UNIVERSITÄT ZU LÜBECK
INSTITUT FÜR INFORMATIONSSYSTEME

Özgür L. Özçep

Query Reformulation

Lecture 13

Foundations of Ontologies and Databases
for Information Systems
CS5130 (Winter 16/17)


LAST 2 MILEs
AHEAD

Recap

- Talked about higher-level declarative stream processing using STARQL as example
  - Declarative: streams have assertional status
  - High-level: Have to incorporate (reason over) a background KB

This Lecture

- Back to general topic of query rewriting
  - Here we call it "Query reformulation"
  - Examples from DB theory: Reformulating a query w.r.t. views

- Uses the "last significant property of FOL that has come to light" namely interpolation (Benthem 2008)

  Lit: J. van Benthem. The many faces of interpolation. Synthese, 164(3):451?460, 2008.

- Sources used for this lecture
  - Mainly: Slides from an invited talk of M. Benedikt at DL 2014 with my annotations
  - B. ten Cate: "Craig Interpolation Theorems and Database Applications", talk given at UC Berkeley - Logic Colloquium, November 7, 2014
  - M. Benedikt, J. Leblay, B. ten Cate, and E. Tsamoura. Generating Plans from Proofs: The interpolation-based Approach to Query Reformulation. Synthesis Lectures on Data Management, 2016.

Example for Rewriting

Following Road Example from (tenCate 2014)

First Example: View-Based Query Reformulation

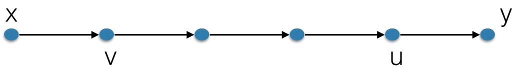
- **Road network database**: Road(x,y)
- **Views**:
  - $V_2(x,y)$ = "∃ path of length 2 from x to y" = $\exists u\ \mathrm{Road}(x,u) \wedge \mathrm{Road}(u,y)$
  - $V_3(x,y)$ = "∃ path of length 3 from x to y" = $\exists u,v\ \mathrm{Road}(x,u) \wedge \mathrm{Road}(u,v) \wedge \mathrm{Road}(v,y)$
  - …
- **Observation**: V_4 can be expressed in terms of V_2.
- **Puzzle** (Afrati'07): can V_5 be expressed (in FO logic) in terms of V_3 and V_4?

Solution to the puzzle

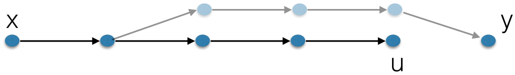
$$V_5(x,y) \Leftrightarrow \exists u \; (\; V_4(x,u) \wedge \forall v \; (\; V_3(v,u) \rightarrow V_4(v,y) \;) \;)$$

Proof:

[$\Longrightarrow$]



[$\Longleftarrow$]



Why this Example is Important

- A **conjunctive query** (CQ) is a FO formula built up using only $\wedge$, $\exists$.
  - Conjunctive queries are the most common type of database queries.
  - Every positive-existential FO formula is equivalent to a union of CQs.

- Remarkable fact:
  - V_3, V_4 and V_5 are all defined by CQs over the base relation (Road).
  - V_5 is definable in terms of V_3 and V_4 but not by means of a CQ.

Classic Results

Querying using views has been around since the 1980s. E.g.,

- **Theorem** (Levy Mendelzon Sagiv Srivastava '95): there is an effective procedure to decide whether a conjunctive query is rewritable as a conjunctive query over a given set of conjunctive views.

- **Open problem** (Nash, Segoufin, Vianu '10): is there an effective procedure to decide if a conjunctive query is answerable on the basis of a set of conjunctive views (a.k.a., is determined by the views)? if so, in what language can we express the rewriting?

NB: The Beth definability theorem (1953) tell us that, if a FO query is answerable on the basis of a set of FO views, then, it has a FO rewriting.

New Perspectives on Query Reformulation

Michael Benedikt, based on joint work with:
Julien Leblay, Efi Tsamoura, Michael Vanden Boom (Oxford)
Balder ten Cate (LogicBlox & UC-Santa Cruz)

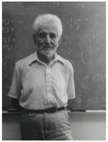
- Overview the theory behind **proof-driven querying**, which takes:

  - a conjunctive query que**ry Q =** $\exists\, \mathbf{x_1 \ldots x_m\, A_1(x_1 \ldots) \wedge \ldots \wedge A_m(x_m \ldots)}$
  - metadata consisting of relation descriptions, access methods, first order constraints, and costs of access

  and generates (if possible) a low cost plan that gives the same answer as Q on every instance satisfying the constraints

- Give connections to:
  - proof theory
  - preservation theorems in model theory
  - data integration
  - work of other DL '14 invited speakers

William Craig's Approach to Query Reformulation



Meta-Algorithm

In two papers from 1957, W. Craig developed a general methodology for reformulating a query **Q** in some restricted target language with respect to constraints:

- Pick out a property that **Q** should possess in order to have the desired reformulation (E.g.: Determinedness)
- Write out the property as an implication/proof goal in logic
- Look for a proof fulfilling the proof goal
- Generate a reformulation from the proof using an **interpolation algorithm**

Craig in Action

Goal: given boolean conjunctive query **Q** find a **first-order logic reformulation** over some restricted signature **T** with respect to constraints Σ (constraints also in FO)

