Data integration: A theoretical perspective

Maurizio Lenzerini

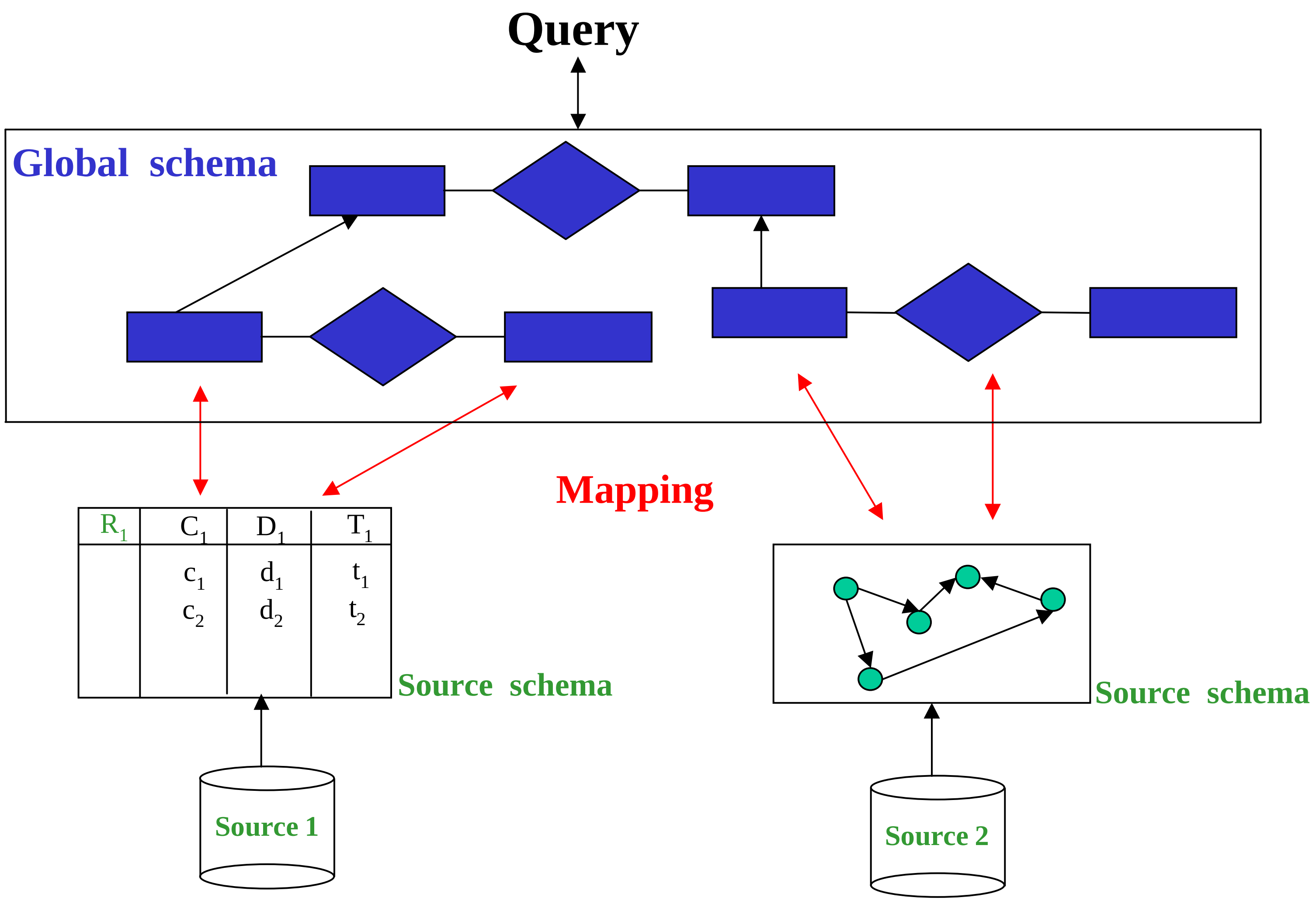
Dipartimento di Informatica e Sistemistica “Antonio Ruberti”

Università di Roma “La Sapienza”

Tutorial at PODS 2002

Madison, Wisconsin, USA, June 2002

Data integration



Outline

- Formal framework for data integration
- Approaches to data integration
- Query answering in different approaches
- Dealing with inconsistency
- Reasoning on queries in data integration
- Conclusions

Formal framework

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

- $\mathcal{G}$ is the global schema (over an alphabet $\mathcal{A}_{\mathcal{G}}$)
- $\mathcal{S}$ is the source schema (over an alphabet $\mathcal{A}_{\mathcal{S}}$)
- $\mathcal{M}$ is the mapping between $\mathcal{G}$ and $\mathcal{S}$

Semantics of $\mathcal{I}$: which are the databases that satisfy $\mathcal{I}$ (models of $\mathcal{I}$)?

We refer only to databases over a fixed infinite domain Γ, and we start with **a source database** $\mathcal{C}$, (data available at the sources, also called source model) over Γ. The set of databases that satisfy $\mathcal{I}$ relative to $\mathcal{C}$ is:

$$sem^{\mathcal{C}}(\mathcal{I}) = \{\, \mathcal{B} \mid \mathcal{B} \text{ is legal wrt } \mathcal{G} \text{ and satisfies } \mathcal{M} \text{ wrt } \mathcal{C} \,\}$$

Semantics of queries to $\mathcal{I}$

A **query** q of arity n is a FOL 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 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 query over $\mathcal{A}_{\mathcal{G}}$), then the set of **certain answers to q wrt $\mathcal{I}$ and $\mathcal{C}$** is

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

Databases with incomplete information

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

  Query answering means evaluating a formula in the model.

- Database with incomplete information: 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 database with incomplete information under constraints.

Outline

- Formal framework for data integration
- Approaches to data integration
- Query answering in different approaches
- Dealing with inconsistency
- Reasoning on queries in data integration
- Conclusions

The mapping

How is the mapping $\mathcal{M}$ between $\mathcal{G}$ and $\mathcal{S}$ 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**.

- Mapping between sources, without global schema?

  Approach called **P2P**.

GAV vs LAV – example

Global schema: movie$(Title, Year, Director)$

european$(Director)$

review$(Title, Critique)$

Source 1: $r_1(Title, Year, Director)$ since 1960, european directors

Source 2: $r_2(Title, Critique)$ since 1990

Query: Title and critique of movies in 1998

$\exists D.$ movie$(T, 1998, D) \wedge$ review(T, R), written

$\{ \ (T, R) \mid$ movie$(T, 1998, D) \wedge$ review$(T, R) \ \}$

Formalization of LAV

In LAV, the mapping $\mathcal{M}$ is constituted by a set of assertions:

$$\begin{array}{llll} s & \subseteq & \phi_{\mathcal{G}} & \text{(sound source)} \quad \forall \vec{\mathbf{x}}\,(s(\vec{\mathbf{x}}) \rightarrow \phi_{\mathcal{G}}(\vec{\mathbf{x}})) \\ s & \equiv & \phi_{\mathcal{G}} & \text{(exact source)} \quad \forall \vec{\mathbf{x}}\,(s(\vec{\mathbf{x}}) \equiv \phi_{\mathcal{G}}(\vec{\mathbf{x}})) \end{array}$$

one for each source element s in $\mathcal{A}_{\mathcal{S}}$, where $\phi_{\mathcal{G}}$ is a **query** over $\mathcal{G}$. Given source database $\mathcal{C}$, a database $\mathcal{B}$ for $\mathcal{G}$ satisfies $\mathcal{M}$ wrt $\mathcal{C}$ if for each $s \in \mathcal{S}$:

$$\begin{array}{llll} s^{\mathcal{C}} & \subseteq & \phi_{\mathcal{G}}{}^{\mathcal{B}} & \text{(sound source)} \\ s^{\mathcal{C}} & = & \phi_{\mathcal{G}}{}^{\mathcal{B}} & \text{(exact source)} \end{array}$$

The mapping $\mathcal{M}$ and the source database $\mathcal{C}$ do **not** provide direct information about which data satisfy the global schema. Sources are views, and we have to answer queries on the basis of the available data in the views.

LAV – example

Global schema: $\text{movie}(Title, Year, Director)$
$\text{european}(Director)$
$\text{review}(Title, Critique)$

LAV: associated to source relations we have views over the global schema

$$\text{r}_1(T, Y, D) \quad \subseteq \quad \{\ (T, Y, D) \mid \text{movie}(T, Y, D) \wedge \text{european}(D) \wedge Y \geq 1960\ \}$$

$$\text{r}_2(T, R) \quad \subseteq \quad \{\ (T, R) \mid \text{movie}(T, Y, D) \wedge \text{review}(T, R) \wedge Y \geq 1990\ \}$$

The query $\{\ (T, R) \mid \text{movie}(T, 1998, D) \wedge \text{review}(T, R)\ \}$ is processed by means of an inference mechanism that aims at re-expressing the atoms of the global schema in terms of atoms at the sources. In this case:

$$\{\ (T, R) \ \mid\ \text{r}_2(T, R) \wedge \text{r}_1(T, 1998, D)\ \}$$

Formalization of GAV

In GAV, the mapping $\mathcal{M}$ is constituted by a set of assertions:

$$\begin{array}{llll} g & \supseteq & \phi_{\mathcal{S}} & \text{(sound source)} \quad \forall \vec{\mathbf{x}}\, (\phi_{\mathcal{S}}(\vec{\mathbf{x}}) \rightarrow g(\vec{\mathbf{x}})) \\ g & \equiv & \phi_{\mathcal{S}} & \text{(exact source)} \quad \forall \vec{\mathbf{x}}\, (\phi_{\mathcal{S}}(\vec{\mathbf{x}}) \equiv g(\vec{\mathbf{x}})) \end{array}$$

one for each element g in $\mathcal{A}_{\mathcal{G}}$, where $\phi_{\mathcal{S}}$ is a **query** over $\mathcal{S}$. Given source database $\mathcal{C}$, a database $\mathcal{B}$ for $\mathcal{G}$ satisfies $\mathcal{M}$ wrt $\mathcal{C}$ if for each $g \in \mathcal{G}$:

$$\begin{array}{llll} g^{\mathcal{B}} & \supseteq & {\phi_{\mathcal{S}}}^{\mathcal{C}} & \text{(sound source)} \\ g^{\mathcal{B}} & = & {\phi_{\mathcal{S}}}^{\mathcal{C}} & \text{(exact source)} \end{array}$$

Given a source database, $\mathcal{M}$ provides direct information about which data satisfy the elements of the global schema. Relations in $\mathcal{G}$ are views, and queries are expressed over the views. Thus, it seems that we can simply evaluate the query over the data satisfying the global relations (as if we had a single database at hand).

GAV – example

Global schema: movie$(\mathit{Title}, \mathit{Year}, \mathit{Director})$

european$(\mathit{Director})$

review$(\mathit{Title}, \mathit{Critique})$

GAV: associated to relations in the global schema we have views over the sources

$$\begin{aligned}
\text{movie}(T, Y, D) &\supseteq \{\, (T, Y, D) \mid \text{r}_1(T, Y, D) \,\} \\
\text{european}(D) &\supseteq \{\, (D) \mid \text{r}_1(T, Y, D) \,\} \\
\text{review}(T, R) &\supseteq \{\, (T, R) \mid \text{r}_2(T, R) \,\}
\end{aligned}$$

GAV – example of query processing

The query $\{ (T,R) \mid \text{movie}(T,1998,D) \wedge \text{review}(T,R) \}$ is processed by means of unfolding, i.e., by expanding the atoms according to their definitions, so as to come up with source relations. In this case:

$$\begin{tikzcd}
\text{movie}(T,1998,D) \arrow[d, "\textit{unfolding}"'] & \wedge & \text{review}(T,R) \arrow[d] \\
r_1(T,1998,D) & \wedge & r_2(T,R)
\end{tikzcd}$$

GAV and LAV – comparison

LAV: (Information Manifold, DWQ, Picsel)

- Quality depends on how well we have characterized the sources
- High modularity and extensibility (if the global schema is well designed, when a source changes, only its definition is affected)
- Query processing needs reasoning (query reformulation complex)

GAV: (Carnot, SIMS, Tsimmis, IBIS, Picsel, . . .)

- Quality depends on how well we have compiled the sources into the global schema through the mapping
- Whenever a source changes or a new one is added, the global schema needs to be reconsidered
- Query processing can be based on some sort of unfolding (query reformulation looks easier)

For more details, see [Ullman, TCS'00], [Halevy, VLDBJ'01].

Beyond GAV and LAV: GLAV

In GLAV, the mapping $\mathcal{M}$ is constituted by a set of assertions:

$$
\begin{array}{llll}
\phi_{\mathcal{S}} & \subseteq & \phi_{\mathcal{G}} & \text{(sound source)} \quad \forall\vec{\mathbf{x}}\,(\phi_{\mathcal{S}}(\vec{\mathbf{x}}) \rightarrow \phi_{\mathcal{G}}(\vec{\mathbf{x}})) \\
\phi_{\mathcal{S}} & \equiv & \phi_{\mathcal{G}} & \text{(exact source)} \quad \forall\vec{\mathbf{x}}\,(\phi_{\mathcal{S}}(\vec{\mathbf{x}}) \equiv \phi_{\mathcal{G}}(\vec{\mathbf{x}}))
\end{array}
$$

where $\phi_{\mathcal{S}}$ is a **query** over $\mathcal{S}$, and $\phi_{\mathcal{G}}$ is a **query** over $\mathcal{G}$. Given source database $\mathcal{C}$, a database $\mathcal{B}$ that is legal wrt $\mathcal{G}$ satisfies $\mathcal{M}$ wrt $\mathcal{C}$ if for each assertion in $\mathcal{M}$:

$$
\begin{array}{llll}
{\phi_{\mathcal{S}}}^{\mathcal{C}} & \subseteq & {\phi_{\mathcal{G}}}^{\mathcal{B}} & \text{(sound source)} \\
{\phi_{\mathcal{S}}}^{\mathcal{C}} & = & {\phi_{\mathcal{G}}}^{\mathcal{B}} & \text{(exact source)}
\end{array}
$$

The mapping $\mathcal{M}$ does **not** provide direct information about which data satisfy the global schema: to answer a query q over $\mathcal{G}$, we have to **infer** how to use $\mathcal{M}$ in order to access the source database $\mathcal{C}$.

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:

$$\begin{array}{lcl}
\{\, (r,f) \mid HasJob(r,f) \,\} & \subseteq & \{\, (r,f) \mid Work(r,p) \;\wedge\; Area(p,f) \,\} \\
\{\, (r,f) \mid Teach(r,c) \;\wedge\; In(c,f) \,\} & \subseteq & \{\, (r,f) \mid Work(r,p) \;\wedge\; Area(p,f) \,\} \\
\{\, (r,p) \mid Get(r,g) \;\wedge\; For(g,p) \,\} & \subseteq & \{\, (r,p) \mid Work(r,p) \,\}
\end{array}$$

Beyond GLAV: P2P data integration

In P2P, the global schema does not exist. Constraints (that we can still call $\mathcal{G}$) are defined over

$$\mathcal{A}_{\mathcal{G}} = \mathcal{A}_{\mathcal{S}_1} \cup \cdots \cup \mathcal{A}_{\mathcal{S}_n}$$

and the mapping $\mathcal{M}$ is constituted by a set of assertions ($\phi_1^{\mathcal{S}_i}$, $\phi_2^{\mathcal{S}_j}$ are queries over the alphabets $\mathcal{A}_{\mathcal{S}_i}$ and $\mathcal{A}_{\mathcal{S}_j}$, respectively): $\phi_1^{\mathcal{S}_i} \subseteq \phi_2^{\mathcal{S}_j}$.

$\mathcal{A}_{\mathcal{S}}$ is a distinguished subset of predicates in $\mathcal{A}_{\mathcal{G}}$, called “base predicates” (where data are). A source database is a database for the base predicates. Given source database $\mathcal{C}$, a database $\mathcal{W}$ that satisfies $\mathcal{I}$ relative to $\mathcal{C}$ is a database for $\mathcal{S}$ such that, for each assertion $\phi_1 \subseteq \phi_2$ in $\mathcal{M}$, $\phi_1^{\mathcal{W}} \subseteq \phi_2^{\mathcal{W}}$.

Queries are now expressed over alphabet $\mathcal{A}_{\mathcal{S}_i}$, and the notion of certain answers is the usual one.

A unified view

- **Alphabet**: $\mathcal{A} = \mathcal{A}_{\mathcal{G}} \cup \mathcal{A}_{\mathcal{S}}$
- **Integrity constraints**: constraints $\mathcal{G}$, and mapping $\mathcal{M}$
- **Partial database**: source database
- **Database**: data for all symbols in $\mathcal{A}$ that are both coherent with the partial database and satisfy the integrity constraints
- **Query answering**: computing the tuples that satisfies the query in every database

Under this view, the difference between LAV, GAV, GLAV, P2P is reflected in the **kinds of integrity constraints that are expressible**.

Query answering with incomplete information

- [Reiter ’84]: relational setting, databases with incomplete information modeled as a first order theory
- [Vardi ’86]: relational setting, complexity of reasoning in closed world databases with unknown values
- Several approaches both from the DB and the KR community
- [van der Meyden ’98]: survey on logical approaches to incomplete information

Connection to query containment

Query containment (under constraints $\mathcal{T}$) *is the problem of checking whether $q_1^{\mathcal{B}}$ is contained in $q_2^{\mathcal{B}}$ for every database $\mathcal{B}$ (satisfying $\mathcal{T}$), where q_1, q_2 are queries with the same arity*.

- A source database $\mathcal{C}$ can be represented as a conjunction $q_{\mathcal{C}}$ of ground literals over $\mathcal{A}_{\mathcal{S}}$ (e.g., if $\vec{\mathbf{x}}$ is in $s^{\mathcal{C}}$, then the corresponding literal is $s(\vec{\mathbf{x}})$)
- If q is a query, and $\vec{\mathbf{t}}$ is a tuple, then we denote by $q_{\vec{\mathbf{t}}}$ the query obtained by substituting the free variables of q with $\vec{\mathbf{t}}$
- The problem of checking whether $\vec{\mathbf{t}} \in q^{\mathcal{I},\mathcal{C}}$ can be reduced to the problem of checking whether $q_{\mathcal{C}}$ **is contained in $q_{\vec{\mathbf{t}}}$ under the constraints $\mathcal{G} \cup \mathcal{M}$**

The **combined complexity** of checking certain answers is identical to the complexity of query containment under constraints, and the **data complexity** is at most the complexity of query containment under constraints.

Outline

Dealing with incompleteness and inconsistency

We analyze the problem of query answering in different cases, depending on two parameters:

- **Global schema**:

  - without constraints,

  - with constraints

- **Mapping**:

  - GAV or LAV,

  - sound or complete

Given a source database $\mathcal{C}$, we call **retrieved global database** any database for $\mathcal{G}$ that satisfies the mapping wrt $\mathcal{C}$.

Incompleteness and inconsistency

Constraints in $\mathcal{G}$	Type of mapping	Incomple-teness	Inconsi-stency
no	GAV/exact	no	no
no	GAV/sound	**yes**/no	no
no	LAV/sound	**yes**	no
no	LAV/exact	**yes**	**yes**
yes	GAV/exact	no	**yes**
yes	GAV/sound	**yes**	**yes**
yes	LAV/sound	**yes**	**yes**
yes	LAV/exact	**yes**	**yes**

Incompleteness and inconsistency

Constraints in $\mathcal{G}$	Type of mapping	Incomple- teness	Inconsi- stency
no	*GAV/exact*	no	no
no	GAV/sound	**yes**/no	no
no	LAV/sound	**yes**	no
no	LAV/exact	**yes**	**yes**
yes	GAV/exact	no	**yes**
yes	GAV/sound	**yes**	**yes**
yes	LAV/sound	**yes**	**yes**
yes	LAV/exact	**yes**	**yes**

INT[noconstr, GAV/exact]: example

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

Global schema $\mathcal{G}$:

$$\text{student}(Scode, Sname, Scity)$$

$$\text{university}(Ucode, Uname)$$

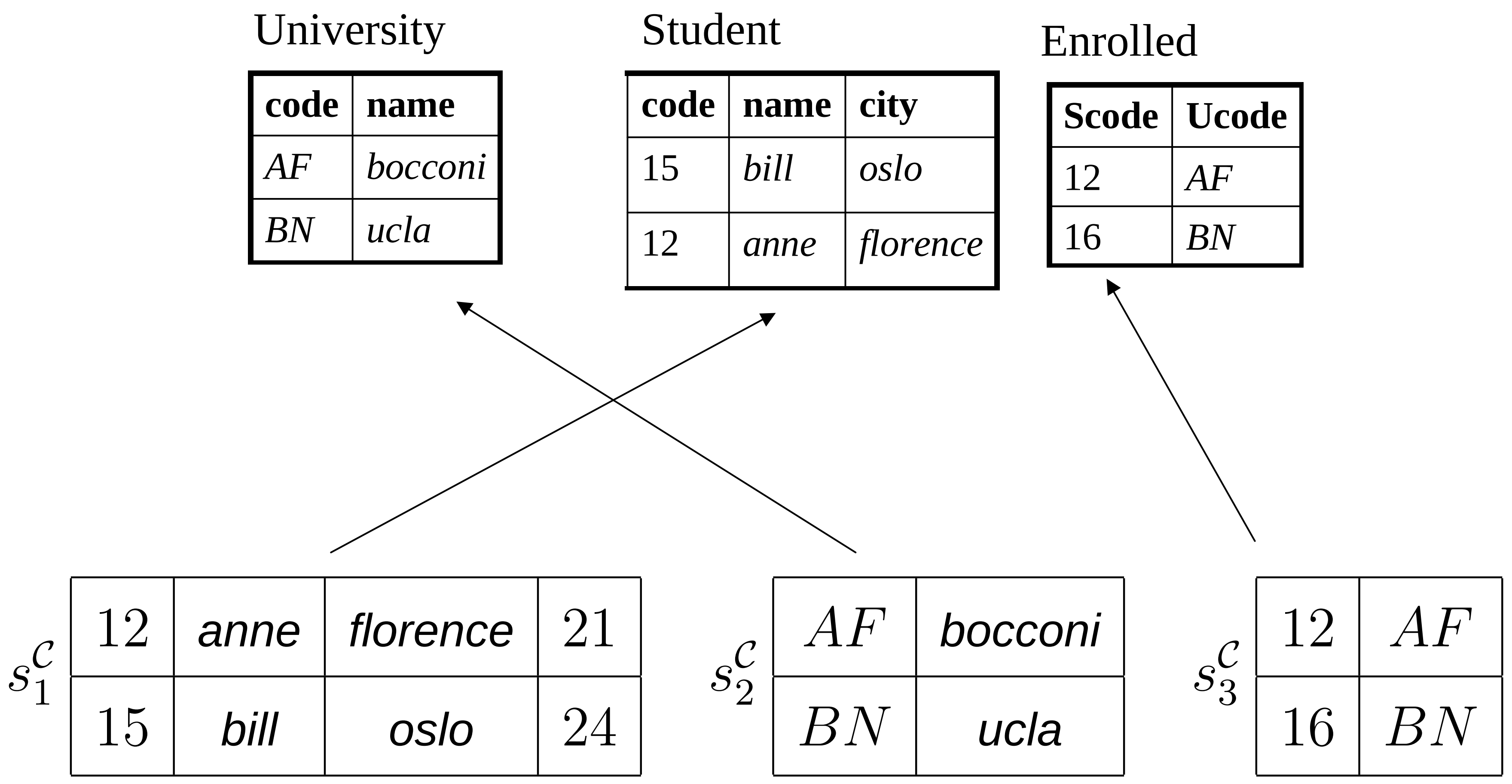
$$\text{enrolled}(Scode, Ucode)$$

Source schema $\mathcal{S}$: database relations $\text{s}_1, \text{s}_2, \text{s}_3$

Mapping $\mathcal{M}$:

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

INT[noconstr, GAV/exact]: 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

$s_2^{\mathcal{C}}$

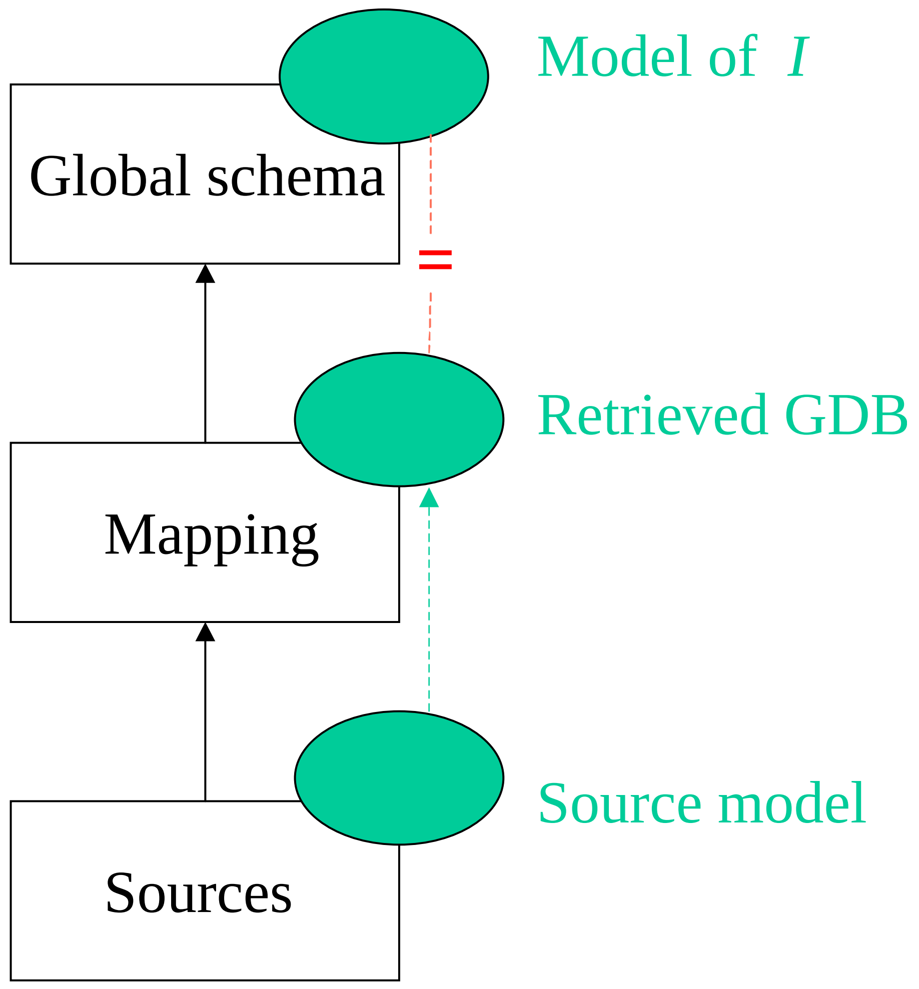
AF	*bocconi*
BN	*ucla*

$s_3^{\mathcal{C}}$

12	AF
16	BN

Example of source database and corresponding retrieved global database

INT[noconstr, GAV/exact]



INT[noconstr, GAV/exact]: query answering

- Use $\mathcal{M}$ for computing from $\mathcal{C}$ the retrieved global database, where each element g of $\mathcal{G}$ satisfies exactly the tuples of $\mathcal{C}$ satisfying the $\phi_{\mathcal{S}}$ that $\mathcal{M}$ associates to g
- Since $\mathcal{G}$ **does not have constraints**, the retrieved global database is legal wrt $\mathcal{G}$
- Actually, it is the only database that is legal wrt $\mathcal{G}$, and that satisfies $\mathcal{M}$ wrt $\mathcal{C}$
- Thus, we can simply evaluate the query q over the retrieved global database, which is equivalent to unfolding the query according to $\mathcal{M}$, in order to obtain a query on $\mathcal{A}_{\mathcal{S}}$ to be evaluated over $\mathcal{C}$

Answering queries to $\mathcal{I}$ means answering queries to a single database.

