

# Abstraction-enriched Formal Methods Integration

**Slavomír Šimoňák<sup>1</sup>, Matúš Uchnár<sup>1</sup>, Peter Fecilák<sup>1</sup>, Eva Chovancová<sup>1</sup>, Martin Chovanec<sup>2</sup>**

<sup>1</sup>Department of Computers and Informatics, Faculty of Electrical Engineering and Informatics, Technical University of Košice, Letná 9, 042 00 Košice, Slovakia

<sup>2</sup>Institute of Computer Technology, Technical University of Košice, B. Němcovej 3, 042 00 Košice, Slovakia

e-mails: slavomir.simonak@tuke.sk, matus.uchnar@tuke.sk,  
peter.fecilak@tuke.sk, eva.chovancova@tuke.sk, martin.chovanec@tuke.sk

---

*Abstract: The paper presents the results of our research in the field of combining process algebra and Petri nets. To provide better support for design and analysis of larger-scale systems by means of abstraction mechanism, the method itself and the tools allowing its practical application have been enhanced significantly. The theoretical aspects and implementation of enhancements are discussed in detail. Careful testing, along with the process of implementing the new functionality into one of the involved tools, helped us to disclose its certain hidden imperfections, which are subsequently addressed.*

*Keywords: process algebra; Petri nets; formal methods integration; abstraction*

---

## 1 Introduction

Formal methods offer a mathematically-based framework, allowing for a systematic specification, development, and analysis of systems. When applying formal methods to the design and analysis of real-life-sized systems, the usage of different methods and different verification techniques can be very useful. It might be either because a particular formalism is most suitable for the design of an individual component or the designer is interested in different system properties to investigate, or to cope with the complexity of the system [1].

The existence of successful series of conferences on Integrated Formal Methods (iFM), refers to the importance of formal methods integration. The conferences cover all aspects of the integration from language design, through the analysis to the tools and their application in software engineering practice [1]. In September

2017, 13<sup>th</sup> International Conference of this kind (iFM 2017) was organized in Torino, Italy.

Formal methods integration, in this particular case, is based on the transformation of process algebra ACP [6] specifications into the corresponding Petri net representations. Source algebraic specification of a system, supplied by using XML-based PAML language [31], is processed by the ACP2Petri tool, which produces corresponding Petri net based representation of the system in PNML [21] format.

## 1.1 Motivation

Despite the large number of existing formal methods, new methods are currently being developed. In such a situation, it is actually very fruitful to study various combinations of several methods with different characteristics and complementary strengths [7]. Petri nets have an intuitive graphical representation, they allow to describe both the states and the actions of the considered system and offer many analytical techniques [11] for investigation of structural as well as the behavioral properties of a model. Process algebra is a symbolic formalism, which is focused on the dynamic behavior of a system. Algebraic specification usually has no explicit representation of states and available proof techniques are generally aimed at investigating the equality of behavioral descriptions [7]. So we can conclude that Petri nets and process algebra can be considered complementary in several aspects.

According to our experience, it is very useful to build the specification of the system once and to obtain the corresponding specification in different formalisms, after automatic transformation, with almost no effort. For the purposes of the analysis, both models can be used and according to the properties of interest, we can choose the best one.

In many cases, the intuitive graphical representation offered by Petri nets, supports a better understanding of the structure and operation of the system under consideration [15]. In the case of modeling larger systems, however, benefits of graphical representation are less evident with an increasing size and complexity of a system. Algebraic specification in such situations is often much more compact than the corresponding Petri net. On the other hand, the main source of motivation for transforming algebraic specifications into the Petri net formalism is the access to analytical techniques and the results available for Petri nets [18]. The design and analysis of communication protocols can serve as an example of application of the approach, mentioned above [32]. We believe that, in many cases, it is simpler to create a specification for a particular communication protocol using process algebra, rather than Petri nets. It can be done by specifying the communicating entities and the communication medium separately and composing them together by the means of process algebra's parallel composition

operation. When it comes to analysis, the powerful analytical apparatus of Petri nets, including automatic generation of system invariants, is highly appreciated.

After years of using the method [32] and the set of tools associated with it, we recognized the need for a major update. The need was connected with the inability of processing abstraction [17] within specifications of bigger systems. The last significant update was oriented towards a graphical user interface [3] and fixing some shortcomings found within the ACP2Petri tool [34]. Such an update not only provided a better user experience, when using the application, but also allowed for tracing the progress of transformation, in a visual way. This enabled a better understanding of how the process of transformation is implemented and it resulted in disclosure of some shortcomings, which were repaired subsequently.

The fact that one of the most advanced toolsets for the mCRL2 specification language [20], based on process algebra ACP, still does not support recursive parallelism [27], can be perceived as a limitation in some cases. Therefore, it can be considered as another source of motivation.

## 1.2 Related Work

Research in the field of relating both process algebra and Petri nets, two fundamental concurrency theories, is not new and many influential works have been published [13, 26, 29]. On the other hand, the research is still active and it produces new and interesting results. In [18] a calculus (Finite-net Multi-CCS), inspired by CCS, is introduced and provided by a labeled transition system as well as Petri net semantics. The ability to represent finite, statically reduced, P/T nets by well-formed finite-net Multi-CCS processes is shown in [19]. A framework is introduced in [9] where a net encoding can be constructed for calculi using different communication patterns.

A simple process calculus of Petri nets (Petri calculus) is defined in [30]. The main motivation here is to provide the compositional approach for defining the semantics of Petri nets. Within the paper, a compositional extension of Condition/Event nets is introduced. A net is associated with interfaces to which its transitions can be connected. Composition of such nets along a common interface is performed by synchronization of transitions. It is shown that the class of nets with boundaries has the same expressiveness as a simple process calculus.

The relations between the Petri Box Calculus and a class of P/T Petri nets are considered in [10]. PBC terms are carefully designed in order to define the transformation producing P/T nets preserving the structural operational semantics of the terms. In such way a composition of P/T nets is allowed. A unique algebraic semantics for Petri nets, based on process algebra ACP, is introduced in [8]. Actions of the PTNA (Place/Transition-Net Algebra) correspond to production and consumption of tokens by Petri net transitions. It is shown, that both Petri net

and its corresponding algebraic representation have identical operational behavior. The results are further enhanced to hierarchical P/T nets.

Compared to the existing approaches, mentioned above, our approach differs in several aspects. We use widely-adopted formalisms without defining their special extensions, as is the case in many available solutions. This fact implies a reasonable support by existing tools and contributes to its practical application. Process algebra ACP has been selected as a part of our integration framework in this case, as we believe it has its own advantages compared to other well-known process algebras such as CCS and CSP. Compared to the other two, ACP emphasizes the algebraic aspect, more. The equational theory is the central point here. It can be equipped with a range of semantical models [5]. The communication scheme of ACP is also more general, since in CCS communication is combined with abstraction and it is combined with restriction in case of CSP. Last, but not least, there is a software toolset for the implementation of our transformation, therefore, it is much more available, for practical utilization.

