UDC ????

A SUPPORT TOOL FOR THE REACHABILITY AND OTHER PETRI NETS-RELATED PROBLEMS AND FORMAL DESIGN AND ANALYSIS OF DISCRETE SYSTEMS

Štefan Hudák, Štefan Korečko, Slavomír Šimoňák

Department of Computers and Informatics, Faculty of Electrical Engineering and Informatics, Technical University of Košice, Letná 9, 042 00 Košice, tel.+42155 602 2525, Fax +42155 602 2746,E-mail: {Stefan.Hudak, Stefan.Korecko, Slavomir.Simonak}@tuke.sk

Annotation. In this paper we deal with the PNtool - a tool for a design, analysis and development of concurrent and time-critical systems specified using the Petri nets (PN) formalism. The PNtool supports four Petri nets dialects: Generalized Petri nets, Time-basic nets, Evaluative and Coloured Petri nets. The tool allows to design and simulate a system using any of the supported PN dialects and to perform an invariants-based analysis and reachability analysis for Generalized PN. Each Generalized and Evaluative PN can be also loaded from and saved in a standard interchange format called PNML. The tool is implemented in Java as a part of the mFDT Environment (mFDTE). The mFDTE is a toolset for the formal design and analysis of concurrent discrete and time-critical systems, developed at the home institution of the authors. It integrates three formal methods with complementary features: Petri nets, process algebra and B-Method. This paper also describes interfaces, which connect the PNtool with other parts of mFDTE. The work presented is supported by the grants No. 1/3140/06 and No. 1/4073/07 of the VEGA- The Scientific Grant Agency of Slovakia and NATO CLG 982698 grant.

В статье идёт речь о программном средстве – PNtool, предназначенным для синтеза и анализа параллельных и критических во времени систем, которые описываются с помощью формализма сетей Петри (СП). PNtool предназначен для поддержки четырёх диалогов СП: обобщённые СП (generalized PN), временно- базисные СП (Time-basic PN), эвалюационные СП (evaluative PN) и раскрашенные СП (Coloured PN). PNtool позволяет конструирование и симуляцию системы с применением любого из упомянутых диалектов СП и также проделать анализ системы с целью одержания её инвариантов , и решения проблемы достижимости для системы представленной обобщённой СП.Спецификация систем на языке обобщённых, или эвалюационных СП может быть представлена и сохраняться в формате PNML.PNtool реализован на языке JAVA, как одна из частей программной среды mFDTE (multi FDT Environment). Среда mFDTE предназначена для синтеза и анализа параллельных (concurrent) дискретных систем , включая и временно-критические системы. Программная среда mFDTE создана в університете авторов статьи. В ней интегрированы три формальных метода, которые обладают взаимно-комплементарными свойствами: СП, процессные алгебры и метод B AMN. В статье описываются интерфейсы связывающие PNtool с остальными частями среды mFDTE. Результаты исследований представленные в статье были достигнуты при поддержке грантов № 1/3140/06 , № 1/4073/07 VEGA – научного грантового агентства Словакии, и гранта NATO CLG 982698.

1. Introduction

The architecture of the multi Formal Description Technique Environment (mFDT Environment, mFDTE) has been proposed in **Ошибка! Источник ссылки не найден.** as, among others, an answer to one of the greatest problems in the practical use of formal methods. The problem is connected with a hypothesis that claims that there does not (and will not) exist any universal formal method, covering all aspects of systems. The mFDTE adopts the idea of formal methods integration to cope with this problem. It integrates three formal methods with complementary features:

1. Petri nets (PN) is a behaviour-oriented method with very nice analytical features, such as an automatic derivation of invariant properties and a reachability analysis.
2. Process algebras view systems as processes, described in an algebraic way. They allow to deal with a de/composition of systems very elegantly. The mFDTE supports two process algebras: the Algebra of Process Components (APC) **Ошибка! Источник ссылки не найден.** and the Algebra of Communicating Processes (ACP) **Ошибка! Источник ссылки не найден.**.
3. B-Method (B) **Ошибка! Источник ссылки не найден.** is a model-oriented method. Contrary to the previous methods, B provides the whole development process from abstract formal specification through the sequence of refinements to the concrete specification, which can be automatically translated to the executable code.

The mFDTE consists of tools for the integrated methods and interfaces between languages of these methods. The tools allow the designer to gain from the advantages of individual methods and the interfaces provide correct and formally proved translation from specification in one method to the equivalent specification in another one.

In this paper we introduce the most advanced of the mFDTE tools – the tool, which implements the Petri nets formalism and is called *PNtool*. After the introduction the paper continues with a short description of four Petri nets dialects supported in the *PNtool*. Section 3 is dedicated to the *PNtool* itself – namely to its modes and features. In Section 4 we describe how the exchange format PNML is supported in the *PNtool* and Section 5 deals with the interfaces, which connect the *PNtool* with the other parts of the mFDTE.

2. PN dialects supported

There are many types, or "dialects", of PN and the *PNtool* supports four of them: Generalized PN, Time-basic nets, Evaluative PN and Coloured Petri nets.

