Representation of Petri Net with OWL DL Ontology

Fu Zhang
College of Information Science & Engineering, Northeastern University, Shenyang, China
zhangfu216@126.com

Z. M. Ma
College of Information Science & Engineering, Northeastern University, Shenyang, China
mazongmin@ise.neu.edu.cn

Slobodan Ribarić
Faculty of Electrical Engineering & Computing, University of Zagreb, Zagreb, Croatia
slobodan.ribaric@fer.hr

Abstract—**With the wide utilization of Petri nets, many researchers suggest that Petri nets should be reused and shared. Emerging the Semantic Web technologies, such as ontologies, can play an important role in this scenario. In this paper, we propose an ontology approach for representing Petri nets. Firstly, we propose a complete formal definition of OWL DL ontologies. Then, Petri nets are introduced and a formal definition of Petri nets is given. On this basis, we propose an approach for representing Petri nets with ontologies, i.e., we translate some key features of Petri nets into classes, properties and axioms of OWL DL ontologies.**

Keywords—**Petri net; Ontology; Representation**

I. INTRODUCTION

As a way of modeling processes, Petri nets, have been extensively used in many areas of data processing for the modeling of software engineer, office automation, and data bases, particularly in the initial phase of system design [10], [16], [17], [25]. Despite the popularity of Petri nets, however, some inconsistent understanding appears in existing Petri net models, since current Petri net interoperability is possible at the level of syntax for model sharing [7]. Therefore, the Petri net models should be reused, shared and standardized, which makes us free of ambiguity in constructing the models.

Ontologies [3], which are supported by the W3C through Web Ontology Language [13] Recommendation, are a formal and explicit specification of a shared conceptualization and can enable semantic interoperability. With ontologies, it is possible to formally capture the shared knowledge of a given domain and represent this knowledge in a declarative and expressive language (explicit) that follows a logical paradigm (formal), i.e., classes, properties and axioms.

On this basis, representing Petri nets using ontologies can provide for semantic description of Petri net concepts and their relationships, which will contribute with a common semantics to improve the communication among communities. Moreover, with the wide utilization of Petri nets and the requirements of the Semantic Web, representing Petri nets by ontologies is needed in order to reuse them more effectively on the Semantic Web. For this purpose, several works made efforts to connect Petri nets with ontologies [4], [5], [6], [7], [9], [10], [15], [18], [20], [21], [23]. However, most of the existing works above focus on how to develop a high-level mapping between Petri nets and ontologies, and do not provide a detailed description about how to represent Petri nets with ontologies, i.e., how to translate some key features of Petri nets into classes, properties and axioms of ontologies.

To this end, in this paper, we propose a formal approach for representing Petri nets by ontologies. Firstly, we propose a kind of complete formal definition of OWL DL ontologies, where ontologies formulated in OWL DL language (a sublanguage of OWL [13]) are called OWL DL ontologies. Then, we introduce some basic notions of Petri nets and further give a definition of Petri nets. On this basis, we propose a formal approach for representing Petri nets with OWL DL ontologies, i.e., we translate some key features of Petri nets into classes, properties and axioms of OWL DL ontologies. Representing Petri nets with ontologies will facilitate the semantic interoperability of Petri nets and the development of the Semantic Web.

The remainder of this paper is organized as follows. Section 2 proposes a formal definition of OWL DL ontologies. Section 3 introduces Petri nets and gives a definition of Petri nets. Section 4 investigates how to represent Petri nets with ontologies. Section 5 shows conclusions and future work.

II. OWL DL ONTOLOGY

In order to represent the translated ontologies from Petri nets, it is necessary for us to give a formal definition of target ontologies. In the following, we propose a formal definition of OWL DL ontologies.

Ontologies can be defined by ontology definition languages such as RDFS, OIL, or OWL. *OWL* (*Web Ontology language*) [13], a W3C recommendation, is to be de-facto standard for ontologies. OWL has three sublanguages OWL Lite, OWL DL, and OWL Full. *OWL DL*, which is equivalent to Description Logic SHOIN(D), is the language chosen by the major ontology editors because it because it provides a good balance of expressive power and computability [8]. Therefore, we choose *OWL DL language* in our work. For the detailed syntax of OWL DL, please refer to [13].

An ontology formulated in OWL DL language is called *OWL DL ontology*, which consists of a set of *OWL DL identifiers* and *axioms*. The following Definition 1 gives a formal definition of OWL DL ontologies.

