Data Conflict Resolution Using Trust Mappings

Wolfgang Gatterbauer & Dan Suciu
University of Washington, Seattle

June 8, Sigmod 2010

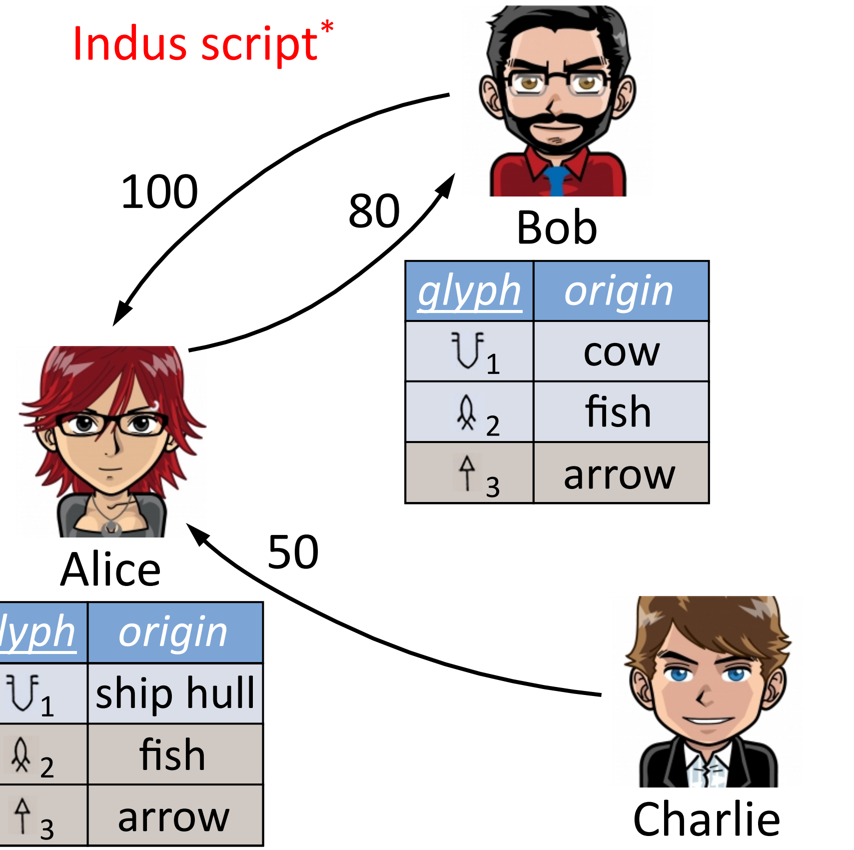
Project web page: http://db.cs.washington.edu/beliefDB

Conflicts & Trust mappings in Community DBs

Background 1: Conflicting beliefs

glyph	origin	
1	ship hull	Alice
1	cow	Bob
1	jar	Charlie
2	fish	Bob
2	knot	Charlie
3	arrow	Charlie

"Beliefs": annotated (key,value) pairs



Bob:

glyph	origin
1	cow
2	fish
3	arrow

Alice:

glyph	origin
1	ship hull
2	fish
3	arrow

Charlie:

glyph	origin
1	jar
2	knot
3	arrow

Background 2: Trust mappings

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

Priorities

Recent work on community databases:

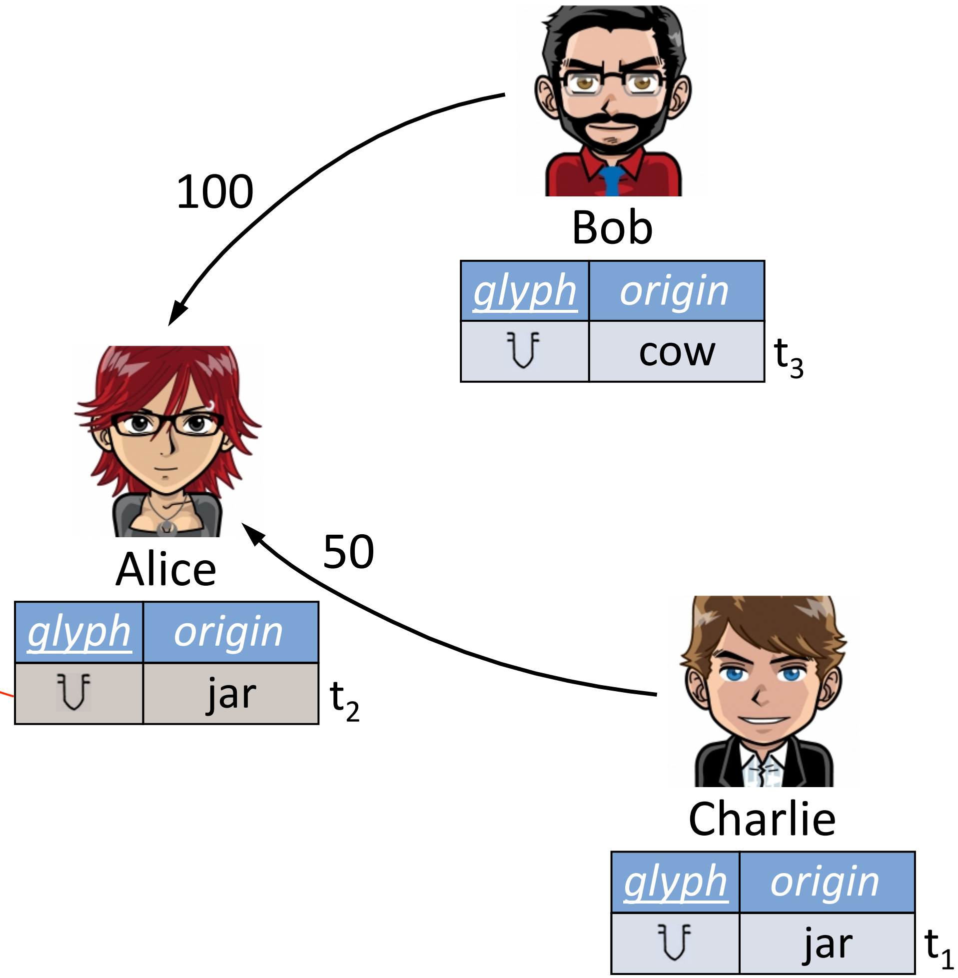
Orchestra [SIGMOD'06, VLDB'07]
Youtopia [VLDB'09], BeliefDB [VLDB'09]

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

Problems due to transient effects

1. Incorrect inserts

– Value depends on order of inserts



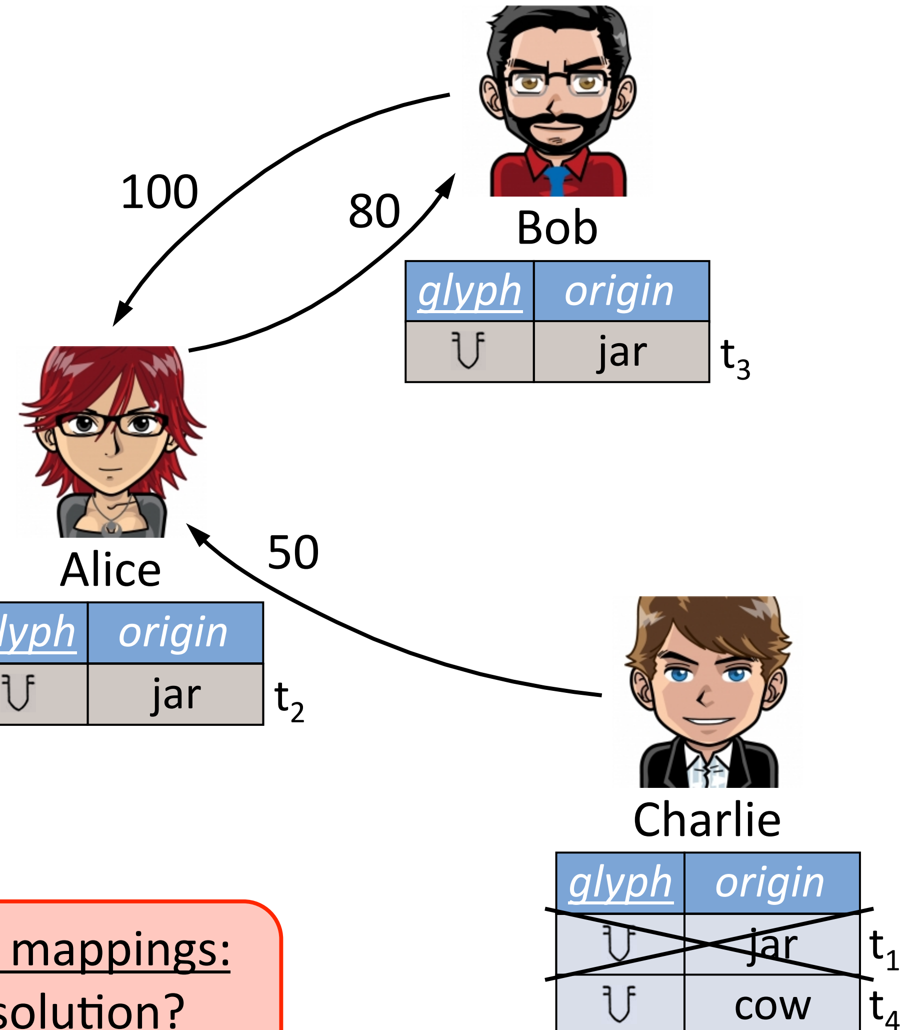
Alice would have preferred Bob's value over Charlie's

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:

<u>Automatic conflict resolution with trust mappings:</u>

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?

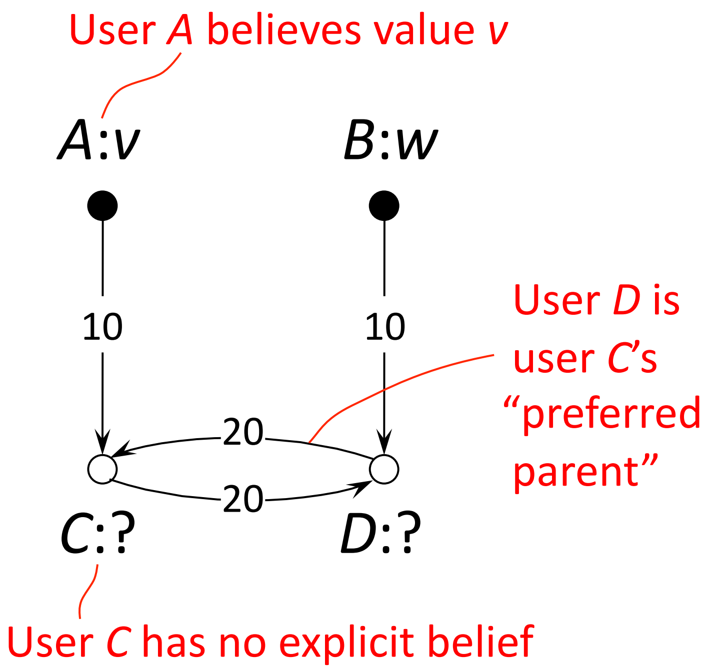
2. Resolution algorithm
   – how to calculate the solution efficiently?