INT[noconstr, GAV/exact]: example of query answering

Mapping $\mathcal{M}$:

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

$s_1^{\mathcal{C}}$

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

$s_2^{\mathcal{C}}$

AF	*bocconi*
BN	*ucla*

$s_3^{\mathcal{C}}$

12	AF
16	BN

Query: $\{\,(X) \mid \text{student}(X,Y,Z), \text{enrolled}(X,W)\,\}$

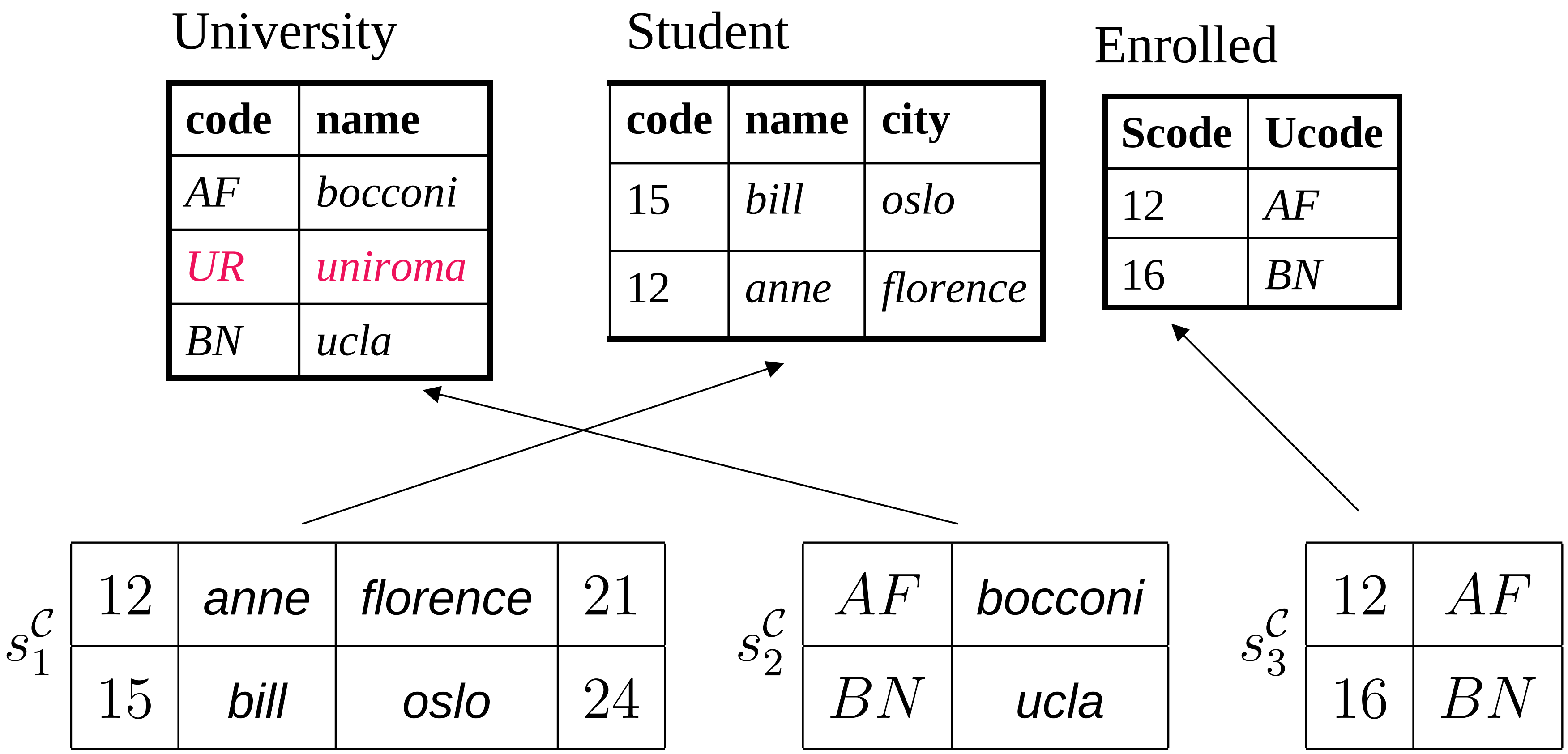
Unfolding wrt $\mathcal{M}$: $\{\,(X) \mid s_1(X,Y,Z,V), s_3(X,W)\,\}$

retrieves the answer $\{12\}$ from $\mathcal{C}$. A simple unfolding strategy is **sufficient** in this context.

Incompleteness and inconsistency

Constraints in $\mathcal{G}$	Type of mapping	Incomple-teness	Inconsi-stency
no	GAV/exact	no	no
no	*GAV/sound*	**yes**/no	no
no	LAV/sound	**yes**	no
no	LAV/exact	**yes**	**yes**
yes	GAV/exact	no	**yes**
yes	GAV/sound	**yes**	**yes**
yes	LAV/sound	**yes**	**yes**
yes	LAV/exact	**yes**	**yes**

INT[noconstr, GAV/sound]: example



University

code	name
AF	*bocconi*
UR	*uniroma*
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 and corresponding retrieved global database

INT[noconstr, GAV/sound]

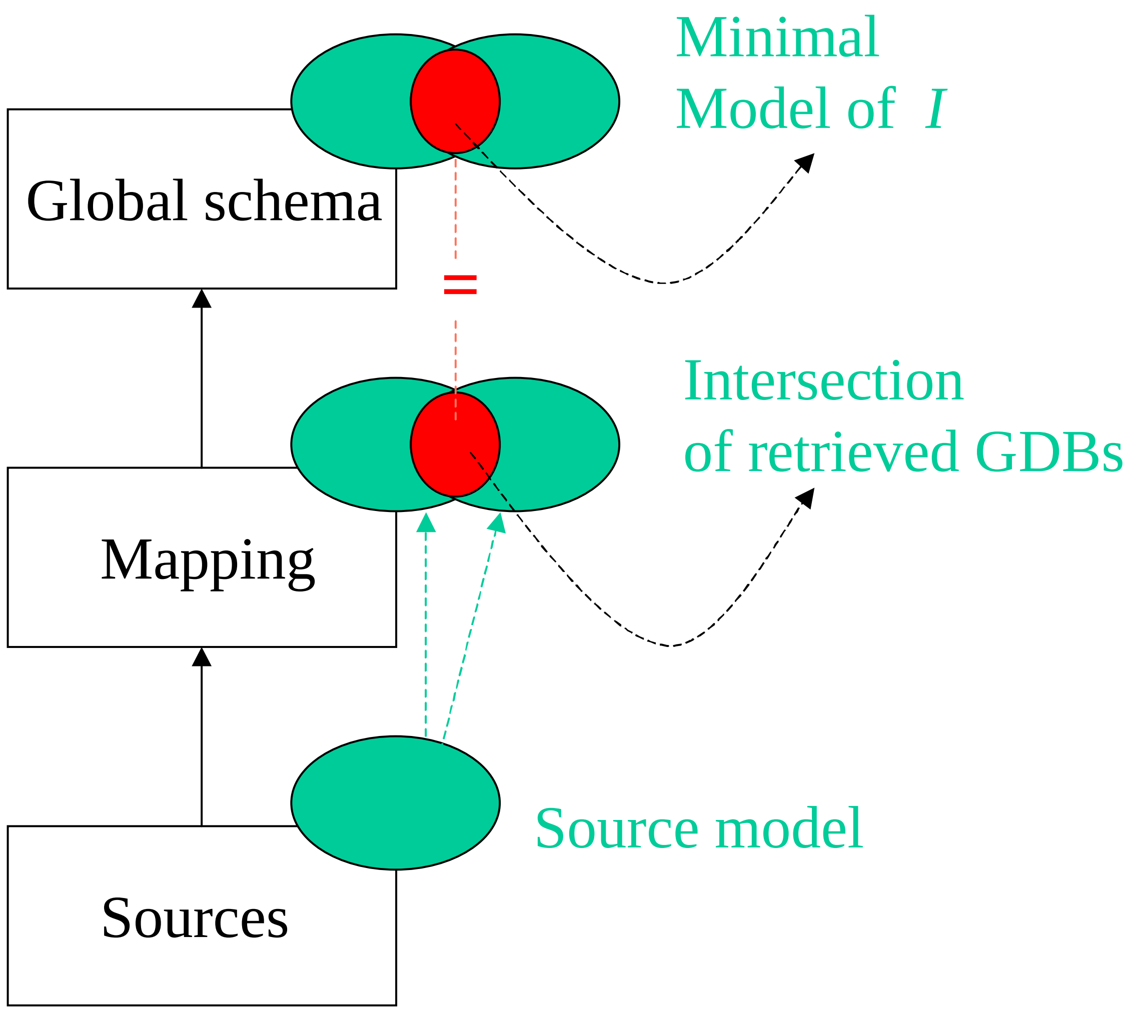
The GAV mapping assertions have the logical form:

$$\forall \vec{\mathbf{x}} \;\; \phi_s(\vec{\mathbf{x}}) \longrightarrow g(\vec{\mathbf{x}})$$

The intersection of all retrieved global databases (which can be computed by letting each element g of $\mathcal{G}$ satisfy exactly the tuples of $\mathcal{C}$ satisfiying the $\phi_{\mathcal{S}}$ that $\mathcal{M}$ associates to g) still satisfies $\mathcal{M}$ wrt $\mathcal{C}$, and therefore, is the **only** "minimal" model of $\mathcal{I}$.

Incompleteness is of special form. For queries without negation, unfolding is sufficient.

INT[noconstr, GAV/sound]



Incompleteness and inconsistency

Constraints in $\mathcal{G}$	Type of mapping	Incomple-teness	Inconsi-stency
no	GAV/exact	no	no
no	GAV/sound	**yes**/no	no
no	*LAV/sound*	**yes**	no
no	LAV/exact	**yes**	**yes**
yes	GAV/exact	no	**yes**
yes	GAV/sound	**yes**	**yes**
yes	LAV/sound	**yes**	**yes**
yes	LAV/exact	**yes**	**yes**

INT[noconstr, LAV/sound]: incompleteness

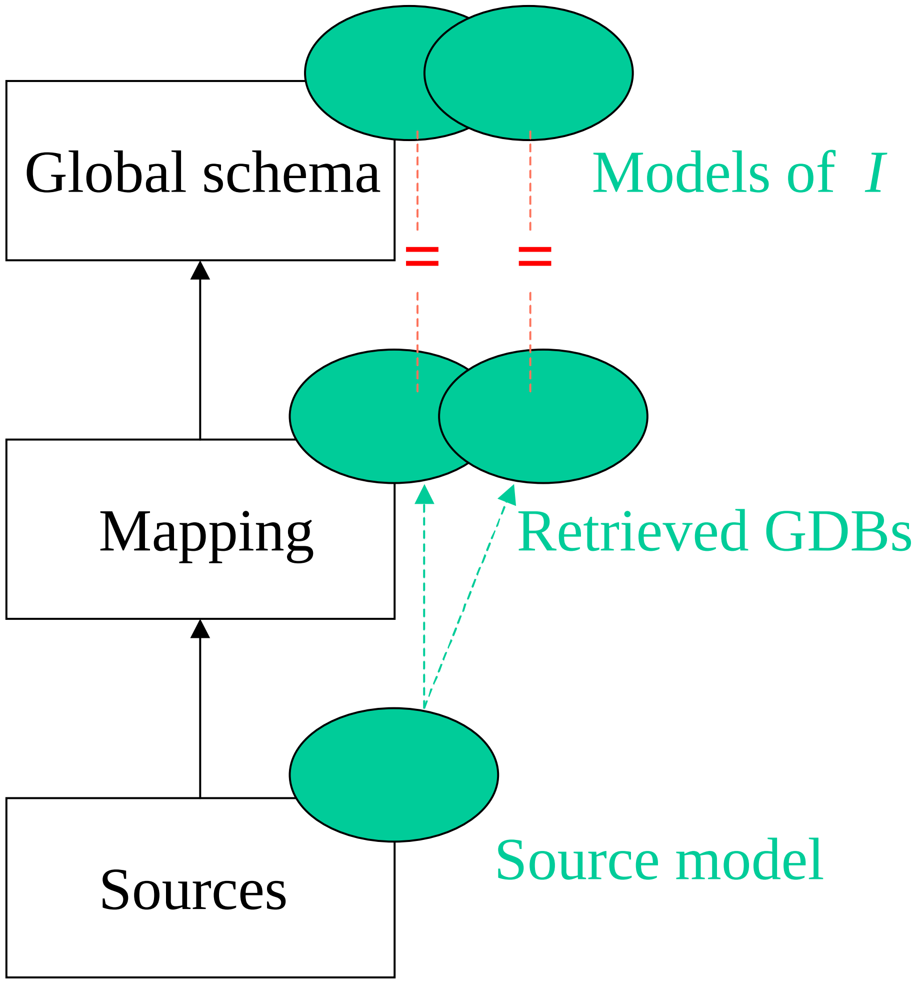
The LAV mapping assertions have the logical form:

$$\forall \vec{\mathbf{x}}\ \ s(\vec{\mathbf{x}}) \rightarrow \phi_{\mathcal{G}}(\vec{\mathbf{x}})$$

In general, given a source database $\mathcal{C}$ there are several solutions of the above assertions (i.e., different databases that are legal wrt $\mathcal{G}$ that satisfies $\mathcal{M}$ wrt $\mathcal{C}$). Incompleteness comes from the mapping. This holds even for the case of simple queries $\phi_{\mathcal{G}}$:

$$\begin{array}{rcl} s_1(x) & \subseteq & \{\ (x) \mid \exists y\ g(x,y)\ \} \\ s_2(x) & \subseteq & \{\ (x) \mid g_1(x) \vee g_2(x)\ \} \end{array}$$

INT[noconstr, LAV/sound]



INT[noconstr, LAV/sound]: dealing with incompleteness

View-based query processing: Answer a query based on a set of materialized views, rather than on the raw data in the database.

Relevant problem in

- Data warehousing
- Query optimization
- Providing physical independence

INT[noconstr, LAV/sound]: dealing with incompleteness

In LAV/sound data integration, **the views are the sources**. Two approaches to view-based query processing:

- **View-based query rewriting**: query processing is divided in two steps
  1. re-express the query in terms of a **given query language** over the alphabet of $\mathcal{A}_{\mathcal{S}}$
  2. evaluate the rewriting over the source database $\mathcal{C}$
- **View-based query answering**: no limitation is posed on how queries are processed, and the only goal is to exploit all possible information, in particular the source database, to compute the certain answers to the query

INT[noconstr, LAV/sound]: connection to query containment

- If queries in $\mathcal{M}$ are conjunctive queries, then we can substitute the query that $\mathcal{M}$ associates to s for every s-literal in $q_{\mathcal{C}}$, and therefore, **checking certain answers can be reduced to checking pure containment (without constraints)** of two queries in the alphabet $\mathcal{A}_{\mathcal{G}}$

- The **data complexity** is at most the complexity of query containment

INT[noconstr, LAV/sound]: some results for query answering

- Conjunctive queries using conjunctive views [Levy&al. PODS'95]
- Recursive queries (datalog programs) using conjunctive views [Duschka&Genesereth PODS'97], [Afrati&al. ICDT'99]
- Complexity analysis [Abiteboul&Duschka PODS'98] [Grahne&Mendelzon ICDT'99]
- Variants of Regular Path Queries [Calvanese&al. ICDE'00, PODS'00] [Deutsch&Tannen DBPL'01], [Calvanese&al. DBPL'01]

INT[noconstr, LAV/sound]: data complexity

From [Abiteboul&Duschka PODS'98]:

Sound sources	CQ	$CQ^{\neq}$	PQ	datalog	FOL
CQ	*PTIME*	*coNP*	*PTIME*	*PTIME*	*undec.*
$CQ^{\neq}$	*PTIME*	*coNP*	*PTIME*	*PTIME*	*undec.*
PQ	*coNP*	*coNP*	*coNP*	*coNP*	*undec.*
datalog	*coNP*	*undec.*	*coNP*	*undec.*	*undec.*
FOL	*undec.*	*undec.*	*undec.*	*undec.*	*undec.*

INT[noconstr, LAV/sound]: basic technique

Consider conjunctive queries and conjunctive views.

$$\begin{array}{lcl} \mathsf{r}_1(T) & \subseteq & \{\ (T) \mid \mathsf{movie}(T,Y,D) \wedge \mathsf{european}(D)\ \} \\ \mathsf{r}_2(T,V) & \subseteq & \{\ (T,V) \mid \mathsf{movie}(T,Y,D) \wedge \mathsf{review}(T,V)\ \} \end{array}$$

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

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

Answering a query means evaluating a goal wrt to this nonrecursive logic program (PTIME data complexity).

INT[noconstr, LAV/sound]: polynomial intractability

Given a graph $G = (N, E)$, we define $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$, and source database $\mathcal{C}$:

$$\begin{array}{lcl} V_b & \subseteq & R_b \\ V_f & \subseteq & R_f \\ V_t & \subseteq & R_{rg} \vee R_{gr} \vee R_{rb} \vee R_{br} \vee R_{gb} \vee R_{bg} \\ V_b{}^{\mathcal{C}} & = & \{(c, a) \mid a \in N, c \notin N\} \\ V_f{}^{\mathcal{C}} & = & \{(a, d) \mid a \in N, d \notin N\} \\ V_t{}^{\mathcal{C}} & = & \{(a, b), (b, a) \mid (a, b) \in E\} \\ Q & \leftarrow & R_b \cdot M \cdot R_f \end{array}$$

where M describes all mismatched edge pairs (e.g., $R_{rg} \cdot R_{rb}$). If G is 3-colorable, then $\exists \mathcal{B}$ where M (and Q) is empty, i.e. $(c, d) \notin Q^{\mathcal{I},\mathcal{C}}$. If G is not 3-colorable, then M is nonempty $\forall \mathcal{B}$, i.e. $(c, d) \in Q^{\mathcal{I},\mathcal{C}}$.

$\Longrightarrow$ **coNP-hard data complexity** for positive queries and positive views.

INT[noconstr, LAV/sound]: in coNP

Consider the case of Datalog queries and positive views.

- $\vec{\mathbf{t}}$ is not a certain answer to Q wrt $\mathcal{I}$ and $\mathcal{C}$, if and only if there is a database $\mathcal{B}$ for $\mathcal{I}$ such that $\vec{\mathbf{t}} \notin Q^{\mathcal{B}}$, and $\mathcal{B}$ satisfies $\mathcal{M}$ wrt $\mathcal{C}$
- Because of the form of $\mathcal{M}$

$$\forall \vec{\mathbf{x}}\, (s(\vec{\mathbf{x}}) \;\rightarrow\; \exists \vec{\mathbf{y_1}} \alpha_1(\vec{\mathbf{x}}, \vec{\mathbf{y_1}}) \;\vee\; \ldots \;\vee\; \exists \vec{\mathbf{y_h}} \alpha_h(\vec{\mathbf{x}}, \vec{\mathbf{y_h}}))$$

  each tuple in $\mathcal{C}$ forces the existence of k tuples in any database that satisfies $\mathcal{M}$ wrt $\mathcal{C}$, where k is the maximal length of conjuncts in $\mathcal{M}$
- If $\mathcal{C}$ has n tuples, then there is a database $\mathcal{B}' \subseteq \mathcal{B}$ for $\mathcal{I}$ that satisfies $\mathcal{M}$ wrt $\mathcal{C}$ with at most $n \cdot k$ tuples. Since Q is monotone, $\vec{\mathbf{t}} \notin Q^{\mathcal{B}'}$.
- Checking whether $\mathcal{B}'$ satisfies $\mathcal{M}$ wrt $\mathcal{C}$ can be done in PTIME wrt the size of $\mathcal{B}'$.

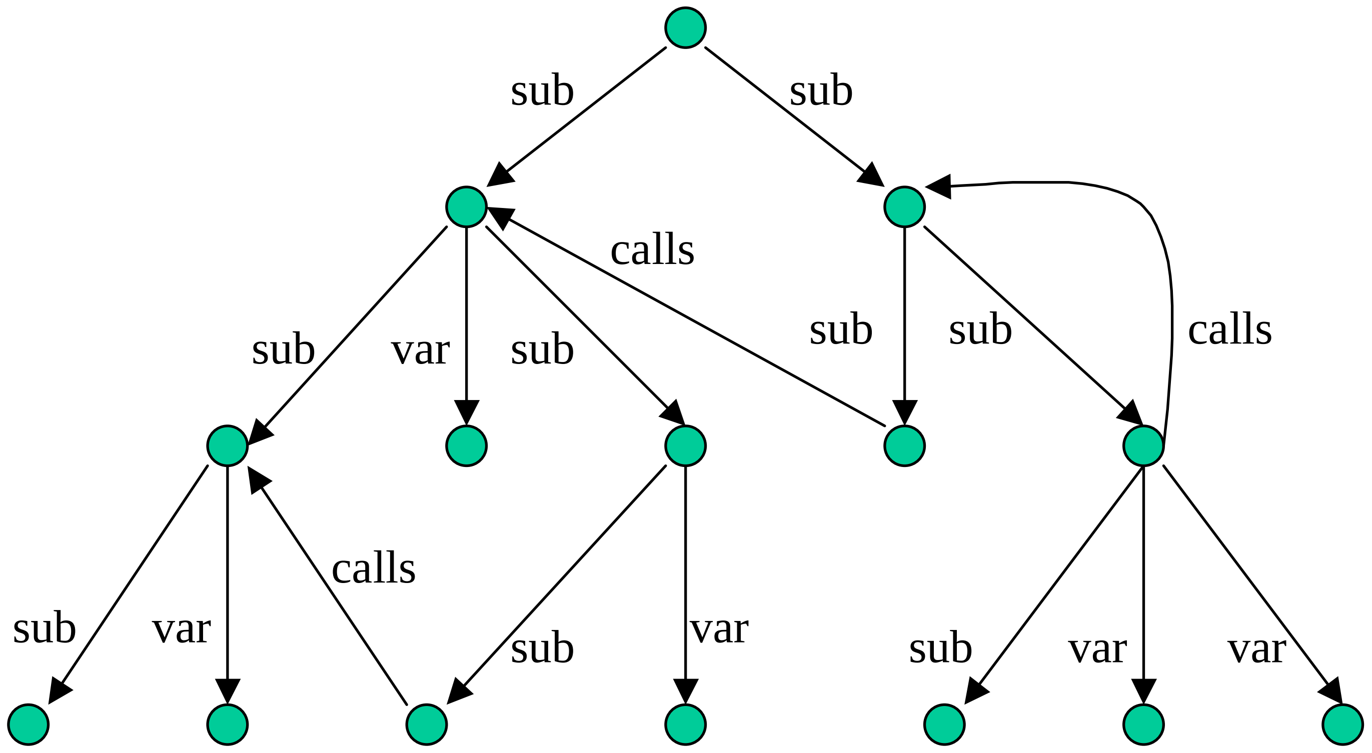
$\Longrightarrow$ **coNP data complexity** for Datalog queries and positive views.

INT[noconstr, LAV/sound]: the case of RPQ

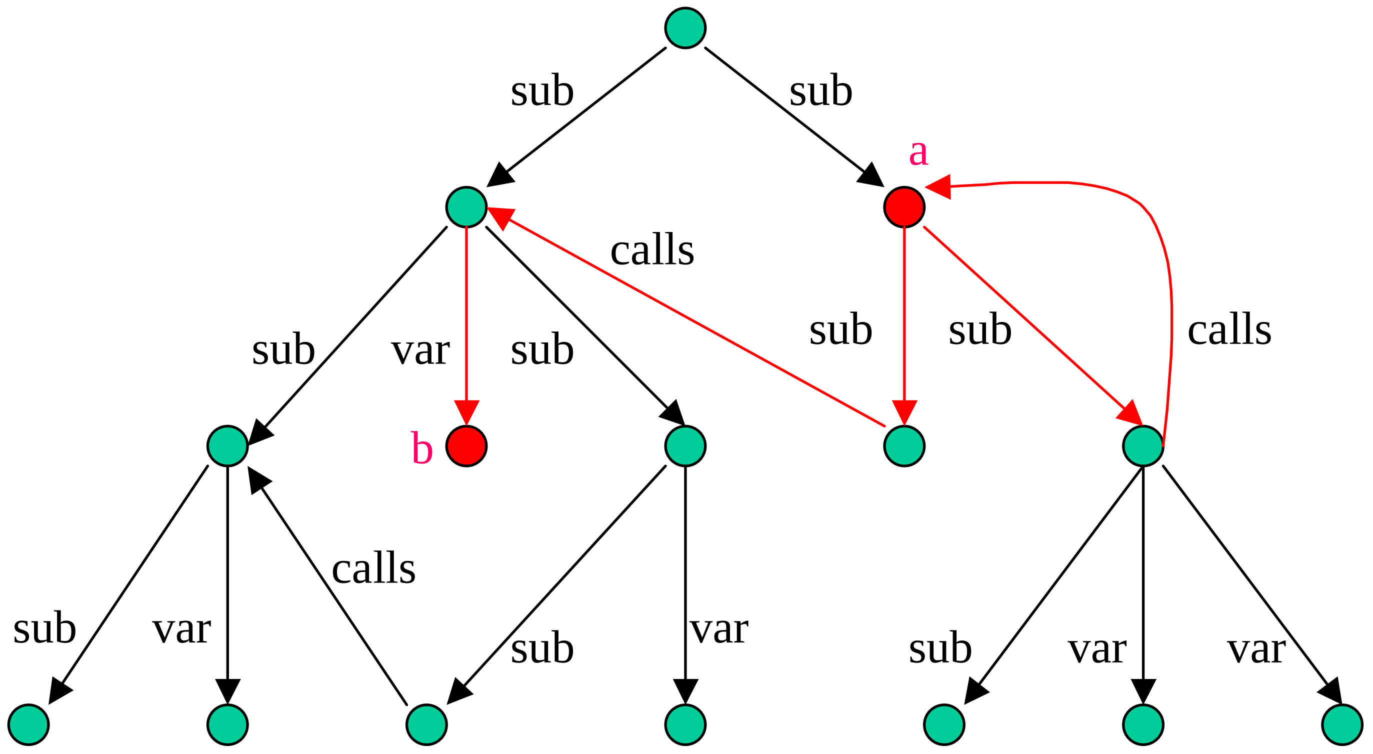
We deal with the problem of answering queries to data integration systems of the form $\langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$, where

- $\mathcal{G}$ simply fixes the labels (alphabet Σ) of a semi-structured database
- the sources in $\mathcal{S}$ are relational
- the mapping $\mathcal{M}$ is of type LAV
- queries are typical of semi-structured data (variants of regular path queries)

Global semi-structured database

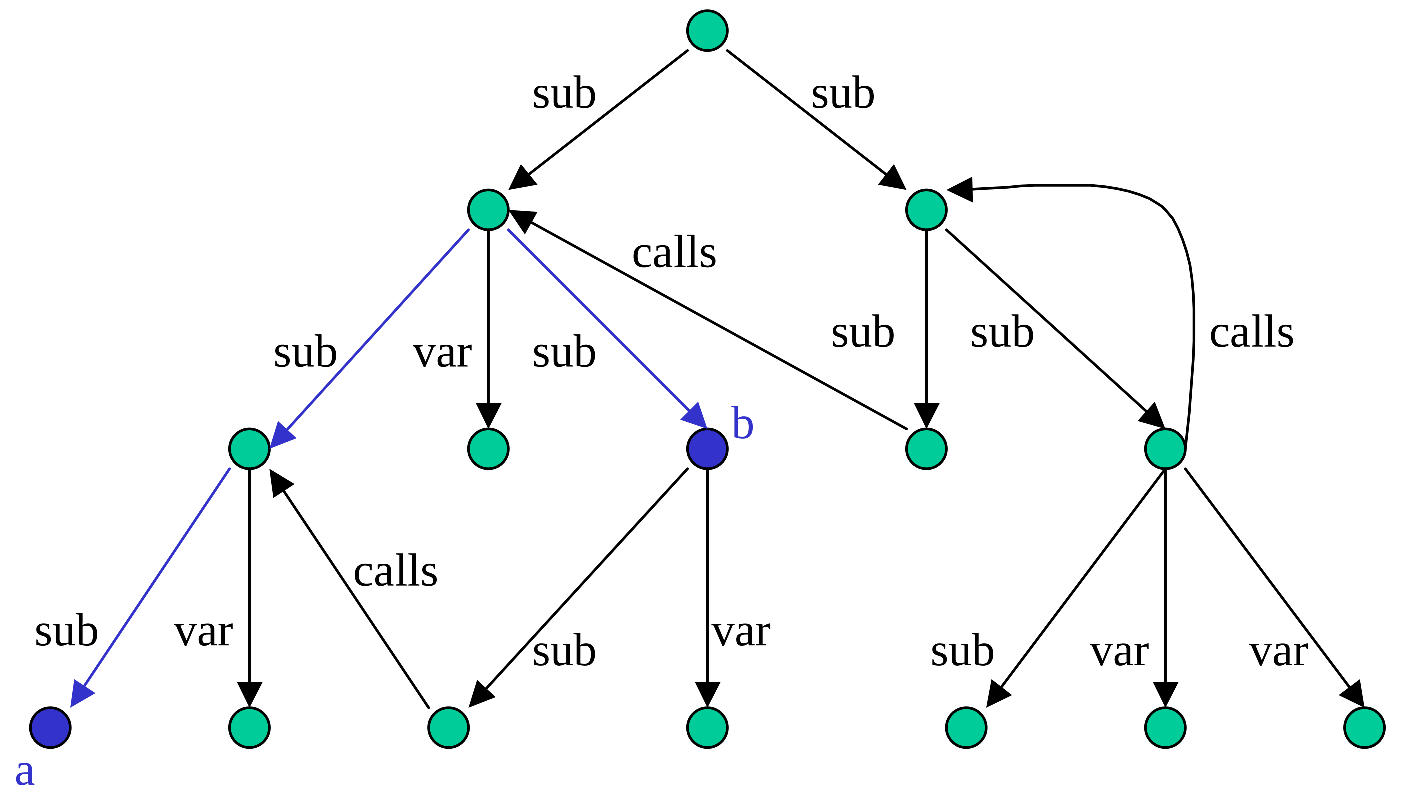


Global semi-structured databases and queries



Regular Path Query (RPQ): $(sub)^* \cdot (sub \cdot (calls \cup sub))^* \cdot var$

Global semi-structured databases and queries



2RPQ: $(sub^{-})^{*} \cdot (var \cup sub)$

INT[noconstr, LAV/sound]: the case of RPQ

Given

- $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$, where
  - $\mathcal{G}$ simply fixes the labels (alphabet Σ) of a semi-structured database
  - the sources in $\mathcal{S}$ are binary relations
  - the mapping $\mathcal{M}$ is of type LAV, and associates to each source s a 2RPQ w over Σ

$$\forall x, y \;\; s(x, y) \subseteq x \xrightarrow{w} y$$

- a source database $\mathcal{C}$
- a 2RPQ Q over Σ
- a pair of objects $\vec{t}$

we want to determine whether $\vec{t} \in Q^{\mathcal{I},\mathcal{C}}$.

