EPNML 1.1 - an XML format for Petri nets

J.M.E.M. van der Werf (jmw@petriweb.org)
R.D.J. Post (rpost@petriweb.org)
TU Eindhoven

21st June 2004

Abstract

This document defines EPNML 1.1, an XML format used for Petri nets in the web application *Petriweb* and the Petrinet editor *Yasper*. EPNML 1.1 is an application of PNML, a developing standard XML format for Petri nets: EPNML 1.1 documents are a particular type of PNML documents.[1]
EPNML is modified over time to incorporate corrections and extensions. The version described here is *EPNML 1.1*, first published in October, 2003; this specification has been modified several times since, due to errors found. At the time of writing we are already considering an updated version of the file format itself, which will be called EPNML 1.2; that slightly modified document format is not described here.

A formal, but incomplete, syntax definition is included.

1 Overview

This document is intended as the full and authoritative definition of a set of document formats used to interchange Petri nets [?] by the tools and utilities developed in the Petriweb project [?].

Petriweb is a web application to manage repositories of Petri nets. It stores Petri nets created and used by various tools for various purposes. Some of the tools are our own, such as the Yasper Petri net editor [?]. However, it is an explicit goal to be compatible with other Petri net tools.

For this reason, Petriweb is based on PNML, the Petri Net Markup Language, which is intended as a common and tool independent description of Petri nets. PNML is an application of XML, the eXtensible Markup Language [?]. Its specification is still in progress; an informal, preliminary definition can be found in [?].

Petri nets are added to Petriweb by uploading them in PNML format, and can be downloaded in PNML format.

PNML is intended as a universal exchange format for Petri nets: all types of Petri nets can in principle be expressed in PNML. What is more, the syntax of a particular type of Petri nets can be defined in PNML

[1] The name EPNML is poorly chosen as it can be mistaken to mean *Extended PNML*, when in fact it stands for *ExSpect PNML*, as it represents Petri net features that roughly corresponds to those of the ExSpect [?] process modelling tool.

using a so-called PNTD (Petri Net Type Definition). The Petriweb repository does not support arbitrary Petri nets with arbitrary features – although it may be extended to do so – and neither do our supporting tools. Therefore, we use PTNDs to provide exact definitions of particular sets of documents supported by our tools.

A PNTD is only a formal grammar, and cannot specify a document format in full detail: it cannot express some of the constraints that hold, and cannot say anything about rules of interpretation necessary to make interoperability between tools work in practice. This document adds this information and thereby serves as the full specification.

Having a precise definition of the interchange format used by tools, even while the tools are developing capabilities, turns out to be very helpful in the development process. It implies that, over time, we have to make changes to the document format itself to correct problems and add new capabilities. Likewise, we have to change their specifications, sometimes to reflect changes in the document formats used, sometimes to reflect changes in PNML, sometimes to simply make corrections in the specification itself. The first version of this document, issued in June, 2003, described EPNML 1.0, which wasn't a proper PNML application, and is no longer supported by our tools; the present version describes EPNML 1.1 (defined in October, 2003); a future version will describe EPNML 1.2.

This document is organised as follows. First, an informal, but precise, definition is given of the document format, with examples, discussing, global document structure (section 2, section 3), places (section 4), transitions (section 5) and their connections (section 6), subnet structure (section 7), and the elements that recur within all of these (section 8). It is intended as a self-sufficient definition, for readers who are looking for an informal introduction.

A full example document is given in section 9.

In section 10, we supply formal specifications, as syntax validation software need them.

Finally, we summarise the specific properties of EPNML documents, compared to PNML documents in general, in section 11.

2 <pnml>: Document structure and general properties

An XML document always has a single top element; in the case of a PNML document, it is `<pnml>`. All nets – a PNML document can contain several – are children of this top element.[2]

In an EPNML 1.1 document, the order in which elements appear is free: it is valid, and does not change the document's meaning, to arbitrarily reorder the children of any element in the document. This slightly complicates parsing: cross-references (`IDREF` attributes) can point forward, and children aren't sorted by element type. The latter is also a problem for syntax definitions in languages other than RELAX NG.[3]

In accordance with the (developing) PNML specification (see [?]), every object (i. e. `<net>`, `<page>`, `<transition>`, `<place>` and `<arc>`) in an EPNML 1.1 can have a `<toolspecific>` element with arbitrary content. This is used by some of our tools to record information that other tools are not expected to understand.

[2] PNML modules [?] are not used in EPNML 1.1.

[3] The next version of EPNML will have provisions to cope with this.

