Query Compilation
(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
GRAND UNIFIED APPROACH TO QUERY COMPILATION

PART II: HOW DOES IT WORK?
The Plan

Definability and Rewriting

<table>
<thead>
<tr>
<th>Queries</th>
<th>range-restricted FOL over $S_L$ \textit{definable} w.r.t. $\Sigma$ and $S_A$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ontology/Schema</td>
<td>range-restricted FOL</td>
</tr>
<tr>
<td>Data</td>
<td>CWA (complete information \textit{for} $S_A$ symbols)</td>
</tr>
</tbody>
</table>

$$\Sigma_L \leftarrow \Sigma_{LP} \leftarrow (\text{rewriting}) \leftarrow \Sigma_P$$

(Logical Schema)

$S_L \subseteq S_{LP}$

(Physical Schema)
## Plans as Formulas

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

<table>
<thead>
<tr>
<th>Atomic formula</th>
<th>$\mapsto$</th>
<th>Access path (<em>get-first-get-next iterator</em>)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conjunction</td>
<td>$\mapsto$</td>
<td>Nested loops join</td>
</tr>
<tr>
<td>Existential quantifier</td>
<td>$\mapsto$</td>
<td>Projection (annotated w/duplicate info)</td>
</tr>
<tr>
<td>Disjunction</td>
<td>$\mapsto$</td>
<td>Concatenation</td>
</tr>
<tr>
<td>Negation</td>
<td>$\mapsto$</td>
<td>Simple complement</td>
</tr>
</tbody>
</table>

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

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

#### Non-logical properties/operators
- Binding patterns
- Duplication of data and duplicate-preserving/eliminating projections
- Sortedness of data (with respect to the iterator semantics) and sorting

#### Cost model
## Query Plans via Rewriting

### Plans as Formulas

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

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Representation</th>
</tr>
</thead>
<tbody>
<tr>
<td>atomic formula</td>
<td>access path (get-first-get-next iterator)</td>
</tr>
<tr>
<td>conjunction</td>
<td>nested loops join</td>
</tr>
<tr>
<td>existential quantifier</td>
<td>projection (annotated w/duplicate info)</td>
</tr>
<tr>
<td>disjunction</td>
<td>concatenation</td>
</tr>
<tr>
<td>negation</td>
<td>simple complement</td>
</tr>
</tbody>
</table>

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

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

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

- Cost model
### Query Plans via Rewriting

#### Plans as Formulas

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

<table>
<thead>
<tr>
<th>Formula Type</th>
<th>Representation</th>
</tr>
</thead>
<tbody>
<tr>
<td>atomic formula</td>
<td>access path (get-first–get-next iterator)</td>
</tr>
<tr>
<td>conjunction</td>
<td>nested loops join</td>
</tr>
<tr>
<td>existential quantifier</td>
<td>projection (annotated w/duplicate info)</td>
</tr>
<tr>
<td>disjunction</td>
<td>concatenation</td>
</tr>
<tr>
<td>negation</td>
<td>simple complement</td>
</tr>
</tbody>
</table>

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

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

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

2. **Cost model**
Chase and Backchase

IDEA #1 (Database Theory, CQ/UCQ)

Inference(s): \( Q, (\forall \bar{x}. Q_1 \rightarrow Q_2) \vdash Q \cup Q_2\theta \) when \( Q_1\theta \subseteq Q \)

1. (chase): expand \( Q \) “maximally” using constraints
2. (plan) choose a “plan” \( P \) from the expansion
3. (backchase): expand \( P \) using constraints to contain \( Q \) (or fail)

⇒ can be extended to UCQ+denial constraints

Views:

- \( V_1(x, y) \leftrightarrow \exists t, u, v. R(t, x) \land R(t, u) \land R(u, y) \)
- \( V_2(x, y) \leftrightarrow \exists u. R(x, u) \land R(u, y) \)
- \( V_3(x, y) \leftrightarrow \exists t, u. R(x, t) \land R(t, u) \land R(u, y) \)

Query:

\( Q(x, y) \leftrightarrow \exists t, u, v. R(t, x) \land R(t, u) \land R(u, v) \land R(v, y) \)

