Optimizing Query Answering under Ontological Constraints

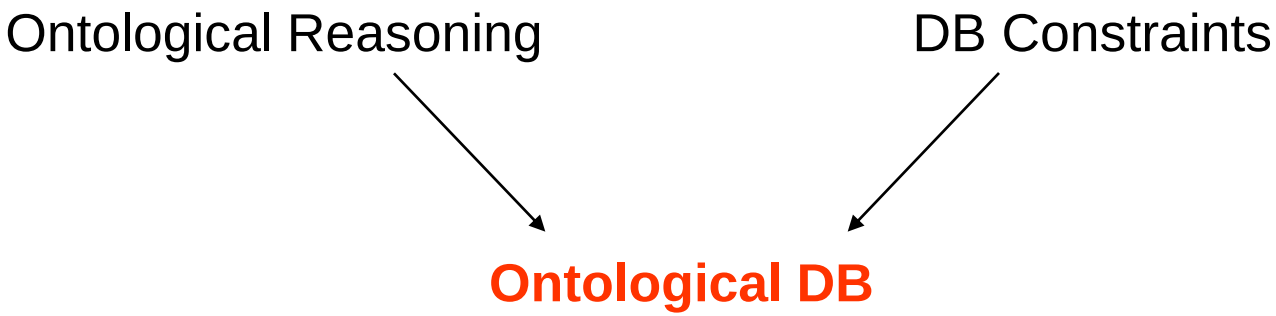
Giorgio Orsi[1,2] and Andreas Pieris[2]

[1]Institute for the Future of Computing
Oxford Martin School
University of Oxford

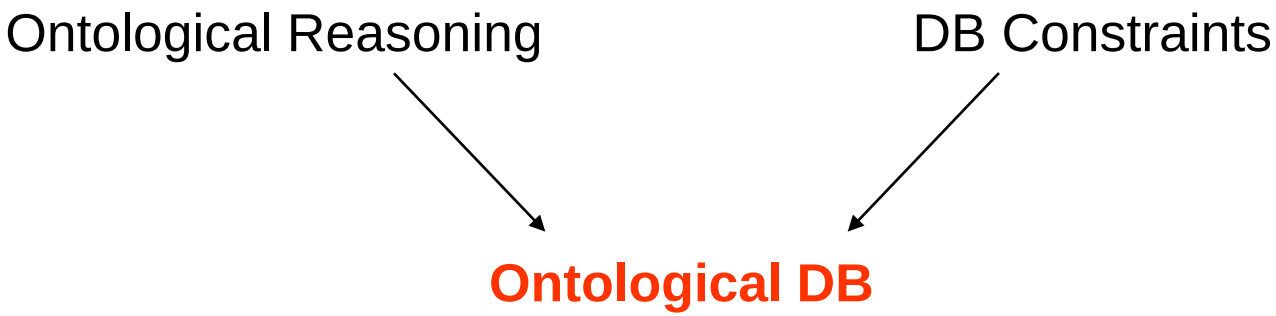
[2]Department of Computer Science
University of Oxford