Show that **Q** is **determined** over **T** with respect to Σ :
for any two instances **I** and **I'** satisfying the constraints Σ
if **I** and **I'** have the same **T** facts, then **Q(I)=Q(I')**

Do this by showing that the following implication, using two copies of relations, is valid:
$\mathbf{Q} \wedge \Sigma \wedge \Sigma\,' \wedge [\bigwedge_{\mathcal{R} \in \mathcal{T}} \forall\, x_1 \ldots x_n\, R(x_1 \ldots x_n) \leftrightarrow R'(x_1 \ldots x_n)] \rightarrow \mathbf{Q'}$
$\Sigma\,'$ = copy of constraints on primed relations
Q' = copy of query on **Q** primed relations

To witness the validity, search for a **proof** that:
$\mathbf{Q} \wedge \Sigma \wedge \Sigma\,' \wedge [\bigwedge_{\mathcal{R} \in \mathcal{T}} \forall\, x_1 \ldots x_n\, R(x_1 \ldots x_n) \leftrightarrow R'(x_1 \ldots x_n)] \vdash \mathbf{Q'}$
Re-arranging: $\mathbf{Q} \wedge \Sigma \vdash [\Sigma\,' \wedge \wedge_{\mathcal{R} \in \mathcal{T}} \forall\, x_1 \ldots x_n\, R(x_1 \ldots x_n) \leftrightarrow R'(x_1 \ldots x_n)] \rightarrow \mathbf{Q'}$
Craig 1957 paper**: from** any **interpolant** for re-arranged proof goal, can obtain a first-order reformulation of **Q** over **T** with respect to Σ

Interpolant: If $\rho_1 \vdash \rho_2$, an interpolant is a formula ρ such that $\rho_1 \vdash \rho \vdash \rho_2$,
and ρ uses only relations common to ρ_1 and ρ_2

Craig's Reformulation Recipe

To find a **first-order reformulation** of query **Q** over some restricted signature **T** with respect to constraints Σ .

Show that **Q** is **determined** over **T** with respect to Σ :
for any two instance **I** and **I'** satisfying the constraints Σ
if **I** and **I'** have the same **T** facts, then **Q(I)=Q(I')**

This condition holds iff there is a **proof** that:

$$\mathbf{Q} \wedge \Sigma \wedge \Sigma' \wedge \ [\textstyle\bigwedge_{\mathcal{R} \in \mathcal{T}} \forall\, \mathbf{x_1 \ldots x_n\ R(x_1 \ldots x_n)} \leftrightarrow \mathbf{R'(x_1 \ldots x_n)}\ )] \vdash \mathbf{Q'}$$

Craig '57:

- ***Gave a proof system complete for finding these implications***
- ***Gave an interpolation algorithm***: from a proof witnessing $\rho_1 \vdash \rho_2$,
efficiently extracts a formula ρ such that $\rho_1 \vdash \rho \vdash \rho_2$, and ρ uses only relations common to ρ_1 and ρ_2
- Showed that any interpolant gives a reformulation, and if there is **any** first-order reformulation, this process gives one

First Order Tableau proofs

To Prove: $\exists x\ A(x) \wedge \neg B(x) \wedge C(x) \vdash \neg \forall y\ [(\neg A(y) \wedge E(y)) \vee B(y)]$

Derive a contradiction from:

$\exists x\ A(x) \wedge \neg B(x) \wedge C(x)\ ,\ \forall y\ [(\neg A(y) \wedge E(y)) \vee B(y)]$

$A(c) \wedge \neg B(c) \wedge C(c)\ ,\ \forall y\ [(\neg A(y) \wedge E(y)) \vee B(y)]$

$A(c) \wedge \neg B(c) \wedge C(c)\ ,\ [(\neg A(c) \wedge E(c)) \vee B(c)]$

$A(c)\ ,\ \neg B(c)\ ,\ C(c)\ ,\ [(\neg A(c) \wedge E(c)) \vee B(c)]$

$A(c)\ ,\ \neg B(c)\ ,\ C(c),\ \neg A(c) \wedge E(c)$ $\qquad$ $A(c)\ ,\ \neg B(c)\ ,\ C(c),\ B(c)$

$A(c)\ ,\ \neg B(c)\ ,\ C(c),\ \neg A(c)\ ,\ E(c)$

Tableau proofs

To Prove: $\exists x \; A(x) \land \neg B(x) \land C(x) \;\vdash\; \neg \forall y \, [(\neg A(y) \land E(y)) \lor B(y)]$

Derive a contradiction from:

$\exists x \; A(x) \land \neg B(x) \land C(x) , \;\; \forall y \, [(\neg A(y) \land E(y)) \lor B(y)]$

$A(c) \land \neg B(c) \land C(c) , \forall y \, [(\neg A(y) \land E(y)) \lor B(y)]$

$A(c) \land \neg B(c) \land C(c) , \;\; [(\neg A(c) \land E(c)) \lor B(c)]$

$A(c) , \;\; \neg B(c) , C(c) , \;\; [(\neg A(c) \land E(c)) \lor B(c)]$

$A(c) , \;\; \neg B(c) , C(c), \;\; \neg A(c) \land E(c)$

$A(c) , \;\; \neg B(c) , C(c), \;\; B(c)$

$A(c) , \;\; \neg B(c) , C(c), \;\; \neg A(c) , \;\; E(c)$

Tableaux and Craig's Interpolation Algorithm

$\exists x\ A(x) \wedge \neg B(x) \wedge C(x) \vdash \neg \forall y\, [\, (\neg A(y) \wedge E(y)\,) \vee B(y)]$

