SomeWhere: a scalable peer-to-peer infrastructure for querying distributed ontologies

M.-C. Rousset[2]
joint work with P. Adjiman[1], P.Chatalic, F. Goasdoué, and L.Simon[1]

[1] LRI, bâtiment 490, Université Paris-Sud 11, 91405 Orsay Cedex, France
[2] LSR-IMAG, BP 72, 38402 St Martin d'Heres Cedex, France

Abstract. In this invited talk, we present the SomeWhere approach and infrastructure for building semantic peer-to-peer data management systems based on simple personalized ontologies distributed at a large scale. Somewhere is based on a simple class-based data model in which the data is a set of resource identifiers (e.g., URIs), the schemas are (simple) definitions of classes possibly constrained by inclusion, disjunction or equivalence statements, and mappings are inclusion, disjunction or equivalence statements between classes of different peer ontologies. In this setting, query answering over peers can be done by distributed query rewriting, which can be equivalently reduced to distributed consequence finding in propositional logic. It is done by using the message-passing distributed algorithm that we have implemented for consequence finding of a clause w.r.t a set of distributed propositional theories. We summarize its main properties (soundness, completeness and termination), and we report experiments showing that it already scales up to a thousand of peers. Finally, we mention ongoing work on extending the current data model to RDF(S) and on handling possible inconsistencies between the ontologies of different peers.

1 Overview of SomeWhere

SomeWhere promotes a "small is beautiful" vision of the Semantic Web [1] based on simple personalized ontologies (e.g., taxonomies of atomic classes) but which are distributed at a large scale. In this vision of the Semantic Web introduced by [2], no user imposes to others his own ontology but logical mappings between ontologies make possible the creation of a web of people in which personalized semantic marking up of data cohabits nicely with a collaborative exchange of data. In this view, the Web is a huge peer-to-peer data management system based on simple distributed ontologies and mappings.

For scalability purpose, we have chosen a simple class-based data model in which the data is a set of resource identifiers (e.g., URIs), the ontologies are (simple) definitions of classes possibly constrained by inclusion, disjunction or equivalence statements, and mappings are inclusion, disjunction or equivalence statements between classes of different peer ontologies. That data model is in accordance with the W3C recommendations since it is captured by the propositional fragment of the OWL ontology language (http://www.w3.org/TR/owl-semantics).

Query answering through ontologies is achieved using a rewrite and evaluate strategy. In SomeWhere the query rewriting problem can be reduced to a consequence finding problem in distributed propositional theories. It is performed by a message-passing algorithm named DeCA: DEcentralized Consequence finding Algorithm [3]. As a result, query answering in SomeWhere is sound, complete and terminates. Moreover, the detailed experiments reported in [4] show that it scales up to 1000 peers.

2 Illustrative example

We illustrate the SomeWhere data model on a small example of four peers modeling four persons Ann, Bob, Chris and Dora, each of them bookmarking URLs about restaurants they know or like, according to their own taxonomy for categorizing restaurants.

Ann, who is working as a restaurant critic, organizes its restaurant URLs according to the following classes:

• the class $Ann{:}G$ of restaurants considered as offering a "good" cooking, among which she distinguishes the subclass $Ann{:}R$ of those which are rated: $Ann{:}R \sqsubseteq Ann{:}G$

• the class $Ann{:}R$ is the union of three disjoint classes $Ann{:}S1$, $Ann{:}S2$, $Ann{:}S3$ corresponding respectively to the restaurants rated with $1, 2$ or 3 stars:

$Ann{:}R \equiv Ann{:}S1 \sqcup Ann{:}S2 \sqcup Ann{:}S3$

$Ann{:}S1 \sqcap Ann{:}S2 \equiv \bot \quad Ann{:}S1 \sqcap Ann{:}S3 \equiv \bot$

$Ann{:}S2 \sqcap Ann{:}S3 \equiv \bot$

• the classes $Ann{:}I$ and $Ann{:}O$, respectively corresponding to Indian and Oriental restaurants

• the classes $Ann{:}C$, $Ann{:}T$ and $Ann{:}V$ which are subclasses of $Ann{:}O$ denoting Chinese, Taï and Vietnamese restaurants respectively: $Ann{:}C \sqsubseteq Ann{:}O$, $Ann{:}T \sqsubseteq Ann{:}O$, $Ann{:}V \sqsubseteq Ann{:}O$

