ABSTRACT

One of the basic problems in semantic business management concerns the validation of semantic business processes. This paper introduces the Unary RDF Annotated Petri net systems as a formalism to model and analyze business processes with RDF annotations to deal with semantic information from both, data and control flow points of view. In addition, it shows how to use the reachability graph in order to analyze some behavioral business process properties including semantic aspects.

Keywords

1. INTRODUCTION

One of the basic problems in semantic business process management concerns the validation of semantic business processes. Business Processes invoke (Web) services to satisfy the needs of clients. In order to describe a business process the preconditions, postconditions (or effects), inputs and outputs of each invoked service must be specified. Different works have studied the semantic validation of business processes with some kind of semantic annotations. For example, [2] adds effects (or postconditions) to the services, [4] adds preconditions and effects, while [5] adds preconditions, effects and also an ontological axiomation. However, none of these works add semantic annotations to the inputs and outputs of services so they cannot study the semantic propagation of data through processes: the semantic description of the outputs of a service depends on the semantics of the inputs, which in turn depends on the semantics of the previous services in the business process. In addition, they avoid loops in the control flow of the business processes they validate. In this paper, we propose a new formalism, the Unary RDF Annotated Petri net systems, that extends classical Petri nets with RDF annotations [1] to deal with semantic information from both, data and control flow points of view. We define its semantics in terms of enabling and firing of transitions. We also give an example that illustrates the usefulness and application of such formalism to analyze some behavioral business process properties including semantic aspects.

The paper is organized as follows. A motivating example is presented in Section 2. Section 3 gives the complete definition of Unary RDF Annotated Petri net systems, as well as its semantics and explains how to use the reachability graph to carry out some behavioral analysis. Section 4 briefly describes an implementation prototype. Finally, Section 5 presents some conclusions.
2. **MOTIVATING EXAMPLE**

Figure 1 shows the BPMN representation of a Business Process called Order Process. Although some works have studied the problem of semantic validation on semantic business process [5, 4], as far as we know, there are no works that can validate at design time a semantically annotate business process like the presented in Figure 1. Their major shortcomings that our formalism can solve are:

1. Current approaches do not consider some control flow operators such as loops. However, our formalism will inherit the control flow constructors of Petri nets.
2. Current approaches do not present a model including annotations with variables in preconditions, effects, inputs and outputs of the invoked services. In contrast, our formalism uses RDF graph patterns to annotate all of them. Thanks to the use of variables our formalism, like the one described in [3], allows representing the semantic propagation of data thought the process.
3. Most of the current approaches are limited to the modeling of processes and do not validate them. In contrast, our approach allows the verification of behavioral properties using the reachability graph.

3. **UNARY RDF ANNOTATED PETRI NET SYSTEMS**

Below we give the definition of Unary RDF Annotated Petri net systems and show how they can be useful for representing business processes with RDF annotations.

3.1 **Basic definitions**

Let us first introduce in a compact way some RDF elements [1] that will be necessary for the definition of Unary RDF Annotated Petri net systems. Let $RDF_L$ be the set of all RDF Literals, let $I$ be the set of all IRI references and let $RDF_B$ be the set of all blank nodes in RDF graphs.

(1) $RDF_T = RDF_L \cup I \cup RDF_B$ is called the set of RDF Terms.

(2) An RDF Graph is defined to be a finite subset of the set $(I \cup RDF_B) \times I \times RDF_T$. The set of RDF Graphs is denoted as $RDFG$. (3) The elements $(s, p, o)$ of an RDF graph are called RDF triples, which consist of a subject, a predicate and an object, respectively. Let $V$ be a set of variables disjoint from $RDF_T$. (1) A Basic Graph Pattern w.r.t. $V$ is defined to be a finite subset of the set $(I \cup RDF_B \cup V) \times (I \cup V) \times (RDF_T \cup V)$. The set of Basic Graph Patterns w.r.t. $V$ is denoted as $BGPV$.

(2) The elements of a Basic Graph Pattern w.r.t. $V$ are called triple patterns w.r.t. $V$. Notice that from the definition, $V$ can be empty or infinite and $BGPV = RDFG$.

Figure 2 represents basic graph pattern voc. It is composed of the triples: $t_1=(?oc, op:hasIdOrder, ?IdOrder), t_2=(?oc, op:hasClient, ?Client), t_3=(?Client, op:hasName, ?Name), etc.$ For example triple $t_1$ is composed of variables $?oc, ?IdOrder as subject and object respectively, and IRI op:hasIdOrder as predicate.

![RDF Graph Pattern](image)

Figure 2: RDF Graph Pattern voc. Semantic annotation describing the output of service Close of Figure 1.

3.2 **Semantics of Unary RDF Annotated Petri Net Systems**

Let us now concentrate on the definition of the formalism this paper is devoted to.

A Unary RDF Annotated Petri net, U-RDF-PN for short, is a tuple $N = (P, T, EV, OV, PC, G, PRE, POST, AV)$ where:

- $P = \{p_1, ..., p_m\}$, $m > 0$, $T = \{t_1, ..., t_n\}$, $n > 0$, are two finite non-empty disjoint sets whose elements are called places and transitions, respectively.
- $EV$ is the enabling variable mapping, defined over $T$, associating to each transition $t$ a finite (maybe empty) set of variables, called the transition enabling variable set while $OV$ is the output variable mapping, defined over $T$, associating to each transition $t$ a finite (maybe empty) set of variables, called the transition output variable set, and so that for any transition $t$, $EV(t)$ and $OV(t)$ are disjoint.
- $G$ is the guard mapping, defined over $T$, associating to each transition $t$ a boolean expression involving the set of enabling variables while $PC$ is the postcondition mapping, defined over $T$, associating to each transition $t$ an assignment to the transition output variables as a function of the transition enabling variables, in the form $\{o_1^t, ..., o_n^t\} := f_t([e_1^t, ..., e_m^t])$, where $OV(t) = \{o_1^t, ..., o_n^t\}$ and $EV(t) = \{e_1^t, ..., e_m^t\}$.
- $PRE \subseteq P \times T$ is the set of input arcs, while $POST \subseteq T \times P$ is the set of output arcs and $AV : PRE \cup POST \rightarrow \bigcup_{t \in T} BGP_{EV(t)\cup OV(t)}$ such that for every $(p,t) \in PRE, AV((p,t)) \in BGP_{EV(t)}$ and for every $(t,p) \in POST, AV(t,p) \in BGP_{EV(t)\cup OV(t)}$ is the arc valuation mapping.
The RDF triples of all RDF graphs of the example (\( \{ o1 \} \)) are a subset of the RDF triples in the example removes \( o1 \) from place \( p1 \) and puts \( r1 \) in place \( p1 \) and \( s1 \) in place \( p3 \) (\( r1 \) and \( s1 \) are represented in Table 1 and will be explained later). Given a Petri net system, the set of its reachable states is composed of the set of markings reachable from the initial marking. In the case of such set to be finite, the reachability graph can be obtained. The nodes of this graph are the set of reachable markings, while there is an arc joining states \( s \) and \( s' \) when marking \( s' \) is reached from \( s \) by firing a transition. The label of such a transition corresponds to the binding that made the firing possible.

3.3 Reachability Graph Based Analysis in Unary RDF Annotated Petri nets

Let us show how the reachability graph can be used to analyze some system properties. As an example, let us assume that the information of the order to be processed is known (\( o1 \) in Table 1 contains only RDF Literals and IRIs but does not contains blank nodes). Figure 4 depicts the reachability graph for this case. Table 1 shows the values of the RDF graphs appearing in the reachability graph.

On the contrary to ordinary Petri nets, the concrete reachability graph has semantic information about the data that is propagated through the system. This way it would be possible to carry any information about the data at any moment in the execution. This will make it possible to ask questions about data (Will the order be carried out?, Will the client receive the receipt?, Is the client of type 1?). For this given concrete order, Will the process terminate?
The firing of a transition RGN.rd f RGN database, variable v in the input places. If the query returns that the value of a enables of a transition AllegroGraph RDFStore U-RDF-PNs. It has been implemented using Java and are proposing, we have implemented a prototype for the executed since every path from the initial state leads to In the example we are considering, we can conclude that properties at design time. On the reachability graph in order to analyze behavioral properties that we are interested in. Our ongoing research includes the development of new techniques that can abstract the data that will be received at runtime and the application of Unary RDF Annotated Petri Nets to verify the correct interaction of several business processes with semantic annotations.

6. REFERENCES