Transforming XPDL to Petri Nets

Conference Paper · September 2007

DOI: 10.1007/978-3-540-78238-4_21 · Source: DBLP

CITATIONS
9

READS
385

4 authors:

Haiping Zha
Tsinghua University
5 PUBLICATIONS **133** CITATIONS
SEE PROFILE

Yun Yang
Fudan University
276 PUBLICATIONS **5,176** CITATIONS
SEE PROFILE

Jianmin Wang
Suzhou University
277 PUBLICATIONS **11,151** CITATIONS
SEE PROFILE

Lijie Wen
Tsinghua University
133 PUBLICATIONS **2,377** CITATIONS
SEE PROFILE

Transforming XPDL to Petri Nets

Haiping Zha[1,3,4], Yun Yang[2], Jianmin Wang[1,3,4], and Lijie Wen[1,3,4]

[1] School of Software, Tsinghua University, Beijing, P.R. China, 100084
{chp04,wenlj00}@mails.tsinghua.edu.cn, jimwang@tsinghua.edu.cn
[2] Faculty of Information and Communication Technologies
Swinburne University of Technology, Melbourne, Australia, 3122
yyang@it.swin.edu.au
[3] Key Laboratory for Information System Security, Ministry of Education, P.R. China, 100084
[4] Tsinghua National Laboratory for Information Science and Technology, P.R. China, 100084

Abstract. As a textual specification for process definition, XPDL lacks formal semantics which hinders the formal analysis and verification of business processes. In this paper, we provide a method for translating XPDL processes into Petri nets for the formal analysis of XPDL processes. The algorithm validity has been proved, and has also been verified by experiments on artificial and practical processes.

1 Introduction

The XML Process Definition Language (XPDL) is a formal standard process definition language proposed by the Workflow Management Coalition (WfMC) [14]. The purpose is to serve as an exchange language between different modeling languages. Today there are over 50 major business process management and application vendors that support the XPDL standard, including IBM, Oracle, BEA, Fujitsu, Tibco, and Global 360 [8]. However, As a textual specification of process definition, XPDL lacks formal semantics which serves as the foundation of formal analysis and computer aided verification.

To analyze process models without formal semantics, usually we can first transform them into formal models. A lot of such efforts have been dedicated to analyze BPEL, e.g., to finite state machines [4,5], to process algebra [3], to Petri nets [11,6], and our former work to OWL-S [10], etc. And a Petri net is a rather desirable target language of such transformation, with its formal semantics and the availability of many analysis techniques and tools [1].

Although XPDL is a competitive standard to BPEL, no attempts on transforming XPDL have been published so far. In this paper, we present a mapping method from XPDL to Petri nets. For each XPDL process model, we can get a Petri net with equivalent structure and behaviors. The Petri net can be exported to a PNML [13] file for further analysis or simulation purposes.

The remainder of this paper is organized as follows. Section 2 gives a brief overview on XPDL and process meta-model. Section 3 presents a mapping semantics between XPDL and Petri nets. Section 4 presents the algorithm step

A. ter Hofstede, B. Benatallah, and H.-Y. Paik (Eds.): BPM 2007 Workshops, LNCS 4928, pp. 197–207, 2008.

by step and its validity proof. Section 5 gives case study details. Section 6 discusses the related work and our contributions. Section 7 concludes the paper and discusses future work.

2 A Brief Introduction to XPDL

The WfMC has identified five functional interfaces to a workflow service as part of its standardization program. XDPL forms part of the documentation relating to *Interface one* - supporting process definition import and export. XPDL is a common meta-model for describing the process definition. The purpose is to serve as an interchange of process definition between different tools and also different vendors. The first version of a standard interchange language was the Workflow Process Definition Language (WPDL), published by the WfMC in 1998. The growing popularity of XML and its use for defining document formats for the Internet, combined with some years of accumulated experience using WPDL in workflow and BPM tools, led to the creation of XPDL 1.0, which was officially released in 2002. XPDL retained the semantics of WPDL but defined a new syntax using an XML schema. Neither WPDL nor XPDL 1.0 proposed a specific graphical representation. Intended to be used as a file format for Business Process Modeling Notation (BPMN), XPDL 2.0 was published in 2005, which is back compatible with XPDL 1.0 [9].