Suppose that the data stored by Ann that she accepts to make available deals with restaurants of various specialties, and only with those rated with 2 stars among the rated restaurants. The extensional classes declared by Ann are then: $Ann{:}ViewS2 \sqsubseteq Ann{:}S2$, $Ann{:}ViewC \sqsubseteq Ann{:}C$, $Ann{:}ViewV \sqsubseteq Ann{:}V$, $Ann{:}ViewT \sqsubseteq Ann{:}T$, $Ann{:}ViewI \sqsubseteq Ann{:}I$

Bob, who is fond of Asian cooking and likes high quality, organizes his restaurant URLs according to the following classes:

• the class $Bob{:}A$ of Asian restaurants

• the class $Bob{:}Q$ of high quality restaurants that he knows

Suppose that he wants to make available every data that he has stored. The extensional classes that he declares are $Bob{:}ViewA$ and $Bob{:}ViewQ$ (as subclasses of $Bob{:}A$ and $Bob{:}Q$): $Bob{:}ViewA \sqsubseteq Bob{:}A$, $Bob{:}ViewQ \sqsubseteq Bob{:}Q$

Chris is more fond of fish restaurants but recently discovered some places serving a very nice cantonese cuisine. He organizes its data with respect to the following classes:

- the class $Chris{:}F$ of fish restaurants,
- the class $Chris{:}CA$ of Cantonese restaurants

Suppose that he declares the extensional classes $Chris{:}ViewF$ and $Chris{:}ViewCA$ as subclasses of $Chris{:}F$ and $Chris{:}CA$ respectively:
$Chris{:}ViewF \sqsubseteq Chris{:}F$, $Chris{:}ViewCA \sqsubseteq Chris{:}CA$

Dora organizes her restaurants URLs around the class $Dora{:}DP$ of her preferred restaurants, among which she distinguishes the subclass $Dora{:}P$ of pizzerias and the subclass $Dora{:}SF$ of seafood restaurants.
Suppose that the only URLs that she stores concerns pizzerias: the only extensional class that she has to declare is $Dora{:}ViewP$ as a subclass of $Dora{:}P$: $Dora{:}ViewP \sqsubseteq Dora{:}P$

Ann, **Bob**, **Chris** and **Dora** express what they know about each other using mappings stating properties of class inclusion or equivalence.

Ann is very confident in Bob's taste and agrees to include Bob' selection as good restaurants by stating $Bob{:}Q \sqsubseteq Ann{:}G$. Finally, she thinks that Bob's Asian restaurants encompass her Oriental restaurant concept: $Ann{:}O \sqsubseteq Bob{:}A$

Bob knows that what he calls Asian cooking corresponds exactly to what Ann classifies as Oriental cooking. This may be expressed using the equivalence statement : $Bob{:}A \equiv Ann{:}O$ (note the difference of perception of Bob and Ann regarding the mappings between $Bob{:}A$ and $Ann{:}O$)

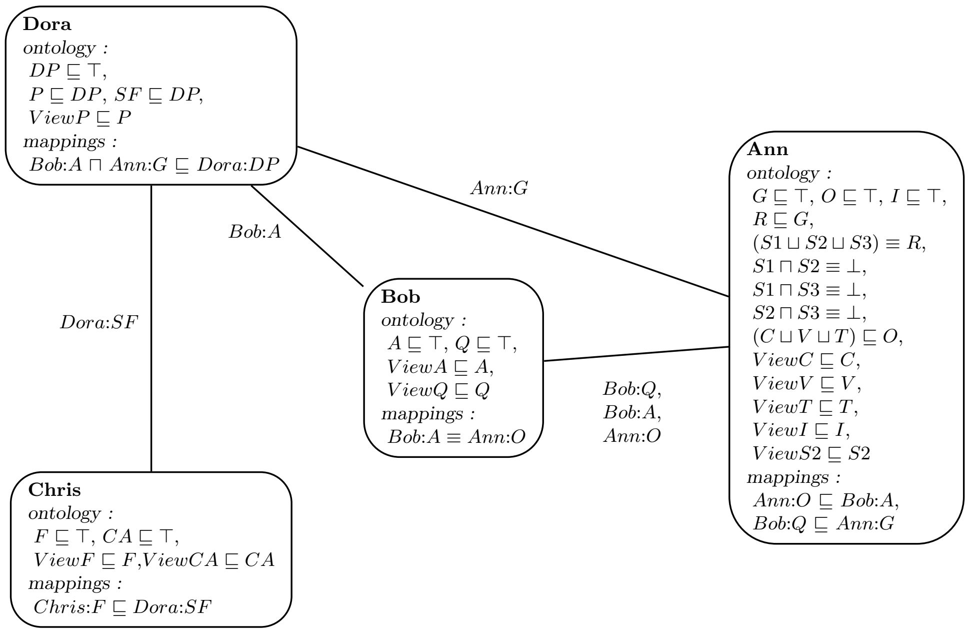
Chris considers that what he calls fish specialties is a particular case of Dora seafood specialties: $Chris{:}F \sqsubseteq Dora{:}SF$