3 <net>: Petri Nets

A Petri net is defined with the element <net>. This element has the following attributes:

id Unique identifier (see section 8.4); allows the net to be referenced from other nets, although the elements defined in EPNML 1.1 themselves do not support such references.

type The Petri net type: http://www.petriweb.org/specs/epnml11.[4]

The net's places, transitions, arcs and subnets are children of the <net> element;[5] and it may further contain the following elements:

graphics See section 8.1. This element is optional; it is used if an overview page exists with an item for all nets in the document. It specifies the position and optionally the size of the item within the overview.[6]

name Optional; see section 8.2.

description Optional; see section 8.3.

4 <place>: Places

A place is defined with the element <place>. This element has the following attributes:

id Unique identifier (see section 8.4); allows the place to be referenced. Such references are made in <referencePlace>s and <arc>s.

A <place> may further contain the following elements:

graphics Optional; see section 8.1.

name Optional; see section 8.2.

description Optional; see section 8.3.

initialMarking Optional. The <initialMarking> element provides the initial marking of a place. Its required subelement <text> specifies the value; for uncoloured nets, it is a nonnegative decimal number. This element may have a subelement <graphics> to specify a relative position for displaying the content in a diagram.

[4] The purpose of this attribute, mandated by the PNML specification, isn't quite clear. Considering that its value is a URI, it is expected to somehow point at a syntax definition for an individual net. If PNML is as compatible as it wants to be, different Petri net type definitions always differ in syntax, so the PNTD(s) a net conforms to are already apparent from its syntax; hence, the value indicated here can be derived automatically, and an explicit value can be in error.
Petriweb related tools do not all output this value, and all ignore it on input documents, validating the whole document against formal syntax definitions instead.

[5] Unlike a <page>, a <net> cannot contain any reference places.

[6] Note that its use is deprecated: our tools do not support it, and the next version of EPNML will not define it.

type In EPNML, a place can either be a `store` or a `channel`. A channel is a regular Petri net place, while a store serves as a data storage component, a concept also used in the *ExSpect* [?] process modelling software. Optional; a place is assumed to be a channel. The element `<text>`, which is required for `<type>`, holds the type information.

Below is an example of the definition of a place.

```
<place id="p1">
  <graphics>
    <position x="10" y="20" />
    <dimension x="20" y="20"/>
  </graphics>
  <name>
    <text>employee available</text>
  </name>
  <description>
    <text>A TU Eindhoven employee is ready to edit the specification.</text>
  </description>
  <initialMarking>
    <text>2</text>
  </initialMarking>
  <type>
    <text>channel</text>
  </type>
</place>
```

5 `<transition>`: Transitions

A transition is defined by the element `<transition>`. This element has the following attributes:

id Unique identifier (see section 8.4); allows the transition to be referenced. Such references are made in `<arc>`s and `<referencePlace>`s.

A `<transition>` may further contain the following elements:

graphics See section 8.1. See section 4 for more information about its use.

description Optional; see section 8.3.

transformation The `<transformation>` element specifies the token value transformation performed by the transition, as used in ExSpect ([?]. This element is optional. The transfomation is the content of the required subelement `<text>`. Optionally, information about relative positioning can be added with the element `<graphics>`.

type In EPNML, a transition can be either an `AND` or an `XOR`. An AND transition has normal Petri net semantics, while an XOR transition transfers a token in one of its input places nondeterministically to one of its output places.[7] This element is optional; by default, a transition is assumed to be an `AND`. The element `<text>`, which is required for `<type>`, holds the type information.

Below is an example of a transition.

```
<transition id="t1">
  <graphics>
    <position x="-30" y="0"/>
  </graphics>
  <name>
    <text>employee edits spec</text>
  </name>
  <description>
    <text>A TU employee is editing the EPNML specification.</text>
  </description>
  <transformation>
    <text>lines := lines + added(spec)</text>
  </transformation>
  <type>
    <text>AND</text>
  </type>
</transition>
```

6 `<arc>`: Arcs

An arc is defined by the element `<arc>`. This element has the following attributes:

id Unique identifier (see section 8.4); allows the arcs to be referenced, although EPNML itself doesn't support such references.

source Required; the `id` of a `<transition>`, `<place>`, or `referencePlace` (see section 8.4) within the same `<net>` or `<page>` as the `<arc>` itself. [8]

target See the `source` attribute.

Either `source` or `target`, but not both, must be a `<transition>`. An (unfinished) Petri net with unconnected or wrongly connected arcs cannot be represented in EPNML.

An `<arc>` may further contain the following elements:

[7] It can be regarded a shorthand for a subnet consisting of a central place surrounded by classical transitions, each with one input and one output.

