Ontology Based Data Access and Data Independence

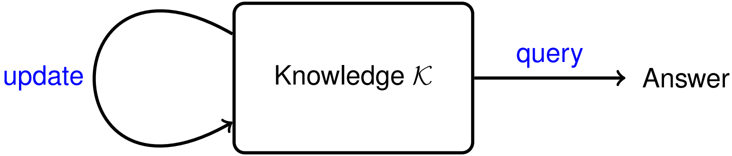
(an alternative look)

David Toman

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

Joint work with Alexander Hudek and Grant Weddell

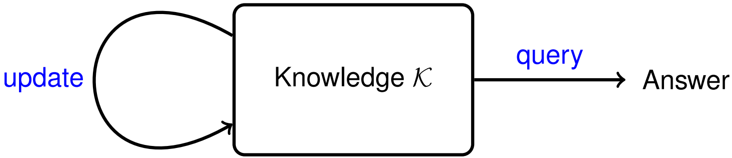
Knowledge Representation: a Big Picture



What is "Knowledge" (how is it represented, and does the user care?)

$\Rightarrow$ not really as long as the updates and queries "play nicely together"

Knowledge Representation: a Big Picture



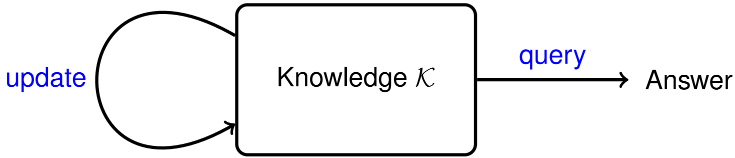
What is “Knowledge” (how is it represented, and does the user care?)

$\Rightarrow$ not really as long as the updates and queries “play nicely together”

Structured World:

- $\mathcal{K}$ is a (first order) theory,
- queries are (FO) formulæ with answers defined by entailment, and
- updates are (variations on) belief revision.

Knowledge Representation: a Big Picture



What is "Knowledge" (how is it represented, and does the user care?)

$\Rightarrow$ not really as long as the updates and queries "play nicely together"

Probabilistic World:

- $\mathcal{K}$ is a ML model (e.g., neural net),
- queries are inputs (e.g., photos) and answers are labels
- updates are pairs of, e.g., photos with their labels.

Ontology-based Data Access (OBDA) [Calvanese et al.: Mastro, 2011]

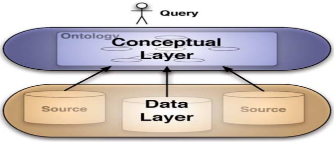
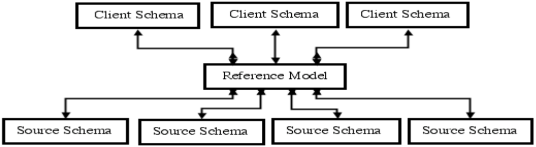


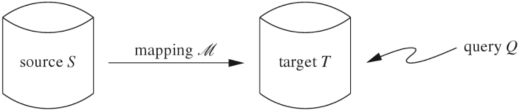
Fig. 1. Ontology-based data access.

Information Integration [Genesereth: Data Integration, 2010]

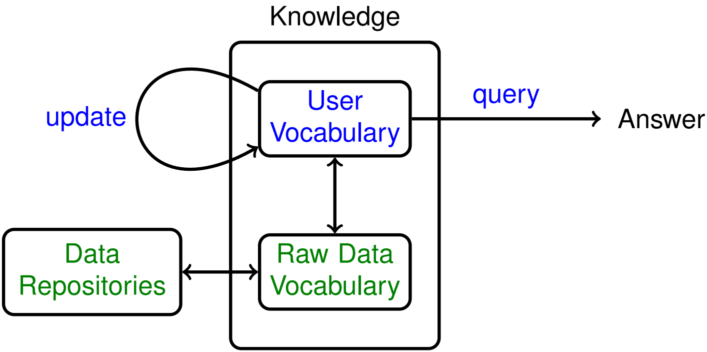


Data Exchange [Arenas et el.: Data Exchange, 2014]

The general setting of data exchange is this:



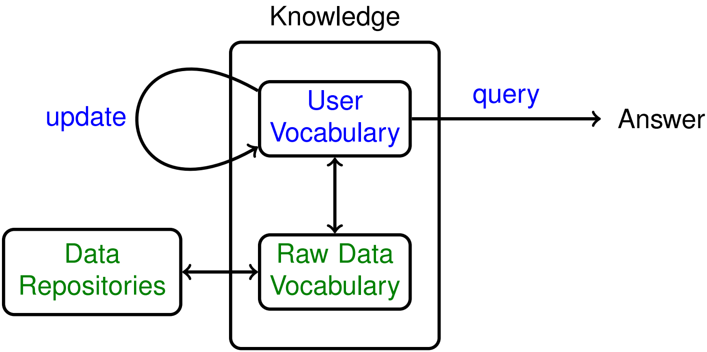
Data vs. Metadata



1. Metadata: constraints formulated in FOL (static)
2. Data: ground tuples (can be "modified")

   $\Rightarrow$ user queries and updates only about data.

Data vs. Metadata



1. Metadata: constraints formulated in FOL (static)
2. Data: ground tuples (can be “modified”)

$\Rightarrow$ user queries and updates only about data.

(Physical) Data Independence

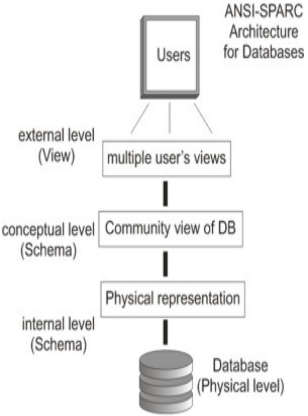
IDEA:

Separate the users' view(s) of the data from the way it is physically represented.

Originally just two levels: conceptual/logical and physical [Codd1970].

Physical Data Independence and ADTs

- data independence [Codd, 1970] and [Bachman, 1969, Date and Hopewell, 1971]
- ADTs [Liskov and Zilles, 1974]



[ANSI/X3/SPARC Standards Planning and Requirements Committee, Bachman, 1975]

(Physical) Data Independence

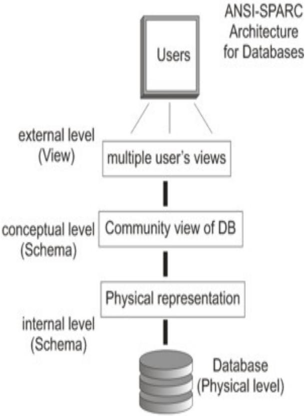
IDEA:

Separate the users' view(s) of the data from the way it is physically represented.

Originally just two levels: conceptual/logical and physical [Codd1970].

Physical Data Independence and ADTs

- data independence [Codd, 1970] and [Bachman, 1969, Date and Hopewell, 1971]
- ADTs [Liskov and Zilles, 1974]



[ANSI/X3/SPARC Standards Planning and Requirements Committee, Bachman, 1975]

(Physical) Data Independence

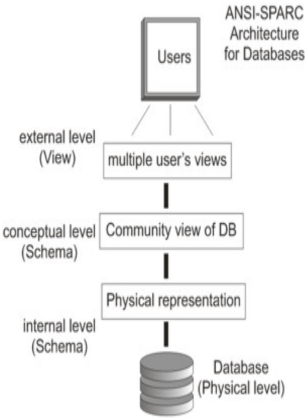
IDEA:

Separate the users' view(s) of the data from the way it is physically represented.

Originally just two levels: conceptual/logical and physical [Codd1970].

Physical Data Independence and ADTs

- data independence [Codd, 1970] and [Bachman, 1969, Date and Hopewell, 1971]
- ADTs [Liskov and Zilles, 1974]



[ANSI/X3/SPARC Standards Planning and Requirements Committee, Bachman, 1975]

Common Threads and Issues

- In general two *schemas*: Conceptual/Logical and Physical

  ⇒ both endowed with *metadata* (vocabulary, constraints, . . .)

  ⇒ mappings that connect the two schemas

- Rules: (1) queries (updates) over the *conceptual/logical* schema only,
  (2) raw data as instance of the *physical* schema only,
  (3) *conceptual/logical* and *physical* schemata are *disjoint*.

- Issues to be formalized/fixed:
  1. Formal description of the two schemas (same formalism for both?)
  2. Language(s) for metadata, mappings, etc.
  3. User Interface:
     - Data Model
     - Query (and Update) Language (e.g., when is an answer really an answer?)

- Algorithms/Execution model for user queries and updates

Common Threads and Issues

- In general two *schemas*: Conceptual/Logical and Physical

  ⇒ both endowed with *metadata* (vocabulary, constraints, . . .)

  ⇒ mappings that connect the two schemas

- Rules: (1) queries (updates) over the *conceptual/logical* schema only,
  (2) raw data as instance of the *physical* schema only,
  (3) *conceptual/logical* and *physical* schemata are *disjoint*.

- Issues to be formalized/fixed:

  1. Formal description of the two schemas (same formalism for both?)
  2. Language(s) for metadata, mappings, etc.
  3. User Interface:
     - Data Model
     - Query (and Update) Language (e.g., when is an answer really an answer?)

- Algorithms/Execution model for user queries and updates

Common Threads and Issues

- In general two *schemas*: Conceptual/Logical and Physical

  $\Rightarrow$ both endowed with *metadata* (vocabulary, constraints, ...)

  $\Rightarrow$ mappings that connect the two schemas

- Rules: (1) queries (updates) over the *conceptual/logical* schema only,
  (2) raw data as instance of the *physical* schema only,
  (3) *conceptual/logical* and *physical* schemata are *disjoint*.

- Issues to be formalized/fixed:

  1. Formal description of the two schemas (same formalism for both?)
  2. Language(s) for metadata, mappings, etc.
  3. User Interface:
     - Data Model
     - Query (and Update) Language (e.g., when is an answer really an answer?)

- Algorithms/Execution model for user queries and updates

Outline

Queries and Query Answering

The Structured/Logical Way (via an OBDA example)

Queries and Ontologies

Queries are answered not only w.r.t. *explicit data* ($\mathcal{A}$)
but also w.r.t. *background knowledge* ($\mathcal{T}$)
$\Rightarrow$ Ontology-based Data Access (OBDA)

Example

- Socrates is a MAN (explicit data)
- Every MAN is MORTAL (ontology)

List all MORTALs $\Rightarrow$ {Socrates} (query)

How do we answer queries?