Figure 1 is an XPDL process meta-model that describes the top-level entities contained within a process definition, with their relationships and attributes. We have skipped XPDL package meta-model, for XPDL package is just a container for XPDL processes. For more details of XPDL meta-model, we can refer to [14].

Currently dozens of vendors have announced conformance of XPDL. A more detailed list of vendors and products can be found on the WfMC website [1].

3 Mapping from XPDL to Petri Nets

Our goal is to translate an XPDL process model into a Petri net. However, we do not pursue a complete Petri net semantics to XPDL. We focus on structure and behavior equivalence. As shown in Figure 1, a process meta-model includes several elements, e.g., activities, participants, applications, transitions and data fields, etc. We skip some elements which are not related to our concerns, e.g., applications, data fields and other extended attributes of various tool vendors, etc. In this section, we introduce the mapping semantics in three aspects, namely, activities, links and routing structures.

3.1 Activities

An XPDL activity can have attributes and child elements, such as *Id*, *Name*, *Performer*, and *Join/Split type*, etc. An activity is either primitive or structured

[1] $http://www.wfmc.org/standards/conformance\_chart.htm$

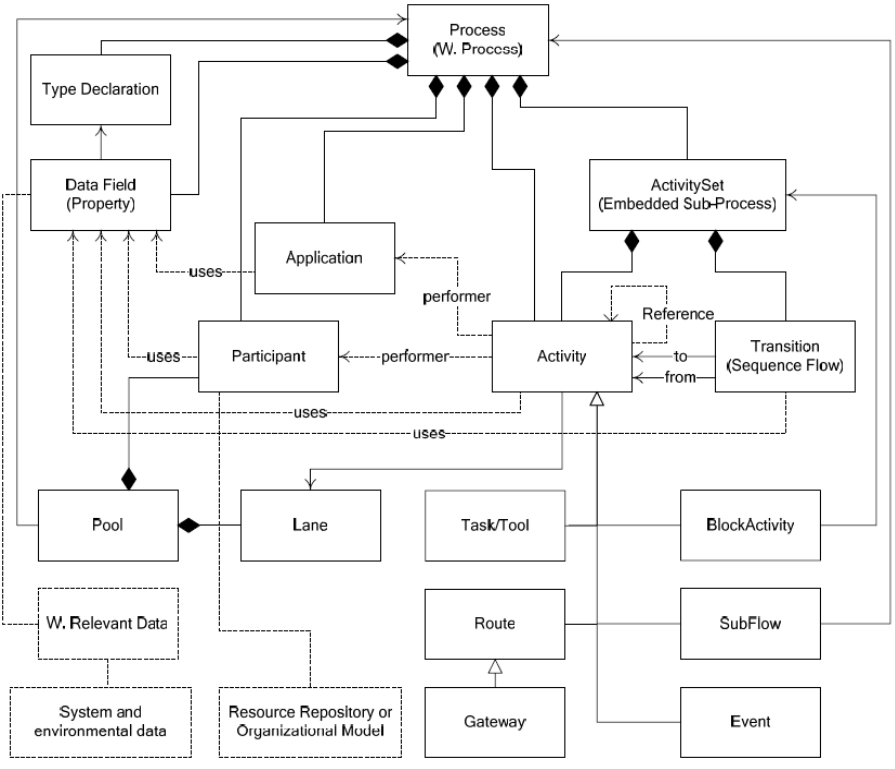

Fig. 1. XPDL Process Meta-Model

(activity sets). An sample of the primitive activity element is shown as follows.

$< Activity \;\; Id = \text{“}Act1\text{”} \;\; Name = \text{“}A\text{”} >$
$\quad + < Implementation >$
$\quad\quad < Performer > Par1 < /Performer >$
$\quad + < StartMode >$
$\quad + < FinishMode >$
$\quad + < TransitionRestrictions >$
$\quad\quad < TransitionRestriction >$
$\quad\quad\quad + < Join \;\; Type = \text{“}AND\text{”} >$
$\quad\quad\quad + < Split \;\; Type = \text{“}XOR\text{”} >$
$\quad\quad < /TransitionRestriction >$
$\quad < /TransitionRestrictions >$
$\quad + < ExtendedAttributes >$
$< /Activity >$

It is instinctive to map *XPDL activities* to *Petri transitions*. A PNML Petri model is ready to integrate limit activity attributes, e.g., *Id*, *Name*. Other attributes and child elements need an extended Petri net model. As to the Join/Split

type attributes, they are no longer a part of Petri net transition. The Join/Split type will be discussed in Section 3.3.