Definition 1 (OWL DL ontology). An OWL DL ontology is a couple $O = (ID_0, Axiom_0)$, where:

(1) $ID_0 = CID_0 \cup IID_0 \cup DRID_0 \cup PID_0$ is an OWL DL identifier set partitioned into:

- a subset CID_0 of class identifiers.
- a subset IID_0 of individual identifiers.
- a subset $DRID_0$ of data range identifiers; each data range identifier is a predefined XML datatype.
- a subset PID_0 of property identifiers, which includes object property identifiers and datatype property identifiers.

(2) $Axiom_0$ is a finite OWL DL axiom set:

- a subset of class/property axioms, used to represent the ontology structure.
- a subset of individual axioms, used to represent the ontology instances.

From a semantic point of view, an OWL DL ontology O is a set of OWL DL axioms. We say that an interpretation I is a model of O iff it satisfies all axioms in O.

III. Petri Nets

In order to help readers to understand what Petri nets are, it is necessary to introduce some basic notions of Petri nets based on [10], [16], [22]. On this basis, we propose a simple formal definition of Petri nets.

A *Petri net* is a particular kind of directed graphs populated by several types of objects, which are *places*, *transitions*, *directed arcs*, *tokens*. *Directed arcs* connect *places* to *transitions* or *transitions* to *places*. A *transition* is activated when every *place* in the precondition of the transition is fulfilled. In briefly, a Petri net can be represented by a transition together with an input place and an output place. *Tokens* are a primitive concept for Petri nets in addition to places and transitions. The presence or absence of a token in a place can indicate whether a condition associated with this place is true or false.

Let us begin with an example to well introduce the Petri net. In a market, *dealers* offer goods for sale and *customers* seek to buy the goods they need. When a dealer and a customer have agreed on a product and a price there is an exchange of goods and money. A business transaction takes place. After that, the dealer can acquire new goods and the customer new money and further business transactions will be possible. Fig. 1 shows a business transaction between a customer y and a dealer x. Here, the arrows of the transition "x with y" are labeled with two variables x and y, and for this transition to occur, y must be replaced by a customer and x by a dealer. Similarly, "x gets goods" for the dealers, and "y gets money" for the customers.

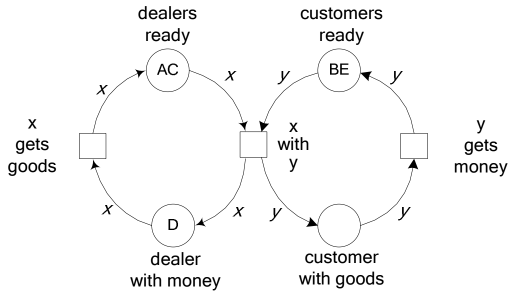


Fig. 1. The market with several customers and dealers represented as a net in which variables are introduced.

Based on the introduction above, in briefly, a Petri net consists of:

- places, represented as circles (○);
- transitions, represented as boxes (□);
- arrows from places to transitions (○→□);
- arrows from transitions to places (□→○);
- individual, distinguishable objects, that can flow through the net as tokens;
- an initial marking that defines for every place what objects it contains in the beginning;
- a label on every arrow can be an expression, which may be individual objects, variables, or arbitrary combinations of constants, variables and operations (e.g., 2x, x+y, x+1, (x, y), etc.).
- a substitution for a transition t is constituted by the fact that on the arrows that begin or end at t every variable in the expressions is replaced by an individual object;
- a transition t activated w.r.t. a substitution will occur in that: (i) the same number of objects will be removed from every place p in the pre-set of t as indicated by the replacement of the expression of the arrow from p to t, and (ii) every place p' in the post-set of t has added to it the same number of objects as indicated by the substitution of the expression of the arrow from t to p'.

The following Definition 2 gives a simple formal definition of Petri nets based on [12], [24]. In briefly, a Petri net mainly consists of *places (tokens)*, *transitions* and *directed arcs*. Here, other constituents (e.g., variables, operations, etc.) are omitted in the definition.

