Retrieving and Matching RDF Graphs by Solving the Satisfiability Problem*

Sheng Liu[1,2] and Jian Zhang[1]
[1] Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences
Beijing 100080, China, {lius,zj}@ios.ac.cn
[2] Graduate University, Chinese Academy of Sciences, Beijing 100049, China

Abstract

The Resource Description Framework (RDF) has been accepted as a standard for semantic representation of resources. Efficient methods and tools are needed to solve problems emerging from RDF based systems, for example, checking equality of two RDF graphs and retrieving subgraphs from another RDF graph. This paper proposes a method that encodes these problems into satisfiability (SAT) instances and solves them by employing efficient SAT solvers. A prototype tool is implemented and preliminary experimental results are given.

1 Introduction

The Resource Description Framework (RDF) [4] is a fundamental lower layer on top of which the semantic web is built. It is proposed by the WWW Consortium (W3C) to express metadata about information resources on the Web. An RDF graph is a set of assertions in the form of subject-predicate-object triples of information resources. Each triple is denoted by $<s, p, o>$ and can be represented as an arc with starting node s, ending node o and the label p of the arc.

Presently, more and more web resources are organized in the form of RDF. Thus, how to efficiently solve RDF related problems becomes an important issue. For example, for testing and debugging purpose, it is necessary for developers of RDF based systems to check the equality of two RDF graphs [2]. While for end users, semantic searching is the main requirement, so checking inference and entailment between RDF graphs is more important. This paper presents a method that is useful for this kind of operations.

Although RDF is based on directed labeled graphs, it is actually a logic, with a model theoretic semantics and an entailment relationship [3]. So, it gives chances for us to use automated reasoning tools to reason about RDF graphs. Using the finite model searching tool SEM [9], Zhang [8] gives some methods to demonstrate satisfiability and non-entailment. The technique proposed in this paper is also based on logic representation of RDF. But we do not use first-order logic tools. Considering the high efficiency of the satisfiability (SAT) solvers, instead, we match RDF graphs and test RDF entailment by means of the propositional logic SAT solvers.

The outline of the paper is as follows. Section 2 introduces some notations and tools of the classical propositional logic. Section 3 introduces the semantics of RDF. In section 4 we propose our method that translates RDF graphs into propositional logic formulas. Experimental results are given in section 5. At last we conclude the paper.

2 Propositional Logic Satisfiability

We assume that the readers already have some knowledge of mathematical logic. So in this section we only briefly recall some notations and introduce some powerful SAT solvers.

In the propositional logic, a propositional formula is a string constructed iteratively from variables and connectives. In a propositional formula, each variable can take only one value from the domain of the truth values $\Sigma = \{TRUE, FALSE\}$ and connectives usually include conjunction, disjunction, negation, and implication (denoted by $\wedge$, $\vee$, $\neg$, $\rightarrow$, respectively). A literal is a variable or its negation. A clause is the disjunction of literals.

A formula is satisfiable if and only if there exists an assignment of truth values to all the variables such that the formula itself gets the $TRUE$ value. The SAT problem is to determine whether a given propositional logic formula is satisfiable or not.

SAT is a central problem in computer science and it is studied widely. Recently, great progress has been made in efficiently solving this problem and many powerful SAT tools are developed, for example Zchaff [5] and SATO [7]. Although SAT is an NP-Complete problem, these SAT tools are very efficient in practice, and they have solved some difficult problems in mathematics and hardware verification.

*Supported by the National Science Foundation of China (NSFC).

Proceedings of the 2006 IEEE/WIC/ACM International Conference
on Web Intelligence (WI 2006 Main Conference Proceedings)(WI'06)
0-7695-2747-7/06 $20.00 © 2006 IEEE

IEEE COMPUTER SOCIETY

Authorized licensed use limited to: Institute of Software. Downloaded on April 04,2022 at 05:34:30 UTC from IEEE Xplore. Restrictions apply.

The method used by these tools is some kind of backtracking search algorithm. Essentially, they examine all the possible assignments of variables. In general cases, they receive a set of clauses as input and output the truth value assignment (if the formula is satisfiable) or information of unsatisfiability (if the formula is not satisfiable).

3 RDF Semantics

In this section we only focus on some simple notations that are related to RDF entailment [1].

