FOL as a Language to Query Databases

Alessandro Artale

Free University of Bozen-Bolzano
Faculty of Computer Science
http://www.inf.unibz.it/~artale

Descrete Mathematics and Logic — BSc course

Thanks to Chiara Ghidini, Annapaola Marconi and Luciano Serafini for provoding the material for these slides

Analogy with Databases

When the signature of the language, $\mathcal{L}$, and the domain of interpretation, Δ, are finite, and $\mathcal{L}$ doesn't contain functional symbols (relational language), there is a strict analogy between FOL and databases.

Analogy with Databases

When the signature of the language, $\mathcal{L}$, and the domain of interpretation, Δ, are finite, and $\mathcal{L}$ doesn't contain functional symbols (relational language), there is a strict analogy between FOL and databases.

- n-predicate symbols of $\mathcal{L}$ correspond to relations (tables) of the database schema

Analogy with Databases

When the signature of the language, $\mathcal{L}$, and the domain of interpretation, Δ, are finite, and $\mathcal{L}$ doesn't contain functional symbols (relational language), there is a strict analogy between FOL and databases.

- n-predicate symbols of $\mathcal{L}$ correspond to relations (tables) of the database schema
- the interpretation domain, Δ, corresponds to the set of values which appear in the tables

Analogy with Databases

When the signature of the language, $\mathcal{L}$, and the domain of interpretation, Δ, are finite, and $\mathcal{L}$ doesn't contain functional symbols (relational language), there is a strict analogy between FOL and databases.

- n-predicate symbols of $\mathcal{L}$ correspond to relations (tables) of the database schema
- the interpretation domain, Δ, corresponds to the set of values which appear in the tables
- constant symbols are interpreted as themselve, i.e., we use the *Standard Name Assumption*

Analogy with Databases

When the signature of the language, $\mathcal{L}$, and the domain of interpretation, Δ, are finite, and $\mathcal{L}$ doesn't contain functional symbols (relational language), there is a strict analogy between FOL and databases.

- n-predicate symbols of $\mathcal{L}$ correspond to relations (tables) of the database schema
- the interpretation domain, Δ, corresponds to the set of values which appear in the tables
- constant symbols are interpreted as themselve, i.e., we use the *Standard Name Assumption*
- the interpretation of a predicate symbol, $R^{\mathcal{I}}$, corresponds to the set of tuples that belong to the relation

Analogy with Databases

When the signature of the language, $\mathcal{L}$, and the domain of interpretation, Δ, are finite, and $\mathcal{L}$ doesn't contain functional symbols (relational language), there is a strict analogy between FOL and databases.

- n-predicate symbols of $\mathcal{L}$ correspond to relations (tables) of the database schema
- the interpretation domain, Δ, corresponds to the set of values which appear in the tables
- constant symbols are interpreted as themselve, i.e., we use the *Standard Name Assumption*
- the interpretation of a predicate symbol, $R^{\mathcal{I}}$, corresponds to the set of tuples that belong to the relation
- FOL formulas on $\mathcal{L}$ correspond to queries over the database

Analogy with Databases

When the signature of the language, $\mathcal{L}$, and the domain of interpretation, Δ, are finite, and $\mathcal{L}$ doesn't contain functional symbols (relational language), there is a strict analogy between FOL and databases.

- n-predicate symbols of $\mathcal{L}$ correspond to relations (tables) of the database schema
- the interpretation domain, Δ, corresponds to the set of values which appear in the tables
- constant symbols are interpreted as themselve, i.e., we use the *Standard Name Assumption*
- the interpretation of a predicate symbol, $R^{\mathcal{I}}$, corresponds to the set of tuples that belong to the relation
- FOL formulas on $\mathcal{L}$ correspond to queries over the database
- interpretation of formulas of $\mathcal{L}$ corresponds to answers.

