Principles of peer-to-peer data integration

Maurizio Lenzerini

Dipartimento di Informatica e Sistemistica “Antonio Ruberti”

Università di Roma “La Sapienza”

Invited talk at DIWeb 2004

Riga, Latvia – June 2004

Three data integration architectures

- Mediator-based data integration

  The traditional architecture for centralized, virtual data integration

- Data exchange

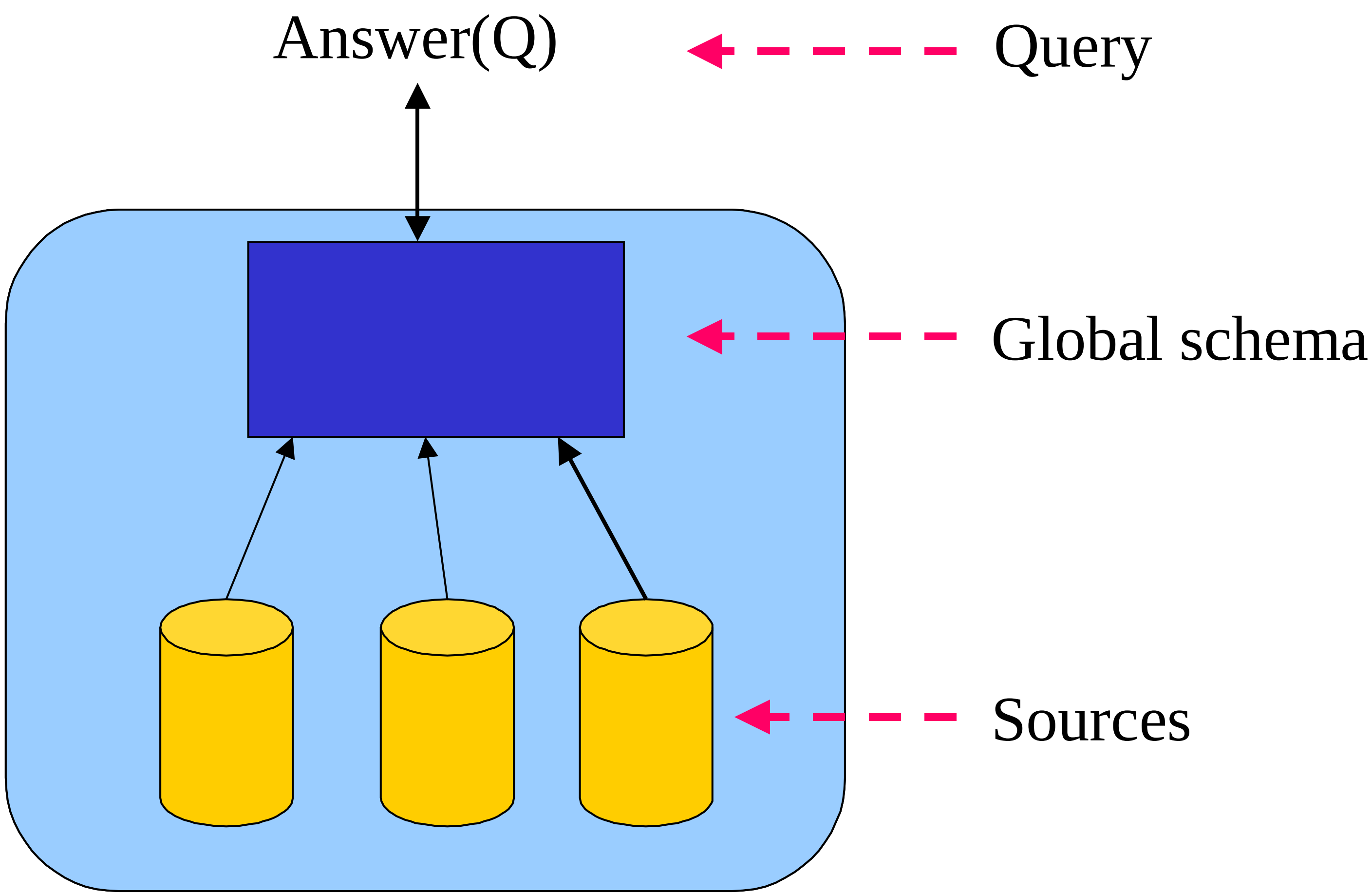
  Materialization of data from a source database to a target database

- Peer-to-peer data integration

  Decentralized, dynamic data-centric coordination between autonomous organization

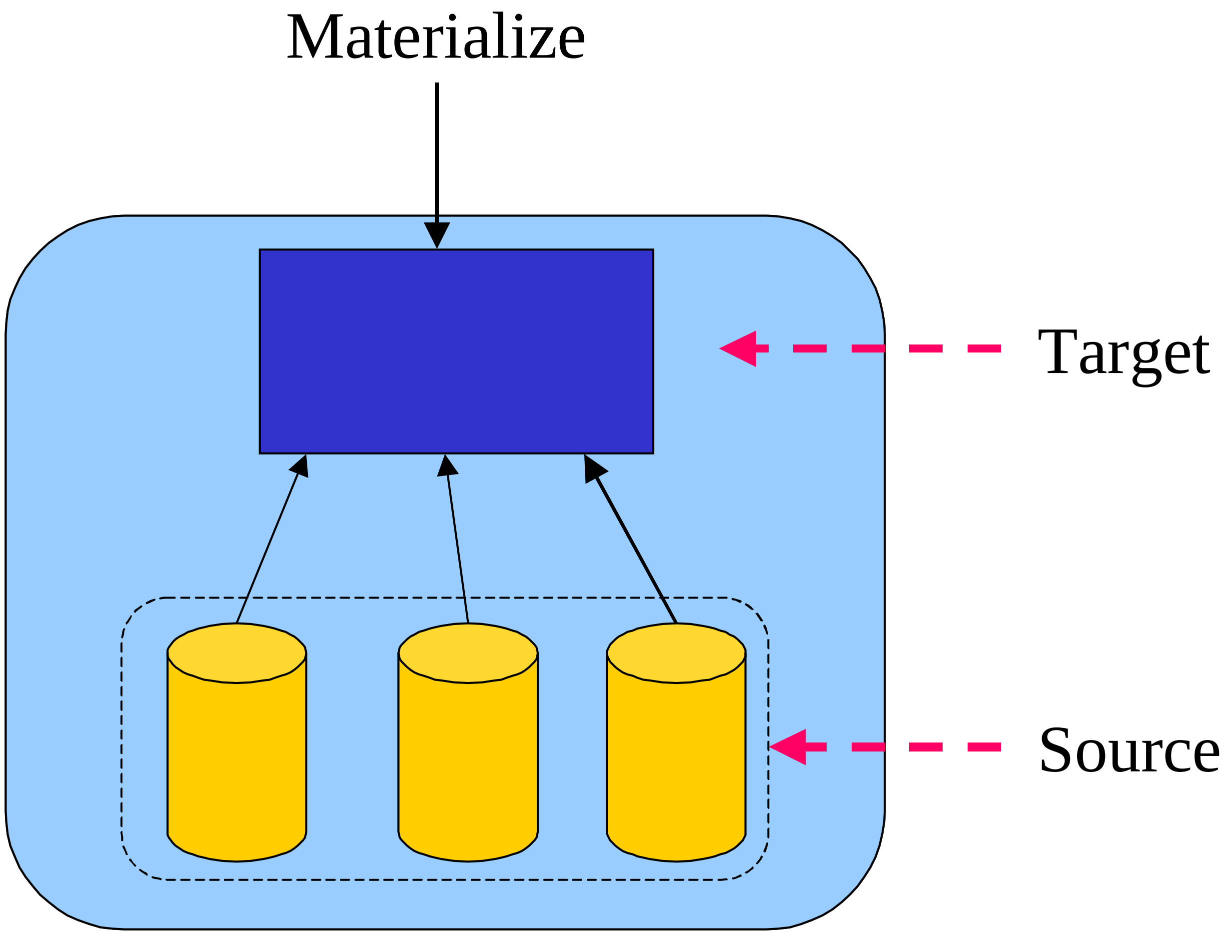
Mediator-based data integration

- Mapping between sources and global schema
- Queries over the global schema



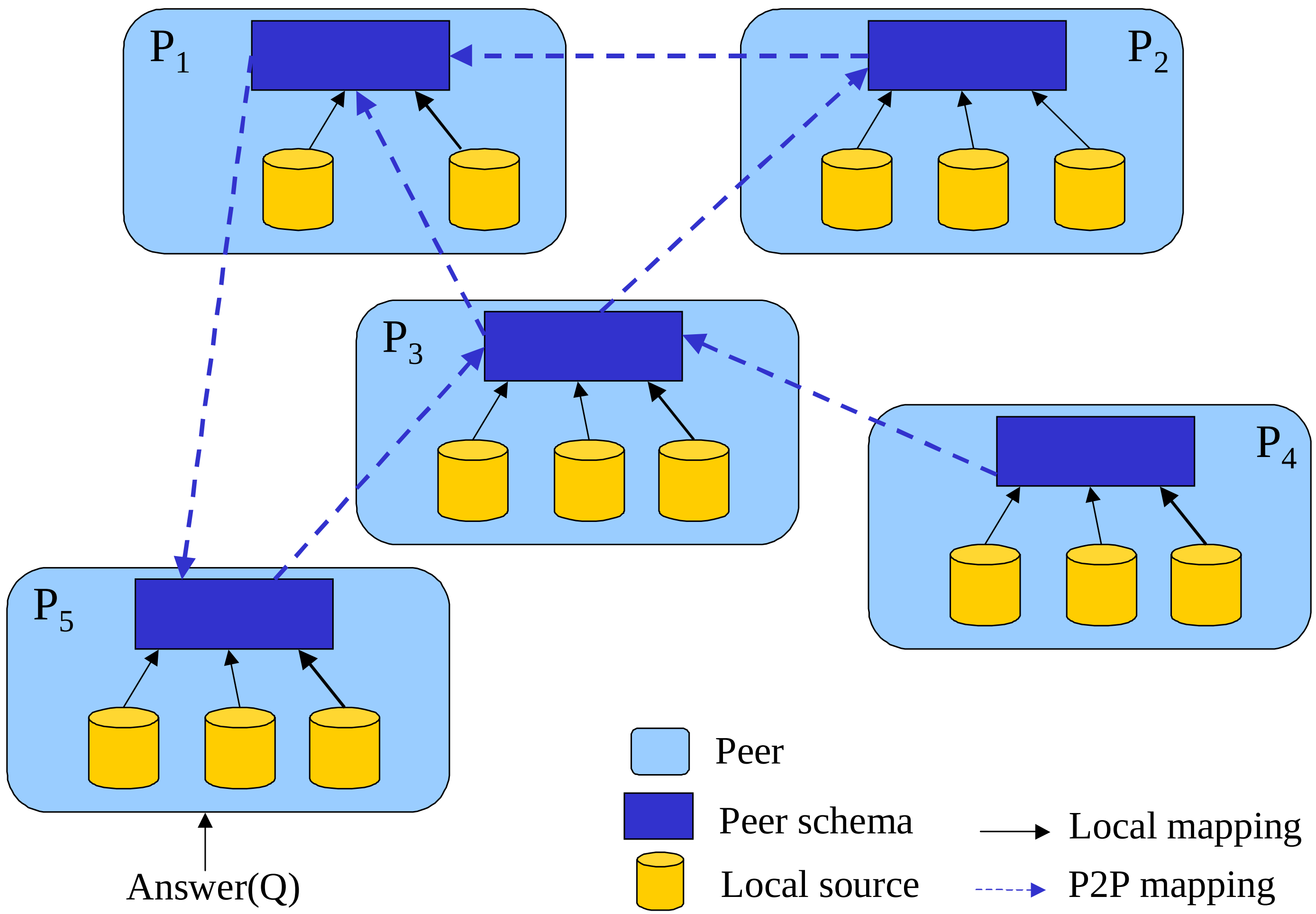
Data exchange

- Mapping between sources and target schema
- Materialization according to the target schema



Peer-to-peer data integration

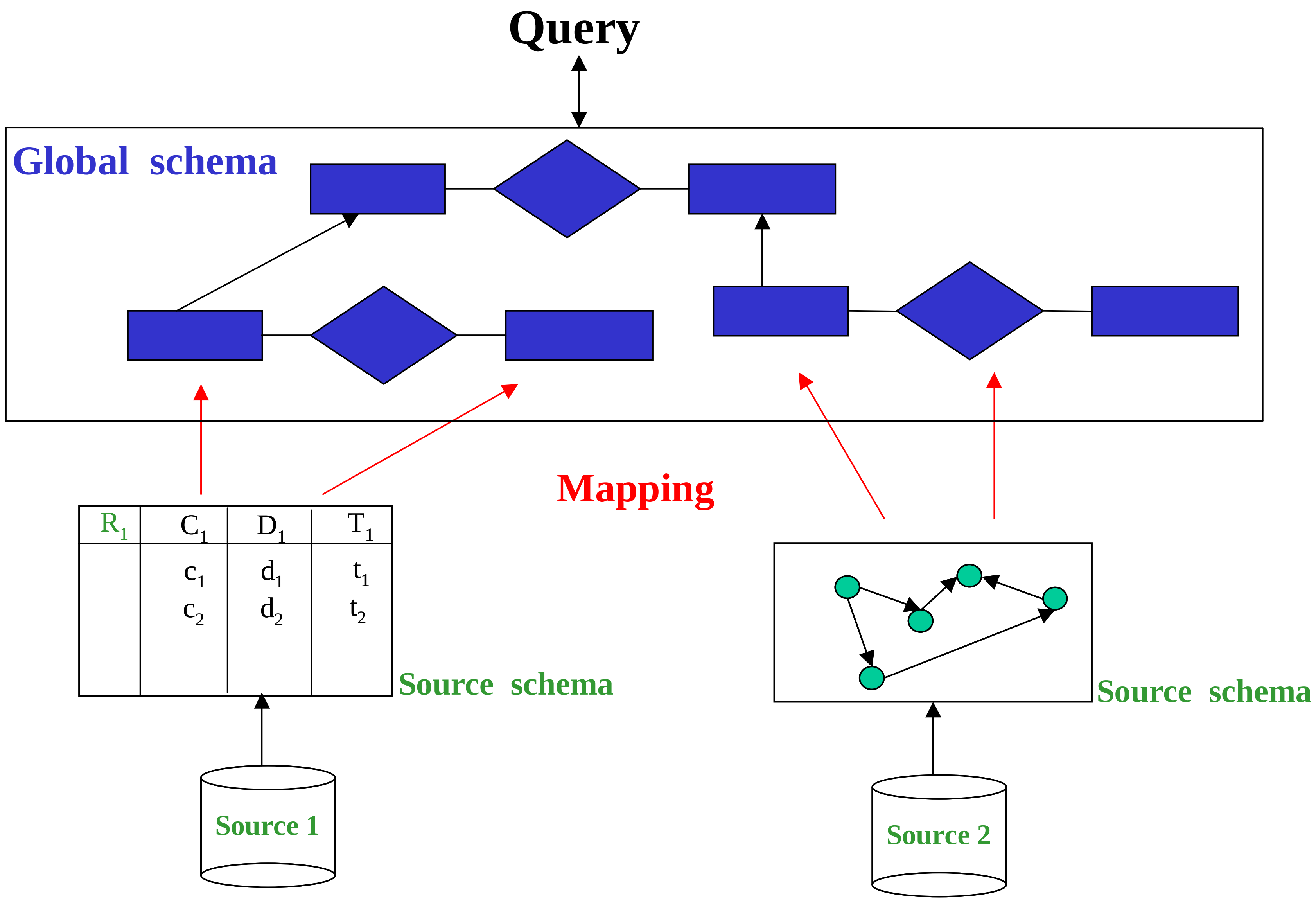
- Several peers
- Local mappings and P2P mappings
- Each query over one peer



Outline

- Peer-based Distributed Information Systems
- Mediator-based data integration
- Data exchange
- P2P data integration
- Conclusions

Data integration



Formal framework for data integration

A data integration system $\mathcal{I}$ is a triple $\langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$, where

- $\mathcal{G}$ is the global schema

  The global schema is a logical theory over an alphabet $\mathcal{A}_{\mathcal{G}}$

- $\mathcal{S}$ is the source schema

  The source schema is constituted simply by an alphabet $\mathcal{A}_{\mathcal{S}}$ disjoint from $\mathcal{A}_{\mathcal{G}}$

- $\mathcal{M}$ is the mapping between $\mathcal{S}$ and $\mathcal{G}$

  Different approaches to the specification of mapping

Semantics of a data integration system

Which are the databases that satisfy $\mathcal{I}$, i.e., which are the logical models of $\mathcal{I}$?

The databases that satisfy $\mathcal{I}$ are logical interpretations for $\mathcal{A}_{\mathcal{G}}$ (called global databases). We refer only to databases over a fixed infinite domain Γ of constants.

Let $\mathcal{C}$ be **a source database** over Γ (also called source model), fixing the extension of the predicates of $\mathcal{A}_{\mathcal{S}}$ (thus modeling the data present in the sources).

The set of models of (i.e., databases for $\mathcal{A}_{\mathcal{G}}$ that satisfy) $\mathcal{I}$ relative to $\mathcal{C}$ is:

$$sem^{\mathcal{C}}(\mathcal{I}) = \{\, \mathcal{B} \mid \mathcal{B} \text{ is a } \mathcal{G}\text{-model (i.e., a global database that is legal wrt } \mathcal{G}) \text{ and is an } \mathcal{M}\text{-model wrt } \mathcal{C} \text{ (i.e., satisfies } \mathcal{M} \text{ wrt } \mathcal{C}) \,\}$$

What it means to satisfy $\mathcal{M}$ wrt $\mathcal{C}$ depends on the nature of the mapping $\mathcal{M}$.

Semantics of queries to $\mathcal{I}$

A query q of arity n is a formula with n free variables.

If $\mathcal{D}$ is a database, then $q^{\mathcal{D}}$ denotes the extension of q in $\mathcal{D}$ (i.e., the set of n-tuples that are valuations in Γ for the free variables of q that make q true in $\mathcal{D}$).

