Fundamentos lógicos de bases de datos
(Logical foundations of databases)

Diego Figueira  Gabriele Puppis

CNRS  LaBRI
About the speakers…

Gabriele Puppis
PhD from Udine (Italy)
post-docs in Oxford
Works in LaBRI, Bordeaux
CNRS researcher

Diego Figueira
PhD from ENS Cachan (France),
post-docs in Warsaw, Edinburgh
Works in LaBRI, Bordeaux
CNRS researcher
Alma Mater: UBA !
About the speakers...

Gabriele Puppis
PhD from Udine (Italy)
post-docs in Oxford
Works in LaBRI, Bordeaux
CNRS researcher

Diego Figueira
PhD from ENS Cachan (France),
post-docs in Warsaw, Edinburgh
Works in LaBRI, Bordeaux
CNRS researcher
Alma Mater: UBA!
Gabriele Puppis
PhD from Udine (Italy)
post-docs in Oxford
Works in LaBRI, Bordeaux
CNRS researcher

Diego Figueira
PhD from ENS Cachan (France),
post-docs in Warsaw, Edinburgh
Works in LaBRI, Bordeaux
CNRS researcher
Alma Mater: UBA!
First and foremost…

interrupt!

ask! (in any language)
Format: 45 min slots / 20 min breaks: 45’ + 20’ + 45’ + 20’ + 45’

Schedule:
- Relational Algebra
- First-Order logic
- EF games, 0-1 law, Locality
- Conjunctive Queries

Evaluation: A short test on Saturday

Slides available at

http://www.labri.fr/perso/dfigueir/ECI15/
Databases

\[
database = \text{a collection of data, structured in some way} + \text{a way of defining, querying, updating the data inside}
\]

* [Abitebou, Hull, Vianu “Foundations of databases”]
Databases

**database** = a collection of data, structured in some way + a way of defining, querying, updating the data inside

**Data model**
- how the data is *logically organized*
- mathematical abstraction for representing data
- independent from physical organisation

DBMS also implement: transactions, concurrency, access control, resiliency…

* [Abitebou, Hull, Vianu “Foundations of databases”]
Relational data model = data logically organised into relations ("tables").

What's a relation?

- a (finite) subset of the cartesian product of sets
- a "table" with rows and columns
Relational databases

**Relational** data model = data logically organised into relations (“tables”).

What’s a **relation**?
- a (finite) subset of the cartesian product of sets
- a “table” with rows and columns

like:

\[
\{ (1,a,2), (2,b,6), (2,a,1) \} \subseteq \mathbb{N} \times \{a,b\} \times \mathbb{N}
\]
Relational data model = data logically organised into relations (“tables”).

What’s a relation?
- a (finite) subset of the cartesian product of sets
- a “table” with rows and columns

like:

\[ \{ (1,a,2), (2,b,6), (2,a,1) \} \subseteq \mathbb{N} \times \{a,b\} \times \mathbb{N} \]

a “tuple” (a “3-tuple”)
Relational data model = data logically organised into relations ("tables").

What's a relation?

- a (finite) subset of the cartesian product of sets
- a "table" with rows and columns

like:

\[ \{ (1,a,2), (2,b,6), (2,a,1) \} \subseteq \mathbb{N} \times \{a,b\} \times \mathbb{N} \]

- a "tuple" (a "3-tuple")
- () \rightarrow 0-tuple
Relational data model = data logically organised into relations ("tables").

What's a relation?

- a (finite) subset of the cartesian product of sets
- a "table" with rows and columns

like:

\[
\{ (1, a, 2), (2, b, 6), (2, a, 1) \} \subseteq \mathbb{N} \times \{a, b\} \times \mathbb{N}
\]

a "tuple" (a "3-tuple")

() \rightarrow 0-tuple

like:

\[
\begin{array}{ccc}
1 & a & 2 \\
2 & b & 6 \\
2 & a & 1
\end{array}
\]
Relational databases

**Relational** data model = data logically organised into relations ("tables").

What’s a **relation**?

- a (finite) subset of the cartesian product of sets
- a “table” with rows and columns

DB = A **schema**: names of tables and attributes

An **instance**: data conforming to the schema
Relational data model = data logically organised into relations ("tables").

What’s a relation?

- a (finite) subset of the cartesian product of sets
- a “table” with rows and columns

DB = A schema: names of tables and attributes

Films (Title:string, Director:string, Actor:string)
Schedule (Theatre:string, Title:string)

An instance: data conforming to the schema
Relational databases

**Relational** data model = data logically organised into relations (“tables”).

What’s a relation?

- a (finite) subset of the cartesian product of sets
- a “table” with rows and columns

DB = A **schema**: names of tables and attributes

<table>
<thead>
<tr>
<th>Films (Title:string, Director:string, Actor:string)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Schedule (Theatre:string, Title:string)</td>
</tr>
</tbody>
</table>

An **instance**: data conforming to the schema

<table>
<thead>
<tr>
<th>Films</th>
<th>Schedule</th>
</tr>
</thead>
<tbody>
<tr>
<td>Title</td>
<td>Director</td>
</tr>
<tr>
<td>-------</td>
<td>----------</td>
</tr>
<tr>
<td>8 1/2</td>
<td>Fellini</td>
</tr>
<tr>
<td>Shining</td>
<td>Kubrick</td>
</tr>
<tr>
<td>Dr. Strangelove</td>
<td>Kubrick</td>
</tr>
<tr>
<td>8 femmes</td>
<td>Ozon</td>
</tr>
</tbody>
</table>
Relational databases

**Relational** data model = data logically organised into relations (“tables”).

⚠️ We assume all elements come from a fixed set of *constants* or *data values* $U$. 
What is a query $q$?

A mapping that takes a database instance $D$ returns a relation $q(D) \subseteq U^r$ of fixed arity $r$. 

Relational databases: queries
What is a query $q$?

A mapping that takes a database instance $D$ returns a relation $q(D) \subseteq U^r$ of fixed arity $r$.
What is a query $q$?

A mapping that takes a database instance $D$ returns a relation $q(D) \subseteq U^r$ of fixed arity $r$.

computable!

generic! (order independent)
Relational databases: queries

What is a query $q$?

A mapping that takes a database instance $D$ returns a relation $q(D) \subseteq U^r$ of fixed arity $r$.

- computable!
- generic! (order independent)
- Boolean query: $r=0$
  Either “yes” {} or “no” {}
Relational databases: queries

What is a query $q$?

A mapping that takes a database instance $D$ returns a relation $q(D) \subseteq U^r$ of fixed arity $r$.

What do we care about queries?

expressive power    evaluation    static analysis
The fundamental questions:

How to query the relational data model?
How efficient/expressive is it?
The fundamental questions:

How to query the relational data model?

How efficient/expressive is it?
Query languages

Query Language

Syntax

Expressions for querying the db, governed by syntactic rules

“Select X from Y”

“y :- ∀x (x ≤ y)”

Semantics

Interpretation of symbols in terms of some structure

Retrieves all strings in column X of table Y

Returns the maximum element of the set.
Relational Algebra (RA)  [Codd, 1970]

Syntax: \[ E := R, S, \ldots \mid E \cup E \mid E \setminus E \mid E \times E \mid \pi_M(E) \mid \sigma_\Theta(E) \]

where \( M \subseteq \mathbb{N} \)
\( \Theta \subseteq \mathbb{N} \times \{=,\neq\} \times \mathbb{N} \)
Relational Algebra (RA)

Syntax: \( E := R, S, \ldots \mid E \cup E \mid E \setminus E \mid E \times E \mid \pi_M(E) \mid \sigma_\Theta(E) \)

where \( M \subseteq \mathbb{N} \)
\( \Theta \subseteq \mathbb{N} \times \{=,\neq\} \times \mathbb{N} \)

- \( R_1 \cup R_2 \): Set union
- \( R_1 \times R_2 \): Cartesian product
- \( R_1 \setminus R_2 \): Set difference
Relational Algebra (RA)

Syntax: \[ E := R, S, \ldots \mid E \cup E \mid E \setminus E \mid E \times E \mid \pi_M(E) \mid \sigma_\Theta(E) \]

where \( M \subseteq \mathbb{N} \)
\( \Theta \subseteq \mathbb{N} \times \{=, \neq \} \times \mathbb{N} \)

- \( R_1 \cup R_2 \): Set union
- \( R_1 \times R_2 \): Cartesian product
- \( R_1 \setminus R_2 \): Set difference
- \( \sigma_{\{i_1=j_1, \ldots, i_n=j_n\}}(R) := \{(x_1, \ldots, x_m) \in R \mid (x_{i_1}=x_{j_1}) \land \ldots \land (x_{i_n}=x_{j_n})\} \): Selection

[Codd, 1970]
Relational Algebra (RA)

Syntax: \[ E := R, S, ... \mid E \cup E \mid E \setminus E \mid E \times E \mid \pi_M(E) \mid \sigma_\Theta(E) \]

where \( M \subseteq \mathbb{N} \)
\( \Theta \subseteq \mathbb{N} \times \{=, \neq\} \times \mathbb{N} \)

- \( R_1 \cup R_2 \): Set union
- \( R_1 \times R_2 \): Cartesian product
- \( R_1 \setminus R_2 \): Set difference

\[ \sigma_{\{i_1=j_1, \ldots, i_n=j_n\}}(R) := \{(x_1, \ldots, x_m) \in R \mid (x_{i_1} \neq x_{j_1}) \land \cdots \land (x_{i_n} \neq x_{j_n})\} : \text{Selection} \]
Relational Algebra (RA)

Syntax: \[ E := R, S, \ldots \mid E \cup E \mid E \setminus E \mid E \times E \mid \pi_M(E) \mid \sigma_\Theta(E) \]

where \( M \subseteq \mathbb{N} \)
\( \Theta \subseteq \mathbb{N} \times \{\ =, \neq \} \times \mathbb{N} \)

