FORMAL METHODS TRANSFORMATION OPTIMIZATIONS WITHIN THE ACP2PETRI TOOL

Slavomír ŠIMOŇÁK
Department of Computers and Informatics, Faculty of Electrical Engineering and Informatics,
Technical University of Košice, Letná 9, 042 00 Košice, Slovak Republic,
tel.: +421 55 602 3178, E-mail: slavomir.simonak@tuke.sk

SUMMARY

The complexity of discrete systems developments process shows, that it is not realistic to expect, that the only universal formal method will exist, which will cover well all the aspects of this process. We are working towards formal methods integration in order to obtain deeper understanding of system under development/analysis. We chose Petri nets and process algebra, notations with complementary properties for integration. The research activity is very high in this area and the work was inspired by [9,10,11] and other papers published in past years. ACP2PETRI is a software tool for formal method transformations based on results of research performed in past years [1].

Keywords: *Petri nets, process algebra ACP, Petri net semantics for process term, elementary (Petri) nets, transformation optimizations*

1. INTRODUCTION

The purpose of the tool is a transformation of source ACP [2,3] specification (written in XML-based language PAML [4]) into the Petri net representation (PNML language [6]). The transformation preserves the linguistic semantics (defined in [1,5] and in short described also in this paper) of source specification in destination one. The transformation mentioned is a part of design and analysis multi-FDT environment [8], based on three formal notations – process algebra, Petri nets and the B-AMN.

The paper provides an insight into some internal mechanisms of ACP2PETRI, which are used in order to obtain resulting specification (net) smaller and clearer, if it is possible.

After sketching few theoretical principles, useful for understanding the net composition mechanisms, optimizations used within the tool are described and, at the end of paper, an example is appended to illustrate the operations mentioned.

2. SHORT THEORETICAL OVERVIEW

As we stated at the beginning of this paper, the ACP2PETRI tool is mostly based on results of research performed by the author in the area of formal methods transformations. The theoretical foundations are published in [1,5]. At this paper only the brief introduction is presented. *Elementary nets* correspond to the atomic actions of the process algebra ACP extended by the empty process notion (ε) and the deadlock (δ) respectively. Let process Q is represented by ACP term a ($Q = a$). Corresponding Petri net ($N(Q) = N_a$) is given by: $N_a = (P, T, pre, post)$, $I(P) = \{Q\}$, $F(P) = \{Q'\}$, $P = \{Q, Q'\}$, $T = \{a\}$, $pre(Q, a) = 1$ and $post(Q', a) = 1$. Shape of the corresponding elementary net is depicted at Fig.1 case a). Similarly, cases b) and c) hold elementary nets for empty process (ε) and the deadlock (δ) respectively. The meaning of notation used: P – stands for set of places, I(P) – set of *initial* places, F(P) – set of *final* places. For example we can observe, the set of final places for the deadlock action (δ) is empty. This means, there is no possibility to append (sequentially) another action (or process) to such an action, which corresponds with behavior desired.

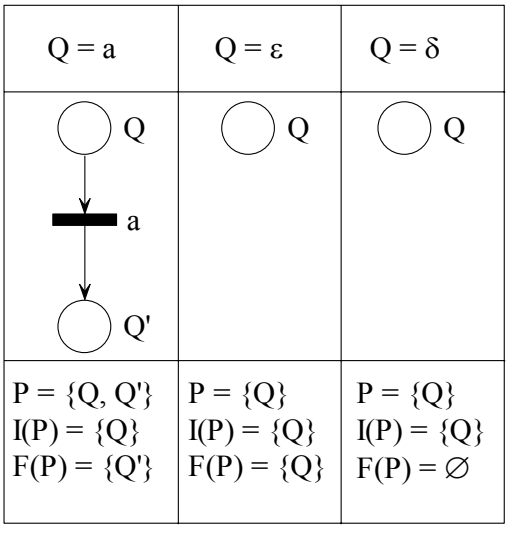

Fig. 1 Elementary nets

Net operations correspond to the operators of the process algebra ACP and are necessary for obtaining a net semantics for more complex terms. The operations discussed are: alternative composition (+), sequential composition (·), parallel composition (‖) with communication (in ACP exactly two actions can communicate each other) and encapsulation operation (∂_H), where set H is the set of actions to encapsulate.

