Carleton
UNIVERSITY

Inconsistency-Driven Information Sharing in Peer Data Exchange Systems

Leopoldo Bertossi

Carleton University
Ottawa, Canada

Join work with: **Loreto Bravo**
Universidad de Concepcion, Chile

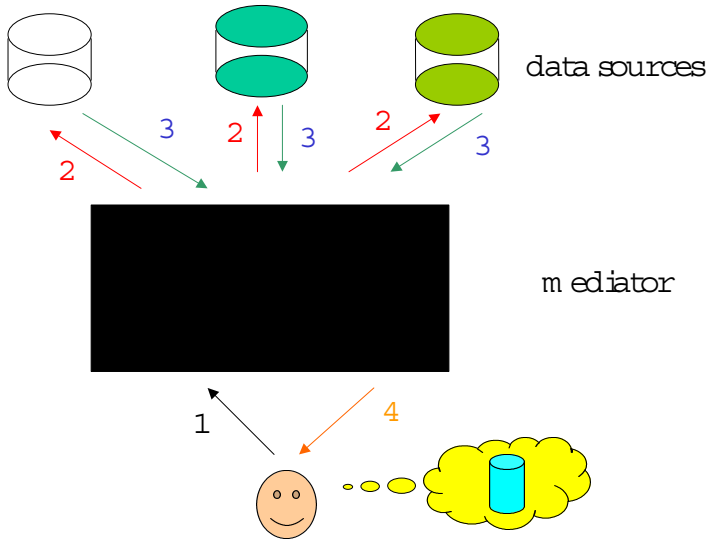
Some Forms of Data Integration

There are different approaches and paradigms for data integration, e.g.

- Materialized: Physical repository is created
- Mediated: Data stay at the sources, a virtual integration system is created
- Data Exchange: Data is exported from one system to another
- Peer-to-Peer Data Exchange: Many peers exchange data without a central control mechanism

  Peer Data Management Systems ...

Mediator-Based Data Integration



A virtual database is created which is accessed via a *mediator*

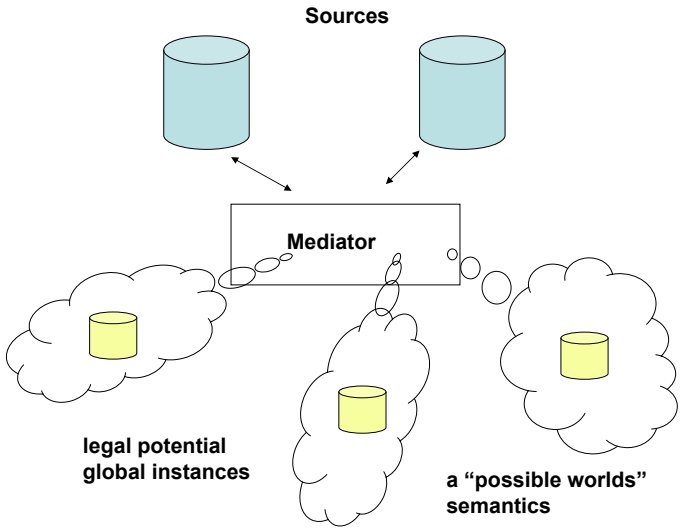
Queries are posed and answered via the mediator

Which are the correct answers?

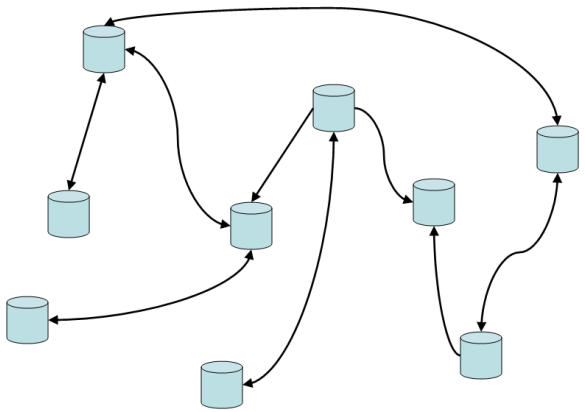
The semantics is given in terms of a collection of legal and intended instances over the global schema: A possible world semantics

It takes into account mappings between global and local schemas

What is true of the system is what is true of all those instances, in particular, query answers



Peer Data Exchange Systems



Each peer has a local and autonomous database

Data at two different peers may be related by data exchange constraints (DECs) (schema mappings)

Local queries are posed to individual peers

Peers exchange data when they answer their queries

Exchange of data depends on: The query, the DECs, the data at the relevant peers, local ICs, and trust relationships between peers

A peer does not update its physical instance according to its DECs and other peers' instances

However, if a peer P is answering a (local) query $Q_{\mathtt{P}}$, it may, at query time:

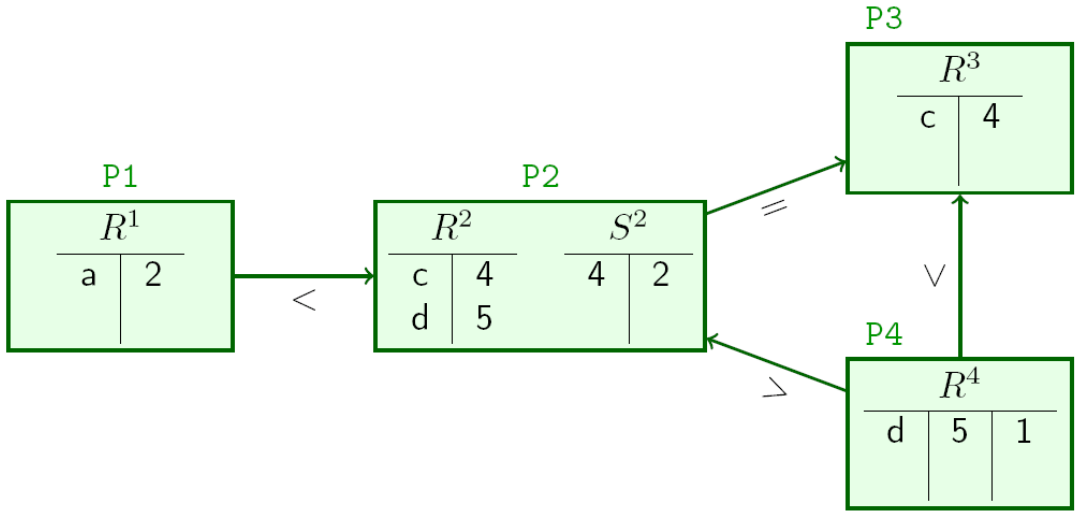
- Import data from other peers to complement its data
- Ignore part of its own data

This depending upon its own DECs, local ICs, and neighboring peers' data

But also upon the trust relationships that P has with its neighbors

We assume that peers have disjoint schemas

Example: Peers in a systems $\mathcal{P}$, their schemas, instances, DECs, trust relationships:



DECs:

$$\Sigma(\mathsf{P1},\mathsf{P2}) = \{\forall x \forall y (R^2(x,y) \rightarrow R^1(x,y))\}$$

$$\Sigma(\mathsf{P2},\mathsf{P3}) = \{\forall x \forall y (R^2(x,y) \wedge R^3(x,y) \rightarrow \mathbf{false})\}$$

$$\Sigma(\mathsf{P4},\mathsf{P2}) = \{\forall x \forall y \forall z (R^2(x,y) \wedge S^2(y,z) \rightarrow R^4(x,y,z))\}$$

$$\Sigma(\mathsf{P4},\mathsf{P3}) = \{\forall x \forall y (R^3(x,y) \rightarrow \exists z R^4(x,y,z))\}$$

Our Motivation and Results

- Develop a a formal semantics for PDESs where peers exchange data for query answering

- Use the semantics to characterize the intended and correct answers to a query posed to and answered by a peer

- We propose a model-theoretic semantics

That is, a collection of possible and admissible models over which the system is interpreted

- The expected answers from a peer to a query are those that are certain; wrt its intended instances

- The semantics can be declarative specified and made executable via logic programs with stable model semantics

General Remarks

- A peer data exchange system (PDES) can be seen as a set of information agents
- Each of them being the owner of a data source
- Each peer P can be seen as an ontology consisting of:
  - the database instance
  - metadata describing the database schema $\mathcal{R}(\mathrm{P})$ and local integrity constraints (ICs)
  - its set $\Sigma(\mathrm{P}) = \bigcup_{\mathrm{P'} \in \mathcal{P}} \Sigma(\mathrm{P}, \mathrm{P'})$ of DECs, and
  - its trust relationships

• These ontologies may be pairwise inconsistent due to the DECs and the database facts

It is easy to extend our framework to handle DECs that contain views, i.e. defined relational predicates

• This kind of consistency issues also emerge when aligning ontologies

• Our notion of DEC largely extends the inclusions of concepts in the ontological scenario

Our DECs can be much more general than inclusions

• In our case, we do not make the ontologies mutually consistent

Whenever possible, inconsistencies are solved at query time

The Idea: "Operational" Semantics

When a peer P receives a query:

- P sends queries to its neighbors, to get data and check the satisfaction of its DECs

  This can be made relative to the specific query at hand (relevance of DECs to the query)

- If they are not satisfied, P tries to restore satisfaction of (consistency wrt) its DECs