Using *logical implication* (to define *certain answers*):

$$\mathrm{Ans}(\varphi, \mathcal{A}, \mathcal{T}) := \{\varphi(a_1, \ldots, a_k) \mid \mathcal{T} \cup \mathcal{A} \models \varphi(a_1, \ldots, a_k)\}$$

$\Rightarrow$ answers are *ground* φ*-atoms* logically implied by $\mathcal{A} \cup \mathcal{T}$.

The Structured/Logical Way (via an OBDA example)

Queries and Ontologies

Queries are answered not only w.r.t. *explicit data* ($\mathcal{A}$)
but also w.r.t. *background knowledge* ($\mathcal{T}$)
$\Rightarrow$ Ontology-based Data Access (OBDA)

Example

- Socrates is a MAN (explicit data)
- Every MAN is MORTAL (ontology)

List all MORTALs $\Rightarrow$ $\{$Socrates$\}$ (query)

How do we answer queries?

Using *logical implication* (to define *certain answers*):

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

$\Rightarrow$ answers are *ground φ-atoms* logically implied by $\mathcal{A} \cup \mathcal{T}$.

The Logical Way: Complexity

How do we answer queries?

Using *logical implication* (to define *certain answers*):

$$\mathrm{Ans}(\varphi, \mathcal{A}, \mathcal{T}) := \{\varphi(a_1, \ldots, a_k) \mid \mathcal{T} \cup \mathcal{A} \models \varphi(a_1, \ldots, a_k)\}$$

$\Rightarrow$ answers are *ground φ-atoms* logically implied by $\mathcal{A} \cup \mathcal{T}$.

The Bad News

- undecidable in general
- coNP-complete (data complexity) from $\mathcal{ELI}$ up to $\mathcal{SROIQ}$ and (U)CQs

The Good News

LOGSPACE/PTIME (data complexity) for query answering:

- DL-Lite/$\mathcal{EL}_\perp$/$\mathcal{CFD}^{\forall}_{nc}$/"rules"-lite (Horn), s-t dependencies, ...
- and (still only) (U)CQ

The Logical Way: Complexity

How do we answer queries?

Using *logical implication* (to define *certain answers*):

$$\mathrm{Ans}(\varphi, \mathcal{A}, \mathcal{T}) := \{\varphi(a_1, \ldots, a_k) \mid \mathcal{T} \cup \mathcal{A} \models \varphi(a_1, \ldots, a_k)\}$$

$\Rightarrow$ answers are *ground φ-atoms* logically implied by $\mathcal{A} \cup \mathcal{T}$.

The Bad News

- undecidable in general
- coNP-complete (data complexity) from $\mathcal{ELI}$ up to $\mathcal{SROIQ}$ and (U)CQs

The Good News

LOGSPACE/PTIME (data complexity) for query answering:

- DL-Lite/$\mathcal{EL}_\perp$/$\mathcal{CFD}^{\forall}_{nc}$/"rules"-lite (Horn), s-t dependencies, ...
- and (still only) (U)CQ

The Logical Way: Complexity

How do we answer queries?

Using *logical implication* (to define *certain answers*):

$$\mathrm{Ans}(\varphi, \mathcal{A}, \mathcal{T}) := \{\varphi(a_1, \ldots, a_k) \mid \mathcal{T} \cup \mathcal{A} \models \varphi(a_1, \ldots, a_k)\}$$

$\Rightarrow$ answers are *ground φ-atoms* logically implied by $\mathcal{A} \cup \mathcal{T}$.

The Bad News

- undecidable in general
- coNP-complete (data complexity) from $\mathcal{ELI}$ up to $\mathcal{SROIQ}$ and (U)CQs

The Good News

LOGSPACE/PTIME (data complexity) for query answering:

- DL-Lite/$\mathcal{EL}_\perp$/$\mathcal{CFD}^{\forall}_{\mathrm{nc}}$/“rules”-lite (Horn), s-t dependencies,...
- and (still only) (U)CQ

Approaches to Ontology-based Data Access

Main Task

INPUT: $\underbrace{\text{Ontology } (\mathcal{T}), \text{ Data } (\mathcal{A})}_{\text{Knowledge Base}(\mathcal{K})}$, and a Query ($Q$)

OUTPUT: $\{a \mid \mathcal{K} \models Q[a]\}$

1. Reduction to *standard reasoning* (e.g., satisfiability)
2. Reduction to *querying a relational database*

   $\Rightarrow$ *very* good at $\{a \mid \mathcal{A} \models Q[a]\}$ for range restricted Q

   $\Rightarrow$ what to do with $\mathcal{T}$??

   1. incorporate into Q (perfect rewriting for DL-Lite et al. (AC^0 logics)); or
   2. incorporate into $\mathcal{A}$ (combined approach for $\mathcal{EL}$ (PTIME-complete logics));

      or sometimes both ($\mathcal{CFDI}$ or Horn-$\mathcal{ALC}*$ logics).

Combined Combined Approach to OMQ

Theorem (Combined Combined Approach [Eiter et al., 2012] [Toman and Weddell, 2013])

Let $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ be a Horn-$\mathcal{SHIQ}$ (or Horn-$\mathcal{DLFD}$) knowledge base and φ a conjunctive query. Then there is

1. *a FO (typically UCQ) query $\varphi_{\mathcal{T}}$, called a query rewriting, and*
2. *a Datalog program $\Pi_{\mathcal{T}}$, called an ABox completion program,*

such that

$$\mathcal{K} \models \varphi[\vec{x} \mapsto \vec{a}] \iff \Pi_{\mathcal{T}}(\mathcal{A}) \models \varphi_{\mathcal{T}}[\vec{x} \mapsto \vec{a}]$$

where $\Pi_{\mathcal{T}}(\mathcal{A})$ is the minimal model of $\Pi_{\mathcal{T}}$ over $\mathcal{A}$, the completion of $\mathcal{A}$ w.r.t. $\mathcal{T}$.

Perfect Reformulation [Calvanese et al., 2007]:

$\Pi_{\mathcal{T}}$ is an identity on $\mathcal{A}$;

Combined Approach [Lutz et al., 2009, Kontchakov et al., 2010]:

$\varphi_{\mathcal{T}}$ does not depend on $\mathcal{T}$.

Combined Combined Approach to OMQ

Theorem (Combined Combined Approach [Eiter et al., 2012] [Toman and Weddell, 2013])

Let $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ be a Horn-$\mathcal{SHIQ}$ (or Horn-$\mathcal{DLFD}$) knowledge base and φ a conjunctive query. Then there is

1. *a FO (typically UCQ) query $\varphi_{\mathcal{T}}$, called a query rewriting, and*
2. *a Datalog program $\Pi_{\mathcal{T}}$, called an ABox completion program,*

such that

$$\mathcal{K} \models \varphi[\vec{x} \mapsto \vec{a}] \iff \Pi_{\mathcal{T}}(\mathcal{A}) \models \varphi_{\mathcal{T}}[\vec{x} \mapsto \vec{a}]$$

where $\Pi_{\mathcal{T}}(\mathcal{A})$ is the minimal model of $\Pi_{\mathcal{T}}$ over $\mathcal{A}$, the completion of $\mathcal{A}$ w.r.t. $\mathcal{T}$.

Perfect Reformulation [Calvanese et al., 2007]:

$\Pi_{\mathcal{T}}$ is an identity on $\mathcal{A}$;

Combined Approach [Lutz et al., 2009, Kontchakov et al., 2010]:

$\varphi_{\mathcal{T}}$ does not depend on $\mathcal{T}$.

The *really* BAD news for OBDA

- no negative queries/sub-queries (SQL??)
- no negations in ABox
- **no closed-world assumption**
- **counter-intuitive query answers**

⇒ the same goes for *information integration, data exchange*, etc.

The *really* BAD news for OBDA

- no negative queries/sub-queries (SQL??)
- no negations in ABox
- **no closed-world assumption**
- **counter-intuitive query answers**

$\Rightarrow$ the same goes for *information integration*, *data exchange*, etc.

Difficulties: (Data) Complexity

Example (Why can't we have CWA?)

- relations: "$ColNode(x, y)$" and "$Edge(x, y)$";
- the data: graph $(Node^{\mathcal{I}}, Edge^{\mathcal{I}})$, and $Colour^{\mathcal{I}} = \{r, g, b\}$.
- ontology: $\forall x.Node(x) \to \exists y.ColNode(x, y)$;
  $\forall x, y.ColNode(x, y) \to Colour(y)$;
- query: $\exists x, y, z.Edge(x, y) \land ColNode(x, z) \land ColNode(y, z)$

PROBLEM: having *exactly* 3 colours is not in DL-Lite (or even HORN).

$\Rightarrow$ same examples for other non-Horn features, e.g., disjunction

Difficulties: (Data) Complexity

Example (Why can't we have CWA?)

- relations: "$ColNode(x, y)$" and "$Edge(x, y)$";
- the data: graph $(Node^{\mathcal{I}}, Edge^{\mathcal{I}})$, and $Colour^{\mathcal{I}} = \{r, g, b\}$.
- ontology: $\forall x.Node(x) \rightarrow \exists y.ColNode(x, y)$,
  $\forall x, y.ColNode(x, y) \rightarrow Colour(y)$;
- query: $\exists x, y, z.Edge(x, y) \wedge ColNode(x, z) \wedge ColNode(y, z)$

PROBLEM: having *exactly* 3 colours is not in DL-Lite (or even HORN).

$\Rightarrow$ same examples for other non-Horn features, e.g., disjunction

Difficulties: (Data) Complexity

Example (Why can't we have CWA?)

- relations: "$ColNode(x, y)$" and "$Edge(x, y)$";
- the data: graph $(Node^{\mathcal{I}}, Edge^{\mathcal{I}})$, and $Colour^{\mathcal{I}} = \{r, g, b\}$.
- ontology: $\forall x.Node(x) \rightarrow \exists y.ColNode(x, y)$,
  $\forall x, y.ColNode(x, y) \rightarrow Colour(y)$;
- query: $\exists x, y, z.Edge(x, y) \wedge ColNode(x, z) \wedge ColNode(y, z)$

  ...says "the graph $(Node^{\mathcal{I}}, Edge^{\mathcal{I}})$ is NOT 3-colourable"

PROBLEM: having *exactly* 3 colours is not in DL-Lite (or even HORN).

$\Rightarrow$ same examples for other non-Horn features, e.g., disjunction

Difficulties: (Data) Complexity

Example (Why can't we have CWA?)

