First Order Logic

Formal Methods

Giuseppe De Giacomo

Sapienza Università di Roma
MSc in Engineering in Computer Science

First-order logic

- First-order logic (FOL) is the logic to speak about objects, 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.

FOL syntax – Terms

We first introduce:

- A set $Vars = \{x_1, \ldots, x_n\}$ of individual variables (i.e., variables that denote single objects).
- A set of functions symbols, each of given arity ≥ 0.
  Functions of arity 0 are called constants.

Def.: The set of *Terms* is defined inductively as follows:

- $Vars \subseteq Terms$;
- If $t_1, \ldots, t_k \in Terms$ and f^k is a k-ary function symbol, then $f^k(t_1, \ldots, t_k) \in Terms$;
- Nothing else is in *Terms*.

FOL syntax – Formulas

Def.: The set of *Formulas* is 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).
- If $t_1, t_2 \in Terms$, then $t_1 = t_2 \in Formulas$.
- If $\varphi \in Formulas$ and $\psi \in Formulas$ then
  - $\neg\varphi \in Formulas$
  - $\varphi \wedge \psi \in Formulas$
  - $\varphi \vee \psi \in Formulas$
  - $\varphi \rightarrow \psi \in Formulas$
- If $\varphi \in Formulas$ and $x \in Vars$ then
  - $\exists x.\varphi \in Formulas$
  - $\forall x.\varphi \in Formulas$
- Nothing else is in *Formulas*.

Note: a predicate of arity 0 is a proposition of propositional logic.

Interpretations

Given an alphabet of predicates $P_1, P_2, \ldots$ and functions $f_1, f_2, \ldots$, each with an associated arity, a FOL interpretation is:

$$\mathcal{I} = (\Delta^{\mathcal{I}}, P_1^{\mathcal{I}}, P_2^{\mathcal{I}}, \ldots, f_1^{\mathcal{I}}, f_2^{\mathcal{I}}, \ldots)$$

where:

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

Assignment

Let *Vars* be a set of (individual) variables.

Def.: Given an interpretation $\mathcal{I}$, an assignment is a function

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

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

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

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

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

Truth in an interpretation wrt an assignment

We define when a FOL formula φ is true in an interpretation $\mathcal{I}$ wrt an assignment α, written $\mathcal{I}, \alpha \models \varphi$:

- $\mathcal{I}, \alpha \models P(t_1, \ldots, t_k)$ if $(\hat{\alpha}(t_1), \ldots, \hat{\alpha}(t_k)) \in P^{\mathcal{I}}$
- $\mathcal{I}, \alpha \models t_1 = t_2$ if $\hat{\alpha}(t_1) = \hat{\alpha}(t_2)$
- $\mathcal{I}, \alpha \models \neg\varphi$ if $\mathcal{I}, \alpha \not\models \varphi$
- $\mathcal{I}, \alpha \models \varphi \wedge \psi$ if $\mathcal{I}, \alpha \models \varphi$ and $\mathcal{I}, \alpha \models \psi$
- $\mathcal{I}, \alpha \models \varphi \vee \psi$ if $\mathcal{I}, \alpha \models \varphi$ or $\mathcal{I}, \alpha \models \psi$
- $\mathcal{I}, \alpha \models \varphi \rightarrow \psi$ if $\mathcal{I}, \alpha \models \varphi$ implies $\mathcal{I}, \alpha \models \psi$
- $\mathcal{I}, \alpha \models \exists x.\varphi$ if for some $a \in \Delta^{\mathcal{I}}$ we have $\mathcal{I}, \alpha[x \mapsto a] \models \varphi$
- $\mathcal{I}, \alpha \models \forall x.\varphi$ if for every $a \in \Delta^{\mathcal{I}}$ we have $\mathcal{I}, \alpha[x \mapsto a] \models \varphi$

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

$$\alpha[x \mapsto a](x) = a$$
$$\alpha[x \mapsto a](y) = \alpha(y) \quad \text{for } y \neq x$$

Open vs. closed formulas

Definitions

- A variable x in a formula φ is free if x does not occur in the scope of any quantifier, otherwise it 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 define what it means to be true in an interpretation, written $\mathcal{I} \models \varphi$, without mentioning the assignment, since the assignment α does not play any role in verifying $\mathcal{I}, \alpha \models \varphi$.

