Enhancement and Implementation of Core Computation

Reinhard Pichler and Vadim Savenkov

Technische Universität Wien
[pichler | savenkov]@dbai.tuwien.ac.at

INFINT 2007

Outline

Outline

Motivation

Starting Point

1. Arguments in favor of the core (Fagin et al., 2003)
2. Tractability of core computation (Gottlob/Nash, 2006)
3. No implementation of core computation

Motivation

Starting Point

1. Arguments in favor of the core (Fagin et al., 2003)
2. Tractability of core computation (Gottlob/Nash, 2006)
3. No implementation of core computation

Goal

1. Prototype Implementation
2. Enhancement:
   - No simulation of target EGDs by TGDs
   - Strict separation of core computation from solving the data exchange problem

Outline

Preliminaries

Basic Definitions (1)

- Embedded dependencies $\forall \vec{x}\,(\phi(\vec{x}\,) \rightarrow \exists \vec{y}\; \psi(\vec{x}, \vec{y}\,))$
  - TGDs: $\psi(\vec{x}, \vec{y}\,)$ is a conjunction of atoms
  - EGDs: $\psi(\vec{x}, \vec{y}\,)$ is a conjunction of equalities
- Data exchange setting $(\mathbf{S}, \mathbf{T}, \Sigma_{st}, \Sigma_t)$:
  - source schema **S**, target schema **T**, STDs Σ_{st}, TDs Σ_t
  - Σ_{st} is a set of TGDs
  - Σ_t is a set of EGDs and *weakly acyclic* TGDs
- Data exchange problem for $(\mathbf{S}, \mathbf{T}, \Sigma_{st}, \Sigma_t)$
  Given source instance S, construct a target instance U, s.t. all of the STDs Σ_{st} and TDs Σ_t are satisfied.

Basic Definitions (2)

- **Solving the data exchange problem via chase.**
  - Preuniversal instance $T = (S, \emptyset)^{\Sigma_{st}}$
  - (Canonical) universal instance $U = T^{\Sigma_t}$
- **Homomorphisms.**
  - endomorphism: homomorphism $h\colon I \to I$
  - retraction: idempotent endomorphism $h\colon I \to I$
  - proper endomorphism/retraction. h non-surjective
- **Core.**
  - Core: instance with no proper retraction
  - Core of instance I: retract of I which is a core
  - Core is unique up to isomorphism
  - Core of data exchange problem: core of a universal solution

Outline

FindCore Algorithm of (Gottlob/Nash, 2006)

Input: Source ground instance S
Output: Core of a universal solution for S

Chase (S,$\emptyset$) with Σ_{st} to obtain $(S, T) := (S, \emptyset)^{\Sigma_{st}}$;
Compute $\bar{\Sigma}_t$ from Σ_t;
Chase T with $\bar{\Sigma}_t$ (using a *nice order*) to get $U := T^{\bar{\Sigma}_t}$;
for each $x \in \mathrm{var}(U)$, $y \in \mathrm{dom}(U)$, $x \neq y$ **do**
  Compute T_{xy};
  Look for $h \colon T_{xy} \to U$ s.t. $h(x) = h(y)$;
  if there is such h **then**
    Extend h to an endomorphism h' on U;
    Transform h' into a retraction r;
    Set $U := r(U)$;
  fi;
od;
return U.

FindCore Algorithm of (Gottlob/Nash, 2006)

Input: Source ground instance S
Output: Core of a universal solution for S

Chase (S,$\emptyset$) with Σ_{st} to obtain $(S, T) := (S, \emptyset)^{\Sigma_{st}}$;
Compute $\bar{\Sigma}_t$ from Σ_t;
Chase T with $\bar{\Sigma}_t$ (using a *nice order*) to get $U := T^{\bar{\Sigma}_t}$;
for each $x \in \mathrm{var}(U)$, $y \in \mathrm{dom}(U)$, $x \neq y$ **do**
    Compute T_{xy};
    Look for $h\colon T_{xy} \to U$ s.t. $h(x) = h(y)$;
    if there is such h **then**
        Extend h to an endomorphism h' on U;
        Transform h' into a retraction r;
        Set $U := r(U)$;
    fi;
od;
return U.

