Datalog$^{\pm}$: A Unifying Framework for Ontological Reasoning and Query Answering

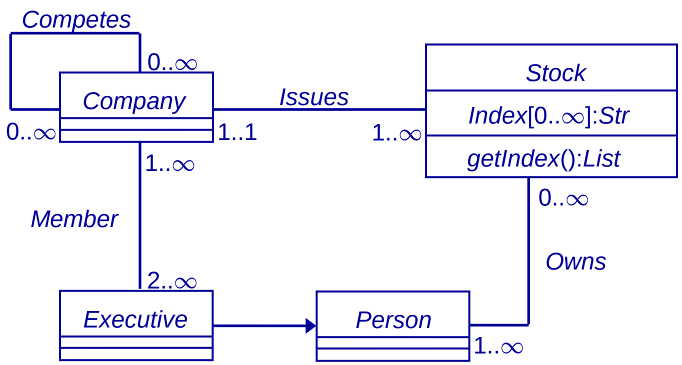
Georg Gottlob and Andreas Pieris

Department of Computer Science

University of Oxford

Ontology, Rules, and Logic Programming for Reasoning and Applications,
October 31, 2013

The Need for Querying Ontologies



UML Class Diagrams

student::person

person[age $*\!\Rightarrow$ number]

person[age {0:1} $*\!\Rightarrow$ number]

person[name {1:$*$} $*\!\Rightarrow$ string]

F-Logic

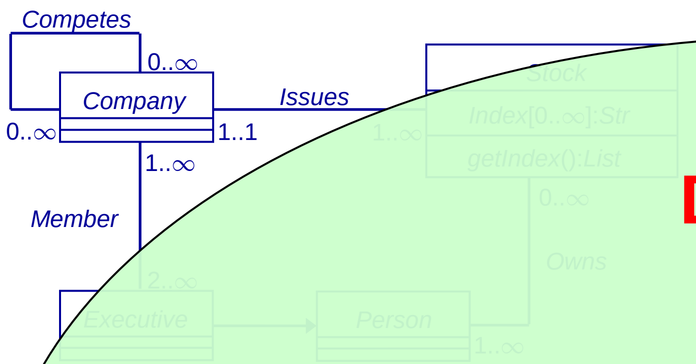
$$
\begin{aligned}
\mathit{Speaker} &\sqsubseteq \mathit{Person} \sqcap \exists \mathit{gives}.\mathit{Talk} \\
\mathit{Tutorial} &\sqsubseteq \forall \mathit{attendedBy}.\mathit{Student} \\
\mathit{attendedBy} &\sqsubseteq \mathit{attended}^{-1} \\
\mathit{Speaker} \sqcap \mathit{Student} &\sqsubseteq \bot
\end{aligned}
$$

Description Logics

```
PREFIX abc: ⟨http://example.com/exampleOntology#⟩
SELECT ?capital ?country
WHERE {
    ?x abc:cityname ?capital ;
    ?x abc:isCapitalOf ?y .
    ?x abc:countryname ?country ;
    ?x abc:isInContinent abc:Africa . }
```

SPARQL (with reasoning capabalities)

A Unifying Framework for Ontology Querying



UML Class Diagrams

student::person
person[age *⇒ number]
person[age {0:1} *⇒ number]
person[name {1:*} *⇒ string]

F-Logic

Datalog±

- Logical foundations, semantics and decidability
- Complexity results for ontological query answering
- Identification of tractable fragments

PREFIX abc: ⟨http://example.com/exampleOntology#⟩
SELECT ?capital ?country
WHERE {
?x abc:cityname ?capital ;
?y abc:isCapitalOf ?y .
?x abc:countryname ?country ;
?x abc:isInContinent abc:Africa . }

Tutorial ⊑ ∀attendedBy.Student
attendedBy ⊑ attended^{-1}
Speaker ⊓ Student ⊑ ⊥

Description Logics

SPARQL (with reasoning capabalities)

The Datalog$^{\pm}$ Family

- Datalog: $\forall X \forall Y$ *fatherOf*(X,Y) $\wedge$ *person*(Y) $\rightarrow$ *person*(X)

- Extend Datalog by allowing in the head:

  Existential quantification (TGDs) – $\forall X$ *person*(X) $\rightarrow \exists Y$ *fatherOf*(Y,X) $\wedge$ *person*(Y)

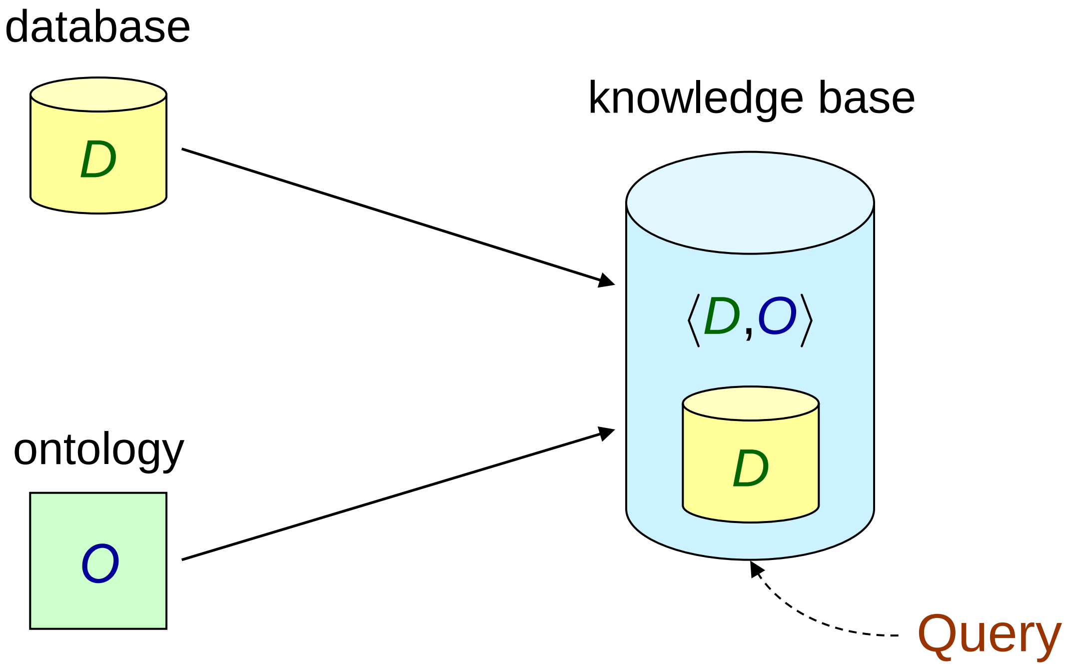
  Equality predicate (EGDs) – $\forall X \forall Y \forall Z$ *fatherOf*(Y,X) $\wedge$ *fatherOf*(Z,X) $\rightarrow$ Y = Z

  Constant false (Negative Constraints) – $\forall X \forall Y$ *fatherOf*(Y,X) $\wedge$ *motherOf*(Y,X) $\rightarrow \perp$

