KR in Database Systems Implementation
(or Life beyond Lite Logics and CQ/UCQ)

David Toman

D.R. Cheriton School of Computer Science
University of Waterloo

Joint work with Alexander Hudek and Grant Weddell
Queries and Ontologies

**IDEA:**
Queries are answered not only w.r.t. *explicit data*
but also w.r.t. *background knowledge*
⇒ Ontology-based Data Access (OBDA)

**Example**
- Bob is a BOSS (explicit data)
- Every BOSS is an EMPloyee (ontology)

List all EMPloyees ⇒ \{Bob\} (query)

**How do we answer queries?**
via *logical implication*:

$$\text{Ans}(Q, A, T) = \{Q(a_1, \ldots, a_k) \mid T \cup A \models Q(a_1, \ldots, a_k)\}$$

i.e., answers are *ground Q-atoms* implied by
the *data A* and the *background knowledge T*. 
Queries and Ontologies

**IDEA:**
Queries are answered not only w.r.t. *explicit data* but also w.r.t. *background knowledge* ⇒ Ontology-based Data Access (OBDA)

**Example**
- Bob is a BOSS (explicit data)
- Every BOSS is an EMPloyee (ontology)

List all EMPloyees ⇒ \{Bob\} (query)

**How do we answer queries?**
via logical implication:

\[ \text{Ans}(Q, A, T) = \{ Q(a_1, \ldots, a_k) \mid T \cup A \models Q(a_1, \ldots, a_k) \} \]

i.e., answers are ground \(Q\)-atoms implied by the data \(A\) and the background knowledge \(T\).
Queries and Ontologies

IDEA:

Queries are answered not only w.r.t. *explicit data* but also w.r.t. *background knowledge* ⇒ Ontology-based Data Access (OBDA)

Example

- Bob is a BOSS (explicit data)
- Every BOSS is an EMPloyee (ontology)

List all EMPloyees ⇒ {Bob}

How do we answer queries?

via *logical implication*:

\[
\text{Ans}(Q, \mathcal{A}, \mathcal{T}) = \{Q(a_1, \ldots, a_k) \mid \mathcal{T} \cup \mathcal{A} \models Q(a_1, \ldots, a_k)\}
\]

i.e., answers are *ground Q-atoms* implied by the *data* \(\mathcal{A}\) and the *background knowledge* \(\mathcal{T}\).
Difficulties

**Good/Standard News**

LOGSPACE/PTIME (data complexity) for query answering:
- (U)CQ and
- DL-Lite/$\mathcal{EL}_{⊥}$/$\mathcal{CFD}_{nc}$/“rules”-lite (Horn)

**Bad News**

Everything else: coNP (or worse), in particular:
- no negations in ABox
- no closed-world assumption

Counter-intuitive query answers
- no negative queries or sub-queries
Difficulties

Good/Standard News

LOGSPACE/PTIME (data complexity) for query answering:
- (U)CQ and
- DL-Lite/\mathcal{EL}_\perp/\mathcal{CFD}^\forall_{nc}/“rules”-lite (Horn)

Bad News

Everything else: coNP (or worse), in particular:
- no negations in ABox
- no closed-world assumption

Counter-intuitive query answers
- no negative queries or sub-queries
Difficulties: Unintuitive Answers

Example: DL-Lite

- \( EMP(Sue) \)
- \( EMP \sqsubseteq \exists \text{PHONENUM} \)

\[(\forall x. EMP(x) \rightarrow \exists y. \text{PHONENUM}(x, y))\]
Difficulties: Unintuitive Answers

Example: DL-Lite

- EMP(Sue)
- EMP ⊑ ∃PHONENUM

\((∀x.EMP(x) → ∃y.PHONENUM(x, y))\)

User: Does Sue have a phone number?

Information System: YES
Difficulties: Unintuitive Answers

Example: DL-Lite

- EMP(Sue)
- EMP ⊑ ∃PHONENUM

(∀x.EMP(x) → ∃y.PHONENUM(x, y))

User: Does Sue have a phone number?
Information System: YES

User: OK, tell me Sue’s phone number!
Information System: (no answer)
Difficulties: Unintuitive Answers

Example: DL-Lite

- EMP(Sue)
- EMP ⊑∃PHONENUM

(∀x.EMP(x) → ∃y.PHONENUM(x, y))

User: Does Sue have a phone number?
Information System: YES

User: OK, tell me Sue’s phone number!
Information System: (no answer)

User:
What to do?