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

FOL queries

Def.: A FOL query is an (open) FOL formula.

When φ is a FOL query with free variables $(x_1, \ldots, x_k)$, then we sometimes write it as $\varphi(x_1, \ldots, x_k)$, and say that φ has arity k.

Given an interpretation $\mathcal{I}$, we are interested in those assignments that map the variables $x_1, \ldots, x_k$ (and only those). We write an assignment α s.t. $\alpha(x_i) = a_i$, for $i = 1, \ldots, k$, as $\langle a_1, \ldots, a_k \rangle$.

Def.: Given an interpretation $\mathcal{I}$, the answer to a query $\varphi(x_1, \ldots, x_k)$ is

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

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

FOL boolean queries

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

Hence, the answer to a boolean query $\varphi()$ is defined as follows:

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

Such an answer is

- $()$, if $\mathcal{I} \models \varphi$
- $\emptyset$, if $\mathcal{I} \not\models \varphi$.

As an obvious convention we read $()$ as "true" and $\emptyset$ as "false".

FOL formulas: logical tasks

Definitions

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

FOL queries – Logical tasks

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

Note: These definitions can be extended to the case where we have axioms, i.e., constraints on the admissible interpretations.

Query evaluation

Let us consider:

- a finite alphabet, i.e., we have a finite number of predicates and functions, and
- a finite interpretation $\mathcal{I}$, i.e., an interpretation (over the finite alphabet) for which $\Delta^{\mathcal{I}}$ is finite.

Then we can consider query evaluation as an algorithmic problem, and study its computational properties.

Note: To study the computational complexity of the problem, we need to define a corresponding decision problem.

Query evaluation problem

Definitions

- Query answering problem: given a finite interpretation $\mathcal{I}$ and a FOL query $\varphi(x_1, \ldots, x_k)$, compute

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

- Recognition problem (for query answering): given a finite interpretation $\mathcal{I}$, a FOL query $\varphi(x_1, \ldots, x_k)$, and a tuple $(a_1, \ldots, a_k)$, with $a_i \in \Delta^{\mathcal{I}}$, check whether $(a_1, \ldots, a_k) \in \varphi^{\mathcal{I}}$, i.e., whether

$$\mathcal{I}, \langle a_1, \ldots, a_k \rangle \models \varphi(x_1, \ldots, x_k)$$

Note: The recognition problem for query answering is the decision problem corresponding to the query answering problem.

Query evaluation algorithm

We define now an algorithm that computes the function $\texttt{Truth}(\mathcal{I}, \alpha, \varphi)$ in such a way that $\texttt{Truth}(\mathcal{I}, \alpha, \varphi) = \texttt{true}$ iff $\mathcal{I}, \alpha \models \varphi$.

We make use of an auxiliary function $\texttt{TermEval}(\mathcal{I}, \alpha, t)$ that, given an interpretation $\mathcal{I}$ and an assignment α, evaluates a term t returning an object $o \in \Delta^{\mathcal{I}}$:

```
Δ^I TermEval(I,α,t) {
   if (t is x ∈ Vars)
      return α(x);
   if (t is f(t_1,...,t_k))
      return f^I(TermEval(I,α,t_1),...,TermEval(I,α,t_k));
}
```

Then, $\texttt{Truth}(\mathcal{I}, \alpha, \varphi)$ can be defined by structural recursion on φ.

