View-based Query Processing over Semistructured Data

Diego Calvanese

Free University of Bozen-Bolzano

View-based Query Processing

Diego Calvanese, Giuseppe De Giacomo, Georg Gottlob,

Maurizio Lenzerini, Riccardo Rosati

Corso di Dottorato – Dottorato in Ingegneria Informatica

University of Rome "La Sapienza"

September–October 2005

VBQP over Semistructured Data – Outline

1. The semistructured data model and regular path queries (RPQs)
2. Containment of RPQs and 2RPQs
3. View-based query rewriting for RPQs and 2RPQs
4. View-based query answering for RPQs and 2RPQs via automata
5. View-based query answering for RPQs and 2RPQs via CSP

VBQP over Semistructured Data – Outline

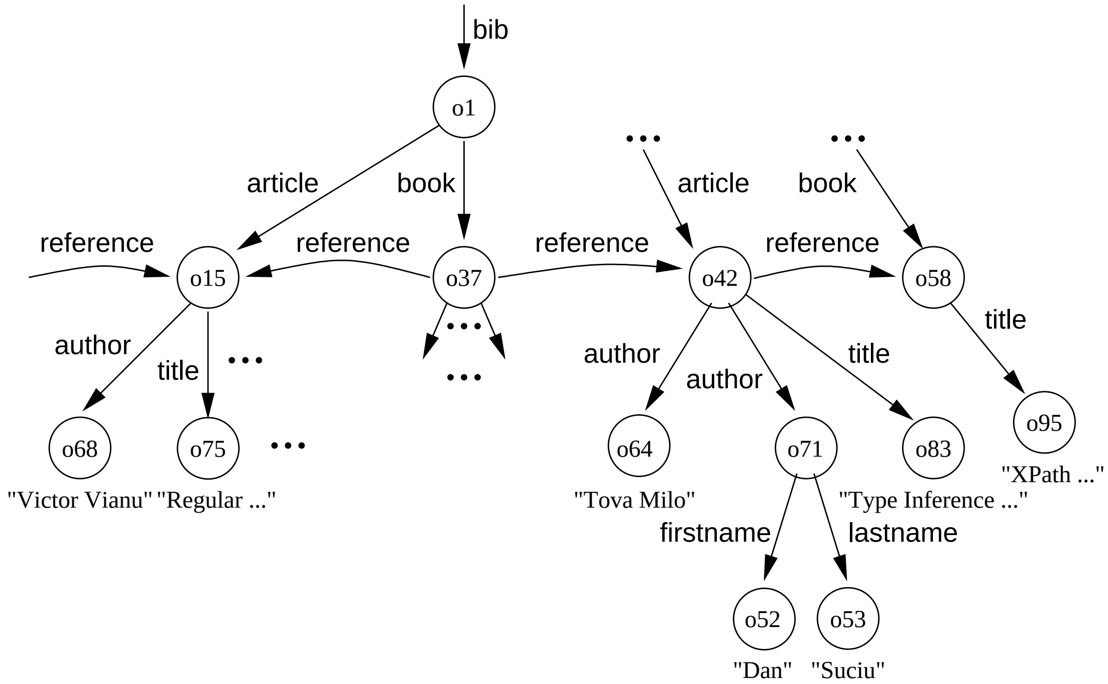
$\Rightarrow$ The semistructured data model and regular path queries (RPQs)

2. Containment of RPQs and 2RPQs
3. View-based query rewriting for RPQs and 2RPQs
4. View-based query answering for RPQs and 2RPQs via automata
5. View-based query answering for RPQs and 2RPQs via CSP

Semistructured data

Semistructured data (SSD) are an abstraction for data on the web, structured documents, XML:

- A (semistructured) database (DB) is a (finite) edge-labeled graph



- In some cases there are restrictions on the structure of the graph, e.g., in XML the graph has to be a tree

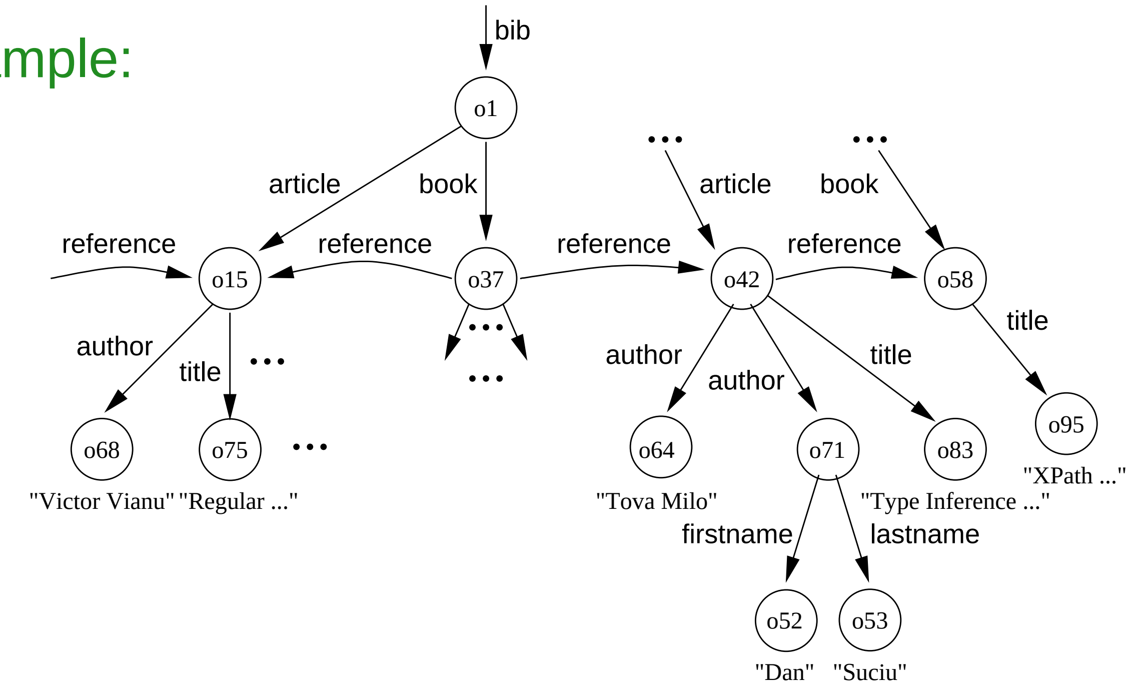
Formalization of semistructured databases

Definition:

- The schema of a DB is a relational alphabet Σ of binary predicates (one for each edge label).
- A SSDB is a set of binary relations.

Note that we do not allow for constraints over the relations in a SSDB.

Example:



$$\Sigma = \{\textbf{bib}, \textbf{article}, \textbf{book}, \textbf{reference}, \textbf{title}, \textbf{author}, \ldots\}$$

Queries over SSD

Queries over SSD are typically constituted by two parts:

- selection part: selects (tuples of) nodes that satisfy some condition
- restructuring part: reorganizes the selected nodes into a graph (or tree)

In this course we deal with the selection part only.

Queries must provide the ability to “navigate” the graph structure to relate pairs of nodes $\leadsto$ must contain some form of recursion:

- Datalog: provides a very expressive form of recursion
- XPath: descendant/ancestor axes refer to successor/predecessor nodes at arbitrary depth in the tree – rather restricted form of recursion
- reflexive transitive closure provides a good tradeoff

Path queries

Are the basic element of all proposals for query languages over SSD.

Definition: A path query Q has the form

$$Q(x, y) \leftarrow x \; L \; y$$

where L is a language over the alphabet Σ of binary DB predicates.

Recall that a DB $\mathcal{B}$ is a set of (binary) relations over Σ, or equivalently a graph whose edges are labeled with elements of Σ.

Definition: The answer $Q(\mathcal{B})$ to Q over $\mathcal{B}$ is the set of pairs of nodes (a, b) such that there is a path $a \xrightarrow{p_1} \cdots \xrightarrow{p_k} b$ in $\mathcal{B}$, with $p_1 \cdots p_k \in L$.