$\exists x\ A(x) \wedge \neg B(x) \wedge C(x)\ ,\ \forall y\, [\, (\neg A(y) \wedge E(y)\,) \vee B(y)]$

$A(c) \wedge \neg B(c) \wedge C(c)\ ,\ \forall y\, [\, (\neg A(y) \wedge E(y)\,) \vee B(y)]$

$A(c) \wedge \neg B(c) \wedge C(c)\ ,\ [\, (\neg A(c) \wedge E(c)\,) \vee B(c)]$

$A(c)\ ,\ \neg B(c)\ ,\ C(c),\ [\, (\neg A(c) \wedge E(c)\,) \vee B(c)]$

$A(c)\ ,\ \neg B(c)\ ,\ C(c),\ \neg A(c) \wedge E(c)$ $\qquad$ $A(c)\ ,\ \neg B(c)\ ,\ C(c),\ B(c)$

$A(c)\ ,\ \neg B(c)\ ,\ C(c),\ \neg A(c)\ ,\ E(c)$

Tableaux and Craig's Interpolation Algorithm

$\exists x\, A(x) \wedge \neg B(x)$

$\exists x\ A(x) \wedge \neg B(x) \wedge C(x) \ \vdash\ \neg \forall y\, [\, (\neg A(y) \wedge E(y)\,) \vee B(y)]$

$\exists x\ A(x) \wedge \neg B(x) \wedge C(x)\ ,\ \ \forall y\, [\, (\neg A(y) \wedge E(y)\,) \vee B(y)]$

$A(c) \wedge \neg B(c) \wedge C(c)\ ,\ \forall y\, [\, (\neg A(y) \wedge E(y)\,) \vee B(y)]$ Exists x A(x) & ┐B(x)

... ...

$A(c) \wedge \neg B(c) \wedge C(c)\ ,\ [\, (\neg A(c) \wedge E(c)\,) \vee B(c)]$ A(c) & ┐B(c)

... ...

$A(c)\ ,\ \neg B(c)\ ,\ C(c),\ [\, (\neg A(c) \wedge E(c)\,) \vee B(c)]$ A(c) & ┐B(c)

$A(c)\ ,\ \neg B(c)\ ,\ C(c),\ \neg A(c) \wedge E(c)$ A(c)

$A(c)$

$A(c)\ ,\ \neg B(c)\ ,\ C(c),\ \neg A(c)\ ,\ E(c)$

$\neg B(c)$

$A(c)\ ,\ \neg B(c)\ ,\ C(c),\ B(c)$

Interpolant (in red) construction by bottom up strategy

Summary: Craig's Methodology Episode 1

To find a **first-order reformulation** of query **Q** over some restricted signature **T** with respect to constraints Σ .

Show that **Q** is **determined** over **T** with respect to Σ :
for any two instance **I** and **I'** satisfying the constraints Σ
if **I** and **I'** have the same **T** facts, then **Q(I)=Q(I')**

This condition holds iff:

$$\mathbf{Q} \wedge \Sigma \wedge \Sigma' \wedge [\bigwedge_{\mathcal{R} \in \mathcal{T}} \forall\, \mathbf{x_1 \ldots x_n\ R(x_1 \ldots x_n) \leftrightarrow R'(x_1 \ldots x_n)}] \vdash \mathbf{Q'}$$

Apply Craig's **interpolation algorithm** to any tableau proof witnessing the re-arranged version of this: it will give a reformulation.

But wait...

- Craig's Lemma works for FOL where we have natural domain semantics and allow for infinite structures
- In DB one normally has active domain semantics and is interested in finite structures (but in the finite interpolation does not hold for FOL)

- Idea
  - Consider fragments of FOL where natural domain semantics (alias classical FOL semantics) = active domain semantics (alias DB semantics) $\Longrightarrow$ RQFO
  - Consider further restrictions on constraints (Guarded fragment logics)

- RQFO = FOL with **relativized quantifiers**
  - $\exists \vec{x} R(\vec{y}, \vec{x}) \wedge \phi(\vec{y}, \vec{x}, \vec{t}))$
  - $\forall \vec{x} R(\vec{y}, \vec{x}) \rightarrow \phi(\vec{y}, \vec{x}, \vec{t}))$
    where ϕ again a RQFO
  - Note that RQFO are not necessarily safe ($A(x) \vee B(y)$ still allowed).

Interpolation does not hold in the Finite for FOL

- $\mathcal{L}$ a logic; $\qquad K =$ class of τ structures
- Call K good iff for class

$$K_< := \{(\mathfrak{A}, <) \mid \mathfrak{A} \in K, < \text{ an ordering on } \mathfrak{A}\}$$

there is sentence ϕ in vocabulary $\tau \cup <$ such that

$$K_< = Mod(\phi)$$

- $\mathcal{L}$ is **closed under order-invariant sentences in the finite** iff every good structure K is axiomatizable by a τ sentence ψ

Observation

Logics with interpolation property are closed under order-invariant sentences in the finite.