2.1. Generalized Petri nets

The *Generalized PN* (*GP nets*, *GPN*) are the "basic" type of Petri nets, supported in mFDTE. This type of PN is also called Place/Transition nets (P/T nets) or, simply, Petri nets. GP net can be defined as 5-tuple

$$N=(P,\ T,\ \text{pre},\ \text{post},\ m_0),$$

where $P=\{p_1,...p_n\}$ is a finite set of places, $T=\{t_1,...t_m\}$ is a finite set of transitions, $pre: P\times T\rightarrow \mathbb{N}$ is a preset function, $post: P\times T\rightarrow \mathbb{N}$ is a postset function and $m_0\in \text{N}^{|P|}$ is the initial marking. $\mathbb{N}$ is the set of natural numbers with 0.

A *marking* of GP net N is a function $m: P\rightarrow \mathbb{N}$. Value of $m(p)$ is the number of tokens in the place p. These tokens have no individuality, so they are undistinguishable. Transition $t\in T$ is *enabled* (*feasible*) in marking m, iff

$$\forall p\in {}^{\cdot}t : m(p)\geq \text{pre}(p,t),$$

where ${}^{\cdot}t=\{p|\ \text{pre}(p,t)\neq 0\}$. When t is enabled, it can be executed (fired). The result of its execution is a new marking $m'\in N^{|P|}$:

$$m'(p)= m(p)- \text{pre}(p,t)+ \text{post}(p,t).$$

Markings represent states of a Petri net. A marking, which can be reached from the initial marking of some GP net N by firing some sequence of enabled transitions is called a reachable marking of N.

A very popular representation of GP net (and other types of PN) is an oriented graph with two types of vertices - places (circles, or ellipses) and transitions (rectangles). When pre(p,t) $\neq 0$, then there is an arc from p to t, when post(p,t) $\neq 0$, then there is an arc from t to p. If the value of pre(p,t) or post(p,t) is greater than 1, then it is written next to the corresponding arc. For example, in the net in Fig. 1 we have pre($p1$,t)=2, pre($p2$,t)=1 and post($p4$,t)=3.

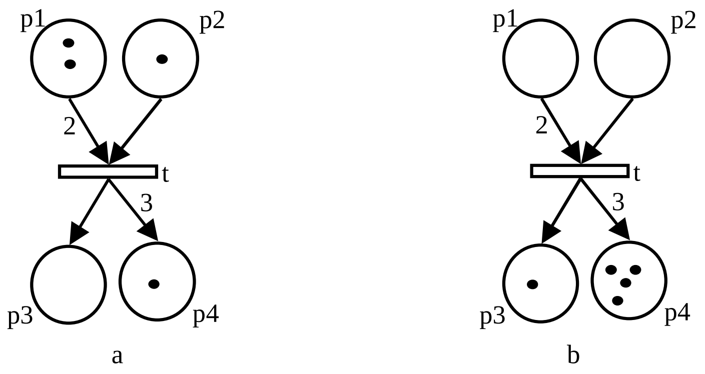


Fig. 1. GP net (fragment) before (a) and after (b) the firing of the transition t

The system specified by GP net can be analysed by means of the place and transition invariants and the reachability analysis, for which the original algorithm has been developed **Ошибка! Источник ссылки не найден.**.

2.2. Time-basic nets

Time-basic nets (TB nets, TBN) are an extension of GPN, which incorporates the concept of the time. They are intended for a design and analysis of time-critical systems. There is a value, called *chronos*, or *timestamp*, associated with each token. The value is the time when the token has been created by a firing of some transition. The *time function* is associated with each transition and describes the relation between the timestamps of tokens removed by the firing and the timestamps of tokens created by the firing.

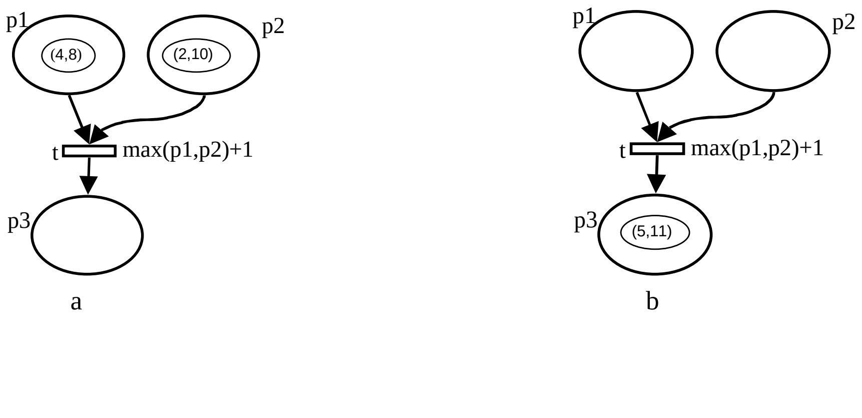


Fig. 2. TB net with TI semantics before (a) and after (b the firing of the transition t

The timestamps can be represented as single values (time points) or time intervals. The TB nets where timestamps are intervals are called TB nets with *time interval* (TI) semantics. An example of TB net with TI semantics is shown in Fig. 2. Here the time function for the transition *t* only defines how the timestamp of a new token, added to *p3*, is computed:

$$\max((4,8),(2,10)) = (4,10) \text{ and } (4,10)+1 = (5,11).$$

More information about TB nets with TI semantics, including canonical representation of time function and an example of a new approach to the time reachability analysis can be found in **Ошибка! Источник ссылки не найден.**.

2.3. Evaluative Petri nets

The Evaluative Petri nets (*EvPN*) are an extension of GPN, introduced in **Ошибка! Источник ссылки не найден.**. They have important analytical properties and the computational power equal to the Turing machines. EvPN differ from GPN in the following way:

- There are 3 sets of places in each EvPN. The first set, designated as *P*, includes places, which are alike the places in GPN. Places from *P* are called individual variables. The second set is called P_f and there is an n-ary integer function assigned to each place from P_f. These functions are functions over the individual variables, i.e. over the places from *P*. The marking of each place from P_f must be always equal to the value of the function assigned to given place. The third set is P_e and there is a predicate over the individual variables assigned to each place from P_e. The value of the predicate must always correspond to the marking of given place: If the predicate, associated with some place p_e, $p_e \in P_e$, is true, then $m(p_e)=1$. Otherwise $m(p_e)=0$.
- The negative values of markings of places from *P* and P_f are allowed.
- The capacity (i.e. maximal value of marking) can be defined for places with respect to individual arcs.
- If some transition *t* fires, then it is possible, that the change of the marking of adjacent places depend also on the marking of places, which are not adjacent with *t*.

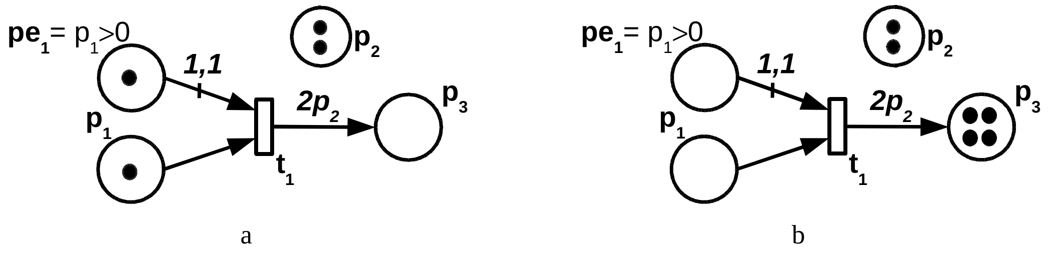


Fig. 3. EvPN before (a) and after (b) the firing of the transition t

In Fig. 3 we have an EvPN with $P = \{p_1, p_2, p_3\}$ and $P_e = \{pe_1\}$ The set P_f of this net is empty. The meaning of the predicate for pe_1 is $m(p_1)>0$. The inscription 1,1 of the arc from pe_1 to t_1 means that a firing of t_1 takes one token from pe_1 and that the capacity of pe_1 is 1. The inscription $2p_2$ of the arc from t_1 to p_3 means that a firing of t_1 adds $2*m(p_2)$ tokens to p_3.

2.4. Coloured Petri nets

Coloured Petri nets (*CP nets, CPN*) **Ошибка! Источник ссылки не найден.** belong to the family of High-level PN, where tokens have individuality. This means that each token have some value. The value of token can be some integer, string, tuple, list of values and so on. These values are called colours and their types are called colour sets. There are arc expressions, specifying the groups of tokens taken from and delivered to places when transitions are fired, and guarding predicates, associated with transitions. The guarding predicates represent additional conditions of transitions feasibility. Each CPN has declarations, where colour sets, functions and variables used in arc expressions and predicates are defined. The declarations, arc expressions and predicates must be written in a language, which meet the specifications formulated in **Ошибка! Источник ссылки не найден.**. To define groups of tokens (in markings, arc expressions…) multisets are used. For example, the initial marking of place p_1 from Fig. 4 is one token of value (colour) 1 and one token of colour 5 (it is the multiset 1\`1++1\`5).

The authors of CP nets **Ошибка! Источник ссылки не найден.** claim that CP nets have the same expressional power as GP nets and each CP net can be translated into equivalent GP net. Therefore the same analysis methods as for GP nets can be used for CP nets.

3. PNtool modes and features

The *PNtool* allows to create, edit, simulate and analyse models specified by means of PN dialects described above. It is a JAVA-based application, which uses the JDOM library to deal with XML (PNML) format and SWING components for graphical user interface. It can be run in one of these three modes: GPN/TBN mode, EvPN mode, CPN mode.

Each mode is dedicated to the corresponding types of PN and offers a plenty of features, which are described in the rest of this section.

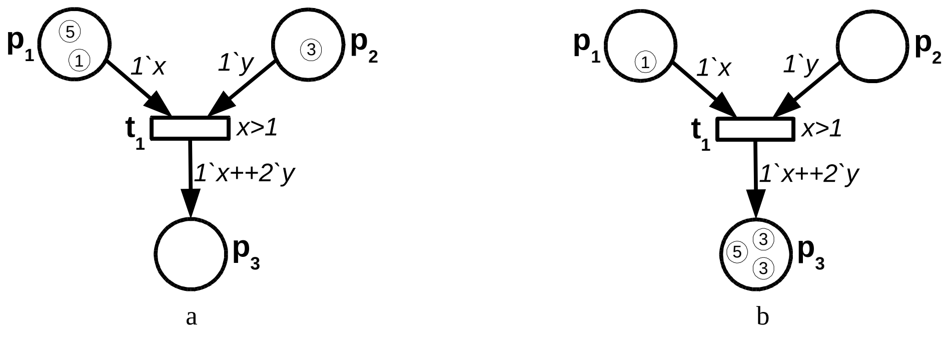
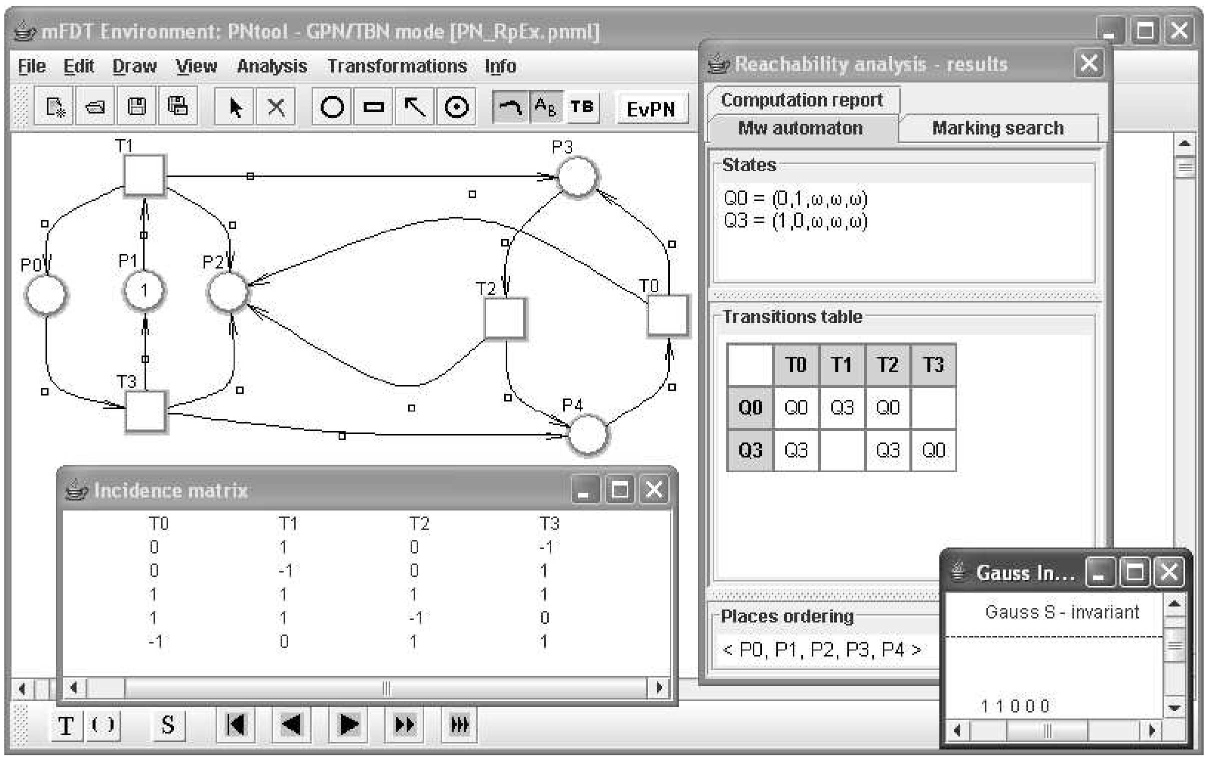


Fig. 4. CP net before (a) and after (b) the firing of the transition t

3.1. GPN/TBN mode

The GPN/TBN mode is the feature-richest mode of the *PNtool*. An appearance of the *PNtool* in this mode can be seen in Fig. 5.



mFDT Environment: PNtool - GPN/TBN mode [PN_RpEx.pnml]
File Edit Draw View Analysis Transformations Info
EvPN
TB
T1
P3
P0
P1
P2
T2
T0
T3
P4
1
Reachability analysis - results
Computation report
Mw automaton
Marking search
Incidence matrix
Gauss In...
T
S

States

Q0 = (0,1,ω,ω,ω)
Q3 = (1,0,ω,ω,ω)

Transitions table

	T0	T1	T2	T3
Q0	Q0	Q3	Q0	
Q3	Q3		Q3	Q0

T0	T1	T2	T3
0	1	0	-1
0	-1	0	1
1	1	1	1
1	1	-1	0
-1	0	1	1

Places ordering

< P0, P1, P2, P3, P4 >

Gauss S - invariant

1 1 0 0 0

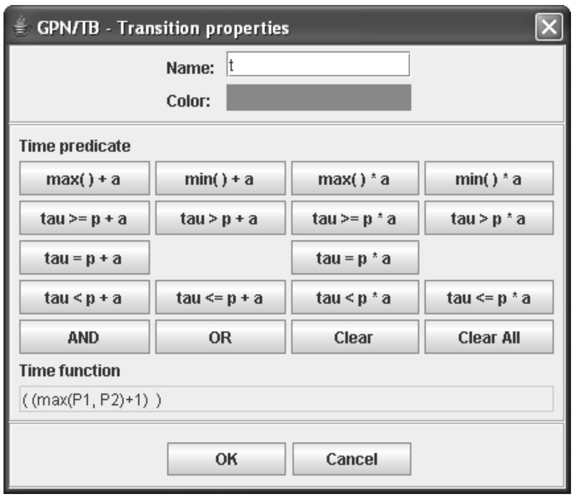
Fig. 5 A screenshot of the *PNtool* in the GPN/TBN mode