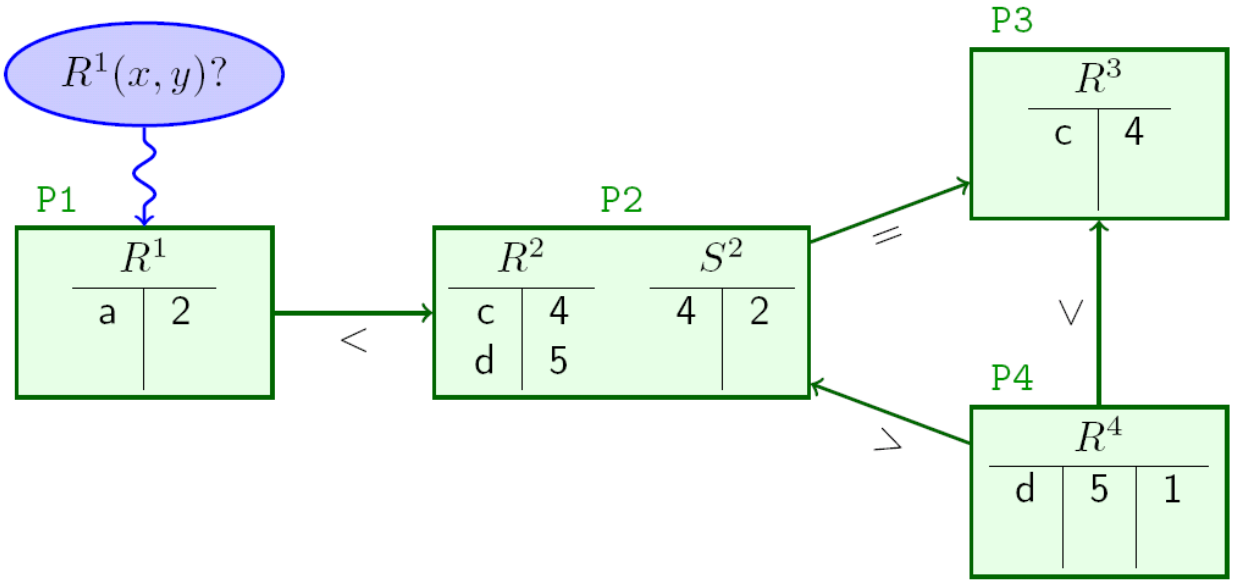
  – A repair, that satisfies the DECs, respects the trust relationships and is "as close as possible" to P's original data is called a *solution* for P

- There might be several solutions
  - Peer P returns as query answers those that are certain, i.e. true in all of P's solutions
  - These are the peer consistent answers (PCAs) from P
- What did P receive from a neighbor P'?

  The answers by P' to the queries from P are also true in all solutions for P'

  The same idea/process has to be applied to P's neighbors, and so on ...

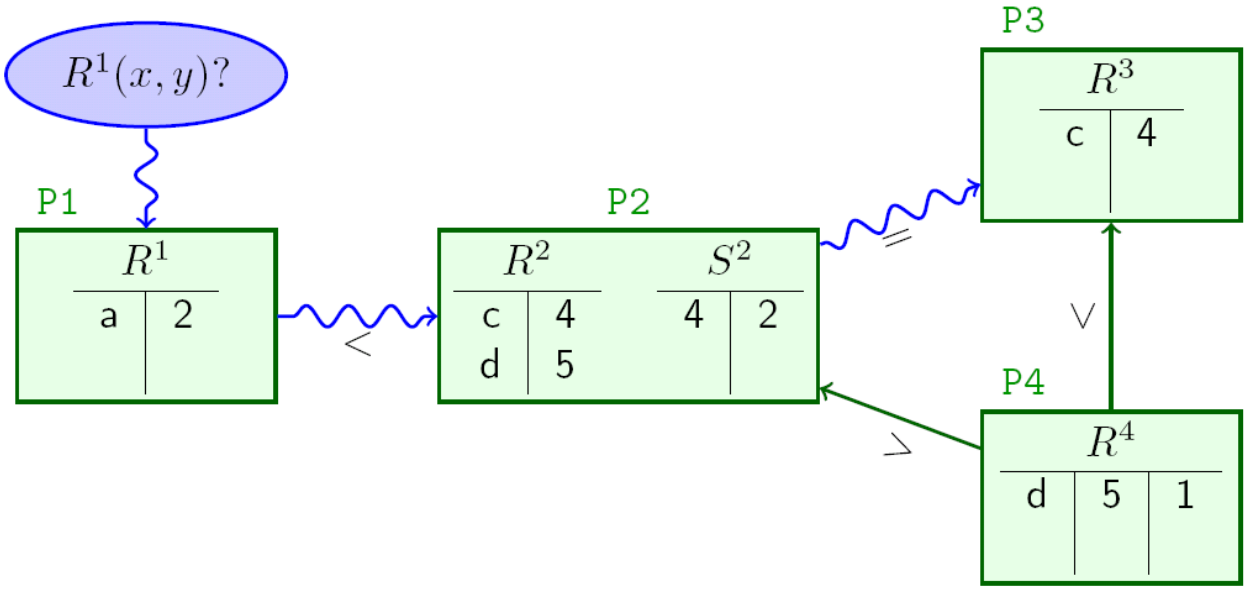
  Peers pass (locally) certain data to other neighboring peers



Example: DECs:

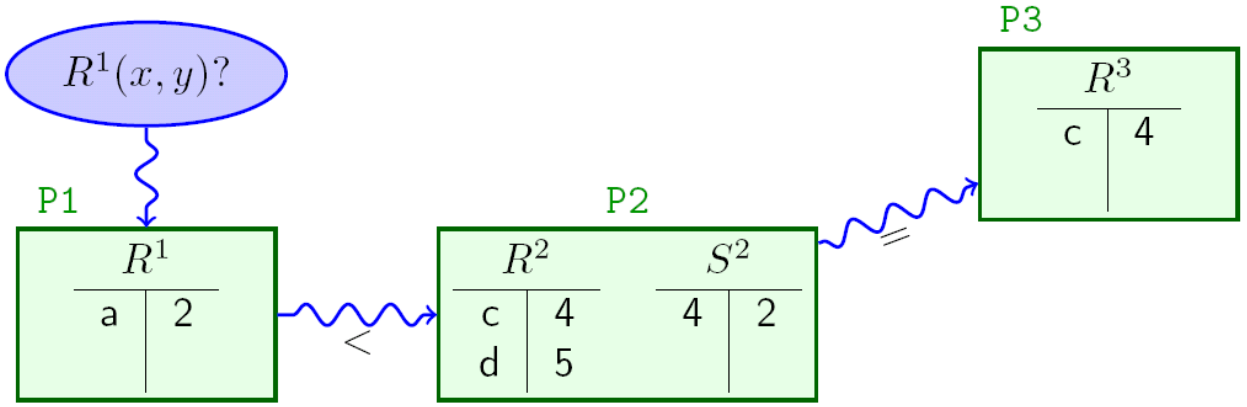
$$\left.\begin{array}{rl} \Sigma(\mathsf{P1},\mathsf{P2}) = & \{\forall x \forall y (R^2(x,y) \rightarrow R^1(x,y))\} \\ \Sigma(\mathsf{P2},\mathsf{P3}) = & \{\forall x \forall y (R^2(x,y) \wedge R^3(x,y) \rightarrow \mathbf{false})\} \\ \Sigma(\mathsf{P4},\mathsf{P2}) = & \{\forall x \forall y \forall z (R^2(x,y) \wedge S^2(y,z) \rightarrow \\ & \qquad R^4(x,y,z))\} \end{array}\right\} \text{Univ. DECs}$$

$$\Sigma(\mathsf{P4},\mathsf{P3}) = \{\forall x \forall y (R^3(x,y) \rightarrow \exists z R^4(x,y,z))\} \quad \} \text{ Ref. DEC}$$



$$\Sigma(\mathsf{P1},\mathsf{P2}) = \{\forall x \forall y (R^2(x,y) \rightarrow R^1(x,y))\}$$
$$\Sigma(\mathsf{P2},\mathsf{P3}) = \{\forall x \forall y (R^2(x,y) \wedge R^3(x,y) \rightarrow \mathbf{false})\}$$
$$\Sigma(\mathsf{P4},\mathsf{P2}) = \{\forall x \forall y \forall z (R^2(x,y) \wedge S^2(y,z) \rightarrow R^4(x,y,z))\}$$
$$\Sigma(\mathsf{P4},\mathsf{P3}) = \{\forall x \forall y (R^3(x,y) \rightarrow \exists z R^4(x,y,z))\}$$

