An interpolation theorem for first-order formulas with relational access restrictions

Balder ten Cate
LogicBlox & UC Santa Cruz

Workshop on the Future of Logic
(on the occasion of Johan van Benthem’s retirement)
Craig Interpolation

- **William Craig (1957)**: For all first-order formulas $\phi$, $\psi$, if $\phi \vdash \psi$, then there is a first-order formula $\chi$ with $\text{Voc}(\chi) \subseteq \text{Voc}(\phi) \cap \text{Voc}(\psi)$ and $\phi \vdash \chi \vdash \psi$. Moreover the formula $\chi$ in question can effectively constructed from a proof of $\phi \vdash \psi$.

- Various extensions and variations have been proved (e.g., Lyndon interpolation, many-sorted interpolation, Otto interpolation).

- **Van Benthem (2008)**: “Craig’s Theorem is about the last significant property of first-order logic that has come to light.”
Relational Access Restrictions

- A **database** is a (finite) relational structure over some schema $S = \{R_1, \ldots, R_n\}$

- **Relational access restrictions**: restrictions on the way we can access the relations $R_1, \ldots, R_n$. 
First Example: View-Based Query Reformulation

- **Road network database**: Road(x,y)

- **Views**:
  - $V_2(x,y) = \exists \text{ path of length 2 from } x \text{ to } y = \exists u \text{ Road}(x,u) \land \text{Road}(u,y)$
  - $V_3(x,y) = \exists \text{ path of length 3 from } x \text{ to } y = \exists u,v \text{ Road}(x,u) \land \text{Road}(u,v) \land \text{Road}(v,y)$
    - ...

- **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 $V_4$?
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 set of 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?
Access Restrictions

• View-Based Query Reformulation:
  
  • Can I reformulate Q as a query using only V₁, ..., Vₙ?
  
  • I.e., is Q equivalent to a query that only uses the symbols V₁, ..., Vₙ (relative to the theory consisting of the view definitions)?

• Query Reformulation w.r.t. Access Methods (more refined):
  
  • Can I find a plan to evaluate Q using only allowed access methods (possibly relative to some theory)?

• First theory work by Chang and Li ’01, followed by work of Nash, Ludaescher, Deutsch, …
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**:

  - \((\text{Telefoongids}(\text{name}, \text{city}, \text{address}, \text{phone#}), \{1,2\})\)
  
  - \((R,\emptyset)\) 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.
Access Methods “Used” by a Formula

- BindPatt(φ) is the set of access methods “used” by φ.

\[
\begin{align*}
\text{BindPatt}(T) &= \text{BindPatt}(x = y) = \emptyset \\
\text{BindPatt}(R(t_1, \ldots, t_n)) &= \{(R, \{1, \ldots, n\})\} \\
\text{BindPatt}(\neg \phi) &= \text{BindPatt}(\phi) \\
\text{BindPatt}(\phi \land \psi) &= \text{BindPatt}(\phi) \cup \text{BindPatt}(\psi) \\
\text{BindPatt}(\phi \lor \psi) &= \text{BindPatt}(\phi) \cup \text{BindPatt}(\psi) \\
\text{BindPatt}(\exists \bar{x}(R(t_1, \ldots, t_n) \land \phi)) &= \text{BindPatt}(\phi) \cup \{(R, \{i \mid t_i \notin \bar{x}\})\} \\
\text{BindPatt}(\forall \bar{x}(R(t_1, \ldots, t_n) \rightarrow \phi)) &= \text{BindPatt}(\phi) \cup \{(R, \{i \mid t_i \notin \bar{x}\})\}
\end{align*}
\]

- For example, BindPatt(∀y(Rxy → Sxy)) = \{(R,\{1\}), (S,\{1,2\})\}

- A “plan” for a query Q is a reformulation Q’ of Q that only uses allowed access methods.
  - Corresponds to a natural operational definition of plans.
Motivation

• **Query Reformulation w.r.t. Access Methods** (more refined):
  
  • *Can I find a plan to evaluate Q using only allowed access methods (possibly relative to some theory)?*

• **Example**: In the road network example, $V_5(x,y)$ admits a first-order plan using only the access methods $(V_2,\emptyset)$ and $(V_3,\{1,2\})$.

• **Motivation**:
  
  • Answering queries using data behind webforms.

  • Query optimization (*if a relation $R(x,y)$ is stored in order sorted on $x$, access method $(R,\{2\})$ is much more costly than access method $(R,\{1\})$.*)

  • ...
The Interpolation-Based Approach to View-Based Query Reformulation
Key concepts

- **Determinacy**: $V_4$ is “determined by” (or “answerable from”) $V_2$.

