The Quest for Definitive Results: Two Examples in Georg Gottlob's Work

Phokion G. Kolaitis

UC Santa Cruz & IBM Research - Almaden

GC@60 Workshop in Genoa, December 2016

From “Reflections on Georg Gottlob” in the Gottlobiana volume:

“When one reflects on Georg's scientific output, one sees clearly that his work is characterized by scholarship of the highest caliber, penetrating conceptual insights, technical prowess, impeccable taste, and a unique ability to obtain definitive results.

Georg never settles for partial results; instead, he is always after complete classifications, full taxonomies, and crisp boundaries that completely settle the question at hand.”

Two Examples of the Quest for Definitive Results

- The Complexity of Acyclic Conjuctive Queries

  Georg Gottlob, Nicola Leone, Francesco Scarcello

  JACM 2001 (67 pages)

  – Preliminary version in FOCS 1998

- Efficient Core Computation in Data Exchange

  Georg Gottlob and Alan Nash

  JACM 2008 (49 pages)

  – Preliminary versions in:

    PODS 2005 (Georg Gottlob)

    PODS 2006 (Georg Gottlob and Alan Nash)

Conjunctive Query Evaluation

- The Conjunctive Query Evaluation Problem (CQE):
  Given a Boolean conjunctive query Q and a database D, is Q(D) = 1? (i.e., does D satisfy Q?)

- Fact: CQE is NP-complete
  G has a clique of size k if and only if $Q_k(G) = 1$, where

$$Q_k \text{ :- } \exists x_1 \ldots \exists x_k \bigwedge_{i \neq j} E(x_i,x_j)$$

The Pursuit for Islands of Tractability

- Extensive pursuit of tractable cases of CQE by the database theory community and the constraint satisfaction community

Conjunctive Query Evaluation
$\equiv$ (Chandra-Merlin, 1977)
Homomorphism Problem
$\equiv$ (Feder-Vardi, 1993)
Constraint Satisfaction Problem

- In 1981, Mihalis Yannakakis discovered a large and useful tractable case of CQE by showing that CQE is tractable for Acyclic Conjunctive Queries.

Acyclic Conjunctive Queries

Definition: A conjunctive query Q is acyclic if it has a join tree.

Definition: Let Q be a conjunctive query of the form

$$Q(\mathbf{x}) : \exists \mathbf{y} \, (R_1(\mathbf{z}_1) \wedge R_2(\mathbf{z}_2) \wedge \ldots \wedge R_m(\mathbf{z}_m)).$$

