Protocol Specification Using P-Graphs, a Technique Based on Coloured Petri Nets

Jonathan Billington

Telecommunications Systems Engineering Centre
Institute for Telecommunications Research, University of South Australia
Warrendi Road, Mawson Lakes, Adelaide, 5095, Australia

Abstract. P-Graphs combine inhibitor Petri nets and abstract data types within the same algebraic framework. They are useful for the specification of concrete concurrent systems and in particular communication protocols. The inhibitor has been included to allow compact descriptions of systems by promoting the economy of data types. They are also necessary for the purging of resources; a common activity when modelling protocols or their services. This paper introduces P-Graphs with the aid of some simple examples. It also shows how to map P-Graphs to P-nets, which are Coloured Petri Nets (CP-nets) extended with place capacities and inhibitors. This is important for the analysis of P-Graph specifications, as P-nets can be transformed to CP-nets in almost all practical situations. Thus the analysis techniques of CP-nets can then be applied. Useful notation for capacities are introduced and their semantics defined in terms of the P-Graph. A notation for purging places of their tokens is also introduced, involving the superimposition of the inhibitor and normal arc. Two case studies, the Demon game and the M-Access Service of the Cambridge Fast Ring, are included to illustrate the use of the P-Graph and the extended notation for protocol specification.

1 Introduction

Since the early 1970's Petri nets have been used for the modelling and analysis of systems that involve communication, synchronization, co-operation and concurrency [30]. Some of the reasons for this are their foundation in concurrency, their ability to be analysed and executed by machine and their graphical appeal allowing the dynamics of a system to be visualised by playing the token game. This has certainly been the case for communication protocols. The earliest work on the modelling of protocols with nets was probably that of Merlin [55]. Early surveys of the use of Petri net based techniques for the modelling of protocols were carried out by Diaz [27, 28] with a more specific treatment in [6]. Burkhardt et. al. [22] presents a methodology for the specification of Open Systems Interconnection services and protocols using a high-level net technique called Product nets. During the 1980's High-level nets were applied to the specification and analysis of many complex protocols and services [12, 25, 3, 49, 21, 4, 2, 38, 5, 26, 34, 33, 53]. More recently a tutorial [15] and a workshop [16] on the application of Petri nets to protocols have been held in conjunction with the annual Petri net conferences.

A further workshop [29] has shown how nets can be applied to multimedia systems. The third volume [47] of Jensen's book on Coloured Petri Nets contains some excellent examples of their application to communication protocols and services. Recent applications are to mobility services [64], information infrastructure such as traders [65] and multi-agent systems [18]. The most recent papers on the application of Petri nets to Communication networks are being compiled for a special issue of Advances in Petri nets [19].

Although Petri nets have sufficient modelling power for most (if not all) practical systems, they suffer from a lack of modelling convenience or elegance particularly for the representation of data. This led to the development of a number of hybrid net/data models [59, 58, 50, 63], that associated a set of variables with the net and/or attributes with tokens, that could be modified on the occurrence of transitions. Thus the 1970's witnessed a number of useful experiments with adding a more convenient data representation to nets, driven by the needs of practical applications. These models could be analysed using reachability techniques and simulation, but other techniques used to analyse nets (structure theory, invariants, reductions and synchronic distance) were no longer applicable as these extended nets did not come with an underlying Petri net semantics.

This problem was tackled in the next decade, where we have seen the development of *high-level* nets where tokens are data items and arcs and transitions are inscribed with symbolic expressions. The earliest of these were Predicate/Transition nets (PrT nets) [35, 37] and Coloured Petri nets (CP-nets) [42] which included methods for calculating invariants. Since then, Predicate/Event (P/E) nets, Relation nets [62] and Algebraic nets [61] have been developed and PrT nets and CP-nets have been reformulated [36, 43, 45, 46]. The links between abstract data types (ADTs) [31] and high-level nets appear to have been discovered in the mid 1980's [51, 67, 7, 52, 1, 13, 9, 60]. Here the approach has been to combine the strengths of ADTs for data representation with the strengths of Petri nets (synchronisation, concurrency, graphics) within the same algebraic framework.

Following on from [9], the main purpose of this paper is to present a class of high-level inhibitor nets, known as the P-Graph, that combines abstract data types and Petri nets, and to illustrate its use with some simple examples and then to apply it to two case studies. The approach to the definition of P-Graphs has been inspired by [36, 61, 67, 43] and is similar to that of [67] but differs in a number of aspects. Firstly we consider nets with inhibitors and capacities and more general arc inscriptions. (In this presentation we do not consider the axioms of ADTs but they can easily be added). Secondly P-Graphs are defined at the concrete level, where places are typed by sets chosen from the carriers of a many-sorted algebra that satisfies the ADT signature, and markings and capacities are multisets over these sets. Inscriptions are at the level of terms. This has similarities with [61] and is appropriate for the specification of concrete systems.

P-Graphs can be interpreted by Coloured Petri Nets, extended by capacities and inhibitors, known as P-nets [8]. This is important for analysis because in most practical situations, P-nets can be transformed to CP-nets [8] and all the analysis techniques applicable to CP-nets can be employed.

Why have inhibitors and capacities been introduced? The main reason is to provide for modelling convenience when specifying protocols and their services, while retaining analysis possibilities as discussed above. This is demonstrated in two ways. An abstract railway signalling protocol is firstly modelled by a CP-Graph, a subclass of P-Graphs without capacities and inhibitors. This model necessitates the introduction of a data type to specify the control aspects of the system. It is then modelled with a P-Graph, where it is shown that this extra control data type is no longer required. Secondly, the inhibitor allows a theory of purging places of their tokens with a single event, to be developed [10]. Purging places is useful when modelling protocol procedures such as aborting, disconnecting or resetting connections as recognised in [12] for example. It turns out that so long as a finite capacity (and colour set) can be associated with the place to be purged, then the P-Graph can be transformed to a finite CP-net for analysis. The use of a purging construct (the reset arc) is illustrated when modelling the M-Access Service of the Cambridge Fast Ring.

The paper is organised as follows. After giving a definition of P-nets in section 2, concepts from algebraic specification are recalled in section 3 to provide the necessary background for the definition of P-Graphs in section 4 and for their interpretation as a P-net in section 5. Section 6 discusses the graphical representation of the P-Graph and in section 7, CP-Graphs are defined as a subclass of P-Graphs and illustrated with two examples. The first is the abstract railway protocol and the second is a resource sharing management scheme which illustrates the use of more complex arc inscriptions. The railway system is remodelled with a P-Graph in section 8. It illustrates the use of the inhibitor and capacity extensions. A notation for capacity for the P-Graph is developed in sections 9 and 10. Two case studies follow. Firstly, the specification of the Demon Game, an example used by the International Standards Organisation as a test of formal description techniques for protocols and services, is presented in section 11. Secondly, a specification of the M-Access Service of the Cambridge Fast Ring is used to illustrate the capacity and purging notations. Finally some conclusions are drawn in the closing section.

No attempt to compare P-Graphs with other formal description techniques has been undertaken as this is beyond the scope of this paper. The interested reader is referred to [11] for example.

2 P-nets

P-nets are Coloured Petri Nets [43] (CP-nets) extended by the capacity function and the threshold inhibitor map. This section provides the basic definitions. Further discussion including P-net to CP-net transformations may be found in

[8]. All notation and terminology for sets, multisets, vectors and their associated operations is defined in the Appendix.

2.1 Definition

A **P-net** is a structure $P = (S, T, \mathcal{C}; C, Pre, Post, I, K, M_0)$ where

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

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

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

- $I : TRANS \longrightarrow \mu_\infty PLACE$ is the threshold inhibitor map
- $K \in \mu^+_\infty PLACE$ is a multiset known as the place capacity; and
- $M_0 \in \mu PLACE$ is a multiset known as the initial marking which must comply with the place capacity so that $M_0 \leq K$

2.2 Marking

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

2.3 Enabling

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

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

where

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

is a linear extension with $P = Pre$ or $Post$, and $\forall tr \in TRANS$, and $T1_\mu, T2_\mu \in \mu TRANS$ we have

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

Thus a multiset of transition modes 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 to receive tokens when the transition modes occur, and the inhibitor thresholds are not exceeded.

2.4 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 $M[T_\mu\rangle M'$ or for a single mode $M \xrightarrow{tr} M'$.

2.5 Set of Reachable Markings

The set of reachable markings, $[M_0\rangle$, of P is obtained inductively as follows.

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

3 Concepts from Algebraic Specification

In the P-Graph, we shall inscribe arcs with multisets of terms involving variables, and transitions with Boolean expressions. Many-sorted signatures 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 P-Graph and abstract P-Graph. We make use of the ideas found in [31, 54] for example. This section is only included to make the paper self-contained and to introduce the required terminology. Those familiar with algebraic specification may like to skip this section.

3.1 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, e.g. *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 the set of operator names 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* or *argument* sorts, and r as the *output* or *range* 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.

3.2 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, in a similar way to that of constants, from the set of variable names 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 (i.e. $v_a \in V_r$ *iff* $a = r$).

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

3.3 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 (N).

3.4 Terms of a Signature with Variables

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

1. $V_r \subseteq TERM(\Omega \cup V)_r$;
2. For all $w_{(\varepsilon,r)} \in \Omega$, $w_{(\varepsilon,r)} \in TERM(\Omega \cup V)_r$; and
3. If $e_1 : r_1, \ldots, e_n : r_n$ are terms and $w_{(r_1 \ldots r_n,r)} \in \Omega$, is an operator, then $w_{(r_1 \ldots r_n,r)}(e_1, \ldots, e_n) \in TERM(\Omega \cup V)_r$

Thus if *Int* is a sort, integer constants and variables, and operators (with appropriate arguments) of output sort *Int* are terms of sort *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, also known as *ground* terms) by $TERM(\Omega)$. Thus

$$TERM(\Omega \cup V) = \bigcup_{r \in R} TERM(\Omega \cup V)_r$$

3.5 Multisets of Terms

Multisets or *bags* of terms can also be built inductively from the signature if we assume that we have a Natural signature. We define multisets of terms this way to allow the multiplicities to be terms of sort Nat, rather than just the Naturals themselves. (This allows, for example, the introduction of conditions into arc expressions - see sections 4.2.)

Let $BTERM(\Omega \cup V)$ denote the set of multisets of terms, defined inductively as follows, using the symbolic sum representation for multisets defined in Appendix A. ($TERM(\Omega \cup V)$ is considered as a special set of multisets, where each member of $TERM(\Omega \cup V)$ is a multiset.)

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

Where there is no confusion the '$\times$' will be dropped and juxtaposition will be used for scalar multiplication (e.g. '$3 \times x$ can be replaced by $3x$ and $4 \times 3 \times x$ by $4 \times 3x$ which is distinctly different from $43x$.)

The set of bags with infinite multiplicities, $B_\infty TERM(\Omega \cup V)$, may now be defined as follows

- $BTERM(\Omega \cup V) \subset B_\infty TERM(\Omega \cup V)$; and
- if $b \in BTERM(\Omega \cup V)$, then $\infty \times b \in B_\infty TERM(\Omega \cup V)$.

where multiplication by ∞ is defined in appendix A.

3.6 Many-sorted Algebras

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

$$w_H : H_{r_1} \times \ldots \times H_{r_n} \rightarrow H_r.$$

In case an operator is a constant, w_r, then there is a corresponding element $w_H \in H_r$. They may be considered as functions of arity zero.

Definition: A many-sorted Algebra, H, is a pair

$$H = (R_H, \Omega_H)$$

where $R_H = \{H_r | r \in R\}$ is the set of carriers and
$\Omega_H = \{w_H | w_{\sigma,r} \in \Omega, \sigma \in R^* \text{and } r \in R\}$ the set of corresponding functions.

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

$$H = (Z, Boolean; lessthan)$$

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, variables are R-sorted. In the algebra, the variable is typed by the carrier corresponding to the sort.

3.7 Assignment and Evaluation

Given an R-sorted algebra, H, with variables in V, an *assignment* [1] for H and V is a family 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 ass comprising

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

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 $r, r_1, \ldots, r_n \in R$ and $e, e_1, \ldots, e_n \in TERM(\Omega \cup V)$,

- If $e \in V_r$ is a variable, then $ass_r(e) = \alpha_r(e)$
- For a constant, $w_r \in \Omega$, $ass_r(w_r) = w_H \in H_r$.
- If $e = w_{(\sigma,r)}(e_1, \ldots, e_n)$, then
  $ass_r(w_{(\sigma,r)}(e_1, \ldots, e_n)) = w_H(ass_{r_1}(e_1), \ldots, ass_{r_n}(e_n)) \in H_r$, where $e_1 : r_1 \ldots e_n : r_n$.

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

- $Val_\alpha(i \times a) = ass(i) \times ass(a)$
- $Val_\alpha(b1 + b2) = Val_\alpha(b1) + Val_\alpha(b2)$

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

4 P-Graphs

This section defines a **P-Graph**. A P-Graph consists of an inhibitor net where the arcs are annotated by multisets 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. A colour function associates a colour set with each place. The colour set is a carrier of the many-sorted algebra. The capacity and initial marking are multisets over the place's colour set.

4.1 Definition

A **P-Graph** is a structure

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

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.
- $\Sigma = (R, \Omega, V)$ is a Natural-Boolean signature with variables.
- $H = (R_H, \Omega_H)$ a corresponding Σ-Algebra.
- $C : S \rightarrow R_H$ is the colour function, such that $\forall s \in S, C(s) \neq \emptyset$.
- $AN = (A, IA, TC)$ is a triple of net annotations.
  - $A : F \rightarrow BTERM(\Omega \cup V)$ such that for $C(s) = H_r$, then for all $(s,t),(u,s) \in F$, $A(s,t), A(u,s) \in BTERM(\Omega \cup V)_r$. It is a function that annotates arcs with a multiset of terms of the same sort as the carrier associated with the arc's place.
  - $IA : IF \rightarrow B_\infty TERM(\Omega \cup V)$ such that for $C(s) = H_r$, then for all $(s,t) \in IF$, $IA(s,t) \in B_\infty TERM(\Omega \cup V)_r$. It is a function that annotates inhibitor arcs with a multiset of terms of the same sort as the carrier associated with the arc's place.
  - $TC : T \rightarrow TERM(\Omega \cup V)_{Bool}$ where for all $t \in T$, $TC(t)$ belongs to $TERM(\Omega \cup V(t))_{Bool}$ and $V(t)$ is the set of variables occurring in the arc inscriptions associated with t.
    TC 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 $\forall s \in S$, $M_0(s) \leq K(s)$, is the initial marking.

4.2 Discussion

When generating multisets of terms 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 as a special case, the *generalised Kronecker delta* extension to PrT-nets [36]. An example of a variant of the readers/writers problem is given in [17] to illustrate the utility of this extension.

5 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: 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 [43], 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 colour sets for the places are obtained from the colour function. Thus the structuring set (of colour sets) is given by $\mathcal{C} = \{C(x) | x \in S \cup T\}$.
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 family of 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 $\forall m \in C(t)$, $Post_{(s,t)}(m) = \emptyset$.
   Thus for all $t \in T$ and for all $m \in C(t)$
   $$Pre(t,m) = \{(s,b) \mid s \in S, b \in Pre_{(s,t)}(m)\}$$
   $$Post(t,m) = \{(s,b) \mid s \in S, b \in Post_{(s,t)}(m)\}$$

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.

With this translation from the P-Graph to P-nets in place, we may now use the definitions of marking, enabling and transition rule for P-nets to allow the P-Graph to be executed. (Alternatively, we could define the enabling condition and the transition rule directly for the P-Graph, by considering assignments for terms in a similar way to [61].)

6 Graphical Form of P-Graph

6.1 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.

6.2 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. (As mentioned above, $C(s)$, $K(s)$ 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 if a place $s \in S$ is not annotated by a capacity multiset, then it will have infinite capacity for all tokens in $C(s)$, unless specified otherwise in the Declaration.

Useful notation for $K(s)$ is given later in sections 9 and 10.

6.3 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 for transition t, $TC(t)$, only involves the variables of the inscriptions of its surrounding arcs. If a transition, t, is left blank, then the Transition Condition is true ($TC(t) = true$).

6.4 Arcs

As usual arcs are represented by arrows. For $(s,t) \in F$, an arrow is drawn from place s to transition t and vice versa for $(t,s) \in F$. If (s,t) and (t,s) have the same inscriptions (s is a side place of t), $A(s,t) = A(t,s)$, then this may be shown by a single arc with an arrowhead at both ends and annotated by single inscription.

An inhibitor arc, $(s,t) \in IF$, is represented by an edge from place s to transition t with a small circle instead of an arrow head at its destination.

The arcs will be annotated with multisets of terms of the same type (or subtype) of their associated place. We therefore need a convenient representation for multisets. We use the symbolic sum or vector representation described in appendix A. In order to distinguish multiplicities from terms, the convention is adopted that terms may be enclosed in angular brackets.

6.5 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 again use the symbolic sum representation and distinguish multiplicities from tokens, by enclosing tokens in angular brackets. Thus if $g \in M(s)$, g or $<g>$ could 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 (e.g. (a,b,c) would be represented by $<a,b,c>$ and not $<(a,b,c)>$.)

7 CP-Graphs

On removing the inhibitor arcs and the place capacities from the P-Graph, we obtain a subclass that is very similar to Jensen's 'CP-**graph**' [43]. We shall distinguish our class, called CP-Graphs, from that of Jensen by using an upper case 'G' in 'Graph'. The CP-**graph** differs from the CP-Graph defined here in two respects:

- it is a multigraph (i.e. multiple arcs are allowed between places and transitions); and
- the arc inscriptions and transition conditions ('**guards**') are not explicitly defined.

Jensen [43] states that the expressions and guards may be defined by means of a many-sorted algebra (but excludes this from his scope of concern) and that has provided part of the stimulus for the definition of P-Graphs.

Definition

A CP-Graph (**CPG**) is a P-Graph, $(IN, \Sigma, H, C, AN, K, M_0)$, with the following restrictions

- $IN = (S, T; F, \emptyset)$ i.e. no inhibitor arcs.
- $AN = (A, \emptyset, TC)$ i.e. no inhibitor arc annotations.
- For all $s \in S$, $K(s) = \{(g, \infty) | g \in C(s)\}$ i.e. the capacities of the places are infinite.

7.1 Abstract Railway Signalling Protocol

In [36], 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 CP-Graph is given in figure 1 where any number of sections greater than 4 is allowed.

In this introductory example we have jumped to the level of the algebra by defining functions and sets, and typing variables, rather than including the signature explicitly. Full details are given in the next example (section 8).

A token in place $p1$ represents a train on a particular section of track. Place $p2$ represents the control data - i.e. the vacant sections. The occurrence of transition $t1$ represents movement of the trains along the track. The variable 'x' ranges over the set of trains, and 'i' over the sections. The arc inscriptions ensure that a train on section 'i' can only move to section 'i$\oplus$1' when sections 'i$\oplus$1' and 'i$\oplus$2' are vacant.