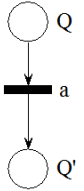


## 2 Theoretical Background

Previous enhancements were connected mainly with the changes within the main transformation tool - the ACP2Petri. The current extension, on the other hand, is more profound and it affects both the theoretical foundations and the supporting tools. The theoretical foundations of the transformation were published in [35] and are only shortly summarized here and extended by the new properties.

Elementary nets represent the basic building blocks of more complex specifications and they correspond to the notion of atomic actions of process algebra ACP. Except those, we defined also the elementary nets corresponding to the empty process ( $\varepsilon$ ) and the deadlock ( $\delta$ ).

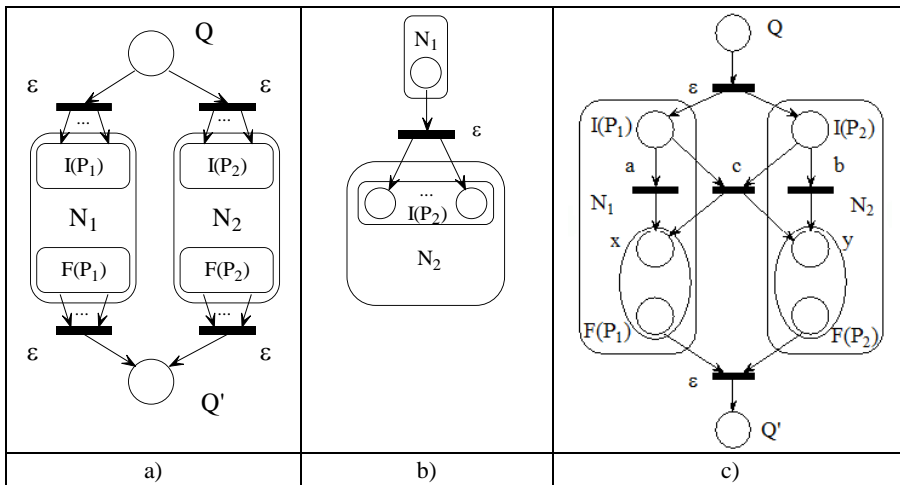
Let a process  $Q$  be represented by the term  $a$  ( $Q=a$ ). Then corresponding elementary net ( $N(Q)=N_a$ ) is given by:  $N_a=(P,T,pre,post)$ , where  $P=\{Q,Q'\}$ ,  $T=\{a\}$ ,  $pre(Q,a)=1$ ,  $post(Q',a)=1$ ,  $I(P)=\{Q\}$ , and  $F(P)=\{Q'\}$ . Here  $P,T$  stand for sets of places and transitions respectively.  $pre()$  and  $post()$  represent pre- and post- transition relation, giving the structure of the net.  $I(P)$  and  $F(P)$  are the sets of initial and final places of the given Petri net, respectively. Here,  $N()$  stands for the mapping from the process term to the corresponding Petri net. In Table 1, Petri net configurations are summarized for all elementary net types, together with their graphical representations.

Table 1  
Elementary nets

$Q = a$	$Q = \varepsilon$	$Q = \delta$
 <p> <math>P = \{Q, Q'\}</math>  <math>I(P) = \{Q\}</math>  <math>F(P) = \{Q'\}</math> </p>	 <p> <math>P = \{Q\}</math>  <math>I(P) = \{Q\}</math>  <math>F(P) = \{Q\}</math> </p>	 <p> <math>P = \{Q\}</math>  <math>I(P) = \{Q\}</math>  <math>F(P) = \{ \}</math> </p>
a)	b)	c)

The net operations are defined, corresponding to the operators of process algebra ACP, which are necessary for expressing the net semantics, of more complex algebraic terms.

Table 2  
Petri net composition operations



The net operations defined correspond to alternative composition (+), sequential composition ( $\cdot$ ), parallel composition with communication ( $\parallel$ ) and encapsulation operation ( $\partial$ ) of the process algebra ACP. The net operations mentioned above are only briefly discussed here. They are depicted in Table 2 and explained deeper in [35]. The alternative composition (case a) in Table 2) of two Petri nets is constructed by enhancing a set of places (given by union of sets of places of composed nets) by two additional places ( $Q, Q'$ ), where  $Q$  will be the initial place and  $Q'$  the final place of the composition. In the case of sequential composition (case b) in Table 2), the final place of the first of composed nets ( $N_1$ ) is connected to the initial place(s) of the second of nets ( $N_2$ ) by the new,  $\varepsilon$ -labeled transition.

As we can observe, the set of final places in the elementary net corresponding to deadlock ( $\delta$ ) is empty, meaning that there is no possibility to append another Petri net to such net by operation of sequential composition, which corresponds to the desired behavior.

The parallel composition of two Petri nets can be slightly more complicated, especially when the communication of the processes represented by the Petri nets is considered. Figure depicted in c) of Table 2 illustrates the situation, where two actions  $a$  and  $b$  are able to communicate and the result of such communication is the action  $c$ . Within the process algebra ACP such communication possibility can be expressed by means of communication function  $\gamma(a, b) = c$ . In the situation depicted by the figure, a Petri net denoted by  $x$  represents the net obtained from the net  $N_l$  by removing its initial place, transition  $a$ , and corresponding arcs. Petri net denoted by  $y$  can be obtained analogically. Petri net corresponding to the application of the encapsulation operation is constructed in such way that transitions labeled by the actions from the encapsulation set ( $H$ ) are removed from the net as well as the arcs connected to those transitions. For expressing the Petri net semantics of APC terms inductive rules were defined:

$$N(Q + R) = N(Q) + N(R) \quad (1)$$

$$N(Q \cdot R) = N(Q) \cdot N(R) \quad (2)$$

$$N(Q \parallel R) = N(Q) \parallel N(R) \quad (3)$$

$$N(\partial_H(Q)) = \partial_H(N(Q)) \quad (4)$$

While on the left side of equations (1) – (4), there are operators (+, ·, ||,  $\partial$ ) of process algebra ACP, operators on the right side of the equations refer to their equivalents on Petri nets. To distinguish them, the Petri net operators are emphasized using the bold face text.

Abstraction [24] is a fundamental mechanism in the design of hierarchical systems. Such mechanism allows us to abstract away from the internal operation of modules from which larger systems are composed. Without such a mechanism, it would be virtually impossible to specify anything useful, except in a very small system [4].