- Proof: Consider $\phi = \phi(<)$ with $Mod(K) = \phi$. Then $\phi(<) \models_{fin} (\text{“} <' \text{ is an ordering''} \to \phi(<'))$. If ψ is interpolant, then $Mod(K) = \psi$.

Interpolation does not hold in the Finite for FOL

- $\mathcal{L}$ a logic; K = class of τ structures
- Call K good iff for class

$$K_< := \{(\mathfrak{A}, <) \mid \mathfrak{A} \in K, < \text{ an ordering on } \mathfrak{A}\}$$

  there is sentence ϕ in vocabulary $\tau \cup <$ s.t. $K_< = Mod(\phi)$
- $\mathcal{L}$ is **closed under order-invariant sentences in the finite** iff every good structure K is axiomatizable by a τ sentence ψ

Proposition

1. *FOL is not closed under order-invariant sentences in the finite*
2. *FOL does not have interpolation property in the finite*

- Proof of 1.:
  - K = boolean algebras with even number of atoms.
  - K not axiomatizable in FOL (use Fraissee game)
  - But $K_<$ is axiomatizable by ϕ:
  - ϕ = boolean algebra axioms + order axioms + sentence A
  - A = there is an element covering exactly atoms in even position and the last atom.

Craig works for database-style queries + constraints

To find a **relational algebra** reformulation of conjunctive query **Q** over subschema **T** with respect to **relational algebra constraints** Σ equivalently, constraints and reformulation built up via **relative quantifiers**:

$\exists\, \mathbf{x_1 \ldots\, x_m}\ \mathbf{R(x_1 \ldots\, x_m\, ,\, y_1\, \ldots\, y_n\,)} \wedge \phi$ „Safe quantification for domain independence“

$\forall\, \mathbf{x_1\, \ldots\, x_m}\ \mathbf{R(x_1 \ldots\, x_m\, ,\ y_1\, \ldots\, y_n\,)} \rightarrow \phi$

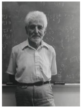
Show that **Q** is determined over **T** with respect to Σ :
for any two instance **I** and **I'** satisfying the constraints Σ
if **I** and **I'** have the same **T** facts, then **Q(I)=Q(I')**

To do this, need to show:

$$\mathbf{Q} \wedge \Sigma \wedge \Sigma' \wedge [\textstyle\bigwedge_{\mathcal{R} \in \mathcal{T}} \forall\, \mathbf{x_1 \ldots\, x_n\ R(x_1 \ldots x_n)} \leftrightarrow \mathbf{R'(x_1\, \ldots\, x_n)}] \vdash \mathbf{Q'}$$

Apply Craig's **interpolation algorithm** to any tableau proof of re-arranged proof goal to extract a reformulation. ***Observe*** *that if all formulas involved are relativized, the resulting interpolant can be put in relativized form as well*.

Based on a true story






This is a reconstruction of Craig's work.

The exposition of interpolation comes from Fitting's 1996 textbook.

The connection between interpolation and query reformulation was discovered by Segoufin and Vianu [PODS 2005] and explored more fully in Nash, Segoufin, Vianu [TODS 2010]. They make use of interpolation results of Otto [BSL 2000].

A. Nash, L. Segoufin, and V. Vianu. Views and queries: Determinacy and rewriting. ACM Trans. Database Syst., 35(3):21:1–21:41, 2010.

M. Otto. An interpolation theorem. Bulletin Symbolic Logic, 6(4):447–462, 12 2000.

The fact that interpolation can be used to generate reformulations constructively has been observed several places:

- Toman and Weddell, Fundamentals of Physical Design and Query Reformulation (Book)
- Franconi, Kerhet and Ngo, JAIR 2013



L. Segoufin and V. Vianu. Views and queries: determinacy and rewriting. PODS 2005, pages 49–60. ACM, 2005.

E. Franconi, V. Kerhet, and N. Ngo.Exact query reformulation over databases with first-order and description logics ontologies. J. Artif. Intell. Res. (JAIR), 48:885–922, 2013.

Craig works for finding USPJ Reformulations

To find a **positive existential** (i.e. **UCQ/USPJ** with inequalities) reformulation of conjunctive query **Q** over subschema **T** with respect to (relativized FO) constraints Σ .

Show that **Q** is **monotone** over **T** with respect to Σ :

for any two instances **I** and **I'** satisfying the constraints Σ
if **T** facts of **I** are contained in the **T** facts of **I'**, then **Q(I)** $\subseteq$ **Q(I')**

To do this need to show:

$$\mathbf{Q} \wedge \Sigma \wedge \Sigma' \wedge [\bigwedge_{\mathcal{R} \in \mathcal{T}} \forall \mathbf{x_1 \ldots x_n}\ \mathbf{R(x_1 \ldots x_n)} \rightarrow \mathbf{R'(x_1 \ldots x_n)}] \vdash \mathbf{Q'}$$

Apply Craig's **interpolation algorithm** to a tableau proof of the re-arranged proof goal and extract a reformulation. *Argue that the resulting reformulation will be positive existential.*

(Lyndon: There is positive (negative) occurrence of relation symbol in interpolant iff is positive (negative) occurrence in premise and conclusion)

Interpolants produced by Craig's algorithm "respect positivity of relations"**:**
Lyndon Interpolation Theorem

Craig works for Existential Reformulations

To find an **existential** (UCQ with atomic negation) reformulation of conjunctive query **Q** over some restricted signature **T** with respect to constraints Σ .

