Data Conflict Resolution Using Trust Mappings

Wolfgang Gatterbauer & Dan Suciu
University of Washington, Seattle

June 8, Sigmod 2010

Project web page: http://db.cs.washington.edu/beliefDB
Conflicts & Trust mappings in Community DBs

Background 1: Conflicting beliefs

<table>
<thead>
<tr>
<th>glyph</th>
<th>origin</th>
</tr>
</thead>
<tbody>
<tr>
<td>��(goal)</td>
<td>ship hull</td>
</tr>
<tr>
<td>الحوثي (goal)</td>
<td>cow</td>
</tr>
<tr>
<td>��(goal)</td>
<td>jar</td>
</tr>
<tr>
<td>有关部门 (official)</td>
<td>fish</td>
</tr>
<tr>
<td>有关部门 (official)</td>
<td>knot</td>
</tr>
<tr>
<td>有关部门 (official)</td>
<td>arrow</td>
</tr>
</tbody>
</table>

“Beliefs”: annotated (key,value) pairs

Background 2: Trust mappings

<table>
<thead>
<tr>
<th>glyph</th>
<th>origin</th>
</tr>
</thead>
<tbody>
<tr>
<td>��(goal)</td>
<td>ship hull</td>
</tr>
<tr>
<td>有关部门 (official)</td>
<td>fish</td>
</tr>
<tr>
<td>有关部门 (official)</td>
<td>arrow</td>
</tr>
</tbody>
</table>

“Explicit belief”

Priorities

“Implicit belief”

Recent work on community databases:

Orchestra [SIGMOD’06, VLDB’07]
Youtopia [VLDB’09], BeliefDB [VLDB’09]

Problems due to transient effects

1. Incorrect inserts
   – Value depends on order of inserts

Alice would have preferred Bob’s value over Charlie’s
**Problems due to transient effects**

1. Incorrect inserts
   – Value depends on order of inserts

2. Incorrect updates
   – Mis-handling of revokes

---

Alice and Bob trust each other most, but have lost “justification” for their beliefs

This paper:

**Automatic conflict resolution with trust mappings:**
1. How to define a globally consistent solution?
2. How to calculate it efficiently?
3. Some extensions
Agenda

1. Stable solutions
   – how to define a unique and consistent solution?

2. Resolution algorithm
   – how to calculate the solution efficiently?

3. Extensions
   – how to deal with “negative beliefs”?
Stable solutions

• Priority trust network (TN)
  – assume a fixed key
  – users (nodes): A, B, C
  – values (beliefs): v, w, u
  – trust mappings (arcs) from “parents”

• Stable solution
  – assignment of values to each node*,
    s.t. each belief has a “non-dominated lineage” to an explicit belief

* each node with at least one ancestor with explicit belief
Stable solutions

- Priority trust network (TN)
  - assume a fixed key
  - users (nodes): A, B, C
  - values (beliefs): v, w, u
  - trust mappings (arcs) from “parents”

- Stable solution
  - assignment of values to each node*, s.t. each belief has a “non-dominated lineage” to an explicit belief

*A: v

B: w

C: v

D: v

Lineage

SS1 = (A:v, B:w, C:v, D:v)

* each node with at least one ancestor with explicit belief
**Stable solutions**

- **Priority trust network (TN)**
  - assume a fixed key
  - users (nodes): A, B, C
  - values (beliefs): v, w, u
  - trust mappings (arcs) from “parents”

- **Stable solution**
  - assignment of values to each node*, s.t. each belief has a “*non-dominated lineage*” to an explicit belief

- **Possible / Certain semantics**
  - a stable solution determines, for each node, a possible value (“poss”)
  - certain value (“cert”) = intersection of all stable solutions, per user

* each node with at least one ancestor with explicit belief
Stable solutions

- Priority trust network (TN)
  - assume a fixed key
  - users (nodes): A, B, C
  - values (beliefs): v, w, u
  - trust mappings (arcs) from “parents”

- Stable solution
  - assignment of values to each node*, s.t. each belief has a “non-dominated lineage” to an explicit belief

- Possible / Certain semantics
  - a stable solution determines, for each node, a possible value (“poss”)”
  - certain value (“cert”) = intersection of all stable solutions, per user

* each node with at least one ancestor with explicit belief

Parent “B:w (10)” dominates and is inconsistent with “E:u (5)”

Is this a stable solution?

