Logical foundations of databases

Diego Figueira

Gabriele Puppis

CNRS     LaBRI
Recap

- **Relational model** (tables)

- **Relational Algebra** (union, product, difference, selection, projection)

- **SQL** (SELECT ... FROM ... WHERE ...)

- **RA ≈ basic SQL**

- **First-order logic** (syntax, semantics)

- **Expressiveness**: FO =* RA
Formulas as queries

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

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

[E.F. Codd 1972]
Formulas as queries

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

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

\[
RA = ^* FO \\
\text{How} = \text{What}
\]

[E.F. Codd 1972]
FO can serve as a **declarative** query language on relational databases: we express the properties of the answer

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

**RA** =* FO

**How** = **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

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

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

...
Formulas as queries

$\mathbf{FO} \subseteq \mathbf{RA}$ does not hold in general!
Formulas as queries

FO ⊆ RA does not hold in general!

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

FO ∉ RA

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

\[ \text{FO} \notin \text{RA} \]

“the complement of R” \[ \notin \text{RA} \] \[ \in \text{FO} : \neg R(x) \]

\[ \Rightarrow \text{We restrict variables to range over active domain} \]
Formulas as queries

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

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

(elements in the relations)

\[ \Rightarrow \text{We restrict variables to range over active domain} \]

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

$FO \not\subseteq RA$

"the complement of $R$" $\not\in RA$ $\in FO : \neg R(x)$

$\Leftarrow$ We restrict variables to range over active domain

$\forall x \in RA \exists y E(y,x)$

$\phi_1(x) = \forall y E(y,x)$

$\phi_1(G) = \{v_2\}$

$\phi_2(x,y) = \neg E(x,y)$

$\phi_2(G) = \{(v_1,v_1),(v_3,v_1),(v_2,v_3)\}$

$G = \{v_1,v_2,v_3,v_4\}$

$FO^{act}$

$= FO$ restricted to active domain
First-order logic restricted to active domain

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

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

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

$G \models_\alpha \phi \land \psi \text{ iff } G \models_\alpha \phi \text{ and } G \models_\alpha \psi$

$G \models_\alpha \neg \phi \text{ iff it is not true that } G \models_\alpha \phi$

$G \models_\alpha x = y \text{ iff } \alpha(x) = \alpha(y)$

$G \models_\alpha E(x, y) \text{ iff } (\alpha(x), \alpha(y)) \in E$

$\text{ACT}(G) = \{v \mid \text{ for some } v' : (v, v') \in E \text{ or } (v', v) \in E\}$
First-order logic restricted to active domain

\[ FO^{\text{act}} \subseteq RA \]
First-order logic restricted to active domain

\[ \text{FO}^{\text{act}} \subseteq \text{RA} \]

Assume:

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

\[ \exists x_1 \exists x_2 \neg \exists x_3 \exists x_4 \ (E(x_1, x_3) \land \neg E(x_4, x_2)) \lor (x_1 = x_3) \]
First-order logic restricted to active domain

\[ \text{FO}^{\text{act}} \subseteq \text{RA} \]

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

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

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

- \( (R(x_{i_t},\ldots,x_{i_t}))^+ \sim R \)
- \( (x_i = x_j)^+ \sim \sigma_{\{i=j\}}(\text{Adom} \times \cdots \times \text{Adom}) \)
- \( (\psi_1 \land \psi_2)^+ \sim \psi_1^+ \land \psi_2^+ \)
- \( (\neg \psi)^+ \sim \text{Adom} \times \cdots \times \text{Adom} \setminus \psi^+ \)
- \( (\exists x_i \phi(x_{i_1},\ldots,x_{i_t}))^+ \sim \pi_{\{i_1,\ldots,i_t\}\setminus\{i\}}(\phi^+) \)

Translation
First-order logic restricted to active domain

\[ \mathcal{FO}^{\text{act}} \subseteq \mathcal{RA} \]

