FOL Query Evaluation

Giuseppe De Giacomo

Università di Roma “La Sapienza”

Corso di Seminari di Ingegneria del Software:
Data and Service Integration
Laurea Specialistica in Ingegneria Informatica
Università degli Studi di Roma “La Sapienza”
A.A. 2006-07

First-order logic

- First-order logic (FOL) is the logic to speak about object, which are the domain of discourse or universe.
- FOL is concerned about Properties of these objects and Relations over objects (resp. unary and n-ary Predicates)
- FOL also has Functions including Constants that denote objects.

First-order logic: syntax - terms

Terms: defined inductively as follows

- *Vars*: A set $\{x_1, \ldots, x_n\}$ of individual variables (variables that denote single objects)
- Function symbols (including constants: a set of functions symbols of given arity > 0. Functions of arity 0 are called constants.
- *Vars* $\subseteq$ *Terms*
- if $t_1, \ldots, t_k \in$ *Terms* and f^k is a k-ary function, then $f^k(t_1, \ldots, t_k) \in$ *Terms*
- nothing else is in *Terms*.

First-order logic: syntax - formulas

Formulas: defined inductively as follows

- if $t_1, \ldots, t_k \in$ *Terms* and P^k is a k-ary predicate, then $P^k(t_1, \ldots, t_k) \in$ *Formulas* (atomic formulas)
- $\phi \in$ *Formulas* and $\psi \in$ *Formulas* then
  - $\neg\phi \in$ *Formulas*
  - $\phi \wedge \psi \in$ *Formulas*
  - $\phi \vee \psi \in$ *Formulas*
  - $\phi \supset \psi \in$ *Formulas*
- $\phi \in$ *Formulas* and $x \in$ *Vars* then
  - $\exists x.\phi \in$ *Formulas*
  - $\forall x.\phi \in$ *Formulas*

- nothing else is in *Formulas*.

Note: if a predicate is of arity P_i , then it is a proposition of propositional logic.

First-order logic: Semantics - interpretations

Given an alphabet of predicates and functions, each with associated arity, $P_1, \ldots P_i, \ldots, f_1, \ldots, f_i, \ldots$, A FOL interpretation is

$$\mathcal{I} = (\Delta^{\mathcal{I}}, P_1^{\mathcal{I}}, \ldots P_i^{\mathcal{I}}, \ldots, f_1^{\mathcal{I}}, \ldots, f_i^{\mathcal{I}}, \ldots)$$

where:

- $\Delta^{\mathcal{I}}$ is the domain (a set of objects)
- if P_i is a k-arity predicate, then $P_i^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \cdots \times \Delta^{\mathcal{I}}$ (k times)
- if f_i is a k-arity function, then $f_i^{\mathcal{I}} : \Delta^{\mathcal{I}} \times \cdots \times \Delta^{\mathcal{I}} \longrightarrow \Delta^{\mathcal{I}}$ (k times)
- if f_i is a constant (i.e., 0-arity function), then $f_i^{\mathcal{I}} : () \longrightarrow \Delta^{\mathcal{I}}$ (i.e., denotes exactly one object of the domain)

First-order logic: Semantics - assignment

Let *Vars* be a set of (individual) variables, then given an interpretation $\mathcal{I}$ an assignment is a function

$$\alpha : \textit{Vars} \longrightarrow \Delta^{\mathcal{I}}$$

that assigns to each variable $x \in$ *Vars* an object $\alpha(x) \in \Delta^{\mathcal{I}}$.

It is convenient to extend the notion of assignment to terms. We can do it by defining a function $\bar{\alpha} :$ *Terms* $\longrightarrow \Delta^{\mathcal{I}}$ inductively as follows:

- $\bar{\alpha}(x) = \alpha(x)$, if $x \in$ *Vars*
- $\bar{\alpha}(f(t_1, \ldots, t_k)) = f^{\mathcal{I}}(\bar{\alpha}(t_1), \ldots, \bar{\alpha}(t_k))$

Note: for constants $\bar{\alpha}(c) = c^{\mathcal{I}}$.