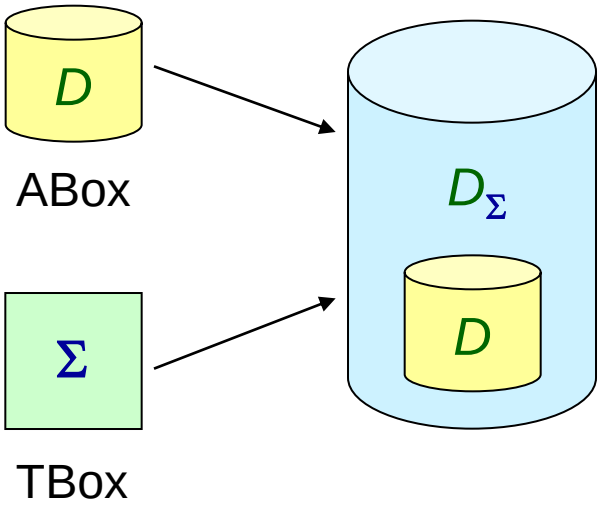
VLDB 2011

Ontological Databases

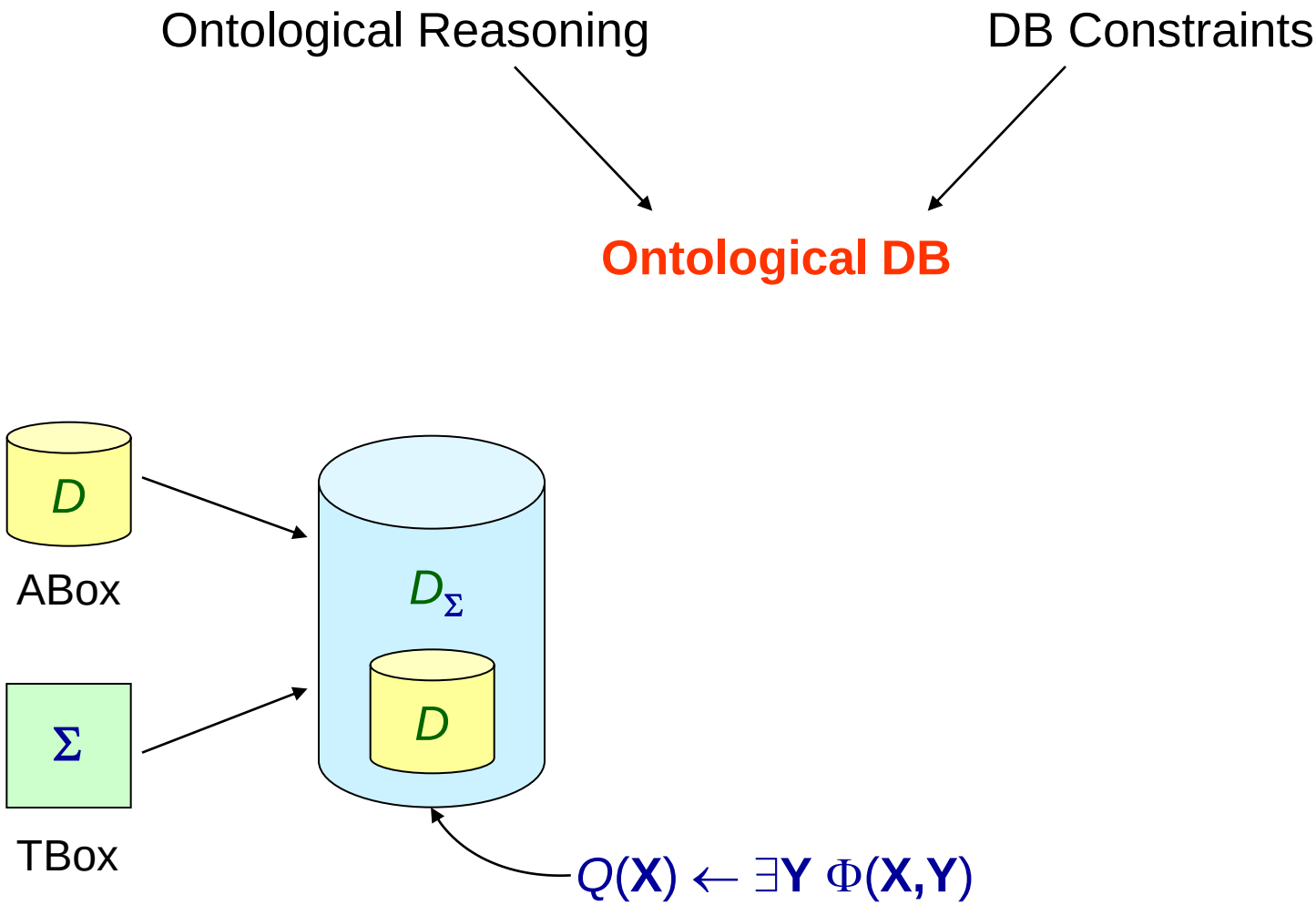


Ontological Databases

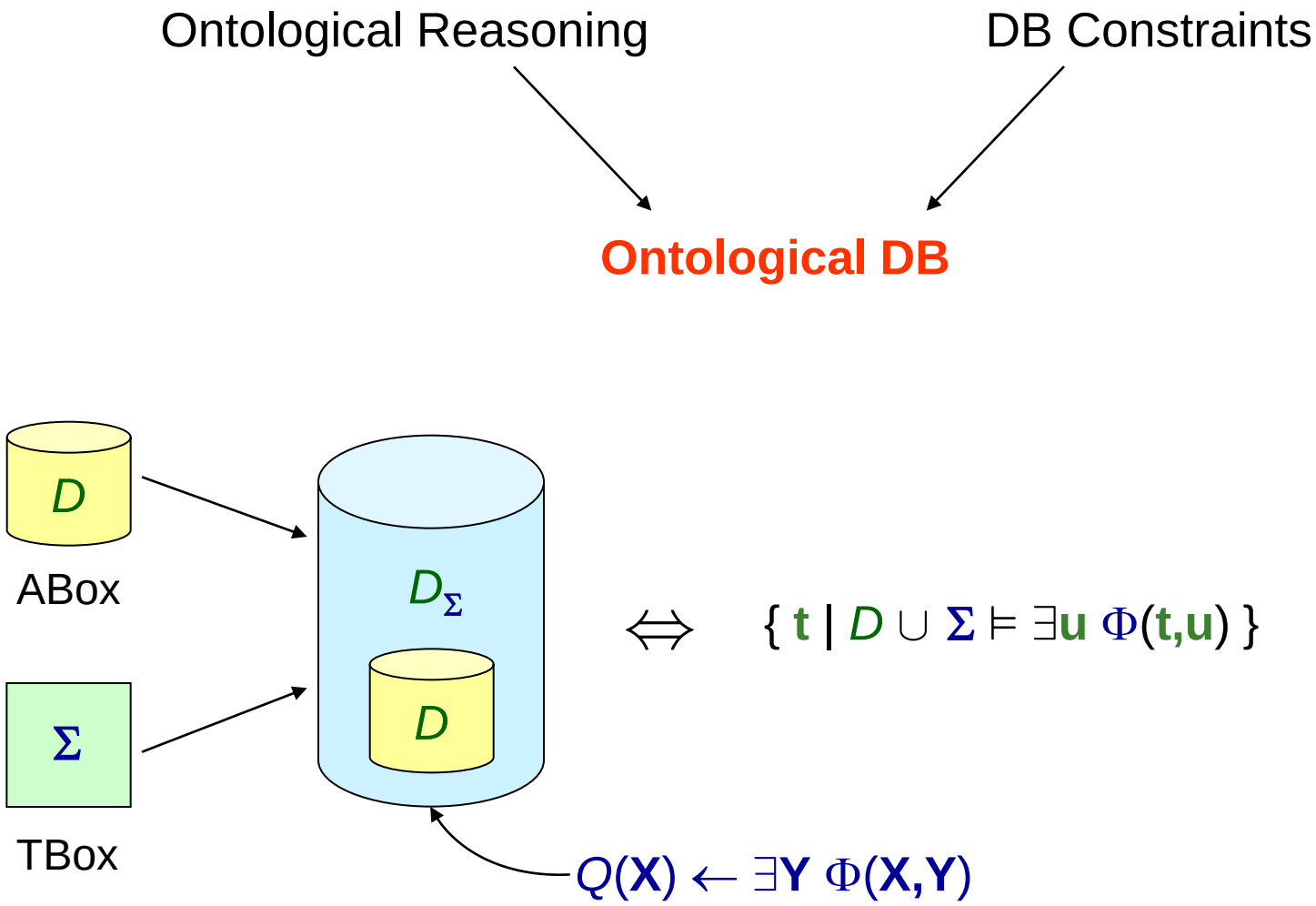




Ontological Databases



Ontological Databases



Ontological Constraints (examples)

Concept Inclusions:	$\forall X\ emp(X) \rightarrow person(X)$
(Inverse) Relation Inclusion:	$\forall X \forall Y\ manages(X,Y) \rightarrow isManaged(Y,X)$
Relation Transitivity:	$\forall X \forall Y \forall Z\ mgs(X,Y), mgs(Y,Z) \rightarrow mgs(X,Z)$
Participation:	$\forall X\ emp(X) \rightarrow \exists Y\ report(X,Y)$
Disjointness:	$\forall X\ emp(X), customer(X) \rightarrow \bot$
Functionality:	$\forall X \forall Y \forall Z\ reports(X,Y), reports(X,Z) \rightarrow Y = Z$

Datalog$^{\pm}$ [Cali' et Al, PODS 09]

– Datalog variant allowing in the head:
  - $\exists$-variables $\rightarrow$ TGDs $\forall \mathbf{X} \forall \mathbf{Y}\ \Phi(\mathbf{X},\mathbf{Y}) \rightarrow \exists \mathbf{Z}\ \Psi(\mathbf{X},\mathbf{Z})$
  - Equality atoms $\rightarrow$ EGDs $\forall \mathbf{X}\ \Phi(\mathbf{X}) \rightarrow X_i = X_j$
  - Constant false ($\bot$) $\rightarrow$ NCs $\forall \mathbf{X}\ \Phi(\mathbf{X}) \rightarrow \bot$

