Core Computation for Data Exchange

Vadim Savenkov

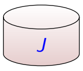
Vienna University of Technology

DEIS 2010
November 9, 2010

Talk Outline

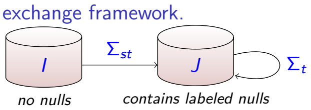
Preliminaries: Labeled nulls and homomorphisms

Consider a database model based on v-relations: unknown values are labeled, and the same label can have several occurrences in a database, unlike the usual SQL nulls ("Codd" tables).



$dom(J) = const(J) \cup var(J)$
$const(J) \cap var(J) = \emptyset$

A basic data exchange framework.



Definition

A homomorphism h between two instances I and J maps $dom(I)$ on $dom(J)$ such that $\forall c \in const(I)\ h(c) = c$, and whenever $R(\bar{x}) \in I$ it holds that $R(h(\bar{x})) \in J$.

Embedded implicational dependencies

Tuple-generating dependencies

- $Employee(Name, Project, Salary) \rightarrow \exists Id \exists Dep\,(Staff(Id, Name, Dep) \wedge Wage(Id, Salary))$
- Source-to-target (st) tgds: How the data must be transferred.
- Target tgds: generalize inclusion / join dependencies.
- Naive chase: $\forall \langle Name, Salary \rangle$ add the instantiation of the conclusion atoms to the db. Replace existential variables by fresh distinct labeled nulls.

Equality-generating dependencies

- $Staff(Id, Name_1, Dep_1) \wedge Staff(Id, Name_2, Dep_2) \rightarrow Dep_1 = Dep_2$
- Generalize functional dependencies.

Chase delivers a canonical universal solution.

Example

τ_{st}^1: $\text{BasicUnit}(C) \rightarrow \text{Course}(Idc, C)$.
τ_{st}^2: $\text{Tutorial}(C, T) \rightarrow \text{Course}(Idc, C), \text{Tutor}(Idt, T), \text{Teaches}(Idt, Itc)$.

BasicUnit('C#') $\Rightarrow$ Course(C_1, 'C#')
Tutorial('C#', 'Joe') $\Rightarrow$ Course(C_2, 'C#'), Tutor(T_1, 'Joe'), Teaches(T_1, C_2)

Formalizing "redundancy"

Endomorphism is a homomorphism from an instance onto itself. If an endomorphism maps an instance onto its proper subset, it is called proper endomorphism. Nulls that can be eliminated by proper endomorphisms are redundnant.

Definition

Let J be an instance. Core of J (denoted $core(J)$) is an endomorphic image of J, for which no proper endomorphism exists.

Cores and endomorphisms

Fundamental paper "Core of a graph" by Hell and Nesetril [1992]

- Cores of any relational structure are isomorphic $\Rightarrow$ "the core"
- Homomorphically equivalent structures have isomorphic cores.
  - Contrast with: typically, there is infinitely many universal solutions for each source instance. (Just add tuples of distinct fresh labeled nulls.) All universal solutions are hom. equivalent.
  - Thus, a single core captures the whole infinite set $USol(I, \mathcal{M})$.

Bet

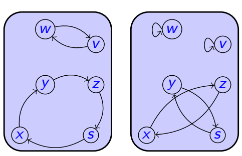
Let Σ be set of tgds and egds, J be an instance satisfying Σ and J' an endomorphic image of J. **Does it hold that $J' \models \Sigma$?**

Consider $\Sigma = \{R(u, w), R(w, w), R(w, v) \rightarrow R(u, v)\}$ and $J = \{(x, z), (x, a), (z, y), (a, z), (a, a)\}$. Let $h = \{z \rightarrow a, y \rightarrow z\}$ be endomorphism, then $h(J) = \{(x, a), (a, z), (a, a)\} \not\models \Sigma$ holds. However, $core(J) = \{(x, a), (a, a)\} \models \Sigma$.

Cores and embedded dependencies

Property ([Hell and Nesetril, 1992])

Let A be a relational structure and C its core. Then, there exists a homomorphism $h: A \rightarrow C$, such that for all $v \in dom(C)$, $h(v) = v$.



Consider a homomorphism $r: A \rightarrow C$. Restricted to $dom(C)$, r is one-to-one (otherwise, C would not be a core).