First-order logic: Semantics - truth in an interpretation wrt an assignment

We say that a FOL formula ϕ is true in an interpretation $\mathcal{I}$ wrt an assignment α, written $\mathcal{I}, \alpha \models \phi$

- $\mathcal{I}, \alpha \models P(t_1, \ldots, t_k)$ if $(\bar{\alpha}(t_1), \ldots, \bar{\alpha}(t_k)) \in P^{\mathcal{I}}$;
- $\mathcal{I}, \alpha \models \neg\phi$ if $\mathcal{I}, \alpha \not\models \phi$
- $\mathcal{I}, \alpha \models \phi \wedge \psi$ if $\mathcal{I}, \alpha \models \phi$ and $\mathcal{I}, \alpha \models \psi$
- $\mathcal{I}, \alpha \models \phi \vee \psi$ if $\mathcal{I}, \alpha \models \phi$ or $\mathcal{I}, \alpha \models \psi$
- $\mathcal{I}, \alpha \models \phi \supset \psi$ if, $\mathcal{I}, \alpha \models \phi$ implies $\mathcal{I}, \alpha \models \psi$
- $\mathcal{I}, \alpha \models \exists x.\phi$ if for some $a \in \Delta^{\mathcal{I}}$ we have $\mathcal{I}, \alpha[x \mapsto a] \models \phi$
- $\mathcal{I}, \alpha \models \forall x.\phi$ if for every $a \in \Delta^{\mathcal{I}}$ we have $\mathcal{I}, \alpha[x \mapsto a] \models \phi$

Here $\alpha[x \mapsto a]$ stands for the new assignment obtained from α as follows:

$$\begin{aligned} \alpha[x \mapsto a](x) &= a \\ \alpha[y \mapsto a](y) &= \alpha(y) \qquad (y \neq x) \end{aligned}$$

Note: for constants $\bar{\alpha}(c) = c^{\mathcal{I}}$.

First-order logic: open vs. closed formulas

A variable x in a formula ϕ is free if x does not occur in the scope of any quantifier, otherwise is bounded.

An open formula is a formula that has some free variable.

A closed formula, also called sentence, is a formula that has no free variables.

For closed formulas (but not for open formulas) we can straightforwardly define what it means to true in an interpretation, written $\mathcal{I} \models \phi$, without mentioning the assignment, since the assignment α does not play any role in verifying $\mathcal{I}, \alpha \models \phi$.

Instead open formulas are strongly related to queries – cf. relational databases.

FOL queries

A FOL query is an (open) FOL formula.

Let ϕ be a FOL query with free variables $(x_1, \ldots, x_k)$, then we sometimes write it as $\phi(x_1, \ldots, x_k)$.

Given an interpretation $\mathcal{I}$, the assignments we are interested in are those that map the variables $x_1, \ldots, x_k$ (and only those). We will write such assignment explicitly sometimes: i.e., $\alpha(x_i) = a_i$ $(i = 1, \ldots, k)$, is written simply as $\langle a_1, \ldots, a_k \rangle$.

Now we define the answer to a query $\phi(x_1, \ldots, x_k)$ as follows

$$\phi(x_1, \ldots, x_k)^{\mathcal{I}} = \{(a_1, \ldots, a_k) \mid \mathcal{I}, \langle a_1, \ldots, a_k \rangle \models \phi(x_1, \ldots, x_k)\}$$

Note: We will also use the notation: $\phi^{\mathcal{I}}$, keeping the free variables implicit, and $\phi(\mathcal{I})$ making apparent that ϕ becomes a functions from interpretations to set of tuples.

FOL boolean queries

A FOL boolean query is a FOL query without free variables.

Hence the answer to a boolean query $\phi()$ as follows

$$\phi()^{\mathcal{I}} = \{() \mid \mathcal{I}, \langle\rangle \models \phi()\}$$

Such an answer is $\langle\rangle$ if $\mathcal{I} \models \phi$ and $\emptyset$ is $\mathcal{I}\neg \models \phi$. As an obvious convention we read $\langle\rangle$ as "true" and $\emptyset$ as "false".