The GPN/TBN mode provides the following components for Generalised and Time-basic nets:
Graphical editor. In this essential component it is possible to draw, save, load and modify a graph of any GP or TB net. The properties of places and transitions, such as name, appearance (colour), initial marking (i.e. a number and time intervals of tokens in places) and time function can be set in special dialog windows. One of them, the transitions properties window, is shown in Figure 6. Each time function is composed from the simple time predicates, which can be seen in Fig. 6, too.

Simulator. A simulation of PN is controlled by a simulation toolbar, which is situated in the left lower corner of the main *PNtool* window (Fig. 5). When the simulation is turned on, all enabled transitions of a given net are highlighted in the graphical editor. By clicking on some enabled transition we fire the transition. The effect of firing – a new marking and a new set of enabled transitions - is shown directly in the graph. The simulation toolbar also allows to fire one random transition or a sequence of 5 or 50 randomly chosen transitions. In addition, it is possible to go back to the previous marking or to the initial marking.

Even more simulation possibilities are available for TB nets. Namely, we can choose from 3 simulation modes:

- GPN-like mode, which ignores timestamps and time functions,
- Time point semantics mode, which takes into account only beginnings of time intervals and
- Time interval semantics mode.



Name: t

((max(P1, P2)+1))

Fig. 6 The transitions properties window from the GPN/TB mode

Invariants-based analyser (GPN only). An automatic derivation of invariants is one of the most valuable properties of Petri nets. There are two kinds of invariants for PN: S- and T-invariants (also called place and transition invariants). An S-invariant represents a set of places for which a weighted sum of tokens remains the same for any reachable marking and a T-invariant defines a sequence of transitions which, if executed, leads from a given marking *m* back to *m*.

Both types of invariants are integer vectors and are acquired by solving a linear equations system. The *PNtool* can compute both of them, using Silva or Gaussian elimination method. If the net analysed has infinitely many invariants, the *PNtool* computes only basic ones. The other invariants can be derived from these basic ones, because any linear combination of invariants is also an invariant.

An example of invariant analysis results can be seen in the right lower corner of Fig. 5. Here we got the basic S-invariant (1,1,0,0,0) of the net displayed in the editor. This means that net's places *P0* and *P1* always share exactly one token.

It is also possible to compute invariants by calling an Adriana plug-in, developed by D.A. Zaitsev's research team, from *PNtool*. The Adriana plug-in implements a decompositional method, in which the net is first decomposed into its so-called functional subnets. Then the invariants are computed for these subnets and combined into the invariants of the whole net. This method, which is described in **Ошибка! Источник ссылки не найден.**, is well-suited for the analysis of large Petri nets.

Reachability analyser (GPN only). This component helps to solve a *reachability problem* (RP) for GPN. The RP is one of the crucial problems related to Petri nets and is formulated as follows:

Assume a GPN N, defined as in Section 2.1, with *n* places and *n*-ary nonnegative integer vector q. Then an instance of RP for N and q is the problem whether q is a reachable marking of N.

An original algorithm to solve RP was designed by the first of the authors **Ошибка! Источник ссылки не найден.**. The algorithm has been further extended by including de/composition strategies and time issues **Ошибка! Источник ссылки не найден.**. In general, the algorithm consists of two steps:

1. Construction of an M_w automaton for the net *N*.

The M_w automaton is a special type of finite automaton, which represents the (possibly infinite) state space of *N*. States of M_w automaton are labelled by vectors, which are reachable markings of *N* or cover some infinite subset of reachable markings of *N*. In the latter case some members of these vectors are ω symbols and states labelled by them are called *macrostates*. The ω stands for a number greater than any natural number. For example, the both states of the M_w from Fig. 5 are macrostates. Arcs of M_w are labelled by PN transitions. One of the advantages of M_w is that once it is obtained, it can be used for any instance of RP for *N*.

2. Creation and solving of the integer linear programming problem for M_w and q – the $ILP(M_w, q)$ **Ошибка! Источник ссылки не найден.**.

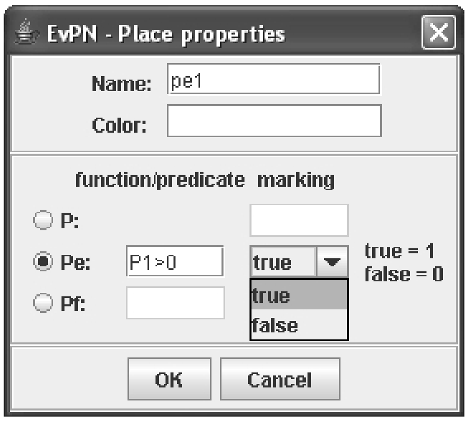
The *PNtool* currently fully implements the first step of RP algorithm. It displays the M_w automaton created in the form of transition table and a list of states with their labels (see the right part of Fig. 5). It also provides a step-by-step visualization of the Mw automaton construction for educational purposes. The second step is supported only partially: the tool compute some elements for $ILP(M_w, q)$ solving, such as the vector form of simple loops in M_w and identification of M_w state, which covers the vector q. Because $ILP(M_w, q)$ is nothing else than a problem of linear equations system solving, it will be relatively easy to adopt the equations system solver of invariants analyser for the ILP task.

