Abstract Petri Net Notation

Falko Bause   Peter Kemper   Pieter Kritzinger

Forschungsbericht
der Universität Dortmund
1994

Lehrstuhl Informatik IV
Universität Dortmund
D-44221 Dortmund
Germany
e-mail: {bause,kemper}@ls4.informatik.uni-dortmund.de

Pieter Kritzinger is from
Data Network Architecture Laboratory
Department of Computer Science
University of Cape Town
Private Bag, RONDEBOSCH
7700 South Africa
psk@cs.uct.ac.za

Abstract

The need to interchange the description of Petri Net models amongst researchers and users was recognised as long ago as 1988. This document proposes an Abstract Petri Net Notation (APNN) in which various nets can be described using a common notation. The use of the notation in the context of Petri Net software tools is shown and the general requirements for such a notation to be generally acceptable, are suggested. Keywords in the notation are similar to LaTeX-commands in order to support readability. By employing an appropriate style-file a net description can thus be included directly into a LaTeX source document. The notation is given for untimed Place/Transition Nets and Coloured Petri Nets as well as Generalised Stochastic Petri Nets and Hierarchical Petri Nets.

So eine Arbeit wird eigentlich nie fertig,
man muss sie für fertig erklären,
wenn man nach Zeit und Umständen
das Möglichste getan hat.
— J.W. Goethe, Italienische Reise, 1787.

1 Introduction

Petri Nets[17] have become widely accepted as a formalism for describing concurrent systems and an active community of researchers exist in the field. As the theory of Petri Nets developed, so were software tools developed to assist the analysis of Petri Nets[12].

It was thus recognised as long ago as 1988 by Berthelot et. al. [9] that it would be an advantage if Petri Net researchers could share their model descriptions in some common notation or interface and thereby exploit the existing variety of tools and user interfaces amongst themselves. Such an interface would have the functionality illustrated in Fig. 1 where the P_i indicate translations from one to the other of the functions illustrated in that figure.

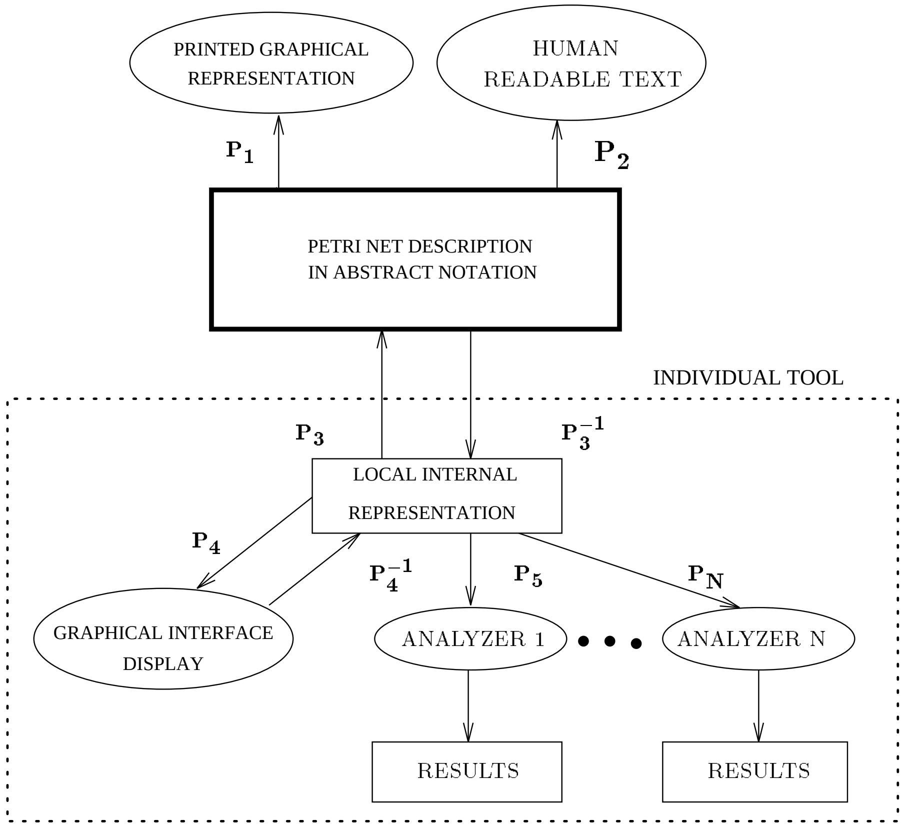

Figure 1: The context of the APNN.

The biggest advantage of such a common notation is the ability to share analysis algorithms and the implementations of these. In this way one can compare the correctness and efficiency of such algorithms and avoid their repeated implementation. The authors have for instance been able to use the Markov chain analyser USENUM[16] developed at one institution for the analysis of Petri Net models created with a Petri Net user interface developed at the other.

Once one accepts the idea of a common interface description various other advantages other than sharing code, become apparent. One could translate the description of a net directly into a printed graphical representation or textual description for documentation.

2 Requirements of the Notation

In general the net notation should satisfy the following criteria for it to be useful to a wide community.

Exchangeable A net description should be in such a format that it can be easily exchanged in electronic form (via, e.g., email or ftp) between different sites. Exchangeability is the major reason for having such a textual notation at all.

Extensible More powerful, i.e. high-level Petri Nets should have a notation such that simpler net classes like Condition/Event Nets appear as special cases of such a formalism in a natural way, i.e. intuitive and concise. Missing information should default to appropriate values. A notation should extend from simple nets to high-level nets in a straightforward way and be flexible enough to allow for future extensions.

Modular and hierarchical These two requirements are strongly related to each other. The notation should be such that Petri Net descriptions in a file can be re-used as part(s) of the description of another, larger net or higher level Petri Net.

Readable Although the graphical description of Petri Nets is one of its major advantages in terms of human readability, it is a drawback for inter-tool communication. Software tools always have a textual description for nets in order to store them in files. Such a file format is in most cases very concise, but not human readable. A useful notation should compromise between conciseness and human readability at least in a way that it is easily transformed into a human readable format. Ideally, this transformation should be performed by public domain software which is generally available.

The general idea is then,

- other persons should be able to "read" the description even without access to the corresponding tools or graphical user interfaces (GUIs).
- it should be easy to include the description into printed documents.

Portability of the graphical representation of the net derived directly from an abstract description is another matter. Graphical layout, where and whether arcs cross and so on, is very much a matter of personal preference. One could specify the position of elements of the net with respect to some predefined origin in the notation as TEX does in fact. This is not simple to do without drawing the net in some form or other on a coordinate system. Although not impossible to do, the authors have decided to avoid the issue for now.

For the notation, the authors decided to choose key-words in the textual description of a net which are similar to LATEX-commands in order to support readability. By employing an appropriate style-file a net description can be directly included into a LATEX-document[1]. Since LATEX is public-domain and widely accepted within the scientific community, we decided to use the descriptive power of LATEX as a formal language to define a textual description of Petri Nets.

The remainder of this report is structured as follows: We distinguish between timed Petri Nets and untimed Petri Nets. In the former category we define Place/Transition Nets and give the grammar describing their abstract notation in Sec. 3.1. This grammar is then extended to Coloured Petri Nets in Sec. 3.2.

Where timed Petri Nets are concerned we start with Generalised Stochastic Petri Nets (GSPN), defined in Sec. 4.1 and Coloured Generalised Stochastic Petri Nets in Sec. 4.2. Hierarchies are considered in Sec. 5. In Sec. 5.7 we define Hierarchical Queueing Petri Nets which are new in that they include both places in the ordinary sense of Petri Nets and a new type of place called queueing places.

The abstract notation of every class of Petri Net is described in Backus Naur Form (BNF) throughout the remainder of this report. Common features of the extended BNF, such as repetition, are not used since it clashes with the terminals used for the description of the notation itself. Non-terminals are written in upper-case, terminals in lower-case.

3 Untimed Petri Nets

Petri Nets were invented in 1962 by Carl Adam Petri[19] and are a formalism for the description of concurrency and synchronisation in systems. Since first described by Petri many variations of his original nets have been defined.

Since the simplest of such nets, known as Condition/Event Nets, are easily described by Place/Transition Nets with a place capacity of one, we shall regard Place/Transition Nets as the basic kind of net to begin with. In this section we define an APNN for Place/Transition Nets and then we extend this notation for Coloured Petri Nets[15].

[1] In this document we have used simple style-file macros, but these can be changed and/or extended easily to improve readability.

In general, Petri Nets comprise the following components:

places, usually drawn by circles, which typically model conditions or objects.

tokens, usually drawn by black dots which represent the specific value of the condition or object.

transitions, usually drawn by rectangles. These model activities which change the values of conditions and objects.

arcs, specifying the interconnection of places and transitions.

Petri Nets are clearly bipartite graphs, i.e., one may only connect a place to a transition or vice versa, but may not connect places to places or transitions to transitions.

3.1 Place/Transition Nets

Definition 1 *A* Place/Transition Net *is a 5-tuple* $PN = (P, T, I^-, I^+, M_0)$ *where*

- $P = \{p_1, \ldots, p_n\}$ *is a finite and non-empty set of places,*
- $T = \{t_1, \ldots, t_m\}$ *is a finite and non-empty set of transitions,*
- $P \cap T = \emptyset$,
- $I^-, I^+ : P \times T \rightarrow \mathbb{N}_0$ *are the backward and forward incidence functions, respectively*
- $M_0 : P \rightarrow \mathbb{N}_0$ *is the initial marking.*

3.1.1 APN Notation for Place/Transition Nets

In the notation which follows the authors have adhered to the requirements described in Sec. 2. Nevertheless, such a notation is obviously not unique and frequently a matter of personal preference. For instance, in the notation, ID is a non-terminal which denotes a unique place, transition or arc identifier. Place and transition identifiers are necessary to specify arcs uniquely. The existence of a unique identifier is useful in that it allows one node description to refer to another one by an entity `\like{ID}`.

