

MODELING PETRI NETS SEMANTICS IN CSP

assoc. prof., dr. Vladimir Dimitrov
University of Sofia

Abstract

Standard notations for specification of business processes, communication protocols and distributed systems are Petri nets based. Such notations are BPMN, UML activity diagrams, WS-BPEL etc. Petri nets have no Turing machine power, but they are more expressive than finite state machines. They have well defined and investigated mathematical properties for analyses and verification. CSP is used as more powerful mathematical tool for modeling and analyses semantics of specifications given in above mentioned notations. In this approach, it is needed Petri nets semantics to be modeled in terms of chosen mathematical tool.

Keywords: Communicating Sequential Processes (CSP), Petri nets, business processes, semantics.

MOTIVATION

Most of the today broadly accepted standard notations for specification of business processes, communication protocols and distributed systems are Petri nets based. Such notations are BPMN [1], UML activity diagrams [2], WS-BPEL [3] etc. This means that tools supporting them for analyses and verification can use Petri nets. But, is it so? Occasionally, these notations are extensions of Petri nets that have expressive power of Turing machine [4]. Petri nets have no Turing machine power, but they are more expressive than finite state machines [5]. They have well defined and investigated mathematical properties for analyses and verification. That is why, Petri nets are preferred model for building notations for specification of distributed systems.

But in the real situation, specifications in notations with Turing machine power could not be directly analyzed and verified. There are two approaches to solve this problem. The first approach is to abstract the specification to pure Petri net and then to apply to result Petri net some Petri net tool for analyses. This approach is implemented in many researches like [6-8]. Good news for this approach is that distributed system backbone is specified in Petri net and the abstraction do not differs too much from detailed one, and that there are many tools for analyses of Petri nets like [9].

The other approach is to be used more powerful mathematical tool for modeling and analyses semantics of specifications given in

such notations. For such purpose can be used CSP [10], π -calculus [11] etc. In this approach, it is needed Petri nets semantics to be modeled in terms of chosen mathematical tool.

In this paper, CSP is used as mathematical modeling tool for analyses and verification. Subject of this paper is to investigate how Petri nets semantics could be modeled in CSP. This approach then can be applied on Petri nets based notations. The consequence is that distributed systems specified in Petri nets based notation can be more precisely analyzed and verified.

DEFINITIONS

Petri net graph is a triple (S, T, W) where:

- S is the *set of states*;
- T is the *set of transitions*;
- $S \cap T = \emptyset$
- $W: (S \times T) \cup (T \times S) \rightarrow \mathbb{N}$ is the *bag of arcs*.

Every arc connects place with transition or transition with place, but not place with place or transition with transition. Every arc has multiplicity that is a nonnegative number. Every place and every transition has at least one input or one output arc.

There is definition of Petri nets in which W is a set – this means that arcs have no multiplicity, or more precisely have multiplicity of 1. From the theory these two kinds of Petri nets have the same expressive power.

Petri net graph is a bipartite multidigraph in which nodes S and T are separated.

The *set of input places* for a given transition t is defined as:

$$\text{inputs}(t) = \{s \in S \mid W(s, t) > 0\}$$

The *set of output places* for a given transition t is defined as:

$$\text{outputs}(t) = \{s \in S \mid W(t, s) > 0\}$$

Petri net *marking* is a bag of its places:

$$M: S \rightarrow \mathbb{N}$$

This means that every place can contain zero or more markers.

Marked Petri net is the tuple (S, T, W, M_0) where M_0 is the initial marking.

Petri net behavior is defined as relation among markings.

Execution of Petri net is a relation $\rightarrow_G$ between two markings via a transition t where:

- $M \rightarrow_{G,t} M' \Leftrightarrow \exists M'': S \rightarrow \mathbb{N}: M = M'' + \sum_{s \in S} W(s, t) \wedge M' = M'' + \sum_{s \in S} W(t, s)$
- $M \rightarrow_G M' \Leftrightarrow \exists t \in T: M \rightarrow_{G,t} M'$

In words: firing of transition t for marking M consumes $W(s, t)$ markers from transition's input places s and adds $W(t, s)$ markers in transition's output places s .