Definability and Rewriting

- Queries: range-restricted FOL (a.k.a. SQL)
- Ontology/Schema: range-restricted FOL
- Data: CWA (complete information)

\[ \Sigma_L \]

\[ S_L \]

( Logical Schema )
What to do?

Definability and Rewriting

Queries
range-restricted FOL over $S_L$ definable w.r.t. $\Sigma$ and $S_A$

Ontology/Schema
range-restricted FOL

Data
CWA (complete information for $S_A$ symbols)

$\Sigma_L$ $\models$ $S_L$ $\leftarrow$ $\varphi$ (Logical Schema)

$\Sigma_{LP}$ $\downarrow$ $\downarrow$

$\cdot$ $\cdot$

$\Sigma_P$ $\models$ $S_A \subseteq S_P$ $\leftarrow$ $\psi$ (Physical Schema)

Compiler
Relational Algebra (over $S_A$)

$\uparrow$ $\uparrow$

Evaluator
Answers

$\models$ $S_A \subseteq S_P$
What to do?

Definability and Rewriting

Queries range-restricted FOL over $S_L$ definable w.r.t. $\Sigma$ and $S_A$
Ontology/Schema range-restricted FOL
Data CWA (complete information for $S_A$ symbols)

Query ($S_L$) \arrow{2} \text{Compiler} \arrow{2} \text{Evaluator} \arrow{2} \text{Answers}

Schema ($S_L \cup S_P$) \arrow{2} \text{Compiler} \arrow{2} \text{Evaluator} \arrow{2} \text{Answers}

Data ($S_A \subset S_P$)
Definability and Rewriting

Queries
Ontology/Schema
Data

range-restricted FOL over \( S_L \)
range-restricted FOL
CWA (complete information for \( S_P \))

Query \((S_L)\)
Schema \((S_L \cup S_P)\)
Data \((S_A \subset S_P)\)

Compiler
Relational Algebra (over \( S_A \))
Evaluator
Answers

- users: looks like a single model (of the conceptual schema)
- implementation: many models

but definable queries answer the same in each of them
What to do?

Definability and Rewriting

<table>
<thead>
<tr>
<th>Queries</th>
<th>range-restricted FOL over $S_L$ definable in $S_L$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ontology/Schema</td>
<td>range-restricted FOL</td>
</tr>
<tr>
<td>Data</td>
<td>CWA (complete information for $S_L$)</td>
</tr>
</tbody>
</table>

- Query ($S_L$)
- Schema ($S_L \cup S_P$)
- Data ($S_A \subset S_P$)

- decidable (schema) languages?
- finite models?

Compiler

Relational Algebra (over $S_A$)

Evaluator

Answers
GRAND UNIFIED APPROACH TO QUERY COMPILATION

PART I: WHAT CAN IT DO?
What can this do?

**GOAL**

Generate query plans *that compete with hand-written programs in C*

1. linked data structures, pointers, . . .
2. access to search structures (index access and selection),
3. hash-based access to data (including hash-joins),
4. multi-level storage (aka disk/remote/distributed files), . . .
5. materialized views (FO-definable),
6. updates through logical schema (*needs id invention!*), . . .

. . . all *without* having to code (too much) in C/C++ !
Lists and Pointers

1. **Logical Schema**

   - Employee
     - Number
     - Name
     - Salary
   - Works
     - Employee number
     - Department number
   - Department
     - Number
     - Name
     - Manager

2. **Physical Design:** A linked list of emp records pointing to dept records.

