New Tool for Model-Based Testing using Petri nets and Constraint Programming

Tomáš POSPÍŠIL[1]

[1] Dept. of Measurement, Czech Technical University, Technická 2, 166 27 Praha, Czech Republic
pospito7@fel.cvut.cz

Abstract. *Model-Based Testing is an automated test approach used for automated test generation, execution, and evaluation. For this purposes, it uses a formal model of system behavior. In this paper, an advanced variant of Petri net is presented as a tool for description of the systems under test. Petri nets have been selected for their graphical and executable nature and for the ability to describe parallel systems. For tests generation, Constraint programming is used, which is a software technology used for description and solving of combinatorial problems.*

This paper proposes a non-autonomous timed Petri net with discrete time, where inputs and outputs are modeled as global variables, as behavioral system model. Places in net represent a performance of the task, and a token movement represents a transition between the tasks. Test input generation is based on the Active token approach and the input values are retrospectively deducted by using Constraint programming.

Keywords

Model-Based Testing, Petri nets, Test generation, Constraint programming.

1. Introduction

At present the Model Based Testing (MBT) and test generation are still active research topics ([1]). In the past, a number of articles was written on the subject, and there is currently a wide range of MBT tools. However, only a small fraction of them uses Petri nets (PNs) as the model ([2]). The MBT is not tightly defined concept and it has different meanings in various areas. Our group is mainly focused on the automotive industry, and there MBT is frequently considered a generation of test cases with oracles from a behavior model. In this approach, the MBT is a kind of black box testing, where an information about the tested system (SUT - system under test) come from an external behavior model. This means that there is no access to internals of SUT such as a source code.

Petri nets were originally introduced by C. A. Petri [3], they are graphical modeling formalism for conceptual modeling of the flow of information in systems. Currently, there is a wide range of PNs variants, which extend their descriptive power. From our perspective, are intriguing Time-Arc Petri nets (TAPNs) [4] that are suitable for the specification of real-time systems. A limited set of basic constructs in PNs often leads to high model complexity in case of large systems. Therefore the Petri nets with discrete variables (PNDVs) [5] were chosen; they primarily add modeling convenience and compactness to PNs. PNDVs extend PNs with a set of finite global integer variables, used in pre-conditions and post-assignments on transitions.

Constraint programming (CP) is a software technology for description and solving combinatorial problems (constraint satisfaction problems (CSPs)) using general search algorithms with heuristics. It is especially used in planning and scheduling areas. The basic idea behind CP is that users specify their constraints, and solver finds solutions, which should satisfy all of them.

This paper describes the combination of PNs with the discrete variables along with CP approach for automatic generation of the input and output variables of the modeled system. They may serve as test inputs and the oracle for MBT.

1.1 Recent work

The connection between PNs and constraint-based approaches is not deeply investigated. The research is mainly focused on the Colored PNs (CPNs) and multi-agent technology. In the paper [6], authors present the method for unfolding colored PNs using the constraint satisfaction approach to improve the unfolding efficiency. Paper [7] describes a research on multi-robot system planning based on a combination of multi-agent system and constraint satisfaction problem approach. Authors use specifically developed CPN models for a distributed constraint satisfaction algorithm. Paper [8] investigated High-Level PN formalisms that use the Constraint Logic Programming framework. Authors outline applications for hybrid and real-time systems, or flexible constraint satisfaction problems. Paper [9] introduces a Colored Petri net (CPN) model for a distributed constraint satisfaction algorithm to be applied in holonic multi-agent systems. In [10], a translation between timed Petri nets and CSP is demonstrated and the paper [11] presents an object-oriented Petri net model with aspects of a CP language.

In the MBT, there exists a wide range of approaches for tests generation; many of them use timed automata, UML diagrams, etc. but only a small set of them uses PNs. The PNs are used in a simulation, verification, and validation of the designed system due to their graphical and executable nature. Often, they are also used for the modeling of parallel systems. Also, for the PN there are several basic approaches for the test generation; moreover, there are a plethora of different PNs variants. In this section, some examples are presented. Article [12] presents test sequence optimization method based on CPN and improved ant colony algorithm, which is used to generate the optimal sequences for China train operation and control system. Paper [13] shows usage of an advanced kind of Petri nets as a test model for Model-Based Statistical Testing. Paper [14] shows High level PN with constraints for Scenario-based Testing application.