If q is a query of arity n posed to a data integration system $\mathcal{I}$ (i.e., a formula over $\mathcal{A}_{\mathcal{G}}$ with n free variables), then the set of certain answers to q wrt $\mathcal{I}$ and $\mathcal{C}$ is

$$ans(q, \mathcal{I}, \mathcal{C}) = \{(c_1, \ldots, c_n) \in q^{\mathcal{B}} \mid \forall \mathcal{B} \in sem^{\mathcal{C}}(\mathcal{I})\}.$$

Note: query answering is logical implication.

Note: complexity will be mainly measured wrt the size of the source database $\mathcal{C}$, and will refer to the problem of deciding whether $\vec{\mathbf{c}} \in ans(q, \mathcal{I}, \mathcal{C})$, for a given $\vec{\mathbf{c}}$.

Databases with incomplete information, or Knowledge Bases

- Traditional database: one model of a first-order theory

  Query answering means evaluating a formula in the model

- Database with incomplete information, or Knowledge Base: set of models (specified, for example, as a restricted first-order theory)

  Query answering means computing the tuples that satisfy the query in all the models in the set

There is a strong connection between query answering in data integration and query answering in databases with incomplete information under constraints (or, query answering in knowledge bases).

The mapping

How is the mapping $\mathcal{M}$ between $\mathcal{S}$ and $\mathcal{G}$ specified?

- Are the sources defined in terms of the global schema?

  Approach called source-centric, or local-as-view, or LAV

- Is the global schema defined in terms of the sources?

  Approach called global-schema-centric, or global-as-view, or GAV

- A mixed approach?

  Approach called GLAV

Example of GLAV

Global schema: $Work(Person, Project), \quad Area(Project, Field)$

Source 1: $HasJob(Person, Field)$

Source 2: $Teach(Professor, Course), \; In(Course, Field)$

Source 3: $Get(Researcher, Grant), \; For(Grant, Project)$

GLAV mapping:

$$\{ (r,f) \mid HasJob(r,f) \} \quad \leadsto \quad \{ (r,f) \mid Work(r,p) \wedge Area(p,f) \}$$

$$\{ (r,f) \mid Teach(r,c) \wedge In(c,f) \} \quad \leadsto \quad \{ (r,f) \mid Work(r,p) \wedge Area(p,f) \}$$

$$\{ (r,p) \mid Get(r,g) \wedge For(g,p) \} \quad \leadsto \quad \{ (r,p) \mid Work(r,p) \}$$

Query answering in different approaches

The problem of query answering comes in different forms, depending on several parameters:

- Global schema
  - without constraints (i.e., empty theory)
  - with constraints
- Mapping
  - GAV
  - LAV
  - GLAV
- Queries
  - user queries
  - queries in the mapping

Two observations

- Unless otherwise specified, we consider conjunctive queries (or, unions thereof) as both user queries and queries in the mapping. A conjunctive query has the form

$$\{ \ (\vec{\mathbf{x}}) \mid \exists \vec{\mathbf{y}} \ \ p_1(\vec{\mathbf{x}}, \vec{\mathbf{y}}) \land \cdots \land p_m(\vec{\mathbf{x}}, \vec{\mathbf{y}}) \ \}$$

- Given a source database $\mathcal{C}$, we call retrieved global database, denoted $\mathcal{M}(\mathcal{C})$, the global database obtained by “applying” the queries in the mapping, and “transferring” to the elements of $\mathcal{G}$ the corresponding retrieved tuples.

Incompleteness and inconsistency

Query answering heavily depends upon whether incompleteness/inconsistency shows up.

Constraints in $\mathcal{G}$	Type of mapping	Incompleteness	Inconsistency
no	GAV	yes/no	no
no	GLAV	yes	no
yes	GAV	yes	yes
yes	GLAV	yes	yes

Incompleteness and inconsistency

Constraints in $\mathcal{G}$	Type of mapping	Incompleteness	Inconsistency
no	*GAV*	yes/no	no
no	GLAV	yes	no
yes	GAV	yes	yes
yes	GLAV	yes	yes

INT[noconstr, GAV]: example

Consider $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$, with

Global schema $\mathcal{G}$:

$$\text{student}(code, name, city)$$

$$\text{university}(code, name)$$

$$\text{enrolled}(Scode, Ucode)$$

Source schema $\mathcal{S}$: relations $\mathsf{s}_1(X, Y, W, Z)$, $\mathsf{s}_2(X, Y)$, $\mathsf{s}_3(X, Y)$

Mapping $\mathcal{M}$:

$$\begin{aligned}
\text{student}(X, Y, Z) &\rightsquigarrow \{ (X, Y, Z) \mid \mathsf{s}_1(X, Y, Z, W) \} \\
\text{university}(X, Y) &\rightsquigarrow \{ (X, Y) \mid \mathsf{s}_2(X, Y) \} \\
\text{enrolled}(X, W) &\rightsquigarrow \{ (X, W) \mid \mathsf{s}_3(X, W) \}
\end{aligned}$$

INT[noconstr, GAV]: example

university

code	name
AF	*bocconi*
BN	*ucla*

student

code	name	city
15	*bill*	*oslo*
12	*anne*	*florence*

enrolled

Scode	Ucode
12	*AF*
16	*BN*

$s_1^{\mathcal{C}}$

12	*anne*	*florence*	21
15	*bill*	*oslo*	24

$s_2^{\mathcal{C}}$

AF	*bocconi*
BN	*ucla*

$s_3^{\mathcal{C}}$

12	AF
16	BN

Example of source database $\mathcal{C}$ and corresponding retrieved global database $\mathcal{M}(\mathcal{C})$

INT[noconstr, GAV]: minimal model

GAV mapping assertions $g \rightsquigarrow \phi_{\mathcal{S}}$ have the logical form:

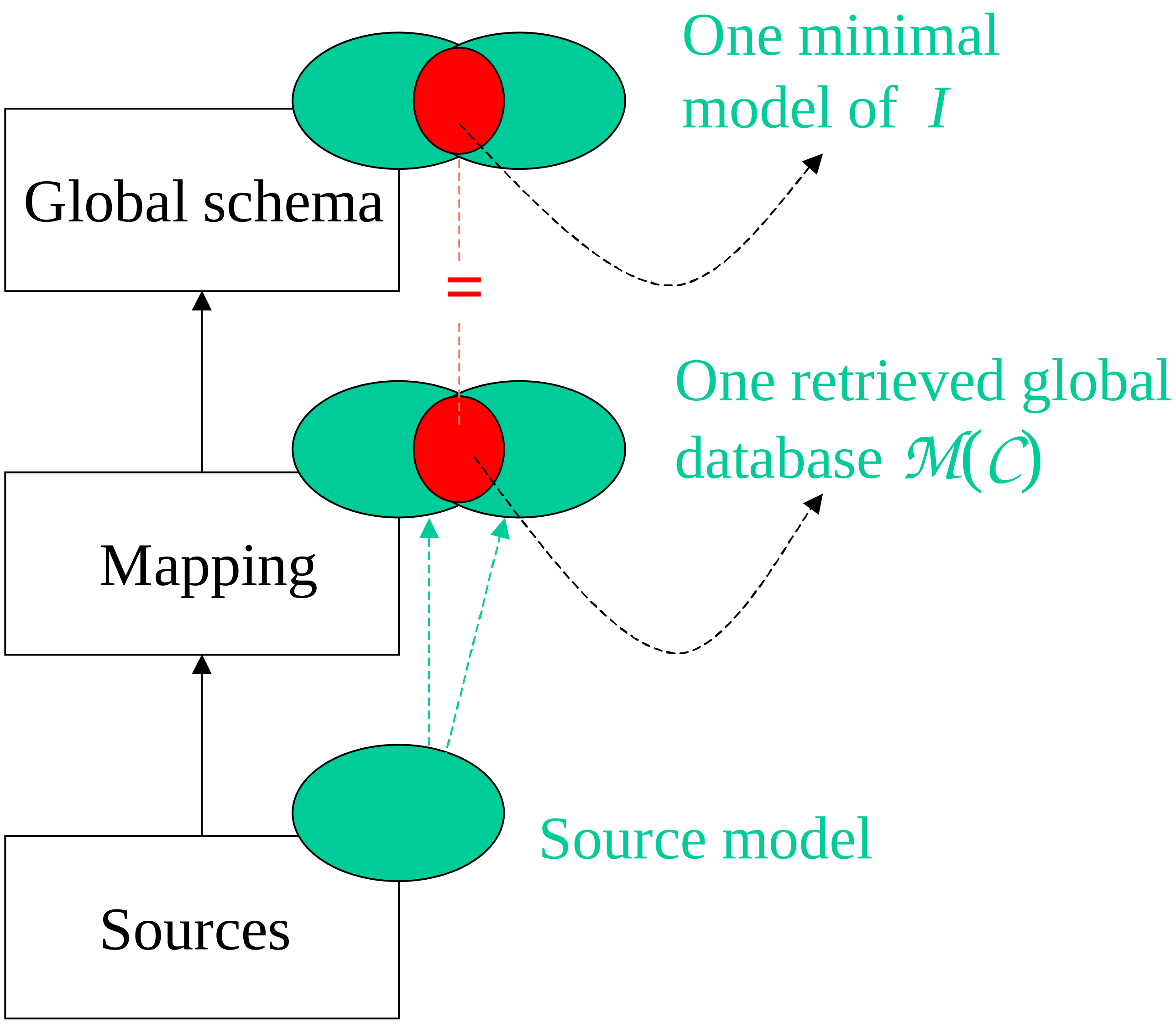
$$\forall \vec{\mathbf{x}} \;\; \phi_{\mathcal{S}}(\vec{\mathbf{x}}) \rightarrow g(\vec{\mathbf{x}})$$

where $\phi_{\mathcal{S}}$ is a conjunctive query, and g is an element of $\mathcal{G}$.

In general, given a source database $\mathcal{C}$ there are several databases that are legal wrt $\mathcal{G}$ that satisfies $\mathcal{M}$ wrt $\mathcal{C}$.

However, it is easy to see that $\mathcal{M}(\mathcal{C})$ is the intersection of all such databases, and therefore, is the only “minimal” model of $\mathcal{I}$.

INT[noconstr, GAV]



INT[noconstr, GAV]: query answering

- If q is a conjunctive query, then $\vec{\mathbf{t}} \in ans(q, \mathcal{I}, \mathcal{C})$ if and only if $\vec{\mathbf{t}} \in q^{\mathcal{M}(\mathcal{C})}$
- If q is query over $\mathcal{G}$, then the unfolding of q wrt $\mathcal{M}$, *unf*$_{\mathcal{M}}(q)$, is the query over $\mathcal{S}$ obtained from q by substituting every symbol g in q with the query $\phi_{\mathcal{S}}$ that $\mathcal{M}$ associates to g
- It can be shown that evaluating a query q over $\mathcal{M}(\mathcal{C})$ is equivalent to evaluating *unf*$_{\mathcal{M}}(q)$ over $\mathcal{C}$. It follows that, if q is a conjunctive query, then $\vec{\mathbf{t}} \in ans(q, \mathcal{I}, \mathcal{C})$ if and only if $\vec{\mathbf{t}} \in$ *unf*$_{\mathcal{M}}(q)^{\mathcal{C}}$

  Unfolding is therefore a perfect rewriting
- (Data) complexity of query answering is polynomial ($|\mathcal{M}(\mathcal{C})|$ is polynomial wrt $|\mathcal{C}|$)

INT[noconstr, GAV]: example

university

code	name
AF	*bocconi*
BN	*ucla*

student

code	name	city
15	*bill*	*oslo*
12	*anne*	*florence*

{ x | student(15,x,y) }

unfolding

$s_1^{\mathcal{C}}$

12	*anne*	*florence*	21
15	*bill*	*oslo*	24

$s_2^{\mathcal{C}}$

AF	*bocconi*
BN	*ucla*

$$\{ x \mid s_1(15, x, y, z) \}$$

INT[noconstr, GAV]: another view

Let B_1 and B_2 be two global databases with values in $\Gamma \cup$ Var.

- A homomorphism $h : B_1 \longrightarrow B_2$ is a mapping from ($\Gamma \cup$ Var(B_1)) to ($\Gamma \cup$ Var(B_2)) such that
  1. $h(c) = c$, for every $c \in \Gamma$
  2. for every fact $R_i(t)$ of B_1, we have that $R_i(h(t))$ is a fact in B_2 (where, if $t = (a_1, \ldots, a_n)$, then $h(t) = (h(a_1), \ldots, h(a_n))$)
- B_1 is homomorphically equivalent to B_2 if there is a homomorphism $h : B_1 \longrightarrow B_2$ and a homomorphism $h' : B_2 \longrightarrow B_1$

Let $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$ be a data integration system. If $\mathcal{C}$ is a source database, then a universal solution for $\mathcal{I}$ relative to $\mathcal{C}$ is a model J of $\mathcal{I}$ relative to $\mathcal{C}$ such that for every model J' of $\mathcal{I}$ relative to $\mathcal{C}$, there exists a homomorphism $h : J \longrightarrow J'$ (see [Fagin&al. ICDT'03]).

INT[noconstr, GAV]: another view

- Homomorphism preserves satisfaction of conjunctive queries: if there exists a homomorphism $h : J \rightarrow J'$, and q is a conjunctive query, then $\vec{\mathbf{t}} \in q^J$ implies $\vec{\mathbf{t}} \in q^{J'}$
- Let $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$ be a GAV data integration system without constraints in the global schema. If $\mathcal{C}$ is a source database, then $\mathcal{M}(\mathcal{C})$ is the minimal universal solution for $\mathcal{I}$ relative to $\mathcal{C}$
- We derive again the following results
  - if q is a conjunctive query, then $\vec{\mathbf{t}} \in ans(q, \mathcal{I}, \mathcal{C})$ if and only if $\vec{\mathbf{t}} \in q^{\mathcal{M}(\mathcal{C})}$
  - complexity of query answering is polynomial

INT[noconstr, LAV]: basic technique

Consider conjunctive queries and conjunctive views.

$$\mathsf{r}_1(T) \quad \rightsquigarrow \quad \{ \ (T) \mid \mathsf{movie}(T,Y,D) \land \mathsf{european}(D) \ \}$$

$$\mathsf{r}_2(T,V) \quad \rightsquigarrow \quad \{ \ (T,V) \mid \mathsf{movie}(T,Y,D) \land \mathsf{review}(T,V) \ \}$$

$$\begin{array}{rcl} \forall T \ \mathsf{r}_1(T) & \rightarrow & \exists Y \exists D \ \mathsf{movie}(T,Y,D) \land \mathsf{european}(D) \\ \forall T \ \forall V \ \mathsf{r}_2(T,V) & \rightarrow & \exists Y \exists D \ \mathsf{movie}(T,Y,D) \land \mathsf{review}(T,V) \end{array}$$

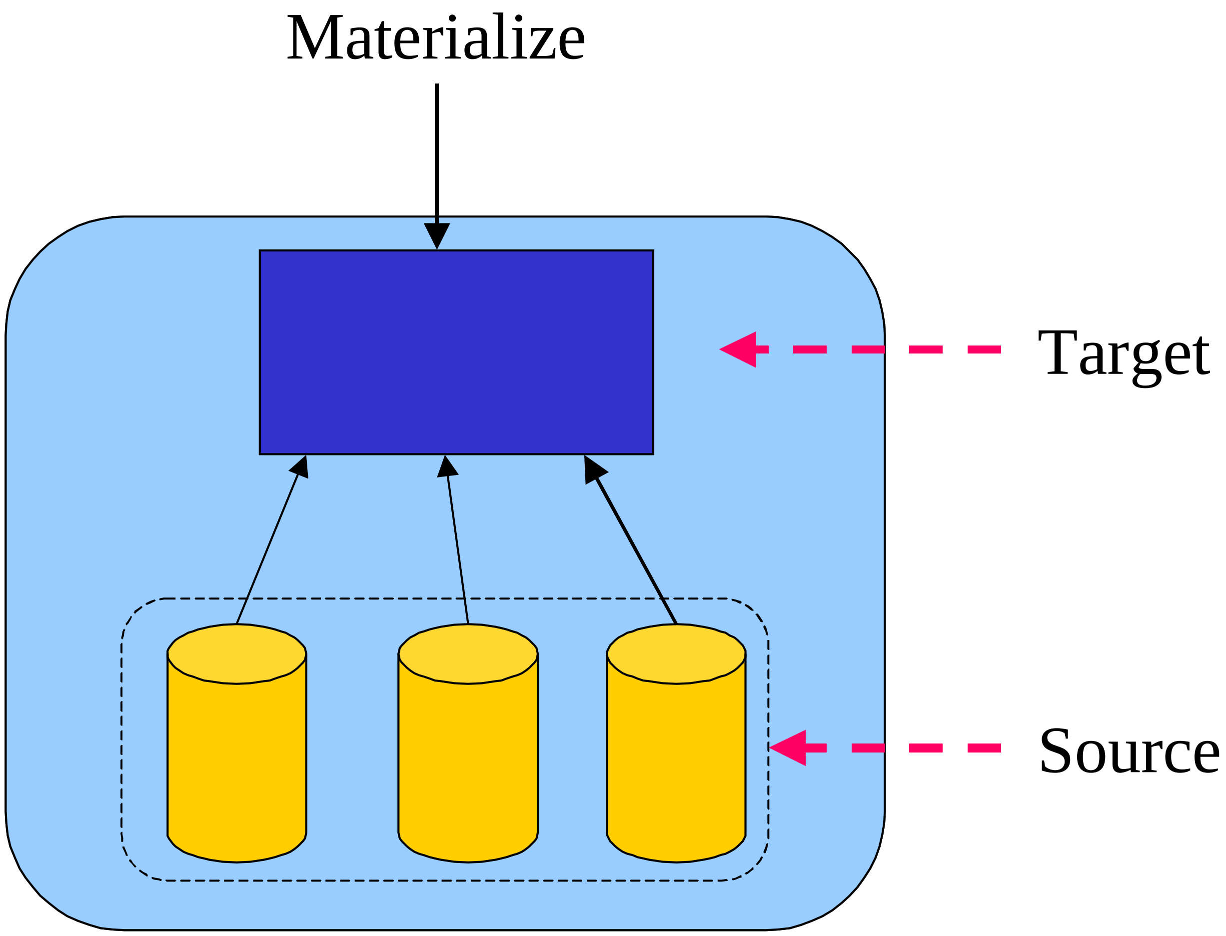
$$\begin{array}{rcl} \mathsf{movie}(T, f_1(T), f_2(T)) & \leftarrow & \mathsf{r}_1(T) \\ \mathsf{european}(f_2(T)) & \leftarrow & \mathsf{r}_1(T) \\ \mathsf{movie}(T, f_4(T,V), f_5(T,V)) & \leftarrow & \mathsf{r}_2(T,V) \\ \mathsf{review}(T,V)) & \leftarrow & \mathsf{r}_2(T,V) \end{array}$$

Answering a query means evaluating a goal wrt to this nonrecursive logic program (PTIME data complexity), i.e., this logic program is a perfect rewriting.

