Transformation and Visualization of BPMN Models to Petri Nets

Lei Li1, Fei Dai2*

1 School of Information Science & Engineering, Yunnan University, Kunming, 650091, China
2 National Pilot School of Software, Yunnan University, Kunming, 650091, China

Abstract. The Business Process Modeling Notation (BPMN) is a language for capturing business processes. It is able to model the complex concepts of systems such as: choice, parallel, and synchronization. However it suffers from the lack of an unambiguous definition, so we may produces models with semantic errors. On the other hand, Petri nets are a formal tool with solid mathematical principles [4]. Moreover, it offers a number of analytical techniques. So Petri nets and BPMN have complementary abilities: BPMN can handle the modeling and Petri nets can handle the analysis. In this paper, we propose an approach for transforming BPMN models to Petri nets models, then assign coordinates of Petri nets elements and control the path of every arc. Finally, we get a valid and neat Petri net model which is both convenient and practical for human analysts and other Petri nets analysis techniques.

1. Introduction

The Business Process Modeling Notation (BPMN) [1] is a language for capturing business processes. It is able to model the complex concepts of systems such as: choice, parallel, and synchronization. It combines elements from many notations for business process modeling, including Activity Diagrams component of the Unified Modeling Notation (UML) [2], the XML Process Definition Language (XPDL) and so on. But BPMN suffers from the lack of an unambiguous definition, so we may produces models with semantic errors.

Petri nets are a formal tool with solid mathematical principles [3]. Moreover, it offers a number of analytical techniques. So Petri nets and BPMN have complementary abilities: BPMN can handle the modeling and Petri nets can handle the analysis. Analyses of BPMN models capturing critical applications can be helpful to ensure the absence of errors and the correctness. There are many previous works focusing on the transformation of BPMN models to Petri nets models. In [4], the authors used the meta-modelling and model-transforming to generate Petri nets models for BPMN using AToM3 graph tool. In [5], the authors map the BPMN models to Kripke structure and check the validity of major properties.

In this paper, we propose an approach for transforming BPMN models to Petri nets models, then assign coordinates of Petri nets elements and control the path of every arc. Finally, we get a valid and neat Petri net model which is both convenient and practical for human analysts and other Petri nets analysis techniques.
2. Subset of BPMN
This paper reports experiences in defining a mapping between a subset of BPMN and Petri nets. The subset of BPMN mainly consists of elements for control-flow perspective, it is illustrated in Figure 1(Icons for BPMN elements are collected from Yaoqiang BPMN Editor [6], it is a graphical editor for business process diagrams, compliant with OMG specifications). Other non-functional features such as artifacts, groups and associations will be dealt with in future work.

![Figure 1. The subset of BPMN to be mapped](image)

3. Mapping
For each element in the subset that we proposed, we define mapping rules in Table 1(Icons for Petri nets are gathered from Platform Independent Petri net Editor [7]). The mapping rules are atomic, so they are applicable for an arbitrary combination of elements in the subset.

| BPMN element | Petri net element | Rule description |
|--------------|------------------|-----------------|
| Start Event  | place transition | Start Event in BPMN is mapped into a Petri net element combination including a place, an arc and a transition. |
| End Event    | transition place | End Event in BPMN is mapped into a Petri net element combination including a transition, an arc and a place. |
| Intermediate Throw Event | transition | Intermediate Throw Event, Choreography Task and Parallel Gateway in BPMN are mapped into a transition. |
| Exclusive Gateway | place | Exclusive Gateway and Inclusive Gateway in BPMN are mapped into a place. |
In a valid Petri net, no arc may connect two places or two transitions, so the mapping rule for sequence flow is divided into four conditions shown in the left column.

### 4. Preprocessing
Before mapping, we should preprocess the input BPMN model. The preprocessing meets the need for semantic equivalence and is without loss of generality. BPMN standards allow for graphs which are unconnected, having no start or end events and so on. Such BPMN models are not applicable for mapping. Therefore we need to transform BPMN models into well-formed ones according to rules presented in Table 2.

| No. | Before transforming | After transforming | Rule description |
|-----|---------------------|--------------------|------------------|
| 1   | ![Diagram](example1.png) | ![Diagram](example2.png) | Insert a parallel gateway for a start event which has an outdegree of more than one. |
| 2   | ![Diagram](example3.png) | ![Diagram](example4.png) | Insert an exclusive gateway for an end event which has an indegree of more than one. |
| 3   | ![Diagram](example5.png) | ![Diagram](example6.png) | Insert a parallel gateway for an intermediate throw event which has an outdegree of more than one. |
| 4   | ![Diagram](example7.png) | ![Diagram](example8.png) | Insert an exclusive gateway for an intermediate throw event which has an indegree of more than one. |
| 5   | ![Diagram](example9.png) | ![Diagram](example10.png) | Insert a parallel gateway after a parallel gateway which has an indegree of more than one and outdegree of more than one. Then apply the rule No.6 below. |
| 6   | ![Diagram](example11.png) | ![Diagram](example12.png) | If a parallel gateway connects a parallel gateway, we insert a choreography task between them. The input may be the result of the rule No.5 above or exists independently. |
| 7   | ![Diagram](example13.png) | ![Diagram](example14.png) | Insert an exclusive (inclusive) gateway after an exclusive (inclusive) gateway which has an indegree of more than one and outdegree of more than one. Then apply the rule No.8 below. |
| 8   | ![Diagram](example15.png) | ![Diagram](example16.png) | If an exclusive (inclusive) gateway connects an exclusive (inclusive) gateway, we insert a choreography task between them. The input may be the result of the rule No.7 above or exists independently. |
5. Layout of Petri net elements
Preprocessing transforms BPMN models into well-formed BPMN models into Petri net elements. Some analysis of Petri net can work well without displaying elements in the user interface such as the reachability problem for Petri nets which is EXPSPACE-hard, it gives a Petri net N and a marking M, then walk the reachability graph until either we reach the requested marking or we know it cannot be found in further attempts [8]. However in many cases, it’s necessary for researchers to analyse manually the semantics of Petri nets. To bolster human analysts, we should assign coordinates of Petri net elements. Otherwise, elements will overlap together at the top-left corner of the Petri net editor such as PIPE.

