Representation of RDF-oriented Composition with OWL DL Ontology

Thi Hoa Hue Nguyen, Nhan Le Thanh

Representation of RDF-oriented Composition with OWL DL Ontology

Thi-Hoa-Hue Nguyen
WIMMICS - The I3S laboratory - CNRS
University of Nice Sophia Antipolis
Sophia Antipolis, France
Email: nguyenth@i3s.unice.fr

Nhan Le-Thanh
WIMMICS - The I3S laboratory - CNRS
University of Nice Sophia Antipolis
Sophia Antipolis, France
Email: Nhan.LE-THANH@unice.fr

***Abstract*—This paper introduces a solution for representing RDF-oriented compositions with OWL DL ontologies. Firstly, we present an overview of RDF-oriented Composition Definition Language (RDFCDL), which is defined for creating/composing RDF manipulation operations. Secondly, we propose an approach for representing RDFNet with OWL DL ontology. We focus on translating some key components of the RDFCDL language into classes, properties and axioms of OWL DL ontology.**

***Index Terms*—RDF, Coloured Petri Net, RDFNet, OWL DL ontology, representation**

I. Introduction

In general, regulations in the industry play an important role in ensuring the quality of a product [1]. They are defined by regulatory bodies, informational or cultural standards. It would be clearly useful to check whether all existing and/or potential systems satisfy these recommended regulations.

We consider here the case of ontologies expressed in RDF/S, OWL which represent a project and the regulations formalized as SPARQL[1] queries. Our work aims to support both non-expert and expert end-users to manipulate RDF data for checking the conformance of a project against a set of regulations. We define RDF-oriented Composition Definition Language (RDFCDL), which is an important part in our overall approach, based on Coloured Petri Nets.

Ontologies with their components (e.g., classes, attributes, relations, restrictions, axioms,...) provide machine-readable definitions of concepts, and therefore can facilitate interoperability by aligning different terms used in different workflow models. They are a means of representing semantic knowledge.

Representing RDF-oriented Compositions using ontologies can express semantic description of composition concepts and their relationship. Consequently, this will contribute with a common semantics to the improvement of communication among communities. We propose a formal approach for representing RDF-oriented Composition with OWL DL ontology in order to share and reuse the processes of manipulating RDF data. We focus on translating some key components of the RDFCDL language into classes, properties and axioms of OWL DL ontology.

The rest of this paper is structured as follows: In Section 2, related work is given. Section 3 describes an overview of RDFCDL in order to manipulate RDF data. In Section 4, we focus on the approach for representing RDFNet with ontology. Finally, Section 5 shows conclusions and ongoing works.

[1]http://www.w3.org/TR/rdf-sparql-query/

II. Related work

Coloured Petri Nets (CPNs) [5] have been developed to being a full-fledged language for design, specification, simulation, validation and implementation of large software systems. The CPN language is supported by CPN tools [10]. CPNs are thus a well-proven language which is suitable for modelling of workflows or work processes [6].

With regard to the conformity-checking problem, various efforts have been made to check the conformance of a product according to defined rules, e.g., [1], [2], [8]. However, to the best of our knowledge, existing approaches/techniques do not allow end-users to create/compose conformity checking processes. In this paper, based on CPNs, we propose the RDFCFL language to support end-users to manipulate RDF data.

Up to now, the combination of Petri Nets/high-level Petri Nets and ontologies has been studied in some research works [3], [4], [7], [9] to support (semi-)automatic system collaboration, provide machine-readable definitions of concepts and interpretable format. In [7], we presented an ontology approach for representing CPNs restricted to the workflow domain. Since the RDFCDL language is based on CPNs, we here rely on our work in [7] to propose an approach for representing RDFNet with OWL DL ontology.

III. RDF-oriented Composition Definition Language

RDF-oriented Composition Definition Language (RDFCDL) allows end-users to create/compose RDF-oriented manipulation operations using node functions (NFs), which are system-defined functions. There are three main parts in RDFCDL as follows:

- *The Inputs* are ontologies expressed in RDF/S or OWL. They are a representation of a project, for example a construction project.
- *The NFs* and *the compositions* which compose the RDFCDL core. They are defined as CPNs.

- *The Outputs* are stored as RDF annotations. They are RDF annotations of manipulating processes. They describe semantically the result of manipulating processes.

The syntax and semantics of the RDFCDL core are based on the grammar RDFNet (RDF-oriented Composition Grammar Net) defined using CPNs. Since RDFNet is based on CPNs, it inherits the features and operational semantics from CPNs, e.g., the firing rule.