If we want to abstract from certain actions, it does not mean that we can simply remove those actions, because we want to preserve the behavior of original process apart from the abstracted actions [6, 17]. So, the silent step ( $\tau$ ), is introduced, which can be removed in some cases, but cannot be removed in other cases. In [6] two  $\tau$ -laws are formulated, giving the exact behavior of the silent step (Table 3).

Table 3  
Behavior of silent step

$x\tau = x$	B1
$x(\tau(y+z) + y) = x(y+z)$	B2

So the abstraction essentially represents a renaming of given actions into  $\tau$ . The abstraction operator ( $\tau_I$ ) is introduced, which renames all actions from the set  $I$  into  $\tau$ . As a consequence, additional axioms (Table 4) are included to the axiom system of process algebra ACP. It holds that  $\delta \notin I$ , since  $\delta \notin A$  and  $I \subseteq A$ , so  $\delta$  is never renamed into  $\tau$ . In the Table 4 it is assumed that  $a \in A \cup \{\delta, \tau\}$ .

Table 4  
Axioms for abstraction

$\tau_I(a) = a$ if $a \notin I$	TI1
$\tau_I(a) = \tau$ if $a \in I$	TI2
$\tau_I(x+y) = \tau_I(x) + \tau_I(y)$	TI3
$\tau_I(xy) = \tau_I(x) \cdot \tau_I(y)$	TI4

The silent step is not allowed to communicate with other actions and therefore it is defined (5) that communication involving  $\tau$  results in deadlock [17, 6].

$$\tau | a = \delta, \text{ for } a \in A \cup \{\delta, \tau\} \quad (5)$$

The complete axiom system for process algebra ACP extended by the notion of silent step (ACP $^\tau$ ) including axioms B1, B2 and TI1-TI4 can be found in [6].

### 3 Adding Abstraction Support within the Toolset

A new unary operation was added to the existing set of net operations, which corresponds to the application of the abstraction operator ( $\tau_I$ ) of process algebra ACP. The result is a Petri net where the transitions with labels from the set  $I$  are renamed to the silent action  $\tau$ . So the new operation can be expressed more formally in a following way:

$$N(\tau_I(Q)) = \tau_I(N(Q)) \quad (6)$$

While the left side of equation (6),  $\tau_I$  represents the abstraction operator of process algebra ACP,  $\tau_I$  and the right side refers to its equivalent in Petri nets. The toolset including the PATool, as well as, the ACP2Petri needed update and incorporate the new transformation possibilities.

### 3.1 Updating the PATool

The PATool [31] provides some useful functionality supporting the integration of process algebra and Petri nets. It is able to work with the various formats used in algebraic specifications, provides valuable capabilities of conversion and serves as an interface to additional transformation tools, including the ACP2Petri. PATool in this case is used to translate a text based ACP specification to the PAML format, which is suitable for processing by the ACP2Petri. New elements allowing for use of abstraction within specifications were added to the input (text-based) as well as the output (XML-based) language. The input language enhancements include two new statements: `tauset`, for specifying the set of actions to be abstracted away and the `tau` for applying the abstraction renaming to a particular process. The updated DTD specification of the output language can be found in Table 5. Within the table, updated parts are emphasized using the bold face text.

Table 5  
Updated DTD specification

```

<!-- ACP tau DTD for process specifications -->
<!ELEMENT ACPSPEC (GAMMA*,ENCSET*,TAUSET*,ACPEQUATION+)>
<!ELEMENT ACPEQUATION (VAR,ACPTERM)>
<!ATTLIST ACPEQUATION INIT CDATA #REQUIRED>
<!ELEMENT ACPTERM (ALTCMP|SEQCMP|PARCMP|ACTION|VAR|ENCAPS|TAU)>
<!ELEMENT ALTCMP ((ALTCMP|SEQCMP|PARCMP|ACTION|VAR|ENCAPS|TAU),
(ALTCMP|SEQCMP|PARCMP|ACTION|VAR|ENCAPS|TAU))>
<!ELEMENT SEQCMP ((ALTCMP|SEQCMP|PARCMP|ACTION|VAR|ENCAPS|TAU),
(ALTCMP|SEQCMP|PARCMP|ACTION|VAR|ENCAPS|TAU))>
<!ELEMENT PARCMP ((ALTCMP|SEQCMP|PARCMP|ACTION|VAR|ENCAPS|TAU),
(ALTCMP|SEQCMP|PARCMP|ACTION|VAR|ENCAPS|TAU))>
<!ELEMENT ENCAPS (ALTCMP|SEQCMP|PARCMP|ACTION|VAR|ENCAPS|TAU)>
<!ATTLIST ENCAPS ENCID CDATA #REQUIRED>
<!ELEMENT TAU (ALTCMP|SEQCMP|PARCMP|ACTION|VAR|ENCAPS|TAU)>
<!ATTLIST TAU TAUID CDATA #REQUIRED>
<!ELEMENT ACTION EMPTY>
<!ATTLIST ACTION NAME CDATA #REQUIRED>
<!ELEMENT VAR EMPTY>
<!ATTLIST VAR NAME CDATA #REQUIRED>
<!ELEMENT GAMMA EMPTY>
<!ATTLIST GAMMA ACT1 CDATA #REQUIRED ACT2 CDATA #REQUIRED RES CDATA #REQUIRED>
<!ELEMENT ENCSET (ACTION*)>
<!ATTLIST ENCSET ENCID CDATA #REQUIRED>
<!ELEMENT TAUSET (ACTION*)>
<!ATTLIST TAUSET TAUID CDATA #REQUIRED>

```

To illustrate a text-based ACP specification and a corresponding PAML specification we provide an example in Table 6 and Table 7 respectively.



Table 6  
Example of text-based ACP specification

```

gamma (a, b) = c
encset[H] (b)
tauset[I] (d)
X = encaps[H] (tau[I] (Y || Z))
Y = b.e
Z = b.d

```

Within the example,  $\text{gamma}(a, b) = c$  represents the definition of communication function, where two actions  $a, b$  are able to communicate and the result of such communication is the action  $c$ .  $\text{encset}[H](b)$  defines the set of actions ( $H$ ) to be encapsulated, while encapsulation itself is applied by the  $\text{encaps}[H]$  operator. Similarly,  $\text{tauset}[I]$  defines the set of actions ( $I$ ) for renaming to silent step. Abstraction renaming is applied by  $\text{tau}[I]$  to the parallel composition of processes  $Y$  and  $Z$ .