No SOLUTION (using C&B); rewritings

\( \exists z. V_1(x, z) \land \forall v. V_2(v, z) \rightarrow V_3(v, y) \) and
\( \exists z. V_3(z, y) \land \forall v. V_2(v, z) \rightarrow V_1(x, v) \)
Chase and Backchase

IDEA #1 (Database Theory, CQ/UCQ)

Inference(s): $Q, (\forall \bar{x}.Q_1 \rightarrow Q_2) \vdash Q \cup Q_2\theta$ when $Q_1\theta \subseteq Q$

1. (chase): expand $Q$ “maximally” using constraints
2. (plan) choose a “plan” $P$ from the expansion
3. (backchase): expand $P$ using constraints to contain $Q$ (or fail)

⇒ can be extended to UCQ+denial constraints

Views:
- $V_1(x, y) \leftrightarrow \exists t, u, v. R(t, x) \land R(t, u) \land R(u, y)$
- $V_2(x, y) \leftrightarrow \exists u. R(x, u) \land R(u, y)$
- $V_3(x, y) \leftrightarrow \exists t, u. R(x, t) \land R(t, u) \land R(u, y)$

Query:
- $Q(x, y) \leftrightarrow \exists t, u, v. R(t, x) \land R(t, u) \land R(u, v) \land R(v, y)$

No SOLUTION (using C&B); rewritings

$\exists z. V_1(x, z) \land \forall v. V_2(v, z) \rightarrow V_3(v, y)$ and
$\exists z. V_3(z, y) \land \forall v. V_2(v, z) \rightarrow V_1(x, v)$
Chase and Backchase

**IDEA #1 (Database Theory, CQ/UCQ)**

**Inference(s):** \( Q, (\forall \bar{x}. Q_1 \rightarrow Q_2) \vdash Q \cup Q_2 \theta \) when \( Q_1 \theta \subseteq Q \)

1. **(chase):** expand \( Q \) “maximally” using constraints
2. **(plan) choose a “plan”** \( P \) from the expansion
3. **(backchase):** expand \( P \) using constraints to contain \( Q \) (or fail)

\( \Rightarrow \) can be extended to UCQ+denial constraints

**Views:**

\[
V_1(x, y) \leftrightarrow \exists t, u, v. R(t, x) \land R(t, u) \land R(u, y) \\
V_2(x, y) \leftrightarrow \exists u. R(x, u) \land R(u, y) \\
V_3(x, y) \leftrightarrow \exists t, u. R(x, t) \land R(t, u) \land R(u, y)
\]

**Query:**

\[
Q(x, y) \leftrightarrow \exists t, u, v. R(t, x) \land R(t, u) \land R(u, v) \land R(v, y)
\]

No SOLUTION (using C&B); rewritings

\[
\exists z. V_1(x, z) \land \forall v. V_2(v, z) \rightarrow V_3(v, y) \text{ and} \\
\exists z. V_3(z, y) \land \forall v. V_2(v, z) \rightarrow V_1(x, v)
\]
Chase and Backchase

IDEA #1 (Database Theory, CQ/UCQ)

Inference(s): \( Q, (\forall \bar{x}. Q_1 \rightarrow Q_2) \vdash Q \cup Q_2 \theta \) when \( Q_1 \theta \subseteq Q \)

1. (chase): expand \( Q \) “maximally” using constraints
2. (plan) choose a “plan” \( P \) from the expansion
3. (backchase): expand \( P \) using constraints to contain \( Q \) (or fail)

\( \Rightarrow \) can be extended to UCQ+denial constraints

Views:
\[
\begin{align*}
V_1(x, y) & \leftrightarrow \exists t, u, v. R(t, x) \land R(t, u) \land R(u, y) \\
V_2(x, y) & \leftrightarrow \exists u. R(x, u) \land R(u, y) \\
V_3(x, y) & \leftrightarrow \exists t, u. R(x, t) \land R(t, u) \land R(u, y)
\end{align*}
\]

Query:
\[
Q(x, y) \leftrightarrow \exists t, u, v. R(t, x) \land R(t, u) \land R(u, v) \land R(v, y)
\]