FOL formulas: logical tasks

- Validity: ϕ is valid iff for all $\mathcal{I}$ and α we have $\mathcal{I}, \alpha \models \phi$;
- Satisfiability: ϕ is satisfiable iff there exists an $\mathcal{I}$ and α such that $\mathcal{I}, \alpha \models \phi$; unsatisfiable otherwise;
- Logical implication: ϕ logically implies ψ, written $\phi \models \psi$ iff for all $\mathcal{I}$ and α, if $\mathcal{I}, \alpha \models \phi$ then $\mathcal{I}, \alpha \models \psi$;
- Logical equivalence: ϕ is logically equivalent to ψ, iff for all $\mathcal{I}$ and α, $\mathcal{I}, \alpha \models \phi$ iff $\mathcal{I}, \alpha \models \psi$ (i.e., $\phi \models \psi$ and $\psi \models \phi$);

FOL queries: logical tasks

- Validity: if ϕ is valid, then $\phi^{\mathcal{I}} = \Delta^{\mathcal{I}} \times \ldots \times \Delta^{\mathcal{I}}$, i.e., the query returns all the tuples of $\mathcal{I}$.
- Satisfiability: ϕ is satisfiable, then $\phi^{\mathcal{I}} \neq \emptyset$, i.e., the query returns some tuples.
- Logical implication: ϕ logically implies ψ, then $\phi^{\mathcal{I}} \subseteq \psi^{\mathcal{I}}$ for all $\mathcal{I}$, written $\phi \subseteq \psi$, i.e., the answer to ϕ is contained in that of ψ in every interpretation; this is called query containment;
- Logical equivalence: ϕ is logically equivalent to ψ, then $\phi^{\mathcal{I}} = \psi^{\mathcal{I}}$ for all $\mathcal{I}$, written $\phi = \psi$, i.e., the answer to the two queries is the same in every interpretation. This is called query equivalence and correspond to query containment in both directions.

Note: We have analogous tasks if we have axioms, i.e., constraints on the

admissible interpretations.

Query evaluation problem

Let us consider a finite alphabet (i.e., we have a finite number of predicates and functions) and a finite interpretation $\mathcal{I}$ (an interpretation over a finite alphabet, where $\Delta^{\mathcal{I}}$ is finite).

Then we can define query evaluation (aka query answering) as an algorithmic problem and study its computational complexity. In fact since to study complexity we need to look at the recognition problem, which is a decision

- query answering problem: given finite interpretation $\mathcal{I}$ and a FOL query ϕ, compute:
$$\phi^{\mathcal{I}} = \{(a_1, \ldots, a_k) \mid \mathcal{I}, \langle a_1, \ldots, a_k \rangle \models \phi\}$$
- (query answering) recognition problem: given finite interpretation $\mathcal{I}$ and a FOL query ϕ and a tuple $\langle a_1, \ldots, a_k \rangle$ ($a_i \in \Delta^{\mathcal{I}}$), check whether

$(a_1, \ldots, a_k) \in \phi^{\mathcal{I}}$, i.e., whether
$$\mathcal{I}, \langle a_1, \ldots, a_k \rangle \models \phi$$

Query evaluation algorithm

```
boolean Truth($\mathcal{I}$,$\alpha$,$\phi$ ) {
  if($\phi$ is $t\_1 = t\_2$)
     return TermEval($t\_1$) = TermEval($t\_2$);
  if($\phi$ is $P(t\_1,\ldots,t\_k)$)
     return $P$^$\mathcal{I}$(TermEval($t\_1$),...,TermEval($t\_k$));
  if($\phi$ is $\neg\psi$)
     return $\neg$Truth($\mathcal{I}$,$\alpha$,$\psi$);
  if($\phi$ is $\psi \circ \psi'$)
     return Truth($\mathcal{I}$,$\alpha$,$\psi$) $\circ$ Truth($\mathcal{I}$,$\alpha$,$\psi'$);
  if($\phi$ is $\exists\ x.\ \psi$) {
     boolean b = false;
     forall($a \in \Delta^{\mathcal{I}}$)
        b = b $\vee$ Truth($\mathcal{I}$,$\alpha[x \mapsto a]$,$\psi$);
     return b;
```

```
  }
  if($\phi$ is $\forall\ x.\ \psi$) {
     boolean b = true;
     forall($a \in \Delta^{\mathcal{I}}$)
        b = b $\wedge$ Truth($\mathcal{I}$,$\alpha[x \mapsto a]$,$\psi$);
     return b;
  }
}


$o \in \Delta^{\mathcal{I}}$ TermEval($\mathcal{I}$,$\alpha$,$t$) {
   if($t$ is $x \in Vars$) return $\alpha(x)$;
   if($t$ is $f(t\_1,\ldots,t\_k)$)
      return $f$^$\mathcal{I}$(TermEval($t\_1$),...,TermEval($t\_k$));
}
```

Query evaluation: results

Thm1(Termination): The algorithm `Truth` terminates.