The notation also allows for an additional description (which may be empty and need not be unique) associated with an entity `\name` at each node description to allow the model builder to include a name, explanation or comment about the function of the node in the model.

A net description using the notation will obviously be more concise if certain attributes are assumed to have generally accepted default values. We thus define the default value for an arc weight to be 1 and 0 for the initial marking of a place. A place has a capacity of infinity unless explicitly stated otherwise.

In the notation we assume the usual production rules for integers and strings. The keyword `empty` denotes the empty string.

Terminal symbols:

```
\beginnet, \endnet, {, }, \place, \transition, \like, \arc,
\name, \init, \from, \to, \capacity, \weight
```

Non-Terminal symbols:

```
NET, ELEMENT, PLACE, TRANSITION, ARC, ID, NAME, INIT,
WEIGHT, CAP, STRING, INTEGER
```

Productions:

```
NET     ::= \beginnet{ID} ELEMENT \endnet

ELEMENT ::= empty
          | PLACE      ELEMENT
          | TRANSITION ELEMENT
          | ARC        ELEMENT

ID      ::= STRING

PLACE   ::= \place{ID}{ NAME INIT CAP }
          | \place{ID}{ \like{ID} }
NAME    ::= empty | \name{ STRING }
INIT    ::= empty | \init{ INTEGER }
CAP     ::= empty | \capacity{ INTEGER }


TRANSITION ::= \transition{ID}{ NAME }

ARC     ::= \arc{ID}{ \from{ID} \to{ID} WEIGHT }
WEIGHT  ::= empty | \weight{ INTEGER }

Start-symbol: NET
```

We omit the explicit definition of non-terminal and terminal symbols in the remainder of the descriptions of the APNN, since these are distinguished anyway by upper- and lower-case letters respectively.

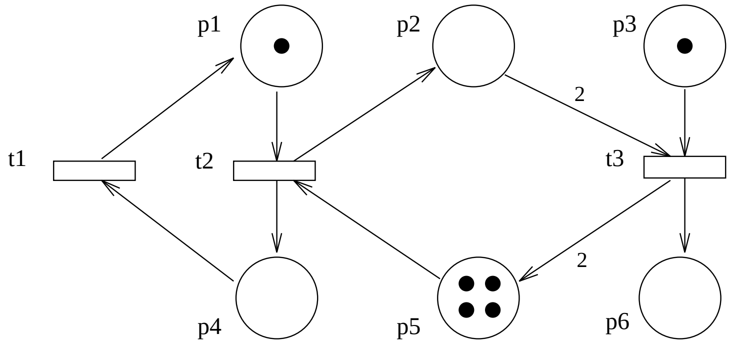

Figure 2: Example Place/Transition Net.

Example 1 *Consider the Place/Transition Net illustrated in Fig. 2. The following is the corresponding APNN description of that net:*

```
\beginnet{examplenet}
\place{p1}{\init{1}}  \place{p2}{}
\place{p3}{\like{p1}} \place{p4}{}
\place{p5}{\init{4}}  \place{p6}{}

\transition{t1}{} \transition{t2}{} \transition{t3}{}

\arc{a1}{\from{p1} \to{t2}} \arc{a2}{\from{t2} \to{p4}}
\arc{a3}{\from{p4} \to{t1}} \arc{a4}{\from{t1} \to{p1}}
\arc{a5}{\from{t2} \to{p2}} \arc{a6}{\from{p5} \to{t2}}
\arc{a7}{\from{p3} \to{t3}} \arc{a8}{\from{t3} \to{p6}}
\arc{a9}{\from{p2} \to{t3}  \weight{2}}
\arc{a10}{\from{t3} \to{p5} \weight{2}}
\endnet
```

As previously explained the authors decided to choose key-words in the textual description of a net which are similar to LaTeX-commands. In this way, the net definition in the previous example can be translated by use of appropriate LaTeX-macros to the following description, intended to be read rather than be processed by machine:

Example 2 *examplenet:*

`Place` *p1* , `marking` {*1*}
`Place` *p2*
`Place` *p3* `is defined as for` *p1*
`Place` *p4*
`Place` *p5* , `marking` {*4*}
`Place` *p6*

```
Transition(s) t1 , t2 , t3

Arc a1 from p1 to t2
Arc a2 from t2 to p4
Arc a3 from p4 to t1
Arc a4 from t1 to p1
Arc a5 from t2 to p2
Arc a6 from p5 to t2
Arc a7 from p3 to t3
Arc a8 from t3 to p6
Arc a9 from p2 to t3 ,weight 2
Arc a10 from t3 to p5 ,weight 2
```

3.2 Coloured Petri Nets

In Coloured Petri Nets, as defined by K. Jensen[15], each token has an associated data value or *colour* which belongs to a specified type of arbitrary complexity. A place can contain various colours which form a set, the so-called *colour set*. A colour function assigns a colour set to each place of the net.

Since it is widely in use, we follow Jensen[15] and use the conventions:

- The type of a variable, v, is denoted by Type(v).
- The type of an expression, *expr*, is denoted by Type(*expr*).
- The set of variables in an expression, *expr*, is denoted by Var(*expr*).
- S_{MS} denotes the set of all multisets over S.

Definition 2 *A Coloured Petri Net is a tuple* $(\Sigma, P, T, A, C, G, E, I)$ *where*

- Σ *is a set of non-empty colour sets,*
- P *is a finite and non-empty set of places,*
- T *is a finite and non-empty set of transitions,*
- A *is a finite set of arcs such that:*
  - $P \cap T = P \cap A = T \cap A = \emptyset$
- C *is a colour function* $P \mapsto \Sigma$
- G *is the guard function from* T *into boolean expressions such that:*
  - $\forall t \in T, \quad Type(G(t)) = \mathbb{B}$ *and* $Type(Var(G(t))) \subseteq \Sigma$
- E *is the arc function from* A *into expressions such that:*

- $\forall a \in A, \quad Type(E(a)) = C(p(a))_{MS}$ *and* $Type(Var(E(a))) \subseteq \Sigma$ *where* $p(a)$ *denotes the place connected by arc* a

- *I is the initialisation function defined from P into closed expressions such that:*
  - $\forall p \in P, \quad Type(I(p)) = C(p)_{MS}$

Programming languages can be employed for the precise definition of expressions and colour sets. In [15], a derivative of Standard ML, so-called CPN ML, is used. In this report we do not distinguish between these two languages explicitly.

3.2.1 APN Notation for Coloured Petri Nets

In order to avoid tedious repetition, only the differences between the notation for Place/Transition Nets and that needed to specify Coloured Petri Nets are given next.

We assume that the non-terminal ML-EXPRESSION is an ML-expression according to [20], which evaluates to the appropriate type, i.e., either integer, boolean or multiset. In fact, since ML-expressions can contain names of arbitrary complex, user-defined ML-functions, such ML-functions can be specified in a separate ML-specification file which can be then referred to in the net description by the APNN `\seeML{FILENAME}` statement. The same holds for ML definitions, which are denoted by the non-terminal ML-DEFINITION.

Productions:

```
ELEMENT    ::= empty
             | PLACE      ELEMENT
             | TRANSITION ELEMENT
             | ARC        ELEMENT
             | TYPEDEF    ELEMENT
             | \seeML{FILENAME} ELEMENT

FILENAME ::= STRING

PLACE  ::= \place{ID}{ NAME INIT CAP COLOUR }
        |  \place{ID}{ \like{ID} }
INIT   ::= empty | \init{ MULTISET }
CAP    ::= empty | \capacity{ MULTISET }
COLOUR ::= empty | \colour{ COLOURSET }

COLOURSET  ::= \like{ID} | ML-DEFINITION
MULTISET   ::= INTEGER | INTEGER`STRING MSLIST
MSLIST     ::= empty | + MULTISET
```

```
TRANSITION ::= \transition{ID}{ NAME GUARD }
            |  \transition{ID}{ \like{ID} }
GUARD      ::= empty | \guard{BOOLEXPR}

BOOLEXPR   ::= \like{ID} | ML-EXPRESSION

ARC        ::= \arc{ID}{ \from{ID} \to{ID} WEIGHT}
WEIGHT     ::= empty | \weight{ MULTISETEXPR }

MULTISETEXPR ::= \like{ID} | ML-EXPRESSION

TYPEDEF    ::= \typedef{ID}{ ML-DEFINITION }
```

It is possible to omit the guard in a transition specification, as defined by the grammar. In this case the guard expression defaults to *true*. Since Place/Transition Nets are considered as a special case of CPNs, we define a default colour set $\{\omega\}$ such that a multiset can be a simple integer value n which is equivalent to $n`\omega$. If the colour set consists of a single colour $x$, a missing arc weight defaults to $1`x$, and an integer arc weight n is equivalent to $n`x$.

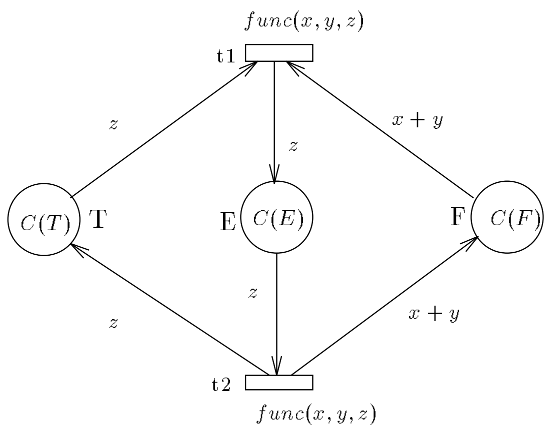

Figure 3: Example Coloured Petri Net.

Example 3 *Consider the simple Coloured Petri Net illustrated in Fig. 3 which models the well-known* dining philosophers *problem: The example is for 5 philosophers, so that*

$$C(p) = \begin{cases} \{p_1, \ldots, p_5\}, \text{ if } p = T \text{ or } p = E, \\ \{f_1, \ldots, f_5\}, \text{ if } p = F, \end{cases}$$

The guard function func(x,y,z) in that figure is defined to be

$$func(x,y,z) = \begin{array}{rl} if & (z=p_i)\wedge(x=f_i)\wedge(y=f_{i\oplus 1}) \\ then & TRUE \\ else & FALSE \end{array}$$

and $i \oplus 1$ *means i+1 modulo 5.*

The following is the description of that net in the APN notation:

```
\beginnet{Coloured Petri Net}
\seeML{MLSpecification}
\place{T}{\init{1`p1+1`p2`+1`p3+1`p4+1`p5} \colour{ with p1|p2|p3|p4|p5 }}
\place{E}{\colour{ \like{T} }}
\place{F}{\init{1`f1+1`f2`+1`f3+1`f4+1`f5} \colour{ with f1|f2|f3|f4|f5 }
\transition{t1}{ \guard{func(x,y,z)} }
\transition{t2}{ \like{t1} }
\arc{a1}{\from{T}  \to{t1} \weight{z}}
\arc{a2}{\from{t1} \to{E}  \weight{z}}
\arc{a3}{\from{E}  \to{t2} \weight{z}}
\arc{a4}{\from{t2} \to{T}  \weight{z}}
\arc{a5}{\from{t2} \to{F}  \weight{x+y}}
\arc{a6}{\from{F}  \to{t1} \weight{x+y}}
\endnet
```

The style macros for the corresponding LaTeX description given below differ somewhat from those used for Place/Transition Nets.

Coloured Petri Net:
`for ML function specification see` MLSpecification
`Place` T , `marking` {1‘p1+1‘p2‘+1‘p3+1‘p4+1‘p5} ,
`colourset`{ with p1|p2|p3|p4|p5 }
`Place` E ,
`colourset`{ `is defined as for` T }
`Place` F , `marking` {1‘f1+1‘f2‘+1‘f3+1‘f4+1‘f5} ,
`colourset`{ with f1|f2|f3|f4|f5 }
`Transition` t1 guard{ func(x,y,z) }
`Transition` t2 `is defined as for` t1
`Arc` a1 `from` T `to` t1 ,`weight` z
`Arc` a2 `from` t1 `to` E ,`weight` z
`Arc` a3 `from` E `to` t2 ,`weight` z
`Arc` a4 `from` t2 `to` T ,`weight` z
`Arc` a5 `from` t2 `to` F ,`weight` x+y
`Arc` a6 `from` F `to` t1 ,`weight` x+y

3.2.2 Further APN Notation for Coloured Petri Nets

The definition of CPNs as given by Jensen in [15] employs ML expressions for the description of multisets, arc expressions and guards. An earlier definition of CPNs[14] explicitly specifies all firing modes of a transition by a finite set of (transition) colours. This representation, known as the functional representation, has to be used for the purpose of analysis[15]. It is also the approach used for Coloured GSPNs defined in [7] and QPNs[3], both of which use a flow oriented view of a system.

When all firing modes of a transition explicitly specified, the definition of CPNs becomes:

Definition 3 *A* Coloured Petri Net *(CPN) is a 6-tuple* $CPN = (\Sigma, P, T, C, I^-, I^+, M_0)$*, where*

- Σ *is a set of non-empty colour sets,*
- P *is a finite and non-empty set of places,*
- T *is a finite and non-empty set of transitions, with* $P \cap T = \emptyset$*,*
- C *is a colour function* $P \cup T \mapsto \Sigma$*,*
- I^- *and* I^+ *are the backward and forward incidence functions defined on* $P \times T$ *such that* $I^-(p,t), I^+(p,t) \in [C(t) \to C(p)_{MS}], \forall\ (p,t) \in P \times T$*,*
- M_0 *is a function defined on* P *describing the initial marking such that* $M_0(p) \in C(p)_{MS}, \forall p \in P$*.*

This representation can be described in the APNN for CPNs of Sec. 3.2.1 by modifying the following productions.

```
COLOURSET   ::= \like{ID}
              |   with STRING COLOUR_REST
COLOUR_REST ::= empty
              |  '|' STRING COLOUR_REST

GUARD       ::= empty | \guard{ GUARDEXPR }
GUARDEXPR   ::= \like{ID}
              |   MODE_VAR = MODE_CONST GUARD_REST
GUARD_REST  ::= empty
              |  orelse MODE_VAR = MODE_CONST GUARD_REST

MODE_VAR    ::= STRING
MODE_CONST  ::= STRING
```

```
MULTISETEXPR ::= \like{ID}
               | MULTISET
               | case MODE_VAR of MODE_CONST
                 => MULTISET MSEXPR_REST
               | if MODE_VAR = MODE_CONST
                 then MULTISET  MSEXPR_ELSE

MSEXPR_REST  ::= empty
               | '|' MODE_CONST => MULTISET MSEXPR_REST
MSEXPR_ELSE  ::= empty
               | else MULTISET
               | else MULTISETEXPR

MULTISET ::= INTEGER
           | INTEGER'STRING MSLIST
MSLIST   ::= empty | + INTEGER'STRING MSLIST
```

Note that a guard expression lists all possible firing modes by a disjunction over all possible bindings of a mode variable, e.g., a transition with firing modes *black* and *blue* would have a guard `\guard{mode=black orelse mode =blue}`.

Obviously this is not a straightforward solution in terms of a functional representation, but it integrates both kinds of CPNs into a single notational framework.

4 Stochastic Petri Nets

In this section we consider Petri Nets to which time has been added. There are a lot of concepts for introducing time, e.g., deterministic and stochastic times, timing information attached to places or transitions etc. (see e.g. [18]). Here we consider time-augmented nets to show the extensibility of the APNN. The most well-known of such nets are Generalised Stochastic Petri Nets (GSPNs).

4.1 Generalised Stochastic Petri Nets

As the name indicates, Generalised Stochastic Petri Nets (GSPNs) are timed nets. An exponentially distributed firing time is associated with timed transitions. Such nets also contain immediate transitions which fire instantaneously and with priority over timed transitions. Furthermore the definition of a GSPN permits inhibitor arcs and different levels of priority for immediate transitions.

Definition 4 *A GSPN (cf. [1, 2]) is a 4-tuple* $GSPN = (PN, Pr, H, W)$ *where*

- $PN = (P, T, I^-, I^+, M_0)$ *is the underlying Place/Transition Net,*
- $H \subset P \times T$ *is a set of inhibitor arcs,*
- Pr *is an assignment of priorities to transitions, with priority 0 for timed transitions and higher priorities* (≥ 1) *for immediate transitions, and*
- $W = (w_1, \ldots, w_m)$ *is an array whose entry* $w_i \in \mathbb{R}^+$
  - *is a rate of a negative exponential distribution specifying the firing delay, when transition* t_i *is a timed transition, or*
  - *is a weight specifying the relative firing frequency, when transition* t_i *is an immediate transition.*

4.1.1 APN Notation for GSPNs

The default value for arc weights and firing rates or frequencies is 1. Unless otherwise specified, a transition is considered to be timed. Note that if all priorities equal zero then the definition of a GSPN coincides with the definition of a Stochastic Petri Net (SPN)[2].

The grammar for GSPNs extends the grammar of Place/Transition Nets by the concepts of priority, weights and inhibitor arcs. This leads to the following modifications:

```
TRANSITION ::= \transition{ID}{ NAME PRIO T_WEIGHT }
             | \transition{ID}{ \like{ID} }

PRIO       ::= empty | \prio{ INTEGER }
T_WEIGHT   ::= empty | \weight{ REAL }

ARC        ::= \arc{ID}{ \from{ID} \to{ID} WEIGHT A_TYPE }
A_TYPE     ::= empty | \type{ordinary}  | \type{inhibitor}
```

A missing type specification of an arc defaults to an ordinary arc. We assume the usual production rules for the derivation of real constants from REAL. Note that marking dependent firing rates are not considered for now.

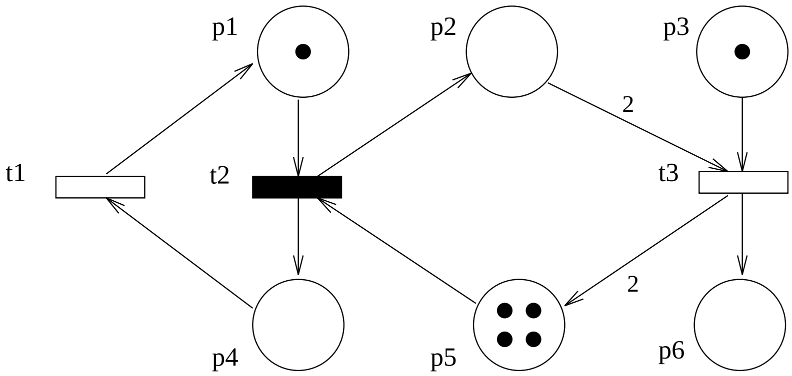

Figure 4: Example GSPN.

Example 4 *Consider the simple GSPN illustrated in Fig. 4. The following is the definition of that net using the APN notation:*

```
\beginnet{gspnexample}
\place{p1}{\init{1}}  \place{p2}{}
\place{p3}{\like{p1}} \place{p4}{}
\place{p5}{\init{4}}  \place{p6}{}

\transition{t1}{\prio{0} \weight{2.0}}
\transition{t2}{\prio{1} \weight{1.0}}
\transition{t3}{}

\arc{a1}{\from{p1} \to{t2}} \arc{a2}{\from{t2} \to{p4}}
\arc{a3}{\from{p4} \to{t1}} \arc{a4}{\from{t1} \to{p1}}
\arc{a5}{\from{t2} \to{p2}} \arc{a6}{\from{p5} \to{t2}}
\arc{a7}{\from{p3} \to{t3}} \arc{a8}{\from{t3} \to{p6}}
\arc{a9}{\from{p2}  \to{t3} \weight{2}}
\arc{a10}{\from{t3} \to{p5} \weight{2}}
\endnet
```

The LaTeX *Output is:*

gspnexample:
`Place` *p1* , `marking` {*1*}
`Place` *p2*
`Place` *p3* `is defined as for` *p1*
`Place` *p4*
`Place` *p5* , `marking` {*4*}
`Place` *p6*

`Transition` *t1* , `priority` *0* ,`weight` *2.0*
`Transition` *t2* , `priority` *1* ,`weight` *1.0*
`Transition` *t3*

```
Arc a1  from p1   to t2
Arc a2  from t2   to p4
Arc a3  from p4   to t1
Arc a4  from t1   to p1
Arc a5  from t2   to p2
Arc a6  from p5   to t2
Arc a7  from p3   to t3
Arc a8  from t3   to p6
Arc a9  from p2   to t3  ,weight 2
Arc a10  from t3   to p5  ,weight 2
```

4.2 Coloured Generalised Stochastic Petri Nets

In the case of Generalised Stochastic Petri Nets, tokens do not have colours. In order to merge the two concepts of coloured tokens and timed nets, tokens have colours and an execution rate is now associated with every *firing mode* of a transition. We call the resultant net, a Coloured Generalised Stochastic Petri Net (CGSPN).

Such nets are however, not yet considered to be hierarchical. Hierarchical nets are described in Sec. 5.

Definition 5 *Using the same convention as that in Sec. 3.2, we define a CGSPN as a tuple* $CGSPN = (\Sigma, P, T, A, C, Pr, H, G, W, E, I)$ *where*

- Σ *is a set of non-empty colour sets,*
- P *is a finite and non-empty set of places,*
- T *is a finite and non-empty set of transitions,*
- A *is a finite set of arcs such that:*
  - $P \cap T = P \cap A = T \cap A = \emptyset$.
- C *is a colour function* $P \mapsto \Sigma$.
- Pr *is an assignment of priorities to transitions, with priority 0 for timed transitions and higher priorities* (≥ 1) *for immediate transitions.*
- $H \subset P \times T$ *is a set of inhibitor arcs*[2].
- G *is the guard function from* T *into expressions such that:*
  - $\forall t \in T, \quad Type(G(t)) = \mathbb{B}$ *and* $Type(Var(G(t))) \subseteq \Sigma$.

[2] The semantics of inhibitor arcs in CGSPNs follows that of [7] .

- *W is the weight function from the transitions T into expressions such that $\forall t \in T$:*

  - *$Type(W(t)) = \mathbb{R}^+$ and $Type(Var(W(t))) \subseteq \Sigma$ which specifies the firing delay when transition t is a timed transition and the relative firing frequency, when transition t is an immediate transition.*

- *E is the arc function from A into expressions such that:*

  - *$\forall a \in A, \quad Type(E(a)) = C(p(a))_{MS}$ and $Type(Var(E(a))) \subseteq \Sigma$ where we $p(a)$ denotes the place connected by arc a.*

- *I is the initialisation function defined from P into closed expressions such that:*

  - *$\forall p \in P, \quad Type(I(p)) = C(p)_{MS}$*

4.2.1 APN Notation for CGSPNs

The notation clearly has to provide the specification of colour sets for places and types of colour sets. Furthermore transitions can fire according to different firing modes which results in generalising arc weights to expressions that evaluate to multisets and a guard expression which evaluates to a boolean value. Therefore transition descriptions are extended by guards, arc weights are generalised to contain expressions which evaluate to a multiset over the colour set attached to the input place. Place descriptions have a colour set description and its attributes for initialisation and capacity are generalised to expressions of the appropriate type.

In the following the complete set of productions for CGSPNs is given.

```
NET        ::= \beginnet{ID} ELEMENT \endnet
ELEMENT    ::= empty
             | PLACE      ELEMENT
             | TRANSITION ELEMENT
             | ARC        ELEMENT
             | TYPEDEF    ELEMENT
             | \seeML{ FILENAME } ELEMENT

FILENAME ::= STRING
ID       ::= STRING

PLACE  ::= \place{ID}{ NAME INIT CAP COLOUR }
        |  \place{ID}{ \like{ID} }
NAME   ::= empty | \name{ STRING }
INIT   ::= empty | \init{ MULTISET }
CAP    ::= empty | \capacity{ MULTISET }
COLOUR ::= empty | \colour{ COLOURSET }
```

```
COLOURSET  ::= \like{ID} | ML-DEFINITION

TRANSITION ::= \transition{ID}{ NAME PRIO T_WEIGHT GUARD }
             |  \transition{ID}{ \like{ID} }
PRIO       ::= empty | \prio{ INTEXPR }
T_WEIGHT   ::= empty | \weight{ REALEXPR }
GUARD      ::= empty | \guard{ BOOLEXPR }

ARC        ::= \arc{ID}{ \from{ID} \to{ID} WEIGHT A_TYPE }

WEIGHT     ::= empty | \weight{ MULTISETEXPR }
A_TYPE     ::= empty | \type{ordinary}  | \type{inhibitor}

MULTISET     ::= INTEGER`STRING MSLIST
MSLIST       ::= empty | + MULTISET

INTEXPR      ::= \like{ID} | ML-EXPRESSION
REALEXPR     ::= \like{ID} | ML-EXPRESSION
MULTISETEXPR ::= \like{ID} | ML-EXPRESSION
BOOLEXPR     ::= \like{ID} | ML-EXPRESSION

TYPEDEF    ::= \typedef{ID}{ ML-DEFINITION }
```

Definitions of integer, boolean, real and multiset expressions follow the ML-syntax[20]. Arcs default to those of type "ordinary".

Example 5 *Consider the familiar Producer-Consumer problem, modified to represent a gizmo assembly process. The associated CGSPN is illustrated in Fig. 5. Gizmos are only assembled as needed and require three wheels, a body and a door. It takes different exponentially distributed times to build the wheels, body and door, respectively and assembling a gizmo takes an exponential time with a different mean rate.*

In other words, transition CONSUME (transition CO) is enabled when three of the tokens with colour "WHEEL" *and one token each of colour* "BODY" *and* "DOOR" *are on place B. A token of colour* "GIZMO" *on place* C_1 *indicates that a gizmo needs to be assembled, if possible. Parts are only produced as they are consumed.*

The various functions are defined as follows:

$$C(p) = \begin{cases} \{\text{WHEEL, BODY, DOOR}\}, \textit{if } p \in \{B, P_1\}, \\ \{\text{GIZMO}\}, \textit{if } p \in \{C_1, C_2, P_2\}. \end{cases}$$

$$
\begin{aligned}
Pr(t) &= \begin{cases} 1, & \mathit{if}\; t \in \{I_1, I_2\}, \\ 0, & \mathit{otherwise.} \end{cases} \\
H &= \emptyset \\
G(t) &= \begin{cases} x = \text{WHEEL} \wedge y = \text{BODY} \wedge z = \text{DOOR} \wedge g = \text{GIZMO}, \;\; \mathit{if}\; t \in \{I_1, CO\}, \\ \mathit{TRUE}, \;\; \mathit{otherwise.} \end{cases} \\
W(t) &= \begin{cases} \begin{array}{lll} \mathit{case}\; x\; \mathit{of} & \text{WHEEL} & => 5 \\ \quad\quad | & \text{BODY} & => 2 \\ \quad\quad | & \text{DOOR} & => 1, \;\; \mathit{if}\; t = PR, \end{array} \\ 20, \;\; \mathit{if}\; t = CO, \\ 1, \;\; \mathit{otherwise.} \end{cases} \\
E(a) &= \begin{cases} x, \;\; \mathit{if}\; a \in \{(P_1, PR), (PR, B)\}, \\ 3`x + 1`y + 1`z, \;\; \mathit{if}\; a \in \{(B, CO), (I_1, P_1)\}, \\ g, \;\; \mathit{if}\; a \in \{(CO, P_2), (P_2, I_1), (CO, C_2), (C_2, I_2), (I_2, C_1), (C_1, CO)\}. \end{cases} \\
I(p) &= \begin{cases} \{\text{GIZMO}\}, \mathit{if}\; p \in \{C_1, P_2\}, \\ \emptyset, \;\; \mathit{otherwise.} \end{cases}
\end{aligned}
$$

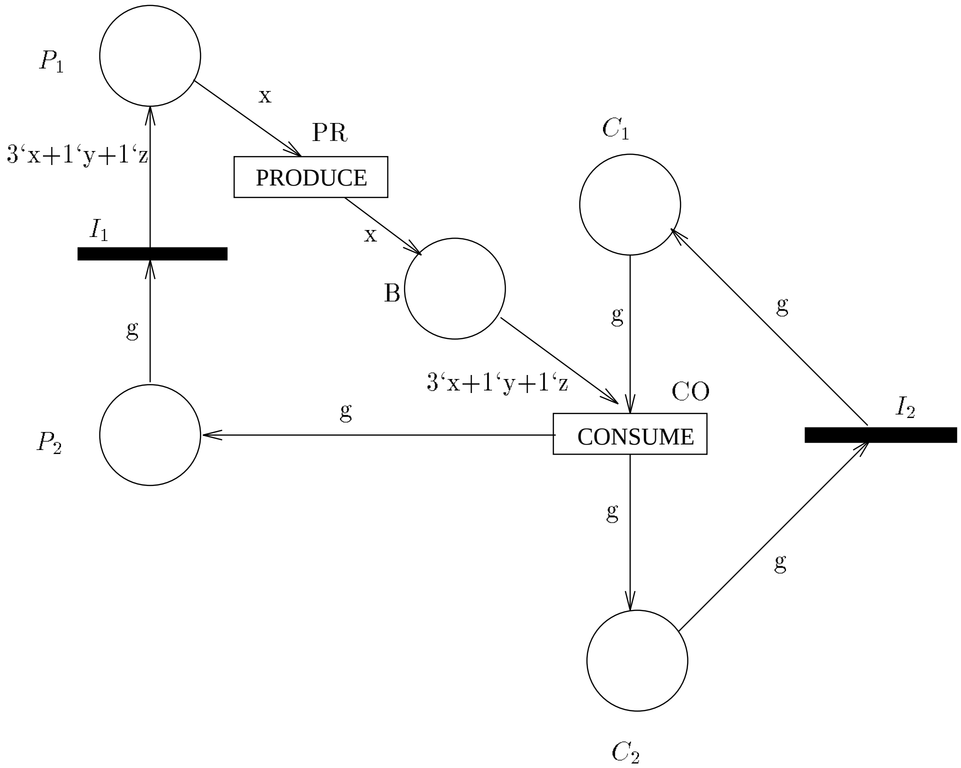

Figure 5: Coloured GSPN of a gizmo assembly process.

```
\beginnet{cgspnexample}
\place{P1}{ \colour{with WHEEL|BODY|DOOR} }
\place{P2}{ \init{GIZMO} \colour{with GIZMO} }
\place{B} { \like{P1} }
\place{C1}{ \init{GIZMO} \colour{with GIZMO} }
\place{C2}{ \colour{with GIZMO} }

\transition{I1}{ \prio{1}
            \guard{x=WHEEL and y=BODY and z=DOOR and g=GIZMO} }
\transition{I2}{ \prio{1} }
\transition{PR}{ \name{PRODUCE} \prio{0}
            \weight{ case x of WHEEL => 5 | BODY => 2 | DOOR => 1} }
\transition{CO}{ \name{CONSUME} \prio{0}
            \guard{ \like{I1} } \weight{20} }

\arc{ } { \from{P1} \to{PR} \weight{x} }
\arc{ } { \from{PR} \to{B}  \weight{x} }
\arc{ } { \from{B}  \to{CO} \weight{3`x+1`y+1`z} }
\arc{ } { \from{CO} \to{P2} \weight{g} }
\arc{ } { \from{P2} \to{I1} \weight{g} }
\arc{ } { \from{I1} \to{P1} \weight{3`x+1`y+1`z} }
\arc{ } { \from{C1} \to{CO} \weight{g} }
\arc{ } { \from{CO} \to{C2} \weight{g} }
\arc{ } { \from{C2} \to{I2} \weight{g} }
\arc{ } { \from{I2} \to{C1} \weight{g} }
\endnet
```

5 Hierarchies

Petri Nets tend to become very large if used for modeling real world problems. One way of avoiding the difficulty of representing such nets graphically, is to define one or more subnets and to possibly build a hierarchy of subnets. In this way, large nets can be represented in a structured way which, moreover, allows one to use a top-down design for complex models as well as the reuse of net descriptions or identical submodels.

Huber et. al. [13] describe five concepts of hierarchical CPNs: substitution transitions, substitution places, invocation transitions, fusion sets for places and fusion sets for transitions. They define the semantics of these hierarchy concepts by giving a translation into an equivalent non-hierarchical CPN. In the following we demonstrate how these concepts can be expressed in an extended APNN but give only a very brief, informal description of the semantics of these hierarchical concepts; for details the reader is referred to [13].

5.1 Substitution transitions

We use the notation in [13] and consider a hierarchical CPN as a set of related pages. If a page has a refined node n, this page is called superpage in relation to the subpage which gives the refinement of n. In this section we consider the refinement of transitions by substitution. Substitution of transitions is very much like macro-expansion in programming languages.

Figure 6 gives an example with a page N containing a substitution transition n and the subpage N', which corresponds to n. Nodes in the preset or postset of n are called socket nodes. Their counterparts inside the subpage are called port nodes and a binding function b must be given, which assigns a port node to each socket node. Note that this function need not be injective or surjective, e.g. $b(p_1) = b(p_2)$ and p'_4 is not bound to any socket node of n.

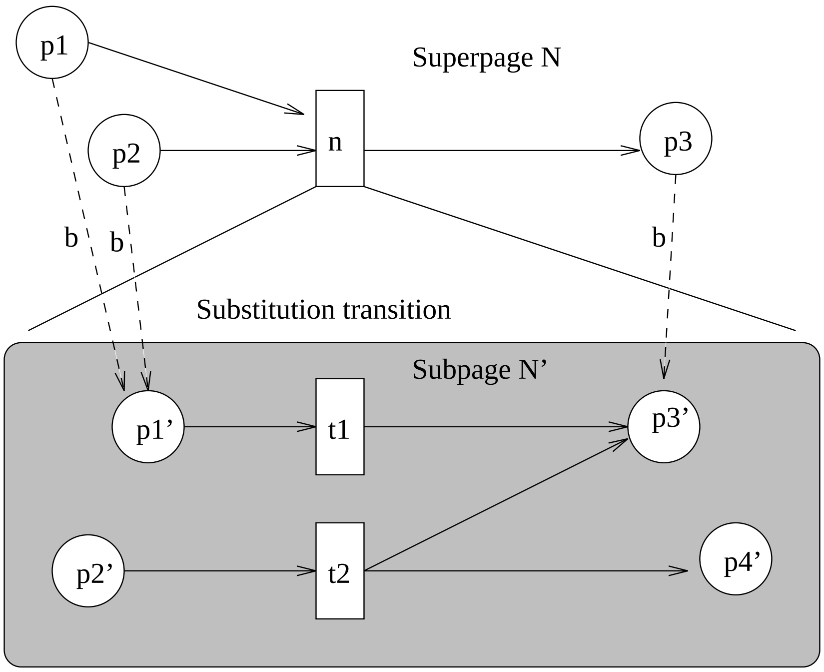

Figure 6: Trivial example of substitution transition

An APNN description has to provide the following information:

- for the substitution transition:
  - the ID of the subpage given by `\substitute{ID}`
  - the binding function b, specified in the arc description, which connects the transition with its input and output places.

    The binding b of an input place is given within an arc description by a sequence of `\bind{SOCKETID}\with{TRANSITIONID}` `\cont{PORTID}` arc attributes.

`\cont{PORTID}` means that a node with identity `PORTID` is an element of the subpage *contained* in the node of the preceding attribute, namely the node with `TRANSITIONID`.

For an output place the corresponding arc attributes are `\bind{TRANSITIONID}\cont{PORTID}\with{SOCKETID}`.

- for nodes of a subpage:
  - an attribute which denotes input and output ports, i.e. `\port{in}`, `\port{out}` and `\port{io}`.

Since the preset and postset of a substitution transition define the set of socket nodes, a special notation to distinguish socket nodes is not necessary.

The new elements of an APNN description for the example pages N and N' are:

```
\beginnet{N}
\transition{n}{ \substitute{N'} ... }
\arc{p1n}{\from{p1} \to{n} \bind{p1} \with{n} \cont{p1'} ... }
\arc{p2n}{\from{p2} \to{n} \bind{p2} \with{n} \cont{p1'} ... }
\arc{np3}{\from{n} \to{p3} \bind{n} \cont{p3'} \with{p3} ... }
\endnet

\beginnet{N'}
\place{p1'}{\port{in}} \place{p2'}{\port{in}}
\place{p3'}{\port{out}} \place{p4'}{\port{out}}
...
\endnet
```

The reason for distributing the binding information over arcs connecting the preset, transition, and postset is, that the existence of such a binding depends on the existence of such an arc. Furthermore in case of refining adjacent nodes, the binding information depends on both refinements and is not exclusive to one of its nodes.

5.2 Substitution places

Substitution places are analogous to substitution transitions apart from a different semantics, i.e. the translation into a non-hierarchical net is different from the translation of substitution transitions. Nevertheless the APNN description can be handled analogously, a substitution place specification contains the subpage ID by `\substitute{ID}` and the binding function is distributed over the arc description just as for substitution transitions. Port nodes of the subpage have an additional attribute `\port{in}`, `\port{out}` or `\port{io}`.

5.3 Invocation transitions

Invocation transitions are like function calls, in fact they even allow recursive re-use of subpages. The binding of socket- and port nodes is equivalent to substitution transitions. We change `\substitute{ID}` to `\invoke{ID}` to distinguish invocation transitions from substitution transitions in the APNN. Since subpages of invocation transitions allow for explicit exit--statements to denote termination conditions of an invocation, place and transition specifications can be extended by an `\exit` attribute in the APN notation.

5.4 Fusion sets for places and fusion sets for transitions

The main idea about fusion is to fold a set of nodes into a single node. Consequently a fusion set contains an arbitrary number of places or an arbitrary number of transitions.

Huber et. al. distinguish three types of fusion sets:

Page fusion set. This is just a drawing convenience in graphical user interfaces in case of a single page instance[3]. In case of several page instances, a page fusion set folds all members from all page instances into a single node.

