Query Compilation

David Toman

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

Joint work with Alexander Hudek and Grant Weddell

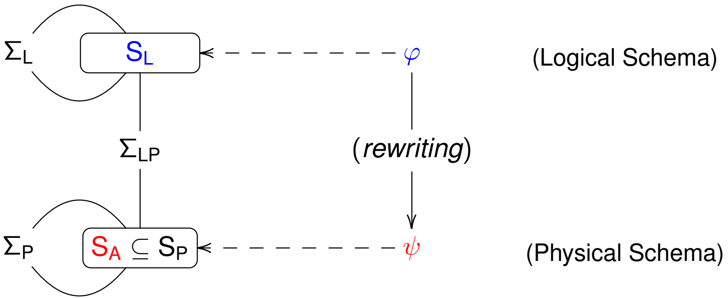
Grand Unified Approach to Query Compilation

Part II: How does it work?

The Plan

Definability and Rewriting

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



Query Plans via Rewriting

Plans as Formulas

Represent *query plans* as (annotated) range-restricted formulas ψ 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

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

Non-logical (but necessary) Addons

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 ψ 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

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

Non-logical (but necessary) Additions

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 ψ 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

$\Rightarrow$ reduces correctness of ψ 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

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

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

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

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

Chase and Backchase

IDEA #1 (Database Theory, CQ/UCQ)

Inference(s): $Q, (\forall \bar{x}.Q_1 \to Q_2) \vdash Q \cup Q_2\theta$ 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: $V_1(x, y) \leftrightarrow \exists t, u, v.R(t, x) \wedge R(t, u) \wedge R(u, y)$

$V_2(x, y) \leftrightarrow \exists u.R(x, u) \wedge R(u, y)$

$V_3(x, y) \leftrightarrow \exists t, u.R(x, t) \wedge R(t, u) \wedge R(u, y)$

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

Chase and Backchase

IDEA #1 (Database Theory, CQ/UCQ)

Inference(s): $Q, (\forall \bar{x}. Q_1 \to Q_2) \vdash Q \cup Q_2\theta$ 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:
$$V_1(x, y) \leftrightarrow \exists t, u, v. R(t, x) \wedge R(t, u) \wedge R(u, y)$$
$$V_2(x, y) \leftrightarrow \exists u. R(x, u) \wedge R(u, y)$$
$$V_3(x, y) \leftrightarrow \exists t, u. R(x, t) \wedge R(t, u) \wedge R(u, y)$$

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

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

University of Waterloo

Chase and Backchase

IDEA #1 (Database Theory, CQ/UCQ)

Inference(s): $Q, (\forall \bar{x}.Q_1 \to Q_2) \vdash Q \cup Q_2\theta$ 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:
$$V_1(x, y) \leftrightarrow \exists t, u, v.R(t, x) \wedge R(t, u) \wedge R(u, y)$$
$$V_2(x, y) \leftrightarrow \exists u.R(x, u) \wedge R(u, y)$$
$$V_3(x, y) \leftrightarrow \exists t, u.R(x, t) \wedge R(t, u) \wedge R(u, y)$$

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

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

University of Waterloo

Chase and Backchase

IDEA #1 (Database Theory, CQ/UCQ)

Inference(s): $Q, (\forall \bar{x}.Q_1 \to Q_2) \vdash Q \cup Q_2\theta$ 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:
$$V_1(x,y) \leftrightarrow \exists t,u,v.R(t,x) \wedge R(t,u) \wedge R(u,y)$$
$$V_2(x,y) \leftrightarrow \exists u.R(x,u) \wedge R(u,y)$$
$$V_3(x,y) \leftrightarrow \exists t,u.R(x,t) \wedge R(t,u) \wedge R(u,y)$$

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

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

but no sol’s in C&B

University of Waterloo

INTERPOLATION

(THE NEW WAY—THAT IS REALLY OLD)

Beth Definability and Interpolation

IDEA #2: What Queries do we allow?

We only allow queries that have *the same answer* in every model of Σ

. . . for a fixed interpretation of S_A (i.e., where the actual data is).

How do we test for this?

φ is *Beth definable* [Beth'56] if $\Sigma \cup \Sigma' \models \varphi \rightarrow \varphi'$ where Σ' (φ') is Σ (φ)

in which symbols *NOT in* S_A are *primed*, respectively.

How do we find the rewriting ψ?

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

. . . ψ is called the *Craig Interpolant* [Craig'57].

. . . we can extract an *interpolant* ψ 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 Σ

... for a fixed interpretation of S_A (i.e., where the actual data is).

How do we test for this?

φ is *Beth definable* [Beth'56] if $\Sigma \cup \Sigma' \models \varphi \to \varphi'$ where Σ' (φ') is Σ (φ)

in which symbols *NOT in* S_A are *primed*, respectively.

How do we find the rewriting ψ?

If $\Sigma \cup \Sigma' \models \varphi \to \varphi'$ then there is ψ s.t. $\Sigma \cup \Sigma' \models \varphi \to \psi \to \varphi'$ with $\mathcal{L}(\psi) \subseteq \mathcal{L}(S_A)$.

... ψ is called the *Craig Interpolant* [Craig'57].

... we can extract an *interpolant* ψ from a (LK) proof of $\Sigma \cup \Sigma' \models \varphi \to \varphi'$

Beth Definability and Interpolation

IDEA #2: What Queries do we allow?

We only allow queries that have *the same answer* in every model of Σ

…for a fixed interpretation of S_A (i.e., where the actual data is).

How do we test for this?

φ is *Beth definable* [Beth'56] if $\Sigma \cup \Sigma' \models \varphi \to \varphi'$ where Σ' (φ') is Σ (φ)

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?

If $\Sigma \cup \Sigma' \models \varphi \to \varphi'$ then there is ψ s.t. $\Sigma \cup \Sigma' \models \varphi \to \psi \to \varphi'$ with $\mathcal{L}(\psi) \subseteq \mathcal{L}(S_A)$.

… ψ 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 Σ

…for a fixed interpretation of S_A (i.e., where the actual data is).

How do we test for this?

φ is *Beth definable* [Beth'56] if $\Sigma \cup \Sigma' \models \varphi \rightarrow \varphi'$ where Σ' (φ') is Σ (φ)

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?

if $\Sigma \cup \Sigma' \models \varphi \rightarrow \varphi'$ then there is ψ s.t. $\Sigma \cup \Sigma' \models \varphi \rightarrow \psi \rightarrow \varphi'$ with $\mathcal{L}(\psi) \subseteq \mathcal{L}(S_A)$

… ψ 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 Σ

. . . for a fixed interpretation of $\mathsf{S_A}$ (i.e., where the actual data is).

How do we test for this?

φ is *Beth definable* [Beth'56] if $\Sigma \cup \Sigma' \models \varphi \rightarrow \varphi'$ where Σ' (φ') is Σ (φ)

in which symbols *NOT in* $\mathsf{S_A}$ are *primed*, respectively.

How do we find the rewriting ψ?

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

. . . ψ is called the *Craig Interpolant* [Craig'57].

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

Sequent Calculus: LK

Identity Rules:

$$\frac{}{\Gamma, \varphi \vdash \varphi, \Delta}\ (\mathit{Axiom}) \qquad \frac{\Gamma \vdash \varphi, \Delta \qquad \Gamma, \varphi \vdash \Delta}{\Gamma \vdash \Delta}\ (\mathit{Cut})$$

Logical Rules:

$$\frac{\Gamma \vdash \varphi, \Delta}{\Gamma, (\neg\varphi) \vdash \Delta}\ (\neg L) \qquad \frac{\Gamma, \varphi \vdash \Delta}{\Gamma \vdash (\neg\varphi), \Delta}\ (\neg R)$$

$$\frac{\Gamma, \varphi \vdash \Delta \qquad \Gamma, \psi \vdash \Delta}{\Gamma, (\varphi \vee \psi) \vdash \Delta}\ (\vee L) \qquad \frac{\Gamma \vdash \varphi, \psi, \Delta}{\Gamma \vdash (\varphi \vee \psi), \Delta}\ (\vee R)$$

$$\frac{\Gamma, \varphi, \psi \vdash \Delta}{\Gamma, (\varphi \wedge \psi) \vdash \Delta}\ (\wedge L) \qquad \frac{\Gamma \vdash \varphi, \Delta \qquad \Gamma \vdash \psi, \Delta}{\Gamma \vdash (\varphi \wedge \psi), \Delta}\ (\wedge R)$$

Sequent Calculus: Cut Elimination

Theorem (Hauptsatz)

For every proof of a sequent $\Gamma \vdash \Delta$ *in* LK *there is also proof of the same sequent in* $LK - \{Cut\}$.

Sequent Calculus (for NNF)

Identity Rules:

$$\frac{}{\Gamma, \varphi \vdash \varphi, \Delta}\ (\textit{Axiom LR})$$

$$\frac{}{\Gamma, \varphi, \neg\varphi \vdash \Delta}\ (\textit{Axiom RR}) \qquad \frac{}{\Gamma \vdash \varphi, \neg\varphi, \Delta}\ (\textit{Axiom LL})$$

Logical Rules:

$$\frac{\Gamma, \varphi \vdash \Delta \qquad \Gamma, \psi \vdash \Delta}{\Gamma, (\varphi \vee \psi) \vdash \Delta}\ (\vee L) \qquad \frac{\Gamma \vdash \varphi, \psi, \Delta}{\Gamma \vdash (\varphi \vee \psi), \Delta}\ (\vee R)$$

$$\frac{\Gamma, \varphi, \psi \vdash \Delta}{\Gamma, (\varphi \wedge \psi) \vdash \Delta}\ (\wedge L) \qquad \frac{\Gamma \vdash \varphi, \Delta \qquad \Gamma \vdash \psi, \Delta}{\Gamma \vdash (\varphi \wedge \psi), \Delta}\ (\wedge R)$$

Sequent Calculus (for NNF) and Interpolation

Identity Rules:

$$\frac{}{\Gamma, \varphi \vdash \varphi, \Delta \leadsto \varphi}$$

$$\frac{}{\Gamma, \varphi, \neg\varphi \vdash \Delta \leadsto \bot} \qquad \frac{}{\Gamma \vdash \varphi, \neg\varphi, \Delta \leadsto \top}$$

Logical Rules:

$$\frac{\Gamma, \varphi \vdash \Delta \leadsto \alpha \qquad \Gamma, \psi \vdash \Delta \leadsto \beta}{\Gamma, (\varphi \vee \psi) \vdash \Delta \leadsto \alpha \vee \beta} \qquad \frac{\Gamma \vdash \varphi, \psi, \Delta \leadsto \alpha}{\Gamma \vdash (\varphi \vee \psi), \Delta \leadsto \alpha}$$

$$\frac{\Gamma, \varphi, \psi \vdash \Delta \leadsto \alpha}{\Gamma, (\varphi \wedge \psi) \vdash \Delta \leadsto \alpha} \qquad \frac{\Gamma \vdash \varphi, \Delta \leadsto \alpha \qquad \Gamma \vdash \psi, \Delta \leadsto \beta}{\Gamma \vdash (\varphi \wedge \psi), \Delta \leadsto \alpha \wedge \beta}$$

LK and Theories

$$
\begin{aligned}
\Sigma \cup \Sigma' \models \varphi \rightarrow \varphi' &\iff (\bigwedge \Sigma) \wedge (\bigwedge \Sigma)' \models \varphi \rightarrow \varphi' \\
&\iff \models (\bigwedge \Sigma) \rightarrow ((\bigwedge \Sigma)' \rightarrow (\varphi \rightarrow \varphi')) \\
&\iff \models (\bigwedge \Sigma) \rightarrow (\varphi \rightarrow ((\bigwedge \Sigma)' \rightarrow \varphi')) \\
&\iff (\bigwedge \Sigma) \wedge \varphi \models (\bigwedge \Sigma)' \rightarrow \varphi') \\
&\iff (\bigwedge \Sigma) \wedge \varphi \models (\bigvee \neg\Sigma)' \vee \varphi' \\
&\iff \Sigma, \varphi \vdash (\neg\Sigma'), \varphi' \quad \text{(due to soundness/completeness)}
\end{aligned}
$$

Not convenient: needs both Σ and negated Σ'!