3.2 Links

Links are literally *transitions* in XPDL process. To avoid confusing with the term *transition* in Petri nets, we use links instead throughout this paper. A link element is shown as follows.

$< Transition \quad From =$ "$Act1$" $\quad Id =$ "$Tra2$" $\quad To =$ "$Act2$" $>$
$\quad + < ExtendedAttributes >$
$< /Transition >$

A link is a relatively simple element, which maps to an arc in a Petri net. A link can have attributes, such as *Id*, *From* (Source activity) and *To* (target activity).

There is a challenging problem when we map a *XPDL link* to a *Petri net arc*. In the XPDL context, the source object and target object of a link are both activities. But according to the Petri net syntax, an arc cannot connect two transitions directly. So a *Petri net place* must be inserted between two transitions. Roughly speaking, an *XPDL link* maps to two *Petri net arcs*, with one arc's target on the inserted place, and the other arc's source on the same one.

However, the problem is only partly resolved. In a Petri net, a routine structure depends on the connections of arcs between transitions and places. Just inserting a place between two neighboring transition is only correct syntactically, the semantic correctness cannot be guaranteed. In fact the situation is even worse, because the semantic incorrectness may be different under different context. Hence we cannot find a simple method to remedy.

To avoid the uncertainty of semantic incorrectness, we introduce a mapping technique which considers the correctness both syntactically and semantically. In this step, we add an input place for each transition. We map an XPDL link to a Petri net arc between the source transition and the target transition's input place, regardless of the join/split conditions of both XPDL activities. Then we derive a Petri net with correct syntax. Although semantic incorrectness still exists, this time it becomes simpler to fix, because the semantic incorrectness can only be from two routing structures, i.e., AND-join structure and XOR-split structure. The correct measures are presented in Section 3.3.

3.3 Routing Structures

In an XPDL process model, there are explicit routing structures other than links. As mentioned in Section 3.1, each activity can have two kinds of join and split types, i.e., AND-join, XOR-join, AND-split and XOR-split, as shown in Figure 2 (a). However, in a Petri net there is no corresponding structure. A Petri net implements routing structure through right connection between transitions and places, as shown in Figure 2 (b). In Section 3.2 we derived a Petri net without AND-join and XOR-split structure. A correcting adjustment of the derived model is necessary, i.e., (1) correcting an XOR-join structure to an AND-join structure, and (2) correcting an AND-split structure to an XOR-join.

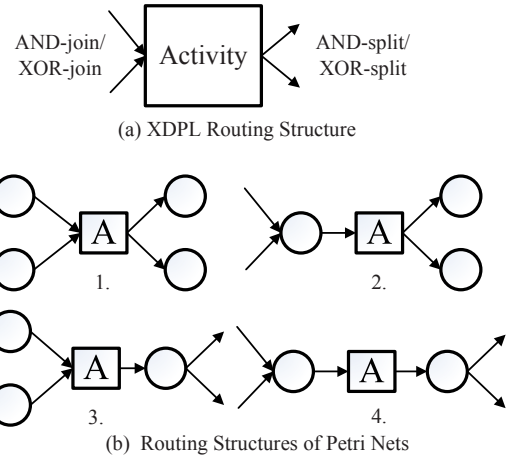

Fig. 2. Routing Structures Mapping

As shown in Figure 2 (b), 2 and 4 have an XOR-join structure, and 1 and 3 have an AND-join structure. In our algorithm, there is no other output arc of a transition's input place, except the arc to the transition itself. So the adjustment impact is local which means the operation will not involve other transitions, as shown in Figure 3 (a). For each input arc of the input place, the first step splits a new place for each input arc. The second step adds an input arc linking the new place and the transition. The final step deletes the old input place and the input arc.