} Datalog^{+}

Datalog$^{\pm}$ [Cali' et Al, PODS 09]

– Datalog variant allowing in the head:
  - $\exists$-variables $\rightarrow$ TGDs $\forall \mathbf{X} \forall \mathbf{Y}\ \Phi(\mathbf{X},\mathbf{Y}) \rightarrow \exists \mathbf{Z}\ \Psi(\mathbf{X},\mathbf{Z})$
  - Equality atoms $\rightarrow$ EGDs $\forall \mathbf{X}\ \Phi(\mathbf{X}) \rightarrow X_i = X_j$
  - Constant false ($\perp$) $\rightarrow$ NCs $\forall \mathbf{X}\ \Phi(\mathbf{X}) \rightarrow \perp$

} Datalog^{+}

– But, query answering under Datalog^{+} is undecidable

Datalog$^{\pm}$ [Cali' et Al, PODS 09]

– Datalog variant allowing in the head:
  - $\exists$-variables $\rightarrow$ TGDs $\forall \mathbf{X} \forall \mathbf{Y}\ \Phi(\mathbf{X},\mathbf{Y}) \rightarrow \exists \mathbf{Z}\ \Psi(\mathbf{X},\mathbf{Z})$
  - Equality atoms $\rightarrow$ EGDs $\forall \mathbf{X}\ \Phi(\mathbf{X}) \rightarrow X_i = X_j$
  - Constant false ($\bot$) $\rightarrow$ NCs $\forall \mathbf{X}\ \Phi(\mathbf{X}) \rightarrow \bot$

} Datalog^{+}

– But, query answering under Datalog^{+} is undecidable

– Datalog^{+} is syntactically restricted $\rightarrow$ **Datalog$^{\pm}$**

Datalog$^{\pm}$ [Cali' et Al, PODS 09]

– Datalog variant allowing in the head:
  - $\exists$-variables $\rightarrow$ TGDs $\forall \mathbf{X} \forall \mathbf{Y}\ \Phi(\mathbf{X},\mathbf{Y}) \rightarrow \exists \mathbf{Z}\ \Psi(\mathbf{X},\mathbf{Z})$
  - Equality atoms $\rightarrow$ EGDs $\forall \mathbf{X}\ \Phi(\mathbf{X}) \rightarrow X_i = X_j$
  - Constant false ($\bot$) $\rightarrow$ NCs $\forall \mathbf{X}\ \Phi(\mathbf{X}) \rightarrow \bot$

} Datalog^{+}

– But, query answering under Datalog^{+} is undecidable

– Datalog^{+} is syntactically restricted $\rightarrow$ **Datalog$^{\pm}$**

– TGDs more expressive than inclusion dependencies

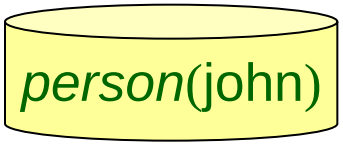
$$\forall D \forall P \forall A\ runs(D,P), area(P,A) \rightarrow \exists E\ employee(E,D,A)$$

The Chase Procedure

Input: Database D, set of TGDs Σ

Output: A model of $D \cup \Sigma$

D



Σ

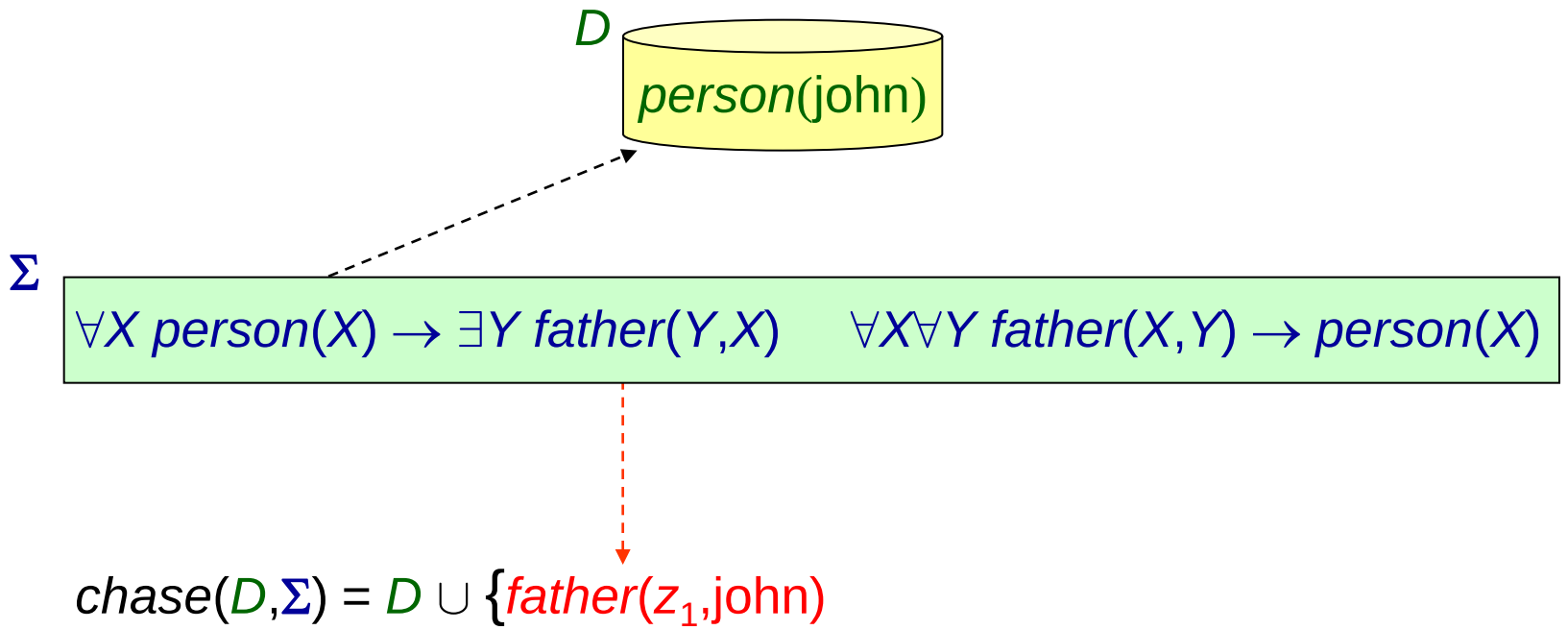
$\forall X\ person(X) \rightarrow \exists Y\ father(Y,X) \qquad \forall X \forall Y\ father(X,Y) \rightarrow person(X)$

