Query Compilation

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

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

\[
\Sigma_L \rightarrow S_L \quad \varphi
\]

(Logical Schema)

\[
\Sigma_{LP} \rightarrow \psi
\]

(rewriting)

\[
\Sigma_P \rightarrow S_A \subseteq S_P \quad \psi
\]

(Physical Schema)
Query Plans via Rewriting

### Plans as Formulas

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

<table>
<thead>
<tr>
<th>Logical Symbol</th>
<th>Translation</th>
</tr>
</thead>
<tbody>
<tr>
<td>atomic formula</td>
<td>access path ((\text{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>

⇒ 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>Operation</th>
<th>Equivalent</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>

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

## 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**
# 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>Operation</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

(THE OLD WAY)
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

Example: (Nash)

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

**Example: (Nash)**

**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:**
- \( Q(x, y) \leftrightarrow \exists t, u, v. R(t, x) \land R(t, u) \land R(u, v) \land R(v, y) \)
Chase and Backchase

IDEA #1 (Database Theory, CQ/UCQ)

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

Example: (Nash)

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

Example (Nash)

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

Solution(s):  
$\exists z. V_1(z, x) \land \forall w. V_2(w, z) \rightarrow V_3(w, y)$  
$\exists z. V_3(z, y) \land \forall w. V_2(w, z) \rightarrow V_1(x, z)$
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

Example (Nash)

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

Solution(s): $\exists z. V_1(x, z) \land \forall v. V_2(v, z) \rightarrow V_3(v, y)$
$\exists z. V_3(z, y) \land \forall v. V_2(v, z) \rightarrow V_1(x, v)$
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

Example (Nash)

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

Solution(s): $\exists z. V_1(x, z) \land \forall v. V_2(v, z) \rightarrow V_3(v, y)$
$\exists z. V_3(z, y) \land \forall v. V_2(v, z) \rightarrow V_1(x, v)$ but no sol’s in C&B
INTERPOLATION

(The New Way—that is really old)
IDEA #2: What Queries do we allow?

We only allow queries that have the same answer in every model of $\Sigma$ ... for a fixed interpretation of $S_A$ (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 NOT in $S_A$ are primed, respectively.

How do we find the rewriting $\psi$?

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 can extract an interpolant $\psi$ from a (LK) proof of $\Sigma \cup \Sigma' \models \varphi \rightarrow \varphi'$. 

...
Beth Definability and Interpolation

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 interpretation of $S_A$ (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 \textit{primed}, respectively.

How do we find the rewriting $\psi$?

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

... $\psi$ is called the \textit{Craig Interpolant} [Craig’57].

... we can extract an \textit{interpolant} $\psi$ from a (LK) proof of $\Sigma \cup \Sigma' \models \varphi \rightarrow \varphi'$. 
Beth Definability and Interpolation

IDEA #2: What Queries do we allow?

We only allow queries that have the same answer in every model of $\Sigma$ for a fixed interpretation of $S_A$ (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 *NOT in* $S_A$ are primed, respectively.

Why??

(i) symbols in $\Sigma \cup \{\varphi\}$ are interpreted as in the 1st model
(ii) symbols in $\Sigma' \cup \{\varphi'\}$ are interpreted as in the 2nd model
(iii) symbols in $S_A$ must be interpreted the same

How do we find the rewriting $\psi$?

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].
Beth Definability and Interpolation

IDEA #2: What Queries do we allow?

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

How do we test for this?

$\phi$ is *Beth definable* [Beth’56] if $\Sigma \cup \Sigma' \models \phi \rightarrow \phi'$ where $\Sigma'$ ($\phi'$) is $\Sigma$ ($\phi$) in which symbols *NOT in $S_A$* are primed, respectively.

Why??

(i) symbols in $\Sigma \cup \{\phi\}$ are interpreted as in the 1st model
(ii) symbols in $\Sigma' \cup \{\phi'\}$ are interpreted as in the 2nd model
(iii) symbols in $S_A$ must be interpreted the same

How do we find the rewriting $\psi$?

If $\Sigma \cup \Sigma' \models \phi \rightarrow \phi'$ then there is $\psi$ s.t. $\Sigma \cup \Sigma' \models \phi \rightarrow \psi \rightarrow \phi'$ with $L(\psi) \subseteq L(S_A)$. $\psi$ is called the *Craig Interpolant* [Craig’57].
IDEA #2: What Queries do we allow?

