UDC 004:519.8
Original Scientific Paper

Enhancing Formal Methods Integration with ACP2Petri

Slavomír Šimoňák *slavomir.simonak@tuke.sk*
Faculty of Electrical Engineering and Informatics
Technical University of Košice, Košice, Slovak Republic

Martin Šolc *martin.solc@student.tuke.sk*
Faculty of Electrical Engineering and Informatics
Technical University of Košice, Košice, Slovak Republic

Abstract

The paper deals with the ACP2Petri tool, providing a transformation of process algebraic specification to equivalent Petri net-based specification. Long-term practical experiences with the tool revealed some suggestions for its update and extension. Shortcomings and limitations found are described and proposed solutions provided within the paper. Implemented extensions, simplifying the usage of the tool and providing more options for analysis of particular transformation, are also presented. One of the most evident extensions of the tool is its graphical user interface, which allows for convenient management and detailed control over the process of transformation.

Keywords: ACP, process algebra, ACP2Petri, formal methods integration, Petri nets

1. Introduction

Formal methods can be applied in different phases of the system development: from the initial requirements, through design, implementation, debugging, to maintenance and verification of the system. Their application at an early development stage of the system can help to recognize and reveal design mistakes, which could otherwise been found only in extensive testing and debugging [1]. At a later development stage they might help to determine the correctness of the implementation or eventually determine equivalence of different system implementations.

The complexity of the development process of discrete systems predetermines, that it is not possible to expect the existence of just one universal formal method, which would cover all aspects of the development process. For this reason various researches explore possibilities in the field of formal methods integration. Usually such formal methods are chosen, functions and features of which are complementary in some respect. Many on importance of formal method integration also suggests a successful series of conferences IFM (Integrated Formal Methods), organized since 1999. Conference represents a place for discussions on effective ways to design

hardware and software in order to increase their reliability and robustness. IFM conferences can also be perceived as a contribution to cope with the complexity of design and analysis of systems by combination of paradigms of specification and design such that at different stages of life cycle, most suitable tools are used [2]. This partially is a question of combination of specification formalisms for creating integrated notation, able to cover wider area of design possibilities, but very important is also the question of verification using integration of formal methods. As an example we can notice Forte, an environment for industrial hardware verification developed and utilized by Intel, based on combination of model checking and theorem proving integrated within a functional programming language [3].

As another example can serve the integration of Petri nets and B-method. While Petri nets offer a simple graphical representation, valuable analytical features and can express parallelism in a natural way, the B-method in turn provides a precisely determined and well-defined development process, that allows to specify the system as a collection of components, B-machines, and enables an abstract specification to be refined into the concrete one, which can be easily translated into programming language [4]. There are two basic types of embedding of Petri nets into the B-method: shallow and semantic embedding. Approach presented in [4] can be seen as an example of former, while approach of [5] as a latter type of embedding.

There has been many approaches proposed to combine Petri nets and process algebra too, but we briefly mention only some of them in the following text. While the exceptional properties of Petri nets were mentioned above already, we believe that operations like de/composition and communication can be expressed more naturally using special algebraic constructs than using Petri nets [6]. A CCS-like calculus is introduced in [7], provided by a labeled transition system and a P/T net semantics. Well-formed processes represented by the calculus are able to represent finite P/T nets [8]. In [9] a Petri calculus is defined, related to Petri nets with boundaries and the same expressiveness of both is shown. In [10] we provided an APC semantics for Petri nets. APC (Algebra of Process Components) [11] has been defined in order to describe processes represented by the class of (ordinary) Petri nets in a natural way. Research in the area of combining Petri nets and process algebra includes also many older, but highly influential works, like [12], [13] or [14]. Our approach to combining Petri nets and process algebra ACP (Algebra of Communicating Processes), based on composition of Petri nets by means of operations corresponding to operations of ACP was introduced in [15]. To facilitate its practical application, the ACP2Petri supporting tool was developed later [16], allowing for transformation of ACP-based specification to Petri nets.