[8] It cannot be a `<page>`, and it cannot cross any page or net borders.

graphics See section 8.1. For an `<arc>`, this element can occur 0, 1, or multiple times; it specifies intermediate support points for layout purposes. It is up to the drawing tool to interpret the values. Start and end points are not specified; the layout algorithms must be deduce them from `source` and `target`.

name See section 8.2.

description See section 8.3.

inscription Optional; its value must be in the subelement `<text>`. A subelement `<graphics>` may be added with relative positioning for layout purposes. This attribute is present to support coloured Petri nets as used in Design/CPN [?], CPN Tools [?] and other tools.

type Optional; not in the PNML definition. Defines the operation an arc performs on the place it is connected to. Possible values are

- `inhibitor` if that place is a `channel`;[9]
- `C`, `R`, `U` or `D` if it is a `store`; not specifying the type in that case is equivalent to specifying `R`.[10]

The values of `type` have the following meanings:

inhibitor The transition is prevented from firing if the channel is nonempty.

biflow Stands for a pair of opposite, untyped arcs.

C The transition inserts a datum into the store.

R The transition replaces a datum in the store.

U The transition updates a datum in the store.

D The transition deletes a datum from the store.

An untyped arc connecting directly or indirectly to a channel represents a standard, directed connection; no specifiable type is equivalent to this.

Below an example of an arc; `t1` and `st1` are assumed to be the `id` attributes of respectively a transition and a place with type `store`.

```
<arc id="a1" source="t1" target="st1">
  <inscription>
    <graphics>
      <offset x="20" y="0" />
    </graphics>
    <text>1</text>
  </inscription>
  <graphics>
    <position x="100" y="100" />
  </graphics>
```

[9] Some of our tools also support the value `biflow` here, with the obvious meaning; the next version of EPNML will make this official.
[10] The next version of EPNML will support combinations of these values.

```
  <type>
    <text>U</text>
  </type>
</arc>
```

7 Subnets

Nets can contain subnets. Different variants of Petri nets support different notions of subnets; one difference between them is how the subnet can be connected with its environment.

EPNML supports subnets as used at TU Eindhoven in education and research (see [?]), and in the ExSpect software tool (see [?]). These subnets connect to their environment by having references to places in the context. This is expressed with the `<referencePlace>` element, that PNML provides for this purpose.

In many Petri net tools, including ExSpect[?], it is possible for a subnet to have a single definition with multiple uses, such that changes to the definition apply everywhere. Our present applications do not support this, and neither does EPNML 1.1. A future version of EPNML will support it, using PNML `<module>`s[?].

7.1 <page>: Subnet definitions

EPNML uses PNML's `<page>` element to represent a subnet. It has the following attributes:

id Unique identifier (see section 8.4); allows the subnet to be referenced, although no elements in EPNML 1.1 support such references, not even `<arc>` or `<referencePlace>`.)[11]

A `<page>` may further contain the following elements:

graphics See section 8.1.

description Optional; see section 8.3.

type Optional. The transition type; see section 5 for details on the syntax allowed. There is a difference in interpretation: whereas for transitions, an omitted type is equivalent to a type of `AND`, for subnets, an omitted type means the subnet is neither an `XOR` nor an `AND`.

A `<page>` is like a `<net>` in every way, except that its `type` has different values and a different meaning, that it occurs within a `<net>` or `<page>`, and that it may have `<referencePlace>`s (see section 3).

Positioning of elements within a page is relative to the page. This allows subnets to be moved to different contexts, or copied, without any change to the information specified within.

[11] An `<arc>` is used to connect `<transition>`s, but not `<page>`s, while a `<referencePlace>`'s `ref` attribute connects pages, implicitly, by connecting their member `<referencePlace>`s. Note that this makes it inmpossible to specify support points for connections between subnets and places.

7.2 `<referencePlace>`: A subnet's interface with the environment.

PNML allows places and transitions to serve as placeholders for other places or transitions. EPNML only uses this mechanism for places, and only allows references from inside a subnet to the direct context it appears in. This models the use of *pin*s in ExSpect.

The PNML element used for this purpose is named `<referencePlace>`. It has the following attributes:

id Unique identifier (see section 8.4); in EPNML, `<arc>`s and `<referencePlace>`s can make such references.

ref The `id` of a place or reference place that is a sibling of the page it occurs in.[12] Required, as PNML mandates.[13]

There is no `type` attribute.[14]

A `<referencePlace>` may further contain the following elements:

graphics See section 8.1.

name Optional; see section 8.2.

description Optional; see section 8.3.

An example of a reference place, which refers to a (reference)place with id `p1`:

```
<referencePlace id="rp1" ref="p1">
  <name>
    <text>Pin</text>
  </name>
  <description>
    <text>description of the pin</text>
  </description>
  <graphics>
    <position x="10" y="5" />
  </graphics>
```

8 Common elements

Here we describe the elements common to (almost) all elements of PNML.