Initially train 'a' is on section 0 and train 'b' is on section 2, fulfilling the requirement that the trains cannot be on the same or adjacent sections. Thus sections 0 and 2 are not vacant and hence $0, 2 \notin M(p2)$. We can interpret the dynamics of the net in a very similar way to PrT-nets. Thus we can bind 'x' to any value in 'T' and 'i' to any value in 'I'. For example, if we have 'x=a' and 'i=0' then the demand on place $p1$ is satisfied, $M(p1) \geq$<0,a>, but the demand on place $p2$ is not satisfied as $2 \notin M(p2)$. Hence $t1$ is not enabled. If however we have 'x=b' and 'i=2', then $t1$ is enabled (and this is the only binding for which $t1$ is enabled in the initial marking). When $t1$ occurs, token <2,b> of place $p1$ is replaced by <3,b> and token 3 of place $p2$ is replaced by 2.

The new marking now allows both trains to move concurrently (so long as $n > 5$). This can be seen by the bindings: 'x=a', 'i=0'; and 'x=b', 'i=3'. Thus a step may occur by $t1$ occurring in both these modes.

The model is a little different from the PrT-net in [36]. Apart from the minor difference of generalising the number of track sections, the functions are total

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 $\oplus$:I×I→I is modulo n addition
Place $p1$: Sections occupied by trains
Place $p2$: Vacant sections
$M_0(p1) = \{<0,a>,<2,b>\}$
$M_0(p2) = I \setminus \{0, 2\}$

Graph

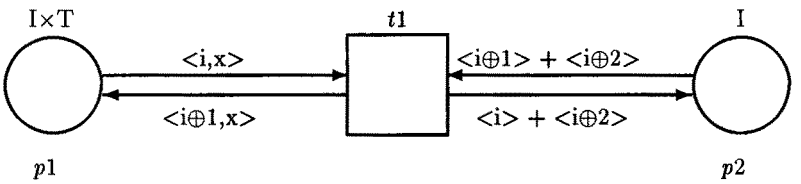

Fig. 1. CP-Graph of Safe Train Operation

and the variables are explicitly typed. There is also no need for a transition condition.

8 P-Graph Example: Genrich's Train revisited

The train example above provides us with a very simple illustration of the use of the inhibitor arc. Given that this is the first example of the use of the inhibitor and capacity extensions, we shall describe it in full detail.

8.1 Linear P-Graph

The linear P-Graph for the safe train is given in figure 2.

In this example we have explicitly shown how tupling (in this case pairing) can be achieved with a suitable tupling operator declared in the signature.

8.2 Graphical Form

The graphical form of the P-Graph for the operation of the train is given in figure 3. As usual we include only the information about the algebra in the

Linear P-Graph

$S = \{p1\}$, $T = \{t1\}$, $F = \{(p1, t1), (t1, p1)\}$, $IF = \{(p1, t1)\}$
$R = \{r1, r2, r3, Nat, Bool\}$
$\Omega = \{\oplus_{r1r1,r1}, (\_,\_)_{r1r2,r3}\} \cup Natconst \cup \{a_{r2}, b_{r2}, true_{Bool}\}$
where $Natconst$ is the set of natural constants including infinity
$V = \{i_{r1}, x_{r2}\}$
$H = (R_H, \Omega_H)$; $R_H = \{H_{r1}, H_{r2}, H_{r3}, H_{Nat}, H_{Bool}\}$
$H_{r1} = \mathrm{I} = \{0, 1, \ldots, n-1 \mid n > 4\}, n \in N$
$H_{r2} = \mathrm{T} = \{\mathrm{a}, \mathrm{b}\}$
$H_{r3} = \mathrm{I} \times \mathrm{T}$
$H_{Nat} = N_\infty$; $H_{Bool} = \{true, false\}$
$\Omega_H = \{\oplus_H, (\_,\_)_H, a_H, b_H, true_H\}$
$a_H = \mathrm{a}; b_H = \mathrm{b}; true_H = true$
$\oplus_H : \mathrm{I} \times \mathrm{I} \rightarrow \mathrm{I}$ is modulo n addition
$(\_,\_)_H : \mathrm{I} \times \mathrm{T} \rightarrow \mathrm{I} \times \mathrm{T}$ is a pairing function where
$\forall j \in \mathrm{I}, \forall \mathrm{t} \in \mathrm{T}, (j, \mathrm{t})_H = (j, \mathrm{t})$
$C(p1) = \mathrm{I} \times \mathrm{T}$
$A(p1, t1) = (i, x)$, $A(t1, p1) = (i \oplus 1, x)$
$IA(p1, t1) = 0 \sum_{j=1}^{2} \sum_{u \in U} (i \oplus j, u) + \infty \sum_{j \in J} \sum_{u \in U} (i \oplus j, u)$
where $J = \{0, 3, 4, \ldots, n-1\}$ and $U = \{\mathrm{a}, \mathrm{b}\}$
$TC(t1) = true$
$K(p1) = \{((j, u), 1) | (j, u) \in \mathrm{I} \times \mathrm{T}\}$ (i.e. the set $\mathrm{I} \times \mathrm{T}$)
$M_0(p1) = \{(0, \mathrm{a}), (2, \mathrm{b})\}$

Fig. 2. Linear P-Graph of Safe Train Operation

Declaration and type variables with the appropriate carrier. The tupling operator and function are considered primitive without any need to define them each time in a Declaration. I have also been less formal with the use of operator names and functions in not distinguishing between them (i.e. $\oplus$ has been used as an operator and also as a function). Also infix notation has been used as it is customary.

For inhibitor arcs we use the convention that zero multiplicities are shown explicitly, whereas infinite multiplicities are assumed for any term that is not shown explicitly (c.f. pre map arcs which assume that zero multiplicities are not shown in the sum). We have also used '$*$' notation to represent sums of tuples. It is defined as follows:

Let $(x, y) : A \times B$, then $(x, *) = \sum_{b \in B} (x, b)$.

This can be generalised to tuples of any length, by allowing the sum to be over the domains of all the variables replaced by stars.

The graphical form provides a compact specification of the behaviour of the trains on the track. The occurrence of $t1$ again indicates the movement of a train from section i to section i$\oplus$1. This is possible if there is a train on section i, (pre condition) and there are no trains on sections i$\oplus$1 and i$\oplus$2 (inhibitor condition). Of course the concurrent moving of trains is allowed, so long as the conditions are met for different trains on different sections of track. For example, on a 10

Declarations

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

Graph

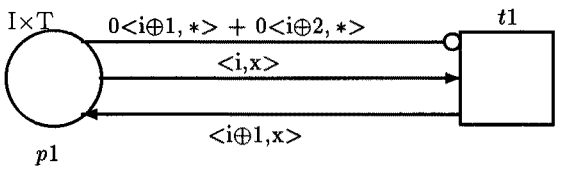

Fig. 3. P-Graph of Safe Train Operation

section track (n = 10) if train 'a' is on section 4 and train 'b' on section 9, then the bindings of i=4 and x=a **and** i=9 and x=b, both satisfy the enabling condition when taken together.

With the CP-Graph model, the control flow aspects are separated out (using place $p2$) from the data flow and thus emphasised. In effect another (redundant) data type has been introduced. This is fine when we wish to emphasise this aspect. At a more abstract level we may not want to make this separation. In this case CP-Graphs do not have the necessary modelling convenience. The P-Graph allows us to have just the one data type; there is no need for any redundancy.

The P-Graph has also made us think about resource limitations. Here it is sensible that only one train can be on one track at any time, instead of an unlimited number. This provides a further check on the system and allows the P-Graph to be transformed into an equivalent CP-net (see [8] for details) for analysis.

8.3 Equivalent P-net

To illustrate the transformation from a P-Graph to a P-net, the equivalent P-net is given in figure 4.

P-net

$S = \{p1\}$, $T = \{t1\}$
$C = \{\mathrm{I} \times \mathrm{T}\}$
$C(p1) = C(t1) = \mathrm{I} \times \mathrm{T}$
$Pre(p1, t1) = \lambda(i, x).(i, x)$, the identity function on $\mathrm{I} \times \mathrm{T}$
$Post(t1, p1) = \lambda(i, x).(i \oplus 1, x)$, a permutation of $\mathrm{I} \times \mathrm{T}$
$I(p1, t1) : \mathrm{I} \times \mathrm{T} \rightarrow \mu(\mathrm{I} \times \mathrm{T})$
$I(p1, t1) = \lambda(i, x).(0 \sum_{j=1}^{2} \sum_{u \in U} (i \oplus j, u) + \infty \sum_{j \in J} \sum_{u \in T} (i \oplus j, u))$
where $J = \{0, 3, 4, \ldots, n-1\}$
$K(p1) = \mathrm{I} \times \mathrm{T}$
$M_0(p1) = \{(0, \mathrm{a}), (2, \mathrm{b})\}$

Fig. 4. P-net of Safe Train Operation

9 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 when annotating places of the P-Graph, as the place is indicated by the proximity of the annotation to the place. Thus for the capacity of token $g_1 \in C(s)$, we may write (for $n_1 \in N_{\infty}^{+}$) $K(g_1) = n_1$ next to place s, instead of $K(s; g_1) = n_1$. Of course, this will only be practical when $C(s)$ is a very small set, or when most of the capacities are the same.

A special case is when the capacity for each token $g \in C(s)$ is the same, say $n \in N^{+}$. This is the same as the capacity defined for PrT-nets [37], and we use the same notation. Thus if place s is annotated by $K = n$ in the P-Graph, then this means $\forall g \in C(s), K(s; g) = n$.