Query evaluation algorithm (cont'd)

```
boolean Truth(I,α,φ) {
  if (φ is t_1 = t_2)
    return TermEval(I,α,t_1) = TermEval(I,α,t_2);
  if (φ is P(t_1,...,t_k))
    return P^I(TermEval(I,α,t_1),...,TermEval(I,α,t_k));
  if (φ is ¬ψ)
    return ¬Truth(I,α,ψ);
  if (φ is ψ ∘ ψ')
    return Truth(I,α,ψ) ∘ Truth(I,α,ψ');
  if (φ is ∃x.ψ) {
    boolean b = false;
    for all (a ∈ Δ^I)
       b = b ∨ Truth(I,α[x ↦ a],ψ);
    return b;
  }
  if (φ is ∀x.ψ) {
    boolean b = true;
    for all (a ∈ Δ^I)
       b = b ∧ Truth(I,α[x ↦ a],ψ);
    return b;
  }
}
```

Query evaluation – Results

Theorem (Termination of $\texttt{Truth}(\mathcal{I}, \alpha, \varphi)$)

The algorithm $\texttt{Truth}$ *terminates.*

Proof. Immediate. □

Theorem (Correctness)

The algorithm $\texttt{Truth}$ *is sound and complete, i.e.,* $\mathcal{I}, \alpha \models \varphi$ *if and only if* $\texttt{Truth}(\mathcal{I}, \alpha, \varphi) = \texttt{true}$.

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

Query evaluation – Time complexity I

Theorem (Time complexity of $\texttt{Truth}(\mathcal{I}, \alpha, \varphi)$)

The time complexity of $\texttt{Truth}(\mathcal{I}, \alpha, \varphi)$ *is* $O((|\mathcal{I}| + |\alpha| + |\varphi|)^{|\varphi|})$*, i.e., polynomial in the size of* $\mathcal{I}$ *and exponential in the size of* φ.

Proof.

- $f^{\mathcal{I}}$ (of arity k) can be represented as k-dimensional array, hence accessing the required element can be done in time linear in $|\mathcal{I}|$.
- $\texttt{TermEval}(\ldots)$ visits the term, so it generates a linear number of recursive calls, hence its time cost is $O(|\varphi| \cdot (|\mathcal{I}| + |\alpha|))$, i.e., polynomial time in $(|\mathcal{I}| + |\alpha| + |\varphi|)$.
- $P^{\mathcal{I}}$ (of arity k) can be represented as k-dimensional boolean array, hence accessing the required element can be done in time linear in $|\mathcal{I}|$.
- $\texttt{Truth}(\ldots)$ for the boolean cases simply visits the formula, so generates either one or two recursive calls.

Query evaluation – Time complexity II

- `Truth`$(\ldots)$ for the quantified cases $\exists x.\varphi$ and $\forall x.\psi$ involves looping for all elements in $\Delta^{\mathcal{I}}$ and testing the resulting assignments.
- The total number of such testings is $O(|\Delta^{\mathcal{I}}|^{\sharp Vars})$.

Considering that
$O((|\varphi| \cdot (|\mathcal{I}| + |\alpha|)) \cdot |\Delta^{\mathcal{I}}|^{\sharp Vars}) \leq O(|\mathcal{I}| + |\alpha| + |\varphi|)^{(2+|\varphi|)})$, the claim holds. □

Query evaluation – Space complexity I

Theorem (Space complexity of `Truth`$(\mathcal{I}, \alpha, \varphi)$)

The space complexity of `Truth`$(\mathcal{I}, \alpha, \varphi)$ *is* $O(|\varphi| \cdot (|\varphi| \cdot \log |\mathcal{I}|))$*, i.e., logarithmic in the size of* $\mathcal{I}$ *and polynomial in the size of* φ*.*

Proof.

- $f^{\mathcal{I}}(\ldots)$ can be represented as k-dimensional array, hence accessing the required element requires $O(\log |\mathcal{I}|)$;
- `TermEval`$(\ldots)$ simply visits the term, so it generates a linear number of recursive calls. Each activation record has a size $O(\log |\mathcal{I}|)$ to evaluate the function call it represent, and we need $O(|\varphi|)$ activation records;
- $P^{\mathcal{I}}(\ldots)$ can be represented as k-dimensional boolean array, hence accessing the required element requires $O(\log |\mathcal{I}|)$;
- `Truth`$(\ldots)$ for the boolean cases simply visits the formula, so generates either one or two recursive calls, each requiring constant size;
- `Truth`$(\ldots)$ for the quantified cases $\exists x.\varphi$ and $\forall x.\psi$ involves looping for all elements in $\Delta^{\mathcal{I}}$ and testing the resulting assignments;

Query evaluation – Space complexity II

- The total number of activation records that need to be at the same time on the stack is $O(\sharp Vars)$.