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

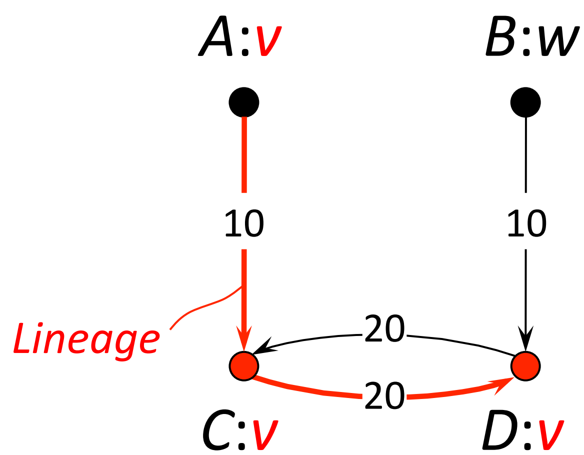


* each node with at least one ancestor with explicit belief

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



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

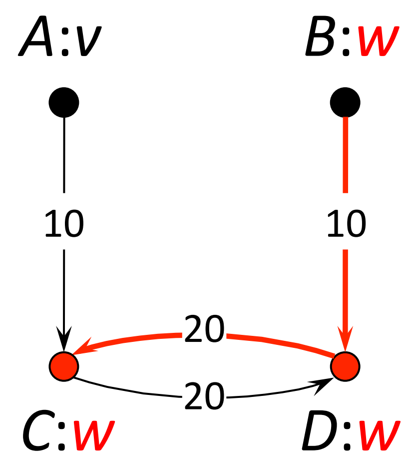
* each node with at least one ancestor with explicit belief

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



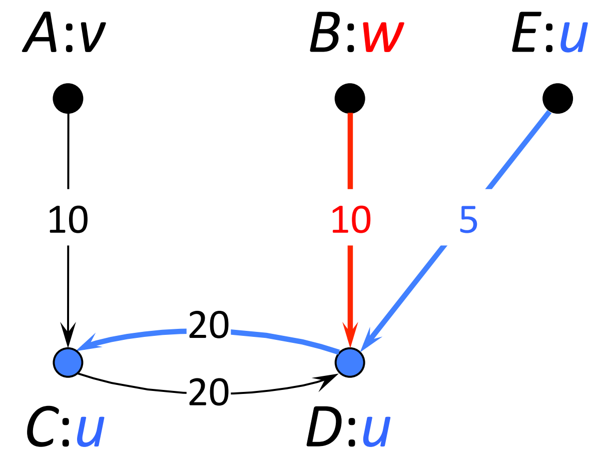
SS1=(*A*:*v*, *B*:*w*, *C*:*v*, *D*:*v*)
SS2=(*A*:*v*, *B*:*w*, *C*:*w*, *D*:*w*)

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

* each node with at least one ancestor with explicit belief

Stable solutions

Parent "*B*:*w* (10)" dominates
and is inconsistent with "*E*:*u* (5)"



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

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!

- 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

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

Now how to calculate **poss** / **cert** ?

* each node with at least one ancestor with explicit belief

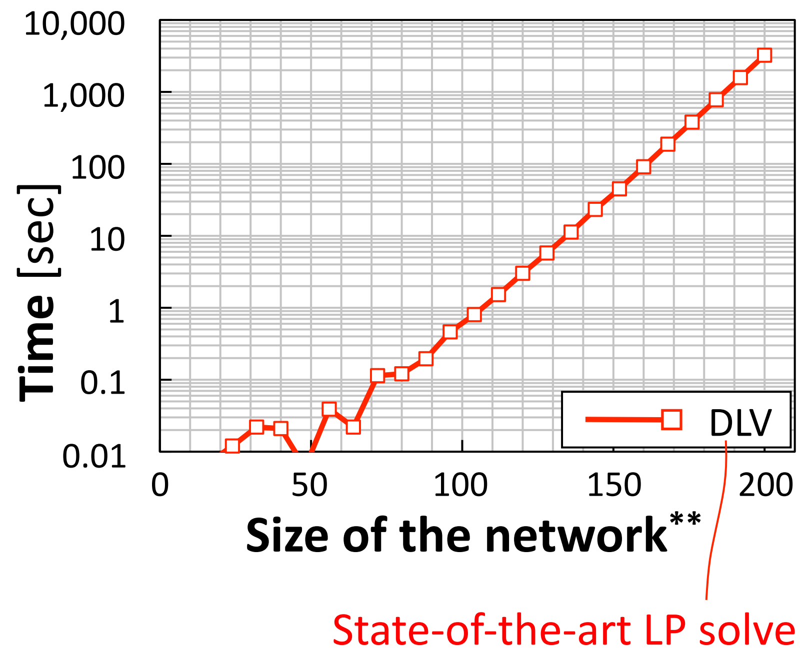
Logic programs (LP) with stable model semantics

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

Greco et al. [TKDE'03]
Arenas et al. [TLP'03]
Barcelo, Bertossi [PADL'03]
Bertossi, Bravo [LPAR'07]

But solving LPs is **hard** ☹



How can we calculate **poss / cert** efficiently?

* keynote Joe Hellerstein
** size of the network = users + mappings; simple network of several "osciallators" (see paper)

Agenda

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

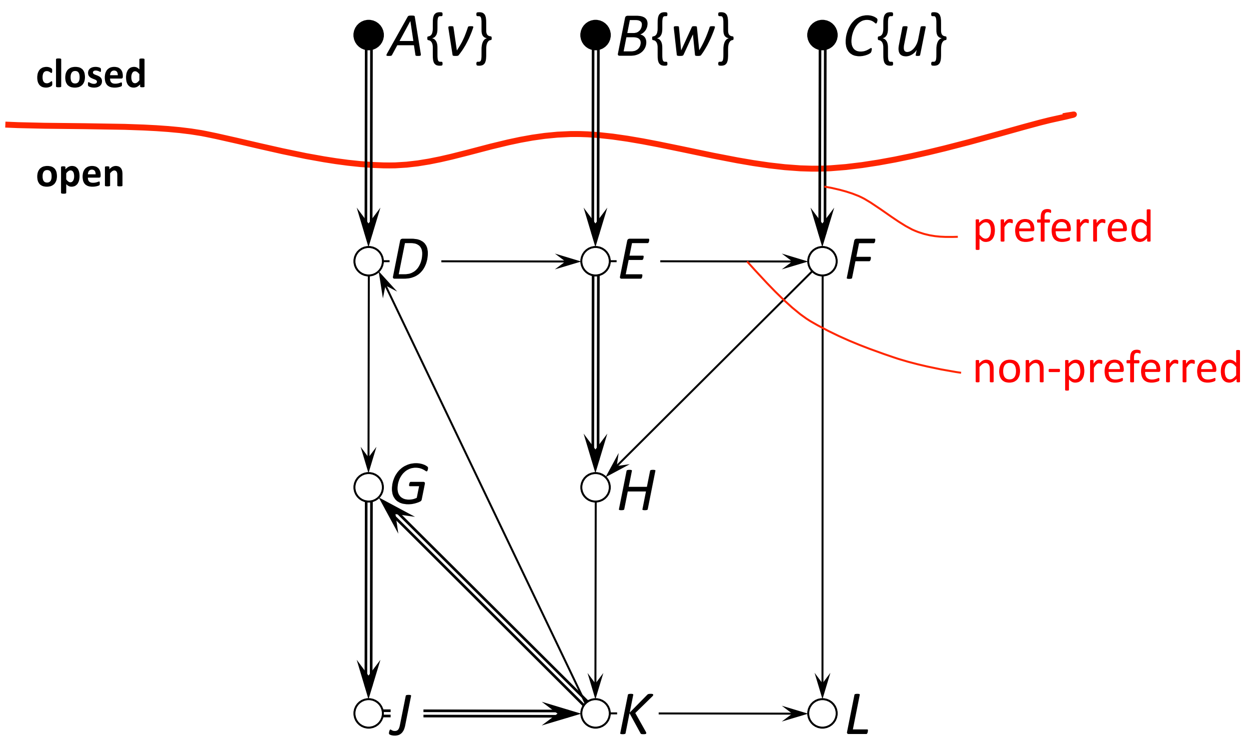
2. Resolution algorithm
   – how to calculate the solution efficiently?

3. Extensions
   – how to deal with "negative beliefs"?

Resolution Algorithm

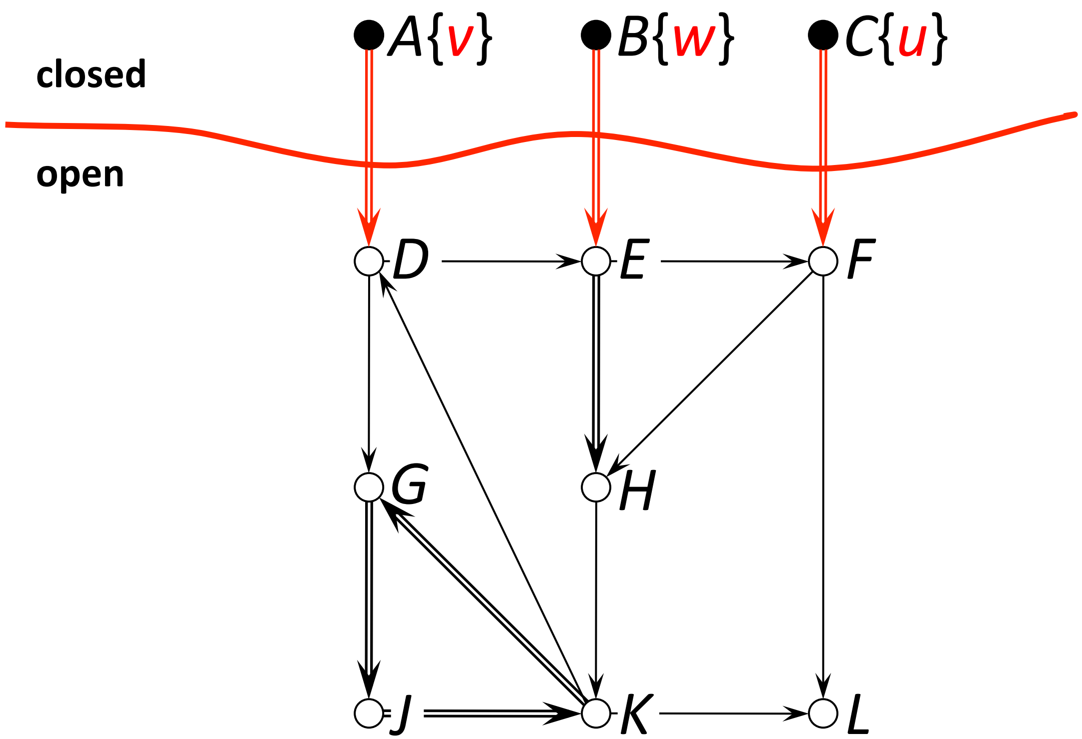
Focus on binary trust network

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



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

Resolution Algorithm

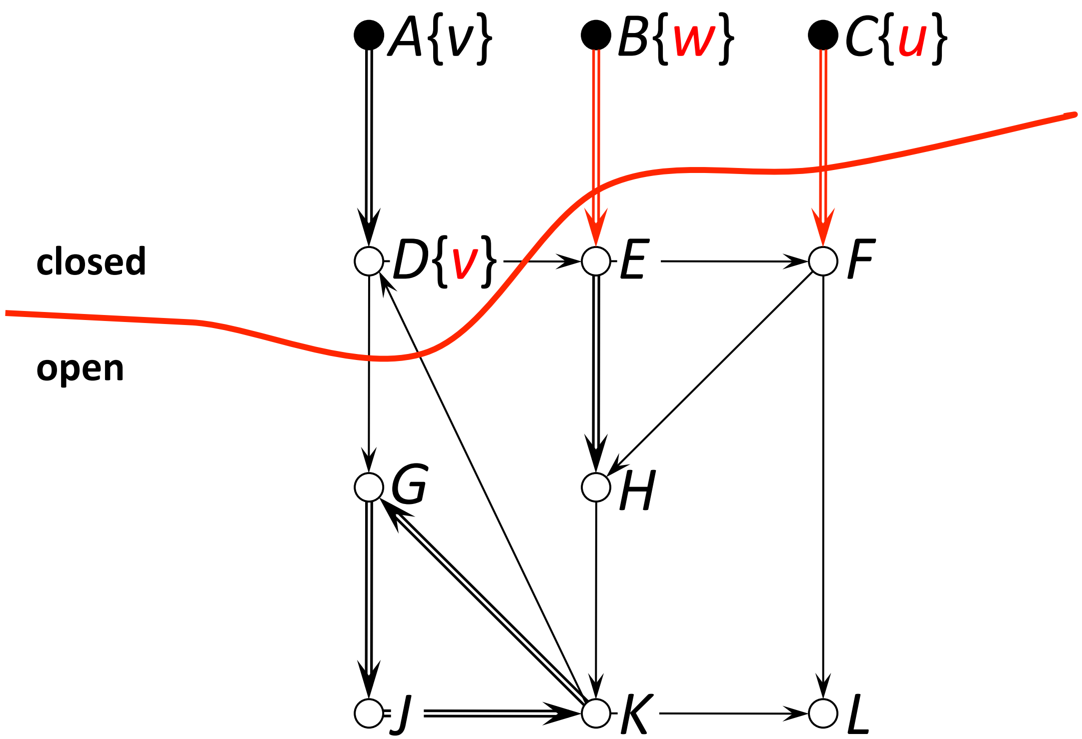


- Keep 2 sets: **closed** / **open**
  Initialize **closed** with explicit beliefs
- MAIN
  Step 1: if ∃ preferred edges from **open** to **closed**
  → follow

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

Resolution Algorithm

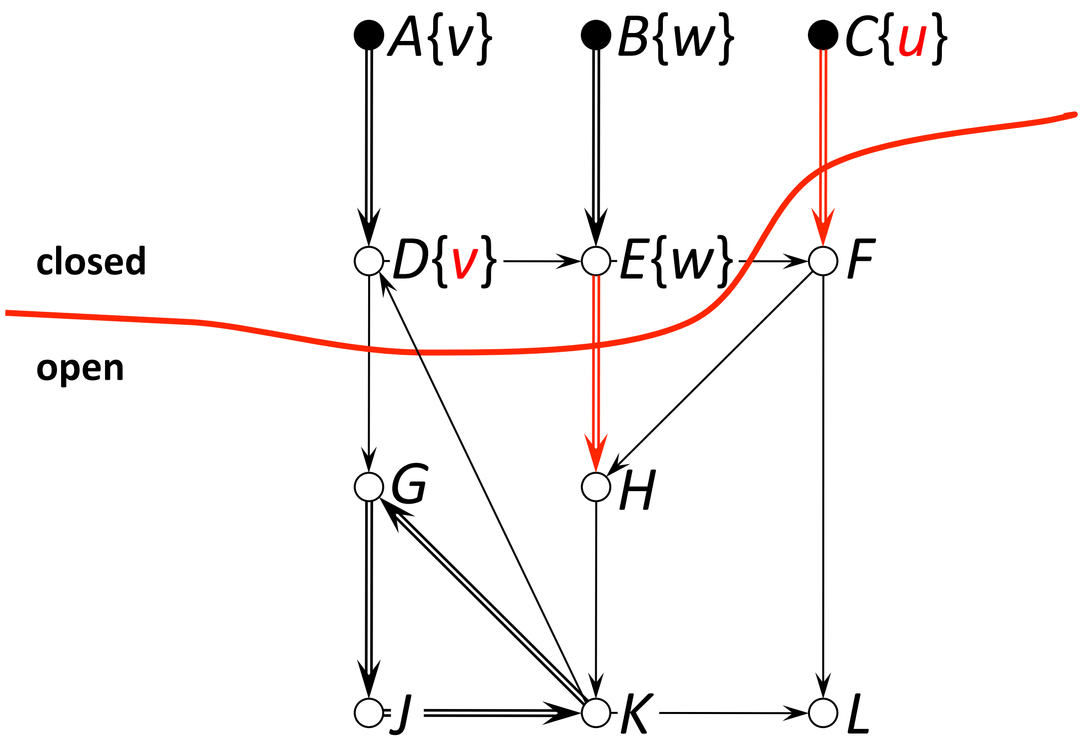
- Keep 2 sets: **closed** / **open**
  Initialize **closed** with explicit beliefs
- MAIN
  - Step 1: if ∃ preferred edges from **open** to **closed** → follow



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

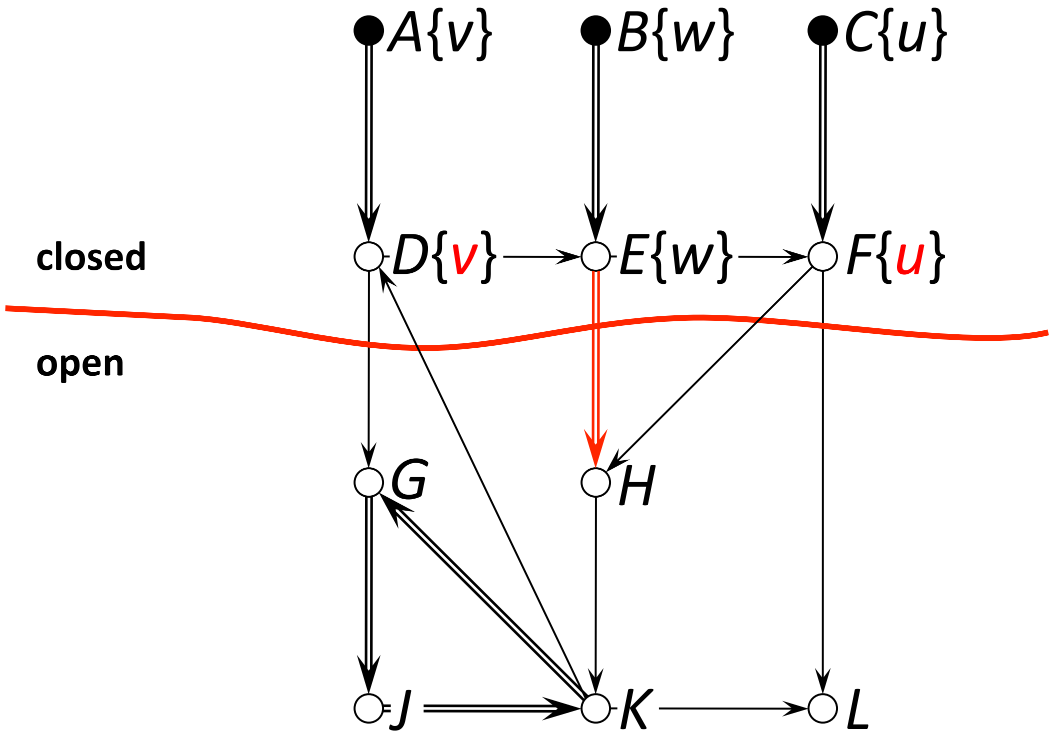
Resolution Algorithm

- 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	$\{w\}$	$\{w\}$
F	?	?
G	?	?
H	?	?
J	?	?
K	?	?
L	?	?

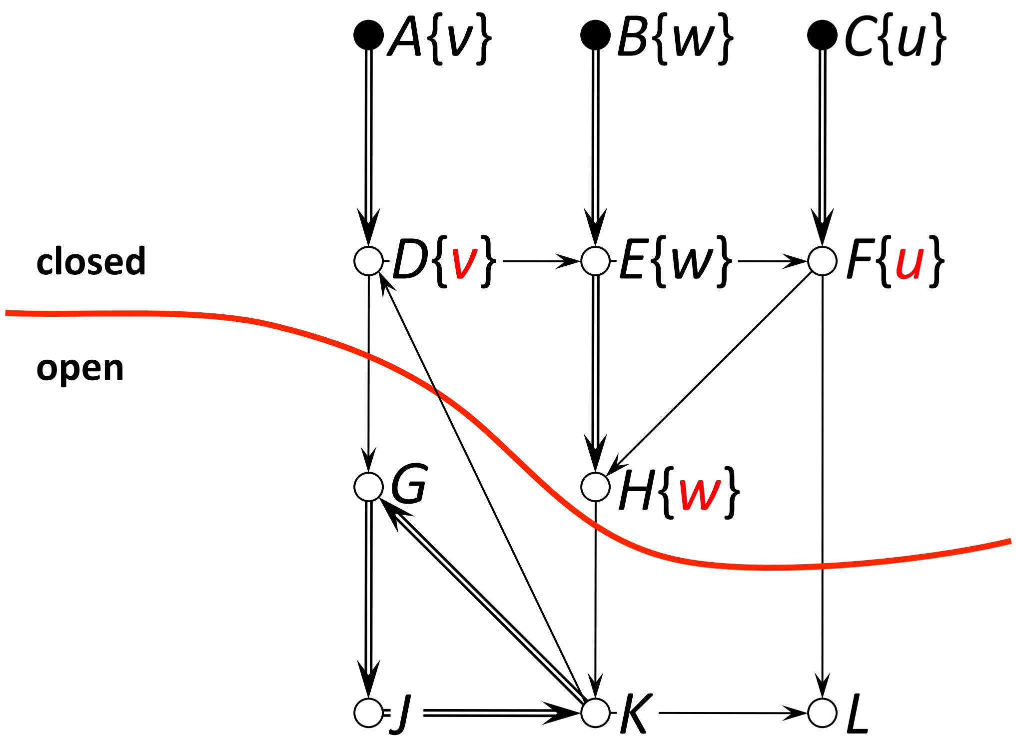
Resolution Algorithm



- 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	$\{w\}$	$\{w\}$
F	$\{u\}$	$\{u\}$
G	?	?
H	?	?
J	?	?
K	?	?
L	?	?