[12] This implies that chains of references cannot form cycles.

[13] Our tools actually allow this attribute to be omitted; this allows an (unfinished) net with unconnected pins to be saved. The next EPNML version will support this.

[14] Some of our applications constrain pins to have at most one arc. The next EPNML version will support this constraint.

8.1 <graphics>: Graphics

EPNML uses the <graphics> element to specify position and size of an element for layout purposes. Unlike some other PNML applications, it does not provide any other graphical attributes.

The <graphics> element is optional.[15]

It can have the following subelements: <position>, <offset>, and <dimension>.

These elements, when they occur, have two required attributes:

x The x coordinate of the element.

y The y coordinate of the element.

They do not have any further content.

The existing PNML documentation is not altogether clear on how to interpret these values. Using the PNK editor[?], a PNML reading and writing tool, and the requirements for our own applications as references, we have arrived at the following interpretation, which is assumed for EPNML.

- Coordinate values denote numbers of screen pixels.
- Sizes are absolute, i.e. relative to the including <net>.
- Positions are always relative to the position of the nearest containing element.
- The x coordinate increases to the right; the y coordinate increases downward; the origin is the containing element's upper left hand corner.
- The <position> subelement is used for <place>s, <transition>s, and <page>s ("nodes", in PNML parlance). It is required. It is not used anywhere else.
- The <offset> subelement is used for node attributes that may be displayed in the diagram ("attributes", in PNML parlance); in EPNML 1.1, they are <name>s, <description>s, <transformation>s, <initialMarking>s, and <inscription>s. It is optional; when used, it is relative to the object to which it refers.
- The <dimension> element is used to denote the size of an element within its containing element, even for <page>s and <net>s.

  There is no way in EPNML 1.1 to specify the dimension of the sheet onto which the contents of a <net> or <page> are displayed; PNML interpreters are assumed to deduce one. An obvious method is to use a standard dimension, or to use the sum of the minimal and maximal positions of its containing elements.

  It is always optional, and cannot assume negative values. To make alignment of shapes work correctly, the origin of an element is its center, as specified by PNML (see [?]).

[15] Most of the tools for Petriweb however, and many other tools that display Petri nets as diagrams, require it for layout purposes (see [?]).

An example:

```
<place id="p1">
<graphics>
  <position x="100" y="100" />
  <dimension x="10" y="10" />
</graphics>
  <initialMarking>
    <text>8</text>
    <graphics>
      <offset x="-10" y="20" />
    </graphics>
  </initialMarking>
</place>
```

8.2 <name>: Name

The element <name> contains the name of the element. A name is optional and not necessarily unique – it is legal for two different elements to carry the same name. It has no attributes and the name is inserted in the the element <text>. Optionally, graphical information can be added with <graphics>. Below an example of a <name> element.

```
<name>
  <graphics>
    <offset x="-10" y="20" />
  </graphics>
  <text>Name of the element</text>
</name>
```

8.3 <description>: Description

The element <description> contains the description of an element. It is optional and not necessarily unique. This element doesn't have attributes and the value of the description is inserted into the element <text>. Optionally, graphical information can be added with the element <graphics>. Below is an example of a description.

```
<description>
  <graphics>
    <offset x="10" y="20" />
  </graphics>
  <text>Description of the element</text>
</description>
```

8.4 `id`: Identifiers and references

Every `<net>`, `<page>`, `<transition>`, `<place>`, `<referencePlace>` and `<arc>` must have an `id` attribute; its presence is required by the PNML specification (see [?]). It is an *ID* attribute in the sense of a DTD; that is, its value has a certain format (see [?]), and no two elements within the same document can have the same `<id>` value.

The `source` and `target` attributes of `<arc>`s and the `ref` attribute of `<referencePlace>`s are *IDREF* attributes: their values must occur as the value of some `id` attribute in the document.

This is the standard way provided by XML to establish relationships between elements other than the hierarchical parent-child relationship.