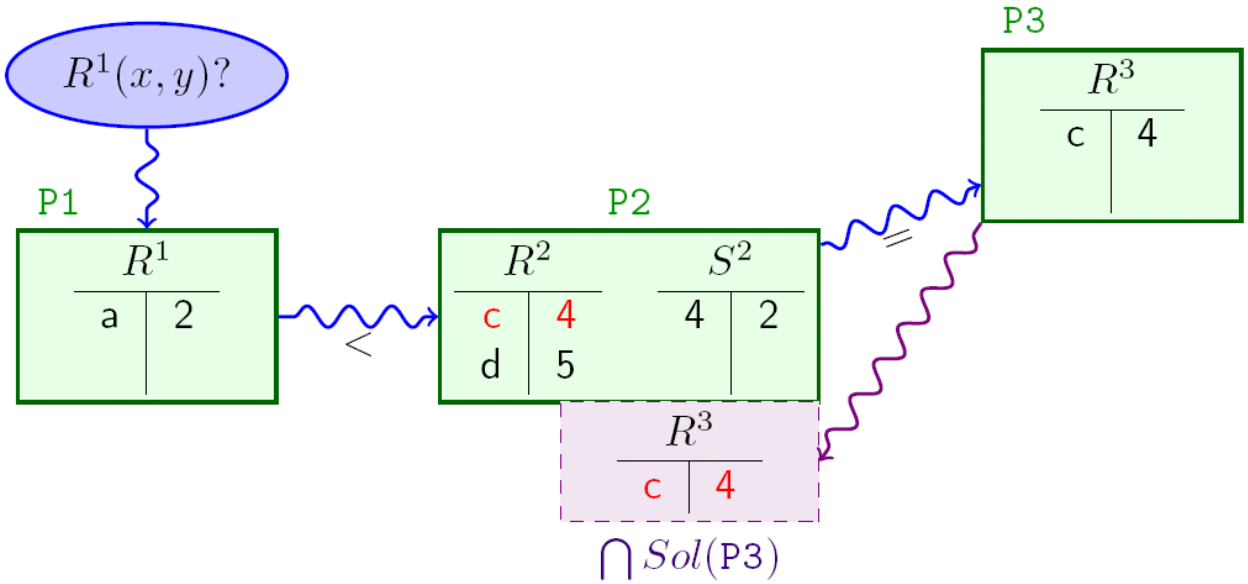
Peer P4 will have no effect on the query!



$$\Sigma(\mathsf{P1},\mathsf{P2}) = \{\forall x \forall y (R^2(x,y) \rightarrow R^1(x,y))\}$$

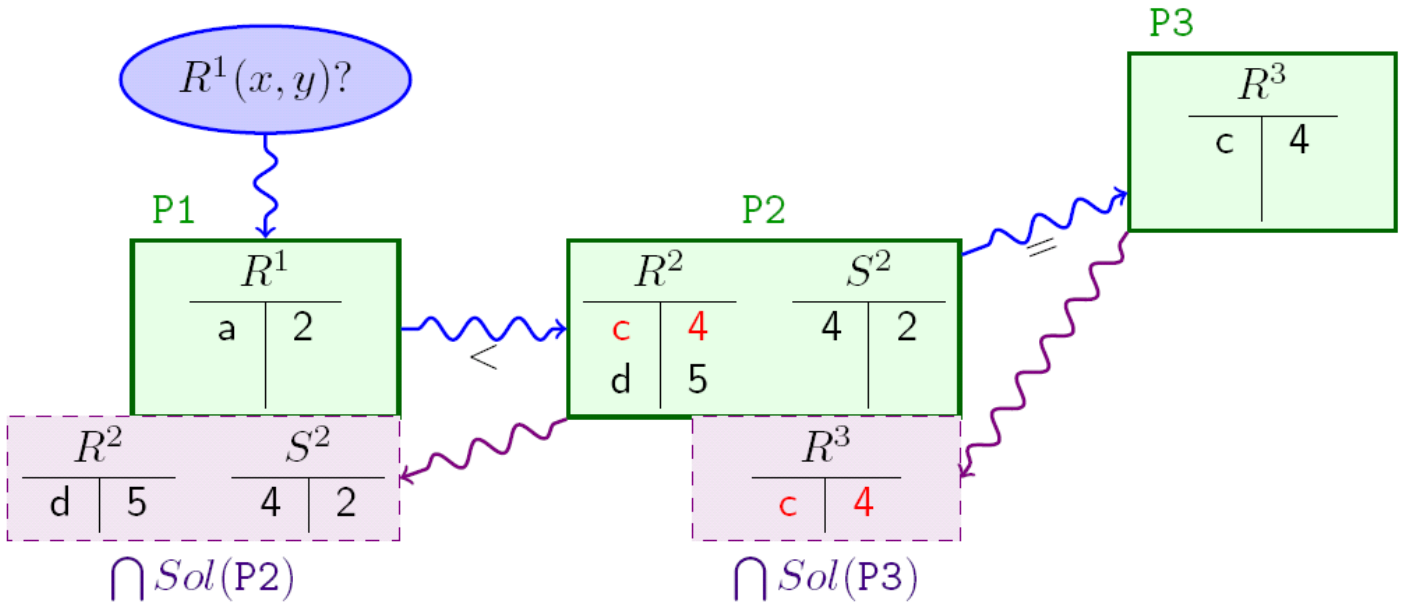
$$\Sigma(\mathsf{P2},\mathsf{P3}) = \{\forall x \forall y (R^2(x,y) \wedge R^3(x,y) \rightarrow \mathbf{false})\}$$

(a denial DEC)



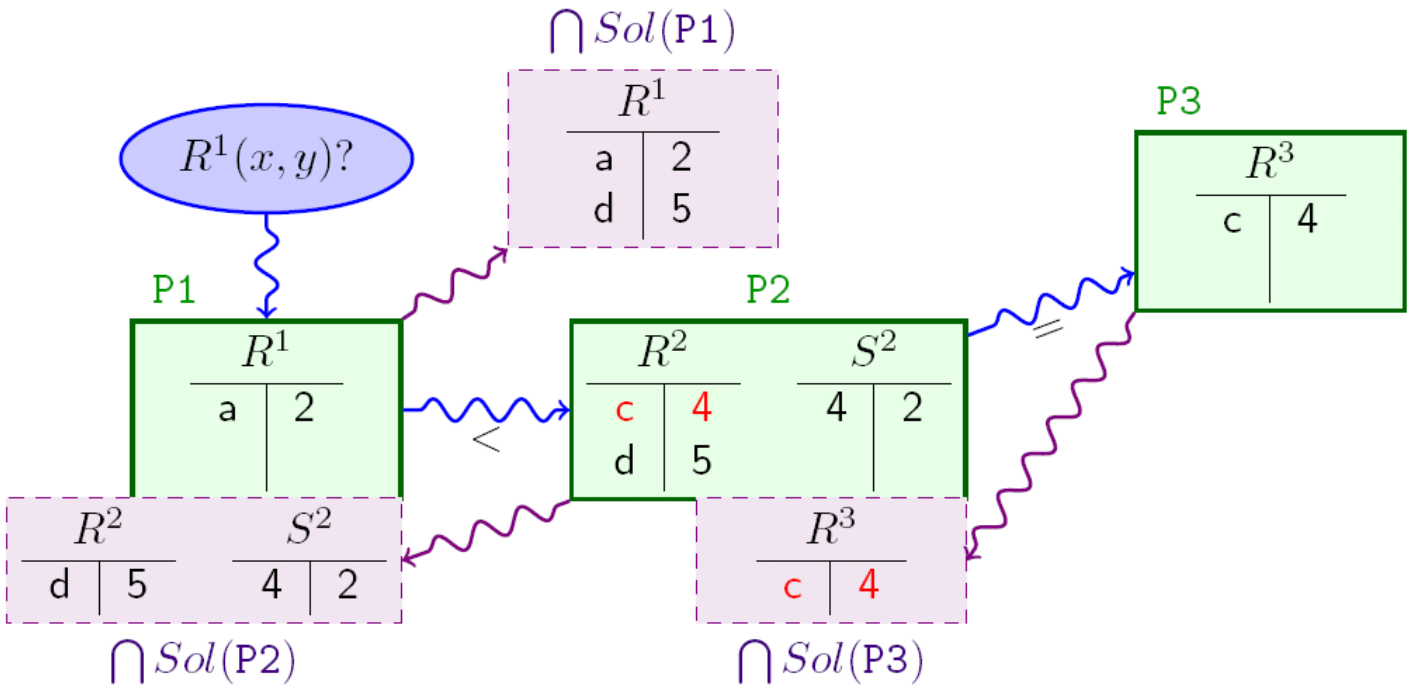
$$\Sigma(\mathtt{P1},\mathtt{P2}) = \{\forall x \forall y (R^2(x,y) \rightarrow R^1(x,y))\}$$

$$\Sigma(\mathtt{P2},\mathtt{P3}) = \{\forall x \forall y (R^2(x,y) \wedge R^3(x,y) \rightarrow \mathbf{false})\}$$



$$\Sigma(\mathsf{P1},\mathsf{P2}) = \{\forall x \forall y (R^2(x,y) \rightarrow R^1(x,y))\}$$

$$\Sigma(\mathsf{P2},\mathsf{P3}) = \{\forall x \forall y (R^2(x,y) \wedge R^3(x,y) \rightarrow \mathbf{false})\}$$



$\Sigma(\mathtt{P1},\mathtt{P2}) = \{\forall x \forall y (R^2(x,y) \rightarrow R^1(x,y))\}$

$\Sigma(\mathtt{P2},\mathtt{P3}) = \{\forall x \forall y (R^2(x,y) \wedge R^3(x,y) \rightarrow \mathbf{false})\}$