Resolution Algorithm

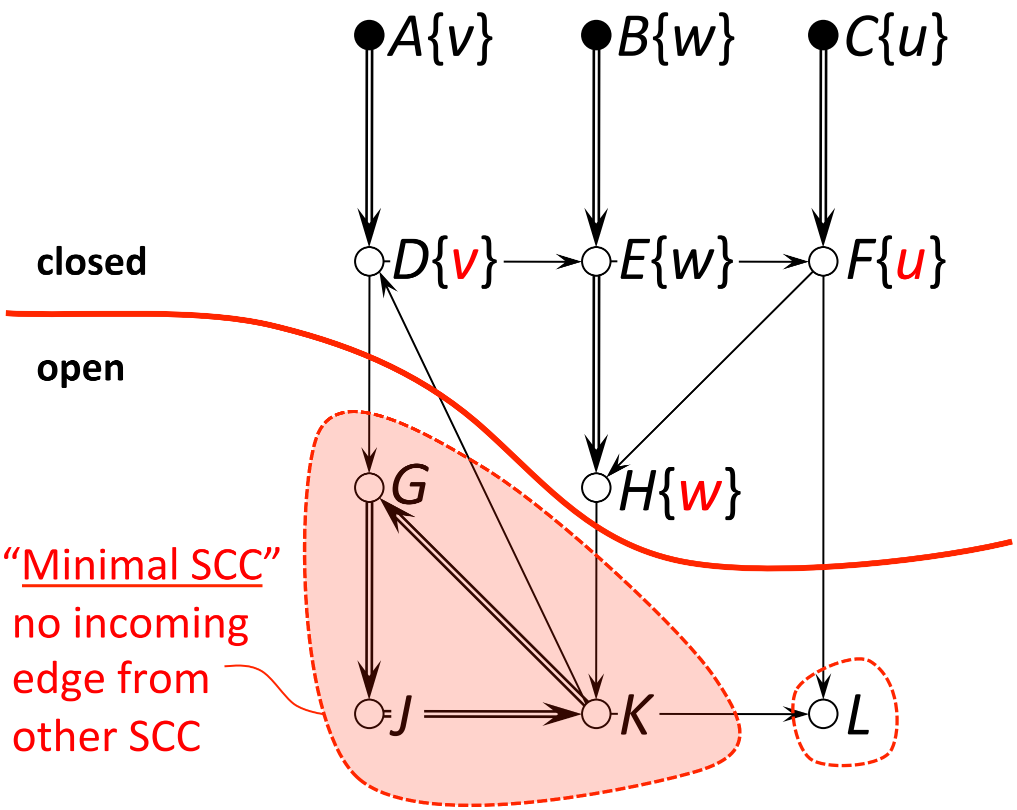


- Keep 2 sets: **closed** / **open**
  Initialize **closed** with explicit beliefs
- MAIN
  - Step 1: if ∃ preferred edges from **open** to **closed** → follow

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

Resolution Algorithm

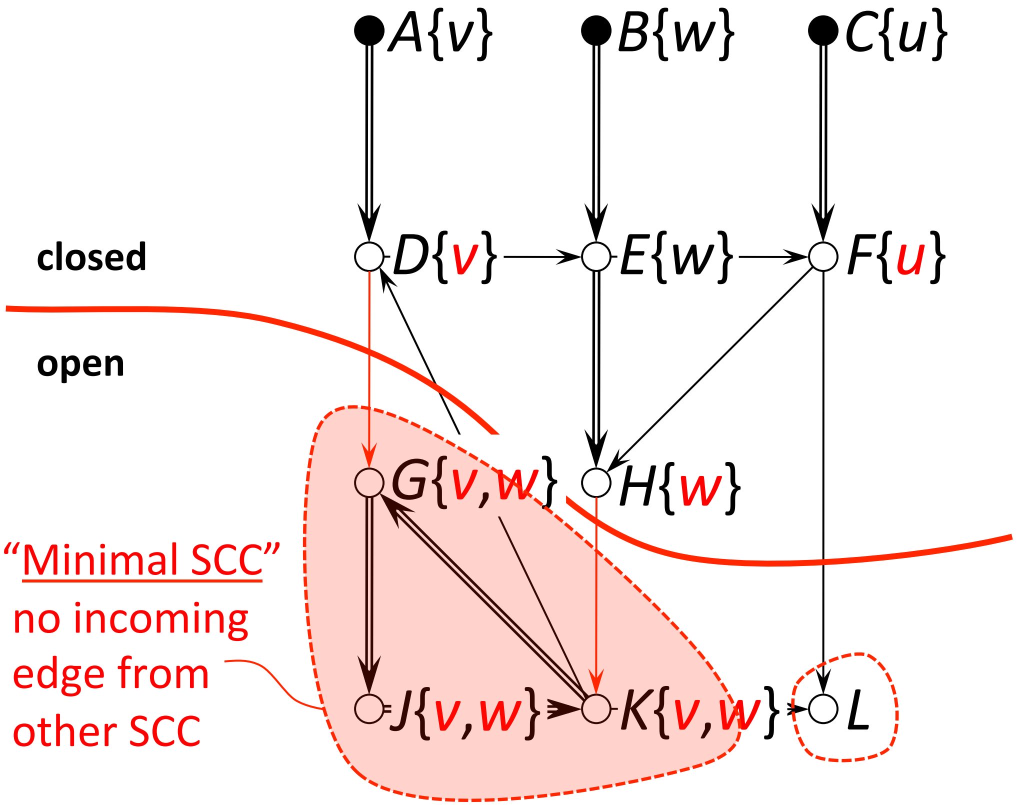
- Keep 2 sets: **closed** / **open**
  Initialize **closed** with explicit beliefs
- MAIN
  Step 1: if ∃ preferred edges from
  open to **closed**
  → follow
  Step 2: else
  → construct SCC graph of **open**



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

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

Resolution Algorithm

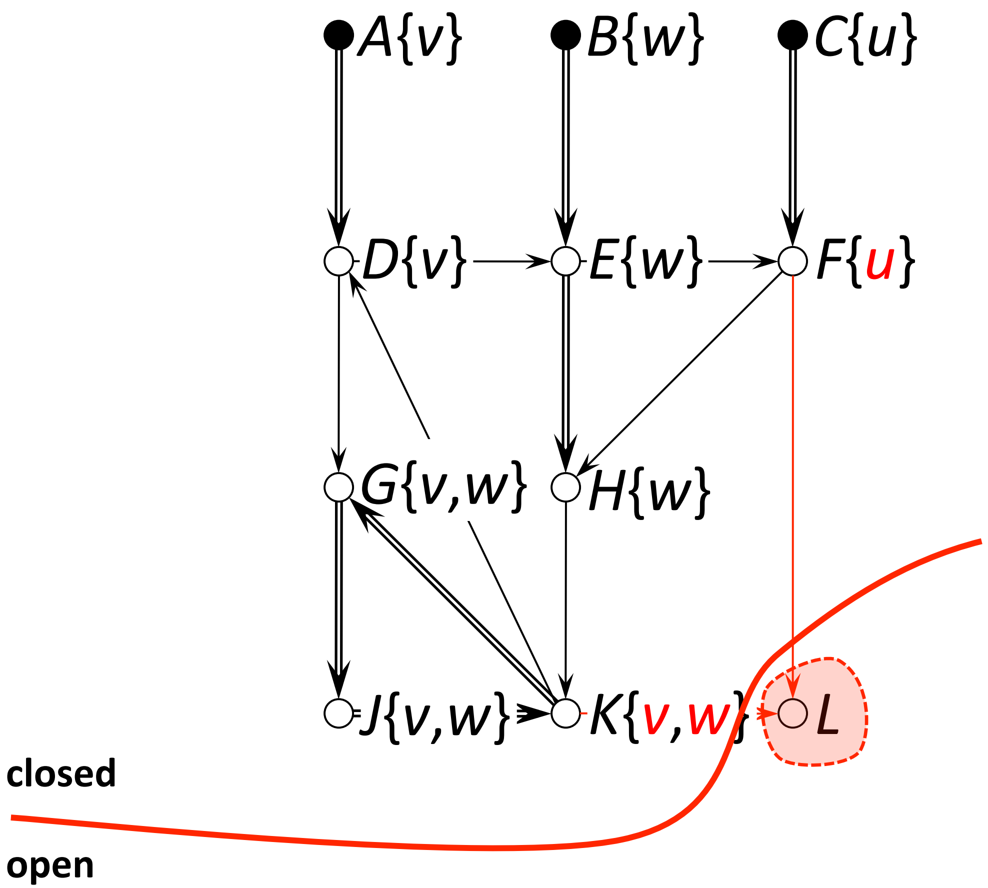


- Keep 2 sets: **closed** / **open**
  Initialize **closed** with explicit beliefs
- MAIN
  Step 1: if ∃ preferred edges from **open** to **closed**
  → follow
  Step 2: else
  → construct SCC graph of **open**
  → resolve minimum SCCs

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}	$\varnothing$
H	{w}	{w}
J	{v,w}	$\varnothing$
K	{v,w}	$\varnothing$
L	?	?