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

2. \( \phi \) has \( n \) variables

Assume:

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

\( \text{Adom} \) = RA expression for active domain

\[ \text{Adom} = \pi_1(E) \cup \pi_2(E) \]

- \( (R(x_{i1},\ldots,x_{it}))^+ \sim R \)

- \( (x_i = x_j)^+ \sim \sigma_{i=j}(\text{Adom} \times \cdots \times \text{Adom}) \)

- \( (\psi_1 \land \psi_2)^+ \sim \psi_1^+ \land \psi_2^+ \)

- \( (\neg \psi)^+ \sim \text{Adom} \times \cdots \times \text{Adom} \setminus \psi^+ \)

- \( (\exists x_i \phi(x_{i1},\ldots,x_{it}))^+ \sim \pi_{\{i_1,\ldots,i_t\}\setminus\{i\}}(\phi^+) \)
First-order logic restricted to active domain

\( \text{FO}^{\text{act}} \subseteq \text{RA} \)

Assume:

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

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

\( \text{Adom} \) = RA expression for active domain = "\( \pi \)

- \( (R(x_{i_1}, \ldots, x_{i_t}))^+ \sim R \)
- \( (x_i = x_j)^+ \sim \sigma_{i=j} (\text{Adom} \times \cdots \times \text{Adom}) \)
- \( (\psi_1 \land \psi_2)^+ \sim \psi_1^+ \land \psi_2^+ \)
- \( (\neg \psi)^+ \sim \text{Adom} \times \cdots \times \text{Adom} \setminus \psi^+ \)
- \( (\exists x_i \phi(x_{i_1}, \ldots, x_{i_t}))^+ \sim \pi_{i_1, \ldots, i_t \setminus \{i\}}^+ (\phi^+ ) \)
First-order logic restricted to active domain

\[ \text{FO}^{\text{act}} \subseteq \text{RA} \]

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

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

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

- \((R(x_{i_1},...,x_{i_t}))^+ \sim R \)
- \((x_i = x_j)^+ \sim \sigma_{i=j}(\text{Adom} \times \cdots \times \text{Adom})\)
- \((\psi_1 \land \psi_2)^+ \sim \psi_1^+ \cap \psi_2^+ \)
- \((\neg \psi)^+ \sim \text{Adom} \times \cdots \times \text{Adom} \setminus \psi^+ \)
- \((\exists x_i \phi(x_{i_1},...,x_{i_t}))^+ \sim \pi_{\{i_1,...,i_t\}\setminus\{i\}}(\phi^+) \)

\[
A \cap B = ((A \cup B) \setminus (A \setminus B)) \setminus (B \setminus A)
\]
First-order logic restricted to active domain

\[ \text{FO}^{\text{act}} \subseteq \text{RA} \]

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

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

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

- \( (R(x_{i_1},\ldots,x_{i_t}))^+ \rightarrow R \)
- \( (x_i = x_j)^+ \rightarrow \sigma_{\{i=j\}}(\text{Adom} \times \cdots \times \text{Adom}) \)
- \( \psi_1 \land \psi_2)^+ \rightarrow \psi_1^+ \land \psi_2^+ \)
- \( \neg \psi)^+ \rightarrow \text{Adom} \times \cdots \times \text{Adom} \setminus \psi^+ \)
- \( (\exists x_i \phi(x_{i_1},\ldots,x_{i_t}))^+ \rightarrow \pi_{\{i_1,\ldots,i_t\}\setminus\{i\}}(\phi^+) \)
Corollary

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

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

Remember: $R_1, R_2$ are binary

Answer: $\exists x_1, x_3, x_4 \left( R_1(x_1, x_2) \land R_2(x_3, x_4) \land x_1 = x_3 \right)$

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)