Peer consistent answers from P1: $(a,2)$ and $(d,5)$

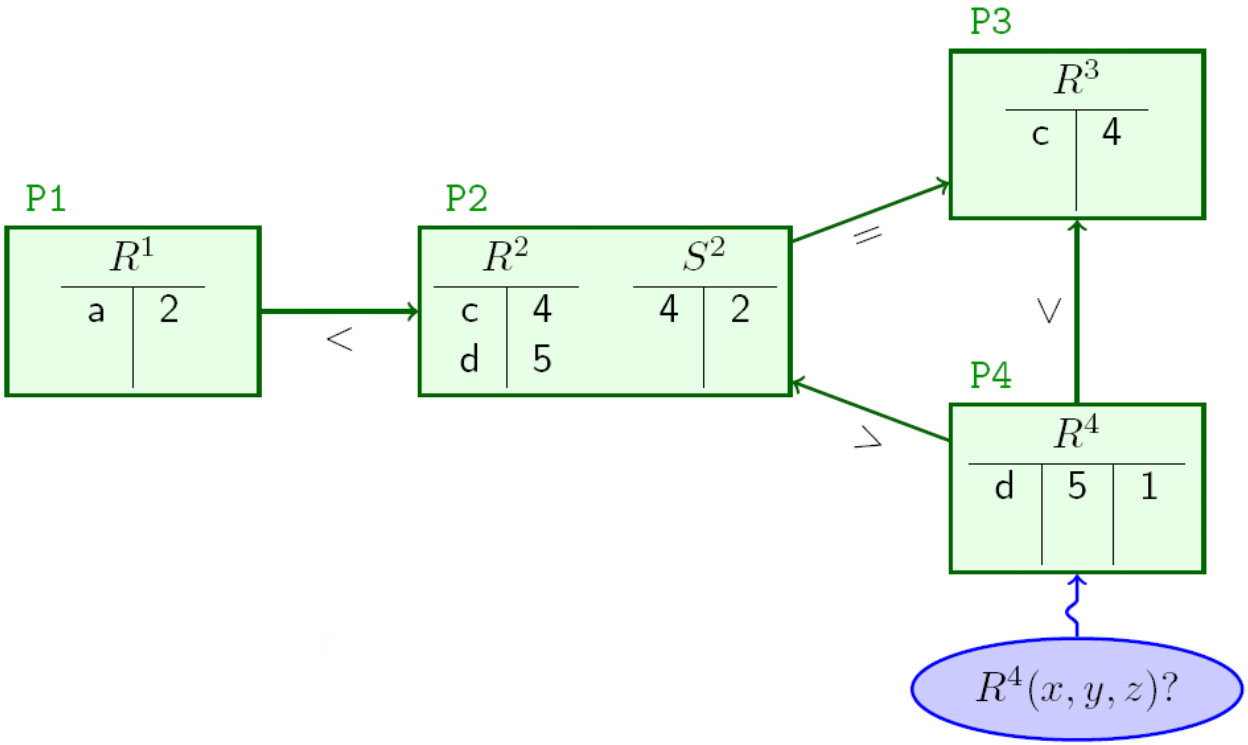
In this example, a peer passes a complete certain instance back to a neighbor, e.g. P2 to P1

This may be more than what P1 needs to answer the original query, e.g. P1 does not need S_2 from P2

Just to show the issues behind the semantics, and of each peer in particular

Each peer will have a set of local solution instances, and this set is not determined by a particular query

In terms of data movement, many things can be optimized wrt the example



Example:

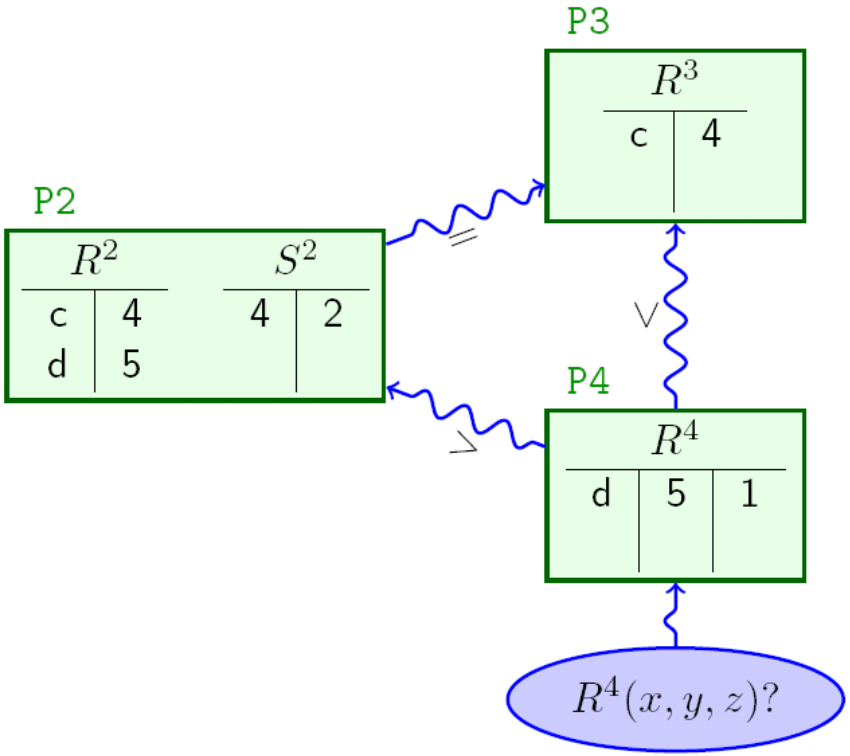
$$\Sigma(\mathsf{P1}, \mathsf{P2}) = \{\forall x \forall y (R^2(x, y) \rightarrow R^1(x, y))\}$$

$$\Sigma(\mathsf{P2}, \mathsf{P3}) = \{\forall x \forall y (R^2(x, y) \wedge R^3(x, y) \rightarrow \mathbf{false})\}$$

$$\Sigma(\mathsf{P4}, \mathsf{P2}) = \{\forall x \forall y \forall z (R^2(x, y) \wedge S^2(y, z) \rightarrow R^4(x, y, z))\}$$

$$\Sigma(\mathsf{P4}, \mathsf{P3}) = \{\forall x \forall y (R^3(x, y) \rightarrow \exists z R^4(x, y, z))\}$$

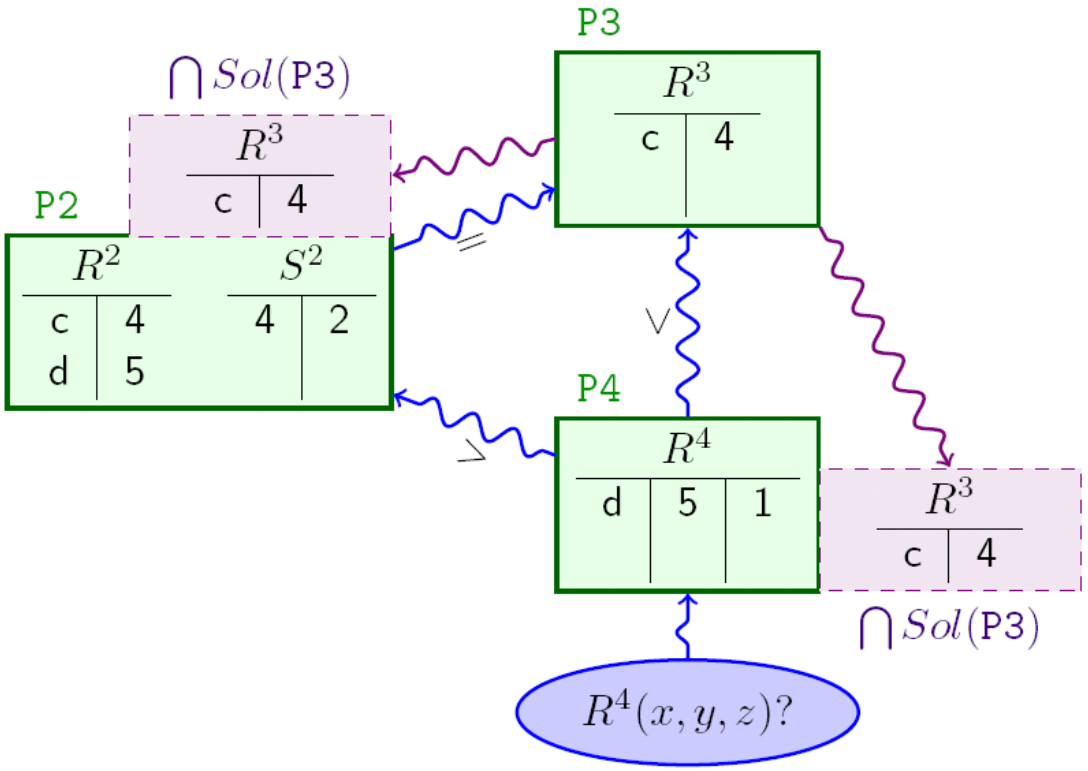
We are left with:



$$\Sigma(\mathsf{P2}, \mathsf{P3}) = \{\forall x \forall y (R^2(x, y) \wedge R^3(x, y) \rightarrow \mathbf{false})\}$$

$$\Sigma(\mathsf{P4}, \mathsf{P2}) = \{\forall x \forall y \forall z (R^2(x, y) \wedge S^2(y, z) \rightarrow R^4(x, y, z))\}$$