Notable example: regular path queries (RPQs), in which L is a regular language over Σ

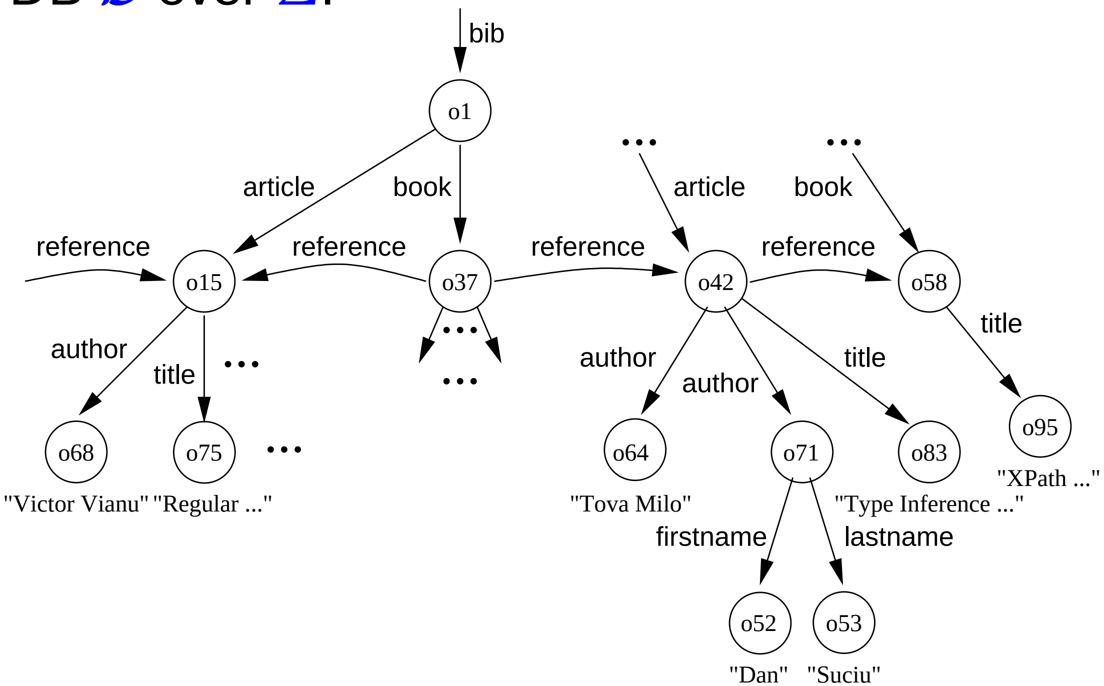
Regular path queries

In an RPQ, we can specify the regular language through a regular expression.

Example: DB alphabet: $\Sigma = \{\mathbf{bib}, \mathbf{article}, \mathbf{book}, \mathbf{reference}, \mathbf{title}, \ldots\}$

Query: $Q(x, y) \leftarrow x \; ((\mathbf{article} + \mathbf{book}) \cdot \mathbf{reference}^* \cdot \mathbf{title}) \; y$

Consider the DB $\mathcal{B}$ over Σ:



$Q(\mathcal{B})$

o_1	o_{75}
o_1	o_{83}
o_1	o_{95}
$\vdots$	$\vdots$

Regular path queries – Observations

Expressive power of RPQs:

- Not expressible in first-order logic
- Are a fragment of transitive-closure logic
- Are a fragment of binary Datalog
  - Concatenation: $P(x,y) \;\leftarrow\; E_1(x,z), E_2(z,y)$
  - Union: $$\begin{array}{lcl} P(x,y) & \leftarrow & E_1(x,y) \\ P(x,y) & \leftarrow & E_2(x,y) \end{array}$$
  - Reflexive-transitive closure: $$\begin{array}{lcl} P(x,y) & \leftarrow & E(x,y) \\ P(x,y) & \leftarrow & E(x,z), P(z,y) \end{array}$$

VBQP over Semistructured Data – Outline

$\surd$ The semistructured data model and regular path queries (RPQs)

$\Rightarrow$ Containment of RPQs and 2RPQs

3. View-based query rewriting for RPQs and 2RPQs
4. View-based query answering for RPQs and 2RPQs via automata
5. View-based query answering for RPQs and 2RPQs via CSP

Path query containment

Given $Q_1(x, y) \leftarrow x\, L_1\, y$

$Q_2(x, y) \leftarrow x\, L_2\, y$

check whether $Q_1 \sqsubseteq Q_2$, i.e., for every DB $\mathcal{B}$, we have $Q_1(\mathcal{B}) \subseteq Q_2(\mathcal{B})$.

Language-Theoretic Lemma 1: $Q_1 \sqsubseteq Q_2$ iff $L_1 \subseteq L_2$

Proof: "Only if": Consider a DB $a \xrightarrow{p_1} \cdots \xrightarrow{p_k} b$ with $p_1 \cdots p_k \in L_1$ and $p_1 \cdots p_k \notin L_2$.

"If": If $(a, b) \in Q_1(\mathcal{B})$, then $\mathcal{B}$ contains a path $a \xrightarrow{p_1} \cdots \xrightarrow{p_k} b$ with $p_1 \cdots p_k \in L_1$. But then $p_1 \cdots p_k \in L_2$, and $(a, b) \in Q_2(\mathcal{B})$.

Corollary: Path query containment is

- undecidable for context-free path queries
- PSPACE-complete for regular path queries [Stockmeyer 1973]

Containment of RPQs

Via language containment. We exploit that $L_1 \subseteq L_2$ iff $L_1 - L_2 = \emptyset$.

Algorithm for checking whether $\mathcal{L}(E_1) \subseteq \mathcal{L}(E_2)$ (for regular expr. E_1, E_2)

1. Construct NFAs A_i such that $\mathcal{L}(A_i) = \mathcal{L}(E_i)$
   $\leadsto$ linear blowup
2. Construct NFA $\overline{A_2}$ such that $\mathcal{L}(\overline{A_2}) = \Sigma^* - \mathcal{L}(A_2)$
   $\leadsto$ exponential blowup
3. Construct $A = A_1 \times \overline{A_2}$ such that $\mathcal{L}(A) = \mathcal{L}(E_1) - \mathcal{L}(E_2)$
   $\leadsto$ quadratic blowup
4. Check whether there is a path from the initial state to a final state in A
   $\leadsto$ NLOGSPACE

Theorem: Containment of RPQs is in PSPACE, and hence PSPACE-complete.

Two-way regular path queries (2RPQs)

- Provide the ability to navigate DB edges in both directions. Allow one to capture, e.g., the predecessor axis of XPath.
- We introduce an extended alphabet $\Sigma^{\pm} = \Sigma \cup \Sigma^{-}$, where $\Sigma^{-} = \{p^{-} \mid p \in \Sigma\}$.
- We call the elements of Σ^{-} inverse edge labels.

Definition: A two-way regular path query over Σ has the form

$$Q(x, y) \leftarrow x \; E \; y$$

with E a regular expression over the extended alphabet $\Sigma^{\pm}$.

Example: $Q_2(x, y) \leftarrow x \; (\mathbf{article} \cdot (\mathbf{reference} + \mathbf{reference}^{-})^{*} \cdot \mathbf{title}) \; y$

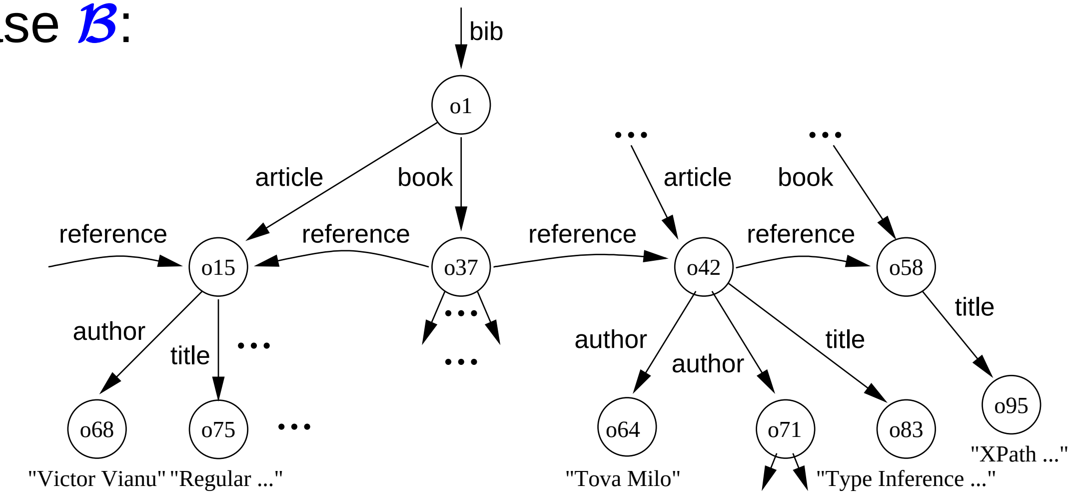
Note: the edges of the DB are still labeled with elements of Σ only.

Semantics of two-way regular path queries

- Consider a 2RPQ $Q(x, y) \leftarrow x\ E\ y$ and a DB $\mathcal{B}$ over Σ.
- A semi-path $a_1 \xrightarrow{r_1} a_2 \cdots a_k \xrightarrow{r_k} a_{k+1}$ in $\mathcal{B}$ is a sequence of nodes with:
  - either $a_i \xrightarrow{p_i} a_{i+1}$ in $\mathcal{B}$, and $r_i = p_i$,
  - or $a_{i+1} \xrightarrow{p_i} a_i$ in $\mathcal{B}$, and $r_i = p_i^-$.
- The answer $Q(\mathcal{B})$ to Q over $\mathcal{B}$ is the set of pairs of nodes (a, b) s.t. there is a semi-path $a \xrightarrow{r_1} \cdots \xrightarrow{r_k} b$ in $\mathcal{B}$, with $r_1 \cdots r_k \in \mathcal{L}(E)$.