Show that **Q** is **induced-subinstance monotone** with respect to Σ :

for any two instances **I** and **I'** satisfying the constraints Σ
if **T** facts of **I** contained in the **T** facts of **I'**, **and I'** has no **T** fact using elements from **I,**
then $\mathbf{Q(I)} \subseteq \mathbf{Q(I')}$

Second condition

To do this, need to show:

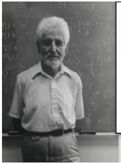
$$\mathbf{Q} \wedge \Sigma \wedge \Sigma' \wedge \left[\bigwedge_{\mathcal{R} \in \mathcal{T}} \forall\, \mathbf{x_1 \ldots x_n}\ \mathbf{R(x_1 \ldots x_n)} \rightarrow \mathbf{R'(x_1 \ldots x_n)} \wedge \mathbf{\ldots}\ \right] \vdash \mathbf{Q'}$$

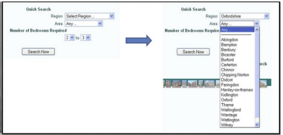
Rewrite and apply Craig's interpolation algorithm to tableau proof of re-arranged goal.
Argue that this gives the desired existential reformulation.

Summary: Craig's Recipe for Restricted Vocabulary Reformulations

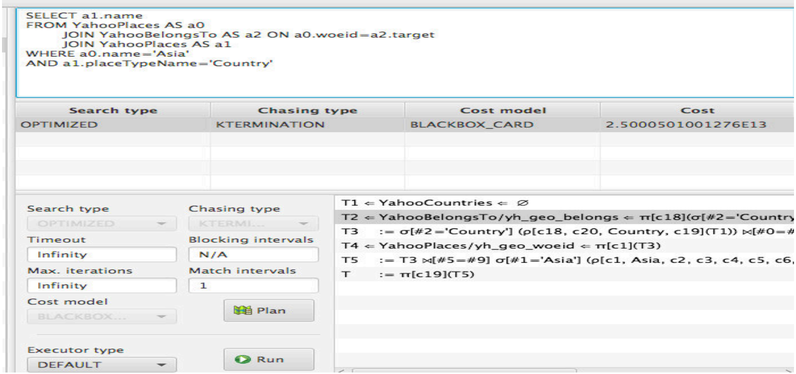
Reformulation Goal	Semantic Property	Search for proof of this entailment, apply interpolation	Parallel with Theorem in Model Theory
Q is FO-rewritable over **T** w.r.t. constraints	**Q** is determined by **T** w.r.t constraints	$\mathbf{Q} \wedge \Sigma \wedge \wedge_{\mathbf{T}} \mathbf{R_i = R'_i} \vdash \Sigma\mathbf{'} \rightarrow \mathbf{Q'}$	Projective Beth Definability
Q is $\exists^+$-rewritable over **T** w.r.t constraints	**Q** is monotone in **T** w.r.t. constraints	$\mathbf{Q} \wedge \Sigma \wedge \wedge_{\mathbf{T}} \mathbf{R} \subseteq \mathbf{R'_i} \vdash \Sigma\mathbf{'} \rightarrow \mathbf{Q'}$	Rough analogy: Lyndon Preservation Theorem/ Hom. Pres. Thm
Q $\exists$-rewritable over **T** w.r.t. constraints	**Q** induced-subinst. monotone in **T** w.r.t constraints	$\mathbf{Q} \wedge \Sigma \wedge \wedge_{\mathbf{T}} \mathbf{R_i} \subseteq \mathbf{R'_i} \wedge \ldots \vdash \Sigma\mathbf{'} \rightarrow \mathbf{Q'}$	(Projective) Los-Tarski Preservation Theorem

Craig works for Access Patterns





To find a **relational algebra plan using a fixed set of access methods** equivalent to query **Q** with respect to constraints Σ .



Access Methods

- **Access method**: a pair (R,X) where R is an n-ary relation and $X \subseteq \{1, \ldots, n\}$ is a set of "input positions"
  - *Relation R can be accessed if specific values are provided for the positions in X.*
- **Examples**:
  - (Yellowpages(name,city,address,phone#), {1,2})
  - $(R, \emptyset)$ means **free (unrestricted) access** to R.
  - $(R, \{1, \ldots, n\})$ means only **membership tests** for specific tuples.
- There may be any number of access methods for a given relation. The allowed access methods for a relation can be assumed to be an upwards closed set.

Craig works for Access Patterns

To find a **relational algebra plan using a fixed set of access methods** equivalent to query **Q** with respect to constraints Σ .

Show that **Q** is **access-determined** with respect to access methods and Σ :
for any two instances **I** and **I'** satisfying the constraints Σ
if **I** and **I'** have the same accessible data, then **Q(I) = Q(I')**

Need to prove:
$\mathbf{Q} \wedge \Sigma \wedge \Sigma' \wedge$ **["Accessibility Axioms"]** $\vdash$ **Q'**

Example accessibility axiom:
Suppose $\mathbf{R}(x_1, x_2)$ has an access method on the first position.