2. The ACP2Petri tool

The tool has been practically used with only minor tweaks for several years. For example, it played an important role in analyzing communication protocols [6]. Over the course of its deployment, however, some shortcomings have been revealed, limiting its practical application, or even lead to incorrect results in some situations. Within the paper we therefore summarize the shortcomings found and propose the

appropriate solutions. To further improve and enhance the possibilities of the tool, new options for detailed analysis of system transformation were also implemented, together with a packet of extensions including graphical user interface [17], allowing for stepping of transformation process, possibility to export each step represented by Petri nets into a file, comparison of two Petri nets or simple Petri nets editor. The editor enables for quick adjustments to the layout of nets generated within the process of transformation, which can be sometimes complicated and less intuitive.

The tool is implemented using Java programming language, providing it a good degree of platform independence and a rich availability of packages for processing XML files. The source specification is written using an XML-based language PAML (Process Algebra Markup Language) [18], designed to represent the algebraic specifications in a machine-readable form. Resulting Petri net is written in a form of the PNML (Petri Net Markup Language) file, which is accepted by many Petri net tools.

3. Analysis of limitations and shortcomings of the ACP2Petri

At the beginning it was found, that in the case of reshuffling equations defining the processes of the ACP specifications, resulting Petri nets did not match. This behavior was further analyzed, where a visualization of the actual transformation process played a very important role. As a consequence, the following imperfections were observed:

- application of communication functions to incomplete nets,
- application of the encapsulation operator to an incomplete net,
- missing application of net optimizations after the encapsulation,
- equations order affects the output net,
- PNML output file does not match the PNML standard,
- missing identification of the initial place,
- incorrect transformation of processes, which definition contains parallel composition with itself.

Some of the mentioned shortcomings are described and illustrated using simple examples below.

`gamma(r2, s2) = c2`
`gamma(r6, s6) = c6`
`encset[H](s2, r2, r6, s6)`
`K = (s6 + r2).K`
`S1 = r1.S1d`
`S1d = s2.T1d`
`T1d = r6.S1d + r3`
`S = S1.S`
`ABP = encaps[H](S || K)`
	`gamma(r2, s2) = c2`
	`gamma(r6, s6) = c6`
	`encset[H](s2, r2, r6, s6)`
	`K = (s6 + r2).K`
	`T1d = r6.S1d + r3`
	`S1d = s2.T1d`
	`S1 = r1.S1d`
	`S = S1.S`
	`ABP = encaps[H](S || K)`

a)	b)

Table 1. Example 1, a) incorrect order of equations, b) correct order of equations.

Examples do not represent any real system in order to keep them as simple as possible and to illustrate the majority of deficiencies found within the tool at the same time. ACP specification of the Example 1 can be found in the Table 1. Equations in ACP specifications defining processes of the system in Example 1 are intentionally arranged in the order, which allows us to capture the most of the shortcomings found within the operation of the ACP2Petri tool.

3.1. Application of communication functions to incomplete nets

Communication between actions of two processes occurs within a parallel composition of processes, if communication function for given actions is defined. If the parallel composition of the system is processed before the individual processes forming the composition are represented by corresponding Petri nets, then communication functions are applied to incomplete nets. When equations defining processes are in the order given by the case a) of Example 1, then the final composition of the system, will be applied to incorrect Petri net representations of processes, which are involved in the composition:

```
ABP = encaps[H](S||K)
```

While in both cases transformation works with the correct net representation of the process K (see Figure 1), this is not true for the process S. As we can see in the Figure 2, the case a) contains incomplete representation of process S, whereas the case b) correct one.

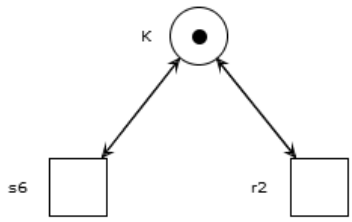

Figure 1. Both cases of Example 1 use correct representation of the process K.

When parallel composition is applied to incomplete net as it is in the case a) of Example 1, communication is not performed, even though the following communication functions are defined:

```
γ(r2,s2) = c2
γ(r6,s6) = c6
```

Transitions s6 and r2 are then removed by subsequent encapsulation operation from the process K. As a consequence, the resulting net in Figure 3, case a) does not contain transitions c2 and c6, representing communication between the processes K and S, as it is within the correctly generated net depicted at Figure 3, case b).