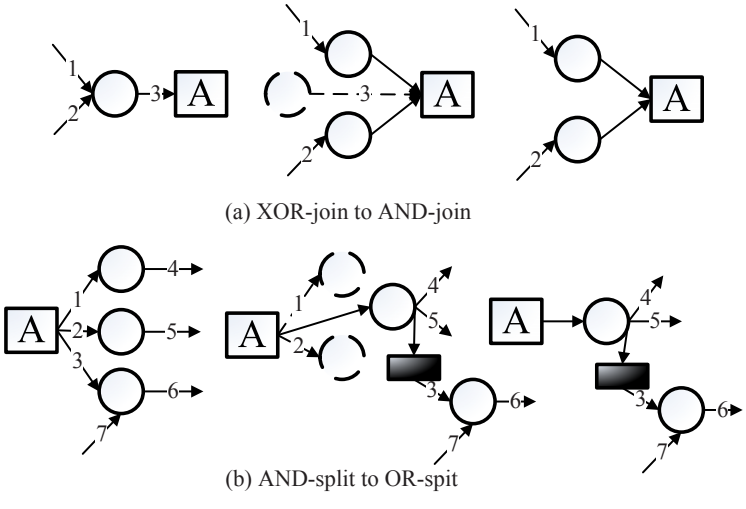

Fig. 3. Correction of Routing Structure

To correct an AND-split structure to an XOR-split it is similar, just through incorporating output arcs and output places. But there is an exception when the output place's in-degree is more than 1. This case needs to introduce a routing transition, as shown in 3 (b). If we do not distinguish Arc 6 from Arc 4 and Arc 5 in 3 (b), behaviors of the transformed Petri net will be changed.

4 Algorithm

The following steps are the mapping algorithm from an XPDL process model to a Petri net.

1. Map each *activity* in XPDL to a *transition* in Petri nets, along with their *attributes*.
2. Add an input *place* and an input *arc* for each transition, except for which has no input links in XPDL.
3. Map each *link* to an *arc*, linking source transition and target transition's input place.
4. Correct *AND-join* structure.
5. Correct *XOR-split* structure.
6. Add a *source place* and a *sink place*.

The details of each step have been discussed in Section 3 except Step 6. The purpose of Step 6 is to construct a Wf-net [1], because a Wf-net is more desirable for the analysis purpose. The only one source place serves as the input place for all transitions without any input places. And the only one sink place serves as the output place for all transitions without any output arcs.

Theorem 1. *The Petri net is equivalent with the XPDL model in structure and observed behaviors, i.e., the transforming algorithm is correct.*

Proof. 1. Step 1 guarantees that the two models have the same activity set.
2. Step 2 and Step 3 make that all activity ordering relations are the same in the two models.
3. Since the algorithm treats any *join structure* as XOR-join and any *split structure* as AND-split, the original XOR-join and AND-split structures remain unchanged after transformation.
4. In the previous steps, AND-join structures are treated as XOR-join structures and XOR-split structures as AND-split structures improperly. Step 4 and Step 5 remedy these flaws respectively.

Therefore, the transforming algorithm guarantees the same structure of the two models. Consequently, it guarantees the same observed behaviors. □

5 Case Study

In this section, we present our case study. The sample process is modelled by JaWE [7], which is a compatible process editor to XPDL, as shown in Figure 4. In