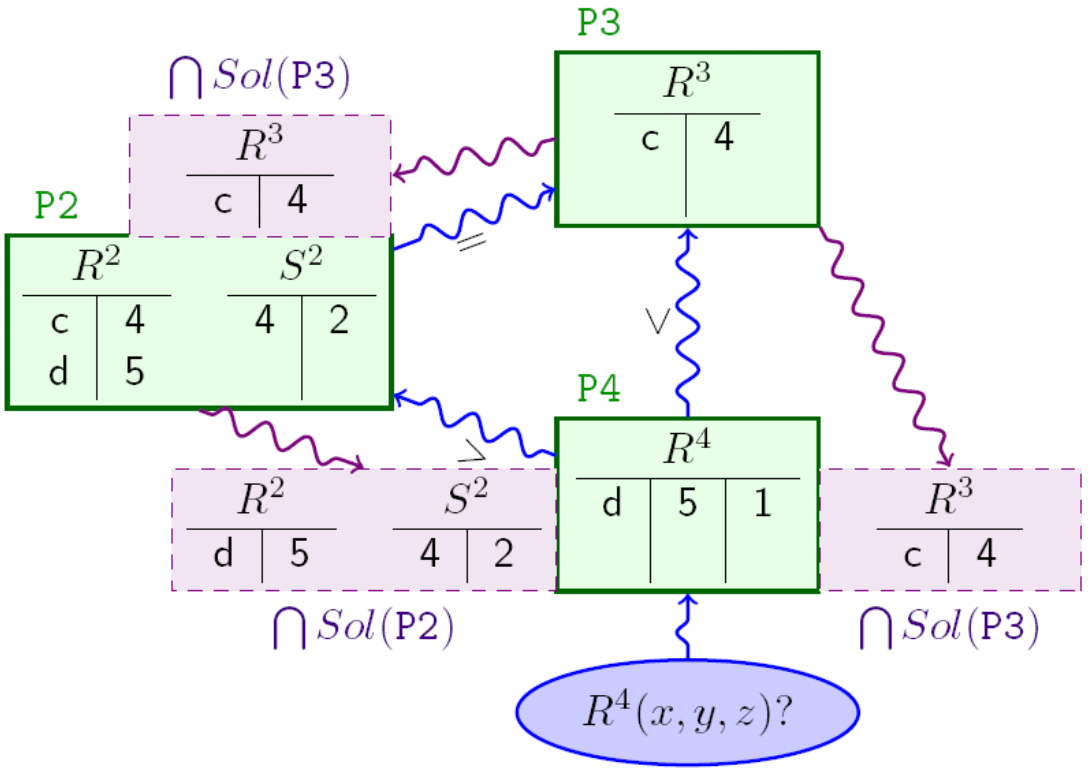
$$\Sigma(\mathsf{P4}, \mathsf{P3}) = \{\forall x \forall y (R^3(x, y) \rightarrow \exists z R^4(x, y, z))\}$$



$$\Sigma(\mathsf{P2},\mathsf{P3}) = \{\forall x \forall y (R^2(x,y) \wedge R^3(x,y) \rightarrow \mathbf{false})\}$$

$$\Sigma(\mathsf{P4},\mathsf{P2}) = \{\forall x \forall y \forall z (R^2(x,y) \wedge S^2(y,z) \rightarrow R^4(x,y,z))\}$$

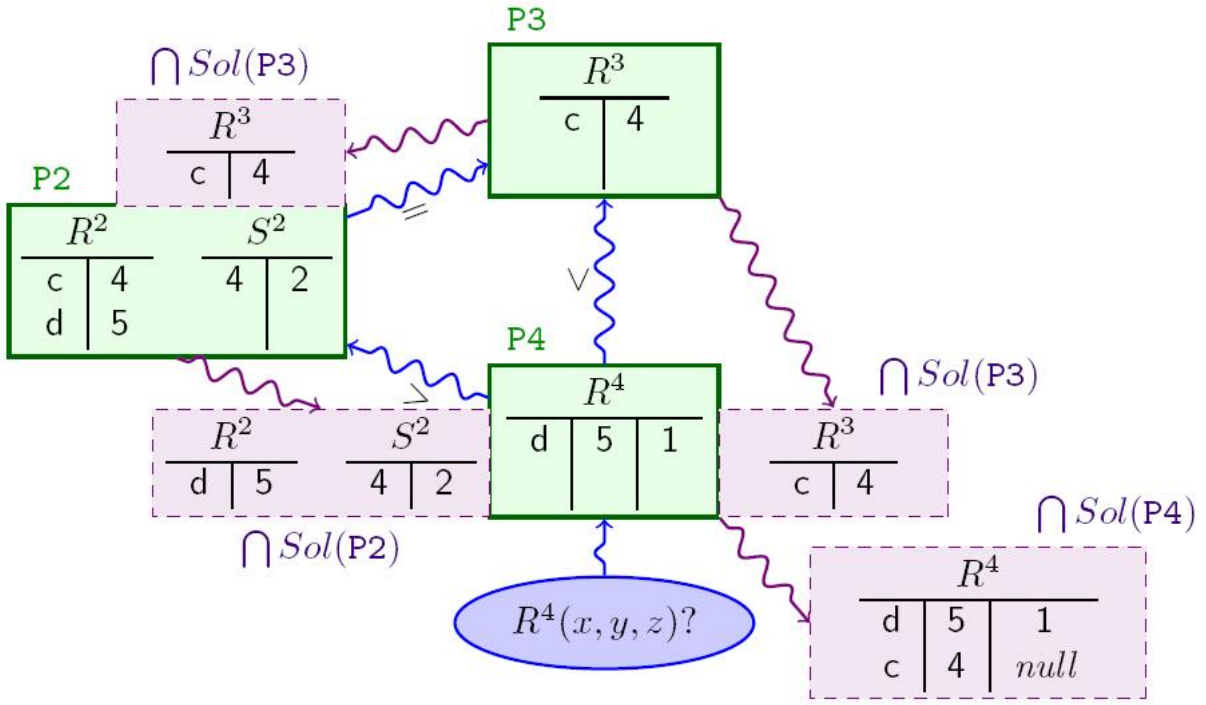
$$\Sigma(\mathsf{P4},\mathsf{P3}) = \{\forall x \forall y (R^3(x,y) \rightarrow \exists z R^4(x,y,z))\}$$



$$\Sigma(\mathsf{P2},\mathsf{P3}) = \{\forall x \forall y (R^2(x,y) \wedge R^3(x,y) \rightarrow \mathbf{false})\}$$

$$\Sigma(\mathsf{P4},\mathsf{P2}) = \{\forall x \forall y \forall z (R^2(x,y) \wedge S^2(y,z) \rightarrow R^4(x,y,z))\}$$

$$\Sigma(\mathsf{P4},\mathsf{P3}) = \{\forall x \forall y (R^3(x,y) \rightarrow \exists z R^4(x,y,z))\}$$



$$\Sigma(\mathsf{P2},\mathsf{P3}) = \{\forall x \forall y (R^2(x,y) \wedge R^3(x,y) \rightarrow \mathbf{false})\}$$

$$\Sigma(\mathsf{P4},\mathsf{P2}) = \{\forall x \forall y \forall z (R^2(x,y) \wedge S^2(y,z) \rightarrow R^4(x,y,z))\}$$

$$\Sigma(\mathsf{P4},\mathsf{P3}) = \{\forall x \forall y (R^3(x,y) \rightarrow \exists z R^4(x,y,z))\}$$

PCAs from P4: $(d,5,1)$ and $(c,4,null)$

Our Framework

Our data exchange constraints (DECs):

- *Universal data exchange constraint* (UDEC) between P1, P2:

$$\forall \bar{x}(\bigwedge_{i=1}^{n} R_i(\bar{x}_i) \longrightarrow (\bigvee_{j=1}^{m} Q_j(\bar{y}_j) \vee \varphi))$$

  $R_i, Q_j \in \mathcal{R}(\mathtt{P1}) \cup \mathcal{R}(\mathtt{P2})$, and φ a disjunction of built-ins

  We can have a predicates of both peers on both sides of implication ....

- *Referential data exchange constraint* (RDEC) between peers P1, P2:

$$\forall \bar{x}(R(\bar{x}) \longrightarrow \exists \bar{y}\ Q(\bar{x}', \bar{y}))$$

  $R, Q \in \mathcal{R}(\mathtt{P1}) \cup \mathcal{R}(\mathtt{P2})$, and $\bar{x}' \subseteq \bar{x}$
  (conjunctions can be added)

- Incomplete information is represented by means of nulls

Actually, they follow a FO semantics that is a "logical reconstruction" of IC satisfaction with nulls in the SQL Standard (Bravo, Bertossi; IIDB 2006)

- The reason for not having joins in the consequents

This restriction can be lifted if labeled nulls or constants are used instead

Other semantics for incomplete databases (and null values) could be easily adopted in our framework

Being able to introduce more complex DECs

- It is also possible to impose local ICs on peers' instances

They can be uniformly handled as before by means of DECs of the form $\Sigma(\mathtt{P}, \mathtt{P})$ and an $=$-trust relationship

The Semantics

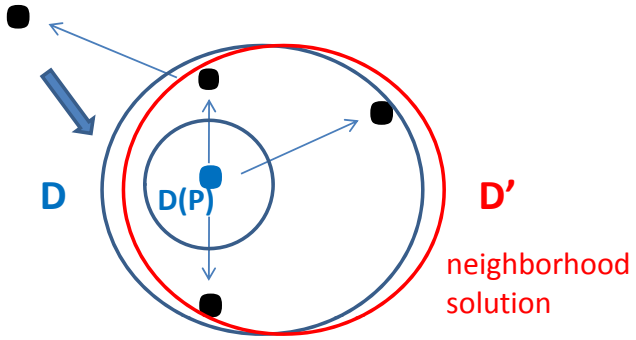
We define the solution instances of a peer; in two steps:

1. First locally for a peer and its neighbors: Neighborhood solutions
2. Using 1., recursively consider transitive relations to other peers

Start from a peer $\mathtt{P}$ with local instance $D(\mathtt{P})$

Can be seen as extended as an instance D for neighborhood $\mathcal{N}(\mathtt{P})$, over the union schema

D may not satisfy the DECs from $\mathtt{P}$ to its neighbors, and inconsistencies have to be solved, minimally ...