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

**Remember**: $R_1,R_2$ are binary

**Answer**: $\exists x_1,x_3,x_4 \ (R_1(x_1,x_2) \land R_2(x_3,x_4) \land x_1 = x_3)$

**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)

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

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

FO = RA = SQL

Programming language
\[ \text{FO} = \text{RA} = \text{SQL} \]

- Logic over active domain
- Algebra on finite domains
- Programming language very basic
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?
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)
Algorithmic problems for query languages

**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)

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

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

How **hard** is it to mechanise?  \(\sim\) complexity classes
What can be *mechanized*? $\sim$ decidable/undecidable

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

What can be mechanized? ~ decidable/undecidable

How hard is it to mechanise? ~ complexity classes

usage of resources:
• time
• memory
Complexity theory

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

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

\[\text{Algorithm } \textbf{Alg} \text{ is } \textit{TIME}\text{-bounded}\]

by a function \(f: \mathbb{N} \to \mathbb{N}\) if

\[\text{\textbf{Alg}(input)} \text{ uses less than } f(|\text{input}|) \text{ units of TIME.}\]
Complexity theory

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

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

- usage of resources: • time
  • memory

Algorithm **Alg** is **TIME**-bounded by a function \( f: \mathbb{N} \rightarrow \mathbb{N} \) if

\[
\text{Alg}(\text{input}) \text{ uses less than } f(|\text{input}|) \text{ units of TIME.}
\]
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 \(\text{TIME}\)-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 \(\text{SPACE}\).
What can be **mechanized**?  \(\sim\) **decidable/undecidable**

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

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**.

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

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

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

usage of resources:
- time
- memory

**SPACE**
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**.

**LOGSPACE** \(\subseteq\) **PTIME** \(\subseteq\) **PSPACE** \(\subseteq\) **EXPTIME** \(\subseteq\) \ldots

**SPACE**-bounded by a polynomial

**SPACE**-bounded by \(\log(n)\)

**TIME**-bounded by a polynomial
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 $\phi(x_1, ..., x_n)$, 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 $\phi(x_1, ..., x_n)$, a graph $G$, and a binding $\alpha$, does $G \models_\alpha \phi$?

**DECIDABLE $\leftrightarrow$ 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 $\leftrightarrow$ 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 $\alpha$?
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 **$\rightarrow \text{foundations of the database industry}$

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

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

**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 **$\rightarrow \text{by reduction to the satisfiability problem}$
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 $\not\models$ and $\models_{\text{finite}}$ [Trakhtenbrot ’50]
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 $\vDash_{\text{finite}}$ [Trakhtenbrot ’50]

Proof: By reduction from the Domino (aka Tiling) problem.
Algorithmic problems for FO

**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 \( \vDash_{\text{finite}} \) [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: 

Domino
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:

- (head is elsewhere, symbol is not modified)
- (head is here, symbol is rewritten, head moves right)
- (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:

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

Domino - Why is it undecidable?

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

- **Initial configuration**
  - (head is elsewhere, symbol is not modified)
  - (head is here, symbol is rewritten, head moves right)
  - (head is here, symbol is rewritten, head moves left)

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

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

$D_x(\ x\ )$

for each domino $\square$
Domino $\rightarrow$ Sat-FO (domino has a solution iff $\phi$ 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

   $D_a(x)$

   for each domino $\Box$

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 

\[ D_x(x) \]

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 \( \text{(Idem vertically)} \)

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 $G \models_{\alpha} \phi$?

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
Equivalence problem: Given FO formulae $\phi, \psi$, is

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

for all graphs $G$ and bindings $\alpha$?