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

Resolution Algorithm

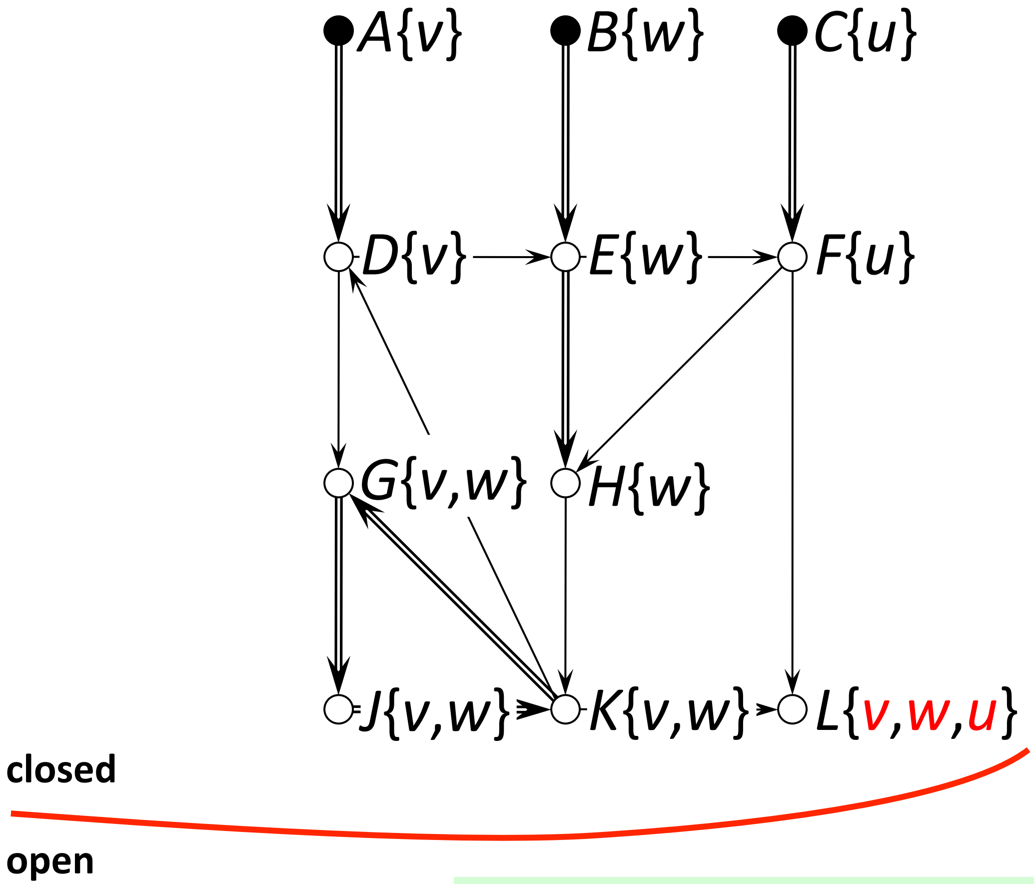


- Keep 2 sets: **closed** / **open**
  Initialize **closed** with explicit beliefs
- MAIN
  Step 1: if ∃ preferred edges from
  open to **closed**
  → follow
  Step 2: else
  → construct SCC graph of **open**
  → resolve minimum SCCs

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	?	?

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



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}	$\varnothing$
H	{w}	{w}
J	{v,w}	$\varnothing$
K	{v,w}	$\varnothing$
L	{v,w,u}	$\varnothing$

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?

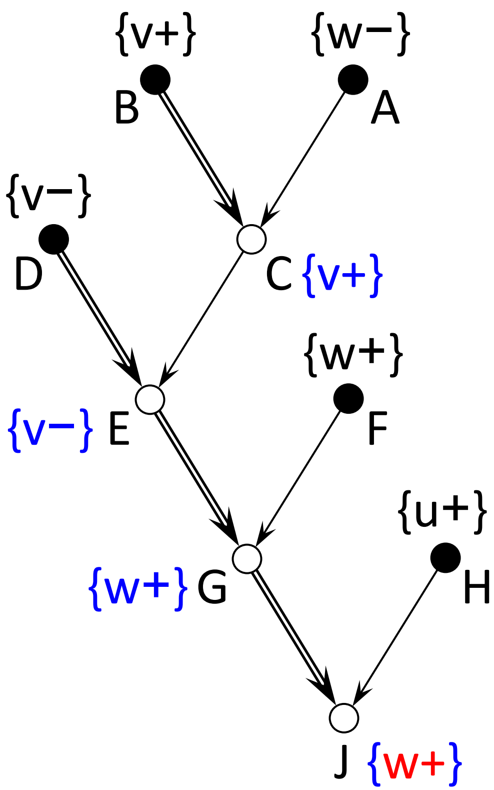
2. Resolution algorithm
   – how to calculate the solution efficiently?

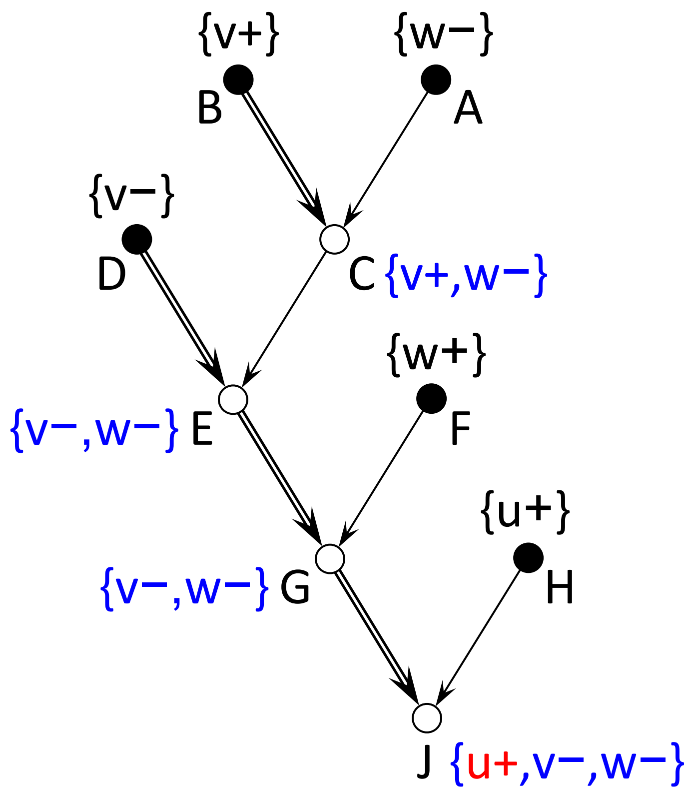
3. Extensions
   – how to deal with “negative beliefs”?

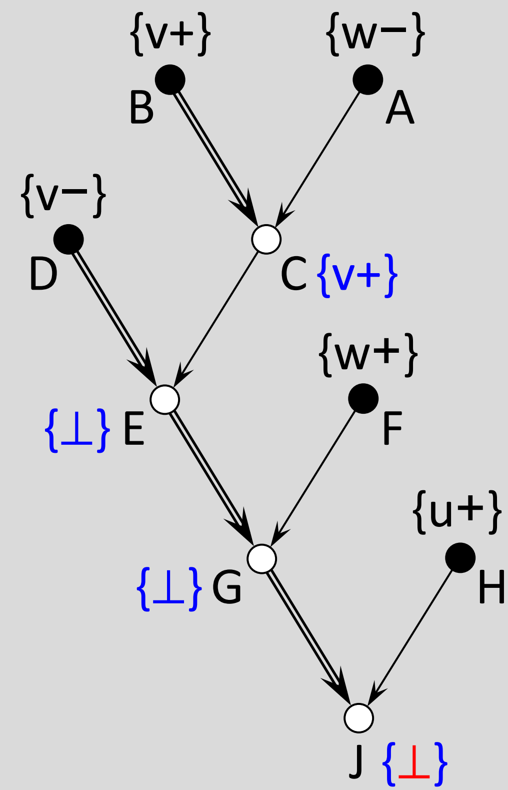
3 semantics for negative beliefs

Our recommendation

	Agnostic	Eclectic	Skeptic







	Agnostic	Eclectic	Skeptic
w/o cycles*	O(*n*)	O(*n*)	O(*n*)
w cycles	NP-hard**	NP-hard**	O(n^2)

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

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

in the paper & TR

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*

glyph	origin	
1	ship hull	Alice
1	cow	Bob
1	jar	Charlie
2	fish	Bob
2	knot	Charlie
3	arrow	Charlie

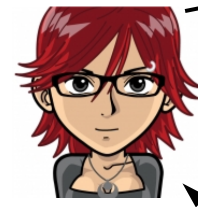
"Beliefs": annotated (key,value) pairs


Bob

glyph	origin
1	cow
2	fish
3	arrow

100

80


Alice

glyph	origin
1	ship hull
2	fish
3	arrow

"Explicit belief"

"Implicit belief"

50


Charlie

glyph	origin
1	jar
2	knot
3	arrow

Background 2: Trust mappings

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

Priorities

Recent work on community databases:

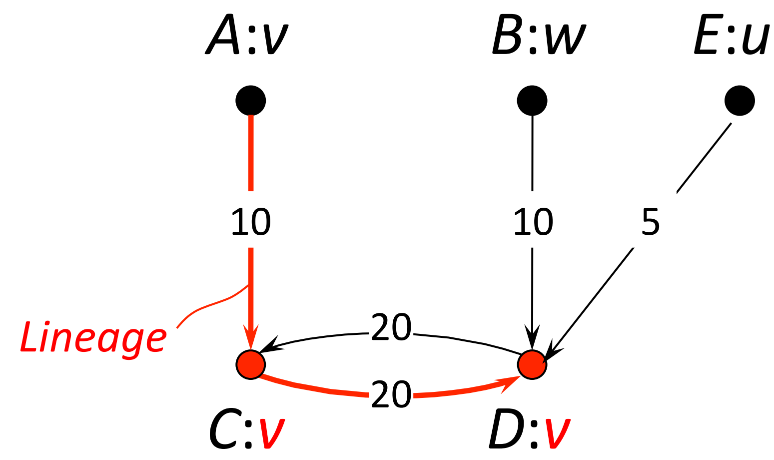
Orchestra [SIGMOD'06, VLDB'07]
Youtopia [VLDB'09], BeliefDB [VLDB'09]

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*)