To cope with a tremendous complexity of RP solving de/compositional approaches to RP solving, based on T-junction and P-junction de/composition of PN **Ошибка! Источник ссылки не найден.**, have been added to the PNtool.

3.2. EvPN mode

In the EvPN mode a graphical editor and a simulator of Evaluative PN are available. These components work similarly to their counterparts from the GPN/TBN mode.

Among others the EvPN mode introduces new arc and place properties dialogs. The latter one, shown in Fig. 7 allows to define predicates and functions for the places from P_e and P_f. However, it is not checked, whether the structure of the net obeys these predicates and functions.



Name: pe1

Pe: P1>0 true

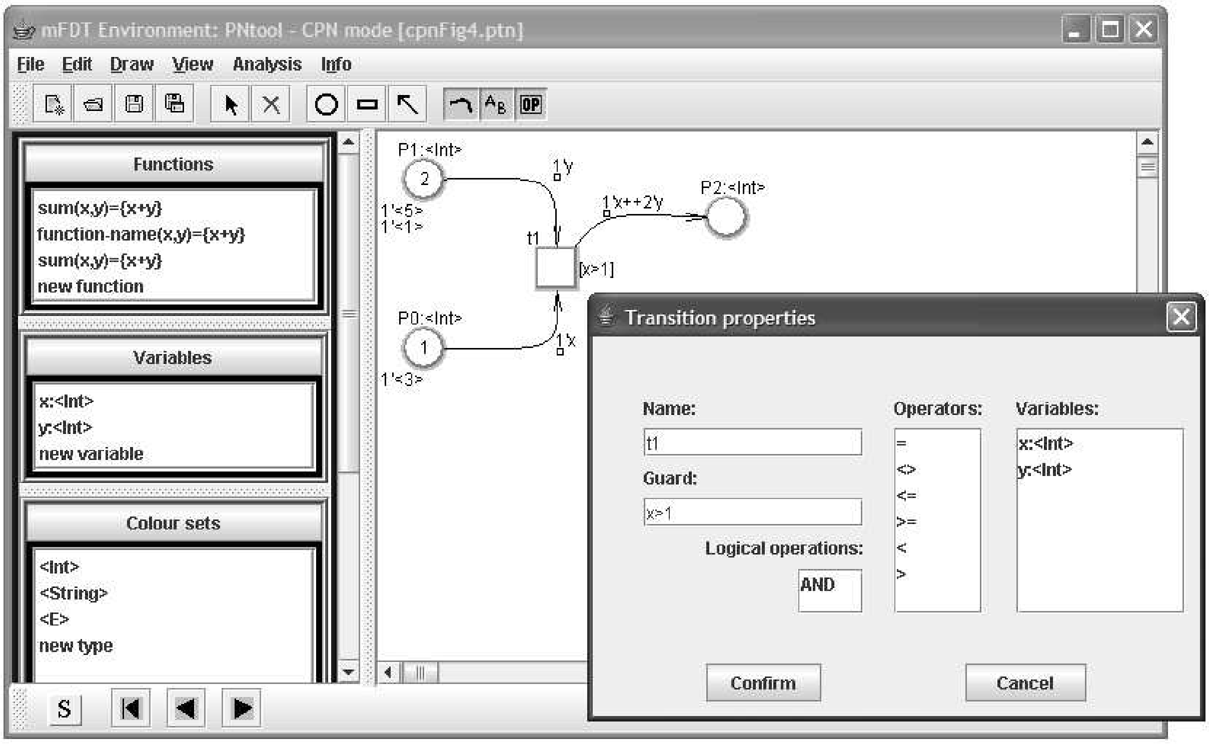
Fig. 7. The place properties window from the EvPN mode

It is also possible to convert a GPN to EvPN and vice versa. Of course, the conversion from EvPN to GPN often leads to loss of some information because of the greater computational power of EvPN.

3.3. CPN mode

As in the case of EvPN mode, the CPN mode offers a graphical editor and simulator. Currently the CPN mode supports integer, string and *E* colour sets. The *E* colour set contains only one element – the constant *e*. Tokens of colour *e* are equal to the tokens of GPN. A new colour sets can be created from the existing ones using the Cartesian product and list operator. The CPN mode also offers basic integer operators (+,-,*,/) for arc expressions and basic integer predicates (>,<,=, ≥ ,≤) for guards. A logical operator "*and*" can be used in guards, too.