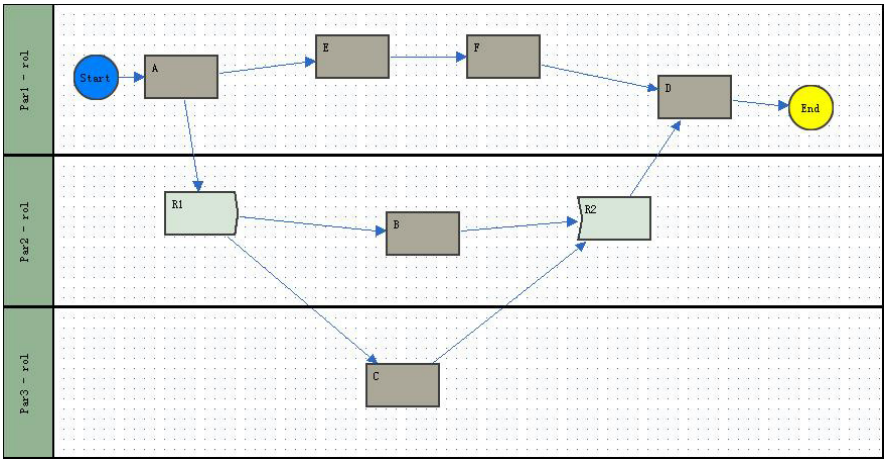

Fig. 4. Sample Process Definition in XPDL

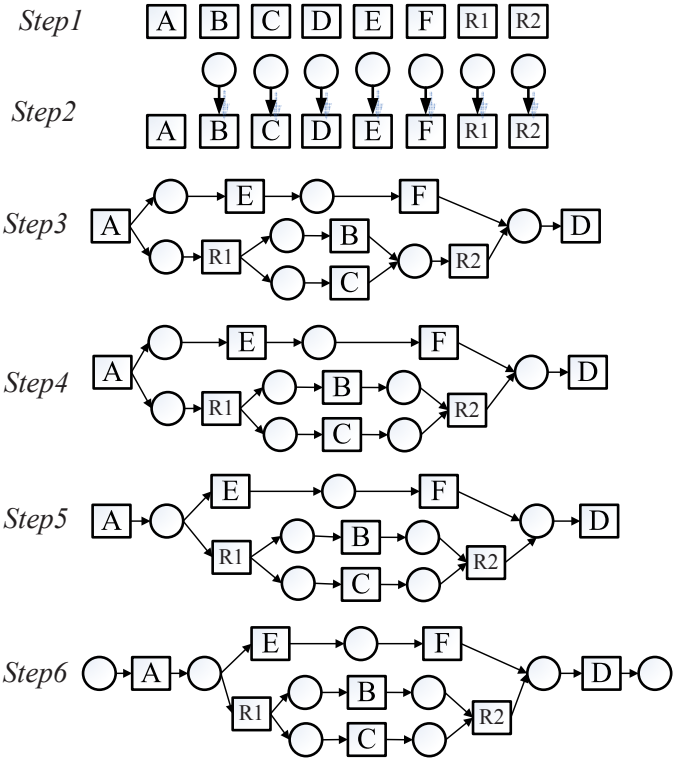

Fig. 5. Illustration of XPDL Transformation Algorithm Steps

the process, there are eight activities including two routing activities. There are three participants named par1 to par3. The routing structures include sequence structure (e.g., EF), split structure (e.g., AE or AR1), and parallel structure (e.g., BC). Correspondingly, there exist all possible Join and Split types, i.e., XOR-split (e.g., A), AND-split (e.g., R1), XOR-join (e.g., D) and AND-join (e.g., R2).

Figure 5 illustrates the transformation steps. In Step 1, each activity in the XPDL process model maps to a transition of a Petri net, with the same Name. In Step 2, each transition is added with an input place and a connect arc except Transition A, because corresponding activity of Transition A has no input link in the XPDL process. After Step 3, a Petri net with correct syntax is generated, but it is not correct semantically. Note that there are only one type of the join structure, i.e., the XOR-join structure, and only one type of the split structure, i.e., the AND-split structure. Therefore, the following two steps are necessary to correct the Petri net with AND-join structure (in Step 4) and XOR-split structure (in Step 5). Finally, Step 6 is an additional step to construct a Wf-net.