Query answering: Technique

- We search for a **counterexample** to $\vec{t} \in Q^{\mathcal{I},\mathcal{C}}$, i.e., a database $\mathcal{B}$ legal for $\mathcal{I}$ wrt $\mathcal{C}$ such that $\vec{t} \notin Q^{\mathcal{B}}$
- **Crucial point**: it is sufficient to restrict our attention to **canonical** databases, i.e., databases $\mathcal{B}$ that can be represented by a word $w_{\mathcal{B}}$

$$\$\,\mathbf{d_1\,w_1\,d_2}\,\$\,\mathbf{d_3\,w_2\,d_4}\,\$\,\cdots\,\$\,\mathbf{d_{2m-1}\,w_m\,d_{2m}}\,\$$$

where $d_1, \ldots, d_{2m}$ are constants in $\mathcal{C}$, $w_i \in \Sigma^+$, and \$ acts as a separator

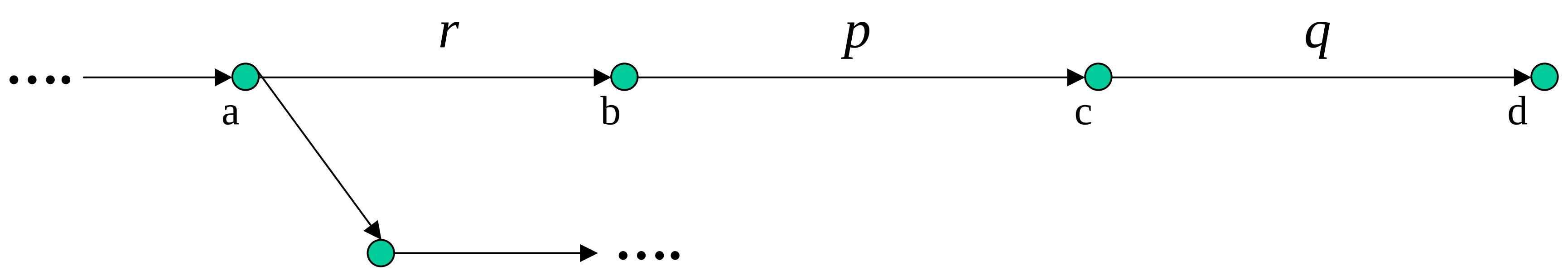
$\Rightarrow$ **Use word-automata theoretic techniques!**

We need techniques for ...

checking whether a pair of objects satisfies a 2RPQ query in the case of

- a word representing a path
- a word representing semipath
- a word representing a canonical database

Finite-state automata and RPQs



$$Q = r \cdot (p \cup q) \cdot q \cdot q^*$$

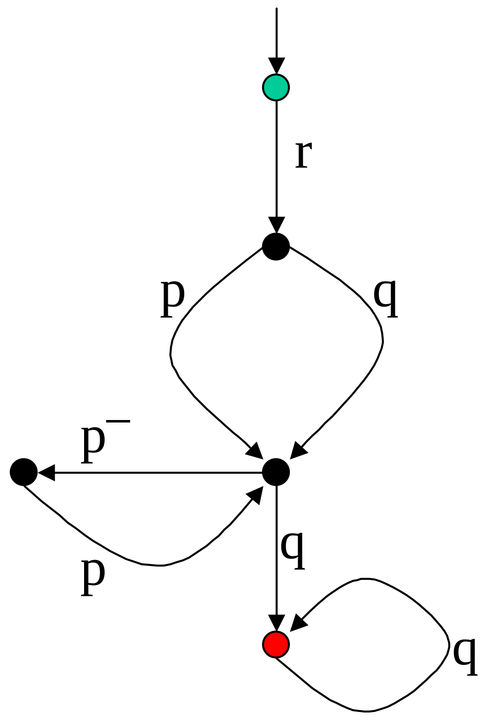
Automaton for Q $\begin{cases} s_1 \in \delta(s_0, r),\ s_2 \in \delta(s_1, p),\ s_2 \in \delta(s_1, q), \\ s_3 \in \delta(s_2, q),\ s_3 \in \delta(s_3, q) \end{cases}$

The computation for RPQs is captured by finite-state automata.

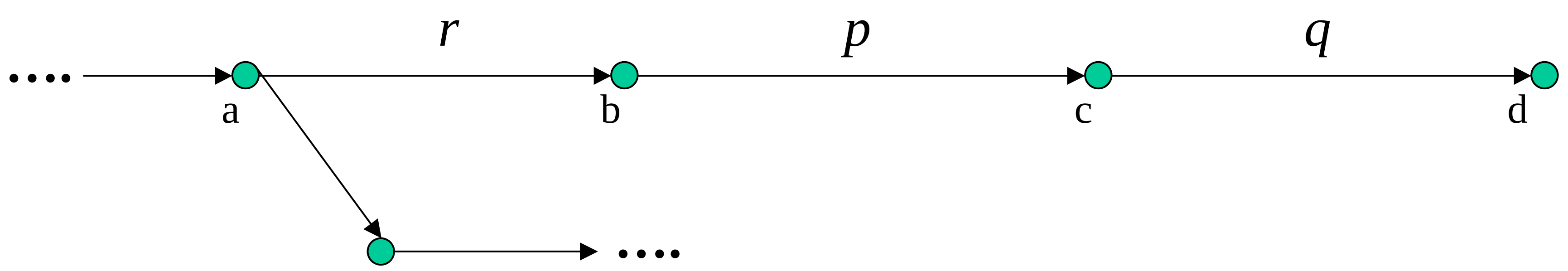
2way Regular Path Queries

2way Regular Path Queries (2RPQ) are expressed by means of finite-state automata over $\Sigma' \cup \{p^- \mid p \in \Sigma'\}$.

$$r \cdot (p \cup q) \cdot (p^- \cdot p)^* \cdot q \cdot q^*$$



Finite-state automata and 2RPQs



Word: $r\ p\ q$

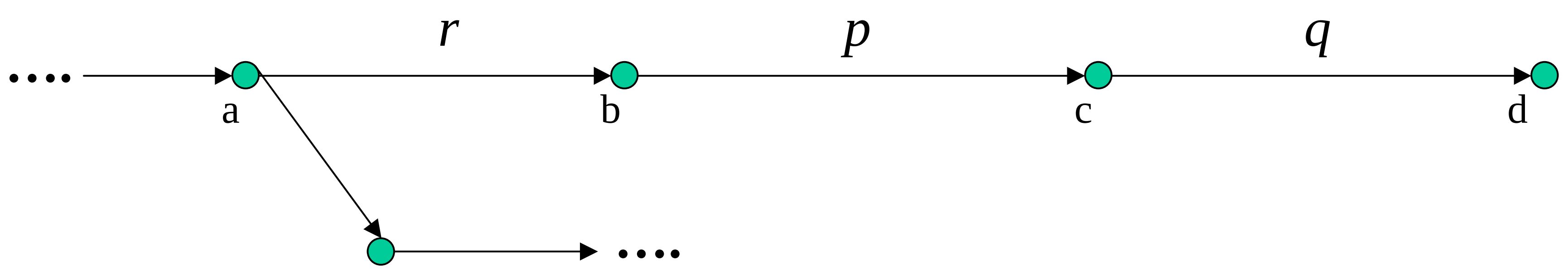
Query: $Q = r \cdot (p \cup q) \cdot p^{-} \cdot p \cdot q \cdot q^{*}$

Automaton for Q $\left\{ \begin{array}{l} s_1 \in \delta(s_0, r),\ s_2 \in \delta(s_1, p),\ s_2 \in \delta(s_1, q), \\ s_3 \in \delta(s_2, p^{-}),\ s_4 \in \delta(s_3, p),\ s_5 \in \delta(s_4, q),\ s_5 \in \delta(s_5, q) \end{array} \right.$

State: s_0

Transition: $s_1 \in \delta(s_0, r)$

Finite-state automata and 2RPQs



Word: $r\ p\ q$

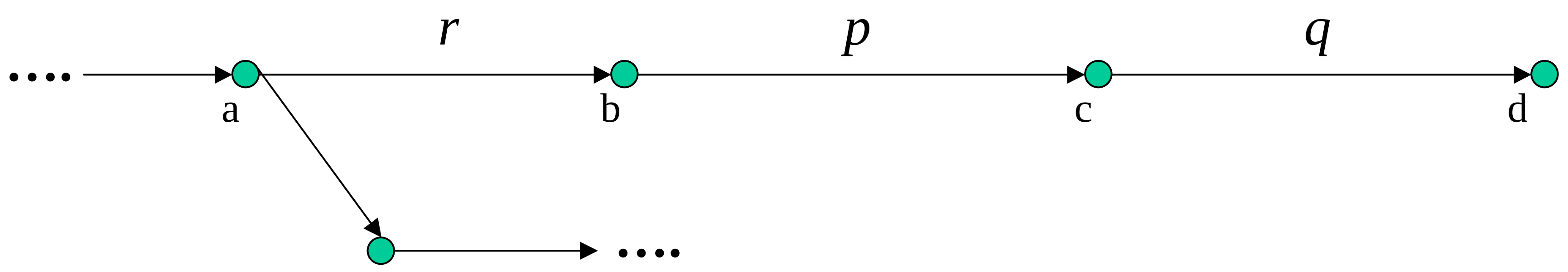
Query: $Q = r \cdot (p \cup q) \cdot p^{-} \cdot p \cdot q \cdot q^{*}$

Automaton for Q $\left\{ \begin{array}{l} s_1 \in \delta(s_0, r),\ s_2 \in \delta(s_1, p),\ s_2 \in \delta(s_1, q), \\ s_3 \in \delta(s_2, p^{-}),\ s_4 \in \delta(s_3, p),\ s_5 \in \delta(s_4, q),\ s_5 \in \delta(s_5, q) \end{array} \right.$

State: s_1

Transition: $s_2 \in \delta(s_1, p)$

Finite-state automata and 2RPQs



Word: $r\ p\ q$

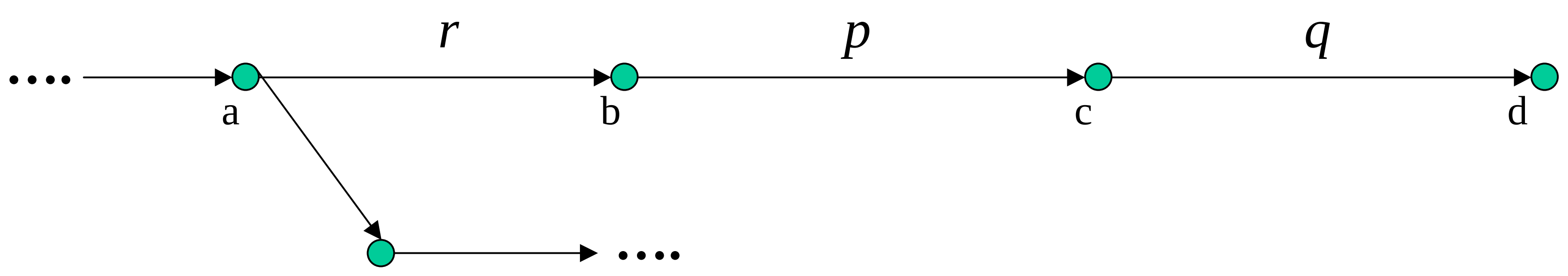
Query: $Q = r \cdot (p \cup q) \cdot p^{-} \cdot p \cdot q \cdot q^{*}$

Automaton for Q $\left\{ \begin{array}{l} s_1 \in \delta(s_0, r),\ s_2 \in \delta(s_1, p),\ s_2 \in \delta(s_1, q), \\ s_3 \in \delta(s_2, p^{-}),\ s_4 \in \delta(s_3, p),\ s_5 \in \delta(s_4, q),\ s_5 \in \delta(s_5, q) \end{array} \right.$

State: s_2

Transition: *none*

Finite-state automata and 2RPQs



Word: $r\ p\ q$

Query: $Q = r \cdot (p \cup q) \cdot p^{-} \cdot p \cdot q \cdot q^{*}$

State: s_2

Transition: *none*

(a, d) satisfies query Q, but the path from a to d is not accepted by the 1NFA corresponding to Q: **the computation for 2RPQs is not captured by finite-state automata**.

2way automata (2NFA)

A ***2way automaton*** $A = (\Gamma, S, S_0, \rho, F)$ consists of an alphabet Γ, a finite set of states S, a set of initial states $S_0 \subseteq S$, a transition function

$$\rho : S \times \Sigma \rightarrow 2^{S \times \{-1,0,1\}}$$

and a set of accepting states $F \subseteq S$.

Given a 2way automaton A with n states, one can construct a one-way automaton B_1 with $O(2^{n \log n})$ states such that $L(B_1) = L(A)$, and a one-way automaton B_2 with $O(2^n)$ states such that $L(B_2) = \Gamma^* - L(A)$.

2way automata and 2RPQs

Given a 2RPQ $E = (\Sigma, S, I, \delta, F)$ over the alphabet Σ, the corresponding 2way automaton A_E is:

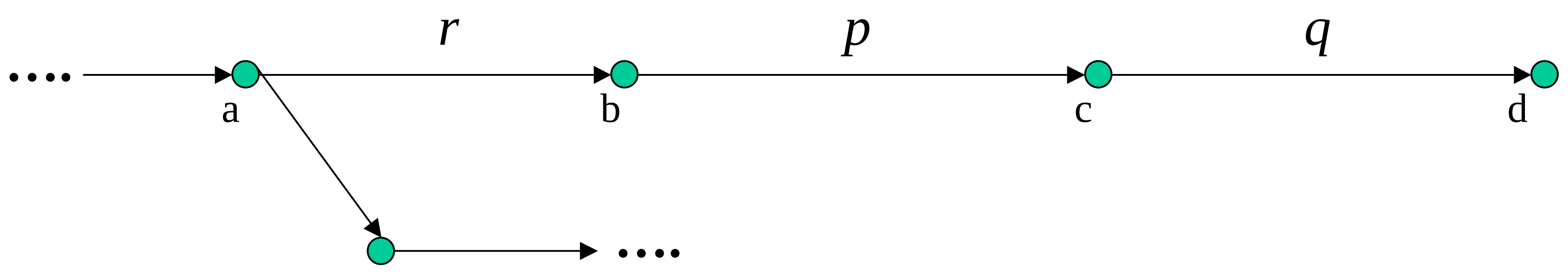
$$(\Sigma_A = \Sigma \cup \{\$\}, S_A = S \cup \{s_f\} \cup \{s^{\leftarrow} \mid s \in S\}, I, \delta_A, \{s_f\})$$

where δ_A is defined as follows:

- $(s_2, 1) \in \delta_A(s_1, r)$, for each transition $s_2 \in \delta(s_1, r)$ of E
- enter backward mode: $(s^{\leftarrow}, -1) \in \delta_A(s, \ell)$, for each $s \in S$ and $\ell \in \Sigma_A$
- exit backward mode: $(s_2, 0) \in \delta_A(s_1^{\leftarrow}, r)$, for each $s_2 \in \delta(s_1, r^-)$ of E
- $(s_f, 1) \in \delta_A(s, \$)$, for each $s \in F$.

$\Longrightarrow$ **w satisfies E iff $w\$ \in L(A_E)$.**

2way automata and 2RPQs



$$Q = r \cdot (p \cup q) \cdot p^{-} \cdot p \cdot q \cdot q^{*}$$

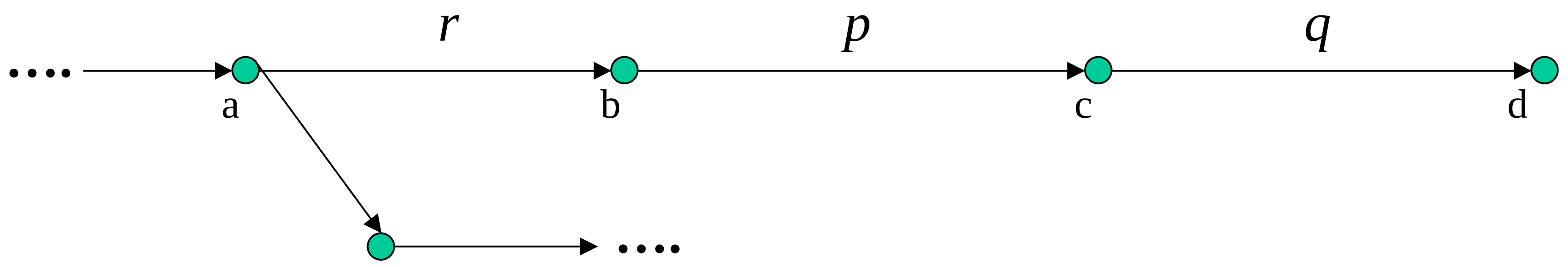
Automaton for Q

$$\begin{cases} s_1 \in \delta(s_0, r),\ s_2 \in \delta(s_1, p),\ s_2 \in \delta(s_1, q), \\ \mathbf{s_3 \in \delta(s_2, p^{-})},\ s_4 \in \delta(s_3, p),\ s_5 \in \delta(s_4, q),\ s_5 \in \delta(s_5, q) \end{cases}$$

2way automaton

$$\begin{cases} (s_1, 1) \in \delta_A(s_0, r),\ (s_2, 1) \in \delta_A(s_1, p),\ (s_2, 1) \in \delta_A(s_1, q), \\ \mathbf{(s_2^{\leftarrow}, -1) \in \delta_A(s_2, q)},\ \mathbf{(s_3, 0) \in \delta_A(s_2^{\leftarrow}, p)}, \\ (s_4, 1) \in \delta_A(s_3, p),\ (s_5, 1) \in \delta_A(s_4, q),\ (s_f, 1) \in \delta_A(s_5, \$) \end{cases}$$

2NFA and 2RPQs



Word: $r\ p\ q\ \$$

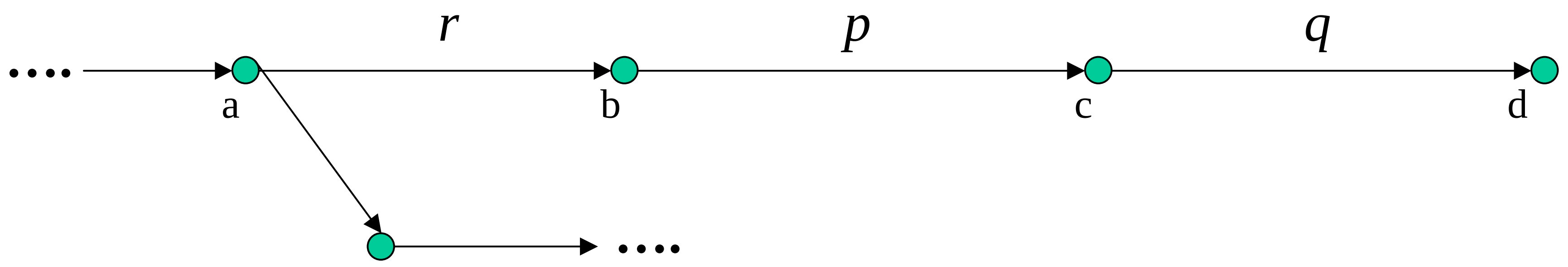
Query: $Q = r \cdot (p \cup q) \cdot p^{-} \cdot p \cdot q \cdot q^{*}$

Automaton for Q $\left\{\begin{array}{l} (s_1, 1) \in \delta_A(s_0, r),\ (s_2, 1) \in \delta_A(s_1, p), \\ (s_2^{\leftarrow}, -1) \in \delta_A(s_2, q),\ (s_3, 0) \in \delta_A(s_2^{\leftarrow}, p), \\ (s_4, 1) \in \delta_A(s_3, p),\ (s_5, 1) \in \delta_A(s_4, q),\ (s_f, 1) \in \delta_A(s_5, \$) \end{array}\right.$

State: s_0

Transition: $(s_1, 1) \in \delta_A(s_0, r)$

2NFA and 2RPQs



Word: $r\ p\ q\ \$$

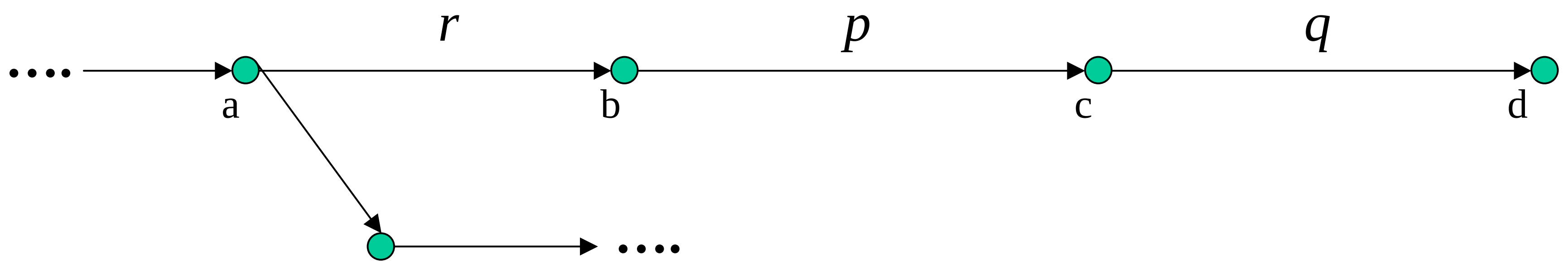
Query: $Q = r \cdot (p \cup q) \cdot p^{-} \cdot p \cdot q \cdot q^{*}$

Automaton for Q
$$\left\{\begin{array}{l}(s_1,1) \in \delta_A(s_0,r),\ (s_2,1) \in \delta_A(s_1,p),\\ (s_2^{\leftarrow},-1) \in \delta_A(s_2,q),\ (s_3,0) \in \delta_A(s_2^{\leftarrow},p),\\ (s_4,1) \in \delta_A(s_3,p),\ (s_5,1) \in \delta_A(s_4,q),\ (s_f,1) \in \delta_A(s_5,\$)\end{array}\right.$$

State: s_1

Transition: $(s_2,1) \in \delta_A(s_1,p)$

2NFA and 2RPQs



Word: $r\ p\ q\ \$$

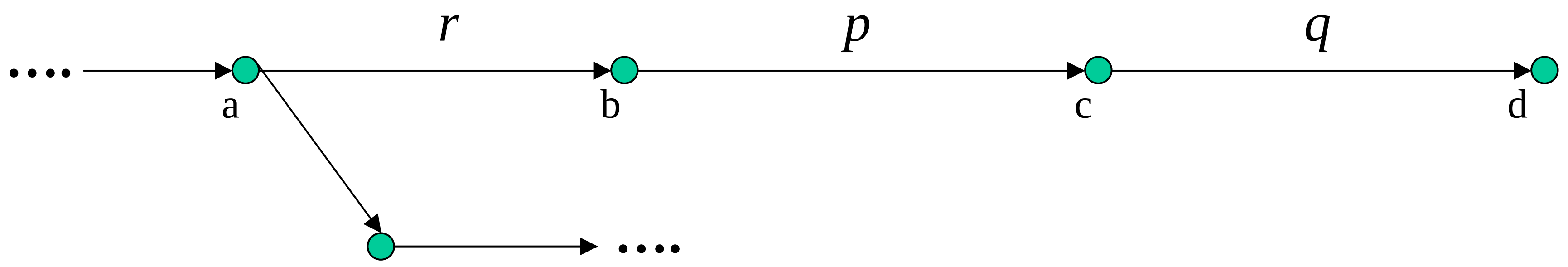
Query: $Q = r \cdot (p \cup q) \cdot p^{-} \cdot p \cdot q \cdot q^{*}$

Automaton for Q
$$\begin{cases} (s_1,1) \in \delta_A(s_0,r),\ (s_2,1) \in \delta_A(s_1,p), \\ (s_2^{\leftarrow},-1) \in \delta_A(s_2,q),\ (s_3,0) \in \delta_A(s_2^{\leftarrow},p), \\ (s_4,1) \in \delta_A(s_3,p),\ (s_5,1) \in \delta_A(s_4,q),\ (s_f,1) \in \delta_A(s_5,\$) \end{cases}$$

State: s_2

Transition: $(s_2^{\leftarrow},-1) \in \delta_A(s_2,q)$

2NFA and 2RPQs



Word: $r\ p\ q\ \$$

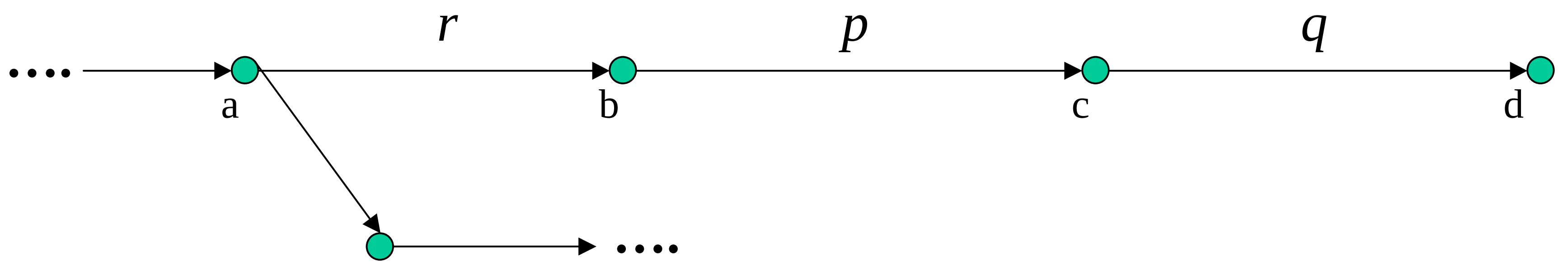
Query: $Q = r \cdot (p \cup q) \cdot p^{-} \cdot p \cdot q \cdot q^{*}$

Automaton for Q $\left\{\begin{array}{l} (s_1, 1) \in \delta_A(s_0, r),\ (s_2, 1) \in \delta_A(s_1, p), \\ (s_2^{\leftarrow}, -1) \in \delta_A(s_2, q),\ (s_3, 0) \in \delta_A(s_2^{\leftarrow}, p), \\ (s_4, 1) \in \delta_A(s_3, p),\ (s_5, 1) \in \delta_A(s_4, q),\ (s_f, 1) \in \delta_A(s_5, \$) \end{array}\right.$

State: $s_2^{\leftarrow}$

Transition: $(s_3, 0) \in \delta_A(s_2^{\leftarrow}, p)$

2NFA and 2RPQs



Word: $r\ p\ q\ \$$

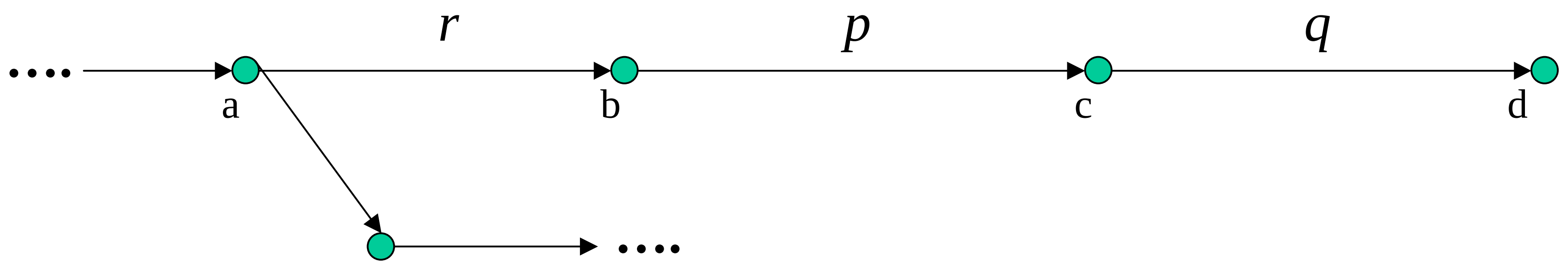
Query: $Q = r \cdot (p \cup q) \cdot p^- \cdot p \cdot q \cdot q^*$

Automaton for Q
$$\left\{\begin{array}{l} (s_1,1) \in \delta_A(s_0,r),\ (s_2,1) \in \delta_A(s_1,p), \\ (s_2^{\leftarrow},-1) \in \delta_A(s_2,q),\ (s_3,0) \in \delta_A(s_2^{\leftarrow},p), \\ (s_4,1) \in \delta_A(s_3,p),\ (s_5,1) \in \delta_A(s_4,q),\ (s_f,1) \in \delta_A(s_5,\$) \end{array}\right.$$

