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

Marie-Christine Rousset

Joint work with Philippe Adjiman, Philippe Chatalic, François Goasdoué, Gia-Hien Nguyen, Laurent Simon
How to make semantic approaches scalable to the web?

- A data centered vision of the Semantic Web
  - viewed as a huge semantic and distributed data management system

- SomeWhere
  - a peer to peer infrastructure
  - based on simple personalized ontologies and mappings distributed at large scale
P2P Data Management Systems

- Logical network of peers (≠ physical network)
  - each peer is characterized by
    - its physical address (IP)
    - a description of the stored resources
    - its neighbors in the network
      - the peers to which it can transmit messages (queries,...)

- Various topologies
  - random and dynamic (Gnutella)
  - fixed (Chord, Hypercube)
  - guided by the semantics
    - SON, Edutella, Piazza, DRAGO, coDB, Somewhere
SomeWhere logical networks

- The topology is not fixed
- Guided by mappings
  - A peer
    - joins by declaring mappings between its ontology and the ontologies of some peers that it knows
    - leaves by removing the mappings with its acquaintances in the network
SomeWhere in a nutshell

- Simple data model based on a propositional language of classes
  - for defining ontologies, mappings, a
  - a sublanguage of OWL DL (W3C)

- Scales up to one thousand peers
  - logical network : « small world »
SomeWhere Data Model

Ontology: hierarchy of intentional classes

Storage description: extensional classes

More complex inclusion statement: St\_A₁ ⊆ A₁ \sqcap \neg A₂
SomeWhere Data Model

Mappings:

Logical combination of class literals: $A_1 \sqcap A_2 \sqcap A_3 \sqsubseteq B_3 \sqcap \neg B_3$

$A_1 \sqcap (A_2 \sqcup \neg A_3) \sqsubseteq B_3$
Semantics

- Standard FO logical semantics
  - one single domain of interpretation
  - a distributed set of formulas interpreted in the same way as if they were not distributed
  - in contrast with some other approaches
    - coDB: epistemic logic
    - DRAGO: distributed semantics of DDL or DFOL based on a collection of domains of interpretations

- Our assumption:
  - the objects have a unique URI
  - objects stored at different peers and having the same URI are interpreted as being the same
Data model: example
Query answering: illustration

Rewritings:
- St_Pop_Rock
- St_Francais \(\sqsubseteq\) St_Pop
- St_US \(\sqsubseteq\) St_Pop
- St_Mouv \(\sqsubseteq\) St_Pop
Query rewriting in SomeWhere

- Decomposition of queries/recombination of rewritings
  - only atomic queries are transmitted to peers
    - a complex query is split into atomic queries
  - each solicited peer processes a given atomic query \( q \) and incrementally sends back intentional answers for it
    - (conjunction of) extensional classes that are rewritings of \( q \)
  - intentional answers of different atomic queries resulting from the split of a complex query must be recombined
    - intentional answers can combine extensional classes of different peers

- Can be reduced to a consequence finding problem in distributed clausal propositional theories
  - Ontologies and mappings are encoded as clauses
  - The **maximal conjunctive rewritings** of a conjunctive query \( Q \) correspond exactly to the negation of the clauses that are **proper prime implicates** of the negation of \( Q \) w.r.t. the union of the local theories and the mappings
Query rewriting algorithm

- Message based local algorithm running on each peer
  - query, answer, and termination messages

- Global properties
  - soundness
  - completeness
  - termination (even for cyclic networks)
Flash demo of the SomeWhere Firefox extension the browser, reloaded
Q₁: Thriller?
Q₂: NOT Adult?
Q₃: Thriller AND Comedy?
Rewritings:

- P3: Thriller
- P1: Action  □  P1: Suspense
- P5: Drama
- P6: DramaComedy
- P2: BruceWillis  □  P1: Suspense
Rewritings of Thriller: evaluation

P₃: Thriller

P₁: Action ⊻ P₁: Suspense

P₅: Drama

P₆: DramaComedy

P₂: Bruce Willis ⊻ P₁: Suspense

Rewritings of Thriller:

Local
Navigational
Navigational
Integration
SomeWhere infrastructure

N machines
1 peer per machine

1 machine
N peers

N machines
K peers per machine
SomeWhere infrastructure

Zoom on one machine

100 % JAVA 1.5
somewhere.jar ~ 250 Ko
Scalability experiments [IJCAI 05]

- on randomly generated networks
  - 1000 peers deployed on a cluster of 75 machines
  - small world topology
    - Close to the topology of the web

- peers
  - ontologies
    - random clauses of length 2
  - mappings
    - random clauses of length 2 or 3
Varying topologies

1000 peers Ring, 10 neighbours/peer

P: probability of redirecting an edge

P = 0.01
Model of Watts and Strogatz

P = 0.1
Small world

P = 1
Random graph
Scalability results

Varying parameters
- Number of mappings between peers
- Complexity of mappings
  - Ratio of clauses of length 3 (0%, 20%, 100%)

Timeout: 30 s/query

Depth of query processing
- Small depth (less than 7) even on the hard cases

Time to produce a number of answers
- In 90% cases, the first answer is produced within 2 seconds
- Easy cases (simple mappings):
  - Few answers per query (5 on average)
  - Very fast (less than 0.1s) to compute all the answers without timeouts
- Hard cases (complex and more mappings per edge)
  - Around 1000 answers per query (but > 30% queries not complete: timeouts)
  - Quite fast to obtain them (less than 20s)
Handling inconsistencies

● How to define them?
  - unsatisfiability => derivation of the empty clause
  - empty classes => derivation of unit negative clauses

there exists $A$ such that $A$ is empty in every model: $S \models \neg A$

● How to detect inconsistencies?
  - at each join of a new peer

● How to deal with inconsistencies?
  - avoid them when reasoning
Illustration

Path m1: AIPubli is a subclass of Conf.
Path m0 -> m2: AIPublic is a subclass of Journal.