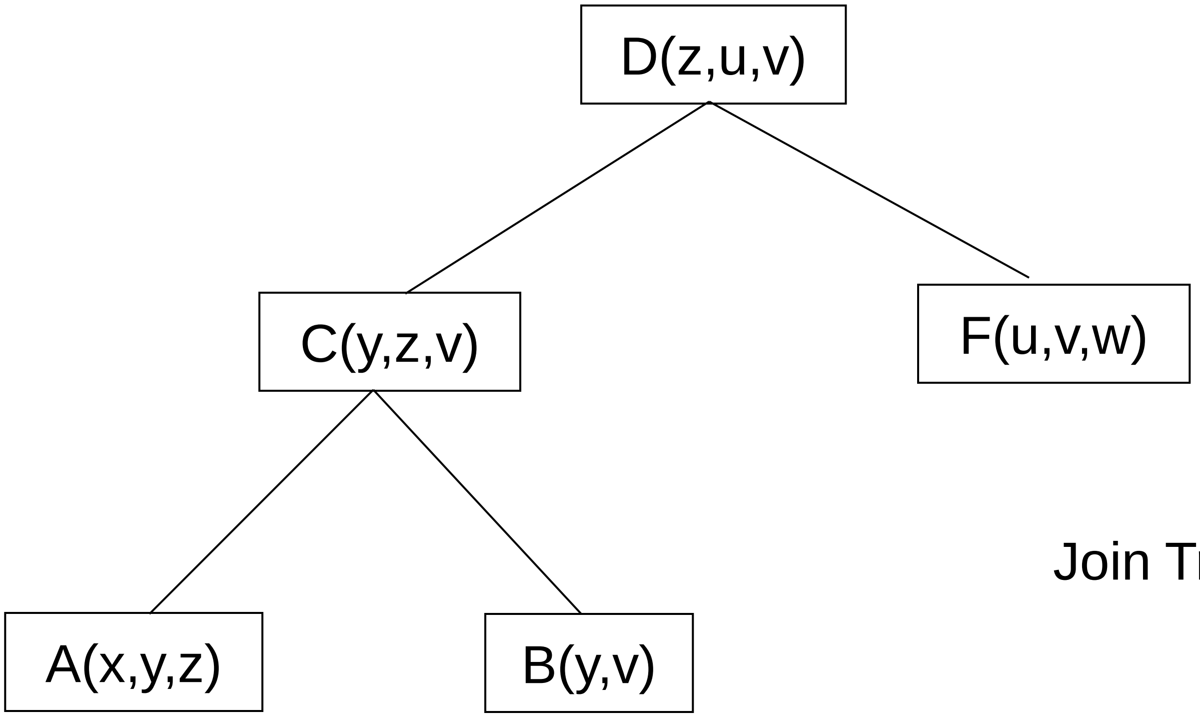
A join tree for Q is a tree T such that

- The nodes of T are the atoms $R_i(\mathbf{z}_i)$, $1 \leq i \leq m$, of Q.
- For every variable w occurring in Q, the set of the nodes of T that contain w forms a subtree of T;

  in other words, if a variable w occurs in two different atoms $R_j(\mathbf{z}_j)$ and $R_k(\mathbf{z}_k)$ of Q, then it occurs in each atom on the unique path of T joining $R_j(\mathbf{z}_j)$ and $R_k(\mathbf{z}_k)$.

Acyclic Conjunctive Queries

$$Q(\,):\quad \exists\, x\ y\ z\ u\ v\ w$$
$$(A(x,y,z) \wedge B(y,v) \wedge C(y,z,v) \wedge D(z,u,v) \wedge F(u,v,w))$$

