PN^3-Editor: Compositional Petri Net Editor for protocol specification *

Nikolay Anisimov Aleksey Kovalenko
Pavel Postupalski
Institute for Automation and Control Processes
Far East Branch of the Russian Academy of Sciences
5 Radio Str., Vladivostok, 690041, Russia
E-Mail: anisimov@iapu2.marine.su

Abstract

The problem of Petri nets application to real-world parallel and distributed systems of industrial size causes developing of the compositionality within the theory of Petri nets. This paper presents a tools which is an attempt in resolving this problem. We suggest solution based on an algebraic approach of Petri net representation. Two levels of compositionality are introduced. The first one, algebraic level offers to a designer a set of net operations (sequential, alternative and parallel composition, iteration and disruption), which allow to build complex Petri nets from simpler ones. The second architectural level allows user to manipulate with Petri net entities, where the entity is defined to be a Petri net equipped with access points. The suggested technique is implemented in three-level Petri net editor comprising basic, algebraic and architectural editors.

Key words and phrases: *Petri nets, compositionality, specification, communication protocols.*

1 Introduction

It is well-recognized that Petri nets and related models are good for the representation and analysis of parallel and distributed systems [7, 6]. First, they enable one to represent concurrency and asynchrony in the most adequate and natural way. Second, the Petri net theory comprises a large body of models, techniques, algorithms, computer-aided tools which could be successively exploited for solving different problems. As a matter of fact, all these approaches suggest only one compositional technique – *hierarchy/refinement* technique. We decided to replace the above technique in Petri nets, at least at high levels of abstraction, by a less elegant, but still simple notation where modules could be represented by *boxes*, as is usual for many practitioners. We believe that at higher levels this model, see [1, 3], would be congruent to the thinking of many designers. More information, concerning the theoretical base of this model, you may find in [4].

*This work is supported by the Russian Fundamental Research Fund (Project 93-013-17372).

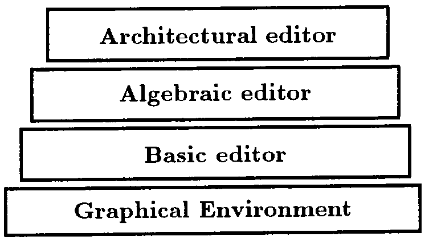


Figure 1: Architecture of PN^3-Editor

We will introduce the Petri net editor, called PN^3-Editor, which reflects all of the advantages of the suggested approach. PN^3-Editor comprises three subsystems: basic Petri net editor, algebraic editor, and architectural editor, see Fig.1. PN^3-Editor is implemented within MS Windows 3.1 graphical operation system, so it may be easily installed on many PC platforms.

2 Basic level

To work with classic Petri nets in PN^3–editor, a basic PN–editor was designed and implemented. It contains all editing facilities used in many Petri net tools [5], including:

- drawing/moving/removing of Petri net elements (boxes and circles);
- connecting places and transitions by directed arcs;
- changing dimensions of boxes and circles;
- setting the initial marking of Petri nets by placing tokens into circles;
- writing/moving/removing text information; and
- simulating of nets (both manually and automatically).

This Editor can build hierarchical nets by attaching to any transition, or place, of one net, the reference to other net, allowing any depth of inclusion. One can walk through this hierarchical structure in order to better understand it. In Fig.2, the example of a net built with the aid of basic PN-editor is presented.

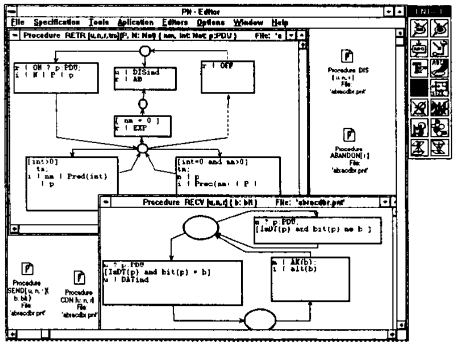


Figure 2: Basic PN–Editor: example net

When we deal with nets of reasonable size (on one screen), the basic PN–editor seems to be rarely satisfactory. Our experience with designing many of large size nets reveals some severe drawbacks to the classic approach:

- The drawing of a net, that was expected to be interesting and fascinating, turns out to be a routine and boring job, which may cause additional errors;
- The basic PN–editor forces a designer to think in terms of local states (places) and actions (transitions), while at some levels of abstraction it is more natural and efficient to think in terms of temporal ordering of events, sequential composition, etc.