Instance fusion set. This means that all members per instance of a page are folded into a single instance-specific node.

Global fusion set. This allows fusion set members from all pages and all page instances to be merged into a single node.

Fusion sets are integrated into the APNN by extending a net description with `\fuse{ID}{TYPE}{IDLIST}`, where ID provides a unique identifier for a fusion set. `TYPE` is a string constant `page`, `inst` or `global`. `IDLIST` is a |-separated list of unique node identifier. This is not critical in case of page or instance fusion sets, but multiple reuse of a page can lead to difficulties in global fusion sets. In order to handle this we assume an appropriate naming convention as mentioned in e.g. [13] for Design/CPN, to obtain unique identifier.

The information contained in a fusion set definition is aggregated into a new net element in APNN, instead of distributing the information over its members. One reason for this is to avoid repetitive specification of type and ID. But mainly because conceptually a new node is created by a fusion set, i.e., the fusion of, e.g., transitions results into a new transition whose guard is a conjunction of all guards of the fusion set members.

[3] Using independent copies of subpages lead to the notion of instances of a page.

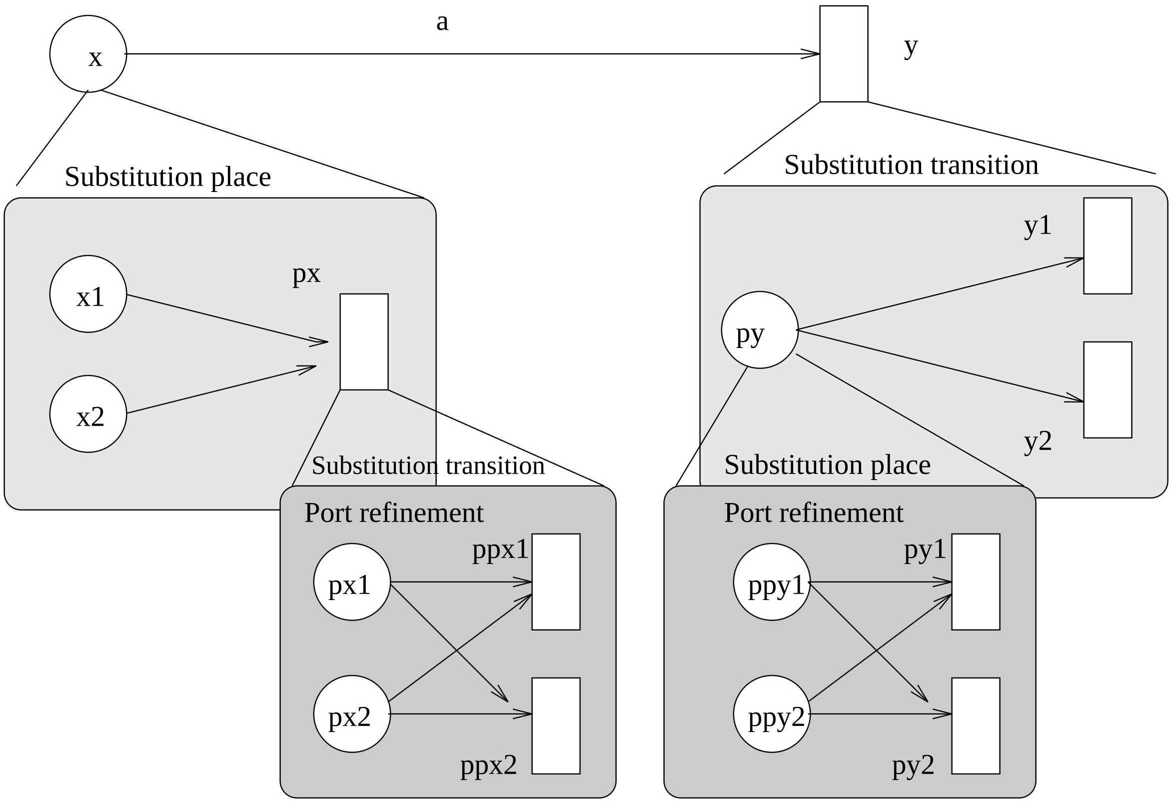

Figure 7: Pathological example for refined adjacent nodes with port refinement

5.5 Refinement of adjacent nodes

The hierarchical concepts regarded so far reveal certain difficulties to handle the refinement of adjacent nodes, a case which is especially considered in [11]. We try to demonstrate the difficulties by means of an artificial example as shown in Fig. 7, which even contains refinement of port nodes.

In [10], Chapter 4.2.4.2, R. Fehling suggests to specify a binding between refined ports, e.g. between ppx1 and ppy1. Here, for consistency, we suggest instead an explicit binding between socket nodes, e.g. x1 and x2, and refined port nodes, e.g. ppy1 and ppy2. The binding information is described with arc a=(x,y) in APNN by the following statement:

```
\arc{a}{ \from{x}  \to{y}
\bind{x} \cont{px} \cont{ppx1} \with{y} \cont{y1}
\bind{x} \cont{px} \cont{ppx2} \with{y} \cont{y2}
\bind{x} \cont{x1} \with{y} \cont{py} \cont{ppy1}
\bind{x} \cont{x2} \with{y} \cont{py} \cont{ppy2}
... }
```

In [11] port refinements do not contain port nodes to match their environment. We decided to support this as well in order to avoid arcs between separate page descriptions and to keep the descriptions homogeneous. In our example arcs (x1,px) and (x2,px) carry the binding for the port refinement of px:

```
\arc{x1px}{\from{x1} \to{px}
\bind{x1} \with{px} \cont{px1}
...}
\arc{x2px}{\from{x2} \to{px}
\bind{x2} \with{px} \cont{px2}
...}
```

Note that all implicit bindings like the binding between px and y can be extracted from the given binding specification by shortening the sequences of cont-statements, e.g. `\bind{x} \cont{px} \with{y}`. Implicit bindings and implicit arcs are regarded in detail in [11].

All these concepts of a hierarchy are now used for the notation of Hierarchical CGSPNs, which serve as an example. The grammar of CGSPNs is extended by the following to accommodate Hierarchical CGSPNs.

5.6 APN Notation for Hierarchical CGSPNs

Modified Productions:

```
NET  ::= empty
      | \inputnet{ID}  NET
      | \beginnet{ID}  ELEMENT   \endnet NET
      | \beginnet{ID}  \like{ID} \endnet NET

ELEMENT    ::= empty
             | PLACE      ELEMENT
             | TRANSITION ELEMENT
             | ARC        ELEMENT
             | TYPEDEF    ELEMENT
             | \seeML{ FILENAME } ELEMENT
             | FUSION     ELEMENT

FUSION ::= \fuse{ID}{TYPE}{ID IDLIST}
TYPE   ::= page  | inst | global
IDLIST ::= empty | '|' ID IDLIST

PLACE  ::= \place{ID}{ NAME INIT CAP COLOUR P_TYPE PORT EXIT}
        |  \place{ID}{ \like{ID} }
P_TYPE ::= empty | \substitute{ID}
PORT   ::= empty | \port{in} | port{out} | \port{io}
EXIT   ::= empty | \exit
```

```
TRANSITION ::= \transition{ID}{ NAME T_TYPE PRIO T_WEIGHT GUARD
                  PORT EXIT}
             |  \transition{ID}{ \like{ID} }
T_TYPE     ::= empty | \substitute{ID} | \invoke{ID}

ARC        ::= \arc{ID}{ \from{ID} \to{ID} WEIGHT A_TYPE BIND }
BIND       ::= empty | \bind{ID} CONT \with{ID} CONT
CONT       ::= empty | \cont{ID} CONT
```

5.7 Hierarchical Queueing Petri Nets

Hierarchical Queueing Petri Nets (HQPNs) are a special kind of hierarchical Petri Nets with certain extensions to integrate queueing networks. The main motivation is to combine hierarchical description techniques with corresponding hierarchical solution techniques [5]. We describe the APNN for this modeling formalism for several reasons:

- to demonstrate the extensibility of the APNN to further Petri net formalisms,
- currently work is under way to support HQPNs by an adequate software tool which supports the APNN.

In 1989 Bause *et al.*[3, 4] introduced the concept of a "queued place" into net theory for the purpose of performance analysis (see Fig. 8). A queued place combines the notion of a queue, where tokens are queued and served according to a certain scheduling strategy, with a depository, which is identical to an ordinary place. Input transitions of a queued place fire tokens into the queue where they are not available to the output transitions until service completion. On service completion the tokens are put into the depository whence they are available to output transitions in the conventional way. They called the new type of net a Queueing Petri Net (QPN).

QPNs additionally allow the refinement of places leading to Hierarchical QPNs (see Fig. 9).