Example: Query $Q_2(x, y) \leftarrow x\ (\textbf{article}\cdot(\textbf{reference} + \textbf{reference}^-)^*\cdot\textbf{title})\ y$

Database $\mathcal{B}$:



$Q_2(\mathcal{B})$

o_1	o_{75}
o_1	o_{83}
o_1	o_{95}
$\vdots$	$\vdots$

Two-way regular path queries – Observations

Language-Theoretic Lemma 1 does not hold anymore.

Reason: sequences of direct and inverse edge labels may be "folded" away.

Example: $\Sigma = \{P\}$

$$Q_1(x, y) \leftarrow x\ P\ y$$

$$Q_2(x, y) \leftarrow x\ P \cdot P^- \cdot P\ y$$

We have that:

- $Q_1 \sqsubseteq Q_2$: consider any path $a \xrightarrow{P} b$ in a DB $\mathcal{B}$
- but $\mathcal{L}(P) \not\subseteq \mathcal{L}(P \cdot P^- \cdot P)$.

Foldings

Definition: Let u, v be words over $\Sigma^{\pm}$. We say that v folds onto u, denoted $v \rightsquigarrow u$, if we can transform v into u by repeatedly:

- replacing each occurrence in v of $p \cdot p^{-} \cdot p$ with p, and
- replacing each occurrence in v of $p^{-} \cdot p \cdot p^{-}$ with p^{-}.

Example: $rss^{-}st \rightsquigarrow rst$

Pictorially: $\xrightarrow{r} \cdot \xrightarrow{s} \cdot \xleftarrow{s} \cdot \xrightarrow{s} \cdot \xrightarrow{t} \quad \rightsquigarrow \quad \xrightarrow{r} \cdot \xrightarrow{s} \cdot \xrightarrow{t}$

Definition: Let E be a RE over $\Sigma^{\pm}$.
Then $fold(E) = \{v \mid v \rightsquigarrow u, \text{ for some } u \in \mathcal{L}(E)\}$

The notion of folding allows us to reduce containment of 2RPQs to a language-theoretic problem.

Containment of 2RPQs

Consider two 2RPQs $$Q_1(x,y) \leftarrow x\; E_1\; y$$ $$Q_2(x,y) \leftarrow x\; E_2\; y$$

Language-Theoretic Lemma 2: $Q_1 \sqsubseteq Q_2$ iff $\mathcal{L}(E_1) \subseteq fold(E_2)$

Proof: by considering simple semi-paths $a \xrightarrow{r_1} \cdots \xrightarrow{r_k} b$ in a DB, where $r_1 \cdots r_k \in \mathcal{L}(E_1)$.

To decide $\mathcal{L}(E_1) \subseteq fold(E_2)$ we resort to two-way automata on words.

Two-way automata on words (2NFA)

A 2NFA is similar to a standard one-way automaton (1NFA)

$$A = (\Sigma, S, S_0, \delta, F)$$

but the transition function $\delta : S \times \Sigma \to 2^{S \times \{-1,0,1\}}$ maps each state to a set of pairs

- new state
- moving direction (left, don't move, or right)

Theorem [Rabin&Scott, Shepherdson 1959]: 2NFAs accept regular languages

Given a 2NFA A with n states, one can construct a 1NFA with $O(2^{n \log n})$ states accepting $\mathcal{L}(A)$.

2NFAs and foldings

Theorem: Let E be a RE over $\Sigma^{\pm}$. There is a 2NFA $\tilde{A}_E$ such that

- $\mathcal{L}(\tilde{A}_E) = fold(E)$
- The number of states of $\tilde{A}_E$ is linear in the size of E

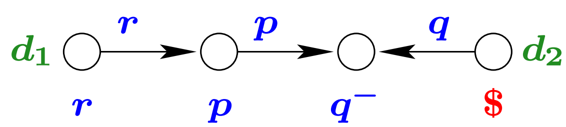
In the construction of $\tilde{A}_E$ we exploit the fact that 2NFAs can move backwards on a word. E.g., to fold pp^-p onto p, the 2NFA:

1. Moves forward on p.
2. Makes a step backward and expects to see p while staying in place (this corresponds to moving according to p^-, i.e., backward on p).
3. Moves forward again on p.

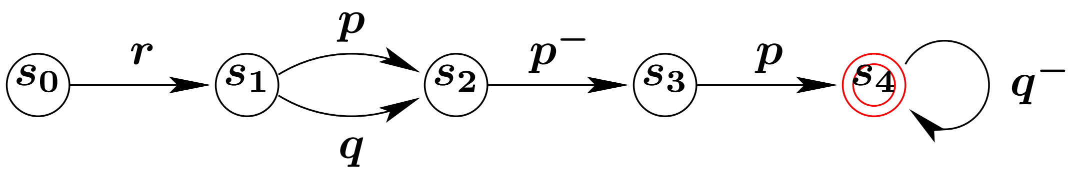
2NFAs and foldings – Example

Regular expression over $\Sigma^{\pm}$: $\quad \boldsymbol{E} = r \cdot (p + q) \cdot p^- \cdot p \cdot q^{-*}$

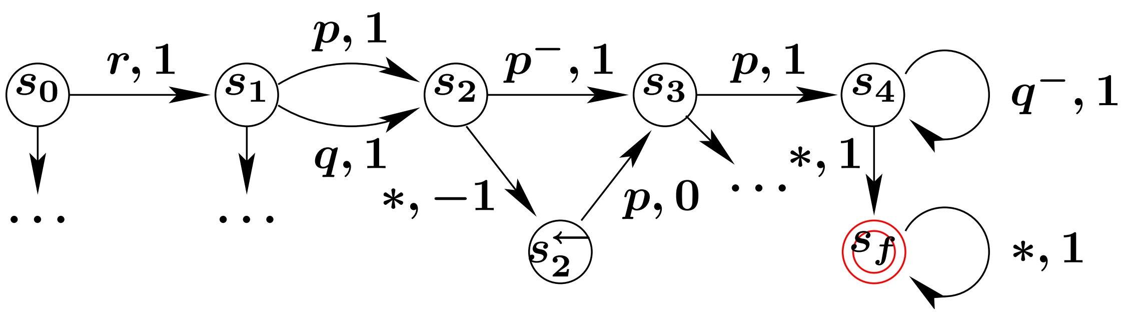
Word in $\mathcal{L}(\boldsymbol{E})$ viewed as a path in a DB:



1NFA that accepts $\mathcal{L}(\boldsymbol{E})$



2NFA that accepts $fold(\boldsymbol{E})$



2NFAs and foldings – Construction

Let E be a RE over $\Sigma^{\pm}$ and $A = (\Sigma^{\pm}, S, S_0, \delta, F)$ a 1NFA with $\mathcal{L}(A) = \mathcal{L}(E)$.

We construct the 2NFA

$$\tilde{A}_E \;=\; (\Sigma^{\pm} \cup \{\$\}, \quad S \cup \{s_f\} \cup \{s^{\leftarrow} \mid s \in S\}, \quad S_0, \quad \delta_A, \quad \{s_f\})$$

where δ_A is defined as follows:

- $(s^{\leftarrow}, -1) \in \delta_A(s, \ell)$, for each $s \in S$ and $\ell \in \Sigma^{\pm} \cup \{\$\}$
- $(s_2, 1) \in \delta_A(s_1, r)$ and $(s_2, 0) \in \delta_A(s_1^{\leftarrow}, r^{-})$, for each transition $s_2 \in \delta(s_1, r)$ of E.
- $(s_f, 1) \in \delta_A(s, \ell)$, for each $s \in F$ and $\ell \in \Sigma^{\pm} \cup \{\$\}$

We have that: $\quad w \in \mathit{fold}(E) \quad$ iff $\quad w\$ \in \mathcal{L}(\tilde{A}_E)$

(We can also get rid of the \$ at the end of words in $\mathcal{L}(\tilde{A}_E)$.)

Containment of 2RPQs (cont'd)

Consider two 2RPQs $Q_1(x, y) \leftarrow x\ E_1\ y$

$Q_2(x, y) \leftarrow x\ E_2\ y$

Summing what we have seen till now, we have that

$$
\begin{array}{ll}
Q_1 \sqsubseteq Q_2 & \text{iff} \\
\mathcal{L}(E_1) \subseteq fold(E_2) & \text{iff} \\
\mathcal{L}(E_1) \subseteq \mathcal{L}(\tilde{A}_{E_2}) &
\end{array}
$$

To check $\mathcal{L}(E_1) \subseteq \mathcal{L}(\tilde{A}_{E_2})$ we have to look into the transformation of 2NFAs to 1NFAs.