SS1=(A:v, B:w, C:v, D:v, E:u)
SS2=(A:v, B:w, C:w, D:w, E:u)
SS3=(A:v, B:w, C:u, D:u, E:u)

No!

<table>
<thead>
<tr>
<th>X</th>
<th>poss(X)</th>
<th>cert(X)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>{v}</td>
<td>{v}</td>
</tr>
<tr>
<td>B</td>
<td>{w}</td>
<td>{w}</td>
</tr>
<tr>
<td>C</td>
<td>{v,w}</td>
<td>∅</td>
</tr>
<tr>
<td>D</td>
<td>{v,w}</td>
<td>∅</td>
</tr>
<tr>
<td>E</td>
<td>{u}</td>
<td>{u}</td>
</tr>
</tbody>
</table>

Now how to calculate poss / cert?
Logic programs (LP) with stable model semantics

LP & Stable model semantics

• “Declarative imperative”*

• Natural correspondence
  Brave (credulous) reasoning
  ~ possible tuple semantics
  Cautious (skeptical) reasoning
  ~ certain tuple semantics

• Previous work on consistent query answering & peer data exchange

But solving LPs is hard 😞

![Graph showing time vs. size of the network]

How can we calculate poss / cert efficiently?

* keynote Joe Hellerstein
** size of the network = users + mappings; simple network of several “oscillators” (see paper)
1. Stable solutions
   – how to define a unique and consistent solution?

2. Resolution algorithm
   – how to calculate the solution efficiently?

3. Extensions
   – how to deal with “negative beliefs”?
Resolution Algorithm

Focus on binary trust network

- Keep 2 sets: closed / open
  Initialize closed with explicit beliefs

```
Initalize closed with explicit beliefs
 Keep 2 sets: closed / open
```

<table>
<thead>
<tr>
<th>X</th>
<th>poss(X)</th>
<th>cert(X)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>{v}</td>
<td>{v}</td>
</tr>
<tr>
<td>B</td>
<td>{w}</td>
<td>{w}</td>
</tr>
<tr>
<td>C</td>
<td>{u}</td>
<td>{u}</td>
</tr>
<tr>
<td>D</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>E</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>F</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>G</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>H</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>J</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>K</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>L</td>
<td>?</td>
<td>?</td>
</tr>
</tbody>
</table>
Resolution Algorithm

- Keep 2 sets: closed / open
  Initialize closed with explicit beliefs
- MAIN
  
  **Step 1**: if ∃ preferred edges from open to closed
  → follow

<table>
<thead>
<tr>
<th>X</th>
<th>poss(X)</th>
<th>cert(X)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>{v}</td>
<td>{v}</td>
</tr>
<tr>
<td>B</td>
<td>{w}</td>
<td>{w}</td>
</tr>
<tr>
<td>C</td>
<td>{u}</td>
<td>{u}</td>
</tr>
<tr>
<td>D</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>E</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>F</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>G</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>H</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>J</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>K</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>L</td>
<td>?</td>
<td>?</td>
</tr>
</tbody>
</table>
Resolution Algorithm

- Keep 2 sets: closed / open
  - Initialize closed with explicit beliefs
- MAIN
  - Step 1: if ∃ preferred edges from open to closed
    → follow

<table>
<thead>
<tr>
<th>X</th>
<th>poss(X)</th>
<th>cert(X)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>{v}</td>
<td>{v}</td>
</tr>
<tr>
<td>B</td>
<td>{w}</td>
<td>{w}</td>
</tr>
<tr>
<td>C</td>
<td>{u}</td>
<td>{u}</td>
</tr>
<tr>
<td>D</td>
<td>{v}</td>
<td>{v}</td>
</tr>
<tr>
<td>E</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>F</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>G</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>H</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>J</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>K</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>L</td>
<td>?</td>
<td>?</td>
</tr>
</tbody>
</table>
Resolution Algorithm

• Keep 2 sets: closed / open
  Initialize closed with explicit beliefs
• MAIN
  Step 1: if ∃ preferred edges from open to closed
  → follow