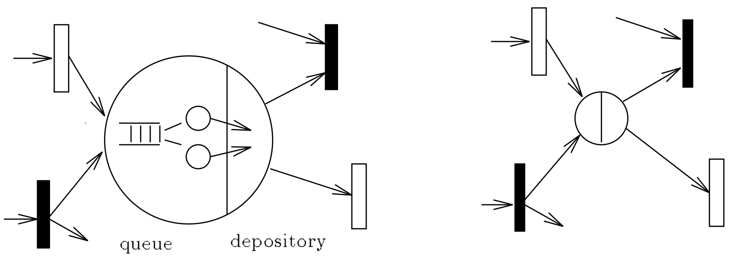

Figure 8: Example queued place of a HQPN and its shorthand notation

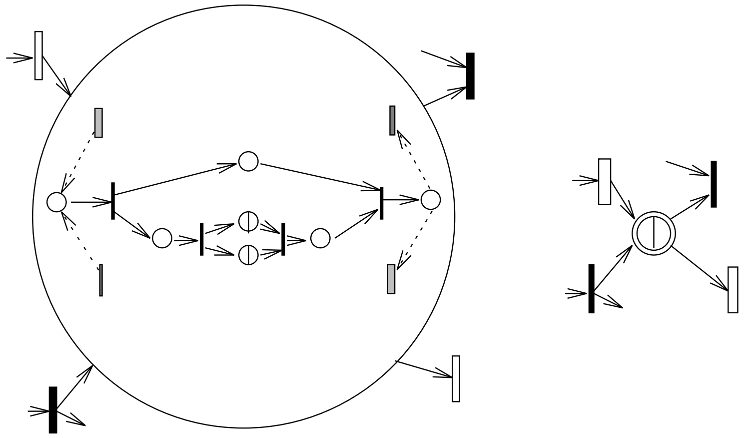

Figure 9: Example subnet place of a HQPN and its shorthand notation

Definition 6 (HQPN) *A* Hierarchically Queueing Petri Net *(HQPN) is a triple* $HQPN = (CGSPN, QS, RF)$ *where*

- $CGSPN = (\Sigma, P, T, A, C, Pr, H, G, W, E, I)$ *as defined before, is the underlying coloured GSPN,*
- QS *is a function defined on* $P_Q \subseteq P$ *giving an assignment of a queueing specification to a queued place* $p \in P_Q$*,*
- RF *is a function defined on* $P_R \subseteq P \setminus P_Q$ *giving the refinement of a place* $p \in P_R$ *as a HQPN.*

Hence HQPNs are CGSPNs where some places are queued places and some places are substitution places. Note that by definition nodes cannot be both a substitution place and a queued place. Fusion sets are supported with the constraint that only substitution places can be fused and that all substitution places of a fusion set share the same subpage. The page hierarchy which results from a HQPN forms a directed acyclic graph.

The description of the underlying CPN is based on the APNN of the functional representation as given in Sec. 3.2.2, such that all firing modes are explicitly described and hence no external ML-definitions are necessary.

5.7.1 APN Notation for HQPNs

Productions:

```
NET         ::= empty
             | \inputnet{ID}  NET
             | \beginnet{ID}  ELEMENT \endnet NET
             | \beginnet{ID}  \like{ID} \endnet NET
```

```
ELEMENT    ::= empty
             | PLACE      ELEMENT
             | TRANSITION ELEMENT
             | ARC        ELEMENT
             | TYPEDEF    ELEMENT
             | FUSION     ELEMENT

ID      ::= STRING
TYPEDEF ::= \typedef{ID}{ COLOURSET }

FUSION ::= \fuse{ID}{TYPE}{ID IDLIST}
TYPE   ::= page  | inst | global
IDLIST ::= empty | '|' ID IDLIST


PLACE  ::= \place{ID}{ NAME INIT CAP COLOUR P_TYPE }
        |  \place{ID}{ \like{ID} }
NAME   ::= empty | \name{ STRING }
INIT   ::= empty | \init{ MULTISET }
CAP    ::= empty | \capacity{ MULTISET }
COLOUR ::= empty | \colour{ COLOURSET }
P_TYPE ::= empty
        |  \substitute{ID}
        |  \queue{immediate} SCHED \weight{ TUPLEEXPR }
        |  \queue{timed}     SCHED \weight{ TUPLEEXPR }

SCHED  ::= empty
        |  \sched{STRING} SCHED
        |  \noserver{ INTEGER } SCHED
        |  \rank{ INTEXPR } SCHED
        |  \prio{ INTEXPR } SCHED

TRANSITION ::= \transition{ID}{ NAME PRIO T_WEIGHT GUARD PORT}
            |  \transition{ID}{ \like{ID} }
PRIO       ::= empty | \prio{ INTEXPR }
T_WEIGHT   ::= empty | \weight{ REALEXPR }
GUARD      ::= empty | \guard{ GUARDEXPR }
PORT       ::= empty | \port{in} | \port{out} | \port{io}

ARC        ::= \arc{ID}{ \from{ID} \to{ID} WEIGHT A_TYPE BIND }
WEIGHT     ::= empty | \weight{ MULTISETEXPR }
A_TYPE     ::= empty | \type{ordinary}  | \type{inhibitor}
BIND       ::= empty | \bind{ID} CONT \with{ID} CONT
CONT       ::= empty | \cont{ID} CONT
```

`TUPLEEXPR` denotes an ML-expression which evaluates to a tuple (*mean*,*coeff_var*) of real values where *mean* is the mean and *coeff_var* the coefficient of variation for the service time of a customer in the queue. The `rank` expression allows the specification of priorities for different colours of tokens to determine the state of the queue in case of bulk arrivals.

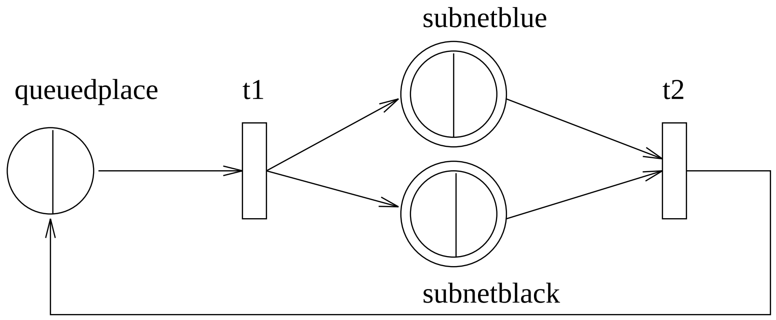

Figure 10: Example net with a queued place and two refined places.

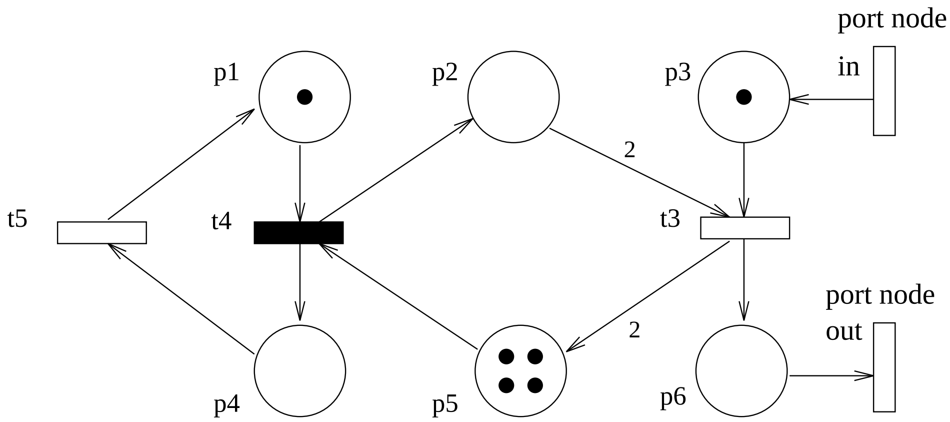

Figure 11: Subpage for substitution places

Fig. 10 is an example intended merely to demonstrate the APNN for HQPNs. The queued place has a FCFS scheduling strategy and its colour set contains only two colours: *black* and *blue*. Transitions t1 and t2 each have two firing modes: *fireblack* and *fireblue*. The transitions merely transport a token to or from the queued place to *subnetblack* or to *subnetblue*. The refinement of the places *subnetblack* and *subnetblue* are instances of the single-coloured GSPN given in Fig. 11 with input port *in* and output port *out*.

Specification of the example:

```
\inputnet{subpagespecification}

\beginnet{hqpnexample}
\place{queuedplace}{\init{1'black + 1'blue} \colour{ with black | blue}
\queue{timed} \sched{FCFS} \noserver{1}
\rank{case colour of black => 1 | blue => 2}
\weight{case colour of black => (3.5,1.0) | blue => (1.2,1.5)}}
```

```
\place{subnetblack}{\substitute{subpagespecification}}
\place{subnetblue}{\substitute{subpagespecification}}

\transition{t1}{\prio{0}
\weight{case mode of fireblack => 1.5 | fireblue => 2.1}
\guard{mode = fireblack orelse mode = fireblue}}
\transition{t2}{\like{t1}}

\arc{a}{\from{t2} \to{queuedplace}
\weight{case mode of fireblack => 1`black | fireblue => 1`blue}}
\arc{b}{\from{queuedplace} \to{t1} \weight{\like{a}}}
\arc{c}{\from{t1} \to{subnetblue}
\weight{if mode = fireblue => 1 else empty}
\bind{t1} \with{subnetblue} \cont{in}}
\arc{d}{\from{t1} \to{subnetblack}
\weight{if mode = fireblack => 1 else empty}
\bind{t1} \with{subnetblack} \cont{in}}
\arc{e}{\from{subnetblue} \to{t2}
\weight{if mode = fireblue => 1 else empty}
\bind{subnetblue} \cont{out} \with{t2}}
\arc{f}{\from{subnetblack} \to{t2}
\weight{if mode = fireblack => 1 else empty}
\bind{subnetblack} \cont{out} \with{t2}}
\endnet
```

In this case the keyword `empty` in the weight expressions of arcs denotes the empty multiset. Note that `\substitute{ID}` creates a fresh instance from the net with this ID. If two substitution places shall share the same instance, they must be members of the same fusion set.

