Description Logics

Description Logics and Databases

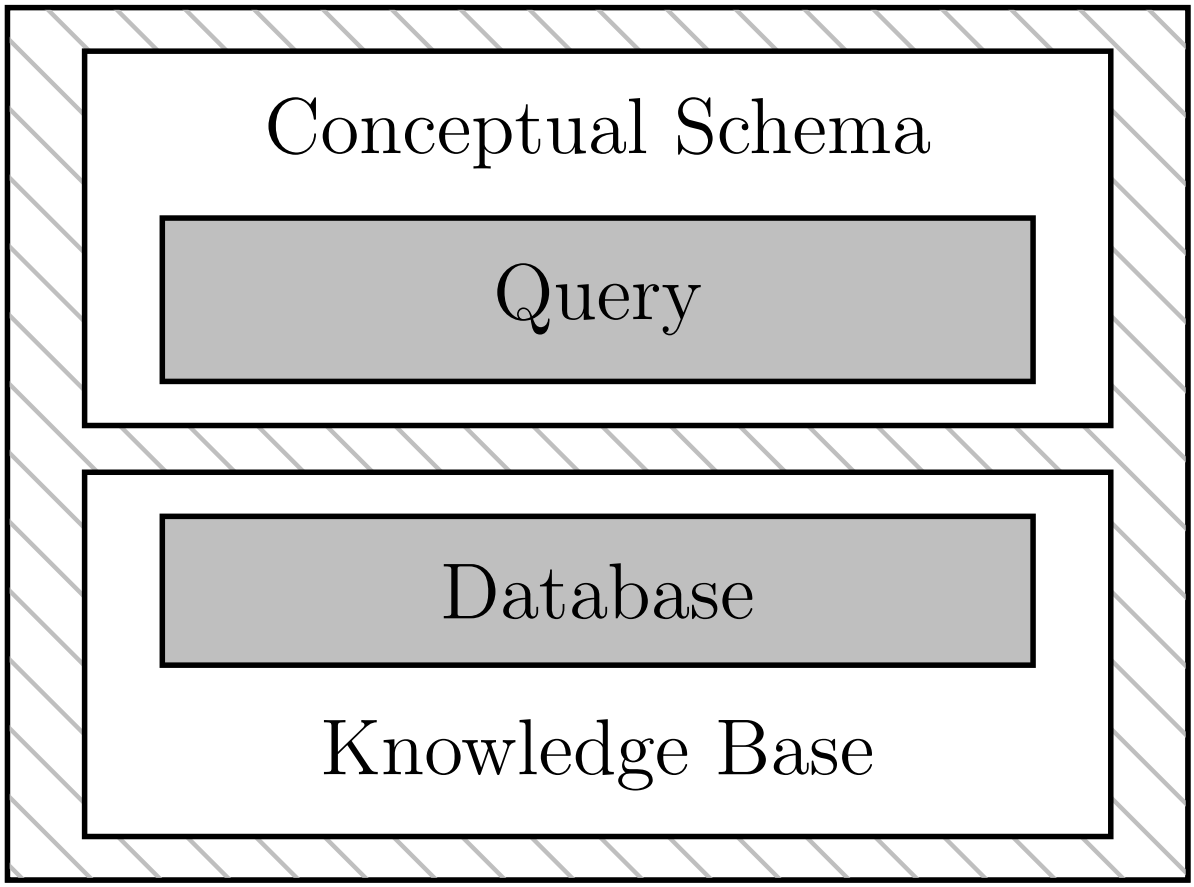
Enrico Franconi

Department of Computer Science
University of Manchester

http://www.cs.man.ac.uk/~franconi

Description Logics and Databases

- Queries
- Conceptual Modeling
- Finite Model Reasoning



Relational Algebra

SINGER		
NAME	COUNTRY	TYPE
Pavarotti	Italy	Classic
Eagles	U.S.A.	Pop
Ramazzotti	Italy	Pop
Queen	U.K.	Rock

CONCERT			
ARTIST	PLACE	DATE	TICKET
Eagles	Paris	22/6/1998	YES
Pavarotti	Barcelona	28/6/1997	NO
Pavarotti	Bologna	27/4/1998	YES

Tell me the places and the dates of italian artists concerts.

$$\pi_{\{\text{PLACE,DATE}\}} \Big(\sigma_{\text{COUNTRY}=\texttt{Italy}} \big(\mathit{SINGER} \bowtie_{\text{NAME}=\text{ARTIST}} \mathit{CONCERT} \big) \Big)$$

RESULT	
PLACE	DATE
Barcelona	28/6/1997
Bologna	27/4/1998

Relational Calculus

SINGER		
NAME	COUNTRY	TYPE
Pavarotti	Italy	Classic
Eagles	U.S.A.	Pop
Ramazzotti	Italy	Pop
Queen	U.K.	Rock

CONCERT			
ARTIST	PLACE	DATE	TICKET
Eagles	Paris	22/6/1998	YES
Pavarotti	Barcelona	28/6/1997	NO
Pavarotti	Bologna	27/4/1998	YES

Tell me the places and the dates of italian artists concerts.

$$\exists z, k, w. \quad CONCERT(z, \mathbf{x}, \mathbf{y}, k) \wedge SINGER(z, \texttt{Italy}, w)$$

RESULT	
PLACE	DATE
Barcelona	28/6/1997
Bologna	27/4/1998

Relational Theory

FACT: $SINGER\,(\texttt{Pavarotti}, \texttt{Italy}, \texttt{Classic})$.