Transforming 2NFAs to 1NFAs

Theorem [Vardi 1988]: Let $A = (\Sigma, S, S_0, \delta, F)$ be a 2NFA.
There is a 1NFA A^c such that

- A^c accepts the complement of A, i.e., $\mathcal{L}(A^c) = \Sigma^* - \mathcal{L}(A)$
- A^c is exponential in A, i.e., $||A^c||$ is $2^{O(||A||)}$

Proof: guess a subset-sequence counterexample.

$a_0 \cdots a_{k-1} \notin \mathcal{L}(A)$ iff there is a sequence $T_0, \ldots, T_k$ of subsets of S such that:

- $S_0 \subseteq T_0$ and $T_k \cap F = \emptyset$
- If $s \in T_i$ and $(t, +1) \in \delta(s, a_i)$, then $t \in T_{i+1}$, for $0 \leq i < k$
- If $s \in T_i$ and $(t, 0) \in \delta(s, a_i)$, then $t \in T_i$, for $0 \leq i < k$
- If $s \in T_i$ and $(t, -1) \in \delta(s, a_i)$, then $t \in T_{i-1}$, for $0 \leq i < k$

The 1NFA A^c guesses such a seQuence $T_0, \ldots, T_k$ and checks that it satisfies the 4 conditions above.

Containment of 2RPQs (finally done!)

Consider two 2RPQs
$$Q_1(x, y) \leftarrow x\ E_1\ y$$
$$Q_2(x, y) \leftarrow x\ E_2\ y$$

Summing up, we have that

$$\begin{array}{rl}
Q_1 \sqsubseteq Q_2 & \text{iff} \\
\mathcal{L}(E_1) \subseteq fold(E_2) & \text{iff} \\
\mathcal{L}(E_1) \subseteq \mathcal{L}(\tilde{A}_{E_2}) & \text{iff} \\
\mathcal{L}(E_1) \cap \mathcal{L}(\tilde{A}^c_{E_2}) = \emptyset & \text{iff} \\
\mathcal{L}(A_{E_1} \times \tilde{A}^c_{E_2}) = \emptyset &
\end{array}$$

Theorem: Containment of 2RPQs is in PSPACE, hence PSPACE-complete.

Containment of queries over semistructured data

Complexity of containment for various classes of queries over semistructured data:

Language	Complexity	
RPQs	PSPACE	[PODS'99]
2RPQs	PSPACE	[PODS'00]
Tree-2RPQs	PSPACE	[DBPL'01]
Conjunctive-2RPQs	EXPSPACE	[KR'00]
Datalog in Unions of C2RPQs	2EXPTIME	[ICDT'03]

VBQP over Semistructured Data – Outline

√ The semistructured data model and regular path queries (RPQs)

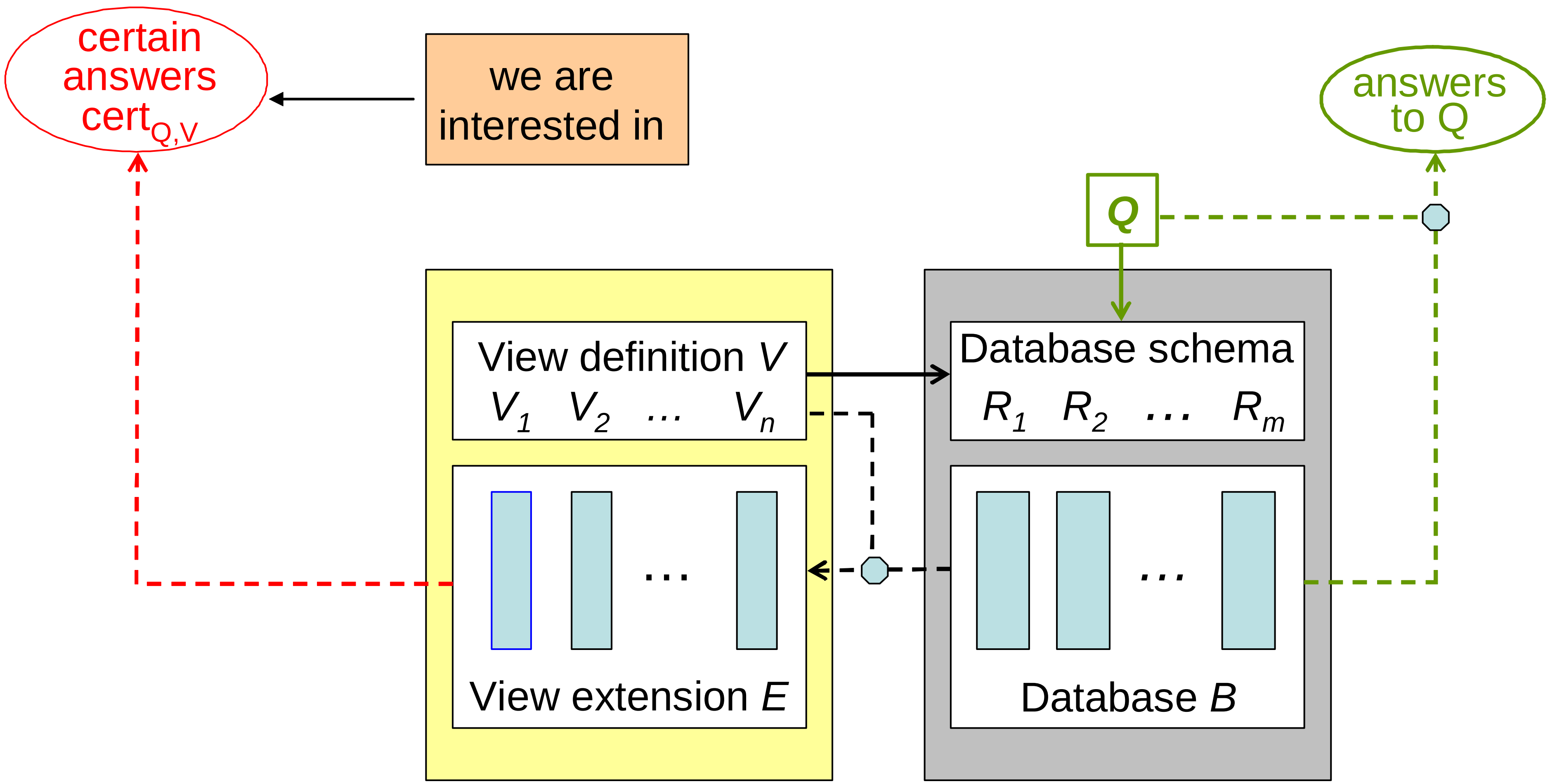
√ Containment of RPQs and 2RPQs

⇒ View-based query rewriting for RPQs and 2RPQs

4. View-based query answering for RPQs and 2RPQs via automata

5. View-based query answering for RPQs and 2RPQs via CSP

View-based query processing



View-based query processing for SSD

We consider the setting where:

- The schema Σ is a set of binary predicates without constraints.
- Each view symbol in $\mathcal{V}$ is a binary predicate.
- Each view definition in $\mathcal{V}^{\Sigma}$ is an RPQ/2RPQ over Σ.
- Each view extension in $\mathcal{E}$ is a binary relation.
- The query Q is an RPQ/2RPQ over Σ.
- Views are sound, complete, or exact views (will discuss mostly the case of sound views).
- The domain is open.

Problem: Given a schema Σ, views $\mathcal{V}$ with definitions $\mathcal{V}^{\Sigma}$ and extensions $\mathcal{E}$, a query Q over Σ, and a tuple $\vec{t}$, decide whether $\vec{t} \in cert(Q, \Sigma, \mathcal{V}^{\Sigma}, \mathcal{E})$.