   - Record emp of:
     - Integer num
     - String name
     - Integer salary
     - Reference dept
   - Record dept of:
     - Integer num
     - String name
     - Reference manager

3. **Access Paths:** empfile/1/0, emp-num/2/1, ... (but no deptfile)

4. **Integrity Constraints** (many), e.g.,

   \[ \forall x, y, z. \text{employee}(x, y, z) \rightarrow \exists w. \text{empfile}(w) \land \text{emp-num}(w, x), \]
   \[ \forall a, x. \text{empfile}(a) \land \text{emp-num}(a, x) \rightarrow \exists y, z. \text{employee}(x, y, z), \ldots \]
What can this do: navigating pointers

List all employee numbers and names ($\exists z.\text{employee}(x, y, z)$):

$$\exists a.\text{empfile}(a) \land \text{emp-num}(a, x) \land \text{emp-name}(a, y)$$
What can this do: navigating pointers

List all employee numbers and names ($\exists z. \text{employee}(x, y, z)$):

$$\exists a. \text{empfile}(a) \land \text{emp-num}(a, x) \land \text{emp-name}(a, y)$$

or, in C-like syntax:

```c
for a in empfile do
  x := a->num;
  y := a->name;
```
What can this do: navigating pointers

1. List all employee numbers and names ($\exists z.\text{employee}(x, y, z)$):

   $\exists a.\text{empfile}(a) \land \text{emp-num}(a, x) \land \text{emp-name}(a, y)$

2. List all department numbers with their manager names

   ($\exists z, u, v.\text{department}(x, z, u) \land \text{employee}(u, y, v)$):

   $\exists a.\text{empfile}(a) \land \text{emp-name}(a, y) \land \text{emp-dept}(a, d) \land \text{dept-num}(d, x) \land \text{dept-mgr}(d, e) \land \text{compare}(a, e)$
What can this do: navigating pointers

1. List all employee numbers and names ($\exists z.\text{employee}(x, y, z)$):
   $$\exists a.\text{empfile}(a) \land \text{emp-num}(a, x) \land \text{emp-name}(a, y)$$

2. List all department numbers with their manager names
   ($\exists z, u, v.\text{department}(x, z, u) \land \text{employee}(u, y, v)$):
   $$\exists a, d, e.\text{empfile}(a) \land \text{emp-dept}(a, d)$$
   $$\land \text{dept-num}(d, x) \land \text{dept-mgr}(d, e) \land \text{emp-name}(e, y)$$
   $$\Rightarrow \text{needs “departments have at least one employee”.$$}
What can this do: navigating pointers

1. List all employee numbers and names ($\exists z.\text{employee}(x, y, z)$):
   $$\exists a.\text{empfile}(a) \land \text{emp-num}(a, x) \land \text{emp-name}(a, y)$$

2. List all department numbers with their manager names
   ($\exists z, u, v.\text{department}(x, z, u) \land \text{employee}(u, y, v)$):
   $$\exists a, d, e.\text{empfile}(a) \land \text{emp-dept}(a, d)$$
   $$\land \text{dept-num}(d, x) \land \text{dept-mgr}(d, e) \land \text{emp-name}(e, y)$$
   $$\Rightarrow \text{needs “departments have at least one employee”}.$$

   $$\exists a, b, d.\text{empfile}(a) \land \text{emp-name}(a, y) \land \text{emp-dept}(a, d)$$
   $$\land \text{dept-num}(d, x) \land \text{dept-mgr}(d, b) \land \text{compare}(a, b)$$
   $$\Rightarrow \text{needs “managers work in their own departments”}.$$
What can this do: navigating pointers

1. List all employee numbers and names (\(\exists z.\)employee\((x, y, z)\)):
   \[
   \exists a.\text{empfile}(a) \land \text{emp-num}(a, x) \land \text{emp-name}(a, y)
   \]

2. List all department numbers with their manager names
   (\(\exists z, u, v.\)department\((x, z, u) \land \text{employee}(u, y, v)\)):
   \[
   \exists a, d, e.\text{empfile}(a) \land \text{emp-dept}(a, d) \\
   \land \text{dept-num}(d, x) \land \text{dept-mgr}(d, e) \land \text{emp-name}(e, y)
   \Rightarrow \text{needs "departments have at least one employee".}
   \]
   \[
   \ldots \text{needs } \text{duplicate elimination} \text{ during projection.}
   \]
   \[
   \exists a, b, d.\text{empfile}(a) \land \text{emp-name}(a, y) \land \text{emp-dept}(a, d) \\
   \land \text{dept-num}(d, x) \land \text{dept-mgr}(d, b) \land \text{compare}(a, b)
   \Rightarrow \text{needs "managers work in their own departments".}
   \]
   \[
   \ldots \text{NO } \text{duplicate elimination} \text{ during projection.}
   \]
What can it do: Hashing, Lists, et al.