Definition 2 (Petri nets). A Petri net is formally defined as a tuple $PN = (P, T, F, M_0)$, where:

- $P = \{p_1, p_2, \dots, p_n\}$ is a finite set of places;
- $T = \{t_1, t_2, \dots, t_n\}$ is a finite set of transitions, $P \cup T \neq \emptyset$, and $P \cap T = \emptyset$;
- $F \sqsubseteq P{\times}T \cup T{\times}P$, which may be an *input function* I: $P \times T \rightarrow N$ defining directed arcs from places to transitions; or an *output function* O: $T \times P \rightarrow N$ defining directed arcs from transitions to places, where N is a set of nonnegative integers;
- M_0: $P \rightarrow N$ is the *initial marking*.

Note that, a *marking* in a Petri net is an assignment of tokens to the places of a Petri net. Tokens, which reside in the places of a Petri net, are used to define the execution of a Petri net. The number and position of tokens may change during the execution of a Petri net.

IV. Representation of Petri Net with OWL DL Ontology

This section proposes an approach for representing Petri nets with OWL DL ontologies, i.e., we translate some key features of Petri nets into classes, properties and axioms of OWL DL ontologies.

Definition 3 gives a formal approach for representing Petri nets with OWL DL ontologies, i.e., translating Petri nets into OWL DL ontologies. Starting with the construction of *OWL*

DL identifiers from *places*, *transitions*, *directed arcs* and *tokens* of a Petri net (e.g., a *place* and an *arc* correspond to a class and a property in ontology, respectively), Definition 3 induces a set of *axioms* from the Petri net. The ontology can then be evaluated by the domain experts.

Definition 3 (representation of Petri nets). Let $PN = (P, T, F, M_0)$ be a Petri net. The OWL DL ontology $O = \varphi(PN) = (ID_0, Axiom_0)$ can be defined by transformation function φ as follows:

(1) The OWL DL identifier set ID_0 of $\varphi(PN)$ contains following elements:

- Each *place* symbol $p_i \in P$ is mapped into a class identifier $\varphi(p_i) \in CID_0$;
- Each *transition* symbol $t_i \in T$ is mapped into a class identifier $\varphi(t_i) \in CID_0$;
- Each *arc* from a place to a transition $f_{ptot} \in F$ is mapped into a property identifier $\varphi(f_{ptot}) \in PID_0$;
- Each *arc* from a transition to a place $f_{ttop} \in F$ is mapped into a property identifier $\varphi(f_{ttop}) \in PID_0$;
- Each *token* inside in a place $tk_i \in M_0$ is mapped into an individual identifier $\varphi(tk_i) \in IID_0$;
- A class identifier φ(CPlace) $\in CID_0$ denotes all the places in *PN*;
- A class identifier φ(CTran) $\in CID_0$ denotes all the transitions in *PN*;
- A class identifier φ(CNode) $\in CID_0$ denotes all the places and transitions in *PN*;
- A property identifier $\varphi(\text{arc}_{ptot}) \in PID_0$ denotes all the arcs from places to transitions;
- A property identifier $\varphi(\text{arc}_{ttop}) \in PID_0$ denotes all the arcs from transitions to places;
- A property identifier φ(arc) $\in PID_0$ denotes all the arcs in *PN*;
- A class identifier owl:Thing $\in CID_0$ is used to denote all elements in Petri net *PN*, and another class identifier owl:Nothing $\in CID_0$ denotes empty sets;

(2) The OWL DL axiom set $Axiom_0$ of $\varphi(PN)$ contains following elements:

- An arc in *PN* means that its input is a place (transition) and output is a transition (place), and thus it is necessary to define the domain and range of the corresponding property identifiers $\varphi(\text{arc}_{ptot})$ and $\varphi(\text{arc}_{ttop})$. Then creating the property axioms:

  ObjectProperty ($\varphi(\text{arc}_{ptot})$ domain (φ(CPlace)) range (φ(CTran)));

  ObjectProperty ($\varphi(\text{arc}_{ttop})$ domain (φ(CTran)) range (φ(CPlace)));

- A place can not be a transition simultaneously in *PN*, i.e., a class identifier φ(CNode) is either a class identifier φ(CPlace) or a φ(CTran). Then creating the class axiom:

  Class (φ(CNode) partial unionOf
  (intersectionOf (φ(CPlace) complementOf(φ(CTran)))
  intersectionOf (φ(CTran) complementOf(φ(CPlace)))));

- A property identifier $\varphi(\text{arc}_{ptot})$ is a subproperty of φ(arc), similarly for $\varphi(\text{arc}_{ttop})$. Then creating the property axioms:

  SubPropertyOf ($\varphi(\text{arc}_{ptot})$ φ(arc));