The combination of CP and PN with global variables in the MBT application has not been found, hence the proposed approach could be innovative.

2. Control-Flow Petri nets

In this section, Control-Flow Petri nets (CFPN) and its properties are described. In PNs, variables are typically modeled locally (CPNs). In the CPNs, tokens carry data structures (colors) and an assigned program code uses them to control when to fire transitions. Unlike CPNs, in the CFPNs tokens do not carry data, the data structures are global and finite, and only Integer and Boolean (modeled as Integer) variable types are allowed. As a basis for presented variant of the PNs, P-Free Petri Nets with Discrete Variables (P-Free PNDV from [5]) are used. In addition, the discrete time semantic is added to the CFPN in the form of Discrete Timed-Arc PN (DTAPN). This step is basically redundant because the authors of the article [5] show that DTAPN can be reduced to PNDV. The cause of this choice is to increase the clarity of PN for system modelers.

CFPN is a PN extended with a finite set of integer variables, $\boldsymbol{X} = \{x_1, \ldots, x_m\}$, and pre- and post-conditions on transitions. Meta-variables and syntactic categories are defined for expressions (**Expr**), conditions (**Cond**) and assignments (**Assign**).

Definition 1: The language describing expressions and conditions over $\boldsymbol{X}$ is

Expression $e \in \boldsymbol{Expr}$

$e ::= x \mid z \mid e \oplus e$, where $x \in \boldsymbol{X}$, $z \in \mathbb{Z}$, $\oplus \in \{+, -, *\}$

Conditions $c \in \boldsymbol{Cond}$

$c ::= e \bowtie e \mid c \vee c \mid c \wedge c \mid \neg c$, where $\bowtie \in \{=, <, \leq, >, \geq, \neq\}$

Assignments $a \in \boldsymbol{Assign}$

$a ::= (x_1 := e_1, x_2 := e_2, \ldots, x_m := e_m)$, where $e_1, \ldots, e_m \in \boldsymbol{Expr}$

Control-Flow Petri nets are defined as follows.

Definition 2: Control-Flow Petri net

An CFPN is a 9-tuple ($\boldsymbol{P}$, $\boldsymbol{T}$, $\boldsymbol{F}$, ***times***, $\boldsymbol{X}$, ***Range***, ***Pre, Post, V***) such that:

$\boldsymbol{P}$	*is a finite set of places,*
$\boldsymbol{T}$	*is a finite set of transitions, where* $\boldsymbol{P} \cap \boldsymbol{T} = \emptyset$,
$\boldsymbol{F}$	is flow function $\boldsymbol{F} \subseteq (\boldsymbol{P} \times \boldsymbol{T}) \cup (\boldsymbol{T} \times \boldsymbol{P})$,
times	maps intervals to input arcs $\boldsymbol{F}\vert_{\boldsymbol{P}\times\boldsymbol{T}} \rightarrow \{[a, b] \mid a \in \mathbb{N}, b \in \mathbb{N} \cup \{\infty\}\}$,
$\boldsymbol{X}$	$\boldsymbol{X} = \{x_1, \ldots, x_m\}$ is finite set of integer variables,
Range	is a function from the $\boldsymbol{X}$ to $\mathbb{N} \setminus \{0\}$; it assigns the maximum values for variables in $\boldsymbol{X}$,
Pre	is a function from the set $\boldsymbol{T}$ to ***Cond***; it maps transitions to pre-conditions,
Post	is a function from the set $\boldsymbol{T}$ to ***Assign***; it maps transitions to post-assignments,
$\boldsymbol{V}$	is a valuation; function from the $\boldsymbol{X}$ to $\mathbb{N}$, where $\boldsymbol{V}(x) \leq \boldsymbol{Range}(x)$ for all $x \in \boldsymbol{X}$.