We only allow queries that have the same answer in every model of $\Sigma$ … for a fixed interpretation of $S_A$ (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 NOT in $S_A$ are primed, respectively.

How do we find the rewriting $\psi$?

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 can extract an interpolant $\psi$ from a (LK) proof of $\Sigma \cup \Sigma' \models \varphi \rightarrow \varphi'$
Sequent Calculus: LK

Identity Rules:

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

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

Logical Rules:

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

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

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

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

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

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

Identity Rules:

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

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

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

Logical Rules:

\[ \Gamma, \varphi \vdash \Delta \quad \Gamma, \psi \vdash \Delta \]

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

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

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

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

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

\[ \Gamma \vdash \varphi, \Delta \quad \Gamma \vdash \psi, \Delta \]

\[ \Gamma \vdash (\varphi \land \psi), \Delta \]  (\land R)
Sequent Calculus (for NNF) and Interpolation

### Identity Rules:

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

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

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

### Logical Rules:

\[
\begin{align*}
\Gamma, \varphi & \vdash \Delta \leadsto \alpha & \Gamma, \psi & \vdash \Delta \leadsto \beta \\
\hline
\Gamma, (\varphi \lor \psi) & \vdash \Delta \leadsto \alpha \lor \beta
\end{align*}
\]

\[
\begin{align*}
\Gamma & \vdash \varphi, \psi, \Delta \leadsto \alpha \\
\Gamma & \vdash (\varphi \lor \psi), \Delta \leadsto \alpha
\end{align*}
\]

\[
\begin{align*}
\Gamma, \varphi, \psi & \vdash \Delta \leadsto \alpha \\
\Gamma, (\varphi \land \psi) & \vdash \Delta \leadsto \alpha
\end{align*}
\]

\[
\begin{align*}
\Gamma & \vdash \varphi, \Delta \leadsto \alpha & \Gamma & \vdash \psi, \Delta \leadsto \beta \\
\hline
\Gamma & \vdash (\varphi \land \psi), \Delta \leadsto \alpha \land \beta
\end{align*}
\]
LK and Theories

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

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

\[ \Rightarrow \text{we use ANALYTIC TABLEAU: a refutation variant of LK to show} \]
\[ \Sigma, \Sigma', \varphi, \neg \varphi' \vdash \bot \quad \text{a.k.a. is inconsistent} \]

\[ \Rightarrow \text{need to tag left (L)/right(R) formulae to simulate sequent sides (for interpolation)!} \]
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 ANALYTIC TABLEAU: a refutation variant of LK to show

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

⇒ need to tag left (L)/right(R) formulae to simulate sequent sides (for interpolation)!
LK and Theories

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

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

\[ \Rightarrow \text{we use } \textsc{Analytic Tableau}: \text{a refutation variant of LK to show} \]
\[ \Sigma, \Sigma', \varphi, \neg \varphi' \vdash \bot \quad \text{a.k.a. is inconsistent} \]

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

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

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

\( \Rightarrow \) 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 sequent sides (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):
  - **LL clash** \( S \cup \{ P^L, \neg P^L \} \xrightarrow{\text{int}} \bot \) (similar for “RR”: \( \top \))
  - **LR clash** \( S \cup \{ P^L, \neg P^R \} \xrightarrow{\text{int}} P \) where \( P \in S_A \)
  - **RL clash** \( S \cup \{ \neg P^L, P^R \} \xrightarrow{\text{int}} \neg P \) where \( P \in S_A \)
  - **L-conjunction** \( S \cup \{ \alpha^L, \beta^L \} \xrightarrow{\text{int}} \delta \)
    \( S \cup \{ (\alpha \land \beta)^L \} \xrightarrow{\text{int}} \delta \)
  - **R-Disjunction** \( S \cup \{ \alpha^R \} \xrightarrow{\text{int}} \delta_\alpha \)
    \( S \cup \{ \beta^R \} \xrightarrow{\text{int}} \delta_\beta \)
    \( S \cup \{ (\alpha \lor \beta)^R \} \xrightarrow{\text{int}} \delta_\alpha \land \delta_\beta \)
  - etc. (see [Fitting] for details)
Tableau and Interpolant Extraction (by example)

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

- tableau rules (sample):
  
