cases translation of an NPN to a P-net will be straightforward. It is conjectured that the *Pre*2 and *I* maps of an NPN can be translated into the *Pre* and *I* maps of a P-net and the *Pre* and *Post* maps of an NPN can be can be simulated by the *Pre* and *Post* maps of a P-net. For example, if *Pre*2 just involves inclusion, then it translates to *Pre* in the P-net. If *Pre*2 involves equality, then it translates to $Pre = I = Pre2$. If *Pre*2 involves superbag, then it translates to *I*. *I* in the NPN will translate to *I* in the P-net.

In most cases, $Post - Pre$ in the NPN will equal $Post - Pre$ in the P-net where subtraction is vector subtraction, with the restriction that for any $tr \in TRANS$, $[Post(tr;p)]_{P-net} \geq 0$. When $\exists p \in PLACE$ such that

$$[Post(tr;p)]_{NPN} - [Pre(tr;p)]_{NPN} + [Pre(tr;p)]_{P-net} < 0$$

the translation is not straightforward and a single mode of an NPN transition would have to be mapped onto several modes of a P-net transition.

It is not the intent of this appendix to provide a detailed translation as this would be a major task and there is little justification for it. This is because alot of the more unusual NPN constructs have very rarely been used in applications and that it is no longer intended that NPNs be used as a specification technique for distributed systems. The intent has been to show that NPNs can be cast into the same framework as P-nets and that most of the time a translation from NPNs to P-nets will be relatively straightforward.