Conf and Journal are disjoint, therefore AIPublic is necessarily empty.

Inconsistencies are caused by mappings.
P2P detecting of inconsistencies

- AIPubli v 2005
- BDPUbl v 2005
- AIPubli v Theory
- Theory v Article
- Expe v Article
- Theory v Journal
- Theory v Journal
- AIPubli
- Theory v Article
- AIPubli v 2005
- BDPUbl v 2005
- 2005 v Conf

**Propagation of m1:**

\{ ¬AIPubli v Conf;
¬AIPubli v Publi;
¬AIPubli v ¬Journal;
¬BDPUbl v Conf;
¬BDPUbl v Publi;
¬BDPUbl v ¬Journal \}.

No production of unit clause
No inconsistency

**Propagation of m2:**

\{ ¬Theory v Journal;
¬AIPubli v Journal;
.....;
¬AIPubli;
.....;
¬AIPubli v ¬Conf \}.

Production of a unit clause
Inconsistency
\{m1,m2\} is a NoGood stored at P3
Distributed storage of the NoGoods
P2P well-founded reasoning

- **Principle:**
  - avoid the inconsistencies when constructing answers

- **Semantics of « well-founded » answer:**
  - obtained from a consistent subset of formulas

- **Algorithm:**
  - for each answer,
    - build its set of mapping supports and return the set of NoGoods encountered during the reasoning,
    - discard the mapping supports including a NoGood
  - return the answers having a not empty set of mapping supports

It has been presented at ECAI 06
Extending the data model to RDF(S)
- W3C recommendation for describing web resources
- Classes and (binary) relations between objects
- each object is identified by a URI

Triple notation: <resource, property, value>
Relational notation: property(resource, value)
### FOL axiomatization

<table>
<thead>
<tr>
<th>Constructor</th>
<th>FOL axiomatization</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class inclusion</td>
<td>$\forall X (C_1(X) \Rightarrow C_2(X))$</td>
</tr>
<tr>
<td>Property inclusion</td>
<td>$\forall X \forall Y (P_1(X,Y) \Rightarrow P_2(X,Y))$</td>
</tr>
<tr>
<td>Domain typing of a property</td>
<td>$\forall X \forall Y (P(X,Y) \Rightarrow C(X))$</td>
</tr>
<tr>
<td>Range typing of property</td>
<td>$\forall X \forall Y (P(X,Y) \Rightarrow C(Y))$</td>
</tr>
</tbody>
</table>
SomeRDFS: data model

a simple fragment of RDFS
distributed through simple mappings
(using the same constructors)

Q(X,Y): P2.Work(X) ∧ P2.refersTo(X,Y)
Query rewriting in SomeRDFS

- Propositionalisation of the RDFS statements and the query: removing the variables

<table>
<thead>
<tr>
<th>Constructor</th>
<th>FOL axiomatization</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class inclusion</td>
<td>$\forall X (C_1(X) \Rightarrow C_2(X))$</td>
</tr>
<tr>
<td>Property inclusion</td>
<td>$\forall X \forall Y (P_1(X,Y) \Rightarrow P_2(X,Y))$</td>
</tr>
<tr>
<td>Domain typing of a property</td>
<td>$\forall X \forall Y (P(X,Y) \Rightarrow C(X))$</td>
</tr>
<tr>
<td>Range typing of property</td>
<td>$\forall X \forall Y (P(X,Y) \Rightarrow C(Y))$</td>
</tr>
</tbody>
</table>

- Propositional query rewriting using SomeWhere
- Building the relational rewritings by adding the variables at the right place.

To appear in Journal of Data Semantics
Q(X,Y): P2.Work(X) \land P2.refersTo(X,Y)

P2.Work^dom  \rightarrow  SomeWhere rewriting

P2.Work^range  \rightarrow  SomeWhere rewriting

P2.refersTo^rel  \rightarrow  SomeWhere rewriting

P2.Painting^dom  \rightarrow  P2.Painting(X)

P2.Painting(X)  \rightarrow  R1(X,Y): P2.Painting(X) \land P1.belongsTo(X,Y)

P1.Paints^rel  \rightarrow  P1.Paints(Z,X)

P1.belongsTo^rel  \rightarrow  P1.belongsTo(X,Y)
Q(X,Y): P2.Work(X) \land P2.refersTo(X,Y)

P2.Work_{dom} \downarrow

SomeWhere rewriting

P2.Painting_{dom} \quad P2.Painting(X)

P2.Painting_{range} \downarrow

SomeWhere rewriting

P1.Paints_{rel} \quad P1.Paints(Z,X)

P2.refersTo_{rel} \downarrow

SomeWhere rewriting

P1.belongsTo_{rel} \quad P1.belongsTo(X,Y)

R2(X,Y): P1.Paints(Z,X) \land P1.belongsTo(X,Y)
Ongoing work

- Coupling SomeWhere to a DHT for optimizing lookup queries

- Adapting the SomeWhere algorithm to support the epistemic semantics

- Modeling and handling trust in P2P Semantic overlay networks
  - based on a logical approach

- P2P discovery and composition of smart devices
  - based on a semantic description of the functionality, inputs and outputs of devices