Hash Index with (list-based) Separate Chaining

Hash Array  Separate Chaining Linked Lists  Dept Records

$\ldots$

$i$: $\bullet \rightarrow \bullet \rightarrow \bullet \rightarrow \bot$

$j$: $\bot$

$n$: $\bullet \rightarrow \bullet \rightarrow \bot$

$D1$

$D2$

$D3$
Hash Index on department’s name:

Access paths:

\[ S_A \supseteq \{ \text{hash/2/1, hasharraylookup/2/1, listscan/2/1} \} \].

Physical Constraints:

\[ \Sigma_{LP} \supseteq \{ \forall x, y.((\text{deptfile}(x) \land \text{dept-name}(x, y)) \rightarrow \exists z, w. (\text{hash}(y, z) \land \text{hasharraylookup}(z, w) \land \text{listscan}(w, x))), \\
\forall x, y. (\text{hash}(x, y) \rightarrow \exists z. \text{hasharraylookup}(y, z)), \\
\forall x, y. (\text{listscan}(x, y) \rightarrow \text{deptfile}(y)) \} \]
What can it do: Hashing, Lists, et al.

Hash Index on department’s name:

Access paths:
\[ S_A \supseteq \{ \text{hash/2/1, hasharraylookup/2/1, listscan/2/1} \} \].

Physical Constraints:
\[ \Sigma_{LP} \supseteq \{ \forall x, y.((\text{deptfile}(x) \land \text{dept-name}(x, y)) \rightarrow \exists z, w.(\text{hash}(y, z) \land \text{hasharraylookup}(z, w) \land \text{listscan}(w, x))), \forall x, y.(\text{hash}(x, y) \rightarrow \exists z.\text{hasharraylookup}(y, z)), \forall x, y.(\text{listscan}(x, y) \rightarrow \text{deptfile}(y)) \} \]

Query:
\[ \exists y, z.(\text{department}(x_1, p, y) \land \text{employee}(y, x_2, z))\{p\}. \]
What can this do: two-level store

The access path $\text{empfile}$ is refined by $\text{emppages}/1/0$ and $\text{emprecords}/2/1$:
- $\text{emppages}$ returns (sequentially) disk pages containing $\text{emp}$ records, and
- $\text{emprecords}$ given a disc page, returns $\text{emp}$ records in that page.

List all employees with the same name
$(\exists z, u, v, w, t. \text{employee}(x_1, z, u, v) \land \text{employee}(x_2, z, w, t))$:

$$
\exists y, z, w, v, p, q. \text{emppages}(p) \land \text{emppages}(q) \\
\land \text{emprecords}(p, y) \land \text{emp-num}(y, x_1) \land \text{emp-name}(y, w) \\
\land \text{emprecords}(q, z) \land \text{emp-num}(z, x_2) \land \text{emp-name}(z, v) \\
\land \text{compare}(w, v).
$$

$\Rightarrow$ this plan implements the block nested loops join algorithm.

...more examples in ...
User updates **only through logical schema**: 

⇒ supplying “delta” relations (sets of tuples)

- Two copies of the schema: $\Sigma^{old}$ and $\Sigma^{new}$;
- Delta relations: $R^+$ (insertions) and $R^-$ (deletions);
- Constraints:
  \[
  \forall \bar{x}. (R^{old}(\bar{x}) \lor R^+(\bar{x})) \equiv (R^{new}(\bar{x}) \lor R^-(\bar{x})),
  \forall \bar{x}. (R^+(\bar{x}) \land R^-(\bar{x})) \rightarrow \perp
  \]
User updates *only through logical schema*:
⇒ supplying “delta” relations (sets of tuples)