- **Query reformulations**: $V_4$ “can be reformulated as a query over $V_2$.”

\[
?xy.V_2(x,y) \models ?xy.V_4(x,y)
\]
View-Based Query Reformulation

- Base relations $R_1 \ldots R_n$, view names $V_1 \ldots V_m$

- View definition theory: $T = \{ \forall x (V_1(x) \leftrightarrow \psi_1(x)), \ldots \}$, query $Q$

- The following are equivalent:
  
  1. $Q$ is determined by $V_1 \ldots V_m$ (w.r.t. the theory $T$).
  
  2. A certain FO implication $\theta_{T,Q}$ is valid

  3. $Q$ can be reformulated as a FO query over $V_1 \ldots V_m$. In fact, every Craig interpolant of $\theta_{T,Q}$ is such a reformulation.
What is going on?

• *From a proof of determinacy we are obtaining an actual reformulation.*

• This way of using interpolation to get explicit definitions from implicit ones goes right back to Craig’s work.

• Same technique works for arbitrary theories $T$ (not only view definitions).

• In principle this gives a method for finding query reformulations (but FO theorem proving is difficult).
• **Question:** can we do the same for the case with access methods?

• Answer: yes, using a suitable generalization of Craig interpolation.
Access Interpolation

• **Access interpolation theorem** (Benedikt, tC, Tsamoura, 2014): for all first-order formulas $\phi$, $\psi$, if $\phi \models \psi$, then there is a first-order formula $\chi$ with $\text{BindPatt}(\chi) \subseteq \text{BindPatt}(\phi) \cap \text{BindPatt}(\psi)$ and $\phi \models \chi \models \psi$. Moreover the formula $\chi$ in question can effectively constructed from a proof of $\phi \models \psi$ (in a suitable proof system).

• Can be further refined by distinguishing positive/negative uses of binding patterns.

• Generalizes many existing interpolation theorems (Lyndon, many-sorted interpolation, Otto interpolation)

• Gives rise to a way of testing “access-determinacy” and the existence of reformulations w.r.t. given access methods, as well as a method for finding such reformulations.
Examples in Mathematical Logic

• In set theory, a $\Delta_0$-formula is a formula that only uses access method ($\in, \{2\}$).

• In bounded arithmetic, we study formulas that only use access method ($\leq, \{2\}$).

• The access interpolation theorem generalizes an interpolation theorem for “$\leq$-persistent” formulas by Feferman (1967).

• Closely related: an interpolation theorem for the bounded fragment (equivalently, hybrid logic) by Areces, Blackburn and Marx (2001).
Summary

Querying under Access Restrictions

1. **View-based query reformulation** (restricting to a subset of the signature)

   This is the setting of the (projective) Beth theorem. We look for a proof of implicit definability (“determinacy”) and, from it, compute an explicit definition (“query reformulation”) using Craig interpolation.

2. **Query reformulations given access methods** (more refined)

   Same general technique applies, using a suitable adaptation of Craig interpolation: access interpolation.
Three Important Subtleties

1. Databases are finite structures. But Craig interpolation for first-order logic fails in the finite.

2. For practical applications, we need effective algorithms. But first-order logic is undecidable (we cannot effectively decide if the implication $\theta_{T,Q}$ is valid).

3. For practical applications, we don’t want just any query reformulation, we want one of low cost.
Solutions

• The solution for 1 and 2 is to move to a fragment of first-order logic that is **decidable** and that has the **finite model property**, while still being sufficiently expressive.

• Natural candidate: **the guarded fragment**.
Guar...
Guarded Negation Fragment

- Guarded-Negation fragment (GNFO): a slight further extension of the guarded fragment that does have Craig interpolation.

- Instead of guarding quantifiers we guard the negation.

  \( \phi ::= R(x_1 \ldots x_n) \ | \ x=y \ | \ G(x) \land \neg \phi(x) \ | \ \phi \land \phi \ | \ \exists y. \phi \)

- Note: sentential and unary negation can be trivially guarded by the identity guard \( x=x \).

- GNFO retains all the good properties of GF (Barany, tC, Segoufin 2011),

- It also has (effective) Craig interpolation (Barany, Benedikt, tC 2013; Benedikt, tC, Vanden Boom 2014).
Cost-sensitive Query Reformulation

• Every real-world database management system has a cost-estimate function for query plans (what is the expected execution time).

• We are looking for a proof of \( \theta_{T,Q} \) such that the interpolant obtained from it constitutes a plan that has a low cost.

• Strategy: explore the space of possible proofs guided by a (monotone) plan cost function.

• Ongoing research, collaboration between Oxford University (Michael Benedikt) and LogicBlox. There will be openings for postdocs at Oxford on this.
Thank you
Solution to the puzzle

- $V_5(x,y) \iff \exists u \ ( V_4(x,u) \land \forall v \ ( V_3(v,u) \rightarrow V_4(v,y) ) )$

Diagrammatic proof: