Towards a Generic Interchange Format for Petri Nets
– Position Paper –

Matthias Jüngel, Ekkart Kindler, and Michael Weber

Humboldt-Universität zu Berlin, Institut für Informatik, D-10099 Berlin, Germany

`juengel|kindler|mweber@informatik.hu-berlin.de`

1 Introduction

XML is in vogue! An XML based interface is a sales promotion for today's software products. Consequently, many software developers have started to design and to implement an XML based interface for their products. Developers of Petri net tools are not excepted from this trend. Recent postings on the Petri Net Mailing List (`PetriNets@daimi.au.dk`) show that several Petri net tools are currently being equipped with an XML based file format.

But, is an XML based file format of a Petri net tool more than a sales promotion? In fact, it is not if each Petri net tool uses a different file format. But, it is much more than a sales promotion if there is an XML based file format that helps to interchange Petri nets between different tools.

In the past, there have been some attempts to define a standard interchange format for Petri nets—without much success. Basically, the problems in defining a standard interchange format are the following:

1. Historically, each Petri net tool was equipped with its own file format, which was designed for the particular purpose of this tool. Implementing a parser for another file format was tedious, and the willingness to spend this extra implementation effort was low because no file format seemed to have the potential to serve as a standard (see below).
2. Even worse, each Petri net tool supports a different version of Petri nets—some with only slight variations, others with significant differences. Thus, an agreement on an interchange format meant to give up some features of a Petri net tool—maybe, the outstanding features.

The success of XML rules out the first problem: Several tool developers are willing to implement an XML based interface anyway. This eliminates the extra effort. Due to the sudden success of XML, all tool developers have started to develop an XML based file format at the same time. This eliminates historic bulk and allows a fresh approach towards a standard file format. In addition, the costs for implementing an XML based interface are significantly reduced by existing XML tools. Altogether, these arguments rule out the first problem—at least for a short period of time. And we should make the best use of this time.

But, how about the second, more serious problem? In the rest of this paper, we argue that an XML based format could rule out the second problem if designed in an appropriate way: We advocate a *generic* interchange format for Petri nets. In particular, we propose a document description for Petri net files that consists of two parts: a general part, which is independent of a specific version of Petri nets, and a specific part. We call the general part the *Petri Net Markup Language* (PNML), and we call the specific part the *Petri Net Type Definition* (PNTD). Of course, a generic interchange format will be irrelevant if all relevant information of a particular net type is defined in the PNTD—in that case, there is no difference to a collection of different file formats. Therefore, a good generic interchange format must capture the essence of most Petri net types in the general part: The PNML distils the essence of Petri nets, and the PNTD defines additional features, which are not captured by the PNML.

In this paper, we argue in favour of a generic interchange format (Sect. 2) and show that the idea of a generic interchange format for Petri nets is feasible (Sect. 3). Moreover, we identify some objectives and issues, which should be kept in mind when designing an interchange format (Sect. 4).

2 Generic Interchange Format

There is a large number of different Petri net tools. Each of them has its specific features and strengths. Some tools focus on a nice graphical representation. Other tools are convincing by a variety of simulation and visualization techniques. Other tools are not equipped with graphics at all—they provide strong algorithms for analysis of Petri nets.

Let us assume that a user wants to use different Petri net tools to analyse, to simulate, and to verify the same net. Of course, he does not want to edit the same net in each tool from scratch. The best way to solve this problem is an *interchange format* that can be read by each of the used tools. The tools, however, might support different versions of Petri nets and need to represent their special features in the interchange format. Thus, a general interchange format must support all features of all Petri net tools—existing and forthcoming. In order to avoid a clutter of unrelated net features in the interchange format, we propose a *generic interchange format*, which consists of two parts: One part, called PNML, defines the typical features of all Petri nets; the other part, called PNTD, defines the special features of a particular Petri net type. By the way, a generic interchange format is open for future extensions of Petri nets: We need to define only the special features of a new Petri net type in a PNTD.

The typical features of a Petri net are: it consists of places, transitions, and arcs; places are associated with a marking; the net elements may be annotated by some additional information. Moreover, Petri nets have a typical graphical representation. Extracting the typical features of Petri nets into a PNML results in a uniform file structure for all kinds of Petri nets. For particular net types, it remains to fix the special features of this type in a PNTD.

Of course, there are features of Petri net, which occur in some but not in all Petri nets (e. g. different versions of time). The interchange format should allow to exchange nets of different but compatible Petri net types. In order to interpret the compatible features in the same way in both types, the interchange format must provide naming conventions. For example, it should be clear from the name whether a timing constraint should be interpreted as a firing duration or a firing delay. Some of these conventions can be supported by a separate XML-document, which comprises a definition for each standard feature of Petri nets. We call this document the *conventions document*. When designing a new Petri net type, the standard features need not be defined from scratch—rather the new PNTD can refer to the features defined in the conventions document. This way, different net types can be exchanged between different tools—at least as far as the features from the conventions document are concerned.

Altogether, the interchange format consists of a PNML, several PNTDs, and conventions for defining new PNTDs and for interpreting special features.

3 Feasibility

A generic interchange format for Petri nets has some appeal. But, is it feasible? In this section, we discuss the feasibility of a generic interchange format for Petri nets based on XML by the help of an example. For lack of space, we can discuss the XML file of a single net only; we can neither present the XML Schema for the PNML nor the XML Schema for the corresponding PNTD. These documents can be found on our web page [4].

Figure 1 shows a simple algebraic system net, which is taken from Reisig's book [3]. Algebraic system nets are a particular version of high-level nets—that's all we need to know for this paper. Listing 2 shows the XML file corresponding to this net. The `pnml` tag indicates that the file is a PNML file, which is defined by the XML Schema[1] `pnml.xsd`. The `net` tag indicates the begin of a net, where the reference to `hlNet.xsd` refers to the XML Schema of the corresponding PNTD (algebraic high-level nets in our example). Then, the different components of the net, its name, its places, its transitions, and its arcs follow. Moreover, there is a specific `netInscription`, which defines the underlying data type of the net (i. e. sorts and operations).

The general structure of this file is defined in `pnml.xsd`; the specific parts are defined in `hlNet.xsd`. The general structure says that there are places, transitions, and arcs. The concrete definition of a marking as defined for algebraic nets, however, is given in the PNTD `hlNet.xsd`. Similar comments apply to annotations of arcs, which are concretely defined in `hlNet.xsd`.

Note that, on a high-level of abstraction, the file from List. 2 can be understood without knowing the detailed definitions of the PNTD in `hlNet.xsd`. Here, we cannot discuss the details of the different XML Schemas. The basic idea, however, is that `pnml.xsd` defines the components and their relation which are relevant in most Petri nets. A Petri Net Type Definition, i. e. a `pntd.xsd` file, gives a concrete definition for these components, and may add other components, attributes etc.

Note that the PNML as proposed in [4] provides only the minimum requirements on the general Petri net structure—the current proposal is not meant to be complete. The main purpose of this proposal is to demonstrate that a generic interchange format for Petri nets is feasible. What should be included in the PNML, what should be not included in the PNML, and which conventions should be imposed on the PNTDs is subject to discussions. Existing proposals like the LaTeX-based Abstract Petri Net Notation [1] can serve as a guideline for this discussion.

4 Issues

Hitherto, we focussed our discussion on being generic. Of course, there are other important issues that should be kept in mind when designing the interchange format. Here, we will only mention the following issues:

Graphics. Graphical information should be independent from a particular Petri net type—or at least as independent as possible.

Structuring. There should be a mechanism for structuring Petri net models; if possible, this mechanism should be independent from a particular Petri net type.

Markings. There should be a way to save a marking (or a list of markings) independently from the net itself. The same holds for other information associated with some net elements.

Rewriting. There should be a statement about how each tool behaves in case of rewriting a net: Let us assume, a tool A reads

---

[1] Note that XML Schema has not yet reached its final stage; thus, we can only present a preliminary proposal, here. The W3C working group on XML Schema believes that the current version [5, 6] is 'feature-complete': i. e. the functionality included in XML Schema will not change.