- **R_1 \cup R_2** : Set union
- **R_1 \times R_2** : Cartesian product
- **R_1 \setminus R_2** : Set difference
- \[ \sigma_{\{i_1=j_1, \ldots, i_n=j_n\}}(R) := \{(x_1, \ldots, x_m) \in R \mid (x_{i_1} \neq x_{j_1}) \land \cdots \land (x_{i_n} \neq x_{j_n})\} : \text{Selection} \]
Relational Algebra (RA)

Syntax:  \[ E := R, S, ... \mid E \cup E \mid E \setminus E \mid E \times E \mid \pi_M(E) \mid \sigma_\Theta(E) \]

where \( M \subseteq \mathbb{N} \)
\[ \Theta \subseteq \mathbb{N} \times \{=, \neq\} \times \mathbb{N} \]

- \( R_1 \cup R_2 \): Set union
- \( R_1 \times R_2 \): Cartesian product
- \( R_1 \setminus R_2 \): Set difference

\[ \sigma_{\{i_1=j_1, \ldots, i_n=j_n\}}(R) := \{(x_1, \ldots, x_m) \in R \mid (x_{i_1} \neq x_{j_1}) \land \cdots \land (x_{i_n} \neq x_{j_n})\} : \text{Selection} \]

\[ \pi_{\{i_1, \ldots, i_n\}}(R) := \{(x_{i_1}, \ldots, x_{i_n}) \mid (x_1, \ldots, x_m) \in R\} : \text{Projection} \]
Relational Algebra (RA)

Syntax: \( E := R, S, \ldots | E \cup E | E \setminus E | E \times E | \pi_M(E) | \sigma_\Theta(E) \)

where \( M \subseteq \mathbb{N} \)
\( \Theta \subseteq \mathbb{N} \times \{=, \neq\} \times \mathbb{N} \)

- **\( R_1 \cup R_2 \):** Set union
  \[ \sigma_{\{1=3,1\neq2\}}(\{(1,2,1), (2,2,2)\}) = \{(1,2,1)\} \]
  \[ \pi_{\{1,3\}}(\{(1,2,1),(2,2,2)\}) = \{(1,1), (2,2)\} \]

- **\( R_1 \times R_2 \):** Cartesian product

- **\( R_1 \setminus R_2 \):** Set difference

- **\( \sigma_{\{i_1=j_1,\ldots,i_n=j_n\}}(R) := \{(x_1, \ldots, x_m) \in R | (x_{i_1} \neq x_{j_1}) \land \cdots \land (x_{i_n} \neq x_{j_n})\} : Selection \)**

- **\( \pi_{\{i_1,\ldots,i_n\}}(R) := \{(x_{i_1},\ldots,x_{i_n}) | (x_1, \ldots, x_m) \in R\} : Projection \)**
Relational Algebra (RA)

- $R_1 \cup R_2$ : Set union
- $R_1 \times R_2$ : Cartesian product
- $R_1 \setminus R_2$ : Set difference

$\sigma_{i_1 \neq j_1, \ldots, i_n \neq j_n}(R) := \{(x_1, \ldots, x_m) \in R \mid (x_{i_1} \neq x_{j_1}) \land \cdots \land (x_{i_n} \neq x_{j_n})\}$ : Selection

$\pi_{\{i_1, \ldots, i_n\}}(R) := \{(x_{i_1}, \ldots, x_{i_n}) \mid (x_1, \ldots, x_m) \in R\}$ : Projection

**Question 1:** What is the RA expression for

\[
\{ (v_1, v_2) \mid \text{there are } w_1 \neq w_2 \text{ so that } (v_1, w_1) \in R_1 \text{ and } (v_2, w_2) \in R_2 \}\ ?
\]

**Question 2:** $\pi_2(\sigma_{1=3}(\pi_2(\sigma_{1=3}(R_1 \times R_2)) \times R_2)) = ?$

<table>
<thead>
<tr>
<th>$R_1$</th>
<th>$R_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>a</td>
</tr>
<tr>
<td>b</td>
<td>b</td>
</tr>
<tr>
<td>c</td>
<td>c</td>
</tr>
<tr>
<td>b</td>
<td>b</td>
</tr>
<tr>
<td>a</td>
<td>a</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$R_1$</th>
<th>$R_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
</tr>
</tbody>
</table>
Relational Algebra (RA)

• \( R_1 \cup R_2 \) : Set union
• \( R_1 \times R_2 \) : Cartesian product
• \( R_1 \setminus R_2 \) : Set difference
• \( \sigma_{\{i_1 \neq j_1, \ldots, i_n \neq j_n\}}(R) := \{(x_1, \ldots, x_m) \in R \mid (x_{i_1} \neq x_{j_1}) \land \cdots \land (x_{i_n} \neq x_{j_n})\} : \) Selection
• \( \pi_{\{i_1, \ldots, i_n\}}(R) := \{(x_{i_1}, \ldots, x_{i_n}) \mid (x_1, \ldots, x_m) \in R\} : \) Projection

**Question 1:** What is the RA expression for
\( \{ (v_1, v_2) \mid \) there are \( w_1 \neq w_2 \) so that \( (v_1, w_1) \in R_1 \) and \( (v_2, w_2) \in R_2 \} \) ?

**Answer:** \( \pi_{\{1, 3\}}(\sigma_{1 \neq 3}(R_1 \times R_2)) \)

**Question 2:** \( \pi_2(\sigma_{1=3}(\pi_2(\sigma_{1=3}(R_1 \times R_2)) \times R_2)) = \) ?
Relational Algebra (RA)

- $R_1 \cup R_2$: Set union
- $R_1 \times R_2$: Cartesian product
- $R_1 \setminus R_2$: Set difference

$\sigma_{i_1\neq j_1, \ldots, i_n\neq j_n}(R) := \{(x_1, \ldots, x_m) \in R \mid (x_{i_1} \neq x_{j_1}) \land \cdots \land (x_{i_n} \neq x_{j_n})\}$: Selection

$\pi_{i_1, \ldots, i_n}(R) := \{(x_{i_1}, \ldots, x_{i_n}) \mid (x_1, \ldots, x_m) \in R\}$: Projection

**Question 1:** What is the RA expression for

\[ \{ (v_1, v_2) \mid \text{there are } w_1 \neq w_2 \text{ so that } (v_1, w_1) \in R_1 \text{ and } (v_2, w_2) \in R_2 \} \]?

**Answer:** $\pi_{1,3}(\sigma_{1\neq3}(R_1 \times R_2))$

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>b</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td></td>
<td>b</td>
</tr>
<tr>
<td>b</td>
<td>a</td>
<td></td>
</tr>
<tr>
<td>c</td>
<td>a</td>
<td>b</td>
</tr>
</tbody>
</table>

**Question 2:** $\pi_2(\sigma_{1=3}(\pi_2(\sigma_{1=3}(R_1 \times R_2)) \times R_2)) = ?$

**Answer (only one element):** b
RA = Basic SQL

Select $X$
From $R_1, \ldots, R_n$
Where $Z$

\[ \iff \pi_X(\sigma_Z( R_1 \times \cdots \times R_n )) \]

... or ...
\[ \iff \text{union} \]

... not in (...
\[ \iff \text{difference} \]
RA = Basic SQL

Select X
From R₁, ..., Rₙ  ⇔  \( \pi_X(\sigma_Z( R_1 \times \cdots \times R_n ) ) \)

... or ...
... not in (…)

\( \pi_2(\sigma_{1 \neq 3}(R_1 \times R_2)) \)  \( \Rightarrow \)

\begin{tabular}{|c|c|}
\hline
R₁ & R₂ \\
\hline
a & 3 \\
\hline
b & 2 \\
\hline
c & 4 \\
\hline
b & 3 \\
\hline
a & 2 \\
\hline
\end{tabular}
RA = Basic SQL

Select $X$
From $R_1, \ldots, R_n$  $\iff \pi_X (\sigma_Z (R_1 \times \cdots \times R_n))$

... or ...
$\iff$ union
... not in (...)  $\iff$ difference

\[ \pi_2 (\sigma_{1 \neq 3} (R_1 \times R_2)) \sim \]
Select $R_1.2$ as foo
From $R_1$, $R_2$
Where $R_1.1 \neq R_2.1$

<table>
<thead>
<tr>
<th>$R_1$</th>
<th></th>
<th>$R_2$</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>3</td>
<td>a</td>
<td>4</td>
</tr>
<tr>
<td>b</td>
<td>2</td>
<td>b</td>
<td>1</td>
</tr>
<tr>
<td>c</td>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>b</td>
<td>3</td>
<td>a</td>
<td>1</td>
</tr>
<tr>
<td>a</td>
<td>2</td>
<td>b</td>
<td>3</td>
</tr>
</tbody>
</table>
RA = Basic SQL

Select $X$
From $R_1,\ldots, R_n$  \iff  $\pi_X(\sigma_Z(R_1 \times \cdots \times R_n))$
Where $Z$

... or ...
\iff union
... not in (...) \iff difference

\[ \pi_2(\sigma_1 \neq 3(R_1 \times R_2)) \sim \]
\[ \pi_2(\sigma_1 = 3(\neq \times R_2)) \sim \]

Select $R_1.2$ as $\text{foo}$
From $R_1$, $R_2$
Where $R_1.1 \neq R_2.1$

<table>
<thead>
<tr>
<th>$R_1$</th>
<th>$R_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>a</td>
</tr>
<tr>
<td>b</td>
<td>b</td>
</tr>
<tr>
<td>c</td>
<td>c</td>
</tr>
<tr>
<td>b</td>
<td>b</td>
</tr>
<tr>
<td>a</td>
<td>a</td>
</tr>
<tr>
<td>b</td>
<td>b</td>
</tr>
<tr>
<td>a</td>
<td>a</td>
</tr>
<tr>
<td>b</td>
<td>b</td>
</tr>
</tbody>
</table>

no domain-specific features, aggregation, etc
RA = Basic SQL

no domain-specific features, aggregation, etc

Select $X$
From $R_1, \ldots, R_n$  $\iff$  $\pi_X(\sigma_Z( R_1 \times \cdots \times R_n ))$
Where $Z$

... or ...  $\iff$  union
... not in (...)  $\iff$  difference

\[ \pi_2(\sigma_{1\neq 3}(R_1 \times R_2)) \sim \]