9 Example document

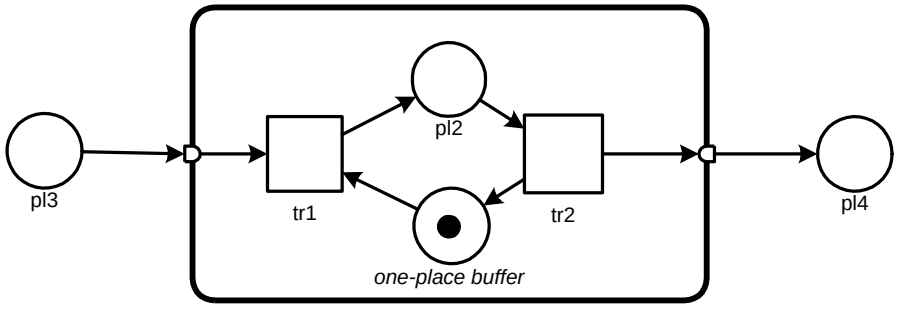

Figure 1: A Petri net representing a one-place buffer

In this section we describe an example for the representation of a net and subnet in PNML. Our example is the one-place buffer diagrammed in figure 1.[16]

A valid EPNML 1.1 representation of this net:

```
<?xml version="1.0"?>
<pnml>
    <net type="EPNML11" id="n3">
        <name>
            <text>bufferdef.vsd/Page-1</text>
        </name>
        <graphics>
            <dimension x="496" y="702"/>
            <position x="0" y="0"/>
        </graphics>
        <page id="n4">
            <name>
                <text>one-place buffer (sn1)</text>
            </name>
            <graphics>
                <position x="189" y="76"/>
                <dimension x="165" y="94"/>
            </graphics>
            <place id="pl5">
                <graphics>
                    <position x="83" y="70"/>
                    <dimension x="24" y="24"/>
                </graphics>
                <type>
                    <text>channel</text>
                </type>
                <initialMarking>
                    <text>1</text>
                </initialMarking>
            </place>
            <transition id="tr3">
                <name>
                    <text>tr1</text>
                </name>
                <graphics>
                    <position x="36" y="47"/>
                    <dimension x="24" y="24"/>
                </graphics>
                <type>
                    <text>AND</text>
                </type>
```

[16] Note that `id`s normally do not appear in diagrams; they are displayed here for reference purposes only.

```
</transition>
<place id="pl6">
    <name>
        <text>pl2</text>
    </name>
    <graphics>
        <position x="82" y="23"/>
        <dimension x="24" y="24"/>
    </graphics>
    <type>
        <text>channel</text>
    </type>
</place>
<transition id="tr4">
    <name>
        <text>tr2</text>
    </name>
    <graphics>
        <position x="119" y="47"/>
        <dimension x="25" y="25"/>
    </graphics>
    <type>
        <text>AND</text>
    </type>
</transition>
<referencePlace id="pi3">
    <graphics>
        <position x="0" y="47"/>
        <dimension x="5" y="5"/>
    </graphics>
</referencePlace>
<referencePlace id="pi4">
    <graphics>
        <position x="165" y="47"/>
        <dimension x="5" y="5"/>
    </graphics>
    </referencePlace>
    <arc id="ar7" target="tr3" source="pl5"/>
    <arc id="ar8" source="tr3" target="pl6"/>
    <arc id="ar9" source="tr4" target="pl5"/>
    <arc id="ar10" target="tr4" source="pl6"/>
    <arc id="ar11" target="tr3" source="pi3"/>
    <arc id="ar12" source="tr4" target="pi4"/>
</page>
<place id="pl7">
    <name>
        <text>pl3</text>
    </name>
    <graphics>
        <position x="59" y="75"/>
        <dimension x="24" y="24"/>
    </graphics>
    <type>
        <text>channel</text>
    </type>
</place>
<place id="pl8">
    <name>
        <text>pl4</text>
    </name>
    <graphics>
        <position x="319" y="76"/>
        <dimension x="24" y="24"/>
    </graphics>
    <type>
        <text>channel</text>
    </type>
</place>
</net>
</pnml>
```

10 Formal syntax definitions for EPNML 1.1

10.1 XML syntax definition formalisms

The syntax of EPNML 1.1 is the set of XML documents that are valid EPNML 1.1 documents. The informal description above isn't suitable for an automatic syntax checker; we need a formal one. Fortunately, there are several popular and standardized formalisms to describe sets of XML documents, the most important of which are:

- the DTD (Document Type Definition) [?];
- XML Schema [?], a standard developed from the viewpoint of database schema and data structure definition;
- RELAX NG [?], a standard developed from the viewpoint of formal language theory (grammar);
- Schematron [?], a simple constraint specification language based on XPath [?].

RELAX NG is used to describe the syntax of PNML. In particular, RELAX NG's extension mechanism, which allows a specification to be based on another, is used to describe Petri Net Type Definitions (PNTDs) as extensions to other PNTDs and sets of common base definitions.