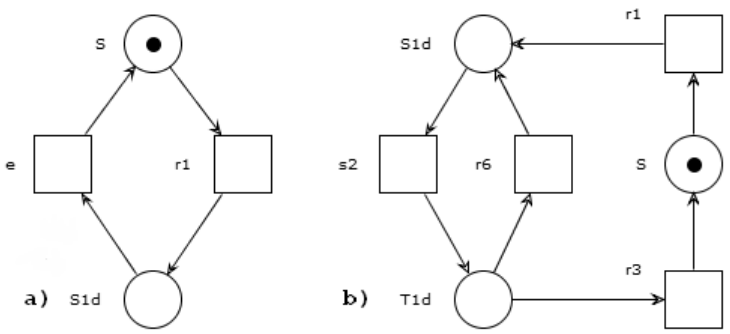

Figure 2. Process S, a) incomplete Petri net representation of process S, b) correct representation of process S.

3.2. Application of encapsulation operator to an incomplete net

As it was mentioned above, after parallel composition with incorrect net-based representation of the process S, communication does not occur, which will further affect the resulting net. To such composition, the encapsulation operation is applied:

```
ABP = encaps[H](S||K)
```

Since the encapsulation set is defined by `H = {s2,r2,r6,s6}`, transitions s6 and r2 are removed from the process K. Resulting Petri net representations for specifications given by Example 1 can be found in Figure 3.

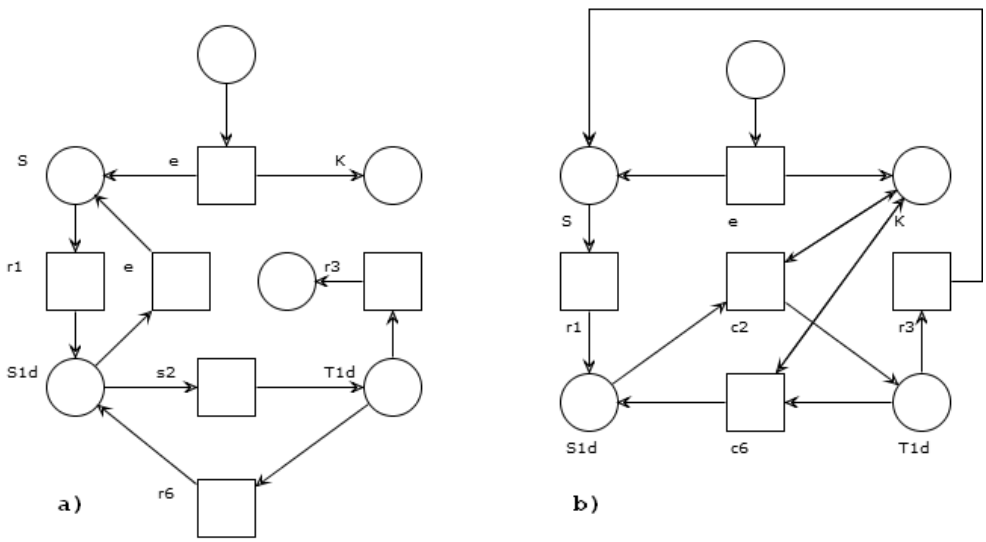

Figure 3. Resulting Petri net representation for specification from Example 1, a) incorrect, b) correct representation.

3.3. Missing net optimization after the encapsulation

This behavior was detected during the visualization of a transformation, when a net was displayed after encapsulation, but was not optimized. Specification, which induced mentioned behavior can be found in Table 2.

```
gamma(r2, s2) = c2
gamma(r6, s6) = c6
encset[H](s2, r2, r6, s6)
K = (s6 + r2).K
ABP = encaps[H](S || K)
S1d = s2.T1d
T1d = r6.S1d + r3
S1 = r1.S1d
S = S1.S
```

Table 2. Example 2.

Generated Petri net is depicted in the Figure 4, where an arc coming from the place S heads to a Ɛ-transition and further to an unlabeled place. It is a type of transition, which together with the unlabeled place had to be removed in the process of optimization. In this case, an exception was thrown and the transformation failed.