Hence, we have $O(\sharp Vars \cdot (|\varphi| \cdot log(|\mathcal{I}|)) \leq O(|\varphi| \cdot (|\varphi| \cdot \log(|\mathcal{I}|))$ the claim holds. □

Note: the worst case form for the formula is

$$\forall x_1.\exists x_2.\cdots\forall x_{n-1}.\exists x_n.P(x_1, x_2, \ldots, x_{n-1}, x_n).$$

Query evaluation – Complexity measures [Var82]

Definition (Combined complexity)

The combined complexity is the complexity of $\{\langle \mathcal{I}, \alpha, \varphi\rangle \mid \mathcal{I}, \alpha \models \varphi\}$, i.e., interpretation, tuple, and query are all considered part of the input.

Definition (Data complexity)

The data complexity is the complexity of $\{\langle \mathcal{I}, \alpha\rangle \mid \mathcal{I}, \alpha \models \varphi\}$, i.e., the query φ is fixed (and hence not considered part of the input).

Definition (Query complexity)

The query complexity is the complexity of $\{\langle \alpha, \varphi\rangle \mid \mathcal{I}, \alpha \models \varphi\}$, i.e., the interpretation $\mathcal{I}$ is fixed (and hence not considered part of the input).

Query evaluation – Combined, data, query complexity

Theorem (Combined complexity of query evaluation)

The complexity of $\{\langle \mathcal{I}, \alpha, \varphi \rangle \mid \mathcal{I}, \alpha \models \varphi\}$ *is:*

- *time: exponential*
- *space:* PSPACE*-complete — see [Var82] for hardness*

Theorem (Data complexity of query evaluation)

The complexity of $\{\langle \mathcal{I}, \alpha \rangle \mid \mathcal{I}, \alpha \models \varphi\}$ *is:*

- *time: polynomial*
- *space:* LOGSPACE

Theorem (Query complexity of query evaluation)

The complexity of $\{\langle \alpha, \varphi \rangle \mid \mathcal{I}, \alpha \models \varphi\}$ *is:*

- *time: exponential*
- *space:* PSPACE*-complete — see [Var82] for hardness*

Example

We consider FOL interpretations exactly as used in relational databases. This requires to drop functions except for constants. Moreover we assume that the interpretation of constants is the identity function, that is constants are interpreted as themselves. This allows us to drop also the interpretation of constants from our interpretations, which now have the form:

$$\mathcal{I} = (\Delta^{\mathcal{I}}, P_1^{\mathcal{I}}, P_2^{\mathcal{I}}, \ldots, P_n^{\mathcal{I}}).$$

Interpretation: $\mathcal{I}$ is as follows (also given in relational notation):

- $\Delta^{\mathcal{I}} = \{john, paul, george, mick, ny, london, 0, 1, \ldots, 100\}$
- $Person^{\mathcal{I}} = \{(john, 30), (paul, 60), (george, 35), (mick, 35)\}$
- $Lives^{\mathcal{I}} = \{(john, ny), (paul, ny), (george, london), (mick, london)\}$
- $Manages^{\mathcal{I}} = \{(paul, john), (george, mick), (paul, mick)\}$

$Person^{\mathcal{I}}$

name	age
john	30
paul	60
george	35
mick	35

$Lives^{\mathcal{I}}$

name	city
john	ny
paul	ny
george	london
mick	london

$Manages^{\mathcal{I}}$

boss	emp. name
paul	john
george	mick
paul	mick

Query: find name and age of persons who live in the same city as their boss.

$$\exists z, w.\mathsf{Person}(x, y) \wedge \mathsf{Manages}(z, x) \wedge \mathsf{Lives}(x, w) \wedge \mathsf{Lives}(z, w)$$

Example - Interpretation

Consider the following interpretation $\mathcal{I}$:

- $\Delta^{\mathcal{I}}$ is equal to the *active domain*: all objects occurring in any predicate extension.
- $Sailors^{\mathcal{I}}$ see table below
- $Boats^{\mathcal{I}}$ see table below
- $Reserves^{\mathcal{I}}$ see table below

$Sailors^{\mathcal{I}}$

sid	sname
22	dustin
31	lubber
58	rusty

$Boats^{\mathcal{I}}$

bid	color
101	red
102	green
103	red
104	blue

$Reserves^{\mathcal{I}}$

sid	bid	day
22	101	10/10/96
58	103	11/12/96

Example - Queries

- Find the names of the sailors who have reserved boat 103.
- Find the names of the sailors who have reserved a red boat.
- Find the colors of the boats reserved by Bob.
- Find the names of the sailors who have reserved at least one boat.
- Find the names of the sailors who have reserved a red and a green boat.
- Find the names of the sailors who have reserved a red or a green boat.
- Find the names of the sailors who have reserved at least two boats.
- Find the names of the sailors who have not reserved a red boat.
- Find the names of the sailors who have reserved all boats.
- Find the names of the sailors who have reserved all red boats.

Example - Queries

- Find the names of the sailors who have reserved boat 103.

Example - Queries

- Find the names of the sailors who have reserved boat 103.

$$\exists x.Sailors(x, y) \wedge \exists w.Reserves(x, 103, z)$$

$$q(y) \leftarrow Sailors(x, y), Reserves(x, 103, z)$$

Example - Queries

- Find the names of the sailors who have reserved boat 103.

$$\exists x.Sailors(x, y) \wedge \exists w.Reserves(x, 103, z)$$

$$q(y) \leftarrow Sailors(x, y), Reserves(x, 103, z)$$

- Find the names of the sailors who have reserved a red boat.

Example - Queries

- Find the names of the sailors who have reserved boat 103.

$$\exists x.Sailors(x, y) \wedge \exists w.Reserves(x, 103, z)$$

$$q(y) \leftarrow Sailors(x, y), Reserves(x, 103, z)$$

- Find the names of the sailors who have reserved a red boat.

$$\exists x.Sailors(x, y) \wedge \exists z, w.Reserves(x, z, w) \wedge Boats(z, red)$$

$$q(y) \leftarrow Sailors(x, y), Reserves(x, z, w), Boats(z, red)$$

Example - Queries

- Find the names of the sailors who have reserved boat 103.

$$\exists x.Sailors(x, y) \wedge \exists w.Reserves(x, 103, z)$$

$$q(y) \leftarrow Sailors(x, y), Reserves(x, 103, z)$$

- Find the names of the sailors who have reserved a red boat.

$$\exists x.Sailors(x, y) \wedge \exists z, w.Reserves(x, z, w) \wedge Boats(z, red)$$

$$q(y) \leftarrow Sailors(x, y), Reserves(x, z, w), Boats(z, red)$$

- Find the colors of the boats reserved by Bob.

Example - Queries

- Find the names of the sailors who have reserved boat 103.

$$\exists x.Sailors(x, y) \wedge \exists w.Reserves(x, 103, z)$$

$$q(y) \leftarrow Sailors(x, y), Reserves(x, 103, z)$$

- Find the names of the sailors who have reserved a red boat.

$$\exists x.Sailors(x, y) \wedge \exists z, w.Reserves(x, z, w) \wedge Boats(z, red)$$

$$q(y) \leftarrow Sailors(x, y), Reserves(x, z, w), Boats(z, red)$$

- Find the colors of the boats reserved by Bob.

$$\exists x.Boats(x, y) \wedge \exists z, w.Reserves(z, y, w) \wedge Sailor(z, Bob)$$

$$q(y) \leftarrow Boats(x, y), Reserves(z, y, w), Sailor(z, Bob)$$

Example - Queries

- Find the names of the sailors who have reserved boat 103.

$$\exists x.Sailors(x, y) \wedge \exists w.Reserves(x, 103, z)$$

$$q(y) \leftarrow Sailors(x, y), Reserves(x, 103, z)$$

- Find the names of the sailors who have reserved a red boat.

$$\exists x.Sailors(x, y) \wedge \exists z, w.Reserves(x, z, w) \wedge Boats(z, red)$$

$$q(y) \leftarrow Sailors(x, y), Reserves(x, z, w), Boats(z, red)$$

- Find the colors of the boats reserved by Bob.

$$\exists x.Boats(x, y) \wedge \exists z, w.Reserves(z, y, w) \wedge Sailor(z, Bob)$$

$$q(y) \leftarrow Boats(x, y), Reserves(z, y, w), Sailor(z, Bob)$$

- Find the names of the sailors who have reserved at least one boat.

Example - Queries

- Find the names of the sailors who have reserved boat 103.

$$\exists x.Sailors(x, y) \wedge \exists w.Reserves(x, 103, z)$$

$$q(y) \leftarrow Sailors(x, y), Reserves(x, 103, z)$$

- Find the names of the sailors who have reserved a red boat.

$$\exists x.Sailors(x, y) \wedge \exists z, w.Reserves(x, z, w) \wedge Boats(z, red)$$

$$q(y) \leftarrow Sailors(x, y), Reserves(x, z, w), Boats(z, red)$$

- Find the colors of the boats reserved by Bob.

$$\exists x.Boats(x, y) \wedge \exists z, w.Reserves(z, y, w) \wedge Sailor(z, Bob)$$

$$q(y) \leftarrow Boats(x, y), Reserves(z, y, w), Sailor(z, Bob)$$

- Find the names of the sailors who have reserved at least one boat.

$$\exists x.Sailors(x, y) \wedge \exists z, w.Reserves(x, z, w)$$

$$q(y) \leftarrow Sailors(x, y), Reserves(x, z, w)$$

Example - Queries

- Find the names of the sailors who have reserved boat 103.

$$\exists x.Sailors(x, y) \wedge \exists w.Reserves(x, 103, z)$$

$$q(y) \leftarrow Sailors(x, y), Reserves(x, 103, z)$$

- Find the names of the sailors who have reserved a red boat.

$$\exists x.Sailors(x, y) \wedge \exists z, w.Reserves(x, z, w) \wedge Boats(z, red)$$

$$q(y) \leftarrow Sailors(x, y), Reserves(x, z, w), Boats(z, red)$$

- Find the colors of the boats reserved by Bob.

$$\exists x.Boats(x, y) \wedge \exists z, w.Reserves(z, y, w) \wedge Sailor(z, Bob)$$

$$q(y) \leftarrow Boats(x, y), Reserves(z, y, w), Sailor(z, Bob)$$

- Find the names of the sailors who have reserved at least one boat.

$$\exists x.Sailors(x, y) \wedge \exists z, w.Reserves(x, z, w)$$

$$q(y) \leftarrow Sailors(x, y), Reserves(x, z, w)$$

- Find the names of the sailors who have reserved a red and a green boat.

Example - Queries

- Find the names of the sailors who have reserved boat 103.

$$\exists x.Sailors(x, y) \wedge \exists w.Reserves(x, 103, z)$$

$$q(y) \leftarrow Sailors(x, y), Reserves(x, 103, z)$$

- Find the names of the sailors who have reserved a red boat.

$$\exists x.Sailors(x, y) \wedge \exists z, w.Reserves(x, z, w) \wedge Boats(z, red)$$

$$q(y) \leftarrow Sailors(x, y), Reserves(x, z, w), Boats(z, red)$$

- Find the colors of the boats reserved by Bob.

$$\exists x.Boats(x, y) \wedge \exists z, w.Reserves(z, y, w) \wedge Sailor(z, Bob)$$

$$q(y) \leftarrow Boats(x, y), Reserves(z, y, w), Sailor(z, Bob)$$

- Find the names of the sailors who have reserved at least one boat.

$$\exists x.Sailors(x, y) \wedge \exists z, w.Reserves(x, z, w)$$

$$q(y) \leftarrow Sailors(x, y), Reserves(x, z, w)$$

- Find the names of the sailors who have reserved a red and a green boat.

$$\exists x.S(x, y) \wedge \exists z, w.R(x, z, w) \wedge B(z, red) \wedge \exists z', w'.R(x, z', w') \wedge B(z', green)$$

Example - Queries

- Find the names of the sailors who have reserved a red or a green boat.