<table>
<thead>
<tr>
<th>X</th>
<th>poss(X)</th>
<th>cert(X)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>{v}</td>
<td>{v}</td>
</tr>
<tr>
<td>B</td>
<td>{w}</td>
<td>{w}</td>
</tr>
<tr>
<td>C</td>
<td>{u}</td>
<td>{u}</td>
</tr>
<tr>
<td>D</td>
<td>{v}</td>
<td>{v}</td>
</tr>
<tr>
<td>E</td>
<td>{w}</td>
<td>{w}</td>
</tr>
<tr>
<td>F</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>G</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>H</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>J</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>K</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>L</td>
<td>?</td>
<td>?</td>
</tr>
</tbody>
</table>
Resolution Algorithm

- Keep 2 sets: closed / open
  Initialize closed with explicit beliefs
- MAIN
  
  **Step 1**: if ∃ preferred edges from open to closed
  → follow

<table>
<thead>
<tr>
<th>$X$</th>
<th>poss($X$)</th>
<th>cert($X$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A$</td>
<td>${v}$</td>
<td>${v}$</td>
</tr>
<tr>
<td>$B$</td>
<td>${w}$</td>
<td>${w}$</td>
</tr>
<tr>
<td>$C$</td>
<td>${u}$</td>
<td>${u}$</td>
</tr>
<tr>
<td>$D$</td>
<td>${v}$</td>
<td>${v}$</td>
</tr>
<tr>
<td>$E$</td>
<td>${w}$</td>
<td>${w}$</td>
</tr>
<tr>
<td>$F$</td>
<td>${u}$</td>
<td>${u}$</td>
</tr>
<tr>
<td>$G$</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>$H$</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>$J$</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>$K$</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>$L$</td>
<td>?</td>
<td>?</td>
</tr>
</tbody>
</table>
Resolution Algorithm

- Keep 2 sets: closed / open
- Initialize closed with explicit beliefs
- MAIN
  - Step 1: if ∃ preferred edges from open to closed
    - follow

<table>
<thead>
<tr>
<th>X</th>
<th>poss(X)</th>
<th>cert(X)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>{v}</td>
<td>{v}</td>
</tr>
<tr>
<td>B</td>
<td>{w}</td>
<td>{w}</td>
</tr>
<tr>
<td>C</td>
<td>{u}</td>
<td>{u}</td>
</tr>
<tr>
<td>D</td>
<td>{v}</td>
<td>{v}</td>
</tr>
<tr>
<td>E</td>
<td>{w}</td>
<td>{w}</td>
</tr>
<tr>
<td>F</td>
<td>{u}</td>
<td>{u}</td>
</tr>
<tr>
<td>G</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>H</td>
<td>{w}</td>
<td>{w}</td>
</tr>
<tr>
<td>J</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>K</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>L</td>
<td>?</td>
<td>?</td>
</tr>
</tbody>
</table>
Resolution Algorithm

• Keep 2 sets: closed / open
  Initialize closed with explicit beliefs
• MAIN
  Step 1: if ∃ preferred edges from open to closed
    → follow
  Step 2: else
    → construct SCC graph of open

For every cyclic or acyclic directed graph:
- The Strongly Connected Components graph is a DAG
- can be calculated in $O(n)$ [Tarjan [1972]]

<table>
<thead>
<tr>
<th>$X$</th>
<th>poss($X$)</th>
<th>cert($X$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A$</td>
<td>{v}</td>
<td>{v}</td>
</tr>
<tr>
<td>$B$</td>
<td>{w}</td>
<td>{w}</td>
</tr>
<tr>
<td>$C$</td>
<td>{u}</td>
<td>{u}</td>
</tr>
<tr>
<td>$D$</td>
<td>{v}</td>
<td>{v}</td>
</tr>
<tr>
<td>$E$</td>
<td>{w}</td>
<td>{w}</td>
</tr>
<tr>
<td>$F$</td>
<td>{u}</td>
<td>{u}</td>
</tr>
<tr>
<td>$G$</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>$H$</td>
<td>{w}</td>
<td>{w}</td>
</tr>
<tr>
<td>$J$</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>$K$</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>$L$</td>
<td>?</td>
<td>?</td>
</tr>
</tbody>
</table>
Resolution Algorithm

- Keep 2 sets: closed / open
  Initialize closed with explicit beliefs
- MAIN
  Step 1: if ∃ preferred edges from open to closed
    → follow
  Step 2: else
    → construct SCC graph of open
    → resolve minimum SCCs