No SOLUTION (using C&B): rewritings

\[
\begin{align*}
\exists z. V_1(x, z) \land \forall v. V_2(v, z) \rightarrow V_3(v, y) \quad \text{and} \\
\exists z. V_3(z, y) \land \forall v. V_2(v, z) \rightarrow V_1(x, v)
\end{align*}
\]
IDEA #1 (Database Theory, CQ/UCQ)

Inference(s): $Q, (\forall \bar{x}. Q_1 \rightarrow Q_2) \vdash Q \cup Q_2 \theta$ when $Q_1\theta \subseteq Q$

1. (chase): expand $Q$ “maximally” using constraints
2. (plan) choose a “plan” $P$ from the expansion
3. (backchase): expand $P$ using constraints to contain $Q$ (or fail)

$\Rightarrow$ can be extended to UCQ+denial constraints

Views: $V_1(x, y) \iff \exists t, u, v. R(t, x) \land R(t, u) \land R(u, y)$
$V_2(x, y) \iff \exists u. R(x, u) \land R(u, y)$
$V_3(x, y) \iff \exists t, u. R(x, t) \land R(t, u) \land R(u, y)$

Query: $Q(x, y) \iff \exists t, u, v. R(t, x) \land R(t, u) \land R(u, v) \land R(v, y)$

No SOLUTION (using C&B); rewritings

$\exists z. V_1(x, z) \land \forall v. V_2(v, z) \rightarrow V_3(v, y)$ and
$\exists z. V_3(z, y) \land \forall v. V_2(v, z) \rightarrow V_1(x, v)$
IDEA #2: What Queries do we allow?

We only allow queries that have \textit{the same answer} in every model of $\Sigma$ ... for a fixed signature $S_A$ (i.e., where the actual data is).

How do we test for this?

$\varphi$ is Beth definable [Beth'56] if

$$\Sigma \cup \Sigma' \models \varphi \rightarrow \varphi'$$

where $\Sigma'$ ($\varphi'$) is $\Sigma$ ($\varphi$) in which symbols \textit{NOT in $S_A$} are primed, respectively.
IDEA #2: What Queries do we allow?

We only allow queries that have \textit{the same answer} in every model of $\Sigma$ ... for a fixed signature $S_A$ (i.e., where the actual data is).

How do we test for this?

$\varphi$ is \textit{Beth definable} [Beth’56] if

$$\Sigma \cup \Sigma' \models \varphi \rightarrow \varphi'$$

where $\Sigma'$ ($\varphi'$) is $\Sigma$ ($\varphi$) in which symbols \textit{NOT in} $S_A$ are primed, respectively.
Sequent Calculus: LK

Identity Rules:

\[ \Gamma, \varphi \vdash \varphi, \Delta \]  (Axiom)

\[ \Gamma \vdash \varphi, \Delta \quad \Gamma, \varphi \vdash \Delta \]  (Cut)

Logical Rules:

\[ \Gamma \vdash \varphi, \Delta \]  \[ \Gamma, \varphi \vdash \Delta \]  (\neg L)

\[ \Gamma \vdash (\neg \varphi), \Delta \]  (\neg R)

\[ \Gamma, \varphi \vdash \Delta \quad \Gamma, \psi \vdash \Delta \]  \[ \Gamma \vdash (\varphi \lor \psi), \Delta \]  (∨ L)

\[ \Gamma \vdash \varphi, \psi, \Delta \]  (∨ R)

\[ \Gamma, \varphi, \psi \vdash \Delta \]  \[ \Gamma \vdash (\varphi \land \psi), \Delta \]  (∧ L)

\[ \Gamma \vdash \varphi, \Delta \quad \Gamma \vdash \psi, \Delta \]  (∧ R)

\[ \Gamma, \varphi, \psi \vdash \Delta \]  \[ \Gamma \vdash (\varphi \land \psi), \Delta \]  (∧ L)
**Theorem (Hauptsatz)**

*For every proof of a sequent $\Gamma \vdash \Delta$ in $LK$ there is also a proof of the same sequent in $LK - \{\text{Cut}\}$.***
Sequent Calculus (for NNF)

Identity Rules:

\[
\Gamma, \varphi \vdash \varphi, \Delta \quad (Axiom \ LR)
\]

\[
\Gamma, \varphi, \neg \varphi \vdash \Delta \quad (Axiom \ RR)
\]

\[
\Gamma \vdash \varphi, \neg \varphi, \Delta \quad (Axiom \ LL)
\]

Logical Rules:

\[
\Gamma, \varphi \vdash \Delta \quad \Gamma, \psi \vdash \Delta \quad \Gamma, (\varphi \lor \psi) \vdash \Delta \quad (\lor L)
\]

\[
\Gamma \vdash \varphi, \psi, \Delta \quad \Gamma \vdash (\varphi \lor \psi), \Delta \quad (\lor R)
\]

\[
\Gamma, \varphi, \psi \vdash \Delta \quad \Gamma, (\varphi \land \psi) \vdash \Delta \quad (\land L)
\]

\[
\Gamma \vdash \varphi, \Delta \quad \Gamma \vdash \psi, \Delta \quad \Gamma \vdash (\varphi \land \psi), \Delta \quad (\land R)
\]
Interpolation

How do we find $\psi$?

If $\Sigma \cup \Sigma' \models \varphi \rightarrow \varphi'$ then there is $\psi$ s.t. $\Sigma \cup \Sigma' \models \varphi \rightarrow \psi \rightarrow \varphi'$ with $L(\psi) \subseteq L(S_A)$.

... $\psi$ is called the *Craig Interpolant* [Craig’57].

... we extract an *interpolant* $\psi$ from a (LK) proof of $\Sigma \cup \Sigma' \models \varphi \rightarrow \varphi'$
Identity Rules:

\[ \Gamma, \varphi \vdash \varphi, \Delta \sim \varphi \]

\[ \Gamma, \varphi, \neg \varphi \vdash \Delta \sim \bot \]

\[ \Gamma \vdash \varphi, \neg \varphi, \Delta \sim \top \]

Logical Rules:

\[ \Gamma, \varphi \vdash \Delta \sim \alpha \quad \Gamma, \psi \vdash \Delta \sim \beta \]

\[ \Gamma, (\varphi \lor \psi) \vdash \Delta \sim \alpha \lor \beta \]

\[ \Gamma \vdash \varphi, \psi, \Delta \sim \alpha \]

\[ \Gamma \vdash (\varphi \lor \psi), \Delta \sim \alpha \]

\[ \Gamma, \varphi, \psi \vdash \Delta \sim \alpha \]

\[ \Gamma, (\varphi \land \psi) \vdash \Delta \sim \alpha \]

\[ \Gamma \vdash \varphi, \Delta \sim \alpha \quad \Gamma \vdash \psi, \Delta \sim \beta \]

\[ \Gamma \vdash (\varphi \land \psi), \Delta \sim \alpha \land \beta \]
LK and Theories

\[ \Sigma \cup \Sigma' \models \varphi \rightarrow \varphi' \iff (\bigwedge \Sigma) \wedge (\bigwedge \Sigma') \models \varphi \rightarrow \varphi' \]
\[ \iff \models (\bigwedge \Sigma) \rightarrow ((\bigwedge \Sigma') \rightarrow (\varphi \rightarrow \varphi')) \]
\[ \iff \models (\bigwedge \Sigma) \rightarrow (\varphi \rightarrow ((\bigwedge \Sigma') \rightarrow \varphi')) \]
\[ \iff (\bigwedge \Sigma) \wedge \varphi \models (\bigwedge \Sigma') \rightarrow \varphi' \]
\[ \iff (\bigwedge \Sigma) \wedge \varphi \models (\bigvee \neg \Sigma) \lor \varphi' \]
\[ \iff \Sigma, \varphi \vdash (\neg \Sigma'), \varphi' \quad \text{(due to soundness/completeness)} \]

Not convenient: needs both \( \Sigma \) and negated \( \Sigma' \)!

we use ANALYTIC TABLEAU: a refutation variant of LK to show

\[ \Sigma, \Sigma', \varphi, \neg \varphi' \vdash \bot \quad \text{a.k.a. is inconsistent} \]

\( \rightarrow \) need to tag left (L)/right(R) formulae to simulate sequents (for interpolation)!