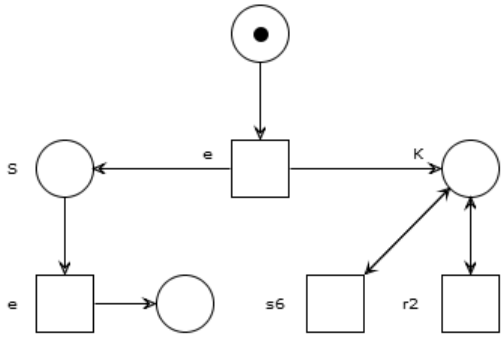

Figure 4. Missing optimization.

In the case a) of Example 1, however, transformation was completed, but generated Petri net was not correct (case a) of Figure 3). Such situation is highly tricky, as a user is not notified on possibly incorrect result.

3.4. Influence of order of equations to resulting net

Order of equations has an influence on the output net not only in the case of parallel composition and encapsulation as it was mentioned above, but also in the case of other operations utilized in definitions of processes. In the case a) of Example 1 processes contributing to definition of the process S are given by equations in the following order:

```
S1 = r1.S1d
S1d = s2.T1d
T1d = r6.S1d + r3
S = S1.S
```

The definition of process S contains the process S1 and the S1 includes the process S1d in its definition. The problem of determining the order comes with processes S1d and T1d, because the process S1d includes within its definition the process T1d and vice versa. The original transformation algorithm works in a way that the net representation of the processed definition is saved into the list of processed definitions. When the processed definition is used within the definition of another process, in this case the process T1d in the definition of process S1d, the net representation of processed definition is used in composition and removed from the list. Situation is depicted in the Figure 5, where the case a) represents incorrect and case b) correct representation of the process S1d.

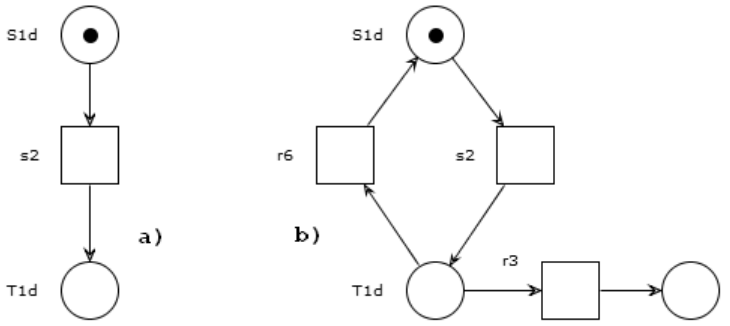

Figure 5. Petri net representations of the processes S1d.

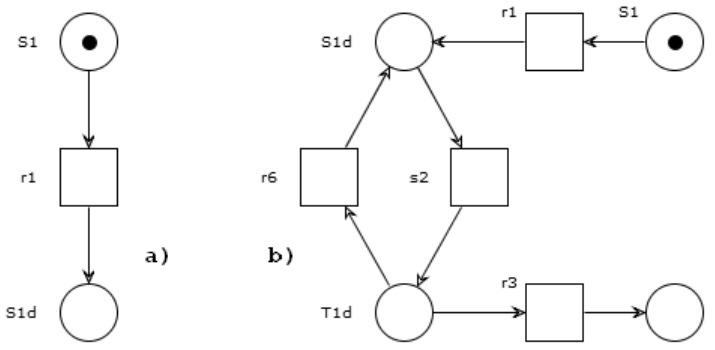

Figure 6. Petri net representations of the process S1.

Similarly, in time of creating a net representation of the process S1, the process S1d is represented only as a labeled place. Situation might be seen in Figure 6, where in the case a) incorrect and in the case b) correct representation is depicted. The place is within the final optimization identified with the place of the same label, which is the initial place of the net representation of the process S1d.

Following the definition of the process S, after executing the process S1 it will start itself again. Therefore, the net of process S in the Figure 2, case a) contains Ɛ-transition connected back to the place labelled by name S. In such a way, the entire execution of the process S1d was omitted, since within the composition just a single place was used, instead of its correct Petri net representation. The correct Petri net representation of the process S is depicted in the Figure 2, case b).