$\vdots$

$CONCERT\,(\texttt{Pavarotti}, \texttt{Bologna}, 27/4/1998, \texttt{YES})$.

UNA: $\texttt{Eagles} \neq \texttt{Queen}, \ldots, \texttt{Paris} \neq \texttt{Barcelona}$.

DO: $\forall x.\ ((x = \texttt{Pavarotti}) \vee \cdots \vee (x = \texttt{Eagles}))$.

CO: $\forall x_1 \cdots x_3.\ (SINGER\,(x_1, \ldots, x_3) \longrightarrow$
$(x_1 = \texttt{Pavarotti} \wedge \cdots \wedge x_3 = \texttt{Classic}) \vee$
$\cdots \vee$
$(x_1 = \texttt{Queen} \wedge \cdots \wedge x_3 = \texttt{Rock}))$.

$\forall x_1 \cdots x_4.\ (CONCERT\,(x_1, \ldots, x_4) \longrightarrow$
$(x_1 = \texttt{Eagles} \wedge \cdots \wedge x_4 = \texttt{YES}) \vee \cdots \vee$
$(x_1 = \texttt{Pavarotti} \wedge \cdots \wedge x_4 = \texttt{YES}))$.

$\exists z, k, w.\quad CONCERT(z, \mathbf{x}, \mathbf{y}, k) \wedge$
$SINGER(z, \texttt{Italy}, w)$

Autoepistemic Logic

FACT: $SINGER\,(\texttt{Pavarotti}, \texttt{Italy}, \texttt{Classic})$.

$\vdots$

$CONCERT\,(\texttt{Pavarotti}, \texttt{Bologna}, 27/4/1998, \texttt{YES})$.

UNA: $\texttt{Eagles} \neq \texttt{Queen}, \ldots, \texttt{Paris} \neq \texttt{Barcelona}$.

$$\exists z, k, w. \quad \mathbf{K}CONCERT(z, \mathbf{x}, \mathbf{y}, k) \wedge \mathbf{K}SINGER(z, \texttt{Italy}, w)$$

DL as a query language for DB

Description Logics parallel the four approaches:

- The $\ddot{\mathcal{C}}^3$ fragment can be easily translated into relational algebra.
- Model checking techniques can be generally applied, up to a CO-NP-complete complexity for querying with the full featured description logic.
- The description logic can express the uniqueness of a model for a relational theory.
- Decidable autoepistemic extensions of description logics exist.

DL and Relational Algebra: the $\mathcal{ALC}$ example

$$
\begin{array}{rcl}
\overline{\top}^{\mathcal{I}} & = & \mathcal{J}^{\mathcal{I}} \\
\overline{\bot}^{\mathcal{I}} & = & \emptyset \\
\overline{A}^{\mathcal{I}} & = & (\mathbf{K}A)^{\mathcal{I}} \\
\overline{\neg C}^{\mathcal{I}} & = & \mathcal{J}^{\mathcal{I}} \setminus \overline{C}^{\mathcal{I}} \\
\overline{C \sqcap D}^{\mathcal{I}} & = & \overline{C}^{\mathcal{I}} \cap \overline{D}^{\mathcal{I}} \\
\overline{C \sqcup D}^{\mathcal{I}} & = & \overline{C}^{\mathcal{I}} \cup \overline{D}^{\mathcal{I}} \\
\overline{\forall R.\, C}^{\mathcal{I}} & = & \mathcal{J}^{\mathcal{I}} \setminus \pi_1(\overline{R}^{\mathcal{I}} \underset{2=1}{\bowtie} (\mathcal{J}^{\mathcal{I}} \setminus \overline{C}^{\mathcal{I}})) \\
\overline{\exists R.\, C}^{\mathcal{I}} & = & \pi_1(\overline{R}^{\mathcal{I}} \underset{2=1}{\bowtie} \overline{C}^{\mathcal{I}})
\end{array}
$$

$$
\overline{R}^{\mathcal{I}} = (\mathbf{K}R)^{\mathcal{I}}
$$

Example of query

Given the theory (ABox):

`CHILD(john,mary), Female(mary).`

Which are the individuals in the extension of the query:

$\forall$`CHILD.Female`

- with classical DL semantics
- by considering the ABox as a database and using the relational algebra equivalent for the query:
  $\mathcal{J} \setminus \pi_1(\texttt{CHILD} \underset{2=1}{\bowtie} (\mathcal{J} \setminus \texttt{Female}))$

Relational theories in DL

- **FACT** is an ABox knowledge base.
- **UNA** is built-in in description logics.
- **DO** can be expressed by means of an axiom of the type

$$\top \dot{\sqsubseteq} \{\ a\ \} \sqcup \{\ b\ \} \sqcup \ldots$$

  for all individuals in the database.