We consider a set of terms υ which is composed of three disjoint sets: a set μ of URI references (urirefs), a set β of blank nodes, and a set ι of literals. The set ι is composed of two disjoint sets: the set ι_P of plain literals and the set ι_T of typed literals. Let V be a subset of υ, then we denote the set of urirefs of V by $\mu(V)$, denote the set of blank nodes of V by $\beta(V)$. The rest may be deduced by analogy.

Definition 1 (RDF tripleset) *An RDF tripleset is a subset of $(\mu \cup \beta) \times \mu \times \upsilon$. Its elements are called RDF triples.*

Let G be an RDF tripleset. We denote by $\upsilon(G)$ the set of terms that appear in any triples of G:

$$\upsilon(G) = \{x \in \upsilon | \exists <s, p, o> \in G, x = s \vee x = p \vee x = o\}$$

We denote by $D(G)$ the set of terms that contains all the non-blank terms of $\upsilon(G)$. We call $D(G)$ a domain of G, which means that all blank nodes in G can have values in $D(G)$. Note that $D(G)$ can be arbitrarily large, so G can have an arbitrarily large domain.

Definition 2 (Simple Interpretation) *Let V be a set of terms. An interpretation on V is a 5-tuple $<I_R, I_P, I_{EXT}, I_S, I_L>$ where I_R is a non-empty set of resources, I_P is a set of properties, I_{EXT} is a mapping from I_P into the powerset of $I_R \times I_R$, I_S is a mapping from $\mu(V)$ into $I_R \cup I_P$, I_L is a mapping from $\iota_T(V)$ into I_R.*

Definition 3 (Model) *Let G be an RDF tripleset. An interpretation $<I_R, I_P, I_{EXT}, I_S, I_L>$ on $D(G)$ is a model of G iff there exists a mapping*

$$\gamma : \upsilon(G) \rightarrow I_R \cup I_P$$

such that:

$\forall l \in \iota_P(\upsilon(G)), \gamma(l) = l$
$\forall l \in \iota_T(\upsilon(G)), \gamma(l) = I_L(l)$
$\forall u \in \mu(\upsilon(G)), \gamma(u) = I_S(u)$
$\forall b \in \beta(\upsilon(G)), \gamma(b) \in I_R$
$\forall <s, p, o> \in G, <\gamma(s), \gamma(o)> \in I_{EXT}(\gamma(p))$

Definition 4 (Satisfiability, Entailment) *Let G and H be two RDF triplesets. We say that G is satisfiable if there is a model of G. We say that H is a semantic consequence of G (we also say that G entails H, denoted by $G \models H$) if every model of G is also a model of H.*

Definition 5 (Instance) *Let G be an RDF tripleset. Let us consider an instance mapping*

$$\alpha : \upsilon(G) \rightarrow D(G)$$

α maps each blank node of G to a term of $D(G)$, and each uriref or literal to itself. The RDF tripleset

$$G_\alpha = \{<\alpha(s), p, \alpha(o)> | <s, p, o> \in G\}$$

is called an instance of G.

Lemma 1 (Interpolation Lemma) *Let G and H be two triplesets. Then $G \models H$ iff there exists an instance H_α of H such that $H_\alpha \subseteq G$.*

4 Encoding Entailment and Matching Problems as SAT

From Definition 4 we know that in order to prove $G \models H$, we must test all models of G and make sure that each model of G is also a model of H. But this method is sometimes infeasible in practice because G can have an infinite number of models.

However, the $InterpolationLemma$ (Lemma 1) seems to be more useful in practice. If we want to prove $G \models H$, we only need to find an instance H_α of H such that $H_\alpha \subseteq G$. In order to find an instance of H, first of all we should define the domain of H. It can be proved that it is proper to set the domain of H to be $\upsilon(G)$. Then we construct H_α by letting each blank node in H take a value in $\upsilon(G)$ and check whether H_α is a subset of G. Remember that we want to determine whether H_α exists, so we have to try each possible H_α. That is to say, we have to let each blank node take all possible values. Of course it is impossible for us to do it by hand. Fortunately, we can employ the powerful SAT solvers. Next we give our encoding and introduce our method.