<table>
<thead>
<tr>
<th>X</th>
<th>poss(X)</th>
<th>cert(X)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>{v}</td>
<td>{v}</td>
</tr>
<tr>
<td>B</td>
<td>{w}</td>
<td>{w}</td>
</tr>
<tr>
<td>C</td>
<td>{u}</td>
<td>{u}</td>
</tr>
<tr>
<td>D</td>
<td>{v}</td>
<td>{v}</td>
</tr>
<tr>
<td>E</td>
<td>{w}</td>
<td>{w}</td>
</tr>
<tr>
<td>F</td>
<td>{u}</td>
<td>{u}</td>
</tr>
<tr>
<td>G</td>
<td>{v,w}</td>
<td>∅</td>
</tr>
<tr>
<td>H</td>
<td>{w}</td>
<td>{w}</td>
</tr>
<tr>
<td>J</td>
<td>{v,w}</td>
<td>∅</td>
</tr>
<tr>
<td>K</td>
<td>{v,w}</td>
<td>∅</td>
</tr>
<tr>
<td>L</td>
<td>?</td>
<td>?</td>
</tr>
</tbody>
</table>

For every cyclic or acyclic directed graph:
- The Strongly Connected Components graph is a DAG
- can be calculated in $O(n)$ \textbf{Tarjan [1972]}
Resolution Algorithm

- Keep 2 sets: closed / open
  - Initialize closed with explicit beliefs
- MAIN
  - **Step 1**: if ∃ preferred edges from open to closed → follow
  - **Step 2**: else → construct SCC graph of open → resolve minimum SCCs

<table>
<thead>
<tr>
<th>X</th>
<th>poss(X)</th>
<th>cert(X)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>{v}</td>
<td>{v}</td>
</tr>
<tr>
<td>B</td>
<td>{w}</td>
<td>{w}</td>
</tr>
<tr>
<td>C</td>
<td>{u}</td>
<td>{u}</td>
</tr>
<tr>
<td>D</td>
<td>{v}</td>
<td>{v}</td>
</tr>
<tr>
<td>E</td>
<td>{w}</td>
<td>{w}</td>
</tr>
<tr>
<td>F</td>
<td>{u}</td>
<td>{u}</td>
</tr>
<tr>
<td>G</td>
<td>{v,w}</td>
<td>∅</td>
</tr>
<tr>
<td>H</td>
<td>{w}</td>
<td>{w}</td>
</tr>
<tr>
<td>J</td>
<td>{v,w}</td>
<td>∅</td>
</tr>
<tr>
<td>K</td>
<td>{v,w}</td>
<td>∅</td>
</tr>
<tr>
<td>L</td>
<td>?</td>
<td>?</td>
</tr>
</tbody>
</table>
Resolution Algorithm

- Keep 2 sets: closed / open
  Initialize closed with explicit beliefs
- MAIN
  
  **Step 1:** if ∃ preferred edges from open to closed
  → follow
  
  **Step 2:** else
  → construct SCC graph of open
  → resolve minimum SCCs

<table>
<thead>
<tr>
<th>X</th>
<th>poss(X)</th>
<th>cert(X)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>{v}</td>
<td>{v}</td>
</tr>
<tr>
<td>B</td>
<td>{w}</td>
<td>{w}</td>
</tr>
<tr>
<td>C</td>
<td>{u}</td>
<td>{u}</td>
</tr>
<tr>
<td>D</td>
<td>{v}</td>
<td>{v}</td>
</tr>
<tr>
<td>E</td>
<td>{w}</td>
<td>{w}</td>
</tr>
<tr>
<td>F</td>
<td>{u}</td>
<td>{u}</td>
</tr>
<tr>
<td>G</td>
<td>{v,w}</td>
<td>∅</td>
</tr>
<tr>
<td>H</td>
<td>{w}</td>
<td>{w}</td>
</tr>
<tr>
<td>J</td>
<td>{v,w}</td>
<td>∅</td>
</tr>
<tr>
<td>K</td>
<td>{v,w}</td>
<td>∅</td>
</tr>
<tr>
<td>L</td>
<td>{v,w,u}</td>
<td>∅</td>
</tr>
</tbody>
</table>