- **CO** can be expressed by means of axioms of the type

$$A_i \doteq \{\ a_1^i, a_2^i, \ldots\ \}$$

  $R_j \doteq (\{\ a_1^j\ \} \times \{\ b_1^j\ \}) \sqcup (\{\ a_2^j\ \} \times \{\ b_2^j\ \}) \sqcup \ldots$

  (if the $(C \times D)$ operator is lacking, a more careful encoding is necessary)

- Reasoning is now on the unique identified model.

Exercise

The encoding of the role expression $R \doteq (C \times D)$, where
$(C \times D)^{\mathcal{I}} = \{(i,j) \in \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \mid C^{\mathcal{I}}(i) \wedge D^{\mathcal{I}}(j)\}$

$C \sqsubseteq \exists R$

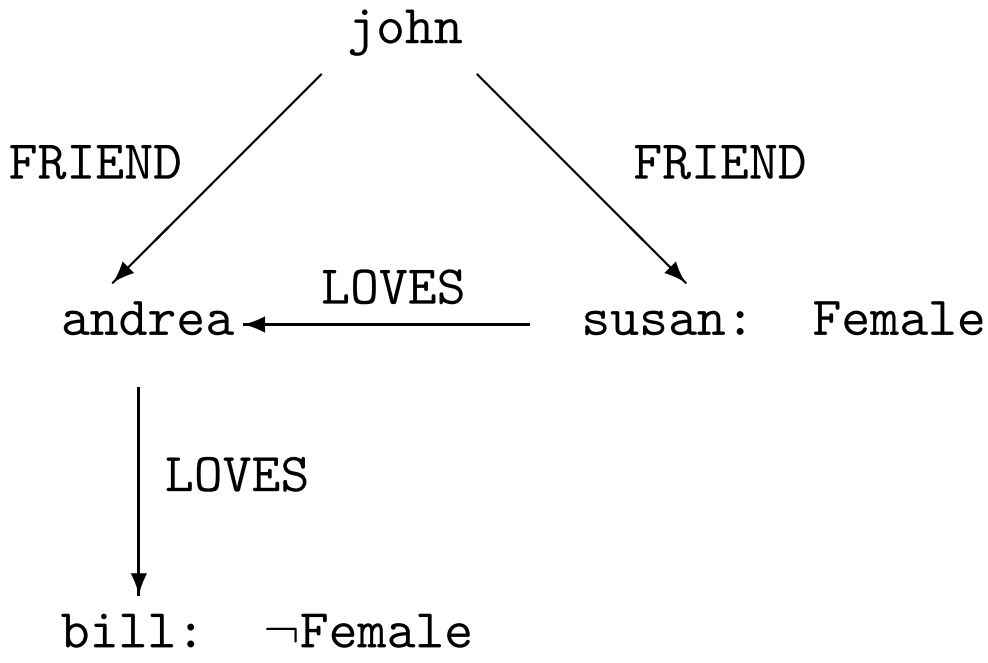
$D \sqsubseteq \exists R^{-1}$

$\top \sqsubseteq \forall R.\, D$

$\top \sqsubseteq \forall R^{-1}.\, C$

Our old example

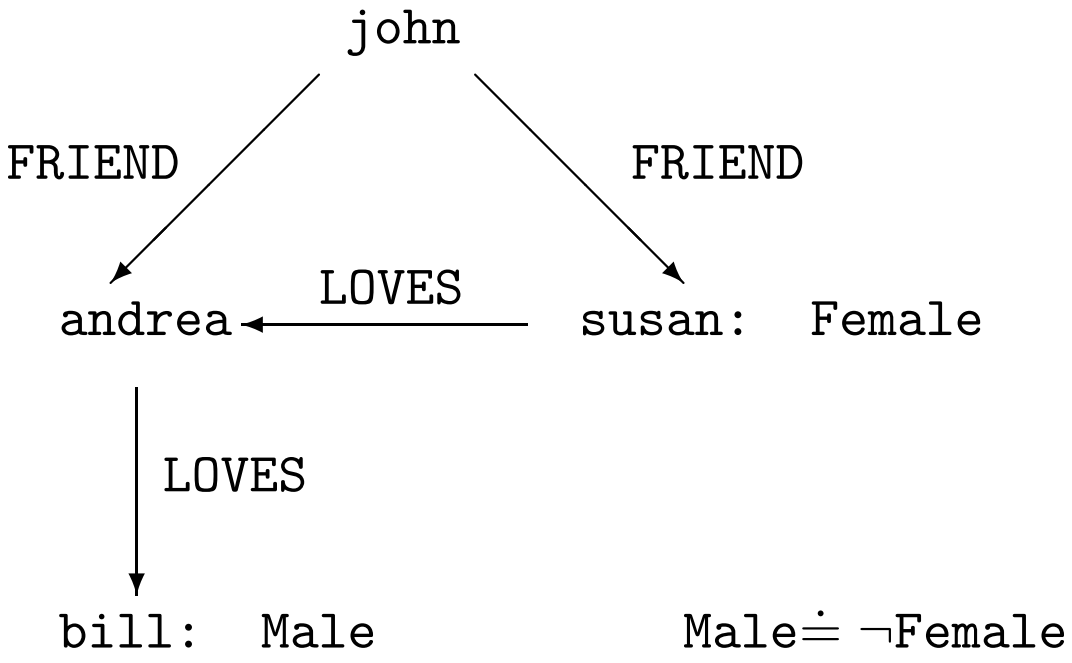
$\Gamma =$



Does John have a female friend loving a not female person?