Let D' be an instance for $\mathcal{N}(\mathtt{P})$

D' is a neighborhood solution for $\mathtt{P}$ and D:

1. D' satisfies the DECs and local constraints of $\mathtt{P}$

$$D' \models \bigcup_{\mathtt{P}' \in \mathcal{N}(\mathtt{P})} \Sigma(\mathtt{P}, \mathtt{P}')$$

2. The data in D' associated to the peers that $\mathtt{P}$ trusts more than itself is the same as the one in D

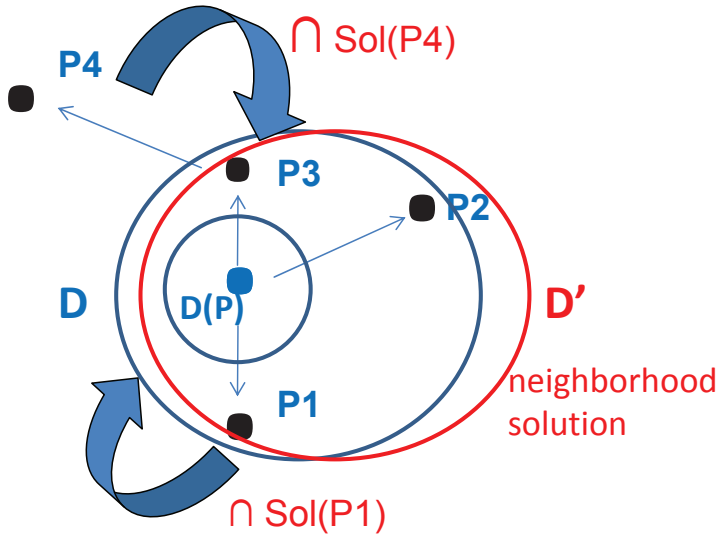
3. There is no instance D'' that satisfies 1. and 2. and is "closer" to D than D'

Distance can be defined in different ways: Ours

- Changes are minimal under set inclusion of sets of tuples (Arenas, Bertossi, Chomicki, PODS 1999), and
- Referential DECs are repaired by insertions of a single $null$, as in (Bravo, Bertossi; IIDB 2006)

  The repair semantics takes $null$ into account

There may be more than one neighborhood solution D'



Now we define the solution instances (solutions) for P

Now consider transitive relationships too

Transitive peers will contribute to the creation of the D above; which will lead to the D's, etc.

Peers will be passing back the intersection of their own solutions

Transitive peers will contribute to D with the intersection of their own solutions, iteratively/recursively ...

Let $D(\mathtt{P})$ be the database instance for peer $\mathtt{P}$

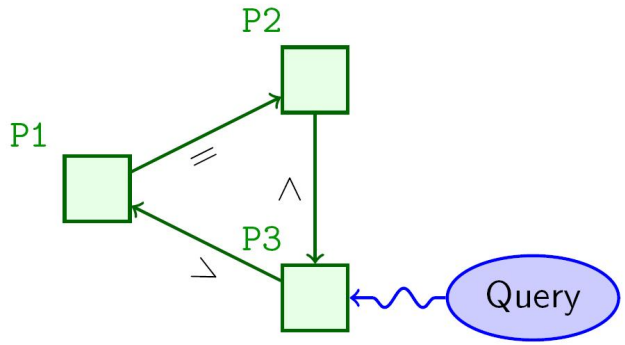
A solution for $\mathtt{P}$ can be recursively defined as:

- If $\mathtt{P}$ has no DECs, then the solution is $D(\mathtt{P})$
- Otherwise:
  1. Let D be a database instance that is the union of $D(\mathtt{P})$, and, for each neighbor, the intersection of its solutions:
     $$D = D(\mathtt{P}) \cup \bigcup_{\substack{\mathtt{P}' \in \mathcal{N}(\mathtt{P}) \\ \mathtt{P}' \neq \mathtt{P}}} \bigcap Sol(\mathtt{P}')$$
     (an instance for the union schema around $\mathtt{P}$)
  2. Let D' be a neighborhood solution for $\mathtt{P}$ and D

  Then, D' restricted to the schema of $\mathtt{P}$ is a solution for $\mathtt{P}$

For this definition to work, we restrict ourselves to acyclic peer data exchange systems, i.e. graphs of neighbors is acyclic (not necessarily the DECs)

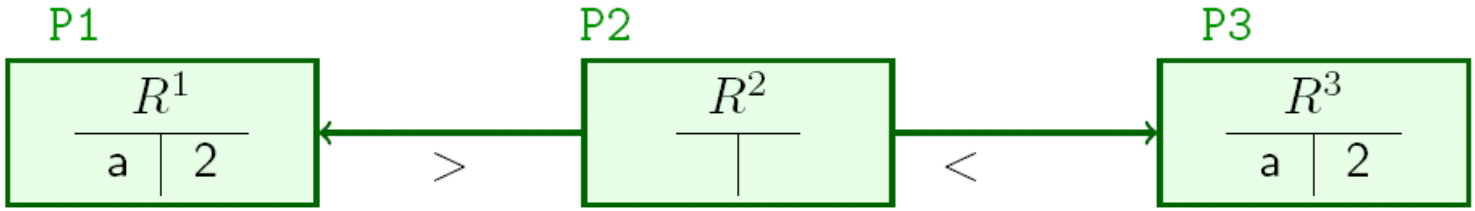
We also want to avoid the following:



P3 needs the intersection of P1's solutions $\Rightarrow$ P1 needs the intersection of P2's solutions $\Rightarrow$ P2 needs the intersection of P3's solutions $\Rightarrow \cdots$

Cycles can be detected by using a query identifier that is propagated as an annotation and detected

It might be the case that a peer has no solution:



DECs:

$$\Sigma(\mathsf{P2},\mathsf{P1}) = \{\forall x \forall y (R^1(x,y) \rightarrow R^2(x,y))\}$$

$$\Sigma(\mathsf{P2},\mathsf{P3}) = \{\forall x \forall y (R^2(x,y) \wedge R^3(x,y) \rightarrow \mathbf{false})\}$$

Peer P2 trusts P1 and P3 more than itself, but both provide contradictory information

First-order query $\mathcal{Q}(\bar{x}) \in L(\mathtt{P})$ posed to peer $\mathtt{P}$:

> The ground tuple $\bar{t}$ is a *peer consistent answer* to $\mathcal{Q}$ from $\mathtt{P}$ iff $D' \models \mathcal{Q}[\bar{t}]$ for every $D' \in Sol(\mathtt{P})$

Peer consistent query answering is decidable

This relies on our null-based repair semantics

Even with acyclic neighbors' graph it be undecidable depending on interaction between DECs if arbitrary constants are used

Theorem: Given a peer $\mathtt{P}$ and, for each peer, the intersections of its solutions, deciding if a tuple is a peer consistent answer from $\mathtt{P}$ to a FO query is Π_2^P-complete

Logic Programs for Peers' Solutions

- Answer set programs have been used to specify and compute repairs of databases that are inconsistent wrt ICs

(Barcelo, Bertossi; PADL'03), (Barcelo, Bertossi, Bravo; LNCS 2582)

- Those programs can be adapted to our framework, and there is a one-to-one correspondence between their answer sets (stable models) and the solutions of peer

  - The trust relationships, DECs and local ICs have to be taken into consideration

- They provide a compact representation of the class of solutions and the possibility of reasoning about that class

• The program uses annotation constants to indicate the atoms that may virtually inserted or deleted in order to restore consistency:

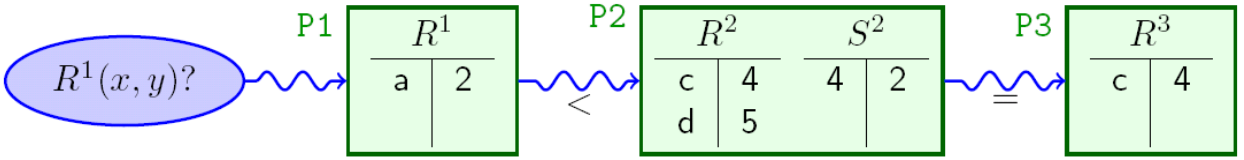
Annotation	Atom	The tuple $P(\bar{a})$ is ...
$\mathbf{t}$	$P\_(\bar{a}, \mathbf{t})$	made true (inserted)
$\mathbf{f}$	$P\_(\bar{a}, \mathbf{f})$	made false (deleted)
$\mathbf{t}^\star$	$P\_(\bar{a}, \mathbf{t}^\star)$	true or becomes true
$\mathbf{f}^\star$	$P\_(\bar{a}, \mathbf{f}^\star)$	false or becomes false
$\mathbf{t}^{\star\star}$	$P\_(\bar{a}, \mathbf{t}^{\star\star})$	true in the solution

