Petri Net File Formats

Bernd Grahlmann

Universität Hildesheim
Institut für Informatik

bernd@informatik.uni-hildesheim.de
www.informatik.uni-hildesheim.de/~bernd

Abstract

The main intention of this paper is to encourage discussion about standards for Petri net file formats. We present criteria for good formats and estimate their significance for different classes of involved persons. The following persons are considered:

1. the designer of a Petri net tool who should integrate analysis algorithms and editors implemented by different persons at different locations;
2. the programmer of an analysis algorithm or a net editor, and
3. a user who wants to edit or analyse nets.

The **PEP** file format serves as an example to explain design decisions.

Keywords: File Formats, **PEP**, Petri Nets, Standards.

1 Introduction

Presently, there exists a large number of different Petri net tools, analysis algorithms and net editors. Every year many new ones are coming into existence. We do not want to explain all reasons for this, but we would like to concentrate on one of the reasons and its impact.

Analysis algorithms are often developed at universities. Frequently, students are entrusted with the implementations. In general, these students do not know enough about existing tools and editors to take the decision to do the implementation in such a way that it fits perfectly into an existing tool or editor. Thus the interfaces (which are basically file formats) are defined ad hoc, and in many cases a new editor (which offers only a basic functionality) is implemented in order to do the necessary testing of new algorithms.

This scenario is slightly exaggerated. Nevertheless, we think that interfaces are sufficiently important such that their proper design and useful standardisations need to be discussed. We have chosen file formats as we made the experience that these are the most realistic approach to integrate algorithms and editors implemented by different persons (in general from different universities or even different countries) into one Petri net tool. High fluctuation (e.g. of students) is a further argument.

The paper is structured as follows. In the second section, we give criteria for what we believe, from our experiences, to be a good net file format. Different kinds of 'net-like' objects handled by the **PEP** tool are described in the third section. In the fourth section, the **PEP** file format for low-level (LL) nets serves as an example to explain design decisions with respect to the given criteria. The conclusion ends this paper and hopefully starts some discussion.

The **PEP** (**P**rogramming **E**nvironment based on **P**etri nets) project is financed by the DFG (German Research Foundation).

2 Criteria for a good net file format

There are many criteria for a good net file format. Different persons have different opinions depending on their point of interest. Fig. 2.1 identifies the six (in our opinion) most important criteria and shows three different points of view of what is the main technical purpose of a Petri net file, viz. to *represent information*.

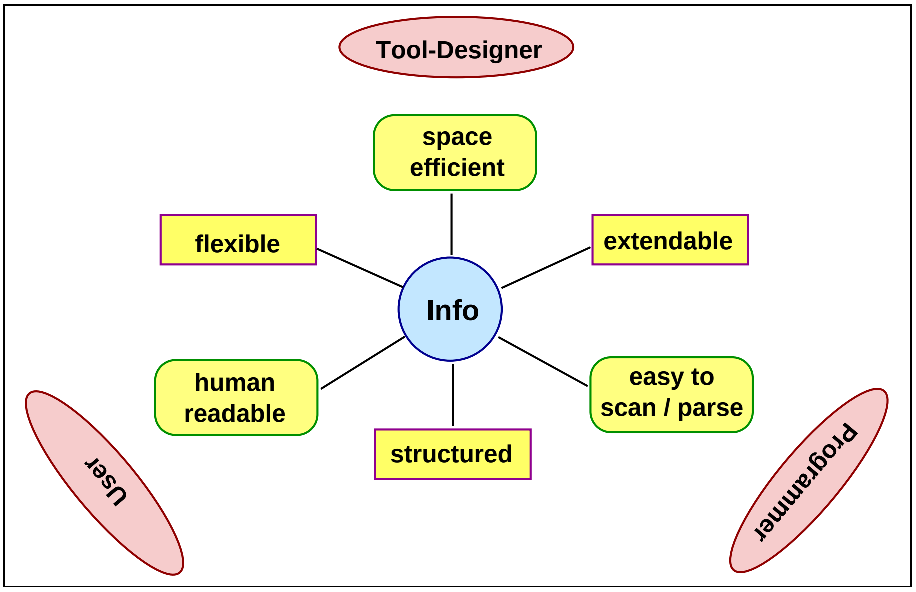


Figure 2.1: Representation of information and three interpretations.

The first three criteria

C1: space efficient information storage,

C2: easy parsability and scannability, and

C3: human readability

are, resp., slightly simplified and related to the points of view of the involved persons, viz.:

- the designer of a Petri net tool,
- the programmer and
- a user.

What we mean by simplification is, e. g., the fact that a good tool designer naturally takes care of the interest of users and programmers.

The classification of the last three criteria is more difficult:

C4: structure,

C5: extendability and

C6: flexibility.

It should also be mentioned that a Petri net file must, of course, contain all necessary information. The answer to the question 'What is necessary' (once more) depends on the point of view.

3 The PEP framework

This section briefly describes the **PEP** framework. The different kinds of 'net-like' objects handled by the **PEP** tool [BG95] are introduced and their main functionalities are explained in Fig. 3.1.