- But query answering under Datalog[$\exists$] is already undecidable
  see, e.g., [Beeri & Vardi, ICALP 1981] and [Calì, G. & Kifer, KR 2008]

- Datalog[$\exists$,=,$\perp$] is syntactically restricted $\rightarrow$ **Datalog$^{\pm}$**

Ontological Query Answering



Ontological Query Answering: Example

D = *person*(john)

O =

$\forall P\ \textit{person}(P) \rightarrow \exists F\ \textit{fatherOf}(F,P)$

$\forall F \forall P\ \textit{fatherOf}(F,P) \rightarrow \textit{person}(F)$

each model of $\langle D, O\rangle$ $\supseteq$ …*fatherOf*(*z*,john) *person*(*z*)…

$\exists X\ \textit{fatherOf}(X,\text{john}) \wedge \textit{person}(X)$ ✓

$\exists X\ \textit{fatherOf}(\text{john},X)$ ✗

Decidable Languages

- Guarded Datalog[∃]

- Linear Datalog[∃]

- Sticky Datalog[∃]

Guarded Datalog[$\exists$]

- All $\forall$-variables occur in one body atom – guard atom

$$\forall E \forall D\ \mathit{employee}(E) \wedge \mathit{managerOf}(E,D) \rightarrow \exists F\ \mathit{supervisorOf}(E,F)$$

guard

- Models of finite treewidth $\Rightarrow$ decidability of query answering

  [Calì, G. & Kifer, KR 2008, JAIR 2013] related to [Andréka, Németi & van Benthem, J. Philosophical Logic 1998] and [Grädel, J. Symb. Log. 1999]

- Query answering is PTIME-complete in data complexity

[Calì, G. & Kifer, KR 2008, JAIR 2013]

Linear Datalog[$\exists$]

- Rules with just one body-atom

$$\forall E\ \mathit{supervisorOf}(E,E) \rightarrow \exists D\ \mathit{managerOf}(E,D)$$

- Query answering is first-order rewritable

- Same data complexity as plain SQL queries (in AC_0)

[Calì, G. & Lukasiewicz, J. Web. Sem. 2012]

Sticky Datalog[∃]

- Join-variables "stick" to the inferred atoms

D is the director of P, and (s)he is an expert in A

$$\forall P \forall N \forall A\ \mathit{project}(P,N) \wedge \mathit{inArea}(P,A) \rightarrow \exists D\ \mathit{directorOf}(D,P,A)$$

N is the name of project P

- Query answering is first-order rewritable

- Same data complexity as plain SQL queries (in AC_0)

[Calì, G. & Pieris, Artif. Intell. 2012]

Datalog[$\exists$,=,$\perp$]

- Every decidable Datalog[$\exists$] language can be enriched with:
  - Non-Conflicting EGDs – no interaction with TGDs

    $\forall X \forall Y \forall Z\ \mathit{fatherOf}(Y,X) \wedge \mathit{fatherOf}(Z,X) \rightarrow Y = Z$

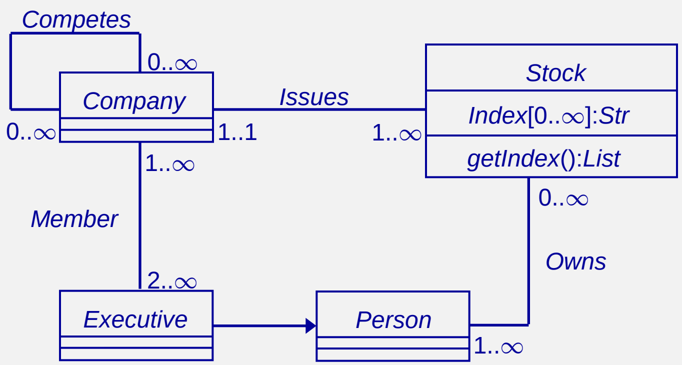
  - Negative constraints

    $\forall X \forall Y\ \mathit{fatherOf}(Y,X) \wedge \mathit{motherOf}(Y,X) \rightarrow \perp$

- The complexity of query answering remains the same

see, e.g., [Calì, G. & Pieris, Artif. Intell. 2012]

A Unifying Framework for Ontology Querying



UML Class Diagrams

student::person

person[age $\ast\!\Rightarrow$ number]

person[age {0:1} $\ast\!\Rightarrow$ number]

person[name {1:$\ast$} $\ast\!\Rightarrow$ string]

F-Logic

$$
\begin{aligned}
\mathit{Speaker} &\sqsubseteq \mathit{Person} \sqcap \exists \mathit{gives}.\mathit{Talk} \\
\mathit{Tutorial} &\sqsubseteq \forall \mathit{attendedBy}.\mathit{Student} \\
\mathit{attendedBy} &\sqsubseteq \mathit{attended}^{-1} \\
\mathit{Speaker} \sqcap \mathit{Student} &\sqsubseteq \bot
\end{aligned}
$$

Description Logics

```
PREFIX abc: ⟨http://example.com/exampleOntology#⟩
SELECT ?capital ?country
WHERE {
    ?x abc:cityname ?capital ;
    ?x abc:isCapitalOf ?y .
    ?x abc:countryname ?country ;
    ?x abc:isInContinent abc:Africa . }
```

SPARQL (with reasoning capabalities)

Querying UML Class Diagrams: Example

Executive(john)

Member(john,LU)

Stock(BAY)

Issues(BA,BAY)

Owns(john,BAY)

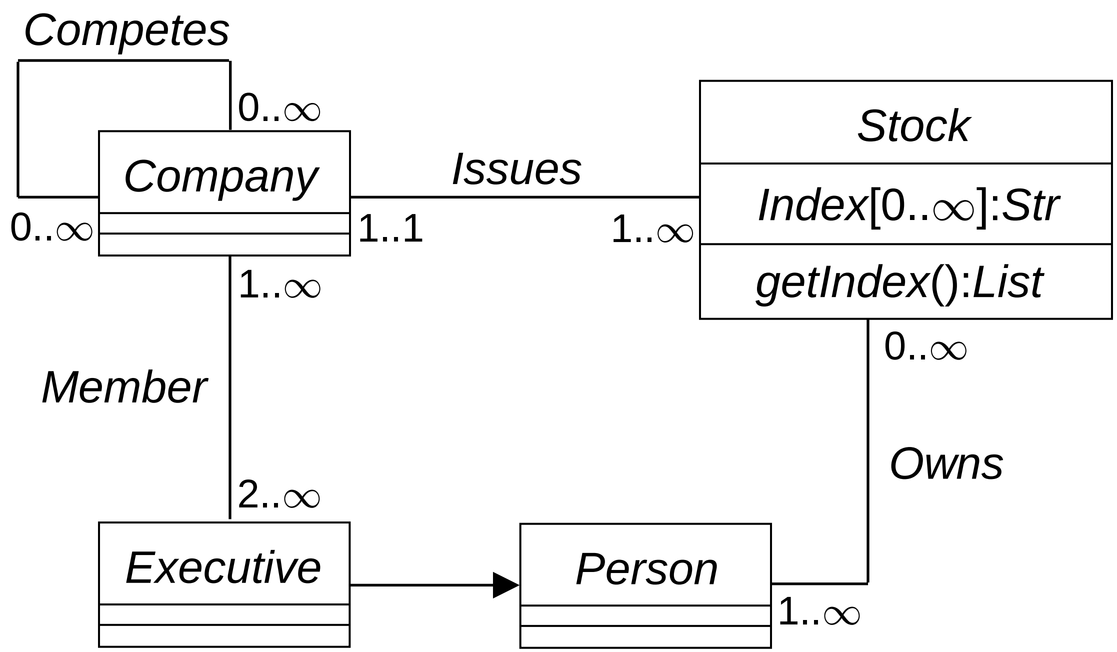
Competes(LU,BA)

***Person*(john)**

***Company*(LU)**

***Company*(BA)**

+



Does anybody have a potential conflict of interest?

