What is a Query?

The relational queries considered so far produced a result table from a database. We generalize slightly.

**Definition 2.1:**
- Syntax: a query expression $q$ is a word from a query language (algebra expression, logical expression, etc.)
- Semantics: a query mapping $M[q]$ is a function that maps a database instance $I$ to a database instance $M[q](I)$

$\leadsto$ a “result table” is a result database instance with one table.

$\leadsto$ for some semantics, query mappings are not defined on all database instances

---

Generic Queries

We only consider queries that do not depend on the concrete names given to constants in the database:

**Definition 2.2:** A query $q$ is generic if, for every bijective renaming function $\mu : \text{dom} \rightarrow \text{dom}$ and database instance $I$:

$$\mu(M[q](I)) = M[\mu(q)](\mu(I)).$$

In this case, $M[q]$ is closed under isomorphisms.

---

Review: Example from Previous Lecture

**Lines:**

<table>
<thead>
<tr>
<th>Line</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>85</td>
<td>bus</td>
</tr>
<tr>
<td>3</td>
<td>tram</td>
</tr>
<tr>
<td>F1</td>
<td>ferry</td>
</tr>
</tbody>
</table>

**Stops:**

<table>
<thead>
<tr>
<th>SID</th>
<th>Stop</th>
<th>Accessible</th>
</tr>
</thead>
<tbody>
<tr>
<td>17</td>
<td>Hauptbahnhof</td>
<td>true</td>
</tr>
<tr>
<td>42</td>
<td>Helmholtzstr.</td>
<td>true</td>
</tr>
<tr>
<td>57</td>
<td>Stadtgutstr.</td>
<td>true</td>
</tr>
<tr>
<td>123</td>
<td>Gustav-Freytag-Str.</td>
<td>false</td>
</tr>
</tbody>
</table>

**Connect:**

<table>
<thead>
<tr>
<th>From</th>
<th>To</th>
<th>Line</th>
</tr>
</thead>
<tbody>
<tr>
<td>57</td>
<td>42</td>
<td>85</td>
</tr>
<tr>
<td>17</td>
<td>789</td>
<td>3</td>
</tr>
</tbody>
</table>

Every table has a schema:

- **Lines**[Line:string, Type:string]
- **Stops**[SID:int, Stop:string, Accessible:bool]
- **Connect**[From:int, To:int, Line:string]
First-order Logic as a Query Language

Idea: database instances are finite first-order interpretations

\( \sim \) use first-order formulae as query language

\( \sim \) use unnamed perspective (more natural here)

Examples (using schema as in previous lecture):

- Find all bus lines: Lines(x, "bus")
- Find all possible types of lines: \( \exists y. \) Lines(y, x)
- Find all lines that depart from an accessible stop:
  \[ \exists y. \text{Stop}(y, x) \land \text{Connect}(y, x, "true") \land \text{Connect}(y, x, y) \]

First-order Logic Syntax: Simplifications

We use the usual shortcuts and simplifications:

- flat conjunctions (\( \varphi_1 \land \varphi_2 \land \varphi_3 \) instead of \( \varphi_1 \land (\varphi_2 \land \varphi_3) \))
- flat disjunctions (similar)
- flat quantifiers (\( \exists x, y, z. \varphi \) instead of \( \exists x. \exists y. \exists z. \varphi \))

- \( \varphi \rightarrow \psi \) as shortcut for \( \neg \varphi \lor \psi \)
- \( \varphi \leftrightarrow \psi \) as shortcut for \( (\varphi \rightarrow \psi) \land (\psi \rightarrow \varphi) \)
- \( \neg t_1 \equiv t_2 \) as shortcut for \( \neg (t_1 \equiv t_2) \)

But we always use parentheses to clarify nesting of \( \land \) and \( \lor \):

No "\( \varphi_1 \land \varphi_2 \lor \varphi_3 \)"!
### First-order Logic Queries

**Definition 2.3:** An \( n \)-ary first-order query \( q \) is an expression \( \varphi[x_1, \ldots, x_n] \) where \( x_1, \ldots, x_n \) are exactly the free variables of \( \varphi \) (in a specific order).

**Definition 2.4:** An answer to \( q = \varphi[x_1, \ldots, x_n] \) over an interpretation \( I \) is a tuple \( \langle a_1, \ldots, a_n \rangle \) of constants such that

\[
I \models \varphi[x_1/a_1, \ldots, x_n/a_n]
\]

where \( \varphi[x_1/a_1, \ldots, x_n/a_n] \) is \( \varphi \) with each free \( x_i \) replaced by \( a_i \).

The result of \( q \) over \( I \) is the set of all answers of \( q \) over \( I \).

---

### Boolean Queries

A Boolean query is a query of arity 0

\(~ \varphi \) we simply write \( \varphi \) instead of \( \varphi[] \)

\( \varphi \) is a closed formula (a.k.a. sentence)

What does a Boolean query return?

Two possible cases:

- \( I \models \varphi \), then the result of \( \varphi \) over \( I \) is \( \{ \langle \rangle \} \) (the unit table)
- \( I \not\models \varphi \), then the result of \( \varphi \) over \( I \) is \( \emptyset \) (the empty table)

Interpreted as Boolean check with result true or false (match or no match)

---

### Domain Dependence

We have defined FO queries over interpretations

\(~ \) How exactly do we get from databases to interpretations?

- Constants are just interpreted as themselves: \( a^I = a \)
- Predicates are interpreted according to the table contents
- But what is the domain of the interpretation?

What should the following queries return?

1. \( \neg \text{Lines}(x, "bus")[x] \)
2. \( (\text{Connect}(x_1, "42", "85") \lor \text{Connect}("57", x_2, "85"))[x_1, x_2] \)
3. \( \forall y. p(x, y)[x] \)

\(~ \) Answers depend on the interpretation domain, not just on the database contents

---

### Natural Domain

First possible solution: the natural domain

**Natural domain semantics (ND):**

- fix the interpretation domain to \( \text{dom} \) (infinite)
- query answers might be infinite (not a valid result table)
- \(~ \) query result undefined for such databases

Markus Krötzsch, 10th Apr 2018 Database Theory
Active Domain: Examples

Query answers under active domain semantics:
(1) \( \neg \text{Lines}(x, "bus")[x] \)
   Undefined on all databases
(2) \( (\text{Connect}(x_1, "42", "85") \lor \text{Connect}("57", x_2, "85")[x_1, x_2] \)
   Undefined on databases with matching \( x_1 \) or \( x_2 \) in Connect, otherwise empty
(3) \( \forall y. p(x, y)[x] \)
   Empty on all databases

Active Domain

Alternative: restrict to constants that are really used
\( \sim \) active domain
- for a database instance \( I \), \( \text{dom}(I) \) is the set of constants used in relations of \( I \)
- for a query \( q \), \( \text{dom}(q) \) is the set of constants in \( q \)
- \( \text{dom}(I, q) = \text{dom}(I) \cup \text{dom}(q) \)

Active domain semantics (AD):
consider database instance as interpretation over \( \text{dom}(I, q) \)

Domain Independence

Observation: some queries do not depend on the domain
- \( \text{Stops}(x, y, "true")[x, y] \)
- \( (x \approx q)[x] \)
- \( p(x) \land \neg q(x)[x] \)
- \( \forall y.(q(x, y) \rightarrow p(x, y))[x] \) (exercise: why?)

In contrast, all example queries on the previous few slides are not domain independent

Domain independent semantics (DI):
consider only domain independent queries
use any domain \( \text{dom}(I, q) \subseteq \Delta \subseteq \text{dom} \) for interpretation

Natural Domain: Examples

Query answers under natural domain semantics:
(1) \( \neg \text{Lines}(x, "bus")[x] \)
   Undefined on all databases
(3) \( \forall y. p(x, y)[x] \)
   Empty on all databases

Domain Independence

Observation: some queries do not depend on the domain
- \( \text{Stops}(x, y, "true")[x, y] \)
- \( (x \approx q)[x] \)
- \( p(x) \land \neg q(x)[x] \)
- \( \forall y.(q(x, y) \rightarrow p(x, y))[x] \) (exercise: why?)

In contrast, all example queries on the previous few slides are not domain independent

Domain independent semantics (DI):
consider only domain independent queries
use any domain \( \text{dom}(I, q) \subseteq \Delta \subseteq \text{dom} \) for interpretation
How to Compare Query Languages

We have seen three ways of defining FO query semantics
~ how to compare them?

**Definition 2.5:** The set of query mappings that can be described in a query language \( L \) is denoted \( QM(L) \).

- \( L_1 \) is subsumed by \( L_2 \), written \( L_1 \sqsubseteq L_2 \), if \( QM(L_1) \subseteq QM(L_2) \)
- \( L_1 \) is equivalent to \( L_2 \), written \( L_1 \equiv L_2 \), if \( QM(L_1) = QM(L_2) \)

We will also compare query languages under named perspective with query languages under unnamed perspective.
This is possible since there is an easy one-to-one correspondence between query mappings of either kind (see exercise).