Outline

- Peer-based Distributed Information Systems
- Mediator-based data integration
- Data exchange
- P2P data integration
- Conclusions

Data exchange



Formal framework for data exchange

From [Fagin&al. ICDT'03], a data exchange setting $\mathcal{E} = (S, T, \Sigma_{st}, \Sigma_t)$ consists of

- a source schema S
- a target schema T
- a set Σ_{st} of source-to-target dependencies, each one of the form (tuple generating dependency, tgd)

$$\forall \vec{\mathbf{x}} \ (\phi_S(\vec{\mathbf{x}}) \rightarrow \exists \vec{\mathbf{y}} \phi_T(\vec{\mathbf{x}}, \vec{\mathbf{y}}))$$

  with $\phi_S(\vec{\mathbf{x}})$ conjunction of atoms over S, and $\phi_T(\vec{\mathbf{x}}, \vec{\mathbf{y}})$ conjunction of atoms over T (cfr. GLAV mappings in data integration)
- a set Σ_t of target dependencies, each one of the form (tgd, or equality generating dependency)

$$\forall \vec{\mathbf{x}} \ (\phi_S(\vec{\mathbf{x}}) \rightarrow \exists \vec{\mathbf{y}} \phi_T(\vec{\mathbf{x}}, \vec{\mathbf{y}})) \quad \text{or} \quad \forall \vec{\mathbf{x}} \ (\phi_T(\vec{\mathbf{x}}) \rightarrow (x_1 = x_2))$$

Formal framework for data exchange

The data exchange problem associated with the data exchange setting $\mathcal{E} = (S, T, \Sigma_{st}, \Sigma_t)$ is the following:

- given a finite instance $\mathcal{C}$ of S (source instance)
- find a finite instance J of T (target instance) such that (I, J) satisfies Σ_{st}, and J satisfies Σ_t.

Such a J is called a solution for $\mathcal{E}$ wrt $\mathcal{C}$, or simply for $\mathcal{C}$. The set of all solutions is denoted by Sol($\mathcal{C}$).

Example of data exchange

$$\Sigma_{st}: \\ \{ \forall a \forall b \forall c \, (P(a,b,c) \rightarrow \exists Y \exists Z \;\; T(a,Y,Z)) \\ \forall a \forall b \forall c \, (Q(a,b,c) \rightarrow \exists X \exists U \;\; T(X,b,U)) \\ \forall a \forall b \forall c \, (R(a,b,c) \rightarrow \exists V \exists W \;\; T(V,W,c)) \, \}$$

$$\mathcal{C} = \{P(a_0, b_1, c_1),\; Q(a_2, b_0, c_2),\; R(a_3, b_3, c_0)\}$$

Some possible solutions:

$$J = \{T(a_0, Y_0, Z_0),\; T(X_0, b_0, U_0),\; T(V_0, W_0, c_0)\}$$

$$J_1 = \{T(a_0, b_0, c_0)\}$$

$$J_2 = \{T(a_0, b_0, Z_1),\; T(V_1, W_1, c_0)\}$$

Data exchange: results

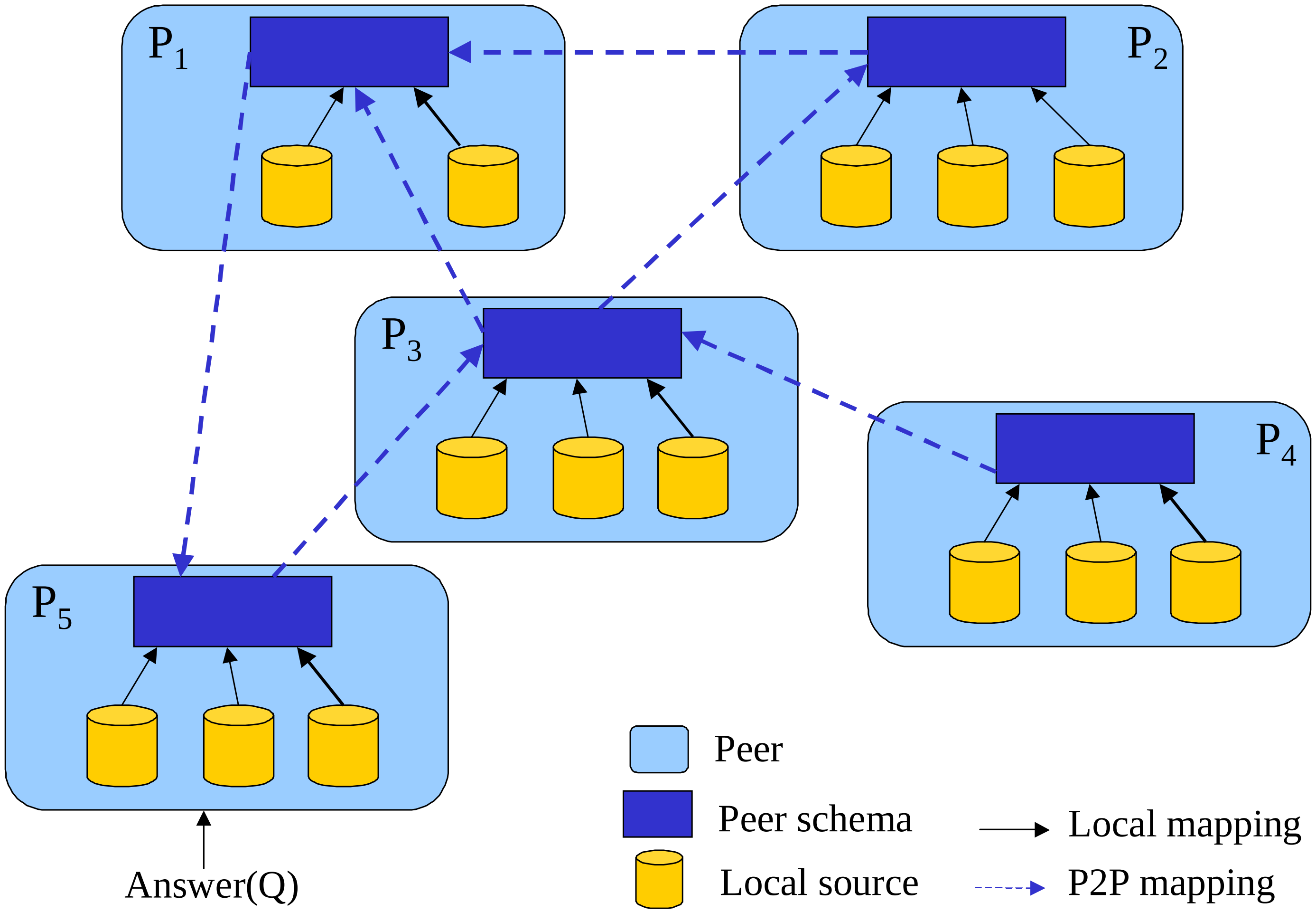
The following results appear in [Fagin&al. ICDT'03]:

- If $\mathcal{C}$ is a source instance and J, J' are universal solutions for $\mathcal{C}$, then J and J' are homomorphically equivalent
- Let $\mathcal{C}, \mathcal{C}'$ be two source instances, J a universal solution for $\mathcal{C}$, and J' a universal solution for $\mathcal{C}'$. Then Sol($\mathcal{C}$)=Sol($\mathcal{C}'$) if and only if J and J' are homomorphically equivalent
- If the tgds in Σ_t are weakly acyclic (i.e., cycles do not involve existentially quantified variables), then the existence of a solution for $\mathcal{C}$ can be checked in polynomial time wrt the size of $\mathcal{C}$. Moreover, if a solution for $\mathcal{C}$ exists, then a universal solution for $\mathcal{C}$ can be produced in polynomial time wrt the size of $\mathcal{C}$ (by "chasing" $\mathcal{C}$)

Outline

- Peer-based Distributed Information Systems
- Mediator-based data integration
- Data exchange
- P2P data integration
- Conclusions

P2P data integration



P2P data integration: general framework

A P2P system Π is a set $\{P_1, \ldots, P_n\}$ of peers, where each peer P_i models an autonomous information site, that

- exports its information content in terms of a schema
- stores actual data in a set of (local) sources
- is related to other peers in Π by means of a set of P2P mappings, where each P2P mapping is a schema level assertion relating information in another peer P_j to information in P_i

Inspired by [Catarci&Lenzerini COOPIS '92], Halevy&al. ICDE'03]. Other related work: [Ghidini&Serafini FCS '98], [Bernstein&al. WebDB '02, Franconi&al.'03].

Logic-based formal framework for P2P data integration

Each peer of Π is a tuple $P = (G, S, L, M)$ constituted by

- a schema G, i.e., a set of FOL formulas over a peer alphabet A_G
- a set S of (local) sources (simply a finite relational alphabet)
- a set L of local mapping assertions between G and S, each of the form

$$cq_S \leadsto cq_G$$

  where cq_S and cq_G are conjunctive queries over S and G, respectively
- a set M of P2P mapping assertions, each of the form

$$cq' \leadsto cq$$

where:
- cq' is a conjunctive query over one of the other peers in Π
- cq is a conjunctive query over the peer schema of P

(cq' and cq are of the same arity)

Formal framework for P2P data integration: semantics

- We assume that all peers are interpreted over a fixed infinite domain
- We also refer to a fixed, infinite, denumerable, set Γ of constants, that act as standard names (i.e., Γ is isomorphic to the interpretation domain)
- ... but see the conclusions for a more general approach

In the following, we will use constants of Γ to denote elements of the interpretation domain

Semantics of one peer

For each peer $P = (G, S, L, M)$ we define a FOL theory T_P as follows:

- The alphabet of T_P is obtained as union of the alphabets of the schema G and of the sources S
- The axioms of T_P are as follows:
  - all FOL formulas in the schema G
  - for each local mapping assertion $\{\vec{\mathbf{x}} \mid \exists\vec{\mathbf{y}}\,\varphi_S(\vec{\mathbf{x}},\vec{\mathbf{y}})\} \leadsto \{\vec{\mathbf{x}} \mid \exists\vec{\mathbf{z}}\,\varphi_G(\vec{\mathbf{x}},\vec{\mathbf{z}})\}$ in L, one formula of the form

$$\forall\vec{\mathbf{x}}\,(\exists\vec{\mathbf{y}}\,\varphi_S(\vec{\mathbf{x}},\vec{\mathbf{y}}) \supset \exists\vec{\mathbf{z}}\,\varphi_G(\vec{\mathbf{x}},\vec{\mathbf{z}}))$$

Notice that T_P does not consider the P2P mappings in M

It follows that we are modeling each peer P as a GLAV data integration system, in turn modeled as a FOL theory T_P (ignoring the P2P mappings M)

Semantics of one peer (cont'd)

We resort to the standard notion of certain answer in data integration:

- We start from a source database D for the peer P, i.e., a database over Γ for the source relation symbols in S
- A model of P based on D is an interpretation of T_P that
  - coincides with D on the source relation symbols
  - satisfies all formulas in T_P
- Given a query Q of arity n expressed over (the schema G of) P, and a source database D, the (local) certain answers of P to Q based on D are

$$ans(Q, P, D) \;=\; \{\, \vec{\mathbf{t}} \in \Gamma^n \mid \vec{\mathbf{t}} \in Q^{\mathcal{I}}, \text{ for every model } \mathcal{I} \text{ of } P \text{ based on } D\}$$

Semantics of a P2P system

- A source database $\mathcal{D}$ for Π is the disjoint union of one source database for each peer P_i in Π
- Given a source database $\mathcal{D}$ for Π, the set of models of Π relative to $\mathcal{D}$ is:

$$sem^{\mathcal{D}}(\Pi) = \{\, \mathcal{I} \mid \begin{array}{l} \mathcal{I} \text{ is a model of all peer theories } T_{P_i} \text{ based on } \mathcal{D}, \text{ and} \\ \mathcal{I} \text{ satisfies all P2P mapping assertions} \end{array} \}$$

  The meaning of $\mathcal{I}$ satisfying a P2P mapping assertion may vary in the various approaches

- Given a query Q of arity n posed to a peer P_i of Π, and a source database $\mathcal{D}$, the certain answers to Q based on $\mathcal{D}$ are

$$ans(Q, \Pi, \mathcal{D}) = \{\, \vec{\mathbf{t}} \in \Gamma^n \mid \vec{\mathbf{t}} \in Q^{\mathcal{I}}, \text{ for every } \mathcal{I} \in sem^{\mathcal{D}}(\Pi) \,\}$$

Possible formalizations of P2P mappings

We consider two alternatives for specifying the semantics of P2P mappings:

- Based on First-Order Logic

  P2P mappings are considered as material logical implications

- Based on Epistemic Logic

  P2P mappings are considered as specifications of exchange of certain answers

First-Order Logic semantics of P2P mappings

The semantics of P2P mapping assertions is given in terms of First-Order Logic [Halevy&al. ICDE'03, Bernstein&al. WebDB '02]

An interpretation $\mathcal{I}$ satisfies a P2P mapping assertion

$$\{\vec{\mathbf{x}} \mid \exists \vec{\mathbf{y}}\, \varphi_1(\vec{\mathbf{x}}, \vec{\mathbf{y}})\} \leadsto \{\vec{\mathbf{x}} \mid \exists \vec{\mathbf{z}}\, \varphi_2(\vec{\mathbf{x}}, \vec{\mathbf{z}})\}$$

if it satisfies the FOL formula

$$\forall \vec{\mathbf{x}}\, (\exists \vec{\mathbf{y}}\, \varphi_1(\vec{\mathbf{x}}, \vec{\mathbf{y}}) \supset \exists \vec{\mathbf{z}}\, \varphi_2(\vec{\mathbf{x}}, \vec{\mathbf{z}}))$$

which is equivalent to the condition

$$\{\vec{\mathbf{x}} \mid \exists \vec{\mathbf{y}}\, \varphi_1(\vec{\mathbf{x}}, \vec{\mathbf{y}})\}^{\mathcal{I}} \subseteq \{\vec{\mathbf{x}} \mid \exists \vec{\mathbf{z}}\, \varphi_2(\vec{\mathbf{x}}, \vec{\mathbf{z}})\}^{\mathcal{I}}$$

Inadequacy of FOL semantics of P2P mappings

The FOL semantics is not adequate for P2P data integration:

- Lack of modularity
  - the system is modeled by a flat FOL theory, with no formal separation between the various peers
  - the modular structure of the system is not reflected in the semantics
- Bad computational properties

  Computing the set of certain answers to a conjunctive query Q posed to a peer is undecidable, even when all peer schemas are empty [Halevy&al. ICDE'03, Koch FOIKS'02]
- Lack of generality

  To recover decidability, one has to limit the expressive power of P2P mappings (e.g., assume acyclicity) [Halevy&al. ICDE'03]

Epistemic semantics for P2P mappings: objectives

A new semantics for P2P mappings, with the following aims:

- Peers in our context are to be considered autonomous sites that exchange information
- We do not want to limit a-priori the topology of the mapping assertions among the peers in the system
- Defining a setting where query answering is decidable, and possibly, polynomially tractable

Epistemic semantics for P2P mappings: basic idea

The new semantics is based on epistemic logic [Reiter TARK'88]

- A P2P mapping $cq_i \rightsquigarrow cq_j$ (with cq_i over P_i and cq_j over P_j) is interpreted as an epistemic formula which imposes that only the certain answers to cq_i in P_i (i.e., the facts that are known by P_i) are transferred to P_j as facts satisfying cq_j.
  In other words, peer P_i communicates to peer P_j only facts that are certain, i.e., true in every model of the P2P system
- The modular structure of the system is now reflected in the semantics (by virtue of the modal semantics of epistemic logics)
- Good computational properties: computing the certain answers to a conjunctive query Q based on a source database $\mathcal{D}$ is polynomial time in the size of $\mathcal{D}$, even for cyclic mappings