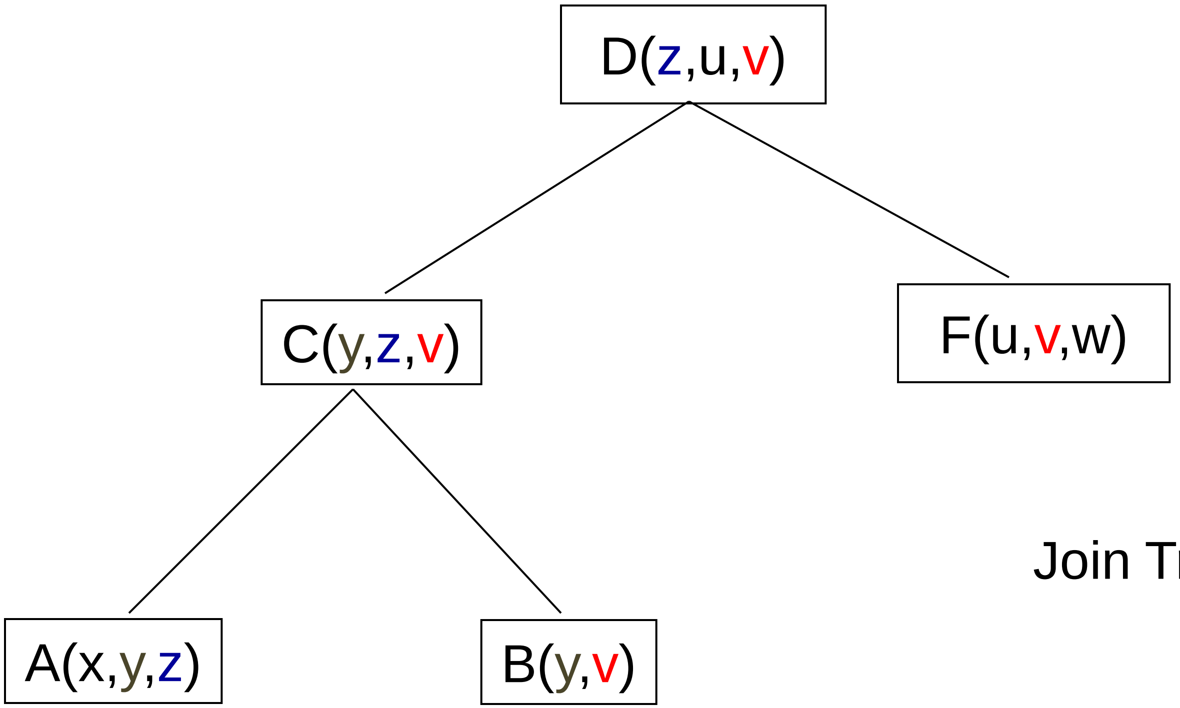


Join Tree for Q

Acyclic Conjunctive Queries

$$Q(\,):\ \exists\, x\ y\ z\ u\ v\ w$$

$$(A(x,y,z) \wedge B(y,v) \wedge C(y,z,v) \wedge D(z,u,v) \wedge F(u,v,w))$$



Join Tree for Q

Yannakakis' PTIME-Algorithm for Acyclic CQE

Dynamic Programming Algorithm

Input: Acyclic Boolean conjunctive query Q, database D

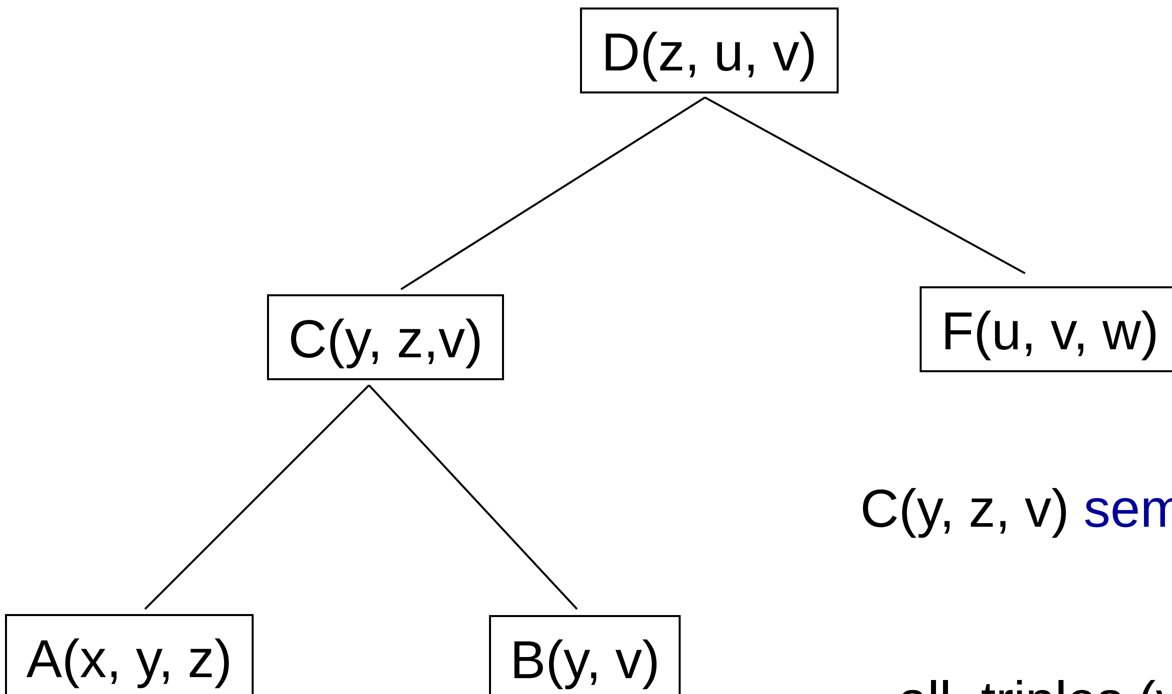
1. Construct a join tree T of Q
2. Populate the nodes of T with the matching relations of D.
3. Traverse the tree T bottom up:

   For each node $R_k(z_k)$, compute the semi-joins of the (current) relation in the node $R_k(z_k)$ with the (current) relations in the children of the node $R_k(z_k)$.
4. Examine the resulting relation R at the root of T
   - If R is non-empty, then output Q(D) = 1 (D satisfies Q).
   - If R is empty, then output Q(D) = 0 (D does not satisfy Q).