A simple layout scheme is to use coordinates of the source BPMN model. However, we define a new layout scheme instead of adopting it for two reasons. Firstly, a standardised layout of a Petri net is propitious to visual analysis. We can easily seek out differences among Petri nets by checking elements at the same position. Coordinates assigned manually cannot offer such a convenience. Secondly, some Petri net elements have features because of the layout rules. For example, elements which have an indegree of zero will always locate on the left side of the Petri net editor.

We use a modified version of the topological sort algorithm [9] to determine the order for assigning coordinates of Petri net elements. A topological sort is a linear ordering of vertices in a directed graph. One of the topological ordering algorithms works as follows. Firstly, find all nodes that have no incoming edges and add them into a set S. While S is non-empty, we continuously remove a node n from S, remove edges come from the node n and insert nodes which have no incoming edge into S. Until the loop ends, if the graph still contains edges, the graph has at least one cycle. Otherwise, we acquire a topologically sorted order successfully.

Since a Petri net can be cyclic, we require a few modifications of the topological ordering algorithm. We show the procedures of our algorithm in Figure 2.
6. Controlling the path of an arc
In a Petri net instance, if we only define the start point and the end point of an arc, there will be a straight line between two points. In a complex Petri net, these straight lines may overlap with other elements and arcs. It’s a confusing situation which is bad for human analysts.

We use a modified version of A* search algorithm to control paths of arcs. A* (pronounced as "A star") is a computer algorithm that is widely used in path finding. Specifically, A* finds a path which minimizes $f(n)=g(n)+h(n)$, $g(n)$ is the total cost from the start node to $n$, and $h(n)$ is a heuristic which values the cost from $n$ to the goal. Figure 3 shows details, modifications are underlined. We introduce some definitions of this flowchart: $S(x, y)$ stands for the start point of an arc; $E(x, y)$ stands for the end point of the arc. A variable $D$ stands for the minimal distance between two points in the horizontal and vertical direction. A variable $RA$ stands for the number of the right angles in the path. When $RA\leq 2$, we will get a neat path in our experiments.
7. A transforming case
To support the validation and correctness of our approach, we have implemented a tool written in JAVA that transform the BPMN models which is a XML serialization supported by an existing BPMN tool, into Petri net models which are described in the Petri Net Markup Language (PNML). After transforming, we will get a valid and neat Petri net model which is both convenient and practical for human analysts and other Petri nets analysis techniques.

We use a sample BPMN model “Incident Management as Choreography” which is offered by Business Process Incubator to show our approach and implementation. Business Process Incubator is the best BPM resource which encompasses everything BPM related on the web. Figure 4 shows the BPMN model graphic of the sample “Incident Management as Choreography”. Figure 5 shows the output Petri net provided by our program for the input sample.
Figure 4. The input BPMN model “Incident Management as Choreography”

Figure 5. The output Petri net model of “Incident Management as Choreography”

References

[1] Object Management Group. Business Process Modeling Notation (BPMN) Version 1.0. OMG Final Adopted Specification. Object Management Group, 2006.

[2] Object Management Group. Unified Modeling Language: Superstructure. UML Superstructure Specification v2.0, formal/05-07-04. Object Management Group, 2005.

[3] Rozenburg, G.; Engelfriet, J. (1998). "Elementary Net Systems". In Reisig, W.; Rozenberg, G. Lectures on Petri Nets I: Basic Models – Advances in Petri Nets. Lecture Notes in Computer Science. 1491. Springer. pp. 12–121.

[4] R. El Mansouri "On the use of Meta-Modelling and Graph Grammars to Generate Petri nets models for Business Processes", IRECONS Journal, January, 2008 issue.

[5] Oussama Mohammed Kherrouche, Adeel Ahmad, and Henri Basson," Using model checking to control the structural errors in BPMN models". RCIS2013: 1-12.

[6] Yaoqiang, Inc., Yaoqiang BPMN Editor - an Open Source BPMN 2.0 Modeler, http://www.yaoqiang.org/, March 23 2018.

[7] N.J. Dingle, W.J. Knottenbelt and T. Suto. PIPE2: A Tool for the Performance Evaluation of Generalised Stochastic Petri Nets (PDF format). ACM SIGMETRICS Performance Evaluation Review (Special Issue on Tools for Computer Performance Modelling and Reliability Analysis), Vol. 36(4), March 2009, pp. 34-39.

[8] Lipton, R. (1976). "The Reachability Problem Requires Exponential Space". Technical Report 62. Yale University.

[9] Kahn, Arthur B. (1962), "Topological sorting of large networks", Communications of the ACM, 5 (11): 558–562, doi:10.1145/368996.369025.