David Toman (et al.)
LK and Theories

\[ \Sigma \cup \Sigma' \models \varphi \to \varphi' \iff (\land \Sigma) \land (\land \Sigma)' \models \varphi \to \varphi' \]
\[ \iff \models (\land \Sigma) \to ((\land \Sigma)' \to (\varphi \to \varphi')) \]
\[ \iff \models (\land \Sigma) \to (\varphi \to ((\land \Sigma)' \to \varphi')) \]
\[ \iff (\land \Sigma) \land \varphi \models (\land \Sigma)' \to \varphi' \]
\[ \iff (\land \Sigma) \land \varphi \models (\lor \neg \Sigma)' \lor \varphi' \]
\[ \iff \Sigma, \varphi \vdash (\neg \Sigma'), \varphi' \quad \text{(due to soundness/completeness)} \]

Not convenient: needs both \( \Sigma \) and negated \( \Sigma' \)!

we use Analytic Tableau: a refutation variant of LK to show

\[ \Sigma, \Sigma', \varphi, \neg \varphi' \vdash \bot \quad \text{a.k.a. is inconsistent} \]

\[ \Rightarrow \text{need to tag left (L)/right(R) formulae to simulate sequents} \]
\[ (\text{for interpolation})! \]
LK and Theories

\[ \Sigma \cup \Sigma' \models \varphi \rightarrow \varphi' \iff (\bigwedge \Sigma) \land (\bigwedge \Sigma') \models \varphi \rightarrow \varphi' \]
\[ \iff \models (\bigwedge \Sigma) \rightarrow ((\bigwedge \Sigma') \rightarrow (\varphi \rightarrow \varphi')) \]
\[ \iff \models (\bigwedge \Sigma) \rightarrow (\varphi \rightarrow ((\bigwedge \Sigma') \rightarrow \varphi')) \]
\[ \iff (\bigwedge \Sigma) \land \varphi \models (\bigwedge \Sigma') \rightarrow \varphi' \]
\[ \iff (\bigwedge \Sigma) \land \varphi \models (\bigvee \neg \Sigma') \lor \varphi' \]
\[ \iff \Sigma, \varphi \vdash (\neg \Sigma'), \varphi' \quad \text{(due to soundness/completeness)} \]

Not convenient: needs both \( \Sigma \) and negated \( \Sigma' \)!

We use \textsc{Analytic Tableau}: a refutation variant of LK to show

\[ \Sigma, \Sigma', \varphi, \neg \varphi' \vdash \bot \quad \text{a.k.a. is inconsistent} \]

\[ \Rightarrow \text{need to \textit{tag} left (L)/right(R) formulae to simulate sequents (for interpolation)}! \]
Tableau and Interpolant Extraction (by example)

- an interpolant $S \xrightarrow{\text{int}} \psi$; invariant $(\land S^L) \rightarrow \psi$ and $\psi \rightarrow (\neg \land S^R)$
  
  where $S^L$ and $S^R$ are the left/right subsets of $S$;

- tableau rules (sample):
  
  **LR clash**
  
  $S \cup \{ R^L, \neg R^R \} \xrightarrow{\text{int}} R$, where $R \in S_A$

  as $(\land S^L \land R^L) \rightarrow R$ and $R \rightarrow (R^R \lor \neg \land S^R)$;

  **L-conjunction**
  
  $S \cup \{ \alpha^L, \beta^L \} \xrightarrow{\text{int}} \delta$

  $S \cup \{ (\alpha \land \beta)^L \} \xrightarrow{\text{int}} \delta$

  as $(\land S^L \land \alpha^L \land \beta^L) \rightarrow \delta$ implies $(\land S^L \land (\alpha \land \beta)^L) \rightarrow \delta$;