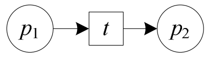
  SubPropertyOf ($\varphi(\text{arc}_{ttop})$ φ(arc));

- A class identifier φ(CPlace) is a subconcept of φ(CNode), similarly for φ(CTran). Then creating the class axioms:

  SubClassOf (φ(CPlace) φ(CNode));

  SubClassOf (φ(CTran) φ(CNode));

- A substitution for a transition *t*, such as



  where the labels on arrows associated with the places and the transition are omitted for the sake of simplicity, and the following several substitutions are also similar. Then creating the axioms:

  Class ($\varphi(p_1)$ partial restriction ($\varphi(f_{p1tot})$ allValuesFrom ($\varphi(t)$) cardinality(x)));

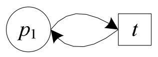
  Class ($\varphi(t)$ partial restriction ($\varphi(f_{ttop2})$ allValuesFrom ($\varphi(p_2)$) cardinality(x)));

  ObjectProperty ($\varphi(f_{p1tot})$ domain ($\varphi(p_1)$) range ($\varphi(t)$));

  ObjectProperty ($\varphi(f_{ttop2})$ domain ($\varphi(t)$) range ($\varphi(p_2)$));

  where $\varphi(p_1)$, $\varphi(p_2)$, $\varphi(t)$, $\varphi(f_{p1tot})$ and $\varphi(f_{ttop2})$ are created in step (1) above, and x is the number of tokens restricted by an expression on arrow as mentioned in Section 4.

- A substitution for a transition *t*, such as



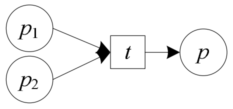
  Then creating the axioms:

  Class ($\varphi(p_1)$ partial restriction ($\varphi(f_{p1tot})$ allValuesFrom ($\varphi(t)$) cardinality(x)));

  Class ($\varphi(t)$ partial restriction ($\varphi(f_{ttop1})$ allValuesFrom ($\varphi(p_1)$) cardinality(x)));

  ObjectProperty ($\varphi(f_{p1tot})$ domain ($\varphi(p_1)$) range ($\varphi(t)$) inverseOf($\varphi(f_{ttop1})$));

- A conflict-free substitution for a transition *t*:



  Then creating the axioms:

  Class ($\varphi(p_1)$ partial restriction ($\varphi(f_{p1tot})$ allValuesFrom ($\varphi(t)$) cardinality(x)));

  Class ($\varphi(p_2)$ partial restriction ($\varphi(f_{p2tot})$ allValuesFrom ($\varphi(t)$) cardinality(x)));

Class ($\varphi(t)$ partial restriction ($\varphi(f_{\mathrm{ttop}})$ allValuesFrom ($\varphi(p)$) cardinality(x)));

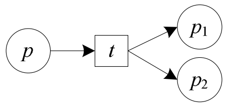
SubClassOf (intersectionOf ($\varphi(p_1)$ $\varphi(p_2)$) $\varphi(p)$);

ObjectProperty ($\varphi(f_{\mathrm{p1tot}})$ domain ($\varphi(p_1)$) range ($\varphi(t)$));

ObjectProperty ($\varphi(f_{\mathrm{p2tot}})$ domain ($\varphi(p_2)$) range ($\varphi(t)$));

ObjectProperty ($\varphi(f_{\mathrm{ttop}})$ domain ($\varphi(t)$) range ($\varphi(p)$));

- A conflict-free substitution for a transition t:



Then creating the axioms:

Class ($\varphi(p)$ partial restriction ($\varphi(f_{\mathrm{ptot}})$ allValuesFrom ($\varphi(t)$) cardinality(x)));

Class ($\varphi(t)$ partial restriction ($\varphi(f_{\mathrm{ttop1}})$ allValuesFrom ($\varphi(p_1)$) cardinality(x)) restriction ($\varphi(f_{\mathrm{ttop2}})$ allValuesFrom ($\varphi(p_2)$) cardinality(x)));

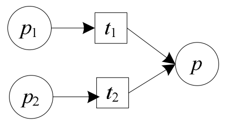
Class ($\varphi(p)$ partial $\varphi(p_1)$ $\varphi(p_2)$);

ObjectProperty ($\varphi(f_{\mathrm{ptot}})$ domain ($\varphi(p)$) range ($\varphi(t)$));