3.5. PNML output file does not match the PNML standard

Generated output file has been tested using PNML validator on the basis of which it was found, that the <pnml> element is missing an attribute defining a namespace of the document.

```
<?xml version="1.0" encoding="UTF-8"?>
<pnml>
<net id="net1" type="PTNet">
...
</net>
</pnml>
```

Element <net> contains the attribute type, but the validator did not recognize the specified type of Petri net. Within the file an element <page> is also missing, representing the page of PNML file, which is an important part of the structure of PNML document.

3.6. Lack of initial marking

While forming a Petri net representation of ACP processes, every intermediate net has assigned an initial place. Initial places of nets are used at the process of their composition. That is why every such net comprises information, which of its places is the initial one. Place itself, however, does not contain the information, that it is the initial place of net. This leads to the situation, that after writing the resulting net into the PNML file, the file does not contain information about the initial place.

3.7. Processes, which definition contains parallel composition with itself

Incorrect resulting net is also generated in the case, where the process definition contains a parallel composition of any process with itself and the corresponding communication function is defined. An example of such situation is illustrated by

the following simple ACP specification:

```
gamma(a,b) = c
X = a.b.(X||X)
```

Specification contains definition of process X, which has parallel composition X||X in its definition, so the process itself is a part of its parallel composition. Because the process X in the moment of constructing the parallel composition X||X does not have its final Petri nets representation yet, it is represented as a labeled place only. So the communication is applied only to two places (and not the corresponding net representations of process X) and the processes (by means of actions a, b) will not communicate.

4. Solutions to shortcomings revealed within the analysis

Solutions to revealed deficiencies were proposed and implemented within the tool and their description is provided in the following part of the paper.

4.1. Solution to problems related to encapsulation, communication and order of equations

Solution is based on checking the order of equations defining a system and change the order (if needed) before the transformation itself is started.

Equations order checking starts at the time of parsing the equations from input PAML document. Nodes fetched from the PAML document, representing definitions of processes, are stored in a list. For every equation, a left-hand side of the equation (the name of the process), and a list of variables, which are part of the process definition are stored. Equations from the list are one by one reviewed and the main equation representing the system composition is determined. It is the equation, whose left-hand side does not appear in definition of any other process. The order of equations is checked starting from the main equation to the first equation of the specification. The equation is in the right position with respect to other equations, when all of its variables are representing names of processes (left-hand sides of equations), which are defined before this process, i.e. they are positioned within the specification before this equation. If the order of equations according to main equation is not correct, then new, corrected order is generated. Above mentioned steps are parts of newly proposed and implemented preprocessing algorithm. An option is given to a user, whether to continue with the original order of equations or to change the order to the proposed one (Figure 7).

Checking the order of equations provided by the algorithm is demonstrated using the Example 1, case a) mentioned earlier within the paper. At the beginning, equation defining the process ABP is chosen as the main equation. Afterwards it is checked, whether the processes represented by variables S and K in definition of process ABP are defined before the definition of the process ABP. The second of the

two processes (K) is defined by the first equation only, so it is considered correctly positioned. The definition of the first of the two processes (S), however, does not fulfill this condition, and so the order of equations is considered incorrect. A new order is generated, based on the same conditions, which determine the correctness of the order of equations.

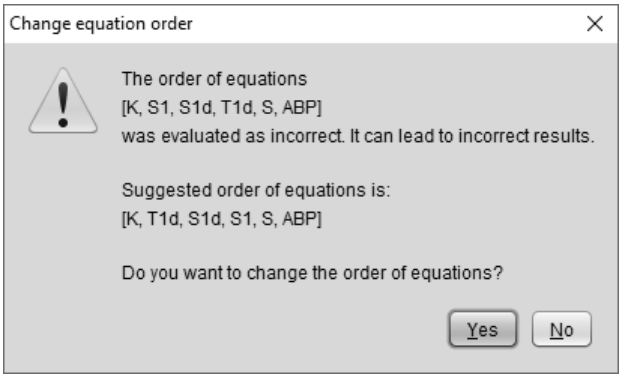

Figure 7. Proposing the order of equations.

