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 Ribaric  
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 $O = (ID_0, Axioms_0)$, 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 “$x$ gets $y$” are labeled with two variables $x$ and $y$, and for this transition to occur, $y$ must be replaced by a customer $y$ and $x$ by a dealer. Similarly, “$x$ gets goods” for the dealers, and “$y$ gets money” for the customers.

Based on the introduction above, in briefly, a Petri net consists of:
• places, represented as circles ($\bigcirc$);
• transitions, represented as boxes ($\square$);
• arrows from places to transitions ($\bigcirc$$\rightarrow$$\square$);
• arrows from transitions to places ($\square$$\rightarrow$$\bigcirc$);
• 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:
- $P = \{p_1, p_2, \ldots, p_n\}$ is a finite set of places;
- $T = \{t_1, t_2, \ldots, t_m\}$ is a finite set of transitions, $P \cup T \neq \emptyset$, and $P \cap T = \emptyset$;
- $F \subseteq P \times T \cup T \times P$, 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...
Definition 3 (representation of Petri nets). Let $PN = (P, T, F, M_0)$ be a Petri net. The OWL DL ontology $O = \varphi(PN) = (ID_0, \text{Axiom}_0)$ can be defined by transformation function $\varphi$ 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 $f_{p\rightarrow t} \in F$ is mapped into a property identifier $\varphi(f_{p\rightarrow t}) \in PID_0$;
- Each arc from a transition to a place $f_{t\rightarrow p} \in F$ is mapped into a property identifier $\varphi(f_{t\rightarrow p}) \in PID_0$;
- Each token inside in a place $t_{k_i} \in M_0$ is mapped into an individual identifier $\varphi(t_{k_i}) \in IID_0$;
- A class identifier $\varphi(\text{CPlace}) \in CID_0$ denotes all the places in $PN$;
- A class identifier $\varphi(\text{CTran}) \in CID_0$ denotes all the transitions in $PN$;
- A class identifier $\varphi(\text{CNode}) \in CID_0$ denotes all the places and transitions in $PN$;
- A property identifier $\varphi(\text{arc}_{p\rightarrow t}) \in PID_0$ denotes all the arcs from places to transitions;
- A property identifier $\varphi(\text{arc}_{t\rightarrow p}) \in PID_0$ denotes all the arcs from transitions to places;
- A property identifier $\varphi(\text{arc}) \in PID_0$ 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 $\text{Axiom}_0$ 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 $\varphi(\text{arc}_{p\rightarrow t})$ and $\varphi(\text{arc}_{t\rightarrow p})$. Then creating the property axioms:
  - ObjectProperty ($\varphi(\text{arc}_{p\rightarrow t})$) domain ($\varphi(\text{CPlace})$), range ($\varphi(\text{CTran})$);
  - ObjectProperty ($\varphi(\text{arc}_{t\rightarrow p})$) domain ($\varphi(\text{CTran})$), range ($\varphi(\text{CPlace})$);
- A place can not be a transition simultaneously in $PN$, i.e., a class identifier $\varphi(\text{CNode})$ is either a class identifier $\varphi(\text{CPlace})$ or a $\varphi(\text{CTran})$. Then creating the class axiom:
  - Class ($\varphi(\text{CNode})$) partial unionOf (intersectionOf ( $\varphi(\text{CPlace})$ complementOf ($\varphi(\text{CTran})$)) intersectionOf ( $\varphi(\text{CTran})$ complementOf ($\varphi(\text{CPlace})$)));

- A property identifier $\varphi(\text{arc}_{p\rightarrow t})$ is a subproperty of $\varphi(\text{arc})$, similarly for $\varphi(\text{arc}_{t\rightarrow p})$. Then creating the property axioms:
  - SubPropertyOf ($\varphi(\text{arc}_{p\rightarrow t})$ $\varphi(\text{arc})$);
  - SubPropertyOf ($\varphi(\text{arc}_{t\rightarrow p})$ $\varphi(\text{arc})$);