Table 7  
Example of corresponding PAML specification

<pre> &lt;ACPSPEC&gt;   &lt;GAMMA ACT1="a" ACT2="b" RES="c"&gt;&lt;/GAMMA&gt;   &lt;ENCSET ENCID="H"&gt;     &lt;ACTION NAME="b"&gt;&lt;/ACTION&gt;   &lt;/ENCSET&gt;   &lt;TAUSET TAUID="I"&gt;     &lt;ACTION NAME="d"&gt;&lt;/ACTION&gt;   &lt;/TAUSET&gt;   &lt;ACPEQUATION INIT="true"&gt;     &lt;VAR NAME="X"&gt;&lt;/VAR&gt;     &lt;ACPTERM&gt;       &lt;ENCAPS ENCID="H"&gt;         &lt;TAU TAUID="I"&gt;           &lt;PARCMP&gt;             &lt;VAR NAME="Y"&gt;&lt;/VAR&gt;             &lt;VAR NAME="Z"&gt;&lt;/VAR&gt;           &lt;/PARCMP&gt;         &lt;/TAU&gt;       &lt;/ENCAPS&gt;     &lt;/ACPTERM&gt;   &lt;/ACPEQUATION&gt; </pre>	<pre> &lt;ACPEQUATION INIT="false"&gt;   &lt;VAR NAME="x"&gt;&lt;/VAR&gt;   &lt;ACPTERM&gt;     &lt;SEQCMP&gt;       &lt;ACTION NAME="b"&gt;&lt;/ACTION&gt;       &lt;ACTION NAME="e"&gt;&lt;/ACTION&gt;     &lt;/SEQCMP&gt;   &lt;/ACPTERM&gt; &lt;/ACPEQUATION&gt; &lt;ACPEQUATION INIT="false"&gt;   &lt;VAR NAME="z"&gt;&lt;/VAR&gt;   &lt;ACPTERM&gt;     &lt;SEQCMP&gt;       &lt;ACTION NAME="b"&gt;&lt;/ACTION&gt;       &lt;ACTION NAME="d"&gt;&lt;/ACTION&gt;     &lt;/SEQCMP&gt;   &lt;/ACPTERM&gt; &lt;/ACPEQUATION&gt; &lt;/ACPSPEC&gt; </pre>
--	---

With respect to new elements of the language to be processed by the PATool, the core functionality of the tool has been updated to reflect the changes. Now the tool provides conversion of text-based ACP specifications, including the abstraction related elements, to the PAML format, suitable for further processing by the ACP2Petri.

### 3.2 Enhancing the ACP2Petri Tool

The ACP2Petri also required substantial updates to reflect the new transformation properties. It was necessary to create sets of actions defined by the `TAUSET` element of source specification and identified by `TAUID` attribute in order to apply abstraction renaming performed by `TAU` operation to correct actions of the particular process.

Special attention has been paid to more complicated cases, like the nested application of abstraction operator, or its combination with recursion and parallelism, as it is illustrated in Figure 1, while the full specification of example is provided in Table 8.

Table 8  
Specification with application of abstraction operator

```

tauset[I] (a)
tauset[J] (b)
tauset[K] (c)
X = tau[I] (tau[J] (tau[K] (a.b.c.d. (X || X))))

```

Within the specification in Table 8, all the actions ( $a, b, c$ ) except the action  $d$  are renamed to silent action by the means of nested application of abstraction operator.

However, not all actions renamed to silent action  $\tau$  can be simply removed from the resulting Petri net. The reasons were discussed in section 2 of this paper and are connected with the effort to preserve the behavior of the original process apart from the abstracted actions. To remove those silent actions, which can be removed safely, we implemented a special "Tau removing mode" functionality within the tool for the case the system designer wishes to remove them. This functionality can be switched on using the command line option `(-t)` when starting the ACP2Petri tool.

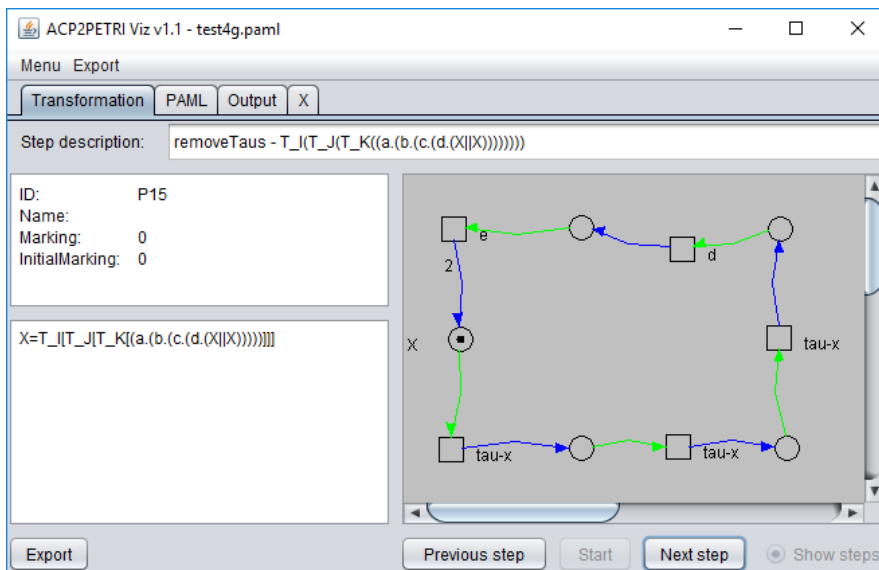


Figure 1

Application of abstraction operator in ACP2Petri

Since the program has been started with the  $-t$  option for removing silent actions, all three of them (named  $\tau$ -x in Figure 1) are removed in the next step and the resulting Petri net is depicted in Figure 2. As it was stated above, not all of the silent actions can be removed automatically.

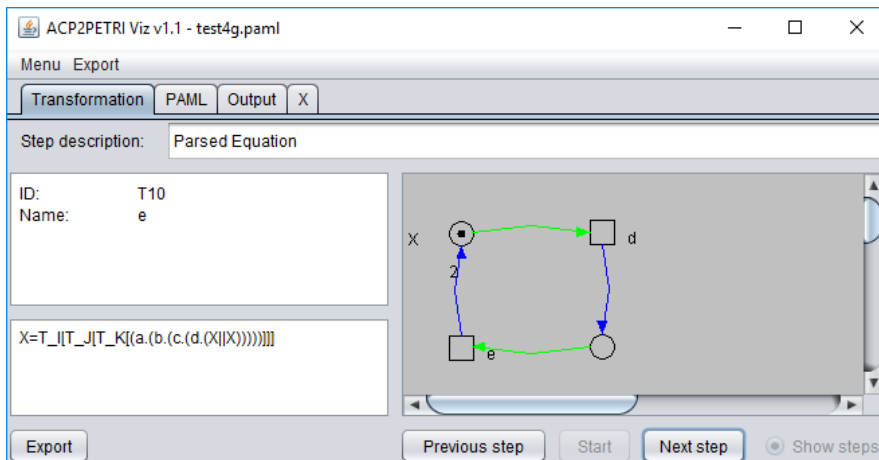


Figure 2

Removing silent actions in ACP2Petri

To illustrate the difference, we provide another example (Table 9) containing also the silent action which will not be automatically removed by the tool.