Let N be a CFPN. A state on $\boldsymbol{N}$ is a 3-tuple $\boldsymbol{S} = (\boldsymbol{M}, \boldsymbol{V}, \boldsymbol{B})$, where $\boldsymbol{M}$ is a marking, $\boldsymbol{V}$ is a valuation and $\boldsymbol{B}$ is a finite multiset of natural numbers. That numbers correspond to the age of tokens at all places. The initial state of marked CFPN is a pair (N, S_0), where $S_0 = (M_0, V_0, B_0)$; M_0 is initial marking and V_0 corresponds to initial values of variables, in initial marking all tokens have age 0.

Pre-condition $c \in \boldsymbol{Cond}$ is satisfied by a state (denoted $S \models c$), if by replacing the variables in c with the corresponding values from $\boldsymbol{V}$ and the formula evaluates to true.

An assignment $a = (x_1 := e_1, \ldots, x_m := e_m) \in \boldsymbol{Assign}$ evaluates to a new valuation V' for given state S. It means that each expression e from a evaluates to a new value $\boldsymbol{V} \in \mathbb{Z}$ (denoted $\boldsymbol{V} = eval(e,S)$) for appropriate variables. More over each variable x_i is limited by $\boldsymbol{Range}(x_i)$ function, which specifies the largest value x_i can reach. Subsequently, it can be written $V' = eval(a, S)$.

Let $S = (M, V, B)$ be a state on N and $t \in \boldsymbol{T}$ a transition, transition t is enabled if there exists a token $x \in M(p)$ where $x \in \boldsymbol{times}(p, t)$, for all $p \in {\bullet}t$ and $S \models \boldsymbol{Pre}(t)$. If t is enabled in S, it can fire, which leads to a new state S'. From each input place ($p \in {\bullet}t$), a single token place satisfying the age constraint is removed and new token of age 0 is added to every output place ($p \in t{\bullet}$). The new state $S' = (M', eval(\boldsymbol{Post}(t), S), B')$, where M' new marking and B' is a new multiset of tokens ages.

3. System modeling

Our approach to system modeling is inspired by Workflow - Petri nets (WFPN) [15]. The system is modeled using CFPNs as a set of tasks (activities) and

transitions between them. Token's position represents a performance of the task and its movement represents a transition between the tasks. It was decided to model the activity as a place ([16]). This may at first seem counterintuitive because places often represent a static state. The execution of an activity is represented by a place and it is surrounded by two transitions. These transitions represent beginning and end of an activity. Similarly to the WFPN, the net contains only one token, but there is no need for the network to contain the start and end place. In the future, the concurrence of multiple parallel PNs is planned; they will be enhanced with synchronization mechanisms. The extending structures AND and XOR are also added in forms *split* and *join* (see [15]), for increased model transparency. These structures can be broken down into a form PNs and thus they bring no further issues. Next, more of supporting structures from WFPN as sub-processes or some routing patterns will be added.

3.1 Task duration

For modeling of the tasks duration, token locks are inserted into places that prevent tokens movement from the places for a specified period of time. This time is set fixed by the modeler, it represents the expected behavior of the user (inputs) or the duration of an internal task carried out by the system. In the future, the fixed values will be replaced with a stochastic approach. This solution was chosen for better transparency and it is based on the idea that the places correspond to tasks; additionally, this approach can be converted back to the timing arcs.

3.2 Variables

For the purpose of test inputs generation and oracle's outputs, variables of the CFPN are divided into two groups. In the first group, there are the inputs variables. They correspond to the input of the system and their actual values are generated by the CP (see section 4.2). In the second group, there are the state (output) variables, which can be used as a support for the oracle. Their actual values are computed from the initial values and the changes that are made using assignments (see section 4.3).

Definition 3: Variables

Let $\boldsymbol{X_{input}}$ and $\boldsymbol{X_{state}}$ are finite sets of integer variables, where $\boldsymbol{X} = \boldsymbol{X_{input}} \cup \boldsymbol{X_{state}}$, and $\boldsymbol{X_{input}} \cap \boldsymbol{X_{state}} = \emptyset$.

4. Implementation

The CFPNs are actually non-autonomous (reactive) PNs [17], they respond to inputs (events) from the external environment. Moreover, they are timed PNs with the clocks and a synchronous environment. That means CFPNs internal time is a sequence of discrete time steps (the basic clock strokes) and outside these time steps the system inputs are not valid. Model interpreter must guarantee a complete and coherent evolution of the model for every time step. In fact, for every time step, a cycle of transition firing is performed, bringing the CFPN from an initial stable state to a new stable one, according to the external inputs which are present at the same time step.