$\mathit{Conflict} \leftarrow \mathit{Person}(P), \mathit{Company}(C_1), \mathit{Company}(C_2), \mathit{Stock}(S),$
$\mathit{Owns}(P,S), \mathit{Member}(P,C_1), \mathit{Issues}(C_2,S), \mathit{Competes}(C_1,C_2)$ ✓

Querying UML Class Diagrams: Example

Executive(john)

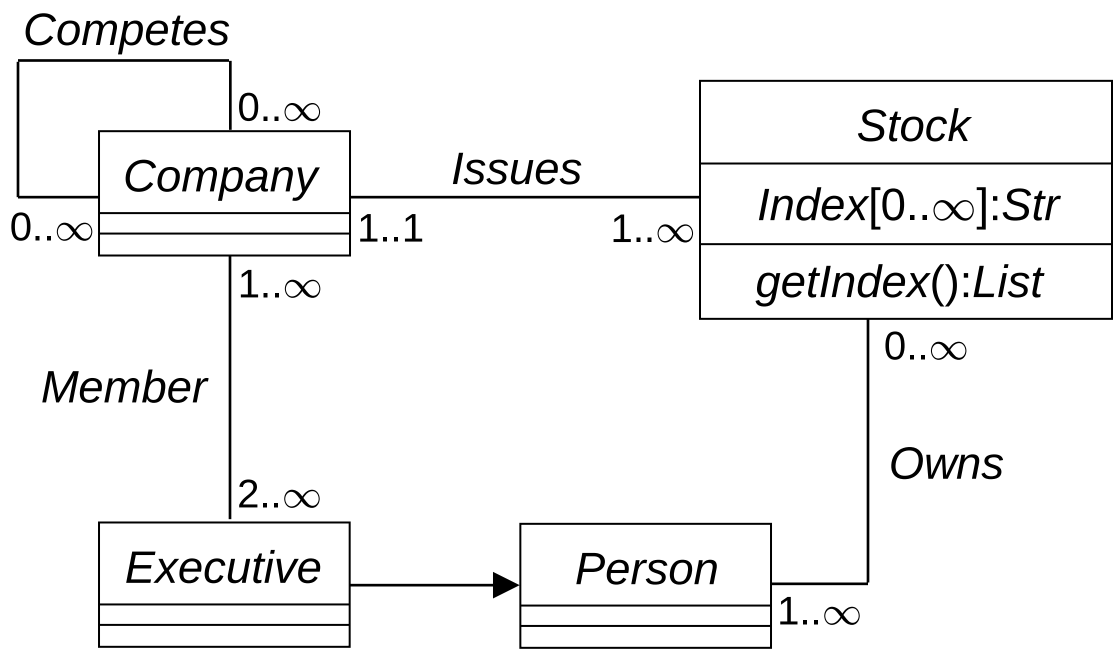
Member(john,LU)

Stock(BAY)

Issues(BA,BAY)

Owns(john,BAY)

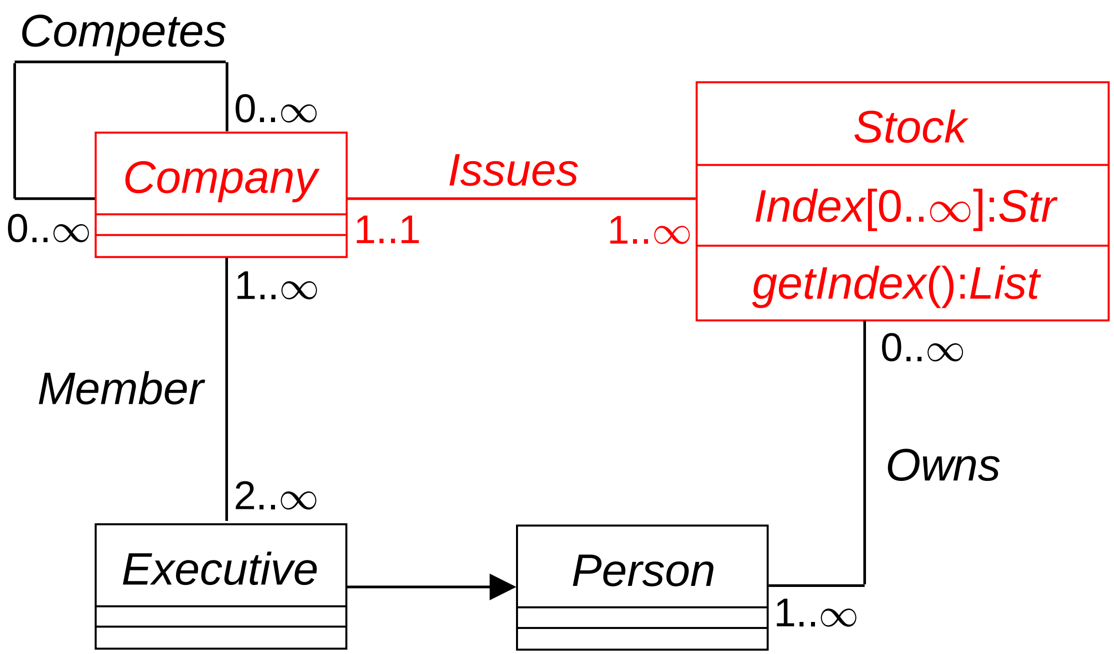
Competes(LU,BA)



Does anybody have a potential conflict of interest?

$$\mathit{Conflict} \leftarrow \mathit{Person}(P), \mathit{Company}(C_1), \mathit{Company}(C_2), \mathit{Stock}(S), \mathit{Owns}(P,S), \mathit{Member}(P,C_1), \mathit{Issues}(C_2,S), \mathit{Competes}(C_1,C_2)$$

From Diagrams to First-Order Logic (Datalog$^{\pm}$)



$\forall X\ \mathit{Company}(X) \rightarrow \exists Y\ \mathit{Issues}(X,Y)$

$\forall X\ \mathit{Stock}(X) \rightarrow \exists Y\ \mathit{Issues}(Y,X)$

$\forall X \forall Y \forall Z\ \mathit{Stock}(X) \wedge \mathit{Issues}(Y,X) \wedge \mathit{Issues}(Z,X) \rightarrow Y = Z$