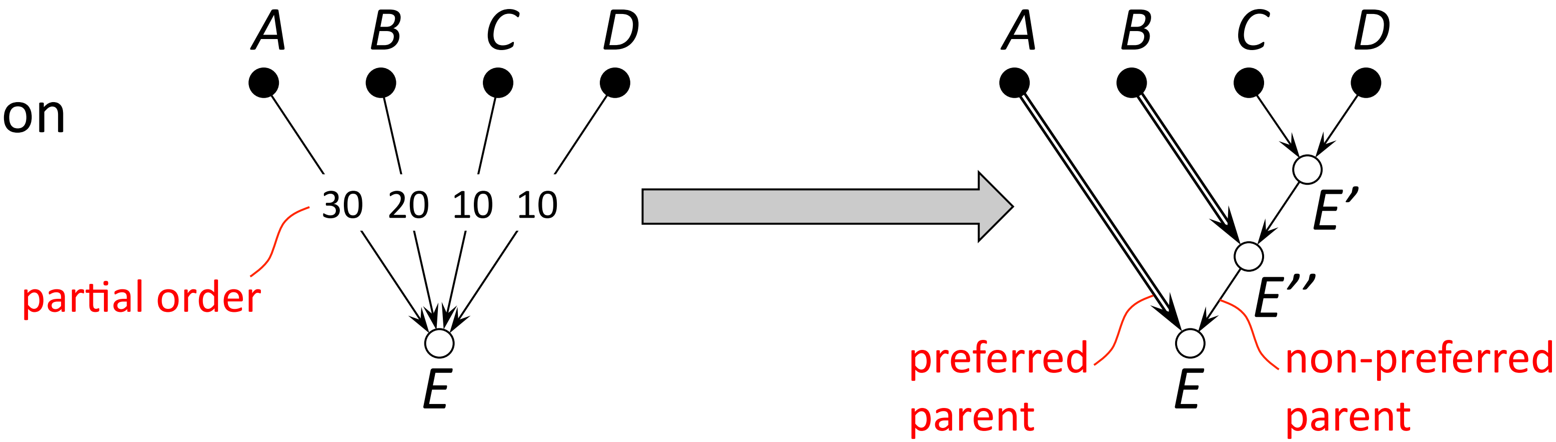
- Possible / Certain semantics
  - a stable solution determines, for each node, a possible value ("**poss**")
  - certain value ("**cert**") = intersection of all stable solutions, per user

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

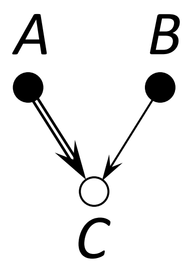
* each node with at least one ancestor with explicit belief

3. Logic programs with stable model semantics

Step 1:
Binarization



Step 2:
Logic program

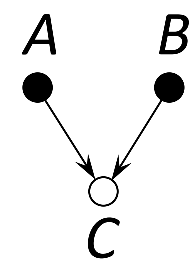


1: accept all **poss** of preferred parent

$P(C,x) \leftarrow P(A,x)$

$F(C,B,y) \leftarrow P(B,y), P(C,x), x \neq y$

$P(C,y) \leftarrow P(B,y), \neg F(C,B,y$



$F(C,A,y) \leftarrow P(A,y), P(C,x), x \neq y$

$P(C,y) \leftarrow P(A,y), \neg F(C,A,y)$

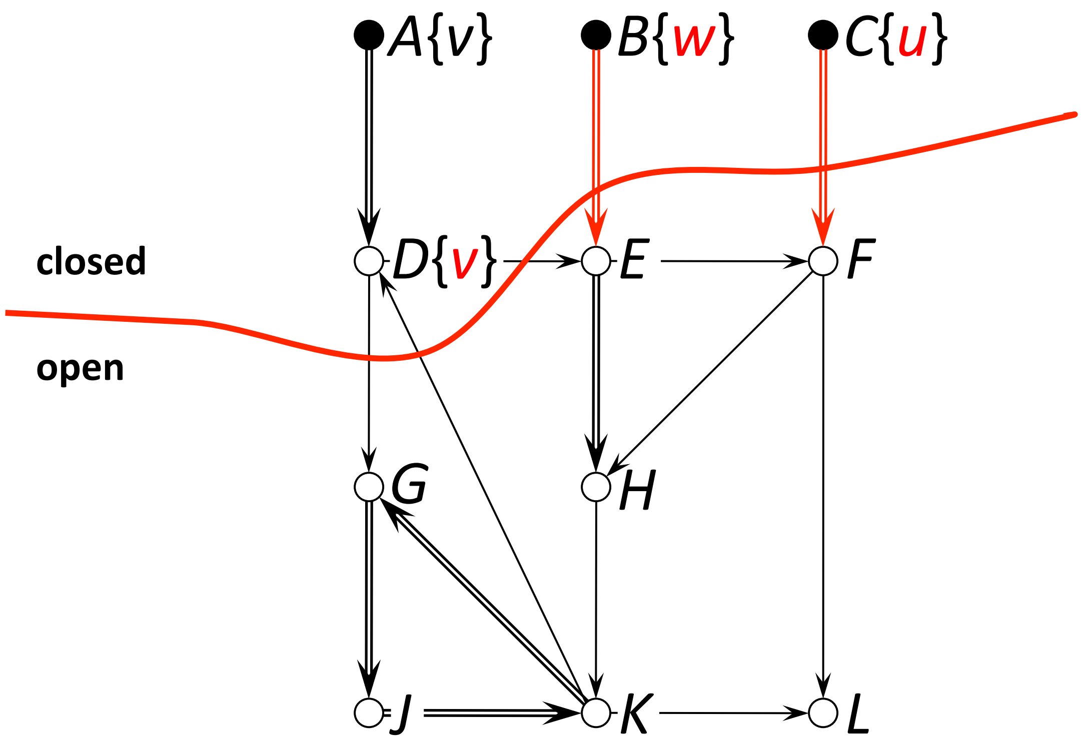
$F(C,B,y) \leftarrow P(B,y), P(C,x), x \neq y$

$P(C,y) \leftarrow P(B,y), \neg F(C,B,y)$

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

4. Resolution Algorithm (1/2)

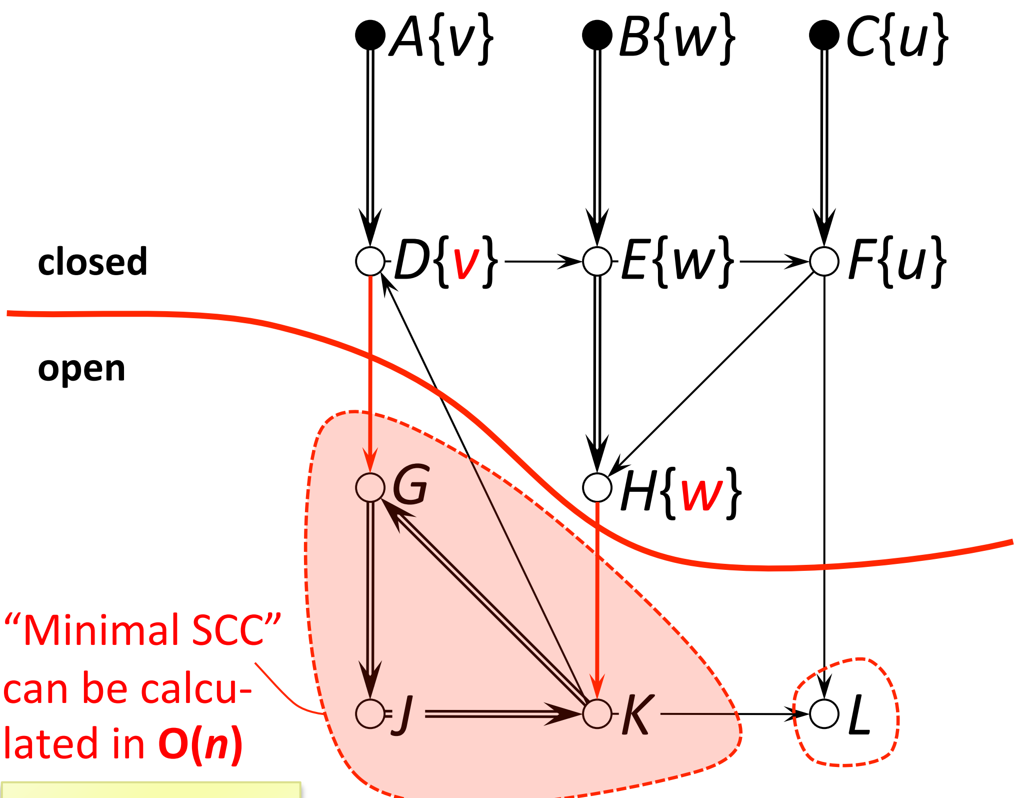
- Keep 2 sets: **closed** / **open**
  Initialize **closed** with explicit beliefs
- MAIN
  - Step 1: if ∃ preferred edges from **open** to **closed**
    → follow



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



Tarjan [1972]

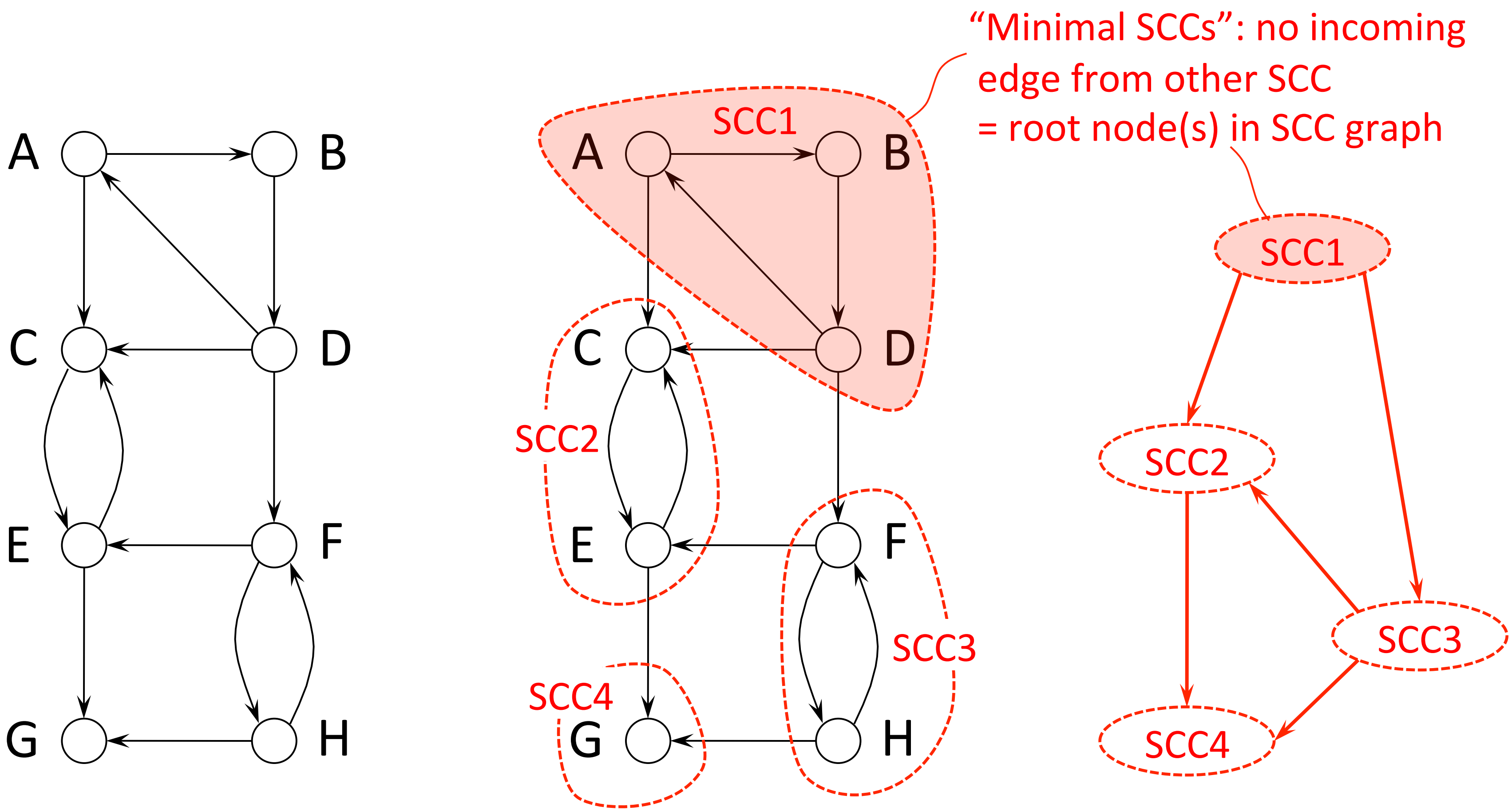
X	**poss**(X)	**cert**(X)
A	{v}	{v}
B	{w}	{w}
C	{u}	{u}
D	{v}	{v}
E	{w}	{w}
F	{u}	{u}
G	{v,w}	$\varnothing$
H	{w}	{w}
J	{v,w}	$\varnothing$
K	{v,w}	$\varnothing$
L	?	?