$$\Gamma \models \exists X, Y.\ \texttt{FRIEND}(\texttt{john}, X) \wedge \texttt{Female}(X) \wedge \texttt{LOVES}(X, Y) \wedge \neg\texttt{Female}(Y)$$

$$\Gamma \models \Big(\exists\texttt{FRIEND}.\big(\texttt{Female} \sqcap (\exists\texttt{LOVES}.\,\neg\texttt{Female})\big)\Big)(\texttt{john})$$

$\Gamma_1 =$ 

Does John have a female friend loving a male person?

$$\Gamma_1 \models \exists X, Y.\ \texttt{FRIEND}(\texttt{john}, X) \wedge \texttt{Female}(X) \wedge \texttt{LOVES}(X, Y) \wedge \texttt{Male}(Y)$$

$$\Gamma_1 \models \Big(\exists \texttt{FRIEND}.\big(\texttt{Female} \sqcap (\exists \texttt{LOVES}.\texttt{Male})\big)\Big)(\texttt{john})$$

$$\Gamma \not\models \texttt{Female}(\texttt{andrea})$$
$$\Gamma \not\models \neg\texttt{Female}(\texttt{andrea})$$

$$\Gamma_1 \not\models \texttt{Female}(\texttt{andrea})$$
$$\Gamma_1 \not\models \neg\texttt{Female}(\texttt{andrea})$$
$$\Gamma_1 \not\models \texttt{Male}(\texttt{andrea})$$
$$\Gamma_1 \not\models \neg\texttt{Male}(\texttt{andrea})$$

Γ as a logic program (DATALOG$^\neg$)

EDB:
```
friend(john,susan).
friend(john,andrea).
loves(susan,andrea).
loves(andrea,bill).
female(susan).
```

Querying Γ:

```
?- friend(john,X), female(X),
   loves(X,Y), ¬female(Y).

X = susan, Y = andrea
yes

?- ¬female(andrea).

yes

?- female(andrea).

no
```

$$\begin{aligned}\Gamma = \;& \texttt{FRIEND(john,susan)} \wedge \\ & \texttt{FRIEND(john,andrea)} \wedge \\ & \texttt{LOVES(susan,andrea)} \wedge \\ & \texttt{LOVES(andrea,bill)} \wedge \\ & \texttt{Female(susan)} \wedge \\ & \neg\texttt{Female(bill)}\end{aligned}$$

$\Delta^{\mathcal{I}} = \{\texttt{john}, \texttt{susan}, \texttt{andrea}, \texttt{bill}\}$ $\Longleftarrow$ unique minimal model.
$\texttt{Female}^{\mathcal{I}} = \{\texttt{susan}\}$

$$\begin{aligned}\Gamma_1 = \;& \texttt{FRIEND(john, susan)} \wedge \\ & \texttt{FRIEND(john, andrea)} \wedge \\ & \texttt{LOVES(susan, andrea)} \wedge \\ & \texttt{LOVES(andrea, bill)} \wedge \\ & \texttt{Female(susan)} \wedge \\ & \texttt{Male(bill)} \wedge \\ & \forall X.\ \texttt{Male}(X) \leftrightarrow \neg\texttt{Female}(X)\end{aligned}$$

$\Delta^{\mathcal{I}1} = \{\texttt{john}, \texttt{susan}, \texttt{andrea}, \texttt{bill}\}$
$\texttt{Female}^{\mathcal{I}1} = \{\texttt{susan}, \texttt{andrea}\}$
$\texttt{Male}^{\mathcal{I}1} = \{\texttt{bill}, \texttt{john}\}$

$\Delta^{\mathcal{I}2} = \{\texttt{john}, \texttt{susan}, \texttt{andrea}, \texttt{bill}\}$
$\texttt{Female}^{\mathcal{I}2} = \{\texttt{susan}\}$
$\texttt{Male}^{\mathcal{I}2} = \{\texttt{bill}, \texttt{andrea}, \texttt{john}\}$

$\Delta^{\mathcal{I}1} = \{\texttt{john}, \texttt{susan}, \texttt{andrea}, \texttt{bill}\}$
$\texttt{Female}^{\mathcal{I}1} = \{\texttt{susan}, \texttt{andrea}, \texttt{john}\}$
$\texttt{Male}^{\mathcal{I}1} = \{\texttt{bill}\}$

