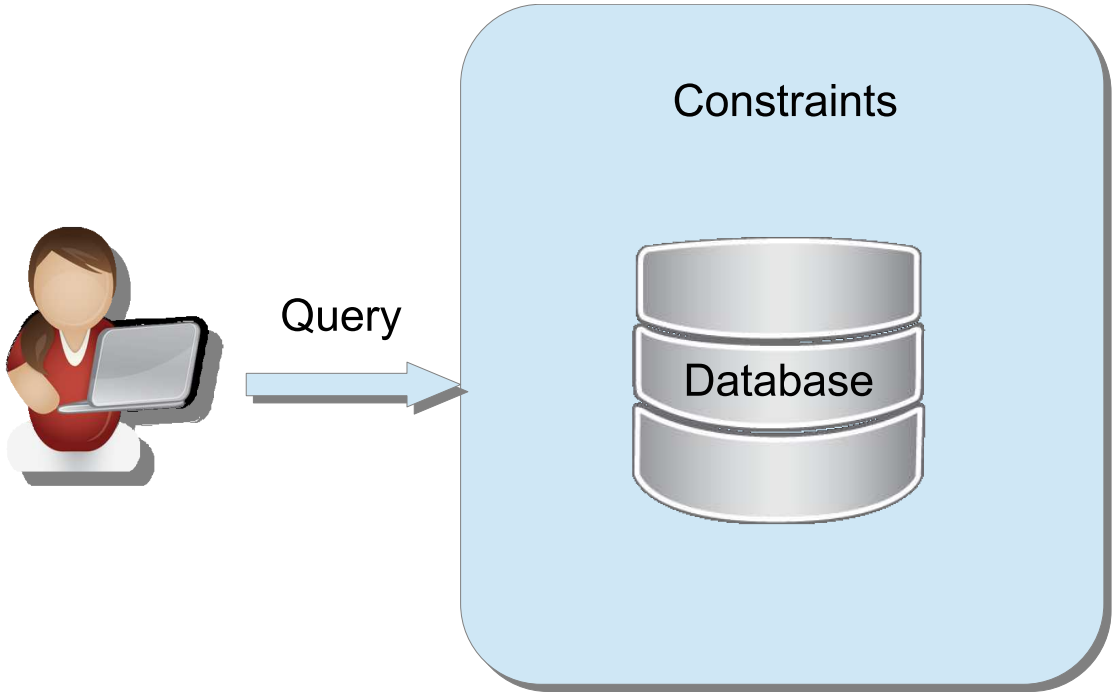
Exact Query Reformulation with First-order Ontologies and Databases

Nhung Ngo

Free University of Bolzano
A joint work with Enrico Franconi and Volga Kernet

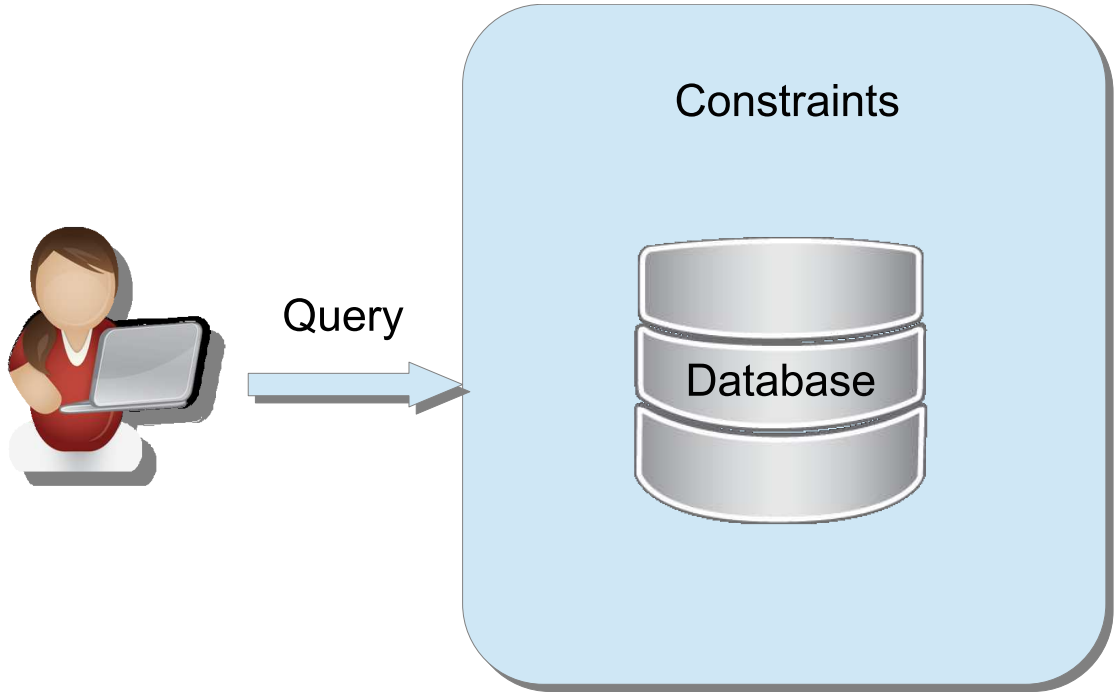
Nov 13, 2013

Query Answering Under Contraints and Databases



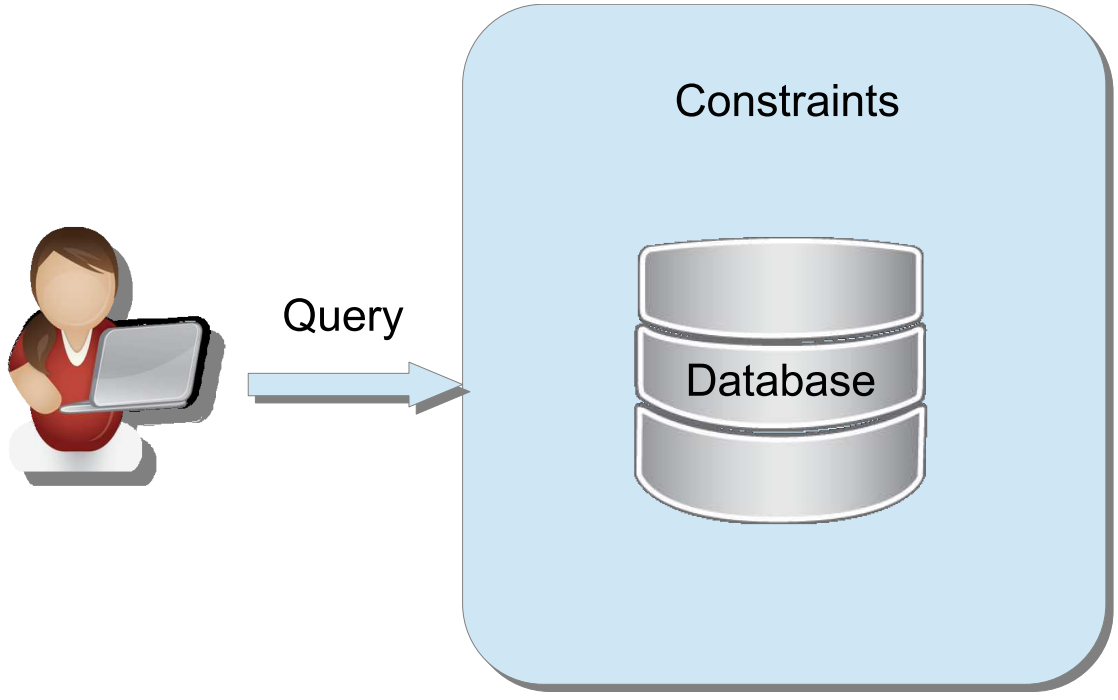
- Given a FOL fragment $\mathcal{L}$
- Constraints: a set of L-sentences $\mathcal{KB}$
- Database: a finite set of facts $\mathcal{DB}$ over a relational signature $\mathcal{P}_{\mathcal{DB}}$
- Query: a (open) $\mathcal{L}$-formula

Query Answering Under Contraints and Databases



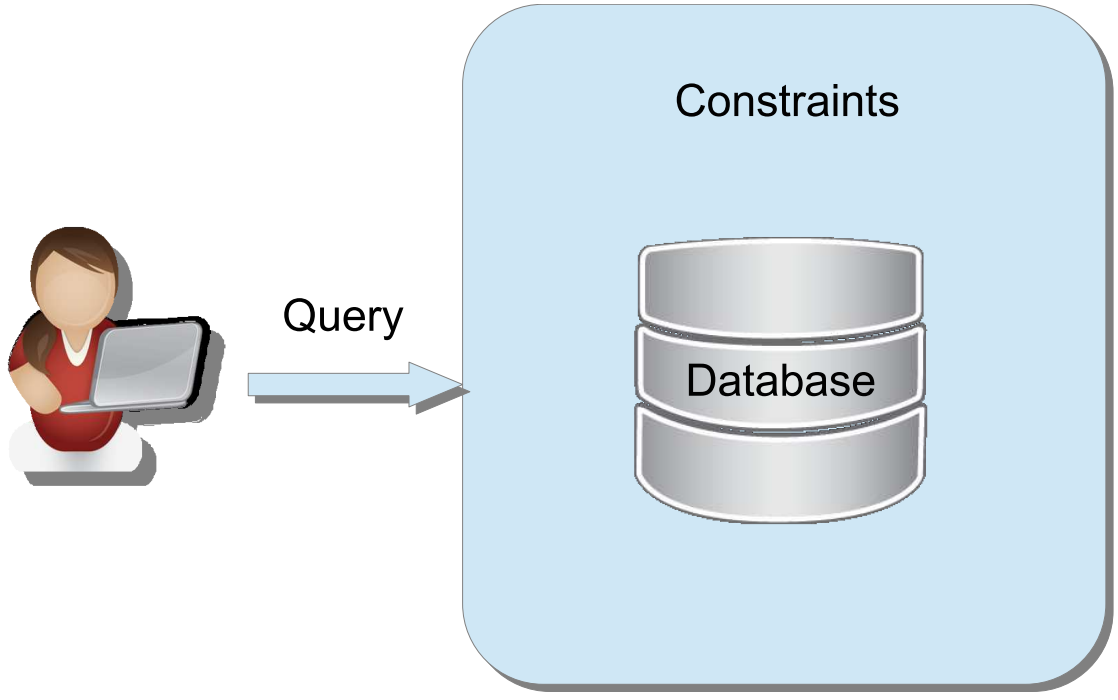
- Given a FOL fragment $\mathcal{L}$
- Constraints: a set of L-sentences $\mathcal{K}B$
- Database: a finite set of facts $\mathcal{D}B$ over a relational signature $\mathcal{P}_{\mathcal{D}\mathcal{B}}$
- Query: a (open) $\mathcal{L}$-formula

Query Answering Under Contraints and Databases



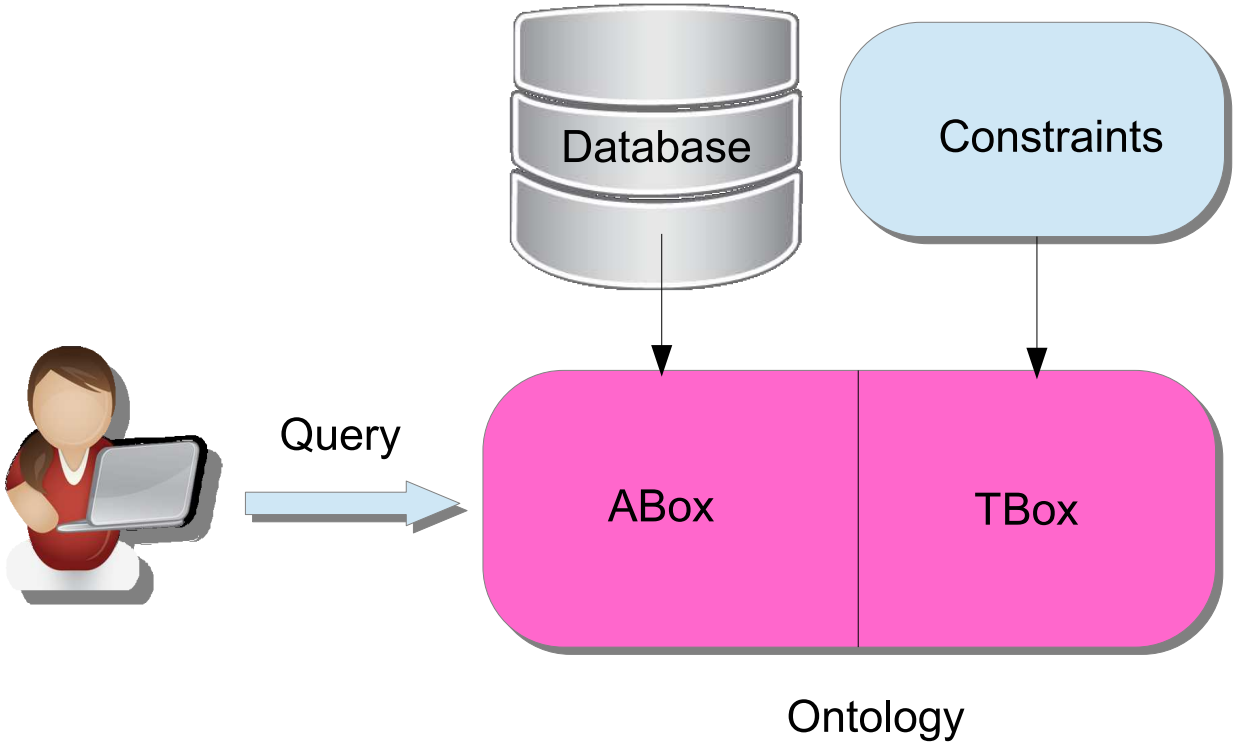
- Given a FOL fragment $\mathcal{L}$
- Constraints: a set of L-sentences $\mathcal{K}B$
- Database: a finite set of facts $\mathcal{D}B$ over a relational signature $\mathcal{P}_{\mathcal{DB}}$
- Query: a (open) $\mathcal{L}$-formula

Query Answering Under Contraints and Databases



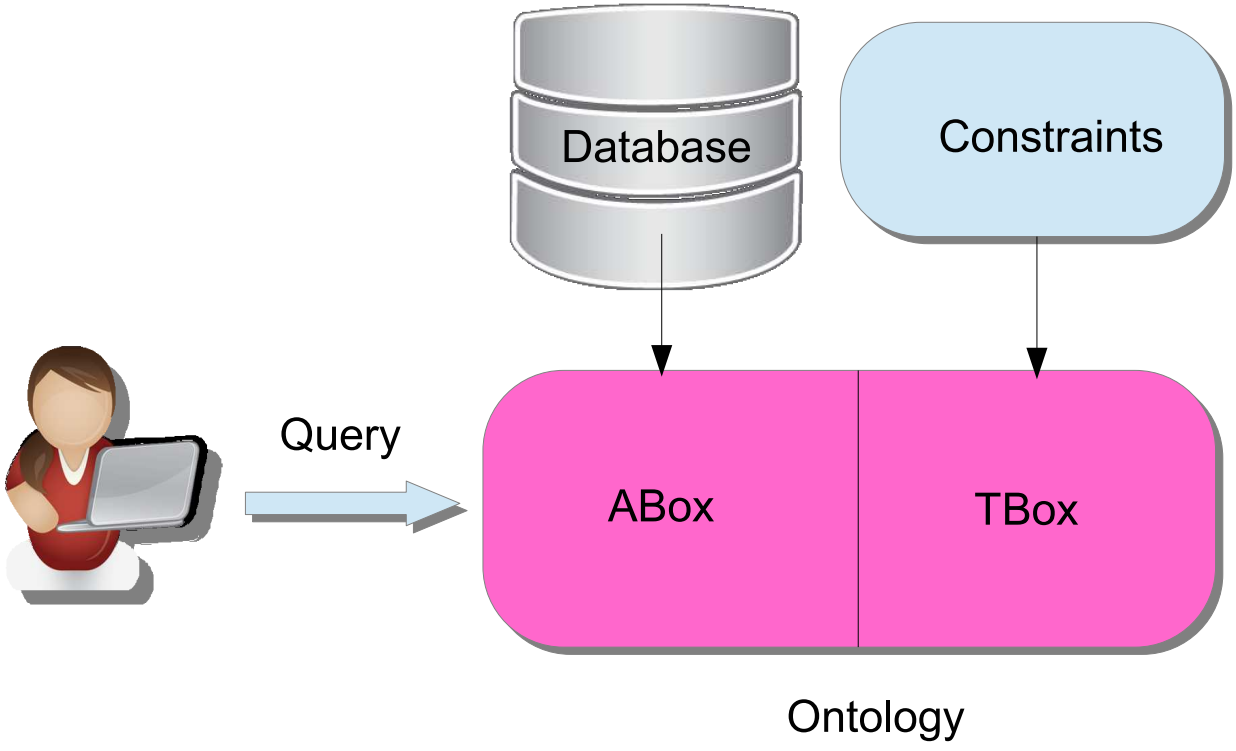
- Given a FOL fragment $\mathcal{L}$
- Constraints: a set of L-sentences $\mathcal{K}B$
- Database: a finite set of facts $\mathcal{D}B$ over a relational signature $\mathcal{P}_{\mathcal{DB}}$
- Query: a (open) $\mathcal{L}$-formula

Ontology-based Data Access



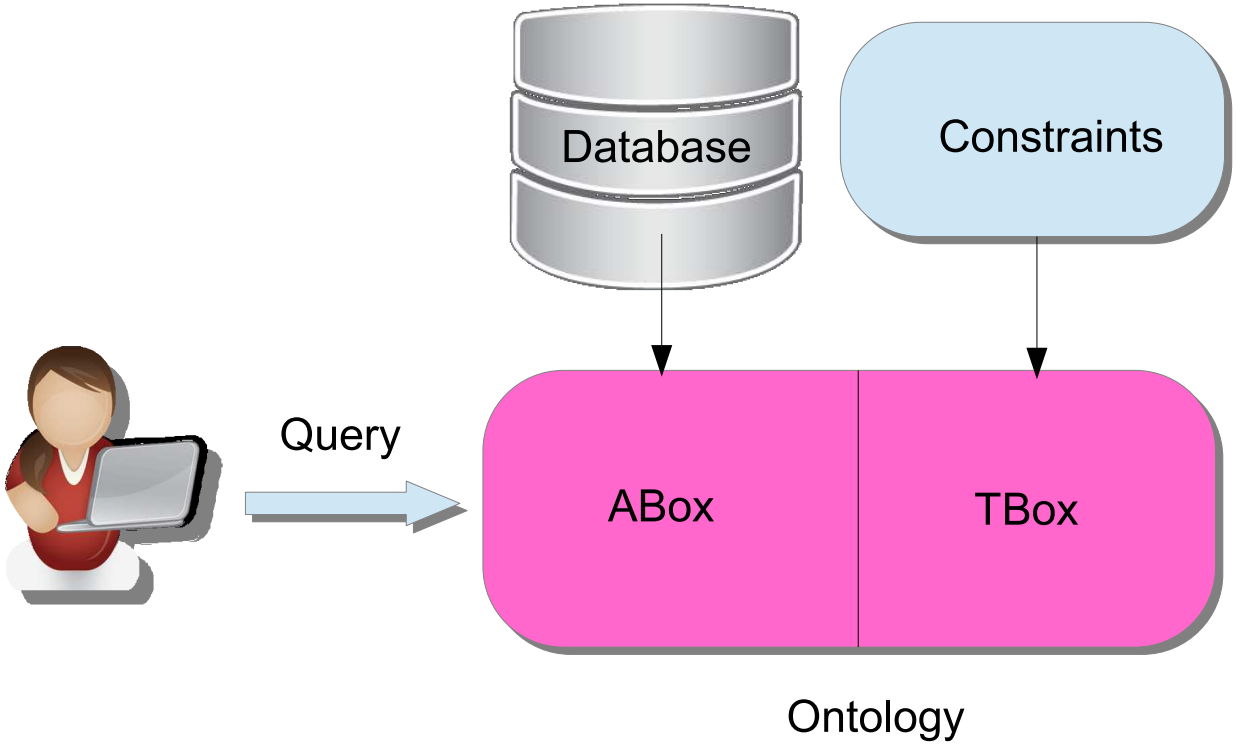
- Given a description logic fragment $\mathcal{DL}$
- TBox: a set of TBox assertions $\mathcal{T}$
- Databases: a set of ABox assertions $\mathcal{A}$
- Query: a concept query

Ontology-based Data Access



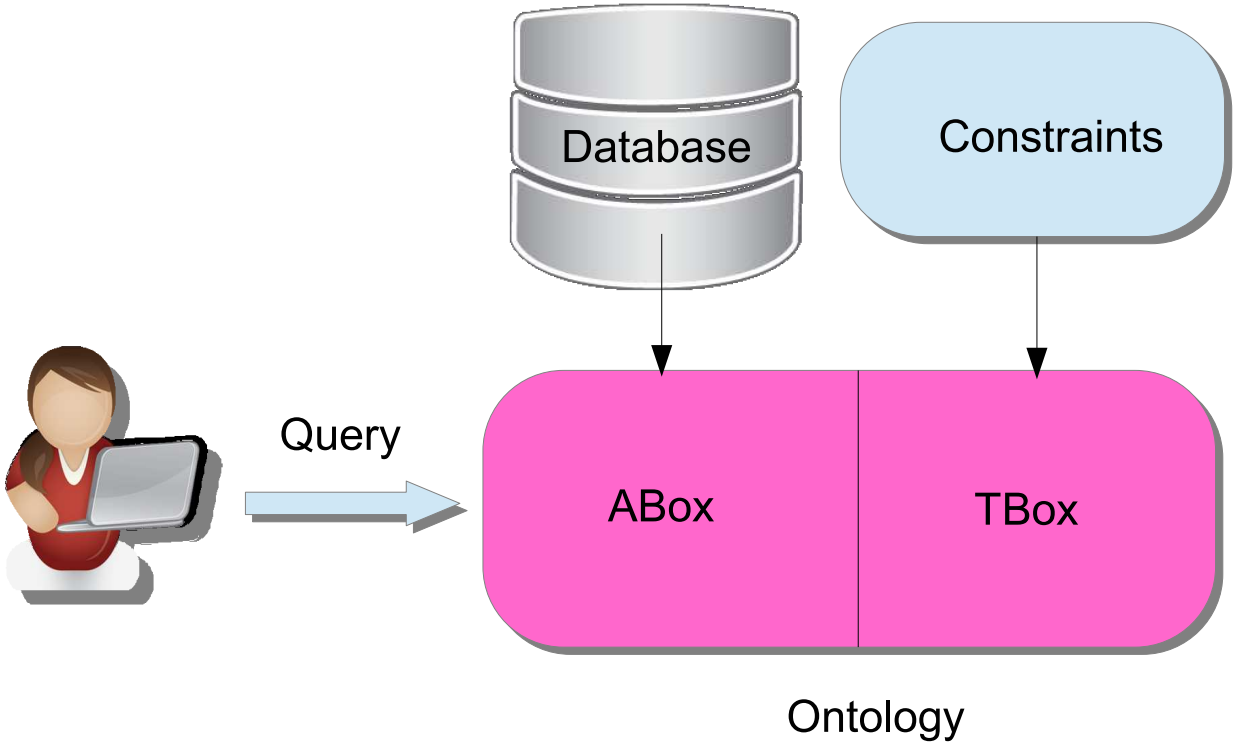
- Given a description logic fragment $\mathcal{DL}$
- TBox: a set of TBox assertions $\mathcal{T}$
- Databases: a set of ABox assertions $\mathcal{A}$
- Query: a concept query