PTIME resolution algorithm
O(n^2) worst case
O(n) on reasonable graphs
Agenda

1. Stable solutions
   – how to define a unique and consistent solution?

2. Resolution algorithm
   – how to calculate the solution efficiently?

3. Extensions
   – how to deal with “negative beliefs”?
3 semantics for negative beliefs

Agnostic

Eclectic

Skeptic

w/o cycles*  O(n)  O(n)

w cycles  NP-hard**  NP-hard**  O(n)  O(n^2)

* assuming total order on parents for each node
** checking if a belief is possible at a give node is NP-hard, checking if it is certain is co-NP-hard

Our recommendation  with a variation of resolution algorithm
Take-aways automatic conflict resolution

Problem
• Given explicit beliefs & trust mappings, how to assign consistent value assignment to users?

Our solution
• Stable solutions with possible/certain value semantics
• PTIME algorithm \([O(n^2)] \) worst case, \([O(n)] \) experiments
• Several extensions
  – negative beliefs: 3 semantics, two hard, one \([O(n^2)] \)
  – bulk inserts
  – agreement checking
  – consensus value
  – lineage computation

Please visit us at the poster session Th, 3:30pm
or at: http://db.cs.washington.edu/beliefDB
poster
1. Conflicts & Trust mappings in Community DBs

Background 1: Conflicting beliefs*

<table>
<thead>
<tr>
<th>glyph</th>
<th>origin</th>
</tr>
</thead>
<tbody>
<tr>
<td>∪₁</td>
<td>ship hull</td>
</tr>
<tr>
<td>∪₁</td>
<td>cow</td>
</tr>
<tr>
<td>∪₁</td>
<td>jar</td>
</tr>
<tr>
<td>/apt₂</td>
<td>fish</td>
</tr>
<tr>
<td>/apt₂</td>
<td>knot</td>
</tr>
<tr>
<td>/apt₃</td>
<td>arrow</td>
</tr>
</tbody>
</table>

“Beliefs”: annotated (key,value) pairs

Bob ➔ Alice (100)
Bob ➔ Charlie (80)
Alice ➔ Bob (100)
Alice ➔ Charlie (50)
Bob ➔ Alice (80)

Priorities

<table>
<thead>
<tr>
<th>glyph</th>
<th>origin</th>
</tr>
</thead>
<tbody>
<tr>
<td>∪₁</td>
<td>cow</td>
</tr>
<tr>
<td>/apt₂</td>
<td>fish</td>
</tr>
<tr>
<td>/apt₃</td>
<td>arrow</td>
</tr>
</tbody>
</table>

“Explicit belief”

Background 2: Trust mappings

Alice ➔ Bob (100)
Alice ➔ Charlie (50)
Bob ➔ Alice (80)

“Implicit belief”

Recent work on community databases:

Orchestra [SIGMOD'06, VLDB'07]
Youtopia [VLDB'09], BeliefDB [VLDB’09]

How to unambiguously assign beliefs to all users?

2. Stable solutions

- **Priority trust network (TN)**
  - assume a fixed key
  - users (nodes): A, B, C
  - values (beliefs): v, w, u
  - trust mappings (arcs) from “parents”

- **Stable solution**
  - assignment of values to each node*, s.t. each belief has a “non-dominated lineage” to an explicit belief

- **Possible / Certain semantics**
  - a stable solution determines, for each node, a possible value (“poss”)
  - certain value (“cert”) = intersection of all stable solutions, per user

* each node with at least one ancestor with explicit belief

\[
\begin{align*}
\text{SS1} &= (A:v, B:w, C:v, D:v, E:u) \\
\text{SS2} &= (A:v, B:w, C:w, D:w, E:u)
\end{align*}
\]
3. Logic programs with stable model semantics

Step 1: Binarization

Step 2: Logic program

1: accept all **poss** of preferred parent

- \( P(C,x) \leftarrow P(A,x) \)
- \( F(C,B,y) \leftarrow P(B,y), P(C,x), x \neq y \)
- \( P(C,y) \leftarrow P(B,y), \neg F(C,B,y) \)

2: accept **poss** from non-preferred parent, that are not conflicting with an existing value