Equation defining the process ABP is positioned as the last one. It contains the variables S and K in its definition and that is why the equation S is the equation preceding it within the order. Process S contains the variable S1, so the next equation is the equation of process S1. The same principle is applied to equations of processes S1d and T1d. Process T1d contains the variable S1d, but the position of S1d was determined yet. Process T1d does not have any other variables in its definition, so this determining branch is completed now. The last equation to process is the equation defining the process K. In such a way the correct order of equations was constructed, see Figure 7.

4.2. Adding net optimizations after the encapsulation

Problem of missing optimizations after applying the encapsulation operation in process of transformation is solved by calling the corresponding methods providing Petri net optimizations. Petri nets optimizations are defined and described in [5].

4.3. Corrections to the PNML output file

Generated PNML output file is modified in a way that the attribute determining the namespace of a document was added to the element `<pnml>`. Attribute type of the element `<net>` was modified too, and now contains the valid Petri net type. The element `<page>` was also added to the document structure. Modified header is a part of the following fragment of the output PNML file:

```
<?xml version="1.0" encoding="UTF-8"?>
<pnml xmlns="http://www.pnml.org/version-2009/grammar/pnml">
```

```
<net id="net1" type="http://www.pnml.org/version-2009/grammar/
ptnet">
<page id="top-level">
...
</page>
</net>
</pnml>
```

4.4. Initial marking

Information on initial marking was added to the initial place while exporting the generated Petri net. By this modification, the information is now present in the output PNML file, too. The information on initial places is also displayed during the system transformation.

4.5. Transformation of processes, which definition contains parallel composition with itself

A test, whether a process participating in a parallel composition does have its final net representation, was added to the transformation of parallel composition. In case of situation it does not, nets participating in parallel composition are stored in a list and the communication functions application is postponed. When the net representations of participating process are complete, communication functions are applied and transitions representing the result of communication are added to the net of the resulting process.

5. Extensions of the tool

Implemented extensions of the tool simplify its usage and provide new options and functionality for analysis of systems and their transformations.

5.1. Graphical user interface

The most perceptible extension of the tool is a graphical user interface, which allows for comfortable management and control of the transformation steps execution. Buttons for transformation progress control are available during the process of transformation and allow for executing and displaying the results of the next and previous steps, respectively. All displayed steps and generated Petri nets are stored within application's data model, from where they can be accessed and displayed as previous steps of the transformation.

Main window of the graphical user interface (Figure 8) displays an information related to the state and the progress of particular transformation. On the right hand side of the window a Petri net of actually processed part of ACP specification is displayed. Detailed information about elements of the Petri net are available, too. ACP equations being currently processed and performed action description are also displayed.

Main window content depends on a tab selected. The „Transformation“ tab provides detailed information on running transformation, the „PAML“ tab shows an input ACP specification, the „Output“ tab displays standard output. Other tabs are added throughout the transformation, having the names based on processed equations and containing the Petri nets representations of process. Additional functionality, like comparison of Petri nets, settings of export formats, and running a new transformation are available from the menu of the main window.