Yannakakis' PTIME-Algorithm for Acyclic CQE

Q( ) : $\exists$ x y z u v w

$$(A(x,y,z) \wedge B(y,v) \wedge C(y,z,v) \wedge D(z,u,v) \wedge F(u,v,w))$$



C(y, z, v) semi-join A(x, y, z)

=

all triples (y, z, v) in C that "match" a triple (x, y, z) in A

The World inside P

Suppose that a decision problem is shown to be in P.

- Is the problem P-complete (under logspace-reductions)?
  - Is the problem inherently sequential?
  - Note that Yannakakis' algorithm is sequential.

- Does the problem belong to some complexity class inside P?
  - There is a rich world of complexity classes inside P
    - L and NL Deterministic and non-deterministic logspace
    - Parallel complexity classes (fast parallel time)

$$NC = \bigcup_i NC^i, \text{ where}$$

NC^i = The class of decision problems solvable in $O(\log^i(n))$-time using polynomially-many processors

Fact: $NC^1 \subseteq L \subseteq NL \subseteq NC^2 \subseteq \ldots \subseteq NC \subseteq P$

The Complexity of Acyclic CQE

Question: What is the exact complexity of Acyclic CQE?

Theorem (Dalhaus – 1990)
Acyclic CQE is in NC^2 (hence, Acyclic CQE is parallelizable).

Theorem (Gottlob, Leone, Scarcello – 1998)
Acyclic CQE is LOGCFL-complete, where

- LOGCFL is the class of all decision problems having a logspace-reduction to some context-free language.

Fact: $NC^1 \subseteq L \subseteq NL \subseteq LOGCFL \subseteq NC^2 \subseteq \ldots \subseteq NC \subseteq P$

LOGCFL

Definition: LOGCFL is the class of all decision problems having a logspace-reduction to some context-free language.

Fact: LOGCFL was known to have complete problems:

– Greibach's hardest context-free language L_0 (1973):

(every context-free lang. is an inverse homomorphic image of L_0)

Fact: No problem from databases was known to be LOGCFL-complete, prior to the Gottlob-Leone-Scarcello paper.

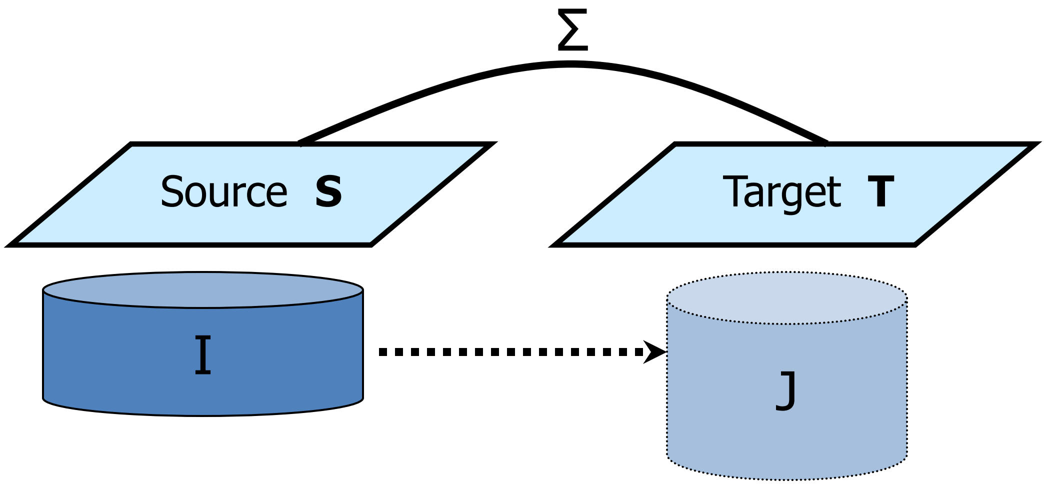
The Complexity of Acyclic CQE: Definitive Result