- \( F(C,A,y) \leftarrow P(A,y), P(C,x), x \neq y \)
- \( P(C,y) \leftarrow P(A,y), \neg F(C,A,y) \)
- \( F(C,B,y) \leftarrow P(B,y), P(C,x), x \neq y \)
- \( P(C,y) \leftarrow P(B,y), \neg F(C,B,y) \)
4. Resolution Algorithm (1/2)

- Keep 2 sets: closed / open
  - Initialize closed with explicit beliefs
- MAIN
  - Step 1: if there exist preferred edges from open to closed
    - follow

<table>
<thead>
<tr>
<th>$X$</th>
<th>poss($X$)</th>
<th>cert($X$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A$</td>
<td>{v}</td>
<td>{v}</td>
</tr>
<tr>
<td>$B$</td>
<td>{w}</td>
<td>{w}</td>
</tr>
<tr>
<td>$C$</td>
<td>{u}</td>
<td>{u}</td>
</tr>
<tr>
<td>$D$</td>
<td>{v}</td>
<td>{v}</td>
</tr>
<tr>
<td>$E$</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>$F$</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>$G$</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>$H$</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>$J$</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>$K$</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>$L$</td>
<td>?</td>
<td>?</td>
</tr>
</tbody>
</table>
5. Resolution Algorithm (2/2)

- Keep 2 sets: closed / open
  Initialize closed with explicit beliefs
- MAIN

  Step 1: if ∃ preferred edges from open to closed
  → follow

  Step 2: else
  → construct SCC graph of open
  → resolve minimum SCCs

<table>
<thead>
<tr>
<th></th>
<th>poss(X)</th>
<th>cert(X)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>{v}</td>
<td>{v}</td>
</tr>
<tr>
<td>B</td>
<td>{w}</td>
<td>{w}</td>
</tr>
<tr>
<td>C</td>
<td>{u}</td>
<td>{u}</td>
</tr>
<tr>
<td>D</td>
<td>{v}</td>
<td>{v}</td>
</tr>
<tr>
<td>E</td>
<td>{w}</td>
<td>{w}</td>
</tr>
<tr>
<td>F</td>
<td>{u}</td>
<td>{u}</td>
</tr>
<tr>
<td>G</td>
<td>{v, w}</td>
<td>∅</td>
</tr>
<tr>
<td>H</td>
<td>{w}</td>
<td>{w}</td>
</tr>
<tr>
<td>J</td>
<td>{v, w}</td>
<td>∅</td>
</tr>
<tr>
<td>K</td>
<td>{v, w}</td>
<td>∅</td>
</tr>
<tr>
<td>L</td>
<td>?</td>
<td>?</td>
</tr>
</tbody>
</table>

PTIME resolution algorithm
O(n^2) worst case
O(n) on reasonable graphs

Tarjan [1972]
6. Detail: Strongly Connected Components (SCCs)

For every cyclic or acyclic directed graph:
- The Strongly Connected Components graph is a DAG
- can be calculated in $O(n)$

“Minimal SCCs”: no incoming edge from other SCC = root node(s) in SCC graph

Tarjan [1972]
7. Experiments on large network data

Calculating \textbf{poss / cert} for fixed key
- \textbf{DLV}: State-of-the-art logic programming solver
- \textbf{RA}: Resolution algorithm

Network 1: “Oscillators”

Network 2: “Web link data”
Web data set with 5.4m links between
270k domain names. Approach:
• Sample links with increasing ratio
• Include both nodes in sample
• Assign explicit beliefs randomly

Network 3: “Worst case” $O(n^2)$
8. Three semantics for negative beliefs

**Agnostic**

- $\{v^+\}$
- $\{w^-\}$
- $\{v^-\}$
- $\{w^+\}$
- $\{v^-\}$
- $\{w^+\}$
- $\{v^-, w^-\}$
- $\{v^-, w^+\}$
- $\{v^-, w^-\}$
- $\{v^+, v^-, w^-\}$
- $\{v^+, v^-, w^+\}$
- $\{v^+, v^-, w^-, w^\}$
- $\{v^+, v^-, w^-\}$
- $\{v^+, v^-, w^+, w^-\}$