FindCore Algorithm of (Gottlob/Nash, 2006)

Simulation of EGDs by TGDs

- Transformation of Σ_t into $\bar{\Sigma}_t$
  - Replace all equations $x = y$ with $E(x, y)$.
  - Add the following *equality* constraints:
    – $E(x, y) \rightarrow E(y, x)$
    – $E(x, y), E(y, z) \rightarrow E(x, z)$
    – $R(x_1, \ldots, x_k) \rightarrow E(x_i, x_i)$
  - Add the following *consistency* constraints:
    – $R(x_1, \ldots, x_k), E(x_i, y) \rightarrow R(x_1, \ldots, y, \ldots, x_k)$
- Chase with $\bar{\Sigma}_t$
  - $\bar{\Sigma}_t$ is, in general, not weakly acyclic.
  - A *nice chase order* guarantees termination.
  - $U := T^{\bar{\Sigma}_t}$ is not a solution.
  - The core of U is a solution.

FindCore Algorithm of (Gottlob/Nash, 2006)

Input: Source ground instance S
Output: Core of a universal solution for S

Chase (S,$\emptyset$) with Σ_{st} to obtain $(S, T) := (S, \emptyset)^{\Sigma_{st}}$;
Compute $\bar{\Sigma}_t$ from Σ_t;
Chase T with $\bar{\Sigma}_t$ (using a *nice order*) to get $U := T^{\bar{\Sigma}_t}$;
for each $x \in \mathrm{var}(U)$, $y \in \mathrm{dom}(U)$, $x \neq y$ **do**
    Compute T_{xy};
    Look for $h\colon T_{xy} \to U$ s.t. $h(x) = h(y)$;
    if there is such h **then**
        Extend h to an endomorphism h' on U;
        Transform h' into a retraction r;
        Set $U := r(U)$;
    fi;
od;
return U.

FindCore Algorithm of (Gottlob/Nash, 2006)

Input: Source ground instance S
Output: Core of a universal solution for S

Chase (S,$\emptyset$) with Σ_{st} to obtain $(S, T) := (S, \emptyset)^{\Sigma_{st}}$;
Compute $\bar{\Sigma}_t$ from Σ_t;
Chase T with $\bar{\Sigma}_t$ (using a *nice order*) to get $U := T^{\bar{\Sigma}_t}$;
for each $x \in \mathrm{var}(U)$, $y \in \mathrm{dom}(U)$, $x \neq y$ **do**
    Compute T_{xy};
    Look for $h\colon T_{xy} \rightarrow U$ s.t. $h(x) = h(y)$;
    if there is such h **then**
        Extend h to an endomorphism h' on U;
        Transform h' into a retraction r;
        Set $U := r(U)$;
    fi;
od;
return U.

FindCore Algorithm of (Gottlob/Nash, 2006)

Search for a proper endomorphism $h' : U \rightarrow U$

- Observation.
  - Search for homomorphism is exponential w.r.t. block size.
  - Block size in T is bounded by a constant; but not in U.

FindCore Algorithm of (Gottlob/Nash, 2006)

Search for a proper endomorphism $h': U \to U$

- **Observation.**
  - Search for homomorphism is exponential w.r.t. block size.
  - Block size in T is bounded by a constant; but not in U.
- **Idea: Split search for h' into 2 steps.**
  - Search for a homomorphism $h\colon T_{xy} \to U$ with $h(x) = h(y)$.
  - Then h is extended to endomorphism $h': U \to U$.