10.2 RELAX NG definitions for EPNML 1.1

The three given RELAX NG specifications do this for EPNML 1.1, thereby proving it a regular application of PNML in all respects. Their structure is as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
  for SGML validation, use this first line instead:
  <!DOCTYPE grammar PUBLIC "-//thaiopensource//DTD RNG 20010705//EN" "">
 -->

<grammar
  xmlns="http://relaxng.org/ns/structure/1.0"
  xmlns:a="http://relaxng.org/ns/compatibility/annotations/1.0">

  <a:documentation>
    EPNML 1.1 label conventions - RELAX NG implementation
    version: $Id: conv.rng,v 1.5 2004/06/11 12:44:46 pnml Exp $
    (c) 2003, Technische Universiteit Eindhoven
        JMEM van der Werf, jmw@petriweb.org
        some editing by R. Post, rp@petriweb.org
     human-readable documentation:
        EPNML 1.1 - an XML format for Petri nets
        http://www.petriweb.org/specs/epnml-1.1/pnmldef.pdf
  </a:documentation>

  <a:documentation>
    We use the standard annotations and conventions of PNML,
    by including them directly.
  </a:documentation>
  <include href="http://www.informatik.hu-berlin.de/top/pnml/conv.rng"/>

  <a:documentation>
    We add netgraphics. In EPNML a net can have positions, this because we see
    an PNML document as a container of nets.
  </a:documentation>
  <define name="netgraphics.content" combine="interleave">
    <optional>
      <ref name="nodegraphics.content"/>
    </optional>
  </define>

  <define name="common.labels">
    <a:documentation>
      We use some different common labels:
        *) Name
        *) Description
        *) Graphics. But this is already in basicPNML.
      Everything already has a name, because it's defined in the conv.rng doc.
      See [1], section 9
    </a:documentation>
    <interleave>
      <optional>
        <ref name="Name"/>
      </optional>
      <optional>
        <ref name="Description"/>
      </optional>
    </interleave>
  </define>


  <define name="Description">
    <a:documentation>
      The name of an element, has element text in it, just like name.
      See [1], section 9.3
    </a:documentation>
    <element name="description">
      <ref name="simpletextlabel.content"/>
    </element>
  </define>

  <define name="place.type">
    <a:documentation>
      The type of a place is either a channel (common) or a store.
      See [1], section 5
    </a:documentation>
    <element name="type">
      <element name="text">
        <choice>
          <value>channel</value>
          <value>store</value>
        </choice>
      </element>
    </element>
  </define>


  <define name="transition.type">
    <a:documentation>
      The type of a transition is AND (common) or XOR.
      See [1], section 6
    </a:documentation>
    <element name="type">
      <element name="text">
        <choice>
          <value>AND</value>
          <value>XOR</value>
        </choice>
      </element>
    </element>
  </define>


  <define name="Transformation">
    <a:documentation>
      A transition also has a transformation: it defines what a transition does.
    </a:documentation>
    <element name="transformation">
      <ref name="simpletextlabel.content"/>
    </element>
  </define>


  <define name="arc.type">
    <a:documentation>
      An arc has a type with some more possibilities:
      See [1], section 7
    </a:documentation>
    <element name="type">
      <element name="text">
        <choice>
          <value>C</value>
          <value>R</value>
          <value>U</value>
          <value>D</value>
          <value>inhibitor</value>
        </choice>
      </element>
    </element>
  </define>