In the classic non-autonomous PNs, the firing of some transition is conditioned by a combination of external inputs. In the proposed method, the process works vice versa. From the firings made during a time step, the method retrospectively deducts which inputs lead to the chosen course. This procedure is based on the Active token approach and the generation of variables values by using the CP. The implementation of CFPNs is developed in JAVA and Java Constraint Programming solver - JaCoP ([18]) is used.

4.1 Active token

The proposed method is based on the principle of Active token; it is inspired by articles [19] and [20]. Common PNs are focused on transitions; when transition fires, it changes a token position and thus the marking of the net. From a token's perspective, it is a passive process. In the proposed method, the token "manages" its movement through the net itself. It means that it selects an appropriate output arc from the current place. Appropriate output arcs are selected such that the firing of conditional transitions maintains the consistency of the input variables (see. the following subsection). The method of selection may be based on search algorithms that meet certain cover criteria, on a heuristic method or a stochastic approach can be used. In future, the above-described approaches will be explored.

4.2 Input generation

Test inputs are generated based on the conditions that belong to transitions which are fired during a single time step. In the case of conditional transition firing, the appropriate condition is set as *active*. During the time step, only those transitions whose preconditions are not in contradiction with existing *active* conditions can be fired. This feature keeps the consistency of the input variables. At the end of the time step and after the stable state is reached, all *active* conditions are submitted to the constraint solver which calculates a solution, which contains a domain for each variable. The solver is able to select the values of variables from the resulting domains according to various criteria (the maximal, minimal, middle or random value of the domain). In this way, the values of the input variables are retrospectively determined for actual time step, and variables that are not included in the *active* conditions retain their values form the previous time step.

4.3 Output generation

In case of the CFPNs, **only** the values of the state (output) variables are changed by assignments. The values do not change along a simulation step ([21]), but they are changed at the end of time step after the input variables are

generated and all assignments are performed in the order they occurred during a time step.

5. Conclusion and future work

In this paper, the new tool that uses the Petri nets with discrete variables (CFPNs) in combination with a Constraint programming solver has been shown. The CFPNs are timed non-autonomous PNs with the clocks and a synchronous environment, which are used for system modeling. The paper outlines the possibility of using such tools for generating test in the MBT concept. Presented tool is based on the Active token approach and the generation of variables values by using the CP. In this approach, the system inputs are retrospectively deduced from the conditions belonging to the transitions that are fired during a time step, and the system outputs are calculated using assignments.

In the future, we want to focus on the parallel run of multiple PNs, with an emphasis on synchronization and system modularity. Furthermore, the problem of the token routing for test cases generation seems to be interesting. Particularly, we would like to focus on metaheuristic methods like Ant Colony Optimization. Last but not least, the JaCoP provides a broad array of useful constraints, which could be incorporated into the CFPNs.

Acknowledgements

The research described in the paper was supervised by Assoc. Prof. J. Novák, FEE CTU in Prague and supported by Grant Agency of the Czech Technical University in Prague [SGS16/171/OHK3/2T/13], by by Technologická Agentura ČR, grant no. TE01020020 - Centrum kompetence automobilového průmyslu Josefa Božka.

References

[1] RÖSCH, S., ULEWICZ, S., PROVOST, J. and VOGEL-HEUSER, B. Review of Model-Based Testing Approaches in Production Automation and Adjacent Domains—Current Challenges and Research Gaps in J. *Softw. Eng. Appl.*, 2015, vol. 08, no. 09, p. 499–519.

[2] MISTA: Model-based Integration and System Test Automation http://cs.boisestate.edu/~dxu/research/MBT.html

[3] PETRI, C., *Kommunikation mit automaten*, Tech. rep. Bonn: Rheinisch-Westfälisches Institut für Instrumentelle Mathematik an der Universität Bonn, 1966.