- Two copies of the schema: \( \Sigma^{\text{old}} \) and \( \Sigma^{\text{new}} \);
- Delta relations: \( R^+ \) (insertions) and \( R^- \) (deletions);
- Constraints:
  \[
  \forall \bar{x} (R^{\text{old}}(\bar{x}) \lor R^+ (\bar{x})) \equiv (R^{\text{new}}(\bar{x}) \lor R^-(\bar{x})), \\
  \forall \bar{x} (R^+(\bar{x}) \land R^-(\bar{x})) \rightarrow \bot
  \]

Update turned into *definability* question

Is \( A^{\text{new}} \) (or \( A^+, A^- \)) definable in terms of \( A_i^{\text{old}} \in S_A^{\text{old}} \) (old access paths) and \( U_j^+, U_j^- \) (user updates) for every access path \( A \in S_A \)?
What can this do: updates

**User updates only through logical schema:**
⇒ supplying “delta” relations (sets of tuples)

- Two copies of the schema: \( \Sigma^{\text{old}} \) and \( \Sigma^{\text{new}} \);
- Delta relations: \( R^+ \) (insertions) and \( R^- \) (deletions);
- Constraints:
  \[
  \forall \bar{x}. (R^{\text{old}}(\bar{x}) \lor R^+(\bar{x})) \equiv (R^{\text{new}}(\bar{x}) \lor R^-(\bar{x})),
  \forall \bar{x}. (R^+(\bar{x}) \land R^-(\bar{x})) \rightarrow \bot
  \]

**Update turned into definability question**

Is \( A^{\text{new}} \) (or \( A^+, A^- \)) definable in terms of \( A_i^{\text{old}} \in S_A^{\text{old}} \) (old access paths) and \( U_j^+, U_j^- \) (user updates) for every access path \( A \in S_A \)?

**How do we get “anonymous values”?**

*Constant Complement* tables (access paths that do `malloc` and such);
cyclic dependencies between *anonymous values* broken via *reification*. 
GRAND UNIFIED APPROACH TO QUERY COMPILATION

PART II: HOW DOES IT WORK?
The Plan

Definability and Rewriting

Queries
- range-restricted FOL over $S_L$ definable w.r.t. $\Sigma$ and $S_A$

Ontology/Schema
- range-restricted FOL

Data
- CWA (complete information for $S_A$ symbols)

(Logical Schema)

(SA ⊆ SP)

(rewriting)

(Physical Schema)
Idea #1: Plans as Formulas

Represent *query plans* as (annotated) range-restricted formulas $\psi$ over $S_A$:

- atomic formula $\leadsto$ access path (*get-first–get-next iterator*)
- conjunction $\leadsto$ nested loops join
- existential quantifier $\leadsto$ projection (annotated w/duplicate info)
- disjunction $\leadsto$ concatenation
- negation $\leadsto$ simple complement
Query Plans via Interpolation

**Idea #1: Plans as Formulas**

Represent *query plans* as (annotated) range-restricted formulas $\psi$ over $S_A$:

- **atomic formula** $\mapsto$ access path (get-first–get-next iterator)
- **conjunction** $\mapsto$ nested loops join
- **existential quantifier** $\mapsto$ projection (annotated w/duplicate info)
- **disjunction** $\mapsto$ concatenation
- **negation** $\mapsto$ simple complement

**Non-logical (but necessary) Add-ons**

1. **Non-logical properties/operators**
   - binding patterns
   - duplication of data and duplicate-preserving/elminating projections
   - sortedness of data (with respect to the *iterator semantics*) and sorting

2. **Cost model(s)**
Idea #1 (cont):

Represent *physical design* as *access paths* ($S_A$) and *constraints* ($\Sigma$).

Represent *query plans* as (annotated) range-restricted formulas $\psi$ over $S_A$.

\[ \Rightarrow \text{reduces correctness of } \psi \text{ to logical implication } \Sigma \models \varphi \leftrightarrow \psi \]
Query Plans via Interpolation

Idea #1 (cont):

Represent *physical design* as *access paths* ($S_A$) and *constraints* ($\Sigma$).
Represent *query plans* as (annotated) range-restricted formulas $\psi$ over $S_A$.