  **R-Disjunction**
  
  $S \cup \{ \alpha^R \} \xrightarrow{\text{int}} \delta_\alpha$

  $S \cup \{ \beta^R \} \xrightarrow{\text{int}} \delta_\beta$

  $S \cup \{ (\alpha \lor \beta)^R \} \xrightarrow{\text{int}} \delta_\alpha \land \delta_\beta$

  as $(\land S^L \rightarrow \delta_\alpha, \delta_\alpha \rightarrow \neg (\alpha^R \land \land S^R))$ and $(\land S^L \rightarrow \delta_\beta, \delta_\beta \rightarrow \neg (\beta^R \land \land S^R))$

  implies $(\land S^L) \rightarrow \delta_\alpha \land \delta_\beta, \delta_\alpha \land \delta_\beta \rightarrow \neg ((\alpha \lor \beta)^R \land \land S^R)$;

  etc. (see [Fitting] for details)
Tableau and Interpolant Extraction (by example)

- an interpolant $S \xrightarrow{\text{int}} \psi$; invariant $(\bigwedge S^L) \rightarrow \psi$ and $\psi \rightarrow (\neg \bigwedge S^R)$
  where $S^L$ and $S^R$ are the left/right subsets of $S$;

- tableau rules (sample):
  
  - LR clash $S \cup \{ R^L, \neg R^R \} \xrightarrow{\text{int}} R$, where $R \in S_A$

  as $(\bigwedge S^L \land R^L) \rightarrow R$ and $R \rightarrow (R^R \lor \neg \bigwedge S^R)$;

- L-conjunction $S \cup \{ \alpha^L, \beta^L \} \xrightarrow{\text{int}} \delta$

  as $(\bigwedge S^L \land \alpha^L \land \beta^L) \rightarrow \delta$ implies $(\bigwedge S^L \land (\alpha \land \beta)^L) \rightarrow \delta$;

- R-Disjunction $S \cup \{ \alpha^R \} \xrightarrow{\text{int}} \delta_{\alpha}$

  $S \cup \{ (\alpha \lor \beta)^R \} \xrightarrow{\text{int}} \delta_{\alpha}$

  $S \cup \{ \beta^R \} \xrightarrow{\text{int}} \delta_{\beta}$

  as $\bigwedge S^L \rightarrow \delta_{\alpha}$, $\delta_{\alpha} \rightarrow \neg (\alpha^R \land \bigwedge S^R)$ and $\bigwedge S^L \rightarrow \delta_{\beta}$, $\delta_{\beta} \rightarrow \neg (\beta^R \land \bigwedge S^R)$

  implies $(\bigwedge S^L) \rightarrow \delta_{\alpha} \land \delta_{\beta}$, $\delta_{\alpha} \land \delta_{\beta} \rightarrow \neg ((\alpha \lor \beta)^R \land \bigwedge S^R)$;

- etc. (see [Fitting] for details)
Tableau and Interpolant Extraction (by example)

- an interpolant $S \xrightarrow{\text{int}} \psi$; invariant $(\land S^L) \rightarrow \psi$ and $\psi \rightarrow (\neg \land S^R)$
  
  where $S^L$ and $S^R$ are the left/right subsets of $S$;

- tableau rules (sample):
  
  - LR clash $S \cup \{R^L, \neg R^R\} \xrightarrow{\text{int}} R$, where $R \in S_A$
    
    as $(\land S^L \land R^L) \rightarrow R$ and $R \rightarrow (R^R \lor \neg \land S^R)$;

  - L-conjunction $S \cup \{\alpha^L, \beta^L\} \xrightarrow{\text{int}} \delta$
    
    $S \cup \{(\alpha \land \beta)^L\} \xrightarrow{\text{int}} \delta$
    
    as $(\land S^L \land \alpha^L \land \beta^L) \rightarrow \delta$ implies $(\land S^L \land (\alpha \land \beta)^L) \rightarrow \delta$;

  - R-Disjunction $S \cup \{\alpha^R\} \xrightarrow{\text{int}} \delta_\alpha$
    
    $S \cup \{\beta^R\} \xrightarrow{\text{int}} \delta_\beta$
    
    $S \cup \{(\alpha \lor \beta)^R\} \xrightarrow{\text{int}} \delta_\alpha \land \delta_\beta$
    
    as $\land S^L \rightarrow \delta_\alpha, \delta_\alpha \rightarrow \neg (\alpha^R \land \land S^R)$ and $\land S^L \rightarrow \delta_\beta, \delta_\beta \rightarrow \neg (\beta^R \land \land S^R)$
    
    implies $(\land S^L) \rightarrow \delta_\alpha \land \delta_\beta, \delta_\alpha \land \delta_\beta \rightarrow \neg ((\alpha \lor \beta)^R \land \land S^R)$;