Dora counts on both Ann and Bob to obtain good Asian restaurants : $Bob{:}A \sqcap Ann{:}G \sqsubseteq Dora{:}DP$

Figure 1 describes the resulting overlay network. In order to alleviate the notations, we omit the local peer name prefix except for the mappings. Edges are labeled with the class identifiers that are shared through the mappings between peers.

3 Query Rewriting in SomeWhere through Propositional Encoding

In SomeWhere, each user interrogates the peer-to-peer network through one peer of his choice, and uses the vocabulary of this peer to express his query. Therefore, queries are logical combinations of classes of a given peer ontology.



Dora
ontology :
$DP \sqsubseteq \top$,
$P \sqsubseteq DP$, $SF \sqsubseteq DP$,
$ViewP \sqsubseteq P$
mappings :
$Bob{:}A \sqcap Ann{:}G \sqsubseteq Dora{:}DP$
Ann:G
Bob:A
Dora:SF
Bob
ontology :
$A \sqsubseteq \top$, $Q \sqsubseteq \top$,
$ViewA \sqsubseteq A$,
$ViewQ \sqsubseteq Q$
mappings :
$Bob{:}A \equiv Ann{:}O$
Bob:Q,
Bob:A,
Ann:O
Ann
ontology :
$G \sqsubseteq \top$, $O \sqsubseteq \top$, $I \sqsubseteq \top$,
$R \sqsubseteq G$,
$(S1 \sqcup S2 \sqcup S3) \equiv R$,
$S1 \sqcap S2 \equiv \bot$,
$S1 \sqcap S3 \equiv \bot$,
$S2 \sqcap S3 \equiv \bot$,
$(C \sqcup V \sqcup T) \sqsubseteq O$,
$ViewC \sqsubseteq C$,
$ViewV \sqsubseteq V$,
$ViewT \sqsubseteq T$,
$ViewI \sqsubseteq I$,
$ViewS2 \sqsubseteq S2$
mappings :
$Ann{:}O \sqsubseteq Bob{:}A$,
$Bob{:}Q \sqsubseteq Ann{:}G$
Chris
ontology :
$F \sqsubseteq \top$, $CA \sqsubseteq \top$,
$ViewF \sqsubseteq F$,$ViewCA \sqsubseteq CA$
mappings :
$Chris{:}F \sqsubseteq Dora{:}SF$

Fig. 1. The restaurants network

The corresponding answer sets are expressed in intention in terms of the combinations of extensional classes that are *rewritings* of the query. The point is that extensional classes of several distant peers can participate to the rewritings, and thus to the answer of a query posed to a given peer.

In general, finding all answers in a peer data management system is a critical issue [5]. In our setting however, we are in a case where all the answers can be obtained using *rewritings* of the query: it has been shown [6] that when a query has a *finite number of maximal conjunctive rewritings*, then *all* its answers (a.k.a. certain answers) can be obtained as the union of the answer sets of its rewritings.

In the SomeWhere setting, query rewriting can be equivalently reduced to distributed reasoning over logical propositional clausal theories by a straighforward propositional encoding of the query and of the distributed ontologies and mappings of a SomeWhere network. It consists in transforming the query and each ontology and mapping statement into a propositional formula using class identifiers as propositional variables.

The propositional encoding of a class description D, and thus of a query, is the propositional formula $Prop(D)$ obtained inductively as follows:

- $Prop(\top) = true$, $Prop(\bot) = false$
- $Prop(A) = A$, if A is an atomic class
- $Prop(D_1 \sqcap D_2) = Prop(D_1) \wedge Prop(D_2)$
- $Prop(D_1 \sqcup D_2) = Prop(D_1) \vee Prop(D_2)$

- $Prop(\neg D) = \neg(Prop(D))$

The global schema $\mathcal{S}$ of a SomeWhere peer-to-peer network is the union of the ontology and mapping statements distributed over the network. Its propositional encoding is the distributed propositional theory $Prop(\mathcal{S})$ made of the formulas obtained inductively from the statements in $\mathcal{S}$ as follows:

- $Prop(C \sqsubseteq D) = Prop(C) \Rightarrow Prop(D)$
- $Prop(C \equiv D) = Prop(C) \Leftrightarrow Prop(D)$
- $Prop(C \sqcap D \equiv \bot) = \neg Prop(C) \vee \neg Prop(D)$

We have shown in [3] that the maximal conjunctive rewritings of a query q in SomeWhere are the negation of the prime proper implicates of $\neg q$ w.r.t the propositional encoding of the schema. We use the distributed message-passing DeCA algorithm [3] to compute them.