State: s_3

Transition: $(s_4,1) \in \delta_A(s_3,p)$

2NFA and 2RPQs



Word: $r\ p\ q\ \$$

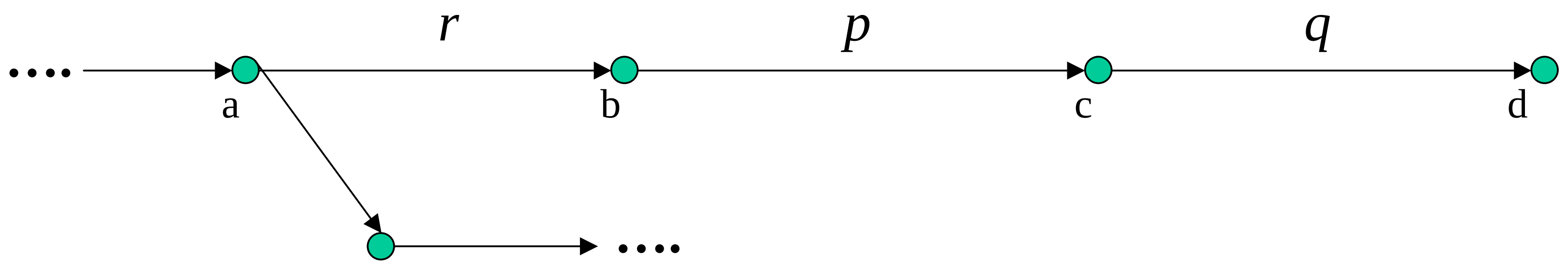
Query: $Q = r \cdot (p \cup q) \cdot p^{-} \cdot p \cdot q \cdot q^{*}$

Automaton for Q $\left\{ \begin{array}{l} (s_1, 1) \in \delta_A(s_0, r),\ (s_2, 1) \in \delta_A(s_1, p), \\ (s_2^{\leftarrow}, -1) \in \delta_A(s_2, q),\ (s_3, 0) \in \delta_A(s_2^{\leftarrow}, p), \\ (s_4, 1) \in \delta_A(s_3, p),\ (s_5, 1) \in \delta_A(s_4, q),\ (s_f, 1) \in \delta_A(s_5, \$) \end{array} \right.$

State: s_4

Transition: $(s_5, 1) \in \delta_A(s_4, q)$

2NFA and 2RPQs



Word: $r\ p\ q\ \$$

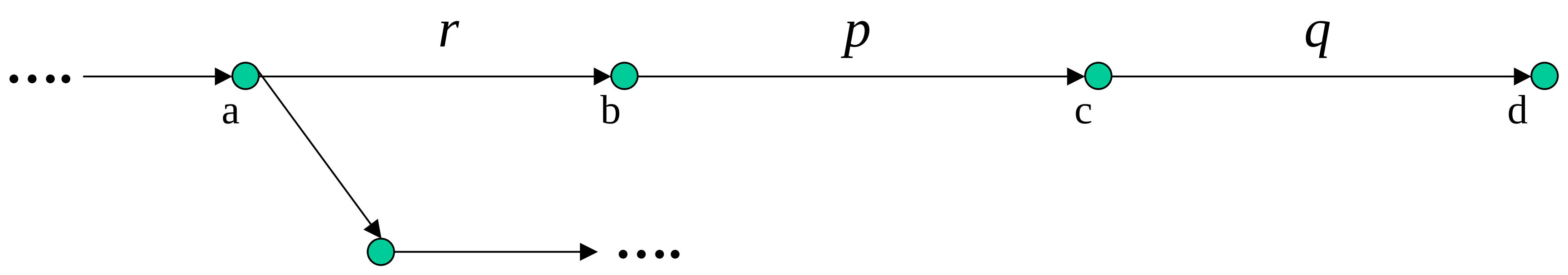
Query: $Q = r \cdot (p \cup q) \cdot p^{-} \cdot p \cdot q \cdot q^{*}$

Automaton for Q
$$\begin{cases} (s_1,1) \in \delta_A(s_0,r),\ (s_2,1) \in \delta_A(s_1,p), \\ (s_2^{\leftarrow},-1) \in \delta_A(s_2,q),\ (s_3,0) \in \delta_A(s_2^{\leftarrow},p), \\ (s_4,1) \in \delta_A(s_3,p),\ (s_5,1) \in \delta_A(s_4,q),\ (s_f,1) \in \delta_A(s_5,\$) \end{cases}$$

State: s_5

Transition: $(s_f,1) \in \delta_A(s_5,\$)$

2NFA and 2RPQs



Word: $r\ p\ q\ \$$

Query: $Q = r \cdot (p \cup q) \cdot p^{-} \cdot p \cdot q \cdot q^{*}$

State: s_f

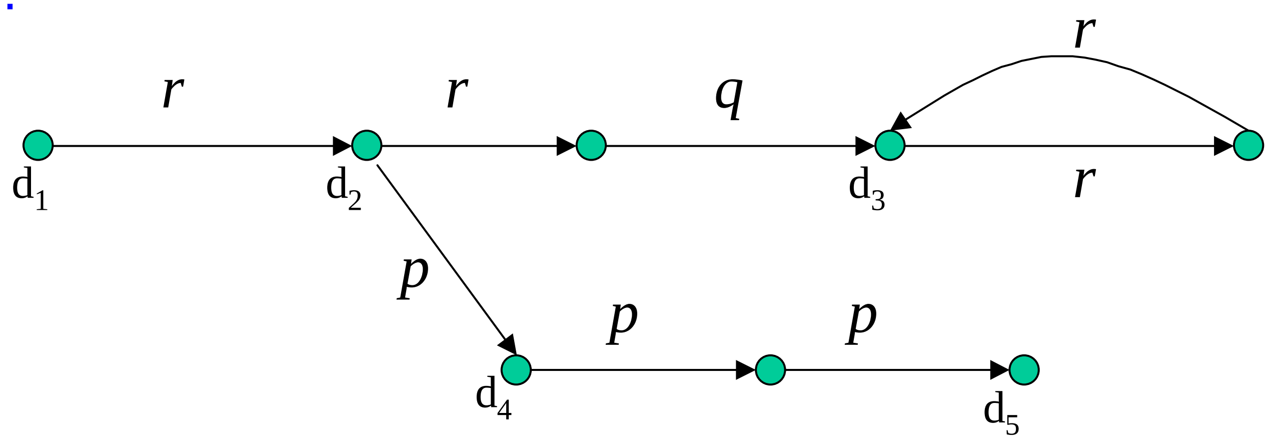
(a, d) satisfies query Q, and the path from a to d is accepted by the 2NFA corresponding to Q: **the computation for 2RPQs is captured by 2way automata**.

2NFA and view extensions

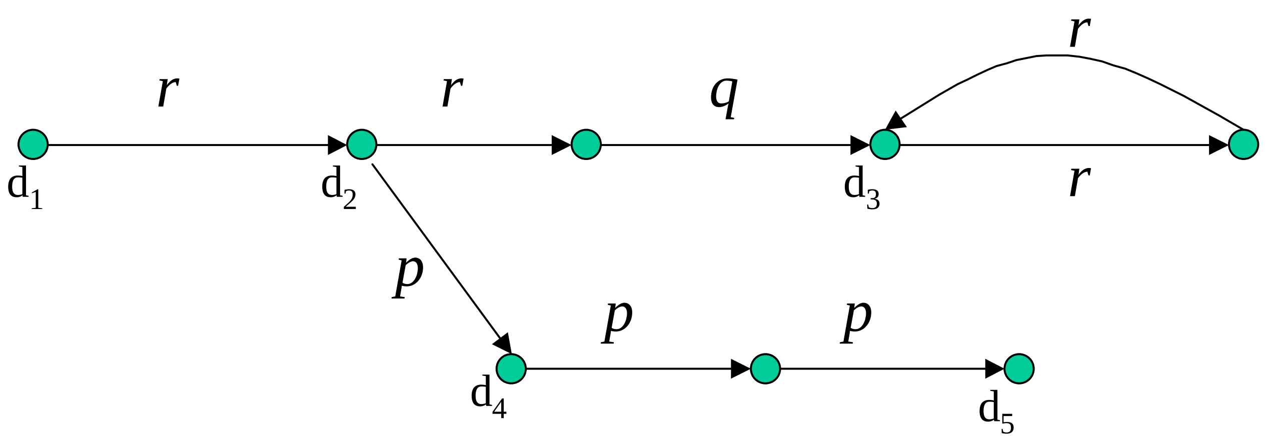
Global schema G: $(r \cup p \cup q \cup r^- \cup p^- \cup q^-)^*$

	$r \cup q^*$	$(p \cup r) \circ p$	$(p^- \cup q)^*$	$r \circ r$	$r \circ (q \cup p)$
Sources:	(d_1,d_2)	(d_4,d_5)	(d_4,d_2)	(d_3,d_3)	(d_2,d_3)

Database for G:



2NFA and view extensions

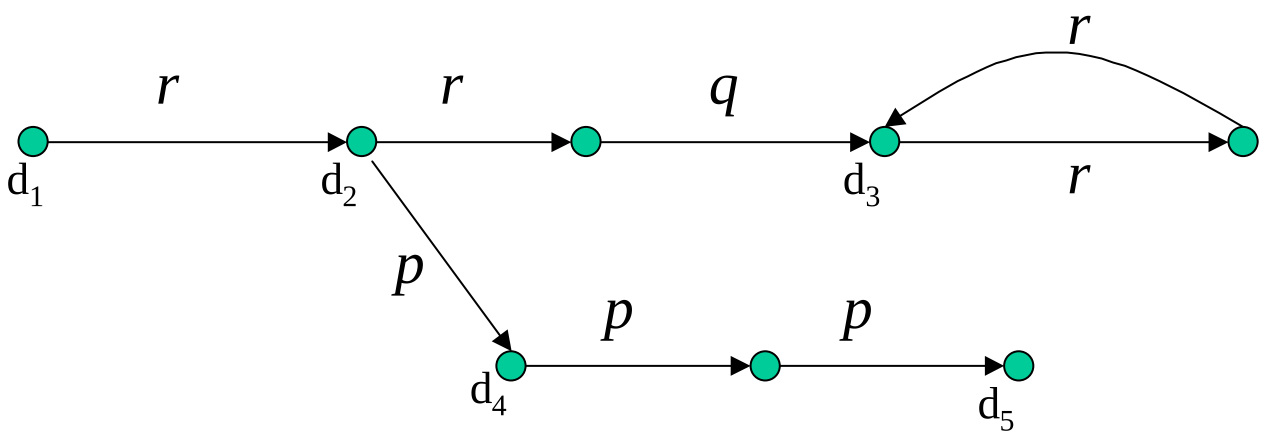


Database $\mathcal{B}$ as a word: $\$ d_4\ p\ p\ d_5\ \$\ d_1\ r\ d_2\ \$\ d_4\ p^-\ d_2\ \$\ d_3\ r\ r\ d_3\ \$\ d_2\ r\ q\ d_3\ \$$

$$Q = r \cdot (p \cup q) \cdot (p^- \cdot r)^* \cdot q \cdot q^*$$

To verify that (d_1, d_3) satisfies Q in the above database $\mathcal{B}$, we build $A_{(Q,d_1,d_3)}$, by exploiting not only the ability of 2way automata to move on the word both forward and backward, but also the ability to **jump** from one position in the word representing a node to any other position (either preceding or succeeding) representing the same node.

A run of $A_{(Q,d_1,d_3)}$



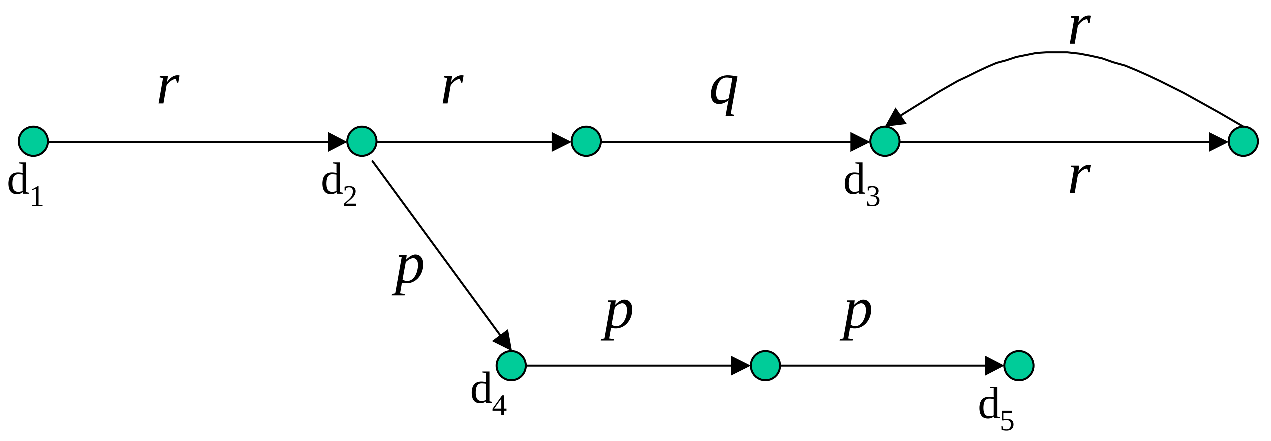
Word: $\$ \; d_4 \; p \; p \; d_5 \; \$ \; d_1 \; r \; d_2 \; \$ \; d_4 \; p^- \; d_2 \; \$ \; d_3 \; r \; r \; d_3 \; \$ \; d_2 \; r \; q \; d_3 \; \$$

$$Q = r \cdot (p \cup q) \cdot (p^- \cdot r)^* \cdot q \cdot q^*$$

State: s_0

Transition: $(s_0, 1) \in \delta_A(s_0, \ell)$, for each $\ell \in \Sigma_A$

A run of $A_{(Q,d_1,d_3)}$



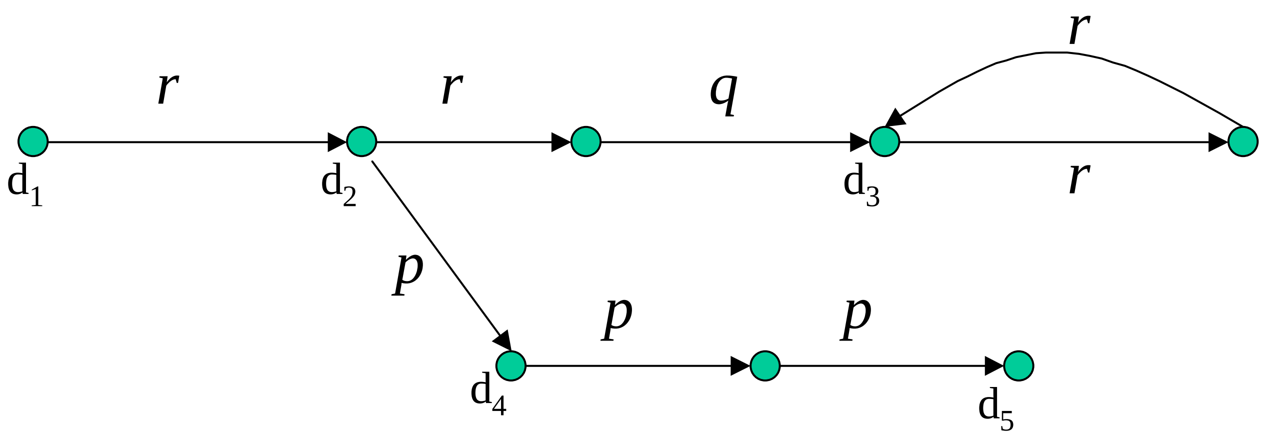
Word: $\$ \; d_4 \; p \; p \; d_5 \; \$ \; d_1 \; r \; d_2 \; \$ \; d_4 \; p^- \; d_2 \; \$ \; d_3 \; r \; r \; d_3 \; \$ \; d_2 \; r \; q \; d_3 \; \$$

$$Q = r \cdot (p \cup q) \cdot (p^- \cdot r)^* \cdot q \cdot q^*$$

State: s_0

Transition: $(s_0, 1) \in \delta_A(s_0, \ell)$, for each $\ell \in \Sigma_A$

A run of $A_{(Q,d_1,d_3)}$



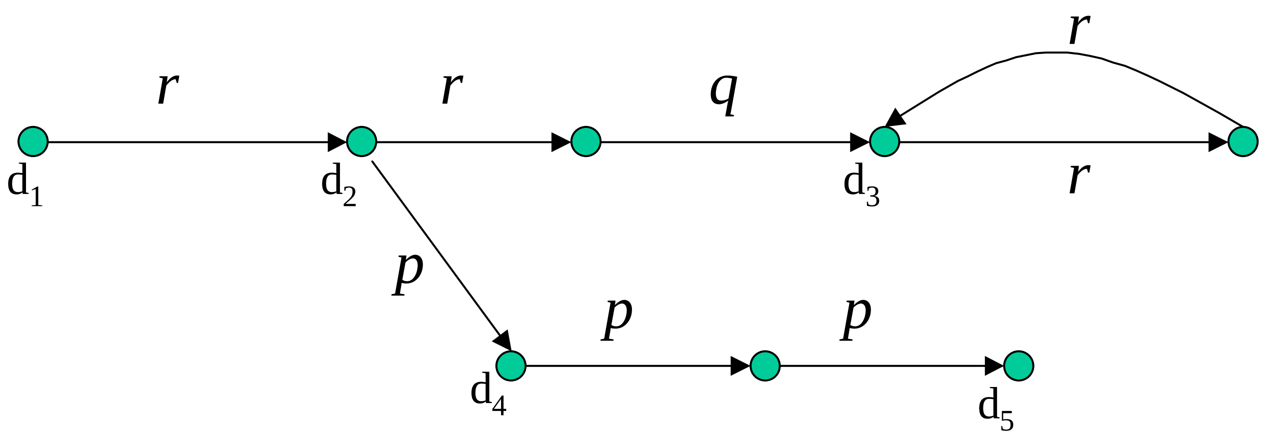
Word: $\$ \; d_4 \; p \; p \; d_5 \; \$ \; d_1 \; r \; d_2 \; \$ \; d_4 \; p^- \; d_2 \; \$ \; d_3 \; r \; r \; d_3 \; \$ \; d_2 \; r \; q \; d_3 \; \$$

$$Q = r \cdot (p \cup q) \cdot (p^- \cdot r)^* \cdot q \cdot q^*$$

State: s_0

Transition: $(s_0, 1) \in \delta_A(s_0, \ell)$, for each $\ell \in \Sigma_A$

A run of $A_{(Q,d_1,d_3)}$



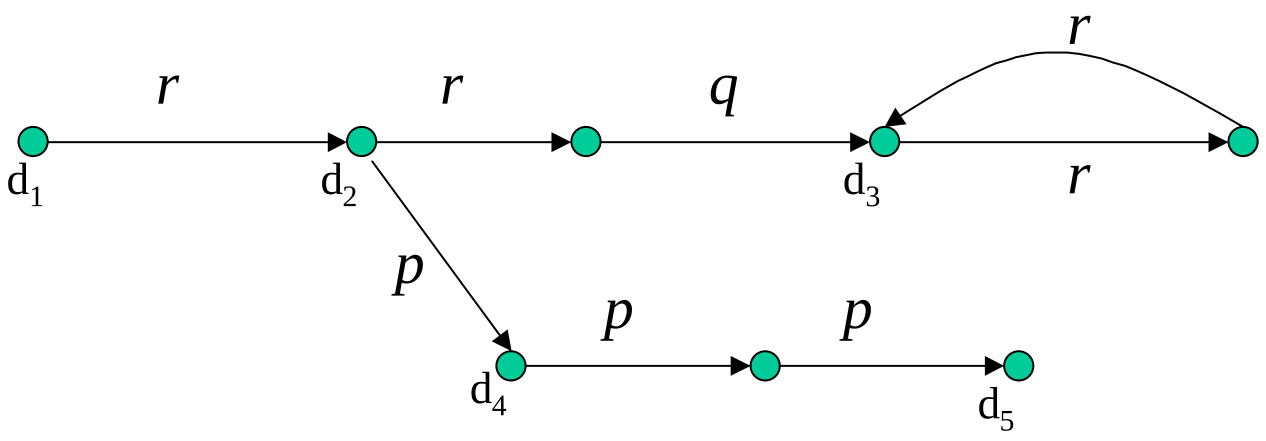
Word: $\$\ d_4\ p\ p\ d_5\ \$\ d_1\ r\ d_2\ \$\ d_4\ p^-\ d_2\ \$\ d_3\ r\ r\ d_3\ \$\ d_2\ r\ q\ d_3\ \$$

$$Q = r \cdot (p \cup q) \cdot (p^- \cdot r)^* \cdot q \cdot q^*$$

State: s_0

Transition: $(s_0, 1) \in \delta_A(s_0, \ell)$, for each $\ell \in \Sigma_A$

A run of $A_{(Q,d_1,d_3)}$



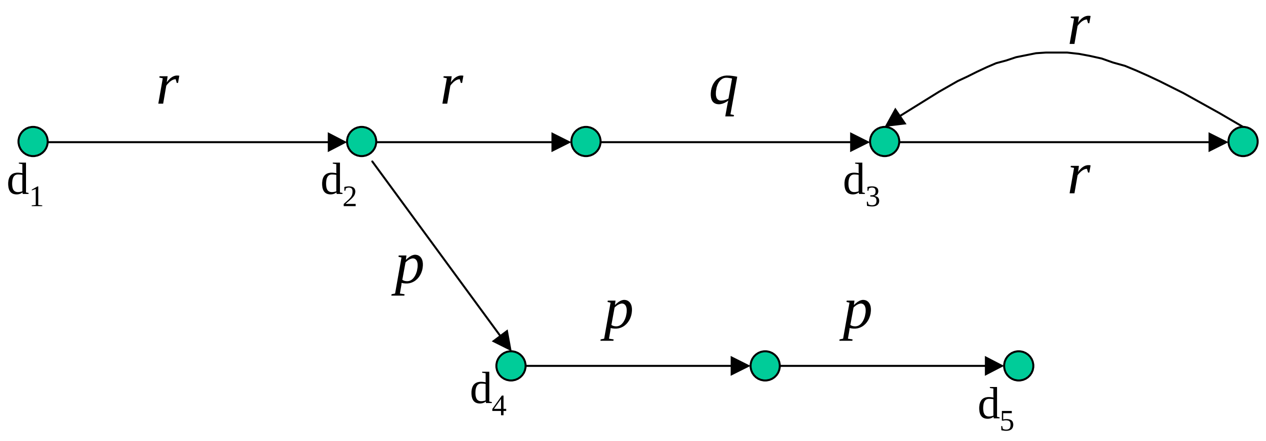
Word: $\$\ d_4\ p\ p\ d_5\ \$\ d_1\ r\ d_2\ \$\ d_4\ p^-\ d_2\ \$\ d_3\ r\ r\ d_3\ \$\ d_2\ r\ q\ d_3\ \$$

$$Q = r \cdot (p \cup q) \cdot (p^- \cdot r)^* \cdot q \cdot q^*$$

State: s_0

Transition: $(s_0, 1) \in \delta_A(s_0, \ell)$, for each $\ell \in \Sigma_A$

A run of $A_{(Q,d_1,d_3)}$



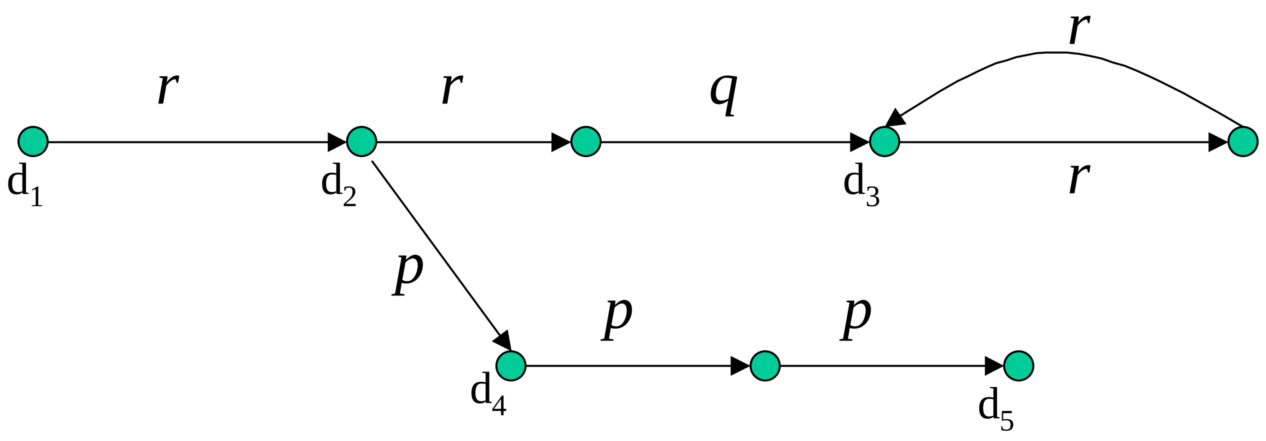
Word: $\$\ d_4\ p\ p\ d_5\ \$\ d_1\ r\ d_2\ \$\ d_4\ p^-\ d_2\ \$\ d_3\ r\ r\ d_3\ \$\ d_2\ r\ q\ d_3\ \$$

$$Q = r \cdot (p \cup q) \cdot (p^- \cdot r)^* \cdot q \cdot q^*$$

State: s_0

Transition: $(s_0, 1) \in \delta_A(s_0, \ell)$, for each $\ell \in \Sigma_A$

A run of $A_{(Q,d_1,d_3)}$



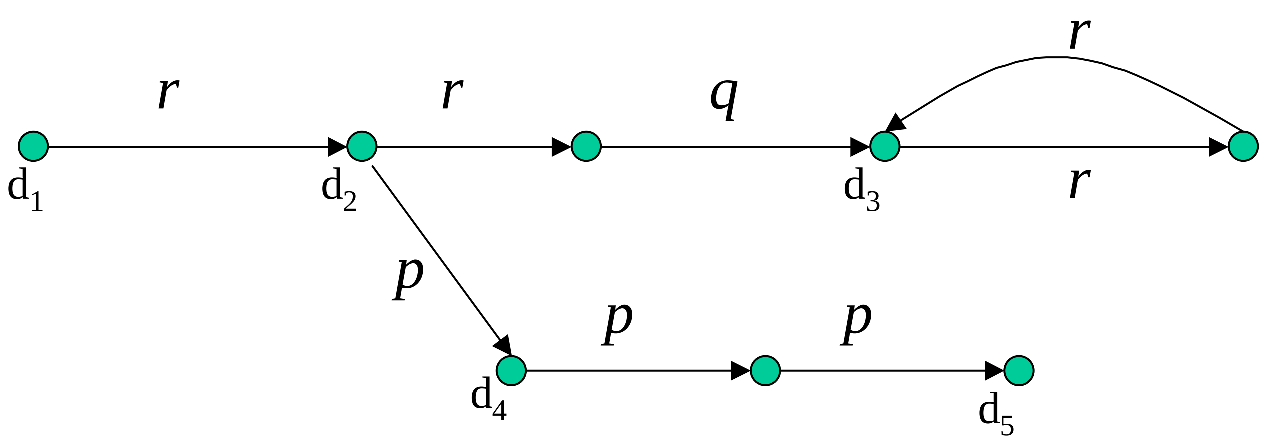
Word: $\$\ d_4\ p\ p\ d_5\ \$\ d_1\ r\ d_2\ \$\ d_4\ p^-\ d_2\ \$\ d_3\ r\ r\ d_3\ \$\ d_2\ r\ q\ d_3\ \$$

$$Q = r \cdot (p \cup q) \cdot (p^- \cdot r)^* \cdot q \cdot q^*$$

State: s_0

Transition: $(s_1, 0) \in \delta_A(s_0, d_1)$, s_1 initial state for Q

A run of $A_{(Q,d_1,d_3)}$



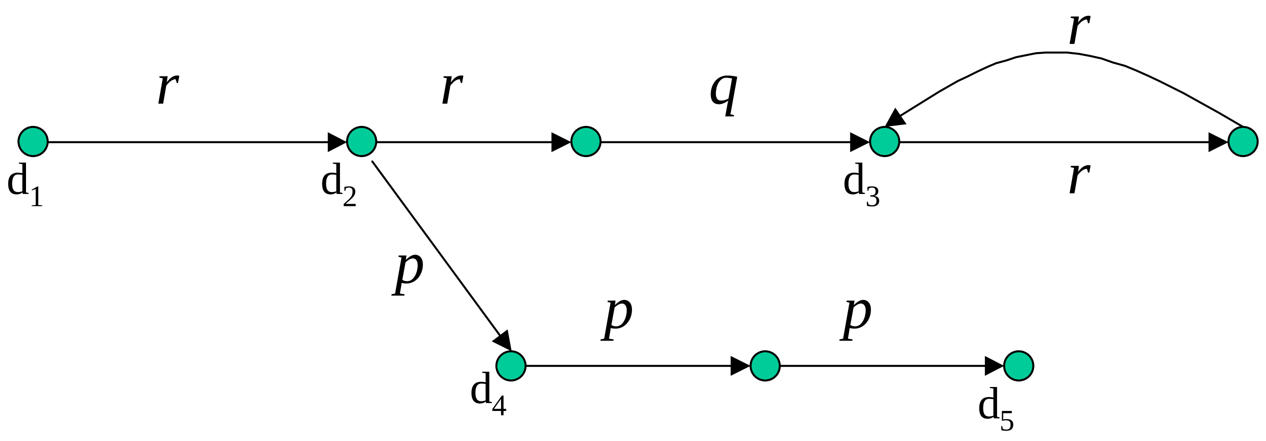
Word: $\$\ d_4\ p\ p\ d_5\ \$\ d_1\ r\ d_2\ \$\ d_4\ p^-\ d_2\ \$\ d_3\ r\ r\ d_3\ \$\ d_2\ r\ q\ d_3\ \$$

$$Q = r \cdot (p \cup q) \cdot (p^- \cdot r)^* \cdot q \cdot q^*$$

State: s_1

Transition: $(s_1, 1) \in \delta_A(s_1, d_1)$

A run of $A_{(Q,d_1,d_3)}$



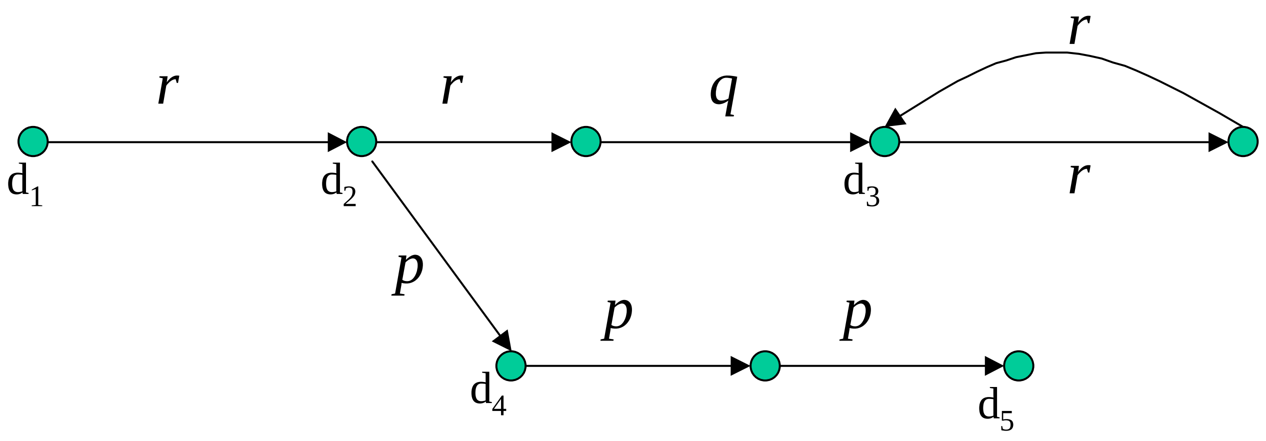
Word: $\$\ d_4\ p\ p\ d_5\ \$\ d_1\ r\ d_2\ \$\ d_4\ p^-\ d_2\ \$\ d_3\ r\ r\ d_3\ \$\ d_2\ r\ q\ d_3\ \$$

$$Q = r \cdot (p \cup q) \cdot (p^- \cdot r)^* \cdot q \cdot q^*$$

State: s_1

Transition: $(s_2, 1) \in \delta_A(s_1, r)$, transition coming from Q

A run of $A_{(Q,d_1,d_3)}$



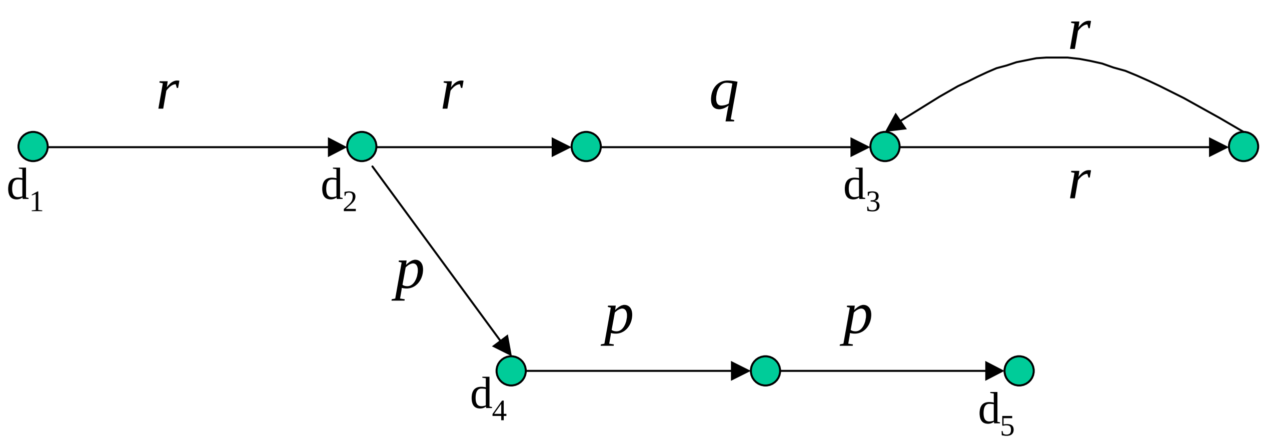
Word: $\$\ d_4\ p\ p\ d_5\ \$\ d_1\ r\ d_2\ \$\ d_4\ p^-\ d_2\ \$\ d_3\ r\ r\ d_3\ \$\ d_2\ r\ q\ d_3\ \$$

$$Q = r \cdot (p \cup q) \cdot (p^- \cdot r)^* \cdot q \cdot q^*$$

State: s_2

Transition: $((s_2, d_2), 1) \in \delta_A(s_2, d_2)$, search for d_2

A run of $A_{(Q,d_1,d_3)}$



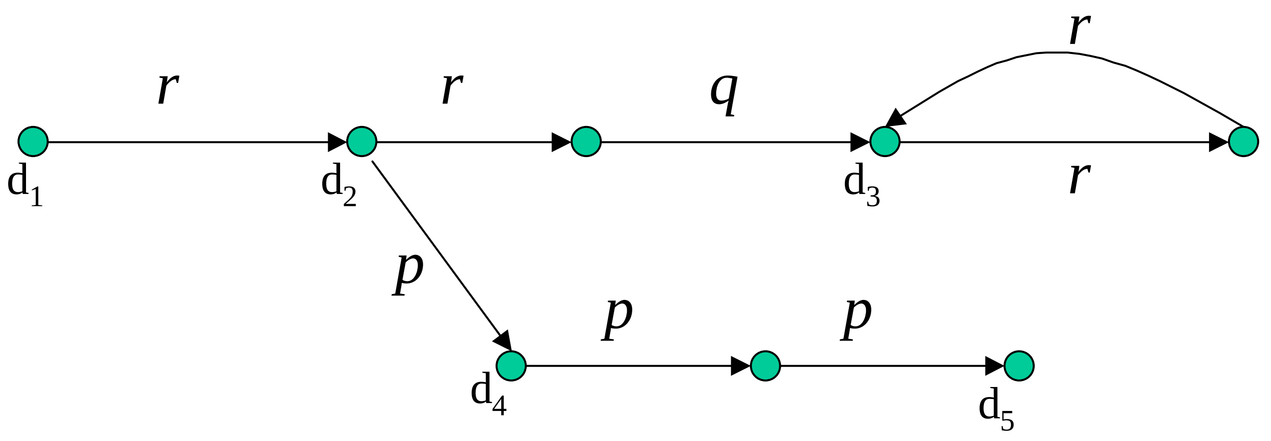
Word: $\$ \; d_4 \; p \; p \; d_5 \; \$ \; d_1 \; r \; d_2 \; \$ \; d_4 \; p^- \; d_2 \; \$ \; d_3 \; r \; r \; d_3 \; \$ \; d_2 \; r \; q \; d_3 \; \$$

$$Q = r \cdot (p \cup q) \cdot (p^- \cdot r)^* \cdot q \cdot q^*$$

State: (s_2, d_2)

Transition: $((s_2, d_2), 1) \in \delta_A((s_2, d_2), \$)$, search for d_2

A run of $A_{(Q,d_1,d_3)}$



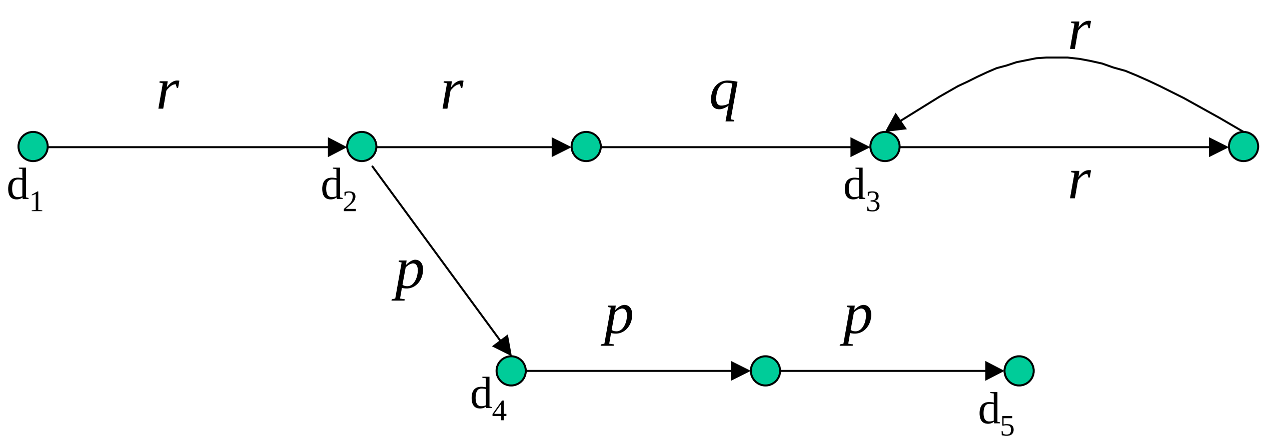
Word: $\$ \ d_4 \ p \ p \ d_5 \ \$ \ d_1 \ r \ d_2 \ \$ \ d_4 \ p^- \ d_2 \ \$ \ d_3 \ r \ r \ d_3 \ \$ \ d_2 \ r \ q \ d_3 \ \$$

$$Q = r \cdot (p \cup q) \cdot (p^- \cdot r)^* \cdot q \cdot q^*$$

State: (s_2, d_2)

Transition: $((s_2, d_2), 1) \in \delta_A((s_2, d_2), d_4)$, search for d_2

A run of $A_{(Q,d_1,d_3)}$



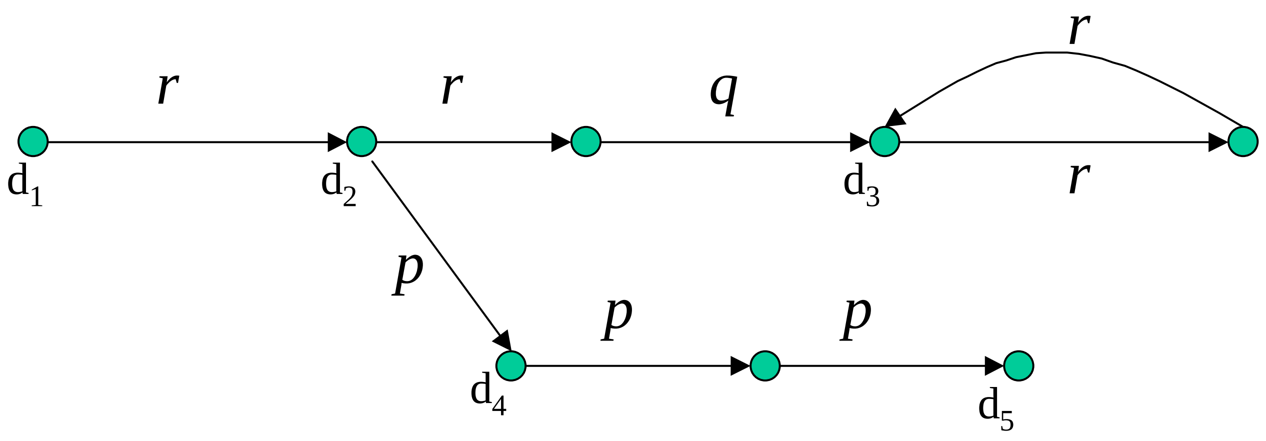
Word: $\$\ d_4\ p\ p\ d_5\ \$\ d_1\ r\ d_2\ \$\ d_4\ p^-\ d_2\ \$\ d_3\ r\ r\ d_3\ \$\ d_2\ r\ q\ d_3\ \$$

$$Q = r \cdot (p \cup q) \cdot (p^- \cdot r)^* \cdot q \cdot q^*$$

State: (s_2, d_2)

Transition: $((s_2, d_2), 1) \in \delta_A((s_2, d_2), p^-)$, search for d_2

A run of $A_{(Q,d_1,d_3)}$



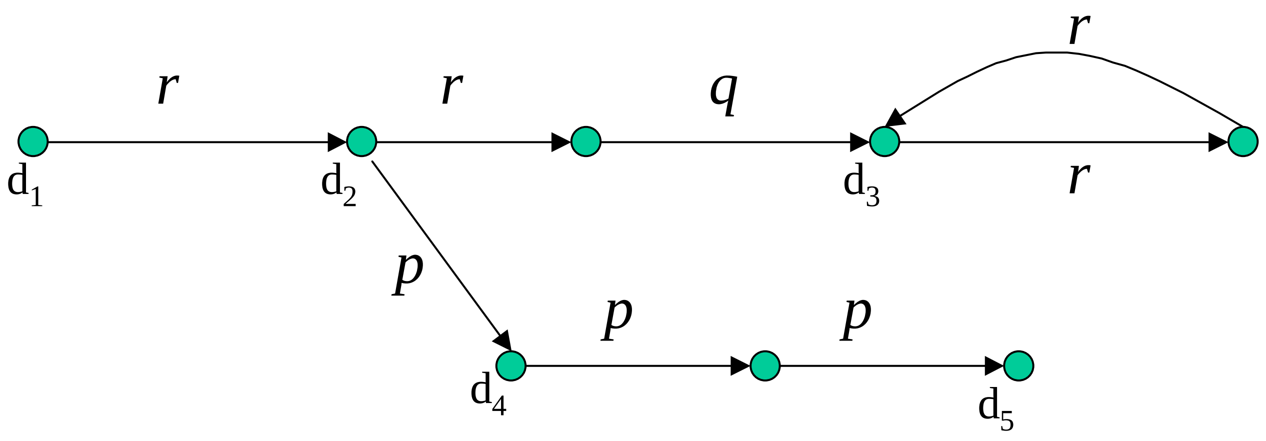
Word: $\$\ d_4\ p\ p\ d_5\ \$\ d_1\ r\ d_2\ \$\ d_4\ p^-\ d_2\ \$\ d_3\ r\ r\ d_3\ \$\ d_2\ r\ q\ d_3\ \$$

$$Q = r \cdot (p \cup q) \cdot (p^- \cdot r)^* \cdot q \cdot q^*$$

State: (s_2, d_2)

Transition: $(s_2, 0) \in \delta_A((s_2, d_2), d_2)$, exit search mode

A run of $A_{(Q,d_1,d_3)}$



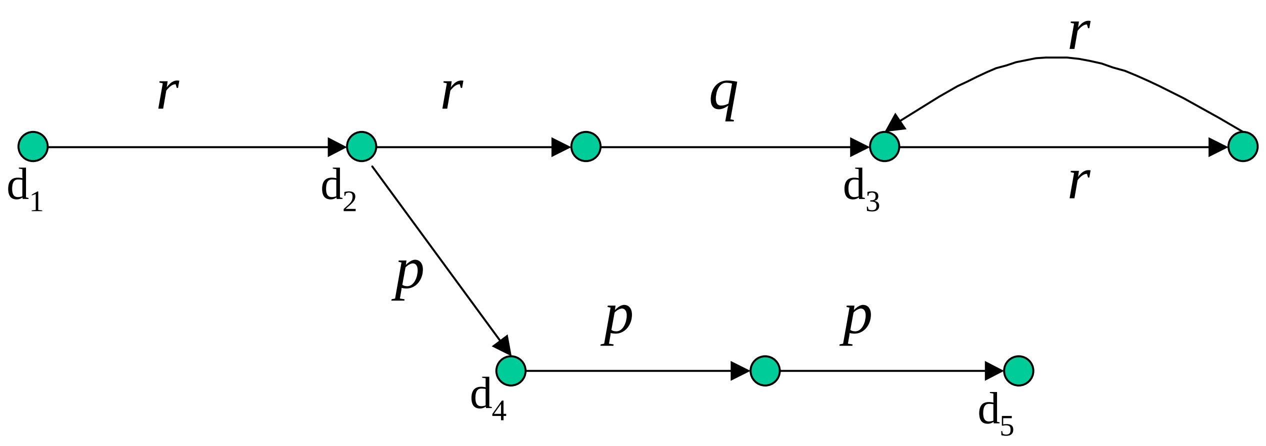
Word: $\$\ d_4\ p\ p\ d_5\ \$\ d_1\ r\ d_2\ \$\ d_4\ p^-\ d_2\ \$\ d_3\ r\ r\ d_3\ \$\ d_2\ r\ q\ d_3\ \$$

$$Q = r \cdot (p \cup q) \cdot (p^- \cdot r)^* \cdot q \cdot q^*$$

State: s_2

Transition: $(s_2^{\leftarrow}, -1) \in \delta_A(s_2, d_2)$, backward mode

A run of $A_{(Q,d_1,d_3)}$



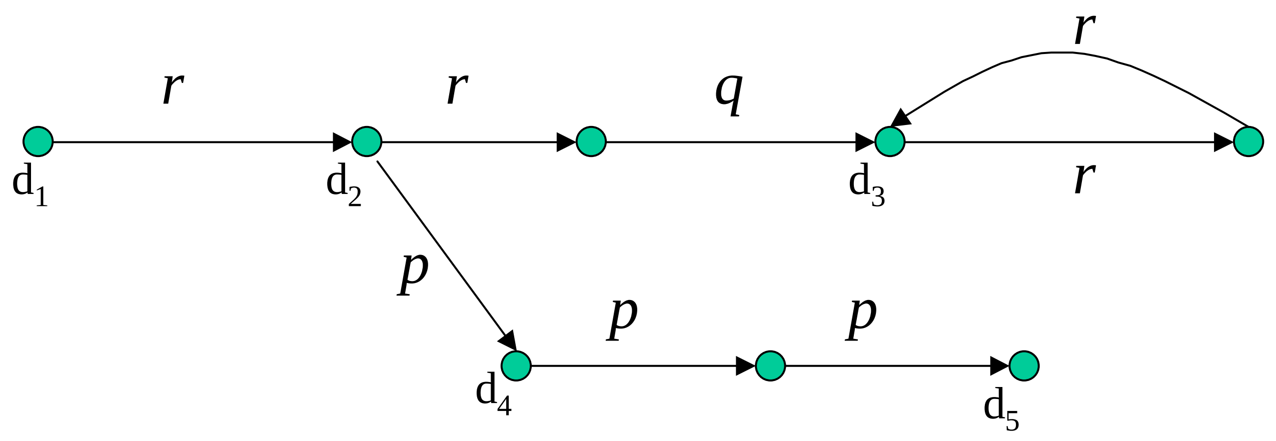
Word: $\$\ d_4\ p\ p\ d_5\ \$\ d_1\ r\ d_2\ \$\ d_4\ p^-\ d_2\ \$\ d_3\ r\ r\ d_3\ \$\ d_2\ r\ q\ d_3\ \$$

$$Q = r \cdot (p \cup q) \cdot (p^- \cdot r)^* \cdot q \cdot q^*$$

State: $s_2^{\leftarrow}$

Transition: $(s_3, 0) \in \delta_A(s_2^{\leftarrow}, p^-)$, transition coming from Q

A run of $A_{(Q,d_1,d_3)}$



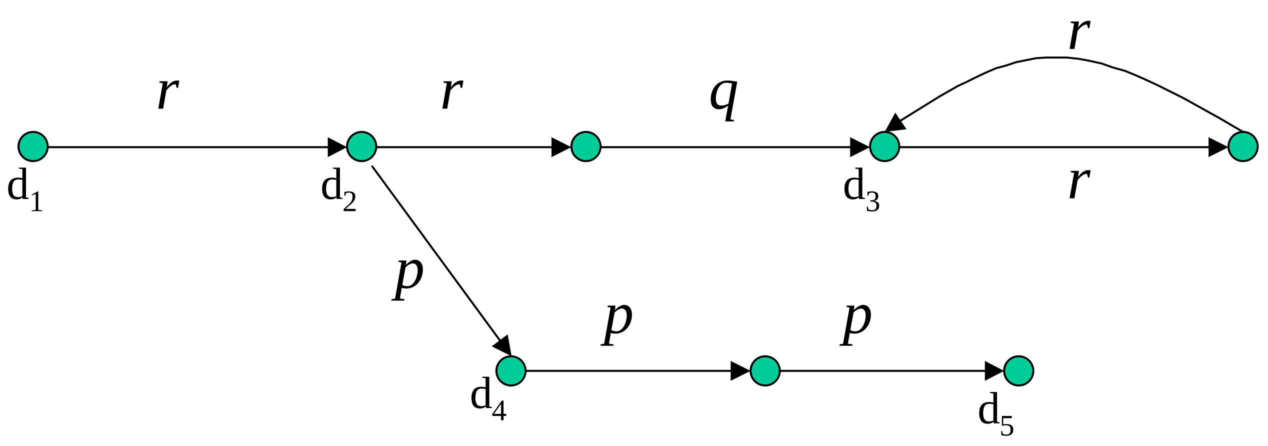
Word: $\$\ d_4\ p\ p\ d_5\ \$\ d_1\ r\ d_2\ \$\ d_4\ p^-\ d_2\ \$\ d_3\ r\ r\ d_3\ \$\ d_2\ r\ q\ d_3\ \$$

$$Q = r \cdot (p \cup q) \cdot (p^- \cdot r)^* \cdot q \cdot q^*$$

State: s_3

Transition: $(s_4, 1) \in \delta_A(s_3, p^-)$, transition coming from Q

A run of $A_{(Q,d_1,d_3)}$



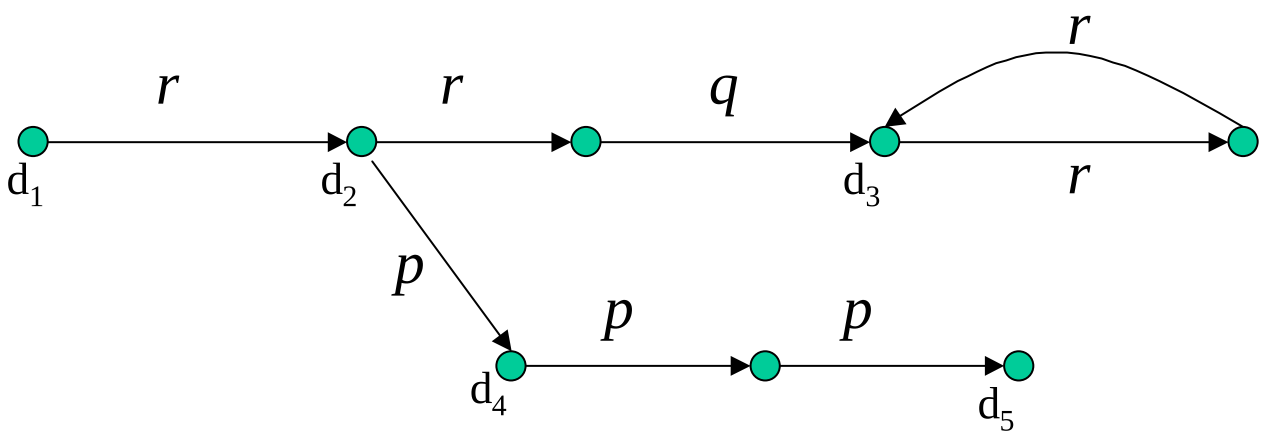
Word: $\$\ d_4\ p\ p\ d_5\ \$\ d_1\ r\ d_2\ \$\ d_4\ p^-\ d_2\ \$\ d_3\ r\ r\ d_3\ \$\ d_2\ r\ q\ d_3\ \$$

$$Q = r \cdot (p \cup q) \cdot (p^- \cdot r)^* \cdot q \cdot q^*$$

State: s_4

Transition: $((s_4, d_2), 1) \in \delta_A(s_4, d_2)$, search for d_2

A run of $A_{(Q,d_1,d_3)}$



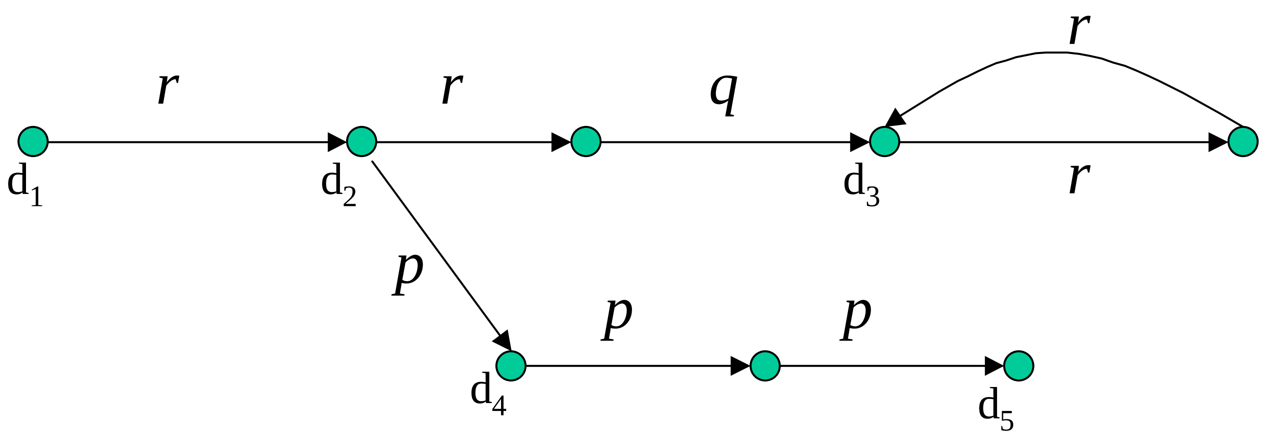
Word: $\$\ d_4\ p\ p\ d_5\ \$\ d_1\ r\ d_2\ \$\ d_4\ p^-\ d_2\ \$\ d_3\ r\ r\ d_3\ \$\ d_2\ r\ q\ d_3\ \$$

$$Q = r \cdot (p \cup q) \cdot (p^- \cdot r)^* \cdot q \cdot q^*$$

State: (s_4, d_2)

Transition: $((s_4, d_2), 1) \in \delta_A((s_4, d_2), \$)$, search for d_2

A run of $A_{(Q,d_1,d_3)}$



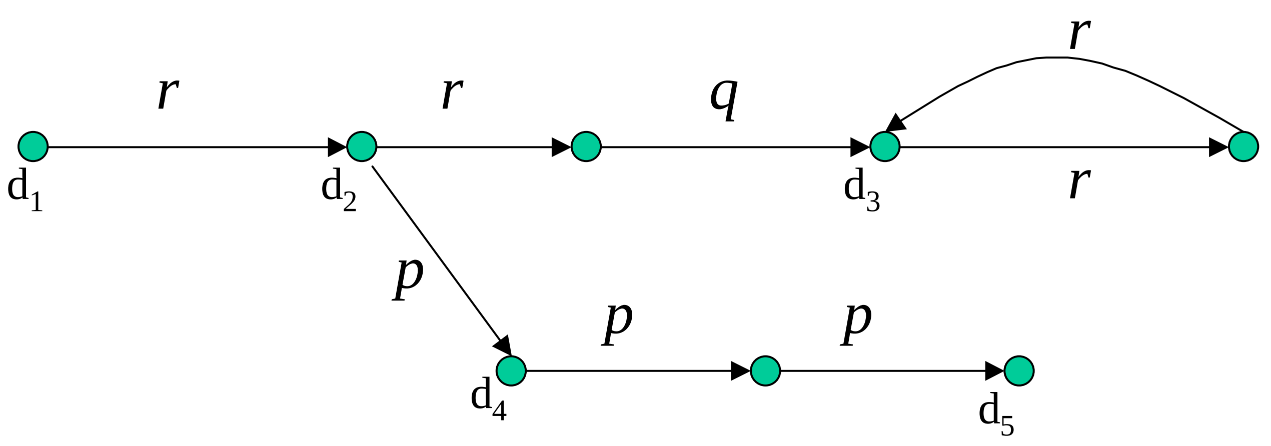
Word: $\$ \; d_4 \; p \; p \; d_5 \; \$ \; d_1 \; r \; d_2 \; \$ \; d_4 \; p^- \; d_2 \; \$ \; d_3 \; r \; r \; d_3 \; \$ \; d_2 \; r \; q \; d_3 \; \$$

$$Q = r \cdot (p \cup q) \cdot (p^- \cdot r)^* \cdot q \cdot q^*$$

State: (s_4, d_2)