⇒ reduces correctness of $\psi$ to logical implication $\Sigma \models \varphi \leftrightarrow \psi$

IDEA #2:

Use *interpolation* to search for $\psi$:

extract an *interpolant* $\psi$ from a (TABLEAU) proof of $\Sigma \cup \Sigma^* \models \varphi \rightarrow \varphi^*$
Query Plans via Interpolation

Idea #1 (cont):
Represent physical design as *access paths* ($S_A$) and *constraints* ($\Sigma$).
Represent query plans as (annotated) range-restricted formulas $\psi$ over $S_A$.

$\implies$ reduces correctness of $\psi$ to logical implication $\Sigma \models \varphi \iff \psi$

IDEA #2:
Use *interpolation* to search for $\psi$:
extract an *interpolant* $\psi$ from a (TABLEAU) proof of $\Sigma \cup \Sigma^* \models \varphi \rightarrow \varphi^*$

$\implies$ Beth definability of $\varphi$ over $\Sigma$ and $S_A$ resolves the existence of $\psi$. 
Issues with TABLEAU

Dealing with the *subformula property* of Tableau

⇒ analytic tableau *explores* formulas *structurally*
⇒ (to large degree) the structure of interpolant depends on where access paths are present in queries/constraints.

Factoring *logical reasoning from plan enumeration*

⇒ backtracking tableau to get alternative plans: too slow, too few plans
Issues with TABLEAU

Dealing with the subformula property of Tableau

⇒ analytic tableau explores formulas structurally
⇒ (to large degree) the structure of interpolant depends on where access paths are present in queries/constraints.

IDEA #3:
Separate general constraints from physical rules in the formulation of the definability question (and the subsequent interpolant extraction):

\[ \Sigma^L \cup \Sigma^R \cup \Sigma^{LR} \models \varphi^L \rightarrow \varphi^R \text{ where } \Sigma^{LR} = \{ \forall \bar{x}. P^L \leftrightarrow P \leftrightarrow P^R \mid P \in S_A \} \]

Factoring logical reasoning from plan enumeration
⇒ backtracking tableau to get alternative plans: too slow, too few plans

IDEA #4:
Define conditional tableau exploration (using general constraints) and separate it from plan generation (using physical rules)
Conditional Formulæ and Tableau

Conditional Formulæ

$\varphi[C]$ where $C$ is a set of (ground) atoms over $S_A$

$\varphi$ only exists if all atoms in $C$ are “used” in a plan tableau.

Absorbed Range-restricted Formulæ: ANF

$$Q ::= R(\bar{x}) \mid \bot \mid Q \land Q \mid Q \lor Q \mid \forall \bar{x}.R(\bar{x}) \rightarrow Q,$$

...and all $\exists$’s are Skolemized.

Conditional Tableau Rules for ANF

$$\frac{S \cup \{\varphi[C], \psi[C]\}}{(\varphi \land \psi)[C] \in S} \quad \text{(conj)}$$

$$\frac{S \cup \{(\varphi[\bar{t}/\bar{x}])[C \cup D]\}}{\{R(\bar{t})[C], (\forall \bar{x}.R(\bar{x}) \rightarrow \varphi)[D]\} \subseteq S} \quad \text{(abs)}$$

$$\frac{S \cup \{\varphi[C]\}}{(\varphi \lor \psi)[C] \in S} \quad \text{(disj)}$$

$$\frac{S \cup \{\psi[C]\}}{R(\bar{x}) \in S_A \quad \text{(phys)}}$$
Goal: a *closed tableau* of the form

- $T^L$ and $T^R$ are *logical left- and right-tableau* (only logical constraints $\Sigma^L$ and $\Sigma^R$, built 1st)
- $T^P$ is a *physical tableau* (only physical rules $\Sigma^{LR}$, built 2nd)
Conditional Tableau: Logical Parts

Conditional Tableau for \((Q, \Sigma, S_A)\)

Proof trees \((T^L, T^R)\):
- \(T^L\) for \(\Sigma^L \cup \{Q^L(\bar{a})\}\) over \(\{P^L \mid P \in S_A\}\)
- \(T^R\) for \(\Sigma^R \cup \{Q^R(\bar{a}) \rightarrow \bot\}\) over \(\{P^R \mid P \in S_A\}\)

Closing Set(s)

We call \(S\) of atoms over \(S_A\) a closing set for \(T\) if, for every branch,

1. there is an atom \(R(\bar{t})[C]\), \(R(\bar{t}) \not\in C\), such that \(C \cup \{\neg R(\bar{t})\} \subseteq S\), or
2. there is \(\bot[C]\) such that \(C \subseteq S\).

\[\Rightarrow\] there are many different minimal closing sets for \(T\).