Definition 1 (RDFNet (RDF-oriented Composition Grammar Net)). RDFNet represents the grammar of the RDFCDL in compliance with CPNs. It is defined as a 9-tuple: $RDFNet = (\sum, P, T, A, F, C, G, E, I)$ where:

- $\sum$ is a finite set of non-empty types available in the RDFCDL, called colour sets:
  $\sum = \{Char, String, Integer, Double, Boolean, Date, RDFNode\}$
  where $Char$, $String$, $Integer$, $Double$, $Boolean$, $Date$ are standard types and $RDFNode$ is a super-type (see Definition 2).
- $P = P_{in} \cup P_{out}$ is a finite set of places. P_{in} and P_{out} denote the input and output states of the functions used in RDFCDL respectively.
  The number of tokens in place p: $\forall p \in P, [w(p) = 1]$.
- T is a finite set of transitions. The behavior of the functions and operators in RDFCDL are represented by transitions.
- $A \subseteq (P \times T) \cup (T \times P)$ is a set of directed arcs connecting input places to transitions or transitions to output places. $\forall a \in A : a.p$ and $a.t$ stand for the place and transition linked by a, respectively.
- F[2] is a set of operators/functions available in the libraries.
- $C : P \rightarrow \sum$ is a colour function.
  Each place has only one type from $\sum$:
  $$\forall p \in P : [|C(p) = 1|]$$
- $G : T \rightarrow F$ is a guard function associating an operation to a transition.
- $E : A \rightarrow Expr$ is an arc expression function. It is defined from A into $Expr$ such that:
  $$\forall a \in A : [Type(E(a)) = C(a.p) \wedge Type(Var(E(a)))]$$
- $I : P \rightarrow Init$ is an initialization function associating initial values to places. It is defined from P into $Init$ such that: $\forall p \in P : [Type(I(p)) = C(p)]$.

We introduce the following definition of $RDFNode$ that designates an RDF[3] (Resource Description Framework) component.

Definition 2 (RDFNode)**.** $RDFNode$ contains three subtypes that are $RDFNode : URI$, $RDFNode : Literal$ and $RDFNode : Blank$, where:

[2]Since F is added to the initial CPN definition and has no effect on the CPN's functionality, in the rest of the definitions based on CPNs it is bypassed.

[3]http://www.w3.org/TR/2004/REC-rdf-concepts-20040210/

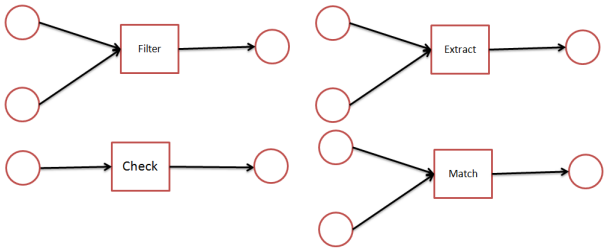


Fig. 1. Some sample functions defined in RDFCDL

- $RDFNode : URI$ defines the RDF URI reference type.
- $RDFNode : Literal$ defines the RDF literal type.
- $RDFNode : Blank$ defines the RDF blank node type.

The NFs and the compositions are defined based on CPNs. Consequently, the inputs and outputs of NFs are defined as places and drawn as ellipses. Note that in this study, a function can have multiple inputs but only one output. Each place has a single colour defining its type. A transition, which is drawn as a rectangle, represents the operation of the function. It operates on the inputs and sends the result to the output. The input and output places are linked to transitions via directed arcs drawn as arrows. A directed arc connects a place with a transition or vice versa. Several sample functions are shown in Figure 1.

Definition 3 (NF (Node Function))**.** NF is a system-defined function based on CPNs. It describes an operation and is defined as:
$NF = (\sum, P, T, A, C, G, E, I)$ where:

- $\sum$ is a finite set of non-empty types available in the NF, where: $\sum \subseteq RDFNet.\sum$
- $P = P_{in} \cup P_{out}$ is a finite set of places defining the input and output states of the NF.
  P_{in} and P_{out} are the set of input and output places respectively where: $P = P_{in} \cup P_{out}$; $P_{in} \cap P_{out} = \emptyset$; $P_{in} = \{p_{in1}, p_{in2}, \ldots, p_{inN}\}$; $P_{out} = \{p_{out}\}$.
- $T = \{t\}$ is a finite set of transitions denoting the behavior of the NF. Transition t contains the operation to be performed.
- $A \subseteq (P \times \{t\}) \cup (\{t\} \times P)$ is a set of directed arcs connecting input places to transition t or transition t to output places.
- $C : P \rightarrow \sum$ is a colour function associating a type to each place. It is defined from P into $\sum$.
- $G : \{t\} \rightarrow F$ is a guard function associating an operation to transition t. It is defined from $T = \{t\}$ into F where: $Type(G(t)) = Type(Var(G(t))) \wedge C(p_{out}) \subseteq \sum$
- $E : A \rightarrow Expr$ is an arc expression function where $Expr$ is a set of expressions. It is defined from A into $Expr$ where:
  $$\forall a \in A : E(a) = \begin{cases} M(a.p) & \text{if } a.p \in P_{in} \\ G(a.t) & \text{otherwise} \end{cases}$$
  $M(p)$ is the value of the token in p.
- $I : P_{in} \rightarrow Init$ is an initialization function associating initial values to input places.

A composition is defined by a mapping between the outputs and the inputs of NFs. It is expressed by a combination of graphical functions via operators. We use a suitable operator having dashed arcs for one link between two functions (Figure 2).

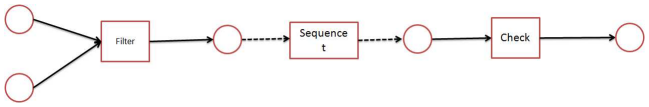


Fig. 2. Graphical composition in RDFCDL

For the purpose of manipulating RDF data, a composition might be sequential, parallel or conditional. We use the operators including Sequence, And-split, And-join, Xor-split and Xor-join (Figure 3) to create the compositions. These operators are defined based on CPNs, which are compliant to RDFNet.

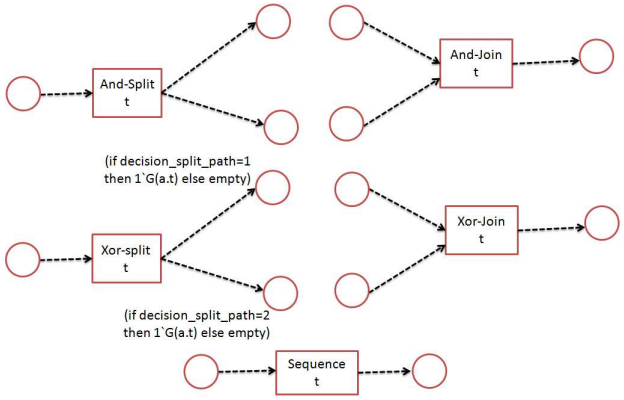


Fig. 3. Operators are defined in RDFCDL

IV. Representation of RDFNet with OWL DL ontology

In this Section, we propose an approach for representing RDFNet with OWL DL ontology. We translate some key features of RDFNet into classes, properties and axioms of OWL DL ontology.

OWL DL, which stands for OWL Description Logic, is equivalent to Description Logic $\mathcal{SHOIN}(\mathcal{D})$. OWL DL supports all OWL language constructs with restrictions (e.g., type separation), provides maximum expressiveness while always keeping computational completeness and decidability. Therefore, we choose OWL DL language to represent RDFNet. For more details on OWL DL, please refer to [11].

A formal definition of OWL DL ontologies is given in Definition 4.

Definition 4 (OWL DL ontology)**.** An OWL DL ontology is a couple $O = (ID_0, Axiom_0)$, where:

1) $ID_0 = CLID_0 \cup INID_0 \cup DRID_0 \cup PODID_0$ is an OWL DL identifier set including five subsets:
   - $CLID_0$ is a subset of class identifiers.
   - $INID_0$ is a subset of individual identifiers.
   - $DRID_0$ is a subset of data range identifiers which are predefined XML datatypes.
   - $PODID_0$ is a subset of property identifiers containing object property identifiers ($OPIDs$) and datatype property identifiers ($DPIDs$).
2) $Axiom_0 = CLPA_0 \cup INA_0$ is a finite set of OWL DL axioms including two subsets:
   - $CLPA_0$ is a subset of class/property axioms which is used to represent the ontology structure.
   - INA_0 is a subset of individual axioms which is used to represent the ontology instances.

Table I shows the mapping between RDFNet and ontology. We then present the definition of representation of RDFNet with OWL DL ontology in Definition 5.