Let G_r be a graph whose vertices are elements of $dom(C)$, and an edge (x, y) denotes $r(x) = y$. Every edge of such graph belongs to a cycle. For cycle of length n, vertices that occur in it are mapped to themselves by r^n.

Moreover, r^n is still a homomorphism and thus must be one-to-one on C. Now, consider the graph G_{r^n}, etc.

Definition

Idempotent endomorphism, i.e. r such that $r(r(x)) = r(x)$, for all x is called a retraction. Any endomorphism can be transformed into a retraction simply by iterating it long enough.

As we just showed, core of a structure is a retract.

Theorem (Fagin, Kolaitis, and Popa [2005b])

Let $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{st} \cup \Sigma_t)$ be a mapping where Σ_{st} is a set of source-to-target tgds, and Σ_t consists of target tgds and egds. Then, if $J \in Sol(I, \mathcal{M})$, and J' is a retract of J, then also $J' \in Sol(I, \mathcal{M})$.

Proof (Excerpt).

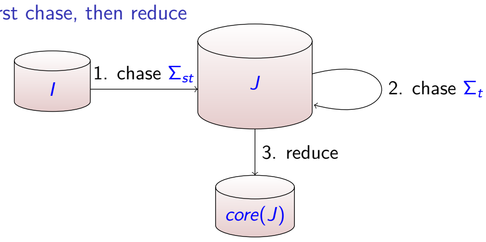
Consider a target tgd $\tau: \phi(\bar{x}) \rightarrow (\exists \bar{y})\psi(\bar{x}, \bar{y})$ in Σ_t. To show: $J' \models \tau$. Assume that for some $\bar{a}$, $J' \models \phi(\bar{a})$. Then, by $J \models \tau$, $\exists \bar{b} \in dom(J)$ such that $J \models \psi(\bar{a}, \bar{b})$. J' being a retract, means there exists $h: J \rightarrow J'$ such that $\forall v \in var(J')\ h(v) = v$.
Hence, $J' \models \psi(h(\bar{a}), h(\bar{b}))$. Since $h(\bar{a}) = \bar{a}$, we have $J' \models \psi(\bar{a}, h(\bar{b}))$ and thus, also $J' \models \tau$. □

Timeline

- 2003 "Getting to the core" paper by Fagin, Kolaitis, and Popa at PODS *(TODS version: 2005)*. Introduced cores in the context of data exchange. ST tgds + target egds.
- 2005 In his PODS paper, Gottlob addresses full target tgds (very tricky!).
- 2006 "Computing cores in polynomial time" paper by Gottlob and Nash *(JACM version: 2010)* Weakly-acyclic sets of target tgds + egds (simulated by full tgds).
- 2008 Pichler and S. add direct support for target egds along with weakly acyclic sets of tgds. *(LPAR, TCS version: 2010)*
- 2009 (i) SIGMOD paper by Mecca, Papotti and Raunich, and "Laconic Schema Mappings" @ VLDB by ten Cate, Chiticariu, Kolaitis, and Tan. Computing cores directly, as part of the chase; no target constraints. (ii) PODS paper by Marnette presents a robust core-based semantics for data exchange.
- 2010 Marnette, Mecca and Papotti consider direct core computation under target functional dependencies. *(VLDB)*.

Core Computation as a Postprocessing Step

First chase, then reduce



+ Most general approach (handles also target constraints)

- Performance

Greedy algorithm [Fagin et al., 2005b], target egds

Input: Source instance I, st tgds Σ_{st}, target egds Σ_t
Output: A core of a universal solution for I under $\Sigma_{st} \cup \Sigma_t$

(1) Chase I with $\Sigma_{st} \Rightarrow$ Canonical pre-universal instance $\tilde{J}$.

(2) Chase $\tilde{J}$ with Σ_t
If the chase fails $\Rightarrow$ stop and return "failure";
otherwise, let J be a canonical universal solution.

(3) Initialize J^* to be J.

(4) While there is a fact $R(\bar{x}) \in J^*$ such that $\langle I, J^* - \{R(\bar{x})\}\rangle \models \Sigma_{st}$, set J^* to be $J^* - \{R(\bar{x})\}$.

(5) Return J^*.

Question

As is, works only with target egds. Why?

- source instance has to be available

Descent to the core via proper retractions

- As we have shown, a retract of a solution is itself a solution.
- Moreover, the core of a structure is unique (up to isomorhpism).

$\Rightarrow$ Compute an ever shrinking sequence of proper retractions: J, $r_1(J)$, $r_2(r_1(J)), \ldots$

Retracts are solutions, so no need to test $\langle I, r_n(J)\rangle \models \Sigma$

- How to find a proper retraction? Iterate a proper endomorphism.
- How to find a proper endomorphism? For general structures, we are likely to need exhaustive search.
  - CoreIdentification is DP-complete [Fagin et al., 2005b]
  - CoreRecognition is coNP-complete [Fagin et al., 2005b]
- What about solutions in data exchange?

Blocks algorithm: idea

Key idea

Blocks are mutually independent partitions of $var(J)$.

Gaifman Graph $\mathcal{G}_J$ of instnance J

Undirected graph (V, E) where V represents $var(J)$ and $(v_1, v_2) \in E$ whenever there is $R(\bar{v}) \in J$ such that $v_1, v_2 \in \bar{v}$.
Blocks correspond to connected components of $\mathcal{G}_J$.

Example

$R(x, y), R(y, z), R(v, w)$
$R(1, 2), R(2, 3), R(4, 5)$

Blocks algorithm: idea (2)

Each homomorphism $h \colon J \rightarrow K$ can be represented as a union of $h_{B_i} \colon J[B_i] \rightarrow K$ for blocks B_i of J.

Recall how the canonical universal solution is created during the chase of the source instance I:

- For each st tgd $\phi(\bar{x}) \rightarrow (\exists \bar{y})\psi(\bar{x}, \bar{y})$
  For each $\bar{a}$, such that $I \models \phi(\bar{a})$, $\psi(\bar{a}, \bar{y})$ is instantiated by replacing the elements of $\bar{y}$ with fresh labeled nulls.

Question

If $\Sigma_t = \emptyset$ and J was created by chasing $\Sigma = \Sigma_{st}$. What can be said about the block size of J?

Blocks algorithm: no target constraints

Input: Source instance I, mapping Σ_{st}
Output: A core of a universal solution for I under Σ_{st}

(1) Chase I with $\Sigma_{st} \Rightarrow$ Canonical universal solution J.

(2) Compute the blocks B_i of J, and initialize J' to be J

(3) Check if $h_i \colon J'[B_i] \rightarrow J'$ exists, s.t. $h(x) = h(y)$ for some $x \in B_i$ and $y \neq x$.

(4) Set $J' = h(J')$, where h extends h_i to $dom(I)$ as identity mapping

(5) Return to step (3).

Blocks algorithm: target egds

A nice property allows to lift the blocks algorithm to target egds.

Rigidity Lemma [Fagin et al., 2005b]

Let $\tilde{J}$ be the canonical preuniversal instance for some source I and mapping $\Sigma_{st} \cup \Sigma_t$ where Σ_t consists of egds. Moreover, let x and y be nulls from different blocks of $\tilde{J}$. If, in the course of the chase of $\tilde{J}$ with Σ_t, an equality $x = y$ is enforced, the term $[x](= [y])$ standing for both x and y in the canonical universal solution J, is rigid: any endomorphism of J maps $[x]$ on itself.

Example

$J = \{R(1, x), R(y, 2), R(1, 3), R(3, 2)\}$
$\Sigma_t = \{R(1, x), R(y, 2) \rightarrow x = y\}$

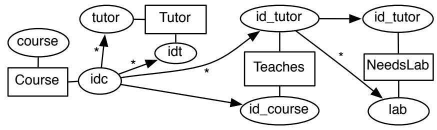
Effectively, target egds can be simply ignored.

Target tgds? Weak acyclicity

Dependency graph [Fagin, Kolaitis, Miller, and Popa, 2005a] of the mapping $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$

Directed graph $(V, E \cup E^*)$. V represents attributes of $\mathbf{T}$. $(a_1, a_2) \in E$ whenever a tgd copies a value from a_1 into a_2. Special edges: $(a_1, a_2) \in E^*$ whenever a_1 occurs in the antecedent of a tgd in which a_2 is occupied by an existentially quantified variable.
Dependency graphs of weakly-acyclic sets of tgds have no cycles through special edges.