ObjectProperty ($\varphi(f_{\mathrm{ttop1}})$ domain ($\varphi(t)$) range ($\varphi(p_1)$));

ObjectProperty ($\varphi(f_{\mathrm{ttop2}})$ domain ($\varphi(t)$) range ($\varphi(p_2)$));

- A conflict-free substitution for transitions t_1, t_2:



Then creating the axioms:

Class ($\varphi(p_1)$ partial restriction ($\varphi(f_{\mathrm{p1tot1}})$ allValuesFrom ($\varphi(t_1)$) cardinality(x)));

Class ($\varphi(p_2)$ partial restriction ($\varphi(f_{\mathrm{p2tot2}})$ allValuesFrom ($\varphi(t_2)$) cardinality(x)));

Class ($\varphi(t_1)$ partial restriction ($\varphi(f_{\mathrm{t1top}})$ allValuesFrom ($\varphi(p)$) cardinality(x)));

Class ($\varphi(t_2)$ partial restriction ($\varphi(f_{\mathrm{t2top}})$ allValuesFrom ($\varphi(p)$) cardinality(x)));

SubClassOf (unionOf ($\varphi(p_1)$ $\varphi(p_2)$) $\varphi(p)$);

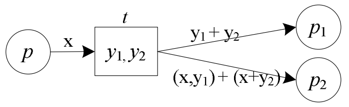
ObjectProperty ($\varphi(f_{\mathrm{p1tot1}})$ domain ($\varphi(p_1)$) range ($\varphi(t_1)$));

ObjectProperty ($\varphi(f_{\mathrm{p2tot2}})$ domain ($\varphi(p_2)$) range ($\varphi(t_2)$));

ObjectProperty ($\varphi(f_{\mathrm{t1top}})$ domain ($\varphi(t_1)$) range ($\varphi(p)$));

ObjectProperty ($\varphi(f_{\mathrm{t2top}})$ domain ($\varphi(t_2)$) range ($\varphi(p)$));

- The following discusses several special substitutions for a transition t (i.e., the labels on arrows are some related expressions), which may occur in a Petri net. A substitution for a transition t, such as:

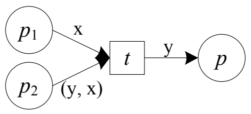


Since the transition t implicitly denotes that $\forall x.(p(x) \rightarrow \exists y_1, y_2\ (p_2(x, y_1) \wedge p_1(y_1) \wedge p_2(x, y_2) \wedge p_1(y_2)))$, and thus creating the axiom:

Class ($\varphi(p)$ partial restriction ($\varphi(p_2)$ someValuesFrom ($\varphi(p_1)$) minCardinality(2)));

here, to provide the intuition on the above translation process from the transition t to the ontology axioms, we represent the translated axioms with a simple and direct form, where class and property identifiers such as $\varphi(t)$, $\varphi(f_{\mathrm{ptot}})$, $\varphi(f_{\mathrm{ttop1}})$, $\varphi(f_{\mathrm{ttop2}})$, and the corresponding axioms are omitted, they can be created following the similar procedures mentioned in the previous translations. Intuitively, the above axiom is equivalent to the form $p \sqsubseteq \exists p_2.\mathrm{p}_1 \sqcap \geq 2p_2$.

- A substitution for transition t, such as:



Since the transition t implicitly denotes that $\exists \mathrm{y}.(p_2(y, x) \wedge p_1(x)) \rightarrow p(y)$, and thus creating the axioms:

SubClassOf (restriction ($\varphi(p_2)$ someValuesFrom ($\varphi(p_1)$)) $\varphi(p)$);

here, similarly for the above case, we represent the translated axioms with a simple and direct form (which is equivalent to the form $\exists p_2.p_1 \sqsubseteq p$), where class and property identifiers such as $\varphi(t)$, $\varphi(f_{\mathrm{p1tot}})$, $\varphi(f_{\mathrm{p2top}})$, $\varphi(f_{\mathrm{ttop}})$, and the corresponding axioms are also omitted.

- In addition, for each token tk_{i} inside in a place $p \in P$, creating the axiom:

  Individual ($\varphi(tk_{\mathrm{i}})$ type($\varphi(p)$));

  Moreover, for different tokens $tk_{\mathrm{i}},\ldots,tk_{\mathrm{n}}$, creating the axiom: DifferentIndividuals ($\varphi(tk_1)\ldots\varphi(tk_{\mathrm{n}})$);

- For $P \cap T = \varnothing$ as mentioned in Definition 2, i.e., class identifiers φ(CPlace) and φ(CTran) are disjoint, and thus it is necessary to create the class axiom:

  DisjointClasses (φ(CPlace) φ(CTran)).