The LaTeX output of the example is:

`Net description imported from` subpagespecification
hqpnexample:
`Place` queuedplace ,
`marking` {1'black + 1'blue} ,
`colourset`{ with black | blue } ,
timed `queueing place with` FCFS scheduling , 1 `server(s)` ,
`order of arrival in case of bulk arrivals:`
case colour of black => 1 | blue => 2 ,
`weight` case colour of black => (3.5,1.0) | blue => (1.2,1.5)
`Place` subnetblack `is an instance of` subpagespecification
`Place` subnetblue `is an instance of` subpagespecification

`Transition` t1 , `priority` 0 ,
`weight` case mode of fireblack => 1.5 | fireblue => 2.1 ,
`guard`{mode = black orelse mode = blue}

```
Transition t2 is defined as for t1

Arc a  from t2   to queuedplace  ,
weight case mode of fireblack => 1‘black | fireblue => 1‘blue
Arc b  from queuedplace   to t1  ,
weight is defined as for a
Arc c  from t1   to subnetblue  ,
weight if mode = fireblue => 1 else empty
binding t1 with subnetblue wherein it binds in
Arc d  from t1   to subnetblack  ,
weight if mode = fireblack => 1 else empty
binding t1 with subnetblack wherein it binds in
Arc e  from subnetblue   to t2  ,
weight if mode = fireblue => 1 else empty
binding subnetblue wherein it binds out  with t2
Arc f  from subnetblack   to t2  ,
weight if mode = fireblack => 1 else empty
binding subnetblack wherein it binds out  with t2
```

6 Summary

The first attempt at defining a universal notation for describing Petri Net models was made in 1988. The authors could find no other, or any subsequent reference to the subject when the need arose for such an interface in the collaboration between their respective research groups. In their particular case they wanted to exchange model descriptions for analysis by the QPN-analyser[6] developed, and being developed, by one group with the DNA-net[8] developed by the other.

The notation described in this report proposes an uniform, extensible formalism which should accommodate a large class of nets. For Coloured Petri Nets the APNN employs ML so far. The APNN can easily be modified to support other “programming languages”. The notation moreover allows for a human readable model description which can be derived directly using LaTeX commands known to a wide audience allowing a single format for the incorporation of the net description.

At the same time the APNN allows different researchers to discuss and experiment with variations of models of actual systems using possibly different analysis algorithms implemented in tools which, for different reasons, they do not share.

Comments about the APNN are clearly most welcome and should be addressed to anyone of the authors. We trust that, while almost certainly not acceptable in its entirety at present, it will be the start of a discussion leading to a generally accepted abstract formalism for describing high-level Petri Nets.

References

[1] M. Ajmone-Marsan, G. Balbo, and G. Conte. A class of Generalised Stochastic Petri Nets for the performance evaluation of multiprocessor systems. *ACM Transactions on Computer Systems*, 2:93–122, 1984.

[2] M. Ajmone-Marsan, G. Balbo, and G. Conte. *Performance Models of Multiprocessor Systems.* MIT Press Series in Computer Science, 1986.

[3] F. Bause. Queueing Petri Nets — a formalism for the combined qualitative and quantitative analysis of systems. In *5th International Workshop on Petri Nets and Performance Models, Toulouse (France)*, pages 14–23, 1993.

[4] F. Bause and H. Beilner. Eine Modellwelt zur Integration von Warteschlangen- und Petri-Netz-Modellen. In *Proceedings of the 5th GI/ITG-Fachtagung, Messung, Modellierung und Bewertung von Rechensystemen und Netzen*, pages 190–204. Braunschweig, Gesellschaft für Informatik (GI), Germany, 9 1989.

[5] F. Bause, P. Buchholz, and P. Kemper. Hierarchically combined Queueing Petri Nets. In *11th International Conference on Analysis and Optimizations of Systems, Discrete Event Systems, Sophia-Antipolis (France)*, June 1994.

[6] F. Bause and P. Kemper. QPN-tool for qualitative and quantitative analysis of queueing Petri Nets. In *7th International Conference on Modelling Techniques and Tools for Computer Performance Evaluation, Vienna (Austria)*, pages 321–334, 1994.

[7] G. Chiola, G. Bruno, and T. Demaria. Introducing a color formalism into Generalized Stochastic Petri Nets. In *Proceedings of the 9th International Workshop on Application and Theory of Petri Nets, Venice (Italy)*, pages 202–215, 1988.

[8] A. Attieh et al. Functional and temporal analysis of concurrent systems. Technical report, University of Cape Town, 1994.

[9] G. Berthelot et al. A syntax for the description of Petri Nets. *Petri Net Newsletter*, pages 4 – 15, 29 April 1988.

[10] R. Fehling. *Hierarchische Petrinetze.* PhD thesis, Universität Dortmund, Verlag Dr. Kovac, 1991.

[11] R. Fehling. A concept of hierarchical Petri Nets with building blocks. In G. Rozenberg, editor, *Advances in Petri Nets 1993*, LNCS 674, pages 148–168. Springer, Berlin, 1993.

[12] F. Feldbrugge. Petri net tools. *Lecture Notes in Computer Science, No 222*, pages 203–223, 1985.

[13] P. Huber, K. Jensen, and R.M. Shapiro. Hierarchies in Coloured Petri Nets. In G. Rozenberg, editor, *Advances in Petri Nets*, LNCS 483, pages 313–341. Springer, Berlin, 1990.

[14] K. Jensen. Coloured Petri Nets and the invariant method. *Mathematical Foundations on Computer Science, Lecture Notes in Computer Science*, 118:327–338, 1981.

[15] K. Jensen. *Coloured Petri Nets: Basic Concepts, Analysis Methods and Practical Use*. EATCS Monographs on Theoretical Computer Science, Vol. I, 1992.

[16] U. Krieger, B. Müller-Clostermann, and M. Sczittnick. Modelling and analysis of communication systems based on computational methods for markov chains. *IEEE Transactions on Communications: Special Issue on Modelling and Analysis of Telecommunication Systems*, 40(4):109–137, 1991.

[17] T. Murata. Petri Nets: Properties, analysis and applications. *Proceedings of IEEE*, 77(4):541–580, April 1989.

[18] A. Pagnoni. Stochastic nets and performance evaluation. In *Petri Nets: Central Models and Their Properties. Advances in Petri Nets*, pages 460–478. Lecture Notes in Computer Science, No. 254, 1986.

[19] C. A. Petri. *Kommunikation mit Automaten*. PhD thesis, Universität Bonn, 1962.

[20] A. Wikstrom. *Functional Programming using Standard ML*. Prentice Hall International, 1987.

A Complete APNN grammar

In the following we present the productions of a complete APNN for CGSPNs including hierarchical description and queued places. As throughout the report terminal symbols are given by lower-case letters, non-terminals by upper-case letters. For conciseness we use the APNN for CPNs with general ML-expressions as given in Sec. 3.2.1. Note that one can easily modify the APNN to support other "programming languages". `empty` denotes the empty string.

With this APNN a wide range of Petri Nets is covered. Possible extensions of the notation can be incorporated as demonstrated in this report.

```
NET  ::= empty
      |  \inputnet{ID}  NET
      |  \beginnet{ID}  ELEMENT   \endnet NET
      |  \beginnet{ID}  \like{ID} \endnet NET

ELEMENT    ::= empty
             | PLACE      ELEMENT
             | TRANSITION ELEMENT
             | ARC        ELEMENT
             | TYPEDEF    ELEMENT
             | \seeML{ FILENAME } ELEMENT
             | FUSION     ELEMENT

ID       ::= STRING
FILENAME ::= STRING
TYPEDEF  ::= \typedef{ID}{ COLOURSET }

FUSION ::= \fuse{ID}{TYPE}{ID IDLIST}
TYPE   ::= page  | inst | global
IDLIST ::= empty | '|' ID IDLIST

PLACE  ::= \place{ID}{ NAME INIT CAP COLOUR P_TYPE PORT EXIT }
        | \place{ID}{ \like{ID} }
NAME   ::= empty | \name{ STRING }
INIT   ::= empty | \init{ MULTISETEXPR }
CAP    ::= empty | \capacity{ MULTISETEXPR }
COLOUR ::= empty | \colour{ COLOURSET }
P_TYPE ::= empty
        |  \substitute{ID}
        |  \queue{immediate} SCHED \weight{ TUPLEEXPR }
        |  \queue{timed}     SCHED \weight{ TUPLEEXPR }
SCHED  ::= empty
        |  \sched{STRING} SCHED
        |  \noserver{ INTEGER } SCHED
        |  \rank{ INTEXPR } SCHED
        |  \prio{ INTEXPR } SCHED
PORT   ::= empty | \port{in} | port{out} | \port{io}
EXIT   ::= empty | \exit

TRANSITION ::= \transition{ID}{ NAME T_TYPE PRIO T_WEIGHT GUARD
               PORT EXIT }
            |  \transition{ID}{ \like{ID} }
T_TYPE     ::= empty | \substitute{ID} | \invoke{ID}
PRIO       ::= empty | \prio{ INTEXPR }
T_WEIGHT   ::= empty | \weight{ REALEXPR }
GUARD      ::= empty | \guard{ BOOLEXPR }
```

```
ARC        ::= \arc{ID}{ \from{ID} \to{ID} WEIGHT A_TYPE BIND }
WEIGHT     ::= empty | \weight{ MULTISETEXPR }
A_TYPE     ::= empty | \type{ordinary}  | \type{inhibitor}
BIND       ::= empty | \bind{ID} CONT \with{ID} CONT
CONT       ::= empty | \cont{ID} CONT

INTEXPR      ::= \like{ID} | ML-EXPRESSION
REALEXPR     ::= \like{ID} | ML-EXPRESSION
BOOLEXPR     ::= \like{ID} | ML-EXPRESSION
TUPLEEXPR    ::= \like{ID} | ML-EXPRESSION
MULTISETEXPR ::= \like{ID} | ML-EXPRESSION

COLOURSET  ::= \like{ID} | ML-DEFINITION
TYPEDEF    ::= \typedef{ID}{ ML-DEFINITION }

Start-symbol: NET
```