The Petri Net Markup Language

Matthias Jüngel[*], Ekkart Kindler, and Michael Weber[*]
Humboldt-Universität zu Berlin
juengel|kindler|mweber@informatik.hu-berlin.de

31st August 2000

1 Introduction

At the 'Meeting on XML/SGML based Interchange Formats for Petri Nets' [BBK+00] held in Aarhus in June 2000, different aspects of interchange formats for Petri nets were discussed, requirements were identified, and several interchange formats were proposed. Here, we present an interchange format for Petri nets that is based on this discussion and on our preliminary proposal [JKW00]. We call it the *Petri Net Markup Language* (*PNML*). The proposed format is quite basic, but it is open for future extensions.

In this paper, we present the concepts and terminology of the interchange format as well as its syntax, which is based on XML. It should provide a starting point for the development of a standard interchange format for Petri nets.

2 Concepts and terminology

Before introducing the syntax of the interchange format, we briefly discuss its basic concepts and terminology, which is independent of XML.

Petri nets and objects A file that meets the requirements of the interchange format is called a *Petri net file*; it may contain several *Petri nets*. Each Petri net consists of *objects*, where the objects, basically, represent the graph structure of the Petri net. Thus, an object is a *place*, a *transition*, or an *arc*. For structuring a Petri net, there are three other kinds of objects, which will be explained later in this section: *pages*, *reference places*, and *reference transitions*. Each object within a Petri net file has a unique *identifier*, which can be used to refer to this object.

For convenience, we call places, transitions, reference places, and reference transitions *nodes*, and we call a reference place and a reference transition a *reference node*.

[*]Supported by the Deutsche Forschungsgemeinschaft within the project "Petri Net Technology"

In: S. Philippi (Ed.): 7. Workshop Algorithmen und Werkzeuge für Petrinetze.
Fachberichte Informatik 7/2000, Universität Koblenz-Landau
http://www.informatik.hu-berlin.de/top/pnml/download/JKW_PNML.ps

Labels In order to assign further meaning to an object, each object may have some labels. Typically, a label represents the name of a node, the marking of a place, the guard of a transition, or the inscription of an arc. The legal labels—and the legal combinations of labels—of an object are defined by the *type* of the Petri net, which will be defined later in this section. In addition, the Petri net itself may have some labels. For example, the declaration of functions and variables that are used in the arc-inscriptions could be the labels of a Petri net.

We distinguish between two kinds of labels: *annotations* and *attributes*. Typically, an annotation is a label with an infinite domain of legal values. For example, names, markings, arc-inscriptions, and transition guards are annotations. An attribute is a label with a finite (and small) domain of legal values. For examples, an arc-type could be a label of an arc with domain: `normal`, `read`, `inhibitor`, `reset` (and maybe some more). Another example are attributes for classifying the nodes of a net as proposed by Mailund and Mortensen [MM00]. Besides this pragmatic difference, annotations have graphical information whereas attributes do not have graphical information[1].

Graphical information Each object and each annotation is equipped with some graphical information. For a node, this information is its position; for an arc, it is a list of positions that defines intermediate points of this arc. For an annotation, the graphical information is its relative position with respect to the corresponding object[2]. Absolute as well as relative positions refer to the *reference point* of an object or of an annotation respectively. By default, the reference point is the middle of the graphical representation for an object; it is the lower left point of the graphical representation for an annotation. For an arc, the reference point is the middle of the first segment of the arc. Future extensions might allow us to define the reference point of objects and annotations explicitly. All positions refer to (x,y)-coordinates, where the x-axis runs from left to right and the y-axis runs from top to bottom; but, we do not fix a unit[3].