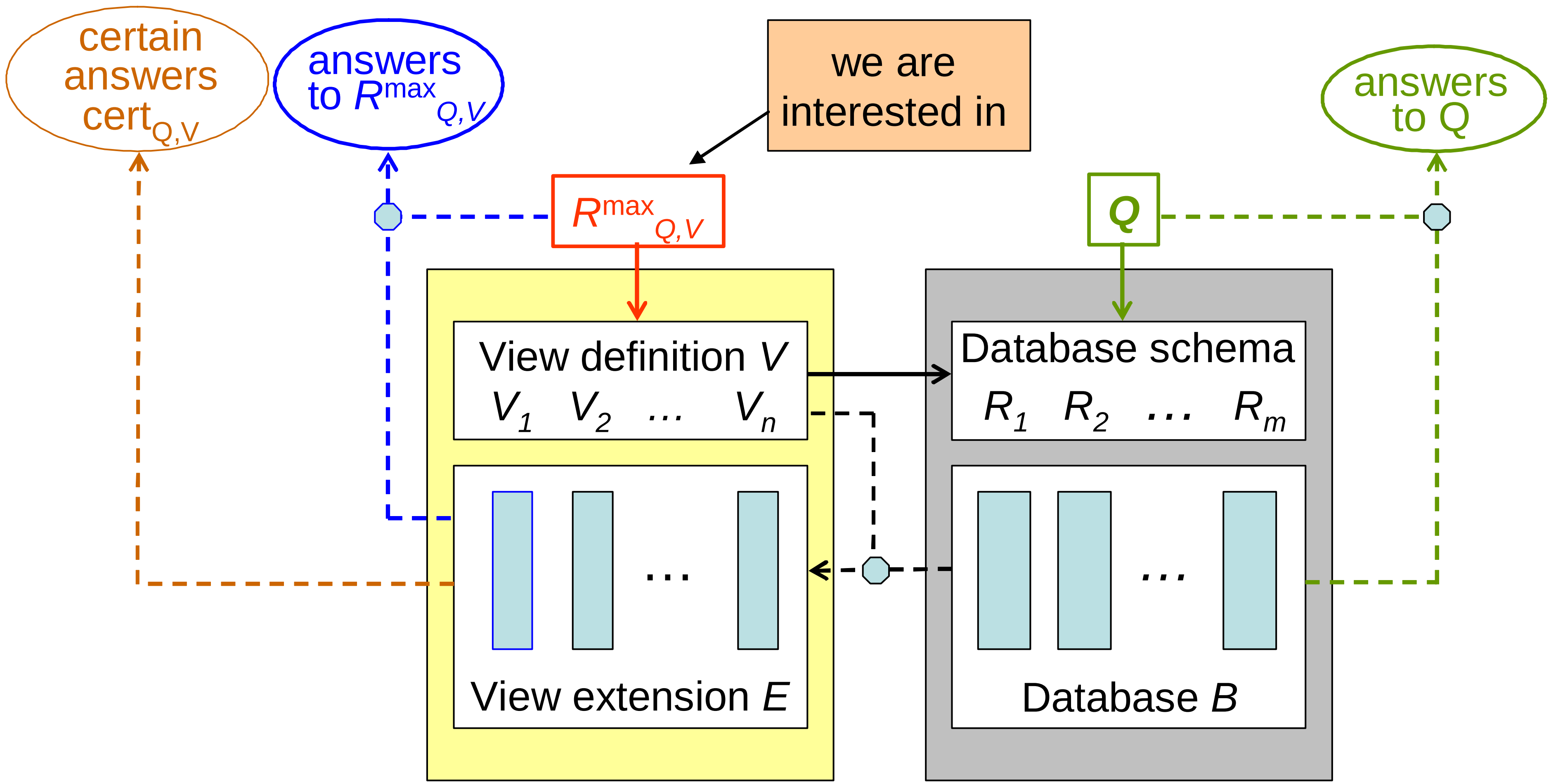
View-based query answering for SSD – Example

- Schema $\Sigma = \{\text{article}, \text{reference}, \text{title}, \text{author}, \ldots\}$
- Set of views $\mathcal{V} = \{V_1, V_2, V_3\}$
- View definitions $\mathcal{V}^\Sigma = \{V_1^\Sigma, V_2^\Sigma, V_3^\Sigma\}$, with

$$
\begin{aligned}
V_1^\Sigma: \quad V_1(b, a) &\leftarrow b\ (\text{article})\ a \\
V_2^\Sigma: \quad V_2(p_1, p_2) &\leftarrow p_1\ (\text{reference}^*)\ p_2 \\
V_3^\Sigma: \quad V_3(p, t) &\leftarrow p\ (\text{title})\ t
\end{aligned}
$$

- View extensions $\mathcal{E} = \{E_1, E_2, E_3\}$, where
  - E_1 stores for each bibliography its articles
  - E_2 stores for each publ. the ones it references directly or indirectly
  - E_3 stores for each publication its title

View-based query answering via rewriting



Note: the class $\mathcal{C}$ of queries in which to express the rewriting is fixed a priori.

View-based query rewriting for SSD

Problem: Given a schema Σ, views $\mathcal{V}$ with definitions $\mathcal{V}^{\Sigma}$, a query Q over Σ, and a class $\mathcal{C}$ of queries over $\mathcal{V}$, compute the query R over $\mathcal{V}$ such that:

1. R is a query in $\mathcal{C}$.
2. R is a sound rewriting, i.e., for every database $\mathcal{B}$ over Σ and every view extension $\mathcal{E}$ that is sound wrt $\mathcal{B}$ (i.e., such that $\mathcal{E} \subseteq \mathcal{V}^{\Sigma}(\mathcal{B})$), we have $R(\mathcal{E}) \subseteq Q(\mathcal{B})$.
3. R is the maximal (wrt containment) query in $\mathcal{C}$ satisfying condition (2).

Such a query is called the $\mathcal{C}$-maximal rewriting of Q wrt Σ and $\mathcal{V}$, and is denoted $rew_{\mathcal{C}}(Q, \Sigma, \mathcal{V})$.

In the following, we assume that the class $\mathcal{C}$ of queries in which the rewriting is expressed coincides with that of Q (i.e., is that of RPQs / 2RPQs), and we do not mention it explicitly.

View-based query rewriting for 2RPQs – Example

Consider three views with definitions:

$$\begin{aligned} V_1(b, a) &\leftarrow b\ (\mathsf{article})\ a \\ V_2(p_1, p_2) &\leftarrow p_1\ (\mathsf{reference}^*)\ p_2 \\ V_3(p, t) &\leftarrow p\ (\mathsf{title})\ t \end{aligned}$$

Query: $Q(x, y) \leftarrow x\ (\mathsf{article}\cdot(\mathsf{reference} + \mathsf{reference}^-)^*\cdot\mathsf{title})\ y$

- $R(x, y) \leftarrow x\ (\mathsf{v}_1\cdot\mathsf{v}_2\cdot\mathsf{v}_3)\ y$ is an RPQ rewriting of Q
- $R(x, y) \leftarrow x\ (\mathsf{v}_1\cdot(\mathsf{v}_2 + \mathsf{v}_2{}^-)\cdot\mathsf{v}_3)\ y$ is a 2RPQ rewriting of Q
- $R(x, y) \leftarrow x\ (\mathsf{v}_1\cdot(\mathsf{v}_2 + \mathsf{v}_2{}^-)^*\cdot\mathsf{v}_3)\ y$ is a 2RPQ-maximal rewriting of Q that is also exact

Sound rewritings

Since RPQs and 2RPQs are monotone queries, the following characterizations of a sound rewriting R of Q wrt Σ and $\mathcal{V}$ are equivalent:

- For every database $\mathcal{B}$ over Σ and every view extension $\mathcal{E}$ with $\mathcal{E} \subseteq \mathcal{V}^{\Sigma}(\mathcal{B})$, we have that $R(\mathcal{E}) \subseteq Q(\mathcal{B})$.
- For every database $\mathcal{B}$ over Σ and every view extension $\mathcal{E}$ with $\mathcal{E} = \mathcal{V}^{\Sigma}(\mathcal{B})$, we have that $R(\mathcal{E}) \subseteq Q(\mathcal{B})$.
- For every database $\mathcal{B}$ over Σ, we have that $R(\mathcal{V}^{\Sigma}(\mathcal{B})) \subseteq Q(\mathcal{B})$.

Observe: The following two ways of computing $R(\mathcal{V}^{\Sigma}(\mathcal{B}))$ are equivalent:

- First evaluate the view definitions $\mathcal{V}^{\Sigma}$ over $\mathcal{B}$, and then evaluate R over the obtained view extension.
- First expand in R the view symbols $\mathcal{V}$ with the corresponding definitions $\mathcal{V}^{\Sigma}$, and then evaluate the resulting query over $\mathcal{B}$.

Expansion of an RPQ / 2RPQ

We call a query over Σ a Σ-query, and a query over $\mathcal{V}$ a $\mathcal{V}$-query.