Databases as Interpretations

Let us consider the following DB:

Students

NAME	ID	COURSE
Smith	123	CS
Johnson	522	ENG
Maria	124	CS

Teaches

NAME	COURSE
Alex	CS
Enrico	CS
Diego	ENG

[Interpretation Induced by a DB]. The above DB induces the following interpretation, $\mathcal{I}_{db} = (\Delta^{\mathcal{I}_{db}}, \cdot^{\mathcal{I}_{db}})$:

$\Delta^{\mathcal{I}_{db}} =$
$\{Smith, Johnson, Maria, Alex, Enrico, Diego, CS, ENG, 123, 124, 522\}$
$\text{Student}^{\mathcal{I}_{db}} =$
$\{(Smith, 123, CS), (Johnson, 522, ENG), (Maria, 124, CS)\}$
$\text{Teaches}^{\mathcal{I}_{db}} = \{(Alex, CS), (Enrico, CS), (Diego, ENG)\}$

Formulas as Queries

- A DB query with n answer variables, $(x_1, \ldots, x_n)$, can be formulated as an FOL formula over the signature $\mathcal{L}$, $\varphi(x_1, \ldots, x_n)$, with $\{x_1, \ldots, x_n\} = FREE(\varphi)$.
- Given a DB, $\mathcal{D}$, the Answer Set of an FOL query $\varphi(x_1, \ldots, x_n)$ with respect to $\mathcal{D}$ is:

$$ans(\varphi, \mathcal{D}) = \{(a_1, \ldots, a_n) \in \Delta^{\mathcal{I}_\mathcal{D}} \times \ldots \times \Delta^{\mathcal{I}_\mathcal{D}} \mid \mathcal{I}_\mathcal{D}, \alpha[x_1/a_1, \ldots, x_n/a_n] \models \varphi(x_1, \ldots, x_n)\}$$

Formulas as Queries: Example

Let us consider the following DB:

Students

NAME	ID	COURSE
Smith	123	CS
Johnson	522	ENG
Maria	124	CS

Teaches

NAME	COURSE
Alex	CS
Enrico	CS
Diego	ENG

Query. *Give Names of Students enrolled in CS.*

$$\varphi(x) = \exists y.Student(x, y, CS)$$

$$ans(\varphi(x), DB) = \{Smith, Maria\}$$

Analogy with Databases

FOL	**DB**
friends	`CREATE TABLE FRIENDS (friend1 : INTEGER` `friend2 : INTEGER)`

Analogy with Databases

FOL	DB
friends	`CREATE TABLE FRIENDS (friend1 : INTEGER friend2 : INTEGER)`
friends(x, y)	`SELECT friend1 AS x friend2 AS y FROM FRIENDS`

Analogy with Databases

FOL	DB
friends	`CREATE TABLE FRIENDS (friend1 : INTEGER friend2 : INTEGER)`
friends(x, y)	`SELECT friend1 AS x friend2 AS y FROM FRIENDS`
friends(x, x)	`SELECT friend1 AS x FROM FRIENDS WHERE friend1 = friend2`

Analogy with Databases

FOL	DB
friends	`CREATE TABLE FRIENDS (friend1 : INTEGER friend2 : INTEGER)`
$friends(x, y)$	`SELECT friend1 AS x friend2 AS y FROM FRIENDS`
$friends(x, x)$	`SELECT friend1 AS x FROM FRIENDS WHERE friend1 = friend2`
$friends(x, y) \wedge x = y$	`SELECT friend1 AS x friend2 AS y FROM FRIENDS WHERE friend1 = friend2`

Analogy with Databases

FOL	DB
friends	`CREATE TABLE FRIENDS (friend1 : INTEGER friend2 : INTEGER)`
$friends(x, y)$	`SELECT friend1 AS x friend2 AS y FROM FRIENDS`
$friends(x, x)$	`SELECT friend1 AS x FROM FRIENDS WHERE friend1 = friend2`
$friends(x, y) \wedge x = y$	`SELECT friend1 AS x friend2 AS y FROM FRIENDS WHERE friend1 = friend2`
$\exists x.friends(x, y)$	`SELECT friend2 AS y FROM FRIENDS`

FOL as queries over Databases: Examples

Example

Consider the following database schema:

- `Students(Name, University, OriginT, LiveT)`
- `Universities(Name, Town)`
- `Town(Name, Country)`

Express each of the following queries in FOL formulas with free variables.

1. Give `Names` of students living in `Trento`

FOL as queries over Databases: Examples

Example

Consider the following database schema:

- `Students(Name, University, OriginT, LiveT)`
- `Universities(Name, Town)`
- `Town(Name, Country)`

Express each of the following queries in FOL formulas with free variables.

1. Give `Names` of students living in `Trento`
   $\exists y \exists z.Students(x, y, z, Trento)$

FOL as queries over Databases: Examples

Example

Consider the following database schema:

- `Students(Name, University, OriginT, LiveT)`
- `Universities(Name, Town)`
- `Town(Name, Country)`

Express each of the following queries in FOL formulas with free variables.