$\forall X \forall Y\ \mathit{Stock}(X) \wedge \mathit{Index}(X,Y) \rightarrow \mathit{Str}(Y)$

$\forall X \forall Y\ \mathit{Stock}(X) \wedge \mathit{getIndex}(X,Y) \rightarrow \mathit{List}(Y)$

[Berardi et al., Artif. Intell. 2005]

Lean UML Class Diagrams as Guarded Rules

$\forall X \forall Y\ C(X) \wedge Attr(X,Y) \rightarrow T(Y)$

$\forall X\ C(X) \rightarrow \exists Y_1 \ldots \exists Y_n\ Attr(X,Y_1) \wedge \ldots \wedge Attr(X,Y_n)$

$\forall X\ C_1(X) \rightarrow C_2(X)$

classes

$\forall X_1 \forall X_2\ A(X_1,X_2) \rightarrow C_1(X_1) \wedge C_2(X_2)$

$\forall X_1 \forall X_2 \forall Y\ A(X_1,X_2) \wedge glue(X_1,X_2,Y) \rightarrow C_A(Y)$

$\forall X_1 \forall X_2\ A(X_1,X_2) \rightarrow \exists Y\ glue(X_1,X_2,Y)$

$\forall X\ C(X) \rightarrow \exists Y_1 \ldots \exists Y_n\ A(X,Y_1) \wedge \ldots \wedge A(X,Y_n)$

$\forall X\ C(X) \rightarrow \exists Y_1 \ldots \exists Y_n\ A(Y_1,X) \wedge \ldots \wedge A(Y_n,X)$

$\forall X_1 \forall X_2\ A_1(X_1,X_2) \rightarrow A_2(X_1,X_2)$

associations

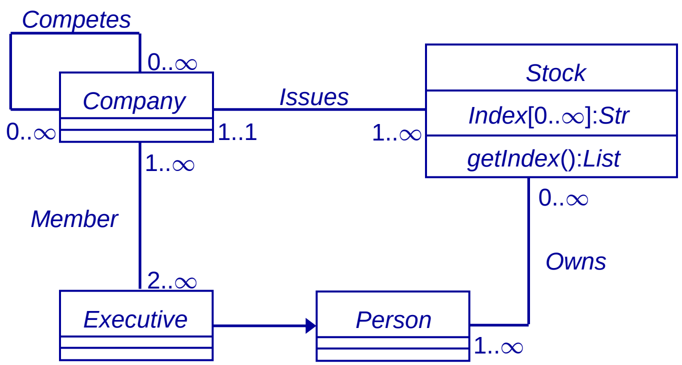
$\forall X\ C_1(X) \wedge \ldots \wedge C_n(X) \rightarrow C(X)$

$\forall X \forall Y\ C(X) \wedge Attr(Y,X) \rightarrow T(Y)$

additional constraints

[Calì, G., Orsi & Pieris, FOSSACS 2012]

A Unifying Framework for Ontology Querying



UML Class Diagrams

student::person

person[age ∗⇒ number]

person[age {0:1} ∗⇒ number]

person[name {1:∗} ∗⇒ string]

F-Logic

$$\begin{aligned} \mathit{Speaker} &\sqsubseteq \mathit{Person} \sqcap \exists \mathit{gives}.\mathit{Talk} \\ \mathit{Tutorial} &\sqsubseteq \forall \mathit{attendedBy}.\mathit{Student} \\ \mathit{attendedBy} &\sqsubseteq \mathit{attended}^{-1} \\ \mathit{Speaker} \sqcap \mathit{Student} &\sqsubseteq \bot \end{aligned}$$

Description Logics

PREFIX abc: ⟨http://example.com/exampleOntology#⟩
SELECT ?capital ?country
WHERE {
    ?x abc:cityname ?capital ;
    ?x abc:isCapitalOf ?y .
    ?x abc:countryname ?country ;
    ?x abc:isInContinent abc:Africa . }

SPARQL (with reasoning capabalities)

The DL-Lite Family

Popular family of DLs with AC_0 data complexity

DL-Lite TBox	First-Order Representation (Datalog$^{\pm}$)
DL-Lite$_{core}$	
$professor \sqsubseteq \exists teachesTo$	$\forall X\ professor(X) \rightarrow \exists Y\ teachesTo(X,Y)$
$professor \sqsubseteq \neg student$	$\forall X\ professor(X) \wedge student(X) \rightarrow \bot$
DL-Lite$_R$ (OWL 2 QL)	
$hasTutor^{-} \sqsubseteq teachesTo$	$\forall X \forall Y\ hasTutor(X,Y) \rightarrow teachesTo(Y,X)$
DL-Lite$_F$	
$\mathrm{funct}(hasTutor)$	$\forall X \forall Y \forall Z\ hasTutor(X,Y) \wedge hasTutor(X,Z) \rightarrow Y = Z$

[Calvanese, De Giacomo, Lembo, Lenzerini & Rosati, J. Autom. Reasoning 2007]

The $\mathcal{EL}$ Family

Popular family of DLs with PTIME data complexity (OWL 2 EL)

$\mathcal{ELHI}^{\neg}$ TBox	First-Order Representation (Datalog$^{\pm}$)
$A \sqcap B \sqsubseteq C$	$\forall X \; A(X) \wedge B(X) \rightarrow C(X)$
$A \sqsubseteq \exists R$	$\forall X \; A(X) \rightarrow \exists Y \; R(X,Y)$
$A \sqsubseteq \exists R.B$	$\forall X \; A(X) \rightarrow \exists Y \; R(X,Y) \wedge B(Y)$
$\exists R \sqsubseteq A$	$\forall X \forall Y \; R(X,Y) \rightarrow A(X)$
$\exists R.A \sqsubseteq B$	$\forall X \forall Y \; R(X,Y) \wedge A(X) \rightarrow B(X)$
$A \sqsubseteq \neg B$	$\forall X \; A(X) \wedge B(X) \rightarrow \bot$
$R \sqsubseteq \neg P$	$\forall X \forall Y \; R(X,Y) \wedge P(X,Y) \rightarrow \bot$

see, e.g., [Baader, IJCAI 2003], [Rosati, DL 2007] and [Pérez-Urbina et al., J. Applied Logic 2010]

DLs vs. Datalog[$\exists$,=,$\perp$]