Epistemic logic: basic notions

In epistemic logic, we have a new form of atoms, namely (φ is again a formula):

$$\mathbf{K}\,\varphi$$

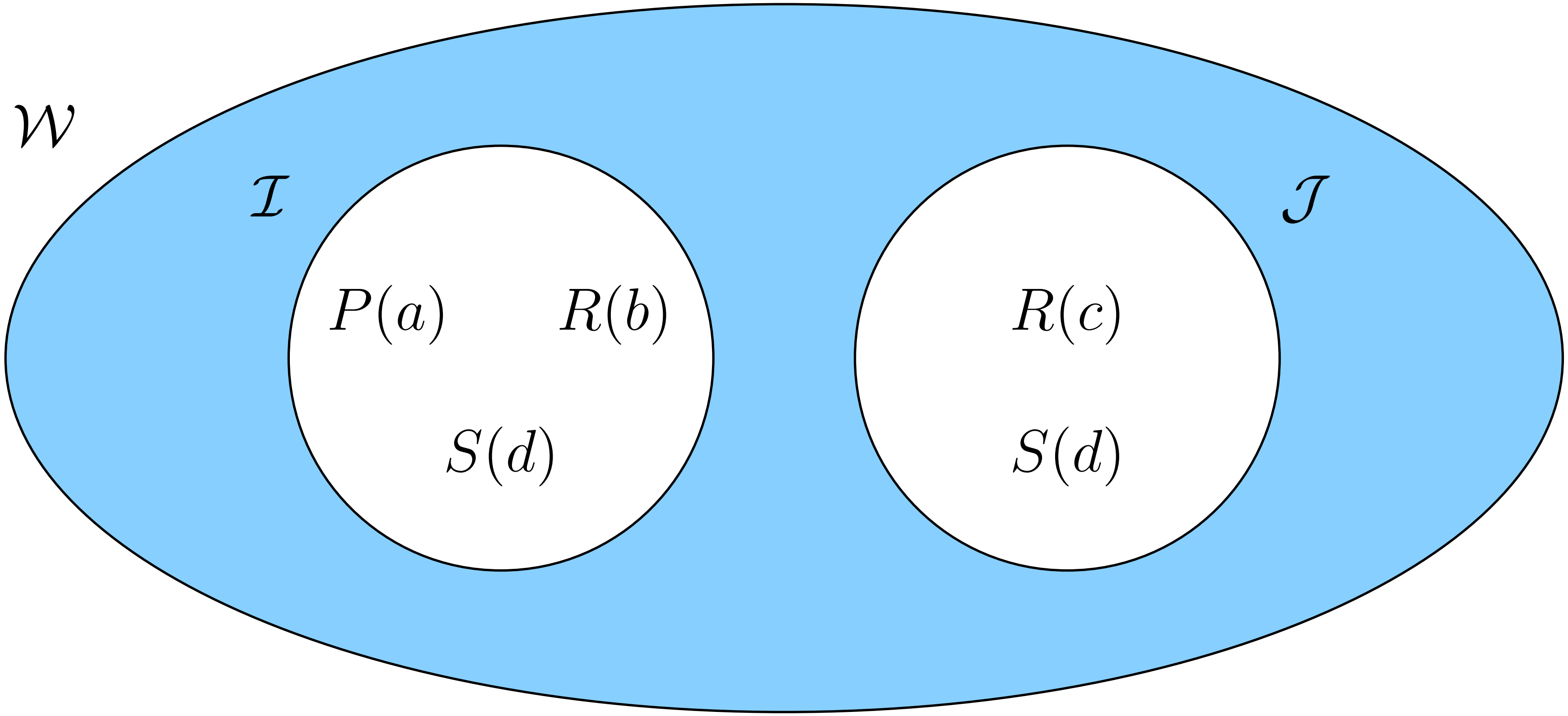
An epistemic interpretation is a pair $\langle \mathcal{I}, \mathcal{W} \rangle$, where $\mathcal{W}$ is a set of FOL interpretations and $\mathcal{I} \in \mathcal{W}$

- a FOL formula constituted by an atom $a(\vec{\mathbf{x}})$ is satisfied in $\langle \mathcal{I}, \mathcal{W} \rangle$ by the tuples $\vec{\mathbf{t}}$ of constants such that $a(\vec{\mathbf{t}})$ is true in $\mathcal{I}$
- an atom of the form $\mathbf{K}\varphi(\vec{\mathbf{x}})$ is satisfied in $\langle \mathcal{I}, \mathcal{W} \rangle$ by the tuples $\vec{\mathbf{t}}$ of constants such that $\varphi(\vec{\mathbf{t}})$ is satisfied in all epistemic interpretations $\langle \mathcal{J}, \mathcal{W} \rangle$ with $\mathcal{J} \in \mathcal{W}$

An epistemic model of an epistemic logic theory $\{\varphi_1, \ldots, \varphi_t\}$ is an epistemic interpretation $\langle \mathcal{I}, \mathcal{W} \rangle$ that satisfies every axiom φ_i

An axiom φ_i is satisfied in $\langle \mathcal{I}, \mathcal{W} \rangle$ if φ_i is satisfied in all $\langle \mathcal{J}, \mathcal{W} \rangle$ with $\mathcal{J} \in \mathcal{W}$

Epistemic logic: example 1

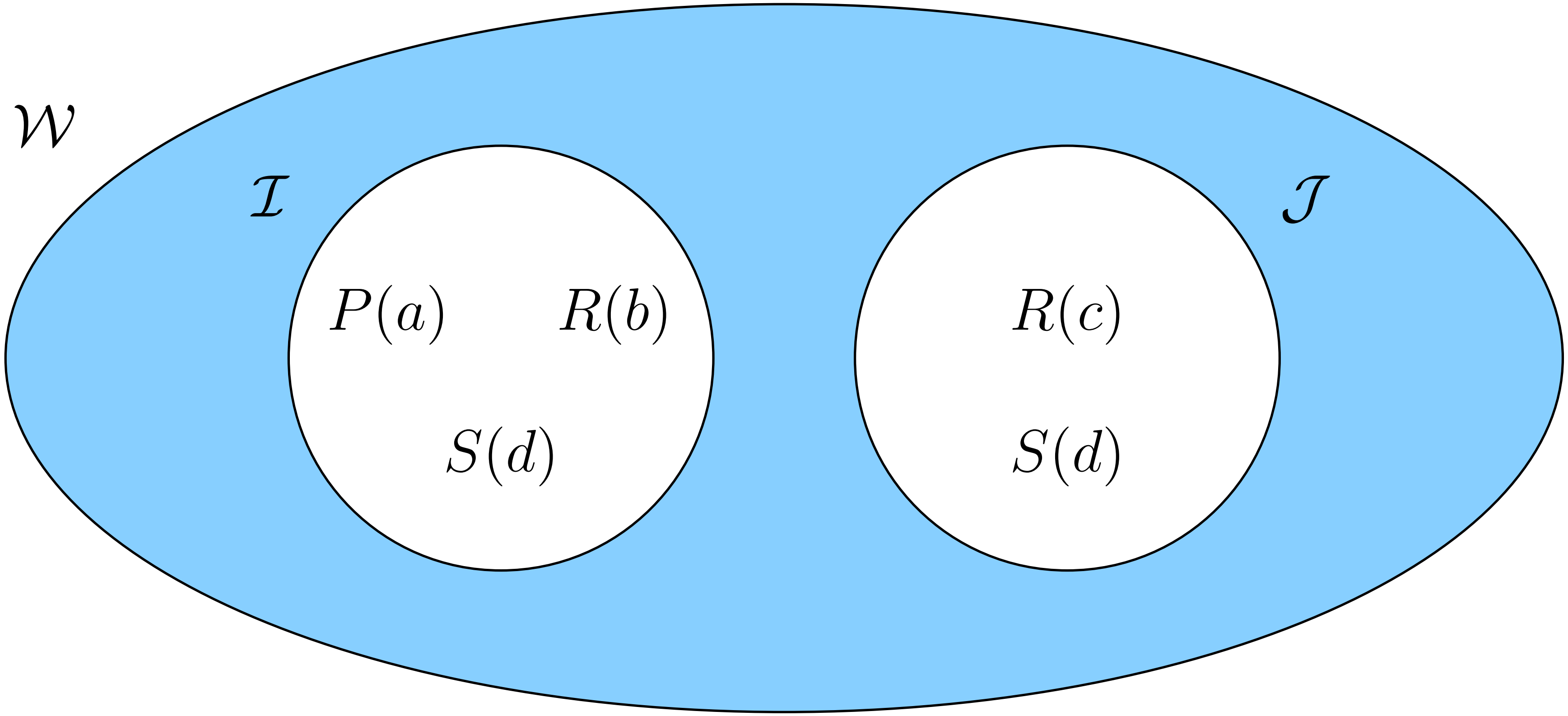


$$\langle \mathcal{I}, \mathcal{W} \rangle \models P(a)$$

$$\langle \mathcal{J}, \mathcal{W} \rangle \not\models P(a)$$

$$\langle \mathcal{I}, \mathcal{W} \rangle \not\models \mathbf{K}\, P(a)$$

Epistemic logic: example 2

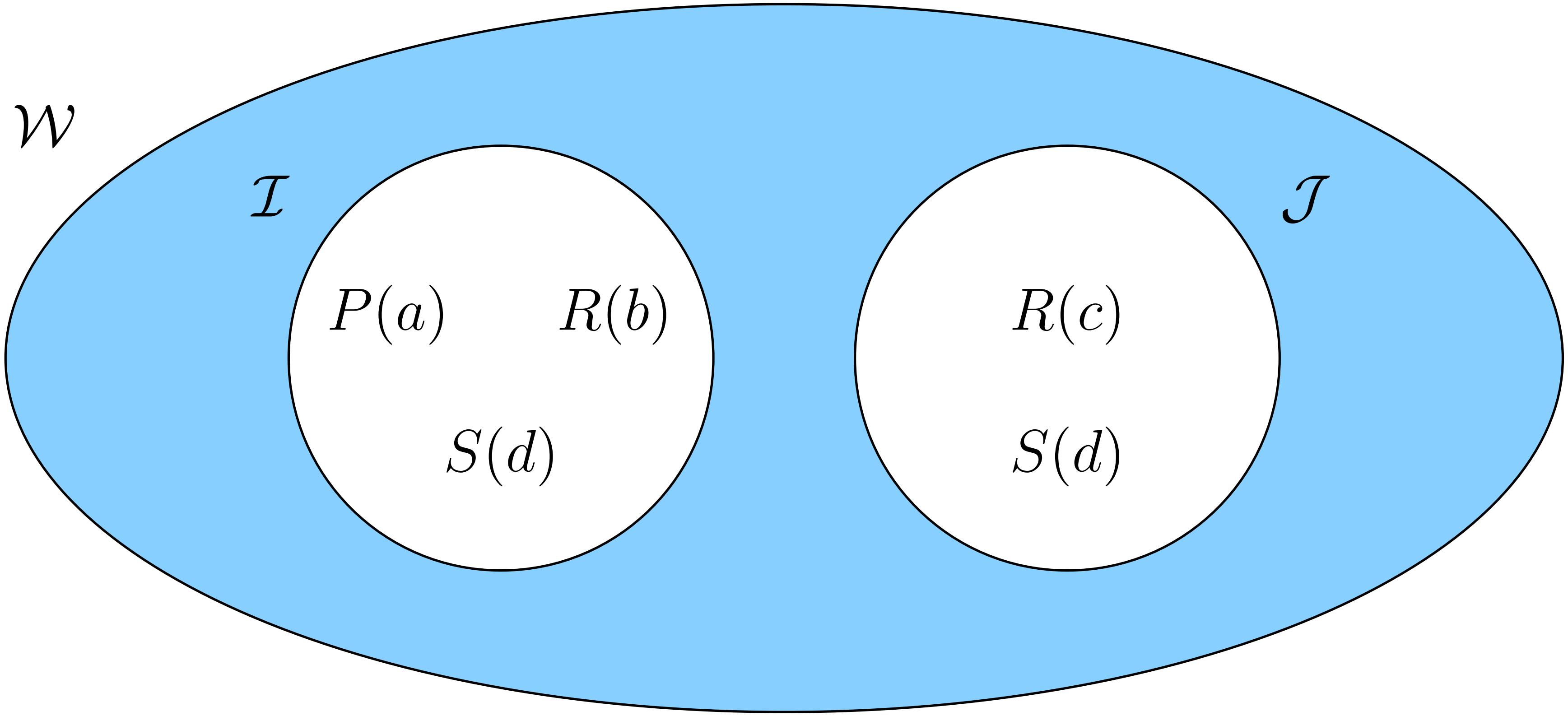


$$\langle \mathcal{I}, \mathcal{W} \rangle \models \mathbf{K}\,(R(b) \vee R(c))$$

$$\langle \mathcal{I}, \mathcal{W} \rangle \not\models (\mathbf{K}\,R(b)) \vee (\mathbf{K}\,R(c))$$

$$\langle \mathcal{I}, \mathcal{W} \rangle \models \mathbf{K}\,S(d)$$

Epistemic logic: example 3



$$\langle \mathcal{I}, \mathcal{W} \rangle \models \mathbf{K}\,(\exists x\, R(x))$$

$$\langle \mathcal{I}, \mathcal{W} \rangle \not\models \exists x\,(\mathbf{K}\, R(x))$$

$$\langle \mathcal{I}, \mathcal{W} \rangle \models \exists x\,(\mathbf{K}\, S(x))$$

Epistemic semantics for P2P mappings: basic idea

We formalize a P2P system Π in terms of the epistemic logic theory E_Π:

- the alphabet $\mathcal{A}_\Pi$ is the disjoint union of the alphabets of the various peer theories T_P, one for each peer P in Π
- all the formulas of the various theories T_P are axioms in E_Π
- for each P2P mapping assertion

$$\{\vec{\mathbf{x}} \mid \exists \vec{\mathbf{y}}\, \varphi_1(\vec{\mathbf{x}}, \vec{\mathbf{y}})\} \leadsto \{\vec{\mathbf{x}} \mid \exists \vec{\mathbf{z}}\, \varphi_2(\vec{\mathbf{x}}, \vec{\mathbf{z}})\}$$

  in the peers of Π, there is one axiom in E_Π of the form

$$\forall \vec{\mathbf{x}}\, ((\mathbf{K}\, \exists \vec{\mathbf{y}}\, \varphi_1(\vec{\mathbf{x}}, \vec{\mathbf{y}})) \supset \exists \vec{\mathbf{z}}\, \varphi_2(\vec{\mathbf{x}}, \vec{\mathbf{z}}))$$

Epistemic semantics for P2P mappings: basic idea

In other words, $\langle \mathcal{I}, \mathcal{W} \rangle$ satisfies the P2P mapping assertion $cq_1 \leadsto cq_2$ if,

for every tuple $\vec{\mathbf{t}}$ of constants in Γ,

when $\vec{\mathbf{t}} \in cq_1^{\mathcal{J}}$ for every FOL model $\mathcal{J}$ in $\mathcal{W}$, then $\vec{\mathbf{t}} \in cq_2^{\mathcal{I}}$

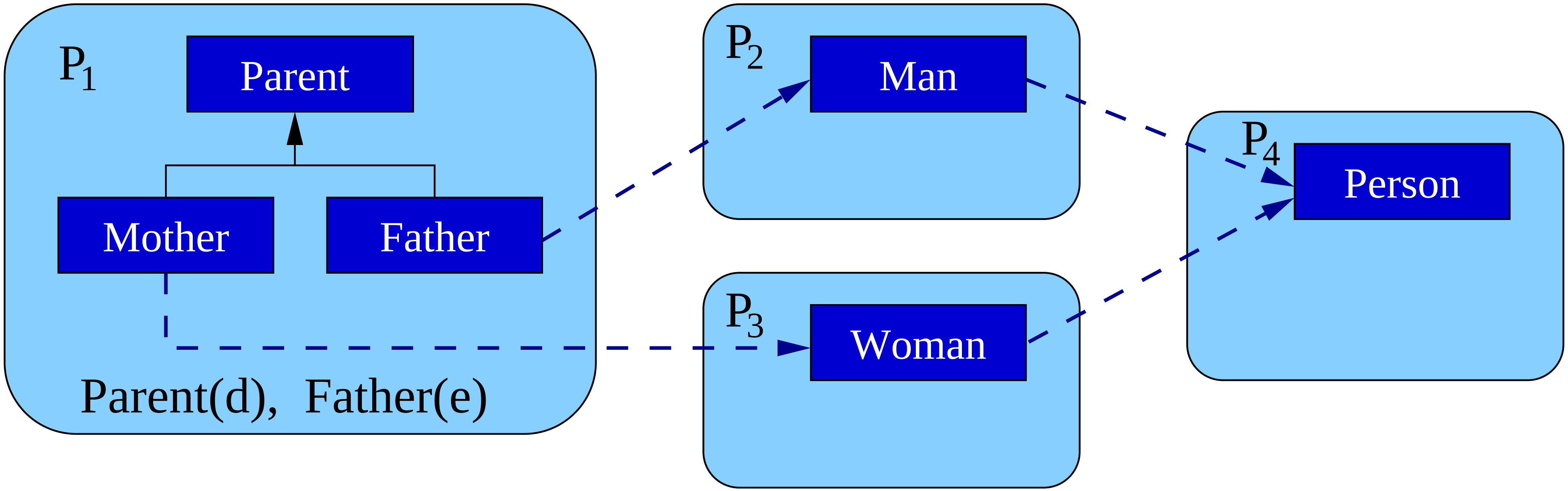
An epistemic model of Π based on $\mathcal{D}$ is an epistemic interpretation $\langle \mathcal{I}, \mathcal{W} \rangle$ such that

- $\mathcal{W}$ is a set of models of T_Π based on $\mathcal{D}$, and
- $\langle \mathcal{I}, \mathcal{W} \rangle$ satisfies all axioms corresponding to the P2P mapping assertions in the peers of Π

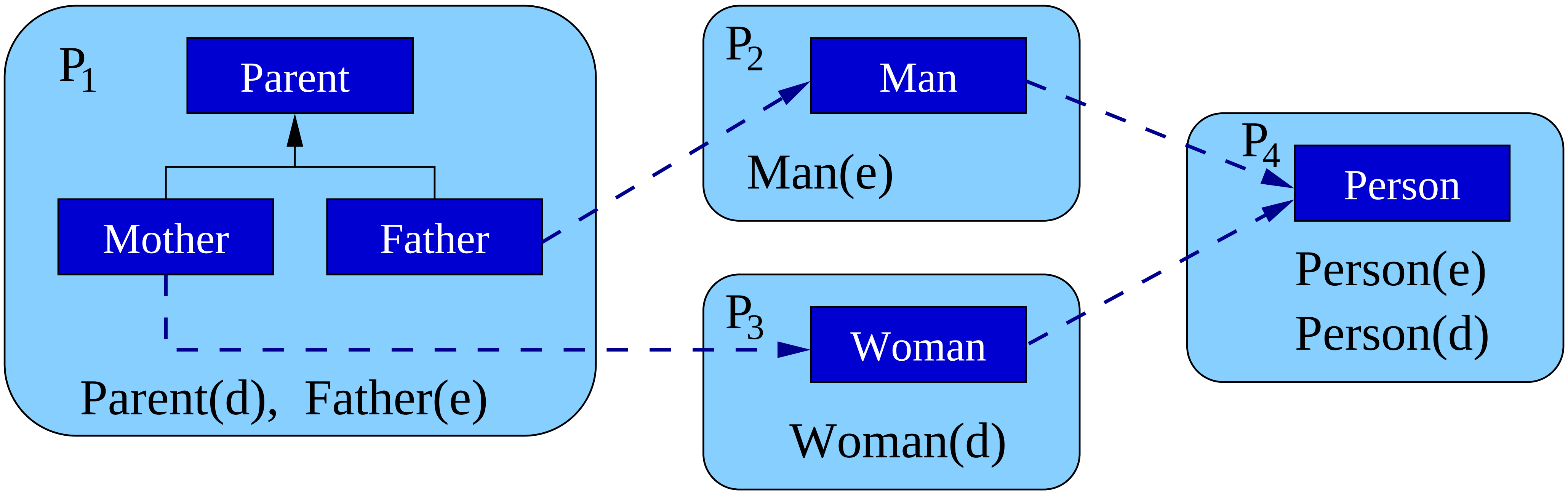
Given a query Q of arity n posed to a peer P_i of Π, and a source database $\mathcal{D}$, the certain answers to Q based on $\mathcal{D}$ under epistemic semantics are

$$ans_{\mathbf{k}}(Q, \Pi, \mathcal{D}) \;=\; \{\, \vec{\mathbf{t}} \in \Gamma^n \mid \vec{\mathbf{t}} \in Q^{\mathcal{I}}, \text{ for every epistemic model } \langle \mathcal{I}, \mathcal{W} \rangle \text{ of } \Pi \text{ based on } \mathcal{D} \,\}$$

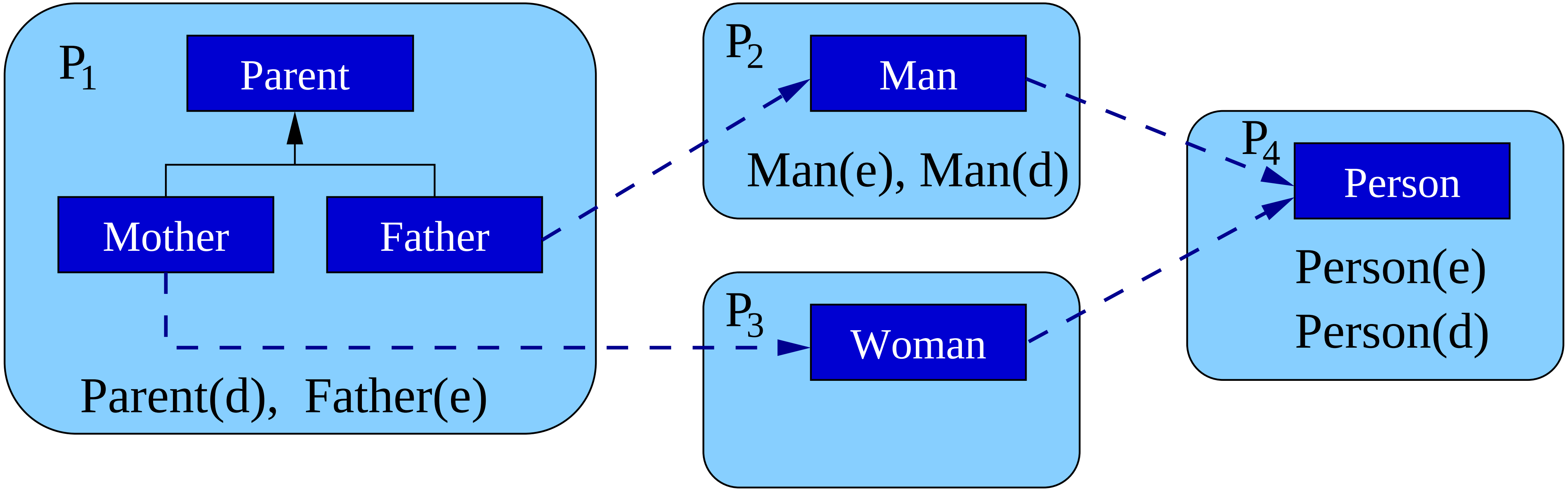
Semantics of P2P mappings: example



FOL semantics of P2P mappings: model 1

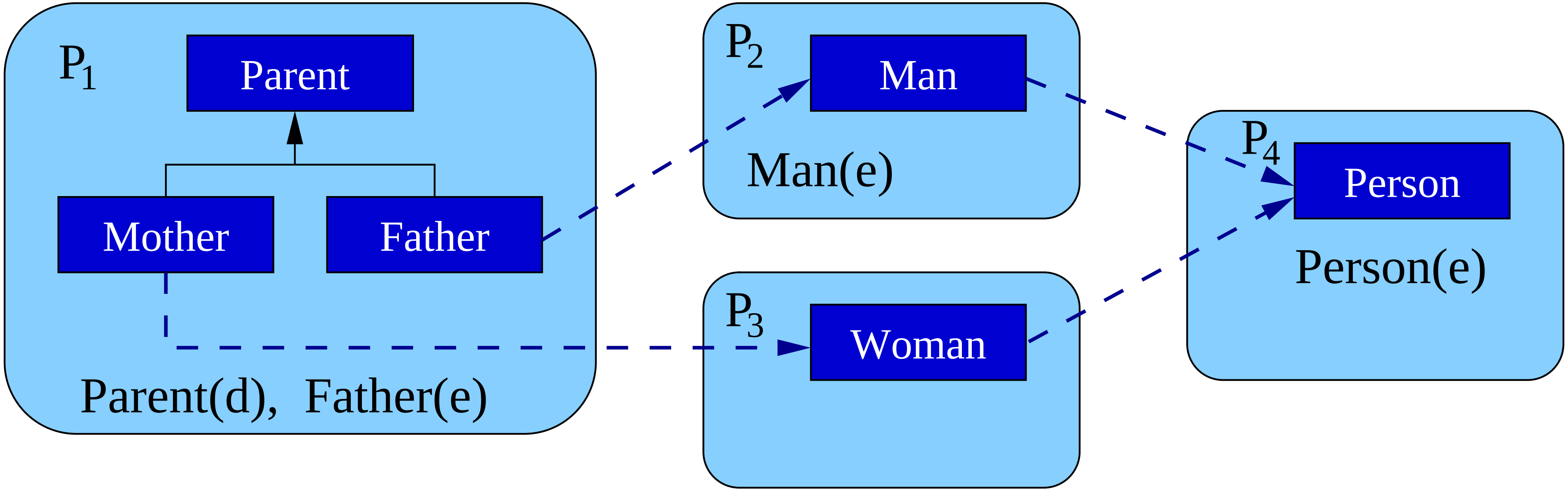


FOL semantics of P2P mappings: model 2



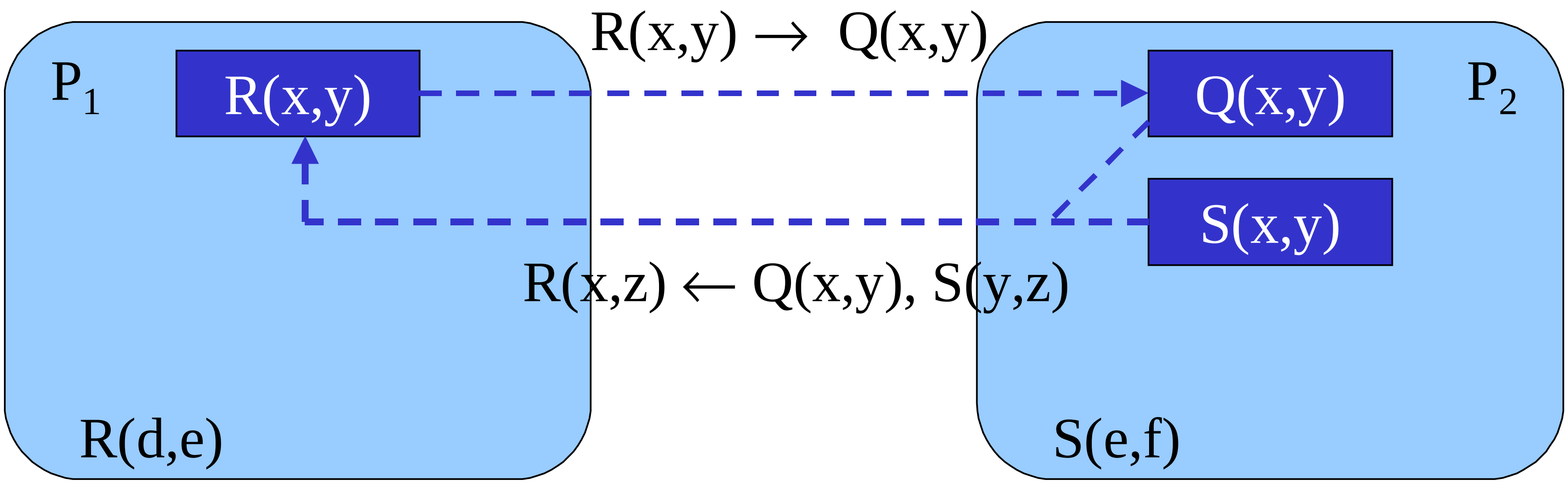
According to the FOL semantics, $\mathrm{Person}(d)$ is true in all cases, and therefore is a certain answer to $\{x \mid \mathrm{Person}(x)\}$

Epistemic semantics of P2P mappings

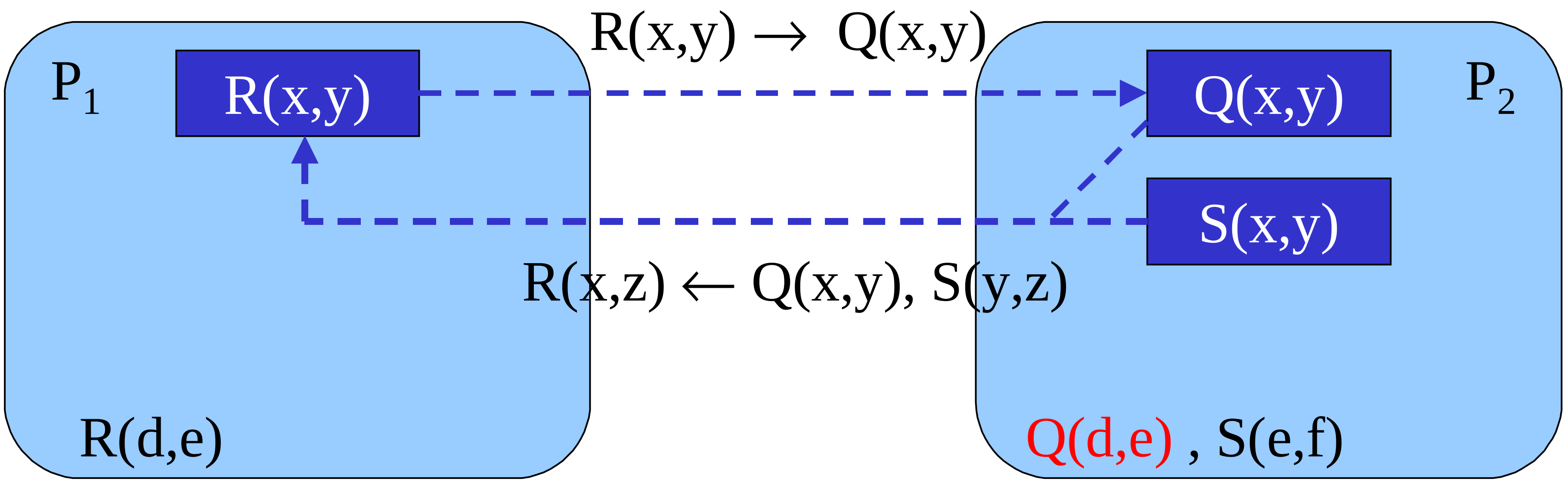


According to the epistemic semantics, $\mathrm{Person}(d)$ is not a certain answer to $\{x \mid \mathrm{Person}(x)\}$

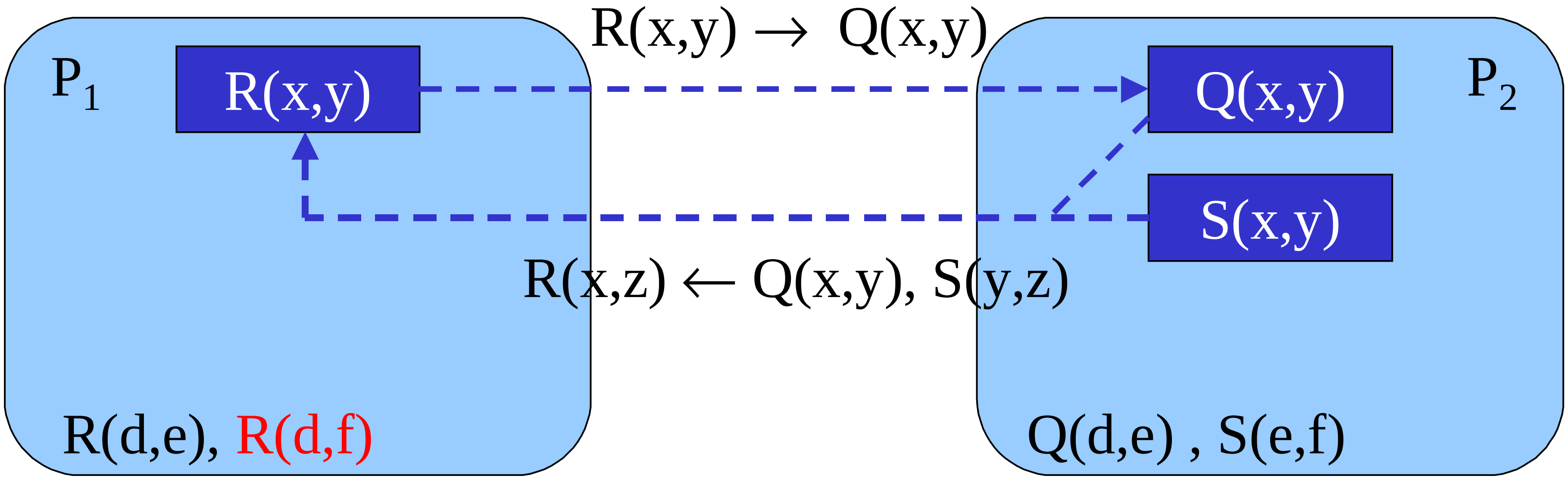
Query answering: example



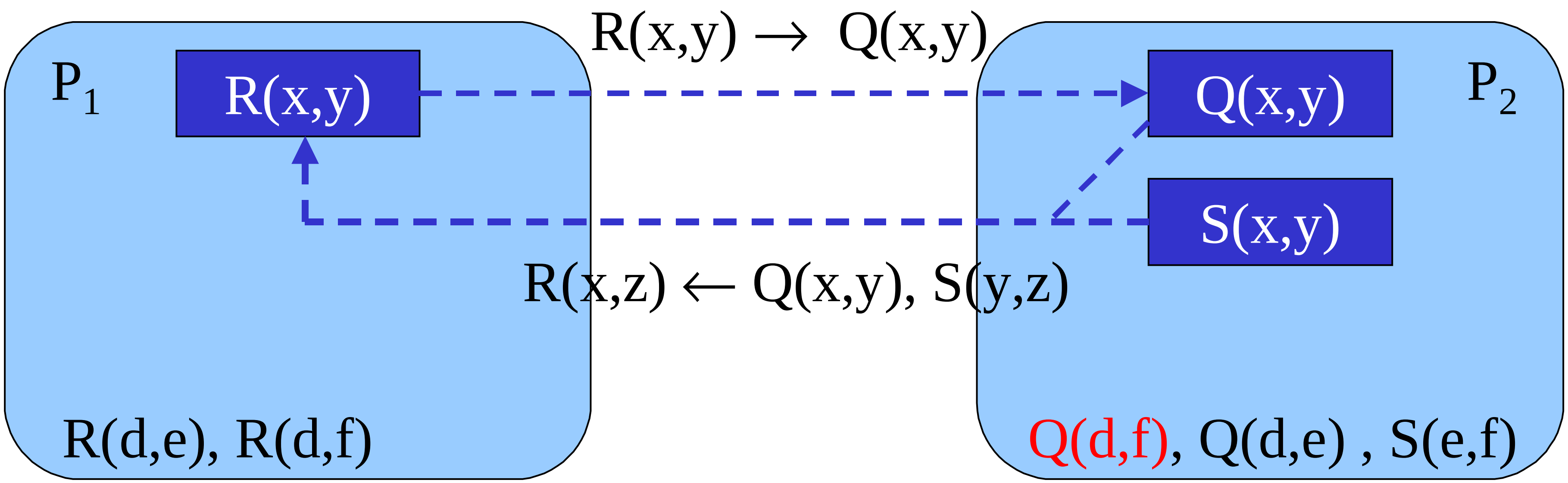
Query answering: example



Query answering: example



Query answering: example



Query answering in P2P systems under epistemic semantics

- We are interested in an algorithm for distributed query answering
  - the query is posed to one peer in the system
  - each peer executes the same algorithm, and in doing so exchanges information only with the peers it is connected to
  - no central coordination or centralized data structures

- We assume that peers accept queries in a query language $\mathcal{L}$ (subsuming at least conjunctive queries)

- We require that each peer, given a query Q in $\mathcal{L}$, is able to compute a Datalog query Q' that is a perfect reformulation of Q