- A class identifier $\varphi(\text{CPlace})$ is a subconcept of $\varphi(\text{CNode})$, similarly for $\varphi(\text{CTran})$. Then creating the class axioms:
  - SubClassOf ($\varphi(\text{CPlace})$ $\varphi(\text{CNode})$);
  - SubClassOf ($\varphi(\text{CTran})$ $\varphi(\text{CNode})$);

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

\[ p_1 \rightarrow t \rightarrow p_2 \]

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 ($\varphi(f_{p_1\rightarrow t})$) allValuesFrom ($\varphi(p_1)$) cardinality($x$));
- Class ($\varphi(t)$) partial restriction ($\varphi(f_{t\rightarrow p})$) allValuesFrom ($\varphi(p_2)$) cardinality($x$));
- ObjectProperty ($\varphi(f_{p_1\rightarrow t})$) domain ($\varphi(p_1)$) range ($\varphi(t)$);
- ObjectProperty ($\varphi(f_{t\rightarrow p})$) domain ($\varphi(t)$) range ($\varphi(p_2)$);

where $\varphi(p_1), \varphi(p_2), \varphi(t), \varphi(f_{p_1\rightarrow t})$ and $\varphi(f_{t\rightarrow p})$ 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

\[ p_1 \rightarrow t \rightarrow p_2 \]

Then creating the axioms:

- Class ($\varphi(p_1)$) partial restriction ($\varphi(f_{p_1\rightarrow t})$) allValuesFrom ($\varphi(p_1)$) cardinality($x$));
- Class ($\varphi(t)$) partial restriction ($\varphi(f_{t\rightarrow p})$) allValuesFrom ($\varphi(p_2)$) cardinality($x$));
- ObjectProperty ($\varphi(f_{p_1\rightarrow t})$) domain ($\varphi(p_1)$) range ($\varphi(t)$) inverseOf ($\varphi(f_{t\rightarrow p})$));

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

\[ p_1 \rightarrow t \rightarrow p_2 \]

Then creating the axioms:

- Class ($\varphi(p_1)$) partial restriction ($\varphi(f_{p_1\rightarrow t})$) allValuesFrom ($\varphi(p_1)$) cardinality($x$));
- Class ($\varphi(p_2)$) partial restriction ($\varphi(f_{p_2\rightarrow t})$) allValuesFrom ($\varphi(p_2)$) cardinality($x$));
Class (ϕ(t) partial restriction (ϕ(ftop) allValuesFrom (ϕ(p)) cardinality(x)));
SubClassOf (intersectionOf (ϕ(p1) ϕ(p2)) ϕ(p));
ObjectProperty (ϕ(f1topa) domain (ϕ(p1)) range (ϕ(t)));
ObjectProperty (ϕ(f2topa) domain (ϕ(p2)) range (ϕ(p)));
ObjectProperty (ϕ(ftop) domain (ϕ(t)) range (ϕ(p)));

- A conflict-free substitution for a transition t:

\[ \begin{array}{c}
\text{p} \\
\rightarrow \\
\text{t} \\
\downarrow \\
\text{p}_1 \\
\rightarrow \\
\text{p}_2
\end{array} \]

Then creating the axioms:
Class (ϕ(p) partial restriction (ϕ(ftop) allValuesFrom (ϕ(t)) cardinality(x)));
Class (ϕ(t) partial restriction (ϕ(ftop1) allValuesFrom (ϕ(p1)) cardinality(x)) restriction (ϕ(ftop2) allValuesFrom (ϕ(p2)) cardinality(x)));
Class (ϕ(p)) partial (ϕ(p1), ϕ(p2));
ObjectProperty (ϕ(ftopa) domain (ϕ(p)) range (ϕ(t)));
ObjectProperty (ϕ(ftopa) domain (ϕ(p)) range (ϕ(p1)));
ObjectProperty (ϕ(ftopa) domain (ϕ(p)) range (ϕ(p2)));

- A conflict-free substitution for transitions t1, t2:

\[ \begin{array}{c}
\text{p}_1 \\
\rightarrow \\
\text{t}_1 \\
\downarrow \\
\text{p}_2 \\
\rightarrow \\
\text{t}_2
\end{array} \]