Instead of previous "transfer axiom" $\forall \mathbf{x_1\, x_2\, R(x_1, x_2)} \rightarrow \mathbf{R'(x_1, x_2)}$

We have, for any n-ary relation **F** in the signature, for every position in **F**, an axiom:

$$\forall \mathbf{x_1\, x_2 \ldots x_n\, F'(\ldots x_1 \ldots)} \wedge \mathbf{R(x_1, x_2)} \rightarrow \mathbf{R'(x_1, x_2)}$$

Craig works for Access Patterns

To find a **relational algebra plan using a fixed set of access methods** equivalent to query **Q** with respect to constraints Σ .

Show that **Q** is **access-determined** with respect to access methods and Σ :
for any two instances **I** and **I'** satisfying the constraints Σ
if **I** and **I'** have the same accessible data, then **Q(I) = Q(I')**

Need to prove:
Q $\wedge \Sigma \wedge \Sigma' \wedge$ **["Accessibility Axioms"]** $\vdash$ **Q'**

Re-arrange and apply Craig's interpolation algorithm to the proof to extract a reformulation. *Argue that this can be converted to a plan that fits the access methods.*

Interpolants produced by Craig's algorithm "reflect the quantification patterns on the left and right of the proof symbol": **Access Interpolation Theorem**

Craig works for Access-Pattern based Restrictions

To **f**ind a **negation-free** (**USPJ) plan using a fixed set of access methods** equivalent to query **Q** with respect to constraints Σ .

Show that **Q** is **access-monotone** with respect to methods and Σ :
for any two instances **I** and **I'** satisfying the constraints Σ
the accessible data in **I** is contained in the accessible data of **I'**
then **Q(I)** $\subseteq$ **Q(I')**

Need to show:
Q $\wedge\ \Sigma \wedge \Sigma$ **'**$\wedge$ **["Uni-directional Accessibility Axioms"]** $\vdash$ **Q'**

Re-arrange and apply Craig's interpolation algorithm to any proof to extract a reformulation. Argue using Access Interpolation Theorem that this gives a plan of the desired shape.

Binding Patterns

Consider first order logic built up from equalities and true/false via relative quantifiers:

$\forall$ $\mathbf{y_1 \ldots y_n\ R(x_1 \ldots x_m , y_1 \ldots y_n)} \rightarrow \phi$

$\exists$ $\mathbf{y_1 \ldots y_n\ \ R(x_1 \ldots x_m , y_1 \ldots y_n)} \wedge \phi\mathbf{(x_1 \ldots x_m, y_1 .. y_n)}$

The **binding pattern** of a formula tells where in each relation we have free variables in relative quantifications.

bindpatt($\forall$ $\mathbf{y_1\ y_2\ R(x_1, y_1, y_2)} \rightarrow \mathbf{S(x_1,y_2))= \{(R, \{1\}), (S,\{1,2\})\}}$

Access Methods "Used" by a Formula

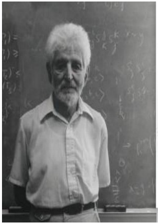
BindPatt(ϕ) is the set of access methods "used" by ϕ.

$$
\begin{array}{lcl}
\mathsf{BindPatt}(\top) = \mathsf{BindPatt}(x=y) & = & \emptyset \\
\mathsf{BindPatt}(R(t_1,\ldots,t_n)) & = & \{(R,\{1,\ldots,n\})\} \\
\mathsf{BindPatt}(\neg\varphi) & = & \mathsf{BindPatt}(\varphi) \\
\mathsf{BindPatt}(\varphi\wedge\psi) & = & \mathsf{BindPatt}(\varphi)\cup\mathsf{BindPatt}(\psi) \\
\mathsf{BindPatt}(\varphi\vee\psi) & = & \mathsf{BindPatt}(\varphi)\cup\mathsf{BindPatt}(\psi) \\
\mathsf{BindPatt}(\exists\vec{x}(R(t_1,\ldots,t_n)\wedge\varphi)) & = & \mathsf{BindPatt}(\varphi)\cup\{(R,\{i\mid t_i\notin\vec{x}\})\} \\
\mathsf{BindPatt}(\forall\vec{x}(R(t_1,\ldots,t_n)\rightarrow\varphi)) & = & \mathsf{BindPatt}(\varphi)\cup\{(R,\{i\mid t_i\notin\vec{x}\})\}
\end{array}
$$

- For example BindPatt($\forall y(Rxy \rightarrow Sxy)$) = { (R,{1}), (S,{1,2}) }
- A FO formula ϕ is executable if BindPatt(ϕ) consists of allowed access methods.
- **Fact**: Each executable FO formula admits a query plan, and, conversely, every formula that admits a query plan is equivalent to an executable FO formula.
  - Query plan = sequence of allowed accesses and/or relational algebra operations.