A transition could be fired only if there are enough markers in all its input places:

$$\forall s \in \text{inputs}(t): M(s) \geq W(s, t)$$

The marking M' is reachable from a marking M in one step if $M \rightarrow_G M'$; more generally, M' is reachable from M if $M \rightarrow_G^* M'$, where $\rightarrow_G^*$ is the transitive closure of $\rightarrow_G$; that is, if it is reachable in 0 or more steps.

The *set of reachable markings* for the marked Petri net $N = (S, T, W, M_0)$ is:

$$R(N) = \{M' \mid M_0 \rightarrow_{(S, T, W)}^* M'\}$$

The *reachability graph* of N is the transition relation $\rightarrow_G$ restricted to its reachable markings $R(N)$. It is the state space of the net.

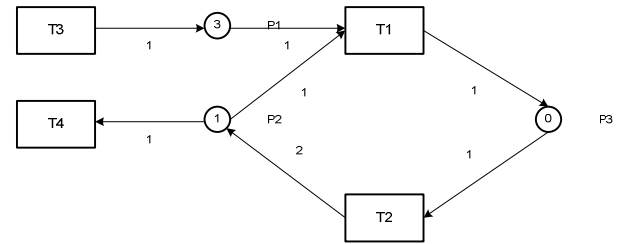
A *firing sequence* for a Petri net with graph G and initial marking M_0 is a sequence of transitions $\sigma = \langle t_1, t_2, \dots, t_n \rangle$ such that:

$$M_0 \rightarrow_{G,t_1} M_1 \wedge \dots \wedge M_{n-1} \rightarrow_{G,t_n} M_n$$

The *set of firing sequences* is denoted as $L(N)$. Example of Petri net graph is given in Fig 1.

In this example transition T_3 is only generating tokens, and T_4 is only consuming.

From T_2 to P_2 the arc has multiplicity 2 – all other arcs are with multiplicity 1.



$$\begin{aligned}
 S &= \{P1, P2, P3\}, T = \{T1, T2, T3\} \\
 W &= \{(T3, P1) \mapsto 1, (P1, T1) \mapsto 1, (T1, P3) \mapsto 1, \\
 &\quad (P3, T2) \mapsto 1, (T2, P2) \mapsto 2, (P2, T4) \mapsto 1\} \\
 \text{inputs}(T1) &= \{P1, P2\}, \text{outputs}(T1) = \{P3\} \\
 \text{inputs}(T2) &= \{P3\}, \text{outputs}(T2) = \{P2\} \\
 \text{inputs}(T3) &= \{\}, \text{outputs}(T3) = \{P1\} \\
 \text{inputs}(T4) &= \{P2\}, \text{outputs}(T4) = \{\} \\
 M_0 &= \{P1 \mapsto 3, P2 \mapsto 1, P3 \mapsto 0\}
 \end{aligned}$$

Fig. 1. Petri net graph.

Z-NOTATION SPECIFICATION

Here, we will be stricter in specifying Petri net in Z-notation [12].

Basic sets are:

$[STATES, TRANSITIONS]$

STATES and TRANSITIONS as different types have no common elements, i.e. implicitly $STATES \cap TRANSITIONS = \emptyset$.

M_0 : bag STATES

INPUT_ARCS: bag $(STATES \times TRANSITIONS)$

OUTPUT_ARCS: bag $(TRANSITIONS \times STATES)$

$$(\forall s: STATES \bullet s \in \text{dom}(\text{dom INPUT_ARCS}) \vee \\
 s \in \text{ran}(\text{dom OUTPUT_ARCS})) \wedge$$

$$(\forall t: TRANSITIONS \bullet t \in \text{ran}(\text{dom INPUT_ARCS}) \vee \\
 t \in \text{dom}(\text{dom OUTPUT_ARCS}))$$

With this axiom initial marking M_0 , Petri net arcs as input and output ones are defined, and the invariant say that every state and transition is connected in the graph.

PN

Marking: bag STATES

Petri net type is defined with the current marking.

PNInit

PN

Marking = M_0

Initially, Petri net marking is M_0 .

<u>Arcs</u>
$t?: TRANSITIONS$
$ins!: bag STATES$
$outs!: bag STATES$
$(\forall s: STATES \bullet$
$INPUT_ARCS \# (s, t?) = ins! \# s \wedge$
$OUTPUT_ARCS \# (t?, s) = outs! \# s)$

This is supporting schema – it, for a given transition $t?$, returns the bag of all input states and the bag of all output states. Every state participates in the input (output) states bag exactly what the multiplicity of the input (output) arc has to (from) the transition $t?$.

<u>Active</u>
<u>Arcs</u>
$Marking: bag STATES$
$(\forall s: STATES \bullet$
$(s, t?) \in INPUT_ARCS \Rightarrow$
$INPUT_ARCS \# (s, t?) \leq Marking \# s)$

This is another supporting schema which simply tests is active the transition $t?$ in the marking $Marking$.

<u>Execution0</u>
ΔPN
$ins?: bag STATES$
$outs?: bag STATES$
$Marking' = Marking \uplus ins? \uplus outs?$

This is supporting schema; it executes the transition that is represented by input and output states bags, but do not test for activity.

$$Execution \equiv Active \setminus (t?) \gg Execution0 \vee$$

$$\neg (Active \setminus (t?, ins!, outs!)) \wedge \exists PN$$

This the execution of Petri net; initially it tests for any active transition, if it find such one, then this transition is executed, but if there are no active transition Petri net marking remains the same.

$$Reachable == \{M: bag STATES \mid$$

$$\begin{aligned} & \exists st: seq TRANSITIONS \bullet \exists sm: seq (bag STATES) \bullet \\ & \#st + 1 = \#sm \wedge M_0 = sm(1) \wedge M = sm(\#sm) \wedge \\ & (\forall i: 1.. \#st \bullet \exists ins: bag STATES \bullet \exists outs: bag STATES \bullet \\ & (\forall s: STATES \bullet INPUT_ARCS \# (s, st(i)) = ins \# s \wedge \\ & OUTPUT_ARCS \# (st(i), s) = outs \# s \wedge \\ & (s, st(i)) \in INPUT_ARCS \Rightarrow \\ & INPUT_ARCS \# (s, st(i)) \leq sm(i) \# s) \wedge \\ & sm(i+1) = sm(i) \uplus ins \uplus outs) \} \end{aligned}$$

This is the definition of all reachable markings in the Petri net starting from the initial marking M_0 . It is defined through the existence of sequence of transitions and sequence of markings where the last one starts from M_0 .

$$\begin{aligned} PossibleTransitionSeqs == \{st: seq TRANSITIONS \mid \\ & \exists sm: seq (bag STATES) \bullet \\ & \#st + 1 = \#sm \wedge M_0 = sm(1) \wedge \\ & (\forall i: 1.. \#st \bullet \exists ins: bag STATES \bullet \exists outs: bag STATES \bullet \\ & (\forall s: STATES \bullet INPUT_ARCS \# (s, st(i)) = ins \# s \wedge \\ & OUTPUT_ARCS \# (st(i), s) = outs \# s \wedge \\ & (s, st(i)) \in INPUT_ARCS \Rightarrow \\ & INPUT_ARCS \# (s, st(i)) \leq sm(i) \# s) \wedge \\ & sm(i+1) = sm(i) \uplus ins \uplus outs) \} \end{aligned}$$

Finally, this is the set of all possible transition sequences for the Petri net. It is defined in the same way as the set of reachable markings.

The above example looks in Z-notation as:
 $STATES = P_1, P_2, P_3$,
 $TRANSITIONS = \{T_1, T_2, T_3\}$
 $INPUT_ARCS = \{(P_1, T_1) \mapsto 1, (P_3, T_2) \mapsto 1, (P_2, T_4) \mapsto 1\}$
 $OUTPUT_ARCS = \{(T_3, P_1) \mapsto 1, (T_1, P_3) \mapsto 1, (T_2, P_2) \mapsto 2\}$
 $M_0 = \{P_1 \mapsto 3, P_2 \mapsto 1, P_3 \mapsto 0\}$

INTO THE CSP

In our approach, Petri nets semantics is modeled with sequential processes; places are represented with variables – every place is represented from a variable with the same name. If $STATES = \{P_1, P_2, \dots, P_n\}$, then in the CSP model we will have the variables $P_1, P_2, \dots, P_n$. This set of variables reflects the Petri net marking. Every P_i for $i = 1, \dots, n$

contains nonnegative integer during the Petri net model execution. These variables are initialized with the M0 marking.

Every transition in the model is represented as a CSP process. If $TRANSITIONS = \{T1, T2, \dots, Tm\}$, then in the CSP model we will have the processes $T1, T2, \dots, Tm$. Every such a process is defined with a guard condition checking activity of the transition for firing. This condition checks the contents of the variables representing input states. It uses input arcs multiplicity.

Let's for Ti the bag of input states (with input arcs multiplicity) is $ins(Ti) = \{I1 \mapsto N1, I2 \mapsto N2, \dots, Ip \mapsto Np\}$, then the guard condition (ready for firing) is $I1 \geq N1 \wedge I2 \geq N2 \wedge \dots \wedge Ip \geq Np$. The process Ti is guarded in the following way:

$Ti <| 1 \geq N1 \wedge I2 \geq N2 \wedge \dots \wedge Ip \geq Np |> SKIP$

That means: Ti is firing if it is active, else it is skipped.

Ti when firing, it changes the marking (variables). Let's the bag of output places (with output arcs multiplicity) is $outs(Ti) = \{O1 \mapsto M1, O2 \mapsto M2, \dots, Oq \mapsto Mq\}$, then the definition of Ti is:

$Ti = (I1 := I1 - N1; I2 := I2 - N2; \dots,$

$Ip := Ip - Np; O1 := O1 + M1;$

$O2 := O2 + M2; \dots, Oq := Oq + Mq; Ti)$

$<| I1 \geq N1 \wedge I2 \geq N2 \wedge \dots \wedge Ip \geq Np |> SKIP$

Finally, the Petri net process is:

$PN = T1 ||| T2 ||| \dots ||| Tm$

When a process is executed, it changes the current marking of the Petri net in the variables.

For Petri net another important term is transition firing sequences. In this model the firing sequence can be registered with a channel: when a transition fires then an event can be generated in the output channel of Petri net. Let's such a channel is ch , and then instead of Ti in above definition we will have: $ch! "Ti" \rightarrow Ti$.

CONCLUSION

In this CSP model of Petri nets variables contents model Petri net marking. All reachable markings for the Petri net are generated by this process. With addition of output channel all

transition sequences could be generated by the process.

Now, with the CSP model we can easily extent Petri nets to UML activity diagrams, BPMN diagrams and WS-BPEL diagrams, and more precisely to analyze them.

Acknowledgments. This research is supported by Project 240/2010 "Development of Grid infrastructure for research and education" funded by the Scientific Research Fund of University of Sofia.

REFERENCE

- [1] BPMN, <http://www.bpmn.org/>
- [2] UML, <http://www.uml.org/>
- [3] OASIS Web Services Business Process Execution Language (WSBPEL) TC, http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsbpel
- [4] Turing machine, http://en.wikipedia.org/wiki/Turing_machine
- [5] Finite-state machine, http://en.wikipedia.org/wiki/Finite-state_machine
- [6] Remco M. Dijkman, Marlon Dumas, Chun Ouyang, Semantics and analysis of business process models in BPMN, http://www.inf.pucrs.br/~bastos/ModelagemSistemasInformacao/IST_SmenaticsandAnalysisofBusinessProcess_ModelsinBPMN.pdf
- [7] S. Bernardi, S. Donatelli, J. Merseguer, From UML Sequence Diagrams and Statecharts to analysable Petri Net models, Proceedings of the 3rd international workshop on Software and performance 2002, Rome, Italy, July 24 - 26, ACM New York, NY, USA, 2002, pp. 35-45.
- [8] Niels Lohmann, A feature-complete Petri net semantics for WS-BPEL 2.0, Proceedings of the 4th international conference on Web services and formal methods, Brisbane, Australia, Springer-Verlag Berlin, Heidelberg, 2007, pp. 77-91.
- [9] Tools and Software, <http://www.informatik.uni-hamburg.de/TGI/PetriNets/tools/>
- [10] Communicating sequential processes, http://en.wikipedia.org/wiki/Communicating_sequential_processes
- [11] Pi calculus, http://en.wikipedia.org/wiki/Pi_calculus
- [12] Z-notation, http://en.wikipedia.org/wiki/Z_notation