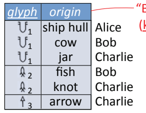
Data Conflict Resolution Using Trust Mappings

Wolfgang Gatterbauer
Dan Suciu

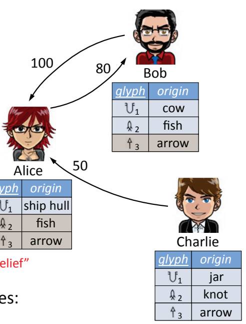
University of Washington
Computer Science and Engineering
Seattle, WA, USA

1. Conflicts & Trust mappings in Community DBs

Background 1: Conflicting beliefs*



"Beliefs": annotated (key,value) pairs



Background 2: Trust mappings

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

Priorities

Recent work on community databases:

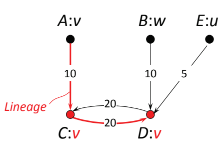
Orchestra [SIGMOD'06, VLDB'07]
Youtopia [VLDB'09], BeliefDB [VLDB'09]

How to unambiguously assign beliefs to all users?

* Current state of knowledge on the Indus Script: Rao et al., Science 324(5931):1165, May 2009

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

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

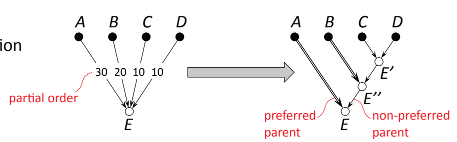
- Possible / Certain semantics
  - a stable solution determines, for each node, a possible value ("**poss**")
  - certain value ("**cert**") = intersection of all stable solutions, per user

X	poss(*X*)	cert(*X*)
A	{*v*}	{*v*}
B	{*w*}	{*w*}
C	{*v*,*w*}	∅
D	{*v*,*w*}	∅
E	{*u*}	{*u*}

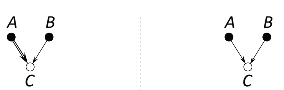
* each node with at least one ancestor with explicit belief

3. Logic programs with stable model semantics

Step 1:
Binarization



Step 2:
Logic program



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

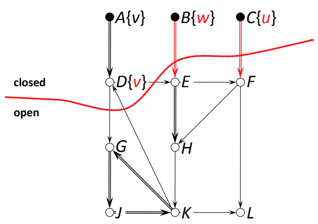
P(C,x) ← P(A,x)
F(C,B,y) ← P(B,y), P(C,x), x≠y
P(C,y) ← P(B,y), ¬F(C,B,y)

F(C,A,y) ← P(A,y), P(C,x), x≠y
P(C,y) ← P(A,y), ¬F(C,A,y)
F(C,B,y) ← P(B,y), P(C,x), x≠y
P(C,y) ← P(B,y), ¬F(C,B,y)

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

4. Resolution Algorithm (1/2)

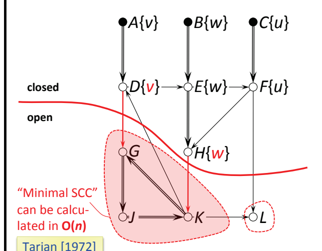
- Keep 2 sets: **closed** / **open**
  Initialize **closed** with explicit beliefs
- MAIN
  Step 1: if ∃ preferred edges from **open to closed**
  → follow



X	poss(*X*)	cert(*X*)
A	{*v*}	{*v*}
B	{*w*}	{*w*}
C	{*u*}	{*u*}
D	{*v*}	{*v*}
E	?	?
F	?	?
G	?	?
H	?	?
J	?	?
K	?	?
L	?	?

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



"Minimal SCC" can be calculated in **O(*n*)**

Tarjan [1972]

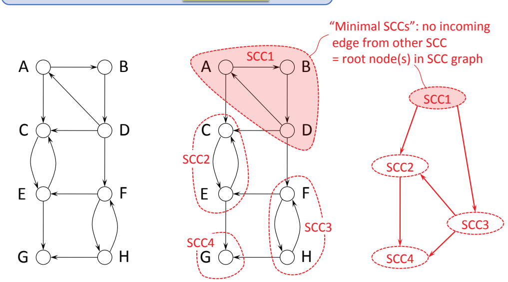
PTIME resolution algorithm
O(*n*²) worst case
O(*n*) on reasonable graphs

X	poss(*X*)	cert(*X*)
A	{*v*}	{*v*}
B	{*w*}	{*w*}
C	{*u*}	{*u*}
D	{*v*}	{*v*}
E	{*w*}	{*w*}
F	{*u*}	{*u*}
G	{*v*,*w*}	∅
H	{*w*}	{*w*}
J	{*v*,*w*}	∅
K	{*v*,*w*}	∅
L	?	?

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*)** Tarjan [1972]

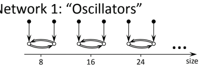
"Minimal SCCs": no incoming edge from other SCC = root node(s) in SCC graph



7. Experiments on large network data

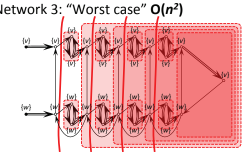
Calculating **poss** / **cert** for fixed key
- **DLV**: State-of-the art logic programming solver
- **RA**: Resolution algorithm

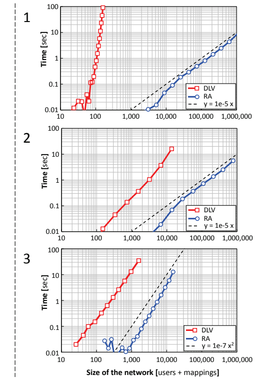
Network 1: "Oscillators"



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

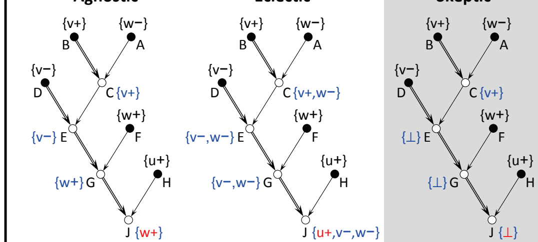
Network 3: "Worst case" **O(*n*²)**





8. Three semantics for negative beliefs

Our recommendation



	Agnostic	Eclectic	Skeptic
w/o cycles*	O(n)	O(n)	O(n)
w cycles	NP-hard**	NP-hard**	O(n²)

with a variation of resolution algorithm

* 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

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(n²)** worst case, **O(n)** experiments]
- Several extensions
  - negative beliefs: 3 semantics, two hard, one **O(n²)**
  - bulk inserts
  - agreement checking
  - consensus value
  - lineage computation

not covered in the talk

Slides soon available on our project page:
http://db.cs.washington.edu/beliefDB