Example - Queries

- Find the names of the sailors who have reserved a red or a green boat.

$$\exists x.Sailors(x, y) \wedge \exists z, w.Reserves(x, z, w) \wedge (Boats(z, red) \vee Boats(z, green))$$

$$q(y) \leftarrow Sailors(x, y), Reserves(x, z, w), Boats(z, red)$$
$$q(y) \leftarrow Sailors(x, y), Reserves(x, z, w), Boats(z, green)$$

Example - Queries

- Find the names of the sailors who have reserved a red or a green boat.

$$\exists x.Sailors(x,y) \wedge \exists z,w.Reserves(x,z,w) \wedge (Boats(z,red) \vee Boats(z,green))$$

$$q(y) \leftarrow Sailors(x,y), Reserves(x,z,w), Boats(z,red)$$
$$q(y) \leftarrow Sailors(x,y), Reserves(x,z,w), Boats(z,green)$$

- Find the names of the sailors who have reserved at least two boats.

Example - Queries

- Find the names of the sailors who have reserved a red or a green boat.

$$\exists x.Sailors(x,y) \wedge \exists z,w.Reserves(x,z,w) \wedge (Boats(z,red) \vee Boats(z,green))$$

$$q(y) \leftarrow Sailors(x,y), Reserves(x,z,w), Boats(z,red)$$
$$q(y) \leftarrow Sailors(x,y), Reserves(x,z,w), Boats(z,green)$$

- Find the names of the sailors who have reserved at least two boats.

$$\exists x.Sailors(x,y) \wedge \exists z,w.Reserves(x,z,w) \wedge \exists z',w'.Reserves(x,z',w') \wedge z \neq z'$$

Example - Queries

- Find the names of the sailors who have reserved a red or a green boat.

$$\exists x.Sailors(x,y) \wedge \exists z,w.Reserves(x,z,w) \wedge (Boats(z,red) \vee Boats(z,green))$$

$$q(y) \leftarrow Sailors(x,y), Reserves(x,z,w), Boats(z,red)$$
$$q(y) \leftarrow Sailors(x,y), Reserves(x,z,w), Boats(z,green)$$

- Find the names of the sailors who have reserved at least two boats.

$$\exists x.Sailors(x,y) \wedge \exists z,w.Reserves(x,z,w) \wedge \exists z',w'.Reserves(x,z',w') \wedge z \neq z'$$

- Find the names of the sailors who have not reserved a red boat.

Example - Queries

- Find the names of the sailors who have reserved a red or a green boat.

$$\exists x.Sailors(x,y) \wedge \exists z,w.Reserves(x,z,w) \wedge (Boats(z,red) \vee Boats(z,green))$$

$$q(y) \leftarrow Sailors(x,y), Reserves(x,z,w), Boats(z,red)$$
$$q(y) \leftarrow Sailors(x,y), Reserves(x,z,w), Boats(z,green)$$

- Find the names of the sailors who have reserved at least two boats.

$$\exists x.Sailors(x,y) \wedge \exists z,w.Reserves(x,z,w) \wedge \exists z',w'.Reserves(x,z',w') \wedge z \neq z'$$

- Find the names of the sailors who have not reserved a red boat.

$$\exists x.Sailors(x,y) \wedge \forall z,w.(Reserves(x,z,w) \rightarrow \neg Boats(z,red))$$

Example - Queries

- Find the names of the sailors who have reserved a red or a green boat.

$$\exists x.Sailors(x,y) \wedge \exists z,w.Reserves(x,z,w) \wedge (Boats(z,red) \vee Boats(z,green))$$

$$q(y) \leftarrow Sailors(x,y), Reserves(x,z,w), Boats(z,red)$$
$$q(y) \leftarrow Sailors(x,y), Reserves(x,z,w), Boats(z,green)$$

- Find the names of the sailors who have reserved at least two boats.

$$\exists x.Sailors(x,y) \wedge \exists z,w.Reserves(x,z,w) \wedge \exists z',w'.Reserves(x,z',w') \wedge z \neq z'$$

- Find the names of the sailors who have not reserved a red boat.

$$\exists x.Sailors(x,y) \wedge \forall z,w.(Reserves(x,z,w) \rightarrow \neg Boats(z,red))$$

- Find the names of the sailors who have reserved all boats.