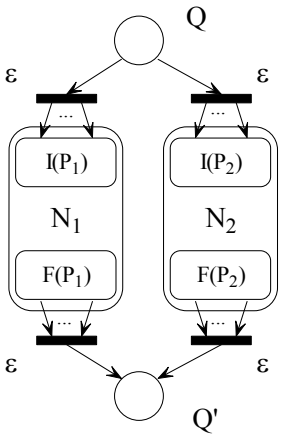

Fig. 2 Alternative composition

Net operations corresponding to the ACP operations mentioned above are explained by the figures Fig. 2, Fig. 3 and the Fig. 4 in brief. The full definitions of these operations are published elsewhere [1, 5]. Alternative composition of two nets is obtained by extending the set of places (union of sets of places of composed nets) by two additional places (Q,Q'), where Q will act as the initial place of composition and Q' as the final one.

When sequential composition is considered (Fig. 3), the final place of first composed net (N_1) is connected via the transition (labeled with an empty action (ε)) to the initial place(s) of the second one (N_2). As we mentioned above, if there is no final place within the first net (N_1), no sequential composition is possible.

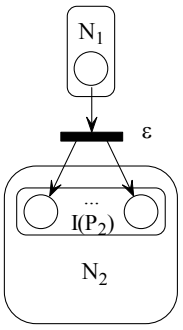

Fig. 3 Sequential composition

The parallel composition operation is slightly more complicated, when compared to operations just described. This is due to the fact, the communication possibility between two actions must also be taken into consideration, when the behavior of ACP processes should be modeled exactly. Fig. 4 depicts the situation, where the actions *a* and *b* are able to communicate, and the result of such a communication is the action *c*. Communication possibility is expressed by means of communication function *γ(a,b)=c* in ACP (for situation on Fig. 4). In situation depicted, Petri net marked *x*, stands for net obtained from the N_1 by removing it's initial place, transition named *a* and the arcs incidental. Similarly for net marked *y*.

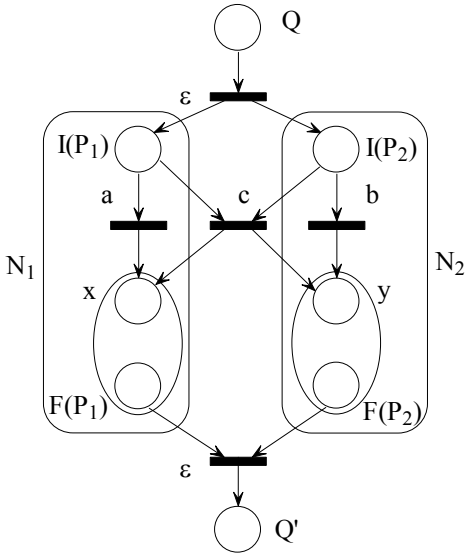

Fig. 4 Parallel composition

The encapsulation operation is an unary one, and the resulting Petri net is constructed in such a way, that transitions with labels from the encapsulation set (H) are removed from the net and incidental arcs are also removed. For obtaining the net-semantics of ACP terms we define the following rules:

- $N(Q+R) = N(Q) + N(R)$
- $N(Q \cdot R) = N(Q) \cdot N(R)$
- $N(Q \,\|\, R) = N(Q) \,\|\, N(R)$
- $N(\partial_H(Q)) = \partial_H(N(Q))$

While on the LHS of these equations symbols $(+,\cdot,\|,\partial_H)$ represents the operators of process algebra ACP, the same symbols at the RHS are the net operations introduced above. The resulting Petri net, corresponding to the transformed ACP term is equipped with the initial marking consisting of one token placed in the initial place of the composition.