Proof. immediate. □

Thm2 (Correctness): The algorithm `Truth` is sound and complete: $\mathcal{I}, \alpha \models \phi$ if and only if `Truth(`$\mathcal{I}$`,`α`,`ϕ`) = true`.

Proof. Easy: the algorithm is very close to the semantic definition of $\mathcal{I}, \alpha \models \phi$. □

Query evaluation: time complexity

Thm (time complexity): $(|\mathcal{I}| + |\alpha| + |\phi|)^{|\phi|}$, i.e., polynomial in the size of $\mathcal{I}$ and exponential in the size of ϕ.

Proof.

1. $f^{\mathcal{I}}(\ldots)$ can be represented as k-dimensional array, hence accessing the required element can be done in linear time in $\mathcal{I}$;
2. `TermEval(...)` simply visits the term, so it generates a polynomial number of recursive calls, hence is time polynomial in $(|\mathcal{I}| + |\alpha| + |\phi|)$;
3. $P^{\mathcal{I}}(\ldots)$ can be represented as k-dimensional boolean array, hence accessing the required element can be done in linear time in $\mathcal{I}$;

4. `Truth(...)` for the boolean cases simply visit the formula, so generate either one or two recursive calls;
5. `Truth(...)` for the quantified cases $\exists x.\phi$ and $\forall x.\psi$ involve looping for all elements in $\Delta^{\mathcal{I}}$ and testing the resulting assignments;
6. The total number of such testings is $O(|\mathcal{I}|^{|Vars|})$;

Hence the thesis □.

Query evaluation: space complexity

Thm (space complexity): $|\phi| * (|\phi| * log(|\mathcal{I}|))$, i.e., logarithmic in the size of $\mathcal{I}$ and polynomial in the size of ϕ.

Proof.

1. $f^{\mathcal{I}}(\ldots)$ can be represented as k-dimensional array, hence accessing the required element requires $O(log(|\mathcal{I}|)$;
2. `TermEval(...)` simply visits the term, so it generates a polynomial number of recursive calls. each activation record has a constant size, and we need $O(|\phi|)$ activation record;
3. $P^{\mathcal{I}}(\ldots)$ can be represented as k-dimensional boolean array, hence accessing the required element requires $O(log(|\mathcal{I}|)$;

4. `Truth(...)` for the boolean cases simply visit the formula, so generate either one or two recursive calls, each of constant size;

5. `Truth(...)` for the quantified cases $\exists x.\phi$ and $\forall x.\psi$ involve looping for all elements in $\Delta^{\mathcal{I}}$ and testing the resulting assignments;

6. The total number of activation records that need to be at the same time on the stack is $O(\#\mathit{Vars}) \leq O(|\phi|)$;
(the worst case form for the formula is
$\forall x_1.\exists x_2.\cdots\forall x_{n-1}.\exists x_n.p(x_1, x_2, \ldots, x_{n-1}, x_n)$.)

Hence the thesis $\Box$.

Query evaluation: combined, data, query complexity

Combined complexity: complexity of $\{\langle\mathcal{I}, \alpha, \phi\rangle \mid \mathcal{I}, \alpha \models \phi\}$, i.e., interpretation, tuple, and query part of the input:

- time: exponential
- space: PSPACE *(PSPACE-complete –see [Vardi82] for hardness)*

Data complexity: complexity of $\{\langle\mathcal{I}, \alpha\rangle \mid \mathcal{I}, \alpha \models \phi\}$, i.e., interpretation fixed (not part of the input):

- time: polynomial
- space: LOGSPACE *(LOGSPACE-complete –see [Vardi82] for hardness)*

Query complexity: complexity of $\{\langle\alpha, \phi\rangle \mid \mathcal{I}, \alpha \models \phi\}$, i.e., query fixed (not part of the input):

- time: exponential
- space: PSPACE *(PSPACE-complete –see [Vardi82] for hardness)*