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

$\Sigma, \Sigma', \varphi, \neg\varphi' \vdash \bot$ a.k.a. is inconsistent

$\Rightarrow$ need to *tag* left (L)/right(R) formulae to simulate sequent sides (for interpolation)!

LK and Theories

$$
\begin{aligned}
\Sigma \cup \Sigma' \models \varphi \to \varphi' &\iff (\bigwedge \Sigma) \wedge (\bigwedge \Sigma)' \models \varphi \to \varphi' \\
&\iff \models (\bigwedge \Sigma) \to ((\bigwedge \Sigma)' \to (\varphi \to \varphi')) \\
&\iff \models (\bigwedge \Sigma) \to (\varphi \to ((\bigwedge \Sigma)' \to \varphi')) \\
&\iff (\bigwedge \Sigma) \wedge \varphi \models (\bigwedge \Sigma)' \to \varphi') \\
&\iff (\bigwedge \Sigma) \wedge \varphi \models (\bigvee \neg\Sigma)' \vee \varphi' \\
&\iff \Sigma, \varphi \vdash (\neg\Sigma'), \varphi' \qquad \text{(due to soundness/completeness)}
\end{aligned}
$$

Not convenient: needs both Σ and negated Σ'!

$\Rightarrow$ 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)!

LK and Theories

$$
\begin{aligned}
\Sigma \cup \Sigma' \models \varphi \to \varphi' &\iff (\textstyle\bigwedge \Sigma) \wedge (\textstyle\bigwedge \Sigma)' \models \varphi \to \varphi' \\
&\iff \models (\textstyle\bigwedge \Sigma) \to ((\textstyle\bigwedge \Sigma)' \to (\varphi \to \varphi')) \\
&\iff \models (\textstyle\bigwedge \Sigma) \to (\varphi \to ((\textstyle\bigwedge \Sigma)' \to \varphi')) \\
&\iff (\textstyle\bigwedge \Sigma) \wedge \varphi \models (\textstyle\bigwedge \Sigma)' \to \varphi') \\
&\iff (\textstyle\bigwedge \Sigma) \wedge \varphi \models (\textstyle\bigvee \neg\Sigma)' \vee \varphi' \\
&\iff \Sigma, \varphi \vdash (\neg\Sigma'), \varphi' \qquad \text{(due to soundness/completeness)}
\end{aligned}
$$

Not convenient: needs both Σ and negated Σ'!

$\Rightarrow$ 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)!

LK and Theories

$$
\begin{aligned}
\Sigma \cup \Sigma' \models \varphi \to \varphi' &\iff (\bigwedge \Sigma) \wedge (\bigwedge \Sigma)' \models \varphi \to \varphi' \\
&\iff \models (\bigwedge \Sigma) \to ((\bigwedge \Sigma)' \to (\varphi \to \varphi')) \\
&\iff \models (\bigwedge \Sigma) \to (\varphi \to ((\bigwedge \Sigma)' \to \varphi')) \\
&\iff (\bigwedge \Sigma) \wedge \varphi \models (\bigwedge \Sigma)' \to \varphi' \\
&\iff (\bigwedge \Sigma) \wedge \varphi \models (\bigvee \neg\Sigma)' \vee \varphi' \\
&\iff \Sigma, \varphi \vdash (\neg\Sigma'), \varphi' \qquad \text{(due to soundness/completeness)}
\end{aligned}
$$

Not convenient: needs both Σ and negated Σ'!