Then creating the axioms:
Class (ϕ(p1)) partial restriction (ϕ(f1top1) allValuesFrom (ϕ(t1)) cardinality(x));
Class (ϕ(p2)) partial restriction (ϕ(f2top2) allValuesFrom (ϕ(t2)) cardinality(x));
Class (ϕ(t1)) partial restriction (ϕ(f1top1) allValuesFrom (ϕ(p1)) cardinality(x));
Class (ϕ(t2)) partial restriction (ϕ(f2top2) allValuesFrom (ϕ(p2)) cardinality(x));
SubClassOf (unionOf (ϕ(p1) ϕ(p2)) ϕ(p));
ObjectProperty (ϕ(f1topa) domain (ϕ(p1)) range (ϕ(t1)));
ObjectProperty (ϕ(f2topa) domain (ϕ(p2)) range (ϕ(t2)));
ObjectProperty (ϕ(f1topa) domain (ϕ(p1)) range (ϕ(p2)));
ObjectProperty (ϕ(f2topa) domain (ϕ(p2)) range (ϕ(p1)));

- 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:

\[ \text{p} \xrightarrow{t} \text{p}_1 \]

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

Class (ϕ(p)) partial restriction (ϕ(p2) someValuesFrom (ϕ(p1)) minCardinality(2));

- A substitution for transition t, such as:

\[ \text{p}_1 \xrightarrow{t} \text{p} \]

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

SubClassOf (restriction (ϕ(p2) someValuesFrom (ϕ(p1))) ϕ(p));

- In addition, for each token tk inside in a place p ∈ P, creating the axiom:

Individual (ϕ(tk) type(ϕ(p)));

Moreover, for different tokens tk1,...,tkn creating the axiom: DifferentIndividuals (ϕ(tk1) ϕ(tkn));

- For \( P \cap T = \emptyset \) 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 \( O = \text{ϕ(PN)} \) 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”.

\[ \begin{array}{c}
\text{p} \\
\rightarrow \\
\text{t}_0 \\
\downarrow \\
\text{p}_1 \\
\rightarrow \\
\text{p}_2 \\
\rightarrow \\
\text{p}_3 \\
\rightarrow \\
\text{p}_4
\end{array} \]
The OWL DL ontology \( O = \phi(PN) = (ID_0, Axiom_0) \), and some similar axioms are omitted here.

\[
\begin{align*}
ID_0 = & \text{CID}_0 \cup ID_0 \cup PID_0 \\
CID_0 = & \{ (\phi(p_1), \phi(p_2), \phi(p_3), \phi(q_1), \phi(q_2), \phi(t_1)) \}; \\
ID_0 = & \{ (\phi(A), \phi(C), \phi(D), \phi(B), \phi(E)) \}; \\
PID_0 = & \{ (\phi(f_{t1top1}), \phi(f_{f2top0}), \phi(f_{f3top1}), \phi(f_{f4tot2}), \phi(f_{f5tot2})) \}; \\
Axiom_0 = & \{ \\
\text{Class} (\phi(p_1)) \text{ partial restriction} (\phi(f_{f3tot2}) \text{ allValuesFrom} (\phi(t_1)) \text{ cardinality}(1))); \\
\text{Class} (\phi(t_1)) \text{ partial restriction} (\phi(f_{f1top2}) \text{ allValuesFrom} (\phi(p_2)) \text{ cardinality}(1)) \text{ restriction} (\phi(f_{f1top2}) \text{ allValuesFrom} (\phi(p_3)) \text{ cardinality}(1))); \\
\text{Individual} (\phi(A) \text{ type}(\phi(p_1))); \\
\text{Class} (\phi(p_2)) \text{ partial restriction} (\phi(f_{f2top0}) \text{ allValuesFrom} (\phi(t_2)) \text{ cardinality}(1))); \\
\text{Class} (\phi(p_3)) \text{ partial restriction} (\phi(f_{f4tot2}) \text{ allValuesFrom} (\phi(t_2)) \text{ cardinality}(1))); \\
\text{DisjointClasses} (\phi(A) \phi(C) \phi(D) \phi(B) \phi(E)); \ldots \}.
\end{align*}
\]

Fig. 2. The OWL DL ontology \( O = \phi(PN) \) 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 interesting 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