2.1. Linguistic semantics

Linguistic semantics for ACP terms is defined as follows:

- $[\![a]\!] = \{a\}$
- $[\![\varepsilon]\!] = \{\varepsilon\}$
- $[\![\delta]\!] = \{\varepsilon\}$
- $[\![u+v]\!] = \{u\} \cup \{v\}$
- $[\![u \cdot v]\!] = [\![u]\!] \cdot [\![v]\!] = \{xy \mid x \in [\![u]\!], y \in [\![v]\!]\}$
- $[\![u \,\|\, v]\!] = [\![u \between v]\!] \cup [\![v \between u]\!] \cup [\![u \mid v]\!] =$ $[\![a(u' \,\|\, v)]\!] \cup [\![b(u \,\|\, v')]\!] \cup [\![c(u' \,\|\, v')]\!]$, kde $u = au', v = bv', \gamma(a,b) = c$
- $[\![\partial_H(a)]\!] = \{a\}$ ak $a \notin H$
- $[\![\partial_H(a)]\!] = \{\varepsilon\}$ ak $a \in H$
- $[\![\partial_H(u+v)]\!] = [\![\partial_H(u)]\!] \cup [\![\partial_H(v)]\!]$
- $[\![\partial_H(u \cdot v)]\!] = [\![\partial_H(u) \cdot \partial_H(v)]\!]$

Where u, v stands for ACP terms, $a \in A$, $\gamma()$ is communication function and $[\![]\!] : ACP \rightarrow A^*$ semantic function.

Linguistic semantics for Petri net is given by its language, i.e. the set of acceptable firing sequences. So if Petri net language is defined by:

$$L(N) = \{\sigma \in T^* \mid m_0 \overset{\sigma}{\vdash} m\}$$

then linguistic sémantics $[\![N]\!]$ is given by names of transitions in the sequence σ :

$$[\![N]\!] = \{\alpha \mid \alpha = l_T(t_1),...,l(t_r); \sigma = t_1,...,t_r; \sigma \in L(N)\}$$

3. OPTIMIZATIONS USED

Composition operations on nets are defined in such a way, that they are robust enough to cover all the needs, which can arise in process of ACP specification transformation. In some cases, some elements can be omitted, or optimized. This could be done in order to obtain the resulting net, which is smaller (the number of places, transitions or arcs is lower), more clear in expressing the system which it represent, but still preserving all the properties (linguistic semantics) of original specification.

Within the ACP2PETRI tool, currently three types of optimizations are used:

1. Removing disconnected places
2. Joining places with the same variable associated
3. Removing transitions
   a) *Simple-loop*
   b) *Without pre-places*
   c) *Simple-ε*

1. Removing disconnected places

When an isolated place (i.e. place with no arcs connected to it) is found, it is removed, because such a place cannot change it's marking. Also it is not possible such a place cause firing of any transition, and thus is not important with respect to linguistic semantics used within this transformation.

2. Joining places with the same variable associated

Within the ACP specification, which represents the source of transformation, one process term is *bound* to the one variable. When this is not true, specification is not deterministic, and transformation is cancelled in that case. Also, when the variable is not bound (associated) to a term, we say, the variable is *free* within this specification, which is parameterized by such a variable. In this case, transformation is cancelled too. When the variable occurs more than once in specification, it occurs also in the net, which is being constructed upon this specification. But in this case it is not necessary to paste the net corresponding to the term, to which the variable is bound, at every occurrence of the variable. More elegant solution seems to be the joining such a places, with preserving all the connections of places involved.