$\Rightarrow$ 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 $S \xrightarrow{int} \psi$; 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):
  - LL clash $S \cup \{P^L, \neg P^L\} \xrightarrow{int} \bot$ (similar for "RR": $\top$)
  - LR clash $S \cup \{P^L, \neg P^R\} \xrightarrow{int} P$, where $P \in S_A$
  - RL clash $S \cup \{\neg P^L, P^R\} \xrightarrow{int} \neg P$, where $P \in S_A$
  - L-conjunction $$\frac{S \cup \{\alpha^L, \beta^L\} \xrightarrow{int} \delta}{S \cup \{(\alpha \wedge \beta)^L\} \xrightarrow{int} \delta}$$
  - R-Disjunction $$\frac{S \cup \{\alpha^R\} \xrightarrow{int} \delta_\alpha \qquad S \cup \{\beta^R\} \xrightarrow{int} \delta_\beta}{S \cup \{(\alpha \vee \beta)^R\} \xrightarrow{int} \delta_\alpha \wedge \delta_\beta}$$
- etc. (see [Fitting] for details)

Tableau and Interpolant Extraction (by example)

- an interpolant $S \xrightarrow{int} \psi$; invariant $(\bigwedge S^L) \to \psi$ and $\psi \to (\neg \bigwedge S^R)$
  where S^L and S^R are the left/right subsets of S;
- tableau rules (sample):
  - LL clash $\boxed{S \cup \{P^L, \neg P^L\} \xrightarrow{int} \bot}$ (similar for “RR”: $\top$)
  - LR clash $\boxed{S \cup \{P^L, \neg P^R\} \xrightarrow{int} P}$, where $P \in \mathsf{S_A}$
  - RL clash $\boxed{S \cup \{\neg P^L, P^R\} \xrightarrow{int} \neg P}$, where $P \in \mathsf{S_A}$
  - L-conjunction $\dfrac{S \cup \{\alpha^L, \beta^L\} \xrightarrow{int} \delta}{S \cup \{(\alpha \wedge \beta)^L\} \xrightarrow{int} \delta}$
  - R-Disjunction $\dfrac{S \cup \{\alpha^R\} \xrightarrow{int} \delta_\alpha \qquad S \cup \{\beta^R\} \xrightarrow{int} \delta_\beta}{S \cup \{(\alpha \vee \beta)^R\} \xrightarrow{int} \delta_\alpha \wedge \delta_\beta}$
  - etc. (see [Fitting] for details)

University of Waterloo

Tableau and Interpolant Extraction (by example)

- an interpolant $S \xrightarrow{int} \psi$; invariant $(\bigwedge S^L) \to \psi$ and $\psi \to (\neg \bigwedge S^R)$
  where S^L and S^R are the left/right subsets of S;
- tableau rules (sample):
  - LL clash $\boxed{S \cup \{P^L, \neg P^L\} \xrightarrow{int} \bot}$ (similar for "RR": $\top$)
  - LR clash $\boxed{S \cup \{P^L, \neg P^R\} \xrightarrow{int} P}$, where $P \in \mathrm{S_A}$
  - RL clash $\boxed{S \cup \{\neg P^L, P^R\} \xrightarrow{int} \neg P}$, where $P \in \mathrm{S_A}$
  - L-conjunction $\dfrac{S \cup \{\alpha^L, \beta^L\} \xrightarrow{int} \delta}{S \cup \{(\alpha \wedge \beta)^L\} \xrightarrow{int} \delta}$
  - R-Disjunction $\dfrac{S \cup \{\alpha^R\} \xrightarrow{int} \delta_\alpha \qquad S \cup \{\beta^R\} \xrightarrow{int} \delta_\beta}{S \cup \{(\alpha \vee \beta)^R\} \xrightarrow{int} \delta_\alpha \wedge \delta_\beta}$
  - etc. (see [Fitting] for details)

University of Waterloo

Tableau and Interpolant Extraction (by example)