Theorem (Gottlob, Leone, Scarcello – 1998)
Acyclic CQE is LOGCFL-complete.
-- Upper Bound: Acyclic CQE is solvable via a non-deterministic auxiliary push-down automaton in logspace and PTIME.
Join tree is processed top-down (unlike Yannakakis' algorithm)
-- Lower Bound: Reduction from SAC^1 circuits:
logspace-uniform semi-unbounded circuits of O(log(n))-depth.

Theorem (Gottlob, Leone, Scarcello – 1998)
The following problems are LOGCFL-complete:

- Acyclic Boolean Conjunctive Query Containment.
- Acyclic Constraint Satisfaction.

Schema Mappings & Data Exchange

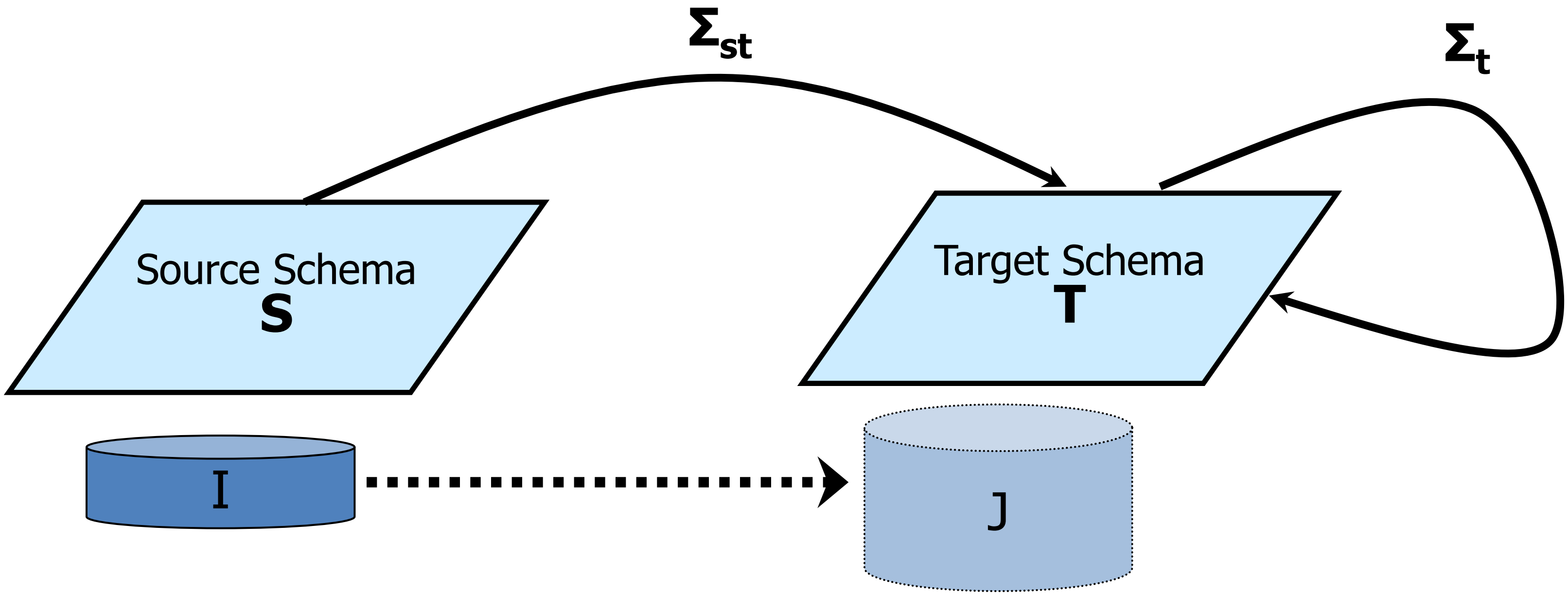


- Schema Mapping **M** = (**S**, **T**, Σ)
  - Source schema **S**, Target schema **T**
  - Set Σ of declarative assertions about **S** and **T**.

