Peer-to-peer Query Answering with Inconsistent Knowledge

Arnold Binas and Sheila A. McIlraith

University of Toronto

September 19, 2008
Reasoning with distributed knowledge

- Distributed nature of knowledge on the (Semantic) Web
- Heterogeneous information sources (xml files, DBs, KBs, . . .)
- Mutual inconsistencies
- Different levels of reliability
Work on distributed reasoning

- Partition-based logical reasoning (propositional and FOL) [Amir and McIlraith, 2005]
- Distributed propositional consequence-finding [Adjiman et al., 2006]
  - with inconsistencies: [Chatalic et al., 2006]
- Peer-to-peer data exchange systems [Bertossi and Bravo, 2004]
- Ontology translation and information integration
- Distributed defeasible reasoning in multi-context systems [Bikakis and Antoniou, 2008]
This talk

- Distributed propositional reasoners in peer-to-peer network
This talk

- Distributed *propositional* reasoners in *peer-to-peer* network
- Mutual inconsistencies possible
This talk

- Distributed propositional reasoners in peer-to-peer network
- Mutual inconsistencies possible
- Priority/trust/preference ordering over peers
This talk

- Distributed propositional reasoners in peer-to-peer network
- Mutual inconsistencies possible
- Priority/trust/preference ordering over peers
- Formalization of distributed entailment
This talk

- Distributed propositional reasoners in peer-to-peer network
- Mutual inconsistencies possible
- Priority/trust/preference ordering over peers
- Formalization of distributed entailment
- Distributed query answering algorithm
This talk

- Distributed propositional reasoners in peer-to-peer network
- Mutual inconsistencies possible
- Priority/trust/preference ordering over peers
- Formalization of distributed entailment
- Distributed query answering algorithm
- Pruning strategy and heuristic for drastic speedup
This talk

- Distributed propositional reasoners in peer-to-peer network
- Mutual inconsistencies possible
- Priority/trust/preference ordering over peers
- Formalization of distributed entailment
- Distributed query answering algorithm
- Pruning strategy and heuristic for drastic speedup
- Empirical evaluation
Outline

1. P2P Query Answering Systems
   - Framework
   - Distributed entailment

2. A Distributed Query Answering Algorithm
   - Algorithm
   - Ordering heuristic and pruning
   - Experiments and results

3. Summary & Conclusion
Outline

1. P2P Query Answering Systems
   - Framework
   - Distributed entailment

2. A Distributed Query Answering Algorithm
   - Algorithm
   - Ordering heuristic and pruning
   - Experiments and results

3. Summary & Conclusion
P2P Query Answering Systems—Example

Customer
- YYZ-SYD
- pay_cc
- arrive_morning → book_cruise
- arrive_evening → ¬book_cruise

United Airlines
- YYZ-SYD ∧ pymt_rcvd → arrive_morning
- 5

Visa Card
- pay_cc → pymt_rcvd
- 5

Qantas
- YYZ-SYD ∧ pymt_rcvd → arrive_evening
- 10
A peer is a tuple $P_i = (KB_i, \models_i, I_i)$
- $KB_i$ is the peer’s local knowledge base
- $\models_i$ is the peer’s local consequence relation
- $I_i$ is the peer’s priority level

Small priorities are good, large ones are bad

A peer’s signature $L_i$ is the set of symbols in its $KB_i$
A P2P query answering system (PQAS)

- A PQAS is a tuple $\mathcal{P} = (P, G)$
  - $P = \{P_i\}_{i=1}^n$ is a set of $n$ peers
  - $G = (V, E)$ is a graph connecting the peers
- Edges are labeled: $(i, j, L_{ij})$, where $L_{ij} \subseteq L_i \cap L_j$
Distributed Entailment—Example

Customer

YYZ-SYD
pay_cc
arrive_morning → book_cruise
arrive_evening → ¬book_cruise

{YYZ-SYD, arrive_morning, arrive_evening}

United Airlines

YYZ-SYD ∧ pymt_rcvd → arrive_morning

Visa Card

pay_cc → pymt_rcvd

{pymt_rcvd}

Qantas

YYZ-SYD ∧ pymt_rcvd → arrive_evening

{pymt_rcvd}

Arnold Binas and Sheila A. McIlraith

Peer-to-peer Query Answering with Inconsistent Knowledge
Distributed Entailment—Example

Customer

YYZ-SYD
pay_cc
arrive.morning → book.cruise
arrive_evening → ¬book.cruise

{YYZ-SYD, arrive.morning, arrive_evening}

book.cruise?