1. $\text{Course}(Idc, C) \rightarrow \text{Tutor}(Idt, T), \text{Teaches}(Idt, Idc)$.
2. $\text{Teaches}(Idt, Idc) \rightarrow \text{NeedsLab}(Idt, L)$.



FindCore algorithm [Gottlob and Nash, 2008]: Idea

Idea

- Take a variable x and a term y, and test if any proper endomorphism can stitch them together.
- Testing for endomorphism existence should use some subset of the full instance which has bounded block size.

Parents, Ancestors, Siblings

- Parent variables: x_p is a parent of x, if the tgd that created x fired on the tuple $\bar{p}$, and $x_p \in \bar{p}$.
- Ancestor relation as a transitive closure of parent. Every null has bounded number of ancestors (by weak acyclicity).
- Siblings of x are nulls created by the same tgd, at the same chase step as x.

Example

Single st tgd $S(x_1, x_2) \rightarrow \exists Y_1 \exists Y_2\, R(x_1, x_2, Y_1, Y_2)$ and two target tgds:

τ_1: $R(x_1, x_2, y_1, y_2) \wedge R(x_2, x_3, y_3, y_4) \rightarrow R(x_1, x_3, y_1, y_4)$

τ_2: $R(x, x, y_1, y_2) \rightarrow \exists Z\, Q(y_1, y_2, Z)$

$I = \{S(1, 2), S(2, 3), S(3, 1)\}$

$\tilde{J} = \{R(1, 2, y_1, y_2), R(2, 3, y_3, y_4), R(3, 1, y_5, y_6)\}$

$J' = chase(\tilde{J}, \{\tau_1\}) = \tilde{J} \cup \{R(2, 1, y_3, y_6), R(1, 3, y_1, y_4), R(1, 1, y_1, y_6)\}$

$chase(J', \{\tau_2\}) = J' \cup \{Q(y_1, y_6, z_1)\}$

Note: y_3 and y_4 were needed to derive z_1, but they don't belong to its ancestors.

FindCore algorithm [Gottlob and Nash, 2008]

Input: Source instance I, st tgds Σ_{st}, weakly-acyclic set of target tgds Σ_t
Output: A core of a universal solution for I under $\Sigma_{st} \cup \Sigma_t$

(1) Let $\tilde{J}$ denote the canonical pre-universal instance, and J be the canonical universal solution obtained by chasing $\tilde{J}$ with Σ_t.
(2) Set $J^* = J$.
(3) Let T_{xy} be $\tilde{J}$ (fixed block size) together with an instance induced by the ancestors of x, y and their siblings (fixed number of variables). Test if a homomorphism $h_0\colon T_{xy} \rightarrow J^*$ exists, such that $h_0(x) = h_0(y)$
(4) By "replaying" the chase, h_0 can always be extended to $h\colon J \rightarrow J^*$.
(5) Transform h to a retraction r, so that $r(J)$ is a solution. Set $J^* = r(J)$.
(6) Repeat until no further variables can be eliminated.
(7) Return J^*.

Target egds by simulation

"Equality predicate" E

- For each egd $\phi(\bar{x}) \rightarrow x_i = x_j$, consider $\phi(\bar{x}) \rightarrow E(x_i, x_j)$
- $E(x, y) \rightarrow E(y, x)$, $\quad E(x, y) \wedge E(y, z) \rightarrow E(x, z)$
- For each target relation R, and each position i in R:
  - $R(\ldots, x_i, \ldots) \rightarrow E(x_i, x_i)$
  - $R(x_1, \ldots x_i, \ldots x_n) \wedge E(x_i, y) \rightarrow R(x_1, \ldots, y, \ldots x_n)$
- "Nice" (non-predefined) chase order required.