Table 9  
Specification containing abstraction operator

```
tauset [I] (b)
X = tau[I] (b. (a.b + b.a))
```

According to the specification above the set  $I$  contains single action  $(b)$  to be abstracted away within the process  $X$  by the application of the abstraction operator. All three occurrences of the action  $b$  are renamed to silent action  $\tau$ , but only two of them are scheduled to be removed by the tool automatically and denoted by  $tau-x$  label, as it is shown in the Figure 3.

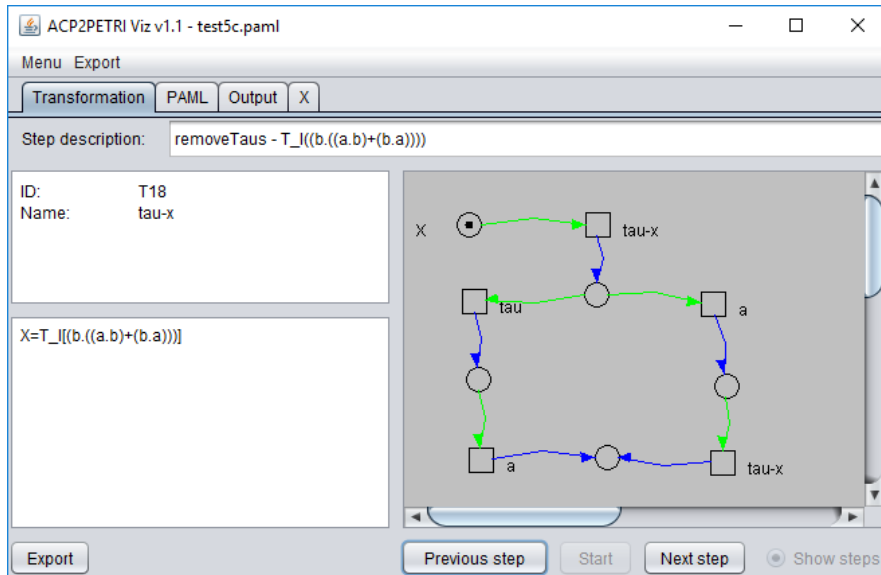


Figure 3  
Silent actions in Petri net

As it can be observed, only two of three silent actions have been removed (described as  $tau-x$  in the Figure 3) by the tool, while one (described as  $tau$ ) has been retained. The resulting Petri net is shown in Figure 4, containing two  $a$ -labeled actions and one silent action  $tau$ .

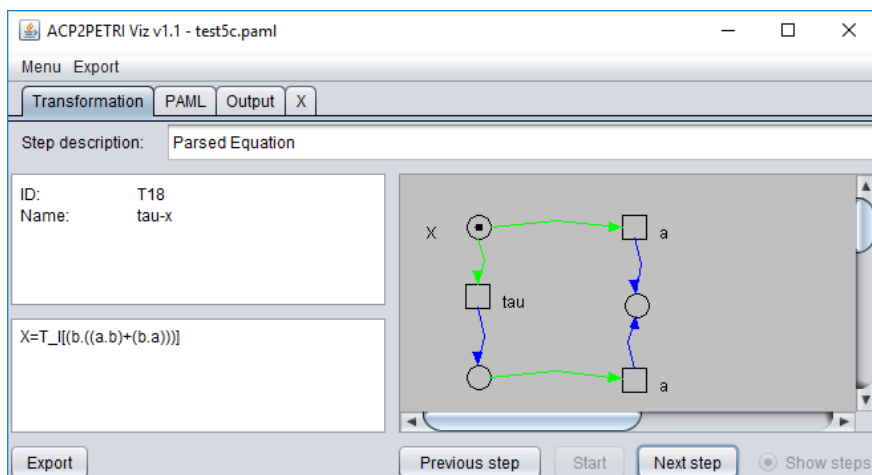


Figure 4

Removing selected silent actions by the ACP2Petri tool

The most important extensions of the tool were described within this section; however, some of the less apparent, but still very useful updates are described in the following one.

## 4 Additional Improvements

Updates we are discussing within the paper, not only introduced the abstraction as a fundamental design mechanism, but lead also to some additional corrections to the process of transformation. An extensive testing of newly implemented extensions allowed us to uncover certain hidden imperfections within some of the older functionality of the ACP2Petri tool. One of such imperfections is illustrated by small example given in Table 10, where recursion and encapsulation are used together.

Table 10

Repairing imperfections present in older version of the tool

```

gamma(a,b) = c
encset[H](c)
X = encaps[H](a.b.X)

```

The older version of the tool (Figure 5) was not able to handle this combination of operations correctly and did not identify the place holding a token with the place marked by  $X$ . In such form the resulting Petri net does not reflect the full behavior of the process  $X$ . It is easy to spot the difference in a small system like this, and adjust it eventually, but for larger-scale systems it could induce serious problems.

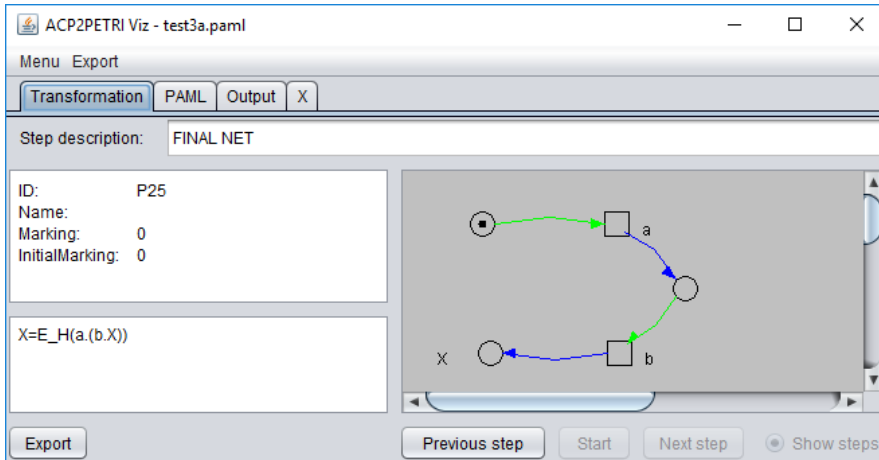


Figure 5

Incorrect Petri net produced by the older version of the tool

The correct Petri net representation of the process, produced by the current version of the tool, is shown in Figure 6. In some rare, more complicated cases, we even found that the transformation was not finished successfully and the exception was generated. Regardless of our current effort, only a further, more practical exploitation of the tool can show if there are some additional shortcomings.