4 Ongoing work

We are involved in two extensions of SomeWhere. The first one concerns SomeRDFS which extends the very simple class-based data model of SomeWhere to RDFS. The second one deals with possible inconsistencies of the global schema of SomeWhere. Though the data model of SomeWhere is very simple, it allows negation. It is thus very likely that mappings between consistent ontologies cause inconsistencies at the global level. The problem is twofold: the inconsistencies have to be detected at join time ; they must then be handled at query time to compute only *well-founded* answers.

4.1 Extending the data model to RDFS

In SomeRDFS the ontologies and mappings are expressed in the core fragment of RDFS allowing to state (sub)classes, (sub)properties, typing of domain and range of properties. The mappings that we consider in SomeRDFS are inclusion statements between classes or properties of two distinct peers, or typing statements of a property of a given peer with a class of another peer. Therefore, mappings are RDFS statements involving vocabularies of different peers which thus establish semantic correspondances between peers.

We have shown that query answering in SomeRDFS can be achieved using a rewrite and evaluate strategy, and that the corresponding rewriting problem can be reduced to the same consequence finding problem in distributed propositional theories as in [3]. Moreover, the consequence finding problem resulting from the propositional encoding of the fragment of RDFS that we consider is tractable since the resulting propositional theories are reduced to clauses of length 2 for which the reasoning problem is in P. The experiments reported in [4] show that it takes in mean 0.07s to SomeWhere for a complete reasoning on randomly generated sets of clauses of length 2 distributed on 1000 peers.

4.2 Dealing with inconsistencies

Since peer ontologies are personalized they model possibly different viewpoints. Therefore, the global schema made of the union of the local ontologies and the mappings may be inconsistent even if each local ontology is consistent. Given the lack of centralized control, all peers should be treated equally. It would be unfair to refuse the join of a new peer just because the resulting global schema becomes inconsistent. Our choice is to accept the presence of inconsistency. The problem is first to *detect* inconsistencies, second to *reason* in spite of them in a satisfactory way. We thus compute only *well-founded* answers to a given query, i.e., answers that can be computed w.r.t. to a consistent subset of the global schema

We assume each local ontology to be consistent. Therefore, the possible inconsistencies result from interactions between local ontologies and are caused by *mappings*. Before adding a mapping, a peer checks whether this mapping (possibly with other mappings) can be the cause of some inconsistency. In that case, the peer stores locally as a *nogood* the set of mappings involved in the corresponding inconsistency. At query time, the concerned distributed nogoods must be collected to check whether the answers under construction are well-founded: an answer is well-founded if the set of mappings used to infer it is not included in any nogood. This can be done using the distributed algorithms extending DeCA that are designed and studied in [7].

References

1. Rousset, M.C.: Small can be beautiful in the semantic web. In: ISWC. (2004)
2. Plu, M., Bellec, P., Agosto, L., van de Velde, W.: The web of people: A dual view on the WWW. In: WWW. (2003)
3. Adjiman, P., Chatalic, P., Goasdoué, F., Rousset, M.C., Simon, L.: Distributed reasoning in a P2P setting: Application to the semantic web. Journal of Artificial Intelligence Research (JAIR) (2006)
4. Adjiman, P., Chatalic, P., Goasdoué, F., Rousset, M.C., Simon, L.: Scalability study of P2P consequence finding. In: IJCAI. (2005)
5. Halevy, A., Ives, Z., Suciu, D., Tatarinov, I.: Schema mediation in peer data management systems. In: ICDE. (2003)
6. Goasdoué, F., Rousset, M.C.: Answering queries using views: a KRDB perspective for the semantic web. ACM Journal - Transactions on Internet Technology (TOIT) **4**(3) (2004)
7. Chatalic, P., Nguyen, G., M-C.Rousset: Reasoning with inconsistencies in propositional peer-to-peer inference systems. Proceedings of ECAI (2006)