  - etc. (see [Fitting] for details)
Tableau and Interpolant Extraction (by example)

- an interpolant \( S \xrightarrow{\text{int}} \psi \); invariant \(( \land S_L ) \rightarrow \psi \) and \( \psi \rightarrow ( \neg \land S_R )\)
  where \( S_L \) and \( S_R \) are the left/right subsets of \( S \);

- tableau rules (sample):
  
  - LR clash \( S \cup \{ R^L, \neg R^R \} \xrightarrow{\text{int}} R \), where \( R \in S_A \)
    as \(( \land S_L \land R^L ) \rightarrow R \) and \( R \rightarrow ( R^R \lor \neg \land S_R )\);
  
  - L-conjunction
    \[
    \frac{S \cup \{ \alpha^L, \beta^L \} \xrightarrow{\text{int}} \delta}{S \cup \{ (\alpha \land \beta)^L \} \xrightarrow{\text{int}} \delta}
    \]
    as \(( \land S_L \land \alpha^L \land \beta^L ) \rightarrow \delta \) implies \(( \land S_L \land (\alpha \land \beta)^L ) \rightarrow \delta\);

  - R-Disjunction
    \[
    \frac{S \cup \{ \alpha^R \} \xrightarrow{\text{int}} \delta_\alpha}{S \cup \{ (\alpha \lor \beta)^R \} \xrightarrow{\text{int}} \delta_\alpha \land \delta_\beta}
    \]
    as \( \land S_L \rightarrow \delta_\alpha, \delta_\alpha \rightarrow ( \neg R^R \land S^R ) \) and \( \land S_L \rightarrow \delta_\beta, \delta_\beta \rightarrow \neg ( \neg R^R \land \land S^R )\)
    implies \(( \land S_L ) \rightarrow \delta_\alpha \land \delta_\beta, \delta_\alpha \land \delta_\beta \rightarrow \neg ((\alpha \lor \beta)^R \land \land S^R)\).

- etc. (see [Fitting] for details)
First-order Variables and Equality

- Quantifier rules
  1. inference rules with Ground constants/terms

  **Quantifier Rules:**
  \[
  \begin{align*}
  & \Gamma, \varphi(t/x) \vdash \Delta \quad (\forall L) \\
  & \Gamma, (\forall x. \varphi) \vdash \Delta \quad (\forall R)
  \end{align*}
  \]

  2. unification tableau and Skolemization (refutation systems)

- Equality
  1. High-school Axioms (immediate implementation)

    \[
    \begin{align*}
    & \Gamma \vdash x = x \\
    & \Gamma \vdash x = y \land \varphi \vdash \varphi(y/x)
    \end{align*}
    \]

  2. Superposition rules (efficient implementation)
First-order Variables and Equality

- Quantifier rules
  - inference rules with Ground constants/terms

  **Quantifier Rules:**

  \[
  \frac{\Gamma, \varphi(t/x) \vdash \Delta}{\Gamma, (\forall x. \varphi) \vdash \Delta} \quad (\forall L) \quad \frac{\Gamma \vdash \varphi(y/x), \Delta}{\Gamma \vdash (\forall R)}
  \]

- unification tableau and Skolemization (refutation systems)

- Equality
  - High-school Axioms (immediate implementation)
    \[
    \vdash x = x \\
    x = y \land \varphi \vdash \varphi(y/x)
    \]
  - Superposition rules (efficient implementation)
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):

\[ \Sigma^L \cup \Sigma^R \cup \Sigma^{LR} \models \phi^L \rightarrow \phi^R \text{ where } \Sigma^{LR} = \{ \forall \vec{x} . P^L \rightarrow P \rightarrow P^R | P \in S_A \} \]

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)
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**

ϕ[C] where C is a set of (ground) atoms over \( S_p \)

ϕ only exists if all atoms in C are “used” in a plan tableau.