The above disadvantages hinder potential developers from using the basic PN–Editor for solving real problems. Moreover, those disadvantages follow from the one fundamental shortcoming of a Petri net — a lack of a compositionality. The next section will show how it is possible to develop compositionality in Petri nets construction.

3 Algebraic level

To construct nets by means of net operations, an algebraic PN–editor is implemented on the top of a basic PN–editor. The notions of ***head*** and ***tail*** places must be used for performing an operation, see [4]. The following net operations was implemented in the algebraic PN–editor:

- putting an *atomic net*, which consists of one transition equipped with one input and one output place. This net may be labeled by naming the transition. See Fig.3.a).
- *alternative composition* or *choice* , where the tail and head places of the first net are merged with the places of the same type of the second net. See Fig.3.b), where $spec3 = spec1[]spec2$.
- *parallel composition with synchronization*, where the transitions that have the same labeling have to be merged. See Fig.3.c), where $spec1 = spec2||_s spec3$.
- *sequential composition*, where the tail places of the first net are merged with the head places of the second net See Fig.3.d), where $spec6 = spec4 >> spec1 >> spec5$.
- *iteration*, where the tail places of the net are merged with head places. See Fig.3.e), where $spec3 = *spec6$.

The scenario of execution of each such operation is as follows: for performing a binary operation one should point out via toolbox two windows with net operands and a third window for a result. In particular, one and the same window can serve for an operand and a result simultaneously. Head and tail places are graphically indicated by incoming and outcoming extra arcs respectively.

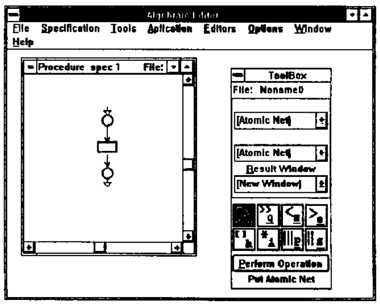
One can observe some obvious advantages of this style of net construction:

- Using only few operations one can obtain a rather complex net that can be read and understood easily, i.e. the reading process reproduces a construction process.
- A designer thinks in terms of a practitioner, such that sequence, alternative, etc. This way of thinking is more natural for a person working at the high level of compositionality.
- A set of net operations is sufficient to construct various systems. In particular it suitable for specification of communication protocols.

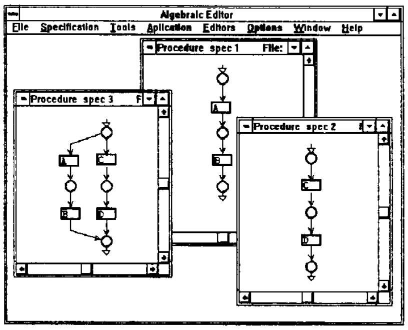
At the same time, the use of the above algebraic approach in Petri net construction may lead to unpleasant results. In other words, the way of algebraic Petri net construction may produced very complex and cumbrous specifications. To pass over this disadvantage we suggest the next level of compositionality in our approach of Petri net construction — the so called architectural level of specification.

4 Architectural level

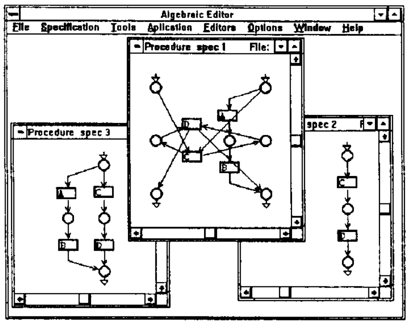
The architectural level of compositionality is the highest level of specification. The main idea underlining this level is to represent graphically Petri nets and their operations by interconnected boxes. To accomplish this, we define a notion of *Petri net entity* that previously was introduced in [1]. Petri net entity represent the Petri net which has the set of named access points. We need to use the names of access points for their identification and references during the operations.