Ontology-based Data Access



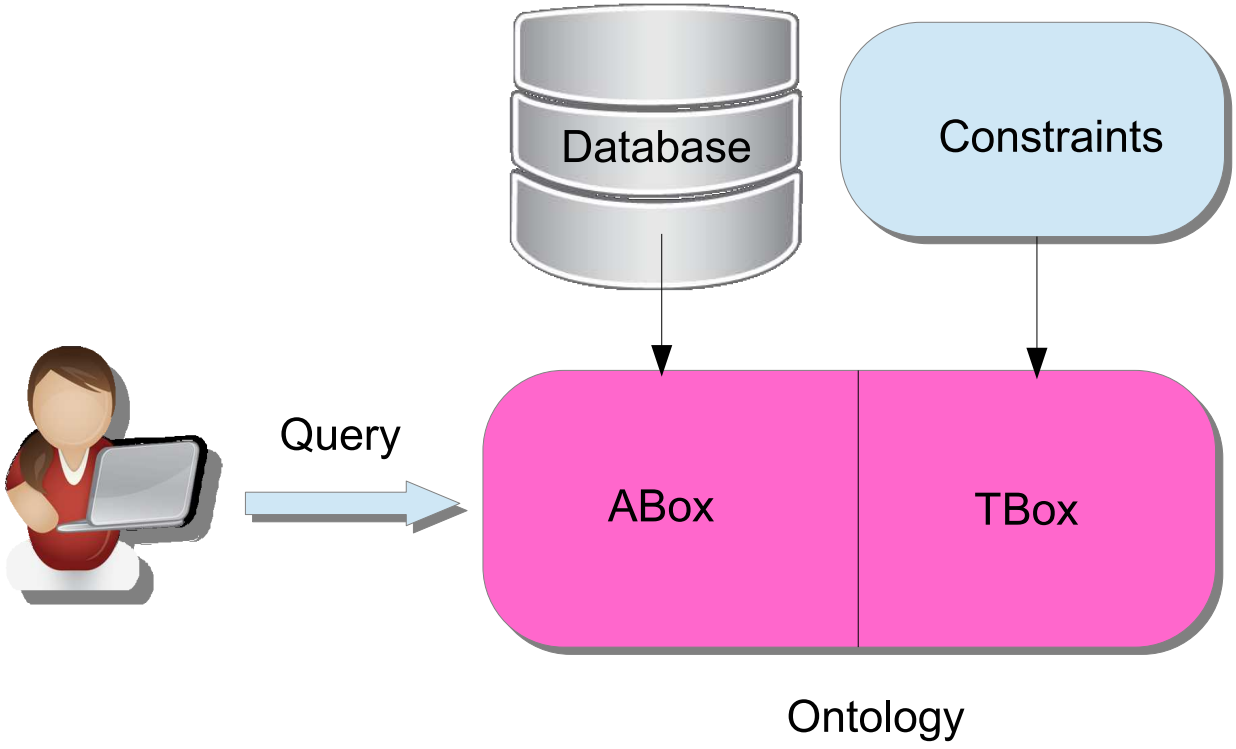
- Given a description logic fragment $\mathcal{DL}$
- TBox: a set of TBox assertions $\mathcal{T}$
- Databases: a set of ABox assertions $\mathcal{A}$
- Query: a concept query

Ontology-based Data Access



- Given a description logic fragment $\mathcal{DL}$
- TBox: a set of TBox assertions $\mathcal{T}$
- Databases: a set of ABox assertions $\mathcal{A}$
- Query: a concept query

Ontology-based Data Access



- Given a description logic fragment $\mathcal{DL}$
- TBox: a set of TBox assertions $\mathcal{T}$
- Databases: a set of ABox assertions $\mathcal{A}$
- Query: a concept query

ABox

- An ABox is a finite set of ground atomic facts, syntactically

Example

$\mathcal{A} = \{\texttt{Employee(Jon)}, \texttt{Project(Winter)}\}$ is an ABox

- The semantics of ABoxes is given by first-order structures.
- An ABox does not correspond to a first-order structure!

ABox

- An ABox is a finite set of ground atomic facts, syntactically

Example

$\mathcal{A} = \{\texttt{Employee(Jon)}, \texttt{ Project(Winter)}\}$ is an ABox

- The semantics of ABoxes is given by first-order structures.
- An ABox does not correspond to a first-order structure!

ABox

- An ABox is a finite set of ground atomic facts, syntactically

Example

$\mathcal{A} = \{\texttt{Employee(Jon)}, \texttt{ Project(Winter)}\}$ is an ABox

- The semantics of ABoxes is given by first-order structures.
- An ABox does not correspond to a first-order structure!

ABox

- An ABox is a finite set of ground atomic facts, syntactically

Example

$\mathcal{A} = \{\texttt{Employee(Jon)}, \texttt{Project(Winter)}\}$ is an ABox

- The semantics of ABoxes is given by first-order structures.
- An ABox does not correspond to a first-order structure!

ABox Semantics

Definition

A first-order structure $\mathcal{I}$ is a model of an ABox $\mathcal{A}$ if $\mathcal{I} \supseteq \mathcal{A}$

An ABox is an incomplete database i.e., an ABox represents a class of databases.

Example

Given the ABox $\mathcal{A} = \{\texttt{Employee(Jon)}, \texttt{Project(Winter)}\}$

- $\{\{\texttt{Employee(Jon)}, \texttt{Project(Winter)}\}$ is a model of $\mathcal{A}$
- $\{\{\texttt{Employee(Jon)}, \texttt{Employee(Rob)}, \texttt{Project(Winter)}\}$
- $\{\{\texttt{Employee(Jon)}, \texttt{Employee(Rob)}, \texttt{Project(River)}\}$ is not a model of $\mathcal{A}$

ABox Semantics

Definition

A first-order structure $\mathcal{I}$ is a model of an ABox $\mathcal{A}$ if $\mathcal{I} \supseteq \mathcal{A}$

An ABox is an incomplete database i.e., an ABox represents a class of databases.

Example

Given the ABox $\mathcal{A} = \{\texttt{Employee(Jon), Project(Winter)}\}$

- $\{\{\texttt{Employee(Jon), Project(Winter)}\}$ is a model of $\mathcal{A}$
- $\{\{\texttt{Employee(Jon), Employee(Rob), Project(Winter)}\}$
- $\{\{\texttt{Employee(Jon), Employee(Rob), Project(River)}\}$ is not a model of $\mathcal{A}$

ABox Semantics

Definition

A first-order structure $\mathcal{I}$ is a model of an ABox $\mathcal{A}$ if $\mathcal{I} \supseteq \mathcal{A}$

An ABox is an incomplete database i.e., an ABox represents a class of databases.

Example

Given the ABox $\mathcal{A} = \{\texttt{Employee(Jon), Project(Winter)}\}$

- $\{\{\texttt{Employee(Jon), Project(Winter)}\}$ is a model of $\mathcal{A}$
- $\{\{\texttt{Employee(Jon), Employee(Rob), Project(Winter)}\}$
- $\{\{\texttt{Employee(Jon), Employee(Rob), Project(River)}\}$ is not a model of $\mathcal{A}$

ABox Semantics

Definition

A first-order structure $\mathcal{I}$ is a model of an ABox $\mathcal{A}$ if $\mathcal{I} \supseteq \mathcal{A}$

An ABox is an incomplete database i.e., an ABox represents a class of databases.

Example

Given the ABox $\mathcal{A} = \{\texttt{Employee(Jon)}, \texttt{Project(Winter)}\}$

- $\{\{\texttt{Employee(Jon)}, \texttt{Project(Winter)}\}$ is a model of $\mathcal{A}$
- $\{\{\texttt{Employee(Jon)}, \texttt{Employee(Rob)}, \texttt{Project(Winter)}\}$
- $\{\{\texttt{Employee(Jon)}, \texttt{Employee(Rob)}, \texttt{Project(River)}\}$ is not a model of $\mathcal{A}$

Certain Answers

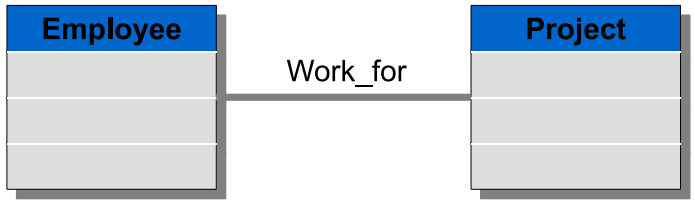
Definition

Given:

- a TBox $\mathcal{T}$, an ABox $\mathcal{A}$ and
- a concept query $\mathcal{Q}$ and an individual a

Question: $(\mathcal{T}, \mathcal{A}) \models \mathcal{Q}(a)$? That is $\forall$ models $\mathcal{I}$ of $(\mathcal{T}, \mathcal{A})$, is it the case that $\mathcal{I} \models \mathcal{Q}(a)$

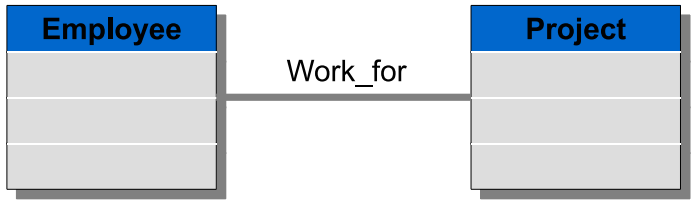
ABox vs Database



- Additional constraint as a standard view over data:
  $\forall x$`Bad-Project(x)` $\leftrightarrow$ `Project(x)` $\wedge \neg \exists y$`Work-for(y,x)`
- Database:
  `Work-for` $= \{$`<Jon, Winter>, <Rob, Winter>`$\}$
  `Project` $= \{$`Winter, River`$\}$
- $\mathcal{Q}(x) :=$ `Bad-Project(x)`
  $\Rightarrow \{$`River`$\}$
- ABox:
  `Work-for` $\supseteq \{$`<Jon, Winter>, <Rob, Winter>`$\}$
  `Project` $\supseteq \{$`Winter, River`$\}$
- $\mathcal{Q}(x) :=$ `Bad-Project(x)`
  $\Rightarrow \{\}$

ABox does not scale down to standard DB answer

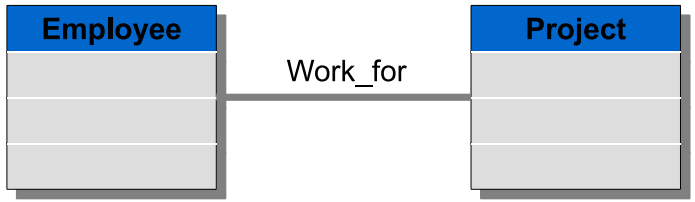
ABox vs Database



- Additional constraint as a standard view over data:
  $\forall x \texttt{Bad-Project(x)} \leftrightarrow \texttt{Project(x)} \wedge \neg\exists y \texttt{Work-for(y,x)}$
- Database:
  $\texttt{Work-for} = \{\texttt{<Jon, Winter>}, \texttt{<Rob, Winter>}\}$
  $\texttt{Project} = \{\texttt{Winter, River}\}$
- $\mathcal{Q}(x) := \texttt{Bad-Project(x)}$
  $\Rightarrow \{\texttt{River}\}$
- ABox:
  $\texttt{Work-for} \supseteq \{\texttt{<Jon, Winter>}, \texttt{<Rob, Winter>}\}$
  $\texttt{Project} \supseteq \{\texttt{Winter, River}\}$
- $\mathcal{Q}(x) := \texttt{Bad-Project(x)}$
  $\Rightarrow \{\}$

ABox does not scale down to standard DB answer

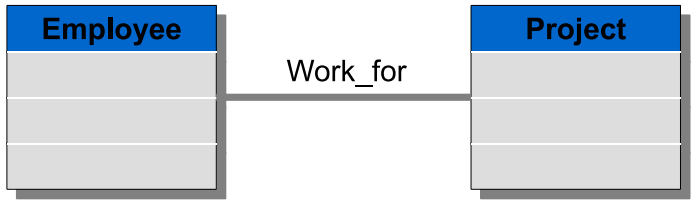
ABox vs Database



- Additional constraint as a standard view over data:
  $\forall x$Bad-Project(x) $\leftrightarrow$ Project(x) $\wedge \neg\exists y$Work-for(y,x)
- Database:
  Work-for $= \{$<Jon, Winter>, <Rob, Winter>$\}$
  Project $= \{$Winter, River$\}$
- $\mathcal{Q}(x) :=$ Bad-Project(x)
  $\Rightarrow \{$River$\}$
- ABox:
  Work-for $\supseteq \{$<Jon, Winter>, <Rob, Winter>$\}$
  Project $\supseteq \{$Winter, River$\}$
- $\mathcal{Q}(x) :=$ Bad-Project(x)
  $\Rightarrow \{\}$

ABox does not scale down to standard DB answer

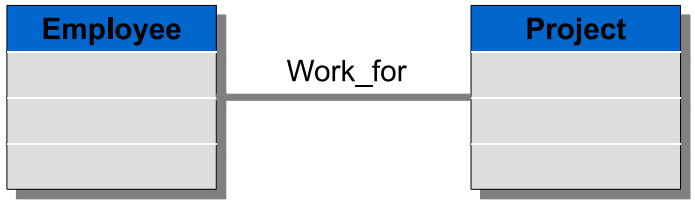
ABox vs Database



- Additional constraint as a standard view over data:
  $\forall x$Bad-Project(x) $\leftrightarrow$ Project(x) $\wedge \neg\exists y$Work-for(y,x)
- Database:
  Work-for $= \{$<Jon, Winter>, <Rob, Winter>$\}$
  Project $= \{$Winter, River$\}$
- $\mathcal{Q}(x) :=$ Bad-Project(x)
  $\Rightarrow \{$River$\}$
- ABox:
  Work-for $\supseteq \{$<Jon, Winter>, <Rob, Winter>$\}$
  Project $\supseteq \{$Winter, River$\}$
- $\mathcal{Q}(x) :=$ Bad-Project(x)
  $\Rightarrow \{\}$

ABox does not scale down to standard DB answer

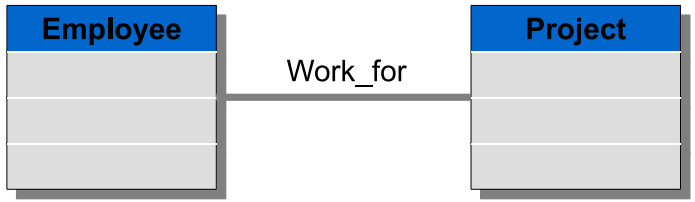
ABox vs Database



- Additional constraint as a standard view over data:
  $\forall x$Bad-Project(x) $\leftrightarrow$ Project(x) $\wedge \neg\exists y$Work-for(y,x)
- Database:
  Work-for $= \{$<Jon, Winter>, <Rob, Winter>$\}$
  Project $= \{$Winter, River$\}$
- $\mathcal{Q}(x) :=$ Bad-Project(x)
  $\Rightarrow \{$River$\}$
- ABox:
  Work-for $\supseteq \{$<Jon, Winter>, <Rob, Winter>$\}$
  Project $\supseteq \{$Winter, River$\}$
- $\mathcal{Q}(x) :=$ Bad-Project(x)
  $\Rightarrow \{\}$

ABox does not scale down to standard DB answer

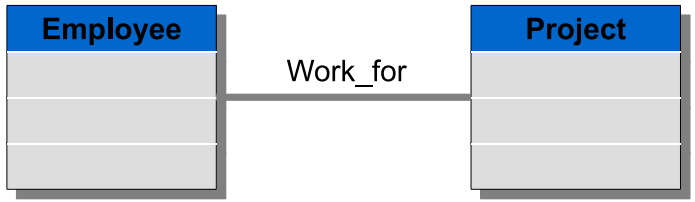
ABox vs Database



- Additional constraint as a standard view over data:
  $\forall x$Bad-Project(x) $\leftrightarrow$ Project(x) $\wedge \neg\exists y$Work-for(y,x)
- Database:
  Work-for $= \{$<Jon, Winter>, <Rob, Winter>$\}$
  Project $= \{$Winter, River$\}$
- $\mathcal{Q}(x) :=$ Bad-Project(x)
  $\Rightarrow \{$River$\}$
- ABox:
  Work-for $\supseteq \{$<Jon, Winter>, <Rob, Winter>$\}$
  Project $\supseteq \{$Winter, River$\}$
- $\mathcal{Q}(x) :=$ Bad-Project(x)
  $\Rightarrow \{\}$

ABox does not scale down to standard DB answer

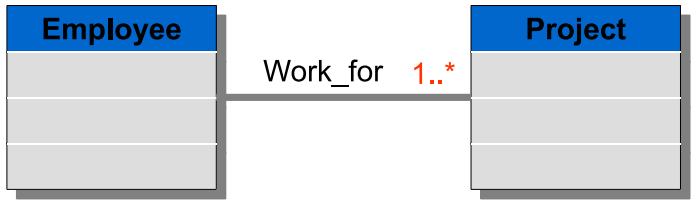
ABox vs Database



- Additional constraint as a standard view over data:
  $\forall x$Bad-Project(x) $\leftrightarrow$ Project(x) $\wedge \neg\exists y$Work-for(y,x)
- Database:
  Work-for $= \{$<Jon, Winter>, <Rob, Winter>$\}$
  Project $= \{$Winter, River$\}$
- $\mathcal{Q}(x) :=$ Bad-Project(x)
  $\Rightarrow \{$River$\}$
- ABox:
  Work-for $\supseteq \{$<Jon, Winter>, <Rob, Winter>$\}$
  Project $\supseteq \{$Winter, River$\}$
- $\mathcal{Q}(x) :=$ Bad-Project(x)
  $\Rightarrow \{\}$

ABox does not scale down to standard DB answer

ABox vs Database



- ABox:

  $\texttt{Work-for} \supseteq \{\ \texttt{<Jon, Winter>}\}$

  $\texttt{Project} \supseteq \{\ \texttt{Winter, River}\}$

- Query as a standard view over database:

  $Q(x) :=\ \texttt{Work-for(y,x)}$ $Q = \Pi_2\ \texttt{Work-for}$

  - $Q =\ \texttt{EVAL}(\Pi_2\ \texttt{Work-for})$

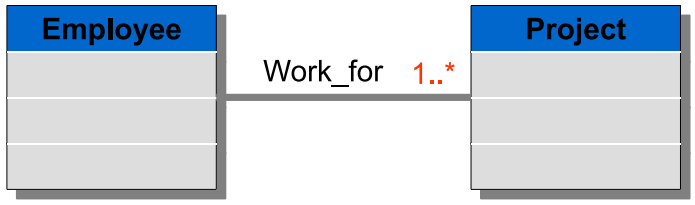
    $\Rightarrow \{\texttt{Winter, River}\}$

  - $Q = \Pi_2(\ \texttt{EVAL}(\ \texttt{Work-for}))$

    $\Rightarrow \{\texttt{Winter}\}$

Queries are not compositional wrt certain answer semantics!

ABox vs Database



- ABox:

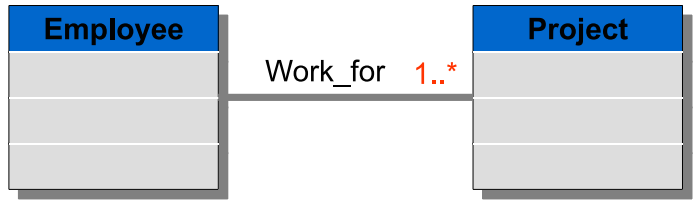
  $\texttt{Work-for} \supseteq \{\ \texttt{<Jon, Winter>}\}$

  $\texttt{Project} \supseteq \{\ \texttt{Winter, River}\}$

- Query as a standard view over database:
  $Q(x) :=\ \texttt{Work-for(y,x)}\ Q = \Pi_2\ \texttt{Work-for}$
  - $Q =\ \texttt{EVAL}(\Pi_2\ \texttt{Work-for})$
    $\Rightarrow \{\texttt{Winter, River}\}$
  - $Q = \Pi_2(\ \texttt{EVAL}(\ \texttt{Work-for}))$
    $\Rightarrow \{\texttt{Winter}\}$

Queries are not compositional wrt certain answer semantics!

ABox vs Database



- ABox:

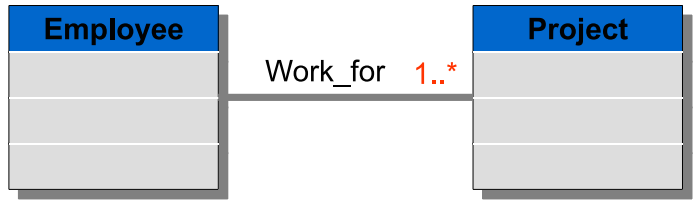
  $\texttt{Work-for} \supseteq \{\ \texttt{<Jon, Winter>}\}$

  $\texttt{Project} \supseteq \{\ \texttt{Winter, River}\}$

- Query as a standard view over database:
  $Q(x) :=\ \texttt{Work-for(y,x)}\ \ Q = \Pi_2\ \texttt{Work-for}$
  - $Q =\ \texttt{EVAL}(\Pi_2\ \texttt{Work-for})$
    $\Rightarrow \{\texttt{Winter, River}\}$
  - $Q = \Pi_2(\ \texttt{EVAL}(\ \texttt{Work-for}))$
    $\Rightarrow \{\texttt{Winter}\}$

Queries are not compositional wrt certain answer semantics!

ABox vs Database



- ABox:

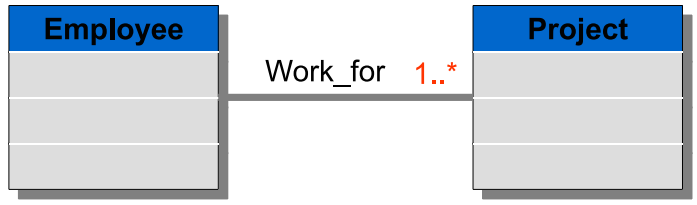
  $\texttt{Work-for} \supseteq \{\ \texttt{<Jon, Winter>}\}$

  $\texttt{Project} \supseteq \{\ \texttt{Winter, River}\}$

- Query as a standard view over database:
  $Q(x) :=\ \texttt{Work-for(y,x)}\ Q = \Pi_2\ \texttt{Work-for}$
  - $Q =\ \texttt{EVAL}(\Pi_2\ \texttt{Work-for})$
    $\Rightarrow \{\texttt{Winter, River}\}$
  - $Q = \Pi_2(\ \texttt{EVAL}(\ \texttt{Work-for}))$
    $\Rightarrow \{\texttt{Winter}\}$

Queries are not compositional wrt certain answer semantics!

ABox vs Database



- ABox:

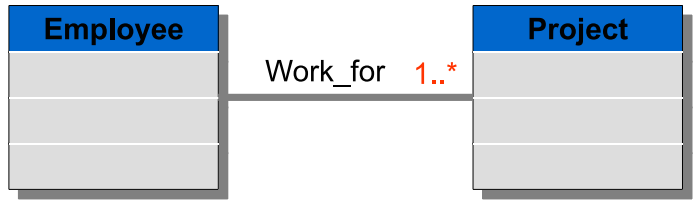
  $\texttt{Work-for} \supseteq \{\ \texttt{<Jon, Winter>}\}$

  $\texttt{Project} \supseteq \{\ \texttt{Winter, River}\}$

- Query as a standard view over database:
  $Q(x) :=\ \texttt{Work-for(y,x)}\ Q = \Pi_2\ \texttt{Work-for}$
  - $Q =\ \texttt{EVAL}(\Pi_2\ \texttt{Work-for})$
    $\Rightarrow \{\texttt{Winter, River}\}$
  - $Q = \Pi_2(\ \texttt{EVAL}(\ \texttt{Work-for}))$
    $\Rightarrow \{\texttt{Winter}\}$

Queries are not compositional wrt certain answer semantics!

ABox vs Database



- ABox:

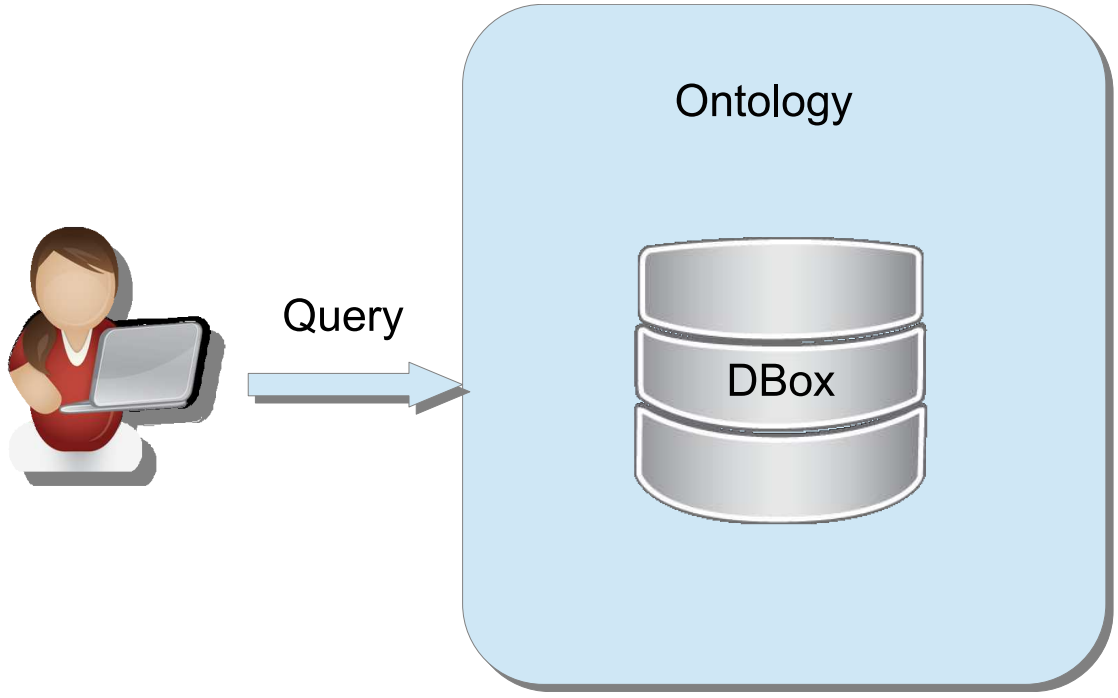
  $\texttt{Work-for} \supseteq \{\ \texttt{<Jon, Winter>}\}$

  $\texttt{Project} \supseteq \{\ \texttt{Winter, River}\}$

- Query as a standard view over database:
  $Q(x) :=\ \texttt{Work-for(y,x)}\ Q = \Pi_2\ \texttt{Work-for}$
  - $Q =\ \texttt{EVAL}(\Pi_2\ \texttt{Work-for})$
    $\Rightarrow \{\texttt{Winter, River}\}$
  - $Q = \Pi_2(\ \texttt{EVAL}(\ \texttt{Work-for}))$
    $\Rightarrow \{\texttt{Winter}\}$

Queries are not compositional wrt certain answer semantics!

Our Proposal



We propose DBox as a way to model complete data

DBox

A DBox is a finite set of ground anomic facts, syntactically

Example

$\mathcal{D} = \{$ `Employee(Jon), Project(Winter)`$\}$ is a DBox

Definition

A first-order structure $\mathcal{I}$ is a model of a DBox $\mathcal{D}$ if

- $\mathcal{I} \supseteq \mathcal{D}$
- for all predicate P in the signature of $\mathcal{D}$, $P^{\mathcal{I}} = \{\bar{a} | P(\bar{a} \in \mathcal{D}\}$

DBox captures the exact semantics of database.

DBox

A DBox is a finite set of ground anomic facts, syntactically

Example

$\mathcal{D} = \{$ `Employee(Jon), Project(Winter)`$\}$ is a DBox

Definition

A first-order structure $\mathcal{I}$ is a model of a DBox $\mathcal{D}$ if

- $\mathcal{I} \supseteq \mathcal{D}$
- for all predicate P in the signature of $\mathcal{D}$, $P^{\mathcal{I}} = \{\bar{a} | P(\bar{a} \in \mathcal{D}\}$

DBox captures the exact semantics of database.

DBox

A DBox is a finite set of ground anomic facts, syntactically

Example

$\mathcal{D} = \{$ `Employee(Jon), Project(Winter)`$\}$ is a DBox

Definition

A first-order structure $\mathcal{I}$ is a model of a DBox $\mathcal{D}$ if

- $\mathcal{I} \supseteq \mathcal{D}$
- for all predicate P in the signature of $\mathcal{D}$, $P^{\mathcal{I}} = \{\bar{a} | P(\bar{a} \in \mathcal{D}\}$

DBox captures the exact semantics of database.

DBox example

Given the DBox $\mathcal{D} = \{$ `Employee(Jon), Project(Winter)`$\}$

- $\{$ `Employee(Jon), Project(Winter)`$\}$ is a model of $\mathcal{D}$
- $\{$ `Employee(Jon), Manager(Rob), Project(Winter)`$\}$
- $\{$ `Employee(Jon), Employee(Rob), Project(Winter)`$\}$ is not a model of $\mathcal{D}$

DBox example

Given the DBox $\mathcal{D} = \{$ `Employee(Jon), Project(Winter)`$\}$

- $\{$ `Employee(Jon), Project(Winter)`$\}$ is a model of $\mathcal{D}$
- $\{$ `Employee(Jon), Manager(Rob), Project(Winter)`$\}$
- $\{$ `Employee(Jon), Employee(Rob), Project(Winter)`$\}$ is not a model of $\mathcal{D}$

DBox example

Given the DBox $\mathcal{D} = \{$ `Employee(Jon), Project(Winter)`$\}$

- $\{$ `Employee(Jon), Project(Winter)`$\}$ is a model of $\mathcal{D}$
- $\{$ `Employee(Jon), Manager(Rob), Project(Winter)`$\}$
- $\{$ `Employee(Jon), Employee(Rob), Project(Winter)`$\}$ is not a model of $\mathcal{D}$

DBox example

Given the DBox $\mathcal{D} = \{$ `Employee(Jon), Project(Winter)`$\}$

- $\{$ `Employee(Jon), Project(Winter)`$\}$ is a model of $\mathcal{D}$
- $\{$ `Employee(Jon), Manager(Rob), Project(Winter)`$\}$
- $\{$ `Employee(Jon), Employee(Rob), Project(Winter)`$\}$ is not a model of $\mathcal{D}$

Certain Answers

Definition

Given:

- an ontology containing: a TBox $\mathcal{T}$, [an ABox $\mathcal{A}$] and
- a DBox $\mathcal{D}$
- a concept query $\mathcal{Q}$ and an individual a

Question: $(\mathcal{T}, [\mathcal{A}], \mathcal{D}) \models \mathcal{Q}(a)$? That is $\forall$ models $\mathcal{I}$ of $(\mathcal{T}, [\mathcal{A}], \mathcal{D})$, is it the case that $\mathcal{I} \models \mathcal{Q}(a)$

Conjunctive Query Answering: DBox vs ABox

- Complexity w.r.t to ABox
  Results by Calvanese, de Giacomo, Lembo, Lenzerini, Rosati, Krisnadhi, Lutz.

Description Logic	KB-type	Combined	Data
$DL-Lite_{[core\|\mathcal{F}]}$	$(\mathcal{T},\mathcal{A})$	NP-complete	in AC^0
$\mathcal{EL}$	$(\mathcal{T},\mathcal{A})$	NP-complete	PTIME-complete

- Complexity w.r.t to DBox Results by Franconi, Seylan, Anglica Ibez-Garca.

Description Logic	KB-type	Combined	Data
$DL-Lite_{[core\|\mathcal{F}]}$	$(\mathcal{T},\mathcal{D})$	coNP-hard	coNP-complete
$DL-Lite_{[core\|\mathcal{F}]}$	$(\mathcal{T},\mathcal{D},[\mathcal{A}])$	EXPTIME-hard	coNP-complete
$\mathcal{EL}$	$(\mathcal{T},\mathcal{A})$	EXPTIME-hard	coNP-hard

DBoxes are natural but expensive formalism.

Conjunctive Query Answering: DBox vs ABox

- Complexity w.r.t to ABox
  Results by Calvanese, de Giacomo, Lembo, Lenzerini, Rosati, Krisnadhi, Lutz.

Description Logic	KB-type	Combined	Data
$DL-Lite_{[core\|\mathcal{F}]}$	$(\mathcal{T},\mathcal{A})$	NP-complete	in AC^0
$\mathcal{EL}$	$(\mathcal{T},\mathcal{A})$	NP-complete	PTIME-complete

- Complexity w.r.t to DBox Results by Franconi, Seylan, Anglica Ibez-Garca.

Description Logic	KB-type	Combined	Data
$DL-Lite_{[core\|\mathcal{F}]}$	$(\mathcal{T},\mathcal{D})$	coNP-hard	coNP-complete
$DL-Lite_{[core\|\mathcal{F}]}$	$(\mathcal{T},\mathcal{D},[\mathcal{A}])$	EXPTIME-hard	coNP-complete
$\mathcal{EL}$	$(\mathcal{T},\mathcal{A})$	EXPTIME-hard	coNP-hard

DBoxes are natural but expensive formalism.

Conjunctive Query Answering: DBox vs ABox

- Complexity w.r.t to ABox
  Results by Calvanese, de Giacomo, Lembo, Lenzerini, Rosati, Krisnadhi, Lutz.

Description Logic	KB-type	Combined	Data
$DL-Lite_{[core\|\mathcal{F}]}$	$(\mathcal{T},\mathcal{A})$	NP-complete	in AC^0
$\mathcal{EL}$	$(\mathcal{T},\mathcal{A})$	NP-complete	PTIME-complete

- Complexity w.r.t to DBox Results by Franconi, Seylan, Anglica Ibez-Garca.

Description Logic	KB-type	Combined	Data
$DL-Lite_{[core\|\mathcal{F}]}$	$(\mathcal{T},\mathcal{D})$	coNP-hard	coNP-complete
$DL-Lite_{[core\|\mathcal{F}]}$	$(\mathcal{T},\mathcal{D},[\mathcal{A}])$	EXPTIME-hard	coNP-complete
$\mathcal{EL}$	$(\mathcal{T},\mathcal{A})$	EXPTIME-hard	coNP-hard

DBoxes are natural but expensive formalism.

Conjunctive Query Answering: DBox vs ABox

- Complexity w.r.t to ABox
  Results by Calvanese, de Giacomo, Lembo, Lenzerini, Rosati, Krisnadhi, Lutz.

Description Logic	KB-type	Combined	Data
$DL-Lite_{[core\|\mathcal{F}]}$	$(\mathcal{T},\mathcal{A})$	NP-complete	in AC^0
$\mathcal{EL}$	$(\mathcal{T},\mathcal{A})$	NP-complete	PTIME-complete

- Complexity w.r.t to DBox Results by Franconi, Seylan, Anglica Ibez-Garca.

Description Logic	KB-type	Combined	Data
$DL-Lite_{[core\|\mathcal{F}]}$	$(\mathcal{T},\mathcal{D})$	coNP-hard	coNP-complete
$DL-Lite_{[core\|\mathcal{F}]}$	$(\mathcal{T},\mathcal{D},[\mathcal{A}])$	EXPTIME-hard	coNP-complete
$\mathcal{EL}$	$(\mathcal{T},\mathcal{A})$	EXPTIME-hard	coNP-hard

DBoxes are natural but expensive formalism.

DBox is expensive

- Do not consider every TBox ⇒ Safe TBox (Lutz & Wolter, 2011)
- Do not consider every query ⇒ Determinacy

DBox is expensive

- Do not consider every TBox ⇒ Safe TBox (Lutz & Wolter, 2011)
- Do not consider every query ⇒ Determinacy

DBox is expensive

- Do not consider every TBox ⇒ Safe TBox (Lutz & Wolter, 2011)
- Do not consider every query ⇒ Determinacy

Determinacy

Given a TBox $\mathcal{T}$ and a DBox $\mathcal{D}$, a query $\mathcal{Q}$ is determined by the DBox predicates $\mathbb{P}_{\mathcal{D}}$ if its answer functionally depends only from the extension of the DBox predicates.

Definition (Determinacy)

Let $\mathcal{I}^i_{(\mathcal{D})}$ and $\mathcal{I}^j_{(\mathcal{D})}$ be any two models of a TBox $\mathcal{T}$ embedding a DBox $\mathcal{D}$. A query $\mathcal{Q}$ is *determined* by the DBox predicates $\mathbb{P}_{\mathcal{D}}$ given the TBox $\mathcal{T}$ *if* the answer of $\mathcal{Q}$ over $\mathcal{I}^i_{(\mathcal{D})}$ is the same as the answer of $\mathcal{Q}$ over $\mathcal{I}^j_{(\mathcal{D})}$.

Checking determinacy of a query with a first-order TBox is reducible to entailment.

Determinacy

Given a TBox $\mathcal{T}$ and a DBox $\mathcal{D}$, a query $\mathcal{Q}$ is determined by the DBox predicates $\mathbb{P}_{\mathcal{D}}$ if its answer functionally depends only from the extension of the DBox predicates.

Definition (Determinacy)

Let $\mathcal{I}^i_{(\mathcal{D})}$ and $\mathcal{I}^j_{(\mathcal{D})}$ be any two models of a TBox $\mathcal{T}$ embedding a DBox $\mathcal{D}$. A query $\mathcal{Q}$ is *determined* by the DBox predicates $\mathbb{P}_{\mathcal{D}}$ given the TBox $\mathcal{T}$ *if* the answer of $\mathcal{Q}$ over $\mathcal{I}^i_{(\mathcal{D})}$ is the same as the answer of $\mathcal{Q}$ over $\mathcal{I}^j_{(\mathcal{D})}$.

Checking determinacy of a query with a first-order TBox is reducible to entailment.

Determinacy

Given a TBox $\mathcal{T}$ and a DBox $\mathcal{D}$, a query $\mathcal{Q}$ is determined by the DBox predicates $\mathbb{P}_{\mathcal{D}}$ if its answer functionally depends only from the extension of the DBox predicates.

Definition (Determinacy)

Let $\mathcal{I}^i_{(\mathcal{D})}$ and $\mathcal{I}^j_{(\mathcal{D})}$ be any two models of a TBox $\mathcal{T}$ embedding a DBox $\mathcal{D}$. A query $\mathcal{Q}$ is *determined* by the DBox predicates $\mathbb{P}_{\mathcal{D}}$ given the TBox $\mathcal{T}$ *if* the answer of $\mathcal{Q}$ over $\mathcal{I}^i_{(\mathcal{D})}$ is the same as the answer of $\mathcal{Q}$ over $\mathcal{I}^j_{(\mathcal{D})}$.

Checking determinacy of a query with a first-order TBox is reducible to entailment.

Determinacy: Example

Let the DBox predicates be $\mathbb{P}_{\mathcal{D}} = \{\texttt{Mother}, \texttt{Father}\}$ and let $\mathcal{T}$ consists of:

1. $\forall x\ \texttt{Mother(x)} \leftrightarrow \exists y\ \texttt{Woman(x)} \wedge \texttt{hasChild(x,y)}$
2. $\forall x\ \texttt{Father(x)} \leftrightarrow \exists y\ \texttt{Man(x)} \wedge \texttt{hasChild(x,y)}$

and let $\mathcal{Q} = \exists x \exists y ((\texttt{Woman(x)} \vee \texttt{Man(x)}) \wedge \texttt{hasChild(x,y)})$

Fact

$\mathcal{Q}$ is determined by the DBox predicates given the TBox.

Determinacy: Example

Let the DBox predicates be $\mathbb{P}_{\mathcal{D}} = \{\, \texttt{Mother}, \texttt{Father}\}$ and let $\mathcal{T}$ consists of:

1. $\forall x\ \texttt{Mother(x)} \leftrightarrow \exists y\ \texttt{Woman(x)} \wedge \texttt{hasChild(x,y)}$
2. $\forall x\ \texttt{Father(x)} \leftrightarrow \exists y\ \texttt{Man(x)} \wedge \texttt{hasChild(x,y)}$

and let $\mathcal{Q} = \exists x \exists y ((\ \texttt{Woman(x)} \vee \texttt{Man(x)}) \wedge \texttt{hasChild(x,y)})$

Fact

$\mathcal{Q}$ is determined by the DBox predicates given the TBox.

Why Determinacy?

- It captures *exactly* the notion of "non ambiguous" answers.
- Consider the models of a TBox $\mathcal{T}$ with a set of DBox predicates $\mathbb{P}_{\mathcal{D}}$. A query $\mathcal{Q}$ determined by $\mathbb{P}_{\mathcal{D}}$ generates an "extended" DBox augmented with the relation associated to the determined query.
- Arbitrary determined queries can be composed and decomposed without affecting the outcome.

Why Determinacy?

- It captures *exactly* the notion of "non ambiguous" answers.
- Consider the models of a TBox $\mathcal{T}$ with a set of DBox predicates $\mathbb{P}_{\mathcal{D}}$. A query $\mathcal{Q}$ determined by $\mathbb{P}_{\mathcal{D}}$ generates an "extended" DBox augmented with the relation associated to the determined query.
- Arbitrary determined queries can be composed and decomposed without affecting the outcome.

Why Determinacy?

- It captures *exactly* the notion of "non ambiguous" answers.
- Consider the models of a TBox $\mathcal{T}$ with a set of DBox predicates $\mathbb{P}_{\mathcal{D}}$. A query $\mathcal{Q}$ determined by $\mathbb{P}_{\mathcal{D}}$ generates an "extended" DBox augmented with the relation associated to the determined query.
- Arbitrary determined queries can be composed and decomposed without affecting the outcome.

Exact reformulation

Theorem (Beth,1953)

Given a TBox $\mathcal{T}$ and a DBox $\mathcal{D}$, a query $\mathcal{Q}$ is determined by the DBox predicates $\mathbb{P}_{\mathcal{D}}$ if and only if there exists an exact reformulation of $\mathcal{Q}$.

Definition

A reformulation $\hat{\mathcal{Q}}$ of a query $\mathcal{Q}$ is exact if $\mathcal{T} \models \forall x.\mathcal{Q}(x) \leftrightarrow \hat{\mathcal{Q}}(x)$.

Exact reformulation

Theorem (Beth,1953)

Given a TBox $\mathcal{T}$ and a DBox $\mathcal{D}$, a query $\mathcal{Q}$ is determined by the DBox predicates $\mathbb{P}_{\mathcal{D}}$ if and only if there exists an exact reformulation of $\mathcal{Q}$.

Definition

A reformulation $\hat{\mathcal{Q}}$ of a query $\mathcal{Q}$ is exact if $\mathcal{T} \models \forall x.\mathcal{Q}(x) \leftrightarrow \hat{\mathcal{Q}}(x)$.

Example

Let the DBox predicates be $\mathbb{P}_{\mathcal{D}} = \{\texttt{Mother}, \texttt{Father}\}$ and let $\mathcal{T}$ consists of:

1. $\forall x\ \texttt{Mother(x)} \leftrightarrow \exists y\ \texttt{Woman(x)} \wedge \texttt{hasChild(x,y)}$
2. $\forall x\ \texttt{Father(x)} \leftrightarrow \exists y\ \texttt{Man(x)} \wedge \texttt{hasChild(x,y)}$

and let $\mathcal{Q} = \exists x \exists y((\texttt{Woman(x)} \vee \texttt{Man(x)}) \wedge \texttt{hasChild(x,y)})$

Fact

$$\mathcal{T} \models \mathcal{Q} \leftrightarrow \exists x\ \mathit{Mother}(x) \vee \mathit{Father}(x).$$

$\exists x\ \mathit{Mother}(x) \vee \mathit{Father}(x)$ is an exact reformulation of $\mathcal{Q}$

Example

Let the DBox predicates be $\mathbb{P}_{\mathcal{D}} = \{\texttt{Mother}, \texttt{Father}\}$ and let $\mathcal{T}$ consists of:

1. $\forall x\ \texttt{Mother(x)} \leftrightarrow \exists y\ \texttt{Woman(x)} \wedge \texttt{hasChild(x,y)}$
2. $\forall x\ \texttt{Father(x)} \leftrightarrow \exists y\ \texttt{Man(x)} \wedge \texttt{hasChild(x,y)}$

and let $\mathcal{Q} = \exists x \exists y ((\texttt{Woman(x)} \vee \texttt{Man(x)}) \wedge \texttt{hasChild(x,y)})$

Fact

$$\mathcal{T} \models \mathcal{Q} \leftrightarrow \exists x\ \mathit{Mother(x)} \vee \mathit{Father(x)}.$$

$\exists x\ \mathit{Mother(x)} \vee \mathit{Father(x)}$ *is an exact reformulation of* $\mathcal{Q}$

Safe-range Queries

- The *safe-range* fragment is a syntactic fragment of FOL
- Intuition: A formula is safe-range if its variables (free and quantified) are bounded by positive predicates or equalities.
  - $\exists x. A(x) \wedge \neg B(x)$ - safe-range
  - $A(x) \vee B(x)$ - safe range
  - $\forall x. C(x)$ - not safe-range
- An open query is *ground safe-range* if its grounding is safe range.
- The safe-range fragment of FOL is equally expressive to the domain independent fragment of FOL and to relational algebra – the core of SQL.

$\Longrightarrow$ SQL can be used for the evaluation of safe-range formulas.

Safe-range Queries

- The *safe-range* fragment is a syntactic fragment of FOL
- Intuition: A formula is safe-range if its variables (free and quantified) are bounded by positive predicates or equalities.
  - $\exists x.\, A(x) \wedge \neg B(x)$ - safe-range
  - $A(x) \vee B(x)$ - safe range
  - $\forall x.C(x)$ - not safe-range
- An open query is *ground safe-range* if its grounding is safe range.
- The safe-range fragment of FOL is equally expressive to the domain independent fragment of FOL and to relational algebra – the core of SQL.

$\Longrightarrow$ SQL can be used for the evaluation of safe-range formulas.

Safe-range Queries

- The *safe-range* fragment is a syntactic fragment of FOL
- Intuition: A formula is safe-range if its variables (free and quantified) are bounded by positive predicates or equalities.
  - $\exists x.\, A(x) \wedge \neg B(x)$ - safe-range
  - $A(x) \vee B(x)$ - safe range
  - $\forall x. C(x)$ - not safe-range
- An open query is *ground safe-range* if its grounding is safe range.
- The safe-range fragment of FOL is equally expressive to the domain independent fragment of FOL and to relational algebra – the core of SQL.

$\Longrightarrow$ SQL can be used for the evaluation of safe-range formulas.

Safe-range Queries

- The *safe-range* fragment is a syntactic fragment of FOL
- Intuition: A formula is safe-range if its variables (free and quantified) are bounded by positive predicates or equalities.
  - $\exists x.\, A(x) \wedge \neg B(x)$ - safe-range
  - $A(x) \vee B(x)$ - safe range
  - $\forall x.C(x)$ - not safe-range
- An open query is *ground safe-range* if its grounding is safe range.
- The safe-range fragment of FOL is equally expressive to the domain independent fragment of FOL and to relational algebra – the core of SQL.

$\Longrightarrow$ SQL can be used for the evaluation of safe-range formulas.

Safe-range Queries

- The *safe-range* fragment is a syntactic fragment of FOL
- Intuition: A formula is safe-range if its variables (free and quantified) are bounded by positive predicates or equalities.
  - $\exists x.\, A(x) \wedge \neg B(x)$ - safe-range
  - $A(x) \vee B(x)$ - safe range
  - $\forall x.C(x)$ - not safe-range
- An open query is *ground safe-range* if its grounding is safe range.
- The safe-range fragment of FOL is equally expressive to the domain independent fragment of FOL and to relational algebra – the core of SQL.

$\Longrightarrow$ SQL can be used for the evaluation of safe-range formulas.

Safe-range Queries

- The *safe-range* fragment is a syntactic fragment of FOL
- Intuition: A formula is safe-range if its variables (free and quantified) are bounded by positive predicates or equalities.
  - $\exists x.\, A(x) \wedge \neg B(x)$ - safe-range
  - $A(x) \vee B(x)$ - safe range
  - $\forall x.C(x)$ - not safe-range
- An open query is *ground safe-range* if its grounding is safe range.
- The safe-range fragment of FOL is equally expressive to the domain independent fragment of FOL and to relational algebra – the core of SQL.

$\Longrightarrow$ SQL can be used for the evaluation of safe-range formulas.

Safe-range Queries

- The *safe-range* fragment is a syntactic fragment of FOL
- Intuition: A formula is safe-range if its variables (free and quantified) are bounded by positive predicates or equalities.
  - $\exists x.\, A(x) \wedge \neg B(x)$ - safe-range
  - $A(x) \vee B(x)$ - safe range
  - $\forall x.C(x)$ - not safe-range
- An open query is *ground safe-range* if its grounding is safe range.
- The safe-range fragment of FOL is equally expressive to the domain independent fragment of FOL and to relational algebra – the core of SQL.

$\Longrightarrow$ SQL can be used for the evaluation of safe-range formulas.

Safe-range Queries

- The *safe-range* fragment is a syntactic fragment of FOL
- Intuition: A formula is safe-range if its variables (free and quantified) are bounded by positive predicates or equalities.
  - $\exists x.\, A(x) \wedge \neg B(x)$ - safe-range
  - $A(x) \vee B(x)$ - safe range
  - $\forall x.C(x)$ - not safe-range
- An open query is *ground safe-range* if its grounding is safe range.
- The safe-range fragment of FOL is equally expressive to the domain independent fragment of FOL and to relational algebra – the core of SQL.

$\Longrightarrow$ SQL can be used for the evaluation of safe-range formulas.

Safe-range Queries

- The *safe-range* fragment is a syntactic fragment of FOL
- Intuition: A formula is safe-range if its variables (free and quantified) are bounded by positive predicates or equalities.
  - $\exists x.\, A(x) \wedge \neg B(x)$ - safe-range
  - $A(x) \vee B(x)$ - safe range
  - $\forall x.C(x)$ - not safe-range
- An open query is *ground safe-range* if its grounding is safe range.
- The safe-range fragment of FOL is equally expressive to the domain independent fragment of FOL and to relational algebra – the core of SQL.

$\Longrightarrow$ SQL can be used for the evaluation of safe-range formulas.

Query Answering: from entailment to model checking

Theorem

Let $\mathcal{T}$ be an ontology, $\mathcal{Q}$ be a query, $\mathcal{D}$ be a consistent DBox for $\mathcal{T}$, and AD the set of all individuals in $\mathcal{D}$ and $\mathcal{Q}$.
If $\widehat{\mathcal{Q}}$ is

- *an* **exact** *reformulation of $\mathcal{Q}$,*
- **safe-range**,

then

$$Ans(\mathcal{Q}, \mathcal{D}, \mathcal{T}) = Ans(\widehat{\mathcal{Q}}, \mathcal{D}, \{\}) = \{a \mid \langle AD, \mathcal{D}\rangle \models \widehat{\mathcal{Q}}(a)\}$$

The original query answering problem (based on entailment) is reduced to the problem of checking the validity of the reformulation $\widehat{\mathcal{Q}}$ over the *single* interpretation given by the DBox with the active domain (model checking problem), which can be executed by an SQL engine.

Query Answering: from entailment to model checking

Theorem

Let $\mathcal{T}$ be an ontology, $\mathcal{Q}$ be a query, $\mathcal{D}$ be a consistent DBox for $\mathcal{T}$, and AD the set of all individuals in $\mathcal{D}$ and $\mathcal{Q}$.
If $\widehat{\mathcal{Q}}$ is

- *an* **exact** *reformulation of $\mathcal{Q}$,*
- **safe-range**,

then

$$Ans(\mathcal{Q}, \mathcal{D}, \mathcal{T}) = Ans(\widehat{\mathcal{Q}}, \mathcal{D}, \{\}) = \{a \mid \langle AD, \mathcal{D}\rangle \models \widehat{\mathcal{Q}}(a)\}$$

The original query answering problem (based on entailment) is reduced to the problem of checking the validity of the reformulation $\widehat{\mathcal{Q}}$ over the *single* interpretation given by the DBox with the active domain (model checking problem), which can be executed by an SQL engine.

Problem Statment

Given

- an ontology $\mathcal{T}$ in $\mathcal{L}$;
- a DBox $\mathcal{D}$ in $\mathcal{L}$;
- a concept query $\mathcal{Q}$ in $\mathcal{L}$.

We need to solve the following PROBLEM:

- find a first-order logic **safe-range exact** reformulation of $\mathcal{Q}$ expressed in terms of DBox predicates.

Problem Statment

Given

- an ontology $\mathcal{T}$ in $\mathcal{L}$;
- a DBox $\mathcal{D}$ in $\mathcal{L}$;
- a concept query $\mathcal{Q}$ in $\mathcal{L}$.

We need to solve the following PROBLEM:

- find a first-order logic **safe-range exact** reformulation of $\mathcal{Q}$ expressed in terms of DBox predicates.

Semantic Characterisation Theorem

Given a set of database predicates $\mathbb{P}_{\mathcal{DB}}$,
a domain independent ontology $\mathcal{T}$,
and a query $\mathcal{Q}$,

a domain independent exact reformulation $\widehat{\mathcal{Q}}$ of $\mathcal{Q}$ over $\mathbb{P}_{\mathcal{DB}}$ under $\mathcal{T}$ exists

if and only if

$\mathcal{Q}$ is implicitly definable from $\mathbb{P}_{\mathcal{DB}}$ under $\mathcal{T}$
and it is domain independent with respect to $\mathcal{T}$.

Constructive Theorem

If:

1. $\mathcal{T} \cup \widetilde{\mathcal{T}} \models \forall \mathbb{X}.\, \mathcal{Q}_{[\mathbb{X}]} \leftrightarrow \widetilde{\mathcal{Q}}_{[\mathbb{X}]}$
   (that is, $\mathcal{Q}_{[\mathbb{X}]}$ is implicitly definable),
2. $\mathcal{Q}$ is safe-range
   (that is, $\mathcal{Q}$ is domain independent),
3. $\mathcal{T}$ is safe-range
   (that is, $\mathcal{T}$ is domain independent),

then there exists an exact reformulation $\widehat{\mathcal{Q}}$ of $\mathcal{Q}$ as a safe-range query in $\mathcal{FOL}(\mathbb{C}, \mathbb{P})$ over $\mathbb{P}_{\mathcal{DB}}$ under $\mathcal{T}$, that can be obtained constructively.

Oops! But databases are finite

Beth's theorem fails if we consider only models with a finite interpretation of database predicates

Let $\mathbb{P} = \{P, R, A\}$, $\mathbb{P}_{\mathcal{DB}} = \{P, R\}$,
$\mathcal{T}$ consists of:

1. $\forall x, y, z.\, R(x, y) \wedge R(x, z) \rightarrow y = z$,
2. $\forall x, y.\, R(x, y) \rightarrow \exists z.\, R(z, x)$,
3. $(\forall x, y.\, R(x, y) \rightarrow \exists z.\, R(y, z)) \rightarrow (\forall x.\, A(x) \leftrightarrow P(x))$

Fact

- $\forall x, y.\, R(x, y) \rightarrow \exists z.\, R(y, z)$ *is entailed from the first two formulas only over finite interpretations of* R.
- *Query* $\mathcal{Q} = A(x)$ *is finitely determined by DBox predicates, but it is not determined under models with an unrestricted interpretation of* R
- *This knowledge base does not enjoy finitely controllable determinacy.*
- *The characterization theorem fails too.*

Oops! But databases are finite

Beth's theorem fails if we consider only models with a finite interpretation of database predicates

Let $\mathbb{P} = \{P, R, A\}$, $\mathbb{P}_{\mathcal{DB}} = \{P, R\}$,

$\mathcal{T}$ consists of:

1. $\forall x, y, z.\, R(x, y) \wedge R(x, z) \rightarrow y = z$,
2. $\forall x, y.\, R(x, y) \rightarrow \exists z.\, R(z, x)$,
3. $(\forall x, y.\, R(x, y) \rightarrow \exists z.\, R(y, z)) \rightarrow (\forall x.\, A(x) \leftrightarrow P(x))$

Fact

- *$\forall x, y.\, R(x, y) \rightarrow \exists z.\, R(y, z)$ is entailed from the first two formulas only over finite interpretations of R.*
- *Query $\mathcal{Q} = A(x)$ is finitely determined by DBox predicates, but it is not determined under models with an unrestricted interpretation of R*
- *This knowledge base does not enjoy finitely controllable determinacy.*
- *The characterization theorem fails too.*

Oops! But databases are finite

Beth's theorem fails if we consider only models with a finite interpretation of database predicates

Let $\mathbb{P} = \{P, R, A\}$, $\mathbb{P}_{\mathcal{DB}} = \{P, R\}$,

$\mathcal{T}$ consists of:

1. $\forall x, y, z.\, R(x, y) \wedge R(x, z) \rightarrow y = z$,
2. $\forall x, y.\, R(x, y) \rightarrow \exists z.\, R(z, x)$,
3. $(\forall x, y.\, R(x, y) \rightarrow \exists z.\, R(y, z)) \rightarrow (\forall x.\, A(x) \leftrightarrow P(x))$

Fact

- $\forall x, y.\, R(x, y) \rightarrow \exists z.\, R(y, z)$ *is entailed from the first two formulas only over finite interpretations of* R.
- *Query* $\mathcal{Q} = A(x)$ *is finitely determined by DBox predicates, but it is not determined under models with an unrestricted interpretation of* R
- *This knowledge base does not enjoy finitely controllable determinacy.*
- *The characterization theorem fails too.*

Finitely controllable determinacy

Definition

Given a set of database predicates $\mathbb{P}_{\mathcal{DB}}$, an ontology $\mathcal{T}$, we say that the determinacy of a query $\mathcal{Q}$ is finitely controllable if, $\mathcal{Q}$ is determined iff $\mathcal{Q}$ is determined under the models of $\mathcal{T}$ where the extensions of database predicates are finite.