- Data Exchange via the schema mapping **M** = (**S**, **T**, Σ)

  Transform a given source instance I to a target instance J so that <I, J> satisfy the specifications Σ of **M**.
  - Such a J is called a solution for I w.r.t. **M**.

Formalization of Data Exchange



Fagin, K …, Miller, Popa (2003)

Schema Mapping **M** = (**S**, **T**, Σ_{st} , Σ_t), where

- Σ_{st} is a set of source-to-target tgds
- Σ_t is a set of target tgds and target egds

Schema-Mapping Language

- Source-to-Target Tuple Generating Dependencies (s-t tgds)

  $\forall \mathbf{x}\ (\varphi(\mathbf{x}) \rightarrow \exists \mathbf{y}\ \psi(\mathbf{x}, \mathbf{y}))$, where

  - $\varphi(\mathbf{x})$ is a conjunction of atoms over the source;
  - $\psi(\mathbf{x}, \mathbf{y})$ is a conjunction of atoms over the target.

  Example:

  $(\text{Student}(s) \wedge \text{Enrolls}(s,c)) \rightarrow \exists t\ \exists g\ (\text{Teaches}(t,c) \wedge \text{Grade}(s,c,g))$

- Target Tgds : $\forall \mathbf{x}\ (\varphi_T(\mathbf{x}) \rightarrow \exists \mathbf{y}\ \psi_T(\mathbf{x}, \mathbf{y}))$
- Target Equality Generating Dependencies (target egds)

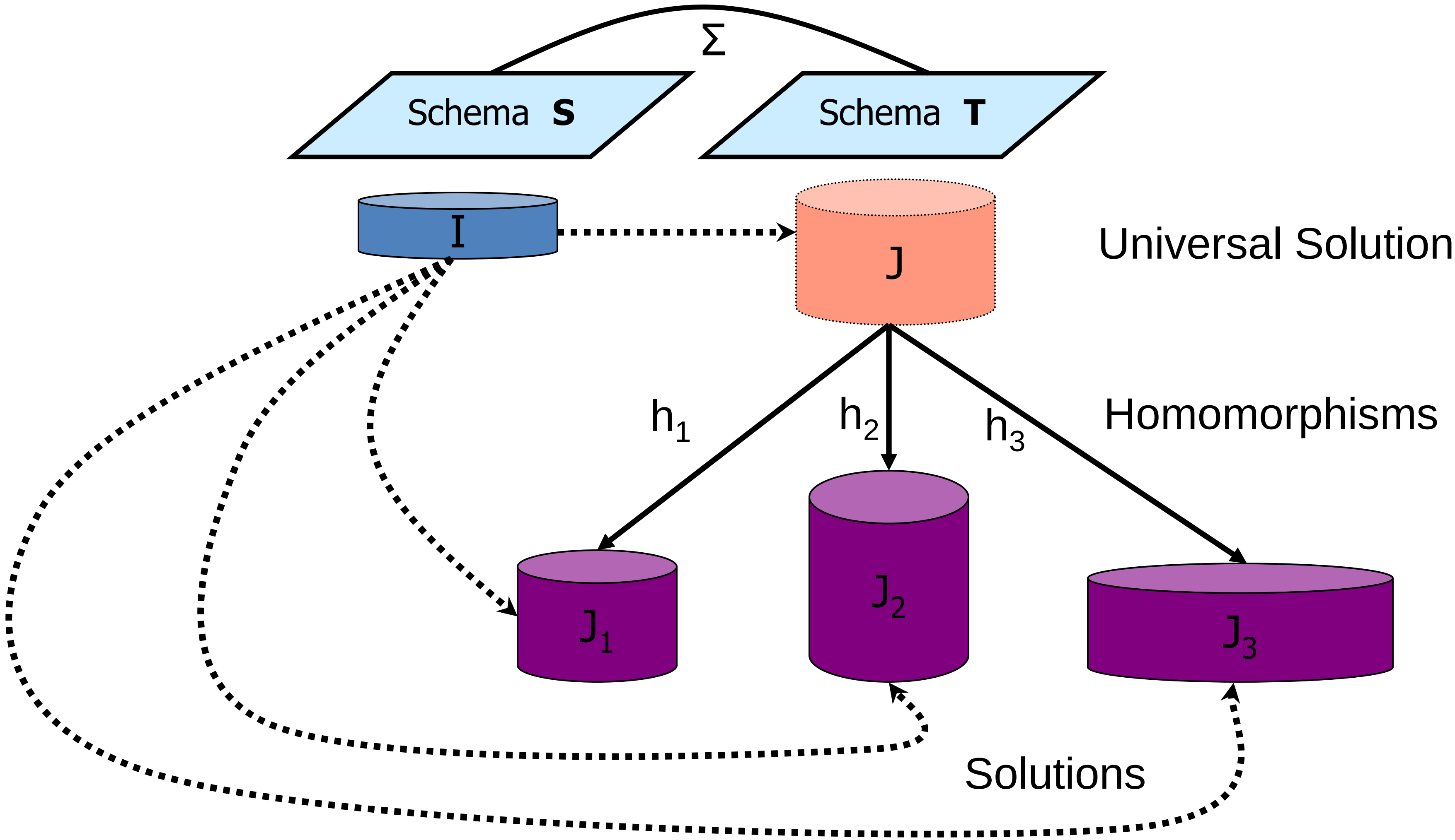
  $\forall \mathbf{x}\ (\varphi_T(\mathbf{x}) \rightarrow (x_1 = x_2))$