Example

Preuniversal instance $\tilde{J} = \{R(x, y), P(y, x)\}$, $\Sigma_t = \{R(z, v), P(v, z) \rightarrow z = v\}$

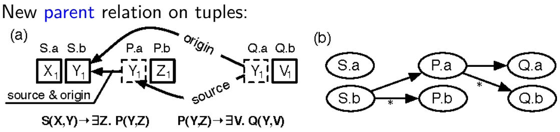
Simulating set $\bar{\Sigma}_t$ of 11 full tgds. $chase(\tilde{J}, \bar{\Sigma}_t) = \{R(x, y), R(x, x), R(y, x), R(y, y), P(y, x), P(y, y), P(x, y), P(x, x), E(x, x), E(x, y), E(y, x), E(y, y)\}$

Core: $\{R(x, x), P(x, x)\}$ resp. $\{R(y, y), P(y, y)\}$.

Support egds directly

- Egds unify variables and merge "families" of nulls.
- Switch to facts instead of variables [Pichler and S., 2010]. Redefine the parent relation.
- Need to be careful to keep the size of the fact "family" fixed in presence of non-special cycles in dependency graph.

New parent relation on tuples:



Parametrized Complexity

Block size is the key complexity parameter of core computation.

Theorem (Gottlob and Nash [2008])

The following search problems are fixed parameter intractable with respect to parameters blocksize(J) and k, respectively:

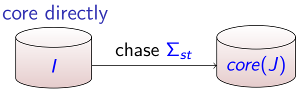
P1: CoreIdentification*: Given an instance J, compute core(J).*

P2: *Given a mapping $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{st} \cup \Sigma_t)$ where $\Sigma_t = \emptyset$ and where the maximum number of variables occurring in a tgd of Σ_{st} is bounded by parameter k, and a source instance I, compute the core of a universal solution for S.*

Laconic schema mappings

Why create redundant tuples in the first place?

Compute the core directly



For settings without target constraints, direct core computation has been proposed [Mecca, Papotti, and Raunich, 2009; ten Cate, Chiticariu, Kolaitis, and Tan, 2009].

Definition

Schema mapping is laconic, if chasing it (naively) produces a core.

Naive chase: fire each st tgd for each distinct tuple satisfying its antecedent.

Example (frightening) [Fagin et al., 2005b]

Consider two st tgds, and a source instance $I = \{R(1,1,2,3)\}$:

$R(a,b,c,d) \rightarrow (\exists x_1, x_2, x_3, x_4, x_5)$
$S(x_5, b, x_1, x_2, a)$
$\wedge S(x_5, c, x_3, x_4, a)$
$\wedge S(d, c, x_3, x_4, b)$

$S(N_5, 1, N_1, N_2, 1)$
$S(N_5, 2, N_3, N_4, 1)$
$S(3, 2, N_3, N_4, 2)$

$R(a,b,c,d) \rightarrow (\exists x_1, x_2, x_3, x_4, x_5)$
$S(d, a, a, x_1, b)$
$\wedge S(x_5, a, a, x_1, a)$
$\wedge S(x_5, c, x_2, x_3, x_4)$

$S(3, 1, 1, N'_1, 1)$
$S(N'_5, 1, 1, N'_1, 1)$
$S(N'_5, 2, N'_2, N'_3, N'_4)$

If fired together, st tgds above generate non-core atoms on I.
However, if fired alone, none of the tgds produce redundant atoms.

Idea of reformulation as a laconic mapping

$R(a,a,c,d) \rightarrow (\exists x_1, x_2)\, S(d, c, x_1, x_2, b)$

$R(a,a,c,d) \rightarrow (\exists y_1)\, S(d, a, a, y_1, b)$

$R(a,b,c,d) \wedge a \neq b \wedge b \neq c \rightarrow (\exists x_1, x_2, x_3, x_4, x_5)\, S(x_5, b, x_1, x_2, a) \wedge S(x_5, c, x_3, x_4, a) \wedge S(d, c, x_3, x_4, b)$

$R(a,b,b,d) \wedge a \neq b \rightarrow (\exists x_1, x_2, x_3)\, S(x_3, b, x_1, x_2, a) \wedge S(d, c, x_1, x_2, b)$

$R(a,b,c,d) \wedge a \neq b \rightarrow (\exists x_1, x_2, x_3, x_4, x_5)\, S(x_5, b, x_1, x_2, a) \wedge S(x_5, c, x_3, x_4, a) \wedge S(d, c, x_3, x_4, b)$

More examples [ten Cate et al., 2009]