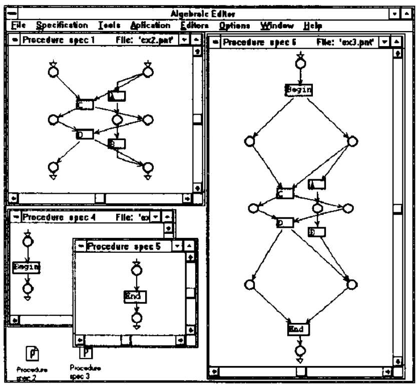
a) Atomic Net



b) Choice

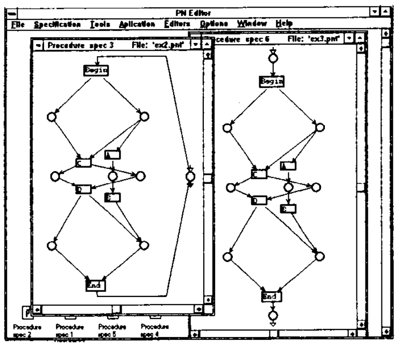

Algebraic Editor
File Specification Tools Aplication Editors Options Window Help
Procedure spec 1 File:
Procedure spec 3
spec 2

c) Parallel Composition with synchronization


Algebraic Editor
File Specification Tools Aplication Editors Options Window Help
Procedure spec 1 File: 'ex2.pnt'
Procedure spec 6 File: 'ex3.pnt'
Procedure spec 4
Procedure spec 5
Begin
End
Procedure spec 2
Procedure spec 3

d) Sequential composition

Figure 3: Examples of net operation (continued)


PN Editor
File Specification Tools Aplication Editors Options Window Help
Procedure spec 3 File: 'ex2.pnt'
cedure spec 6 File: 'ex3.pnt'
Begin
End
Procedure spec 2
Procedure spec 1
Procedure spec 5
Procedure spec 4

e) Iteration

Figure 3: Examples of net operation

The architectural PN-editor is intended to design complex specifications at the higher level of compositionality. An element of the editor is an entity represented as a box with t-access points (TAP) and p-access points (PAP). The editor allows one to build from boxes of various configurations by interconnecting them and refining them.

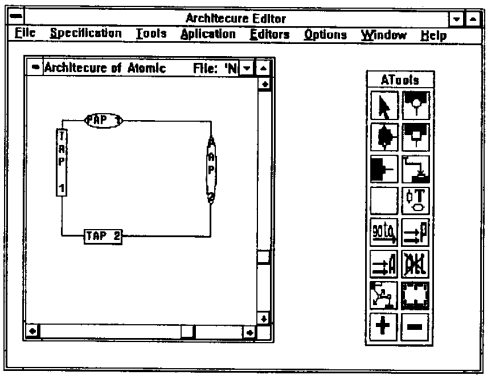
The intuition behind the notion of t-access points is quite simple. A transition access point is nothing but a well-known notion of labelling of a Petri net. The notion is to assign to each transition some visible label or an invisible τ-symbol. Through TAP one can observe executions of transitions with visible labels.

A place access point is a set of safe markings, mutually incompatible with respect to inclusion. The state in which a net reaches one of the markings is considered to be an event in the PAP. PAP can be thought of as a generalization of a Petri net marking, where we take a set of mutually exclusive markings. Reaching one of them is treated as a visible event from this PAP. Note that it is more natural than only using one marking.

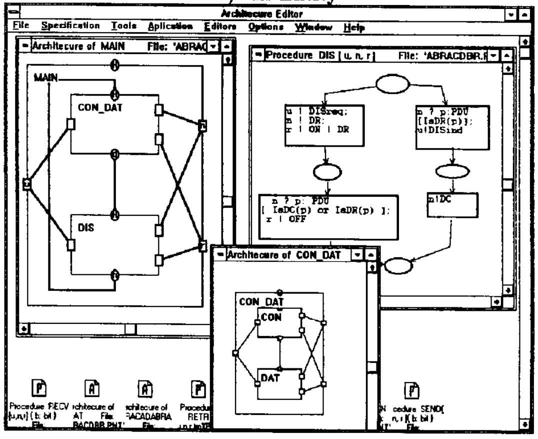
In particular, the architectural editor implements the following functions:

- Drawing of entity as a box with its name within the box. Drawing s– and t–access points in form of small circles and boxes placed on the perimeter of the entity-box, See Fig.4.a);
- Performing entity composition by connecting corresponding s– or t–access points, see Fig.4.b), where entity "MAIN" consists of "DIS" and "CON-DAT";
- Refining of an entity by defining its internal structure that in its turn is a entity configuration assembled from other entities (as "MAIN" on Fig.4.b)). Entities which have internal structure are called *composed* entities.
- Providing an entity hierarchy allowing to represent the internal structure of an compound entity on a separate page. So the architecture of specification may take several hierarchically organized pages.