- relations: "$ColNode(x, y)$" and "$Edge(x, y)$";
- the data: graph $(Node^{\mathcal{I}}, Edge^{\mathcal{I}})$, and $Colour^{\mathcal{I}} = \{r, g, b\}$.
- ontology: $\forall x.Node(x) \rightarrow \exists y.ColNode(x, y)$,
  $\forall x, y.ColNode(x, y) \rightarrow Colour(y)$;
- query: $\exists x, y, z.Edge(x, y) \wedge ColNode(x, z) \wedge ColNode(y, z)$

  ...says "the graph $(Node^{\mathcal{I}}, Edge^{\mathcal{I}})$ is NOT 3-colourable"

PROBLEM: having *exactly* 3 colours is not in DL-Lite (or even HORN).

$\Rightarrow$ same examples for other non-Horn features, e.g., disjunction

(First-order) Query Rewritability

Rewritability (Decision Problem)

Given:

1. a TBox $\mathcal{T}$ and
2. a Query φ.

Decide whether there is a FO query ψ such that

$$\text{Ans}(\varphi, \mathcal{A}, \mathcal{T}) = \text{Ans}(\psi, \mathcal{A}, \emptyset)$$

for every ABox $\mathcal{A}$ (optionally where ψ is over a sub-vocabulary of $\mathcal{T}$).

[Bienvenu, Lutz, Wolter: First-Order Rewritability of Atomic Queries in Horn Description Logics. IJCAI 2013. (and many papers followed...)]

Difficulties: Unintuitive Answers

Example

- $EMP(Sue)$
- $EMP \sqsubseteq \exists PHONENUM$ (or $\forall x.EMP(x) \rightarrow \exists y.PHONENUM(x, y)$)

Difficulties: Unintuitive Answers

Example

- $EMP(Sue)$
- $EMP \sqsubseteq \exists PHONENUM \quad (\text{or } \forall x.EMP(x) \rightarrow \exists y.PHONENUM(x,y))$

User: *Does Sue have a phone number?*

Information System: *YES*

Difficulties: Unintuitive Answers

Example

- $EMP(Sue)$
- $EMP \sqsubseteq \exists PHONENUM$ (or $\forall x.EMP(x) \rightarrow \exists y.PHONENUM(x,y)$)

User: *Does Sue have a phone number?*

Information System: *YES*

User: *OK, tell me Sue's phone number!*

Information System: *(no answer)*

Difficulties: Unintuitive Answers

Example

- $EMP(Sue)$
- $EMP \sqsubseteq \exists PHONENUM$ (or $\forall x.EMP(x) \rightarrow \exists y.PHONENUM(x,y)$)

User: *Does Sue have a phone number?*

Information System: *YES*

User: *OK, tell me Sue's phone number!*

Information System: *(no answer)*

User: 

The problem: Users (essentially) EXPECT CWA

What does $\mathcal{A} = \{EMP(Bob), EMP(Sue)\}$ mean?

OWA: $Bob^{\mathcal{I}} \in EMP^{\mathcal{I}}, Sue^{\mathcal{I}} \in EMP^{\mathcal{I}}$ (KR folks)

CWA: $\{Bob^{\mathcal{I}}, Sue^{\mathcal{I}}\} = EMP^{\mathcal{I}}$ (DB folks and users)

. . . at least for *their* relations (i.e., in the conceptual schema).

The problem: Users (essentially) EXPECT CWA

What does $\mathcal{A} = \{EMP(Bob), EMP(Sue)\}$ mean?

OWA: $Bob^{\mathcal{I}} \in EMP^{\mathcal{I}}, Sue^{\mathcal{I}} \in EMP^{\mathcal{I}}$ (KR folks)

CWA: $\{Bob^{\mathcal{I}}, Sue^{\mathcal{I}}\} = EMP^{\mathcal{I}}$ (DB folks and users)

. . . at least for *their* relations (i.e., in the conceptual schema).

Simulations:

CWA in OWA: *closure axioms*: $\forall x.EMP(x) \rightarrow (x = Bob) \vee (x = Sue)$;

OWA in CWA: *auxiliary symbols*: $ExpEMP(Bob), ExpEMP(Sue)$
and *constraints*: $\forall x.ExpEMP(x) \rightarrow EMP(x)$

What does a User Want? . . . but is afraid to ask

1. what I know and what I don't is just a single model (CWA);
2. queries are model-checked against this model;
3. updates change the model into another single model.

What does a User Want? . . . but is afraid to ask

1. what I know and what I don't is just a single model (CWA);
2. queries are model-checked against this model;
3. updates change the model into another single model.

YES, BUT:

- it better run fast!!
  $\Rightarrow$ preferably without having to code algorithms/data structures by hand
- algorithms/performance/data storage-representation/. . .
  can be changed *without changes to user queries/updates*

User Queries and Updates – for TODAY

Queries: First-order (open) formulae over the user vocabulary

$\Rightarrow$ only *range-restricted* formulae

(i.e., with appropriate *binding pattern restrictions*)

Updates: Instances of *delta-relations* (tuples to be inserted/deleted) for ALL relations in the user vocabulary

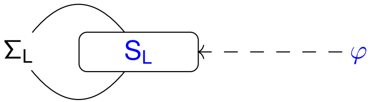
$\Rightarrow$ only *consistency-preserving transactions* allowed

. . . a.k.a. the Relational Model and Relational Calculus [Codd, 1972].

Rewritability and Definability

User and System Expectations

Queries	range-restricted FOL (a.k.a. SQL)
Ontology/Schema	range-restricted FOL $\Sigma := \Sigma_L \cup \Sigma_{LP} \cup \Sigma_P$
Data	CWA (complete information)

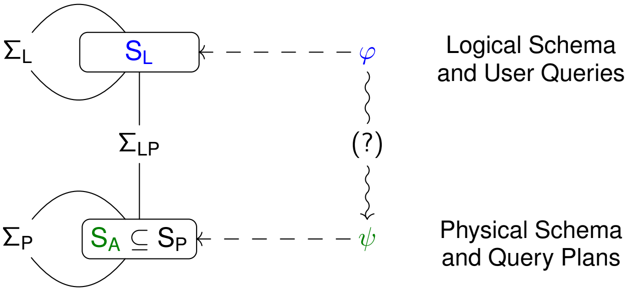


Logical Schema
and User Queries

Rewritability and Definability

User and System Expectations

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



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

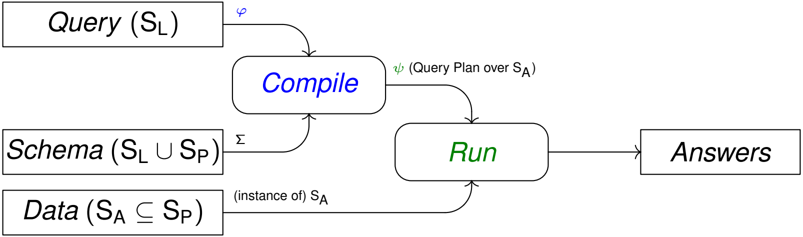
Rewritability and Definability

User and System Expectations

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

- to users it looks like a *single model* (of the logical schema)
- implementation can pick from many models

but *definable* queries answer the same in each of them



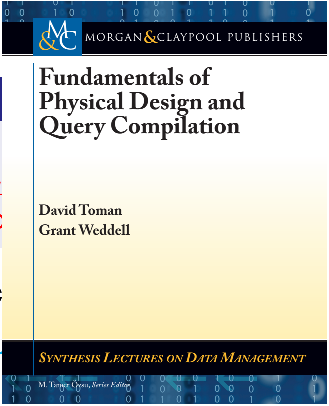
Rewritability and Definability

User and System Expectations

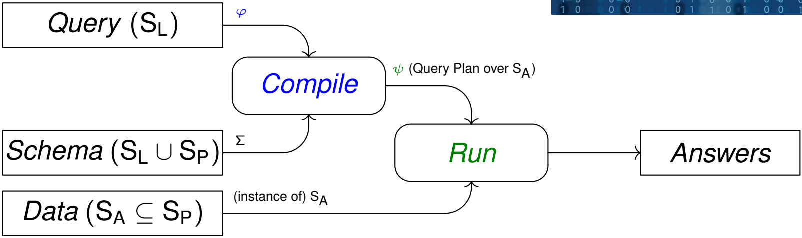
Queries	range-restricted FOL	over S_L
Ontology/Schema	range-restricted FOL	$\Sigma := \Sigma_L$
Data	CWA (complete information	fo

- to users it looks like a *single model* (of the logic
- implementation can pick from many models

but *definable* queries answer



©2011



This is NOT OMQ/OBDA (by example)

Example (Horizontal Partition)

$$\mathsf{S_L} = \{\mathtt{emp}/1, \mathtt{wkr}/1, \mathtt{mgr}/1\} \text{ and } \Sigma_\mathsf{L} = \left\{ \begin{array}{l} \mathtt{mgr}(x) \vee \mathtt{wkr}(x) \leftrightarrow \mathtt{emp}(x) \\ \mathtt{mgr}(x) \wedge \mathtt{wkr}(x) \to \bot \end{array} \right\}$$

This is NOT OMQ/OBDA (by example)

Example (Horizontal Partition)

$\mathsf{S_L} = \{\mathtt{emp}/1, \mathtt{wkr}/1, \mathtt{mgr}/1\}$ and $\Sigma_\mathsf{L} = \left\{ \begin{array}{l} \mathtt{mgr}(x) \lor \mathtt{wkr}(x) \leftrightarrow \mathtt{emp}(x) \\ \mathtt{mgr}(x) \land \mathtt{wkr}(x) \to \bot \end{array} \right\}$

$\mathsf{S_A} = \{\mathtt{emp}/1,\ \mathtt{mgr}/1\}$, note that $\mathtt{wkr}/1$ is NOT in $\mathsf{S_A}$;

This is NOT OMQ/OBDA (by example)

Example (Horizontal Partition)

$\mathsf{S_L} = \{\mathtt{emp}/1, \mathtt{wkr}/1, \mathtt{mgr}/1\}$ and $\Sigma_\mathsf{L} = \left\{ \begin{array}{l} \mathtt{mgr}(x) \vee \mathtt{wkr}(x) \leftrightarrow \mathtt{emp}(x) \\ \mathtt{mgr}(x) \wedge \mathtt{wkr}(x) \rightarrow \bot \end{array} \right\}$

$\mathsf{S_A} = \{\mathtt{emp}/1,\ \mathtt{mgr}/1\}$, note that $\mathtt{wkr}/1$ is NOT in $\mathsf{S_A}$;

Query $\{x \mid \mathtt{wkr}(x)\}$ over an instance $\mathtt{mgr} = \{\mathrm{Fred}\}$, and $\mathtt{emp} = \{\mathrm{Fred}, \mathrm{Wilma}\}$

This is NOT OMQ/OBDA (by example)

Example (Horizontal Partition)

$\mathsf{S_L} = \{\mathtt{emp}/1, \mathtt{wkr}/1, \mathtt{mgr}/1\}$ and $\Sigma_\mathsf{L} = \left\{ \begin{array}{l} \mathtt{mgr}(x) \vee \mathtt{wkr}(x) \leftrightarrow \mathtt{emp}(x) \\ \mathtt{mgr}(x) \wedge \mathtt{wkr}(x) \rightarrow \bot \end{array} \right\}$

$\mathsf{S_A} = \{\mathtt{emp}/1,\ \mathtt{mgr}/1\}$, note that $\mathtt{wkr}/1$ is NOT in $\mathsf{S_A}$;

Query $\{x \mid \mathtt{wkr}(x)\}$ over an instance $\mathtt{mgr} = \{\mathrm{Fred}\}$, and $\mathtt{emp} = \{\mathrm{Fred}, \mathrm{Wilma}\}$

Certain Answer under OWA: $\{\ \}$

Answer under CWA: $\{\mathrm{Wilma}\}$

(obtained by executing the *plan* $\{x \mid \mathtt{emp}(x) \wedge \neg\mathtt{mgr}(x)\}$).