10 Extended Capacity Notation

Although the P-net capacity function and the above notation may be of use in some applications, for others a much richer capacity notation is required. It is often the case that a limit needs to be placed on the cardinality of multisets over (elements of partitions of) a place's colour set. For example, we would like to be able to express constraints like $|M(s)| \leq n$. This represents the *total* capacity of a place (i.e. the sum of all tokens in the place) which could be a resource bound, e.g. 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 (by itself) is inadequate.

10.1 Protocol Example

As a further illustration, consider the following example encountered while modelling the M-Access Service of the Cambridge Fast Ring [39, 14]. (We shall return to this example in more detail in section 12.)

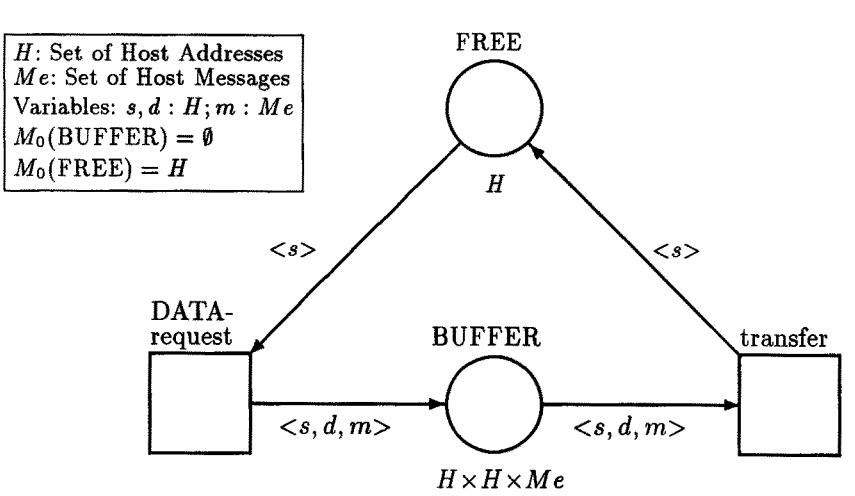

Fig. 5. LAN Access Buffer

A network interconnects a set of computers, known as hosts. Hosts can send messages to each other via the network. Each host has an address. When a host wishes to send a message it appends its own address (source address) and that of the destination (destination address) to the message to form a packet. Each host accesses the network via a one packet buffer. When this buffer is free, the host can store a new packet in the buffer. When network resources are available the packet is transferred into the network for routing and delivery, thus freeing-up the buffer for a new packet.

A P-Graph of the access procedure (for all hosts) is shown in figure 5. Place BUFFER represents the set of access buffers, one for each host. Place FREE indicates which buffers are available. (Initially all the buffers are free: M_0(BUFFER) $= \emptyset$.) If this place contains a token with the value of host a's address, then host a's buffer is free and can be used for the next packet host a wishes to submit to the network (transition DATA-request occurs). Host a's buffer will not be free

again until the network accepts the packet (transition transfer occurs). Hence place FREE provides the control necessary to ensure a capacity limit of one buffer per host.

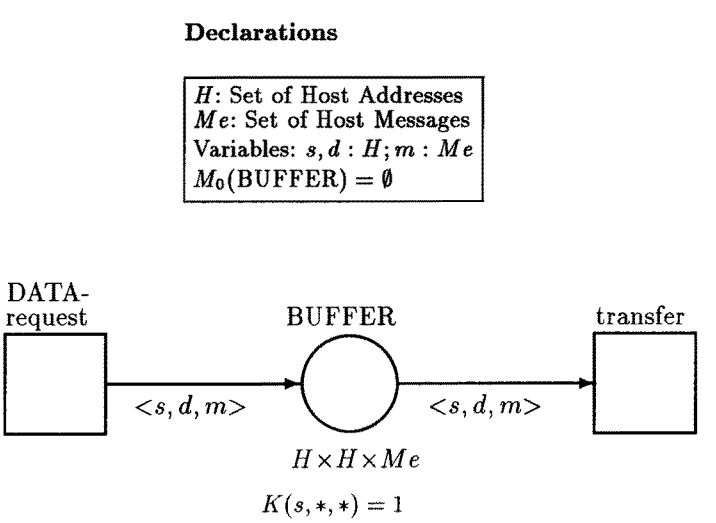

Fig. 6. LAN Access Buffer illustrating extended capacity notation

When visualization of this control mechanism is not required, we would like to replace the capacity control for place BUFFER by an extended capacity inscription. This is shown in figure 6, where place BUFFER is inscribed by '$K(s,*,*) = 1$'. We may interpret this to mean that there is one buffer available for each host, i.e. that the sum of tokens over the set of (destination) host addresses, H, and messages, Me, in place BUFFER for a particular value of s, is at most one. The $*$'s indicate sums over the domains of the variables they replace. This may be viewed as an extended capacity condition on the marking of the place concerned: for all markings of BUFFER, and for each host, $j \in H$, $\sum_{h \in H} \sum_{g \in Me} M(\text{BUFFER}; j, h, g) \leq 1$.

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 $k \in N^+$. The syntax of $a_i, i \in In = \{1, 2, \ldots, n\}$ is given by the production rule $a_i ::= < v_i > \mid *$ where angular brackets denote non-terminals and $v_i : G_i$. (The syntax for variables is left open, but it would normally be a finite string of alphanumeric characters.)

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

10.2 Interpretation of Extended Capacity Notation

When there are no stars present in the argument of $K(a_1, \ldots, a_n)$, it has the same meaning as K, defined in the previous section. This notation is therefore redundant and would not be used.

We now consider two cases:

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

Case A

For case A, for each place, s, inscribed by $K(a_1, \ldots, a_n) = k$, we remove the inscription and replace it by a *projected* complementary place, $\overline{s}$, 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, $\overline{s}$, with colour set $C(\overline{s}) = G_i \times \ldots \times G_j$ derived from the types the variables of the above tuple, where G_i is the type of the variable v_i and so forth.
3. 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'), $t' \in T$.
4. Annotate each arc by the tuple $< v_i, \ldots, v_j >$.
5. The initial Marking, $M_0(\overline{s})$ is related to $M_0(s)$ and the value of k in the following way. For every $g_i \in G_i \ldots g_j \in G_j$

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

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

Case B

For case B, for each place, s, inscribed by $K(*, \ldots, *) = k$, we remove the inscription and replace it by a *completely-projected* complementary place, $\overline{s}$, (a P/T-net 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 (i.e. $K^* = K(*, \ldots, *)$) as in Numerical Petri Nets [69].

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.

11 Demon Game

In this section a small example is presented that was used as a test case for formal methods developed in ISO and CCITT with application to Open Systems Interconnection protocols and services. The example is called the Demon (Daemon) Game [66].

The following provides a description of the demon game which is slightly more abstract than the narrative description in [66] 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!

11.1 Narrative Description

Consider a system in which there lurks a demon which generates *bumps*; the number of bumps not being directly observable from outside the system. 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 by one for a successful guess and decremented by one 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.

11.2 MAN Specification

The Demon Game can be specified using a (strongly-typed) many-sorted algebraic net (MAN) [9]. A MAN is a CP-Graph where all transition conditions are true and the multiplicities of terms in arc expressions are natural numbers rather than natural number terms. (The CP-Graph of the train was a MAN.) 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.

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 the state of the demon's bumps and requesting the cumulative score) and the bottom transition specifies the bumping of the demon.

The convention of double-headed arcs described in section 6.4 is used. That is, 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. For example, see $f1$ = (Scores,Request) and $f2$ = (Request,Scores) in figure 7, where $A(f1) = A(f2) = <i,s>$.

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 $\overline{\ }$: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) = ∅
M_0(Players) = ∅
M_0(Bumps) = even

Graph

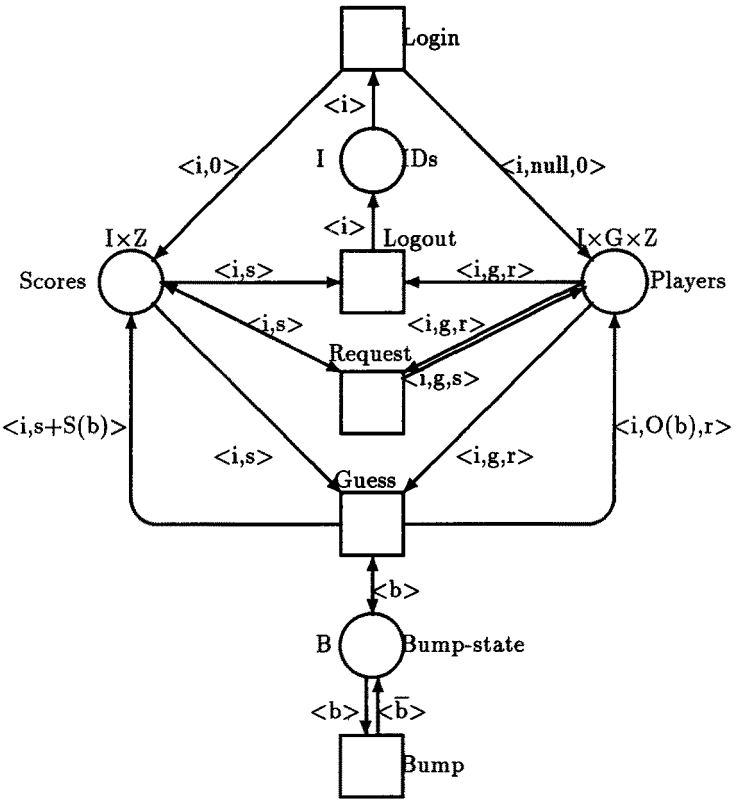

Fig. 7. MAN Specification of Demon Game

Information about players is represented as a triple 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*. 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 is an even number (zero) of bumps, represented by place *Bumps* being 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.