Select $R_1.2$ as $\text{foo}$
From $R_1$, $R_2$
Where $R_1.1 \neq R_2.1$

\[ \pi_2(\sigma_{1=3}( \bullet \times R_2)) \sim \]

<table>
<thead>
<tr>
<th>$R_1$</th>
<th>$R_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>a</td>
</tr>
<tr>
<td>b</td>
<td>4</td>
</tr>
<tr>
<td>c</td>
<td>2</td>
</tr>
<tr>
<td>b</td>
<td>1</td>
</tr>
<tr>
<td>a</td>
<td>1</td>
</tr>
<tr>
<td>a</td>
<td>2</td>
</tr>
<tr>
<td>b</td>
<td>3</td>
</tr>
</tbody>
</table>
RA = Basic SQL

no domain-specific features, aggregation, etc

Select $X$
From $R_1, \ldots, R_n$
$\iff \pi_X ( \sigma_Z ( R_1 \times \ldots \times R_n ) )$

... or ...
$\iff$ union

... not in (...) $\iff$ difference

$\Pi_2 ( \sigma_{1 \neq 3} ( R_1 \times R_2 ) ) \leadsto \star$ Select $R_1.2$ as $\text{foo}$
From $R_1$, $R_2$
Where $R_1.1 \neq R_2.1$

$\Pi_2 ( \sigma_{1=3} ( \star \times R_2 ) ) \leadsto$
Select $\text{foo}$
From $\star$, $R_2$
Where $\text{foo} = R_2.2$

\begin{array}{|c|c|}
\hline
R_1 & R_2 \\
\hline
a & 3 \\
b & 2 \\
c & 4 \\
b & 3 \\
a & 2 \\
\hline
\end{array}
Denotational languages

Algebra $\sim$ *How* to obtain the result

Logics $\sim$ *What* is the property of the result
Denotational languages

Algebra $\sim$ *How* to obtain the result

Logics $\sim$ *What* is the property of the result

**Relational Algebra**
operations on tables

**Procedural**

**Declarative**
Denotational languages

Algebra ~ How to obtain the result

Relational Algebra
operations on tables

Logics ~ What is the property of the result

First Order logic
properties on mathematical structures

Procedural

Declarative
Denotational languages

Algebra $\sim$ How to obtain the result

Logics $\sim$ What is the property of the result

Relational Algebra
operations on tables

First Order logic
properties on mathematical structures

Procedural

Declarative
FO = First-Order logic
A **structure** is:

\[ A = (D, R_1, \ldots, R_n, f_1, \ldots f_n) \]

- \(D\) is a non-empty set, the domain
- \(R_i\) is an \(m\)-ary relation for some \(m\) (ie, \(R_i \subseteq D^m\))
- \(f_i\) is an \(n\)-ary function for some \(n\) (ie, \(f_i : D^n \to D\))
A structure is:

\[ A = (D, R_1, \ldots, R_n, f_1, \ldots f_n) \]

- \( D \) is a non-empty set, the domain
- \( R_i \) is an \( m \)-ary relation for some \( m \) (ie, \( R_i \subseteq D^m \))
- \( f_i \) is an \( n \)-ary function for some \( n \) (ie, \( f_i : D^n \to D \))

A graph \( G = (V,E) \)
- \( V \): nodes
- \( E \subseteq V^2 \): edges (binary relation)
- (no functions)

A group, like \((\mathbb{N},+)\)
- \( \mathbb{N} \): natural numbers
- (no relations)
- \( + : \mathbb{N}^2 \to \mathbb{N} \) addition (binary function)
First-order logic
First-order logic

Variables $x, y, z, \ldots$

Quantifiers: $\exists, \forall$

Boolean connectives: $\neg, \land, \lor$

A language to talk about structures

Variables range over the domain

Atomic formulas: $R(x_1, \ldots, x_m), x=y$
**First-order logic**

**FO**

variables $x, y, z, \ldots$

quantifiers: $\exists, \forall$

Boolean connectives: $\neg, \land, \lor$

A language to talk about structures

Variables range over the domain

Atomic formulas: $R(x_1, \ldots, x_m), x = y$

A graph $G = (V, E)$

- $V$: nodes
- $E \subseteq V^2$: edges (binary relation)
- (no functions)

Language to talk about graphs

Variables range over nodes

Atomic formulas: $E(x, y), x = y$
First-order logic

Variables $x, y, z, \ldots$
Quantifiers: $\exists, \forall$
Boolean connectives: $\neg, \land, \lor$

A language to talk about structures
Variables range over the domain
Atomic formulas: $R(x_1, \ldots, x_m), x=y$

A graph $G = (V,E)$
- $V$: nodes
- $E \subseteq V^2$: edges (binary relation)
- (no functions)

Language to talk about graphs
Variables range over nodes
Atomic formulas: $E(x,y), x = y$

Formulas: Atomic formulas + connectives + quantifiers
“The node x has at least two neighbours”

\[ \exists y \exists z (\neg(y=z) \land E(x,y) \land E(x,z)) \]
“The node $x$ has at least two neighbours”

$$\varphi(x) = \exists y \exists z \ (\neg(y=z) \land E(x,y) \land E(x,z))$$

x is **free** = not quantified

(a property of a **node** in the **graph**)
“The node $x$ has at least two neighbours”

$\varphi(x) = \exists y \exists z (\neg(y=z) \land E(x,y) \land E(x,z))$

“Each node has at least two neighbours”

$\forall x \exists y \exists z (\neg(y=z) \land E(x,y) \land E(x,z))$

x is free = not quantified
(a property of a node in the graph)
“The node $x$ has at least two neighbours”

$$\varphi(x) = \exists y \exists z \ (\neg (y=z) \land E(x,y) \land E(x,z))$$

The formula is a **sentence**

- no free variables

(a property of the **graph**)

“Each node has at least two neighbours”

$$\psi = \forall x \exists y \exists z \ (\neg (y=z) \land E(x,y) \land E(x,z))$$

The formula is a **sentence**

- no free variables

(a property of the **graph**)

X is **free** = not quantified

(a property of a **node** in the **graph**)

free
“The node $x$ has at least two neighbours”
$$\varphi(x) = \exists y \exists z \ (\neg(y=z) \land E(x,y) \land E(x,z))$$

Each node has at least two neighbours”
$$\psi = \forall x \exists y \exists z \ (\neg(y=z) \land E(x,y) \land E(x,z))$$

x is free = not quantified
(a property of a node in the graph)

the formula is a sentence
= no free variables
(a property of the graph)

Question: • How to express in FO

“Every two neighbours have a common neighbour”?

• Does it have free variables? Is it a sentence?
"The node $x$ has at least two neighbours"

$\varphi(x) = \exists y \exists z (\neg(y=z) \land E(x,y) \land E(x,z))$

Each node has at least two neighbours"

$\psi = \forall x \exists y \exists z (\neg(y=z) \land E(x,y) \land E(x,z))$

Question: • How to express in FO

"Every two neighbours have a common neighbour"?

• Does it have free variables? Is it a sentence?

Answer: $\forall x \forall y (\neg E(x,y) \lor \exists z ( (E(x,z) \lor E(z,x)) \land (E(y,z) \lor E(z,y)) ))$
To evaluate a formula $\phi$ we need a graph $G=(V,E)$ and a binding $\alpha$ that maps free variables of $\phi$ to nodes of $G$.

$$G \models_\alpha \phi(x_1,\ldots,x_n) \quad \alpha : \{x_1,\ldots,x_n\} \rightarrow V \quad \text{assigns nodes to free variables}$$
To evaluate a formula $\phi$ we need a graph $G=(V,E)$ and a binding $\alpha$ that maps free variables of $\phi$ to nodes of $G$.

\[ G \models_{\alpha} \phi(x_1,\ldots,x_n) \quad \alpha : \{x_1,\ldots,x_n\} \rightarrow V \quad \text{assigns nodes to free variables} \]
To evaluate a formula $\phi$ we need a graph $G=(V,E)$ and a binding $\alpha$ that maps free variables of $\phi$ to nodes of $G$.

"$G,\alpha$ satisfy $\phi$"  \implies  "$\phi$ is satisfiable"

$G \models_\alpha \phi(x_1,\ldots,x_n)$  \quad $\alpha : \{x_1,\ldots,x_n\} \rightarrow V$ assigns nodes to free variables

"The node $x$ has at least two neighbours"

$\phi(x) = \exists y \exists z (\neg(y=z) \land E(x,y) \land E(x,z))$

$G \models_\alpha \phi$ if $\alpha = \{x \mapsto v\}$
To evaluate a formula $\phi$ we need a graph $G=(V,E)$ and a binding $\alpha$ that maps free variables of $\phi$ to nodes of $G$. 

\[
\alpha \propto G \models \alpha \phi
\]

Assigns nodes to free variables

\[
G \models_\alpha \phi(x_1,\ldots,x_n) \quad \alpha : \{x_1,\ldots,x_n\} \longrightarrow V
\]

\[
\text{ assigns nodes to free variables }
\]

\[
\text{“The node } x \text{ has at least two neighbours”}
\]

\[
\phi(x) = \exists y \exists z (\neg(y=z) \land E(x,y) \land E(x,z))
\]

\[
G \models_\alpha \phi \quad \text{if} \quad \alpha = \{x \mapsto v\}
\]

\[
\text{“Every node has at least two neighbours”}
\]

\[
\psi = \forall x \exists y \exists z (\neg(y=z) \land E(x,y) \land E(x,z))
\]

\[
G \models_\emptyset \psi
\]
Formal Semantics of FO

\(G \vDash_{\alpha} \exists x \, \phi \) iff for some \(v \in V\) and \(\alpha' = \alpha \cup \{x \mapsto v\}\) we have \(G \vDash_{\alpha'} \phi\)

\(G \vDash_{\alpha} \forall x \, \phi \) iff for every \(v \in V\) and \(\alpha' = \alpha \cup \{x \mapsto v\}\) we have \(G \vDash_{\alpha'} \phi\)

\(G \vDash_{\alpha} \phi \land \psi \) iff \(G \vDash_{\alpha} \phi\) and \(G \vDash_{\alpha} \psi\)

\(G \vDash_{\alpha} \neg \phi \) iff it is not true that \(G \vDash_{\alpha} \phi\)

\(G \vDash_{\alpha} x = y \) iff \(\alpha(x) = \alpha(y)\)

\(G \vDash_{\alpha} E(x,y) \) iff \((\alpha(x), \alpha(y)) \in E\)
Formulas as queries

\( \phi(x_1, \ldots, x_n) \) evaluated on \( G = (V, E) \) yields all the bindings that satisfy \( \phi \):

\[
\phi(G) = \{ (\alpha(x_1), \ldots, \alpha(x_n)) \mid G \models_\alpha \phi, \ \alpha: \{x_1, \ldots, x_n\} \rightarrow V \}
\]
Formulas as queries

\( \phi(x_1, \ldots, x_n) \) evaluated on \( G=(V,E) \) yields all the bindings that satisfy \( \phi \):

\[
\phi(G) = \{ (\alpha(x_1),\ldots,\alpha(x_n)) \mid G \vDash_\alpha \phi, \alpha: \{x_1,\ldots,x_n\} \rightarrow V \}
\]

“The node \( x \) has at least two neighbours”

\( \phi(x) = \exists y \exists z (\neg(y=z) \land E(x,y) \land E(x,z)) \)

\( \approx \) “Return all nodes with at least two neighbours”
Formulas as queries

\( \phi(x_1, \ldots, x_n) \) evaluated on \( G=(V,E) \) yields all the bindings that satisfy \( \phi \):

\[
\phi(G) = \{ (\alpha(x_1), \ldots, \alpha(x_n)) \mid G \models \alpha, \alpha: \{x_1, \ldots, x_n\} \to V \}
\]

“The node \( x \) has at least two neighbours”

\( \phi(x) = \exists y \exists z (\neg(y=z) \land E(x,y) \land E(x,z)) \)

\[
\phi(G) = \{v, v', v''\}
\]

\[
\phi(G') = \{v, v'\}
\]

“Return all nodes with at least two neighbours”

G

G'
Formulas as queries

\( \phi(x_1, \ldots, x_n) \) evaluated on \( G = (V, E) \) yields all the bindings that satisfy \( \phi \):

\[
\phi(G) = \{ (\alpha(x_1), \ldots, \alpha(x_n)) \mid G \vdash_{\alpha} \phi, \alpha : \{x_1, \ldots, x_n\} \rightarrow V \}
\]

“The node \( x \) has at least two neighbours”

\[
\phi(x) = \exists y \exists z \left( \neg(y = z) \land E(x, y) \land E(x, z) \right)
\]

\( \phi(G) = \{v, v', v''\} \)

\( \phi(G') = \{v, v'\} \)

“Every node has two neighbours”

\[
\psi = \forall x \exists y \exists z \left( \neg(y = z) \land E(x, y) \land E(x, z) \right)
\]

\( \psi(G) = \{()\} \sim \text{set with one element: the 0-tuple} \)

\( \psi(G') = \{\} \sim \text{empty set} \)
Question: Which bindings $\alpha$ verify $G \models_\alpha \phi$ for

$$\phi(x,y) = \exists z \ (E(x,z) \land E(z,y))$$

and

$$G = \begin{array}{c}
\text{v} \\
\text{v'} \\
\text{v''}
\end{array}$$
**Question:** Which bindings $\alpha$ verify $G \models_\alpha \phi$ for

$$\phi(x,y) = \exists z \ (E(x,z) \land E(z,y))$$

and $G =$

Answer:

- $\alpha = \{ x \mapsto v, y \mapsto v' \}$,
- $\alpha = \{ x \mapsto v, y \mapsto v \}$,
- $\alpha = \{ x \mapsto v', y \mapsto v' \}$,
- $\ldots$ and all the rest

$$\phi(G) = \{v,v', v''\} \times \{v,v', v''\}$$
FO can serve as a **declarative** query language on relational databases: we express the properties of the answer

- **Tables** = **Relations**
- **Queries** = **Formulas**
- **Rows** = **Tuples**
FO can serve as a **declarative** query language on relational databases: we express the properties of the answer

Tables = Relations

Queries = Formulas

Rows = Tuples

**Particular to databases:**

- Use of constants
- No functions
- Finite structure
- Quantification over active domain
Formulas as queries

FO can serve as a **declarative** query language on relational databases: we express the properties of the answer

\[
\text{Tables} = \text{Relations} \\
\text{Queries} = \text{Formulas} \\
\text{Rows} = \text{Tuples}
\]

**Particular to databases:**

- Use of constants
- No functions
- Finite structure
- Quantification over active domain

\( \models_{\text{finite}} \) is different from \( \not\models \)
FO can serve as a **declarative** query language on relational databases: we express the properties of the answer

\[
\text{Tables = Relations} \quad \quad \text{Queries = Formulas} \quad \quad \text{Rows = Tuples}
\]

**Particular to databases:**
- Use of constants
- No functions
- Finite structure
- Quantification over active domain

\( \models_{\text{finite}} \) is different from \( \models \)

There are formulas \( \phi \) that are satisfiable only on infinite structures.

Like which?
Formulas as queries

FO can serve as a **declarative** query language on relational databases: we express the properties of the answer

\[
\text{Tables} = \text{Relations} \\
\text{Queries} = \text{Formulas} \\
\text{Rows} = \text{Tuples}
\]

**Particular to databases:**
- Use of constants
- No functions
- Finite structure
- Quantification over active domain

\[\models_{\text{finite}} \text{ is different from } \models\]

There are formulas \( \phi \) that are satisfiable only on infinite structures.

Like which?

\[\phi = "R(x,y) \text{ is an infinite linear order}"\]
Formulas as queries

FO can serve as a declarative query language on relational databases: we express the properties of the answer

Tables = Relations

Queries = Formulas

Rows = Tuples

Particular to databases:
- Use of constants
- No functions
- Finite structure
- Quantification over active domain

\( \models_{\text{finite}} \) is different from \( \models \)

There are formulas \( \phi \) that are satisfiable only on infinite structures.

Like which?

\[ \phi = \text{"R(x,y) is an infinite linear order"} \]

Finite model theory
Formulas as queries

FO can serve as a **declarative** query language on relational databases: we express the properties of the answer

- **Tables** = **Relations**
- **Queries** = **Formulas**
- **Rows** = **Tuples**

\[ \text{RA} = * \text{ FO} \]

\[ \text{How} = \text{What} \]

RA and FO logic have roughly* the same expressive power!

[E.F. Codd 1972]

*FO without functions, with equality, on finite domains, …
Formulas as queries

$\text{RA} \subseteq \text{FO}$

- $R_1 \times R_2 \sim R_1(x_1, \ldots, x_n) \land R_2(x_{n+1}, \ldots, x_m)$
- $R_1 \cup R_2 \sim R_1(x_1, \ldots, x_n) \lor R_2(x_1, \ldots, x_n)$
- $\sigma_{i_1=j_1, \ldots, i_n=j_n}(R) \sim R(x_1, \ldots, x_m) \land (x_{i_1}=x_{j_1}) \land \cdots \land (x_{i_n}=x_{j_n})$
- $\pi_{i_1, \ldots, i_n}(R) \sim \exists(x_1, \ldots, x_m \setminus \{x_{i_1}, \ldots, x_{i_n}\}). R(x_1, \ldots, x_m)$
- $R_1 \setminus R_2 \sim R_1(x_1, \ldots, x_n) \land \neg R_2(x_1, \ldots, x_n)$
- $\ldots$
Formulas as queries

$\text{FO } \subseteq \text{ RA}$ does not hold in general!
Fo ⊆ RA does not hold in general!

“the complement of R” ∉ RA ∈ FO : ¬R(x)
Formulas as queries

FO $\notin$ RA

"the complement of R" $\notin$ RA $\in$ FO : $\neg R(x)$
Formulas as queries


device

\[ \text{FO} \not\subseteq \text{RA} \]

“the complement of R” \( \not\in \text{RA} \) \( \in \text{FO} : \neg R(x) \)

\( \implies \) We restrict variables to range over active domain
Formulas as queries

\[
\text{FO} \not\subseteq \text{RA}
\]

“the complement of R” \( \not\in \text{RA} \) \( \in \text{FO} : \neg R(x) \)

\( \leadsto \) elements in the relations

\( \leadsto \) We restrict variables to range over \textbf{active domain}

\[
\text{FO}^{\text{act}} = \text{FO restricted to active domain}
\]
Formulas as queries

FO \not\subseteq RA

“the complement of R” \notin RA \in FO: \neg R(x)

We restrict variables to range over **active domain**

**FO**^act

= FO restricted to active domain

\phi_1(x,y) = \neg E(x,y)
\phi_1(G) = \{(v_1,v_1),(v_3,v_1)\}

\phi_2(x) = \forall y E(y,x)
\phi_2(G) = \{v_2\}

G =
First-order logic restricted to active domain

Formal Semantics of $\text{FO}^\text{act}$

$G \vDash_\alpha \exists x \phi$ iff for some $v \in \text{ACT}(G)$ and $\alpha' = \alpha \cup \{x \mapsto v\}$ we have $G \vDash_{\alpha'} \phi$

$G \vDash_\alpha \forall x \phi$ iff for every $v \in \text{ACT}(G)$ and $\alpha' = \alpha \cup \{x \mapsto v\}$ we have $G \vDash_{\alpha'} \phi$

$G \vDash_\alpha \phi \land \psi$ iff $G \vDash_\alpha \phi$ and $G \vDash_\alpha \psi$

$G \vDash_\alpha \neg \phi$ iff it is not true that $G \vDash_\alpha \phi$

$G \vDash_\alpha x = y$ iff $\alpha(x) = \alpha(y)$

$G \vDash_\alpha E(x, y)$ iff $(\alpha(x), \alpha(y)) \in E$

$\text{ACT}(G) = \{v \mid \text{for some } v': (v, v') \in E \text{ or } (v', v) \in E\}$
$\text{FO}^{\text{act}} \subseteq \text{RA}$
\[ \text{FO}^\text{act} \subseteq \text{RA} \]

Assume:

1. \( \phi \) has variables \( x_1, \ldots, x_n \),

2. \( \phi \) in normal form: \( (\exists^* (\neg \exists^*)^* + \text{quantifier-free } \psi(x_1, \ldots, x_n) \)

\[
\exists x_1, x_2 \neg \exists x_3 \exists x_4 . \ ( E(x_1, x_3) \land \neg E(x_4, x_2) ) \lor (x_1 = x_3)
\]
\[ \text{Assume:} \]

1. \( \phi \) has variables \( x_1, \ldots, x_n \),  

2. \( \phi \) in normal form: 
   \( (\exists^* (\neg \exists^*)^* + \text{quantifier-free } \psi(x_1, \ldots, x_n) \)  

\[ \exists x_1, x_2 \neg \exists x_3 \exists x_4 \cdot (E(x_1, x_3) \land \neg E(x_4, x_2)) \lor (x_1 = x_3) \]

\textit{Adom} = RA expression for active domain = “\( \pi_1(E) \cup \pi_2(E) \)”

- \( (\exists x_i \phi(x_{i_1}, \ldots, x_{i_n}))^+ \leadsto \pi\{i_1, \ldots, i_n\}\backslash\{i\}(\phi^+) \)
- \( (x_i = x_j)^+ \leadsto \sigma_{\{i=j\}}(\text{Adom} \times \cdots \times \text{Adom}) \)
- \( (\psi_1(x_1, \ldots, x_n) \land \psi_2(x_1, \ldots, x_n))^+ \leadsto \psi_1^+ \cap \psi_2^+ \)
- \( (\neg \phi(x_{i_1}, \ldots, x_{i_n}))^+ \leadsto \text{Adom} \times \cdots \times \text{Adom} \setminus \phi^+ \)
\( \text{FO}^{\text{act}} \subseteq \text{RA} \)

Assume:

1. \( \phi \) has variables \( x_1, ..., x_n \),
2. \( \phi \) in normal form: \( (\exists^* (\neg \exists)^* + \text{quantifier-free } \psi(x_1, ..., x_n) \)

\[ \exists x_1, x_2 \neg \exists x_3 \exists x_4 . \left( E(x_1, x_3) \land \neg E(x_4, x_2) \right) \lor (x_1 = x_3) \]

**Adom** = RA expression for active domain = “\( \pi_1(E) \cup \pi_2(E) \)”

- \( (\exists x_i \phi(x_{i_1}, ..., x_{i_n}))^+ \sim \pi_{\{i_1, ..., i_n\}\{i\}}(\phi^+) \)
- \( (x_i = x_j)^+ \sim \sigma_{\{i=j\}}(\text{Adom} \times ... \times \text{Adom}) \)
- \( (\psi_1(x_1, ..., x_n) \land \psi_2(x_1, ..., x_n))^+ \sim \psi_1^+ \land \psi_2^+ \)
- \( (\neg \phi(x_{i_1}, ..., x_{i_n}))^+ \sim \text{Adom} \times ... \times \text{Adom} \setminus \phi^+ \)

A \cap B = (A \cup B) \setminus A \setminus B
Corollary

$\text{FO}^{\text{act}}$ is equivalent to RA
Question 1: How is $\pi_2(\sigma_{1=3}(\pi_2(\sigma_{1=3}(R_1 \times R_2)) \times R_2))$ expressed in FO?

Remember: $R_1, R_2$ are binary

Question 2: How is $\exists y,z . (R_1(x,y) \land R_1(y,z) \land x \neq z)$ expressed in RA?

Remember: The signature is the same as before ($R_1, R_2$ binary)

- $R_1 \cup R_2$
- $R_1 \times R_2$
- $R_1 \setminus R_2$
- $\sigma_{\{i_1=j_1, ..., i_n=j_n\}}(R) := \{(x_1, ..., x_m) \in R | (x_{i_1} = x_{j_1}) \land ... \land (x_{i_n} = x_{j_n}) \}$
- $\pi_{\{i_1, ..., i_n\}}(R) := \{(x_{i_1}, ..., x_{i_n}) | (x_1, ..., x_m) \in R\}$
Question 1: How is $\pi_2(\sigma_{1=3}(\pi_2(\sigma_{1=3}(R_1 \times R_2)) \times R_2))$ expressed in FO?
Remember: $R_1, R_2$ are binary

Answer: $\exists x_2 . \left( \exists x_1, x_4 . \left( R_1(x_1, x_2) \land R_2(x_1, x_4) \right) \land R_2(x_2, x_5) \right)$

Question 2: How is $\exists y, z . \left( R_1(x, y) \land R_1(y, z) \land x \neq z \right)$ expressed in RA?
Remember: The signature is the same as before ($R_1, R_2$ binary)

- $R_1 \cup R_2$
- $R_1 \times R_2$
- $R_1 \setminus R_2$
- $\sigma_{\{i_1=j_1, \ldots, i_n=j_n\}}(R) := \{ (x_1, \ldots, x_m) \in R \mid (x_{i_1} \neq x_{j_1}) \land \cdots \land (x_{i_n} \neq x_{j_n}) \}$
- $\pi_{\{i_1, \ldots, i_n\}}(R) := \{ (x_{i_1}, \ldots, x_{i_n}) \mid (x_1, \ldots, x_m) \in R \}$
**Question 1:** How is $\pi_2(\sigma_1=3(\pi_2(\sigma_1=3(R_1 \times R_2)) \times R_2))$ expressed in FO?  
**Remember:** $R_1, R_2$ are binary

**Answer:** $\exists x_2 . \left( \exists x_1, x_4 . \left( R_1(x_1, x_2) \land R_2(x_1, x_4) \right) \land R_2(x_2, x_5) \right)$

**Question 2:** How is $\exists y, z . (R_1(x, y) \land R_1(y, z) \land x \neq z)$ expressed in RA?  
**Remember:** The signature is the same as before ($R_1, R_2$ binary)

- $R_1 \cup R_2$
- $R_1 \times R_2$
- $R_1 \setminus R_2$
- $\sigma_{\{i_1 = j_1, \ldots, i_n = j_n\}}(R) := \{(x_1, \ldots, x_m) \in R \mid (x_{i_1} \neq x_{j_1}) \land \cdots \land (x_{i_n} \neq x_{j_n})\}$
- $\pi_{\{i_1, \ldots, i_n\}}(R) := \{(x_{i_1}, \ldots, x_{i_n}) \mid (x_1, \ldots, x_m) \in R\}$

**Answer:** $\pi_1(\sigma_{\{2=3,1\neq 4\}}(R_1 \times R_1))$
Logic = Algebra = Programming language

FO = RA = SQL
Logic = Algebra = Programming language

- over active domain
- on finite domains
- very basic

FO = RA = SQL

on finite domains
over active domain
very basic
**Evaluation problem:** Given a query $Q$, a database instance $db$, and a tuple $t$, is $t \in Q(db)$?

⇒ How hard is it to retrieve data?
**Evaluation problem:** Given a query $Q$, a database instance $db$, and a tuple $t$, is $t \in Q(db)$?

$\Rightarrow$ How hard is it to retrieve data?

**Emptiness problem:** Given a query $Q$, is there a database instance $db$ so that $Q(db) \neq \emptyset$?

$\Rightarrow$ Does $Q$ make sense? Is it a contradiction? (Query optimization)
**Algorithmic problems for query languages**

**Evaluation problem:** Given a query $Q$, a database instance $db$, and a tuple $t$, is $t \in Q(db)$?

⇒ How hard is it to retrieve data?

**Emptiness problem:** Given a query $Q$, is there a database instance $db$ so that $Q(db) \neq \emptyset$?

⇒ Does $Q$ make sense? Is it a contradiction? (Query optimization)

**Equivalence problem:** Given queries $Q_1$, $Q_2$, is $Q_1(db) = Q_2(db)$ for all database instances $db$?

⇒ Can we safely replace a query with another? (Query optimization)
What can be *mechanized*? $\leadsto$ decidable/undecidable

How *hard* is it to mechanise? $\leadsto$ complexity classes
Complexity theory

What can be **mechanized**? \(\sim\) decidable/undecidable

How **hard** is it to mechanise? \(\sim\) complexity classes
Complexity theory

What can be **mechanized**? \(\sim\) **decidable/undecidable**

How **hard** is it to mechanise? \(\sim\) **complexity classes**

\(\Rightarrow\) usage of resources: • time
• memory
Complexity theory

What can be *mechanized*? $\leadsto$ decidable/undecidable

How *hard* is it to mechanise? $\leadsto$ complexity classes

- usage of resources: 
  - time 
  - memory

Algorithm $\text{Alg}$ is *TIME*-bounded by a function $f : \mathbb{N} \rightarrow \mathbb{N}$ if $\text{Alg}(\text{input})$ uses less than $f(|\text{input}|)$ units of TIME.
Complexity theory

What can be **mechanized**? \(\sim\) decidable/undecidable

How **hard** is it to mechanise? \(\sim\) complexity classes

usage of resources:
- time
- memory

Algorithm \(\text{Alg}\) is **TIME**-bounded by a function \(f: \mathbb{N} \rightarrow \mathbb{N}\) if \(\text{Alg}(\text{input})\) uses less than \(f(|\text{input}|)\) units of **TIME**.
Complexity theory

What can be **mechanized**?

~ decidable/undecidable

How **hard** is it to mechanise?

~ complexity classes

usage of resources: • time
• memory

Algorithm \( \text{Alg} \) is \( \text{TIME} \)-bounded by a function \( f: \mathbb{N} \rightarrow \mathbb{N} \) if \( \text{Alg}(\text{input}) \) uses less than \( f(|\text{input}|) \) units of \( \text{TIME} \).

Algorithm \( \text{Alg} \) is \( \text{SPACE} \)-bounded by a function \( f: \mathbb{N} \rightarrow \mathbb{N} \) if \( \text{Alg}(\text{input}) \) uses less than \( f(|\text{input}|) \) units of \( \text{SPACE} \).
Complexity theory

What can be **mechanized**? $\sim$ **decidable/undecidable**

How **hard** is it to mechanise? $\sim$ **complexity classes**

usage of resources: • time
• memory

Algorithm $\text{Alg}$ is $\text{T}IME$-bounded by a function $f : \mathbb{N} \rightarrow \mathbb{N}$ if $\text{Alg}(\text{input})$ uses less than $f(|\text{input}|)$ units of $\text{T}IME$.

$\text{LOGSPACE} \subseteq \text{PTIME} \subseteq \text{PSPACE} \subseteq \text{EXPTIME} \subseteq \cdots$
What can be **mechanized**?  \sim \text{decidable/undecidable}

**How hard** is it to mechanise?  \sim \text{complexity classes}

\begin{itemize}
\item usage of resources:  
  \begin{itemize}
  \item time
  \item memory
  \end{itemize}
\end{itemize}

**SPACE**

Algorithm **Alg** is **TIME**-bounded by a function \( f: \mathbb{N} \to \mathbb{N} \) if \( \text{Alg}(\text{input}) \) uses less than \( f(|\text{input}|) \) units of **TIME**.

**LOGSPACE \subseteq \textbf{PTIME} \subseteq \textbf{PSPACE} \subseteq \textbf{EXPTIME} \subseteq \cdots**

**SPACE**-bounded by a polynomial

TIME-bounded by a polynomial

**SPACE**-bounded by \( \log(n) \)
Algorithmic problems for FO

Evaluation problem: Given a FO formula $\phi(x_1, ..., x_n)$, a graph $G$, and a binding $\alpha$, does $G \models_\alpha \phi$?

Satisfiability problem: Given a FO formula $\phi$, is there a graph $G$ and binding $\alpha$, such that $G \models_\alpha \phi$?

Equivalence problem: Given FO formulae $\phi, \psi$, is $G \models_\alpha \phi \text{ iff } G \models_\alpha \psi$ for all graphs $G$ and bindings $\alpha$?
Algorithmic problems for FO

Evaluation problem: Given a FO formula $\phi(x_1, ..., x_n)$, a graph $G$, and a binding $\alpha$, does $G \models_{\alpha} \phi$?

Satisfiability problem: Given a FO formula $\phi$, is there a graph $G$ and binding $\alpha$, such that $G \models_{\alpha} \phi$?

Equivalence problem: Given FO formulae $\phi, \psi$, is $G \models_{\alpha} \phi \iff G \models_{\alpha} \psi$ for all graphs $G$ and bindings $\alpha$?
Algorithmic problems for FO

**Evaluation problem:** Given a FO formula \( \phi(x_1, \ldots, x_n) \), a graph \( G \), and a binding \( \alpha \), does \( G \vDash_\alpha \phi \) ?

DECIDABLE \( \iff \) foundations of the database industry

**Satisfiability problem:** Given a FO formula \( \phi \), is there a graph \( G \) and binding \( \alpha \), such that \( G \vDash_\alpha \phi \) ?

💀 UNDECIDABLE \( \iff \) both for \( \vDash \) and \( \vDash_{\text{finite}} \)

**Equivalence problem:** Given FO formulae \( \phi, \psi \), is

\[
G \vDash_\alpha \phi \iff G \vDash_\alpha \psi
\]

for all graphs \( G \) and bindings \( \alpha \)?
Algorithmic problems for FO

**Evaluation problem:** Given a FO formula $\phi(x_1, \ldots, x_n)$, a graph $G$, and a binding $\alpha$, does $G \models_\alpha \phi$?

DECIDABLE $\iff$ foundations of the database industry

**Satisfiability problem:** Given a FO formula $\phi$, is there a graph $G$ and binding $\alpha$, such that $G \models_\alpha \phi$?

💀 UNDECIDABLE $\iff$ both for $\models$ and $\models_{\text{finite}}$

**Equivalence problem:** Given FO formulae $\phi, \psi$, is $G \models_\alpha \phi$ iff $G \models_\alpha \psi$ for all graphs $G$ and bindings $\alpha$?

💀 UNDECIDABLE $\iff$ by reduction to the satisfiability problem
Satisfiability problem: Given a FO formula $\phi$, is there a graph $G$ and binding $\alpha$, such that $G \models_\alpha \phi$?

💀 UNDECIDABLE $\iff$ both for $\models$ and $\models_{finite}$ [Trakhtenbrot ’50]
**Satisfiability problem**: Given a FO formula $\phi$, is there a graph $G$ and binding $\alpha$, such that $G \models \alpha \phi$?

💀 **UNDECIDABLE** $\iff$ both for $\models$ and $\models_{\text{finite}}$  
[Trakhtenbrot ’50]

Proof: By reduction from the Domino (aka Tiling) problem.
Satisfiability problem: Given a FO formula $\phi$, is there a graph $G$ and binding $\alpha$, such that $G \models_\alpha \phi$?

💀 UNDECIDABLE $\iff$ both for $\models$ and $\models_{\text{finite}}$  
[Trakhtenbrot ’50]

Proof: By reduction from the Domino (aka Tiling) problem.

Reduction from $P$ to $P'$: Algorithm that solves $P$ using a $O(1)$ procedure $“P'(x)”$ that returns the truth value of $P'(x)$. 
The (undecidable) Domino problem

Input: 4-sided dominos:
The (undecidable) Domino problem

Input: 4-sided dominos:

Output: Is it possible to form a white-bordered rectangle? (of any size)
The (undecidable) Domino problem

Input: 4-sided dominos:

Output: Is it possible to form a white-bordered rectangle? (of any size)

Rules: sides must match, you can’t rotate the dominos, but you can ‘clone’ them.
The (undecidable) Domino problem

Domino - Why is it undecidable?

It can easily encode *halting* computations of Turing machines:
The (undecidable) Domino problem

Domino - Why is it undecidable?

It can easily encode *halting* computations of Turing machines:

(Head is elsewhere, symbol is not modified)
The (undecidable) Domino problem

Domino - Why is it undecidable?

It can easily encode *halting* computations of Turing machines:

- (head is elsewhere, symbol is not modified)
- (head is here, symbol is rewritten, head moves right)
The (undecidable) Domino problem

Domino - Why is it undecidable?

It can easily encode *halting* computations of Turing machines:

- 0 1 2
  - 0 1 2 (head is elsewhere, symbol is not modified)
- 1 r r
  - q 0 2 (head is here, symbol is rewritten, head moves right)
- l 2 l
  - l 1 q 0 (head is here, symbol is rewritten, head moves left)

[Diagram of Domino tiles and Turing machine states]
The (undecidable) Domino problem

Domino - Why is it undecidable?

It can easily encode *halting* computations of Turing machines:

- \( s0 \) (initial configuration)
- \( 0 \) (head is elsewhere, symbol is not modified)
- \( q0 \) (head is here, symbol is rewritten, head moves right)
- \( l1 \) (head is here, symbol is rewritten, head moves left)

...
The (undecidable) Domino problem

Domino - Why is it undecidable?

It can easily encode *halting* computations of Turing machines:

- **0 0 2 2** (head is elsewhere, symbol is not modified)
- **1 r 2 2** (head is here, symbol is rewritten, head moves right)
- **l2 l 1 q0** (head is here, symbol is rewritten, head moves left)
- **s0 0 0 0** (initial configuration)
- **b0 0 0 0** (halting configuration)

...
1. There is a grid: $H( , )$ and $V( , )$ are relations representing bijections such that...
1. There is a grid: $H(\ ), V(\ )$ are relations representing bijections such that...
1. There is a grid: $H(\ , \ )$ and $V(\ , \ )$ are relations representing bijections such that...
1. There is a grid: $H(,)$ and $V(,)$ are relations representing bijections such that...
Domino \Leftrightarrow \text{Sat-FO} \ (\text{domino has a solution iff } \phi \text{ satisfiable})

1. There is a grid: $H(,)$ and $V(,)$ are relations representing bijections such that...

2. Assign one domino to each node: a unary relation $D(x)$ for each domino
1. There is a grid: $H(\ ,\ )$ and $V(\ ,\ )$ are relations representing bijections such that...

2. Assign one domino to each node: a unary relation

\[ \text{for each domino } \]

3. Match the sides $\forall x,y$

if $H(x,y)$, then $D_a(x) \land D_b(y)$

for some dominos $a, b$ that ‘match’ horizontally (Idem vertically)
1. There is a grid: $H( , )$ and $V( , )$ are relations representing bijections such that...

2. Assign one domino to each node: a unary relation

   $$D_a(x)$$

   for each domino $\square$

3. Match the sides $\forall x,y$ if $H(x,y)$, then $D_a(x) \land D_b(y)$ for some dominos $a,b$ that ‘match’ horizontally (Idem vertically)

4. Borders are white.
Algorithmic problems for FO

**Evaluation problem:** Given a FO formula $\phi(x_1, \ldots, x_n)$, a graph $G$, and a binding $\alpha$, does $G \models_\alpha \phi$?

**Satisfiability problem:** Given a FO formula $\phi$, is there a graph $G$ and binding $\alpha$, such that $G \models_\alpha \phi$?

**Equivalence problem:** Given FO formulae $\phi, \psi$, is $G \models_\alpha \phi$ iff $G \models_\alpha \psi$ for all graphs $G$ and bindings $\alpha$?

DECIDABLE $\iff$ foundations of the database industry

UNDECIDABLE $\iff$ both for $\models$ and $\models_{\text{finite}}$

UNDECIDABLE $\iff$ by reduction to the satisfiability problem
**Equivalence problem:** Given FO formulae $\phi, \psi$, is $G \models_\alpha \phi$ iff $G \models_\alpha \psi$ for all graphs $G$ and bindings $\alpha$?

💀 **UNDECIDABLE** $\iff$ by reduction to the satisfiability problem
Algorithmic problems for FO

\( \phi \) is **satisfiable** iff \( \phi \) is **not equivalent** to \( \perp \)

Satisfiability problem undecidable \( \iff \) Equivalence problem undecidable

**Equivalence problem:** Given FO formulae \( \phi, \psi \), is

\[ G \vDash_{\alpha} \phi \iff G \vDash_{\alpha} \psi \]

for all graphs \( G \) and bindings \( \alpha \)?

💀 **UNDICIDABLE** \( \iff \) by reduction to the satisfiability problem
Algorithmic problems for FO

\[ \phi \text{ is satisfiable iff } \phi \text{ is not equivalent to } \bot \]

Satisfiability problem undecidable \( \implies \) Equivalence problem undecidable

Actually, there are reductions in both senses:

\( \phi(x_1, \ldots, x_n) \) and \( \psi(y_1, \ldots, y_m) \) are equivalent iff

- \( n = m \)
- \( (x_1 = y_1) \land \cdots \land (x_n = y_n) \land \phi(x_1, \ldots, x_n) \land \neg \psi(y_1, \ldots, y_n) \) is unsatisfiable
- \( (x_1 = y_1) \land \cdots \land (x_n = y_n) \land \psi(x_1, \ldots, x_n) \land \neg \phi(y_1, \ldots, y_n) \) is unsatisfiable

Equivalence problem: Given FO formulae \( \phi, \psi \), is

\[ G \models_\alpha \phi \iff G \models_\alpha \psi \]

for all graphs \( G \) and bindings \( \alpha \)?

💀 UNDECIDABLE \( \iff \) by reduction to the satisfiability problem
Algorithmic problems for FO

**Evaluation problem:** Given a FO formula $\phi(x_1, \ldots, x_n)$, a graph $G$, and a binding $\alpha$, does $G \models_\alpha \phi$?

DECIDABLE $\iff$ foundations of the database industry

**Satisfiability problem:** Given a FO formula $\phi$, is there a graph $G$ and binding $\alpha$, such that $G \models_\alpha \phi$?

💀 UNDECIDABLE $\iff$ both for $\models$ and $\models_{\text{finite}}$

**Equivalence problem:** Given FO formulae $\phi, \psi$, is $G \models_\alpha \phi$ iff $G \models_\alpha \psi$ for all graphs $G$ and bindings $\alpha$?

💀 UNDECIDABLE $\iff$ by reduction to the satisfiability problem
Evaluation problem for FO

\[ \phi(x_1, \ldots, x_n) \]

Input: \( G = (V, E) \)

\( \alpha = \{x_1, \ldots, x_n\} \rightarrow V \)

Output: \( G \models_\alpha \phi \)?
Evaluation problem for FO

\[ \phi(x_1, \ldots, x_n) \]

Input: \( G = (V,E) \)

Output: \( G \Vdash_\alpha \phi ? \)

\[ \alpha = \{x_1, \ldots, x_n\} \rightarrow V \]

Encoding of \( G = (V, E) \)

- each node is coded with a bit string of size \( \log(|V|) \),
- edge set is encoded by its tuples, e.g. \((100,101), (010, 010), \ldots\).

Cost of coding: \( ||G|| = |E| \cdot 2 \cdot \log(|V|) \approx |V| \) (mod a polynomial)
Evaluation problem for FO

\[ \phi(x_1, \ldots, x_n) \]

Input: \[ G = (V, E) \]

Output: \[ G \models_{\alpha} \phi ? \]

α = \{x_1, \ldots, x_n\} \rightarrow V

Encoding of \( G = (V, E) \)

- each node is coded with a bit string of size \( \log(|V|) \),
- edge set is encoded by its tuples, e.g. (100,101), (010, 010), ...

Cost of coding: \( ||G|| = |E| \cdot 2 \cdot \log(|V|) \approx |V| \) (mod a polynomial)

Encoding of \( \alpha = \{x_1, \ldots, x_n\} \rightarrow V \)

- each node is coded with a bit string of size \( \log(|V|) \),

Cost of coding: \( ||\alpha|| = n \cdot \log(|V|) \)
Evaluation problem for FO

Input: \( \phi(x_1, \ldots, x_n) \)
\( G = (V, E) \)
\( \alpha = \{x_1, \ldots, x_n\} \rightarrow V \)

Output: \( G \models_{\alpha} \phi \)?
Evaluation problem for FO

Input: \[\phi(x_1, \ldots, x_n)\]
\[G = (V,E)\]
\[\alpha = \{x_1, \ldots, x_n\} \rightarrow V\]

Output: \(G \models_\alpha \phi?\)

- If \(\phi(x_1, \ldots, x_n) = E(x_i, x_j)\):
  answer YES iff \((\alpha(x_i), \alpha(x_j)) \in E\)

- If \(\phi(x_1, \ldots, x_n) = \psi(x_1, \ldots, x_n) \land \psi'(x_1, \ldots, x_n)\):
  answer YES iff \(G \models_\alpha \psi\) and \(G \models_\alpha \psi'\)

- If \(\phi(x_1, \ldots, x_n) = \neg \psi(x_1, \ldots, x_n)\):
  answer NO iff \(G \models_\alpha \psi\)

- If \(\phi(x_1, \ldots, x_n) = \exists y. \psi(x_1, \ldots, x_n, y)\):
  answer YES iff for some \(v \in V\) and \(\alpha' = \alpha \cup \{y \mapsto v\}\)
  we have \(G \models_{\alpha'} \psi\).
Evaluation problem for FO

\[
\begin{align*}
\text{Input:} & \quad \begin{cases} 
\phi(x_1,\ldots,x_n) \\
G = (V,E) \\
\alpha = \{x_1,\ldots,x_n\} \rightarrow V
\end{cases} \\
\text{Output:} & \quad G \vDash_{\alpha} \phi ?
\end{align*}
\]

- If \( \phi(x_1,\ldots,x_n) = E(x_i,x_j) \):
  answer YES iff \( (\alpha(x_i),\alpha(x_j)) \in E \)

- If \( \phi(x_1,\ldots,x_n) = \psi(x_1,\ldots,x_n) \land \psi'(x_1,\ldots,x_n) \):
  answer YES iff \( G \vDash_{\alpha} \psi \) and \( G \vDash_{\alpha} \psi' \)

- If \( \phi(x_1,\ldots,x_n) = \neg \psi(x_1,\ldots,x_n) \):
  answer NO iff \( G \vDash_{\alpha} \psi \)

- If \( \phi(x_1,\ldots,x_n) = \exists y . \psi(x_1,\ldots,x_n,y) \):
  answer YES iff for some \( v \in V \) and \( \alpha' = \alpha \cup \{y\mapsto v\} \) we have \( G \vDash_{\alpha'} \psi \).

Question:
How much space does it take?
Evaluation problem for FO

Input: \( \phi(x_1, \ldots, x_n) \)
\[ G = (V, E) \]
\[ \alpha = \{x_1, \ldots, x_n\} \rightarrow V \]

Output: \( G \models_\alpha \phi \) ?

- If \( \phi(x_1, \ldots, x_n) = E(x_i, x_j) \):
  answer YES iff \( (\alpha(x_i), \alpha(x_j)) \in E \)

- If \( \phi(x_1, \ldots, x_n) = \psi(x_1, \ldots, x_n) \wedge \psi'(x_1, \ldots, x_n) \):
  answer YES iff \( G \models_\alpha \psi \) and \( G \models_\alpha \psi' \)

- If \( \phi(x_1, \ldots, x_n) = \neg \psi(x_1, \ldots, x_n) \):
  answer NO iff \( G \models_\alpha \psi \)

- If \( \phi(x_1, \ldots, x_n) = \exists y . \psi(x_1, \ldots, x_n, y) \):
  answer YES iff for some \( v \in V \) and \( \alpha' = \alpha \cup \{y \mapsto v\} \)
  \( G \models_{\alpha'} \psi \).

Question:
How much space does it take?

use 4 pointers \( \Leftrightarrow \) LOGSPACE
Evaluation problem for FO

Input: \( \phi(x_1, \ldots, x_n) \)
\[ G = (V, E) \]
\[ \alpha = \{x_1, \ldots, x_n\} \rightarrow V \]

Output: \( G \models_\alpha \phi \) ?

- If \( \phi(x_1, \ldots, x_n) = E(x_i, x_j) \):
  answer YES iff \( (\alpha(x_i), \alpha(x_j)) \in E \)

- If \( \phi(x_1, \ldots, x_n) = \psi(x_1, \ldots, x_n) \land \psi'(x_1, \ldots, x_n) \):
  answer YES iff \( G \models_\alpha \psi \) and \( G \models_\alpha \psi' \)

- If \( \phi(x_1, \ldots, x_n) = \neg \psi(x_1, \ldots, x_n) \):
  answer NO iff \( G \models_\alpha \psi \)

- If \( \phi(x_1, \ldots, x_n) = \exists y . \psi(x_1, \ldots, x_n, y) \):
  answer YES iff for some \( v \in V \) and \( \alpha' = \alpha \cup \{y \mapsto v\} \)
  we have \( G \models_{\alpha'} \psi \).

Question:
How much space does it take?
Evaluation problem for FO

Input: \( \phi(x_1,\ldots,x_n) \)
\( G = (V,E) \)
\( \alpha = \{x_1,\ldots,x_n\} \rightarrow V \)

Output: \( G \models_\alpha \phi \) ?

- If \( \phi(x_1,\ldots,x_n) = E(x_i,x_j) \):
  answer YES iff \( (\alpha(x_i),\alpha(x_j)) \in E \)
  use 4 pointers \( \Rightarrow \) LOGSPACE

- If \( \phi(x_1,\ldots,x_n) = \psi(x_1,\ldots,x_n) \land \psi'(x_1,\ldots,x_n) \):
  answer YES iff \( G \models_\alpha \psi \) and \( G \models_\alpha \psi' \)
  \( \Rightarrow \) \( \text{MAX}( \text{SPACE}(G \models_\alpha \psi), \text{SPACE}(G \models_\alpha \psi')) \) \( \Rightarrow \) \( \text{SPACE}(G \models_\alpha \psi) \)

- If \( \phi(x_1,\ldots,x_n) = \neg \psi(x_1,\ldots,x_n) \):
  answer NO iff \( G \models_\alpha \psi \)

- If \( \phi(x_1,\ldots,x_n) = \exists y . \psi(x_1,\ldots,x_n,y) \):
  answer YES iff for some \( v \in V \) and \( \alpha' = \alpha \cup \{y \mapsto v\} \)
  we have \( G \models_{\alpha'} \psi \).