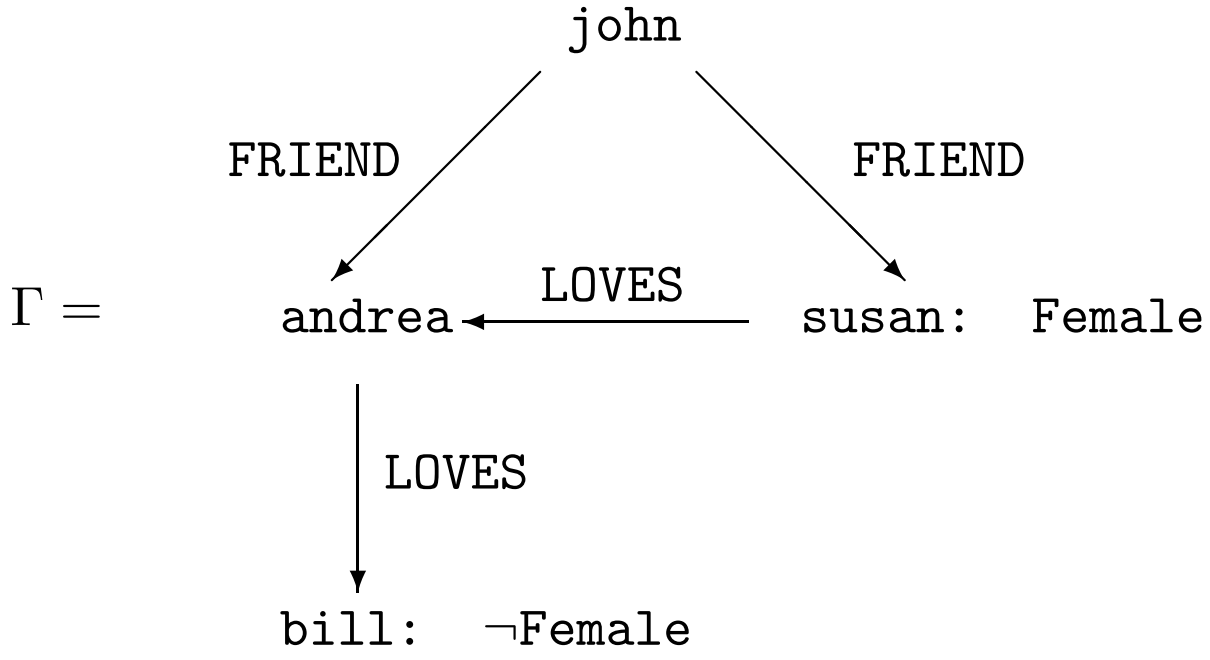
$\Delta^{\mathcal{I}2} = \{\texttt{john}, \texttt{susan}, \texttt{andrea}, \texttt{bill}\}$
$\texttt{Female}^{\mathcal{I}2} = \{\texttt{susan}, \texttt{john}\}$
$\texttt{Male}^{\mathcal{I}2} = \{\texttt{bill}, \texttt{andrea}\}$

Four models; does not exist a unique minimal model.

$\mathcal{ALCK}$

$$C \rightarrow \ldots \mid \mathbf{K}C$$
$$R \rightarrow \ldots \mid \mathbf{K}R$$

- $\mathbf{K}C$ denotes the set of individuals which are *known* to be in the extension of the concept C, in *every* model of the knowledge base.
- Reasoning in $\mathcal{ALCK}$ is PSPACE-complete.
- The evaluation of the *database-oriented* $\mathcal{ALCK}$ queries is polynomial.
- If we limit the expressivity of the DL to ensure the existence of a unique minimal model, then the evaluation of *database-oriented* queries is formally equivalent to CWA.



$\Gamma \models \Big(\exists \texttt{FRIEND}.\big(\texttt{Female} \sqcap (\exists \texttt{LOVES}.\neg\texttt{Female})\big)\Big)(\texttt{john})$

$\Gamma \not\models \texttt{Female}(\texttt{andrea})$

$\Gamma \not\models \neg\texttt{Female}(\texttt{andrea})$

DB-oriented queries:

$\Gamma \models \Big(\exists \mathbf{K}\texttt{FRIEND}.\big(\mathbf{K}\texttt{Female} \sqcap (\exists \mathbf{K}\texttt{LOVES}.\neg\mathbf{K}\texttt{Female})\big)\Big)(\texttt{john})$

$\Gamma \not\models \mathbf{K}\texttt{Female}(\texttt{andrea})$

$\Gamma \models \neg\mathbf{K}\texttt{Female}(\texttt{andrea})$

$\Gamma \not\models \Big(\exists \mathbf{K}\texttt{FRIEND}.\big(\mathbf{K}\texttt{Female} \sqcap (\exists \mathbf{K}\texttt{LOVES}.\mathbf{K}\neg\texttt{Female})\big)\Big)(\texttt{john})$

$\Gamma \not\models \Big(\exists \mathbf{K}\texttt{FRIEND}.\mathbf{K}\big(\texttt{Female} \sqcap (\exists \texttt{LOVES}.\neg\texttt{Female})\big)\Big)(\texttt{john})$

Exercise

Description Logics with the autoepistemic operator can query also the theory Γ_1 – which is completely equivalent to Γ – while Γ_1 can not even be represented in database or logic programming frameworks.

- Try some autoepistemic query to the knowledge base Γ_1.
- Check the CWA with Γ and Γ_1: which is the difference between the two extensions, and which is the difference with the autoepistemic approach?

So what?

Why should we bother of using DL for querying databases, when there are much more expressive languages for that purpose – basically without the "three variables" limit?

- Reasoning over the query is decidable:
  - Query containment,
  - Query satisfiability.
- Evaluation still tractable.
- Two natural extensions:
  - Incomplete information,
  - Querying with a conceptual schema.