$chase(D,\Sigma) = D \cup ?$

The Chase Procedure

Input: Database D, set of TGDs Σ

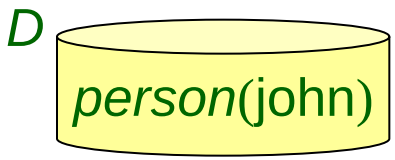
Output: A model of $D \cup \Sigma$



The Chase Procedure

Input: Database D, set of TGDs Σ

Output: A model of $D \cup \Sigma$



Σ

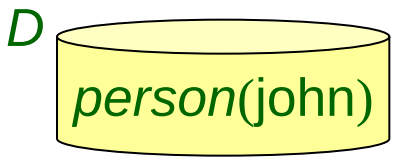
$\forall X\ person(X) \rightarrow \exists Y\ father(Y,X) \qquad \forall X \forall Y\ father(X,Y) \rightarrow person(X)$

$chase(D,\Sigma) = D \cup \{father(z_1,\text{john}),\ person(z_1)\}$

The Chase Procedure

Input: Database D, set of TGDs Σ

Output: A model of $D \cup \Sigma$



Σ

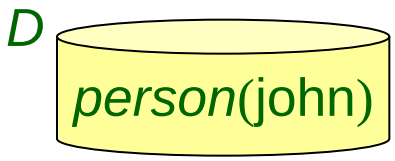
$\forall X\ person(X) \rightarrow \exists Y\ father(Y,X) \qquad \forall X \forall Y\ father(X,Y) \rightarrow person(X)$

$chase(D,\Sigma) = D \cup \{father(z_1,\text{john}), person(z_1), father(z_2,z_1)$

The Chase Procedure

Input: Database D, set of TGDs Σ

Output: A model of $D \cup \Sigma$

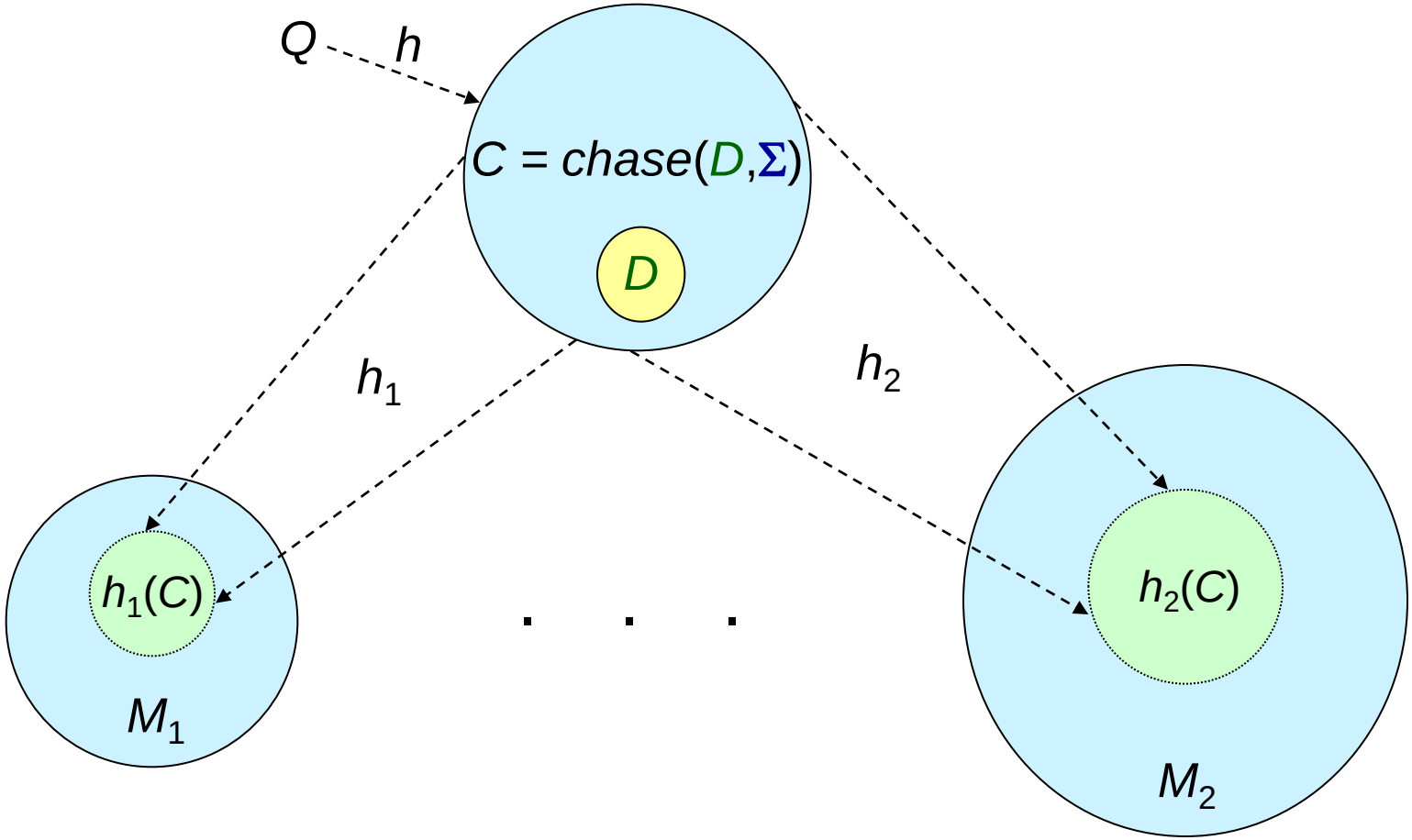


Σ

$\forall X\ person(X) \rightarrow \exists Y\ father(Y,X) \qquad \forall X \forall Y\ father(X,Y) \rightarrow person(X)$

$chase(D,\Sigma) = D \cup \{father(z_1,\text{john}), person(z_1), father(z_2,z_1), \ldots\}$

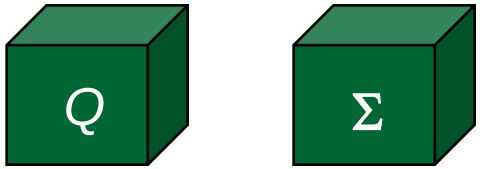
Query Answering via Chase



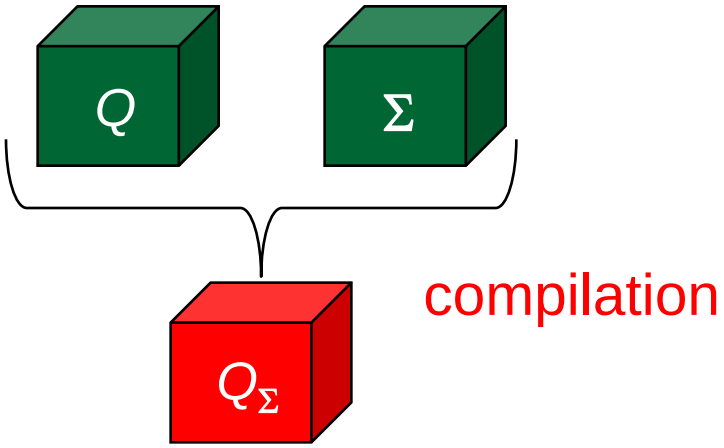
$D \cup \Sigma \models Q \quad \Leftrightarrow \quad chase(D,\Sigma) \models Q$