  $\forall e\ \forall d_1\ \forall d_2\ (\text{Mgr}(e, d_1) \wedge \text{Mgr}(e, d_2)) \rightarrow (d_1 = d_2)$

Universal Solutions in Data Exchange

- FKMP introduced the notion of universal solutions as the "best" solutions in data exchange.
- By definition, a solution is universal if it has homomorphisms to all other solutions (thus, it is a "most general" solution).
  - Constants: entries in source instances
  - Variables (labeled nulls): other entries in target instances
  - Homomorphism h: $J_1 \rightarrow J_2$ between target instances:
    - h(c) = c, for constant c
    - If $P(a_1,\ldots,a_m)$ is in $J_{1,}$, then $P(h(a_1),\ldots,h(a_m))$ is in J_2

Universal Solutions in Data Exchange



Algorithmic Properties of Universal Solutions

Theorem (FKMP): Schema mapping **M**= (**S**, **T**, Σ_{st}, Σ_t) such that:

- Σ_{st} is a set of source-to-target tgds;
- Σ_t is the union of a weakly acyclic set of target tgds with a set of target egds.

Then:

- Universal solutions exist if and only if solutions exist.
- PTIME algorithm for the existence-of-solutions problem for **M**: given I, is there J such that J is a solution for I?
- A *canonical* universal solution (if solutions exist) can be produced in polynomial time using the chase procedure.

The Smallest Universal Solution

Fact: Universal solutions are unique up to homomorphic equivalence, but need not be unique up to isomorphism.

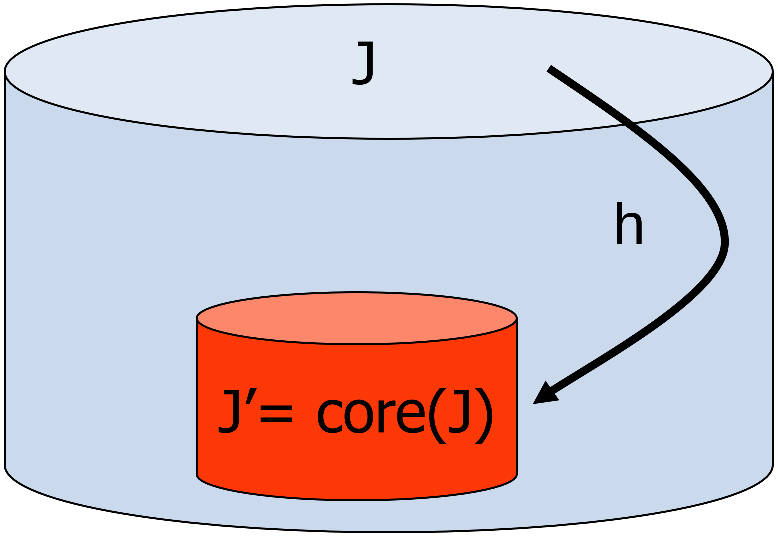
Question: Is there a “best” universal solution?
Answer: Fagin, K …, Popa (PODS 2003): “small is beautiful” approach