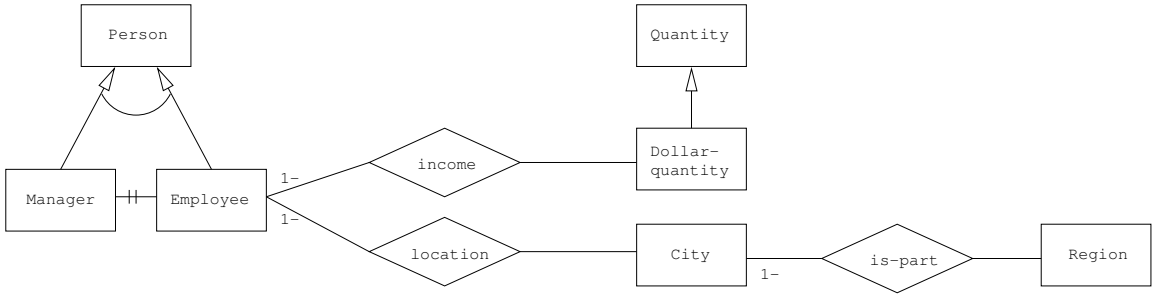
Incomplete Information

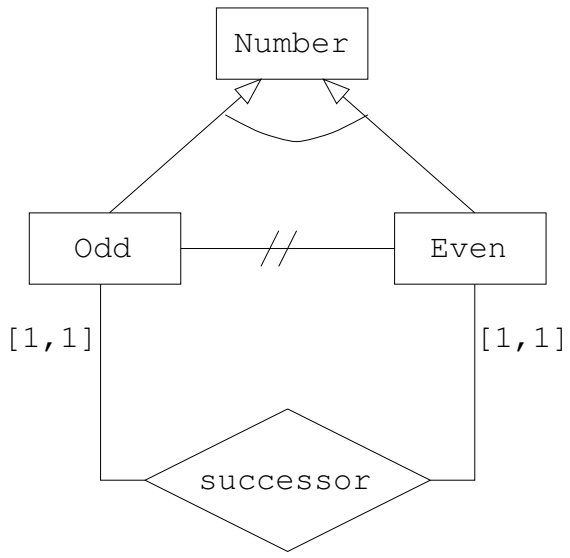
Handling incomplete knowledge is the ability to correctly reason without a complete specification of a situation, but with a under-specified description of a class of possible situations.

- FOL theories.
- KR theories.
  - Unique Name (UNA) assumption.
- Finite Domain theories.
  - Domain Closure (DO) assumption.
- Closed theories (e.g., null values).
  - DO + Completion (CO) assumptions.
- Extended Relational theories.
  - UNA + DO + CO assumptions.

The Entity-Relationship (ER) Conceptual Data Model

The Entity-Relationship (ER) model is the most common semantic data model for database design.

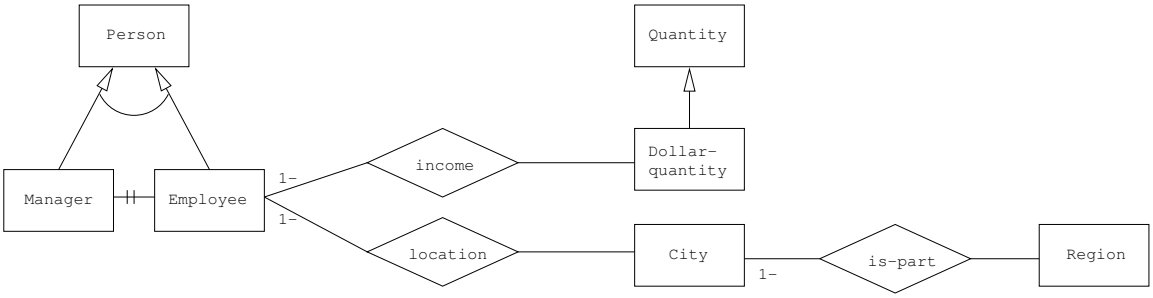




- An ER conceptual schema can be expressed in a suitable description logic theory.
- The models of the DL theory correspond with legal database states of the ER schemas.
- Reasoning services such as satisfiability of a schema or logical implication can be performed by the corresponding DL theory.
- A description logic allows for a greater expressivity than the original ER framework, in terms of full disjunction and negation, and entity definitions by means of both necessary and sufficient conditions.

Mapping an ER schema in a DL theory

- Relations are *reified* in the description logic theory, i.e., they become concepts with n special feature names denoting the n arguments of the n-ary relation.
- The relation `INCOME` becomes a concept with the two features:
  - `incomer` – relating to the first argument of the relation, i.e., an employee,
  - `incoming` – relating to the second argument of the relation, i.e., a dollar quantity.
- `incomer`, `incoming`, `locator`, `place`, `whole`, and `part` are functional roles.