[see, e.g., Deutsch, Nash & Remmel, PODS 08]

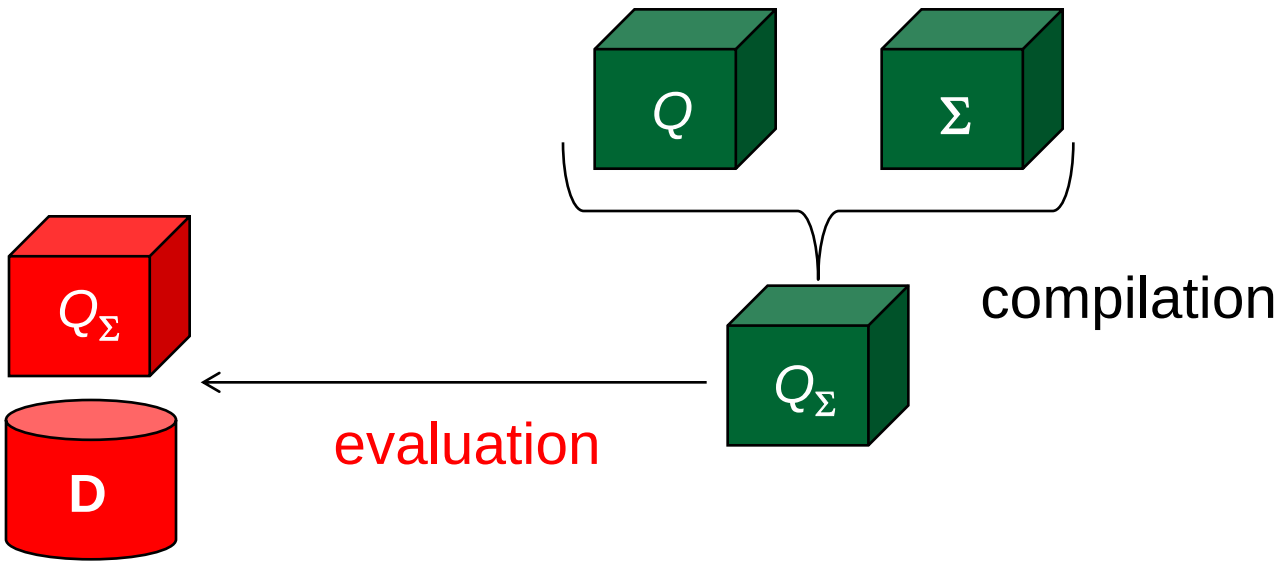
Query Answering via Rewriting



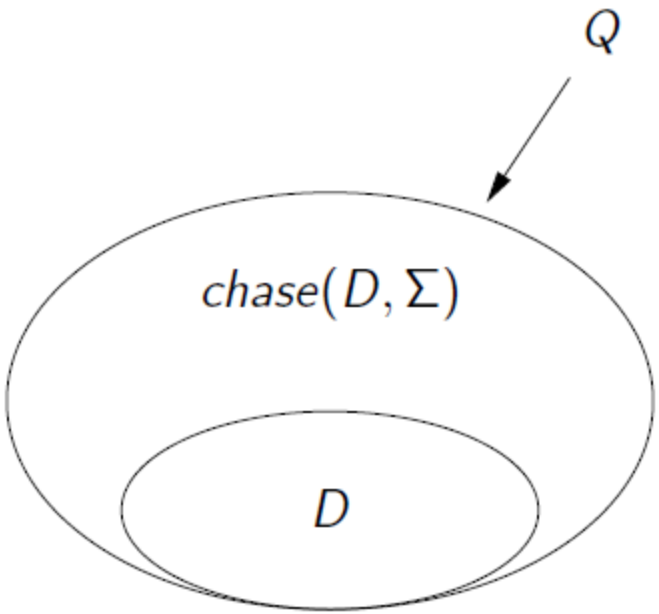
Query Answering via Rewriting

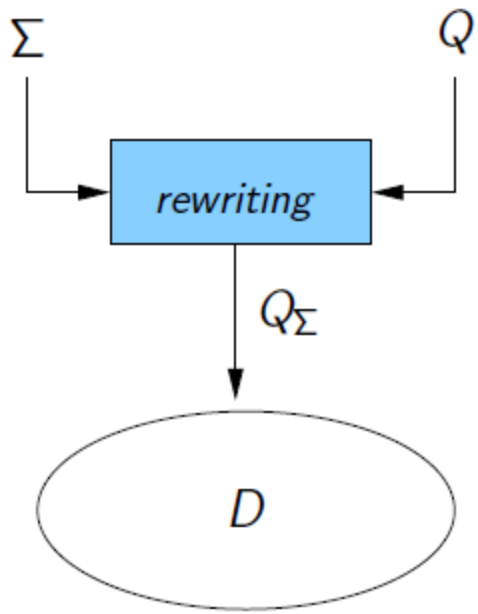


Query Answering via Rewriting



Chase vs Rewriting





Linear TGDs

$$\forall \mathbf{X} \forall \mathbf{Y}\ r(\mathbf{X},\mathbf{Y}) \rightarrow \exists \mathbf{Z}\ \Psi(\mathbf{X},\mathbf{Z})$$

single body atom

- Properly generalize inclusion dependencies.
- Enjoy the bounded-derivation depth property.
- FO-rewritable → Query Answering in **AC0** (data complexity).