  - LL clash \( S \cup \{P^L, \neg P^L\} \overset{\text{int}}{\rightarrow} \bot \) (similar for “RR”: \(\top\))
  
  - LR clash \( S \cup \{P^L, \neg P^R\} \overset{\text{int}}{\rightarrow} P \), where \(P \in S_A\)
  
  - RL clash \( S \cup \{\neg P^L, P^R\} \overset{\text{int}}{\rightarrow} \neg P \), where \(P \in S_A\)
  
  - L-conjunction \( S \cup \{\alpha^L, \beta^L\} \overset{\text{int}}{\rightarrow} \delta \)
  
  \( S \cup \{(\alpha \land \beta)^L\} \overset{\text{int}}{\rightarrow} \delta \)
  
  - R-Disjunction \( S \cup \{\alpha^R\} \overset{\text{int}}{\rightarrow} \delta \)
  
  \( S \cup \{\beta^R\} \overset{\text{int}}{\rightarrow} \delta \)
  
  \( S \cup \{(\alpha \lor \beta)^R\} \overset{\text{int}}{\rightarrow} \delta \land \delta \)
  
  - 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):
  
  - **LL clash** $S \cup \{P^L, \neg P^L\} \xrightarrow{\text{int}} \bot$ (similar for “RR”: $\top$)
  
  - **LR clash** $S \cup \{P^L, \neg P^R\} \xrightarrow{\text{int}} P$, where $P \in S_A$
  
  - **RL clash** $S \cup \{\neg P^L, P^R\} \xrightarrow{\text{int}} \neg P$, where $P \in S_A$
  
  - **L-conjunction**
    
    $S \cup \{\alpha^L, \beta^L\} \xrightarrow{\text{int}} \delta$
    
    $S \cup \{\alpha^L \land \beta^L\} \xrightarrow{\text{int}} \delta$
  
  - **R-Disjunction**
    
    $S \cup \{\alpha^R\} \xrightarrow{\text{int}} \delta\alpha$
    
    $S \cup \{\beta^R\} \xrightarrow{\text{int}} \delta\beta$
    
    $S \cup \{(\alpha \lor \beta)^R\} \xrightarrow{\text{int}} \delta\alpha \lor \delta\beta$

- 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):
  - LL clash \( S \cup \{P^L, \neg P^L\} \xrightarrow{\text{int}} \bot \) (similar for “RR”: \( \top \))
  - LR clash \( S \cup \{P^L, \neg P^R\} \xrightarrow{\text{int}} P \), where \( P \in S_A \)
  - RL clash \( S \cup \{\neg P^L, P^R\} \xrightarrow{\text{int}} \neg P \), where \( P \in S_A \)
  - L-conjunction \( S \cup \{\alpha^L, \beta^L\} \xrightarrow{\text{int}} \delta \)
    \( S \cup \{(\alpha \land \beta)^L\} \xrightarrow{\text{int}} \delta \)
  - R-Disjunction
    \( S \cup \{\alpha^R\} \xrightarrow{\text{int}} \delta_\alpha \)
    \( S \cup \{\beta^R\} \xrightarrow{\text{int}} \delta_\beta \)
    \( S \cup \{(\alpha \lor \beta)^R\} \xrightarrow{\text{int}} \delta_\alpha \land \delta_\beta \)

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

- Quantifier rules

1. inference rules with Ground constants/terms

**Quantifier Rules:**

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

2. 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)
First-order Variables and Equality

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

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

  2. unification tableau and Skolemization (refutation systems)

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

    \[
    \vdash x = x \\
    x = y \land \varphi \vdash \varphi(y/x)
    \]

  2. 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):

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

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)
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):

$$\Sigma^L \cup \Sigma^R \cup \Sigma^{LR} \models \varphi^L \rightarrow \varphi^R$$ where $$\Sigma^{LR} = \{ \forall \bar{x}. P^L \leftrightarrow P \leftrightarrow P^R | P \in S_A \}$$

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)
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**

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

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

**Conditional Tableau Rules for ANF**

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

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

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

\[
\frac{S \cup \{R(\bar{t})[R(\bar{t})]\}}{R(\bar{x}) \in S_A} \quad \text{(phys)}
\]
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 \(\bot(\top)\).
Conditional Tableau and Interpolation: dirty secrets