In the following, we further illustrate the transformation in Definition 3 with Example 1.

Example 1 (a translation example). By applying Definition 3 to the Petri net *PN* in Fig. 1, we can obtain the corresponding OWL DL ontology $O = \varphi(PN)$ in Fig. 2.

Here, for the sake of simplicity, a transition t_0 denotes "x gets goods", t_1 "x with y", and t_2 "y gets money"; a place p_1 denotes "dealers ready", p_2 "dealer with money", p_3 "customers ready", and p_4 "customer with goods".

The OWL DL ontology $O = \varphi(PN) = (ID_0, Axiom_0)$, and some similar axioms are omitted here.
$ID_0 = CID_0 \cup IID_0 \cup PID_0$
$CID_0 = \{\varphi(p_1), \varphi(p_2), \varphi(p_3), \varphi(p_4), \varphi(t_0), \varphi(t_1), \varphi(t_2)\}$;
$IID_0 = \{\varphi(A), \varphi(C), \varphi(D), \varphi(B), \varphi(E)\}$;
$PID_0 = \{\varphi(f_{\text{p1tot1}}), \varphi(f_{\text{t1top2}}), \varphi(f_{\text{p2tot0}}), \varphi(f_{\text{t0top1}}), \varphi(f_{\text{p3tot1}}), \varphi(f_{\text{t1tot4}}), \varphi(f_{\text{p4tot2}}), \varphi(f_{\text{t2top3}})\}$;
$Axiom_0$ = {
Class ($\varphi(p_1)$ partial restriction ($\varphi(f_{\text{p1tot1}})$ allValuesFrom ($\varphi(t_1)$) cardinality(1)));
Class ($\varphi(t_1)$ partial restriction ($\varphi(f_{\text{t1top2}})$ allValuesFrom ($\varphi(p_2)$) cardinality(1)) restriction ($\varphi(f_{\text{t1top4}})$ allValuesFrom ($\varphi(p_4)$) cardinality(1)));
Individual ($\varphi(A)$ type($\varphi(p_1)$));
Class ($\varphi(p_2)$ partial restriction ($\varphi(f_{\text{p2tot0}})$ allValuesFrom ($\varphi(t_0)$) cardinality(1)));
Class ($\varphi(p_4)$ partial restriction ($\varphi(f_{\text{p4tot2}})$ allValuesFrom ($\varphi(t_2)$) cardinality(1)));
DisjointClasses ($\varphi(A)$ $\varphi(C)$ $\varphi(D)$ $\varphi(B)$ $\varphi(E)$); **…** }.

Fig. 2. The OWL DL ontology $O = \varphi(PN)$ representing the Petri net *PN* in Fig. 1.

Until now, the approach presented in this paper may be used to represent with OWL DL ontologies, so that Petri nets may be reused and shared. Also, representing Petri nets with ontologies enables sharing Petri nets on the Semantic Web.

V. Conclusions and Outlook

In this paper, we proposed an ontology approach for representing Petri nets. The formal definitions of OWL DL ontologies and Petri nets were presented. What's more, we proposed a formal approach for representing Petri nets with OWL DL ontologies by translating some key features of Petri nets into classes, properties and axioms of OWL DL ontologies, which will help readers represent Petri nets with ontologies more clearly. Representation of Petri nets with ontologies will facilitate the reusing and sharing of Petri nets, and also enables sharing Petri nets on the Semantic Web.

In the future, we intend to extend the approach to represent more complex Petri nets and develop an automated tool. Moreover, Petri nets have been used to represent much domain knowledge (e.g., spatio-temporal knowledge [17] and others [10], [12]), and thus it would be interest to extend our approach for representing these domain knowledge.

Acknowledgment

The work is supported by the *Ministry of Science and Technology, China, Chinese-Croatian Scientific and Technological Cooperation* "Fuzzy Spatio-temporal Knowledge Representation and Reasoning in the Semantic Web" and by the *National Natural Science Foundation of China* (61073139), and in part by the *Program for New Century Excellent Talents in University* (NCET-05-0288).