$$\texttt{INCOME} \sqsubseteq \texttt{incomer} : \texttt{Employee} \sqcap \texttt{incoming} : \texttt{Dollar-quantity}$$

$$\texttt{LOCATION} \sqsubseteq \texttt{locator} : \texttt{Employee} \sqcap \texttt{place} : \texttt{City}$$

$$\texttt{IS-PART} \sqsubseteq \texttt{part} : \texttt{City} \sqcap \texttt{whole} : \texttt{Region}$$

$$\texttt{Employee} \sqsubseteq \texttt{Person} \sqcap \exists \texttt{incomer}^{-1}.\texttt{INCOME} \sqcap \exists \texttt{locator}^{-1}.\texttt{LOCATION}$$

$$\texttt{Manager} \sqsubseteq \texttt{Person} \sqcap \neg \texttt{Employee}$$

$$\texttt{Person} \sqsubseteq \texttt{Manager} \sqcup \texttt{Employee}$$

$$\texttt{Dollar-quantity} \sqsubseteq \texttt{Quantity}$$

$$\texttt{City} \sqsubseteq \exists \texttt{part}^{-1}.\texttt{IS-PART}$$

Object-Oriented Conceptual Data Models

- Is is intuitive to understand the relationship between a description logic and a generic Object-Oriented formalism.
- Unlike object-oriented systems, description logics do not stress the representation of the behavioral aspect of information, for which they are still considered inadequate.
- The translation of the structural part of an O-O schema into a description logic knowledge base is similar to the one sketched for ER schemas.

Advantages of DL for Conceptual Modeling

- *Ontological organization.* It is possible to capture important basic facets of data semantics, including the structure of complex entities.
- *Consistency checking.* It is possible to check whether the global information conveyed in a schema forces some specific class to be inconsistent. Moreover, one could check the consistency of the whole schema, also with respect to possible integrity constraints.

- *Data entry.* The user is supported in the phase of populating the data base, according to the knowledge of the schema and satisfying the integrity constraints. Then, the system could not only check the consistency of the data base itself, but also make some deductive inferences asserting new facts regarding the data. Moreover, the system supports the user in the refinement of the schema in a populated data base.

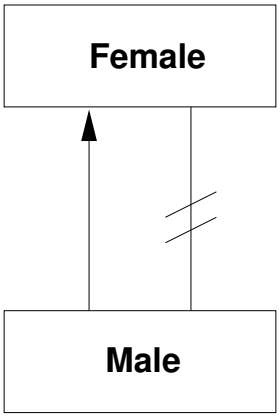
- *Views organization.* Views - i.e., pre-defined descriptions, grounded on the terms of the schema - are automatically organized into a hierarchy, which is a non-trivial task when there are many complex views. Taxonomic relations between views do explicit their meaning and their specificity, allowing for its retrieval and reuse.

- *Schema refinement.* It is possible to reduce the redundancy of a schema, by discovering equivalent descriptions, by reusing descriptions, and by exploiting the description lattice.
- *Inter-schema organization.* It is possible to define inter-schema knowledge describing the constraints among different databases, easing the task of managing multi-databases.

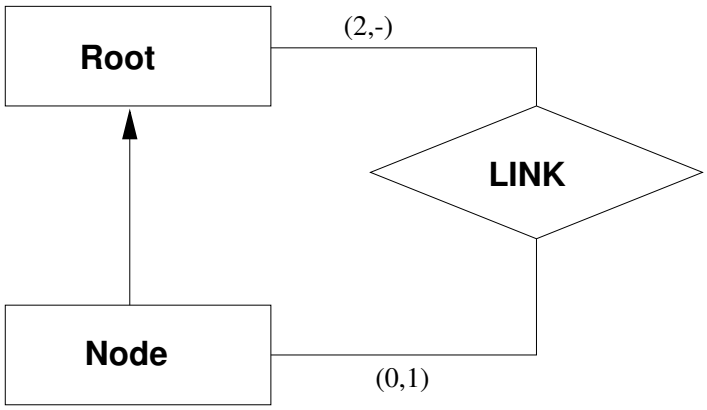
Finite Model Reasoning

- *Simple* description logics do have the Finite Model property: if a formula is satisfiable, then it is finitely satisfiable.
- However, very expressive description logics do not have the finite model property anymore.
- Satisfiability (logical implication) and finite satisfiability (logical implication) may diverge. The theory may infer a property holding only in the finite structures, but classical reasoning may not reveal this fact.
- Finite model reasoning is relevant for database conceptual modeling: databases are always finite.
- In order to model ER schemas, we need a logic which does not have the finite model property.