FindCore Algorithm of (Gottlob/Nash, 2006)

Search for a proper endomorphism $h' : U \rightarrow U$

- **Observation.**
  - Search for homomorphism is exponential w.r.t. block size.
  - Block size in T is bounded by a constant; but not in U.
- **Idea: Split search for h' into 2 steps.**
  - Search for a homomorphism $h \colon T_{xy} \rightarrow U$ with $h(x) = h(y)$.
  - Then h is extended to endomorphism $h' : U \rightarrow U$.
- **Construction of T_{xy}.**
  - Define parent and sibling relation on variables in $T^{\bar{\Sigma}_t}$.
  - Construct T_{xy}, s.t. $T \subseteq T_{xy} \subseteq T^{\bar{\Sigma}_t}$ and dom(T_{xy}) is closed under parents and siblings.
  - The block size of T_{xy} is bounded by a constant.

FindCore Algorithm of (Gottlob/Nash, 2006)

Input: Source ground instance S
Output: Core of a universal solution for S

Chase (S,$\emptyset$) with Σ_{st} to obtain $(S, T) := (S, \emptyset)^{\Sigma_{st}}$;
Compute $\bar{\Sigma}_t$ from Σ_t;
Chase T with $\bar{\Sigma}_t$ (using a *nice order*) to get $U := T^{\bar{\Sigma}_t}$;
for each $x \in \mathrm{var}(U)$, $y \in \mathrm{dom}(U)$, $x \neq y$ **do**
  Compute T_{xy};
  Look for $h\colon T_{xy} \to U$ s.t. $h(x) = h(y)$;
  if there is such h **then**
    Extend h to an endomorphism h' on U;
    Transform h' into a retraction r;
    Set $U := r(U)$;
  fi;
od;
return U.

FindCore Algorithm of (Gottlob/Nash, 2006)

Input: Source ground instance S
Output: Core of a universal solution for S

Chase (S,$\emptyset$) with Σ_{st} to obtain $(S, T) := (S, \emptyset)^{\Sigma_{st}}$;
Compute $\bar{\Sigma}_t$ from Σ_t;
Chase T with $\bar{\Sigma}_t$ (using a *nice order*) to get $U := T^{\bar{\Sigma}_t}$;
for each $x \in \mathrm{var}(U)$, $y \in \mathrm{dom}(U)$, $x \neq y$ **do**
  Compute T_{xy};
  Look for $h\colon T_{xy} \to U$ s.t. $h(x) = h(y)$;
  if there is such h **then**
    Extend h to an endomorphism h' on U;
    Transform h' into a retraction r;
    Set $U := r(U)$;
  fi;
od;
return U.

FindCore Algorithm of (Gottlob/Nash, 2006)

Retractions

- Property 1.
  Let $r \colon A \to A$ *be a retraction with* $B = r(A)$ *and let* Σ *be a set of embedded dependencies. If* $A \models \Sigma$*, then* $B \models \Sigma$*.*
- Property 2.
  Let $h \colon A \to A$ *be an endomorphism s.t.* $h(x) = h(y)$ *for some* $x, y \in \mathrm{dom}(A)$
  - *Then there is a proper retraction* r *on* A *s.t.* $r(x) = r(y)$*.*
  - *Such a retraction can be found in time* $O(|\mathrm{dom}(A)|^2)$*.*

Outline

FindCore Algorithm of (Gottlob/Nash, 2006)

Input: Source ground instance S
Output: Core of a universal solution for S

Chase (S,$\emptyset$) with Σ_{st} to obtain $(S, T) := (S, \emptyset)^{\Sigma_{st}}$;
Compute $\bar{\Sigma}_t$ from Σ_t;
Chase T with $\bar{\Sigma}_t$ (using a *nice order*) to get $U := T^{\bar{\Sigma}_t}$;
for each $x \in \mathrm{var}(U)$, $y \in \mathrm{dom}(U)$, $x \neq y$ **do**
  Compute T_{xy};
  Look for $h\colon T_{xy} \rightarrow U$ s.t. $h(x) = h(y)$;
  if there is such h **then**
    Extend h to an endomorphism h' on U;
    Transform h' into a retraction r;
    Set $U := r(U)$;
  fi;
od;
return U.