FO-rewritability: example [Gottlob et Al., ICDE 11]

Σ promoter(X) $\rightarrow \exists$Y promotesTo(X,Y)
promotesTo(X,Y) $\rightarrow$ customer(Y)

Q q $\leftarrow$ promotesTo(A,B), customer(B)

Q_Σ

q $\leftarrow$ promotesTo(A,B), customer(B) (original query)

FO-rewritability: example [Gottlob et Al., ICDE 11]

Σ $\quad$ promoter(X) $\rightarrow \exists$Y promotesTo(X,Y)
promotesTo(X,Y) $\rightarrow$ customer(Y)

Q $\quad$ q $\leftarrow$ promotesTo(A,B), customer(B)

Q_Σ

q $\leftarrow$ promotesTo(A,B), customer(B) $\quad$ { Y = B }

q $\leftarrow$ promotesTo(A,B), customer(V_0,B) $\quad$ (V_0 is fresh)

FO-rewritability: Example [Gottlob et Al., ICDE 11]

Σ $\quad$ $\text{promoter}(X) \rightarrow \exists Y\ \text{promotesTo}(X,Y)$

$\text{promotesTo}(X,Y) \rightarrow \text{customer}(Y)$

Q $\quad$ $q \leftarrow \text{promotesTo}(A,B), \text{customer}(B)$

Q_Σ

$q \leftarrow \text{promotesTo}(A,B), \text{customer}(B)$

factorization

$q \leftarrow \text{promotesTo}(A,B), \text{promotesTo}(V_0,B) \quad \longrightarrow \quad \text{ans}(A) \leftarrow \text{promotesTo}(A,B)$

$\{ A = V_0 \}$

FO-rewritability: example [Gottlob et Al., ICDE 11]

Σ promoter(X) $\rightarrow \exists$Y promotesTo(X,Y)
promotesTo(X,Y) $\rightarrow$ customer(Y)

Q q $\leftarrow$ promotesTo(A,B), customer(B)

Q_Σ

q $\leftarrow$ promotesTo(A,B), customer(B)

q $\leftarrow$ promotesTo(A,B) {X = A, Y = B}

q $\leftarrow$ promoter(A)

FO-rewritability: example [Gottlob et Al., ICDE 11]

Σ promoter(X) $\rightarrow \exists$Y promotesTo(X,Y)
promotesTo(X,Y) $\rightarrow$ customer(Y)

Q q $\leftarrow$ promotesTo(A,B), customer(B)

Q_Σ

q $\leftarrow$ promotesTo(A,B), customer(B)

q $\leftarrow$ promotesTo(A,B)

q $\leftarrow$ promoter(A)

UCQ rewriting
(first-order)

FO-rewritability

- Desirable properties of a FO-rewriting:
  - → independent on the DB
  - → executable by any DBMS
  - → easy to compute (e.g., polynomial time)
  - → small size (e.g., polynomial size)

FO-rewritability

- Desirable properties of a FO-rewriting:
  - → independent on the DB
  - → executable by any DBMS
  - → easy to compute (e.g., polynomial time)
  - → small size (e.g., polynomial size)

- Unions of Conjunctive Queries (UCQs)
  - → executable by any DBMS
  - → DB independent
  - → easy to optimize and distribute
  - → worst-case exponential size in Q and Σ

Calvanese et Al, JAR 07
Perez Urbina et Al, JAL 09
Cali' et Al, PODS 09
Gottlob et Al, ICDE 11
and others…

FO-rewritability

- Combined and hybrid FO-rewriting
  - → good computational properties (e.g., polynomial in size)
  - → requires access to the DB

Perez Urbina et Al, JAL 09
Kontchakov et Al., KR 10
Gottlob and Schwentick, DL 11

FO-rewritability

- Combined and hybrid FO-rewriting
  - → good computational properties (e.g., polynomial in size)
  - → requires access to the DB

Perez Urbina et Al, JAL 09
Kontchakov et Al., KR 10
Gottlob and Schwentick, DL 11

- Purely intensional Datalog rewriting
  - → very compressed representation
  - → purely intensional
  - → requires view-creation or Datalog engine

Perez Urbina et Al, JAL 09
Rosati and Almatelli., KR 10

Datalog Rewriting: Keep it First-Order!

- A Datalog query is (in general) not a first-order query
  - → a non-recursive Datalog query is a first-order query
  - → a bounded Datalog query is a first-order query

Datalog Rewriting: Keep it First-Order!

- A Datalog query is (in general) not a first-order query
  - → a non-recursive Datalog query is a first-order query
  - → a bounded Datalog query is a first-order query

- Input:
  - → a (w.l.o.g. boolean) conjunctive query $Q = \langle q, \rho \rangle$

$$Q : q(X) \leftarrow p(X), s(X,Y) \Leftrightarrow \langle q, q(X) \leftarrow p(X), s(X,Y) \rangle$$

  - → a set of linear TGDs Σ

- Output:
  - → a bounded Datalog query $Q_\Sigma = \langle q, \pi_\Sigma \rangle$

Datalog Rewriting: skolemization (and renaming)

Σ

$r(X,Y) \rightarrow \exists Z\ s(Y,Z)$

$s(X,Y) \rightarrow \exists Z\ p(Y,Y,Z)$

$p(X,Y,Z) \rightarrow t(Z)$

Datalog Rewriting: skolemization (and renaming)

Σ

$r(X,Y) \rightarrow \exists Z\ s(Y,Z)$

$s(X,Y) \rightarrow \exists Z\ p(Y,Y,Z)$

$p(X,Y,Z) \rightarrow t(Z)$

Σ_f

$r(X_1,Y_1) \rightarrow s(Y_1,f_1(Y_1))$

$s(X_2,Y_2) \rightarrow p(Y_2,Y_2,f_2(Y_2))$

$p(X_3,Y_3,Z_3) \rightarrow t(Z_3)$

Datalog Rewriting: Skolemization (and renaming)

Σ

$r(X,Y) \rightarrow \exists Z\ s(Y,Z)$

$s(X,Y) \rightarrow \exists Z\ p(Y,Y,Z)$

$p(X,Y,Z) \rightarrow t(Z)$

Σ_f

$r(X_1,Y_1) \rightarrow s(Y_1,f_1(Y_1))$

$s(X_2,Y_2) \rightarrow p(Y_2,Y_2,f_2(Y_2))$

$p(X_3,Y_3,Z_3) \rightarrow t(Z_3)$

- Σ_f and Σ are equisatisfiable (not equivalent)
- Introduce one Skolem function for each existential variable