Perfect reformulation

- P2P mappings in a peer P are of the form $cq' \leadsto cq$ (where cq may be an arbitrary conjunctive query)
  - for each such mapping we introduce in P a new source predicate symbol E (called external source)
  - the symbol E has the same arity as cq
  - we add to P a local mapping assertion $\{\, \vec{\mathbf{x}} \mid E(x)\} \leadsto cq$
- Given a query Q in $\mathcal{L}$, a Datalog query Q_1 is a perfect reformulation of Q if
  - Q_1 is expressed over the original and the external source predicates of P
  - for each source database D for P (i.e., over the original and the external sources), we have that

$$Q_1^D = ans(Q, P, D)$$

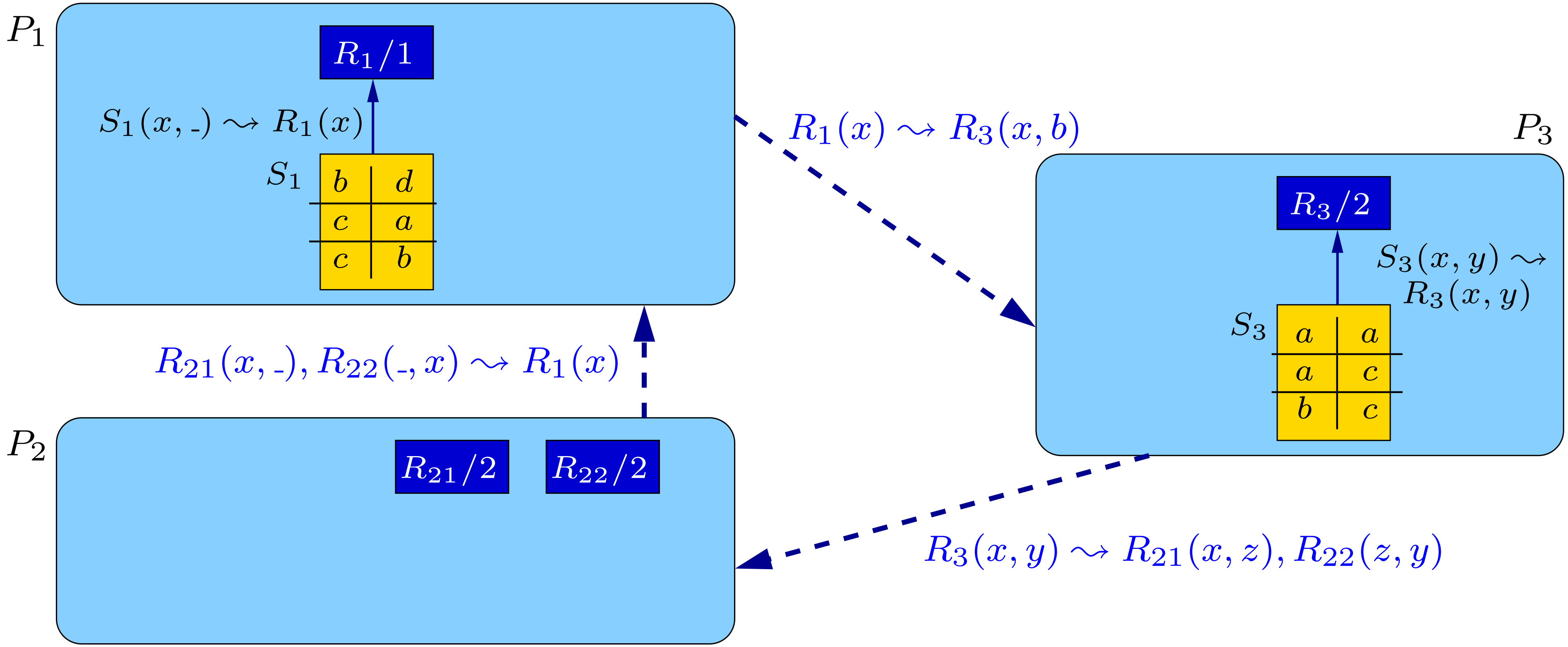
- Perfect reformulations exists in several settings

Distributed query answering in P2P systems

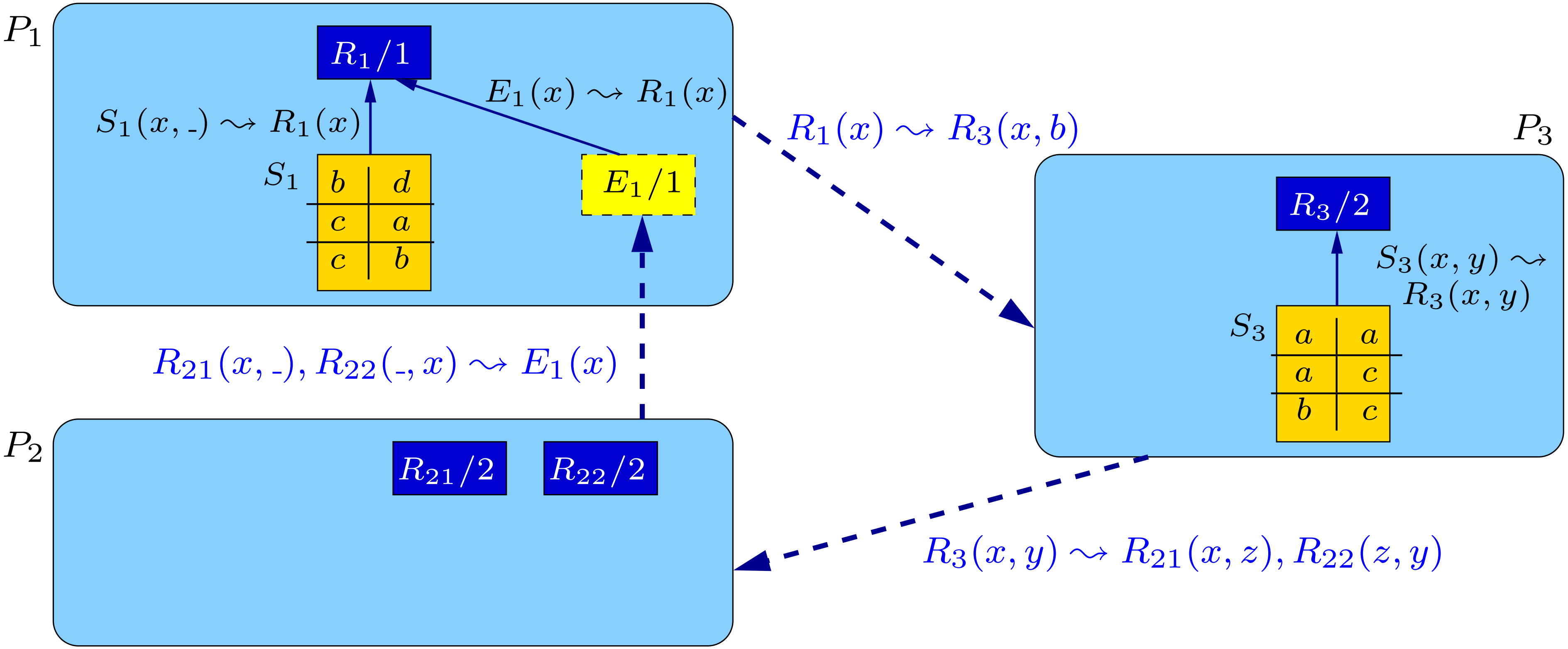
We have devised a distributed query algorithm based on the following ideas

- Each peer reformulates the queries that are requested to it in terms of the local and external sources
- A reference to an external source triggers a request to the peer to which the external source is connected
- Answers to such requests consist of a Datalog program with two parts:
  - an extensional part, which is a set of facts (about source relations received from other peers)
  - an intensional part, which is a set of Datalog rules
- Infinite looping is avoided by:
  - associating to each (user) query a unique (global) transaction id
  - avoiding requests that have already been made for the same transaction id

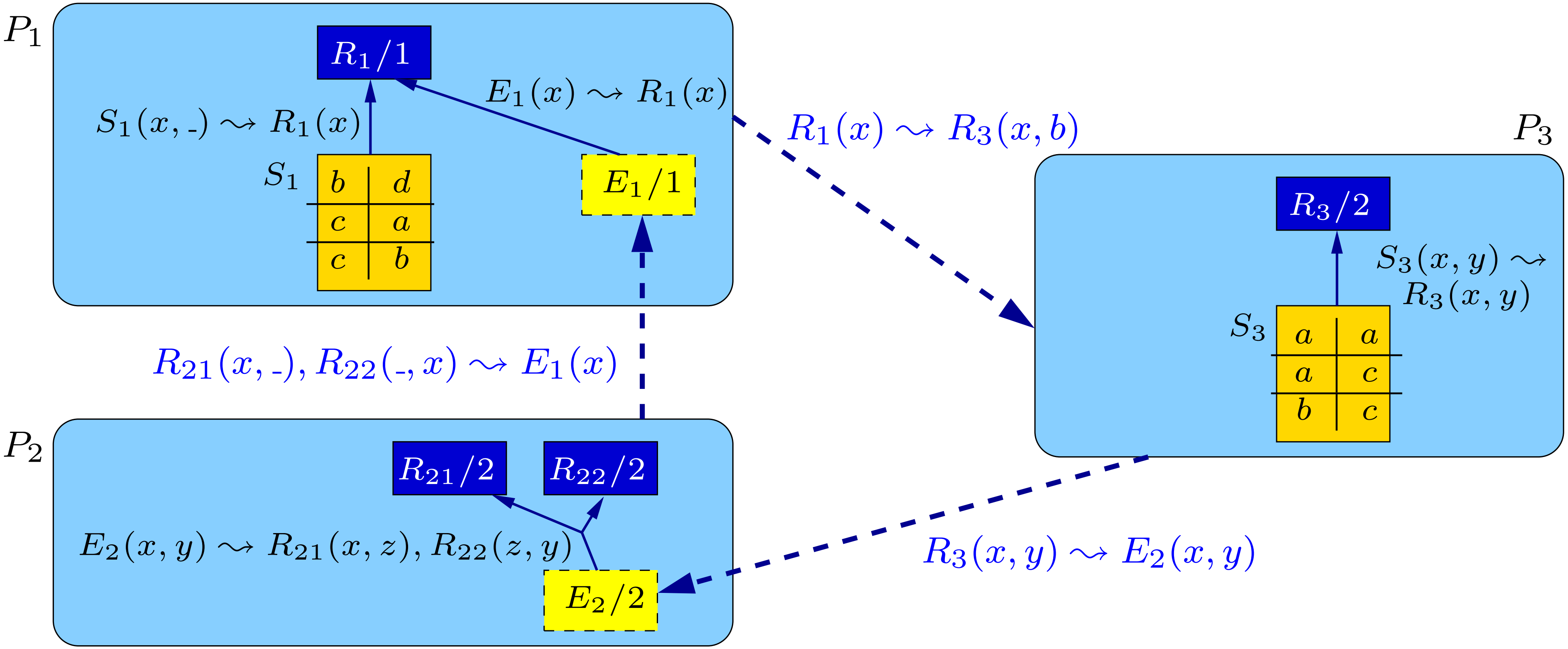
Query answering technique: example



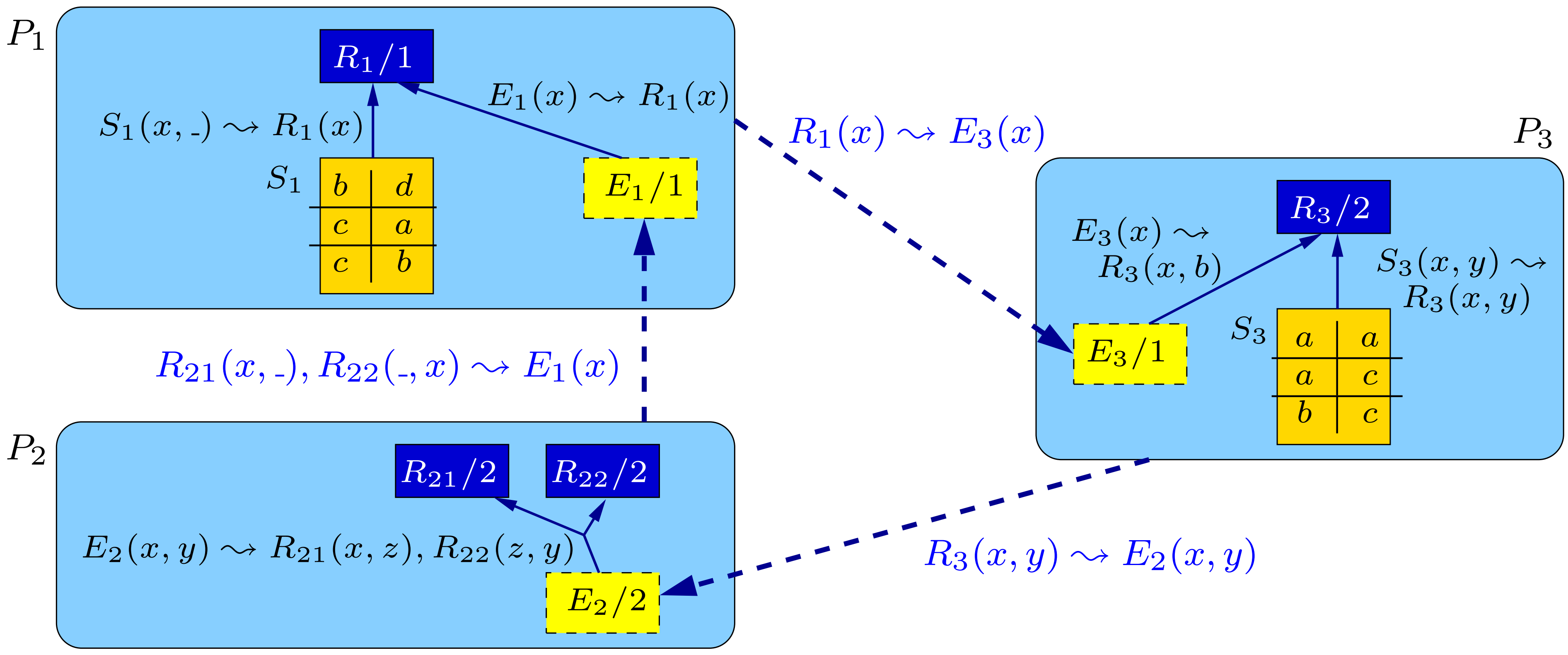
Query answering technique: example



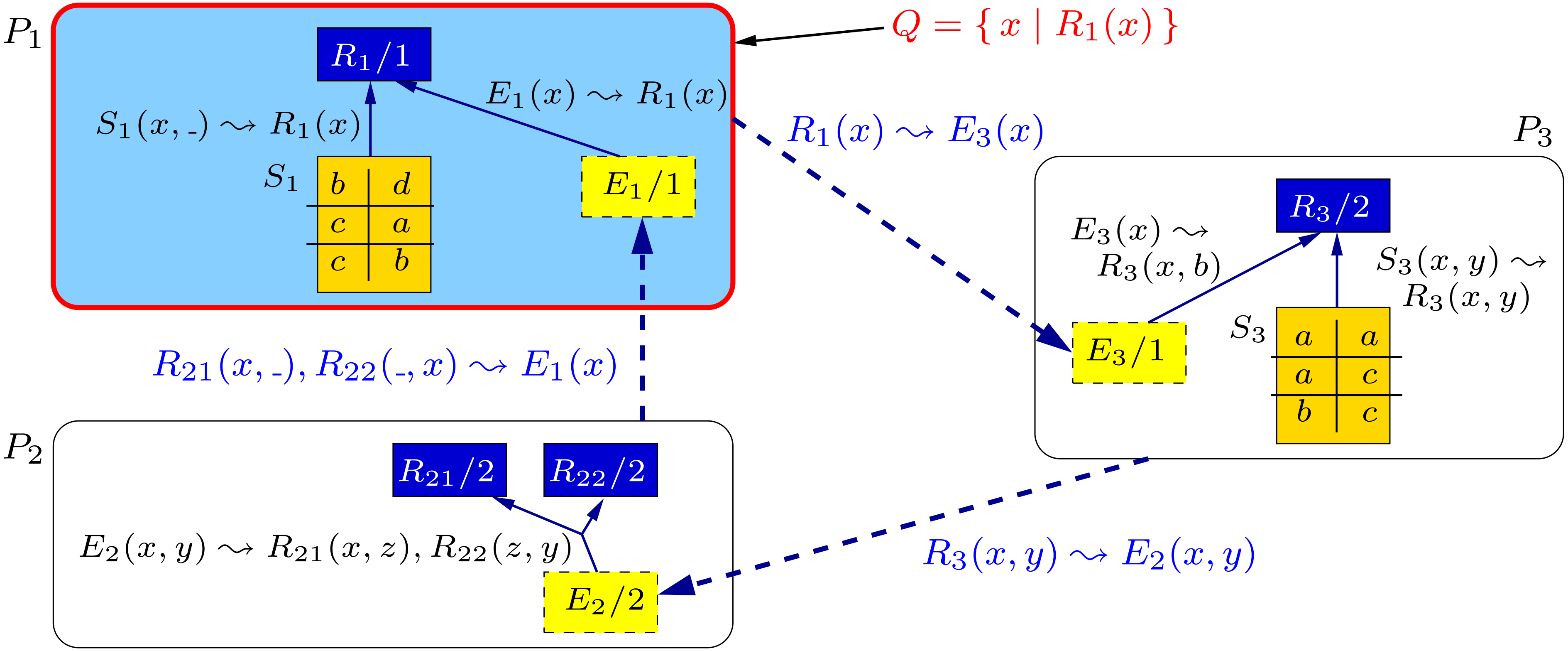
Query answering technique: example



Query answering technique: example

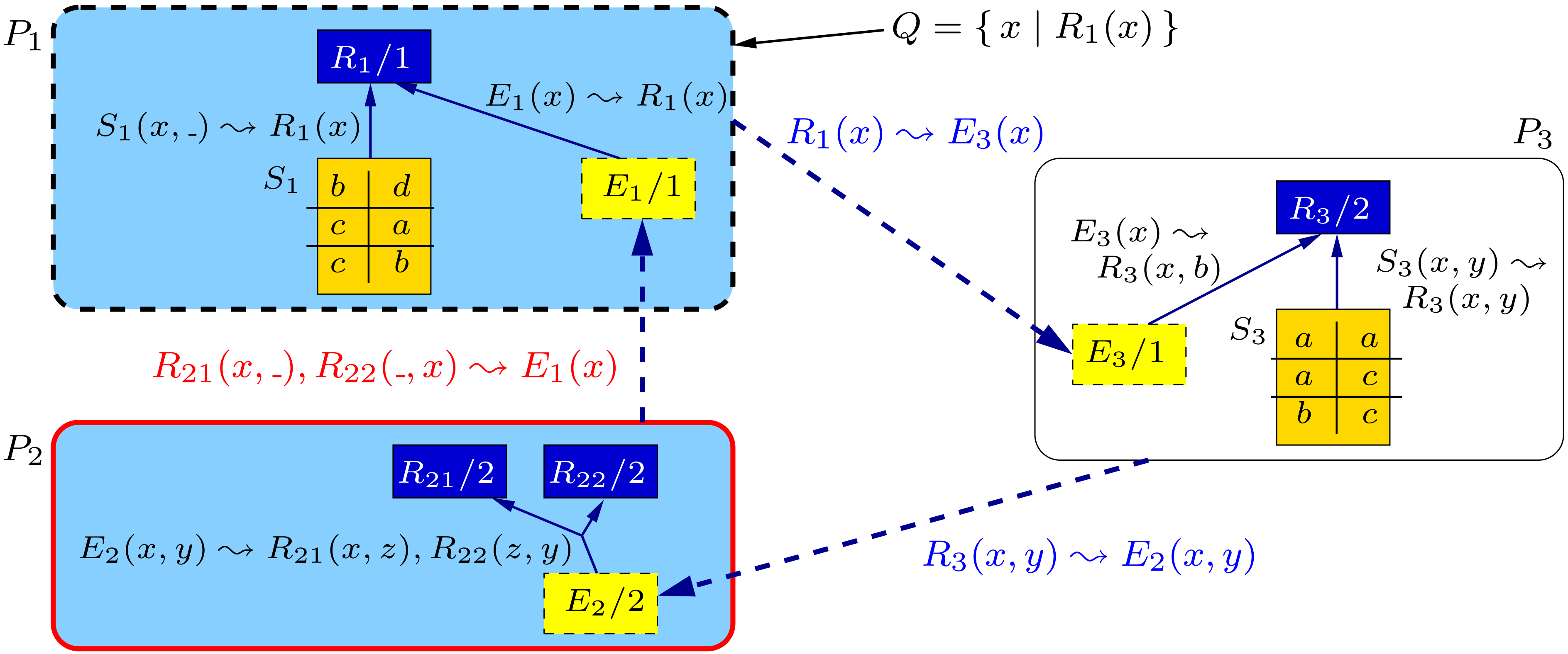


Query answering technique: example



$$1 \left\{ \begin{array}{lcl} Q(x) & \leftarrow & S_1(x, \_) \\ Q(x) & \leftarrow & E_1(x) \end{array} \right.$$