What can we do with this?

Goal #2

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

1. standard RDBMS physical designs (and more),
   - access to search structures (index access and selection),
   - horizontal partitioning/sharding,
   - column store/index-only plans,
2. pointer-based data structures (including main mamory),
3. hash-based access to data (including hash-joins),
4. multi-level storage (aka disk/remote/distributed files), . . .
5. materialized views,
6. updates through logical schema
7. . . .

. . . all without having to code (too much) in C/C++ !

Standard Physical Designs

1. scanning (flat) files
2. primary and secondary indices (via record ids/addresses)
3. horizontal partitioning/sharding
4. column store/index-only plans
5. (disjoint) generalizations

Pointers in Main Memory-Logical Schema

```
CREATE TABLE employee (
  num     INTEGER NOT NULL,
  name    CHAR(20),
  worksin INTEGER  NOT NULL
  PRIMARY KEY (num),
  FOREIGN KEY (worksin)
        REFERENCES department
)
```

```
CREATE TABLE department (
  num     INTEGER NOT NULL,
  name    CHAR(50),
  manager INTEGER NOT NULL,
  PRIMARY KEY (num),
  FOREIGN KEY (manager)
        REFERENCES employee
)
```

this corresponds to

- $S_L = \{\texttt{employee}/3, \texttt{department}/3\}$ and
- $\Sigma_L = \{\, \texttt{employee}(x, y_1, z_1) \wedge \texttt{employee}(x, y_2, z_2) \to y_1 = y_2 \wedge z_1 = z_2,$
  $\texttt{employee}(x, y, z) \to \exists u, v.\texttt{department}(z, u, v), \;\ldots$ and many more $\}$.

Pointers in Main Memory-Logical Schema

```
CREATE TABLE employee (
  num     INTEGER NOT NULL,
  name    CHAR(20),
  worksin INTEGER  NOT NULL
  PRIMARY KEY (num),
  FOREIGN KEY (worksin)
        REFERENCES department
)
```

```
CREATE TABLE department (
  num     INTEGER NOT NULL,
  name    CHAR(50),
  manager INTEGER NOT NULL,
  PRIMARY KEY (num),
  FOREIGN KEY (manager)
        REFERENCES employee
)
```

this corresponds to

- $S_L = \{\texttt{employee}/3, \texttt{department}/3\}$ and
- $\Sigma_L = \{\, \texttt{employee}(x, y_1, z_1) \wedge \texttt{employee}(x, y_2, z_2) \to y_1 = y_2 \wedge z_1 = z_2,$
  $\texttt{employee}(x, y, z) \to \exists u, v.\texttt{department}(z, u, v), \ \ldots$ and many more $\}$.

additional logical constraints (for example):

- managers are employees that manage a department (a view)
- managers work in their own departnemts (business rule)
- workers and managers partition employees (partition), etc.

Pointers in Main Memory-Physical Design

1. Records:

```
struct emp {                          struct dept {
    int      num;                         int      num;
    char[20] name;                        char[50] name;
    dept*    dept; };                     mgr*     emp;   };
```

2. a linked list of `emp` records.

that corresponds to

- Access paths (S_A):
  - `empfile`/1/0: set (list) of *addresses* of `emp` records;
  - `emp-num`/2/1: pairs `emp` record address-`emp` number (pointer navigation)
    same for `emp-name`/2/1 and `emp-dept`/2/1;
  - `dept-num`/2/1: pairs `dept` record address-`dept` number
    same for `dept-name`/2/1 and `dept-mgr`/2/1.
- Integrity constraints ($\Sigma_P \cup \Sigma_{LP}$):

$$\forall x, y, z.\mathtt{employee}(x, y, z) \rightarrow \exists w.\mathtt{empfile}(w) \wedge \mathtt{emp\text{-}num}(w, x),$$
$$\forall a, x.\mathtt{empfile}(a) \wedge \mathtt{emp\text{-}num}(a, x) \rightarrow \exists y, z.\mathtt{employee}(x, y, z), \ldots$$

Query Plans that Navigate Pointers

1. List employee numbers, names, and departments ($\texttt{employee}(x, y, z)$):

$$\exists e, d.\texttt{empfile}(e) \wedge \texttt{emp-num}(e, x) \wedge \texttt{emp-name}(e, y) \wedge \texttt{emp-dept}(e, d) \wedge \texttt{dept-num}(d, z)$$

or, in C-like syntax:

```
for e in empfile do
    x := e->num;
    y := e->name;
    d := e->dept;
    z := d->num;
```

Query Plans that Navigate Pointers

1. List employee numbers, names, and departments ($\texttt{employee}(x, y, z)$):

$$\exists e, d.\texttt{empfile}(e) \wedge \texttt{emp-num}(e, x) \wedge \texttt{emp-name}(e, y) \wedge \texttt{emp-dept}(e, d) \wedge \texttt{dept-num}(d, z)$$

2. List worker numbers and names ($\exists z.\texttt{worker}(x, y, z)$):

$$\exists e, d.\texttt{empfile}(e) \wedge \texttt{emp-num}(e, x) \wedge \texttt{emp-name}(e, y) \wedge \texttt{emp-dept}(e, d) \wedge \neg\texttt{dept-mgr}(d, e)$$

Query Plans that Navigate Pointers

1. List employee numbers, names, and departments ($\texttt{employee}(x, y, z)$):

$$\exists e, d.\texttt{empfile}(e) \land \texttt{emp-num}(e, x) \land \texttt{emp-name}(e, y) \land \texttt{emp-dept}(e, d) \land \texttt{dept-num}(d, z)$$

2. List worker numbers and names ($\exists z.\texttt{worker}(x, y, z)$):

$$\exists e, d.\texttt{empfile}(e) \land \texttt{emp-num}(e, x) \land \texttt{emp-name}(e, y) \land \texttt{emp-dept}(e, d) \land \neg\texttt{dept-mgr}(d, e)$$

3. List all department numbers and their names ($\exists z.\texttt{department}(x, y, z)$):

$\gg$ Caveat: we do NOT have a (direct) way to "scan" `depatment`s! $\ll$

Query Plans that Navigate Pointers

1. List employee numbers, names, and departments ($\texttt{employee}(x, y, z)$):

$$\exists e, d.\texttt{empfile}(e) \land \texttt{emp-num}(e, x) \land \texttt{emp-name}(e, y) \land \texttt{emp-dept}(e, d) \land \texttt{dept-num}(d, z)$$

2. List worker numbers and names ($\exists z.\texttt{worker}(x, y, z)$):

$$\exists e, d.\texttt{empfile}(e) \land \texttt{emp-num}(e, x) \land \texttt{emp-name}(e, y) \land \texttt{emp-dept}(e, d) \land \neg\texttt{dept-mgr}(d, e)$$

3. List all department numbers and their names ($\exists z.\texttt{department}(x, y, z)$):

$$\exists d, e.\texttt{empfile}(e) \land \texttt{emp-dept}(e, d) \land \texttt{dept-num}(d, x) \land \texttt{dept-name}(d, y)$$

$\Rightarrow$ needs "departments have at least one employee".

$$\exists e, d.\texttt{empfile}(e) \land \texttt{emp-dept}(e, d) \land \texttt{dept-num}(d, x) \land \texttt{dept-name}(d, y) \land \texttt{dept-mgr}(d, e)$$

$\Rightarrow$ needs "managers work in their own departments".

Query Plans that Navigate Pointers

1. List employee numbers, names, and departments ($\texttt{employee}(x, y, z)$):

$$\exists e, d.\texttt{empfile}(e) \wedge \texttt{emp-num}(e, x) \wedge \texttt{emp-name}(e, y) \wedge \texttt{emp-dept}(e, d) \wedge \texttt{dept-num}(d, z)$$

2. List worker numbers and names ($\exists z.\texttt{worker}(x, y, z)$):

$$\exists e, d.\texttt{empfile}(e) \wedge \texttt{emp-num}(e, x) \wedge \texttt{emp-name}(e, y) \wedge \texttt{emp-dept}(e, d) \wedge \neg\texttt{dept-mgr}(d, e)$$

3. List all department numbers and their names ($\exists z.\texttt{department}(x, y, z)$):

$$\exists d, e.\texttt{empfile}(e) \wedge \texttt{emp-dept}(e, d) \wedge \texttt{dept-num}(d, x) \wedge \texttt{dept-name}(d, y)$$

$\Rightarrow$ needs "departments have at least one employee".

…needs *duplicate elimination* during projection.

$$\exists e, d.\texttt{empfile}(e) \wedge \texttt{emp-dept}(e, d) \wedge \texttt{dept-num}(d, x) \wedge \texttt{dept-name}(d, y) \wedge \texttt{dept-mgr}(d, e)$$

$\Rightarrow$ needs "managers work in their own departments".

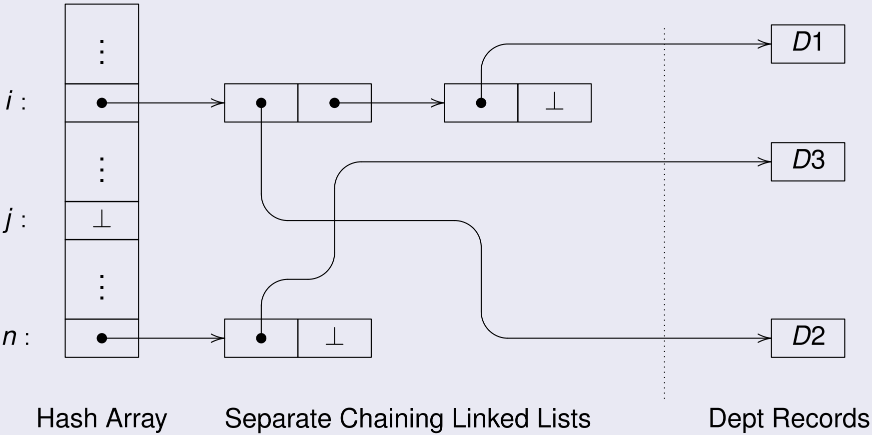
…NO *duplicate elimination* during projection.

. . . and we can actually synthesize this!

```
david$ compile tests/new_fe/book-em-v4-new-query.fol
query(dept2,2,0,[var(0,0,1,int),var(0,0,2,int)]) <->
  ex(var(0,76,4),
    ex(var(0,81,5),
      and (
        and (
          empfile(var(0,76,4))
          emp_dept(var(0,76,4),var(0,81,5))
        )
        and (
          and (
            dept_num(var(0,81,5),var(0,0,1))
            dept_name(var(0,81,5),var(0,0,2))
          )
          dept_mgr(var(0,81,5),var(0,76,4))
        )
      )
    )
```

What can it do: Hashing, Lists, et al.

Hash Index with (list-based) Separate Chaining



What can it do: Hashing, Linked lists, et al.

Hash Index on `department`'s `name`:

Access paths:

$\mathsf{S_A} \supseteq \{\mathtt{hash}/2/1, \mathtt{hasharraylookup}/2/1, \mathtt{listscan}/2/1\}$.

Physical Constraints:

$$\begin{aligned}\Sigma_{LP} \supseteq \{ & \forall x, y.((\mathtt{deptfile}(x) \land \mathtt{dept\text{-}name}(x, y)) \to \exists z, w.(\mathtt{hash}(y, z) \\ & \qquad \land \mathtt{hasharraylookup}(z, w) \land \mathtt{listscan}(w, x))), \\ & \forall x, y.(\mathtt{hash}(x, y) \to \exists z.\mathtt{hasharraylookup}(y, z)), \\ & \forall x, y.(\mathtt{listscan}(x, y) \to \mathtt{deptfile}(y)) \qquad \}\end{aligned}$$

What can it do: Hashing, Linked lists, et al.

Hash Index on `department`'s `name`:

Access paths:

$S_A \supseteq \{\texttt{hash}/2/1, \texttt{hasharraylookup}/2/1, \texttt{listscan}/2/1\}$.

Physical Constraints:

$$\begin{aligned}\Sigma_{LP} \supseteq \{&\forall x, y.((\texttt{deptfile}(x) \wedge \texttt{dept-name}(x, y)) \rightarrow \exists z, w.(\texttt{hash}(y, z)\\ &\qquad\qquad \wedge \texttt{hasharraylookup}(z, w) \wedge \texttt{listscan}(w, x))),\\ &\forall x, y.(\texttt{hash}(x, y) \rightarrow \exists z.\texttt{hasharraylookup}(y, z)),\\ &\forall x, y.(\texttt{listscan}(x, y) \rightarrow \texttt{deptfile}(y)) \qquad \}\end{aligned}$$

4. List departments and their managers given dept name p $(\exists y.\texttt{department}(x_1, p, y) \wedge \texttt{employee}(y, x_2)\{p\})$:

$$\begin{aligned}\exists h, l, d, e.&\texttt{hash}(p, h) \wedge \texttt{hasharraylookup}(h, l) \wedge\\ &\texttt{listscan}(l, d) \wedge \texttt{dept-name}(d, p) \wedge\\ &\texttt{dept-num}(d, x_1) \wedge \texttt{dept-mgr}(d, e) \wedge \texttt{emp-name}(e, x_2)\end{aligned}$$

What can this do: two-level store

The access path `empfile` is refined by `emppages`/1/0 and `emprecords`/2/1:

`emppages` returns (sequentially) disk pages containing `emp` records, and

`emprecords` given a disc page, returns `emp` records in that page.

5. List all employees with the same name
$(\exists z.\mathtt{employee}(x_1, z) \land \mathtt{employee}(x_2, z))$:

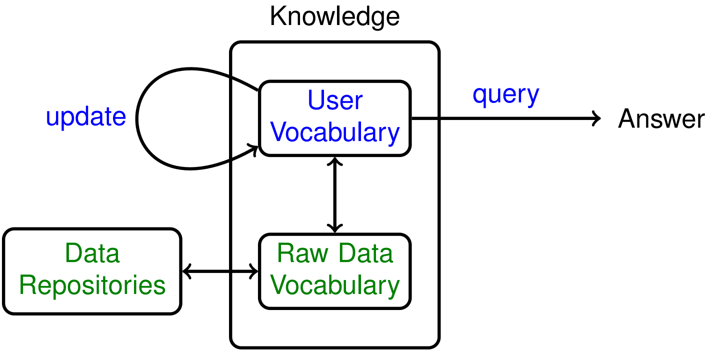
$$\begin{aligned}&\exists y, z, w, v, p, q.\mathtt{emppages}(p) \land \mathtt{emppages}(q)\\ &\quad \land \mathtt{emprecords}(p, y) \land \mathtt{emp\text{-}num}(y, x_1) \land \mathtt{emp\text{-}name}(y, w)\\ &\quad \land \mathtt{emprecords}(q, z) \land \mathtt{emp\text{-}num}(z, x_2) \land \mathtt{emp\text{-}name}(z, v)\\ &\qquad\qquad\qquad\qquad\qquad\qquad \land \mathtt{compare}(w, v).\end{aligned}$$

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

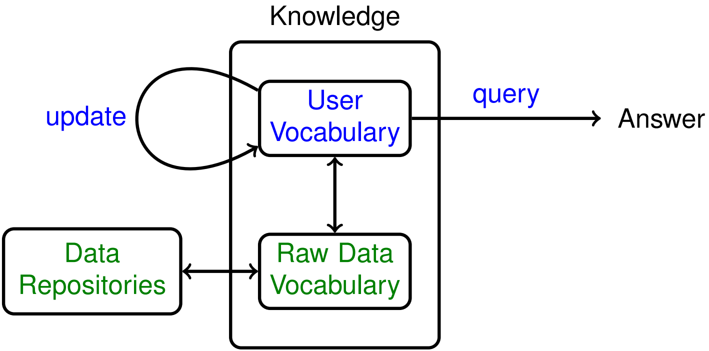
. . . more examples in .

UPDATES

Updates

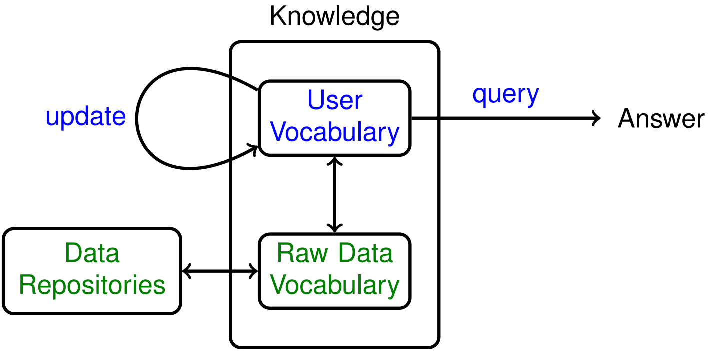

Knowledge
update
User Vocabulary
query
Answer
Data Repositories
Raw Data Vocabulary

Updates



1. Katsuno, Mendelzon: On the Difference between Updating a Knowledge Base and Revising It. KR 1991.
2. De Giacomo, Lenzerini, Poggi, Rosati: On Instance-level Update and Erasure in Description Logic Ontologies. J. Log. Comput. 19(5) 2009.

Updates



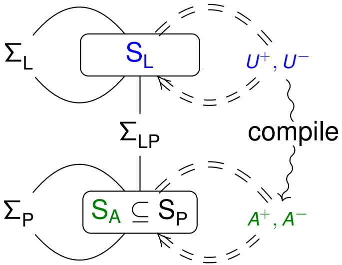
1. Katsuno, Mendelzon: On the Difference between Updating a Knowledge Base and Revising It. KR 1991.
2. De Giacomo, Lenzerini, Poggi, Rosati: On Instance-level Update and Erasure in Description Logic Ontologies. J. Log. Comput. 19(5) 2009.

. . . we use *definable updates* approach instead. . .

Updates and Definability

User updates *through logical schema ONLY*:
$\Rightarrow$ supplying "delta" relations (sets of tuples)

- Delta relations: R^+ (insertions) and R^- (deletions);

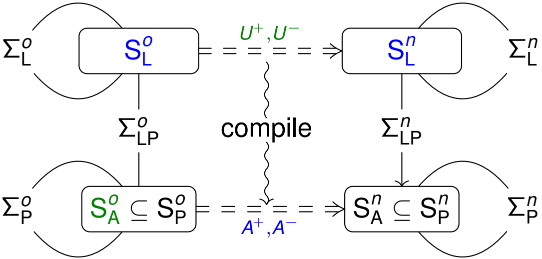


Updates and Definability

User updates *through logical schema ONLY*:

$\Rightarrow$ supplying "delta" relations (sets of tuples)

- Delta relations: R^+ (insertions) and R^- (deletions);



Update turned into *definability* question

Is A^n (or A^+, A^-) definable in terms of $A_i^o \in S_A^o$ (old access paths)
and U_j^+, U_j^- (user updates) for every access path $A \in S_A$?