11.3 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. 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 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 $A(\textit{Bump-state}, \textit{Bump}) = n$ <b> and $A(\textit{Bump}, \textit{Bump-state}) = n$ <$\overline{\text{b}}$> 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 could easily be glossed over with a technique based on interleaving semantics. With an interleaving model, the im-

plementer 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 overlooked.

12 Cambridge Fast Ring Service Specification

12.1 Description of the CFR

The Cambridge Fast Ring (CFR) networking system [39] consists of a cluster of CFRs interconnected by bridges. The CFR is a slotted ring designed during the early 1980s to provide a raw 100 MBit/s transmission speed and to substantially increase the bandwidth between point-to-point users. Hardware for the stations, the monitor and bridges for the Cambridge Fast Ring have been fabricated in VLSI. The hardware implements the low level protocols between the various distributed components.

An initial draft of the protocol architectures for the CFR was compiled in [24], where it is shown that different architectures can co-exist above the basic service provided by the CFR hardware. This service is known as the M-Access Service and has been defined in [23]. The protocol architecture is shown in figure 8. The lower two layers correspond to the CFR hardware. On the left side is the lowest layer of the Unison architecture that is supported by the CFR, the Unison Data Link Layer. On the right side are the lower layers of an architecture that can support the IEEE 802 and Open Systems Interconnection protocols. The M-Segment layer bridges the gap between the standard Media Access Control (MAC) Service of IEEE 802 and the CFR's M-Access Service. Thus M-Segment and M-Access together provide the MAC service over which the Logical Link Control protocol can be implemented.

12.2 CFR M-Access Service

Terminology and Features We shall use the term *packet* to refer to a CFR packet as defined in [39]. The CFR packet includes a Cyclic Redundancy Check (CRC) to detect transmission errors. The term M-Access Service Data Unit (M-SDU) will be used to describe data that is transparently exchanged between users of the M-Access Service.

A draft description of the M-Access Service is given in [23]. The main concern of the M-Access Service is to transfer messages between hosts connected to the CFR. In this paper we shall only consider a single CFR, whereas [10] considers ring clusters.

Service Primitives In the M-Access Service, the M-DATA request and indication service primitives are defined as usual for data transfer, and it is also useful to define an M-TOG indication service primitive. The data primitives and their associated parameters are represented as follows:

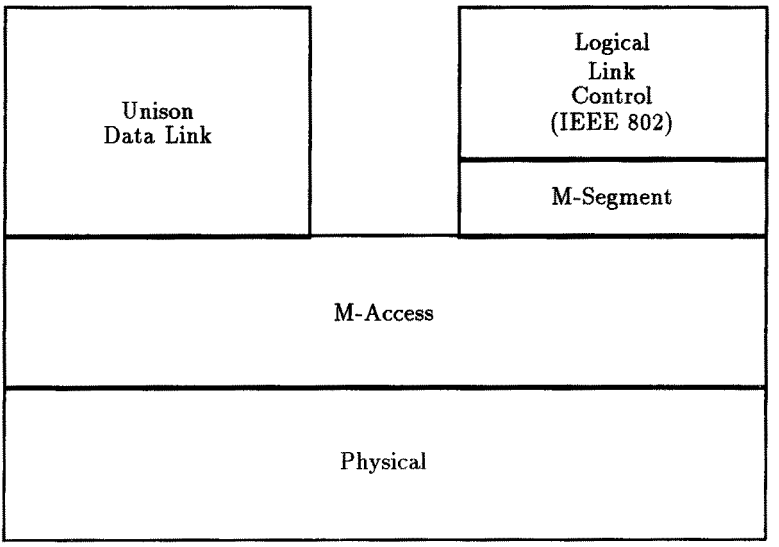

Fig. 8. Lower Layer Protocol Architectures for the CFR

M-DATA request(source-CFR-address, destination-CFR-address, M-data)
M-DATA indication(source-CFR-address, destination-CFR-address, M-data)

The parameters of corresponding M-DATA request and indication primitives have the same values.

The CFR hardware has the capability of telling a user of the M-Access Service that a transmission of a packet has not succeeded. This signal is known as 'Thrown-on-Ground' or TOG for short. This is expressed as a parameterless primitive, M-TOG indication, indicating that the current packet is considered lost by the service provider.

12.3 P-Graph Specification of CFR M-Access Service

Because CFR stations are built from identical chips, the sending (and receiving) operations in each of the stations are the same. We therefore only need to model a generic sender communicating over the ring with a generic receiver, each parameterised by the station address.

We shall consider the following characteristics of a single CFR:

- Arbitrary number of stations
- Point-to-point and broadcast modes
- Single transmit buffer and single receive buffer for each station
- Sequence of M-SDUs preserved per source-destination flow

- Single broadcast by each station (only one broadcast per station is allowed at any one time due to the single transmit buffer)
- Arbitrary loss of M-SDUs
- Three modes of duplication:
  1. Arbitrary duplication in both point-to-point and broadcast mode;
  2. No duplication in broadcast mode, but arbitrary duplication for point-to-point operation; and
  3. No duplication

The duplication case 2 is close to the operation of the CFR, although duplication for point-to-point is very rare and limited. A limit to the amount of duplication can be incorporated into the specification in a straightforward way if desired. (It requires an extra place to store the duplication limit for each station.)

We shall consider the three modes of duplication in separate specifications. The left side of each diagram represents the transmitter and the right side the receiver. The transitions in the centre represent various ways in which the CFR can operate. We represent a set of transmit buffers, one for each station, by the single place 'Transmit-buffers' and we record the stations that have empty buffers in place 'Empty-transmit-buffers'. A similar situation exists for the receive buffers. We also include explicitly which stations are acceptable sources of M-SDUs for each of the destinations, by storing them in place 'Acceptable-sources'.

Arbitrary Duplication The single CFR M-Access service with arbitrary duplication in both broadcast and point-to-point modes is specified in figure 9. The initial state of the service is specified by the initial marking of the net. Each station connected to the CFR will have an empty buffer for transmitting and one for receiving. The presence of an empty transmit buffer is represented by storing the station's source address in place 'Empty-transmit-buffers' and the presence of an empty receive buffer is similarly represented by storing the station's address in place 'Empty-receive-buffers'. The monitor is always attached to a ring, but cannot transmit normal packets [39]. It can, however, receive normal packets. This is why it is excluded from the set of source addresses, but included in the set of destination addresses. Since the monitor is always attached to an operational ring, its address must be included in the initial marking of place 'Empty-receive-buffers'. The addresses of the source stations acceptable to each destination are stored in place 'Acceptable-sources' as source-destination pairs. Initially all the transmit and receive buffers are empty and hence places 'Transmit-buffers' and 'Receive-buffers' are empty.

With this initial state, any number of stations may request the sending of an M-SDU. This is achieved by firing transition 'M-DATA request'. A token representing an M-SDU, is placed in 'Transmit-buffers' and the token representing that the buffer was empty for that station is removed from 'Empty-transmit-buffers'. If the M-SDU is not broadcast, then one of three events may occur:

1. The M-SDU is successfully transferred to the chosen destination. This may only occur if the source is acceptable to the destination. This is achieved by

Declarations

Sets: S,D,D′,M
Constants: b,m ∈ D
Variables: s:S, d:D, d':D′, m:M
Initial Marking
M_0(Transmit-buffers)=M_0(Receive-buffers)=∅
M_0(Empty-transmit-buffers)⊆ S
M_0(Empty-receive-buffers)=M_0(Empty-transmit-buffers)∪{m}
M_0(Acceptable-sources)⊆ S×D′

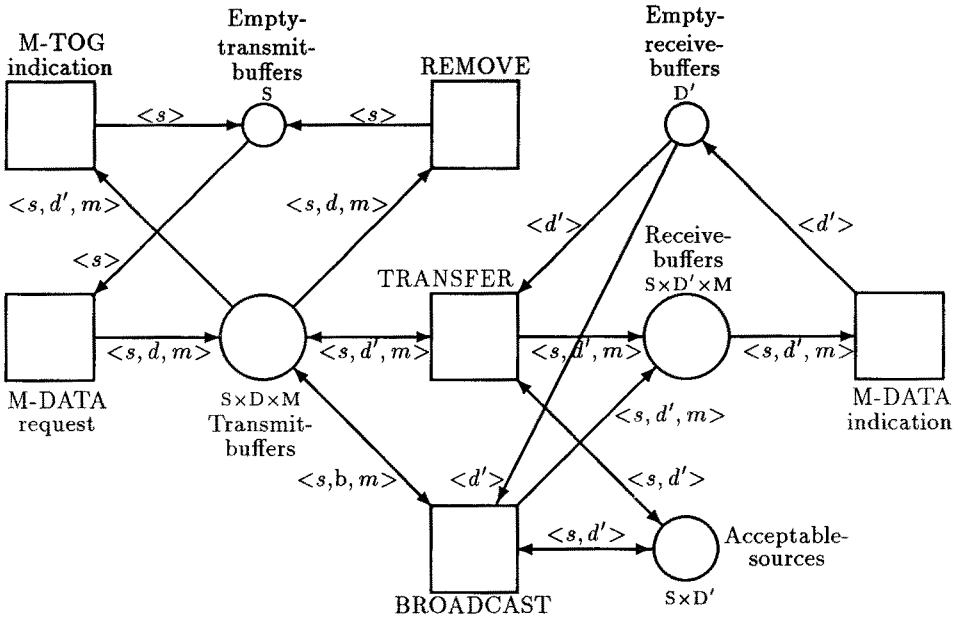

Fig. 9. Single CFR M-Access Service: Duplication

firing transition 'TRANSFER'. A copy of the M-SDU is maintained in the transmit buffer while it is transferred to the destination's receive buffer which is removed from the list of empty buffers. The M-SDU may then be removed from the transmit-buffer which would then be marked free by the occurrence of transition 'REMOVE'. Concurrently, an M-DATA indication may occur at the destination, with the M-SDU being removed from the receive-buffer which is marked free. This may be considered as the normal operation of the service. Duplication may occur by firing 'TRANSFER' twice (or more) before the occurrence of the 'REMOVE' transition.