- **Theorem:** For query answering,

$$\text{DL-Lite} \quad \leq_L \quad \text{Linear and Sticky}$$

$$\mathcal{ELHI}^{\neg} \quad \leq_L \quad \text{Guarded}$$

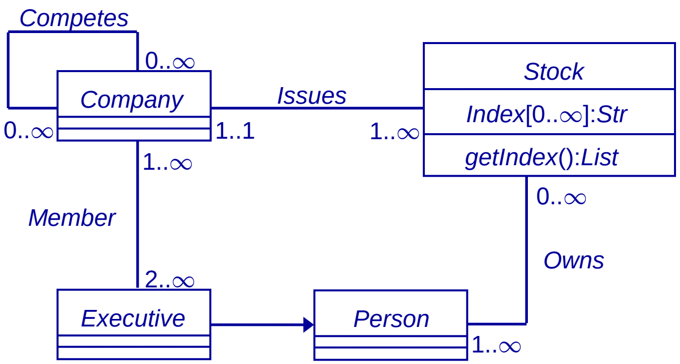
- **Theorem**: Guarded/Linear/Sticky are strictly more expressive

$$\forall E\ \mathit{ceo}(E) \rightarrow \mathit{managerOf}(E,E)$$

$$\forall E\ \mathit{supervisorOf}(E,E) \rightarrow \exists D\ \mathit{managerOf}(E,D)$$

- **Remark:** the data complexity is preserved

A Unifying Framework for Ontology Querying



UML Class Diagrams

student::person

person[age $*\Rightarrow$ number]

person[age {0:1} $*\Rightarrow$ number]

person[name {1:$*$} $*\Rightarrow$ string]

F-Logic

$$
\begin{aligned}
\mathit{Speaker} &\sqsubseteq \mathit{Person} \sqcap \exists \mathit{gives}.\mathit{Talk} \\
\mathit{Tutorial} &\sqsubseteq \forall \mathit{attendedBy}.\mathit{Student} \\
\mathit{attendedBy} &\sqsubseteq \mathit{attended}^{-1} \\
\mathit{Speaker} \sqcap \mathit{Student} &\sqsubseteq \bot
\end{aligned}
$$

Description Logics

```
PREFIX abc: ⟨http://example.com/exampleOntology#⟩
SELECT ?capital ?country
WHERE {
    ?x abc:cityname ?capital ;
    ?x abc:isCapitalOf ?y .
    ?x abc:countryname ?country ;
    ?x abc:isInContinent abc:Africa . }
```

SPARQL (with reasoning capabalities)

From F-Logic Lite to First-Order Logic (Datalog$^{\pm}$)

$type(O,A,T) \wedge data(O,A,V) \rightarrow member(V,T)$

$sub(C_1,C_3) \wedge sub(C_3,C_2) \rightarrow sub(C_1,C_2)$

$member(O,C) \wedge sub(C,C_1) \rightarrow member(O,C_1)$

$data(O,A,V) \wedge data(O,A,W), funct(A,O) \rightarrow V = W$

$mandatory(A,O) \rightarrow \exists V\, data(O,A,V)$

$member(O,C) \wedge type(C,A,T) \rightarrow type(O,A,T)$

$sub(C,C_1) \wedge type(C_1,A,T) \rightarrow type(C,A,T)$

$type(C,A,T_1) \wedge sub(T_1,T) \rightarrow type(C,A,T)$

$sub(C,C_1) \wedge mandatory(A,C_1) \rightarrow mandatory(A,C)$

$member(O,C) \wedge mandatory(A,C) \rightarrow member(A,O)$

$sub(C,C_1) \wedge funct(A,C_1) \rightarrow funct(A,C)$

$member(O,C) \wedge funct(A,C) \rightarrow funct(A,O)$

[Calì & Kifer, VLDB 2006]

From F-Logic Lite to First-Order Logic (Datalog$^{\pm}$)

$type(O,A,T) \wedge data(O,A,V) \rightarrow member(V,T)$

$sub(C_1,C_3) \wedge sub(C_3,C_2) \rightarrow sub(C_1,C_2)$

$member(O,C) \wedge sub(C,C_1) \rightarrow member(O,C_1)$

$\boxed{data(O,A,V) \wedge data(O,A,W), funct(A,O) \rightarrow V = W}$ Non-conflicting EGD

$mandatory(A,O) \rightarrow \exists V\, data(O,A,V)$

$member(O,C) \wedge type(C,A,T) \rightarrow type(O,A,T)$

$sub(C,C_1) \wedge type(C_1,A,T) \rightarrow type(C,A,T)$

$type(C,A,T_1) \wedge sub(T_1,T) \rightarrow type(C,A,T)$

$sub(C,C_1) \wedge mandatory(A,C_1) \rightarrow mandatory(A,C)$

$member(O,C) \wedge mandatory(A,C) \rightarrow member(A,O)$

$sub(C,C_1) \wedge funct(A,C_1) \rightarrow funct(A,C)$

$member(O,C) \wedge funct(A,C) \rightarrow funct(A,O)$

[Calì & Kifer, VLDB 2006]

From F-Logic Lite to First-Order Logic (Datalog$^{\pm}$)

$\boxed{type(O,A,T) \wedge data(O,A,V)} \rightarrow member(V,T)$

$\boxed{sub(C_1,C_3) \wedge sub(C_3,C_2)} \rightarrow sub(C_1,C_2)$

$member(O,C) \wedge sub(C,C_1) \rightarrow member(O,C_1)$

Non-guarded rules!

$mandatory(A,O) \rightarrow \exists V\ data(O,A,V)$

$\boxed{member(O,C) \wedge type(C,A,T)} \rightarrow type(O,A,T)$

$\boxed{sub(C,C_1) \wedge type(C_1,A,T)} \rightarrow type(C,A,T)$

$\boxed{type(C,A,T_1) \wedge sub(T_1,T)} \rightarrow type(C,A,T)$

$\boxed{sub(C,C_1) \wedge mandatory(A,C_1)} \rightarrow mandatory(A,C)$

$\boxed{member(O,C) \wedge mandatory(A,C)} \rightarrow member(A,O)$

$\boxed{sub(C,C_1) \wedge funct(A,C_1)} \rightarrow funct(A,C)$

$\boxed{member(O,C) \wedge funct(A,C)} \rightarrow funct(A,O)$

More expressive

Datalog[∃] language?

[Calì & Kifer, VLDB 2006]

Weakly-Guarded Datalog[∃]

- All ∀-variables at affected positions occur in one body atom

  unify with an ∃-position or an affected position

- Models of finite treewidth ⇒ decidability of query answering

  related to [Andréka, Németi & van Benthem, J. Philosophical Logic 1998] and [Grädel , J. Symb. Log. 1999]

- Query answering is EXPTIME-complete in data complexity – PTIME-complete if the Polynomial Cloud Criterion (PCC) holds

[Calì, G. & Kifer, KR 2008]

F-Logic Lite is Weakly-Guarded

$\mathit{type}(O,A,T) \wedge \mathit{data}(O,A,V) \rightarrow \mathit{member}(V,T)$

$\mathit{sub}(C_1,C_3) \wedge \mathit{sub}(C_3,C_2) \rightarrow \mathit{sub}(C_1,C_2)$

$\mathit{member}(O,C) \wedge \mathit{sub}(C,C_1) \rightarrow \mathit{member}(O,C_1)$

$\mathit{mandatory}(A,O) \rightarrow \exists V\ \mathit{data}(O,A,V)$

$\mathit{member}(O,C) \wedge \mathit{type}(C,A,T) \rightarrow \mathit{type}(O,A,T)$