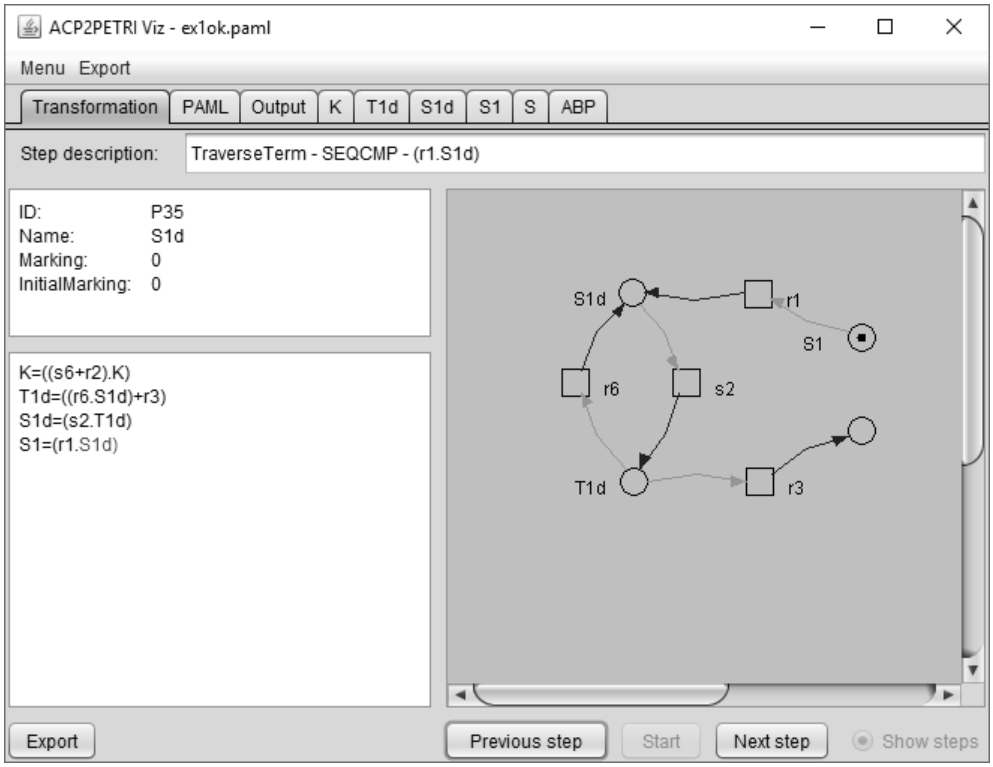

Figure 8. Main window of the graphical user interface.

5.2. Exporting Petri nets

Another extension allows a user to export each displayed Petri net as the PNML file or as an image in PNG format. Inclusion of this feature provides the possibility for further processing and analysis of generated nets, as the PNML is widely accepted interchange file format for Petri nets.

5.3. Petri nets comparison

When analyzing the transformation process by inspection of generated Petri nets, requirements for comparison of displayed Petri nets arise very often. For more

complex nets, however, a manual comparison is very difficult. For this reason an algorithm to determine, whether two Petri nets are identical, was implemented. Petri nets are compared with respect to places, transitions and also the connections between them.

Identical nets have the same number of places, transitions and arcs and the places and transitions of both nets must have the same names. To each element from the first net a set of elements from the second net is assigned, which fulfill the following conditions:

- the same number of incoming and outgoing arcs,
- the same name of pre- and post- transitions and places (for Ɛ-transitions and places without label their count must match).

If empty set of corresponding elements is assigned to a particular element, compared nets are different. Elements from the second net, which are in assigned set, are further examined according to whether the pre- and post- places and transitions of elements are part of set assigned to corresponding element from the first net. If these conditions are satisfied, the nets are considered identical. There are three possibilities for comparing Petri nets available within the tool:

- comparison of displayed net and the net stored in a PNML file,
- comparison of two nets, where both of them are stored in PNML files,
- comparison of displayed net and the net from other running transformation instance of the tool.

5.4. Simple editor of Petri nets

During the process of transformation, complex Petri nets, with lots of elements, are generated very often. Default layout of such nets can be intricate and it can be fairly difficult to recognize their meaning. For that reason a simple editor of displayed nets was implemented, which allows a user to change the positions of places and transitions to create its own, more comprehensible net layout.

6. Conclusions

Within the paper we described the results of analysis performed, classified shortcomings and limitations we found, and proposed solutions to solve them. Limitations revealed by the analysis were subdivided into seven parts and illustrated using simple examples, where needed. Before proposing adequate solutions to some of inaccuracies observed within operation of the tool, a graphical user interface with transformation stepping functionality was implemented and it turned out to be very useful in this respect. By this way, the nature of some resistive shortcomings was disclosed and appropriate steps undertaken to solve them. Proposed and implemented solutions are clearly described within the paper. Having corrections to revealed problems implemented, we expect more confidence in generated specifications without additional verifications.

Increased attention is also devoted to description of extensions to the ACP2Petri, which make it much more convenient instrument to use. The importance of

graphical user interface with transformation stepping functionality was mentioned yet, but the possibility to export generated Petri net, in arbitrary moment of transformation, for further analysis or to compare two Petri nets quickly are valuable additions, too.

We believe that implemented solutions and extensions will contribute to even better and more convenient exploitation of the tool at the tasks connected with the design and analysis of systems using Petri nets and process algebra. In the future it would be interesting to consider an extension to the transformation algorithm in order to adding the support for some of higher-level formalisms, and broaden the scope of its deployment this way.