Datalog Rewriting: Rule Saturation

– Apply resolution inference rule to rules in Σ_f

→ at least one of the rules contains Skolem terms

Σ_f

$\delta_1 : r(X_1,Y_1) \rightarrow s(Y_1,f_1(Y_1))$

$\delta_2 : s(X_2,Y_2) \rightarrow p(Y_2,Y_2,f_2(Y_2))$

$\delta_3 : p(X_3,Y_3,Z_3) \rightarrow t(Z_3)$

Datalog Rewriting: Rule Saturation

– Apply resolution inference rule to rules in Σ_f

→ at least one of the rules contains Skolem terms

Σ_f

$\delta_1 : r(X_1,Y_1) \rightarrow s(Y_1,f_1(Y_1))$

$\delta_2 : s(X_2,Y_2) \rightarrow p(Y_2,Y_2,f_2(Y_2))$

$\delta_3 : p(X_3,Y_3,Z_3) \rightarrow t(Z_3)$

$\Longleftrightarrow$

$[\Sigma_f]$

…

$r(X_1,Y_1) \rightarrow p(f_1(Y_1),f_1(Y_1),f_2(f_1(Y_1)))$

…

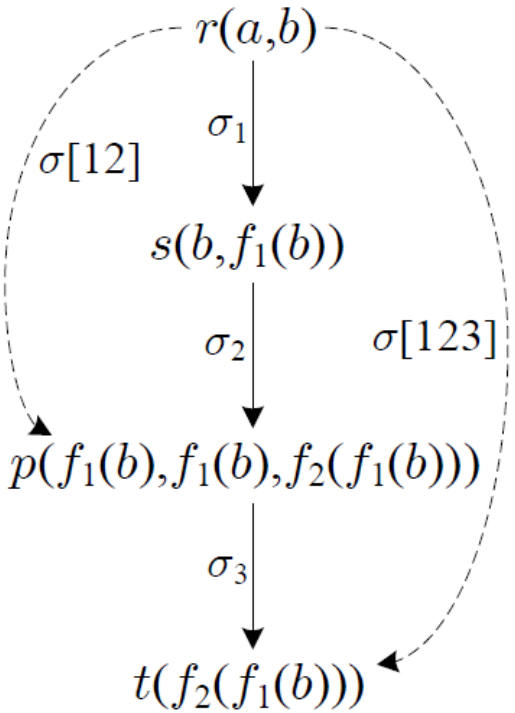
Datalog Rewriting: Properties of Rule Saturation

– $[\Sigma_f]$ *mimics* the chase derivations.

Datalog Rewriting: Properties of Rule Saturation

– **$[\Sigma_f]$** *mimics* the chase derivations.

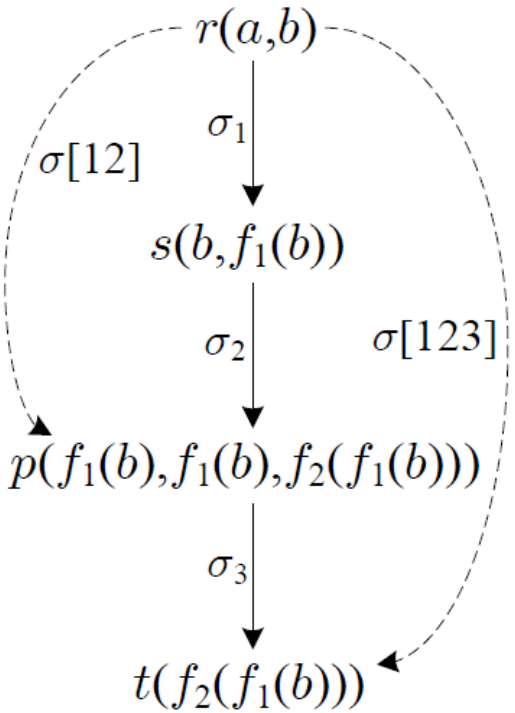
$\delta_1 : r(X_1,Y_1) \rightarrow s(Y_1,f_1(Y_1))$
$\delta_2 : s(X_2,Y_2) \rightarrow p(Y_2,Y_2,f_2(Y_2))$
$\delta_3 : p(X_3,Y_3,Z_3) \rightarrow t(Z_3)$



Datalog Rewriting: Properties of Rule Saturation

– $[\Sigma_f]$ *mimics* the chase derivations.

$\delta_1 : r(X_1,Y_1) \rightarrow s(Y_1,f_1(Y_1))$
$\delta_2 : s(X_2,Y_2) \rightarrow p(Y_2,Y_2,f_2(Y_2))$
$\delta_3 : p(X_3,Y_3,Z_3) \rightarrow t(Z_3)$



– $[\Sigma_f]$ depends only on Σ.

– $[\Sigma_f]$ is possibly infinite

→linear TGDs have BDDP: suffices to construct it up to k steps $[\Sigma_f]_k$.

Datalog Rewriting: Query Saturation

– resolve $[\Sigma_f]$ with the query Q.

→ use only rules with Skolem terms.

Datalog Rewriting: Query Saturation

– resolve $[\Sigma_f]$ with the query Q.

→ use only rules with Skolem terms.

$[\Sigma_f]$

…

$\delta_1 : r(X_1,Y_1) \rightarrow s(Y_1,f_1(Y_1))$

$\delta_2 : s(X_2,Y_2) \rightarrow p(Y_2,Y_2,f_2(Y_2))$

$\delta_3 : p(X_3,Y_3,Z_3) \rightarrow t(Z_3)$

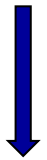
…

$[\delta_{12}] : r(X_1,Y_1) \rightarrow p(f_1(Y_1),f_1(Y_1),f_2(f_1(Y_1)))$

…

Q

$Q \leftarrow s(A,B), p(B,B,C)$



…

$[Q,\Sigma_f]$ $\quad Q \leftarrow r(X_1,Y_1), p(f_1(Y_1), f_1(Y_1),C)$

…

Datalog Rewriting: Query Saturation

– *bypasses* chase derivations with function symbols

Q

$Q \leftarrow s(A,B), p(B,B,C)$

$[\Sigma_f]$

…

$\delta_1 : r(X_1,Y_1) \rightarrow s(Y_1,f_1(Y_1))$

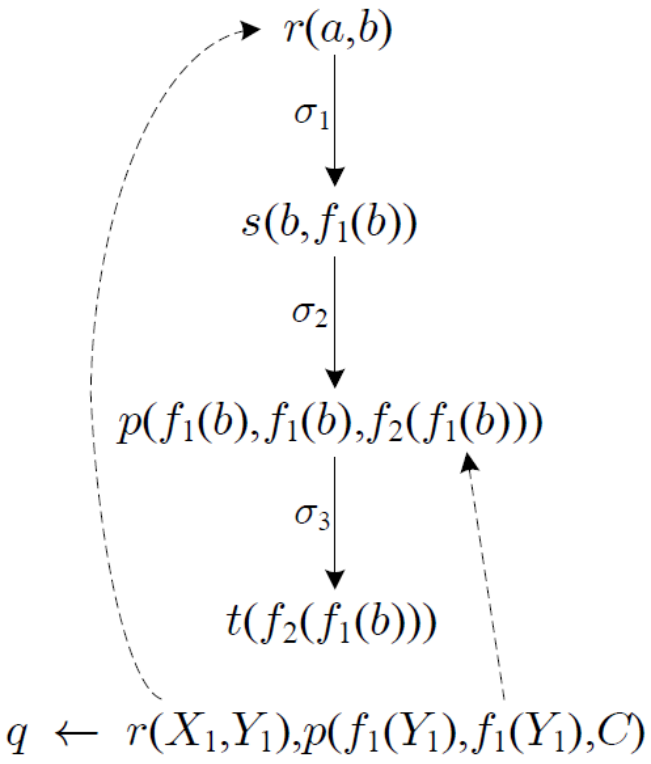
$\delta_2 : s(X_2,Y_2) \rightarrow p(Y_2,Y_2,f_2(Y_2))$