United Airlines

YYZ-SYD ∧ pymt_rcvd → arrive.morning

{pymt_rcvd}

Visa Card

pay_cc → pymt_rcvd

{pymt_rcvd}

Qantas

YYZ-SYD ∧ pymt_rcvd → arrive_evening

{pymt_rcvd}

Customer 3

United Airlines 5

Visa Card 5

Qantas 10
Distributed Entailment—Example

Customer

{YYZ-SYD, arrive_morning, arrive_evening}

YYZ-SYD
pay_cc
arrive_morning → book_cruise
arrive_evening → ¬book_cruise

United Airlines

YYZ-SYD ∧ pymt_rcvd → arrive_morning

pymt_rcvd

Visa Card

pay_cc → pymt_rcvd

Qantas

YYZ-SYD ∧ pymt_rcvd → arrive_evening

Arnold Binas and Sheila A. McIlraith Peer-to-peer Query Answering with Inconsistent Knowledge
Distributed Entailment—Example

Customer

YYZ-SYD
pay_cc
arrive_morning → book_cruise
arrive_evening → ¬book_cruise

{YYZ-SYD, arrive_morning, arrive_evening}

United Airlines

YYZ-SYD ∧ pymt_rcvd → arrive_morning

{pymt_rcvd}

Visa Card

pay_cc → pymt_rcvd

{pymt_rcvd}

Qantas

YYZ-SYD ∧ pymt_rcvd → arrive_evening

{pymt_rcvd}
Reasons (similar to [Benferhat et al., 1999])

A *reason* for a formula $\phi$ is a consistent set $\Sigma$ of formulas in $KB_1 \cup \ldots \cup KB_n$ that derives $\phi$ given the peers’ local consequence relations while respecting the link languages between peers.
Arguments (similar to [Dung, 1995])

An argument is a pair $A = (\Sigma, \phi)$ s.t. $\Sigma$ is a reason for $\phi$. Its rank $R(A)$ is the priority of the worst-priority formula in $\Sigma$. 

$A = (\Sigma, \phi)$

- $\Sigma$ is a reason for $\phi$.
- Its rank $R(A)$ is the priority of the worst-priority formula in $\Sigma$. 

$\neg$book_cruise  book_cruise
Attacking arguments (similar to [Dung, 1995])

$B$ attacks $A$ iff the conclusion of $B$ is inconsistent with the reason of $A$ and $B$ is of better or equal rank.

$B$ attacks $A \iff \phi_B \cup \Sigma_A \models \bot$ and $\mathcal{R}(B) \leq \mathcal{R}(A)$
Argumentation frameworks ([Dung, 1995])

The argumentation framework induced by a PQAS $\mathcal{P}$ is a tuple $AF(\mathcal{P}) = (AR, attacks)$
Conflict-free sets of arguments ([Dung, 1995])

A set $S$ of arguments is conflict-free if no argument in $S$ is attacked by another argument in $S$. 
Acceptable arguments ([Dung, 1995])

An argument $A$ is acceptable wrt. an argument set $S$ iff for each argument $B$, if $B$ attacks $A$ then $B$ is attacked by some argument in $S$. 

\[ A \rightarrow B \rightarrow C \rightarrow E \rightarrow D \]

\[ F \rightarrow G \rightarrow H \]
Admissible sets ([Dung, 1995])

A conflict-free set of arguments $S$ is admissible iff each argument in $S$ is acceptable wrt. $S$.

- consistent
- defends itself against all attacks from the outside
Preferred extensions ([Dung, 1995])

A preferred extension of an argumentation framework $AF$ is a maximal admissible set of $AF$. 
A formula $\phi$ is entailed by the PQAS $\mathcal{P}$ by distributed entailment, denoted $\mathcal{P} \models_D \phi$, iff $\phi$ is the conclusion of an argument in some preferred extension of $AF(\mathcal{P})$. 
Outline

1 P2P Query Answering Systems
   - Framework
   - Distributed entailment

2 A Distributed Query Answering Algorithm
   - Algorithm
   - Ordering heuristic and pruning
   - Experiments and results

3 Summary & Conclusion
Distributed query answering: a hard problem

- Complete propositional inference at each peer
- Consistency checks of candidate reasons
- Arguments must be checked for being attacked
Distributed query answering algorithm