2. The M-SDU is refused by the destination and this is reported to the source user. This is achieved by firing 'M-TOG indication', which removes the M-SDU from the transmit buffer and marks it free.

3. The M-SDU is lost. The CFR transmitter hardware falsely believes that the M-SDU has been accepted by the destination, due to a CRC error in the return path. This is represented by the firing of the 'REMOVE' transition. The M-SDU is discarded and the transmit buffer marked free.

For broadcast M-SDUs, there are two possibilities.

1. The M-SDU is lost by firing transition 'REMOVE'.
2. The M-SDU is broadcast one at a time to any of the allowable destinations by repetitively firing transition 'BROADCAST'. When this transition occurs, a copy of the M-SDU is retained in the transmit buffer, the M-SDU is transferred to an accepting destination and its buffer is removed from the empty list. An M-DATA indication may then occur with the consequent release of the receive buffer. This then allows duplication of the broadcast M-SDU, as the 'BROADCAST' may occur again for the same destination. It may also occur again for any other destination. The broadcast ends with the occurrence of the 'REMOVE' transition, which empties the transmit buffer.

No Duplication in Broadcast mode In order to avoid duplication in broadcast mode we must keep a record of the stations to which we have broadcast. In a single CFR this is relatively easy as no simultaneous transmissions by a particular station are allowed due to single buffering. For each station, only a single point-to-point or broadcast transmission is possible and this must have completed (successfully or not) before the next transmission can occur. This allows us to use the list of allowed source-destination pairs stored in 'Acceptable-sources' to determine which station has received a broadcast M-SDU.

The specification is shown in figure 10. It is the same as figure 9, except that

- The places 'Empty-transmit-buffers' and 'Empty-receive-buffers' and their associated arcs and initial markings have been removed and replaced by the extended capacity notation defined in section 10.
- The place, 'Broadcast-destinations', (with initial null marking), the transition, 'REMOVE-B', and associated arcs and inscriptions have been added.
- The 'REMOVE' transition has been renamed 'REMOVE-P'. 'REMOVE-P' may only remove point-to-point M-SDUs as the tuple annotating the arc now contains the variable d':D$'$, instead of d:D. 'REMOVE-B' may only remove broadcast M-SDUs.
- The *return* arc from transition 'BROADCAST' to 'Acceptable-sources' has been deleted.
- Both 'Acceptable-sources' and 'Broadcast-destinations' have been annotated with a capacity '$K = 1$'.

The specification is the same as figure 9 for point-to-point operation. As before, broadcasting may occur when a broadcast M-SDU is in a transmit buffer and there is a destination (with a free buffer) that will accept M-SDUs from the source of the broadcast. When 'BROADCAST' fires, the destination is removed from the set of accepting destinations stored in 'Acceptable-sources', and

Declarations

Sets: S,D,D',M
Constants: b $\in$ D
Variables: s:S, d:D, d':D', m:M
Initial Marking
M_0(Transmit-buffers) = M_0(Receive-buffers) = M_0(Broadcast-destinations) = $\emptyset$
M_0(Acceptable-sources) $\subseteq$ S×D'

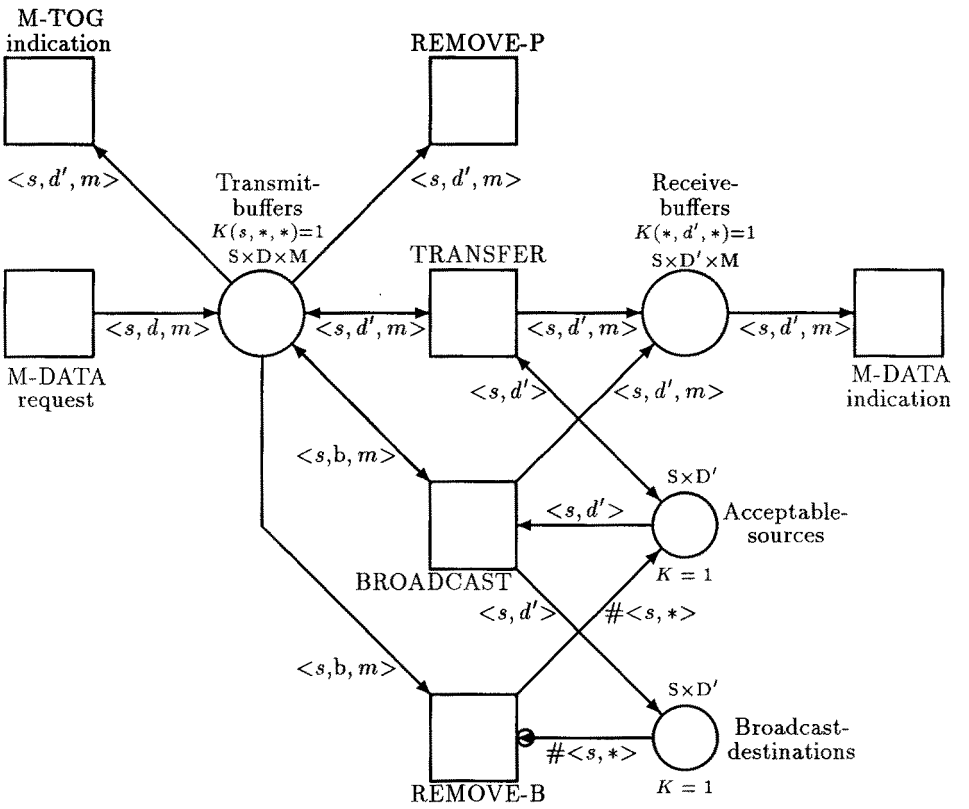

Fig. 10. Single CFR M-Access Service: No Duplication for Broadcast M-SDUs

is written to a set of destinations that have received a broadcast M-SDU. The set is stored in place 'Broadcast-destinations'. The broadcast will continue until either the set of accepting destinations is exhausted (there will no longer be a source-destination pair in 'Acceptable-sources' with the broadcast source address - hence 'BROADCAST' will not be enabled (for this source address) and the only remaining possibility for the broadcast M-SDU is that it is removed from the transmit buffer by firing 'REMOVE-B') or the M-SDU is removed by firing 'REMOVE-B'.

'REMOVE-B' is enabled by a broadcast M-SDU being in a transmit buffer. When it fires, the following actions occur atomically:

1. A particular source's broadcast M-SDU is removed from the transmit buffer.
2. All destinations that have successfully received the source's broadcast are purged from 'Broadcast-destinations' and returned to 'Acceptable-sources'.

Any number of stations can be active at the same time and they operate independently except for contention (conflict) for destination receive-buffers.

To be able to purge 'Broadcast-destinations' of the required destinations for the associated source, we have used the notation for purging a member of a partition developed in chapter 8 of [10], which also provides an interpretation in terms of P-nets. In essence, both an inhibitor and normal arc are required with the same inscription. This is represented by overlaying the normal and inhibitor arcs and is referred to as a reset arc. Here we shall not be concerned with the formalities. The notation $\#<s,*>$ (really a variable ranging over a set of multisets) on the reset arc can be interpreted as follows: for a particular value of s (say a), the demand on 'Broadcast-destinations' is always satisfied and when 'REMOVE-B' occurs, all tokens with the value a in the first position of the pair are removed from 'Broadcast-destinations'. The notation $\#<s,*>$ on the output arc from 'REMOVE-B' to 'Acceptable-sources', implies that these tokens are added to the marking of 'Acceptable-sources'.

The addition of the capacity restriction ($K = 1$) to places 'Acceptable-sources' and 'Broadcast-destinations' better reflects the intent of the specification and guarantees that the P-net to CP-net transformations can be applied.

No Duplication The single CFR M-Access service with no duplication can be derived from figure 10 by deleting the (return) arc from 'TRANSFER' to 'Transmit-buffers'. When 'TRANSFER' fires, the M-SDU is removed from the transmit buffer and hence no duplication can occur. It would also be useful to rename the REMOVE-P transition to LOSE-P as it would only model loss.

We have made the assumption that as far as users of the M-Access Service are concerned, the operation of delivery of an M-SDU to a receiving station and the freeing of the transmit buffer can be considered atomic for point-to-point operation.

13 Analysis and Computer Aided Tools

The analysis of P-Graphs is a large topic and space precludes going into the details here. However, the paper would not be complete without a discussion of the analysis capabilities of P-Graphs. One of the design goals in the development of P-Graphs was to ensure that the analysis capabilities of the high-level nets being developed in main stream net theory [20] could be applied to P-Graphs. This paper shows how P-Graphs can be mapped to P-nets. So long as finite resources are required when the inhibitor is used, P-nets can be transformed

to CP-nets as shown in [8]. This allows the analysis methods of CP-nets [46] to be used for analysing P-nets. These analysis techniques include reachability, invariants and reductions, model checking of the reachability graph, and also analysis via the *skeleton* P/T-system as discussed in [67]. It may also be the case that some of these techniques may be able to be applied directly to the P-net. Direct analysis of subclasses of the P-Graph is possible using invariants as shown in [60] and applied in [32].