4.1 Basic Encoding

Let the set of all subjects and objects in H be $N_H = \{h_1, h_2, \ldots, h_m\}$ ($|N_H| = m$) and the set of all subjects and objects in G be $N_G = \{g_1, g_2, \ldots, g_n\}$ ($|N_G| = n$). Let the set of properties in H be P_H ($|P_H| = l_H$) and the set of properties in G be P_G ($|P_G| = l_G$). Without loss of generality, we assume that all elements in N_H are blank nodes.

Define a mapping

$$f : N_H \times N_G \rightarrow \Sigma$$

Proceedings of the 2006 IEEE/WIC/ACM International Conference
on Web Intelligence (WI 2006 Main Conference Proceedings)(WI'06)
0-7695-2747-7/06 $20.00 © 2006 IEEE

IEEE COMPUTER SOCIETY

Authorized licensed use limited to: Institute of Software. Downloaded on April 04,2022 at 05:34:30 UTC from IEEE Xplore. Restrictions apply.

$f(h_i, g_j) = TRUE$ iff the blank node h_i in N_H takes value g_j of N_G.

Define a mapping

$$p : N_H \times P_H \times N_H \rightarrow \Sigma$$

$p(h_i, j, h_k) = TRUE$ iff $<h_i, j, h_k>$ is a triple of H.

Define a mapping

$$q : N_G \times P_G \times N_G \rightarrow \Sigma$$

$q(g_i, j, g_k) = TRUE$ iff $<g_i, j, g_k>$ is a triple of G.

Because each blank node in N_H should take a value from N_G, we have the following formulas:

$$f(h_1, g_1) \vee f(h_1, g_2) \vee \cdots \vee f(h_1, g_n)$$

$$f(h_2, g_1) \vee f(h_2, g_2) \vee \cdots \vee f(h_2, g_n)$$

$$\vdots \quad \vdots \quad \vdots$$

$$f(h_m, g_1) \vee f(h_m, g_2) \vee \cdots \vee f(h_m, g_n)$$

Each blank node in N_H should not take more than one values from N_G at the same time, so $\forall h_i \in N_H$ we have the following formulas:

$$\begin{array}{ll} \neg f(h_i, g_1) \vee \neg f(h_i, g_2) & \neg f(h_i, g_1) \vee \neg f(h_i, g_3) \\ \cdots & \neg f(h_i, g_1) \vee \neg f(h_i, g_n) \\ \neg f(h_i, g_2) \vee \neg f(h_i, g_3) & \neg f(h_i, g_2) \vee \neg f(h_i, g_4) \\ \cdots & \neg f(h_i, g_2) \vee \neg f(h_i, g_n) \end{array}$$

$$\vdots \quad \vdots$$

$$\neg f(h_i, g_{n-1}) \vee \neg f(h_i, g_n)$$

Of course there are also some unit clauses representing whether $<h_i, j, h_k>$ ($<g_i, j, g_k>$) is a triple of H (G). For example, if $<g_i, j, g_k>$ is a triple in G, then such a clause $q(g_i, j, g_k)$ is inserted; otherwise its negation $\neg q(g_i, j, g_k)$ is inserted.

By now we have encoded the mapping into SAT clauses. Next we will add clauses to make sure that the image set of H under the mapping is a subset of G. It is easy to understand that if f maps h_{x_1} of N_H to g_{y_1} in N_G and maps h_{x_2} of N_H to g_{y_2} in N_G, then, for an arbitrary property z in P_H, if $<h_{x_1}, z, h_{x_2}>$ is a triple in H then $<g_{y_1}, z, g_{y_2}>$ must also be a triple in G. So $\forall z \in P_H$, we have such a formula:

$$f(h_{x_1}, g_{y_1}) \wedge f(h_{x_2}, g_{y_2}) \rightarrow (p(h_{x_1}, z, h_{x_2}) \rightarrow q(g_{y_1}, z, g_{y_2}))$$

Note that the above formula should be reformulated into the form of clause when submitted to SAT solvers:

$$\neg f(h_{x_1}, g_{y_1}) \vee \neg f(h_{x_2}, g_{y_2}) \vee \neg p(h_{x_1}, z, h_{x_2}) \vee q(g_{y_1}, z, g_{y_2})$$

So far we have encoded RDF entailment into propositional logic formulas. While in the case of RDF matching problem where $m = n$, we will have to add some formulas such as

$$\begin{array}{ll} \neg f(h_1, g_i) \vee \neg f(h_2, g_i) & \neg f(h_1, g_i) \vee \neg f(h_3, g_i) \\ \cdots & \neg f(h_1, g_i) \vee \neg f(h_m, g_i) \end{array}$$

to make sure that two different blank nodes in N_H can not be mapped to the same element in N_G.