**Eclectic**

- $\{v^+\}$
- $\{w^-\}$
- $\{v^-\}$
- $\{w^+\}$
- $\{v^-\}$
- $\{w^+\}$
- $\{v^-, w^-\}$
- $\{v^-, w^+\}$
- $\{v^-, w^-\}$
- $\{v^+, v^-, w^-\}$
- $\{v^+, v^-, w^+\}$
- $\{v^+, v^-, w^-, w^\}$
- $\{v^+, v^-, w^-\}$
- $\{v^+, v^-, w^+, w^-\}$

**Skeptic**

- $\{v^+\}$
- $\{w^-\}$
- $\{v^-\}$
- $\{w^+\}$
- $\{v^-\}$
- $\{w^+\}$
- $\{v^-, w^-\}$
- $\{v^-, w^+\}$
- $\{v^-, w^-\}$
- $\{v^+, v^-, w^-\}$
- $\{v^+, v^-, w^+\}$
- $\{v^+, v^-, w^-, w^\}$
- $\{v^+, v^-, w^-\}$
- $\{v^+, v^-, w^+, w^-\}$

w/o cycles* O(n)
w cycles NP-hard**

O(n)

NP-hard**

O(n)

NP-hard**

O(n^2)

* assuming total order on parents for each node
** checking if a belief is possible at a give node is NP-hard, checking if it is certain is co-NP-hard

Our recommendation with a variation of resolution algorithm
9. Take-aways automatic conflict resolution

Problem
- Given explicit beliefs & trust mappings, how to assign consistent value assignment to users?

Our solution
- Stable solutions with possible/certain value semantics
- PTIME algorithm [$O(n^2)$ worst case, $O(n)$ experiments]
- Several extensions
  - negative beliefs: 3 semantics, two hard, one $O(n^2)$
  - bulk inserts
  - agreement checking
  - consensus value
  - lineage computation

Slides soon available on our project page:
http://db.cs.washington.edu/beliefDB
backup
Binarization for Resolution Algorithm*

Example Trust Network (TN)
6 nodes, 9 arcs (size 15)
3 explicit beliefs: A:v, B:w, C:u

Corresponding Binary TN (BTN)
8 nodes, 12 arcs (size 20)

Size increase: ≤ 3

* Note that binarization is not necessary, but greatly simplifies the presentation
Stable solutions: example 2

- Priority trust network (TN)
  - assume a fixed key
  - users (nodes): A, B, C
  - values (beliefs): v, w, u
  - trust mappings (arcs) from “parents”

- Stable solution
  - assignment of values to each node*, s.t. each belief has a “non-dominated lineage” to an explicit belief

- Certain values
  - all stable solution determine, for each node, a possible value (“poss”)
  - certain value (“cert”) = intersection of all stable solutions

* each node with at least one ancestor with explicit belief
Stable solutions: example 2

- Priority trust network (TN)
  - assume a fixed key
  - users (nodes): A, B, C
  - values (beliefs): v, w, u
  - trust mappings (arcs) from “parents”

- Stable solution
  - assignment of values to each node*,
    s.t. each belief has a “non-dominated lineage” to an explicit belief

- Certain values
  - all stable solution determine, for each node, a possible value (“poss”)
  - certain value (“cert”) = intersection of all stable solutions

\[ \text{poss}(G) = \{v, \ldots\} \]

* each node with at least one ancestor with explicit belief
Stable solutions: example 2

- **Priority trust network (TN)**
  - assume a fixed key
  - users (nodes): A, B, C
  - values (beliefs): v, w, u
  - trust mappings (arcs) from “parents”

- **Stable solution**
  - assignment of values to each node*, s.t. each belief has a “non-dominated lineage” to an explicit belief

- **Certain values**
  - all stable solution determine, for each node, a possible value (“poss”)
  - certain value (“cert”) = intersection of all stable solutions

\[
\text{poss}(G) = \{v, w, \ldots\}
\]

* each node with at least one ancestor with explicit belief
Stable solutions: example 2

- Priority trust network (TN)
  - assume a fixed key
  - users (nodes): A, B, C
  - values (beliefs): v, w, u
  - trust mappings (arcs) from “parents”

- Stable solution
  - assignment of values to each node*, s.t. each belief has a “non-dominated lineage” to an explicit belief

- Certain values
  - all stable solution determine, for each node, a possible value (“poss”)
  - certain value (“cert”) = intersection of all stable solutions

\* each node with at least one ancestor with explicit belief
$O(n^2)$-worst-case for Resolution Algorithm