Consider:

- a set of views $\mathcal{V} = \{V_1, \ldots, V_k\}$
- a set of view definitions (RPQs or 2RPQs) $\mathcal{V}^\Sigma = \{V_1^\Sigma, \ldots, V_k^\Sigma\}$, with $V_i^\Sigma : V_i(x, y) \leftarrow x\ E_i\ y$
- a $\mathcal{V}$-query (RPQ or 2RPQ) $R : R(x, y) \leftarrow x\ E_R\ y$

Definition: The expansion $R_{[\mathcal{V} \mapsto \mathcal{V}^\Sigma]}$ of R wrt $\mathcal{V}^\Sigma$ is the Σ-query obtained from R by replacing in E_R:

- each occurrence of a view symbol V_i with E_i.
- each occurrence of an inverse view symbol V_i^- with $inv(E_i)$, where

$$inv(p) = p^- \qquad inv(e_1 + e_2) = inv(e_1) + inv(e_2)$$

$$inv(p^-) = p \qquad inv(e_1 \cdot e_2) = inv(e_2) \cdot inv(e_2)$$

$$inv(e^*) = inv(e)^*$$

Checking rewritings via expansion

Hence, to check whether a rewriting R of Q wrt Σ and $\mathcal{V}^{\Sigma}$ is sound, we can:

1. Compute $R_{[\mathcal{V}\mapsto\mathcal{V}^{\Sigma}]}$ by expanding each direct (resp., inverse) view symbol V (resp., V^{-}) in R with the corresponding definition V^{Σ} (resp., $inv(V^{\Sigma})$).

2. Check whether $R_{[\mathcal{V}\mapsto\mathcal{V}^{\Sigma}]} \sqsubseteq Q$ (notice that both are Σ-queries).

Counterexample method for rewriting of RPQs

Due to Language-Theoretic Lemma 1, we can treat RPQs simply as regular expressions.

Consider a candidate rewriting: $R = u_1 \cdots u_k \in \mathcal{V}^*$

- R is a bad rewriting of Q if $\quad R_{[\mathcal{V} \mapsto \mathcal{V}^\Sigma]} \not\sqsubseteq Q$
- R is a bad rewriting of Q if there are witnesses $w_1, \ldots, w_k \in \Sigma^*$ such that $w_1 \cdots w_k \not\sqsubseteq Q$, where $w_i \in \mathcal{L}(V_j^\Sigma)$ if $u_i = V_j$.

Example: $Q = abcd, \quad \mathcal{V} = \{V_1, V_2\}, \quad V_1^\Sigma = ab, \quad V_2^\Sigma = cd$

- bad rewriting: $V_2 V_1$, with witnesses cd and ab
- good rewriting: $V_1 V_2$

Rewriting of RPQs

Construction is based on 1NFAs:

1. Construct a 1DFA $A_Q = (\Sigma, S, s_0, \delta, F)$ for Q $\leadsto$ exponential blowup
2. Construct the 1NFA $A_b = (\mathcal{V}, S, s_0, \delta', S - F)$, where
   $s_j \in \delta'(s_i, V)$ iff there exists a word $w \in \mathcal{L}(V^\Sigma)$ s.t. $s_j \in \delta^*(s_i, w)$

   Note:
   - A_Q and A_{bad} have the same states, but complementary final states.
   - A_{bad} accepts a word $u_1 \cdots u_k \in \mathcal{V}^*$ iff it is a bad rewriting of Q.
     The words w used in the definition of δ' act as witnesses.

   $\leadsto$ linear blowup
3. Complement A_{bad} to get a 1NFA A_{rew} for good rewritings.
   $\leadsto$ exponential blowup

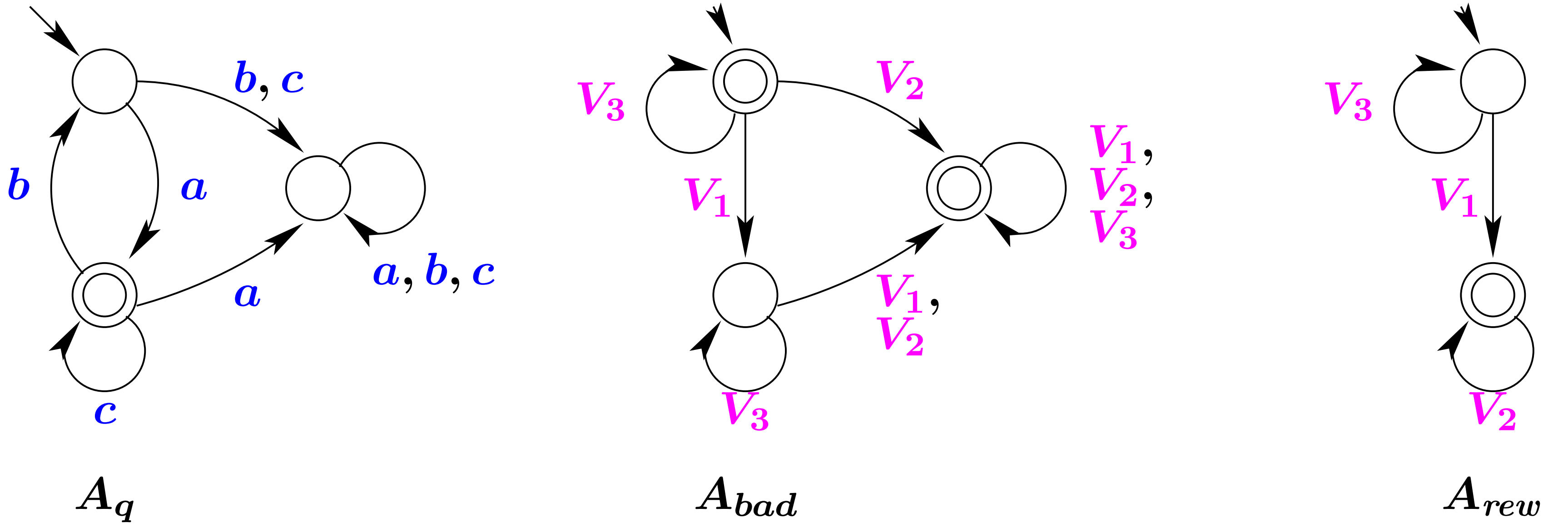
The construction yields the maximal path rewriting.

It is represented by a 1DFA. $\leadsto$ The maximal path rewriting is an RPQ.

Rewriting of RPQs – Example

Query: $Q = a \cdot (b \cdot a + c)^*$

Views: $\mathcal{V} = \{V_1, V_2, V_3\}$, with $V_1^\Sigma = a$, $V_2^\Sigma = a \cdot c^* \cdot b$, $V_3^\Sigma = c$



Rewriting of RPQs – Complexity

Checking nonemptiness of the maximal path-rewriting of an RPQ Query wrt a set of RPQ views is EXPSPACE-complete.

Proof:

- The 1DFA A_{rew} is of double exponential size.
- We can check for its non-emptiness on the fly in NLOGSPACE in the number of its states, i.e., in EXPSPACE in the size of Q
- The matching lower-bound is by a reduction from an EXPSPACE-hard tiling problem.

There exists an RPQ Query Q and RPQ views $\mathcal{V}$ over an alphabet Σ such that the smallest 1NFA for the rewriting $rew(Q, \Sigma, \mathcal{V})$ is of double exponential size.

Exact rewritings

Definition: A rewriting R of an RPQ Q wrt Σ and $\mathcal{V}^{\Sigma}$ is exact if $Q \sqsubseteq R_{[\mathcal{V} \mapsto \mathcal{V}^{\Sigma}]}$.

Note: since R is a rewriting, we also have $R_{[\mathcal{V} \mapsto \mathcal{V}^{\Sigma}]} \sqsubseteq Q$.

To verify whether R is an exact rewriting of Q wrt Σ and $\mathcal{V}^{\Sigma}$:

1. Construct a 1NFA B over Σ accepting $R_{[\mathcal{V} \mapsto \mathcal{V}^{\Sigma}]}$.

   It suffices to replace each V_i edge in R with a 1NFA for V_i^{Σ}.

2. Check whether $\mathcal{L}(A_Q) \subseteq \mathcal{L}(B)$, i.e.,
   - complement B to obtain an 1NFA $\overline{B}$,
   - check whether $\mathcal{L}(A_Q) \cap \mathcal{L}(\overline{B}) = \emptyset$, i.e., whether $\mathcal{L}(A_Q \times \overline{B}) = \emptyset$.

$\overline{B}$ may be of triply exponential size in Q but we can check its emptyness on-the-fly in 2EXPSPACE.

Rewriting of RPQs – Results

Theorem:

- Nonemptiness of the maximal rewriting is EXPSPACE-complete.
- The maximal rewriting may be of double exponential size.
- Existence of an exact rewriting is 2EXPSPACE-complete.

Rewriting of 2RPQs

- The query and the view definitions are 2RPQs over Σ.
- We look for rewritings that are a 2RPQ over $\mathcal{V}$.
- We consider again candidate rewritings and try to characterize bad rewritings.