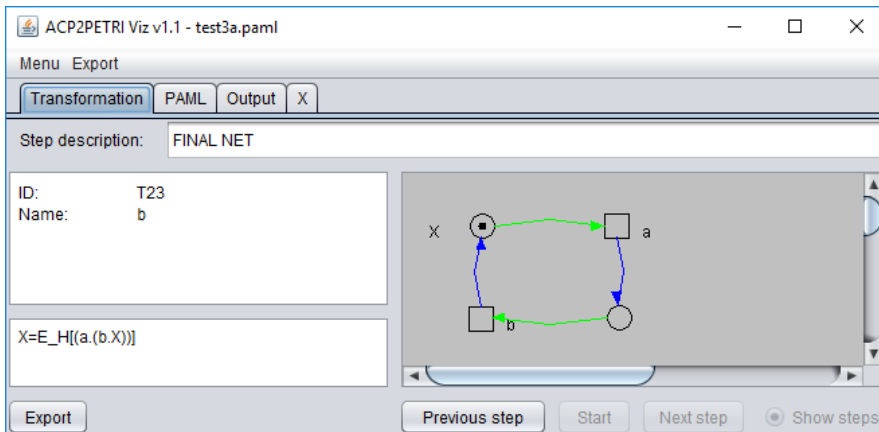


Figure 6

Correct Petri net produced by the current version of the tool

The additional improvement is connected with the ability to further simplify the Petri net generated by the tool in case if it is possible. The Petri net of CB system (Figure 7) can serve as an example, whose operation will be detailed within the next section of the paper. Here the place  $X$  and the  $e$ -labeled transition,

representing the empty action, are present within the net, for the purpose of distributing tokens to the places  $B_{12}$  and  $B_{23}$ , representing the initial places of concurrently working components of the system.

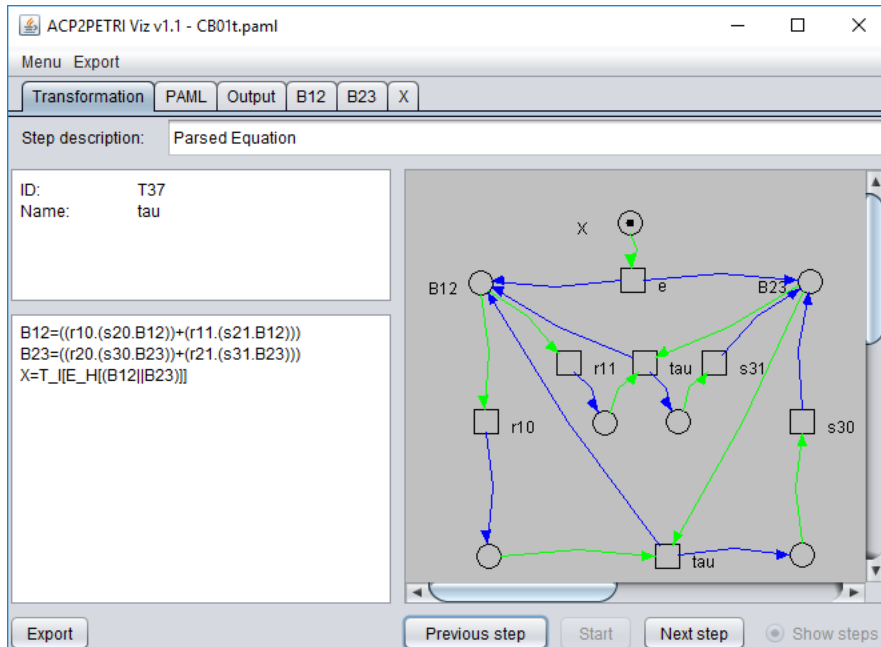


Figure 7

Petri net of the CB system

When places  $B_{12}$  and  $B_{23}$  are marked, then the place  $X$ , the  $e$ -labeled transition that is connected to the place and the corresponding arcs are not required for the correct operation of the system and can be removed. Such functionality was incorporated into the ACP2Petri tool and can be activated using the `-i` command line option. The resulting Petri net can be found in Figure 10.

## 5 Illustrating Example

In this section, there is a small practical example exposing also the newly adopted properties of the transformation presented. We have chosen a Coupling buffers (CB) system adapted from [6], which is depicted in Figure 8. The system represents the buffer with a capacity for two items, composed from two one-element buffers ( $B_{12}$  and  $B_{23}$ ).

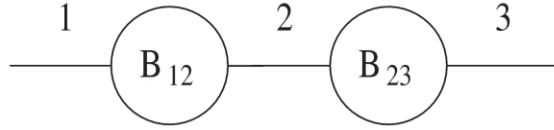


Figure 8  
Coupling buffers system

Ports 1 and 3 of the system represent input and output ports, respectively, whereas port 2 is the internal one, allowing for communication between the two buffers ( $B_{12}$  and  $B_{23}$ ). The algebraic specification of the CB system is given in Table 11.

Table 11  
Specification of the CB system

```

gamma(s20, r20) = c20
gamma(s21, r21) = c21
encset[H](r20, s20, r21, s21)
tauset[I](c20, c21)

B12 = r10.s20.B12 + r11.s21.B12
B23 = r20.s30.B23 + r21.s31.B23
X = tau[I](encaps[H](B12|B23))

```

According to the specification, one-element buffer can read element 0 or 1 at its input port and send the read element to its output port. In case of the buffer  $B_{12}$ , the input port is port 1, so action reading element 0 from this port is named  $r_{10}$  and  $r_{11}$  in case of reading the element 1. The buffer can then send the element that is just read to output port 2. Two buffers are composed using the parallel composition operator and they can communicate over internal port 2, which is expressed by the means of communication function ( $\gamma$ ). Internal actions of the system ( $r_{20}$ ,  $s_{20}$ ,  $r_{21}$ ,  $s_{21}$ ) are further encapsulated by the encapsulation operator ( $\text{encaps}$ ). In this way, the communication between buffers  $B_{12}$  and  $B_{23}$  over internal port 2 is expressed by actions  $c_{20}$  and  $c_{21}$ . Since we are interested only in an external behavior of the CB system, the actions just mentioned, are abstracted away using the  $\tau$  operator.

A simulation of the CB system operation was performed using the PSF Toolkit [14]. Actions executed by the system (recorded in the TRACE window) are indicated by *atom* prefix, and the internal (hidden) actions are indicated by *com.skip* prefix as it is illustrated in Figure 9. The specification of the system, using the PSF language, based on process algebra ACP, is given in the PSF window of the figure.