No self-joins in the conclusion of tgds

- $S_1(x,y) \rightarrow (\exists z)\, P(x,z) \wedge Q(z,y)$
- $S_2(x,v) \rightarrow P(x,v)$
- $S_3(v,y) \rightarrow Q(v,y)$
- Laconic variant of the first tgd:
  $S_1(x,y) \wedge \neg S_2(x,v) \wedge \neg S_3(v,y) \rightarrow (\exists z)\, P(x,z) \wedge Q(z,y)$

Tgds with self-joins in the conclusion

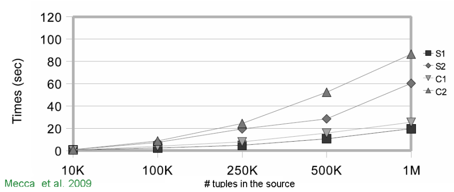
- $R(x,y) \rightarrow (\exists z)\, S(x,z) \wedge S(y,z)$
- Laconic variant:
  $(R(x,y) \vee R(y,x)) \wedge x \leq y \rightarrow (\exists z)\, S(x,z) \wedge S(y,z)$

Laconic mappings [ten Cate et al., 2009]

- Both negation and order on the source domain are necessary.
- Rewritten mappings can be exponential in the number of dependencies of the original, non-laconic mapping.

Skolemized form, suitable for SQL implementation

$S(x_1, x_2, x_3) \rightarrow \exists y\, R(x_1, y)$ $\quad S(x_1, x_2, x_3) \rightarrow R(x_1, f(x_1, x_2, x_3))$
$S(1,3,4) \Rightarrow R(1, \text{'f(1,3,4)'})$



Embracing target constraints

- No complete solution, unless target constraints can be fully "captured" by the st tgds. (E.g.: **bounded chase property**.)
- Best-effort approaches are available and can be helpful in practice.

Target functional dependencies [Marnette, Mecca, and Papotti, 2010]

- A FO implementation Σ_{st}^{FO} of the mapping $\mathcal{M} = \{\mathbf{S}, \mathbf{T}, \Sigma_{st} \cup \Sigma_t\}$ where Σ_t consists of FDs, is a set of st tgds (having UCQs with negation in the antecedents).
  - If $chase(I, \Sigma_{st}^{FO}) \models \Sigma_t$, then Σ_{st}^{FO} *succeeds* on I, and *fails* otherwise.
- Soundness: If Σ_{st}^{FO} succeeds on I, then $chase(I, \Sigma_{st}^{FO})$ is a universal solution. E.g., Σ_{st}^{FO} does not "invent" target artefacts.
- Completeness: Σ_{st}^{FO} succeeds on I iff $\mathcal{M}$ has solutions on I.

Direct Core Computation with target FDs

Theorem (Marnette et al. [2010])

There is a scenario $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{st} \cup \Sigma_t)$ *where* Σ_t *is a set of FDs over* $\mathbf{T}$ *such that no complete FO-implementation exists for* $\mathcal{M}$.

Proof sketch.

S: relation $E(x,y)$ encodes the edges (x,y) of a directed graph.
T: relation $R(v,m)$ marks each vertex v with a conntected component identifier m.

$\Sigma = \{E(x,y) \rightarrow \exists Z\, R(x,Z) \wedge R(y,Z)$
$R(x,z_1) \wedge R(x,z_2) \rightarrow z_1 = z_2\}$

- CQ $q_t(x,y) = \exists Z\, R(x,Z) \wedge R(y,Z)$ finds connected vertices.
- Complete FO-implementation possible $\Rightarrow$ a perfect FO rewriting of q_t over **S** must be obtainable using known techniques.
  Contradiction: reachability is not FO expressible.