The strong relationship with CP-nets will allow the use of automated tools being developed for CP-nets (editors, simulators, analysers) [57, 48], other reachability analysis tools [68] and a compiler [41]. This will form the basis of a powerful systems engineering environment for the development and maintenance of protocols.

14 Conclusions and Future Work

P-Graphs have been motivated by the desire to model protocol mechanisms elegantly. They have been defined as inhibitor nets that include a many-sorted signature and a corresponding algebra. Variables can now be appropriately typed and functions are total. This paper brings together the work of a number of researchers [36, 61, 67, 43]. The contribution of the paper is not only in this synthesis, but also in the development of the inhibitor and capacity extensions, the formalisation of the P-Graph at a concrete level to include natural number terms to be used as multiplicities when defining multisets of terms for arc inscriptions, and the application of P-Graphs to the specification of protocol services and other similar systems. An extended capacity notation has been developed and illustrated in the specification of the Cambridge Fast Ring M-Access Service. The meaning of this notation is provided in terms of the P-Graph. A notation for purging places of tokens has been illustrated with the CFR M-Access Service. It uses a combination of an inhibitor arc and normal arc to form a reset arc [10]. It is further shown how P-Graphs can be mapped to P-nets, which (in most practical circumstances) can be transformed to CP-nets to take advantage of their analysis techniques and automated tools.

A hierarchy of high-level nets can be defined, by restricting the structure of the P-Graph, to include CP-Graphs, many-sorted Algebraic nets and Place Transition nets. It thus provides a basis for the comparison of these classes. An area of further work is in determining the relative merits of these classes for protocol and service specification and for their analysis.

In [9] it is shown how abstract P-Graphs can be defined at the syntactic level, in a similar way to Vautherin [67]. The abstract P-Graph provides a vehicle for the specification of classes of systems and the possibility of their analysis via a single member of the class as has been demonstrated by Vautherin [67]. This opens up possibilities which need to be investigated in the protocol domain.

The aim of the work is that P-Graphs will be able to be applied to large applications. This will require better ways of structuring and refining specifications, than used in the past [12, 3]. At a fundamental level, the development of

refinement morphisms for P/T-systems [56] shows promise, but these ideas will need to be incorporated into high-level nets. The development of hierarchical CP-nets [40] are of practical interest as they have been used to organise the specifications of a significant number of protocol applications [47]. This area has not been addressed in this paper and it will require a significant effort in the future.

Finally, there is a significant international effort going into the development of an international standard for high-level nets [17]. The standard, ISO/IEC CD15909, is currently at Committee Draft status, and is built on many of the ideas presented in this paper. Elaboration of the standard to include inhibitor and other extensions is currently being considered. It is hoped that the standard will reach international standard status in a year or two. This will then provide a consistent set of concepts to allow industry to manadate the standard for contractual work, and to allow the development of a standard means of transferring high-level nets between tools, while (hopefully) maintaining their semantics.

Acknowledgements

This work has benefited from valuable discussions with numerous colleagues, including Kurt Jensen, Geoff Wheeler, Bernd Baumgarten, Wolfgang Reisig, Ekkart Kindler and many others involved in the international standardisation work of ISO/IEC JTC1/SC7/WG11 project 7.19.3 Petri net techniques.

References

1. E. Battiston, F. De Cindio, and G. Mauri. OBJSA nets: a class of high-level nets having objects as domains. In G. Rozenberg, editor, *Advances in Petri Nets 1988, Lecture Notes in Computer Science 340*, pages 20 – 43. Springer-Verlag, Berlin, 1988.
2. Mirion Y. Bearman. Formal specification of the Open Systems Interconnection Transport Protocol class 2 using NPNs. Technical Report 25, CSIRONET, 1986.
3. Mirion Y. Bearman, Michael C. Wilbur-Ham, and Jonathan Billington. Some results of verifying the OSI class 0 Transport Protocol. In *ICCC*, pages 597–602, Sydney, November 1984.
4. Mirion Y. Bearman, Michael C. Wilbur-Ham, and Jonathan Billington. Analysis of Open Systems Interconnection Transport Protocol standard. *Electronics Letters*, 21(15):659–661, July 1985.
5. M.Y. Bearman, K.R. Parker, and R.A. Berger. A formal specification of the OSI Class 3 Transport Protocol using NPNs. Technical Report TR-ED-88-02, CSIRO Division of Information Technology, 1988.
6. Gerard Berthelot and Richard Terrat. Petri nets theory for the correctness of protocols. *IEEE Transactions on Communications*, COM-30(12):2497–2505, Dec 1982.
7. B. Berthomieu, N. Choquet, C. Colin, B. Loyer, J.M. Martin, and A. Mauboussin. Abstract Data Nets: combining Petri nets and abstract data types for high level specifications of distributed systems. In *Proceedings of the Seventh European*

Workshop on Application and Theory of Petri Nets, pages 25 – 48, Oxford, England, 30 June - 2 July 1986.

8. J. Billington. Extensions to Coloured Petri Nets. In *Proceedings of the Third International Workshop on Petri Nets and Performance Models*, pages 61–70, Kyoto, Japan, 11-13 December 1989. IEEE CS Press.
9. J. Billington. Many-sorted high-level nets. In *Proceedings of the Third International Workshop on Petri Nets and Performance Models*, pages 166–179, Kyoto, Japan, 11-13 December 1989. IEEE CS Press.
10. J. Billington. *Extensions to Coloured Petri Nets and their Application to Protocols.* PhD thesis, University of Cambridge, May 1990.
11. J. Billington, G. Wheeler, and M.C. 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.
12. Jonathan Billington. Abstract specification of the ISO Transport Service Definition using labelled Numerical Petri Nets. In Harry Rudin and Colin H. West, editors, *Protocol Specification, Testing and Verification, III*, pages 173–185, Amsterdam, 1983. Elsevier Science Publishers B.V.
13. Jonathan Billington. Extending coloured Petri nets. Technical Report 148, University of Cambridge Computer Laboratory, New Museums Site, Pembroke Street, Cambridge, England, October 1988.
14. Jonathan Billington. A High-Level Petri net specification of the Cambridge Fast Ring M-Access service. In *Proceedings of the BCS-FACS Workshop on Specification and Verification of Concurrent Systems*, University of Stirling, Stirling, Scotland, July 1988. Earlier version published as University of Cambridge Computer Laboratory Technical Report No. 121, December 1987.
15. Jonathan Billington, editor, *Application of Petri Nets to Communication Protocols*, Advanced Practical Tutorial Notes, 15th Int. Conference on Application and Theory of Petri Nets, Zaragoza, Spain, 20 June 1994.
16. Billington J. and Diaz M., editors, *Petri Nets applied to Protocols*, Proceedings, First PetriNets95 Protocol Workshop, Turin, Italy, 26 June 1995.
17. Billington J., Development of an International Standard for High-level Petri Nets, In *Proc. 3rd IEEE International Software Engineering Standards Symposium and Forum* (ISESS'97), Walnut Creek, California, USA, 1-6 June 1997, ISBN: 0-8186-7837-2, pp.155-162.
18. Billington, J., Farrington M., and Du B.B., Modelling and Analysis of Multi-Agent Communication Protocols, to appear in *Proc. 3rd Biennial Engineering Mathematics and Applications Conference* (EMAC'98), Adelaide, 13-16 July 1998.
19. Billington J., Diaz M. and Rozenberg G. (Eds) *Application of Petri Nets to Communication Networks*, Special Issue of Advances in Petri Nets, Lecture Notes in Computer Science, Springer-Verlag, in preparation.
20. 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.
21. H. J. Burkhardt and Hans Eckert. Transport service, formal specification; network service, formal specification; transport protocol class 2, formal specification. early drafts, 1984.
22. H. J. Burkhardt, Hans Eckert, and Rainer Prinoth. Modelling of OSI-communication services and protocols using Predicate/Transition Nets. In

Y. Yemini, R. Strom, and S. Yemini, editors, *Protocol Specification, Testing and Verification, IV*, pages 165 – 192, Amsterdam, 1985. Elsevier Science Publishers B.V.