Update Example

Example (Update Schema)

$$S_L = \{\texttt{emp}^o/1, \texttt{wkr}^o/1, \texttt{mgr}^o/1\} \cup \{\texttt{emp}^n/1, \texttt{wkr}^n/1, \texttt{mgr}^n/1\} \cup$$

$$\Sigma = \left\{ \begin{array}{l} \texttt{mgr}^o(x) \vee \texttt{wkr}^o(x) \leftrightarrow \texttt{emp}^o(x) \\ \texttt{mgr}^o(x) \wedge \texttt{wkr}^o(x) \to \bot \end{array} \right\} \cup \left\{ \begin{array}{l} \texttt{mgr}^n(x) \vee \texttt{wkr}^n(x) \leftrightarrow \texttt{emp}^n(x) \\ \texttt{mgr}^n(x) \wedge \texttt{wkr}^n(x) \to \bot \end{array} \right\} \cup$$

Update Example

Example (Update Schema)

$$S_L = \{\mathtt{emp}^o/1, \mathtt{wkr}^o/1, \mathtt{mgr}^o/1\} \cup \{\mathtt{emp}^n/1, \mathtt{wkr}^n/1, \mathtt{mgr}^n/1\} \cup \{\mathtt{emp}^+/1, \mathtt{wkr}^+/1, \mathtt{mgr}^+/1\} \cup \{\mathtt{emp}^-/1, \mathtt{wkr}^-/1, \mathtt{mgr}^-/1\}$$

$$\Sigma = \left\{\begin{array}{l} \mathtt{mgr}^o(x) \vee \mathtt{wkr}^o(x) \leftrightarrow \mathtt{emp}^o(x) \\ \mathtt{mgr}^o(x) \wedge \mathtt{wkr}^o(x) \rightarrow \bot \end{array}\right\} \cup \left\{\begin{array}{l} \mathtt{mgr}^n(x) \vee \mathtt{wkr}^n(x) \leftrightarrow \mathtt{emp}^n(x) \\ \mathtt{mgr}^n(x) \wedge \mathtt{wkr}^n(x) \rightarrow \bot \end{array}\right\} \cup$$
$$\left\{\begin{array}{l} \mathtt{emp}^o(x) \vee \mathtt{emp}^+(x) \leftrightarrow \mathtt{emp}^n(x) \vee \mathtt{emp}^-(x), \mathtt{emp}^o(x) \wedge \mathtt{emp}^+(x) \rightarrow \bot, \ldots \\ \mathtt{mgr}^o(x) \vee \mathtt{mgr}^+(x) \leftrightarrow \mathtt{mgr}^n(x) \vee \mathtt{mgr}^-(x), \mathtt{mgr}^o(x) \wedge \mathtt{mgr}^+(x) \rightarrow \bot, \ldots \\ \mathtt{wkr}^o(x) \vee \mathtt{wkr}^+(x) \leftrightarrow \mathtt{wkr}^n(x) \vee \mathtt{wkr}^-(x), \mathtt{wkr}^o(x) \wedge \mathtt{wkr}^+(x) \rightarrow \bot, \ldots \end{array}\right\}$$

Update Example

Example (Update Schema)

$$S_L = \{\mathtt{emp}^o/1, \mathtt{wkr}^o/1, \mathtt{mgr}^o/1\} \cup \{\mathtt{emp}^n/1, \mathtt{wkr}^n/1, \mathtt{mgr}^n/1\} \cup \{\mathtt{emp}^+/1, \mathtt{wkr}^+/1, \mathtt{mgr}^+/1\} \cup \{\mathtt{emp}^-/1, \mathtt{wkr}^-/1, \mathtt{mgr}^-/1\}$$

$$\Sigma = \left\{\begin{array}{l} \mathtt{mgr}^o(x) \vee \mathtt{wkr}^o(x) \leftrightarrow \mathtt{emp}^o(x) \\ \mathtt{mgr}^o(x) \wedge \mathtt{wkr}^o(x) \to \bot \end{array}\right\} \cup \left\{\begin{array}{l} \mathtt{mgr}^n(x) \vee \mathtt{wkr}^n(x) \leftrightarrow \mathtt{emp}^n(x) \\ \mathtt{mgr}^n(x) \wedge \mathtt{wkr}^n(x) \to \bot \end{array}\right\} \cup$$
$$\left\{\begin{array}{l} \mathtt{emp}^o(x) \vee \mathtt{emp}^+(x) \leftrightarrow \mathtt{emp}^n(x) \vee \mathtt{emp}^-(x), \mathtt{emp}^o(x) \wedge \mathtt{emp}^+(x) \to \bot, \ldots \\ \mathtt{mgr}^o(x) \vee \mathtt{mgr}^+(x) \leftrightarrow \mathtt{mgr}^n(x) \vee \mathtt{mgr}^-(x), \mathtt{mgr}^o(x) \wedge \mathtt{mgr}^+(x) \to \bot, \ldots \\ \mathtt{wkr}^o(x) \vee \mathtt{wkr}^+(x) \leftrightarrow \mathtt{wkr}^n(x) \vee \mathtt{wkr}^-(x), \mathtt{wkr}^o(x) \wedge \mathtt{wkr}^+(x) \to \bot, \ldots \end{array}\right\}$$

Result(s)

$$\mathtt{emp}^+(x) := (\mathtt{wkr}^+(x) \vee \mathtt{mgr}^+(x)) \wedge \neg\mathtt{emp}^o(x)$$
$$\mathtt{emp}^-(x) := (\mathtt{wkr}^-(x) \vee \mathtt{mgr}^-(x)) \wedge \neg(\mathtt{wkr}^+(x) \vee \mathtt{mgr}^+(x))$$

. . . and we can again synthesize that

```
david$ compile tests/new_fe/mgrwkr-upd.fol
query(empi,1,0,[var(0,0,1,int)]) <->
  and ( or (
      wkri(var(0,0,1))
      mgri(var(0,0,1))
    ) not (
      empo(var(0,0,1))
  ) )

query(empd,1,0,[var(0,0,1,int)]) <->
  and ( or (
      wkrd(var(0,0,1))
      mgrd(var(0,0,1))
    ) not ( or (
        mgri(var(0,0,1))
        wkri(var(0,0,1))
) ) )
```

Unknown/Anonymous Values?

Example (Add a new Worker record (needs an address))

```
INSERT into worker values (1234);
```

⇒ the request then needs to be translated to

```
INSERT into worker-physical values (0xFE1234, 1234);
```

⇒ but where did `0xFE1234` came from? (definability issue!)

Unknown/Anonymous Values?

Example (Add a new Worker record (needs an address))

`INSERT into worker values (1234);`

⇒ the request then needs to be translated to

`INSERT into worker-physical values (0xFE1234, 1234);`

⇒ but where did `0xFE1234` came from? (definability issue!)

Constant Complement: [Bancilhon, Spyratos: Update semantics of relational views. ACM Trans. Database Syst. 6(4), 1981.]

additional access paths that *provide* such values:

⇒ in our case `worker-addr(id,adress)`
⇒ and where $\texttt{worker}^{+} = \{(1234)\}$

$$\texttt{worker-physical}^{+}(x_2, x_1) = \texttt{worker}^{+}(x_1) \wedge \texttt{worker-addr}(x_1, x_2)$$

Unknown/Anonymous Values?

Example (Add a new Worker record (needs an address))

`INSERT into worker values (1234);`

⇒ the request then needs to be translated to

`INSERT into worker-physical values (0xFE1234, 1234);`

⇒ but where did `0xFE1234` came from? (definability issue!)

Constant Complement: [Bancilhon, Spyratos: Update semantics of relational views. ACM Trans. Database Syst. 6(4), 1981.]

additional access paths that *provide* such values:

⇒ in our case `worker-addr(id,adress)`

⇒ and where $\texttt{worker}^+ = \{(1234)\}$

$$\texttt{worker-physical}^+(x_2, x_1) = \texttt{worker}^+(x_1) \land \texttt{worker-addr}(x_1, x_2)$$

The additional access path(s) correspond to *space allocation*

…and cyclic dependencies are broken via *reification*.

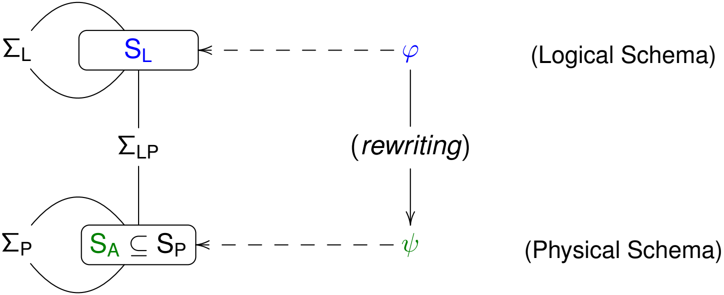
…more details and examples in .

HOW DOES IT ALL 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 Interpolation

IDEA #1: Plans as Formulas

Represent *query plans* as (annotated) range-restricted formulas ψ over $\mathsf{S_A}$:

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

Query Plans via Interpolation

IDEA #1: 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 $\Sigma \models \varphi \leftrightarrow \psi$

Query Plans via Interpolation

IDEA #1: 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

Beth Definability and Craig 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 the signature S_A (i.e., where the actual data is).

Beth Definability and Craig 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 the signature $\mathsf{S_A}$ (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.

Beth Definability and Craig 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 the signature $\mathsf{S_A}$ (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 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].

...and we can extract ψ from a (TABLEAU) proof of $\Sigma \cup \Sigma' \models \varphi \rightarrow \varphi'$

TABLEAU Proofs

1. Make entailment into formula to be proven:

$$\begin{array}{rll} \Sigma \cup \Sigma' \models \varphi \to \varphi' & \text{iff} & \models ((\bigwedge \Sigma) \wedge (\bigwedge \Sigma')) \to (\varphi \to \varphi') \quad \text{iff} \\ & & \models (\bigwedge \Sigma) \to ((\bigwedge \Sigma') \to (\varphi \to \varphi')) \quad \text{iff} \\ & & \models (\bigwedge \Sigma) \to (\varphi \to ((\bigwedge \Sigma') \to \varphi')) \quad \text{iff} \\ & & \models ((\bigwedge \Sigma) \wedge (\varphi) \to ((\bigwedge \Sigma') \to \varphi') \end{array}$$

or, equivalently, $(\bigwedge \Sigma) \ \wedge (\bigwedge \Sigma') \ \wedge \alpha \ \wedge \neg\alpha'$ is inconsistent;

2. Apply TABLEAU expansion rules to $(*)$ until all branches are closed;

$$\frac{S \cup \{\alpha, \beta\}}{S}\ \alpha \wedge \beta \in S\,, \qquad \frac{S \cup \{\alpha\} \qquad S \cup \{\beta\}}{S}\ \alpha \vee \beta \in S\,, \quad \ldots$$

where a branch S is *closed* if $\{\alpha, \neg\alpha\} \subseteq S$ (or $\perp \in S$).

TABLEAU Proofs

1. Make entailment into formula to be proven:

$$\Sigma \cup \Sigma' \models \varphi \to \varphi' \quad \text{iff} \quad \begin{array}{ll} \models ((\bigwedge \Sigma) \wedge (\bigwedge \Sigma')) \to (\varphi \to \varphi') & \text{iff} \\ \models (\bigwedge \Sigma) \to ((\bigwedge \Sigma') \to (\varphi \to \varphi')) & \text{iff} \\ \models (\bigwedge \Sigma) \to (\varphi \to ((\bigwedge \Sigma') \to \varphi')) & \text{iff} \\ \models ((\bigwedge \Sigma) \wedge (\varphi) \to ((\bigwedge \Sigma') \to \varphi') & \end{array}$$

   or, equivalently, $(\bigwedge \Sigma)^L \wedge (\bigwedge \Sigma')^R \wedge \alpha^L \wedge \neg \alpha'^R$ is inconsistent;

2. Apply TABLEAU expansion rules to $(*)$ until all branches are closed;

$$\frac{S \cup \{\alpha, \beta\}}{S} \; \alpha \wedge \beta \in S\,, \qquad \frac{S \cup \{\alpha\} \qquad S \cup \{\beta\}}{S} \; \alpha \vee \beta \in S\,, \quad \ldots$$

   where a branch S is *closed* if $\{\alpha, \neg\alpha\} \subseteq S$ (or $\perp \in S$).

3. Extract an interpolant from (2); this needs extra $.^L$ and $.^R$ annotations.

TABLEAU and Interpolant Extraction (sketch)

- 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 $.^L$ and $.^R$-tagged subsets of S;

TABLEAU and Interpolant Extraction (sketch)

- 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 $.^L$ and $.^R$-tagged subsets of S;
- tableau rules (sample):
  - LR clash $\boxed{S \cup \{R^L, \neg R^R\} \xrightarrow{int} R}$, where $R \in \mathsf{S}_\mathsf{A}$
    as $(\bigwedge S^L \wedge R^L) \to R$ and $R \to (R^R \vee \neg \bigwedge S^R)$;

TABLEAU and Interpolant Extraction (sketch)

- 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 $.^L$ and $.^R$-tagged subsets of S;
- tableau rules (sample):
  - LR clash $\boxed{S \cup \{R^L, \neg R^R\} \xrightarrow{int} R}$, where $R \in \mathsf{S_A}$
    as $(\bigwedge S^L \wedge R^L) \to R$ and $R \to (R^R \vee \neg \bigwedge S^R)$;
  - L-conjunction $\boxed{\dfrac{S \cup \{\alpha^L, \beta^L\} \xrightarrow{int} \delta}{S \cup \{(\alpha \wedge \beta)^L\} \xrightarrow{int} \delta}}$
    as $(\bigwedge S^L \wedge \alpha^L \wedge \beta^L) \to \delta$ implies $(\bigwedge S^L \wedge (\alpha \wedge \beta)^L) \to \delta$;

TABLEAU and Interpolant Extraction (sketch)

- 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 $.^L$ and $.^R$-tagged subsets of S;
- tableau rules (sample):
  - LR clash $\boxed{S \cup \{R^L, \neg R^R\} \xrightarrow{int} R}$, where $R \in \mathsf{S_A}$
    as $(\bigwedge S^L \wedge R^L) \to R$ and $R \to (R^R \vee \neg \bigwedge S^R)$;
  - L-conjunction $\boxed{\dfrac{S \cup \{\alpha^L, \beta^L\} \xrightarrow{int} \delta}{S \cup \{(\alpha \wedge \beta)^L\} \xrightarrow{int} \delta}}$
    as $(\bigwedge S^L \wedge \alpha^L \wedge \beta^L) \to \delta$ implies $(\bigwedge S^L \wedge (\alpha \wedge \beta)^L) \to \delta$;
  - R-Disjunction $\boxed{\dfrac{S \cup \{\alpha^R\} \xrightarrow{int} \delta_\alpha \qquad S \cup \{\beta^R\} \xrightarrow{int} \delta_\beta}{S \cup \{(\alpha \vee \beta)^R\} \xrightarrow{int} \delta_\alpha \wedge \delta_\beta}}$
    as $\bigwedge S^L \to \delta_\alpha, \delta_\alpha \to \neg(\alpha^R \wedge \bigwedge S^R)$ and $\bigwedge S^L \to \delta_\beta, \delta_\beta \to \neg(\beta^R \wedge \bigwedge S^R)$
    implies $(\bigwedge S^L) \to \delta_\alpha \wedge \delta_\beta, \delta_\alpha \wedge \delta_\beta \to \neg((\alpha \vee \beta)^R \wedge \bigwedge S^R)$.
  - etc. (see [Fitting] for details)

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.

Factoring *logical reasoning* from *plan enumeration*

$\Rightarrow$ backtracking tableau to get alternative plans: too slow, too few plans

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):

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

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

$\varphi[C]\{B\}$ where C is a set of (ground) atoms over S_A and B branch descriptors
$\Rightarrow$ φ only exists in T 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

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

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

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 $\Sigma^R \cup \{Q^R(\bar{a}) \to \perp\}$ 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 $\perp[D]$ 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 $\perp(\top)$.

Conditional Tableau through an Example

Example (Horizontal Partition)

$\mathsf{S_L} = \{\mathtt{emp}/1, \mathtt{wkr}/1, \mathtt{mgr}/1\}$ and $\Sigma_\mathsf{L} = \left\{ \begin{array}{l} \mathtt{mgr}(x) \vee \mathtt{wkr}(x) \leftrightarrow \mathtt{emp}(x) \\ \mathtt{mgr}(x) \wedge \mathtt{wkr}(x) \rightarrow \bot \end{array} \right\}$

$\mathsf{S_A} = \{\mathtt{emp}/1,\ \mathtt{mgr}/1\}$, note that $\mathtt{wkr}/1$ is NOT in $\mathsf{S_A}$;

Conditional Tableau through an Example

Example (Horizontal Partition)

$S_L = \{\mathtt{emp}/1, \mathtt{wkr}/1, \mathtt{mgr}/1\}$ and $\Sigma_L = \left\{ \begin{array}{l} \mathtt{mgr}(x) \vee \mathtt{wkr}(x) \leftrightarrow \mathtt{emp}(x) \\ \mathtt{mgr}(x) \wedge \mathtt{wkr}(x) \rightarrow \bot \end{array} \right\}$

$S_A = \{\mathtt{emp}/1,\ \mathtt{mgr}/1\}$, note that $\mathtt{wkr}/1$ is NOT in S_A;

ANF of Σ

$$\left\{ \begin{array}{ll} \mathtt{emp}(x) \rightarrow \mathtt{wkr}(x) \vee \mathtt{mgr}(x), & \mathtt{mgr}(x) \rightarrow \mathtt{emp}(x), \\ \mathtt{mgr}(x) \wedge \mathtt{wkr}(x) \rightarrow \bot, & \mathtt{wkr}(x) \rightarrow \mathtt{emp}(x) \end{array} \right\}$$

Conditional Tableau through an Example

ANF of Σ

$$\left\{ \begin{array}{ll} \mathtt{emp}(x) \to \mathtt{wkr}(x) \vee \mathtt{mgr}(x), & \mathtt{mgr}(x) \to \mathtt{emp}(x), \\ \mathtt{mgr}(x) \wedge \mathtt{wkr}(x) \to \bot, & \mathtt{wkr}(x) \to \mathtt{emp}(x) \end{array} \right\}$$

ANF of the Query

$$\begin{array}{l} \mathtt{query}(x) \to \mathtt{wkr}(x), \quad \mathtt{wrk}(x) \to \mathtt{query}(x) \quad \text{and} \\ \mathtt{query}^{L}(0), \quad \mathtt{query}^{R}(0) \to \bot \end{array}$$

Conditional Tableau through an Example

ANF of Σ

$$\left\{ \begin{array}{ll} \mathtt{emp}(x) \rightarrow \mathtt{wkr}(x) \vee \mathtt{mgr}(x), & \mathtt{mgr}(x) \rightarrow \mathtt{emp}(x), \\ \mathtt{mgr}(x) \wedge \mathtt{wkr}(x) \rightarrow \bot, & \mathtt{wkr}(x) \rightarrow \mathtt{emp}(x) \end{array} \right\}$$

ANF of the Query

$$\begin{array}{l} \mathtt{query}(x) \rightarrow \mathtt{wkr}(x), \quad \mathtt{wrk}(x) \rightarrow \mathtt{query}(x) \quad \text{and} \\ \mathtt{query}^{L}(0), \quad \mathtt{query}^{R}(0) \rightarrow \bot \end{array}$$

$$\begin{array}{lcl} T^{L} & & T^{R} \\ \hline \mathtt{query}(0)[]\{\} & & \\ \mathtt{wkr}(0)[]\{\} & & \\ \mathtt{emp}(0)[]\{\} & \rightarrow & \mathtt{emp}(0)[\mathtt{emp}(0)]\{\} \\ & & \mathtt{wkr}(0)[\mathtt{emp}(0)]\{1.0\} \\ & & \mathtt{query}(0)[\mathtt{emp}(0)]\{1.0\} \\ & & \bot[\mathtt{emp}(0)]\{1.0\} \\ \mathtt{mgr}(0)[\mathtt{mgr}(0)]\{\} & \leftarrow & \mathtt{mgr}(0)[\mathtt{emp}(0)]\{1.1\} \\ \bot[\mathtt{mgr}(0)]\{\} & & \end{array}$$

Conditional Tableau through an Example

T^L		T^R
$\mathtt{query}(0)[]\{\}$		
$\mathtt{wkr}(0)[]\{\}$		
$\mathtt{emp}(0)[]\{\}$	$\to$	$\mathtt{emp}(0)[\mathtt{emp}(0)]\{\}$
		$\mathtt{wkr}(0)[\mathtt{emp}(0)]\{1.0\}$
		$\mathtt{query}(0)[\mathtt{emp}(0)]\{1.0\}$
		$\bot[\mathtt{emp}(0)]\{1.0\}$
$\mathtt{mgr}(0)[\mathtt{mgr}(0)]\{\}$	$\leftarrow$	$\mathtt{mgr}(0)[\mathtt{emp}(0)]\{1.1\}$
$\bot[\mathtt{mgr}(0)]\{\}$		

Closing sets:

$\{\neg\mathtt{emp}(0)\}, \{\mathtt{mgr}(0)\}$ $\qquad$ $\{\neg\mathtt{mgr}(0), \mathtt{emp}(0)\}$

Conditional Tableau through an Example

T^L		T^R
$\mathtt{query}(0)[]\{\}$		
$\mathtt{wkr}(0)[]\{\}$		
$\mathtt{emp}(0)[]\{\}$	$\rightarrow$	$\mathtt{emp}(0)[\mathtt{emp}(0)]\{\}$
		$\mathtt{wkr}(0)[\mathtt{emp}(0)]\{1.0\}$
		$\mathtt{query}(0)[\mathtt{emp}(0)]\{1.0\}$
		$\bot[\mathtt{emp}(0)]\{1.0\}$
$\mathtt{mgr}(0)[\mathtt{mgr}(0)]\{\}$	$\leftarrow$	$\mathtt{mgr}(0)[\mathtt{emp}(0)]\{1.1\}$
$\bot[\mathtt{mgr}(0)]\{\}$		

Closing sets:

$\{\neg\mathtt{emp}(0)\}, \{\mathtt{mgr}(0)\}$ $\qquad$ $\{\neg\mathtt{mgr}(0), \mathtt{emp}(0)\}$

Query Plan:

$$\mathtt{query}(x) \leftrightarrow \mathtt{emp}(x) \wedge \neg\mathtt{mgr}(x)$$

Plan Enumeration

Physical Tableau T^P for a Plan P

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

Observation

For a range-restricted formula P over S_A there is an analytic tableau tree T^P that uses only 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.