Observation

For arbitrary closing set \(S\), an interpolant for \(T^L(T^R)\) is \(\bot(\top)\).
Plan Enumeration

Physical Tableau $T^P$

<table>
<thead>
<tr>
<th>$P$</th>
<th>$L_P$</th>
<th>$R_P$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$R(t)$</td>
<td>${{\neg R^L(t)}}$</td>
<td>${{R^R(t)}}$</td>
</tr>
<tr>
<td>$P_1 \land P_2$</td>
<td>$L_{P_1} \cup L_{P_2}$</td>
<td>${S_1 \cup S_2 \mid S_1 \in R_{P_1}, S_2 \in R_{P_2}}$</td>
</tr>
<tr>
<td>$P_1 \lor P_2$</td>
<td>${S_1 \cup S_2 \mid S_1 \in L_{P_1}, S_2 \in L_{P_2}}$</td>
<td>$R_{P_1} \cup R_{P_2}$</td>
</tr>
<tr>
<td>$\neg P_1$</td>
<td>${{L^L(t) \mid L^R(t) \in S} \mid S \in R_{P_1}}$</td>
<td>${{L^R(t) \mid L^L(t) \in S} \mid S \in L_{P_1}}$</td>
</tr>
<tr>
<td>$\exists x.P_1$</td>
<td>$L_{P_1}[t/x]$</td>
<td>$R_{P_1}[t/x]$</td>
</tr>
</tbody>
</table>

Observation

For a range-restricted formula $P$ over $S_A$ there is an analytic tableau $T^P$ that uses only formulæ in $\Sigma^{LR} \cup \{\forall x.\text{true}^R(x)\}$ such that:

1. Open branches of $T^P$ correspond to sets $S \in L_P$ (left branch) or $S \in R_P$ (right branch); and

2. The interpolant extracted from this tableau is logically equivalent to $P$, assuming that the closure of each left branch interpolates to $\bot$ and for right branch to $\top$. 
Logical & Physical Combined, Controlling the Search

Basic Strategy

1. build \((T^L, T^R)\) for \((Q, \Sigma, S_A)\) to a certain depth,
2. build \(T^P\) and test if each element in \(L_P(R_P)\) closes \(T^L(T^R)\).

If so, \(T^P[T^L, T^R]\) is closed tableau yielding an interpolant equivalent to \(P\);
(\ldots otherwise extend depth in step 1 and repeat.)
Logical & Physical Combined, Controlling the Search

**Basic Strategy**

1. build \((T^L, T^R)\) for \((Q, \Sigma, S_A)\) to a *certain depth*,
2. build \(T^P\) and test if each element in \(L_P(R_P)\) closes \(T^L(T^R)\).

   if so, \(T^P[T^L, T^R]\) is closed tableau yielding an interpolant equivalent to \(P\);

   (\ldots otherwise extend depth in step 1 and repeat.)

In step 2 we can “test” many \(P\)s (plan enumeration), but how do we know which ones to try? while building these bottom-up?

**Controlling the Search**

- only use the (phys) rule in \(T^L(T^R)\) for \(R(\tilde{t})\) that appears in \(T^R(T^L)\),
- only consider *fragments* that help closing \((T^L, T^R)\)
  \(\Rightarrow \) this is determined using the minimal closing sets for \((T^L, T^R)\).