Figure 3.1: **PEP** framework

A distinctive feature of the tool is the possibility to compile parallel finite automata (PFA) [GMA95] or $B(PN)^2$ (**B**asic **P**etri **N**et **P**rogramming **N**otation) programs [BFF$^+$95] via high-level (HL) nets into LL nets in order to use Petri net analysis and verification algorithms for their verification [GB96]. The integrated model checker is based on the finite prefix of the branching process (prefix) [ERV96].

4 The PEP file format

In principle, it would be optimal to have only one basic file format which can be slightly extended in order to handle the four different kinds of objects used by the **PEP** tool:

1. PFA,
2. M-nets (a class of HL nets),
3. Petri boxes (a class of LL nets [BDH92]), and
4. Prefixes.

In practice, however, this would result in major drawbacks due to the different purposes of these objects. The user expects PFA, HL and LL nets to be graphically editable and (s)he appreciates if the corresponding files are human readable. For obvious reasons this contradicts space efficient storage, which is inevitable (at least) for prefixes. Experience has shown that HL and LL nets must also be stored in a space efficient way. Early in the development of the **PEP** tool, a non-optimised file format resulted in 250 MB files[1].

As a consequence, a new basic file format together with the corresponding (slight) extensions for PFA, HL[2] and LL nets was developed. Although this format is human-readable, it is optimised with respect to space efficiency. We have chosen a binary format for prefixes, however, which is even more space optimised and nearly human unreadable.

To enhance human (C3)[3] as well as machine (C2) readability a general *structure* (C4) ordering the presentation of the information is enforced. An LL net file contains the following seven parts (in the given order):

1. A header to specify the type of the file (e. g. **PEP** Petri box in the new format).

[1] Programs with procedures have been compiled into LL nets

[2] In [Bie96], e. g., an extension for hierarchical HL nets is described.

[3] We denote the matched criteria in brackets.

2. Default values (e. g., for the initial marking or the position of a place name) to override the given defaults. This reduces the size (C1), because information which is the same for most places, transitions or arcs, resp., is only stored once. (optional part)

3. A list of all places.

4. A list of all transitions.

5. A list of all arcs from a transition to a place.

6. A list of all arcs from a place to a transition.

7. A list of all additional lines of text displayed together with the net. (optional part)

We concentrate on the LL net file format in order to explain design decisions in more detail.

Depending on the purposes (Petri nets are used for), information can be classified according to their *necessity*. E. g., a pure analysis tool (such as, e. g., INA [Sta92]) does not use the positions of places or transitions. Regarding the fact that the **PEP** tool offers graphical net editing as well as analysis and verification we classified some information as necessary. For example for places these are:

- (unique) numerical place identifiers because they allow to specify arcs in an easy and space efficient way;
- (unique) place names because they allow to express properties by use of temporal logic formulae, and
- positions because they allow the graphical placement.

The **PEP** format even offers a possibility to omit identifiers and names completely (C1). This make sense if nets are generated automatically from programs in such a way that places and transitions are enumerated (e. g., T1, T2, ...).

The commands given in the lower part of the following table can be used to store optional information. An example for a line for a place is:

```
7"Hans"136@73M1b"Meaning"a-156@-29m1R"(6,5;6,9)"
```

Command	**Explanation**
num[4]	Place identifier
str[5]	Place name string
num @ *num*	Coordinates of the midpoint of the place
n *num* @ *num*	Relative position of the place name
b *str*	Meaning string
a *num* @ *num*	Relative position of the meaning string
M *num*	Initial marking
m *num*	Current marking
k *num*	Capacity (infinite if $num < 0$, e.g., '−1')
R *str*	Reference string[6]
e	Entry-place
x	eXit-place

[4] *num* can be an integer (positive or negative).

[5] *str* is a symbol for a string enclosed in double (") quotes.

[6] Gives a reference to the corresponding part of the B(PN)^2 program. The reference is given as a * separated list of terms (*line*,*column*;*line*,*column*).

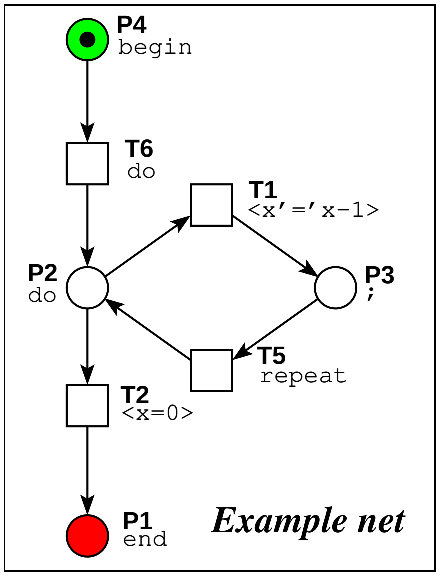
The format is flexible (C6) enough to accept the appearance of these optional information in any *order*. Whereas the order of the necessary information is obliged. A part from the fact, that the further reductions (C1) (complete omission of, e. g., place identifiers and names) otherwise would not be possible, this enhances the scan and parse process (C2).

Using *key letters* (such as `M` for the initial marking) and avoiding unnecessary separators has proven to be space efficient (C1) and extendable (C5) while preserving human readability (C3).

Newlines, however, should be highly recommended to separate the information related to the different places, transitions and arcs, respectively. The gain of human readability (C3) and easier parsability (C2) are worth the minimal space overhead.

Using *lists* for places, transitions and arcs reflects the data structures favoured by programmers (C2). Relating lists of pre- and post-transitions to each place was rejected for the same reason.

Fig. 4.1 gives a small example which illustrates the correlation between the graphical and the textual representation of a net. Lines with comments (% ...) can be omitted. For a full description of the LL file format the reader is referred to [Gra96].




```
PEP
PetriBox
FORMAT_N
% --- Places ---
PL
1"P1"90@330b"end"n17@-12a17@-2x
2"P2"90@210b"do"R"(3,1;3,3)"n-29@-12a-29@0
3"P3"210@210b";"R"(3,14;3,15)"n14@-11a14@-2
4"P4"90@90b"begin"R"(1,1;1,6)"n14@-12a15@0eM1m1
% --- Transitions ---
TR
1"T1"150@170b"<x'='x-1>"R"(3,5;3,14)"n18@-12a17@-1
2"T2"90@267b"<x=0>"R"(4,5;4,10)"n16@-10a16@0
4"T5"150@250b"repeat"R"(3,16;3,22)"n22@-12a23@-1
5"T6"90@150b"do"R"(3,1;3,3)"n18@-12a19@0
% --- Transition -> Place Arcs ---
TP
1<3
2<1
4<2
5<2
% --- Place -> Transition Arcs ---
PT
2>1
2>2
3>4
4>5
% --- Comments  ---
TX
N206@329"Example net"
```

Figure 4.1: Graphical and textual representation of a small LL net.

5 Conclusion

Class libraries are an appropriate approach if a tool is developed within one enterprise or at least at one university. However, file interfaces with properly defined file formats support the design of tools which are developed by loosely coupled groups with high fluctuation (e. g. students) in a better way. In particular, it is a useful mechanism to integrate different analysis algorithms (e. g. the verification methods from INA [Sta92], **PEP** [GB96] and PROD [VHHP95]) as done for example in [HD96].

This paper presented criteria for a good Petri net file format. The influences given by different points of view – those of the tool designer, the programmer and a user – have been considered. We believe that the **PEP** file format, which has been described in all brevity, might serve as basis for an urgently needed standardisation because it optimises[7] the matching of the six criteria.

Acknowledgment: I would like to thank Eike Best, Burkhard Bieber and Stefan Römer for their help.

References

[BDH92] E. Best, R. Devillers, and J. G. Hall. The Box Calculus: a New Causal Algebra with Multi-Label Communication. In *Advances in Petri Nets 92*, Vol. 609 of *LNCS*, pp 21 – 69. Springer, 1992.

[BFF+95] E. Best, H. Fleischhack, W. Frączak, R. P. Hopkins, H. Klaudel, and E. Pelz. A Class of Composable High Level Petri Nets. In *Proc. of ATPN'95, Torino*, Vol. 935 of *LNCS*, pp 103–118. Springer, June 1995.

[BG95] E. Best and B. Grahlmann. PEP: Documentation and User Guide. Universität Hildesheim. http://www.informatik.uni-hildesheim.de/~pep, November 1995. An ftp-able version of the **PEP** tool is available at the same location.

[Bie96] B. Bieber. Blockabstraktion in einer Programmiersprache für verteilte Systeme. Diplomarbeit, Universität Hildesheim, April 1996. (in German).

[ERV96] J. Esparza, S. Römer, and W. Vogler. An Improvement of McMillan's Unfolding Algorithm. In *Proc. of TACAS'96*, Vol. 1055 of *LNCS*. Springer, 1996.

[Gra96] B. Grahlmann. Description of the LL Net Format. Universität Hildesheim. http://www.informatik.uni-hildesheim.de/~pep, March 1996.

[GB96] B. Grahlmann and E. Best. PEP – More than a Petri Net Tool. In *Proc. of TACAS'96*, Vol. 1055 of *LNCS*, pp 397 – 401. Springer, 1996.

[GMA95] B. Grahlmann, M. Moeller, and U. Anhalt. A New Interface for the PEP Tool – Parallel Finite Automata –. In *Proc. of AWPN'95*, AIS 22, pp 21–26. FB Informatik Universität Oldenburg, October 1995.

[HD96] M. Heiner and P. Deussen. Petri Net Based Design and Analysis of Reactive Systems. In *Proc. of WODES'96*, Edinburgh, August 1996.

[Sta92] P. H. Starke. INA: Integrated Net Analyzer. Handbuch, 1992.

[VHHP95] K. Varpaaniemi, J. Halme, K. Hiekkanen, and T. Pyssysalo. PROD Reference Manual. TR 13, Helsinki University of Technology, August 1995.

[7] All six criteria cannot be 100 % matched with one approach.