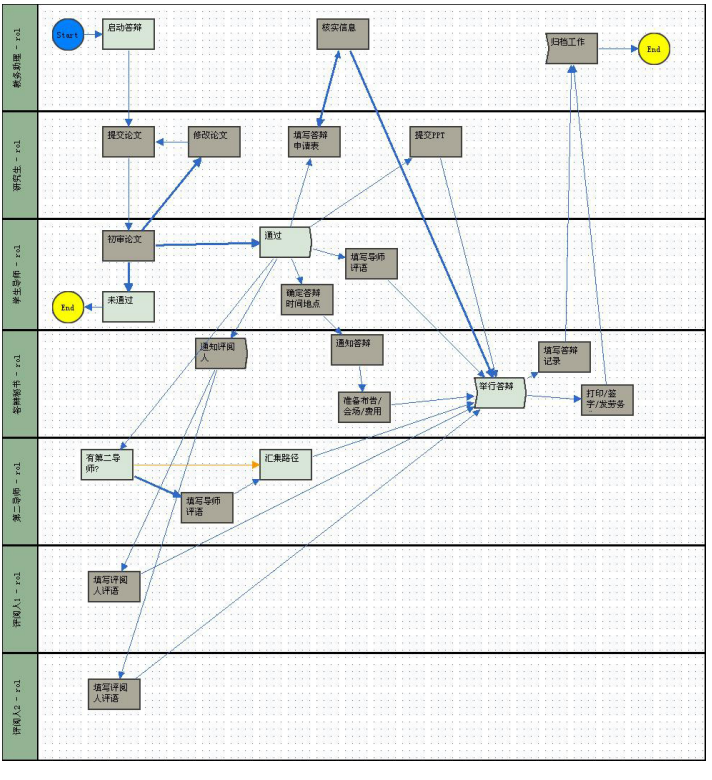

Fig. 6. XPDL Definition Model of Graduate Dissertation Defense Process

Currently we have implemented our algorithm as part of our *ISEflow* project. In the project, we can import XPDL definition file generated by an XPDL conformance process editor, e.g. JaWE [7]. Then the transformed Petri net model can be exported to a PNML [13] file for further analysis, or to a lightweight Petri net based engine to collect execution logs. Since XPDL is popularly used in China, we can study rich practice process models in a sound manner through the transformation.

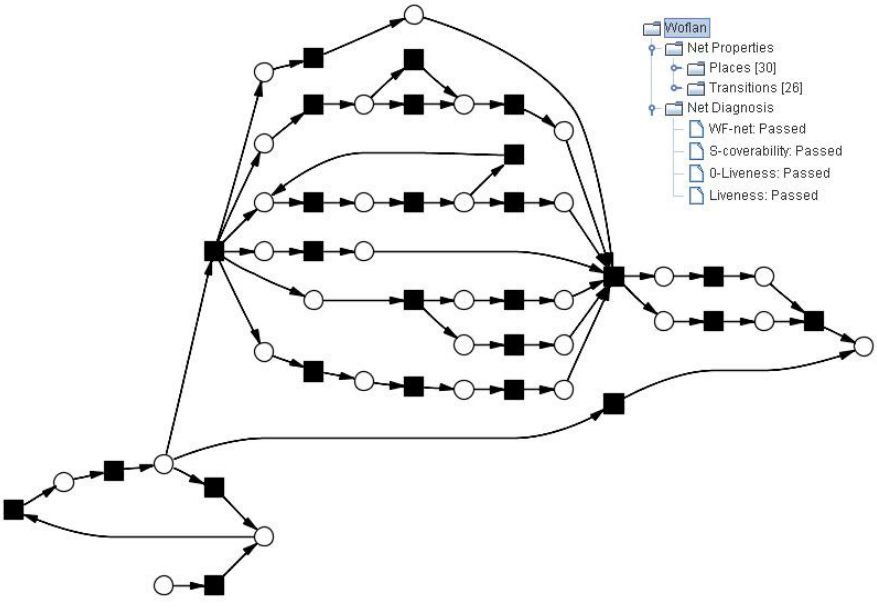

Fig. 7. Petri Net Corresponding to the XPDL Model in Figure 6 with Woflan Analysis Result

We have also conducted a lot of experiments on practical processes. In Figure 6, a real world XPDL model of a graduate dissertation defense process is shown. We reserve the activity labels in original Chinese for it does not matter to our understanding of the process structure and control flow. And in Figure 7, the corresponding Petri net model is presented with the analysis result by Woflan [12] in ProM [2].

6 Related Work

There are several publications that define transformations between different business process modeling languages, with the same motivation to analyze and verify business process. However, most of these efforts are for BPEL [4,5,3,11,6,10].

Petri nets (including Wf-nets) are a desirable target language of transformation, for its formal foundation and lots of available analysis theory and tools.

Similar efforts as ours can be found in efforts to transform BPEL to Petri nets. In [11] authors make an attempt to map a BPEL process model on a Wf-net for verifying process soundness. They classify activities into different categories according to their different semantics. And make a Wf-net mapping for each type activities respectively.

In [6] each construct of BPEL language is separately mapped into a Petri net. Each pattern has an interface for joining it with other patterns as is done with BPEL constructs. The collection of these patterns forms the Petri net semantics for BPEL.

The contribution of our approach lies that: (1) It is the first attempt to transform XPDL to Petri nets for formal analysis. In spite of the comparability between BPEL and XPDL, they are different language with different specification [8]. As to the best of our knowledge, there is no publication on formal transformation of XPDL yet. (2) We focus on structure and behavior equivalence. It enable us to develop an concrete algorithm whose validity can been proved, rather than present a set of sematic mapping rules.