Definition: The core of an instance J is the smallest subinstance J’ that is homomorphically equivalent to J.

Proposition: Let **M** = (**S**, **T**, Σ_{st}, Σ_t) be schema mapping.

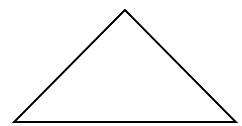
- All universal solutions have the same core.
- The core of the universal solutions is the smallest universal solution (hence, the most compact to materialize).

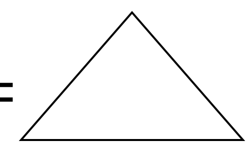
The Core of a Structure



Definition: J′ is the core of J if

- J′ ⊆ J
- there is a hom. h: J → J′
- there is **no** hom. g: J → J″, where J″ ⊂ J′.

Example: If a graph **G** contains a  , then

G is 3-colorable if and only if core(**G**) =  .

Fact: Computing cores of graphs is an NP-hard problem.

Computing the Core - Early Result

Theorem (FKP): **M** = (**S**, **T**, Σ_{st} , Σ_t) a schema mapping such that

- Σ_{st} is a set of s-t tgds
- Σ_t is a set of target egds.

Then the core of universal solutions is polynomial-time computable, i.e.,

there is a polynomial-time algorithm that, given a source instance I, the algorithm computes the core of the universal solutions for I.

Algorithm:

- Step 1: Obtain a target instance J by chasing I with Σ_{st}
- Step 2: Use a greedy algorithm on J and Σ_t to compute the core of the universal solutions for I w.r.t. **M**.

Question: For what schema mappings is the core of the universal solutions polynomial-time computable?

Computing the Core – Definitive Result

Theorem (Gottlob – PODS 2005):

M = (**S**, **T**, Σ_{st}, Σ_t) a schema mapping such that

- Σ_{st} is a set of s-t tgds;
- Σ_t is a set of full target tgds and target egds.

Then the core of universal solutions is polynomial-time computable.

Theorem (Gottlob and Nash 2006):

M = (**S**, **T**, Σ_{st}, Σ_t) a schema mapping such that

- Σ_{st} is a set of source-to-target tgds;
- Σ_t is the union of a weakly acyclic set of target tgds with a set of target egds.

Then the core of universal solutions is polynomial-time computable.

Computing the Core – Definitive Result

Theorem (Gottlob and Nash 2005): **M** = (**S**, **T**, Σ_{st} , Σ_t) such that

- Σ_{st} is a set of source-to-target tgds;
- Σ_t is the union of a weakly acyclic set of target tgds with a set of target egds.

Then the core of universal solutions is polynomial-time computable.

Algorithm: Sophisticated algorithm with several new ideas:

- Systematic use of retractions, instead of endomorphisms.
- Keep track of ancestors and siblings of nulls in chase steps.
- Obtain polynomial-time algorithm for Σ_t with no target egds.
- Simulate target egds using target full tgds.
- Avoid cycles using a particular chase order.

Concluding Remarks

Back to “Reflections on Georg Gottlob” in Gottlobiana

“Brilliant scientist, successful entrepreneur, polyglot, erudite, art lover, wine connoisseur, gracious host, caring husband, proud father. Which of these qualities does Georg Gottlob possess?

To those who have the pleasure to know Georg and the privilege to call him a friend, the answer is simple: all of the above!”