- Binding patterns
  ⇒ needs additional physical atoms that provide *bindings*
  ⇒ must appear in the tableau “on the correct side”
  ⇒ added to closing sets “soundly”

- Functionality (for duplicates)
  ⇒ needs additional physical atoms that *functionally determine quantified variables*
  ⇒ must appear in the tableau “on the correct side”
  ⇒ added to closing sets “soundly”
Conditional Tableau and Interpolation: dirty secrets

- Binding patterns
  - needs additional physical atoms that provide *bindings*
  - must appear in the tableau “on the correct side”
  - added to closing sets “soundly”

- Functionality (for duplicates)
  - needs additional physical atoms
    - that *functionally determine quantified variables*
  - must appear in the tableau “on the correct side”
  - added to closing sets “soundly”
Plan Enumeration

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

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

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

---

David Toman (et al.)

Query Compilation

How does it work? 20 / 24
**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>$P(t)$</td>
<td>${{\neg P^L(t)}}$</td>
<td>${{P^R(t)}}$</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>

**Diagram**

- $\neg P^L$
- $P$
- $\neg P$
- $P^R$

**Interpolant:** $P$

$(P^L \rightarrow P)$

$(P \rightarrow P^R)$
Plan Enumeration

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

\[
\begin{align*}
P & : L_P \\
\neg P(t) & : \{\{\neg P_L(t)\}\} \\
P_1 \land P_2 & : L_{P_1} \cup L_{P_2} \\
P_1 \lor P_2 & : \{S_1 \cup S_2 \mid S_1 \in L_{P_1}, S_2 \in L_{P_2}\} \\
\neg P_1 & : \{\{L^L(t) \mid L^R(t) \in S\} \mid S \in R_{P_1}\} \\
\exists x. P_1 & : L_{P_1}[t/x] \\
R_P & : \{\{P^R(t)\}\} \\
& \{S_1 \cup S_2 \mid S_1 \in R_{P_1}, S_2 \in R_{P_2}\} \\
& R_{P_1} \cup R_{P_2} \\
& \{\{L^R(t) \mid L^L(t) \in S\} \mid S \in L_{P_1}\} \\
& R_{P_1}[t/x]
\end{align*}
\]

\[
\neg P^L \quad P \quad P^R
\]

\[
T^L \quad \times \quad T^R
\]

\[
(P^L \rightarrow P) \\
(P \rightarrow P^R)
\]

interpolant: $P$
## Plan Enumeration

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

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

---

**University of Waterloo**

David Toman (et al.)

Query Compilation

How does it work? 20/24
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.)

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 \((T^L, T^R)\).

\ldots combine with \(A^*\) search (among \(P\)s) with respect to a cost model.
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 \(\{\cdot\}\):

\[
\begin{align*}
Q[[R(x_1, \ldots, x_k)]] & \leftrightarrow Q[R(x_1, \ldots, x_k)] \\
Q[[Q_1 \land Q_2]] & \leftrightarrow Q[[Q_1 \land \{Q_2\}]] \\
Q[\neg Q_1] & \leftrightarrow Q[\neg Q_1] \\
Q[\neg\{Q_1\}] & \leftrightarrow Q[\neg Q_1] \\
Q[[Q_1 \lor Q_2]] & \leftrightarrow Q[[Q_1 \lor \{Q_2\}]] & \text{if } \Sigma \cup \{Q\} \cup \{Q_1 \land Q_2 \rightarrow \top\} \\
Q[[\exists x. Q_1]] & \leftrightarrow Q[\exists x. \{Q_1\}] & \text{if } \\
\Sigma \cup \{Q\} \land (Q_1[y_1/x] \land (Q_1[y_2/x]) & = y_1 \approx y_2
\end{align*}
\]

... reasoning abstracted: a DL \(\forall D_{\text{cf}}\) (a PTIME fragment)
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 $\{\cdot\}$:

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

... reasoning abstracted: a DL $\mathcal{CFD}^\exists_{\mathcal{ID}}$ (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 $\{\cdot\}$:

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

... reasoning abstracted: a DL $\mathcal{CDF}_{nc}$ (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 (DLFDε) tableau based interpolation algorithm
   ⇒ enumeration of plans factored from of tableau reasoning
   ⇒ extra-logical binding patterns and cost model

2. Post processing (using CFDI 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!