University of Waterloo

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*.

Closing Sets (more complex example)

1. Byte code generation for `q/2`

```
q(x,y) <-> ex(z,table(x,x,z) and table(z,y,y)
                                and not table(x,x,x))
```

2. Conditional Tableau Construction

```
L { -p0basetable(sl19:7,sl14:3,sl0:2,sl0:2) }
L { -p0basetable(sl19:5,sl0:1,sl0:1,sl14:3) }
L { +p0basetable(sr19:8,sl0:1,sl0:1,sl0:1) }
R { -p0basetable(sr19:8,sl0:1,sl0:1,sl0:1),
    +p0basetable(sl19:7,sl14:3,sl0:2,sl0:2),
    +p0basetable(sl19:5,sl0:1,sl0:1,sl14:3) }
```

3. Cost-based Optimization (A*)
4. C code Generation (+ compilation/linking w/runtime library)

[Hudek, Toman, Weddell: On Enumerating Query Plans Using Analytic Tableau. TABLEAUX 2015.]
[Toman, Weddell: An Interpolation-based Compiler and Optimizer for Relational Queries (System design Report). IWIL-LPAR 2017.]

CONDITIONAL TABLEAU: Result

```
query(q,2,0,[var(0,0,1,int),var(0,0,2,int)]) <->
  ex(var(0,14,3),
    ex(var(0,19,5),
      ex(var(0,19,7),
        and (
          and (
            p0basetable(var(0,19,7),var(0,14,3),
                        var(0,0,2),var(0,0,2))
            p0basetable(var(0,19,5),var(0,0,1),
                        var(0,0,1),var(0,14,3))
          )
          not (
            ex(var(1,19,8),
              p0basetable(var(1,19,8),var(0,0,1),
                          var(0,0,1),var(0,0,1))
            )
          ) ) ) ) )
```

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 ($\{\cdot\}$).

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 $\{\cdot\}$:

$$
\begin{array}{rcll}
\psi[\{R(x_1,\ldots,x_k)\}] & \leftrightarrow & \psi[R(x_1,\ldots,x_k)] & \\
\psi[\{\psi_1 \wedge \psi_2\}] & \leftrightarrow & \psi[\{\psi_1\} \wedge \{\psi_2\}] & \\
\psi[\{\neg\psi_1\}] & \leftrightarrow & \psi[\neg\psi_1] & \\
\psi[\neg\{\psi_1\}] & \leftrightarrow & \psi[\neg\psi_1] & \\
\psi[\{\psi_1 \vee \psi_2\}] & \leftrightarrow & \psi[\{\psi_1\} \vee \{\psi_2\}] & \text{if } \Sigma \cup \{\psi[]\} \models \psi_1 \wedge \psi_2 \rightarrow \bot \\
\psi[\{\exists x.\psi_1\}] & \leftrightarrow & \psi[\exists x.\{\psi_1\}] & \text{if} \\
 & & \multicolumn{2}{r}{\Sigma \cup \{\psi[] \wedge \psi_1[y_1/x] \wedge \psi_1[y_2/x] \models y_1 \approx y_2}
\end{array}
$$

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 $\{\cdot\}$:

$$
\begin{aligned}
\psi[\{R(x_1,\ldots,x_k)\}] &\leftrightarrow \psi[R(x_1,\ldots,x_k)] \\
\psi[\{\psi_1 \wedge \psi_2\}] &\leftrightarrow \psi[\{\psi_1\} \wedge \{\psi_2\}] \\
\psi[\{\neg\psi_1\}] &\leftrightarrow \psi[\neg\psi_1] \\
\psi[\neg\{\psi_1\}] &\leftrightarrow \psi[\neg\psi_1] \\
\psi[\{\psi_1 \vee \psi_2\}] &\leftrightarrow \psi[\{\psi_1\} \vee \{\psi_2\}] \qquad \text{if } \Sigma \cup \{\psi[]\} \models \psi_1 \wedge \psi_2 \to \bot \\
\psi[\{\exists x.\psi_1\}] &\leftrightarrow \psi[\exists x.\{\psi_1\}] \qquad \text{if} \\
&\qquad \Sigma \cup \{\psi[] \wedge \psi_1[y_1/x] \wedge \psi_1[y_2/x] \models y_1 \approx y_2
\end{aligned}
$$

... reasoning abstracted in FunDL-Lite (a PTIME fragment)

[Toman, Weddell: Using Feature-Based Description Logics to avoid Duplicate Elimination in Object-Relational Query Languages. Künstliche Intell. 34(3): 2020]

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 ($\{\cdot\}$).

$$\exists d, e.\texttt{empfile}(e) \land \texttt{emp-dept}(e, d) \land \texttt{dept-num}(d, x) \land \texttt{dept-name}(d, y)$$

$\Rightarrow$ dept number (x) and name (y) do NOT functionally determine employee (e).

$$\exists e, d.\texttt{empfile}(e) \land \texttt{emp-dept}(e, d) \land \texttt{dept-num}(d, x) \land \texttt{dept-name}(d, y) \land \texttt{dept-mgr}(d, e)$$

$\Rightarrow$ dept number (x) and name (y) do functionally determine employee (e) and dept (d).

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 ($\{\cdot\}$).

$$\{\exists d, e.\texttt{empfile}(e) \wedge \texttt{emp-dept}(e, d) \wedge \texttt{dept-num}(d, x) \wedge \texttt{dept-name}(d, y)\}$$

$\Rightarrow$ dept number (x) and name (y) do NOT functionally determine employee (e).

. . . needs *duplicate elimination* during projection.

$$\exists e, d.\texttt{empfile}(e) \wedge \texttt{emp-dept}(e, d) \wedge \texttt{dept-num}(d, x) \wedge \texttt{dept-name}(d, y) \wedge \texttt{dept-mgr}(d, e)$$

$\Rightarrow$ dept number (x) and name (y) do functionally determine employee (e) and dept (d).

. . . does NOT need *duplicate elimination* during projection.

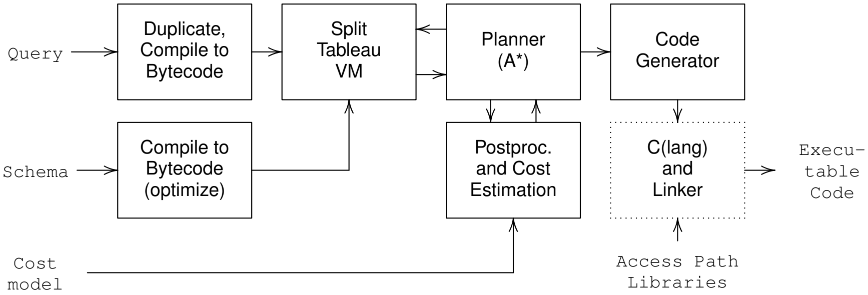
Interpolation (summary)

(A) solution: a *conditional tableau*:

1. reformulate the interpolation problem to
   $\Sigma^L \cup \Sigma^R \cup \Sigma^{LR} \models \varphi^L \to \varphi^R$ where $\Sigma^{LR} = \{\forall \bar{x}.P^L \leftrightarrow P \leftrightarrow P^R \mid P \in \mathrm{S_A}\}$
2. use conditional (ground) atoms to generate
   closing sets: sets of $\mathrm{S_A}$ literals that (fully) close the tableau
3. separate general reasoning from interpolant enumeration
   1) VM-driven conditional tableau for $\Sigma^L \cup \{\varphi^L\}$ and for $\Sigma^R \cup \{\varphi^R \to \bot\}$
   2) A*-based interpolant generator w.r.t. closing sets and Σ^{LR}
4. FunDL-Lite *postprocessing* to deal w/duplicates *et al.*

Details: [Hudek et al., 2015, Toman and Weddell, 2017]

Compiler Architecture



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 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!

Bachman, C. W. (1969).
CODASYL data base task group: October 1969 report.

Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., and Rosati, R. (2007).
Tractable reasoning and efficient query answering in description logics: The DL-Lite family.
J. Autom. Reasoning, 39(3):385–429.

Codd, E. F. (1970).
A relational model of data for large shared data banks.
Commun. ACM, 13(6):377–387.

Codd, E. F. (1972).
Relational completeness of data base sub-languages.
In Rustin, R., editor, *Data Base Systems*, pages 33–64. Prentice-Hall.

Date, C. J. and Hopewell, P. (1971).
Storage structure and physical data independence.
In *Proceedings of the 1971 ACM SIGFIDET (now SIGMOD) Workshop on Data Description, Access and Control*, SIGFIDET '71, pages 139–168. ACM.

Eiter, T., Ortiz, M., Simkus, M., Tran, T., and Xiao, G. (2012).
Query rewriting for horn-shiq plus rules.
In *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence, July 22-26, 2012, Toronto, Ontario, Canada.*

Hudek, A. K., Toman, D., and Weddell, G. E. (2015).
On enumerating query plans using analytic tableau.
In *Automated Reasoning with Analytic Tableaux and Related Methods - 24th International Conference, TABLEAUX 2015, Wrocław, Poland, September 21-24, 2015. Proceedings*, pages 339–354.

Kontchakov, R., Lutz, C., Toman, D., Wolter, F., and Zakharyaschev, M. (2010).
The combined approach to query answering in DL-Lite.

In *Principles of Knowledge Representation and Reasoning: Proceedings of the Twelfth International Conference, KR 2010, Toronto, Ontario, Canada, May 9-13, 2010.*

Liskov, B. H. and Zilles, S. N. (1974).
Programming with abstract data types.
SIGPLAN Notices, 9(4):50–59.

Lutz, C., Toman, D., and Wolter, F. (2009).
Conjunctive query answering in the description logic EL using a relational database system.
In *Proc. Int. Joint Conf. on Artificial Intelligence (IJCAI)*, pages 2070–2075.

Toman, D. and Weddell, G. E. (2013).
Conjunctive Query Answering in $\mathcal{CFD}_{nc}$: A PTIME Description Logic with Functional Constraints and Disjointness.
In *Australasian Conference on Artificial Intelligence*, pages 350–361.

Toman, D. and Weddell, G. E. (2017).
An interpolation-based compiler and optimizer for relational queries (system design report).
In *IWIL@LPAR 2017 Workshop and LPAR-21 Short Presentations, Maun, Botswana, May 7-12, 2017*.