Enhanced Algorithm FINDCORE^E

Input: Source ground instance S
Output: Core of a universal solution for S

```
(1)   Chase (S,∅) with Σ_st to obtain (S, T) := (S, ∅)^{Σ_st};
(2)   Chase T with Σ_t to obtain U := T^{Σ_t};
(3)   for each x ∈ var(U), y ∈ dom(U), x ≠ y do
(4)      Compute T_xy;
(5)      Look for h: T_xy → U s.t. h(x) = h(y);
(6       if there is such h then
(7)         Extend h to an endomorphism h' on U;
(8)         Transform h' into a retraction r;
(9)         Set U := r(U);
(10)     fi;
(11)  od;
(12)  return U.
```

Enhanced Algorithm FINDCORE^E

Input: Source ground instance S
Output: Core of a universal solution for S

(1) Chase (S,$\emptyset$) with Σ_{st} to obtain $(S, T) := (S, \emptyset)^{\Sigma_{st}}$;
(2) Chase T with Σ_t to obtain $U := T^{\Sigma_t}$;
(3) **for each** $x \in \text{var}(U)$, $y \in \text{dom}(U)$, $x \neq y$ **do**
(4) Compute T_{xy};
(5) Look for $h\colon T_{xy} \to U$ s.t. $h(x) = h(y)$;
(6 **if** there is such h **then**
(7) Extend h to an endomorphism h' on U;
(8) Transform h' into a retraction r;
(9) Set $U := r(U)$;
(10) **fi**;
(11) **od**;
(12) **return** U.

Modifications required in $\textsc{FindCore}^E$

Shrinking the canonical universal instance to the core

- compute an instance T_{xy}
- search for a non-injective homomorphism $h\colon T_{xy} \to U$
- lift h to a proper endomorphism $h'\colon U \to U$
- construct a proper retraction r from h' and compute $r(U)$

Modifications required in FINDCORE^E

Shrinking the canonical universal instance to the core

- compute an instance T_{xy}
  - without EGDs, we have $T \subseteq T_{xy} \subseteq T^{\bar{\Sigma}_t}$
  - with EGDs, we do not even have $T \subseteq T^{\Sigma_t}$
- search for a non-injective homomorphism $h\colon T_{xy} \to U$
- lift h to a proper endomorphism $h'\colon U \to U$
- construct a proper retraction r from h' and compute $r(U)$

Modifications required in FINDCORE^E

Shrinking the canonical universal instance to the core

- compute an instance T_{xy}
- search for a non-injective homomorphism $h\colon T_{xy} \to U$
  - positive effect of EGDs: variables may be eliminated
  - negative effect of EGDs: blocks of T may be merged
- lift h to a proper endomorphism $h'\colon U \to U$
- construct a proper retraction r from h' and compute $r(U)$

Modifications required in FINDCORE^E

Shrinking the canonical universal instance to the core

- compute an instance T_{xy}
- search for a non-injective homomorphism $h\colon T_{xy} \to U$
- lift h to a proper endomorphism $h'\colon U \to U$
  - modification required since T_{xy} is defined differently
  - proof has to be completely rewritten
- construct a proper retraction r from h' and compute $r(U)$

Modifications required in $\textsc{FindCore}^E$

Shrinking the canonical universal instance to the core

- compute an instance T_{xy}
- search for a non-injective homomorphism $h\colon T_{xy} \to U$
- lift h to a proper endomorphism $h'\colon U \to U$
- construct a proper retraction r from h' and compute $r(U)$
    - no changes required

Modifications required in FINDCORE^E

Shrinking the canonical universal instance to the core

- compute an instance T_{xy}
- search for a non-injective homomorphism $h\colon T_{xy} \to U$
- lift h to a proper endomorphism $h'\colon U \to U$
- construct a proper retraction r from h'
  - no changes required

Concentrate on Facts (rather than Variables)

Introduction of an id.