Transition: $((s_4, d_2), 1) \in \delta_A((s_4, d_2), d_3)$, search for d_2

A run of $A_{(Q,d_1,d_3)}$



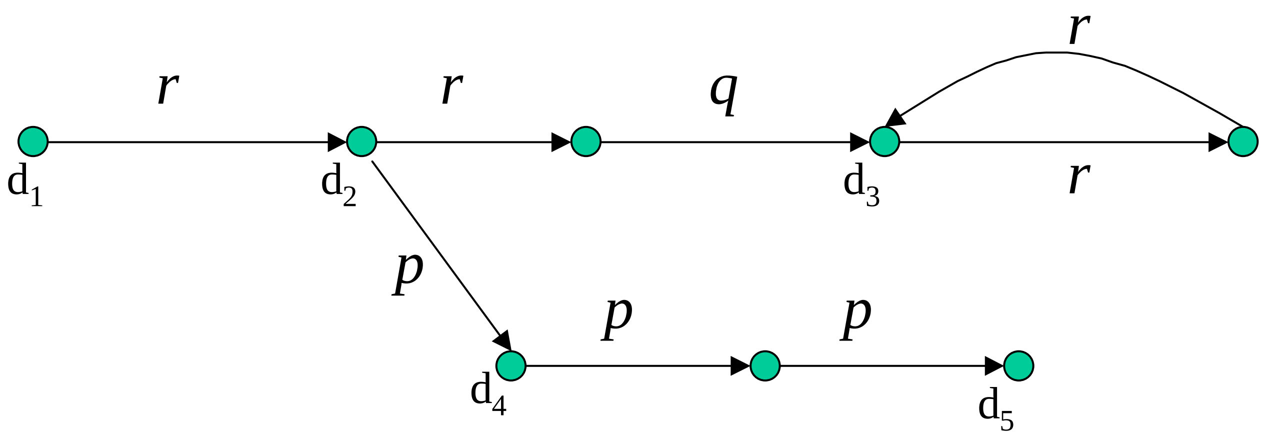
Word: $\$\ d_4\ p\ p\ d_5\ \$\ d_1\ r\ d_2\ \$\ d_4\ p^-\ d_2\ \$\ d_3\ r\ r\ d_3\ \$\ d_2\ r\ q\ d_3\ \$$

$$Q = r \cdot (p \cup q) \cdot (p^- \cdot r)^* \cdot q \cdot q^*$$

State: (s_4, d_2)

Transition: $((s_4, d_2), 1) \in \delta_A((s_4, d_2), r)$, search for d_2

A run of $A_{(Q,d_1,d_3)}$



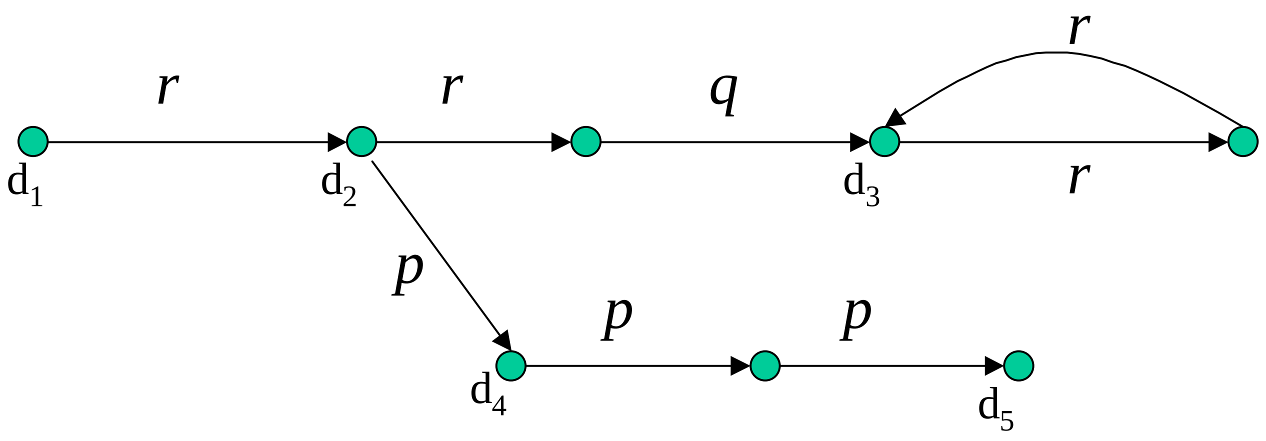
Word: $\$\ d_4\ p\ p\ d_5\ \$\ d_1\ r\ d_2\ \$\ d_4\ p^-\ d_2\ \$\ d_3\ r\ r\ d_3\ \$\ d_2\ r\ q\ d_3\ \$$

$$Q = r \cdot (p \cup q) \cdot (p^- \cdot r)^* \cdot q \cdot q^*$$

State: (s_4, d_2)

Transition: $((s_4, d_2), 1) \in \delta_A((s_4, d_2), r)$, search for d_2

A run of $A_{(Q,d_1,d_3)}$



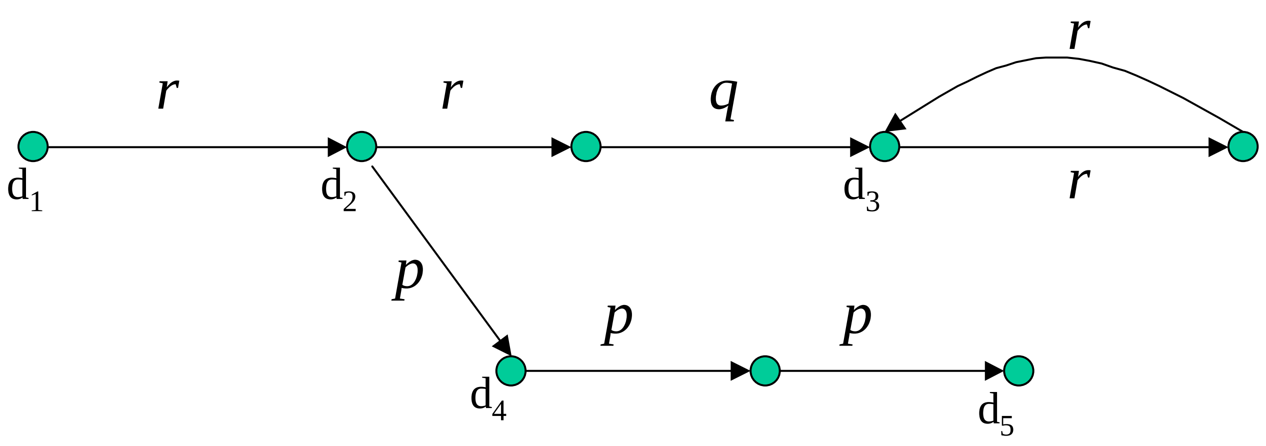
Word: $\$ \; d_4 \; p \; p \; d_5 \; \$ \; d_1 \; r \; d_2 \; \$ \; d_4 \; p^- \; d_2 \; \$ \; d_3 \; r \; r \; d_3 \; \$ \; d_2 \; r \; q \; d_3 \; \$$

$$Q = r \cdot (p \cup q) \cdot (p^- \cdot r)^* \cdot q \cdot q^*$$

State: (s_4, d_2)

Transition: $((s_4, d_2), 1) \in \delta_A((s_4, d_2), d_3)$, search for d_2

A run of $A_{(Q,d_1,d_3)}$



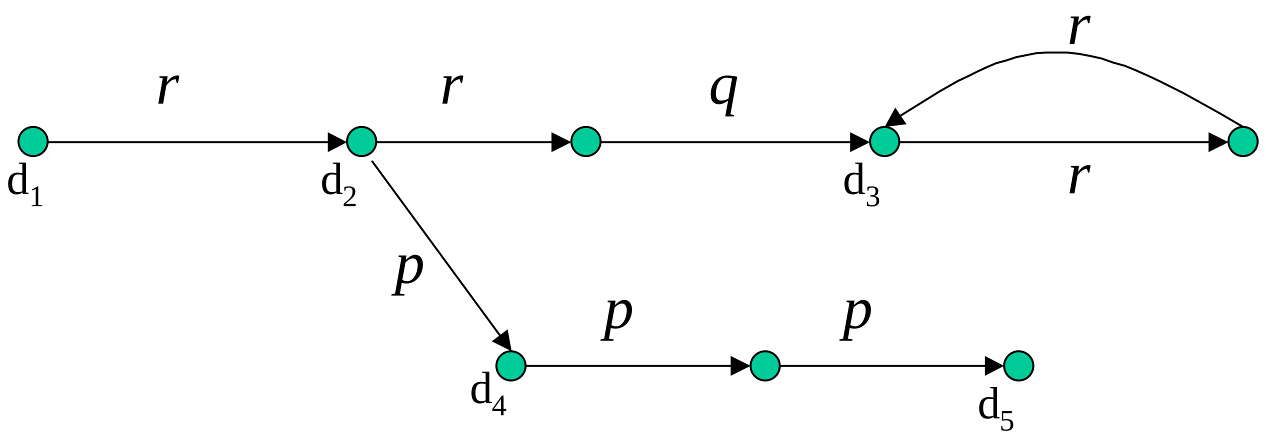
Word: $\$ \; d_4 \; p \; p \; d_5 \; \$ \; d_1 \; r \; d_2 \; \$ \; d_4 \; p^- \; d_2 \; \$ \; d_3 \; r \; r \; d_3 \; \$ \; d_2 \; r \; q \; d_3 \; \$$

$$Q = r \cdot (p \cup q) \cdot (p^- \cdot r)^* \cdot q \cdot q^*$$

State: (s_4, d_2)

Transition: $((s_4, d_2), 1) \in \delta_A((s_4, d_2), \$)$, search for d_2

A run of $A_{(Q,d_1,d_3)}$



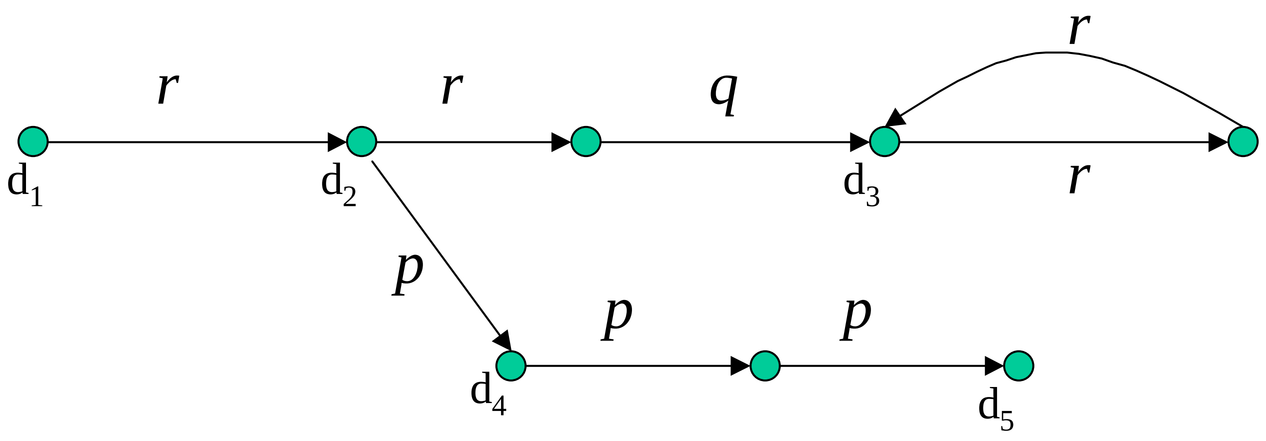
Word: $\$ \; d_4 \; p \; p \; d_5 \; \$ \; d_1 \; r \; d_2 \; \$ \; d_4 \; p^- \; d_2 \; \$ \; d_3 \; r \; r \; d_3 \; \$ \; d_2 \; r \; q \; d_3 \; \$$

$$Q = r \cdot (p \cup q) \cdot (p^- \cdot r)^* \cdot q \cdot q^*$$

State: (s_4, d_2)

Transition: $(s_4, 0) \in \delta_A((s_4, d_2), d_2)$, exit search mode

A run of $A_{(Q,d_1,d_3)}$



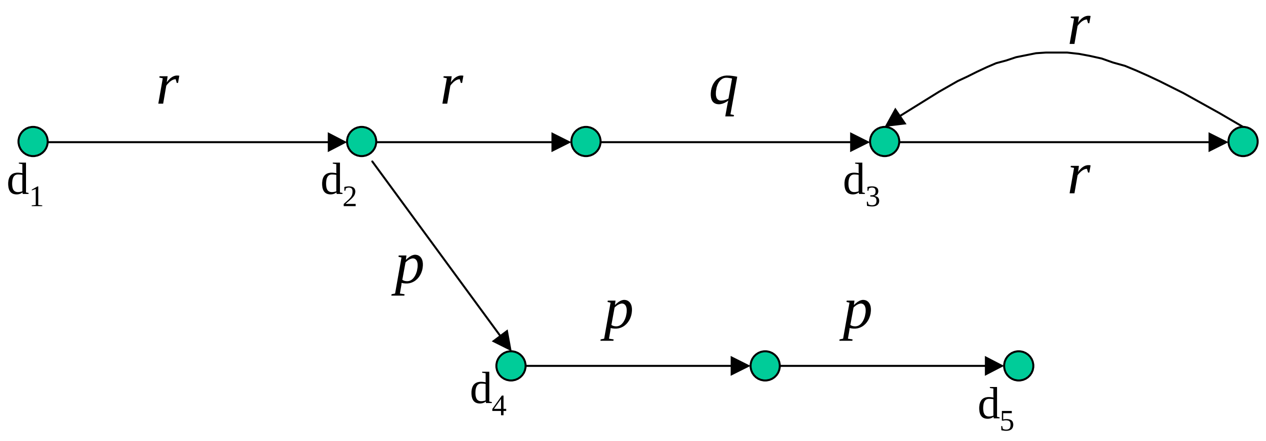
Word: $\$\ d_4\ p\ p\ d_5\ \$\ d_1\ r\ d_2\ \$\ d_4\ p^-\ d_2\ \$\ d_3\ r\ r\ d_3\ \$\ d_2\ r\ q\ d_3\ \$$

$$Q = r \cdot (p \cup q) \cdot (p^- \cdot r)^* \cdot q \cdot q^*$$

State: s_4

Transition: $(s_4, 1) \in \delta_A(s_4, d_2)$

A run of $A_{(Q,d_1,d_3)}$



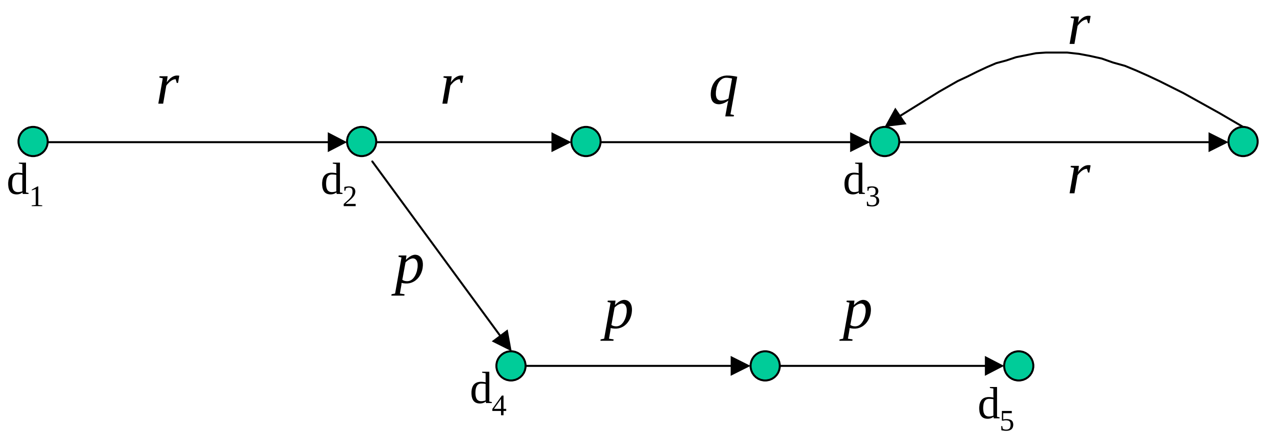
Word: $\$ \; d_4 \; p \; p \; d_5 \; \$ \; d_1 \; r \; d_2 \; \$ \; d_4 \; p^- \; d_2 \; \$ \; d_3 \; r \; r \; d_3 \; \$ \; d_2 \; r \; q \; d_3 \; \$$

$$Q = r \cdot (p \cup q) \cdot (p^- \cdot r)^* \cdot q \cdot q^*$$

State: s_4

Transition: $(s_5, 1) \in \delta_A(s_4, r)$, transition coming from Q

A run of $A_{(Q,d_1,d_3)}$



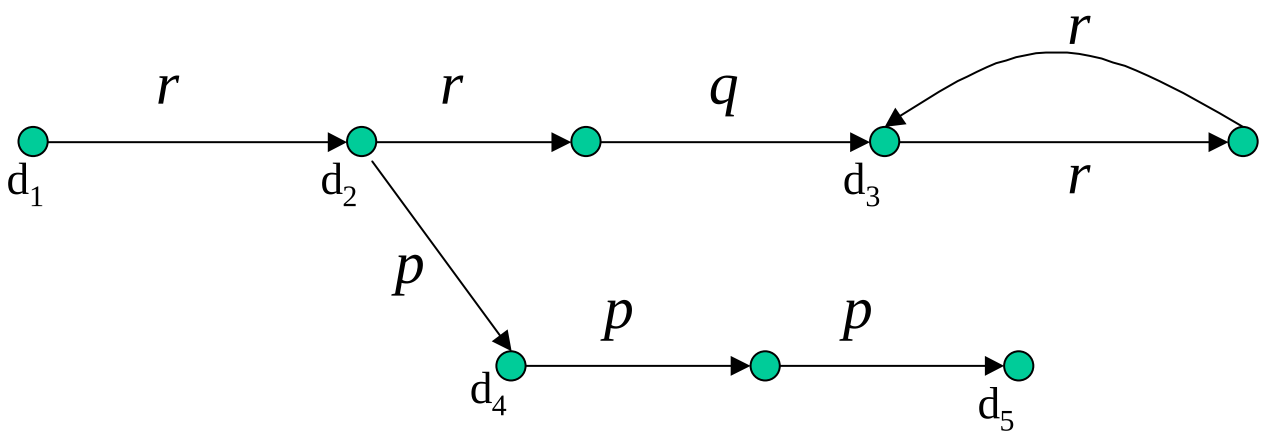
Word: $\$\ d_4\ p\ p\ d_5\ \$\ d_1\ r\ d_2\ \$\ d_4\ p^-\ d_2\ \$\ d_3\ r\ r\ d_3\ \$\ d_2\ r\ q\ d_3\ \$$

$$Q = r \cdot (p \cup q) \cdot (p^- \cdot r)^* \cdot q \cdot q^*$$

State: s_5

Transition: $(s_6, 1) \in \delta_A(s_5, q)$, transition coming from Q

A run of $A_{(Q,d_1,d_3)}$



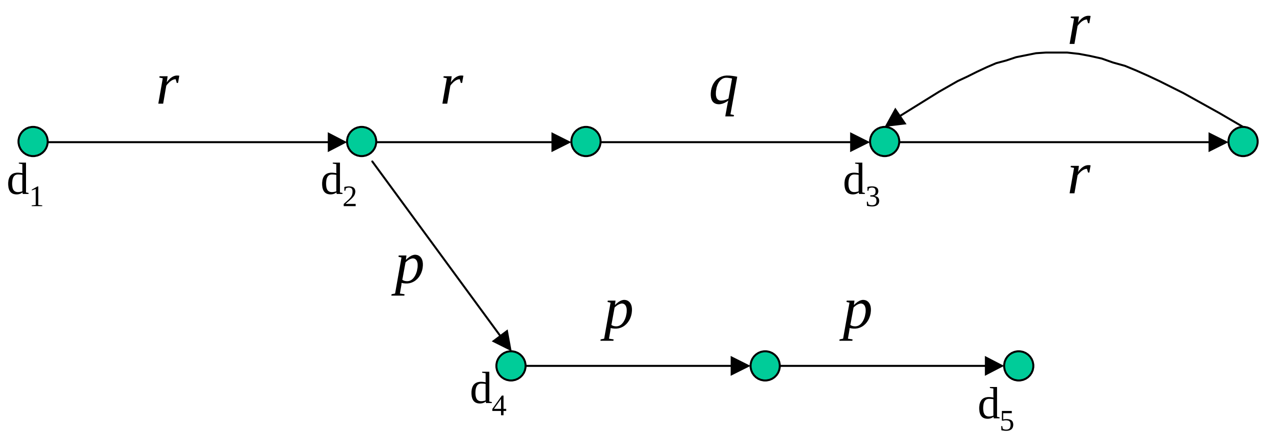
Word: $\$\ d_4\ p\ p\ d_5\ \$\ d_1\ r\ d_2\ \$\ d_4\ p^-\ d_2\ \$\ d_3\ r\ r\ d_3\ \$\ d_2\ r\ q\ d_3\ \$$

$$Q = r \cdot (p \cup q) \cdot (p^- \cdot r)^* \cdot q \cdot q^*$$

State: s_6

Transition: $(s_7, 0) \in \delta_A(s_6, d_3)$, s_7 final state

A run of $A_{(Q,d_1,d_3)}$



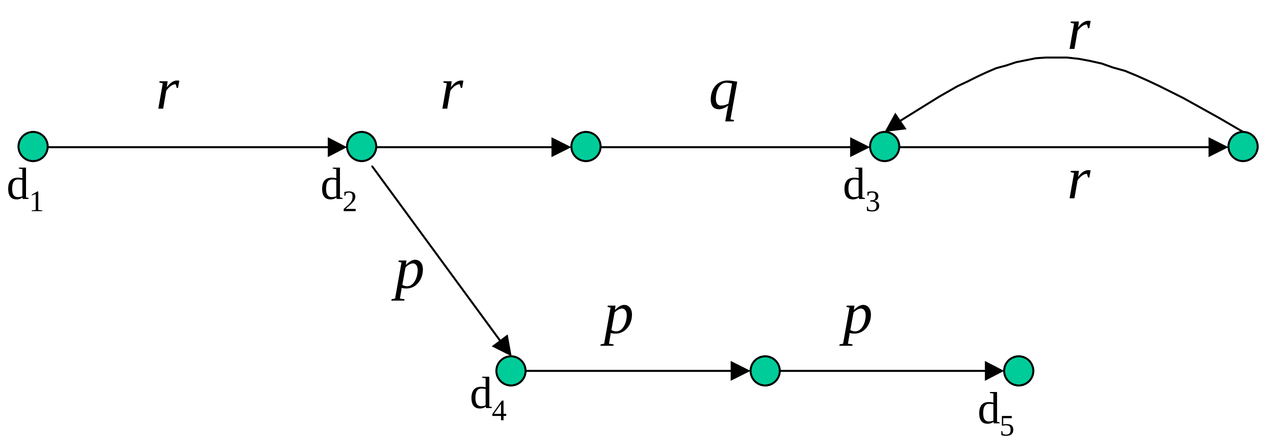
Word: $\$ \ d_4 \ p \ p \ d_5 \ \$ \ d_1 \ r \ d_2 \ \$ \ d_4 \ p^- \ d_2 \ \$ \ d_3 \ r \ r \ d_3 \ \$ \ d_2 \ r \ q \ d_3 \ \$$

$$Q = r \cdot (p \cup q) \cdot (p^- \cdot r)^* \cdot q \cdot q^*$$

State: s_7

Transition: $(s_7, 1) \in \delta_A(s_7, d_3)$, s_7 final state

A run of $A_{(Q,d_1,d_3)}$



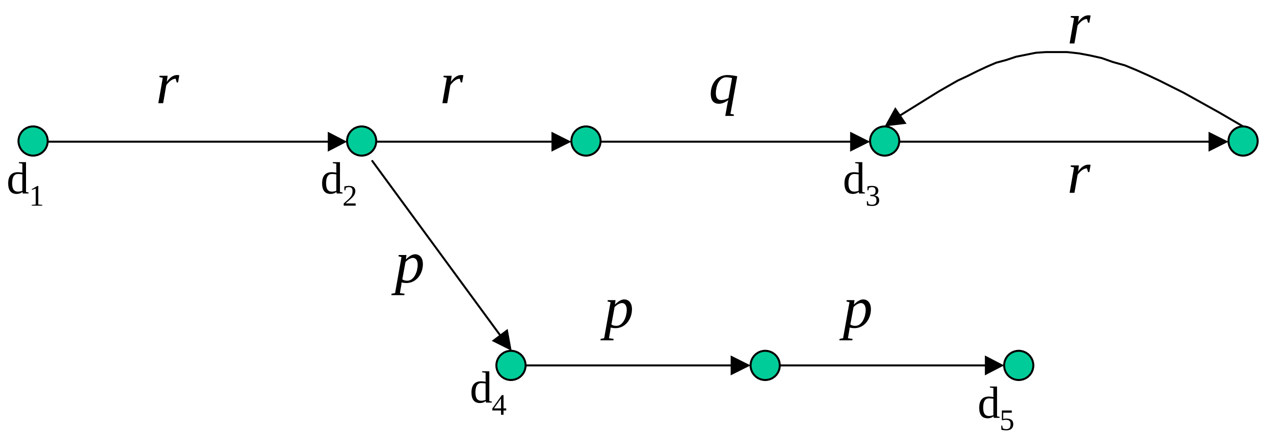
Word: $\$ \; d_4 \; p \; p \; d_5 \; \$ \; d_1 \; r \; d_2 \; \$ \; d_4 \; p^- \; d_2 \; \$ \; d_3 \; r \; r \; d_3 \; \$ \; d_2 \; r \; q \; d_3 \; \$$

$$Q = r \cdot (p \cup q) \cdot (p^- \cdot r)^* \cdot q \cdot q^*$$

State: s_7

Transition: $(s_7, 1) \in \delta_A(s_7, \$)$, s_7 final state

A run of $A_{(Q,d_1,d_3)}$



Word: $\$ \; d_4 \; p \; p \; d_5 \; \$ \; d_1 \; r \; d_2 \; \$ \; d_4 \; p^- \; d_2 \; \$ \; d_3 \; r \; r \; d_3 \; \$ \; d_2 \; r \; q \; d_3 \; \$$

$$Q = r \cdot (p \cup q) \cdot (p^- \cdot r)^* \cdot q \cdot q^*$$

State: s_7 final state

Word accepted by $A_{(Q,d_1,d_3)}$!

Query answering: Technique

To check whether $(c, d) \notin Q^{\mathcal{I},\mathcal{C}}$, we check for nonemptiness of A, that is the intersection of

- the one-way automaton A_0 that accepts words that represent databases, i.e., words of the form $(\$\cdot \mathcal{C}\cdot\Sigma^+\cdot \mathcal{C})^*\cdot\$$
- the one-way automata corresponding to the various $A_{(S_i,a,b)}$ (for each source S_i and for each pair $(a, b) \in S_i^{\mathcal{C}}$)
- the one-way automaton corresponding to the complement of $A_{(Q,c,d)}$

Indeed, any word accepted by such intersection automaton represents a counterexample to $(c, d) \in Q^{\mathcal{I},\mathcal{C}}$.

Query answering: Complexity

- All two-way automata constructed above are of linear size in the size of Q, the queries associated to $S_1, \ldots, S_k$, and $S_1^{\mathcal{C}}, \ldots, S_k^{\mathcal{C}}$. Hence, the corresponding one-way automata would be exponential.

- However, we do not need to construct A explicitly. Instead, we can construct it **on the fly** while checking for nonemptiness.

Query answering for 2RPQs is PSPACE-complete in combined complexity (as for RPQs).

Complexity of query answering for 2RPQs: the complete picture

From [Calvanese&al. PODS'00]:

Assumption on domain	**Assumption on views**	Complexity		
		data	*expression*	*combined*
closed	**all sound**	*coNP*	*coNP*	*coNP*
	all exact	*coNP*	*coNP*	*coNP*
	arbitrary	*coNP*	*coNP*	*coNP*
open	**all sound**	*coNP*	*PSPACE*	*PSPACE*
	all exact	*coNP*	*PSPACE*	*PSPACE*
	arbitrary	*coNP*	*PSPACE*	*PSPACE*

INT[noconstr, LAV/sound]: Connection to rewriting

Query answering by rewriting:

- Given $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$, and given a query Q over $\mathcal{G}$, rewrite Q into a query, called $rew(Q, \mathcal{I})$, in the alphabet $\mathcal{A}_{\mathcal{S}}$ of the sources
- Evaluate the rewriting $rew(Q, \mathcal{I})$ over the source database

We are interested in sound rewritings (computing only certain answers, for every source database $\mathcal{C}$) that are expressed in a given query language, and that are maximal for the class of queries expressible in such language. Sometimes, we are interested in exact rewritings, i.e., rewritings that are logically equivalent to the query, modulo $\mathcal{M}$.

But:

- *When does the rewriting compute all certain answers?*
- *What do we gain or lose by focusing on a given class of queries?*

Perfect rewriting

Let $cert(Q, \mathcal{I}, \mathcal{C})$ be the function that, given query Q, data integration system $\mathcal{I}$, and source database $\mathcal{C}$, computes the certain answers $Q^{\mathcal{I},\mathcal{C}}$ to Q wrt $\mathcal{I}$ and $\mathcal{C}$.

Define $cert_{[Q,\mathcal{I}]}(\cdot)$ to be the function that, with Q and $\mathcal{I}$ fixed, given source database $\mathcal{C}$, computes the certain answers $Q^{\mathcal{I},\mathcal{C}}$.

- $cert_{[Q,\mathcal{I}]}$ can be seen as a query on the alphabet $\mathcal{A}_\mathcal{S}$ that, given $\mathcal{C}$, returns $Q^{\mathcal{I},\mathcal{C}}$
- $cert_{[Q,\mathcal{I}]}$ is a (*sound*) rewriting of Q wrt $\mathcal{I}$
- No sound rewriting exists that is better than $cert_{[Q,\mathcal{I}]}$
- $cert_{[Q,\mathcal{I}]}$ is called the **perfect rewriting** of Q wrt $\mathcal{I}$

Properties of the perfect rewriting

- Can we express the perfect rewriting in a certain query language?
- How does a maximal rewriting for a given class of queries compare with the perfect rewriting?
  - From a semantical point of view
  - From a computational point of view
- Which is the computational complexity of (finding, evaluating) the perfect rewriting?