4.2 Encoding with Simple Reasoning

It is evident that the encoding in subsection 4.1 is easy and direct. But even small RDF entailment problem will induce too many variables and clauses.

In fact, most of the clauses take the following form:

$$\neg f(h_{x_1}, g_{y_1}) \vee \neg f(h_{x_2}, g_{y_2}) \vee \neg p(h_{x_1}, z, h_{x_2}) \vee q(g_{y_1}, z, g_{y_2})$$

Because the truth values of all the $p(h_{x_1}, z, h_{x_2})$-form or $q(g_{y_1}, z, g_{y_2})$-form variables can be known from the input files immediately, we can perform some reasoning on such variables as follows. As for the above clause, if the logic conclusion of the subclause $\neg p(h_{x_1}, z, h_{x_2}) \vee q(g_{y_1}, z, g_{y_2})$ is $TRUE$, then the original clause also has a $TRUE$ value. Thus we need not insert such a clause and can simply delete it. Otherwise, since the truth value of the subclause $\neg p(h_{x_1}, z, h_{x_2}) \vee q(g_{y_1}, z, g_{y_2})$ is $FALSE$, then we have to consider the subclause $\neg f(h_{x_1}, g_{y_1}) \vee \neg f(h_{x_2}, g_{y_2})$. Thus we only need to insert this shorter clause. After that, we can further delete all the $p(h_{x_1}, z, h_{x_2})$-form or $q(g_{y_1}, z, g_{y_2})$-form clauses (they must be satisfied automatically) and variables (because they do not occur in any clauses). Thus the number of variables can also be reduced drastically from $m * n + l_H * m^2 + l_G * n^2$ to $m * n$.

Based on the above encoding we implemented a prototype tool, which reads in two files of RDF triplesets. If the second RDF tripleset entails the first RDF tripleset, then the tool will return the correct mapping in which each blank node of the first tripleset is assigned a value from the second tripleset. Otherwise, it returns UNENTAILMENT. What is more, if entailment exists, we can also return all the correct mappings. We think that this is very useful for end users. Because in practice, the end users usually input the query RDF graphs into the RDF based database or WWW, and they want to find all the possible matchings.

Example Given a data graph in Figure 1, suppose that we want to query all the painters' name, their masterpiece and the creating year. First we may construct a query graph as shown in Figure 2. Then we retrieve the query graph in the data graph with our tool.

$f_1(\_{:}b1) = r1 \quad f_1(\_{:}b2) = r2 \quad f_1(\_{:}b3) = 1638$

$f_1(\_{:}b4) =$ "$Stone\ Bridge$"

Proceedings of the 2006 IEEE/WIC/ACM International Conference
on Web Intelligence (WI 2006 Main Conference Proceedings)(WI'06)
0-7695-2747-7/06 $20.00 © 2006 IEEE

IEEE COMPUTER SOCIETY

Authorized licensed use limited to: Institute of Software. Downloaded on April 04,2022 at 05:34:30 UTC from IEEE Xplore. Restrictions apply.

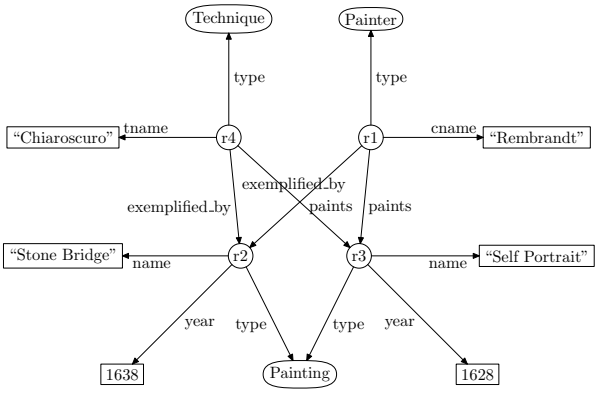

Figure 1. An example graph [6]