Pages and reference nodes A Petri net can be structured by the help of *pages*. A page is an object that may consist of other objects—it may consist even of further pages. An arc, however, may connect nodes on the same page only. In order to connect Petri net nodes on different pages, we can use *reference nodes*: A reference node refers to any node of the Petri net—anywhere in the net. We require only that there are no cyclic references; this guarantees that, in the end, each reference node refers to exactly one place or exactly one transition of the Petri net. A reference node is only a representative for this node. Reference nodes may have labels. But, these labels do not have much meaning. Concerning the semantics of the net, the reference node inherits the labels from the node it refers to. This way, it is always possible to flatten the corresponding net without knowing the meaning of labels at all. Flattening means to merge each reference node to the node it refers to (directly or indirectly) and to ignore the label of the reference nodes.

[1] Of course, the attribute arc-type with value `inhibitor` may affect the appearance of the corresponding arc. But, this effect is in the meaning of the attribute itself and does not come from some extra graphical information in the attribute.

[2] If it is an annotation of the net itself, the position is absolute.

[3] Size of objects and labels as well as units could be included in a future version.

In: S. Philippi (Ed.): 7. Workshop Algorithmen und Werkzeuge für Petrinetze.
Fachberichte Informatik 7/2000, Universität Koblenz-Landau
http://www.informatik.hu-berlin.de/top/pnml/download/JKW_PNML.ps

Modules and instances In the basic interchange format proposed in this paper, pages and reference nodes are the only concepts for structuring a Petri net. In a future version, there will be a concept for defining *modules*, for instantiating modules, and for building Petri nets from these *instances*. This concept is beyond the scope of this paper. However, it is worth noting, that pages can be considered as a special version of modules; this avoids a clutter of many different and redundant concepts in the enhanced interchange format.

Tool specific information For some tools, it might be necessary to store some internal information, which is not supposed to be used by other tools. In order to store internal information, each object and each label may be equipped with *tool specific information*. The internal format of the tool specific information is up to the tool. But, tool specific information is clearly marked and is assigned the name of the specific tool. Therefore, other tools can easily ignore this information. In general, we discourage the use of tool specific information. In some cases, however, tool specific information might be unavoidable—at least in the basic version of PNML as proposed in this paper.

Types and conventions Up to now, we have discussed the general structure of a Petri net file. The available labels and the legal combinations of labels for a particular object are defined by a *Petri net type*. Technically, a Petri net type is a document that defines the XML-syntax of labels; i. e. either a DTD-file or an XML-Schema. In this paper, we concentrate on a single Petri net type: a Petri net type for high-level Petri nets.

In principle, a Petri net type can be freely defined. In practice, however, a Petri net type chooses the labels from a collection of predefined labels, which are provided in a separate document: the *conventions*. The conventions guarantee that the same label has the same meaning in all Petri net types. This allows us to exchange nets between tools with a different, but similar Petri net type.

Up to now, the conventions define only labels that are necessary for high-level nets. Defining further labels and maintaining the conventions document is an on-going process; this process needs the cooperation of all providers of Petri net tools. In our basic interchange format as proposed in this paper, we can provide a technique for defining new Petri net types and new labels only.

3 Examples

In this section, we present some concrete XML syntax in order to exemplify the concepts discussed in Sect. 2. Here, we can only give a flavour of PNML[4] by examples. The examples are pieces of a PNML-coded document representing a Petri net. The examples refer to the PNML version 0.99 [PNM]. In PNML, the net, the Petri net objects, and the labels are represented as XML elements. An XML element is included in a pair of a start tag `<element>` and an end tag `</element>`. An XML element may have XML attributes[5] in order to qualify it. XML attributes of XML elements are denoted by an assignment of a value to a key in the start tag of the XML element `<element key=value>`. XML elements may contain text or further XML elements. An XML element without text or sub-elements is denoted by a single

[4]Please refer to `http://www.informatik.hu-berlin.de/top/pnml/` for a full definition of PNML.
[5]Do not confuse XML attributes with attributes of Petri net objects.