- an interpolant $S \xrightarrow{int} \psi$; 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):
  - LL clash $\boxed{S \cup \{P^L, \neg P^L\} \xrightarrow{int} \bot}$ (similar for "RR": $\top$)
  - LR clash $\boxed{S \cup \{P^L, \neg P^R\} \xrightarrow{int} P}$, where $P \in \mathrm{S_A}$
  - RL clash $\boxed{S \cup \{\neg P^L, P^R\} \xrightarrow{int} \neg P}$, where $P \in \mathrm{S_A}$
  - L-conjunction $$\frac{S \cup \{\alpha^L, \beta^L\} \xrightarrow{int} \delta}{S \cup \{(\alpha \wedge \beta)^L\} \xrightarrow{int} \delta}$$
  - R-Disjunction $$\frac{S \cup \{\alpha^R\} \xrightarrow{int} \delta_\alpha \qquad S \cup \{\beta^R\} \xrightarrow{int} \delta_\beta}{S \cup \{(\alpha \vee \beta)^R\} \xrightarrow{int} \delta_\alpha \wedge \delta_\beta}$$
  - etc. (see [Fitting] for details)

First-order Variables and Equality

- Quantifier rules

  1. inference rules with Ground constants/terms

     Quantifier Rules:

     $$\frac{\Gamma, \varphi(t/x) \vdash \Delta}{\Gamma, (\forall x.\varphi) \vdash \Delta}\ (\forall L) \qquad \frac{\Gamma \vdash \varphi(y/x), \Delta}{\Gamma \vdash (\forall x.\varphi), \Delta}\ (\forall R)$$

  2. unification tableau and Skolemization (refutation systems)

- Equality

  1. High-school Axioms (immediate implementation)

     $$\vdash x = x$$
     $$x = y \wedge \varphi \vdash \varphi(y/x)$$

  2. Superposition rules (efficient implementation)

First-order Variables and Equality

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

     Quantifier Rules:

     $$\frac{\Gamma, \varphi(t/x) \vdash \Delta}{\Gamma, (\forall x.\varphi) \vdash \Delta}\ (\forall L) \qquad \frac{\Gamma \vdash \varphi(y/x), \Delta}{\Gamma \vdash (\forall x.\varphi), \Delta}\ (\forall R)$$

  2. unification tableau and Skolemization (refutation systems)
- Equality
  1. High-school Axioms (immediate implementation)

     $$\vdash x = x$$
     $$x = y \wedge \varphi \vdash \varphi(y/x)$$

  2. Superposition rules (efficient implementation)

Issues with TABLEAU

Dealing with the *subformula property* of Tableau

$\Rightarrow$ analytic tableau *explores* formulas *structurally*

$\Rightarrow$ (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*

$\Rightarrow$ 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

$\Rightarrow$ analytic tableau *explores* formulas *structurally*

$\Rightarrow$ (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):

$$\Sigma^L \cup \Sigma^R \cup \Sigma^{LR} \models \varphi^L \to \varphi^R \text{ where } \Sigma^{LR} = \{\forall \bar{x}.P^L \leftrightarrow P \leftrightarrow P^R \mid P \in \mathsf{S_A}\}$$

Factoring *logical reasoning* from *plan enumeration*

$\Rightarrow$ 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_P

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

Absorbed Range-restricted Formulæ: ANF

$$Q ::= R(\bar{x}) \mid \bot \mid Q \wedge Q \mid Q \vee Q \mid \forall \bar{x}.R(\bar{x}) \to Q,$$

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

Conditional Tableau Rules for ANF

$$\frac{S \cup \{\varphi[C], \psi[C]\}}{(\varphi \wedge \psi)[C] \in S} \text{ (conj)} \qquad \frac{S \cup \{\varphi[C]\} \quad S \cup \{\psi[C]\}}{(\varphi \vee \psi)[C] \in S} \text{ (disj)}$$

$$\frac{S \cup \{(\varphi[\bar{t}/\bar{x}])[C \cup D]\}}{\{R(\bar{t})[C], (\forall \bar{x}.R(\bar{x}) \to \varphi)[D]\} \subseteq S} \text{ (abs)} \qquad \frac{S \cup \{R(\bar{t})[R(\bar{t})]\}}{S} \; R(\bar{x}) \in S_A \text{ (phys)}$$