$\delta_3 : p(X_3,Y_3,Z_3) \rightarrow t(Z_3)$

…

$[\delta_{12}] : r(X_1,Y_1) \rightarrow p(f_1(Y_1),f_1(Y_1),f_2(f_1(Y_1)))$

…



Datalog Rewriting: Finalization

- keep only the function-free rules from $[\Sigma_f] \cup [Q,\Sigma_f]$

- derivations producing certain answers are captured by function-symbol-free rules.

Optimizations: Pruning

– use the predicate graph to reduce the number of rules in Σ_f

Σ_f

$\delta_1 : r(X_1,Y_1) \rightarrow s(Y_1,f_1(Y_1))$
$\delta_2 : s(X_2,Y_2) \rightarrow p(Y_2,Y_2,f_2(Y_2))$
$\delta_3 : p(X_3,Y_3,Z_3) \rightarrow t(Z_3)$

Q

$Q \leftarrow s(A,B), p(B,B,C)$

Optimizations: Pruning

– use the predicate graph to reduce the number of rules in Σ_f

Σ_f

$\delta_1 : r(X_1,Y_1) \rightarrow s(Y_1,f_1(Y_1))$

$\delta_2 : s(X_2,Y_2) \rightarrow p(Y_2,Y_2,f_2(Y_2))$

~~$\delta_3 : p(X_3,Y_3,Z_3) \rightarrow t(Z_3)$~~

Q

$Q \leftarrow s(A,B), p(B,B,C)$

Optimizations: Pruning

– use the predicate graph to reduce the number of rules in Σ_f

Σ_f

$\delta_1 : r(X_1,Y_1) \rightarrow s(Y_1,f_1(Y_1))$

$\delta_2 : s(X_2,Y_2) \rightarrow p(Y_2,Y_2,f_2(Y_2))$

~~$\delta_3 : p(X_3,Y_3,Z_3) \rightarrow t(Z_3)$~~

Q

$Q \leftarrow s(A,B), p(B,B,C)$

– we are no longer independent on Q!

Optimizations: Query Elimination

– eliminate implied atoms during query saturation

Σ_f

$\delta_1 : r(X_1,Y_1) \rightarrow s(Y_1,f_1(Y_1))$

$\delta_2 : s(X_2,Y_2) \rightarrow p(Y_2,Y_2,f_2(Y_2))$

Q

$Q \leftarrow s(A,B),\ p(B,B,C)$

Optimizations: Query Elimination

– eliminate implied atoms during query saturation

Σ_f

$\delta_1 : r(X_1,Y_1) \rightarrow s(Y_1,f_1(Y_1))$

$\delta_2 : s(X_2,Y_2) \rightarrow p(Y_2,Y_2,f_2(Y_2))$

Q

$Q \leftarrow s(A,B), p(B,B,C)$

$s(A,B) \models_\Sigma p(B,B,C)$ atom coverage

Optimizations: Query Elimination

– eliminate implied atoms during query saturation

Σ_f

$\delta_1 : r(X_1,Y_1) \rightarrow s(Y_1,f_1(Y_1))$

$\delta_2 : s(X_2,Y_2) \rightarrow p(Y_2,Y_2,f_2(Y_2))$

Q

$Q \leftarrow s(A,B), p(B,B,C)$

$s(A,B) \models_\Sigma p(B,B,C)$ atom coverage

$Q \leftarrow s(A,B), p(B,B,C) \equiv_\Sigma Q \leftarrow s(A,B)$

Optimizations: Query Elimination

- atom coverage under linear TGDs can be checked in polynomial time
  → see paper.

- unique elimination strategy (w.r.t. the final size of the rewriting)
  → see paper.

- given m = |body(ρ)| and n = |Σ|
  → worst-case size of $\mathbf{[\Sigma_f]} \cup \mathbf{[Q,\Sigma_f]}$ is $O((n \cdot m)^m)$
  → worst-case size of $Q_\Sigma = \langle q, \pi_\Sigma \rangle$ is $O(n+m)^m$

Experimental Results

Ontology	Query	Size				
		QO	RQ	NY	PR	NY^{DTG}
A	Q_1	783	402	249	69	58
	Q_2	1812	103	94	52	41
	Q_3	4763	104	104	55	43
	Q_4	7251	492	456	93	81
	Q_5	66068	624	624	71	65
U	Q_1	5	2	2	6	5
	Q_2	287	148	1	1	1
	Q_3	1260	224	4	8	7
	Q_4	5364	1628	2	6	5
	Q_5	9245	2960	2	11	10
S	Q_1	6	6	6	7	7
	Q_2	204	160	1	3	3
	Q_3	1194	480	2	5	4
	Q_4	1632	960	2	5	4
	Q_5	11487	2880	4	7	6
P5X	Q_1	14	14	10	11	11
	Q_2	86	77	57	16	16
	Q_3	530	390	324	16	16
	Q_4	3,476	1,953	1,642	16	16
	Q_5	23,744	9,766	8,219	16	16

Discussion

- Datalog rewriting is substantially more compact than UCQ rewriting.

- Unclear whether this always leads to increase in performance.

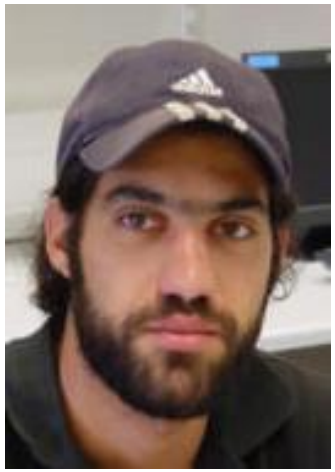
- Extend the procedure to larger classes of TGDs
  → guarded TGDs [Cali' et Al, PODS 09] → non FO-rewritable
  → sticky-join TGDs [Cali' et Al, VLDB 10]

Thank you!

The **Datalog**$^{\pm}$ Family

Georg Gottlob

Thomas Lukasiewicz



Andreas Pieris

Andrea Calì

Giorgio Orsi