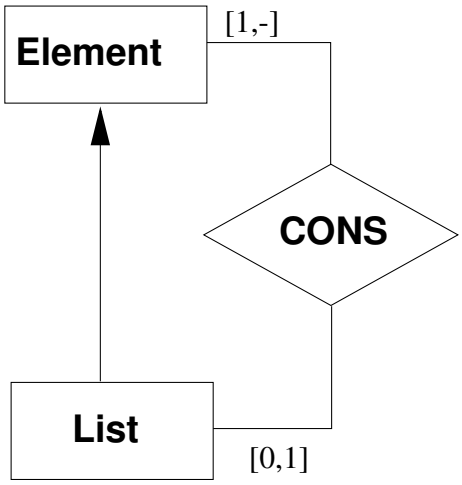
Examples

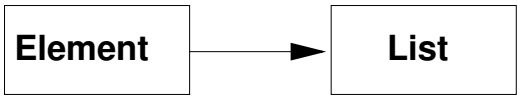


Unsatisfiable



Satisfiable, but not finitely satisfiable





Not logically implied, but finitely logically implied

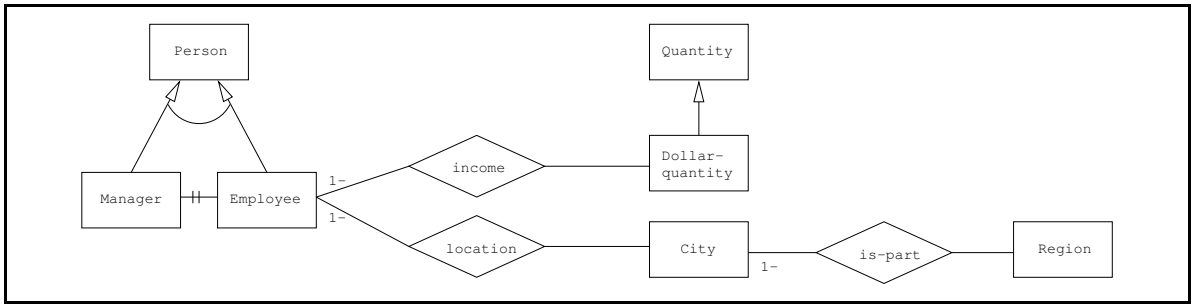
Querying with a conceptual schema

- Query containment of conjunctive queries (SPJ-queries) referring to predicates defined in a $\mathcal{ALCQI}_{reg}$ theory is EXPTIME-complete.
- It is possible to encode constraints over n-ary relations.
- The DL schema allows for recursive definitions, with full negation, disjunction, and universal quantification (compare with DATALOG).
- DL are able to fully encode many O-O and semantic data models, e.g., Entity-Relationship (ER), Object-Role Modeling (ORM), OMT static model. It can extend them in may ways, e.g., with negation, disjointness, covering constraints.

Querying with a conceptual schema

$$\mathtt{Q}(\mathtt{x},\mathtt{y},\mathtt{z}) \doteq \mathtt{Person}(\mathtt{x}) \wedge \neg\mathtt{Manager}(\mathtt{x}) \wedge \mathtt{INCOME}(\mathtt{x},\mathtt{y}) \wedge \mathtt{LOCATION}(\mathtt{x},\mathtt{z}).$$

$$\Downarrow$$



$$\Uparrow$$

$$\mathtt{Table}^{1}_{\mathtt{DB}_1}(\mathtt{x},\mathtt{y}) \doteq \mathtt{Employee}(\mathtt{x}) \wedge \mathtt{INCOME}(\mathtt{x},\mathtt{y}).$$

$$\mathtt{Table}^{2}_{\mathtt{DB}_1}(\mathtt{x},\mathtt{y}) \doteq \mathtt{Employee}(\mathtt{x}) \wedge \mathtt{LOCATION}(\mathtt{x},\mathtt{y}).$$

Advantages of DL

- *Query validation.* Incoherent queries - i.e., queries that can not return any value as answer, given their inconsistent meaning with respect to the schema - are detected, and the user is informed about its ill-formed request.
- *Query generalization.* In many situations, the query, even if it is consistent, can return an empty answer, since there is no actual object in the database satisfying it. In such cases, it is reasonable to generalize the query until a non empty answer is obtained; the description lattice is the obvious space where such generalizations can be searched for.

- *Query organization.* Data exploration may involve a great amount of queries, possibly submitted by different group of persons, in different periods of time, for different purposes. The system can organize the set of queries in a hierarchy, such that it is possible to retrieve already submitted similar or equivalent queries, together with the cached results. This is relevant if the queries need a substantial amount of time to be processed, or if the users associate comments or observations to the queries or to the answers.

- *Query refinement.* Queries can be specified through an iterative refinement process supported by the description lattice for the queries. This process is useful for data exploration tasks. The user may specify his/her request using generic terms; after the query classification, which makes explicit the meaning and the specificity of the query itself and of the terms composing the query, the user may refine some terms of the query or introduce new terms, and iterate the process.

- *Intensional query processing.* Users may explore and discover new generic facts without querying the whole information base, but by giving an explicit meaning to the queries through classification. The system has the ability of answering a query with synthetic concepts representing the general characteristics of the information that satisfy it, as opposed to answering with long sequences of detailed data. Moreover, if the query is classified in a taxonomy of descriptions and queries already computed and indexing the answers, then it can be processed with respect to the indexed objects only, rather than with respect to the whole information base.

- *Query optimization.* Given a schema and a set of views and already processed queries, a query can be optimized by computing an equivalent more efficient one. The optimized query can be obtained by using the cached results - maintained by a semantic indexing technique - retrieved by classification, and/or making more specific the single terms and the complex descriptions used within the query original formulation.

Basic References

- Alex Borgida, 'Description Logics in Data Management', IEEE Transactions on Knowledge and Data Engineering vol.7, No. 5, October 1995.
- Diego Calvanese, Maurizio Lenzerini, Daniele Nardi, 'Description Logics for Conceptual Data Modeling', Logics for Databases and Information Systems, J. Chomicki and G. Saake eds., Kluwer, 1998.
- Klaus Schild, 'Tractable reasoning in a universal description logic', Proceedings of 1st Workshop KRDB'94, Saarbrcken, Germany, September 20-22, 1994.