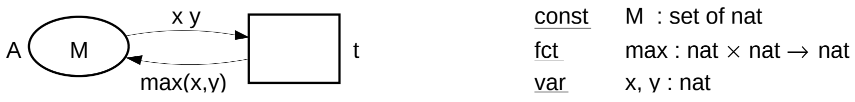

Figure 1: A net

a net generated by another tool B. The net may contain elements that are neither used nor readable by tool A. Now, a user might change the net in tool A and save this net. The question is: What happens to those elements that are not readable by tool A? Tool A might either ignore this information, in which case it is lost for tool B. Or tool A might try to write this information even if it might become inconsistent.

When defining the interchange format, we should be aware that it determines which formalism is accepted as a Petri net and which is not. We should make sure not to exclude some Petri net formalisms unintentionally. For example, there are Signal/Event nets [2], which allow special arcs between transitions. Because of its Petri net like semantics, we might accept a Signal/Event net as a Petri net.

5 Conclusion

In this paper, we have argued that an interchange format for Petri nets should be generic; i. e. it should support different Petri net types, and it should support the definition of new Petri net types. To this end, we proposed to distinguish between a general Petri Net Markup Language (PNML) and specific Petri Net Type Definitions (PNTDs). When defining this generic interchange format for Petri nets, we must take care not to miss the point: The essence of most Petri net types must be captured in the PNML, not in the specific PNTDs. Distilling the essence of Petri nets into a PNML might be a tedious task, which requires many discussions—but we believe that it is a worthwhile task.

Moreover, we have shown that XML Schema provides all concepts for a generic document description. In principle, there are no technical problems in defining an XML based generic interchange format. It remains to distil the essence of Petri nets into a PNML and into some conventions for the definition of PNTDs.

Acknowledgements We would like to thank Olaf Kummer, Stephan Roch and Karsten Schmidt for their comments and suggestions.

References

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

[2] Lüder, Arndt and Hans-Michael Hanisch: A Signal Extension for Petri nets and its Use in Controller Design. In: H.-D. Burkhard; L. Czaja; and P. Starke (eds.), *Proceedings of the CS&P'97 Workshop*, no. 110 in Informatik–Berichte, (pp. 98–105), Humboldt–Universität zu Berlin, Berlin, Germany. Sep. 1998.

[3] Reisig, Wolfgang: *Elements of Distributed Algorithms — Modeling and Analysis with Petri Nets.* Springer-Verlag. 1998.

[4] The PNK Team: Petri Net Markup Language. `http://www.informatik.hu-berlin.de/top/pnml`. Mar. 2000.

[5] XML Schema Working Group: XML Schema Part 1: Structures. *W3C working draft.* `http://www.w3.org/TR/xmlschema-1/`. Feb. 2000.

[6] XML Schema Working Group: XML Schema Part 2: Datatypes. *W3C working draft.* `http://www.w3.org/TR/xmlschema-2/`. Feb. 2000.

Listing 2: The XML file for the net from Fig. 1

```
<?xml version="1.0"?>
<pnml:pnml xmlns:pnml="pnml.xsd">
 <net id="n1" xmlns="hlNet.xsd">
  <name>Distributed Maximum Finding</name>

  <place id="p1">
   <name>A</name>
   <marking const="M"/>
  </place>

  <transition id="t1">
   <name>t</name>
  </transition>

  <arc id="a1" source="p1" target="t1">
   <annotation>
    <var ref="x"/>
    <var ref="y"/>
   </annotation>
  </arc>

  <arc id="a2" source="t1" target="p1">
   <annotation>
    <fct ref="max">
     <var ref="x"/>
     <var ref="y"/>
    </fct>
   </annotation>
  </arc>

  <netInscription>
   <sortDeclaration name="nat" type="basic"/>
   <constDeclaration name="M" sort="nat"/>
   <fctDeclaration name="max">
    <argumentSort ref="nat"/>
    <argumentSort ref="nat"/>
    <targetSort ref="nat"/>
   </fctDeclaration>
   <variableDeclaration name="x" sort="nat"/>
   <variableDeclaration name="y" sort="nat"/>
  </netInscription>
 </net>
</pnml:pnml>
```