1. Give `Names` of students living in `Trento`
   $\exists y \exists z.Students(x, y, z, Trento)$
2. Give `Names` of students studying in a university in `Trento`

FOL as queries over Databases: Examples

Example

Consider the following database schema:

- `Students(Name, University, OriginT, LiveT)`
- `Universities(Name, Town)`
- `Town(Name, Country)`

Express each of the following queries in FOL formulas with free variables.

1. Give `Names` of students living in `Trento`
   $\exists y \exists z . Students(x, y, z, Trento)$
2. Give `Names` of students studying in a university in `Trento`
   $\exists y \exists z \exists v . (Students(x, y, z, v) \wedge Universities(y, Trento))$

FOL as queries over Databases: Examples

Example

Consider the following database schema:

- `Students(Name, University, OriginT, LiveT)`
- `Universities(Name, Town)`
- `Town(Name, Country)`

Express each of the following queries in FOL formulas with free variables.

1. Give `Names` of students living in `Trento`
   $\exists y \exists z.Students(x, y, z, Trento)$
2. Give `Names` of students studying in a university in `Trento`
   $\exists y \exists z \exists v.(Students(x, y, z, v) \wedge Universities(y, Trento))$
3. Give `Names` of students living in their origin town

FOL as queries over Databases: Examples

Example

Consider the following database schema:

- `Students(Name, University, OriginT, LiveT)`
- `Universities(Name, Town)`
- `Town(Name, Country)`

Express each of the following queries in FOL formulas with free variables.

1. Give `Names` of students living in `Trento`
   $\exists y \exists z.Students(x, y, z, Trento)$
2. Give `Names` of students studying in a university in `Trento`
   $\exists y \exists z \exists v.(Students(x, y, z, v) \wedge Universities(y, Trento))$
3. Give `Names` of students living in their origin town
   $\exists y \exists z.Students(x, y, z, z)$

FOL as queries over Databases: Examples

Example

Consider the following database schema:

- `Students(Name, University, OriginT, LiveT)`
- `Universities(Name, Town)`
- `Town(Name, Country)`

Express each of the following queries in FOL formulas with free variables.

4. Give `(Name, University)` pairs for each student studying in `Italy`

FOL as queries over Databases: Examples

Example

Consider the following database schema:

- `Students(Name, University, OriginT, LiveT)`
- `Universities(Name, Town)`
- `Town(Name, Country)`

Express each of the following queries in FOL formulas with free variables.

4. Give `(Name, University)` pairs for each student studying in `Italy`
   $\exists z \exists v \exists w.(Students(x, y, z, v) \wedge Universities(y, w) \wedge Town(w, Italy))$

FOL as queries over Databases: Examples

Example

Consider the following database schema:

- `Students(Name, University, OriginT, LiveT)`
- `Universities(Name, Town)`
- `Town(Name, Country)`

Express each of the following queries in FOL formulas with free variables.

4. Give `(Name, University)` pairs for each student studying in `Italy`
   $\exists z \exists v \exists w.(Students(x, y, z, v) \wedge Universities(y, w) \wedge Town(w, Italy))$
5. Give all `Country` that have at least one university for each town.

FOL as queries over Databases: Examples

Example

Consider the following database schema:

- `Students(Name, University, OriginT, LiveT)`
- `Universities(Name, Town)`
- `Town(Name, Country)`

Express each of the following queries in FOL formulas with free variables.

4. Give `(Name, University)` pairs for each student studying in `Italy`
   $\exists z \exists v \exists w.(Students(x, y, z, v) \wedge Universities(y, w) \wedge Town(w, Italy))$
5. Give all `Country` that have at least one university for each town.
   $\forall x.(Town(x, y) \rightarrow \exists z.Universities(z, x))$

FOL as queries over Databases: Exercise

Exercise

Consider the following database schema

- `Lives(Name,Town)`
- `Works(Name,Company,Salary)`
- `Company_Location(Company,Town)`
- `Reports_To(Name,Manager)`

(you may use the abbreviations `L(N,T)`, `W(N,C,S)`, `CL(C,T)`, and `R(N,M)`).
Express each of the following queries in first order formulas with free variables.

1. Give `(Name,Town)` pairs for each person working for `Fiat`.
2. Find all people who live and work in the same town.
3. Find the maximum salary of all people who work in `Trento`.
4. Find the names of all companies which are located in every city that has a branch of `Fiat`