All of them needed if there are interacting DECs for a peer; and several repair steps become necessary

Example:

$$\Sigma(\mathtt{P1}, \mathtt{P2}) = \{\forall x \forall y (R^2(x,y) \rightarrow R^1(x,y))\}$$

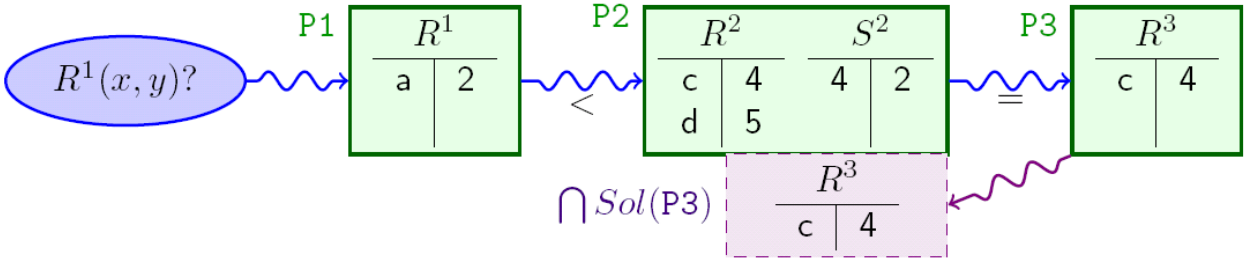
$$\Sigma(\mathtt{P2}, \mathtt{P3}) = \{\forall x \forall y (R^2(x,y) \wedge R^3(x,y) \rightarrow \mathbf{false})\}$$



- Peer P3 has no DECs, therefore its only solution is $D(\mathtt{P3})$

$$\Sigma(\mathsf{P1},\mathsf{P2}) = \{\forall x \forall y (R^2(x,y) \rightarrow R^1(x,y))\}$$

$$\Sigma(\mathsf{P2},\mathsf{P3}) = \{\forall x \forall y (R^2(x,y) \wedge R^3(x,y) \rightarrow \mathbf{false})\}$$



- To find the solution of peer P2 we can use its solution program!

$dom(a).$ $dom(c).$ $\ldots$
$R^3(c,4)$ $R^2(c,4)$ $R^2(d,5)$ $S^2(4,2)$
$R^3_{\_}(x,y,\mathbf{f}) \vee R^2_{\_}(x,y,\mathbf{f}) \leftarrow R^2_{\_}(x,y,\mathbf{t}^\star),\ R^3_{\_}(x,y,\mathbf{t}^\star), x \neq null, y \neq null.$
$R^3_{\_}(x,y,\mathbf{t}^\star) \leftarrow R^3_{\_}(x,y,\mathbf{t}).$
$R^3_{\_}(x,y,\mathbf{t}^\star) \leftarrow R^3(x,y).$
$R^3_{\_}(x,y,\mathbf{f}^\star) \leftarrow R^3_{\_}(x,y,\mathbf{f}).$
$R^3_{\_}(x,y,\mathbf{f}^\star) \leftarrow dom(x), dom(y),\ not\ R^3(x,y).$
$R^2_{\_}(x,y,\mathbf{t}^{\star\star}) \leftarrow R^2_{\_}(x,y,\mathbf{t}^\star),\ not\ R^2_{\_}(x,y,\mathbf{f}).$
$\leftarrow R^3_{\_}(x,y,\mathbf{t}), R^3_{\_}(x,y,\mathbf{f}).$

} (Similarly for R^2)

$dom(a).\quad dom(c).\quad \ldots$

$R^3(c,4)\quad R^2(c,4)\quad R^2(d,5)\quad S^2(4,2)$

$R^3\_(x,y,\mathbf{f}) \vee R^2\_(x,y,\mathbf{f}) \leftarrow R^2\_(x,y,\mathbf{t}^\star),\ R^3\_(x,y,\mathbf{t}^\star), x \neq null, y \neq null.$

$R^3\_(x,y,\mathbf{t}^\star) \leftarrow R^3\_(x,y,\mathbf{t}).$
$R^3\_(x,y,\mathbf{t}^\star) \leftarrow R^3(x,y).$
$R^3\_(x,y,\mathbf{f}^\star) \leftarrow R^3\_(x,y,\mathbf{f}).$
$R^3\_(x,y,\mathbf{f}^\star) \leftarrow dom(x), dom(y),\ not\ R^3(x,y).$
$R^2\_(x,y,\mathbf{t}^{\star\star}) \leftarrow R^2\_(x,y,\mathbf{t}^\star),\ not\ R^2\_(x,y,\mathbf{f}).$
$\leftarrow R^3\_(x,y,\mathbf{t}), R^3\_(x,y,\mathbf{f}).$

} (Similarly for R^2)

- This program has two answer sets
- Collecting atoms $\mathbf{t}^{\star\star}$ we get (neighborhood) solutions:

D_1 :

R^2	
d	5

S^2	
4	2

R^3	
c	4

D_2 :

R^2	
c	4
d	5

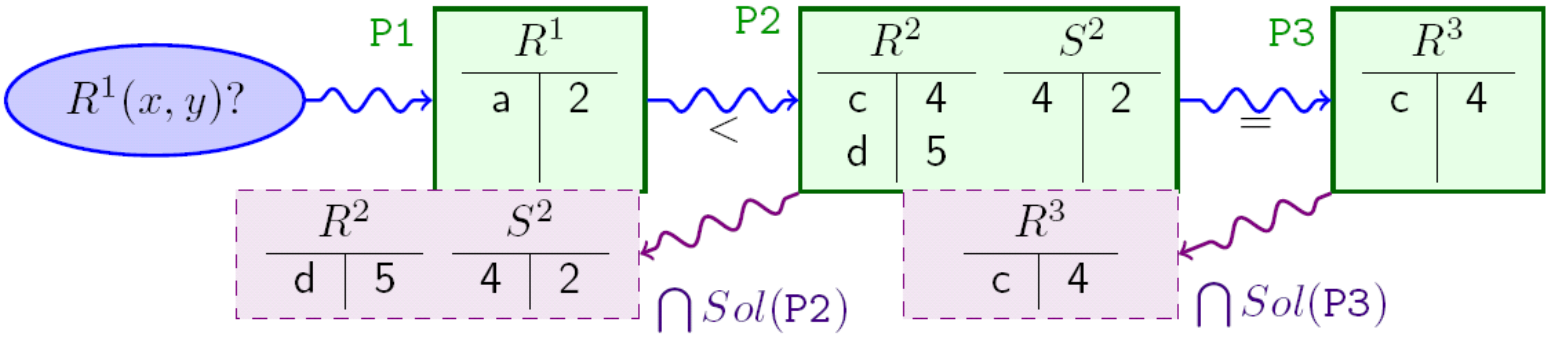
S^2	
4	2

R^3

$\bigcap$?: Just skeptical query answering: $Ans(x,y) \leftarrow R^2(x,y,\mathbf{t}^{\star\star})$

$$\Sigma(\mathsf{P1},\mathsf{P2}) = \{\forall x \forall y (R^2(x,y) \rightarrow R^1(x,y))\}$$

$$\Sigma(\mathsf{P2},\mathsf{P3}) = \{\forall x \forall y (R^2(x,y) \wedge R^3(x,y) \rightarrow \mathbf{false})\}$$



- To find the solution of peer P1 we can use its solution program!

$dom(a).\quad dom(c).\quad \ldots$
$R^1(a,2)\quad R^2(d,5)\quad S^2(4,2)$
$R^1\_(x,y,\mathbf{t}) \leftarrow R^2\_(x,y,\mathbf{t}^\star),\ R^1\_(x,y,\mathbf{f}^\star), x \neq null, y \neq null.$
$R^1\_(x,y,\mathbf{t}^\star) \leftarrow R^1\_(x,y,\mathbf{t}).$
$R^1\_(x,y,\mathbf{t}^\star) \leftarrow R^1(x,y).$
$R^1\_(x,y,\mathbf{f}^\star) \leftarrow R^1\_(x,y,\mathbf{f}).$
$R^1\_(x,y,\mathbf{f}^\star) \leftarrow dom(x), dom(y),\ not\ R^1(x,y).$
$R^1\_(x,y,\mathbf{t}^{\star\star}) \leftarrow R^1\_(x,y,\mathbf{t}^\star),\ not\ R^1\_(x,y,\mathbf{f}).$
$\leftarrow R^1\_(x,y,\mathbf{t}), R^1\_(x,y,\mathbf{f}).$

} (Similarly for R^2)

- This program has one answer set; then one neighbor solution
- Collecting atoms with $\mathbf{t}^{\star\star}$ we get this (neighborhood) solution:

D_1 :

R^1	
a	2
d	5

R^2	
d	5

S^2	
4	2

- The query $R^1(x, y)?$ can also be added to the answer set program:

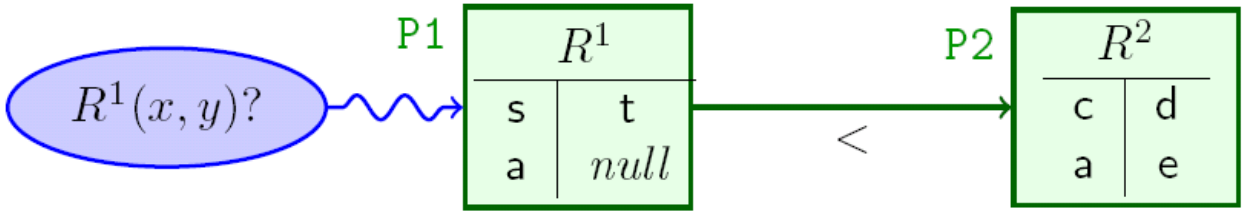
$$Ans(x, y) \leftarrow R^1(x, y, \mathbf{t}^{\star\star})$$

- Final PCAs are obtained: $\{Ans(a, 2), Ans(d, 5)\}$

Example:

$$\Sigma(\mathtt{P1}, \mathtt{P2}) = \{\forall xy\ (R^2(x,y) \rightarrow \exists z\ R^1(x,z))\}$$

$$IC(\mathtt{P1}) = \{\forall xyz\ (R^1(x,y) \wedge R^1(x,z) \rightarrow y = z)\}$$



$dom(a).\ dom(b).\ \ldots\ R^1(a, null).\ R^1(s,t).\ R^2(c,d).\ R^2(a,e).$
$R^1\_(x, null, \mathbf{t}) \leftarrow R^2\_(x, \mathbf{t}^\star),\ not\ aux(x), x \neq null.$
$aux(x) \leftarrow R^1(x, null),\ not\ R^1\_(x, null, \mathbf{f}).$
$aux(x) \leftarrow R^1(x, y, \mathbf{t}^\star),\ not\ R^1\_(x, y, \mathbf{f}), x \neq null, y \neq null.$
$R^1(x, y, \mathbf{f}) \vee R^1(x, z, \mathbf{f}) \leftarrow R^1(x, y, \mathbf{t}^\star), R^1(x, z, \mathbf{t}^\star),\ x \neq null,\ y \neq z.$

The solution with the program is:

R^1	
s	t
a	*null*
c	*null*

A Note of Referential DECs

- A set of DECs and ICs for a peer is Ref-Acyclic if there is no cycles through referential DECs or ICs

- An example of ref-acyclic

$$\Sigma(\mathtt{P1},\mathtt{P2}) = \{\forall xy\ (R^1(x,y) \rightarrow R^2(x,y)),\\ \forall xy(R^2(x,y) \rightarrow R^1(x,y))\}$$
$$\Sigma(\mathtt{P2},\mathtt{P1}) = \{\forall x(S^2(x) \rightarrow \exists y S^1(x,y))\}$$

- An example of non-ref-acyclic

$$\Sigma(\mathtt{P1},\mathtt{P2}) = \{\forall xy\ (R^1(x,y) \rightarrow \exists z\ R^2(x,z)),\\ \forall xy\ (R^2(x,y) \rightarrow R^1(x,y))\}$$

Theorem: For a ref-acyclic set of DECs and ICs for a peer, there is a one-to-one correspondence between answer sets and the neighborhood solutions of the peer

A Special Case and Common Case

- Unrestricted Import Case:
  - The DECs are such that data is only imported to the peer (nothing is deleted)
  - All peers trust other peers more than themselves
  - There are no local ICs
- Nice properties:
  - A unique solution always exist
  - The solution program can be replaced by a non-disjunctive program
  - The solution can be computed in polynomial time

Optimizations and Relaxing Conditions

- It is possible to relax the conditions of ref-acyclicity and acyclicity of the neighbors' graph:
  - A sensible semantics can be provided
  - Correct and complete solution-programs can be given

For example when:

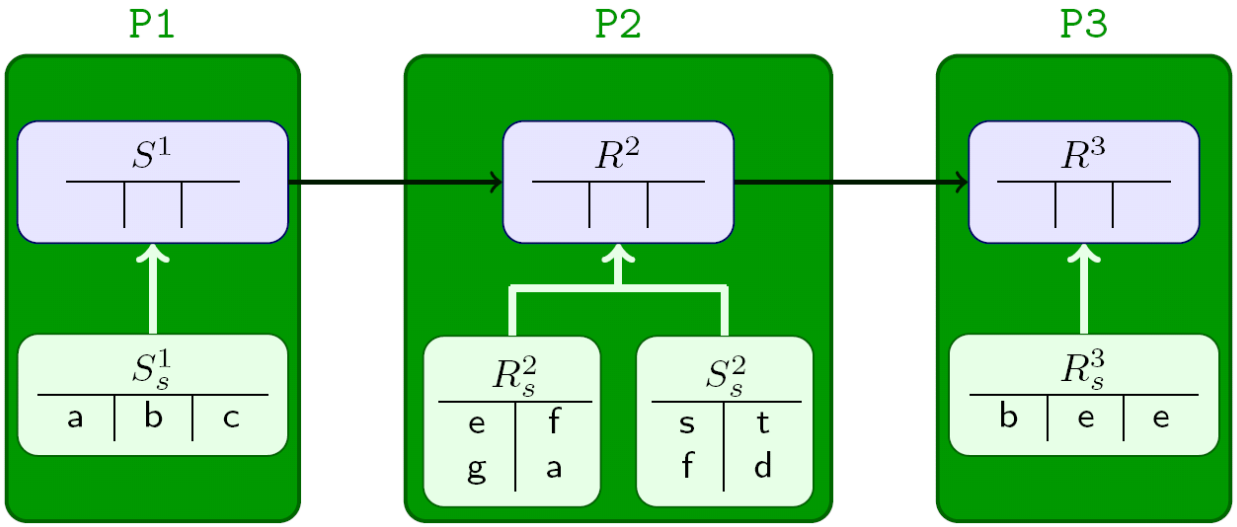
  - The cycles in the graph are not relevant to the query
  - Even if the DECs and ICs are not ref-acyclic, depending on the interaction with the trust relationships, the solution-program can provide exactly the set of solutions

- Instead of requesting all the data of the neighboring sources:
  - restrict to the data that is relevant to check the DECs that have an impact on a query

Related Work

(Calvanese et al.; DBISP2P 2003, DBPL 2005, PODS 2004),
(Franconi et al.; P2P&DB 2004)

- Based on epistemic logic
- DECs are of the form: $cq_i \rightarrow cq_j$ where cq_i and cq_j are conjunctive queries over `Pi` and `Pj`'s schemas, resp.
- No trust relationships
  - Implicitly: peers trust themselves less than other peers
- Local ICs violations are avoided
  - A peer that is inconsistent wrt its local ICs is ignored
  - New atoms are added into a peer by interaction with other peers only if this does not produce a local IC violation



- GAV Local Mappings:

  $\forall xyz(S^1_s(x,y,z) \rightarrow S^1(x,y,z))$

  $\forall xyz(R^2_s(x,y) \wedge R^2_s(y,z) \rightarrow R^2(x,y,z))$

  $\forall xyz(R^3_s(x,y,z) \rightarrow R^3(x,y,z))$

- DECs:

  $\forall xy(R^2(x,y,z) \rightarrow \exists w R^1(x,y,w))$

  $\forall xy(R^3(x,y,y) \rightarrow \exists uv R^3(u,x,v))$

If peer P1 receives a query, it will need the following theory in epistemic logic:

$$\left.\begin{array}{l}\mathbf{K_1}(\forall xyz(S^1_s(x,y,z) \rightarrow S^1(x,y,z))) \\ \forall xy(\mathbf{K_2}(R^2(x,y,z)) \rightarrow \mathbf{K_1}(\exists w R^1(x,y,w)))\end{array}\right\} \text{Specification of P1}$$

$$\left.\begin{array}{l}\mathbf{K_2}(\forall xyz(R^2_s(x,y) \wedge R^2_s(y,z) \rightarrow R^2(x,y,z))) \\ \forall xy(\mathbf{K_3}(R^3(x,y,y)) \rightarrow \mathbf{K_2}(\exists uv R^3(u,x,v)))\}\end{array}\right\} \text{Specification of P2}$$

$$\left.\begin{array}{l}\mathbf{K_3}(\forall xyz(R^3_s(x,y,z) \rightarrow R^3(x,y,z)))\end{array}\right\} \text{Specification of P3}$$

$\mathbf{K_i}\varphi$ can be interpreted as φ is known by peer Pi

"Known" data is passed to other peers

A tuple $\bar{t}$ is a peer consistent answer to a query $\mathcal{Q}$ posed to peer Pi if $\mathbf{K_i}\mathcal{Q}(\bar{t})$ is a logical consequence of the epistemic theory

- Pros: semantics can be applied in the presence of cycles
- Cons: requires (possible massive and complex) reasoning by peer P1
    - Requires data, mappings and DECs not only of neighbors, but of all accessible peers

Our approach can be easily adapted so that each peer is a local data integration system

Conclusions

- We have provided a semantics for peer data exchange systems
  - Respects the modularity and independence of the different peers
  - Takes trust relationships into consideration
  - Uses $null$ to repair referential DECs and ICs considering the same semantics of satisfaction of constraints as commercial DBMSs
- We can say that data movement is triggered by query answering, driven by inconsistency, but guided by trust relationships

Interesting Research: More complex trust relationships, declaratively specified, and to be logically integrated with DECs

- We have provided answer set programs that can be used to specify solutions for a peer and compute PCAs

- Each peer has a single and fixed facts-free logic program, for all queries

Only facts depend on query and other peers' data