7 Conclusion and Future Work

Formal semantics is the foundation of analysis and verification of process definition. This paper aims to automatically transform XPDL process models to Petri nets with equivalent structure and behaviors. We have presented the mapping semantics as well as concrete transforming algorithm. The algorithm has been proved and demonstrated experimentally.

As future work, we will seek to integrate the algorithm into ProM workflow process mining framework, so that XPDL models can be directly imported to ProM, then they can be analyzed by Petri net analysis tools in ProM. In addition, to support various analysis and simulation purposes of XPDL process, we will improve mapping semantics to cover more XPDL 2.0 specification.

Acknowledgements

This work is supported by the 973 Project of China (No. 2002CB312006), the Project of National Natural Science Foundation of China (No. 60473077 and No. 60373011), and the Program for New Century Excellent Talents in University of China.

References

1. van der Aalst, W.M.P.: The Application of Petri Nets to Workflow Management. The Journal of Circuits, Systems and Computers 8(1), 21–66 (1998)
2. van Dongen, B.F., de Medeiros, A.K.A., Verbeek, H.M.W., Weijters, A.J.M.M., van der Aalst, W.M.P.: The ProM framework: A New Era in Process Mining Tool Support. In: Ciardo, G., Darondeau, P. (eds.) ICATPN 2005. LNCS, vol. 3536, pp. 444–454. Springer, Berlin (2005)

3. Ferrara, A.: Web Services: a Process Algebra Approach. In: Proceedings of the 2nd international conference on Service oriented computing, pp. 242–251. ACM Press, New York (2004)
4. Fisteus, J.A., Fernández, L.S., Kloos, C.D.: Formal Verification of BPEL4WS Business Collaborations. In: Bauknecht, K., Bichler, M., Pröll, B. (eds.) EC-Web 2004. LNCS, vol. 3182, pp. 76–85. Springer, Berlin Heidelberg (2004)
5. Fu, X., Bultan, T., Su, J.: Analysis of Interacting BPEL Web Services. In: Proceedings of the 13th international conference on World Wide Web, pp. 621–630. ACM Press, New York (2004)
6. Hinz, S., Schmidt, K., Stahl, C.: Transforming BPEL to Petri Nets. In: van der Aalst, W.M.P., Benatallah, B., Casati, F., Curbera, F. (eds.) BPM 2005. LNCS, vol. 3649, pp. 220–235. Springer, Berlin Heidelberg (2005)
7. ObjectWeb. Enhydra JaWE: Open Source Java XPDL Editor, `http://www.enhydra.org/workflow/jawe/index.html`
8. Pyke, J.: Is XPDL the Silent Workhorse of BPM? (2007), `http://www.ebizq.net/hot_topics/bpm/features/7852.html`
9. Shapiro, R.M.: XPDL 2.0: Integrating Process Interchange and BPMN. In: 2006 Workflow Handbook, WfMC, pp. 183–194 (2006)
10. Shen, J., Yang, Y., Zhu, C., Wan, C.: From BPEL4WS to OWL-S: Integrating E-Business Process Descriptions. In: SCC2005. Proceedings of IEEE International Conference on Services Computing, Orlando, Florida, USA, pp. 181–188 (2005)
11. Verbeek, H.M.W., van der Aalst, W.M.P.: Analyzing BPEL Processes using Petri Nets. In: Proceedings of the Second International Workshop on Applications of Petri Nets to Coordination, Workflow and Business Process Management, pp. 59–78 (2005)
12. Verbeek, H.M.W., Basten, T., van der Aalst, W.M.P.: Diagnosing workflow processes using Woflan. The Computer Journal 44(4), 246–279 (2001)
13. Weber, M., Kindler, E.: The Petri Net Markup Language. In: Ehrig, H., Reisig, W., Rozenberg, G., Weber, H. (eds.) Petri Net Technology for Communication-Based Systems. LNCS, vol. 2472, pp. 124–144. Springer, Berlin Heidelberg (2003)
14. WfMC. Workflow Process Definition Interface – XML Process Definition Language (XPDL) (WFMC-TC-1025). Technical report, Workflow Management Coalition (2005)