In: S. Philippi (Ed.): 7. Workshop Algorithmen und Werkzeuge für Petrinetze.
Fachberichte Informatik 7/2000, Universität Koblenz-Landau
http://www.informatik.hu-berlin.de/top/pnml/download/JKW_PNML.ps

tag `<element/>`. In our examples, we sometimes omit some XML elements. We denote this by an ellipsis (...).

The tags of the following XML elements are named after the concepts given in Sect. 2 except for the labels. Labels are named after their meaning. Thus, an unknown XML element appearing in a Petri net or in an object may indicated as a label of the net or the object.

1 The first example shows the representation of a place with the unique identifier `p1`. The place has two labels, more precisely two annotations. The first one represents the name of the place `<name>`, whereas the second one represents its initial marking `<initialMarking>`. An annotation consists of its value `<value>` and its graphical information (represented by the XML element `<graphics>`). A place furthermore may have graphical information. The XML element `<graphics>` depends on the context in which it appears. A place has a position, whereas an annotation has an offset position.

```
<place id="p1">
  <graphics>
    <position x="10" y="20"/>
  </graphics>
  <name>
    <value>ready to produce</value>
    <graphics>
      <offset x="-5" y="2"/>
    </graphics>
  </name>
  <initialMarking>
    <value>P</value>
    <graphics>
      <offset x="0" y="0"/>
    </graphics>
  </initialMarking>
</place>
```

2 A transition is represented in a similar way. Furthermore, this transition contains a tool specific information marking the transition as hidden in the example tool PN4ALL version 0.1. The XML element `<toolspecific>` must have at least the shown XML attributes and may contain XML elements defined by the tool.

```
<transition id="t1">
  ...
  <toolspecific tool="PN4all" version="0.1">
    <hidden/>
  </toolspecific>
</transition>
```

3 An arc runs from a source node referred to by the XML attribute `source` to a target node (`target`). PNML requires that also each arc has a unique identifier. This simplifies the implementation of XML parsers. The graphical information of the arc contains a list of position points. These points represent intermediate points of

```
<arc id="a1" source="p1" target="t1">
  <graphics>
    <position x="7" y="19"/>
    <position x="5" y="17"/>
  </graphics>
  <inscription>
    <value>x>/value>
    <graphics>
      <offset x="-3" y="5"/>
```


In: S. Philippi (Ed.): 7. Workshop Algorithmen und Werkzeuge für Petrinetze.
Fachberichte Informatik 7/2000, Universität Koblenz-Landau
http://www.informatik.hu-berlin.de/top/pnml/download/JKW_PNML.ps

the arc. The offset in the graphical information of the <inscription> defines the position of the label relative to the reference point of the arc. This arc has an additional attribute called <type> denoting the arc as a normal one.

```
    </graphics>
  </inscription>
  <type value="normal"/>
</arc>
```

4 The next example shows the representation of a page and of reference nodes of a Petri net. A page may have the same Petri net elements as the net itself—even pages and reference nodes. A reference node (a <referencePlace> or a <referenceTransition>) refers to a node of the net via the XML attribute ref. Its value refers to the identifier of a place of this net. Furthermore, a reference node may have its own graphical information, tool specific information, and labels. Remember that these labels have no real meaning, since they are ignored in the underlying flattened net. But, they allow reference nodes to carry their own name etc. Note that an arc on a page starts and ends in nodes or in reference nodes defined on the same page.

```
<page id="pg1">
  <name>
    <value>Example page of the net</value>
  </name>
  <referencePlace id="rp1" ref="p1">
    <name>...</name>
    <graphics>
      <position x="20" y="20"/>
    </graphics>
  </referencePlace>
  <referenceTransition id="rt1" ref="t1">
    ...
  </referenceTransition>
  <place id="p2">...</place>
  <transition id="t2">...</transition>
  <arc id="a2" source="rp1" target="t2">
    ...
  </arc>
  ...
</page>
```