An appearance of the *PNtool* in this mode is slightly different from another modes and can be seen in Fig. 8. The most noticeable difference is the presence of a declarations frame (for variables and colour sets) in the left part of the *PNtool* window.



sum(x,y)={x+y}
function-name(x,y)={x+y}
sum(x,y)={x+y}
new function

x:<Int>
y:<Int>
new variable

<Int>
<String>
<E>
new type

Name: t1
Guard: x>1
Logical operations: AND
Operators: = <> <= >= < >
Variables: x:<Int> y:<Int>

Fig. 8. A screenshot of the *PNtool* in the CPN mode

4. PNML and PNtool

The Petri Net Markup Language (PNML) is an XML-based interchange file format for Petri-nets **Ошибка! Источник ссылки не найден.**. The PNML is flexible enough to integrate different types of Petri Nets and is open for future extensions. There is a variety of software tools with the PNML support, such as Petri Net Kernel, Renew, PEP and TINA. The PNML elements allow describing both the structure and graphical appearance of Petri net. Basic PNML elements are:

- <pnml> – the topmost element;
- <net> – contains the description of the whole net;
- <place> – a sub-element of <net>, describes one place of PN;
- <transition> – a sub-element of <net>, describes one transition of PN;
- <arc> – a sub-element of <net>, describes one arc of PN;
- <name> – holds the name of a net, place, arc and so on. It can be used as a sub-element of the previous four elements;
- <initialMarking> – a sub-element of <place>, holds the value of initial marking of given place;
- <inscription> – a sub-element of <arcs>. In the case of GPN it holds the corresponding *pre* or *post* value.

A hierarchical structure of these basic elements, excluding <name>, can be seen in Fig. 9. There is a lot of additional elements, for example <graphics>, which includes information about rendering of given component, such as position and line and fill colours.

The *PNtool* also belongs to the group of tools supporting PNML. It allows saving and loading of GPN and EvPN in PNML format. However, because of some special features of EvPN, it was needed to extend PNML by a few new elements:

- <evpnPlace> – a sub-element of <place>**,** specifies whether the given place belongs to *P*, P_e or P_f,
- <evpnArc> – a sub-element of <arc>, specifies whether a negative value of marking of the place adjacent to this arc is allowed,
- <capacity> – a sub-element of <evpnArc>, specifies a maximum capacity of the place adjacent to this arc,
- <relPlace> – a sub-element of <evpnArc>, includes the name of a place, not connected to this arc, which affects the *pre* or *post* value of this arc.

The place of these new tags within the PNML structure is also shown in Fig. 9.

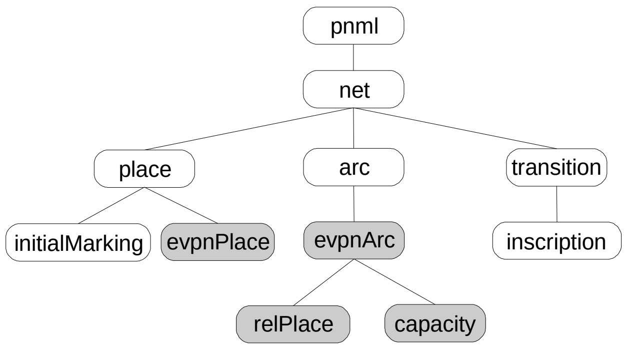


Fig. 9. The hierarchical structure of basic PNML elements. New elements for EvPN are rendered in grey

5. Interfaces

As it has been said before, the *PNtool* is a part of the mFDT Environment, which also incorporates process algebra and B-Method. To interact with the other parts of mFDTE the *PNtool* provides interfaces, which implement the semantics-preserving transformations between Petri nets and process algebra **Ошибка! Источник ссылки не найден.** and between Petri nets and the language of B-Method **Ошибка! Источник ссылки не найден.**.

The theory of transformations between the language of B-Method (also called the B-language) and PN, introduced in **Ошибка! Источник ссылки не найден.**, makes it possible to transform any GPN or EvPN into the computationally equivalent B-machine and almost any B-machine into the equivalent CPN. The B-machine is a specification component of B-Method and its concept is quite close to that of the class in object-oriented programming.

The basic idea of these transformations is to link together the similar behavioural concepts of both methods. Therefore places of PN are transformed to state variables of B-machine, initial marking to initialisation operation, transitions and adjacent arcs to operations and vice versa. Instead of explaining the details of the transformations we present a small example in Fig.10.

The current version of the *PNtool* provides the transformation from GPN or EvPN to the B-language. The far more complicated transformation from the B-language to CPN is about to be implemented in the nearby future.

```
MACHINE EvPNfig3
VARIABLES sv_1, sv_2, sv_3
INVARIANT
 sv_1:NAT & sv_2:NAT & sv_3:NAT
DEFINITIONS
 grd0== sv_1>0 /*from pe1*/
INITIALISATION
 sv_1:=1 || sv_2:=2 || sv_3:=0

OPERATIONS
 op_0= SELECT grd0 & sv_1>=1 THEN
        sv_1:=sv_1 - 1 ||
        sv_3:=sv_3 + 2*sv_2
      END
END
```

Fig. 10. A B-machine transformed from the EvPN shown in Fig. 3

The interface Petri nets – process algebra consist of two parts, namely: linguistic semantics preserving transformation of process algebra ACP **Ошибка! Источник ссылки не найден.** specification into the corresponding Petri net and the operational semantics preserving transformation of (Ordinary) Petri net into the process algebra APC **Ошибка! Источник ссылки не найден.** by the authors.