$\mathit{sub}(C,C_1) \wedge \mathit{type}(C_1,A,T) \rightarrow \mathit{type}(C,A,T)$

$\mathit{type}(C,A,T_1) \wedge \mathit{sub}(T_1,T) \rightarrow \mathit{type}(C,A,T)$

$\mathit{sub}(C,C_1) \wedge \mathit{mandatory}(A,C_1) \rightarrow \mathit{mandatory}(A,C)$

$\mathit{member}(O,C) \wedge \mathit{mandatory}(A,C) \rightarrow \mathit{member}(A,O)$

$\mathit{sub}(C,C_1) \wedge \mathit{funct}(A,C_1) \rightarrow \mathit{funct}(A,C)$

$\mathit{member}(O,C) \wedge \mathit{funct}(A,C) \rightarrow \mathit{funct}(A,O)$

Affected Positions

data[1]

data[3]

type[1]

member[1]

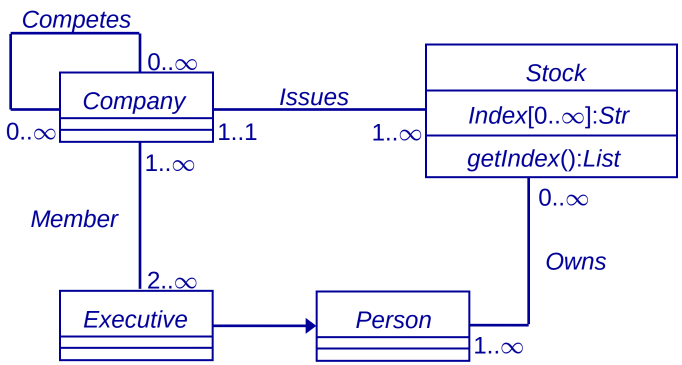
mandatory[2]

funct[2]

the PCC holds

[Calì & Kifer, VLDB 2006]

A Unifying Framework for Ontology Querying



UML Class Diagrams

student::person

person[age ∗⇒ number]

person[age {0:1} ∗⇒ number]

person[name {1:∗} ∗⇒ string]

F-Logic

$$
\begin{aligned}
Speaker &\sqsubseteq Person \sqcap \exists gives.Talk \\
Tutorial &\sqsubseteq \forall attendedBy.Student \\
attendedBy &\sqsubseteq attended^{-1} \\
Speaker \sqcap Student &\sqsubseteq \bot
\end{aligned}
$$

Description Logics

PREFIX abc: ⟨http://example.com/exampleOntology#⟩

SELECT ?capital ?country

WHERE {

?x abc:cityname ?capital ;

?x abc:isCapitalOf ?y .

?x abc:countryname ?country ;

?x abc:isInContinent abc:Africa . }

SPARQL (with reasoning capabalities)

SPARQL Protocol and RDF Query Language

- RDF – data model for representing information in the Web

- … in fact, is a finite set of triples (*subject*, *predicate*, *object*) – or a relational database for the schema {*triple*(.,.,.)}

- SPARQL – the standard language for querying RDF data

Some SPARQL Queries

- P = (?X, name, ?Y) – list of pairs (o_1,o_2) such as o_2 is the name of o_1
- P = (?X, name, B) – list of elements that have a name
- P = (?X, name, ?Y) OPT (?X, phone, ?Y) – for every subject *o*, return *o*, the name of *o*, and the phone number of *o*, if the phone number is available; otherwise, return *o* and its name

From SPARQL to First-Order Logic (Datalog$^{\pm}$)

P = (?X, name, ?Y) – list of pairs (o_1,o_2) such as o_2 is the name of o_1

$$\forall X \forall Y \; \mathit{triple}(X,\mathrm{name},Y) \;\rightarrow\; P(X,Y)$$

From SPARQL to First-Order Logic (Datalog$^{\pm}$)

P = (?X, name, B) – list of elements that have a name

$$\forall X \forall Y \;\; \mathit{triple}(X,\mathrm{name},Y) \rightarrow P(X)$$

From SPARQL to First-Order Logic (Datalog$^{\pm}$)

P = (?X, name, ?Y) OPT (?X, phone, ?Y) – for every subject *o*, return *o*, the name of *o*, and the phone number of *o*, if the phone number is available; otherwise, return *o* and its name

list of individuals with phone number

$\forall X \forall Y\ \mathit{triple}(X,\mathrm{name},Y) \wedge \mathit{triple}(X,\mathrm{phone},Z) \rightarrow P(X,Y,Z) \wedge \mathit{compatible}(Z)$

the third argument (i.e., the phone no.) is missing

$\forall X \forall Y\ \mathit{triple}(X,\mathrm{name},Y) \wedge \neg\mathit{compatible}(X) \rightarrow P'(X,Y)$

From SPARQL to First-Order Logic ($\text{Datalog}^{\pm}$)

P = (?X, name, ?Y) OPT (?X, phone, ?Y) – for every subject *o*, return *o*, the name of *o*, and the phone number of *o*, if the phone number is available; otherwise, return *o* and its name

$$\forall X \forall Y\ \boxed{\mathit{triple}(X,\mathrm{name},Y) \wedge \mathit{triple}(X,\mathrm{phone},Z)} \rightarrow P(X,Y,Z) \wedge \mathit{compatible}(Z)$$

$$\forall X \forall Y\ \mathit{triple}(X,\mathrm{name},Y) \wedge \boxed{\neg \mathit{compatible}(X)} \rightarrow P'(X,Y)$$

Non-guarded rules, and also negation is needed

From SPARQL to First-Order Logic (Datalog$^{\pm}$)

P = (?X, name, ?Y) OPT (?X, phone, ?Y) – for every subject *o*, return *o*, the name of *o*, and the phone number of *o*, if the phone number is available; otherwise, return *o* and its name

$$\forall X \forall Y\ \boxed{triple(X,name,Y) \wedge triple(X,phone,Z)} \rightarrow P(X,Y,Z) \wedge compatible(Z)$$

$$\forall X \forall Y\ triple(X,name,Y) \wedge \boxed{\neg compatible(X)} \rightarrow P'(X,Y)$$

Non-guarded rules, and also negation is needed

Stratified Weakly-Guarded Datalog[∃,¬,⊥]

Additional Functionalities

- Reasoning capabilities – deal with RDFS and OWL vocabularies
- Navigational capabilities – exploit the graph structure of RDF data
- General form of recursion – express natural queries

Theorem: Stratified weakly-guarded Datalog[$\exists,\neg,\perp$] is strictly more expressive than SPARQL enriched with the above functionalities (under the the OWL 2 QL and OWL 2 RL profiles of OWL 2)

Overview

Formalism	**Datalog$^{\pm}$ Language**
Lean UML Class Diagrams	Guarded Datalog[$\exists$,=,$\perp$]
DL-Lite (OWL 2 QL)	Linear and Sticky Datalog[$\exists$,=,$\perp$]
$\mathcal{ELHI}^{\neg}$ (OWL 2 EL)	Guarded Datalog[$\exists$,$\perp$]
F-Logic Lite	Weakly-Guarded Datalog[$\exists$,=] + PCC
SPARQL (with additional features)	Stratified Weakly-Guarded Datalog[$\exists$,$\neg$,$\perp$]