3. Removing transitions

In some cases the structure of the resulting net can be simplified by removing some of transitions, which are labeled with empty action (ε). Such possibility can appear in one of three situations:

a) Transition within 'simple-loop'

Conditions to hold: transition is labeled by an empty action (ε), the cardinality of the set of pre-places (and post-places) is equal to 1, destination place of the outgoing arc is the same, as the source one of the incoming arc. Fig. 11 contains such a transition, which is connected to the initial place A. In this case transition is removed, together with arcs connecting it to the place. Formally this could be expressed as follows:

$$A = \varepsilon A \tag{1}$$

According to the axioms valid for algebra ACP [1,2], the empty action (ε) in equation (1) can be simply omitted in sequential composition operation. Two of axioms mentioned, which are essential for our purposes are given here:

εx=x
xε=x

Symbol *x* represents the ACP process here and ε stands for empty action (as mentioned above).

b) Transition without pre-places

When no arc ends in the transition, it is candidate for this type of optimization. This condition is evaluated by checking the size of the set of pre-places for given transition. Fig. 12 and Fig. 13 can serve as an example of operation mentioned.

c) Simple-ε transition

Some conditions are required to be satisfied, to classify transition as simple-ε type. Firstly, the transition label is empty action (ε). The number of pre-places and post-places is equal to 1 and finally to pre-place or post-place no variable is assigned (i.e. it is possible to identify these places). Such a configuration corresponds to sequential composition with empty action (ε) and same axioms can be used to show the correctness of the operation as in the simple-loop case. The net depicted at Fig. 7 contains simple-ε transition.

4. AN EXAMPLE

As an example, small ACP specification (2) is used, which is interesting in a way, that all of the net composing operations optimizations mentioned above are performed, while the transformation by the ACP2PETRI tool is finished.

$$A = (b.A) + A \tag{2}$$

The DOM (Document Object Model), showing the hierarchical representation of specification (2) is depicted in Fig. 5.

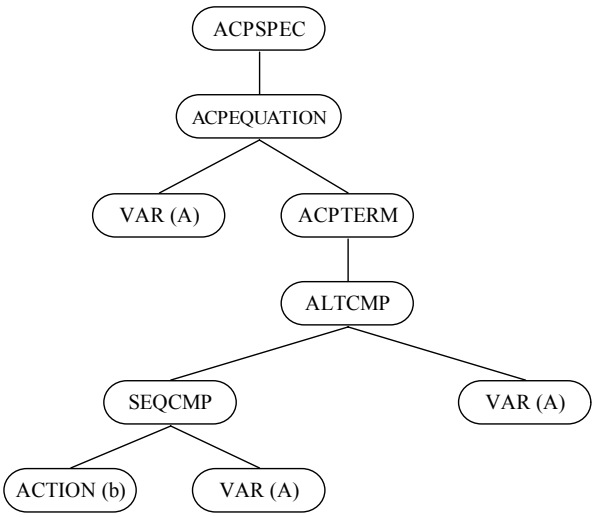

Fig. 5 Hierarchical representation of specification chosen (DOM)

Upon transformation starts, elementary net, corresponding to action b, as well as single place for variable A is created (Fig. 6).

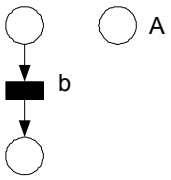

Fig. 6 Elements created at transformation start

Sequential composition of elements created follows, resulting net is depicted at Fig. 7. After this composition operation, net will be optimized for first time.

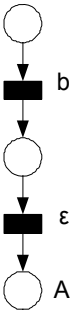

Fig. 7 After sequential composition

First of optimizations (Simple-ε transition) removes transition of empty action (ε) and arcs incidental to this transition. Result is depicted in Fig. 8.

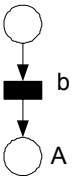

Fig. 8 Optimized net (simple-ε type)

After this optimization, next place corresponding to next occurrence of variable A in term is created. Alternative composition takes place, in order to obtain a net shape as in Fig. 9. At this moment, all the source specification is processed, so the net contains all elements, required, and only optimizing tasks are to be done.

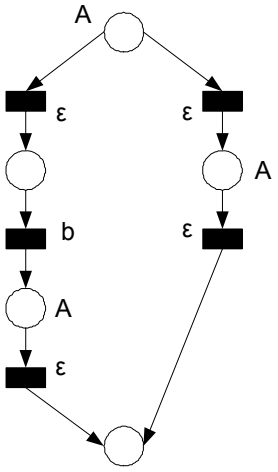

Fig. 9 Alternative composition performed

Two optimizations follow of same type (Joining places with the same variable associated). First of them merges two places corresponding to variable A, which are different from the initial place of whole composition (it also has bound to the variable of this name) (Fig. 10).

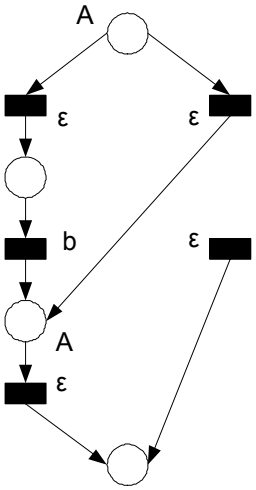

Fig. 10 First of two joining operations

Second joining operation merges two resting places with name A associated (Fig. 11), so only one place with this name is the initial place of the net now.

Next optimization is of type simple-loop, and removes transition labelled with empty action name (ε) and corresponding arcs (connected to initial place, labelled A). The net after this optimization performed is depicted at Fig. 12.

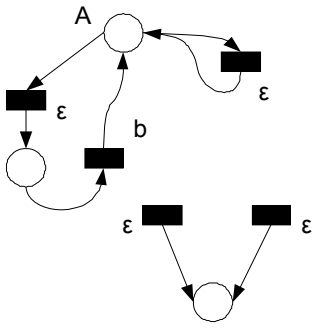

Fig. 11 Second joining operation performed

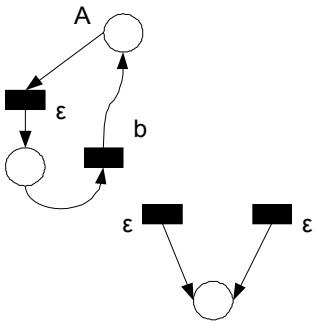

Fig. 12 Simple-loop optimization done

Two following optimizing operations are of type transition without pre-places, and remove transitions and arcs connected to final place of whole net (Fig. 13).

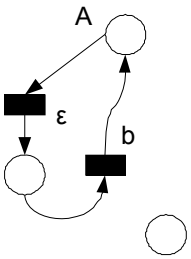

Fig. 13 Two transitions (and arcs) removed

Simple-ε transition optimization is used again, to reduce destination net by one place, one transition and two arcs interconnecting them together. Fig. 14 contains the net after operation is done.

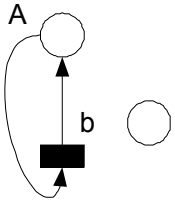

Fig. 14 Simple-ε type optimization

At last, disconnected (final) place is removed (e.g. a process described, still has possibility to continue operation, without reaching the final state, and stop its execution). So Fig. 15 holds the resulting net, after all the optimizations taken place.

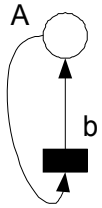

Fig. 15 Resulting net

Resulting Petri net, as obtained by converting the specification (1) by ACP2PETRI tool, opened in Editor of PNK (Petri Net Kernel) [7] environment (Fig. 16). Selfacting simulation functionality and PNML specification import is also available within this environment.

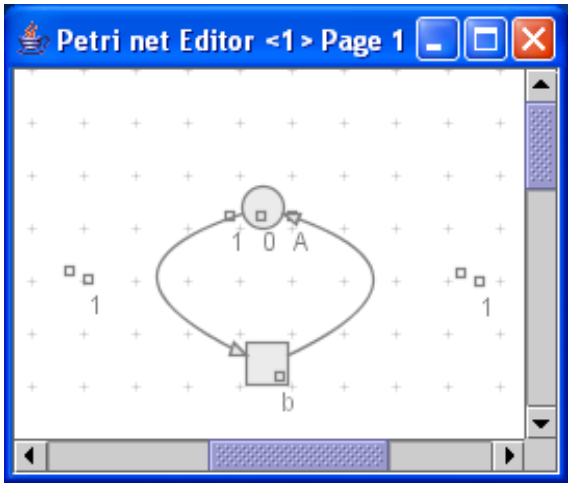

Fig. 16 Resulting net in PNK Editor

5. CONCLUSIONS

The optimization issues of ACP specification into the Petri net notation transformation (with respect to ACP2PETRI tool, which implements such a transformation) are treated in paper. Three types of optimizations are described - removing disconnected places, joining places with the same variable associated and removing transitions (labelled by empty action – ε, in special net configurations). Approaches presented here reflect the current state of solving the problem, as it is implemented in version v1.01 of ACP2PETRI tool. Next optimizations and overall improvements to the transformation are subject of research.

ACKNOWLEDGEMENT

This paper has been prepared under support of grant project VEGA 1/3140/06 of Slovak Grant Agency.

REFERENCES

[1] Šimoňák, S.: *Integrácia formálnych metód s využitím transformácií Petriho sietí a procesných algebier*, Dizertačná práca, KPI FEI TU Košice, pp.112, 2003. (in Slovak)

[2] Baeten, J.C.M., Weijland, W.P.: *Process Algebra*, Cambridge University Press, ISBN 0 521 40043 0, pp.248, 1990.

[3] Baeten, J.C.M.: *Applications of process algebra*, Cambridge University Press, United Kingdom, 1990.

[4] Šimoňák, S., Hudák, Š., Korečko, Š.: The tool for FDT integration support, Zborník prednášok z 8. konferencie so zahraničnou účasťou Informatika 2005, Bratislava, Slovakia, 20.-21. 6. 2005, SSAKI, 2005, pp. 103-108, ISBN 80-969243-3-8

[5] Šimoňák, S., Hudák, Š.: Petri Net Semantics for ACP Terms. In: Acta Electrotechnica et Informatica Vol. 4, No. 1, Košice, 2004, pp. 55 - 59.

[6] PNML - Petri Net Markup Language. URL http://www.informatik.hu-berlin.de/top/pnml/

[7] PNK – The Petri Net Kernel. URL http://www.informatik.hu-berlin.de/top/pnk/

[8] Korečko, Š., Hudák, Š., Šimoňák,S.: Progress in the mFDT Environment Development, Zborník prednášok z 8. konferencie so zahraničnou účasťou Informatika 2005, Bratislava, Slovakia, 20.-21. 6. 2005, SSAKI, 2005, pp. 183-188, ISBN 80-969243-3-8

[9] Olszewski, J.: *Representing Closed CCS Systems by Petri Nets*, UNSW-CSE-TR-9412, University of New South Wales, Australia, pp.19, 1994.

[10] Goltz, U.: *On Representing CCS Programs by Finite Petri Nets*, LNCS 324, pp. 339-350, ISBN 3-540-50110-X, Springer-Verlag, 1988.

[11] Paige, R.F.: *Formal Method Integration via Heterogeneous Notations*, PhD Dissertation, November 1997. http://citeseer.nj.nec.com/paige97formal.html

[12] Vokorokos, L.: Digital Computer Principles. Typotex Ltd. Retek 33-35, Budapest, 2004, p. 230. ISBN 96-39548-09-X.

BIOGRAPHY

Slavomír Šimoňák was born on 23.9.1974. In 1998 he graduated (MSc.) from the Department of Computers and Informatics of the Faculty of Electrical Engineering and Informatics at Technical University in Košice. PhD degree obtained from the same university in 2004. His scientific research is oriented towards formal methods for design and analysis of discrete systems, primarily on Petri nets, process algebra and formal methods transformations. In addition, he also investigates problems related to time-critical systems, programming techniques and machine-oriented languages.

.