Implementing Core Computation for Data Exchange

Diplomarbeitspräsentationen der Fakultät für Informatik

Technische Universität Wien
Institut für Informationssysteme
Arbeitsbereich: Database and Artificial Intelligence
Betreuer: Prof. Dr. Reinhard Pichler

Individuelles Diplomstudium: Computationale Logik

Vadim Savenkov

Homomorphism

— functional mapping h between two databases D and D’, s.t. when a tuple x ∈ D, h(x) ∈ D’
— constants are always mapped on themselves, and labeled nulls can be mapped on constants or on nulls

Endomorphism

— a homomorphism, that maps a database on itself
— endomorphism is proper if it maps a database on its part

Chase transformations to build the database

(Step a) FindCore algorithm by G. Gottlob and A. Nash (PDS 2006)

1. Rewrite the set of target constraints (simulate EGDs by TGDs)
2. Chase the rewritten constraints to build the database D satisfying them
4. Set Core ← D
   5. Run minimization loop until no nulls left or all pairs of terms have been tested minimization loop

a. Choose a labeled null x and a term y from dom(x)
b. Determine a characteristic part of D with Origin(x) and Origin(y), where homomorphism checking is tractable
   c. Check if exists a homomorphism that maps x and y
   d. If yes, transform it into a restriction R on the whole database and, set Core ← R(Core)

Step (b)

Step (c)

Step (d)

Step (e)

Step (f)

Step (g)

Our improvement of FindCore:

Eliminate the need for the first rewriting step

With rewriting

1) Substitute one with 7 new TGDs to a)
   mark equal terms,
   b) add new versions of existing tuples with all possible combinations of equal terms.
2) Compute the core — core satisfies

Core(Salary)

Without

Unified terms via chase

avoid useless blow-up

constraint satisfaction (cheap) and core computation (expensive) are decoupled

database is consistent and ready for querying right after the chase

Our improvement of FindCore: Eliminate the need for the first rewriting step

With rewriting

1) Substitute one with 7 new TGDs to
   a) mark equal terms,
   b) add new versions of existing tuples with all possible combinations of equal terms.
2) Compute the core — core satisfies

Core(Salary)

Without

Unified terms via chase

avoid useless blow-up

constraint satisfaction (cheap) and core computation (expensive) are decoupled

database is consistent and ready for querying right after the chase

Logical Meaning of the Salary table

this equality suggests that the database can be minimised

The most compact database — of which no proper endomorphism exists, is called the CORE

Homomorphism search boils down to SQL evaluation. Step (c) in SQL:

Select E.Project AS P2,
E.Dept AS Y2,
SP.Bonus_plan AS B1
From Salary S JOIN Employee E ON S.Dept = E.Dept
WHERE Label(Salary_plan) = ‘S1’

Labeled nulls and homomorphisms are simulated by views

Homomorphism

— Homomorphism search boils down to SQL evaluation. Step (c) in SQL:

Select E.Project AS P2,
E.Dept AS Y2,
SP.Bonus_plan AS B1
From Salary S JOIN Employee E ON S.Dept = E.Dept
WHERE Label(Salary_plan) = ‘S1’

Labeled nulls and homomorphisms are simulated by views

FindCore algorithm by G. Gottlob and A. Nash (PDS 2006)

1. Rewrite the set of target constraints (simulate EGDs by TGDs)
2. Chase the rewritten constraints to build the database D satisfying them
4. Set Core ← D
   5. Run minimization loop until no nulls left or all pairs of terms have been tested minimization loop

a. Choose a labeled null x and a term y from dom(x)
b. Determine a characteristic part of D with Origin(x) and Origin(y), where homomorphism checking is tractable
   c. Check if exists a homomorphism that maps x and y
   d. If yes, transform it into a restriction R on the whole database D, and set Core ← R(Core)

With rewriting

1) Substitute one with 7 new TGDs to
   a) mark equal terms,
   b) add new versions of existing tuples with all possible combinations of equal terms.
2) Compute the core — core satisfies

Core(Salary)

Without

Unified terms via chase

avoid useless blow-up

constraint satisfaction (cheap) and core computation (expensive) are decoupled

database is consistent and ready for querying right after the chase

Logical Meaning of the Salary table

this equality suggests that the database can be minimised

The most compact database — of which no proper endomorphism exists, is called the CORE

Homomorphism search boils down to SQL evaluation. Step (c) in SQL:

Select E.Project AS P2,
E.Dept AS Y2,
SP.Bonus_plan AS B1
From Salary S JOIN Employee E ON S.Dept = E.Dept
WHERE Label(Salary_plan) = ‘S1’

Labeled nulls and homomorphisms are simulated by views

FindCore algorithm by G. Gottlob and A. Nash (PDS 2006)

1. Rewrite the set of target constraints (simulate EGDs by TGDs)
2. Chase the rewritten constraints to build the database D satisfying them
4. Set Core ← D
   5. Run minimization loop until no nulls left or all pairs of terms have been tested minimization loop

a. Choose a labeled null x and a term y from dom(x)
b. Determine a characteristic part of D with Origin(x) and Origin(y), where homomorphism checking is tractable
   c. Check if exists a homomorphism that maps x and y
   d. If yes, transform it into a restriction R on the whole database D, and set Core ← R(Core)

With rewriting

1) Substitute one with 7 new TGDs to
   a) mark equal terms,
   b) add new versions of existing tuples with all possible combinations of equal terms.
2) Compute the core — core satisfies

Core(Salary)

Without

Unified terms via chase

avoid useless blow-up

constraint satisfaction (cheap) and core computation (expensive) are decoupled

database is consistent and ready for querying right after the chase

Logical Meaning of the Salary table

this equality suggests that the database can be minimised

The most compact database — of which no proper endomorphism exists, is called the CORE

Homomorphism search boils down to SQL evaluation. Step (c) in SQL:

Select E.Project AS P2,
E.Dept AS Y2,
SP.Bonus_plan AS B1
From Salary S JOIN Employee E ON S.Dept = E.Dept
WHERE Label(Salary_plan) = ‘S1’

Labeled nulls and homomorphisms are simulated by views

FindCore algorithm by G. Gottlob and A. Nash (PDS 2006)

1. Rewrite the set of target constraints (simulate EGDs by TGDs)
2. Chase the rewritten constraints to build the database D satisfying them
4. Set Core ← D
   5. Run minimization loop until no nulls left or all pairs of terms have been tested minimization loop

a. Choose a labeled null x and a term y from dom(x)
b. Determine a characteristic part of D with Origin(x) and Origin(y), where homomorphism checking is tractable
   c. Check if exists a homomorphism that maps x and y
   d. If yes, transform it into a restriction R on the whole database D, and set Core ← R(Core)