Equivalence of Relational Query Languages

Theorem 2.6: The following query languages are equivalent:
- Relational algebra \( RA \)
- FO queries under active domain semantics \( AD \)
- Domain independent FO queries \( DI \)

This holds under named and under unnamed perspective.

To prove it, we will show:

\[ RA_{\text{named}} \sqsubseteq DI_{\text{unnamed}} \sqsubseteq AD_{\text{unnamed}} \sqsubseteq RA_{\text{named}} \]

**RA_{\text{named}} \sqsubseteq DI_{\text{unnamed}} (cont’d)**

Remaining cases:
- if \( q = \pi_{a_1, \ldots, a_n}(q') \) for a subquery \( q'[b_1, \ldots, b_m] \) with \( \{b_1, \ldots, b_m\} = \{a_1, \ldots, a_n\} \cup \{c_1, \ldots, c_k\} \), then \( \varphi_q = \exists x_1, \ldots, x_n \, \varphi' \)
- if \( q = q_1 \land q_2 \) then \( \varphi_q = \varphi_{q_1} \land \varphi_{q_2} \)
- if \( q = q_1 \lor q_2 \) then \( \varphi_q = \varphi_{q_1} \lor \varphi_{q_2} \)
- if \( q = q_1 - q_2 \) then \( \varphi_q = \varphi_{q_1} \land \neg \varphi_{q_2} \)

One can show that \( \varphi_q[a_1, \ldots, a_n] \) is domain independent and equivalent to \( q \)

~ exercise
**DI unnamed ⊑ AD unnamed**

This is easy to see:
- Consider an FO query $q$ that is domain independent
- The semantics of $q$ is the same for any domain $\text{dom} \subseteq I \subseteq \Delta I \subseteq \text{dom}$
- In particular, the semantics of $q$ is the same under active domain semantics
- Hence, for every DI query, there is an equivalent AD query

**AD unnamed ⊑ RA named**

Consider an AD query $q = \varphi[x_1, \ldots, x_n]$.

For an arbitrary attribute name $a$, we can construct an RA expression $E_{\text{adom}}$ such that

$$E_{\text{adom}}(I) = \{ [a \mapsto c] \mid c \in \text{adom}(I, q) \}$$

For every variable $x$, we use a distinct attribute name $a_x$.

- if $\varphi = R(t_1, \ldots, t_m)$ with signature $R[a_1, \ldots, a_m]$ with variables $x_1 = t_{i_1}, \ldots, x_n = t_{i_n}$ and constants $c_1 = t_{k_1}, \ldots, c_k = t_{k_2}$,
  then $E_{\varphi} = \delta_{a_1 \ldots a_n = a_{i_1} \ldots a_{i_n}}(\sigma_{a_{i_1} = c_1}(\ldots \sigma_{a_{i_k} = c_k}(R)\ldots))$
- if $\varphi = (x \approx c)$, then $E_{\varphi} = [\{a_x \mapsto c\}]$
- if $\varphi = (x \approx y)$, then $E_{\varphi} = \sigma_{a_x = a_y}(E_{\varphi_{a_x, \text{adom}}} \land E_{\varphi_{a_y, \text{adom}}})$
- other forms of equality atoms are similar

**How to find DI queries?**

Domain independent queries are arguably most intuitive, since their result does not depend on special assumptions.

$$\sim \text{How can we check if a query is in DI? } \text{Unfortunately, we can’t:}$$

**Theorem 2.7:** Given a FO query $q$, it is undecidable if $q \in \text{DI}$.

$$\sim \text{find decidable sufficient conditions for a query to be in DI}$$

_A note on order:_ The translation yields an expression $E_{\varphi}[a_{i_1}, \ldots, a_{i_n}]$. For this to be equivalent to the query $\varphi[x_1, \ldots, x_n]$, we must choose the attribute names such that their global order is $a_{i_1}, \ldots, a_{i_n}$. This is clearly possible, since the names are arbitrary and we have infinitely many names available.
Definition 2.8: An FO query $q = \varphi[x_1, \ldots, x_n]$ is a safe-range query if

$$rr(SRNF(\varphi)) = \{x_1, \ldots, x_n\}.$$
Summary and Outlook

First-order logic gives rise to a relational query language

The problem of domain dependence can be solved in several ways

All common definitions lead to equivalent calculi

\[\leadsto \text{"relational calculus"}\]

Open questions:
- How hard is it to actually answer such queries? (next lecture)
- How can we study the expressiveness of query languages?
- Are there interesting query languages that are not equivalent to RA?