SCALESEM
Evaluation of Semantic Graph based on Model Checking

Mahdi Gueffaz, Sylvain Rampacek and Christophe Nicolle
LE2I, UMR CNRS 5158, University of Bourgogne, BP 47870, 21078 Dijon cedex, France

Keywords: Semantic graph, Model-checking, Temporal logic.

Abstract: Semantic interoperability problems have found their solutions using languages and techniques from the Semantic Web. The proliferation of ontologies and meta-information has improved the understanding of information and the relevance of search engine responses. However, the construction of semantic graphs is a source of numerous errors of interpretation or modelling and scalability remains a major problem. The processing of large semantic graphs is a limit to the use of semantics in current information systems. The work presented in this paper is part of a new research at the border of two areas: the semantic web and the model checking. This line of research concerns the adaptation of model checking techniques to semantic graphs. In this paper, we present a first method of converting RDF graphs into NμSMV and PROMELA languages.

1 INTRODUCTION

W3C[1] aims to standardize the representation and the exchange of information on the WEB. This objective should help make the information understandable for both automated processes and users. The homogenization of computer exchanges took place due to the introduction of the XML (Bray and al, 2006) standard. This standard has enabled the program to manipulate information through languages with hierarchical structure mark-up defined by grammars derived from the XML standard. However, this effort has not helped improve the user's understanding of information. Thus, new standards have been developed to enable the semantic representation of information in the form of XML-derived languages. This base is called Semantic Web standards and it is usually represented as a stack of languages ranging from automatic processes oriented languages to languages representing more abstract concepts of formal semantics (Berners-Lee, 2001). These languages are used to represent the semantics associated with information, whatever its form and structure. To allow the construction of semantic graph, many tools have been developed like Annotea (Kahan and al, 2001) which is a project of the W3C that specifies the infrastructure for the annotation of Web documents. The main format used in the annotation is RDF and the types of documents can be annotated are HTML documents or XML based. However, none provides the functionality to verify the consistency of semantics, and reduce errors annotations.

This paper proposes a new way to check these semantic graphs by model-checking in order to reduce errors in annotation and make the data more relevant. Model checking is an automatic verification technique, it has been applied to many cases in industry, for example (Katoen, 2002), in the Netherlands, model-checking has revealed several serious flaws in the design of control system of a barrier protection against flooding which protects the main port of Rotterdam against floods.

Model checking is a powerful tool for system verification because it can reveal errors that were not discovered by other formal methods such as testing or simulation. Model checking uses temporal logic to describe the properties checking the system model.

[1]World Wide Web Consortium.

Gueffaz M., Rampacek S. and Nicolle C..
SCALESEM - Evaluation of Semantic Graph based on Model Checking.
DOI: 10.5220/0003334703950398
In *Proceedings of the 7th International Conference on Web Information Systems and Technologies* (WEBIST-2011), pages 395-398
ISBN: 978-989-8425-51-5

2 MODEL-CHECKING AND TEMPORAL LOGIC OVERVIEW

Formal methods (Katoen, 2002) offer great potential for an early inclusion of verification in the design process, providing technical audit more efficiently and reduce the verification time. Formal methods are highly recommended techniques for the development of software. Two types of formal verification methods can be distinguished: methods based on the proof of the theorem and the methods based on models.

The model checker examines all relevant system states in order to check whether they satisfy the desired property. The model checker gives a counter example that indicates how the model can violate the property. With a help of a simulator, the user can locate the error and adapt the model or the property to prevent the violation of property (Fig. 1).

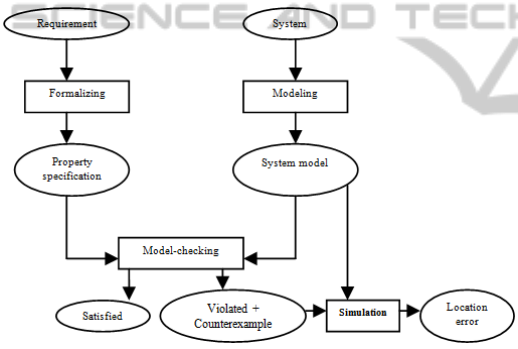

Figure 1: Model Checking Approach.

The concepts of temporal logic used for the first time by Pnueli (Pnueli, 1977) in the specification of formal properties are fairly easy to use. The operators are very close in terms of natural language. The formalization in temporal logic is simple enough although this apparent simplicity therefore requires significant expertise. Temporal logic allows representing and reasoning about certain properties of the system, so it is well-suited for the systems verification.

3 THE SCALESEM APPROACH

We use SPIN (Ben-Ari, 2008) and NµSMV (Cimatti and al, 2000) as model checkers to check the model of semantic graphs. SPIN is a software tool for verifying system models. The system is described in a language model called PROMELA. NµSMV is the amelioration of SMV model checker, working on the same simple principles as SMV.