11

Craig gives decision procedures

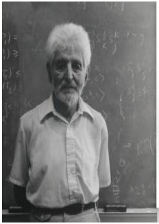


To find a positive existential reformulation of query **Q** over restricted signature **T** with respect to **Tuple-generating Dependencies (TGDs)** Σ

$$\forall\, \mathbf{x_1 \ldots x_m}\ \ [\mathbf{R_1(x_1\ldots\ )} \wedge \mathbf{\ldots} \wedge \mathbf{R_m(x_1\ldots)} \rightarrow \exists\, \mathbf{y_1 \ldots y_m}\ \mathbf{S_1(x_1\ldots.\ y_1\ldots)} \wedge \mathbf{\ldots}]$$

Very common in databases: e.g. inclusion dependencies (referential constraints), data integration mappings

Craig gives decision procedures




To find a positive existential reformulation of query **Q** over restricted signature **T** with respect to **Tuple-generating Dependencies** Σ .

$$\forall \mathbf{x_1 \ldots x_m}\ \ [\mathbf{R_1(x_1\ldots)} \wedge \ldots \wedge \mathbf{R_m(x_1\ldots)} \rightarrow \exists\, \mathbf{y_1 \ldots y_m}\ \mathbf{S_1(x_1\ldots.\ y_1\ldots)} \wedge \ldots]$$

Show that **Q** is monotone with respect to Σ :
for any two instance **I** and **I'** satisfying Σ if **I** and **I'** have the same **T** facts,
then **Q(I)=Q(I')**

Note: This is the same as above;
the only point is that we can use special proof procedures

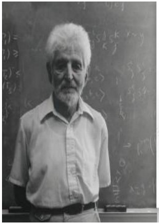
Search for **proof** that:

$$\mathbf{Q} \wedge \Sigma \wedge \Sigma' \wedge\ \ [\textstyle\bigwedge_{\mathcal{R} \in \mathcal{T}} \forall\, \mathbf{x_1 \ldots x_n}\ \mathbf{R(x_1 \ldots x_n)} \rightarrow \mathbf{R'(x_1 \ldots x_n)}] \vdash \mathbf{Q'}$$

Q contained in **Q'** with respect to an augmented set of TGDs

Specialized proof procedures have been developed for proving implications of this form: **the chase**

Craig gives decision procedures



To find a positive existential reformulation of query **Q** over restricted signature **T** with respect to **Tuple-generating Dependencies** Σ .

Show that **Q** is monotone with respect to Σ :
for any two instance **I** and **I'** satisfying Σ **if I** and **I'** have the same **T** facts, then **Q(I)=Q(I')**

Search for **proof** that:

$$\mathbf{Q} \wedge \Sigma \wedge \Sigma' \wedge \ [\textstyle\bigwedge_{\mathcal{R} \in \mathcal{T}} \forall\, \mathbf{x_1 \ldots x_n\ R(x_1 \ldots . x_n) \rightarrow R'(x_1 \ldots x_n)}] \vdash \mathbf{Q'}$$

Q contained in **Q'** with respect to an augmented set of TGDs

Apply Craig's algorithm to a chase proof (which is a tableau proof!) to extract a positive existential reformulation

Chase again

- Remember the chase construction from our lecture on data exchange
  - Use TGDs as firing rules.
  - Chase procedure complete and correct; termination under some constraints für TGDs
- Here for testing $Q \wedge \Sigma \models Q^*$, where Q a CQ
- Canonical database $DB(Q)$ for input Q:

  instance with elements all constants in Q and additional distinct constants for variables; fact in $DB(Q)$ iff it is contained in Q (modulo substituting new constants with variables)
- Chase procedure
  1. start with $DB(Q)$
  2. iteratively apply TGDs
  3. Stop if some fact contained (via a homomorphism) in Q^*
- Chase derivation can be transformed into a tableau derivation

End-to-end reformulation algorithms via Craig's technique

To find a positive existential reformulation of query **Q** over restricted signature **T with respect to some set of TGD constraints Σ where the chase terminates** (i.e. there is a maximal tableau**)**.

Show that **Q** is **monotone** over **T** with respect to Σ:
for any two instances **I** and **I'** satisfying the constraints Σ
if **T** facts of **I** are contained in the **T** facts of **I'**, then **Q(I)** $\subseteq$ **Q(I')**

Perform the chase effectively to determine if there is a proof that:

$$\mathbf{Q} \wedge \Sigma \wedge \Sigma' \wedge [\bigwedge_{\mathcal{R} \in \mathcal{T}} \forall \mathbf{x_1 \ldots x_n} \, \mathbf{R(x_1 \ldots . x_n)} \rightarrow \mathbf{R'(x_1 \ldots x_n)}] \vdash \mathbf{Q'}$$

Apply Craig's interpolation algorithm to the proof to extract a reformulation.

This yields many prior results in the database literature.

End-to-end reformulation algorithms via Craig's technique

Theorem [Levy Mendelzon Sagiv Srivistava PODS 95]:

If the constraints only say that each **R** in **T** is defined by a conjunctive view definition over another set of relations **B**: Compare the road example from the very beginning

$\forall x_1 \ldots x_n\ \mathbf{R}(x_1 \ldots x_n) \leftrightarrow \mathbf{Q_R}(\mathbf{x_1} \ldots \mathbf{x_n})$,

$\mathbf{Q_R}$ a conjunctive query using a set of base relations **B** disjoint from **T.**

One can decide whether a query **Q** written over **B**
can be rewritten as a conjunctive query using only the view relations.