Question:
How much space does it take?
Evaluation problem for FO

Input: 
\[ \phi(x_1, \ldots, x_n) \]
\[ G = (V, E) \]
\[ \alpha = \{x_1, \ldots, x_n\} \rightarrow V \]

Output: 
\[ G \models_\alpha \phi \]

1. If \( \phi(x_1, \ldots, x_n) = E(x_i, x_j) \): 
   answer YES if \( (\alpha(x_i), \alpha(x_j)) \in E \)

2. If \( \phi(x_1, \ldots, x_n) = \psi(x_1, \ldots, x_n) \land \psi'(x_1, \ldots, x_n) \): 
   answer YES if \( G \models_\alpha \psi \) and \( G \models_\alpha \psi' \) 

3. If \( \phi(x_1, \ldots, x_n) = \neg \psi(x_1, \ldots, x_n) \): 
   answer NO if \( G \models_\alpha \psi \)

4. If \( \phi(x_1, \ldots, x_n) = \exists y . \psi(x_1, \ldots, x_n, y) \): 
   answer YES if for some \( v \in V \) and \( \alpha' = \alpha \cup \{y \mapsto v\} \) 
   we have \( G \models_{\alpha'} \psi \).

Question: How much space does it take?
Evaluation problem for FO

Input: \[ \phi(x_1, \ldots, x_n) \]
\[ G = (V, E) \]
\[ \alpha = \{x_1, \ldots, x_n\} \rightarrow V \]

Output: \( G \models_\alpha \phi ? \)

- If \( \phi(x_1, \ldots, x_n) = E(x_i, x_j) \):
  answer YES iff \( (\alpha(x_i), \alpha(x_j)) \in E \)

- If \( \phi(x_1, \ldots, x_n) = \psi(x_1, \ldots, x_n) \land \psi'(x_1, \ldots, x_n) \):
  answer YES iff \( G \models_\alpha \psi \) and \( G \models_\alpha \psi' \)

- If \( \phi(x_1, \ldots, x_n) = \neg \psi(x_1, \ldots, x_n) \):
  answer NO iff \( G \models_\alpha \psi \)

- If \( \phi(x_1, \ldots, x_n) = \exists y . \psi(x_1, \ldots, x_n, y) \):
  answer YES iff for some \( v \in V \) and \( \alpha' = \alpha \cup \{y \mapsto v\} \) we have \( G \models_{\alpha'} \psi \).

Question: How much space does it take?

\[ 2 \cdot \log(|G|) + \cdots + 2 \cdot \log(|G|) + k \cdot \log(|\alpha| + |G|) \text{ space} \]
\[ \leq |\phi| \text{ times} \]
Evaluation problem for FO in PSPACE

Input: \( \phi(x_1,\ldots,x_n) \)

\( G = (V,E) \)

\( \alpha = \{x_1,\ldots,x_n\} \rightarrow V \)

Output: \( G \vDash_\alpha \phi ? \)

- If \( \phi(x_1,\ldots,x_n) = E(x_i,x_j) \): 
  answer YES iff \( (\alpha(x_i),\alpha(x_j)) \in E \)

- If \( \phi(x_1,\ldots,x_n) = \psi(x_1,\ldots,x_n) \land \psi'(x_1,\ldots,x_n) \): 
  answer YES iff \( G \vDash_\alpha \psi \) and \( G \vDash_\alpha \psi' \)

- If \( \phi(x_1,\ldots,x_n) = \neg \psi(x_1,\ldots,x_n) \):
  answer NO iff \( G \vDash_\alpha \psi \)

- If \( \phi(x_1,\ldots,x_n) = \exists y . \psi(x_1,\ldots,x_n,y) \):
  answer YES iff for some \( v \in V \) and \( \alpha' = \alpha \cup \{y \mapsto v\} \) we have \( G \vDash_{\alpha'} \psi \).

Question:
How much space does it take?

\[ 2 \cdot \log(|G|) + \ldots + 2 \cdot \log(|G|) + k \cdot \log(|\alpha| + |G|) \text{ space} \leq |\phi| \text{ times} \]
Problem: Usual scenario in database

A database of size $10^6$
A query of size 100
**Problem:** Usual scenario in database

A **database** of size $10^6$

A **query** of size 100

**Input:** database + query
Problem: Usual scenario in database

A database of size $10^6$

A query of size 100

Input: database

+ query

But we don’t distinguish this in the analysis:

\[
\text{TIME}(2|\text{query}| + |\text{data}|) = \text{TIME}(|\text{query}| + 2|\text{data}|)
\]
Separation of concerns: How the resources grow with respect to

• the size of the data
• the query size

Query and data play very different roles.

[Vardi, 1982]
Combined complexity: input size is $|\text{query}| + |\text{data}|$

Query complexity ($|\text{data}|$ fixed): input size is $|\text{query}|$

Data complexity ($|\text{query}|$ fixed): input size is $|\text{data}|$
Combined complexity: input size is $|\text{query}| + |\text{data}|$

Query complexity ($|\text{data}|$ fixed): input size is $|\text{query}|$

Data complexity ($|\text{query}|$ fixed): input size is $|\text{data}|$

\[
\begin{align*}
O(2^{|\text{query}|} + |\text{data}|) & \text{ is exponential in combined complexity} \\
& \text{exponential in query complexity} \\
& \text{linear in data complexity} \\
O(|\text{query}| + 2^{|\text{data}|}) & \text{ is exponential in combined complexity} \\
& \text{linear in query complexity} \\
& \text{exponential in data complexity}
\end{align*}
\]
Question

What is the data, query and combined complexity for the evaluation problem for FO?

Remember:  
- **data** complexity, input size: $|\text{data}|$
- **query** complexity, input size: $|\text{query}|$
- **combined** complexity, input size: $|\text{data}| + |\text{query}|$

$|\phi| \cdot 2 \cdot \log(|G|) + k \cdot \log(|\alpha| + |G|)$ space
Question

What is the data, query and combined complexity for the evaluation problem for FO?

Remember:  
- **data** complexity, input size: $|\text{data}|$
- **query** complexity, input size: $|\text{query}|$
- **combined** complexity, input size: $|\text{data}| + |\text{query}|$

$$|\phi| \cdot 2 \cdot \log(|G|) + k \cdot \log(|\alpha| + |G|) \text{ space}$$

$O(\log(|\text{data}|) \cdot |\text{query}|) \text{ space}$

**PSPACE** combined and query complexity

**LOGSPACE** data complexity
Evaluation pb for FO is PSPACE-complete (combined complexity)

PSPACE-complete problem: **QBF**

(satisfaction of Quantified Boolean Formulas)

QBF = a boolean formula with quantification over the truth values (T,F)
PSPACE-complete problem: **QBF**

(satisfaction of Quantified Boolean Formulas)

QBF = a boolean formula with quantification over the truth values (T,F)

\[ \exists p \ \forall q . (p \lor \neg q) \text{ where } p,q \text{ range over } \{T,F\} \]
PSPACE-complete problem: \textbf{QBF}  
\textit{(satisfaction of Quantified Boolean Formulas)}

QBF = a boolean formula with quantification over the truth values (T,F)

\[ \exists p \ \forall q . \ (p \lor \neg q) \quad \text{where } p,q \text{ range over } \{T,F\} \]

Theorem: Evaluation for FO is PSPACE-complete (combined c.)
Evaluation pb for FO is PSPACE-complete (combined complexity)

PSPACE-complete problem: **QBF**

(satisfaction of Quantified Boolean Formulas)

QBF = a boolean formula with quantification over the truth values (T,F)

\[ \exists p \forall q . (p \lor \neg q) \] where p,q range over \{T,F\}

Theorem: Evaluation for FO is PSPACE-complete (combined c.)

Polynomial reduction \textbf{QBF} \sim FO:

1. Given \( \psi \in \text{QBF} \),
   let \( \psi'(x) \) be the replacement of each ‘p’ with ‘p=x’ in \( \psi \).

2. Note: \( \exists x \psi' \) holds in a 2-element graph iff \( \psi \) is QBF-satisfiable

3. Test if \( G \vDash \psi' \) for \( G=(\{v,v\}',\{\}) \)
Evaluation pb for FO is PSPACE-complete (combined complexity)

PSPACE-complete problem: **QBF**

(satisfaction of Quantified Boolean Formulas)

QBF = a boolean formula with quantification over the truth values (T,F)

\[ \exists p \forall q . (p \lor \neg q) \] where p,q range over \{T,F\}

Theorem: Evaluation for FO is PSPACE-complete (combined c.)

Polynomial reduction **QBF ~ FO** :

\[ \psi'(x) = \exists p \forall q . ( (p=x) \lor \neg (q=x) ) \]

1. Given \( \psi \in \text{QBF} \), let \( \psi'(x) \) be the replacement of each ‘p’ with ‘p=x’ in \( \psi \).

2. Note: \( \exists x \psi' \) holds in a 2-element graph iff \( \psi \) is QBF-satisfiable

3. Test if \( G \models \psi' \) for \( G = (\{v,v'\},\{\}) \)
Evaluation pb for FO is PSPACE-complete

**PSPACE-complete problem:** \( \text{QBF} \)

(satisfaction of Quantified Boolean Formulas)

\( \text{QBF} = \) a boolean formula with quantification over the truth values (T,F)

\[ \exists p \forall q \cdot (p \lor \neg q) \quad \text{where} \quad p, q \text{ range over } \{T,F\} \]

**Theorem:** Evaluation for FO is PSPACE-complete (combined c.)

Polynomial reduction \( \text{QBF} \sim \text{FO} : \)

1. Given \( \psi \in \text{QBF} \),
   let \( \psi'(x) \) be the replacement of each ‘p’ with ‘p=x’ in \( \psi \).

2. Note: \( \exists x \psi' \) holds in a 2-element graph iff \( \psi \) is QBF-satisfiable

3. Test if \( G \models \psi' \) for \( G=\{v,v'\},\{\}\)

(freely available at http://webdam.inria.fr/Alice/)

Chapters 1, 2, 3