The case of conjunctive queries

Let $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$ be a LAV/sound data integration system, let Q and the queries in $\mathcal{M}$ be CQs, and let Q' be the **union of all maximal rewritings of Q for the class of CQs**. Then ([Levy&al. PODS'95], [Duschka&al.'97], [Abiteboul&al. PODS'98])

- Q' is the maximal rewriting for the class of unions of conjunctive queries (UCQs)
- **Q' is the perfect rewriting of Q wrt $\mathcal{I}$**
- Q' is a PTIME query
- Q' is an exact rewriting (equivalent to Q for each database $\mathcal{B}$ of $\mathcal{I}$), if an exact rewriting exists

Does this "ideal situation" carry on to cases where Q and $\mathcal{M}$ allow for union?

Unions of path queries (UPQs)

Very simple query language (called UPQ) defined as follows:

$$\begin{array}{rcl} Q & \longrightarrow & P \mid Q_1 \cup Q_2 \\ P & \longrightarrow & R \mid P_1 \circ P_2 \end{array}$$

R denotes a **binary database relation**, P denotes a **path query**, which is a chaining of database relations, and Q denotes a **union of path queries**.

UPQs are a simple form of

- Unions of conjunctive queries
- Regular path queries

View-based query processing for UPQs

View-based query answering for UPQs is coNP-complete in data complexity [Calvanese&al. ICDE'00].

In other words, $cert(Q, \mathcal{I}, \mathcal{C})$, with Q and $\mathcal{I}$ fixed, is a coNP-complete function.

$\Rightarrow$ **The perfect rewriting $cert_{[Q,\mathcal{I}]}$ is a coNP-complete query**.

For query languages that include UPQs the perfect rewriting is coNP-hard — we do not have the ideal situation we had for conjunctive queries.

Problem: *Isolate those UPQs Q and $\mathcal{I}$ for which the perfect rewriting $cert_{[Q,\mathcal{I}]}$ is a PTIME function (assuming P$\neq$NP)* [Calvanese&al. LICS'00].

Incompleteness and inconsistency

Constraints in $\mathcal{G}$	Type of mapping	Incomple-teness	Inconsi-stency
no	GAV/exact	no	no
no	GAV/sound	**yes**/no	no
no	LAV/sound	**yes**	no
no	*LAV/exact*	**yes**	**yes**
yes	GAV/exact	no	**yes**
yes	GAV/sound	**yes**	**yes**
yes	LAV/sound	**yes**	**yes**
yes	LAV/exact	**yes**	**yes**

INT[noconstr, LAV/exact]: inconsistency

The LAV mapping assertions have the logical form:

$$\forall \vec{\mathbf{x}} \;\; s(\vec{\mathbf{x}}) \equiv \phi_{\mathcal{G}}(\vec{\mathbf{x}})$$

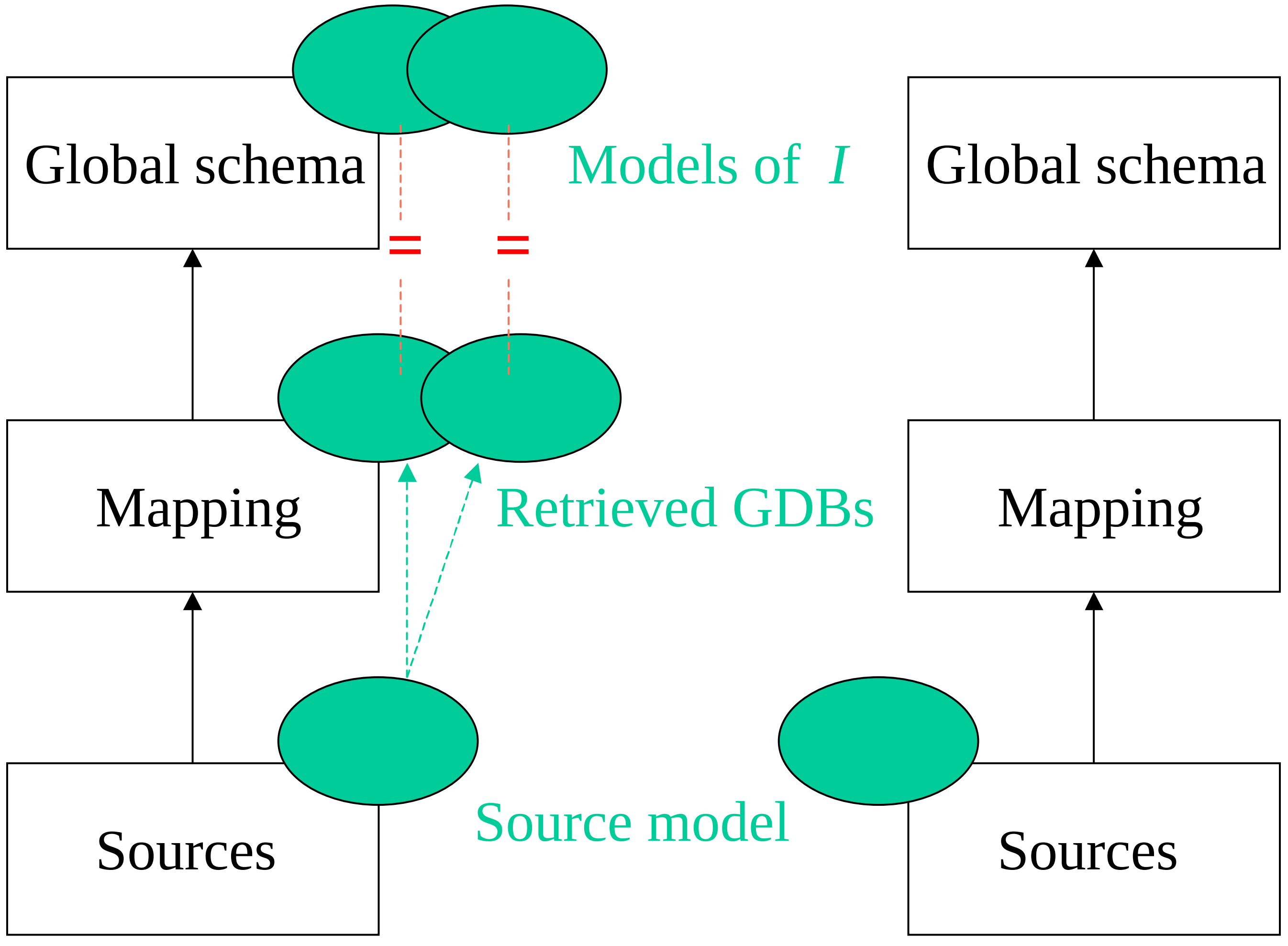
In general, given a source database $\mathcal{C}$, there may be no solution of the above assertions (i.e., no database that is legal wrt $\mathcal{G}$ and that satisfies $\mathcal{M}$ wrt $\mathcal{C}$).

Example:

$$\begin{array}{rcl} s_1(x) & \equiv & \{\ (x) \mid g(x)\ \} \\ s_2(x) & \equiv & \{\ (x) \mid g(x)\ \} \end{array}$$

with ${s_1}^{\mathcal{C}} = \{1\}$, and ${s_2}^{\mathcal{C}} = \{2\}$.

INT[noconstr, LAV/exact]



Incompleteness

Inconsistency

INT[noconstr, LAV/exact]: some results for query answering

- Complexity analysis (sound, complete, exact) [Abiteboul&Duschka PODS'98] [Grahne&Mendelzon ICDT'99]
- Variants of Regular Path Queries [Calvanese&al. ICDE'00, PODS'00]

INT[noconstr, LAV/exact]: data complexity

From [Abiteboul&Duschka PODS'98]:

Sound sources	CQ	CQ$^{\neq}$	PQ	datalog	FOL
CQ	*PTIME*	*coNP*	*PTIME*	*PTIME*	*undec.*
CQ$^{\neq}$	*PTIME*	*coNP*	*PTIME*	*PTIME*	*undec.*
PQ	*coNP*	*coNP*	*coNP*	*coNP*	*undec.*
datalog	*coNP*	*undec.*	*coNP*	*undec.*	*undec.*
FOL	*undec.*	*undec.*	*undec.*	*undec.*	*undec.*

Exact sources	CQ	CQ$^{\neq}$	PQ	datalog	FOL
CQ	*coNP*	*coNP*	*coNP*	*coNP*	*undec.*
CQ$^{\neq}$	*coNP*	*coNP*	*coNP*	*coNP*	*undec.*
PQ	*coNP*	*coNP*	*coNP*	*coNP*	*undec.*
datalog	*undec.*	*undec.*	*undec.*	*undec.*	*undec.*
FOL	*undec.*	*undec.*	*undec.*	*undec.*	*undec.*

INT[noconstr, LAV/exact]: polynomial intractability

Given a graph $G = (N, E)$, we define $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$, and source database $\mathcal{C}$:

$$\begin{array}{lcl}
V_1 & \equiv & \{ (X) \mid color(X, Y) \} \\
V_2 & \equiv & \{ (Y) \mid color(X, Y) \} \\
V_3 & \equiv & \{ (X, Y) \mid edge(X, Y) \} \\
V_1{}^{\mathcal{C}} & = & N \\
V_2{}^{\mathcal{C}} & = & \{ red, \ green, \ blue \} \\
V_3{}^{\mathcal{C}} & = & E \\
Q & \leftarrow & \{ () \mid edge(X, Y) \wedge color(X, Z) \wedge color(Y, Z) \}
\end{array}$$

$Q^{\mathcal{I},\mathcal{C}}$ is true if and only if G is not 3-colorable.

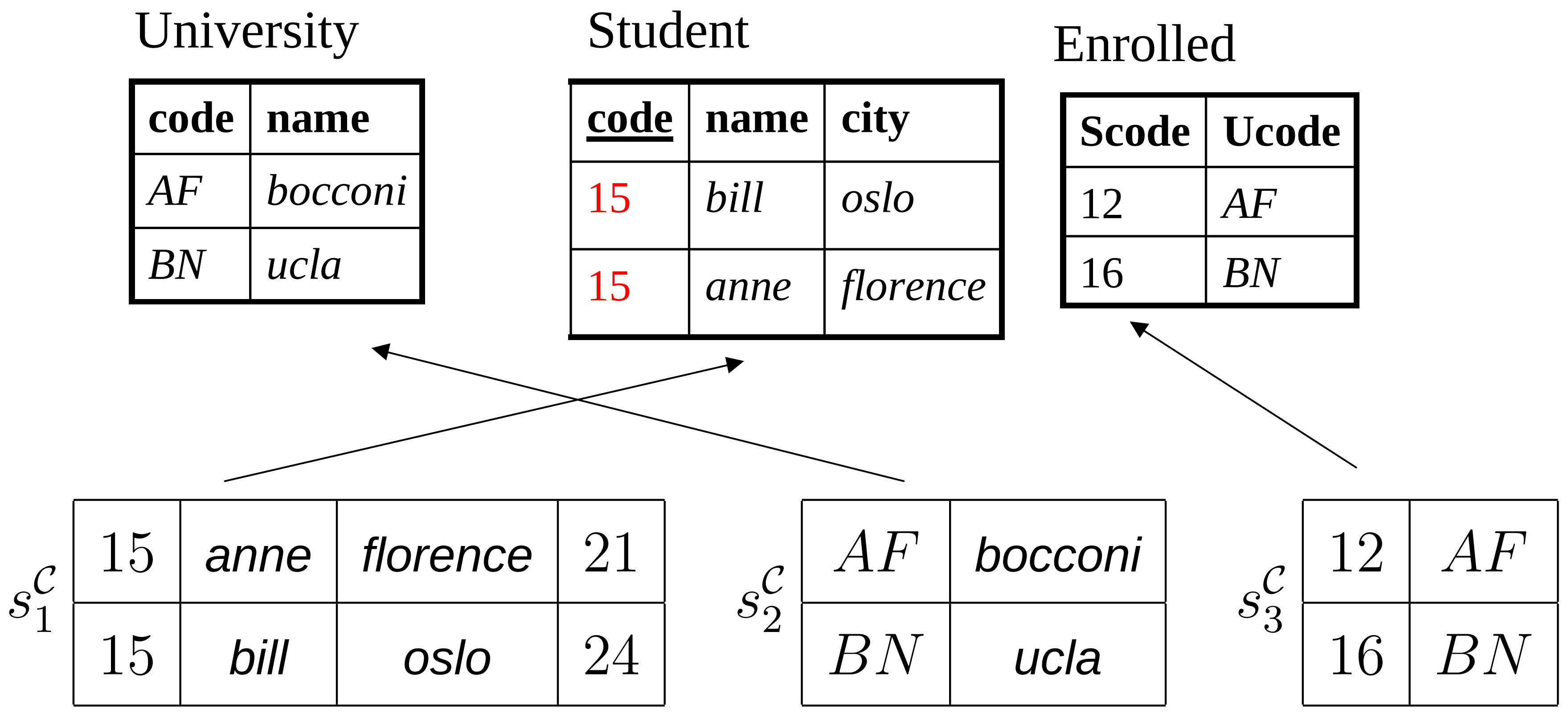
$\Longrightarrow$ **coNP-hard data complexity** for conjunctive queries and views.

Incompleteness and inconsistency

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

INT[constr, GAV/exact]: inconsistency

Given one source database $\mathcal{C}$, there is only one database for $\mathcal{G}$ that satisfies the mapping wrt $\mathcal{C}$. If this is not legal wrt $\mathcal{G}$, then the system is inconsistent ($\mathcal{I}$ has no model), otherwise, the case is similar to INT[noconstr, GAV/exact].



University

code	**name**
AF	*bocconi*
BN	*ucla*

Student

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

Enrolled

Scode	**Ucode**
12	*AF*
16	*BN*

$s_1^{\mathcal{C}}$

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

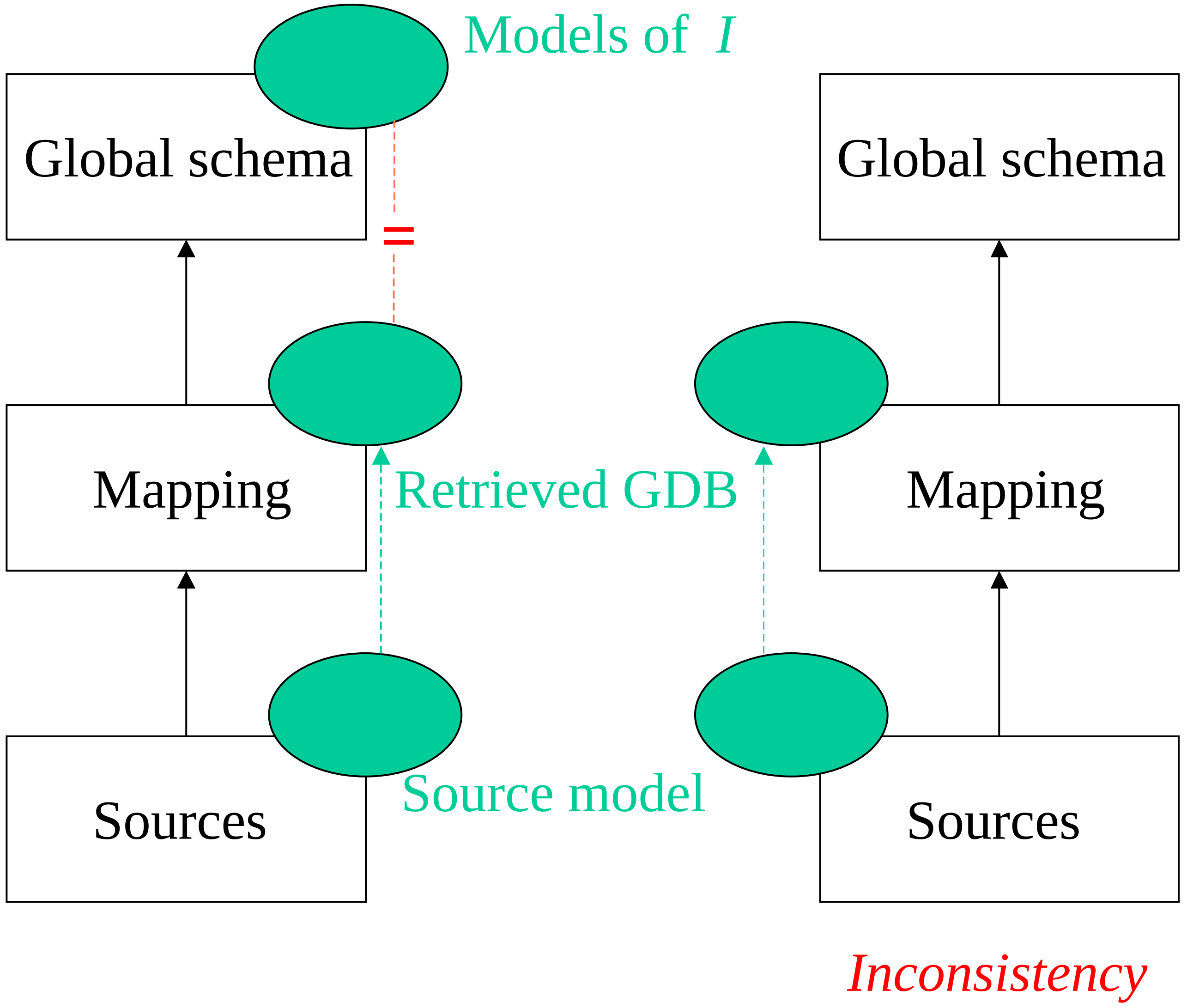
$s_2^{\mathcal{C}}$

AF	*bocconi*
BN	*ucla*

$s_3^{\mathcal{C}}$

12	AF
16	BN

INT[constr, GAV/exact]



Incompleteness and inconsistency

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

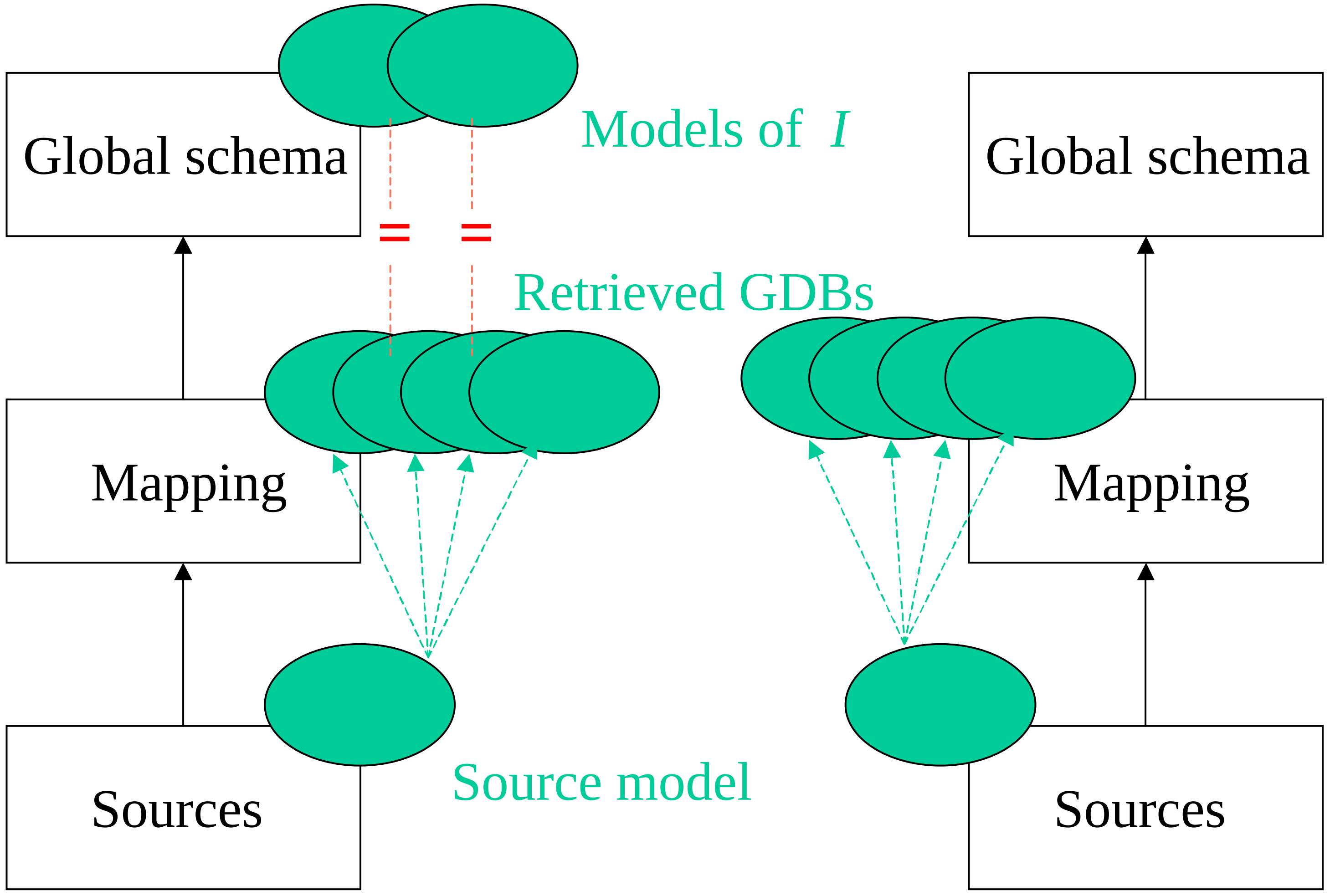
INT[constr, GAV/sound]: incompleteness

Let us consider a system with a global schema with constraints, and with a GAV mapping $\mathcal{M}$ with sound sources, whose assertions have the form

$$g \supseteq \phi_{\mathcal{S}} \quad \text{with the meaning} \quad \forall \mathbf{x}\ (\phi_{\mathcal{S}}(\mathbf{x}) \rightarrow\ g(\mathbf{x}))$$

Since $\mathcal{G}$ **does have constraints**, we cannot simply limit our attention to **one** database of the integration system (as we did for INT[noconstr, GAV/exact] and INT[noconstr, GAV/sound]).

INT[constr, GAV/sound]



Incompleteness

Inconsistency

INT[constr, GAV/sound]: example

Global schema $\mathcal{G}$:

student$(Scode, Sname, Scity)$, $\quad key\{Scode\}$

university$(Ucode, Uname)$, $\quad key\{Ucode\}$

enrolled$(Scode, Ucode)$, $\quad key\{Scode, Ucode\}$

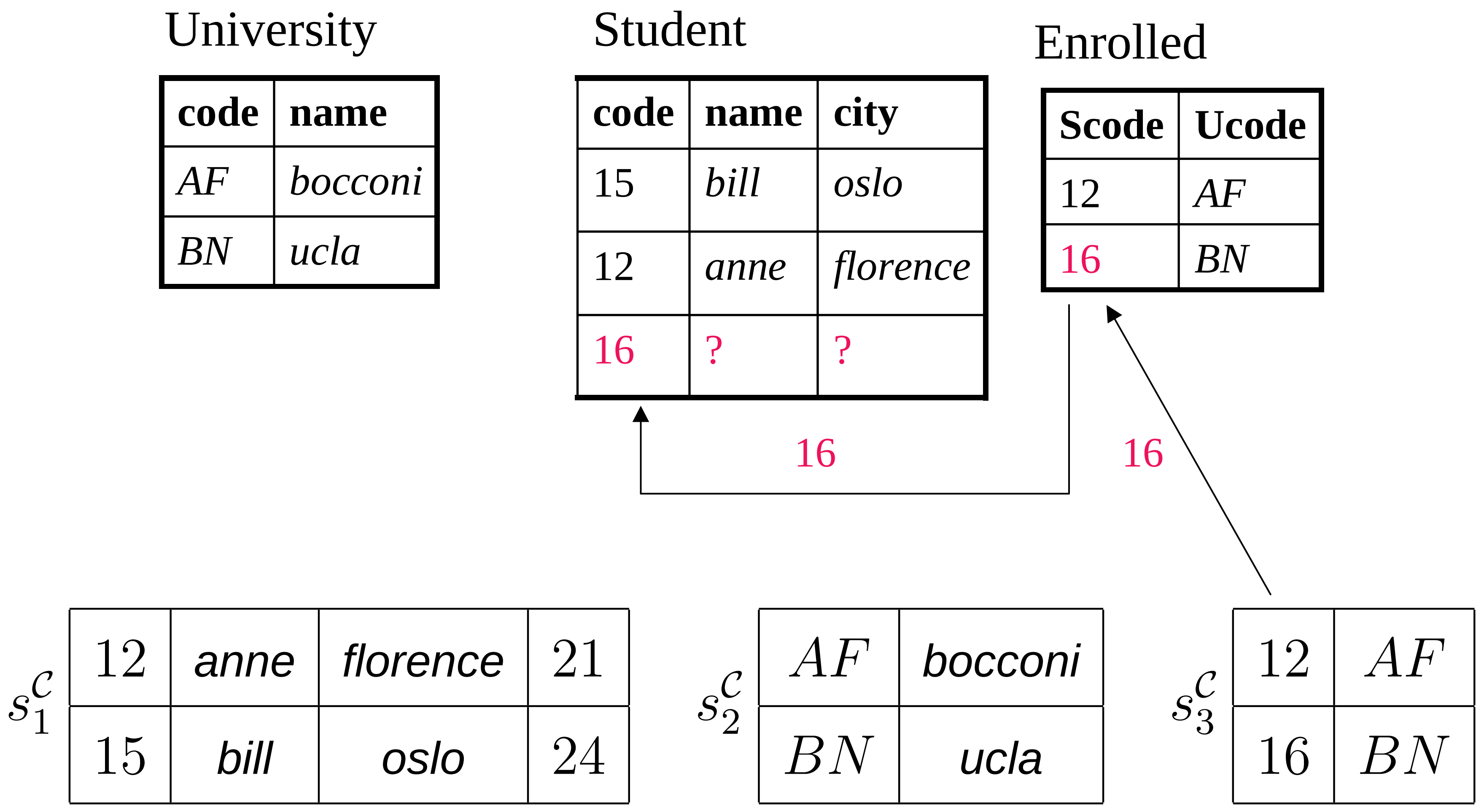
$$\begin{array}{rcl} \text{enrolled}[Scode] & \subseteq & \text{student}[Scode] \\ \text{enrolled}[Ucode] & \subseteq & \text{university}[Ucode] \end{array}$$

Sources $\mathcal{S}$: database relations $\text{s}_1, \text{s}_2, \text{s}_3$

Mapping $\mathcal{M}$:

$$\begin{array}{rcl} \text{student} & \supseteq & \{\, (X,Y,Z) \mid \text{s}_1(X,Y,Z,W)\, \} \\ \text{university} & \supseteq & \{\, (X,Y) \mid \text{s}_2(X,Y)\, \} \\ \text{enrolled} & \supseteq & \{\, (X,W) \mid \text{s}_3(X,W)\, \} \end{array}$$

Constraints in GAV/sound: example



University

code	**name**
AF	*bocconi*
BN	*ucla*

Student

code	**name**	**city**
15	*bill*	*oslo*
12	*anne*	*florence*
16	?	?

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 and corresponding retrieved global database

Constraints in GAV/sound: example

Source database $\mathcal{C}$:

$s_1^{\mathcal{C}}$

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

$s_2^{\mathcal{C}}$

AF	*bocconi*
BN	*ucla*

$s_3^{\mathcal{C}}$

12	AF
16	BN

$s_3^{\mathcal{C}}(16, BN)$ implies enrolled$^{\mathcal{B}}(16, BN)$, for all $\mathcal{B} \in sem^{\mathcal{C}}(\mathcal{I})$.

Due to the integrity constraints in the global schema, 16 **is the code of some student** in all $\mathcal{B} \in sem^{\mathcal{C}}(\mathcal{I})$.

Since $\mathcal{C}$ says nothing about the name and the city of the student with code 16, we must accept as legal for $\mathcal{I}$ wrt $\mathcal{C}$ all virtual global databases that differ in such attributes.

INT[constr, GAV/sound]: unfolding is not sufficient

Mapping $\mathcal{M}$:

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

$s_1^{\mathcal{C}}$

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

$s_2^{\mathcal{C}}$

AF	*bocconi*
BN	*ucla*

$s_3^{\mathcal{C}}$

12	AF
16	BN

Query: $\{\ (X) \mid \textsf{student}(X,Y,Z), \textsf{enrolled}(X,W)\ \}$

Unfolding wrt $\mathcal{M}$: $\{\ (X) \mid s_1(X,Y,Z,V), s_3(X,W)\ \}$

retrieves only the answer $\{12\}$ from $\mathcal{C}$, although $\{12, 16\}$ is the correct answer. The simple unfolding strategy is **not sufficient** in our context.

INT[constr, GAV/sound]: special case

We assume that only key and foreign key constraints are in $\mathcal{G}$, and $\mathcal{M}$ does not violate any key constraint of $\mathcal{G}$ (see later), and we associate to $\mathcal{G}$ a logic program $\mathcal{P}_{\mathcal{G}}$, as follows.

- For each g in $\mathcal{G}$ we have a rule in $\mathcal{P}_{\mathcal{G}}$ of the form:

$$\mathsf{g}'(X_1,\ldots,X_n) \leftarrow \mathsf{g}(X_1,\ldots,X_n)$$

- For each foreign key constraint

$$\mathsf{g}_1[\mathbf{A}] \subseteq \mathsf{g}_2[\mathbf{B}]$$

in $\mathcal{G}$ where $\mathbf{A}$ and $\mathbf{B}$ are sets of attributes, we have a rule in $\mathcal{P}_{\mathcal{G}}$ of the form (the f_i's are fresh Skolem functions):

$$\mathsf{g}_2'(X_1,\ldots,X_h,f_1(X_1,\ldots,X_h),\ldots,f_{n-h}(X_1,\ldots,X_h)) \leftarrow \mathsf{g}_1'(X_1,\ldots,X_h,\ldots,X_m)$$

INT[constr, GAV/sound]: special case

Techniques for processing a conjunctive query q posed to $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$:

- We construct $\mathcal{P}_{\mathcal{G}}$ from $\mathcal{G}$
- We partially evaluate $\mathcal{P}_{\mathcal{G}}$ wrt q, and we obtain another query $exp_{\mathcal{G}}(q)$, called the expansion of q wrt the constraints of $\mathcal{G}$
- We unfold $exp_{\mathcal{G}}(q)$ wrt $\mathcal{M}$, and obtain a query $unf_{\mathcal{M}}(exp_{\mathcal{G}}(q))$ over the sources
- We evaluate $unf_{\mathcal{M}}(exp_{\mathcal{G}}(q))$ over the source database $\mathcal{C}$

$exp_{\mathcal{G}}(q)$ can be of exponential size wrt $\mathcal{G}$, but the whole process has polynomial time complexity wrt the size of $\mathcal{C}$.

INT[constr, GAV/sound]: example

Suppose we have $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$, with $\mathcal{G}$:

$$\text{person}(Pcode, Age, CityOfBirth)$$

$$\text{student}(Scode, University)$$

$$\text{city}(Name, Major)$$

$$\begin{array}{rcl}
key(\text{person}) & = & \{Pcode\} \\
key(\text{student}) & = & \{Scode\} \\
key(\text{city}) & = & \{Name\} \\
\text{person}[CityOfBirth] & \subseteq & \text{city}[Name] \\
\text{city}[Major] & \subseteq & \text{person}[PCode] \\
\text{student}[SCode] & \subseteq & \text{person}[PCode]
\end{array}$$

INT[constr, GAV/sound]: example

The logic program $\mathcal{P}_{\mathcal{G}}$ is

$$\begin{array}{rcl}
\text{person}'(X,Y,Z) & \leftarrow & \text{person}(X,Y,Z) \\
\text{student}'(X,Y) & \leftarrow & \text{student}(X,Y) \\
\text{city}'(X,Y) & \leftarrow & \text{city}(X,Y) \\
\text{city}'(X,f_1(X)) & \leftarrow & \text{person}'(Y,Z,X) \\
\text{person}'(Y,f_2(Y),f_3(Y)) & \leftarrow & \text{city}'(X,Y) \\
\text{person}'(X,f_4(X),f_5(X)) & \leftarrow & \text{student}'(X,Y)
\end{array}$$

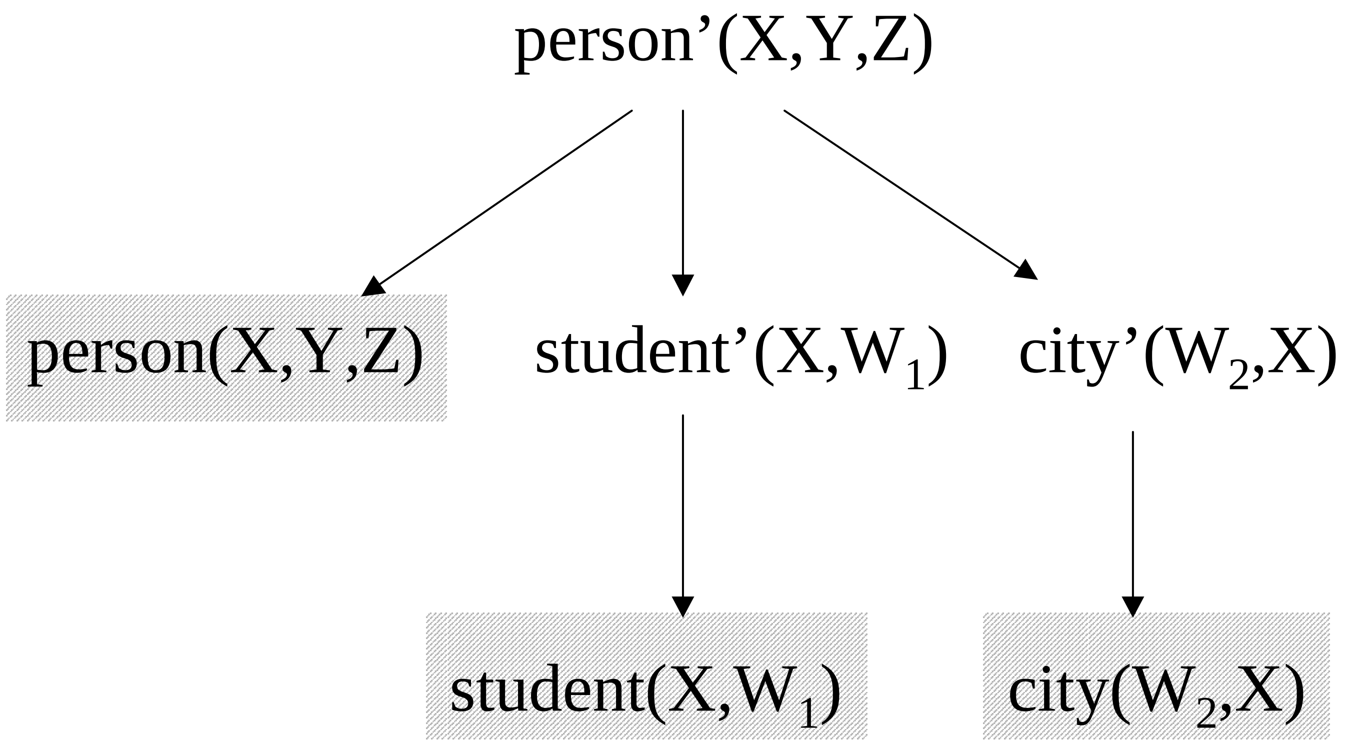
Consider the query

$$\{\ (X) \mid \text{person}(X,Y,Z)\ \}$$

written as the rule

$$\text{q}(X) \leftarrow \text{person}'(X,Y,Z)$$

INT[constr, GAV/sound]: example



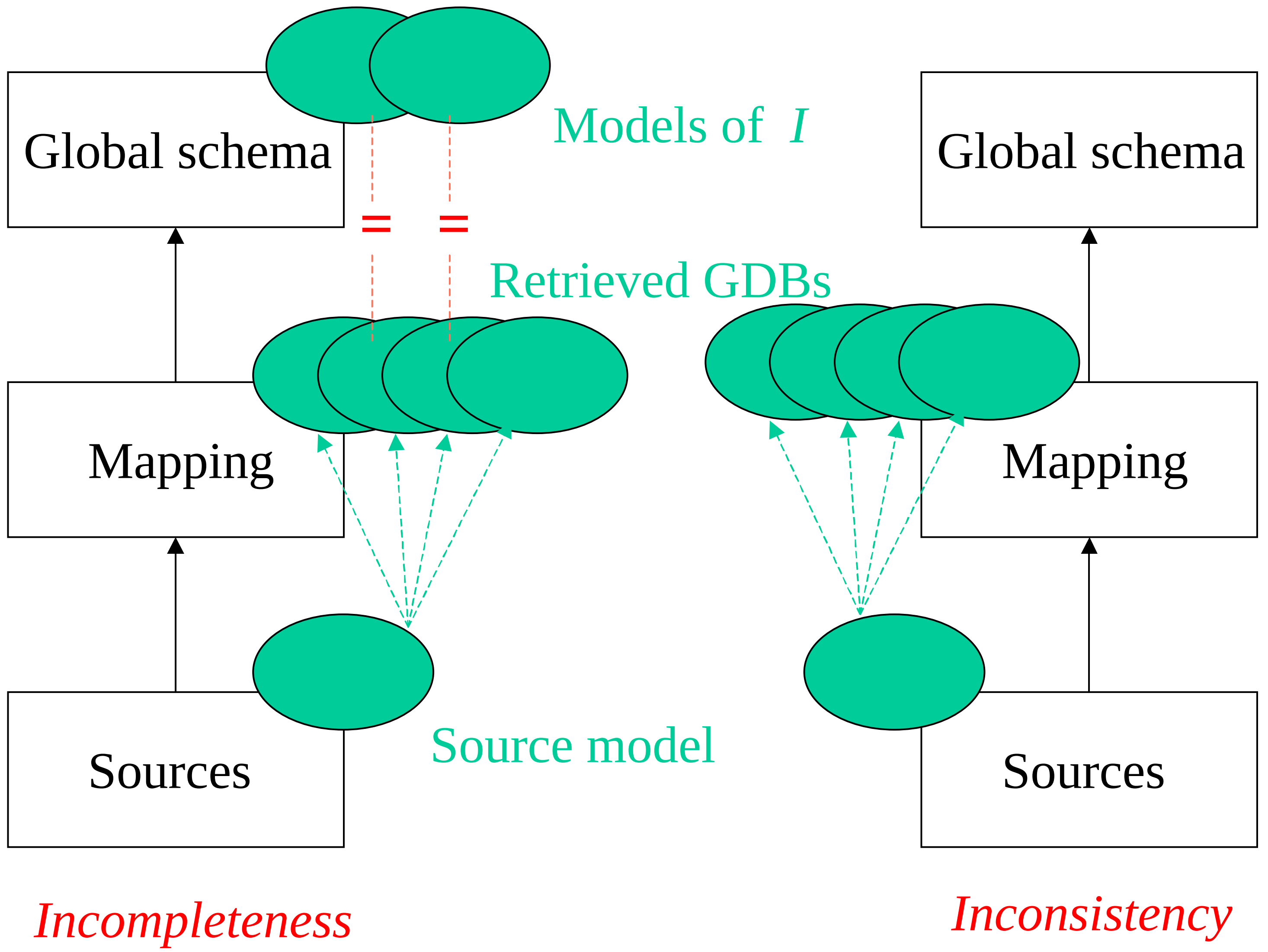
$exp_{\mathcal{G}}(q)$ is

$$\{\ (X) \mid \text{person}(X,Y,Z) \vee \text{student}(X,W) \vee \text{city}(Z,X)\ \}$$

Incompleteness and inconsistency

Constraints in $\mathcal{G}$	Type of mapping	Incomple-teness	Inconsi-stency
no	GAV/exact	no	no
no	GAV/sound	**yes**/no	no
no	LAV/sound	**yes**	no
no	LAV/exact	**yes**	**yes**
yes	GAV/exact	no	**yes**
yes	GAV/sound	**yes**	**yes**
yes	*LAV/sound*	**yes**	**yes**
yes	LAV/exact	**yes**	**yes**

INT[constr, LAV/sound]



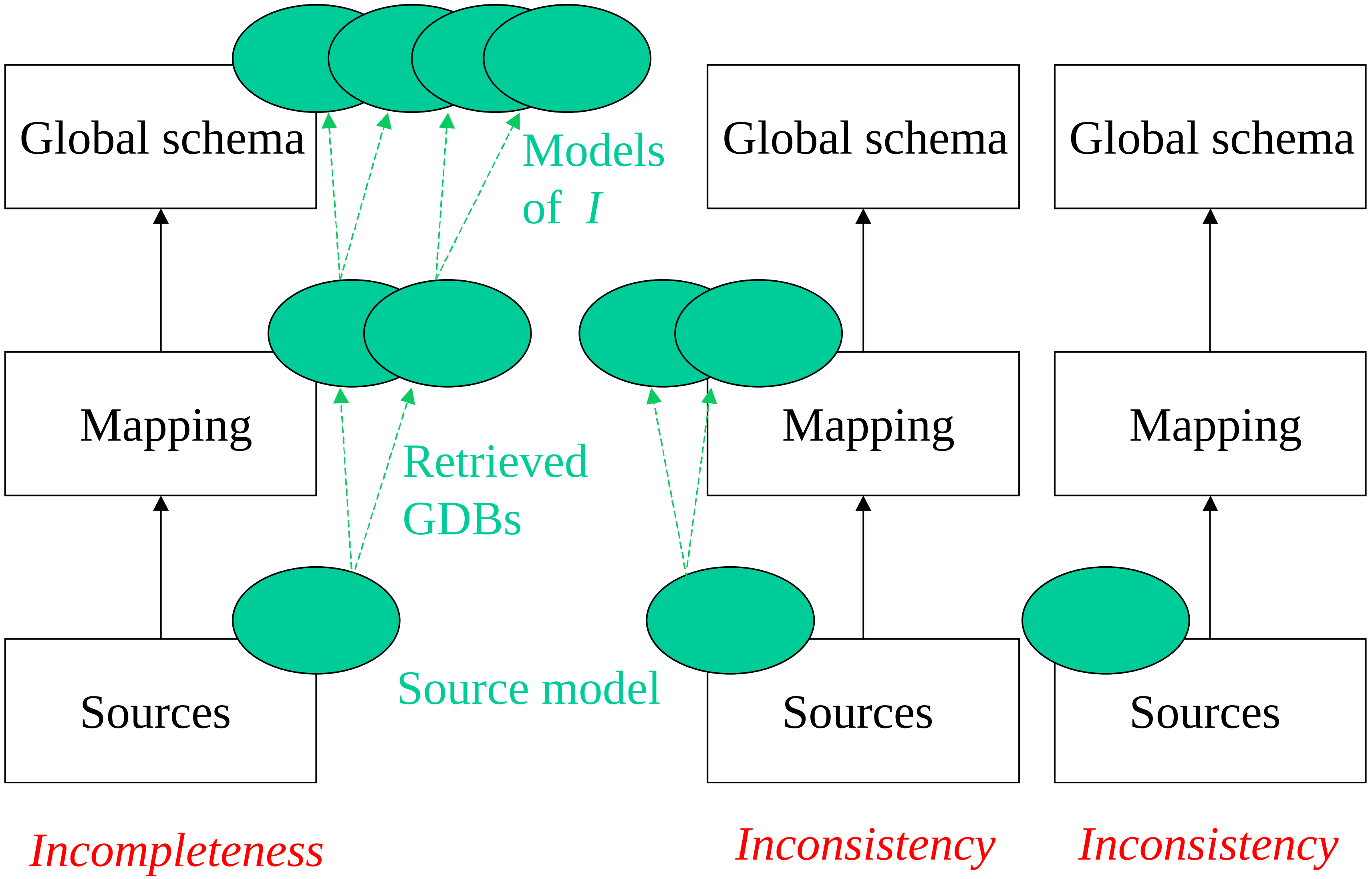
INT[constr, LAV/sound]

- With functional dependencies [Duschka’97]
- With full dependencies [Duschka’97]
- With inclusion dependencies [Gryz’97]
- With Description Logics integrity constraints [Calvanese&al. AAAI’00]

Incompleteness and inconsistency

Constraints in $\mathcal{G}$	Type of mapping	Incomple-teness	Inconsi-stency
no	GAV/exact	no	no
no	GAV/sound	**yes**/no	no
no	LAV/sound	**yes**	no
no	LAV/exact	**yes**	**yes**
yes	GAV/exact	no	**yes**
yes	GAV/sound	**yes**	**yes**
yes	LAV/sound	**yes**	**yes**
yes	*LAV/exact*	**yes**	**yes**

INT[constr, LAV/exact]



INT[constr, LAV/exact]

- With Description Logics integrity constraints [Calvanese&al. AAAI’00]
- Largely unexplored problem

Outline

- Formal framework for data integration
- Approaches to data integration
- Query answering in different approaches
- Dealing with inconsistency
- Reasoning on queries in data integration
- Conclusions

INT[constr, GAV/sound]: Dealing with inconsistency

When for data integration system $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$ and source database $\mathcal{C}$, we have $sem^{\mathcal{C}}(\mathcal{I}) = \emptyset$, the first-order setting described above is **not adequate**.

- [Subrahmanian ACM-TODS'94]
- [Grant&al. IEEE-TKDE'95]
- [Dung CoopIS'96]
- [Lin&al. JICIS'98]
- [Yan&al. CoopIS'99]
- [Arenas&al. PODS'99]
- [Greco&al. LPAR'00]
- many approaches to KB revision and KB/DB update

Beyond first-order logic: example

$$\begin{array}{rcl} key(\text{player}) & = & \{Pcode\} \\ key(\text{team}) & = & \{Tcode\} \\ \text{player}[Pteam] & \subseteq & \text{team}[Tcode] \\ \text{team}[Tleader] & \subseteq & \text{player}[Pcode]. \end{array}$$

$$\begin{array}{rcl} \text{player} & \supseteq & \{\, (X,Y,Z) \mid \mathsf{s}_1(X,Y,Z,W) \,\} \\ \text{team} & \supseteq & \{\, (X,Y,Z) \mid \mathsf{s}_2(X,Y,Z) \vee \mathsf{s}_3(X,Y,Z) \,\} \end{array}$$

$\mathsf{s}_1^{\mathcal{C}}$:

9	*Batistuta*	*RM*	31
10	*Rivaldo*	*BC*	29

$\mathsf{s}_2^{\mathcal{C}}$:

RM	*Roma*	8
BC	*Barcelona*	10

$\mathsf{s}_3^{\mathcal{C}}$:

RM	*Roma*	9

Beyond first-order logic: a proposal

Given

- $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$, with a GAV/sound mapping $\mathcal{M} = \{r_1 \supseteq V_1, \ \ldots \ , r_n \supseteq V_n\}$, and
- source database $\mathcal{C}$ for $\mathcal{S}$,

we would like to focus on those databases for $\mathcal{I}$ that

1. satisfy $\mathcal{G}$ (constraints in $\mathcal{G}$ are rigid), and
2. **approximate as much as possible** the satisfaction of the mapping $\mathcal{M}$ wrt $\mathcal{C}$ (assertions in $\mathcal{M}$ are soft).

Beyond first-order logic: a proposal

We define an ordering between the global databases for $\mathcal{I}$ as follows. If $\mathcal{B}_1$ and $\mathcal{B}_2$ are two databases that satisfy $\mathcal{G}$, we say that $\mathcal{B}_1$ is better than $\mathcal{B}_2$ wrt $\mathcal{I}$ and $\mathcal{C}$, denoted as $\mathcal{B}_1 \gg_{\mathcal{C}}^{\mathcal{I}} \mathcal{B}_2$, if there exists an assertion $r_i \supseteq V_i$ in $\mathcal{M}$ such that

- $(r_i^{\mathcal{B}_1} \cap V_i^{\mathcal{C}}) \supset (r_i^{\mathcal{B}_2} \cap V_i^{\mathcal{C}})$, and
- $(r_j^{\mathcal{B}_1} \cap V_j^{\mathcal{C}}) \supseteq (r_j^{\mathcal{B}_2} \cap V_j^{\mathcal{C}})$ for all $r_j \leadsto_s V_j$ in $\mathcal{M}$ with $j \neq i$.

Intuitively, $\mathcal{B}_1$ has fewer deletions than $\mathcal{B}_2$ wrt the retrieved global database (see [Fagin&al. PODS'83]), and since the mapping is sound, this means that $\mathcal{B}_1$ is closer than $\mathcal{B}_2$ to the retrieved global database. In other words, $\mathcal{B}_1$ approximates the sound mapping better than $\mathcal{B}_2$.

Example

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

- $\mathcal{G}$ containing relation $r(x, y)$ with key x,
- $\mathcal{S}$ containing relations $s_1(x, y)$ and $s_2(x, y)$
- $\mathcal{M} = \{ \ r \ \supseteq \ \{ \ (x, y) \mid s_1(x, y) \vee s_2(x, y) \ \} \ \}$

and consider the source database $\mathcal{C} = \{ \ s_1(a, d), \ s_1(b, d), \ s_2(a, e) \ \}$, so that the retrieved global database is $\{ \ r(a, d), \ r(b, d) \ , \ r(a, e) \ \}$

We have that

- $\{ \ r(a, d), \ r(b, d) \ \} \gg^{\mathcal{I}}_{\mathcal{C}} \{ \ r(a, d) \ \}, \quad \{ \ r(a, e), \ r(b, d) \ \} \gg^{\mathcal{I}}_{\mathcal{C}} \{ \ r(a, e) \ \}$
- $\{ \ r(a, d), \ r(b, d) \ \}$ and $\{ \ r(a, e) \ \}$ are incomparable
- $\{ \ r(a, e), \ r(b, d), \ r(c, e) \ \}$ and $\{ \ r(a, e), \ r(b, d) \ \}$ are incomparable

Beyond first-order logic: a proposal

$\gg^{\mathcal{I}}_{\mathcal{C}}$ is a partial order.

A database $\mathcal{B}$ that satisfy $\mathcal{G}$ satisfies the mapping $\mathcal{M}$ with respect to $\mathcal{C}$ if $\mathcal{B}$ is maximal wrt $\gg^{\mathcal{I}}_{\mathcal{C}}$, i.e., for no other global database $\mathcal{B}'$ that satisfies $\mathcal{G}$, we have that $\mathcal{B}' \gg^{\mathcal{I}}_{\mathcal{C}} \mathcal{B}$:

$$sem^{\mathcal{C}}(\mathcal{I}) = \{\ \mathcal{B} \ \mid \ \mathcal{B} \text{ is a database that satisfies } \mathcal{G}, \text{ and such that } \neg\exists \mathcal{B}' \text{ such that } \mathcal{B}' \text{ satisfies } \mathcal{G} \text{ and } \mathcal{B}' \gg^{\mathcal{I}}_{\mathcal{C}} \mathcal{B} \ \}$$

The notion of legal database for $\mathcal{I}$ with respect to $\mathcal{C}$, and the notion of certain answer remain the same, given the new definition of satisfaction of mapping.

Beyond first-order logic: special case of INT[constr, GAV/sound]

We assume that only key and foreign key constraints are in $\mathcal{G}$. Given $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$, and source database $\mathcal{C}$, we define the DATALOG$^\neg$ program $\mathcal{P}(\mathcal{I}, \mathcal{C})$ obtained by adding to the set of facts $\mathcal{C}$ the following set of rules:

- for each $\mathsf{g} \supseteq \{(\vec{\mathbf{x}}) \mid body_1(\vec{\mathbf{x}}, \vec{\mathbf{y}}_1) \vee \cdots \vee body_m(\vec{\mathbf{x}}, \vec{\mathbf{y}}_m)\}$ in $\mathcal{M}$, the rules:

$$\mathsf{g}_{\mathcal{C}}(\vec{\mathbf{X}}) \leftarrow body_1(\vec{\mathbf{X}}, \vec{\mathbf{Y}}_1) \quad \ldots \quad \mathsf{g}_{\mathcal{C}}(\vec{\mathbf{X}}) \leftarrow body_m(\vec{\mathbf{X}}, \vec{\mathbf{Y}}_m)$$

- for each relation $g \in \mathcal{G}$, the rules

$$\begin{array}{lcl} \mathsf{g}(\vec{\mathbf{X}}, \vec{\mathbf{Y}}) & \leftarrow & \mathsf{g}_{\mathcal{C}}(\vec{\mathbf{X}}, \vec{\mathbf{Y}})\ ,\ not\ \bar{\mathsf{g}}(\vec{\mathbf{X}}, \vec{\mathbf{Y}}) \\ \bar{\mathsf{g}}(\vec{\mathbf{X}}, \vec{\mathbf{Y}}) & \leftarrow & \mathsf{g}(\vec{\mathbf{X}}, \vec{\mathbf{Z}})\ ,\ \vec{\mathbf{Y}} \neq \vec{\mathbf{Z}} \end{array}$$

  - in $\mathsf{g}(\vec{\mathbf{X}}, \vec{\mathbf{Y}})$, $\vec{\mathbf{X}}$ is the key of g
  - $\vec{\mathbf{Y}} \neq \vec{\mathbf{Z}}$ means that there exists i such that $Y_i \neq Z_i$.

Beyond first-order logic: a proposal

The above rules force each stable model T of $\mathcal{P}(\mathcal{I},\mathcal{C})$ to be such that, for each g in $\mathcal{G}$, g^T is a maximal subset of the tuples from the retrieved global database that are consistent with the key constraint for g.

- $t \in q^{\mathcal{I},\mathcal{C}}$ under the new semantics if and only if $t \in q^T$ for each stable model T of the DATALOG$^\neg$ program $\mathcal{P}(\mathcal{I},\mathcal{C}) \cup \{exp_{\mathcal{G}}(q)\}$
- A stable model of a DATALOG$^\neg$ program Π is any set σ of ground atoms that coincides with the unique minimal Herbrand model of the DATALOG progam Π_σ, where Π_σ is obtained from Π by deleting every rule that has a negative literal $\neg B$ with $B \in \sigma$, and all negative literals in the bodies of the remaining rules
- The problem of deciding whether $t \in q^{\mathcal{I},\mathcal{C}}$ is in coNP wrt data complexity
- coNP-complete

Outline

- Formal framework for data integration
- Approaches to data integration
- Query answering in different approaches
- Dealing with inconsistency
- Reasoning on queries in data integration
- Conclusions

Reasoning on queries and views in data integration

Traditional query containment not adequate.

Global schema: movie$(\mathit{Title}, \mathit{Year}, \mathit{Director})$
review$(\mathit{Title}, \mathit{Critique})$

Mapping:

$$\mathsf{r}_1(T, Y, D) \subseteq \{ (T, Y, D) \mid \text{movie}(T, Y, D) \wedge Y \geq 1960 \}$$

$$\mathsf{r}_2(T, R) \subseteq \{ (T, R) \mid \text{review}(T, R) \wedge R \geq 8 \}$$

Queries: $Q_1 : \{ (T, R) \mid \text{movie}(T, 1998, D) \wedge \text{review}(T, R) \}$

$Q_2 : \{ (T, R) \mid \text{movie}(T, 1998, D) \wedge \text{review}(T, R) \wedge R \geq 8 \}$

Q_1 is not contained in Q_2 in the traditional sense, but is contained in Q_2 relative to $\mathcal{I}$.

Relative containment

[Millstein&al. PODS'00] Given data integration system $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$, a query Q_1 is said to be contained in query Q_2 relative to $\mathcal{I}$ (written $Q_1 \subseteq_{\mathcal{I}} Q_2$) if, for every source database $\mathcal{C}$, the set of certain answers to Q_1 wrt $\mathcal{I}$ and $\mathcal{C}$ is contained in the set of certain answers to Q_2 wrt $\mathcal{I}$ and $\mathcal{C}$, i.e., if

$$\forall \mathcal{C}, \mathit{cert}(Q_1, \mathcal{I}, \mathcal{C}) \subseteq \mathit{cert}(Q_2, \mathcal{I}, \mathcal{C})$$

For LAV/sound systems with conjunctive queries in the mapping, deciding relative containment of two conjunctive queries is Π_2^p-complete [Millstein&al. PODS'00].

Lossless views

Given LAV data integration system $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$, and query Q, $\mathcal{I}$ is said to be **lossless** wrt Q if, for every global database $\mathcal{B}$ for $\mathcal{I}$ and for every source database $\mathcal{C}$ such that $\mathcal{B}$ is legal for $\mathcal{I}$ wrt $\mathcal{C}$, we have that $Q^{\mathcal{B}} = Q^{\mathcal{I},\mathcal{C}}$.

If $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$ is lossless wrt Q, then answering Q through the sources of $\mathcal{I}$ (views) is the same as answering Q by accessing the global database.

Note the difference with checking whether the maximally contained rewriting of Q wrt to $\mathcal{I}$ is equivalent to Q.

Comparing the expressive power of sets of views

A set of views V is **p-contained** in another set of views W if all queries that are answerable by V are also answerable by W [Li&al. ICDT'01].

A query is answerable by a set of views V if there is an equivalent rewriting of Q using V.

Given LAV data integration systems $\mathcal{I}_1 = \langle \mathcal{G}, \mathcal{S}_1, \mathcal{M}_1 \rangle$ and $\mathcal{I}_2 = \langle \mathcal{G}, \mathcal{S}_2, \mathcal{M}_2 \rangle$, **$\mathcal{I}_1$ is p-contained in $\mathcal{I}_2$** if, for each query Q, $cert_{[Q,\mathcal{I}_1]}$ equivalent to Q implies $cert_{[Q,\mathcal{I}_2]}$ equivalent to Q.

Outline

- Formal framework for data integration
- Approaches to data integration
- Query answering in different approaches
- Dealing with inconsistency
- Reasoning on queries in data integration
- Conclusions

Conclusions

Many open problems, including

- P2P data integration
- Several interesting classes of integrity constraints
- Global schema expressed in terms of semi-structured data (with constraints)
- Dealing with inconsistencies, data cleaning
- How to go beyond the “unique domain assumption”
- Limitations in accessing the sources
- How to incorporate the notion of data quality (source reliability, accuracy, etc.)
- More on reasoning on queries and views
- Optimization

Acknowledgements

Special thanks to

- **Andrea Calí**
- **Diego Calvanese**
- **Giuseppe De Giacomo**
- **Domenico Lembo**
- **Riccardo Rosati**
- **Moshe Y. Vardi**