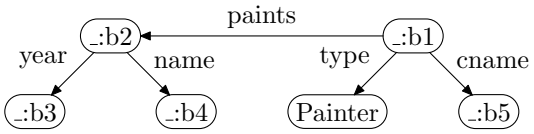

Figure 2. The query graph

$f_1(\_:b5) =$ "$Rembrandt$"
and
$f_2(\_:b1) = r1 \quad f_2(\_:b2) = r3 \quad f_2(\_:b3) = 1628$
$f_2(\_:b4) =$ "$Self\ Portrait$"
$f_2(\_:b5) =$ "$Rembrandt$"

5 Experiments

We use randomly generated RDF triplesets in our experiments. The query graphs have 10 nodes while the data graphs have nodes 50, 60, 70, 80, 90, 100 respectively. Each test is executed 1000 times and the average running times are given in Figure 3 and Figure 4.

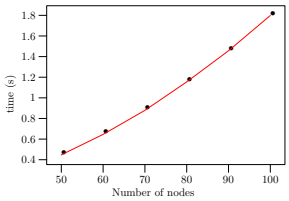

Figure 3. Time of encoding

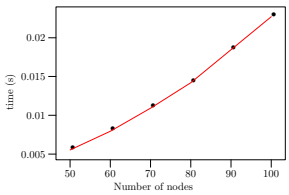

Figure 4. Time of SAT solving

From Figure 3 and Figure 4 we easily find that it takes little time for current SAT solvers to solve the encoded RDF entailment problem while most of the running time is spent on encoding and preprocessing. Despite that, the total time is still not too much.

On the other hand, we notice that the curves in both Figure 3 and Figure 4 exhibit a nearly linear growth. The growing trend of the curves also shows the tool's extensibility to larger graphs.

We also experiment on RDF graph matching problem. But due to the space limit, we do not give the details.

6 Conclusions

In this paper, we introduce a SAT encoding method for RDF entailment problem and RDF graph matching problem. By employing a SAT solver as a subroutine, the tool based on this method works well on randomly generated instances. Of course, more experiments are needed to test the applicability of the method to large graphs and massive resource graph repository. However, we believe that our method is promising. We expect our work will bring more attention to this direction of such problems.

References

[1] J.-F. Baget. RDF entailment as a graph homomorphism. In *Proceedings of the Fourth International Semantic Web Conference (ISWC05)*, pages 82–96. Springer, 2005.

[2] J. J. Carroll. Matching RDF graphs. In *Proceedings of the First International Semantic Web Conference (ISWC02)*, pages 5–15, London, UK, 2002. Springer-Verlag.

[3] P. Hayes, editor. *RDF Semantics*. W3C Recommendation, 10 February 2004. available at `http://www.w3.org/TR/rdf-mt/`.

[4] F. Manola and E. Miller, editors. *RDF Primer*. W3C Recommendation, 10 February 2004. available at `http://www.w3.org/TR/rdf-primer/`.

[5] M. W. Moskewicz, C. F. Madigan, Y. Zhao, L. Zhang, and S. Malik. Chaff: Engineering an Efficient SAT Solver. In *Proceedings of the 38th Design Automation Conference (DAC01)*, 2001.

[6] H. Stuckenschmidt, A. Aerts, J. Broekstra, G.-J. Houben, and J. Broekstra. ISWC04 tutorial: Theory and practice of RDF query processing. In *The Third International Semantic Web Conference (ISWC04)*, 2004. available at `http://www.cs.vu.nl/~heiner/ISWC04Tutorial.html`.

[7] H. Zhang. SATO: an efficient propositional prover. In *Proceedings of the International Conference on Automated Deduction (CADE97)*, pages 272–275, 1997.

[8] J. Zhang. Finding simple RDF interpretations using satisfiability checkers in the classical logics. Technical Report ISCAS-LCS-03-13, Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences, Beijing, China, December 2003.

[9] J. Zhang and H. Zhang. SEM: a system for enumerating models. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence (IJCAI95)*, pages 298–303, 1995.

Proceedings of the 2006 IEEE/WIC/ACM International Conference on Web Intelligence (WI 2006 Main Conference Proceedings)(WI'06)
0-7695-2747-7/06 $20.00 © 2006 IEEE

IEEE COMPUTER SOCIETY

Authorized licensed use limited to: Institute of Software. Downloaded on April 04,2022 at 05:34:30 UTC from IEEE Xplore. Restrictions apply.