Finitely controllable determinacy guarantees the completeness of the characterization theorem

Definition

A fragment $\mathcal{L}$ is said to have finitely controllable determinacy property if the determinacy of every query is finitely controllable.

Finitely controllable determinacy

Definition

Given a set of database predicates $\mathbb{P}_{\mathcal{DB}}$, an ontology $\mathcal{T}$, we say that the determinacy of a query $\mathcal{Q}$ is finitely controllable if, $\mathcal{Q}$ is determined iff $\mathcal{Q}$ is determined under the models of $\mathcal{T}$ where the extensions of database predicates are finite.

Finitely controllable determinacy guarantees the completeness of the characterization theorem

Definition

A fragment $\mathcal{L}$ is said to have finitely controllable determinacy property if the determinacy of every query is finitely controllable.

The logic fragment that we want

- Domain independence/Safe-range
- Finite controllable determinacy
- As expressive as possible

The logic fragment that we want

- Domain independence/Safe-range
- Finite controllable determinacy
- As expressive as possible

The logic fragment that we want

- Domain independence/Safe-range
- Finite controllable determinacy
- As expressive as possible

Syntax and semantics of $\mathcal{ALCHOIQ}$ concepts and roles

Syntax	Semantics
A	$A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$
$\{o\}$	$\{o\}^{I} \subseteq \Delta^{I}$
P	$P^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$
P^{-}	$\{(y,x) \mid (x,y) \in P^{\mathcal{I}}\}$
$\neg C$	$\Delta^{\mathcal{I}} \backslash C^{\mathcal{I}}$
$C \sqcap D$	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$
$C \sqcup D$	$C^{\mathcal{I}} \cup D^{\mathcal{I}}$
$\geq nR$	$\{x \mid \#(\{y \mid (x,y) \in R^{\mathcal{I}}\}) \geq n\}$
$\geq nR.C$	$\{x \mid \#(\{y \mid (x,y) \in R^{\mathcal{I}}\} \cap C^{\mathcal{I}}) \geq n\}$

Syntax of $\mathcal{ALCHOIQ}_{GN}$ concepts and roles

$$
\begin{array}{lcl}
R & ::= & P \mid P^- \\
B & ::= & A \mid \{o\} \mid \geq nR \\
C & ::= & B \mid \geq nR.C \mid \geq nR.\neg C \mid B \sqcap \neg C \mid C \sqcap D \mid C \sqcup D
\end{array}
$$

In GN fragments, only guarded negations are allowed.

$\mathcal{ALCHOIQ}_{GN}$

Syntax of $\mathcal{ALCHOIQ}_{GN}$ concepts and roles

$$
\begin{array}{lcl}
R & ::= & P \mid P^- \\
B & ::= & A \mid \{o\} \mid \geq nR \\
C & ::= & B \mid \geq nR.C \mid \geq nR.\neg C \mid B \sqcap \neg C \mid C \sqcap D \mid C \sqcup D
\end{array}
$$

In GN fragments, only guarded negations are allowed.

Why does $\mathcal{ALCHOIQ}_{GN}$ make sense ?

- Non-guarded negation should not appear in a cleanly designed ontology, and, if present, it should be fixed.
- The use of absolute negative information:
  "a non-male is a female"
  $\neg$ male $\sqsubseteq$ female;
  is not meaningful in conceptual modelling, since the subsumer includes all sorts of objects in the universe.
  Only guarded negative information in the subsumee should be allowed:
  "a non-male person is a female"
  person $\sqcap \neg$ male $\sqsubseteq$ female.

Why does $\mathcal{ALCHOIQ}_{GN}$ make sense ?

- Non-guarded negation should not appear in a cleanly designed ontology, and, if present, it should be fixed.
- The use of absolute negative information:
  "a non-male is a female"
  $\neg$ male $\sqsubseteq$ female;
  is not meaningful in conceptual modelling, since the subsumer includes all sorts of objects in the universe.
  Only guarded negative information in the subsumee should be allowed:
  "a non-male person is a female"
  person $\sqcap \neg$ male $\sqsubseteq$ female.

$\mathcal{ALCHOIQ}_{GN}$

- $\mathcal{ALCHOIQ}_{GN}$ TBoxes and concept queries are domain independent.
- Expressive power equivalence

Theorem

The domain independent fragment of $\mathcal{ALCHOIQ}$ and $\mathcal{ALCHOIQ}_{GN}$ are equally expressive.

- Finitely controllable determinacy

Theorem

$\mathcal{ALCHOIQ}_{GN}$ TBoxes with concept queries have finitely controllable determinacy.

$\mathcal{ALCHOIQ}_{GN}$

- $\mathcal{ALCHOIQ}_{GN}$ TBoxes and concept queries are domain independent.
- Expressive power equivalence

Theorem

The domain independent fragment of $\mathcal{ALCHOIQ}$ and $\mathcal{ALCHOIQ}_{GN}$ are equally expressive.

- Finitely controllable determinacy

Theorem

$\mathcal{ALCHOIQ}_{GN}$ TBoxes with concept queries have finitely controllable determinacy.

- $\mathcal{ALCHOIQ}_{GN}$ TBoxes and concept queries are domain independent.
- Expressive power equivalence

Theorem

The domain independent fragment of $\mathcal{ALCHOIQ}$ and $\mathcal{ALCHOIQ}_{GN}$ are equally expressive.

- Finitely controllable determinacy

Theorem

$\mathcal{ALCHOIQ}_{GN}$ TBoxes with concept queries have finitely controllable determinacy.

- $\mathcal{ALCHOIQ}_{GN}$ TBoxes and concept queries are domain independent.
- Expressive power equivalence

Theorem

The domain independent fragment of $\mathcal{ALCHOIQ}$ and $\mathcal{ALCHOIQ}_{GN}$ are equally expressive.

- Finitely controllable determinacy

Theorem

$\mathcal{ALCHOIQ}_{GN}$ TBoxes with concept queries have finitely controllable determinacy.

$\mathcal{ALCHOIQ}_{GN}$

- $\mathcal{ALCHOIQ}_{GN}$ TBoxes and concept queries are domain independent.
- Expressive power equivalence

Theorem

The domain independent fragment of $\mathcal{ALCHOIQ}$ and $\mathcal{ALCHOIQ}_{GN}$ are equally expressive.

- Finitely controllable determinacy

Theorem

$\mathcal{ALCHOIQ}_{GN}$ TBoxes with concept queries have finitely controllable determinacy.

A complete procedure

Input: An $\mathcal{ALCHOIQ}_{GN}$ TBox $\mathcal{T}$, a concept query $\mathcal{Q}$ in $\mathcal{ALCHOIQ}_{GN}$, and a database signature (database atomic concepts and roles).

1. Check the implicit definability of the query $\mathcal{Q}$ by testing $\mathcal{T} \cup \widetilde{\mathcal{T}} \models \mathcal{Q} \equiv \widetilde{\mathcal{Q}}$ using a standard OWL2 reasoner ($\mathcal{ALCHOIQ}_{GN}$ is a sub-language of OWL2). If it is the case then continue.
2. Compute the Craig interpolant $\widehat{\mathcal{Q}}(x)$ based on the tableau proof of $((\bigwedge \mathcal{T}) \wedge \mathcal{Q}(x)) \rightarrow ((\bigwedge \widetilde{\mathcal{T}}) \rightarrow \widetilde{\mathcal{Q}}(x))$
3. For each free variable x which is not bounded by any positive predicate in $\widehat{\mathcal{Q}}_{[\mathbb{X}]}$ do $\widehat{\mathcal{Q}}_{[\mathbb{X}]} := \widehat{\mathcal{Q}}_{[\mathbb{X}]} \wedge Adom_{\widehat{\mathcal{Q}}}(x)$

Output: A safe-range reformulation $\widehat{\mathcal{Q}}$ expressed over the database signature.

A complete procedure

Input: An $\mathcal{ALCHOIQ}_{GN}$ TBox $\mathcal{T}$, a concept query $\mathcal{Q}$ in $\mathcal{ALCHOIQ}_{GN}$, and a database signature (database atomic concepts and roles).

1. Check the implicit definability of the query $\mathcal{Q}$ by testing $\mathcal{T} \cup \widetilde{\mathcal{T}} \models \mathcal{Q} \equiv \widetilde{\mathcal{Q}}$ using a standard OWL2 reasoner ($\mathcal{ALCHOIQ}_{GN}$ is a sub-language of OWL2). If it is the case then continue.
2. Compute the Craig interpolant $\widehat{\mathcal{Q}}(x)$ based on the tableau proof of $((\bigwedge \mathcal{T}) \wedge \mathcal{Q}(x)) \rightarrow ((\bigwedge \widetilde{\mathcal{T}}) \rightarrow \widetilde{\mathcal{Q}}(x))$
3. For each free variable x which is not bounded by any positive predicate in $\widehat{\mathcal{Q}}_{[\mathbb{X}]}$ do $\widehat{\mathcal{Q}}_{[\mathbb{X}]} := \widehat{\mathcal{Q}}_{[\mathbb{X}]} \wedge Adom_{\widehat{\mathcal{Q}}}(x)$

Output: A safe-range reformulation $\widehat{\mathcal{Q}}$ expressed over the database signature.

A complete procedure

Input: An $\mathcal{ALCHOIQ}_{GN}$ TBox $\mathcal{T}$, a concept query $\mathcal{Q}$ in $\mathcal{ALCHOIQ}_{GN}$, and a database signature (database atomic concepts and roles).

1. Check the implicit definability of the query $\mathcal{Q}$ by testing $\mathcal{T} \cup \widetilde{\mathcal{T}} \models \mathcal{Q} \equiv \widetilde{\mathcal{Q}}$ using a standard OWL2 reasoner ($\mathcal{ALCHOIQ}_{GN}$ is a sub-language of OWL2). If it is the case then continue.
2. Compute the Craig interpolant $\widehat{\mathcal{Q}}(x)$ based on the tableau proof of $((\bigwedge \mathcal{T}) \wedge \mathcal{Q}(x)) \rightarrow ((\bigwedge \widetilde{\mathcal{T}}) \rightarrow \widetilde{\mathcal{Q}}(x))$
3. For each free variable x which is not bounded by any positive predicate in $\widehat{\mathcal{Q}}_{[\mathbb{X}]}$ do $\widehat{\mathcal{Q}}_{[\mathbb{X}]} := \widehat{\mathcal{Q}}_{[\mathbb{X}]} \wedge Adom_{\widehat{\mathcal{Q}}}(x)$

Output: A safe-range reformulation $\widehat{\mathcal{Q}}$ expressed over the database signature.

Conclusion and Future Work

Conclusion:

- We introduced a framework to compute an exact reformulation of a concept query under a description logic ontology over some set of concept and role names (DBox predicates).
- We found the conditions which guarantee that a safe-range exact reformulation exists.
- We proved that such safe-range exact reformulation being evaluated as a relational algebra query over the DBox give the same answer as the original query under the ontology.
- An application of the framework to description logics was studied.

Future Work:

- Study optimisations of reformulations.
  - How to choose the *best* reformulation in terms of query evaluation?

Conclusion and Future Work

Conclusion:

- We introduced a framework to compute an exact reformulation of a concept query under a description logic ontology over some set of concept and role names (DBox predicates).
- We found the conditions which guarantee that a safe-range exact reformulation exists.
- We proved that such safe-range exact reformulation being evaluated as a relational algebra query over the DBox give the same answer as the original query under the ontology.
- An application of the framework to description logics was studied.

Future Work:

- Study optimisations of reformulations.
  - How to choose the *best* reformulation in terms of query evaluation?