University of Waterloo

Conditional Tableau and Interpolation

Conditional Tableau for $(Q, \Sigma, \mathsf{S_A})$

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

Closing Set(s)

We call a set C of literals over $\mathsf{S_A}$ 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
  - $\Rightarrow$ needs additional physical atoms that provide *bindings*
  - $\Rightarrow$ must appear in the tableau "on the correct side"
  - $\Rightarrow$ added to closing sets "soundly"

- Functionality (for duplicates)
  - $\Rightarrow$ needs additional physical atoms that *functionally determine quantified variables*
  - $\Rightarrow$ must appear in the tableau "on the correct side"
  - $\Rightarrow$ 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

P	:	L_P	R_P
$P(\bar{t})$	:	$\{\{\neg P^L(\bar{t})\}\}$	$\{\{P^R(\bar{t})\}\}$
$P_1 \wedge P_2$	:	$L_{P_1} \cup L_{P_2}$	$\{S_1 \cup S_2 \mid S_1 \in R_{P_1}, S_2 \in R_{P_2}\}$
$P_1 \vee P_2$	:	$\{S_1 \cup S_2 \mid S_1 \in L_{P_1}, S_2 \in L_{P_2}\}$	$R_{P_1} \cup R_{P_2}$
$\neg P_1$	:	$\{\{L^L(\bar{t}) \mid L^R(\bar{t}) \in S\} \mid S \in R_{P_1}\}$	$\{\{L^R(\bar{t}) \mid L^L(\bar{t}) \in S\} \mid S \in L_{P_1}\}$
$\exists x.P_1$	:	$L_{P_1[t/x]}$	$R_{P_1[t/x]}$

Observation

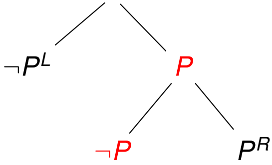
For a range-restricted formula P over S_A there is an analytic tableau tree T^P that uses only formulae in Σ^{LR} such that:

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.

Plan Enumeration

Physical Tableau T^P for a Plan P

P	:	L_P	R_P
$P(\bar{t})$	:	$\{\{\neg P^L(\bar{t})\}\}$	$\{\{P^R(\bar{t})\}\}$
$P_1 \wedge P_2$	:	$L_{P_1} \cup L_{P_2}$	$\{S_1 \cup S_2 \mid S_1 \in R_{P_1}, S_2 \in R_{P_2}\}$
$P_1 \vee P_2$	:	$\{S_1 \cup S_2 \mid S_1 \in L_{P_1}, S_2 \in L_{P_2}\}$	$R_{P_1} \cup R_{P_2}$
$\neg P_1$	:	$\{\{L^L(\bar{t}) \mid L^R(\bar{t}) \in S\} \mid S \in R_{P_1}\}$	$\{\{L^R(\bar{t}) \mid L^L(\bar{t}) \in S\} \mid S \in L_{P_1}\}$
$\exists x.P_1$	:	$L_{P_1[t/x]}$	$R_{P_1[t/x]}$