5 The whole Petri net contains pages and Petri net elements. This example shows a net. It has an annotation defining the name of the net. The type of the net is available via the Uniform Resource Identifier (URI) given in the XML attribute type. In our case, type refers to a file describing the type of the net. The XML attribute pageref of the <graphics> element of the annotation label <name> refers to the page on which this label is displayed.

```
<net id="n1" type="HLnet.xsd">
  <name>
    <value>Example high-level net</value>
    <graphics pageref="pg1"/>
  </name>
  <place id="p1">
  ...
  </place>
  <page id="pg1">
  ...
  </page>
  ...
</net>
```


In: S. Philippi (Ed.): 7. Workshop Algorithmen und Werkzeuge für Petrinetze.
Fachberichte Informatik 7/2000, Universität Koblenz-Landau
http://www.informatik.hu-berlin.de/top/pnml/download/JKW_PNML.ps

4 Conventions

As discussed in Sect. 2, the PNML provides a general format for Petri nets. Which kinds of labels are legal for a particular version of Petri nets and which combinations of labels are legal for a specific object, is defined by a separate *Petri net type*. A Petri net type chooses from a list of labels, which are defined in a *conventions* document. Up to now, the conventions document is at its minimum. The defined labels where chosen with respect to the specific Petri net type used in this paper: high-level Petri nets. The definition of a consistent and appropriate set of labels for all kinds of Petri nets (along with a definition of some standard Petri net types) must be a joint effort—and is an on-going process. It could be based on features supported by the Abstract Petri Net Notation (APNN) [BKK95] and the results of the questionnaire, which was provided and evaluated by Sy et al. [SBB+00].

5 Conclusions

This paper presents, in its first version[6], the concepts and the terminology of the Petri Net Markup Language. It serves as a detailed starting point for the development of a standard interchange format for Petri nets. Some concepts are already fixed. Others are open for future discussion. People who are interested in the development of the standard interchange format and Petri net tool developers are welcome to discuss our proposal directly with us or on the email list PNX[7]. For instance, which Petri net labels are necessary to cooperate with the most Petri net types; which graphical requirements are useful. Furthermore, Petri net tool developers are asked to evaluate the presented concepts with their tools.

References

[BBK+00] Bastide, Rémi; Jonathan Billington; Ekkart Kindler; Fabrice Kordon; and Kjeld H. Mortensen (eds.): *Meeting on XML/SGML based Interchange Formats for Petri Nets*, Århus, Denmark, 21st ICATPN. Jun. 2000.

[BKK95] Bause, Falko; Peter Kemper; and Pieter Kritzinger: Abstract Petri Net Notation. *Petri Net Newsletter 49*:9–27. Oct. 1995.

[JKW00] Jüngel, Matthias; Ekkart Kindler; and Michael Weber: Towards a Generic Interchange Format for Petri Nets. In: Bastide et al. [BBK+00], (pp. 1–5). Jun. 2000.

[MM00] Mailund, Thomas and Kjeld H. Mortensen: Separation of Style and Content with XML in an Interchange Format for High-level Petri Nets. In: Bastide et al. [BBK+00], (pp. 7–11). Jun. 2000.

[PNM] Petri Net Markup Language. `http://www.informatik.hu-berlin.de/top/pnml/`. Last visited: 2000/08/10.

[SBB+00] Sy, Ousmane; Mathiue Buffo; Didier Buchs; Fabrice Kordon; and Rémi Bastide: An Experimental Approach towards the XML Representation of Petri Net Models. *Technical Report 2000/336*, École Polytechnique Fédéral de Lausanne, Departement D'Informatique. Jun. 2000.

[6] This paper will be an on-going document.

[7] `http://www.informatik.hu-berlin.de/top/PNX/`

In: S. Philippi (Ed.): 7. Workshop Algorithmen und Werkzeuge für Petrinetze.
Fachberichte Informatik 7/2000, Universität Koblenz-Landau
http://www.informatik.hu-berlin.de/top/pnml/download/JKW_PNML.ps