Implementation

- **IRIS**$^{\pm}$ – Query rewriting engine for Linear and Sticky Datalog[∃]
    - Developed at the University of Oxford
    - Several effective optimizations towards compact rewritings
    - Supports parallel rewriting
    - Database execution via SQL rewriting
    - Fully open source – https://bitbucket.org/giorsi/nyaya

Other Engines for Datalog$^{\pm}$

- ALASKA – Query rewriting engine for Linear and Sticky Datalog[∃]
  - Developed at the University of Montpellier

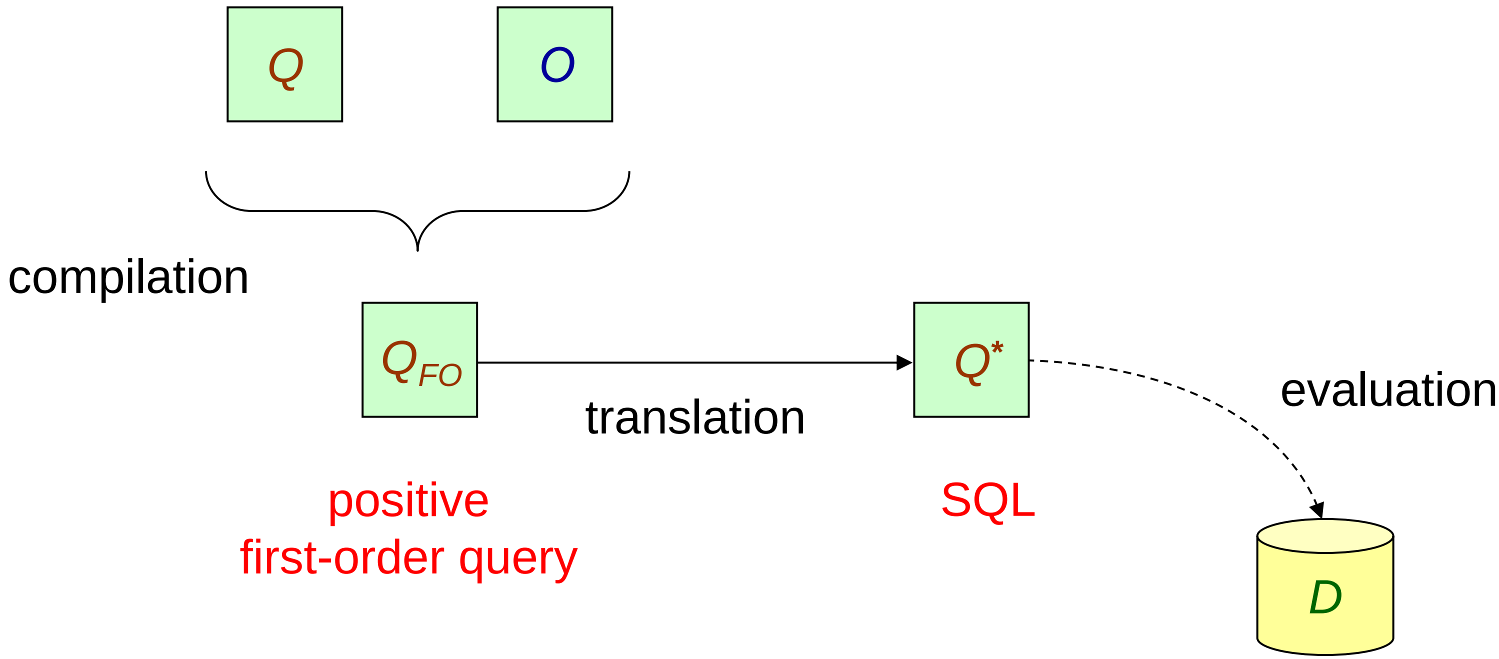
a language which captures Linear Datalog[∃] and plain Datalog

- DLV$^{\exists}$ – Query answering engine for Shy Datalog[∃]
  - Developed at the University of Calabria
  - Extension of the DLV engine

Thank you!

APPENDIX

First-Order Rewritability



$$\forall D \; (D \cup O \models Q \quad \Leftrightarrow \quad D \models Q^{*})$$

Query answering is in AC_0 in data complexity

The DL-Lite Family

Popular family of DLs with AC_0 data complexity

DL-Lite$_F$

(funct R)

DL-Lite$_R$ (OWL 2 QL)

$R \sqsubseteq E$

DL-Lite$_{core}$

$B \sqsubseteq C$

B	::=	$A \mid \exists R \mid \exists R^{-1}$	(basic concept)
C	::=	$B \mid \neg B$	(general concept)
R	::=	$P \mid P^{-1}$	(basic role)
E	::=	$R \mid \neg R$	(general role)

[Calvanese, De Giacomo, Lembo, Lenzerini & Rosati, J. Autom. Reasoning 2007]

The $\mathcal{EL}$ Family

Popular family of DLs with PTIME data complexity (OWL 2 EL)

$\mathcal{ELHI}^{\neg}$	$R \sqsubseteq E$	$E ::= R \mid \neg R$
$\mathcal{ELHI}$	$R \sqsubseteq P$	$R ::= P \mid P^{-1}$
$\mathcal{ELH}$	$P_1 \sqsubseteq P_2$	
$\mathcal{EL}$	$C_1 \sqsubseteq C_2$	$C ::= \top \mid A \mid C_1 \sqcap C_2 \mid \exists P.C$

see, e.g., [Baader, IJCAI 2003], [Rosati, DL 2007] and [Pérez-Urbina et al., J. Applied Logic 2010]

Weakly-Guarded Datalog[∃]

- All ∀-variables at affected positions occur in one body atom

null can occur
during the chase

$$\forall X \forall Y \;\; S(X,Y) \wedge P(X,Y) \rightarrow \exists Z \; P(Y,Z)$$

$$\forall X \forall Y \forall W \;\; P(X,Y) \wedge P(W,X) \rightarrow S(Y,X)$$

Affected positions = ?