[4] HANISCH, H.M. *Analysis of place/transition nets with timed arcs and its application to batch process control*. in: Ajmone Marsan, M. (ed.) Application and Theory of Petri Nets 1993, LNCS,1993, vol. 691, p. 282–299.

[5] JENSEN, J. F., NIELSEN, T. and ØSTERGAARD, L. K. *Petri Nets with Discrete Variables*, Tech. rep. Aalborg Univ., 2012.

[6] LIU, F., HEINER, M., YANG, M. An efficient method for unfolding colored Petri nets. In *Proceedings of the Winter Simulation Conference*, 2012, p. 295.

[7] PANESCU, D., PASCAL, C., A constraint satisfaction approach for planning of multi-robot systems. In: *System Theory, Control and Computing (ICSTCC)*, 2014, p. 157-162.

[8] BOUTET, F., MOTET, G., KUBEK, J. M. Use of constraints in Petri nets and their novel applications, In *Proc. IEEE Int. Conf. Syst. Man, Cybern*, 1998, vol. 1, pp. 32–37.

[9] PASCAL, C., PANESCU, D. A Petri net model for constraint satisfaction application in holonic systems. In *2014 IEEE International Conference on Automation, Quality and Testing, Robotics (AQTR)*. IEEE, 2014. p. 1-6.

[10] MANCEL, C., et al. Relationships between Petri nets and constraint graphs: application to manufacturing. In *15th IFAC world congress*. 2002. p. 634.

[11] SANDERS, M. J. Constraint programming with object-oriented Petri nets. In *Systems, Man, and Cybernetics, 1998. 1998 IEEE International Conference on*. IEEE, 1998. p. 289-294.

[12] ZHENG, W., HU, N. W. Automated Test Sequence Optimization Based on the Maze Algorithm and Ant Colony Algorithm. In *International Journal of Computers, Communications & Control*, 2015, 10.4.

[13] BÖHR, F. Model based statistical testing of embedded systems. In: *Software Testing, Verification and Validation Workshops (ICSTW), 2011 IEEE Fourth International Conference on*. IEEE, 2011. p. 18-25.

[14] REZA, H., KERLIN, S. D. A model-based testing using scenarios and constraints-based modular petri nets. In *Information Technology: New Generations (ITNG), 2011 Eighth International Conference on*. IEEE, 2011. p. 568-573.

[15] VAN DER AALST, W., VAN HEE, K. M. *Workflow management: models, methods, and systems*. MIT press, 2004.

[16] ESHUIS, R., WIERINGA, R. Comparing Petri net and activity diagram variants for workflow modelling–a quest for reactive Petri nets. In *Petri Net Technology for Communication-Based Systems*. Springer Berlin Heidelberg, 2003. p. 321-351.

[17] ALLA, H., DAVID, R. Continuous and hybrid Petri nets. *Journal of Circuits, Systems, and Computers*, 1998, vol. 8, no. 1, p. 159-188.

[18] KUCHCINSKI, K., SZYMANEK, R. Jacop-java constraint programming solver. In *CP Solvers: Modeling, Applications, Integration, and Standardization, co-located with the 19th International Conference on Principles and Practice of Constraint Programming*. 2013.

[19] DAMMASCH, K., HORTON, G., Active Tokens for Modelling Mental Health Care with Coloured Stochastic Petri Nets. In *Innovations in Information Technology, 2007. IIT'07. 4th International Conference on*. IEEE, 2007. p. 541-545.

[20] ATANASSOVA, V., ATANASSOV, K. Ant colony optimization approach to tokens' movement within generalized nets. In: *Numerical Methods and Applications*. Springer Berlin Heidelberg, 2010. p. 240-247.

[21] BARTKEVIČIUS, S., KRAGNYS, R., ŠARKAUSKAS, K. Global Variables in Colored Petri Nets. *Elektronika ir Elektrotechnika*, 2015, vol 69, no. 5, p. 45-48.

About Authors

Tomáš POSPÍŠIL was born in 1986 in Prague, Czech Republic. He received an Ing. degree (M.Sc. equivalent) with a specialization in Cybernetics and Robotics from the Department of Measurement, Faculty of Electrical Engineering (FEE) Czech Technical University (CTU), Prague, in 2012. His research interests include Petri nets and Model-Based Testing.