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

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]

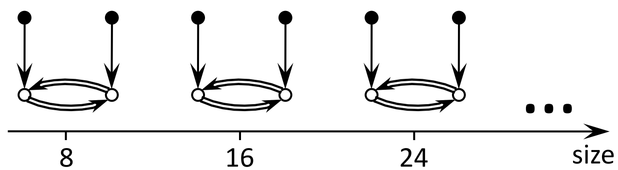


7. Experiments on large network data

Calculating **poss** / **cert** for fixed key

- **DLV**: State-of-the art logic programming solver
- **RA**: Resolution algorithm

Network 1: “Oscillators”

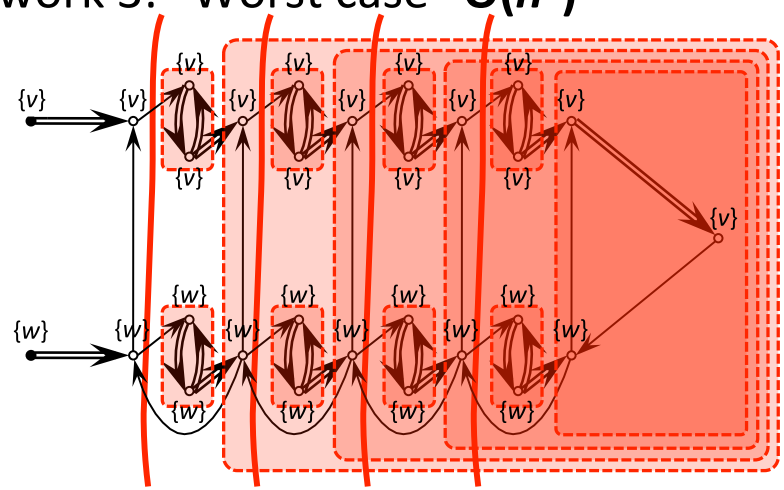


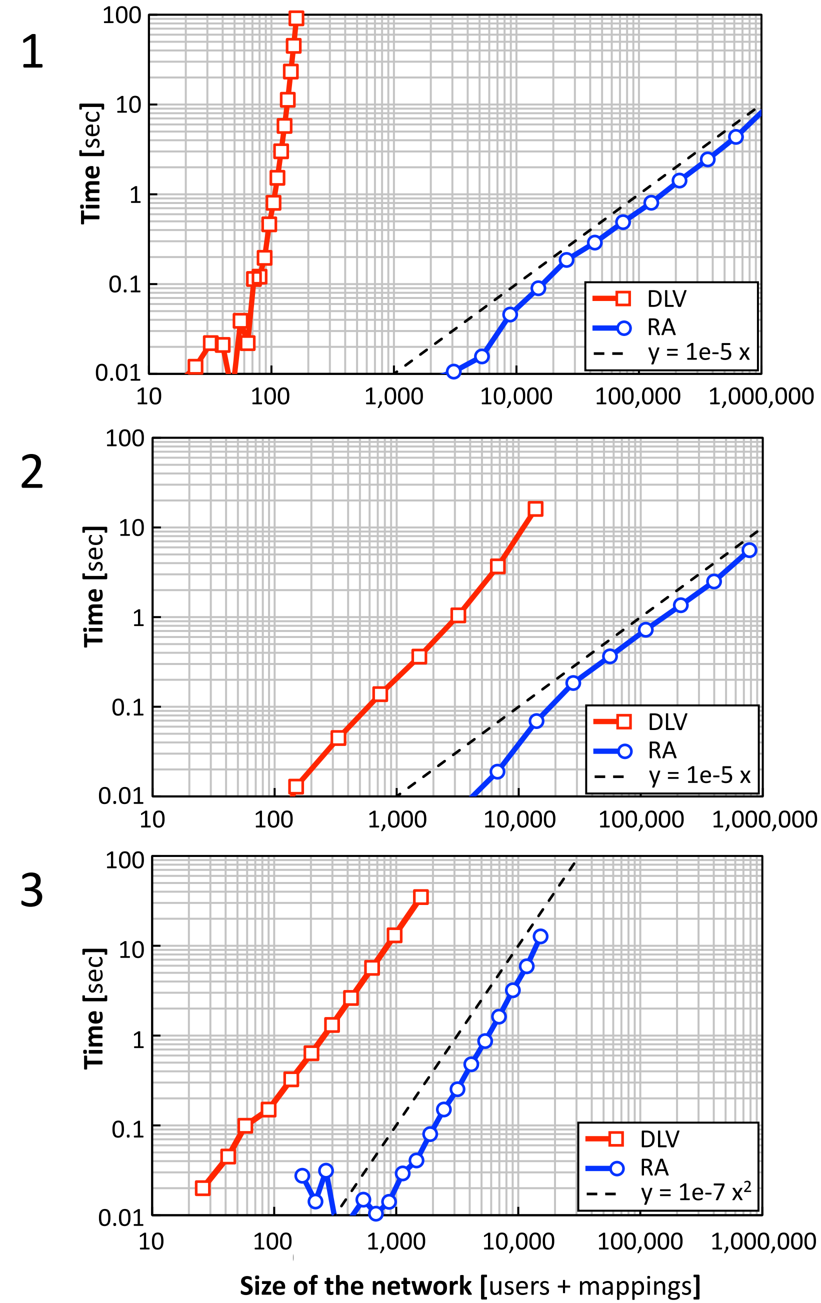
Network 2: “Web link data”

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

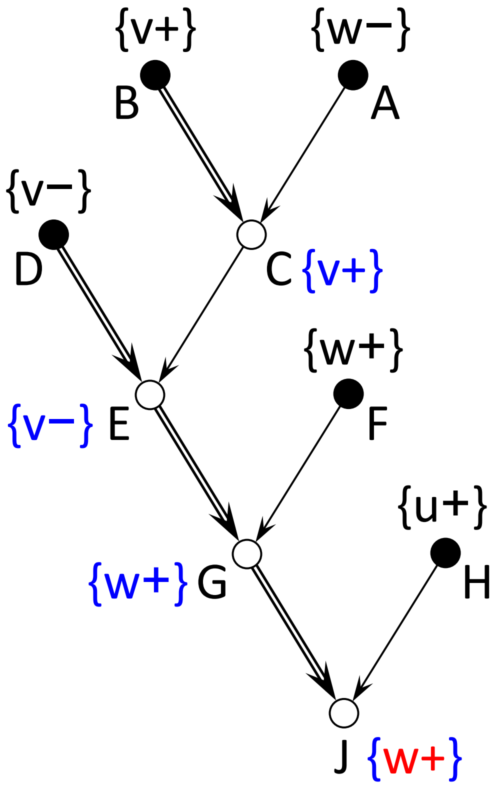
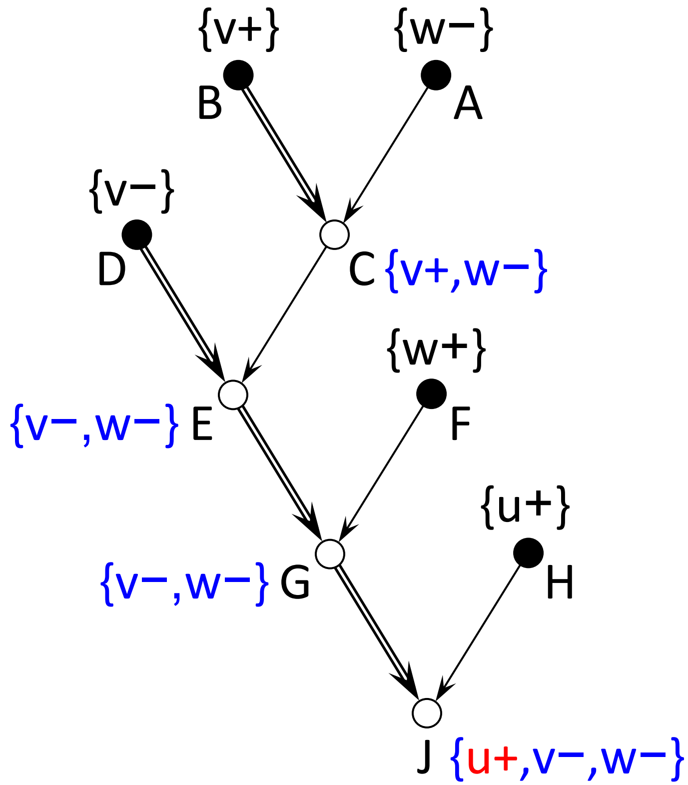
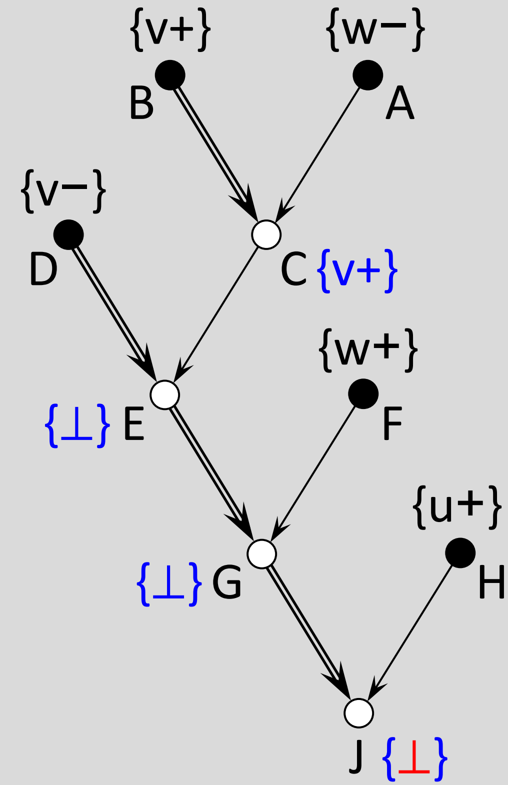
Network 3: “Worst case” $\mathbf{O}(n^2)$