- Assumption: local consequence relations are classical entailment
- Pose both CNF query and its negation for refutation proofs
- Search for arguments for and against query
- Recursively check their being defeated
- Collect ranks of arguments
- Return query answer according to best-rank undefeated argument (in preferred extension)
Consequence-finding algorithm

- Refutation proofs are by consequence finding
- Algorithm modifies and extends distributed consequence-finding algorithm of [Adjiman et al., 2006]
- Negate query and search for contradiction
- Find local consequences and solicit neighboring peers for theirs (recurse)
- Messages types
  - request consequences of a literal
  - report found consequences
  - report termination of a reasoning branch
- Consequences from neighboring peers are returned and merged
- Empty clause consequence indicates contradiction
The algorithm

- Keeps track of original clauses leading to a consequence
- Checks consistency of clause sets $\rightarrow$ reasons
- Keeps track of priority of consequences $\rightarrow$ ranks of arguments
Algorithm—Example

Customer

YYZ-SYD
pay_cc
arrive_morning → book_cruise
arrive_evening → ¬book_cruise

United Airlines

YYZ-SYD ∧ pymt_rcvd → arrive.morning

Visa Card

pay_cc → pymt_rcvd

Qantas

YYZ-SYD ∧ pymt_rcvd → arrive_evening
Algorithm—Example

**Customer**
- YZ-SYD
- pay_cc
- arrive_morning → book_cruise
- arrive_evening → \neg book_cruise

**United Airlines**
- \( YZ-SYD \land pymt_rcvd \rightarrow arrive.morning \)
- \( YZ-SYD \land pymt_rcvd \rightarrow arrive.morning \)

**Visa Card**
- \( pay_cc \rightarrow pymt_rcvd \)

**Qantas**
- \( YZ-SYD \land pymt_rcvd \rightarrow arrive.evening \)

**Algorithm**
- Ordering heuristic and pruning
- Experiments and results

Arnold Binas and Sheila A. McIlraith
Peer-to-peer Query Answering with Inconsistent Knowledge
Algorithm—Example

Customer

- YYZ-SYD
- book_cruise
- pay_cc
- arrive_morning → book_cruise
- arrive_evening → ¬book_cruise

United Airlines

- YYZ-SYD ∧ pymt_rcvd → arrive_morning

Visa Card

- pay_cc → pymt_rcvd

Qantas

- YYZ-SYD ∧ pymt_rcvd → arrive_evening
Algorithm—Example

Customer

- YYZ-SYD
- pay_cc
- arrive_morning → book_cruise
- arrive_evening → ¬book_cruise

United Airlines

- YYZ-SYD ∧ pymt_rcvd → arrive_morning

Visa Card

- ¬pymt_rcvd
- pay_cc → pymt_rcvd

Qantas

- YYZ-SYD ∧ pymt_rcvd → arrive_evening
Algorithm—Example

Customer

- YZ-SYD
- pay_cc
- arrive_morning → book_cruise
- arrive_evening → ¬book_cruise

Visa Card

- YZ-SYD ∧ pymt_rcvd → arrive_morning
- ¬pymt_rcvd
- pay_cc → pymt_rcvd
- ¬pay_cc

Qantas

- YZ-SYD ∧ pymt_rcvd → arrive_evening

United Airlines

- YZ-SYD
- ¬arrive_morning

Arnold Binas and Sheila A. McIlraith
Peer-to-peer Query Answering with Inconsistent Knowledge
Algorithm—Example

Customer

- YYZ-SYD
- pay_cc
- arrive_morning → book_cruise
- arrive_evening → ¬book_cruise

United Airlines

- YYZ-SYD ∧ pymt_rcvd → arrive_morning
- ¬pymt_rcvd

Visa Card

- pay_cc → pymt_rcvd
- ¬pay_cc

Qantas

- YYZ-SYD ∧ pymt_rcvd → arrive_evening

Arnold Binas and Sheila A. McIlraith
Algorithm—Example

Customer

- YYZ-SYD
- ~arrive_morning
- pay_cc
- ~book_cruise
- arrive_morning → book_cruise
- arrive_evening → ~book_cruise

United Airlines

- YYZ-SYD ∧ pymt_rcvd → arrive_morning

Visa Card

- ~pymt_rcvd
- ~pay_cc
- pay_cc → pymt_rcvd

Qantas

- YYZ-SYD ∧ pymt_rcvd → arrive_evening
Algorithm—Example

Customer

YYZ-SYD
¬book_cruise
pay_cc
arrive_morning → book_cruise
arrive_evening → ¬book_cruise

United Airlines

YYZ-SYD ∧ pymt_rcvd → arrive_morning

Visa Card

¬pymt_rcvd
¬pay_cc
pay_cc → pymt_rcvd

Qantas

YYZ-SYD ∧ pymt_rcvd → arrive_evening
Theorem—properties of the algorithm

**Termination**

The algorithm terminates.