Query answering technique: example



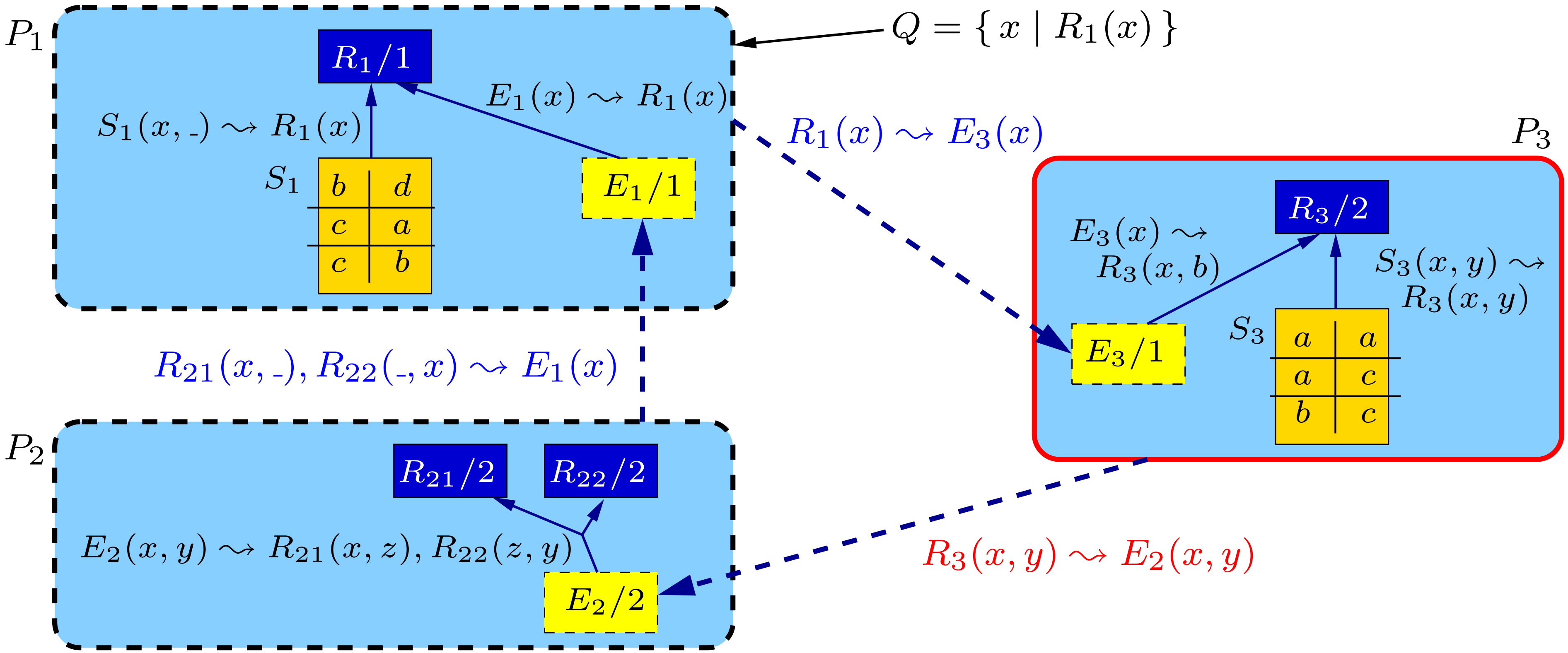
S_1	
b	d
c	a
c	b

S_3	
a	a
a	c
b	c

$$1 \left\{ \begin{array}{lcl} Q(x) & \leftarrow & S_1(x, \_) \\ Q(x) & \leftarrow & E_1(x) \end{array} \right.$$

$$2 \left\{ \begin{array}{lcl} E_1(x) & \leftarrow & E_2(x, \_), E_2(\_, x) \end{array} \right.$$

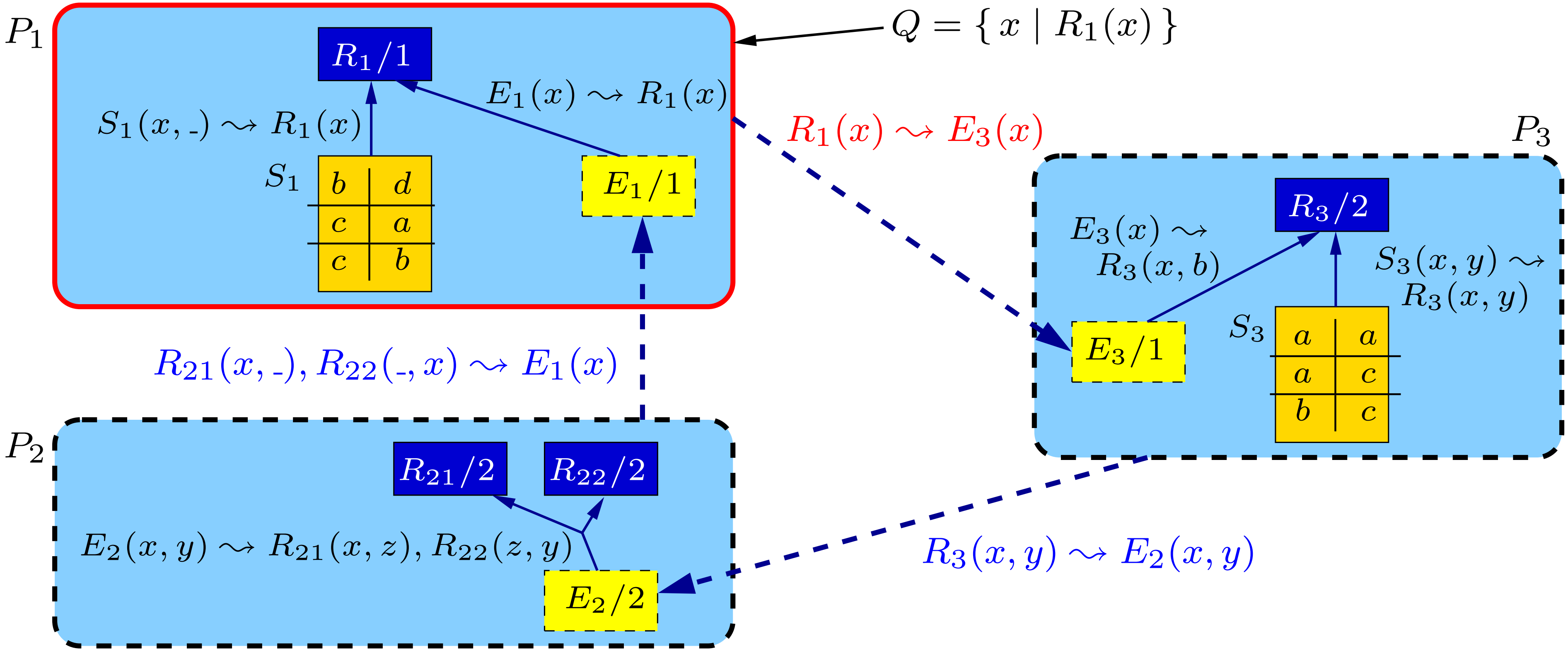
Query answering technique: example



$$1 \left\{ \begin{array}{lcl} Q(x) & \leftarrow & S_1(x, \_) \\ Q(x) & \leftarrow & E_1(x) \end{array} \right. \qquad 3 \left\{ \begin{array}{lcl} E_2(x, y) & \leftarrow & S_3(x, y) \\ E_2(x, y) & \leftarrow & E_3(x), y = b \end{array} \right.$$

$$2 \left\{ \begin{array}{lcl} E_1(x) & \leftarrow & E_2(x, \_), E_2(\_, x) \end{array} \right.$$

Query answering technique: example



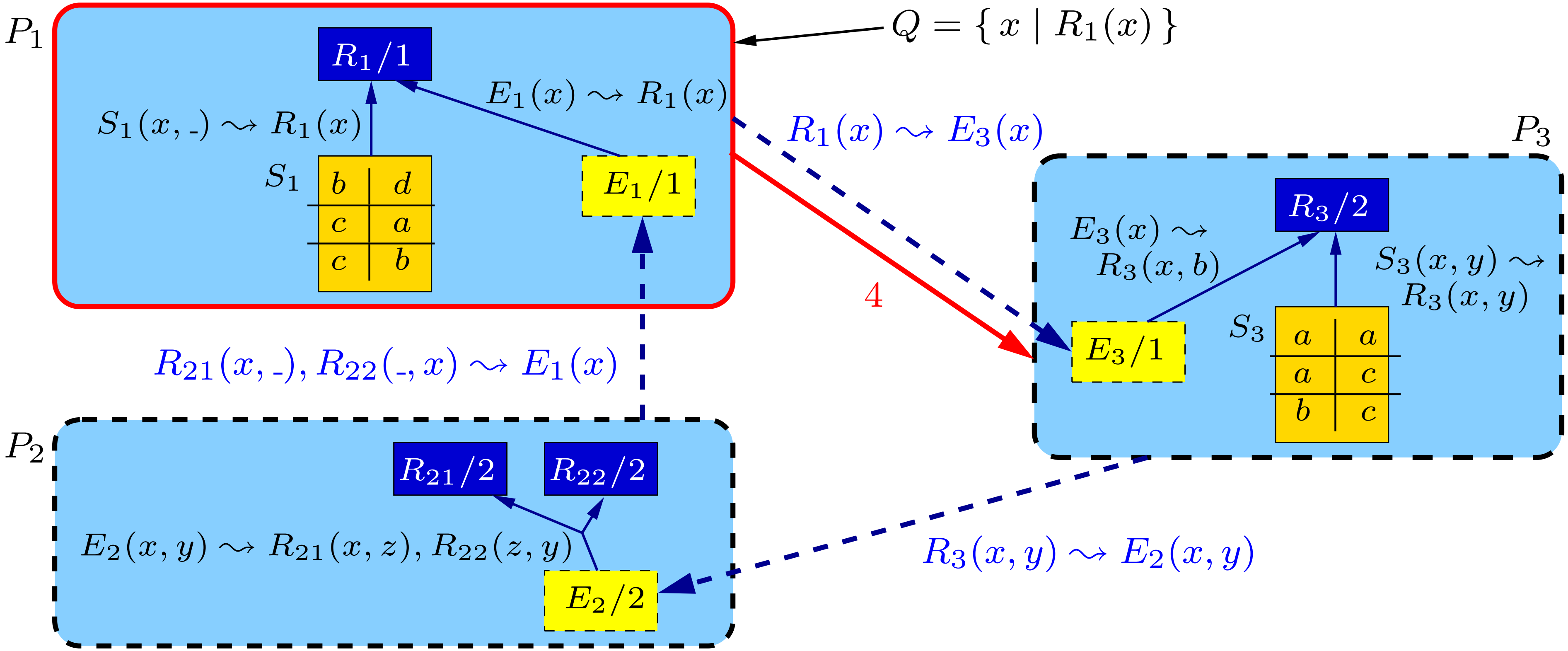
$$1 \left\{ \begin{array}{lcl} Q(x) & \leftarrow & S_1(x,\_) \\ Q(x) & \leftarrow & E_1(x) \end{array} \right.$$

$$2 \left\{ \begin{array}{lcl} E_1(x) & \leftarrow & E_2(x,\_), E_2(\_,x) \end{array} \right.$$

$$3 \left\{ \begin{array}{lcl} E_2(x,y) & \leftarrow & S_3(x,y) \\ E_2(x,y) & \leftarrow & E_3(x), y = b \end{array} \right.$$

$$4 \left\{ \begin{array}{lcl} E_3(x) & \leftarrow & S_1(x,\_) \\ E_3(x) & \leftarrow & E_1(x) \end{array} \right.$$

Query answering technique: example



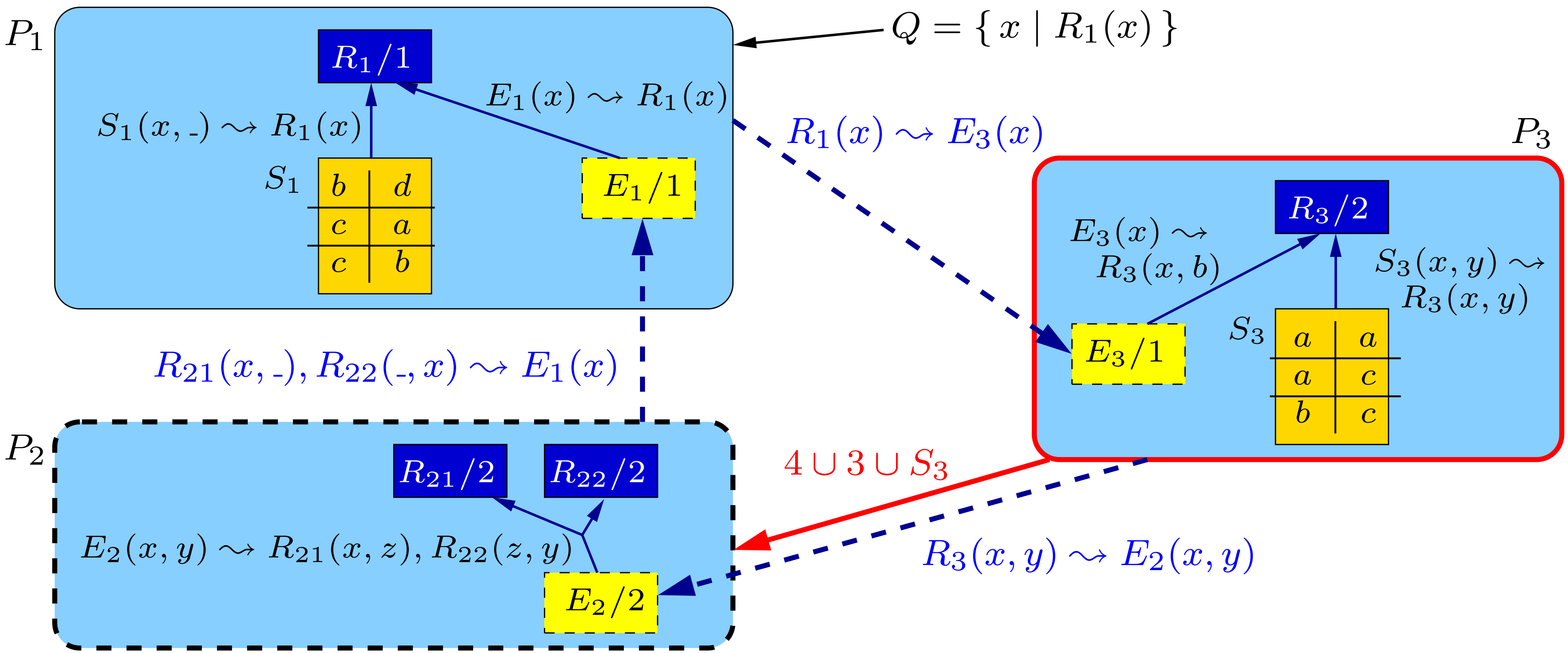
$$1 \left\{ \begin{array}{lcl} Q(x) & \leftarrow & S_1(x, \_) \\ Q(x) & \leftarrow & E_1(x) \end{array} \right.$$

$$3 \left\{ \begin{array}{lcl} E_2(x, y) & \leftarrow & S_3(x, y) \\ E_2(x, y) & \leftarrow & E_3(x), y = b \end{array} \right.$$

$$2 \left\{ \begin{array}{lcl} E_1(x) & \leftarrow & E_2(x, \_), E_2(\_, x) \end{array} \right.$$

$$4 \left\{ \begin{array}{lcl} E_3(x) & \leftarrow & S_1(x, \_) \\ E_3(x) & \leftarrow & E_1(x) \end{array} \right.$$