---

**Absorbed Range-restricted Formulæ: ANF**

\[ Q ::= R(\overline{x}) \mid \bot \mid Q \land Q \mid Q \lor Q \mid \forall \overline{x}.R(\overline{x}) \rightarrow Q, \]

... and all \( \exists \)'s are Skolemized.

---

**Conditional Tableau Rules for ANF**

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

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

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

\[
\frac{S \cup \{R(\overline{t})[R(\overline{t})]\}}{R(\overline{x}) \in S_A} \quad \text{(phys)}
\]
Conditional Tableau and Interpolation

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 a set \(C\) of literals over \(S_A\) a **closing set** for \(T\) if, for every branch

1. there is an atom \(R(\bar{t})[D]\) such that \(D \cup \{\neg R(\bar{t})\} \subseteq C\).
2. there is \(\bot[D]\) such that \(D \subseteq C\).

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

Observation

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

Physical Tableau $T^P$ for a Plan $P$

<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 tree $T^P$ 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 of literals $C \in L_P$ (left branch) or $C \in R_P$ (right branch); and
2. The interpolant extracted from the closed tableau $T^P[T^L, T^R]$, the closure of $(T^L, T^R)$ by (the branches of) $T^P$, is logically equivalent to $P$. 
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, \(T^P[TL, TR]\) is closed tableau yielding an interpolant equivalent to \(P\);
(… otherwise extend depth in step 1 and repeat.)

NOTE: 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 \(R(\bar{t})\) 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 \((T^L, T^R)\).

… combine with \(A^*\) search (among \(P\)s) with respect to a cost model.
Postprocessing: Duplicate Elimination Elimination

**IDEA:**

Separate the projection operation ($\exists \vec{x}.$) to
- a duplicate preserving projection ($\exists$) and
- an explicit (idempotent) duplicate elimination operator ($\{\cdot\}$).

Use the following rewrites to eliminate/minimize the use of $\{\cdot\}$:

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

... reasoning abstracted: a DL $CFD^{\vec{y}}_{\mid\parallel}$ (a PTIME fragment)
Ideas:

- Separate the projection operation \((\exists \bar{x}.)\) to
- a duplicate preserving projection \((\exists)\) and
- an explicit (idempotent) duplicate elimination operator \((\{\cdot\})\).

Use the following rewrites to eliminate/minimize the use of \((\{\cdot\})\):

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

... reasoning abstracted: a DL \(CD_{\mu}^\chi\) (a PTIME fragment)
Postprocessing: Duplicate Elimination Elimination

**IDEA:**
Separate the projection operation ($\exists \bar{x}.$) to
- a duplicate preserving projection ($\exists$) and
- an explicit (idempotent) duplicate elimination operator ($\{\cdot\}$).

Use the following rewrites to eliminate/minimize the use of $\{\cdot\}$:

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

... reasoning abstracted: a DL $\mathcal{CFD}_{\forall}^{-\text{nc}}$ (a PTIME fragment)
Summary

Take Home

While in theory interpolation essentially solves the query rewriting over FO schemas/views problem, the devil is (as usual) in the details.

[Borgida, de Bruijn, Franconi, Seylan, Straccia, Toman, Weddell: On Finding Query Rewritings under Expressive Constraints. SEBD 2010: 426-437

... but an (almost) working system only this year.

1. FO $\mathcal{DLFDE}$ tableau based interpolation algorithm
   ⇒ enumeration of plans factored from of tableau reasoning
   ⇒ extra-logical binding patterns and cost model

2. Post processing (using $CFLDI_{nc}$ approximation)
   ⇒ duplicate elimination elimination
   ⇒ cut insertion

3. Run time
   ⇒ library of common data/legacy structures+schema constraints
   ⇒ finger data structures to simulate merge joins et al.
Research Directions and 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, Certicom, ...
- part of a School of CS with 75+ professors, 300+ grad students, etc.
  - AI&ML, Algorithms&DS, PL, Theory, Systems, Networks, Graphics, ...

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, ...