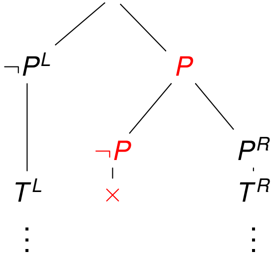
$(P^L \rightarrow P)$

$(P \rightarrow P^R)$

Plan Enumeration

Physical Tableau T^P for a Plan P

P	:	L_P	R_P
$P(\bar{t})$	:	$\{\{\neg P^L(\bar{t})\}\}$	$\{\{P^R(\bar{t})\}\}$
$P_1 \wedge P_2$	:	$L_{P_1} \cup L_{P_2}$	$\{S_1 \cup S_2 \mid S_1 \in R_{P_1}, S_2 \in R_{P_2}\}$
$P_1 \vee P_2$	:	$\{S_1 \cup S_2 \mid S_1 \in L_{P_1}, S_2 \in L_{P_2}\}$	$R_{P_1} \cup R_{P_2}$
$\neg P_1$	:	$\{\{L^L(\bar{t}) \mid L^R(\bar{t}) \in S\} \mid S \in R_{P_1}\}$	$\{\{L^R(\bar{t}) \mid L^L(\bar{t}) \in S\} \mid S \in L_{P_1}\}$
$\exists x.P_1$	:	$L_{P_1[t/x]}$	$R_{P_1[t/x]}$



$(P^L \rightarrow P)$

$(P \rightarrow P^R)$

interpolant: P

Plan Enumeration

Physical Tableau T^P for a Plan P

P	:	L_P	R_P
$P(\bar{t})$	:	$\{\{\neg P^L(\bar{t})\}\}$	$\{\{P^R(\bar{t})\}\}$
$P_1 \wedge P_2$	:	$L_{P_1} \cup L_{P_2}$	$\{S_1 \cup S_2 \mid S_1 \in R_{P_1}, S_2 \in R_{P_2}\}$
$P_1 \vee P_2$	:	$\{S_1 \cup S_2 \mid S_1 \in L_{P_1}, S_2 \in L_{P_2}\}$	$R_{P_1} \cup R_{P_2}$
$\neg P_1$	:	$\{\{L^L(\bar{t}) \mid L^R(\bar{t}) \in S\} \mid S \in R_{P_1}\}$	$\{\{L^R(\bar{t}) \mid L^L(\bar{t}) \in S\} \mid S \in L_{P_1}\}$
$\exists x.P_1$	:	$L_{P_1[t/x]}$	$R_{P_1[t/x]}$

Observation

For a range-restricted formula P over $\mathrm{S_A}$ there is an analytic tableau tree T^P that uses only formulæ in Σ^{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.

Logical&Physical Combined, Controlling the Search

Basic Strategy

1. build (T^L, T^R) for $(Q, \Sigma, \mathsf{S_A})$ 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;
   (…otherwise extend depth in step 1 and repeat.)

NOTE: in step 2 we can "test" many Ps (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 Ps) 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 $\{\cdot\}$:

$$
\begin{aligned}
Q[\{R(x_1,\ldots,x_k)\}] &\leftrightarrow Q[R(x_1,\ldots,x_k)] \\
Q[\{Q_1 \wedge Q_2\}] &\leftrightarrow Q[\{Q_1\} \wedge \{Q_2\}] \\
Q[\{\neg Q_1\}] &\leftrightarrow Q[\neg Q_1] \\
Q[\neg\{Q_1\}] &\leftrightarrow Q[\neg Q_1] \\
Q[\{Q_1 \vee Q_2\}] &\leftrightarrow Q[\{Q_1\} \vee \{Q_2\}] \quad \text{if } \Sigma \cup \{Q[]\} \models Q_1 \wedge Q_2 \to \bot \\
Q[\{\exists x.Q_1\}] &\leftrightarrow Q[\exists x.\{Q_1\}] \quad \text{if} \\
&\quad \Sigma \cup \{Q[] \wedge (Q_1)[y_1/x] \wedge (Q_1)[y_2/x]\} \models y_1 \approx y_2
\end{aligned}
$$

… reasoning abstracted: a DL $\mathcal{CFD}_{nc}^{\forall -}$ (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 $\{\cdot\}$:

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

… reasoning abstracted: a DL $\mathcal{CFD}^{\forall-}_{nc}$ (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 $\{\cdot\}$:

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

... reasoning abstracted: a DL $\mathcal{CFD}_{\text{nc}}^{\forall-}$ (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 $\mathcal{CFDI}_{\text{nc}}$ 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!