  <define name="ArcInscription">
    <a:documentation>
      The inscription of an arc (not implemented in PNML)
      See [1], section 7
    </a:documentation>
    <element name="inscription">
      <ref name="simpletextlabel.content"/>
    </element>
  </define>
</grammar>
```

Figure 2: conv.rng, label conventions for EPNML

figure 2 defines the labels (properties) particular to EPNML nets;

figure 3 defines EPNML nets without subnets;

figure 4 defines EPNML nets with subnets.

In the next EPNML version, many more optional features will be included in this way.

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
  for SGML validation, use this first line instead:
  <!DOCTYPE grammar PUBLIC "-//thaiopensource//DTD RNG 20010705//EN" "">
 -->

<grammar
  xmlns="http://relaxng.org/ns/structure/1.0"
  xmlns:a="http://relaxng.org/ns/compatibility/annotations/1.0">

  <a:documentation>
    EPNML 1.1 without subnets - RELAX NG implementation
    version: $Id: basic.rng,v 1.9 2004/06/11 12:44:46 pnml Exp $
    (c) 2003-2004, Technische Universiteit Eindhoven
        JMEM van der Werf, jmw@petriweb.org
        some editing by R. Post, rp@petriweb.org
     human-readable documentation:
        EPNML 1.1 - an XML format for Petri nets
        http://www.petriweb.org/specs/epnml-1.1/pnmldef.pdf
  </a:documentation>

  <a:documentation>
    We use basic PNML, with some extensions. So we include this document:
  </a:documentation>
  <include href="http://www.informatik.hu-berlin.de/top/pnml/1.3.2/basicPNML.rng"/>

  <define name="anyElement" combine="choice">
    <a:documentation>
      Slightly changed from the included definition
      to avoid data typing conflicts in attributes.
    </a:documentation>
      <element>
        <anyName />
        <zeroOrMore>
          <choice>
            <attribute>
              <anyName>
                <except>
                  <nsName/>
                </except>
              </anyName>
            </attribute>
            <text />
            <ref name="anyElement"/>
          </choice>
        </zeroOrMore>
      </element>
    </define>

  <a:documentation>
    Use our own conventions:
  </a:documentation>
  <include href="conv.rng"/>

  <define name="nettype.uri" combine="choice">
    <a:documentation>
      We define our own net type: epnml-1.1 -> EPNML 1.1
    </a:documentation>
    <value>http://www.petriweb.org/specs/epnml-1.1</value>
  </define>

  <define name="net.labels" combine="interleave">
    <a:documentation>
      A net has common attributes.
      See [1], section 4
    </a:documentation>
    <optional>
      <ref name="common.labels"/>
    </optional>
  </define>


  <define name="place.labels" combine="interleave">
    <a:documentation>
      A place has the common attributes and a type.
      See [1], section 5
    </a:documentation>
    <interleave>
      <optional>
        <ref name="common.labels"/>
      </optional>
      <optional>
        <ref name="place.type"/>
      </optional>
      <optional>
        <ref name="PTMarking"/>
      </optional>
    </interleave>
  </define>


  <define name="transition.labels" combine="interleave">
    <a:documentation>
      A transition has common attributes, a type and a transformation.
      See [1], section 6
    </a:documentation>
    <interleave>
      <optional>
        <ref name="common.labels"/>
      </optional>
      <optional>
        <ref name="transition.type"/>
      </optional>
      <optional>
        <ref name="Transformation"/>
      </optional>
    </interleave>
  </define>


  <define name="arc.labels" combine="interleave">
    <a:documentation>
      An arc has common elements, a type and an inscription.
      See [1], section 7
    </a:documentation>
    <interleave>
      <optional>
        <ref name="common.labels"/>
      </optional>
      <optional>
        <ref name="arc.type"/>
      </optional>
      <optional>
        <ref name="ArcInscription"/>
      </optional>
    </interleave>
  </define>
</grammar>
```

Figure 3: basic.rng, basic EPNML (without subnets)

10.3 Additional constraints

No PNTD defined in RELAX NG can exactly describe the syntax of EPNML 1.1: certain constraints cannot be expressed in RELAX NG.[17]

For EPNML 1.1, they are the following:

- every arc must connect a transition with a (reference) place within the same net or page (cf. section 6)

[17] XML Schema and DTDs suffer from the same problem.
Note that we do not follow the practice, common in computer science, to *define* a specification in DTDs, XML Schema, RELAX NG or any other specification language as "the syntax" of the language, and call all syntactic restrictions that happen not to be expressed in this specification, often because they cannot be expressed in the language at all, "semantics".

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
  for SGML validation, use this first line instead:
  <!DOCTYPE grammar PUBLIC "-//thaiopensource//DTD RNG 20010705//EN" "">
 -->

<grammar
  xmlns="http://relaxng.org/ns/structure/1.0"
  xmlns:a="http://relaxng.org/ns/compatibility/annotations/1.0"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema-datatypes"
  xmlns:rdt="http://relaxng.org/ns/compatibility/datatypes/1.0"
  datatypeLibrary="http://relaxng.org/ns/compatibility/datatypes/1.0">

  <a:documentation>
    EPNML 1.1 with subnets - RELAX NG implementation
    version: $Id: structured.rng,v 1.7 2004/06/11 12:44:46 pnml Exp $
    (c) 2003-2004, Technische Universiteit Eindhoven
        by JMEM van der Werf, jmw@petriweb.org
        some editing by Reinier Post, rp@petriweb.org
    human-readable documentation:
        EPNML 1.1 - an XML format for Petri nets
        http://www.petriweb.org/specs/epnml-1.1/pnmldef.pdf
  </a:documentation>


  <a:documentation>
      We use basic EPNML, with some extensions. So we include this document:
  </a:documentation>
  <!-- PNML extended to EPNML -->
  <include href="basic.rng"/>
  <define name="net.content" combine="choice">
    <a:documentation>
        Unlike in general structured PNML, we do not support
        reference transitions, and allow referencePlaces only within pages.
        See [1], section 4 and 8
    </a:documentation>
    <zeroOrMore>
      <element name="page">
        <ref name="page.content"/>
      </element>
    </zeroOrMore>