**Soundness and completeness**

The algorithm is sound and complete wrt. distributed entailment.
The search space

Recurse to:

- Find consequences in neighboring peers
- Check arguments for being attacked
Pruning

- Message passing algorithms keep track of best-rank currently known, so-far undefeated argument
- Messages and peers of worse rank are pruned from search space for more efficiency
- Does not affect soundness and completeness result
Ordering heuristic

- Process messages and peers in order of priority
- Leads to better-rank arguments being found earlier and thus more effective pruning
Implementation

- Simplified implementation of algorithm (no attack checks)
- Simulate entire PQAS on a single machine
Experiments

- Randomly generated PQASs
- Each proposition asked as a query
- Compared performance of naive algorithm, pruning, and pruning with ordering heuristic
- Performance criterion is number of messages passed between peers (network bottleneck in real-world system)
- Max 10 minutes per query
### Experiment Results

<table>
<thead>
<tr>
<th></th>
<th>naive</th>
<th>pruning</th>
<th>ordering + pruning</th>
</tr>
</thead>
<tbody>
<tr>
<td>queries solved</td>
<td>1341</td>
<td>2292</td>
<td>2842</td>
</tr>
<tr>
<td>avg. required messages</td>
<td>2735</td>
<td>1266</td>
<td>456</td>
</tr>
</tbody>
</table>

#### Pruning vs. No Pruning

![Pruning vs. No Pruning](image1)

#### Pruning vs. Pruning with Ordering Heuristic

![Pruning vs. Pruning with Ordering Heuristic](image2)
## Experiment Results

<table>
<thead>
<tr>
<th></th>
<th>naive</th>
<th>pruning</th>
<th>ordering + pruning</th>
</tr>
</thead>
<tbody>
<tr>
<td>queries solved</td>
<td>1341</td>
<td>2292</td>
<td>2842</td>
</tr>
<tr>
<td>avrg. required messages</td>
<td>2735</td>
<td>1266</td>
<td>456</td>
</tr>
</tbody>
</table>

### Pruning vs. No Pruning

![Pruning vs. No Pruning](image1)

- **x-axis**: Problem Instances
- **y-axis**: Number Messages
- **Lines**: no pruning (blue), pruning (green)

### Pruning vs. Pruning with Ordering Heuristic

![Pruning vs. Pruning with Ordering Heuristic](image2)

- **x-axis**: Problem Instances
- **y-axis**: Number Messages
- **Lines**: pruning (blue), pruning + ordering heuristic (green)
## Experiment Results

<table>
<thead>
<tr>
<th></th>
<th>naive</th>
<th>pruning</th>
<th>ordering + pruning</th>
</tr>
</thead>
<tbody>
<tr>
<td>queries solved</td>
<td>1341</td>
<td>2292</td>
<td>2842</td>
</tr>
<tr>
<td>avrg. required messages</td>
<td>2735</td>
<td>1266</td>
<td>456</td>
</tr>
</tbody>
</table>

### Pruning vs. No Pruning

![Pruning vs. No Pruning Graph](image1)

### Pruning vs. Pruning with Ordering Heuristic

![Pruning vs. Pruning with Ordering Heuristic Graph](image2)
Outline

1. P2P Query Answering Systems
   - Framework
   - Distributed entailment

2. A Distributed Query Answering Algorithm
   - Algorithm
   - Ordering heuristic and pruning
   - Experiments and results

3. Summary & Conclusion
Summary

- Peer-to-peer query answering system (PQAS)
- Distributed entailment relation based on argumentation systems
- Sound and complete distributed query answering algorithm
- Pruning and ordering heuristic for efficiency
- Experimental evaluation
Future work

- Richer priority specification and aggregation schemes
- Algorithm for non-classical local consequence relations (e.g. NAF)
- Information integration by ontology translation between peers
- Richer language