- Every fact $R(x_1, x_2, \ldots, x_n)$ is equipped with a unique id: $R(id, x_1, x_2, \ldots, x_n)$
- Identify facts with their id, i.e.: $R(id_1, x_1, x_2, \ldots, x_n) = R(id_2, y_1, y_2, \ldots, y_n)$ iff $id_1 = id_2$.
- Variables disappear, facts (and positions in facts) persist.

Concentrate on Facts (rather than Variables)

Introduction of an id.

- Every fact $R(x_1, x_2, \ldots, x_n)$ is equipped with a unique id: $R(id, x_1, x_2, \ldots, x_n)$
- Identify facts with their id, i.e.: $R(id_1, x_1, x_2, \ldots, x_n) = R(id_2, y_1, y_2, \ldots, y_n)$ iff $id_1 = id_2$.
- Variables disappear, facts (and positions in facts) persist.

Definition of T_{xy}

- T_{xy} contains facts where x, y were introduced by TGDs.
- All facts of T are in T_{xy}, and $T_{xy} \subseteq T^{\Sigma_t}$.
- T_{xy} is closed under *parents* and *siblings* over facts.

Search for a non-injective homomorphism $h\colon T_{xy} \to U$

Definition

- Effect of EGDs. Let $J' = J^{\Sigma_t}$
  - $[u]$ = term to which u is mapped by the chase
  - $u \sim v$ if $[u] = [v]$
- Rigidity. A domain element y is rigid in an instance K, if $h(y) = y$ for every endomorphism h on K.

Rigidity Lemma – analogously to (Fagin et al, 2003)

Let J be the preuniversal instance and $J' = J^{\Sigma_t}$ the canonical universal instance, and let x and y be nulls of J with $x \backsim y$.
If $[x]$ is non-rigid in J', then x and y are in the same block of J.

FINDCORE VS. FINDCORE^{E}

What the algorithms have in common

- identical overall structure (apart from the target chase)
- asymptotic worst-case complexity

FINDCORE VS. FINDCORE^E

What the algorithms have in common

- identical overall structure (apart from the target chase)
- asymptotic worst-case complexity

Theorem

Let $(\mathbf{S}, \mathbf{T}, \Sigma_{st}, \Sigma_t)$ be a data exchange setting and let S be a ground instance of the source schema $\mathbf{S}$.
If this data exchange problem has a solution, then both FINDCORE *and* FINDCORE^E *correctly compute the core of a canonical universal solution in time $O(|\mathrm{dom}(S)|^b)$ for some b that depends only on $\Sigma_{st} \cup \Sigma_t$.*

FINDCORE vs. FINDCORE^E

Major differences between the algorithms

- Canonical solution vs. core. In FINDCORE^E, the chase first produces a solution of the data exchange problem, while the core computation is considered as an optional add-on.

FINDCORE VS. FINDCORE^E

Major differences between the algorithms

- Canonical solution vs. core. In FINDCORE^E, the chase first produces a solution of the data exchange problem, while the core computation is considered as an optional add-on.
- Chase order. FINDCORE^E selects the TDs with don't care nondeterminism. Hence, several instantiations of a single TGD can be enforced simultaneously.

FINDCORE vs. FINDCORE^E

Major differences between the algorithms

- Canonical solution vs. core. In FINDCORE^E, the chase first produces a solution of the data exchange problem, while the core computation is considered as an optional add-on.
- Chase order. FINDCORE^E selects the TDs with don't care nondeterminism. Hence, several instantiations of a single TGD can be enforced simultaneously.
- Simulation of the EGDs by TGDs. This simulation in FINDCORE increases the set of TDs and the result of the chase (but, of course, this increase easily fits into the polynomial time upper bound).