- Establishing links between non compounded entities and their internal structures designed in terms of Petri nets at lower levels of PN^3-editor. Such entities will call as *atomic* entities (as "DIS" on Fig.4.b)).
- The editor possesses some editing facilities which allow to move/delete boxes-entities, access points, changing their sizes, etc.
- The editor allows one to 'walk through' specifications in order to better understand their structures.


Architecure Editor
File Specification Tools Aplication Editors Options Window Help
Architecure of Atomic File: 'N
ATools
RAP
TAP 1
RAP
TAP 2

a) An Entity


Architecure Editor
File Specification Tools Aplication Editors Options Window Help
Architecure of MAIN File: 'ABRAC
MAIN
CON_DAT
DIS
Procedure DIS [u, n, r] File: 'ABRACDBR.
Architecure of CON_DAT
CON_DAT
CON
DAT

b) Architecture specification

Figure 4: Examples of Architectural Editor

Thus, a complete typical specification built by our PN^3-editor consists of hierarchically organized architectural parts, in which each compounded entity is associated with its own entity configuration and each atomic entity is associated with a Petri net (see Fig.1.b).

5 Concluding remarks

In general, in this paper we suggest a technique for extending Petri net techniques to deal with systems of large size. This approach is demonstrated by the problem of producing specifications of complex system with the aid of compositional Petri net editor. Let us point out some advantages of the approach:

- The suggested technique is precise because it is strongly based on Petri nets theory at all levels of compositionality;
- The technique is flexible enough to allow the user to leave one level compositionality for another one at any point of design process;
- The technique can be successively exploited both for top–down or bottom–up approaches to specification of complex systems, starting from architectural or net representations;
- The technique allows the designer to explicitly define architecture for specifications in graphical form, using different styles of specifications [8].

References

[1] N. A. Anisimov. A Notion of Petri Net Entity for Communication Protocol Design (Institute for Automation and Control Processes, 1989).

[2] N. A. Anisimov, A. A. Kovalenko, P. A. Postupalski, A. S. Simancuk, A Graphical Petri Net Based Editor for Visualization of Distributed and Parallel Systems. *Lecture Notes in Computer Science*, Vol. 631, pp. 847–848.

[3] N. A. Anisimov, A. A. Kovalenko, P. A. Postupalski, Two–Levels Formal Model for Protocol Specification Based on Petri Nets, in *Network Information Processing Systems. Proc. of the IFIP TC6 Int. Symp.* ed.K. Boyanov, (Bulgarian Academy of Sciences, 1993) pp. 143–154.

[4] N. A. Anisimov, A. A. Kovalenko, P. A. Postupalski, Compositional Petri Net Environment, to appear in: *Proc. of the ETFA'94: Seiken/IEEE Symp. on Emerging Technologies and Factory Automation, November 6-10, 1994, Tokio, Japan*

[5] F.Feldbrugge, Petri Net Tool Overview 1992, *Petri Net Newsletter*, 41 (Gesellschaft für Informatik, 1992) pp. 2–42.

[6] T. Murata, Petri Nets, *Proc. IEEE*, Vol. 77 (1989), pp. 541–580.

[7] J.L.Peterson. *Petri Net Theory and the Modelling of Systems* (Prentice–Hall Inc., 1981)

[8] C.A.Vissers, G.Scollo, M.Van Sinderen, Architecture and Specification Style in Formal Description of Distributed Systems, in: *Protocol Specification, Testing, and Verification, VIII: Proc.of the IFIP WG 6.1 8th International Symposium*, eds. S. Aggarval, K. Sabnani (North–Holland, 1988).