References

[1] S. Graf and H. Garavel, *Formal Methods for Safe and Secure Computer Systems*. Bonn, Germany: Federal Office for Information Security, 2013. Available: https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Publikationen/Studien/formal_methods_study_875/formal_methods_study_875.pdf

[2] E. A. Boiten, J. Derrick, G. Smith (Eds.): *Integrated Formal Methods*, 4th International Conference, IFM 2004, Lecture Notes in Computer Science 2999, Springer-Verlag, 2004.

[3] T. Melham: "Integrating Model Checking and Theorem Proving in a Reflective Functional Language", in E. Boiten, J. Derrick, G. Smith (Eds.): *IFM 2004*, LNCS 2999, pp. 36–39, Springer-Verlag Berlin Heidelberg 2004.

[4] Š. Korečko and B. Sobota, "Petri nets to B-language transformation in software development," in *Acta Polytechnica Hungarica*, vol. 11, no. 6, 2014. Available: http://www.uni-obuda.hu/journal/Korecko_Sobota_52.pdf

[5] J. Ch. Attiogbé, "Semantic Embedding of Petri Nets into Event B," in *Int. Workshop on Integration of Model-based Formal Methods and Tools*, Düsseldorf, Germany, 2009.

[6] S. Šimoňák, "Verification of Communication Protocols Based on Formal Methods Integration" in *Acta Polytechnica Hungarica*, vol. 9, no. 4, 2012. Available: http://www.uni-obuda.hu/journal/Simonak_36.pdf

[7] R. Gorrieri and C. Versari: "A Process Calculus for Expressing Finite Place/Transition Petri Nets", in *Proceedings of 17th International Workshop on Expressiveness in Concurrency*, EXPRESS'10, Paris, France, August 30th, 2010.

[8] R. Gorrieri: "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.

[9] P. Sobocinski: "Representations of Petri net interactions", *CONCUR 2010 - Concurrency Theory*, Volume 6269 of the series Lecture Notes in Computer Science, pp. 554-568, 2010.

[10] S. Šimoňák, Š. Hudák, Š. Korečko: "APC semantics for Petri nets", in *Informatica*, Vol. 32, no. 3, 2008, p. 253-260. Available: http://www.informatica.si/index.php/informatica/article/view/197/194

[11] J.C.M. Baeten and W.P. Weijland: *Process Algebra*, Cambridge University Press, 1990.

[12] J.C.M. Baeten and T. Basten: "Partial-Order Process Algebra", in *Handbook of Process Algebra*, Elsevier Science, Amsterdam, The Netherlands, 2000.

[13] T. Basten and M. Voorhoeve: "An Algebraic Semantics for Hierarchical P/T Nets", in Computing Science Report, Eindhoven University of Technology, 1995.

[14] E. Best, R. Devillers, M. Koutny: "Petri Nets, Process Algebras and Concurrent Programming Languages", in *Lectures on Petri Nets II: Applications*, Volume 1492 of the series Lecture Notes in Computer Science, Springer Berlin Heidelberg, 1998, pp. 1-84.

[15] S. Šimoňák, "Formal Methods Integration Using Transformations of Petri Nets and Process Algebra", Ph.D. dissertation, TU FEEI, DCI, Košice, Slovakia, 2003. (in Slovak)

[16] S. Šimoňák, Š. Hudák, and Š. Korečko, "ACP2PETRI: a tool for FDT integration support", in *Analele Universitatii din Oradea, Proc. 8th Int. Conf. Eng. of Modern Elect. Syst.*, Romania, 2005, pp. 122-127.

[17] M. Bačíková and J. Porubän: "Domain usability, user's perception", in Zdzislaw S. Hippe, Juliusz L. Kulikowski, Teresa Mroczek, and Jerzy Wtorek, editors, *Human-Computer Systems Interaction: Backgrounds and Applications 3*, volume 300 of Advances in Intelligent Systems and Computing, pp. 15-26. Springer International Publishing, 2014. Available: http://link.springer.com/chapter/10.1007%2F978-3-319-08491-6_2

[18] S.Šimoňák and I.Peťko: "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. Available: http://www.aei.tuke.sk/papers/2010/1/10_Simonak.pdf