[Calì, G. & Kifer, KR 2008]

Weakly-Guarded Datalog[∃]

- All ∀-variables at affected positions occur in one body atom

null can occur
during the chase

$$\forall X \forall Y \;\; S(X,Y) \wedge P(X,Y) \rightarrow \exists Z \; P(Y,Z)$$

$$\forall X \forall Y \forall W \;\; P(X,Y) \wedge P(W,X) \rightarrow S(Y,X)$$

Affected positions = {*P*[2]}

[Calì, G. & Kifer, KR 2008]

Weakly-Guarded Datalog[∃]

- All ∀-variables at affected positions occur in one body atom

null can occur
during the chase

$$\forall X \forall Y \; S(X,Y) \wedge P(X,Y) \rightarrow \exists Z \; P(Y,Z)$$

$$\forall X \forall Y \forall W \; P(X,Y) \wedge P(W,X) \rightarrow S(Y,X)$$

Affected positions = {*P*[2], *S*[1]}

[Calì, G. & Kifer, KR 2008]

Weakly-Guarded Datalog[∃]

- All ∀-variables at affected positions occur in one body atom

null can occur
during the chase

$$\forall X \forall Y \; \boldsymbol{S(X,Y)} \wedge P(X,Y) \rightarrow \exists Z\, P(Y,Z)$$

$$\forall X \forall Y \forall W \; \boldsymbol{P(X,Y)} \wedge P(W,X) \rightarrow S(Y,X)$$

Affected positions = {*P*[2], *S*[1]}

[Calì, G. & Kifer, KR 2008]

Frame Logic (F-Logic)

- Originally developed for object-oriented deductive databases
- Now used as an ontology language – Semantic Web
- F-Logic is in general undecidable

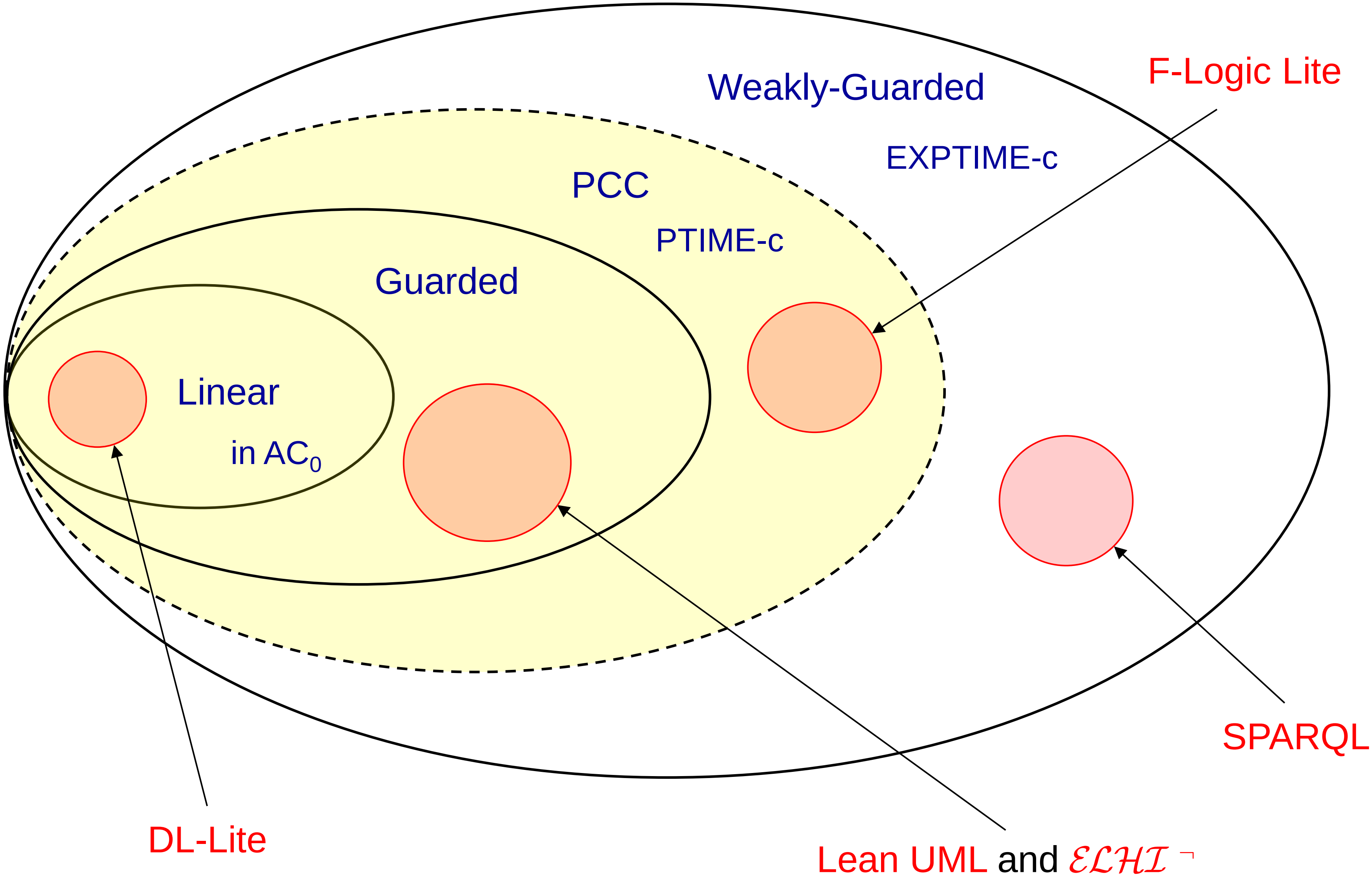
see, e.g., [Kifer, Lausen & Wu, J. ACM 1995]

F-Logic Lite

- An expressive decidable fragment of F-Logic
- Obtained from F-Logic by:
  - Excluding negation and default inheritance
  - Allowing only a limited form of cardinality constraints
- Query answering is PTIME-complete in data complexity

[Calì & Kifer, VLDB 2006]

Overview


Weakly-Guarded
EXPTIME-c
F-Logic Lite
PCC
PTIME-c
Guarded
Linear
in AC_0
SPARQL
DL-Lite
Lean UML and $\mathcal{ELHI}^{\neg}$

Complexity of Datalog[∃,=,¬,⊥]

	Linear	Guarded	Weakly-Guarded	
Arbitrary Query	PSPACE-c	2EXPTIME-c	2EXPTIME-c	Arbitrary Program
Fixed/Atomic Query	PSPACE-c	2EXPTIME-c	2EXPTIME-c	
Arbitrary Query	NP-c	NP-c	EXPTIME-c	Fixed Program
Fixed/Atomic Query	in AC_0	PTIME-c	EXPTIME-c	

Complexity of Datalog[$\exists$,=,$\neg$,$\perp$]

		Linear	Guarded	Weakly-Guarded	
Arbitrary Program	Arbitrary Query	PSPACE-c	2EXPTIME-c	2EXPTIME-c	Arbitrary Program
Arbitrary Program	Fixed/Atomic Query	PSPACE-c	2EXPTIME-c	2EXPTIME-c	Arbitrary Program
Fixed Program	Arbitrary Query	NP-c	NP-c	EXPTIME-c	Fixed Program
Fixed Program	Fixed/Atomic Query	in AC_0	PTIME-c	EXPTIME-c	Fixed Program

Data Complexity

Complexity of Datalog[∃,=,¬,⊥]

	Linear	Guarded	Weakly-Guarded	
Arbitrary Query	PSPACE-c	2EXPTIME-c	2EXPTIME-c	Arbitrary Program
Fixed/Atomic Query	PSPACE-c	2EXPTIME-c	2EXPTIME-c	
Arbitrary Query	NP-c	NP-c	NP-c	Fixed Program
Fixed/Atomic Query	in AC_0	PTIME-c	PTIME-c	

Polynomial Cloud Criterion