Query answering technique: example



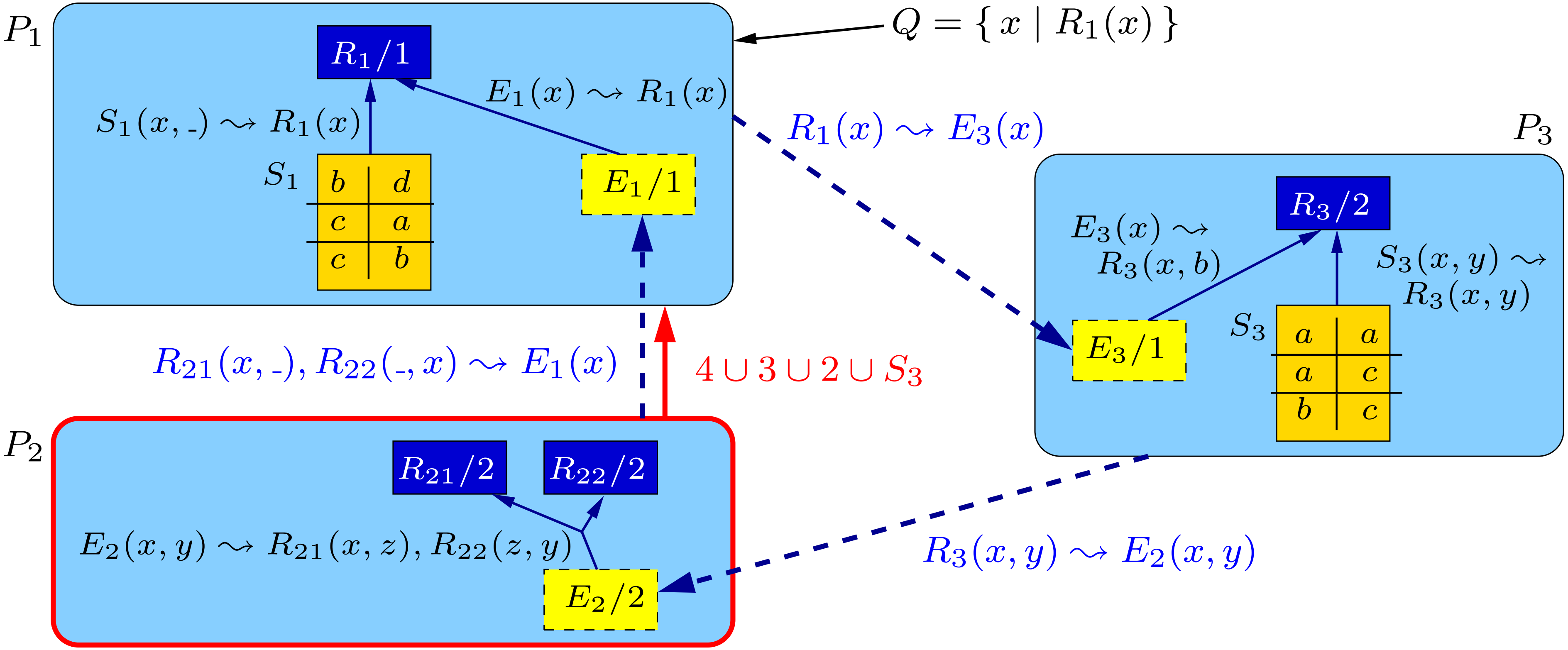
$$1 \begin{cases} Q(x) & \leftarrow & S_1(x,\_) \\ Q(x) & \leftarrow & E_1(x) \end{cases}$$

$$2 \begin{cases} E_1(x) & \leftarrow & E_2(x,\_), E_2(\_,x) \end{cases}$$

$$3 \begin{cases} E_2(x,y) & \leftarrow & S_3(x,y) \\ E_2(x,y) & \leftarrow & E_3(x), y = b \end{cases}$$

$$4 \begin{cases} E_3(x) & \leftarrow & S_1(x,\_) \\ E_3(x) & \leftarrow & E_1(x) \end{cases}$$

Query answering technique: example



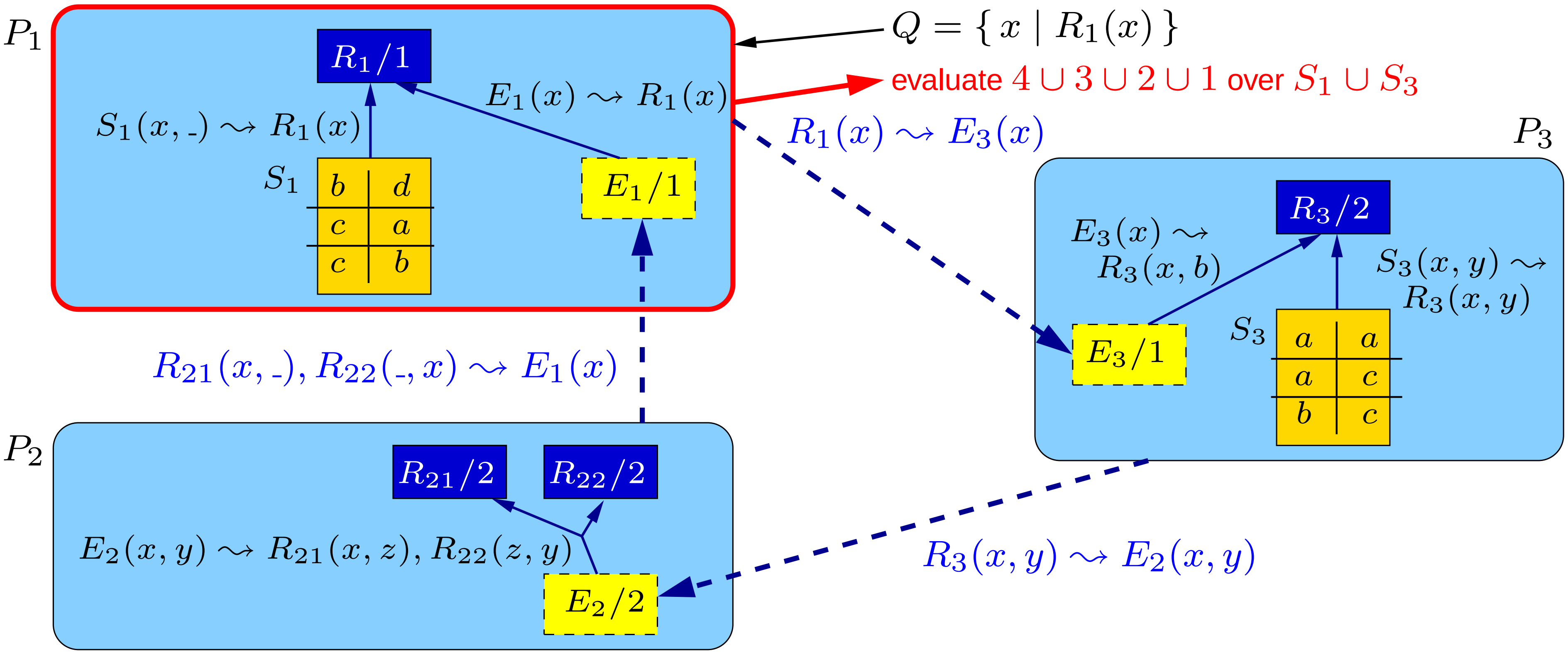
$$1 \left\{ \begin{array}{lcl} Q(x) & \leftarrow & S_1(x, \_) \\ Q(x) & \leftarrow & E_1(x) \end{array} \right.$$

$$2 \left\{ \begin{array}{lcl} E_1(x) & \leftarrow & E_2(x, \_), E_2(\_, x) \end{array} \right.$$

$$3 \left\{ \begin{array}{lcl} E_2(x, y) & \leftarrow & S_3(x, y) \\ E_2(x, y) & \leftarrow & E_3(x), y = b \end{array} \right.$$

$$4 \left\{ \begin{array}{lcl} E_3(x) & \leftarrow & S_1(x, \_) \\ E_3(x) & \leftarrow & E_1(x) \end{array} \right.$$

Query answering technique: example



$$1 \begin{cases} Q(x) & \leftarrow & S_1(x,\_) \\ Q(x) & \leftarrow & E_1(x) \end{cases} \qquad 3 \begin{cases} E_2(x,y) & \leftarrow & S_3(x,y) \\ E_2(x,y) & \leftarrow & E_3(x), y = b \end{cases}$$

$$2 \left\{ E_1(x) \leftarrow E_2(x,\_), E_2(\_,x) \right. \qquad 4 \begin{cases} E_3(x) & \leftarrow & S_1(x,\_) \\ E_3(x) & \leftarrow & E_1(x) \end{cases}$$

Query answering algorithm

Each peer provides two main functionalities:

- answering a user query by initiating a new transaction
- computing the Datalog program for a request coming from a peer

The algorithm is executed over a source database $\mathcal{D}$ representing the state of all peers

Algorithm P.user-query-handler

Input: user query $q \in \mathcal{L}$

Output: set of tuples for q

begin

generate a new transaction id T;

DP $:=$ P.peer-query-handler(q, r_q, T);

return *Eval*$(r_q,$ *DP*$)$

end

Query answering algorithm (cont'd)

Algorithm P.peer-query-handler

Input: query $q \in \mathcal{L}$, query predicate r_q, transaction id T

Output: Datalog program *DP* $= ($*DP*$_I,$ *DP*$_E)$

DP$_I$ $:=$ computePerfectRef(q, r_q, P); *DP*$_E$ $:= \emptyset$;

for each predicate $r \in S \cup$ *AuxAlph*(P) occurring in *DP*$_I$ **do**

if getTransaction$(r, T) =$ *notProcessed* **then**

setTransaction$(r, T,$ *processed*$)$;

if $r \in S$ **then** /* r is a source symbol in P */

DP$_E$ $:=$ *DP*$_E$ $\cup$ *Extension*$(r, \mathcal{D})$

else /* r is an external source symbol */

DP$'$ $:=$ peer(r).peer-query-handler$($query$(r), r, T)$;

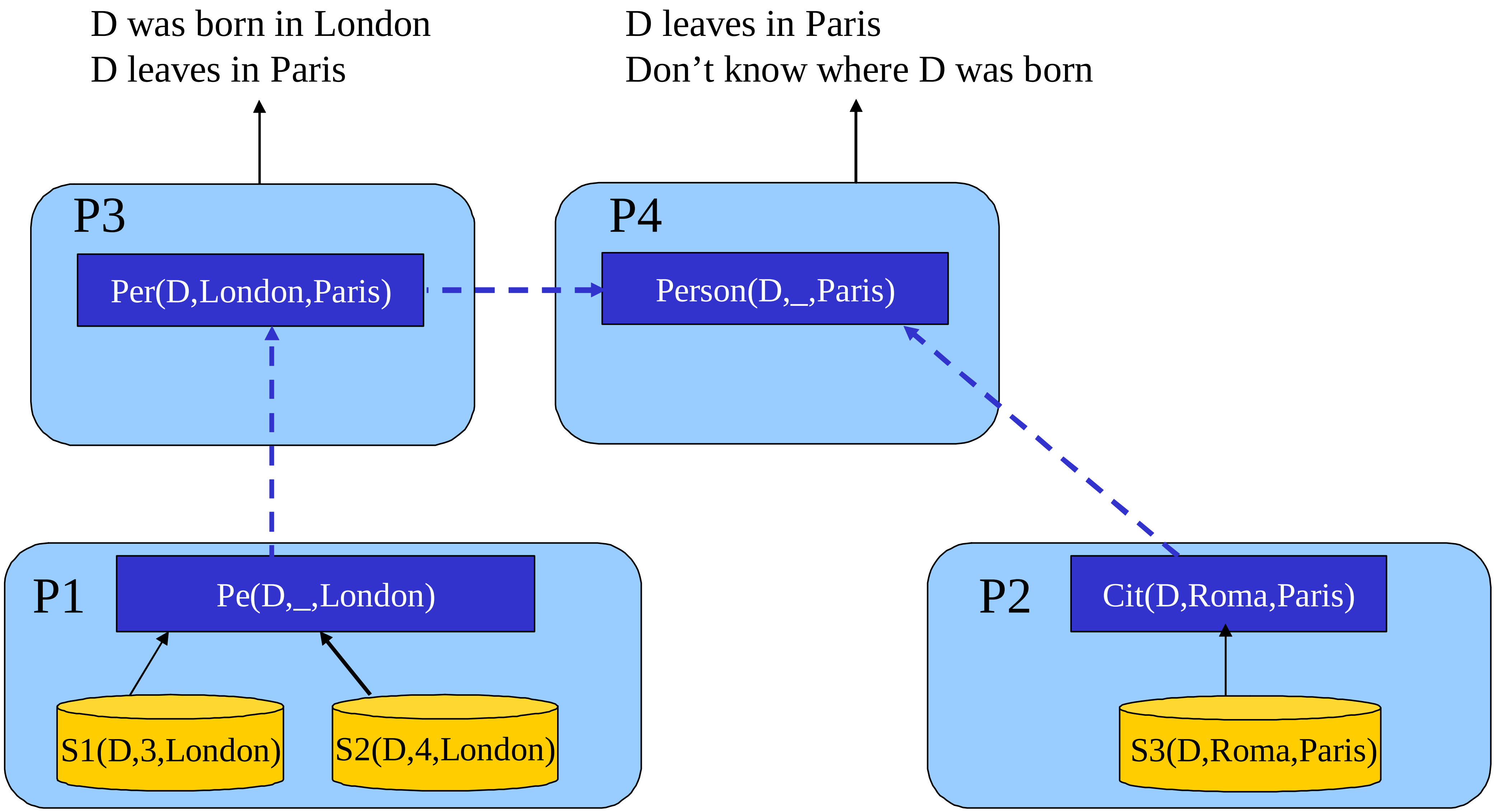
DP$_I$ $:=$ *DP*$_I$ $\cup$ *DP*$'_I$; *DP*$_E$ $:=$ *DP*$_E$ $\cup$ *DP*$'_E$

return *DP*

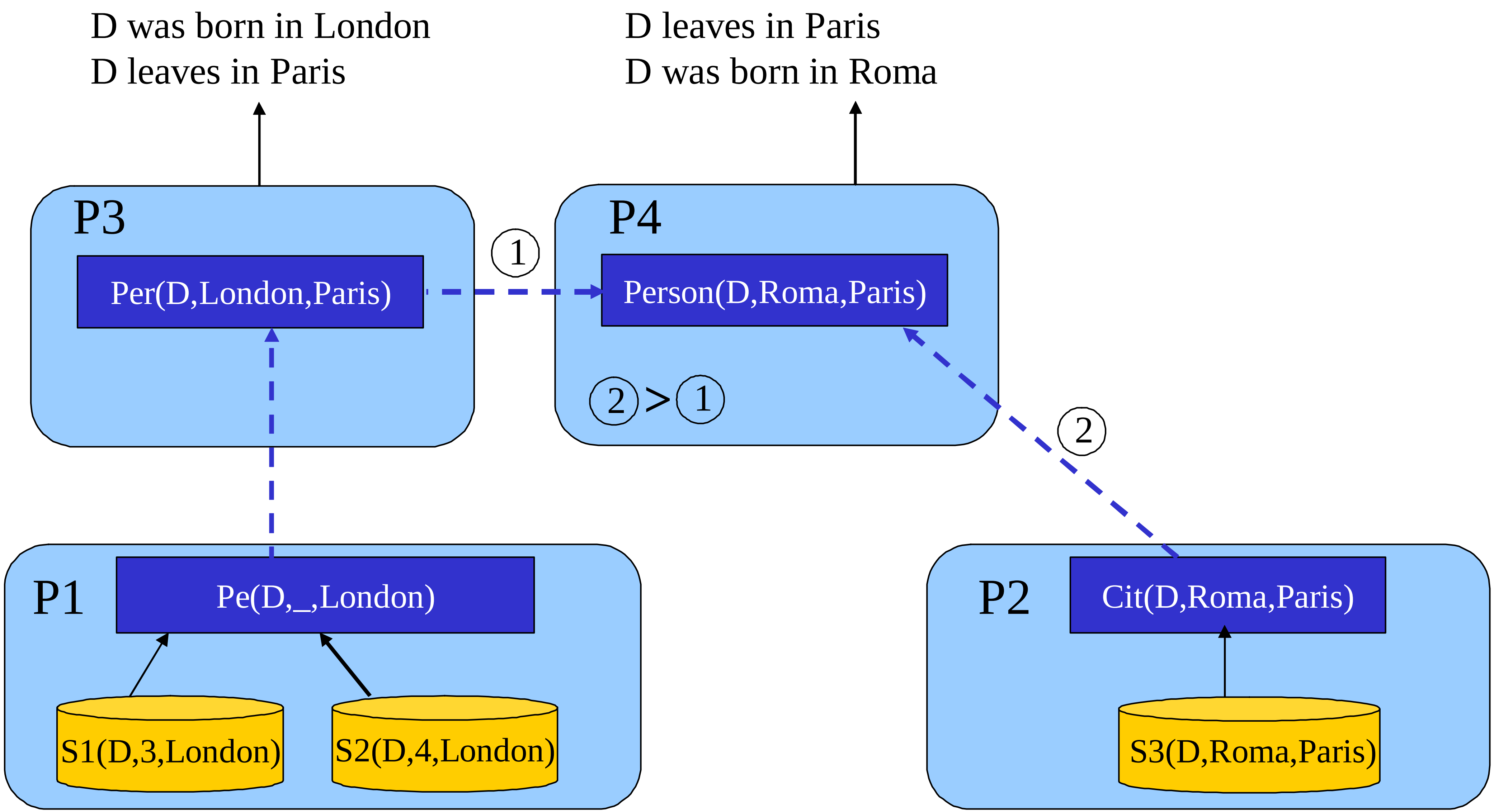
Properties of the algorithm

- It is an algorithm, i.e., it always terminates
- Sound and complete for the epistemic semantics
- Runs in polynomial time in the size of the source database
- Notice that the reformulation step is independent of the data, and hence does not affect data complexity

Dealing with inconsistencies: example



Dealing with inconsistencies and preferences: example



Outline

- Peer-based Distributed Information Systems
- Mediator-based data integration
- Data exchange
- P2P data integration
- Conclusions

Conclusions

Many open problems and issues, including

- Classes of integrity constraints in peer schemas affect the perfect reformulation problem
- Global schema (or target schema, or peer schemas) expressed in terms of semi-structured data (with constraints)
- Limitations in accessing the sources
- Privacy-based restrictions on peer answers
- Dealing with inconsistencies vs. data cleaning
- How to incorporate the notion of data quality (peer reliability, accuracy, etc.)
- Optimization
- Going beyond the “unique domain assumption”, i.e., by means of so-called mapping tables [Arenas&al SIGMOD03]
- Adding materialization