<!--    <a:documentation>
      Because of translations that petriweb makes, (page -> net), also
      referenceplaces and types must be allowed in nets.
    </a:documentation>
    <optional>
      <ref name="transition.type"/>
    </optional>
    <zeroOrMore>
      <ref name="refplace.element"/>
    </zeroOrMore>-->
  </define>


  <define name="page.content">
    <a:documentation>
      An EPNML 1.1 page is a "refined transition".
      It is like a net, except that it is within a net or page,
      can have reference places, and a transition type.
      See [1], section 8
    </a:documentation>
    <interleave>
      <ref name="common.labels"/>
      <zeroOrMore>
        <ref name="net.content"/>
      </zeroOrMore>
      <zeroOrMore>
        <ref name="refplace.element"/>
      </zeroOrMore>
      <ref name="node.content"/>
      <optional>
        <ref name="transition.type"/>
      </optional>
    </interleave>
  </define>

  <define name="refplace.element">
    <element name="referencePlace">
      <ref name="refplace.content"/>
    </element>
  </define>


  <define name="refplace.content">
    <a:documentation>
        A reference place is a placeholder to a place.
        See [1], section 8.2
    </a:documentation>
    <attribute name="ref">
      <data type="IDREF"/>
    </attribute>
    <interleave>
      <ref name="node.content"/>
      <ref name="common.labels"/>
    </interleave>
  </define>

</grammar>
```

Figure 4: structured.rng, structured EPNML (with subnets)

- the reference of every pin (`<referencePlace>`) must be a sibling (reference) place of its containing page (cf. section 7.2)
- arc type and place type must match (even through intermediate pins) (cf. section 7.2)
- (not required): every pin has at most one arc (cf. section 7.2)

None of them can be expressed in XML Schema or DTDs, either, but they can be partly expressed in the constraint specification language Schematron [?]: see figure 5.[18]

11 EPNML 1.1 as an application of PNML

To summarize, EPNML 1.1 documents are very similar to structured place/transition nets as defined in [?], but subject to the following restrictions:

[18] Note that all of them constrain the possible values of `IDREF` attributes, and would be expressible if support for this relation were added to existing XML syntax definition languages, analogous to what is already there for the parent-child relationship [?].

```
<?xml version="1.0" encoding="UTF-8"?>
<grammar xmlns="http://relaxng.org/ns/structure/1.0" xmlns:xsd="http://www.w3.org/2001/XMLSchema-datatypes" xmlns:a="http://relaxng.org/ns/compatibility/annotations/1.0" xmlns:rdt="
  <a:documentation>
some EPNML constraints in Schematron
RELAX NG implementation
version: $Revision: 1.2 $ $Date: 2004/06/11 12:44:46 $
(c) 2004, Technische Universiteit Eindhoven
by Reinier Post, rpost@petriweb.org
</a:documentation>
<!-- the RELAX-NG grammar is minimal: just enough to place the Schematron rules -->
  <start>
    <element name="pnml">
      <ref name="any.content"/>
    </element>
  </start>
  <define name="any.content">
    <zeroOrMore>
      <choice>
        <ref name="any.element"/>
        <text/>
      </choice>
    </zeroOrMore>
  </define>
  <define xmlns:stn="http://www.ascc.net/xml/schematron" name="any.element">
    <element>
      <anyName/>
      <zeroOrMore>
        <attribute>
          <anyName/>
        </attribute>
      </zeroOrMore>
      <zeroOrMore>
      <ref name="any.content"/>
      <stn:assert test="name() != 'arc' or @source = ../*[name() = 'place' or name() = 'transition' or
      </stn:assert>
      <stn:assert test="name() != 'arc' or @target = ../*[name() = 'place' or name() = 'transition' or
      </stn:assert>
      <stn:assert test="name() != 'referencePlace' or @ref = ../../*[name() = 'place' or name() = 'refe
 must be a sibling (reference) place of its containing page.
      </stn:assert>
    <!--
      this assertion isn't complete (doesn't follow references)
      also, it isn't correct (rejects arcs it should accept
      <stn:assert test="name() != 'arc' or @source=../refPlace or ./type/text/text() != 'C' or ../place
      </stn:assert>
    -->
      </element>
    </define>
</grammar>
```

Figure 5: constraints.rng, some EPNML 1.1 constraints expressed in Schematron

- `<referenceTransition>`s do not exist
- the `IDREF` attributes of `<referencePlace>`s and `<arc>`s are severely restricted in what they can point to (section 10.3)

and wih the following additional PNML labels, all optional:

- every object can carry a `<description>`
- a transition can have a `<transformation>`
- `<place>`s, `<transformation>`s and `<arc>`s can all have a `type`, to indicate kinds of elements that do not occur in standard Petri nets, such as inhibitor arcs and access to stores