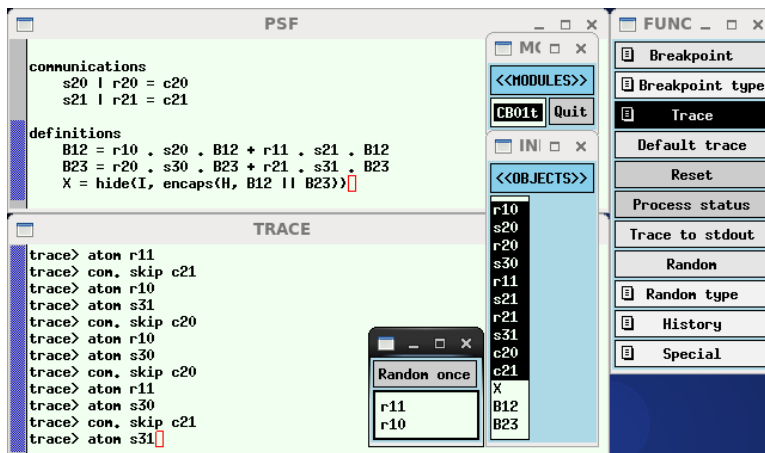


Figure 9

Simulation of the CB system using the PSF Toolkit

After processing the specification from Table 11, by the tools described within the paper, the resulting Petri net can be imported by the variety of available Petri net tools for further processing. Wolfgang Petri net editor [36] e.g. allows us to display the execution trace (Figure 10), which can be easily compared to the trace we got by simulating the algebraic specification using the PSF Toolkit. After removing silent actions we can observe that we were able to simulate the same sequences of actions. In general, for the CB system we can conclude that the ordering of elements entering the system on its input port 1 is the same as ordering of elements leaving the system via the output port 3. Our simulation-based observations correspond to this behavior.

A further analysis can be done by e.g. means of the structure theory of Petri nets [11], which investigates what behavioral properties of particular Petri net can be deduced from its structural properties. By means of instruments like reachability graph, coverability graph, S-Invariants, and T-invariants, many important properties of the system considered can be investigated. The petri net of the CB system, depicted in Figure 10, according to the analysis performed using the tool Netlab [28] is reversible, live, bounded and it has no deadlock.

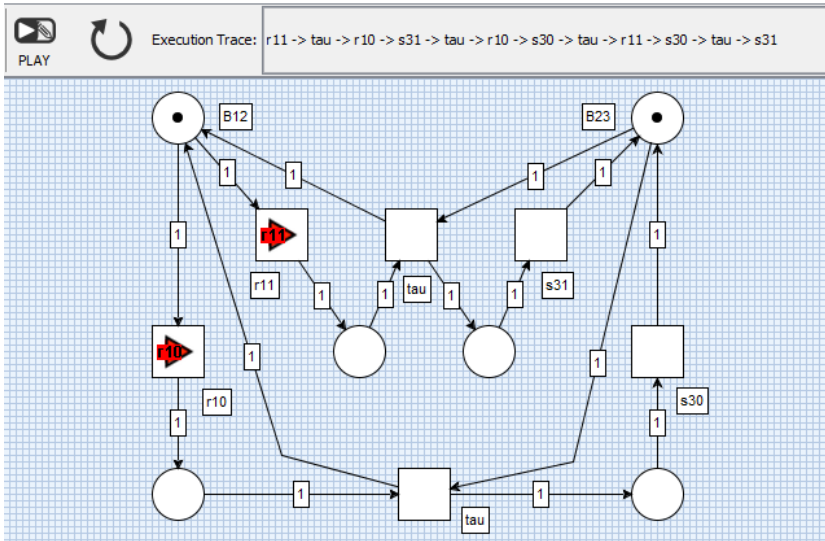


Figure 10

Simulation of the CB system using the Wolfgang tool

## Conclusions

Within the paper we described our latest effort in the field of combining process algebra and Petri nets. The method has been updated by the means required for utilizing the benefits of the abstraction mechanism in order to provide better support for a convenient design and analysis of larger-scale systems.

Except the method itself, the tools allowing the practical application of its benefits were also extended, in order to support the new features. The PATool has been updated to reflect the new enhancements available for use in ACP specifications. The transformation tool, ACP2Petri, now supports abstraction mechanism too, together with the possibility to remove the silent actions, which can be removed safely. The extensive testing within the process of implementing new functionality helped us to uncover some hidden imperfections, which have been subsequently addressed too.

Comparing our method to other integration approaches combining Petri nets and process algebra we consider the availability of its software implementation to be a real benefit allowing for its practical utilization. We can mention its successful application in the field of communication protocols [32, 33], and we believe there will be further promising areas of application after the current update.

We consider support for handling data as one of future extensions, since processes can be understood as mechanisms for the data manipulation [16]. There are process algebras with data support available [12] as well as high-level Petri nets [22] giving such idea a real outline.

Including the notion of time into the process of transformation would also allow us to integrate some of the time-enabled process algebras [12] and Petri nets [37, 38] in order to model and study time-critical systems.

Another extension possibility is connected with the integration of additional formal method allowing for precisely determined and well defined development process, such as the B method [2]. The method allows for developing a system specification in form of a collection of components called B-machines with formally proven properties. It enables to refine an abstract specification into the concrete realm, which can then be translated into programming language [25]. Some activities have been done [23] in this area and it would be very interesting to continue in this direction, in order to utilize the new results in the area of software engineering. It would help to take important steps on the way to generating implementations of verified software components.

### References

- [1] Ábrahám, E., Huisman, M. (Eds.): Integrated Formal Methods, 12<sup>th</sup> International Conference, IFM 2016, Reykjavik, Iceland, June 1-5, 2016, Proceedings
- [2] Abrial, J. R.: Modeling in Event-B: System and Software Engineering, Cambridge University Press, New York, NY, USA, 2010
- [3] Bačíková, M., Porubán, J., Lakatoš, D.: Defining Domain Language of Graphical User Interfaces, Slate 2013: 2<sup>nd</sup> Symposium on Languages, Applications and Technologies, June 20-21, 2013, Porto, Portugal. Wadern: Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2013 pp. 187-202
- [4] Baeten, J. C. M. (ed.): Applications of process algebra, Cambridge University Press, 1990
- [5] Baeten, J. C. M.: A Brief History of Process Algebra, Theoretical Computer Science 335, 2005, pp. 131-146
- [6] Baeten, J. C. M., Weijland, W. P.: Process Algebra, Cambridge University Press, 1990
- [7] Basten, T.: In Terms of Nets: System Design With Petri Nets and Process Algebra, Eindhoven University of Technology, 1998
- [8] Basten, T., Voorhoeve, M.: An Algebraic Semantics for Hierarchical P/T Nets, Computing Science Report, Eindhoven University of Technology, 1995
- [9] Baldan, P., Bonchi, F., Gadducci, F., Monreale, G., V.: Asynchronous Traces and Open Petri Nets, In Programming Languages with Applications to Biology and Security, Volume 9465 of the series Lecture Notes in Computer Science, pp. 86-102, Springer International Publishing, 2015