Perform the chase effectively to determine if there is a proof that:

$\mathbf{Q} \wedge \Sigma \wedge \Sigma' \wedge\ [\bigwedge_{\mathcal{R} \in \mathcal{T}} \forall\, \mathbf{x_1} \ldots \mathbf{x_n}\ \mathbf{R}(\mathbf{x_1} \ldots \mathbf{x_n}) \rightarrow \mathbf{R}'(\mathbf{x_1} \ldots \mathbf{x_n})] \vdash \mathbf{Q}'$

Apply Craig's interpolation algorithm to the proof to extract a reformulation.

New end-to-end reformulation algorithms via Craig's technique

To find a FO reformulation (resp. positive existential, existential reformulation, RA-plan ...) of query **Q** with respect to **some set of TGD constraints Σ where the chase is well-behaved**. e.g. **Inclusion dependencies, LAV schema mappings, Guarded TGDs**.

Show that **Q** is determined over **T** (resp. monotone overt **T**, induced-subinst. monotone, access-determined over the access methods. ..) with respect to Σ :

Effectively determine whether there is **a chase proof** that:
Q $\wedge\ \Sigma \wedge \Sigma' \wedge$ **[........]** $\vdash$ **Q'**

Apply an interpolation algorithm to the proof to extract a reformulation.

Gives first effective reformulation algorithms for very expressive constraint languages (Guarded TGDs, Guarded Fragment, ...)

Craig gives completeness over finite instances

To find a first-order reformulation (resp. positive existential reformulation, RA-plan...) **t**hat is **equivalent to Q over every finite instance**, with respect to some "well-behaved" class of constraints Σ

Show that **Q** is determined (resp. **monotone....)** with respect to Σ**.**:

Search for a proof effectively to determine if:
$\mathbf{Q} \wedge \Sigma \wedge \Sigma' \wedge$ **[...]** $\vdash$ **Q'**

Apply Craig's interpolation algorithm to the proof to extract a reformulation

Use **finite controllability results** to argue that this process is complete for finding plans that work over all finite instances.
E.g. for Guarded TGDs, Guarded Fragment constraints, terminating chase classes....

Craig can find interesting plans

To search for a **USPJ plan** using a fixed set of access methods equivalent to query **Q** with respect to constraints Σ .

Search for a proof that **Q** is **access-monotone** with respect to Σ :
for any two instances **I** and **I'** satisfying the constraints Σ
the accessible data in **I** is contained in the accessible data of **I'**
then **Q(I)** $\subseteq$ **Q(I')**

Do this by searching the space of proofs showing that:
Q $\wedge\ \Sigma \wedge \Sigma'\wedge$ **["Uni-directional Accessibility Axioms"]** $\vdash$ **Q'**

The plans produced by applying Craig's interpolation algorithm to the proofs will include many natural optimizations used in plan search.

Craig can find low-cost plans

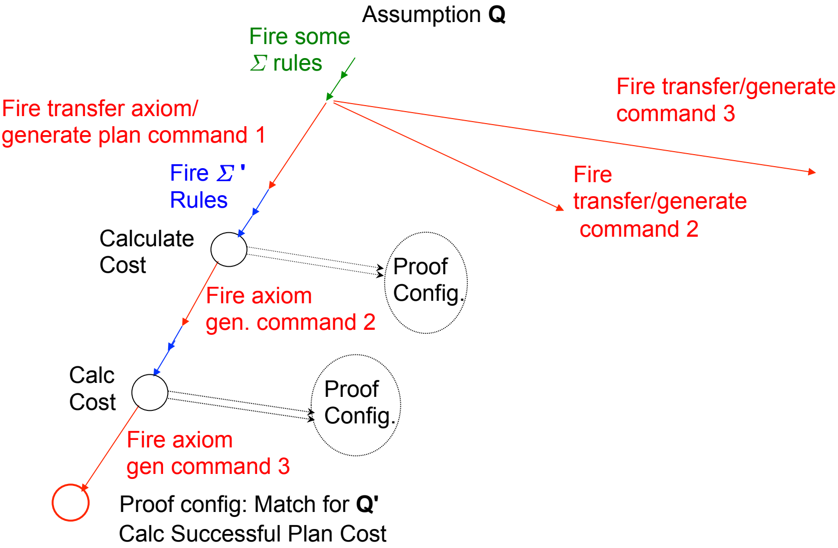
To search for a **low-cost USPJ plan** using a fixed set of access methods equivalent to query **Q** with respect to constraints Σ .

Search for a proof that **Q** is **access-monotone** with respect to Σ :
for any two instances **I** and **I'** satisfying the constraints Σ
the accessible data in **I** is contained in the accessible data of **I'**
then **Q(I)** $\subseteq$ **Q(I')**

Do this by searching the space of proofs showing that:
Q $\wedge$ Σ $\wedge$ Σ **'**$\wedge$ **["Uni-directional Accessibility Axioms"]** $\vdash$ **Q'**

While searching, apply Craig's interpolation algorithm to the proofs, and measure the cost of the resulting plan on-the-fly.

Proof/Plan Search



Craig can find low-cost plans

See **PDQ: Proof-Driven Querying over Web-based Datasources**
(B., Leblay, Tsamoura) in VLDB 2014.

API/examples; **http://www.cs.ox.ac.uk/pdq/**