Example - Queries

- Find the names of the sailors who have reserved a red or a green boat.

$$\exists x.Sailors(x,y) \wedge \exists z,w.Reserves(x,z,w) \wedge (Boats(z,red) \vee Boats(z,green))$$

$$q(y) \leftarrow Sailors(x,y), Reserves(x,z,w), Boats(z,red)$$
$$q(y) \leftarrow Sailors(x,y), Reserves(x,z,w), Boats(z,green)$$

- Find the names of the sailors who have reserved at least two boats.

$$\exists x.Sailors(x,y) \wedge \exists z,w.Reserves(x,z,w) \wedge \exists z',w'.Reserves(x,z',w') \wedge z \neq z'$$

- Find the names of the sailors who have not reserved a red boat.

$$\exists x.Sailors(x,y) \wedge \forall z,w.(Reserves(x,z,w) \rightarrow \neg Boats(z,red))$$

- Find the names of the sailors who have reserved all boats.

$$\exists x.Sailors(x,y) \wedge \forall z,c.(Boats(z,c) \rightarrow \exists w.Reserves(x,z,w))$$

Example - Queries

- Find the names of the sailors who have reserved a red or a green boat.

$$\exists x.Sailors(x,y) \wedge \exists z,w.Reserves(x,z,w) \wedge (Boats(z,red) \vee Boats(z,green))$$

$$q(y) \leftarrow Sailors(x,y), Reserves(x,z,w), Boats(z,red)$$
$$q(y) \leftarrow Sailors(x,y), Reserves(x,z,w), Boats(z,green)$$

- Find the names of the sailors who have reserved at least two boats.

$$\exists x.Sailors(x,y) \wedge \exists z,w.Reserves(x,z,w) \wedge \exists z',w'.Reserves(x,z',w') \wedge z \neq z'$$

- Find the names of the sailors who have not reserved a red boat.

$$\exists x.Sailors(x,y) \wedge \forall z,w.(Reserves(x,z,w) \rightarrow \neg Boats(z,red))$$

- Find the names of the sailors who have reserved all boats.

$$\exists x.Sailors(x,y) \wedge \forall z,c.(Boats(z,c) \rightarrow \exists w.Reserves(x,z,w))$$

- Find the names of the sailors who have reserved all red boats.

Example - Queries

- Find the names of the sailors who have reserved a red or a green boat.

$$\exists x.Sailors(x,y) \wedge \exists z,w.Reserves(x,z,w) \wedge (Boats(z,red) \vee Boats(z,green))$$

$$q(y) \leftarrow Sailors(x,y), Reserves(x,z,w), Boats(z,red)$$
$$q(y) \leftarrow Sailors(x,y), Reserves(x,z,w), Boats(z,green)$$

- Find the names of the sailors who have reserved at least two boats.

$$\exists x.Sailors(x,y) \wedge \exists z,w.Reserves(x,z,w) \wedge \exists z',w'.Reserves(x,z',w') \wedge z \neq z'$$

- Find the names of the sailors who have not reserved a red boat.

$$\exists x.Sailors(x,y) \wedge \forall z,w.(Reserves(x,z,w) \rightarrow \neg Boats(z,red))$$

- Find the names of the sailors who have reserved all boats.

$$\exists x.Sailors(x,y) \wedge \forall z,c.(Boats(z,c) \rightarrow \exists w.Reserves(x,z,w))$$

- Find the names of the sailors who have reserved all red boats.

$$\exists x.Sailors(x,y) \wedge \forall z.(Boats(z,red) \rightarrow \exists w.Reserves(x,z,w))$$

References

[Var82] M. Y. Vardi.
The complexity of relational query languages.
In *Proc. of the 14th ACM SIGACT Symp. on Theory of Computing (STOC'82)*, pages 137–146, 1982.