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**	$O(n^2)$

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 [**$O(n^2)$** 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

not covered in the talk

Slides soon available on our project page:

http://db.cs.washington.edu/beliefDB

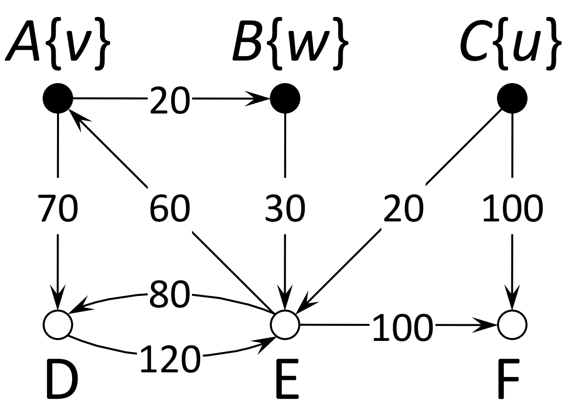
backup

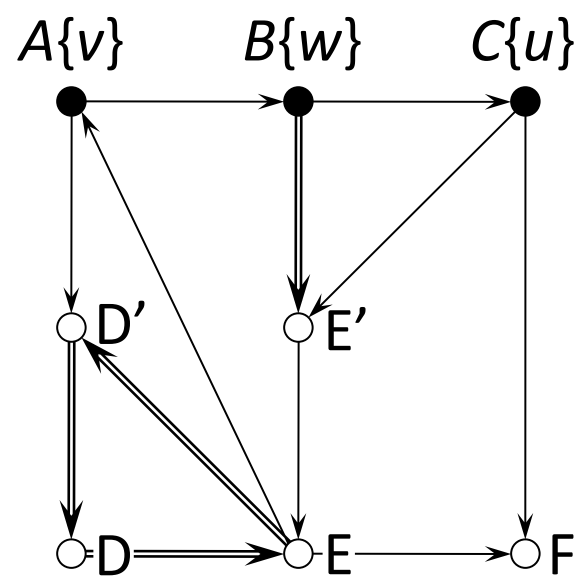
Binarization for Resolution Algorithm*

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

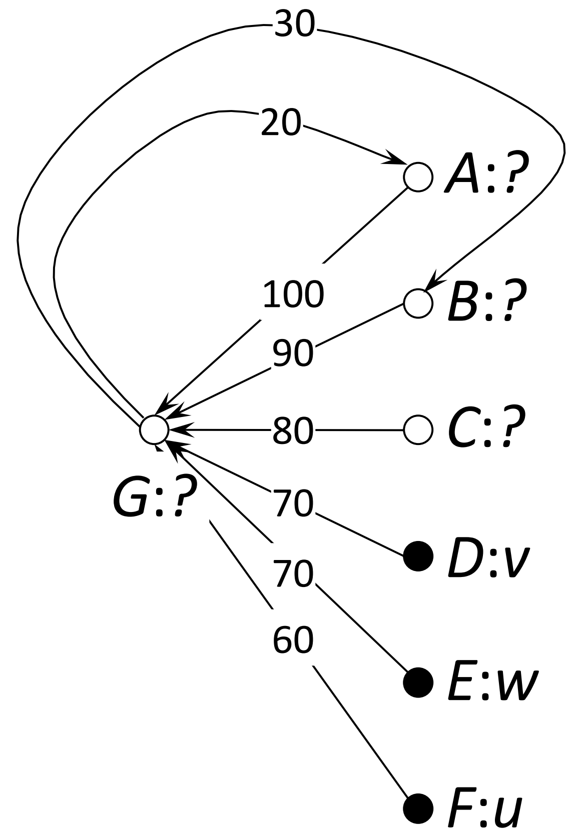




* Note that binarization is not necessary, but greatly simplifies the presentation

Stable solutions: example 2

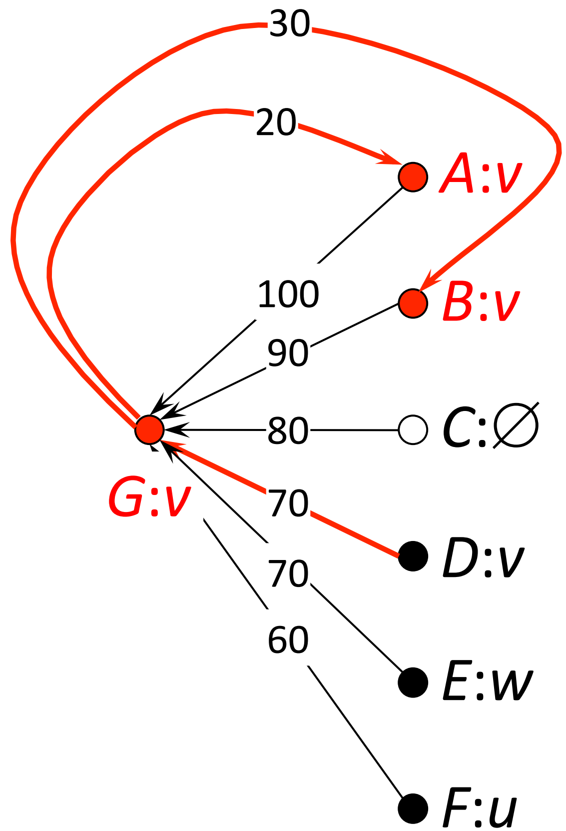
- Priority trust network (TN)
  - assume a fixed key
  - users (nodes): *A*, *B*, *C*
  - values (beliefs): *v*, *w*, *u*
  - trust mappings (arcs) from "parents"
- Stable solution
  - assignment of values to each node*, s.t. each belief has a "*non-dominated lineage*" to an explicit belief
- Certain 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

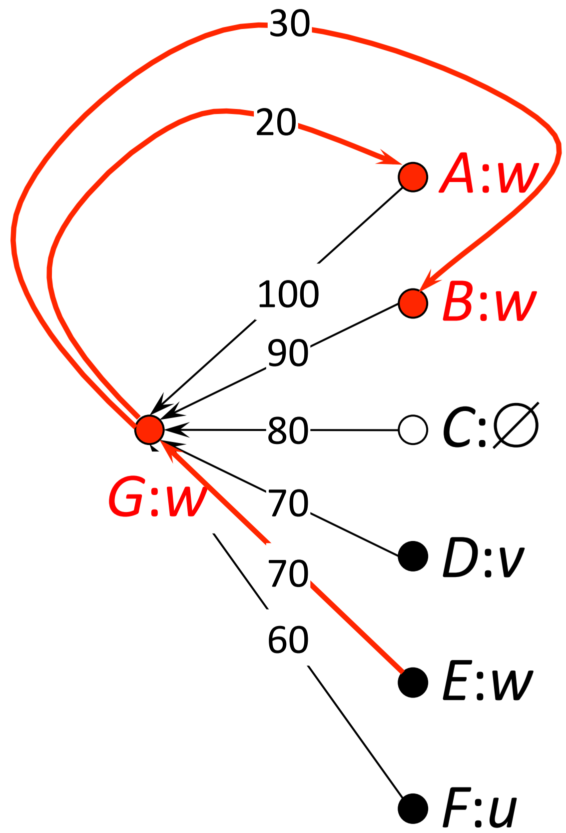


poss(*G*) = {*v*,...}

* 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



poss(*G*) = {*v*,*w*,...}

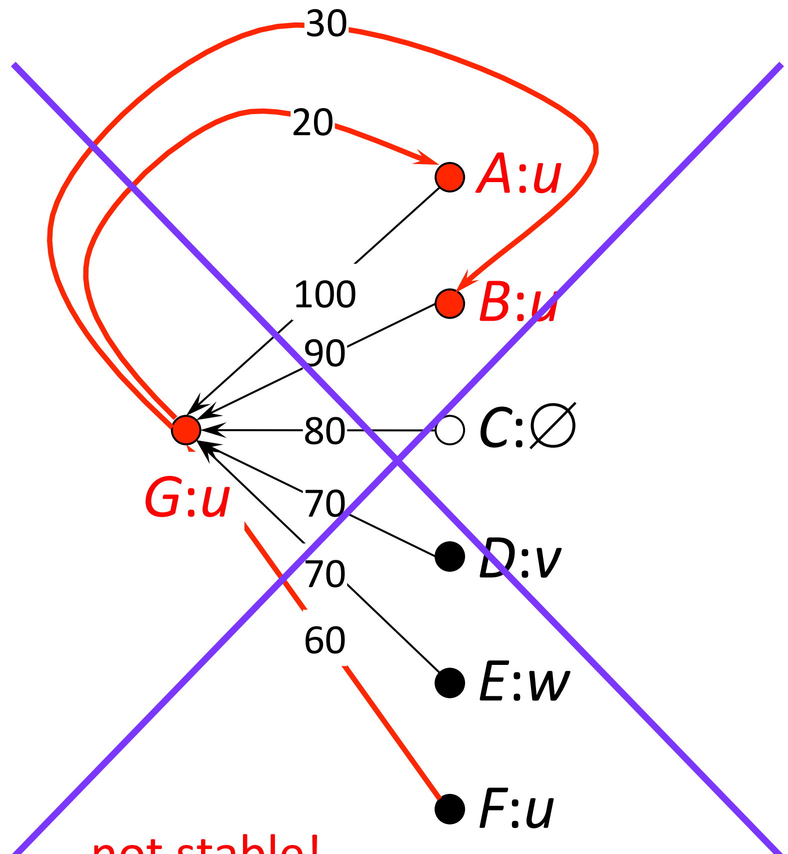
* each node with at least one ancestor with explicit belief

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



not stable!
F→G dominated by E→G

poss(*G*) = {*v*,*w*}

cert(*G*) = ∅

* each node with at least one ancestor with explicit belief

O(n²)-worst-case for Resolution Algorithm