TABLE I
Mapping between RDFNet and ontology

RDFNet	Ontology
Colour Sets	Classes
Places	Classes
Instance Tokens	Individuals
Transitions	Classes
Arcs	Properties
Guard funtions	Individuals
Arc expression functions	Individuals

Definition 5 (Representation of RDFNet)**.** Let an $RDFNet = (\sum, P, T, A, F, C, G, E, I)$ be a CPN. Using a transformation function φ to define the OWL DL ontology $O = \varphi(RDFNet) = (ID_0, Axiom_0)$ as follows:

1) The **identifier set** ID_0 of $\varphi(RDFNet)$ consists of following elements:
   - For each colour set, $\varsigma \in \sum$, map ς into a class identifier $\varphi(\varsigma) \in CLID_0$;
   - For each place, $p_i \in P$, map p_i into a class identifier $\varphi(p_i) \in CLID_0$;
   - For each transition, $t_i \in T$, map t_i into a class identifier $\varphi(t_i) \in CLID_0$;
   - For each arc from a place to a transition, $a_{in} \in A$, map a_{in} into a property identifier $\varphi(a_{in}) \in PODID_0$;
   - For each arc from a transition to a place, $a_{out} \in A$, map a_{out} into a property identifier $\varphi(a_{out}) \in PODID_0$;
   - For each token in a place, $tk_i \in M_0$[4], map tk_i into an individual identifier $\varphi(tk_i) \in INID_0$;
   - For each guard, $g \in F$, map g into an individual identifier $\varphi(g) \in INID_0$
   - For each arc expression, $e \in Expr$, map e into an individual identifier $\varphi(e) \in INID_0$;
   - A class identifier $\varphi(CCSet) \in CLID_0$ refers to all the colour sets in RDFNet;

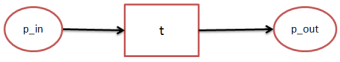
[4] M_0 denotes the set of initial markings of P and $I(p)$ is the initial marking of p where $M_0(p) = I(p)$

- A class identifier $\varphi(CPlace) \in CLID_0$ refers to all the places in RDFNet;
- A class identifier $\varphi(CTran) \in CLID_0$ refers to all the transitions in RDFNet;
- A property identifier $\varphi(arcIn) \in PODID_0$ refers to all the arcs from places to transitions;
- A property identifier $\varphi(arcOut) \in PODID_0$ refers to all the arcs from transitions to places;
- Two OWL class identifiers, $owl : Thing \in CLID_0$ and $owl : Nothing \in CLID_0$ are predefined. The class extension of $owl : Thing$ is employed to denote the set of all individuals. The class extension of $owl : Nothing$ is the empty set.

2) The **OWL DL axiom set** $Axiom_0$ of $\varphi(RDFNet)$ contains the following subsets:
   - a set $CLPA_0$ of class/property axioms;
   - a subset INA_0 of property identifiers.

We now indicate some elements of the set $Axiom_0$.

- For each guard $G(t)$ in transition t having the output place p_{out}, we create the following axiom that corresponds to the form $\varphi(G(t)) \in \varphi(Var(G(t))) \wedge \varphi(p_{out})$ as follows:
  $Individual(\varphi(G(t))\ type(\varphi(Var(G(t))))\ type(\varphi(p_{out})))$;
- Each place $p \in P$ in RDFNet can contain one token tk. We create the following axiom that corresponds to the form $\varphi(tk) \in \varphi(p)$ as follows:
  $Individual(\varphi(tk)\ type(\varphi(p)))$;
- For each arc expression $E(a)$, we create the following axiom that corresponds to the form
  $\varphi(E(a)) \in \varphi(a.p) \wedge \varphi(Var(E(a)))$
  as follows:
  $Individual(\varphi(E(a))\ type(\varphi(a.p))\ type(\varphi(Var(E(a)))))$;
- Consider a substitution for a transition t, such as:



For the sake of simplicity, we assume that the arc expressions on arcs associated with the places and the transition are omitted. We thus create the following axioms that correspond to the forms
$\varphi(p_{in}) \sqsubseteq \forall\varphi(a_{in}).\varphi(t) \sqcap = n\varphi(a_{in})$
$\varphi(t) \sqsubseteq \forall\varphi(a_{out}).\varphi(p_{out}) \sqcap = n\varphi(f_{tTop2})$
$\geqslant 1\ \varphi(a_{in}) \sqsubseteq \varphi(p_{in})$
$\top \sqsubseteq \forall\varphi(a_{in}).\varphi(t)$
$\geqslant 1\ \varphi(a_{out}) \sqsubseteq \varphi(t)$
$\top \sqsubseteq \forall\varphi(a_{out}).\varphi(p_{out})$
as follows:
$Class(\varphi(p_{in})\ partial\ restriction(\varphi(a_{in})$
$allValuesFrom(\varphi(t)\ cardinality(n)))$;
$Class(\varphi(t)\ partial\ restriction(a_{out})$
$allValuesFrom(\varphi(p_{out})\ cardinality(n)))$;
$ObjectProperty(\varphi(a_{in})$
$domain(\varphi(p_{in}))\ range(\varphi(t)))$;
$ObjectProperty(\varphi(a_{out})$
$domain(\varphi(t)\ range(\varphi(p_{out})))$;
where $\varphi(p_{in})$, $\varphi(p_{out})$, $\varphi(t)$, $\varphi(a_{in})$ and $\varphi(a_{out})$ are created in step (1) above, $n = w(p) = 1$ is the number of tokens in place p (Definition 1).

We have created some axioms in the set $Axiom_0$. Other axioms can be created in a same way, e.g., the axioms for a conflict-free substitution for a transition t that is similar to Filter function (Figure 1) or And-join operator (Figure 3).

V. Conclusions

In this paper, we have introduced an overview of the RDFCDL language to allow end-users to create/compose RDF manipulation operations. We based on CPNs to define RDFNet, which represents the grammar of the RDFCDL. On this basic definition, we presented some other related definitions constituting the RDFCDL core.

We have proposed an approach for representing RDFNet with OWL DL ontology, which aims at sharing and reusing the processes of manipulating RDF data not only in the Semantic Web, but also in workflow systems. Some key components of the RDFCDL language were translated into classes, properties and axioms of OWL DL ontology.

For validating the components of the RDFCDL language, our ongoing works focus on defining an internal data model. To execute these components, we then define a run-time environment, which relies on the CORESE[5] semantic engine that answers SPAQRL queries asked against an RDF/OWL knowledge base.

References

[1] Khalil Riad Bouzidi, Bruno Fies, Catherine Faron-Zucker, Alain Zarli and Nhan Le Thanh, *Semantic Web Approach to Ease Regulation Compliance Checking in Construction Industry*, Future Internet, 4, pp. 830-851, 2012.

[2] C. Eastman, J.M. Lee, Y.S. Jeong, J.K. Lee, *Automatic rule-based checking of building designs*, Automation in Construction, Volume 18, Issue 8, pp. 1011-1033, 2009.

[3] Dragan Gašević, Vladan Devedžić, *Reusing Petri Nets Through the Semantic Web*, The Semantic Web: Research and Applications, Lecture Notes in Computer Science Volume 3053, pp. 284-298, Greece, 2004.

[4] D. Gašević, V. Devedzic, *Interoperable Petri net models via ontology*, International Journal of Web Engineering and Technology 3(4), pp. 374-396, 2007.

[5] K. Jensen, *Coloured Petri Nets. Basic Concepts, Analysis Methods and Practical Use. Volume 1*, EATCS monographs on Theoretical Computer Science, Springer-Verlag, Berlin, 1997.

[6] Jens Bæk Jørgensen, Kristian Bisgaard Lassen, Wil M. P. van der Aalst, *From task descriptions via colored Petri nets towards an implementation of a new electronic patient record workflow system*, International Journal on Software Tools for Technology Transfer, Volume 10, Issue 1, pp. 15-28, January 2008.

[7] Thi-Hoa-Hue Nguyen, Nhan Le-Thanh, *Representation of Coloured Workflow Nets with OWL DL Ontoloty*, Second International workshop "Rencontres scientifiques UNS-UD" (RUNSUD2013), pp. 29-41, Vietnam, 2013.

[8] A. Yurchyshyna, *Modélisation Du Contrôle de Conformité en Construction: Une Approche Ontologique* (in French), Ph.D. Dissertation, University of Nice Sophia Antipolis, Sophia Antipolis, France, 2009.

[9] Fu Zhang, Z. M. Ma, Slobodan Ribarić, *Representation of Petri Net with OWL DL Ontology*, Eighth International Conference on Fuzzy Systems and Knowledge Discovery (FSKD), 2011.

[10] CPN Tools, http://cpntools.org/

[11] OWL Web Ontology Language Overview, http://www.w3.org/TR/owl-features/

[5] http://wimmics.inria.fr/corese