The first of two transformations mentioned, is based on construction of *elementary nets*, corresponding to atomic actions of the APC specification, including the empty process (ε) and the deadlock (δ). Additionally, *net operations* are introduced, corresponding to operators of the ACP (alternative composition, sequential composition, parallel composition and the encapsulation), allowing composition of Petri nets in order to obtain the resulting net, corresponding to the original specification. Based on theoretical results obtained, a tool ACP2PETRI has been implemented, using the Java programming environment. As an input format for storing APC specifications, stands an XML-based language PAML by the authors. The output, containing the resulting Petri net is written in the PNML.

The aim of the second transformation mentioned, is to construct the APC specification from the source Petri net. The approach is based on creating special variables (named E-variables) for every place of given Petri net, expressing processes initiated in those places. Algebraic semantics is given as a parallel composition of all such variables, whose corresponding places hold token(s) within the initial marking. The ideas described briefly are implemented within the PETRI2APC tool, coded by using the Java programming platform. The input specification is supposed to be in the PNML format, and the resulting specification is written in PAML format. Both auxiliary tools, the ACP2PETRI and the PETRI2APC, have been designed for a close cooperation with the *PNtool* itself.

6. Conclusion

In this paper we described the *PNtool*, a software tool, which allows a design, analysis and development of concurrent and time-critical systems based on the Petri nets formalism.
The *PNtool*, which supports four Petri nets dialogs, provides a lot of features – from the design and simulation of Petri nets to their invariant and reachability analysis. In addition, the *PNtool* implements some original scientific results by the authors: the Evaluative Petri nets formalism **Ошибка! Источник ссылки не найден.**, the original reachability problem solving algorithm and corresponding de/compositional techniques **Ошибка! Источник ссылки не найден.**, the interval semantics of Time-basic nets **Ошибка! Источник ссылки не найден.** and, finally, the semantics-preserving transformations between Petri nets and process algebra **Ошибка! Источник ссылки не найден.** and between Petri nets and the language of B-Method **Ошибка! Источник ссылки не найден.**.

Despite the rich functionality of the *PNtool* there is still a lot to add and improve. In the nearby future we plan to add a PNML support for TB nets. This will, of course, require another extension of PNML. The opportunity to display and edit a graph of M_w automaton in *PNtool* should be a welcome addition, too. The easiest way to achieve this is to represent Mw automata in the form of equivalent GP nets. We also plan to use some of the available 2D graphics frameworks, such as Piccolo **Ошибка! Источник ссылки не найден.**, for the graphical editor and simulator of the *PNtool*.

1. *Hudák, Š, Grofčík, J.* An Environment for Design and Analysis of Time-Critical Systems, Proceedings of EMES'2001, Oradea, Romania, May 24-26, 2001. – P. 66–75.
2. *Šimoňák, S.* Formal methods integration based on Petri nets and process algebra transformations, PhD dissertation thesis, DCI FEEI TU Košice, 2003 (In Slovak).
3. *Baeten, J.C.M., Weijland, W.P.* Process Algebra, Cambridge University Press, 1990. – P. 248.
4. *Abrial, J.R.* The B-book: assigning programs to meanings, Cambridge University Press, 1996.
5. *Hudák, Š.* Reachability analysis of systems based on Petri nets, Elfa, Košice, Slovakia, 1999.
6. *Hudák, Š.* Reachability analysis of systems based on Petri nets, Elfa, Košice, Slovakia, 1999.
7. *Hudák, Š.* Extensions of Petri Nets, Habilitation thesis, Technical university of Košice, Košice, Slovakia, 1980 (in Slovak).
8. *Jensen, K.* An introduction to the theoretical aspects of Coloured Petri nets, A Decade of Concurrency, Lecture Notes in Computer Science, Volume 803, Springer-Verlag, 1994. – P. 230–272, available from: http://www.daimi.au.dk/~kjensen/.
9. *Zaitsev, D.A.* Decomposition-based calculation of Petri net invariants, Proceedings of Workshop on Token based computing of the 25-th International conference on application and theory of Petri nets, Bologna, Italy, June 21-25 2004. – P. 79–83.
10. *Hudák, Š.* The Recursive Decidability of the Reachability Problem for Vector Addition Systems, The Univ. of Newcastle upon Tyne, Comput. Lab, ASM/84, August 1981. – 78 p.
11. *Billington, J, Christensen, S, van Hee, K, Kindler, E, Kummer, O, Petrucci, L, Post, R, Stehno, Ch, Weber, M.* The Petri Net Markup Language: Concepts, Technology, and Tools, ICATPN 2003, Eindhoven, Netherlands, June 2003.
12. *Korečko, Š.* Integration of Petri Nets and B-Method for the mFDT Environment, PhD dissertation thesis, DCI FEEI TU Košice, 2006 (In Slovak).
13. *Šimoňák, S, Hudák, Š, Korečko, Š.* PETRI2APC: towards unifying Petri nets with other formal methods, Proceedings of the Seventh International Scientific Conference Electronic Computers and Informatics ECI 2006, Košice - Herľany, Slovakia, 2006. – P. 140-144.
14. *Piccolo* Toolkit homepage, URL: http://www.cs.umd.edu/hcil/piccolo/