FINDCORE VS. FINDCORE^E

Example

Let $J = \{R(x, y), P(y, x)\}$ and $\Sigma_t = \{R(z, v), P(v, z) \rightarrow z = v\}$.

FINDCORE VS. FINDCORE^E

Example

Let $J = \{R(x,y), P(y,x)\}$ and $\Sigma_t = \{R(z,v), P(v,z) \rightarrow z = v\}$.

$\bar{\Sigma}_t = \{R(z,v), P(v,z) \rightarrow E(z,v);\ E(x,y) \rightarrow E(y,x);$
$E(x,y), E(y,z) \rightarrow E(x,z);\ R(x,y) \rightarrow E(x,x);$
$R(x,y) \rightarrow E(y,y);\ P(x,y) \rightarrow E(x,x);\ P(x,y) \rightarrow E(y,y);$
$R(x,y), E(x,z) \rightarrow R(z,y);\ R(x,y), E(y,z) \rightarrow R(x,z);$
$P(x,y), E(x,z) \rightarrow P(z,y);\ P(x,y), E(y,z) \rightarrow P(x,z)\}$

FINDCORE vs. FINDCORE^E

Example

Let $J = \{R(x,y), P(y,x)\}$ and $\Sigma_t = \{R(z,v), P(v,z) \rightarrow z = v\}$.

$\bar{\Sigma}_t = \{R(z,v), P(v,z) \rightarrow E(z,v); E(x,y) \rightarrow E(y,x);$
$E(x,y), E(y,z) \rightarrow E(x,z); R(x,y) \rightarrow E(x,x);$
$R(x,y) \rightarrow E(y,y); P(x,y) \rightarrow E(x,x); P(x,y) \rightarrow E(y,y);$
$R(x,y), E(x,z) \rightarrow R(z,y); R(x,y), E(y,z) \rightarrow R(x,z);$
$P(x,y), E(x,z) \rightarrow P(z,y); P(x,y), E(y,z) \rightarrow P(x,z)\}$

$J^{\bar{\Sigma}_t} = \{R(x,y), R(x,x), R(y,x), R(y,y), P(y,x), P(y,y),$
$P(x,y), P(x,x), E(x,x), E(x,y), E(y,x), E(y,y)\}$.

FINDCORE VS. FINDCORE^E

Example

Let $J = \{R(x,y), P(y,x)\}$ and $\Sigma_t = \{R(z,v), P(v,z) \rightarrow z = v\}$.

$\bar{\Sigma}_t = \{R(z,v), P(v,z) \rightarrow E(z,v);\ E(x,y) \rightarrow E(y,x);$
$E(x,y), E(y,z) \rightarrow E(x,z);\ R(x,y) \rightarrow E(x,x);$
$R(x,y) \rightarrow E(y,y);\ P(x,y) \rightarrow E(x,x);\ P(x,y) \rightarrow E(y,y);$
$R(x,y), E(x,z) \rightarrow R(z,y);\ R(x,y), E(y,z) \rightarrow R(x,z);$
$P(x,y), E(x,z) \rightarrow P(z,y);\ P(x,y), E(y,z) \rightarrow P(x,z)\}$

$J^{\bar{\Sigma}_t} = \{R(x,y), R(x,x), R(y,x), R(y,y), P(y,x), P(y,y),$
$P(x,y), P(x,x), E(x,x), E(x,y), E(y,x), E(y,y)\}$.

The core of $J^{\bar{\Sigma}_t}$ is $\{R(x,x), P(x,x)\}$.

FINDCORE VS. FINDCORE^E

Example

Let $J = \{R(x,y), P(y,x)\}$ and $\Sigma_t = \{R(z,v), P(v,z) \rightarrow z = v\}$.