- 
- [10] Best, E., Devillers, R., Koutny, M.: Petri Nets, Process Algebras and Concurrent Programming Languages, Lectures on Petri Nets II: Applications, Volume 1492 of the series Lecture Notes in Computer Science, Springer Berlin Heidelberg, 1998, pp. 1-84
  - [11] Best, E., Wimmel, H.: Structure Theory of Petri Nets, In Transactions on Petri Nets and Other Models of Concurrency VII, Volume 7480 of the series Lecture Notes in Computer Science, pp. 162-224, Springer Berlin Heidelberg, 2013
  - [12] Cranen S. et al.: An Overview of the mCRL2 Toolset and Its Recent Advances. In: Piterman N., Smolka S. A. (eds) Tools and Algorithms for the Construction and Analysis of Systems. TACAS 2013. Lecture Notes in Computer Science, Vol. 7795. Springer, 2013
  - [13] Desel, J., Juhás, G., Lorenz, R.: Process Semantics of Petri Nets over Partial Algebra, Proceedings of ICATPN 2000, 21<sup>st</sup> International Conference on Application and Theory of Petri Nets, Vol. 1825 of LNCS, pp. 146-165, Springer-Verlag, 2000
  - [14] Dierkens, B.: Software Engineering with Process Algebra, Ph.D. Thesis, University of Amsterdam, 2009
  - [15] Ermel, C., Bardohl, R., Ehrig, H.: Specification and Implementation of Animation Views for Petri Nets, Proceedings of 2<sup>nd</sup> International Colloquium on Petri Net Technologies for Modelling Communication Based Systems, Berlin, Germany, 2001
  - [16] Fokkink, W.: Modelling Distributed Systems, Protocol Verification with  $\mu$ CRL, 2<sup>nd</sup> edition, Springer, 2011
  - [17] Fokkink, W.: Introduction to Process Algebra, 2<sup>nd</sup> edition, Springer-Verlag, 2007
  - [18] Gorrieri, R., Versari, C.: A Process Calculus for Expressing Finite Place/Transition Petri Nets, Proceedings of 17<sup>th</sup> International Workshop on Expressiveness in Concurrency, EXPRESS'10, Paris, France, 2010
  - [19] Gorrieri, R.: Language Representability of Finite P/T Nets, In Programming Languages with Applications to Biology and Security, Volume 9465 of the series Lecture Notes in Computer Science pp. 262-282, Springer International Publishing, 2015
  - [20] Groote, J. F., Keiren, J. J. A., Stappers, F. P. M., Wesselink, J. W., Willemse, T. A. C.: Experiences in developing the mCRL2 toolset. In Software: Practice and Experience, 41(2): 143-153, 2011
  - [21] Hillah, L. M., Kindler, E., Kordon, F., Petrucci, L., Trèves, N.: A primer on the Petri Net Markup Language and ISO/IEC 15909-2, Petri Net Newsletter Vol. 76, pp. 9-28, October 2009

- [22] Hostettler, S. P., et al.: High-Level Petri Net Model Checking with AlPiNA. *Fundamenta Informaticae*, 2011, Vol. 113, No. 3-4, pp. 229-264
- [23] Hudák, Š., Zaitsev, D. A., Korečko, Š., Šimoňák, S.: MFDTEPntool - a tool for the rigorous design, analysis and development of concurrent and time-critical systems, *Acta Electrotechnica et Informatica*, Vol. 7, No. 4, 2007, pp. 5-12
- [24] Kollár, J., Pietriková, E., Chodarev, S.: Abstraction in Programming Languages According to Domain-Specific Patterns, *Acta Electrotechnica et Informatica*, Vol. 12, No. 2, 2012, pp. 9-15
- [25] Korečko, Š., Sobota, B.: Petri nets to B-language transformation in software development, *Acta Polytechnica Hungarica*, Vol. 11, No. 6, 2014
- [26] Lodaya, K.: A regular viewpoint on processes and algebra, *Acta Cybernetica* 17, 751-763, 2006
- [27] mCRL2 language reference, Technische Universiteit Eindhoven, [http://www.mcr12.org/web/user\\_manual/language\\_reference/process.html](http://www.mcr12.org/web/user_manual/language_reference/process.html)
- [28] Netlab, <https://www.informatik.uni-hamburg.de/TGI/PetriNets/tools/db/netlab.html>
- [29] Olderog, E. R.: *Nets, Terms and Formulas*, Cambridge University Press, 1991
- [30] Sobocinski, P.: Representations of Petri net interactions, *CONCUR 2010 - Concurrency Theory*, Volume 6269 of the series *Lecture Notes in Computer Science*, pp. 554-568, 2010
- [31] Šimoňák, S., Peřko, I.: PATool – A Tool for Design and Analysis of Discrete Systems Using Process Algebras with FDT Integration Support, in *Acta Electrotechnica et Informatica*, Vol. 10, No. 1, 2010, pp. 59-67
- [32] Šimoňák, S.: Verification of Communication Protocols Based on Formal Methods Integration, *Acta Polytechnica Hungarica*, Vol. 9, No. 4, 2012
- [33] Šimoňák, S., Hudák, Š., Korečko, Š.: Protocol Specification and Verification Using Process Algebra and Petri Nets, *Proceedings of CSSim 2009*, pp. 110-114
- [34] Šimoňák, S., Šolc, M.: Enhancing Formal Methods Integration with ACP2Petri. *Journal of Information and Organizational Sciences*. Vol. 40, No. 2 (2016) pp. 221-235
- [35] Šimoňák, S., Hudák, Š.: Petri net semantics for ACP terms, *Acta Electrotechnica et Informatica*, Vol. 4, No. 1, 2004 (in Slovak)

- [36] Wolfgang - Tool Documentation, BPSec group, Department for Telematics, Institute for Computer Science and Social Studies, University of Freiburg, Germany. Available at: <http://doku.telematik.uni-freiburg.de/wolfgang>
- [37] Zaitsev, D.: Clans of Petri Nets: Verification of protocols and performance evaluation of networks. LAP LAMBERT Academic Publishing, 2013
- [38] Zhang, X., Yao, S.: Bank switching performance verification with object-oriented timed Petri nets, Proceedings of the 2013 IEEE 8<sup>th</sup> Conference on Industrial Electronics and Applications, ICIEA 2013, Melbourne, Australia, pp. 1664-1669