Candidate rewriting: $R = u_1 \cdots u_k \in \mathcal{V}^{\pm *}$

- To check whether a rewriting is bad, we need to expand both direct and inverse view symbols.
- R is a bad rewriting of Q if $\quad R_{[\mathcal{V} \mapsto \mathcal{V}^\Sigma]} \not\sqsubseteq Q$
- R is a bad rewriting of Q if there are witnesses $w_1, \ldots, w_k \in \Sigma^{\pm *}$ such that $w_1 \cdots w_k \not\sqsubseteq Q$, where
  - $w_i \in \mathcal{L}(V_j^\Sigma)$ if $u_i = V_j$.
  - $w_i \in \mathcal{L}(inv(V_j^\Sigma))$ if $u_i = V_j^-$.

Counterexample method for rewriting of 2RPQs

We consider counterexample words, which are obtained from a bad rewriting by inserting after each symbol u its witness w.

Definition: Counterexample word $u_1 w_1 \cdots u_k w_k$

1. $w_i \in \mathcal{L}(V_j^\Sigma)$ if $u_i = V_j$.
2. $w_i \in \mathcal{L}(inv(V_j^\Sigma))$ if $u_i = V_j^-$.
3. $w_1 \cdots w_k \not\sqsubseteq Q$.

Example: $Q = abcd$, $\mathcal{V} = \{V_1, V_2\}$, $V_1^\Sigma = ab$, $V_2^\Sigma = cd$

- bad rewriting: $V_2 V_1$, with witnesses $w_1 = cd$, $w_2 = ab$
- counterexample word: $V_2\, cd\, V_1\, ab$

Checking counterexample words with 2NFAs

- Check (1) and (2) with 2NFAs for V_j^Σ.
- Use folding technique to construct 2NFA to check $w_1 \cdots w_k \sqsubseteq Q$.
- Complement resulting 2NFA.

$\rightsquigarrow$ Complexity is exponential

From counterexamples to rewritings

To construct good rewritings:

1. Construct 1NFA A_1 for counterexample words $u_1 w_1 \cdots u_k w_k$.
   $\leadsto$ exponential
2. Project out witness words w_i to get 1NFA A_2 for bad rewritings $u_1 \cdots u_k$.
   $\leadsto$ linear
3. Complement A_2 to get 1NFA for good rewritings.
   $\leadsto$ exponential

The construction yields the maximal two-way path rewriting

It is represented by a 1DFA. $\leadsto$ The maximal two-way path rewriting is a 2RPQ.

Rewriting of 2RPQs – Results

We get for 2RPQs the same upper (and lower) bounds as for RPQs.

Theorem:

- Nonemptiness of the maximal rewriting is EXPSPACE-complete.
- The maximal rewriting may be of double exponential size.
- Existence of an exact rewriting is 2EXPSPACE-complete.

VBQP over Semistructured Data – Outline

√ The semistructured data model and regular path queries (RPQs)

√ Containment of RPQs and 2RPQs

√ View-based query rewriting for RPQs and 2RPQs

⇒ View-based query answering for RPQs and 2RPQs via automata

5. View-based query answering for RPQs and 2RPQs via CSP

View-based query answering for RPQs / 2RPQs

Given:

- a set $\mathcal{V}$ of view symbols with a corresponding set $\mathcal{V}^{\Sigma}$ of RPQ / 2RPQ view definitions over a relational alphabet Σ
- a corresponding set $\mathcal{E}$ of view extensions
- a 2RPQ Q over Σ
- a pair (c, d) of objects

check whether $(c, d) \in cert(Q, \Sigma, \mathcal{V}^{\Sigma}, \mathcal{E})$.

In other words, check whether for every database $\mathcal{B}$ over Σ such that the view extension $\mathcal{E}$ is sound wrt $\mathcal{B}$ (i.e, $\mathcal{E} \subseteq \mathcal{V}(\mathcal{B})$), we have that $(c, d) \in Q(\mathcal{B})$.

View-based query answering for 2RPQs – Idea

We search for a counterexample database, i.e., a database $\mathcal{B}$ such that $\mathcal{E}$ is sound wrt $\mathcal{B}$, but such that (c, d) is not in the answer to Q over $\mathcal{B}$.

Technique:

1. Encode counterexample databases as finite words.
2. Construct a 2NFA that accepts such words.
3. Check for emptiness of the automaton.

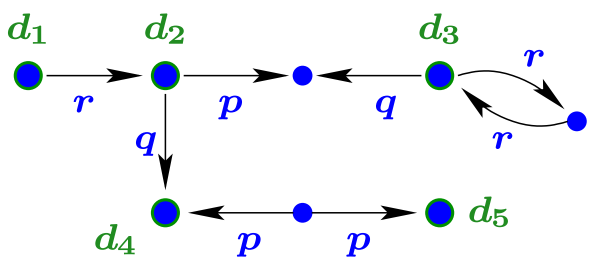
Canonical counterexample databases

A database $\mathcal{B}$ is a counterexample to $(c, d) \in cert(Q, \Sigma, \mathcal{V}, \mathcal{E})$ if

- $\mathcal{E} \subseteq \mathcal{V}^{\Sigma}(\mathcal{B})$, i.e., the view extension $\mathcal{E}$ is sound wrt $\mathcal{B}$,
- $(c, d) \notin Q(\mathcal{B})$.

Observations:

- It is sufficient to restrict the attention to counterexamples of a special form (canonical databases)



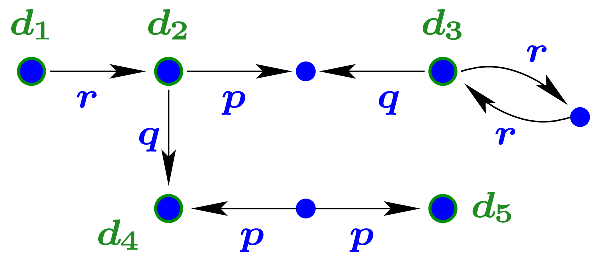
Set of objects in the view extension $\mathcal{E}$:
$\Delta_{\mathcal{E}} = \{d_1, d_2, d_3, d_4, d_5, \ldots\}$

- Each canonical database $\mathcal{B}$ can be represented as a word $w_{\mathcal{B}}$ over $\Sigma_A = \Sigma^{\pm} \cup \Delta_{\mathcal{E}} \cup \{\$\}$ of the form

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

Canonical DBs and 2NFAs

$\mathcal{B}$:



$$Q(x,y) \leftarrow x\,(r\cdot(p+q)\cdot p^{-}\cdot p\cdot q^{-*})\,y$$

Word representing $\mathcal{B}$:

$$\$\ d_1\ r\ d_2\ \$\ d_4\ p^{-}p\ d_5\ \$\ d_4\ q^{-}\ d_2\ \$\ d_3\ rr\ d_3\ \$\ d_2\ pq^{-}\ d_3\ \$$$

To verify whether $(d_i, d_j) \in Q(\mathcal{B})$ we exploit that 2NFAs can:

- move on the word both forward and backward
- jump from one position in the word representing a node to any other position representing the same node (search mode)

$\leadsto$ We can construct a 2NFA $A_{(Q,d_i,d_j)}$ that accepts $w_{\mathcal{B}}$ iff $(d_i, d_j) \in Q(\mathcal{B})$

View-based query answering for 2RPQs – Technique

To check whether $(c, d) \notin cert(Q, \Sigma, \mathcal{V}, \mathcal{E})$, we construct a 1NFA A_{QA} as the intersection of:

- the 1NFA A_0 that accepts $(\$\cdot\Delta_{\mathcal{E}}\cdot\Sigma^{\pm}\cdot\Sigma^{\pm *}\cdot\Delta_{\mathcal{E}})^{*}\cdot\$$
- the 1NFAs corresponding to the various $A_{(V_i^{\Sigma},a,b)}$
  (for each sound or exact view V_i, and for each pair $(a, b) \in E_i$)
- the 1NFAs corresponding to the complement of each A_{V_i}
  (for each complete or exact view V_i, the 2NFA A_{V_i} checks whether a pair of objects other than those in E_i is in $V_i(\mathcal{B})$)
- the 1NFA corresponding to the complement of $A_{(Q,c,d)}$

$\rightsquigarrow$ A_{QA} accepts words representing counterexample DBs

We have that $(c, d) \notin cert(Q, \Sigma, \mathcal{V}, \mathcal{E})$ iff A_{QA} is nonempty.

Complexity of query answering

Can be measured in three different ways:

- Data complexity: as a function of the size of the view extension $\mathcal{E}$
- Expression complexity: as a function of the size of the query Q and of the view definitions $V_1^\Sigma, \ldots, V_k^\Sigma$
- Combined complexity: as a function of the size of both the view extension $\mathcal{E}$ and the expressions Q, $V_1^\Sigma, \ldots, V_k^\Sigma$

View-based query answering for 2RPQs – Upper bounds

- All 2NFAs are of linear size in the size of Q, all views in $\mathcal{V}$ and the view extensions $\mathcal{E}$.
  $\leadsto$ The corresponding 1NFA would be exponential.
- However, we can avoid the explicit construction of A_{QA}, and we can construct it on the fly while checking for nonemptiness.

$\leadsto$ View-based query answering for 2RPQs is in PSPACE wrt expression complexity and combined complexity.

PSPACE-hardness follows immediately by reduction from universality of regular languages.

What about data complexity, i.e., complexity in the size of the view extension $\mathcal{E}$?

VBQP over Semistructured Data – Outline

√ The semistructured data model and regular path queries (RPQs)

√ Containment of RPQs and 2RPQs

√ View-based query rewriting for RPQs and 2RPQs

√ View-based query answering for RPQs and 2RPQs via automata

⇒ View-based query answering for RPQs and 2RPQs via CSP

View-based query answering for 2RPQs – Data complexity

- In the previous algorithm one cannot immediately single out the contribution of $\mathcal{E}$ to the nonemptiness check of the automaton A_{QA}.
- This can however be done by analyzing the transformation from 2NFA to 1NFA, and modifying the construction of the automata to avoid search mode.

We look at an alternative way to analyze data complexity, derived from a tight connection between view-based query answering under *sound views* and constraint satisfaction (CSP).

$\rightsquigarrow$ Better insight into view-based query answering for RPQs and 2RPQs.

$\rightsquigarrow$ Several additional results on various forms of view-based query processing.

Constraint satisfaction problems

Let $\mathcal{A}$ and $\mathcal{B}$ be relational structures over the same alphabet.

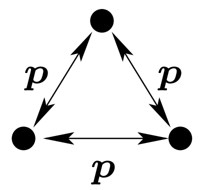
A homomorphism h is a mapping from $\mathcal{A}$ to $\mathcal{B}$ such that for every relation R, if $(c_1, \ldots, c_n) \in R(\mathcal{A})$, then $(h(c_1), \ldots, h(c_n)) \in R(\mathcal{B})$.

Non-uniform constraint satisfaction problem $CSP(\mathcal{B})$: the set of relational structures $\mathcal{A}$ such that there is a homomorphism from $\mathcal{A}$ to $\mathcal{B}$.

Complexity:

- $CSP(\mathcal{B})$ is in NP.
- There are structures $\mathcal{B}$ for which $CSP(\mathcal{B})$ is NP-hard.

  Example:



CSP and VBQA for 2RPQs – Constraint template

From Q and $\mathcal{V}$, we can define a relational structure $\mathcal{T} = CT_{Q,\mathcal{V}}$, called constraint template of Q wrt $\mathcal{V}$:

- The vocabulary of $\mathcal{T}$ is $\{R_1, \ldots, R_k\} \cup \{U_{ini}, U_{fin}\}$, where
  - each R_i corresponds to a view V_i and denotes a binary predicate
  - U_{ini} and U_{fin} denote unary predicates
- Let Q be represented by a 1NFA $(\Sigma^{\pm}, S, S_0, \delta, F)$:
  - The domain $\Delta_{\mathcal{T}}$ of $\mathcal{T}$ is 2^S, i.e., all sets of states of Q
  - $\sigma \in U_{ini}(\mathcal{T})$ iff $S_0 \subseteq \sigma$
  - $\sigma \in U_{fin}(\mathcal{T})$ iff $\sigma \cap F = \emptyset$
  - $(\sigma_1, \sigma_2) \in R_i(\mathcal{T})$ iff there exists a word $p_1 \cdots p_k \in \mathcal{L}(V_i^{\Sigma})$ and a sequence $T_0, \ldots, T_k$ of subsets of S such that:
    1. $T_0 = \sigma_1$ and $T_k = \sigma_2$
    2. if $s \in T_i$ and $t \in \delta(s, p_{i+1})$, then $t \in T_{i+1}$
    3. if $s \in T_i$ and $t \in \delta(s, p_i^-)$, then $t \in T_{i-1}$

CSP and VBQA for 2RPQs – Constraint instance

Observations:

- Intuitively, the constraint template $CT_{Q,\mathcal{V}}$ encodes how the states of Q change when moving along a database according to the views.
- The existence of a word $p_1 \cdots p_k \in \mathcal{L}(V_i^{\Sigma})$ and of a sequence $T_0, \ldots, T_k$ of subsets of S satisfying conditions 1–3 can be checked in PSPACE.
- $CT_{Q,\mathcal{V}}$ can be computed in time exponential in Q and polynomial in $\mathcal{V}$.

From $\mathcal{E}$ and two objects c and d, we can define another relational structure $\mathcal{I} = CI_{\mathcal{E}}^{c,d}$ over the same vocabulary, called the constraint instance:

- The domain $\Delta_{\mathcal{I}}$ of $\mathcal{I}$ is $\Delta_{\mathcal{E}} \cup \{c, d\}$
- $R_i(\mathcal{I}) = E_i$, for $i \in \{1, \ldots, k\}$
- $U_{ini}(\mathcal{I}) = \{c\}$
- $U_{fin}(\mathcal{I}) = \{d\}$

CSP and view-based query answering for 2RPQs

Theorem: (c, d) is not a certain answer to Q wrt $\mathcal{V}$ and $\mathcal{E}$

if and only if

there is a homomorphism from $CI_{\mathcal{E}}^{c,d}$ to $CT_{Q,\mathcal{V}}$, i.e, $CI_{\mathcal{E}}^{c,d} \in CSP(CT_{Q,\mathcal{V}})$

$\leadsto$ Characterization of view-based query answering for 2RPQs in terms of CSP

$\leadsto$ View-based query answering for 2RPQs is in coNP wrt data complexity

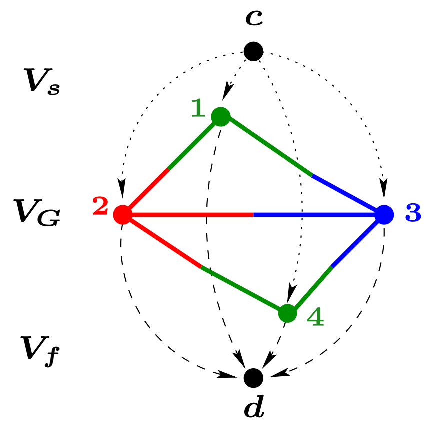
Query answering for RPQs — Data complexity lower bound

coNP-hard wrt data compexity, by reduction from 3-coloring:

$$
\begin{array}{llcl}
\text{Views:} & V_s & = & S_r + S_g + S_b \\
& V_G & = & R_{rg} + R_{gr} + R_{rb} + R_{br} + R_{gb} + R_{bg} \\
& V_f & = & F_r + F_g + F_b \\
\text{Query:} & Q & = & \sum_{x \neq y} S_x \cdot F_y + \sum_{x \neq y \vee w \neq z} S_x \cdot R_{yw} \cdot F_y \quad \text{(color mismatch)}
\end{array}
$$

Only domain and view extensions depend on graph $G = (N, E)$.

$$
\begin{array}{llcl}
\text{Domain:} & \Delta_{\mathcal{E}} & = & N \cup \{c, d\} \\
\text{Extensions:} & E_s & = & \{(c, a) \mid a \in N\} \\
& E_G & = & \{(a, b), (b, a) \mid (a, b) \in E\} \\
& E_f & = & \{(a, d) \mid a \in N\}
\end{array}
$$



Thm: G is 3-colorable iff (c, d) is not a certain answer to Q.

Note: we have used only a query and views that are unions of simple paths.

Complexity of view-based query answering for RPQs / 2RPQs

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*

From [ICDE'00] for RPQs and [PODS'00, ICDT'05] for 2RPQs.

Consequence of complexity results

+ The view-based query answering algorithm provides a set of answers that is sound and complete.

– A coNP data complexity does not allow for effective deployment of the query answering algorithm.

  Note that coNP-hardness holds already for queries and views that are unions of simple paths (no reflexive-transitive closure).

$\leadsto$ Adopt an indirect approach to view-based query answering, via query rewriting.

Query answering by rewriting

To answer a query Q wrt $\mathcal{V}$ and $\mathcal{E}$:

1. re-express Q in terms of the view symbols, i.e., compute a rewriting of Q.
2. directly evaluate the rewriting over $\mathcal{E}$.

Comparison with direct approach to query answering:

- **+** We can consider rewritings in a class with polynomial data complexity (e.g., 2RPQs) $\leadsto$ the data complexity for query answering is polynomial.
- **+/–** We have traded expression complexity for data complexity.
- **–** We may lose completeness (i.e., not obtain all certain answers).

We need to establish the "quality" of a rewriting:

- When does the (maximal) rewriting compute all certain answers?
- What do we gain or lose by considering rewritings in a given class?