$\bar{\Sigma}_t = \{R(z,v), P(v,z) \rightarrow E(z,v); E(x,y) \rightarrow E(y,x);$
$E(x,y), E(y,z) \rightarrow E(x,z); R(x,y) \rightarrow E(x,x);$
$R(x,y) \rightarrow E(y,y); P(x,y) \rightarrow E(x,x); P(x,y) \rightarrow E(y,y);$
$R(x,y), E(x,z) \rightarrow R(z,y); R(x,y), E(y,z) \rightarrow R(x,z);$
$P(x,y), E(x,z) \rightarrow P(z,y); P(x,y), E(y,z) \rightarrow P(x,z)\}$

$J^{\bar{\Sigma}_t} = \{R(x,y), R(x,x), R(y,x), R(y,y), P(y,x), P(y,y),$
$P(x,y), P(x,x), E(x,x), E(x,y), E(y,x), E(y,y)\}$.

The core of $J^{\bar{\Sigma}_t}$ is $\{R(x,x), P(x,x)\}$.

Chasing $J = \{R(x,y), P(y,x)\}$ directly with Σ_t yields the universal solution $J^{\Sigma} = \{R(x,x), P(x,x)\}$.

Outline

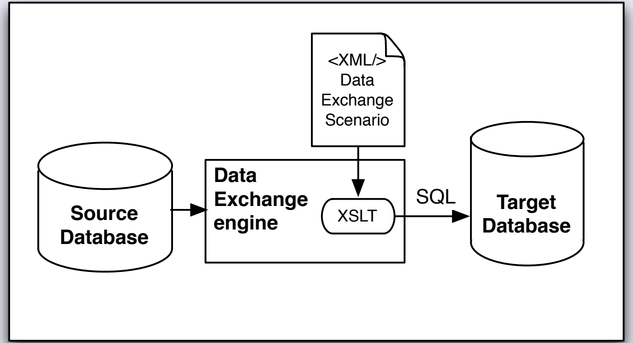
Prototype Implementation

Basic ideas

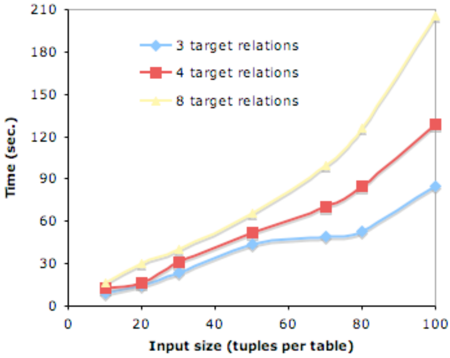
- Java. The core computation is implemented in Java with access to RDBMSs via JDBC.
- XML configuration file for data exchange scenario.
- Use of XSLT to generate the scenario-dependent code parts (in particular, the SQL-statements) from the XML file.
- DBMS back-end. Core computation on top of an RDBMS
  - Add tables (e.g., variable mappings of a homomorphism) and views (e.g. image of a homomorphism).
  - Chase and basic operations of the core computation (e.g., searching for a homomorphism) realized via SQL.

Prototype Implementation

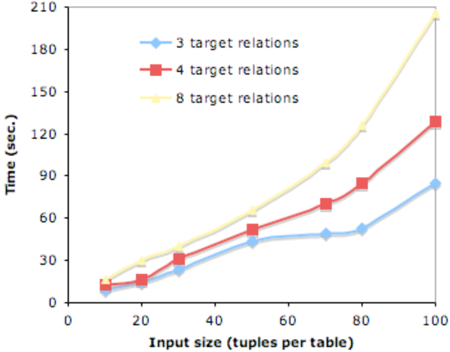
Overview



Experimental Results



Experimental Results



successful scenarios

- 10 relations
- 100 tuples per table
- 1000 variables
- dependencies of 2 to 6 atoms.

Outline

Conclusion

Main Results

- enhanced algorithm for core computation
- prototype implementation
- first experimental results

Conclusion

Main Results

- enhanced algorithm for core computation
- prototype implementation
- first experimental results

Future Work

- bottleneck analysis of implementation
- more efficient implementation
- approximation? subclasses?