\(\ldots\) combine with *search* (among \(P\)’s) with respect to a *cost model*. 
Postprocessing

Duplicate Elimination Elimination

\[ Q[\{R(x_1, \ldots, x_k)\}] \leftrightarrow Q[R(x_1, \ldots, x_k)] \]
\[ Q[\{Q_1 \land Q_2\}] \leftrightarrow Q[\{Q_1\} \land \{Q_2\}] \]
\[ Q[\{\neg Q_1\}] \leftrightarrow Q[\neg Q_1] \]
\[ Q[\neg\{Q_1\}] \leftrightarrow Q[\neg Q_1] \]
\[ Q[\{Q_1 \lor Q_2\}] \leftrightarrow Q[\{Q_1\} \lor \{Q_2\}] \quad \text{if } \Sigma \cup \{Q[]\} \models Q_1 \land Q_2 \rightarrow \bot \]
\[ Q[\exists x. Q_1] \leftrightarrow Q[\exists x.\{Q_1\}] \quad \text{if } \Sigma \cup \{Q[]\} \land Q_1[y_1/x] \land Q_1[y_2/x] \models y_1 \approx y_2 \]

\[ \mathcal{CFD}_{nc} \] logic to approximate \( \Sigma \) and to reason about FDs/disjointness in PTIME.
Postprocessing

Duplicate Elimination

\[ Q[\{R(x_1, \ldots, x_k)\}] \leftrightarrow Q[R(x_1, \ldots, x_k)] \]
\[ Q[\{Q_1 \land Q_2\}] \leftrightarrow Q[\{Q_1\} \land \{Q_2\}] \]
\[ Q[\{\neg Q_1\}] \leftrightarrow Q[\neg Q_1] \]
\[ Q[\neg\{Q_1\}] \leftrightarrow Q[\neg Q_1] \]
\[ Q[\{Q_1 \lor Q_2\}] \leftrightarrow Q[\{Q_1\} \lor \{Q_2\}] \]
\[ Q[\exists x. Q_1] \leftrightarrow Q[\exists x. \{Q_1\}] \]
\[ \text{if } \Sigma \cup \{Q[\cdot]\} \models Q_1 \land Q_2 \rightarrow \bot \]
\[ \text{if } \Sigma \cup \{Q[\cdot]\} \land Q_1[y_1/x] \land Q_1[y_2/x] \models y_1 \approx y_2 \]

$\mathcal{CFD}_{nc}$ logic to approximate $\Sigma$ and to reason about FDs/disjointness in PTIME.

\[ \Rightarrow \text{ also CUT insertion (similar—see } \text{)} \text{.} \]
Summary of the Approach

1. **FO (DLFDE) tableau based interpolation algorithm**
   - Enumeration of plans factored from reasoning
   - Range-restricted queries and constraints $\rightarrow$ ground terms only
   - Extra-logical binding patterns and cost model

2. Post processing (using $\mathcal{CFD}$ approximation)
   - Duplicate elimination elimination
   - Cut insertion

3. Run time
   - Library of common data structures + schema constraints
   - Or an interface to a legacy system
   - Finger data structures to simulate merge joins et al.
Open Issues

1. Dealing with ordered data? (merge-joins etc.: we have a partial solution)
2. Decidable schema languages (decidable interpolation problem)?
3. More powerful schema languages (inductive types, etc.)?
4. Beyond FO Queries/Views (e.g., count/sum aggregates)?
5. Coding extra-logical bits (e.g., binding patterns, postprocessing, etc.) in the schema itself?
6. Standard Designs (a plan can always be found as in SQL)?
7. Explanation(s) of non-definability?
8. Fine(r)-grained updates?
9. . . .

... and, as always, performance, performance, performance!
Message from our Sponsors

Database Group at the University of Waterloo

- 7 professors, affiliated faculty, postdocs, 30+ graduate students, ...
- wide range of research interests
  - Advanced query processing/Knowledge representation
  - System aspects of database systems and Distributed data management
  - Data quality/Managing uncertain data/Data mining
  - New(-ish) domains (text, streaming, graph data/RDF, OLAP)
- research sponsored by governments, and local/global companies
  - NSERC/CFI/OIT and Google, IBM, SAP, OpenText, ...
- part of a School of CS with 75+ professors, 300+ grad students, etc.
  - AI (incl ML), Algorithms&Data Structures, PL, Theory, Systems ...

Cheriton School of Computer Science has been ranked #18 in CS by the world by US News and World Report (#1 in Canada).

... and we are always looking for good graduate students (MMath/PhD)
  ⇒ comes with full support over multiple years