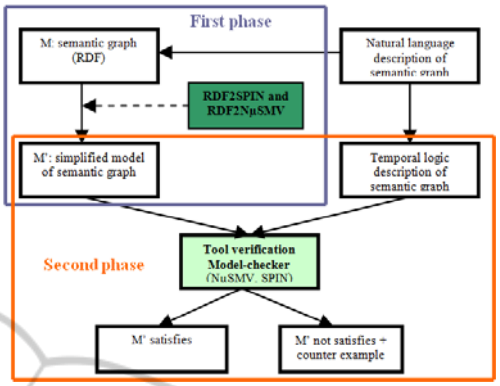

Figure 2: The Scalesem Architecture.

In Fig. 2, we present the architecture of our approach. In this architecture, from a natural language description, we can get the semantic graph (RDF[2]) and its description in temporal logic, as shown in the example found in the section VII. We divide this architecture in two phases. The first phase concerns the transformation of the semantic graph into a model using our tools RDF2SPIN and RDF2NµSMV. There are three steps in this transformation. The first step is to explore the entire RDF graph to obtain the triplets table. The second step is to determine a root for the graph, and the last step is to write the model that represents the semantic graph in the PROMELA or NµSMV languages. The second phase concerns the verification of properties expressed in temporal logic on the model using the model-checker SPIN or NµSMV.

3.1 Introducing RDF

RDF is a language developed by the W3C to bring a semantic layer to the Web (Becket and McBride, 2004). It allows the connection of Web resources using directed labelled edges. The structure of RDF documents is a complex labelled directed graph. An RDF document is a set of triples <subject, predicate, object>. These RDF graphs are not necessarily connected, meaning they may have no root vertex from which all the other vertices are reachable.

3.2 Exploring RDF Graph

We achieve this by appropriate explorations of the RDF graphs, as explained below. Let us consider

[2]Resource Description Framework

that an RDF graph is represented as a couple (V, E), where V is the set of vertices and $E \subseteq VxV$ is the set of edges. For a vertex x, we note $E(x) = \{y \in V | (x,y) \in E\}$ the set of its successor vertices. We use depth-first search algorithm, illustrated below to explore graph, knowing that the breadth-first algorithm also works in this context.

```
Algorithm: procedure Dfs (x):
begin
  visited(x) :=  true;
   // vertex x becomes visited
  p(x) := 0; // start exploring its successors
  stack := push(x, nil);
 while stack ≠ nil do
   y := top(stack);
   if p(y) < |E (y)| then
    // y has some unexplored successors
    z := E (y) p(y);
    p(y) := p(y)+1;
    // take the next successor of y
    if ¬visited (z) then
    visited(z) := true;  // visit it
    p(z) := 0;//start exploring its successors
    stack := push(z, stack)
    endif
   else //all successors of y were explored
   stack := pop(stack)
  endif
 end
end
```

3.3 Determining a Root Vertex

If the RDF graph has no vertex root, we must create a root for the graph.

```
Algorithm: procedure RootElection():
// precondition: ∀ x ∈ V.visited(x) = false
Begin // first phase
 root_list := nil;
 forall  x ∈ V do
  if ¬visited(x) then
   Dfs(x);
   root_list := cons(x, root_list)
  endif
 endfor;
//second phase
 if |root_list|= 1 then
  root := head(root_list)
  // the single partial root is the global root
  else
  forall x ∈ V do  visited(x):= false;
  endfor;
  forall x ∈ root_list do
   // reexplore partial roots in reverse order
   if ¬visited(x) then Dfs(x)
   else
    root_list := root_list \ {x}
     // partial root is not a real one
   endif
  endfor;
  if |root_list| = 1 then
   root := head(root_list)
   // a single partial root is the global root
   else
   root := new_node();
   // new root predecessor of the partial roots
   E(root) := root_list
  endif
 endif
```

The first phase explores the graph until it is fully explored, and inserts in root_list all vertices that have no predecessor. If root_list contains a single vertex, so overall it is the global root of the graph since all the other vertex are accessible from it and it is useless to the second phase has passed. Otherwise, any vertex contained in root_list could also be a root of the graph: the role of the second phase is to determine which of the partial root the root of the global graph is.

The second phase performs a new wave of exploration of the roots contained in partial *root_list* in reverse order in which they were inserted in the list. If a root in *root_list* is to be visited by a partial root, it is removed from the list because it is not a partial root. At the end of this phase, all partial roots of the graph are present in *root_list*. Indeed, each vertex is unreachable from the partial roots which were explored during the second phase. A new root is created (see Fig. 3), having as successor all the partial roots of *root_list*, which ensures that all vertices of the graph are accessible from the new root. Therefore, such a summit is inaccessible from other nodes of the graph.

3.4 Generating the Model

The third step is divided into three sub-steps. The first and the second one consist in generating two tables (triplets table and resources and values table). The last one consists in producing the model writing in PROMELA language for SPIN and in NμSMV language for NμSMV.

4 EXAMPLE AND BENCHMARK

To illustrate our approach, we take the natural language description as follows:

Ninety-three is a novel by Victor Hugo published in 1874, whose theme is the French Revolution. Victor Hugo was born in February 26, 1802 in Besançon.

From the description above, we can easily extract

simple propositions, see Table 1. Table 2 presents the RDF triples derived from the Table 1.

Table 1: Short list of simple propositions.

1	“Ninety-three is a novel”
2	“Ninety-three its author is Victor Hugo”
3	“Ninety-three has been published in 1874”
4	“Ninety-three’s theme is the French revolution”
5	“Victor Hugo was born in February 26, 1802”
6	“Victor Hugo was born in Besançon”

Table 2: Corresponding RDF Triples.

	Subject	Predicate	Object
1	Ninety-three	is	Novel
2	Ninety-three	author	Victor Hugo
3	Ninety-three	Published	1874
4	Ninety-three	theme	French revolution
5	Victor Hugo	Date_born	February 26, 1802
6	Victor Hugo	Place_born	Besançon

From the previous description in natural language, we can express it in temporal logic as shown in Table 3.

Table 3: Example of Temporal Logic representation.

	Temporal logic	Explanation
1	Always (Ninety-three → next novel)	We check that ninety three is a novel
2	Always (Ninety-three → next Victor Hugo)	We check that ninety three is written by Victor Hugo

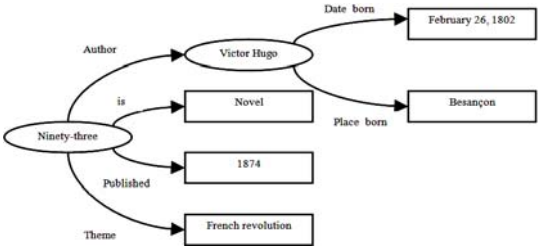

Figure 3: RDF Graph.

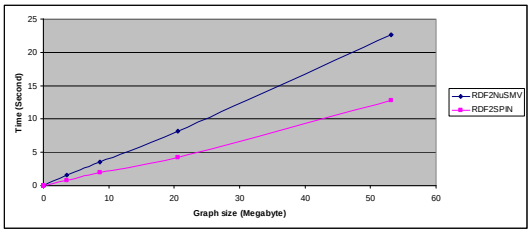

Figure 4: Time conversion of semantic graphs.

Now, we will be able to transform the RDF graph in Fig. 3 with our tools "RDF2SPIN" and RDF2NμSMV" into a model in order to check each formula of temporal logic described in Table 3 and see if each formula is verified or not in the model.

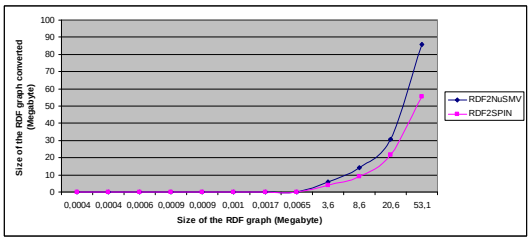

Figure 5: Size of the models.

5 CONCLUSIONS

This paper presents a new technique for the semantic graphs verification by using a model-checker. Knowing that the model-checker does not understand the semantic graphs, we developed two tools RDF2SPIN and RDF2NμSMV to convert them into PROMELA and NμSMV languages in order to be verified with the temporal logics.

In future work, we would like to convert the SPARQL query language for RDF graphs into queries using the operator of the temporal logic, to have a better verification of RDF graphs representing the building industry.

REFERENCES

Becket, D., McBride, B., 2004. RDF/ XML Syntax Specification (Revised). W3C recommandation. http://www.w3.org/TR/2004/REC-rdf-syntax-grammar-20040210/.

Ben-Ari, M., 2008. Principles of the SPIN Model Checker. Springer. ISBN: 978-1-84628-769-5.

Berners-Lee, t., Hendler, J., and Lassila, O., 2001. The Semantic Web. Scientific American. pp. 34–43.

Bray, T., Paoli, J., Sperberg-McQueen., C., M., Maler, E., Yergeau, F., Cowan, J., 2006. Extensible Markup Language (XML) 1.1 (second edition) W3C recommendation, http://www.w3.org/TR/2006/REC-xml11-20060816/.

Cimatti, A., Clarke, E., Giunchiglia, F., Roveri, M., 2000. NuSMV: a new symbolic model checker.

Kahan, J., Koivunen, M., Prud'Hommeaux, E., Swick, R., R, 2001. Annotea: An Open RDF Infrastructure for Shared Web Annotations, in *Proc. of the WWW 10th International Conference*, Hong Kong. http://www10.org/cdrom/papers/488/index.html

Katoen, J. P., 2002. The principal of Model Checking. University of Twente.

Pnueli, A., 1977. The temporal logic of programs. In proc. *18th IEEE Symp. Foundations of Computer Science* (FOCS’77), Providence, RI, USA. pages 46-57.