□

Example #1: Sound implementation Σ_{st}^{FO}

Original mapping

Student(*name*, *bday*) → *Person*(*name*, *bday*, Y_1, Y_2)
Employee(*name*, *salary*) → *Person*(*name*, Y_1, *salary*Y_2)
Driver(*name*, *plate*) → *Person*(n, Y_1, Y_2, Z) ∧ *Car*(Z, *plate*)
Target FDs:
PK(*Person*): *name*, *Car*.⟨*id*⟩ → *plate*, *Car*.⟨*plate*⟩ → *id*

- *Student*(n, *bd*) ∧ *Employee*(n, s) → *Person*(n, *bd*, s, $f(n)$)
- *Student*(n, *bd*) ∧ *Driver*(n, p) → *Person*(n, *bd*, s, $f(n)$)
- *Employee*(n, s) ∧ *Driver*(n, p) → *Person*(n, $g(n)$, s, $f(n)$) ∧ *Car*($f(n)$, *plate*)
- *Student*(n, *bd*) ∧ *Employee*(n, s) ∧ *Driver*(n, p) → *Person*(n, *bd*, $h(n)$, $f(n)$) ∧ *Car*($f(n)$, *plate*)
- ... orignal st tgds enhanced with negated CQs in the antecedents.

Example #2: No complete implementation

Recall the graph connectedness example:

$\Sigma = \{E(x,y) \rightarrow \exists Z\, R(x,Z) \wedge R(y,Z)$
$R(x,z_1) \wedge R(x,z_2) \rightarrow z_1 = z_2\}$

Sound implementations

$\Sigma_{st}^1 = \{E(x,y) \wedge E(y,v) \rightarrow \exists Z\, R(x,Z) \wedge R(y,Z) \wedge R(y,Z)\}$

$\Sigma_{st}^2 = \Sigma_{st}^1 \cup \{E(x,y) \wedge E(y,v) \wedge E(v,w) \rightarrow \exists Z\, R(x,Z) \wedge R(y,Z) \wedge R(y,Z) \wedge R(w,Z)\}$

$\Sigma_{st}^3 = \Sigma_{st}^2 \cup \ldots$

For each n, easy to construct a case when Σ_{st}^n fails (leads to violation of a FD) though Σ has solutions.

Direct core computation in presence of target FDs

Theorem (Marnette et al. [2010])

Given a sound FO implementation Σ_{st}^{FO} of $\mathcal{M}$, it is decidable to check its completeness: Test if chase with Σ_{st}^{FO} can produce an instance violating some target FD in $\mathcal{M}$.

Direct core computation:

1. Work target FDs in st tgds (by combining conclusions of st tgds and chasing them with FDs) to produce a sound FO implementation (best effort).
2. Test FO implementation for completeness.
3. If complete, make the FO implementation laconic, by adapting the rewriting ideas shown before (technical).

R. Fagin, P. G. Kolaitis, R. J. Miller, and L. Popa. Data exchange: semantics and query answering. *Theoretical Computer Science*, 336(1):89 – 124, 2005a.

R. Fagin, P. G. Kolaitis, and L. Popa. Data exchange: getting to the core. *ACM Trans. Database Syst.*, 30(1):174–210, 2005b.

G. Gottlob. Computing cores for data exchange: new algorithms and practical solutions. In *PODS*, pages 148–159, 2005.

G. Gottlob and A. Nash. Data exchange: computing cores in polynomial time. In *PODS*, pages 40–49, 2006.

G. Gottlob and A. Nash. Efficient core computation in data exchange. *J. ACM*, 55(2):1–49, 2008.

P. Hell and J. Nesetril. The core of a graph. *Discrete Mathematics*, 109(1-3): 117 – 126, 1992.

B. Marnette. Generalized schema-mappings: from termination to tractability. In *PODS*, pages 13–22, 2009.

B. Marnette, G. Mecca, and P. Papotti. Scalable data exchange with functional dependencies. *PVLDB*, 3(1):105–116, 2010.

G. Mecca, P. Papotti, and S. Raunich. Core schema mappings. In *SIGMOD Conference*, pages 655–668, 2009.

B. ten Cate, L. Chiticariu, P. G. Kolaitis, and W. C. Tan. Laconic schema mappings: Computing the core with sql queries. *PVLDB*, 2(1):1006–1017, 2009.

Summary

- Core is in many cases the best universal solution to materialize in the target database.
- For core computation, the crucial complexity parameter is the block size of the instance. W.r.t. the block size, CoreIdentification is fixed-parameter intractable.
- Core computation is tractable for target egds and weakly-acyclic sets of target tgds.
- In absence of target constraints, core can be computed directly by chasing rewritten mappings. Rewritten mappings require more expressive language (negation, linear order) and can be exponential in size.
- Direct core computation in presence of target constraints is possible on the best effort basis.