References

[1] G. Antoniou, F. vanHarmelen, A Semantic Web Primer. MIT Press, Cambridge, MA, 2004.
[2] G. Berthelot, J. Vautherin, et al., "A syntax for the description of Petri nets", Petri Net Newsletter 29, 1988, pp. 4-15.
[3] T. Berners-Lee, J. Hendler, O. Lassila, "The Semantic Web", Scientific American 284(5), 2001, pp. 34-43.
[4] E. Breton, J. Bezivin, "Towards an understanding of model executability", Proceedings of the International Conference on Formal Ontology in Information Systems, 2001, pp. 70-80.
[5] A. Brogi, S. Corfini, S. Iardella, "From OWL-S Descriptions to Petri Nets", Proc. of ICSOC'07 Workshops, 2007, pp. 427-438.
[6] J. Billington, et al., "The Petri Net Markup Language: Concepts, Technology, and Tools", Proc. of 24th Int. Conf. on Applications and Theory of Petri Nets, 2003, pp. 483-505.
[7] D. Gasevic, V. Devedzic, "Interoperable Petri net models via ontology", International Journal of Web Engineering and Technology 3(4), 2007, pp. 374-396.
[8] I. Horrocks, P. F. Patel-Schneider, et al., "From SHIQ and RDF to OWL: The Making of a Web Ontology Language", J. Web Semantics 1(1), 2003.
[9] S.F. Lu, C.F. Sun, X.J. Ma, "Using E-Connection and Description Logic for Formalizing and Analyzing High-Level Petri Net", Ninth Int. Symposium on Symbolic and Numeric Algorithms for Scientific Computing, 2007, pp. 514-517.
[10] W.Y. Liu, "Using IDEF0/Petri net for Ontology based task knowledge analysis: the case of emergency response for debris flow", System Sciences, HICSS Volume 4, Issue, 04-07, 2006.
[11] T. Murata, "Petri Nets: Properties, analysis and applications", Proc. of the IEEE, 77(4), 1989, pp. 541-580.
[12] M. Mäkelä, "Introduction to maria and high-level petri nets", Laboratory for Theoretical Computer Science Helsinki University of Technology, 2001.
[13] OWL: Ontology Web Language, http://www.w3.org/2004/OWL/
[14] G. Peterka, T. Murata, "Proof procedure and answer extraction in Petri net model of logic programs", IEEE Trans. Software Eng., 1989, pp. 209-217.
[15] J. Recker, M. Indulska, "An ontology-based evaluation of process modeling with Petri nets", Journal of Interoperability in Business Information Systems, Vol. 2, 2007, pp. 45-64.
[16] W. Reisig, A primer in Petri net design. Berlin: Springer Verlag, 1992.
[17] S. Ribarić, T. Hrka´c, "A knowledge representation and reasoning based on Petri nets with spatio-temporal tokens", in International Conference on Computer As a Tool, EUROCON 2007, 2007, pp. 793-800.
[18] P. Soffer, M. Kaner, Y. Wand, "Assigning Ontology-Based Semantics to Process Models: the Case of Petri Nets", In: Bellahsène, Z. (eds.) CAiSE 2008. 2008, pp. 16-31.
[19] E. Sirin, B. Parsia, B. Cuenca Grau, A. Kalyanpur, Y. Katz, "Pellet: a practical OWL-DL reasoner", Journal of Web Semantics 5, 2007, pp. 51-53.
[20] T. Takai-Igarashi, "Ontology based standardization of Petri net modeling for signaling pathways", In Silico Biol., 2005.
[21] J.C. Vidal, M. Lama, A. Bugar´ın, "A High-level Petri Net Ontology Compatible with PNML", Petri Net Newsletter 71, 2006, pp. 11-23.
[22] H.C. Yen, "Introduction to Petri net theory", Recent Advances in Formal Languages and Applications, 25, 2006, pp. 343-373.
[23] G.S. Zhang, F.Q. Meng, C.J Jiang, J. Pang, "Using Petri Net to Reason with Rule and OWL", Proc. of the sixth IEEE Int. Conf. on Computer and Information Technology (CIT'06), 2006.
[24] I. Zevedei-Oancea, S. Schuster, "Topological analysis of metabolic networks based on Petri net theory", In: Silico Biol. 3, 2003.
[25] M. Zhou, "Modeling, Simulation, and Control of Flexible Manufacturing Systems: A Petri Net Approach", World Scientific Publishing, Hong Kong, 1999.