23. A. M. Chambers. CFR M-Access service definition. Draft Unison Project Document, Ref: UA008, November 1986.
24. A. M. Chambers and D. L. Tennenhouse. Communications architectures for the Cambridge Fast Ring. Draft Unison Project Document, Ref: UA004, October 1986.
25. J. P. Courtiat, J. M. Ayache, and B. Algayres. Petri nets are good for protocols. In *ACM SIGCOMM '84 Symposium, Communications Architectures and Protocols*, pages 66–74, Montreal, Canada, June 1984.
26. B. Cousin, J.M. Couvreur, C. Dutheillet, and P. Estraillier. Validation of a protocol managing a multi-token ring architecture. In *Proceedings of the IX European Workshop on Applications and Theory of Petri Nets*, Venice, Italy, 22-24 June 1988. Fourth paper in volume II.
27. Michel Diaz. Modeling and analysis of communication and co-operation protocols using Petri net based models. *Computer Networks*, 6:419–441, 1982.
28. Michel Diaz. Petri net based models in the specification and verification of protocols. In W. Brauer, W. Reisig, and G. Rozenberg, editors, *Petri Nets: Applications and Relationships to Other Models of Concurrency*, pages 135 – 170. Springer-Verlag, Berlin, 1987. Lecture Notes in Computer Science, Vol. 255.
29. Diaz M., Little T., and Senac P. (Eds), *Multimedia and Concurrency*, Workshop Proceedings, Toulouse, France, 24 June 1997.
30. S. Drees, D. Gomm, H. Plünnecke, W. Reisig, and R. Walter. Bibliography of net theory. In G. Rozenberg, editor, *Advances in Petri Nets 1987*, pages 309 – 451. Springer-Verlag, Berlin, April 1987. Latest update is 'Bibliography of Petri Nets 1988', Arbeitspapiere der GMD, No. 315, June 1988, which contains over 2,500 entries.
31. 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.
32. Greg Findlow and Jonathan Billington. High-level Nets for Dynamic Dining Philosophers Systems. In *Proceedings of the International Workshop on Semantics for Concurrency, Leicester, UK, 23-25 July 1990*, London, 1990. Springer-Verlag.
33. R.J. Fone. An analysis of a Numerical Petri Net description of ISDN call control procedures. Research Laboratories Report 7915, Telecom Australia, August 1988.
34. R.J. Fone. A Numerical Petri Net description of ISDN call control procedures. Research Laboratories Report 7914, Telecom Australia, August 1988.
35. H. J. Genrich and K. Lautenbach. The analysis of distributed systems by means of Predicate/Transition-nets. *Lecture Notes in Computer Science*, 70:123–146, 1979.
36. 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, Berlin, February 1987. Springer-Verlag. Lecture Notes in Computer Science, Volume 254.
37. Hartmann J. Genrich and Kurt Lautenbach. System modelling with high-level Petri nets. *Theoretical Computer Science*, 13:109–136, 1981.
38. C. Girault, C. Chatelain, and S. Haddad. Specification and properties of a cache coherence protocol model. In G. Rozenberg, editor, *Advances in Petri Nets 1987*, pages 1 – 20. Springer-Verlag, Berlin, April 1987. Lecture Notes in Computer Science, Vol. 266.

39. A. Hopper and R.M. Needham. The Cambridge Fast Ring Networking System. *IEEE Transactions on Computers*, 37(10):1214 – 1223, October 1988. Earlier version published as University of Cambridge Computer Laboratory Technical Report, No. 90, June 1986.
40. P. Huber, K. Jensen, and R.M. Shapiro. Hierarchies in Coloured Petri Nets. In *Proceedings of the 10th International Conference on Application and Theory of Petri Nets*, pages 192 – 209, Bonn, West Germany, June 1989.
41. G.C. Illing. Automatic Petri-net based Protocol Implementation. In *IREECON International*, pages 358 – 361, Melbourne, September 1989.
42. Kurt Jensen. Coloured Petri Nets and the invariant-method. *Theoretical Computer Science*, 14:317–336, 1981.
43. 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.
44. Kurt Jensen. Private communication, July 1988.
45. Kurt Jensen. *Coloured Petri Nets: Basic Concepts, Analysis Methods and Practical Use, Volume 1: Basic Concepts*, EATCS Monographs on Theoretical Computer Science. Springer-Verlag, Berlin, 1992.
46. Kurt Jensen. *Coloured Petri Nets: Basic Concepts, Analysis Methods and Practical Use, Volume 2: Analysis Methods*, EATCS Monographs on Theoretical Computer Science. Springer-Verlag, Berlin, 1994.
47. Kurt Jensen. *Coloured Petri Nets: Basic Concepts, Analysis Methods and Practical Use, Volume 3: Practical Use*, EATCS Monographs on Theoretical Computer Science. Springer-Verlag, Berlin, 1997.
48. Jorgensen J.B., and Kristensen L.M., Design/CPN OEOS Graph Manual, University of Aarhus, 1996.
49. W. Jürgensen and S.T. Vuong. Formal specification and validation of ISO Transport Protocol components, using Petri nets. In *ACM SIGCOMM '84 Symposium: Communications Architectures and Protocols*, pages 75–82, Montréal, Canada, June 1984.
50. R.M. Keller. Formal verification of parallel programs. *Communications of the ACM*, 19(7):371 – 384, July 1976.
51. Bernd Krämer. Stepwise construction of non-sequential software systems using a net-based specification language. In G. Rozenberg, editor, *Advances in Petri Nets 1984*, pages 307 – 330. Springer-Verlag, Berlin, 1985. Lecture Notes in Computer Science, Vol. 188.
52. Bernd Krämer. SEGRAS - A Formal and Semigraphical Language combining Petri Nets and Abstract Data Types for the Specification of Distributed Systems. In *Proceedings of the Ninth International Conference on Software Engineering*, pages 116 – 125, Los Alamitos, California, March 1987. CS Press.
53. R.Y.L. Lai. *Formal Specification and Verification of ISO FTAM Protocol.* PhD thesis, Mathematical and Information Sciences, LaTrobe University, Melbourne, Australia, 1989.
54. 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.

55. P.M. Merlin. *A study of the recoverability of computing systems.* PhD thesis, Information and Computer Science Department, UC Irvine, 1974. Tech Report, No. 58.

56. J. Meseguer and U. Montanari. Petri nets are monoids: a new algebraic foundation for net theory. In *Proceedings of the Third Annual Symposium on Logic in Computer Science*, pages 155 – 164, Washington, DC, July 1988. IEEE Comp. Soc. Press.

57. Meta Software. Design/CPN Reference Manual. Meta Software Corporation, 1993.

58. J.D. Noe and G.J. Nutt. Macro *E*-Nets for Representation of Parallel Systems. *IEEE Trans. Comput.*, C-22(8):718 – 727, August 1973.

59. G.J. Nutt. Evaluation nets for computer system performance analysis. In *Proc FJCC, AFIPS*, pages 279 – 286, Montvale, N.J., 1972. AFIPS Press.

60. W. Reisig. Petri Nets and Algebraic Specifications. SFB-Bericht 342/1/90 B, Technische Universität München, Institut für Informatik, München, West Germany, March 1990.

61. 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.

62. Wolfgang Reisig. *Petri Nets, An Introduction*, volume 4 of *EATCS Monographs on Theoretical Computer Science*. Springer-Verlag, Berlin, 1985.

63. 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.

64. Tokmakoff, A. and Billington, J., Service Trading in Mobile Environments, In *Proceedings of the International Conference on Information, Communications & Signal Processing: Trends in Information Systems Engineering and Wireless Multimedia Communications* (ICICS'97), Singapore, 9-12 September 1997, IEEE Singapore Section, Volume 1 of 3, ISBN: 0-7803-3676-3, pp. 417-421.

65. Tokmakoff, A. and Billington, J., Reachability Analysis of the ODP Trader using Equivalence Classes, In *Proceedings of the Software Engineering: Education & Practice Conference* (SE:E&P'98), Dunedin, New Zealand, 26-29 January, 1998.

66. Turner K.J., editor. Using Formal Description Techniques - An Introduction to Estelle, LOTOS and SDL. John Wiley, New York, January 1993.

67. 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.

68. G. Wheeler, A. Valmari, and J. Billington. Baby TORAS eats Philosophers but thinks about Solitaire. In *Proceedings of the Fifth Australian Software Engineering Conference ASWEC'90*, pages 283 – 288, Sydney, Australia, 23-25 May 1990.

69. G. R. Wheeler. Numerical Petri Nets - a definition. Research Laboratories Report 7780, Telecom Australia, May 1985.

Appendix

A Sets, Multisets and Vectors

A.1 Sets

We 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. (We require multisets to have finite support.) There are times when we shall consider a set as a special case of a multiset, where the multiplicities of each of the basis elements is unity.

The set of multisets over A is denoted by μA (i.e. $\mu A = [A \longrightarrow N]$). 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 $\mu_\infty^+ A = [A \longrightarrow N_\infty^+]$.

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

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

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: $\forall a \in A$, $\emptyset(a) = 0$.

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.

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

Equality and Comparison Two multisets, $B1, B2 \in \mu A$, are equal, $B1 = B2$, *iff* $\forall a \in A$, $B1(a) = B2(a)$, and $B1$ is less than or equal to (or contained in) $B2$, $B1 \leq B2$ *iff* $\forall a \in A$, $B1(a) \leq B2(a)$.

Operations We define four binary operations on multisets, $B1, B2 \in \mu A$, known as union, intersection, addition and subtraction, as follows:

$B = B1 \cup B2$ *iff* $\forall a \in A\ B(a) = max(B1(a), B2(a))$

$B = B1 \cap B2$ *iff* $\forall a \in A\ B(a) = min(B1(a), B2(a))$

$B = B1 + B2$ *iff* $\forall a \in A\ B(a) = B1(a) + B2(a)$

$B = B1 - B2$ *iff* $\forall a \in A\ (B1(a) \geq B2(a)) \wedge (B(a) = B1(a) - B2(a))$

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

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

Adding ∞ and Subtracting from ∞ For all $n \in N$, $n + \infty = \infty + n = \infty$. For all $n \in N$, $\infty - n = \infty$.

Multiplication by ∞ For all $n \in N_\infty^+$, $\infty \times n = n \times \infty = \infty$ but $\infty \times 0 = 0 \times \infty = 0$.

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 (i.e. $\nu A = [A \longrightarrow Z]$). 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 $V1, V2 \in \nu A$

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

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

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

Equality and Comparison 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 or equal to $V2$, $V1 \leq V2$, *iff* $\forall a \in A$, $V1(a) \leq V2(a)$.