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

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

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

**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 from the satisfiability problem
Algorithmic problems for FO

φ is satisfiable iff φ is not equivalent to ⊥

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

Actually, there are reductions in both senses:
φ\((x_1,\ldots,x_n)\) and \(ψ(y_1,\ldots,y_m)\) are equivalent iff

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

Equivalence problem: Given FO formulae \(φ,ψ\), is

\[ G \models_α φ \iff G \models_α ψ \]

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

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

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

**DECIDABLE ⇔ 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 ⇔ 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 ⇔ by reduction to the satisfiability problem**
Evaluation problem for FO

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

Output: \[ G \Vdash_\alpha \phi \]
Evaluation problem for FO

\[
\begin{align*}
\phi(x_1, \ldots, x_n) \\
G = (V, E) \\
\alpha = \{x_1, \ldots, x_n\} \rightarrow V
\end{align*}
\]

Input: \( G = (V, E) \)  \hspace{1cm}  Output:  \( G \models_\alpha \phi ? \)

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

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

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

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)

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: \[
\begin{align*}
\phi(x_1, \ldots, x_n) \\
G = (V, E) \\
\alpha = \{x_1, \ldots, x_n\} \rightarrow V
\end{align*}
\]

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 \( G \not\models_\alpha \psi \)

- 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 \( G \not\models_{\alpha'} \psi \).
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 \( \text{iff} \ (\alpha(x_i),\alpha(x_j)) \in E \)

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

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

- If \( \phi(x_1,\ldots,x_n) = \exists y . \psi(x_1,\ldots,x_n,y) \):
  answer YES \( \text{iff} \) 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 \( (\alpha(x_i),\alpha(x_j)) \in E \)

- 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: \[
\begin{align*}
\phi(x_1,\ldots,x_n) \\
G = (V,E) \\
\alpha = \{x_1,\ldots,x_n\} \rightarrow V
\end{align*}
\]

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 \(\phi(x_1,\ldots,x_n) = \exists y \cdot \psi(x_1,\ldots,x_n,y):\)  
  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 \( \phi(x_1,\ldots,x_n) = E(x_i,x_j) \):
  \[ \text{answer YES} \iff (\alpha(x_i),\alpha(x_j)) \in E \]

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

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

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

Question:
How much space does it take?

\[ \text{use 4 pointers } \rightarrow \text{LOGSPACE} \]

\[ \Rightarrow \text{MAX( SPACE}(G \models_\alpha \psi)), \text{ SPACE}(G \models_\alpha \psi')) \]

\[ \Rightarrow \text{SPACE}(G \models_\alpha \psi)) \]
Evaluation problem for FO

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

Input: 

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 \(G \models_\alpha \psi'\)  
  \(\Rightarrow\) MAX( SPACE(\(G \models_\alpha \psi\)), SPACE(\(G \models_\alpha \psi'\)) )

• If \(\phi(x_1, \ldots, x_n) = \neg \psi(x_1, \ldots, x_n)\):  
  answer NO iff \(G \not\models_\alpha \psi\)  
  \(\Rightarrow\) SPACE(\(G \models_\alpha \psi\))

• 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 \(G \models_{\alpha'} \psi\)  
  \(\Rightarrow\) 2 \cdot \log(|G|) + SPACE(\(G \models_{\alpha'} \psi\))

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

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

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?

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

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

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 \( G \not\models_\alpha \psi \)

• 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 pb for FO is PSPACE-complete

PSPACE-complete problem: **QBF**
(satisfaction of Quantified Boolean Formulas)

QBF = a boolean formula with quantification over the truth values (T,F)
Evaluation pb for FO is PSPACE-complete

PSPACE-complete problem: $\text{QBF}$

(satisfaction of Quantified Boolean Formulas)

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

$\exists p \forall q . (p \lor \neg q)$  where $p,q$ range over $\{T,F\}$
Evaluation pb for FO is PSPACE-complete

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.)
Evaluation pb for FO is PSPACE-complete

PSPACE-complete problem: QBF
(satisfaction of Quantified Boolean Formulas)

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

∃p ∀q . (p ∨ ¬q) \hspace{1cm} \text{where } p,q \text{ range over \{T,F\}}

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

Polynomial reduction QBF ∼ FO:

1. Given ψ ∈ QBF, let ψ'(x) be the replacement of each ‘p’ with ‘p=x’ in ψ.

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

PSPACE-complete problem: **QBF**

(satisfaction of Quantified Boolean Formulas)

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

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

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

Polynomial reduction **QBF \sim 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 \( G \vDash \psi' \) for \( G=\{v,v\}',\{\} \)
Evaluation pb for FO is PSPACE-complete

PSPACE-complete problem: **QBF**
(satisfaction of Quantified Boolean Formulas)

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

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

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

Polynomial reduction **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 \( G \models \psi' \) for \( G=(\{v,v'\},\{\}) \)
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: • query +
Problem: Usual scenario in database

Input: • query +
Problem: Usual scenario in database

Input: • query + database

But we don’t distinguish this in the analysis:

\[
\text{TIME}(2|\text{query}| + |\text{data}|) = \text{TIME}(|\text{query}| + 2|\text{data}|)
\]
Query and data play very different roles.

Separation of concerns: How the resources grow with respect to

- the size of the data
- the query size

[Oneil, 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, Query, and Data complexities**

**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 $|\text{data}|$

$O(2^{|\text{query}|} + |\text{data}|)$ is exponential in combined complexity

Exponential in query complexity

Linear in data complexity

$O(|\text{query}| + 2^{|\text{data}|})$ is exponential in combined complexity

Exponential in query complexity

Exponential in data complexity
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
Trading expressiveness for efficiency

Alternation of quantifiers significantly affects complexity (recall that evaluation of QBF is PSPACE-complete: \( \forall x \exists y \forall z \exists w \ldots \phi \)).

What happens if we disallow \( \forall \) and \( \neg \)?
The class NP

\[
\text{LOGSPACE} \subseteq \text{PTIME} \subseteq \text{PSPACE} \subseteq \text{EXPTIME}
\]
The class NP

\[
\text{LOGSPACE} \subseteq \text{PTIME} \subseteq \text{NP} \subseteq \text{PSPACE} \subseteq \text{EXPTIME}
\]

**NP** = Problems whose solutions can be witnessed by a *certificate* to be guessed and checked in *polynomial time* (e.g. a colouring)
The class NP

$$\text{LOGSPACE} \subseteq \text{PTIME} \subseteq \text{NP} \subseteq \text{PSPACE} \subseteq \text{EXPTIME}$$

\[ \text{NP} = \text{Problems whose solutions can be witnessed by a certificate to be guessed and checked in polynomial time} \ (e.g. \ a \ colouring) \]

Examples:

- **3-COLORABILITY**: Given a graph \( G \), can we assign a colour from \{R, G, B\} to each node so that adjacent nodes have always different colours?

- **SAT**: Given a propositional formula, e.g. \((p \lor \neg q \lor r) \land (\neg p \lor s) \land (\neg s \lor \neg p)\), can we assign a truth value to each variable so that the formula becomes true?

- **MONEY-CHANGE**: Given an amount of money \( A \) and a set of coins \{\( B_1, ..., B_n \}\), can we find a subset \( S \subseteq \{B_1, ..., B_n\} \) such that \( \sum S = A \)?
The class NP

\[ \text{LOGSPACE} \subseteq \text{PTIME} \subseteq \text{NP} \subseteq \text{PSPACE} \subseteq \text{EXPTIME} \]

\textbf{NP} = Problems whose solutions can be witnessed by a \emph{certificate} to be guessed and checked in \emph{polynomial time} (e.g. a colouring)
LOGSPACE ⊆ PTIME ⊆ NP ⊆ PSPACE ⊆ EXPTIME

NP = Problems whose solutions can be witnessed by a certificate to be guessed and checked in polynomial time (e.g. a colouring)
The class NP

\[ \text{LOGSPACE} \subseteq \text{PTIME} \subseteq \text{NP} \subseteq \text{PSPACE} \subseteq \text{EXPTIME} \]

**NP** = Problems whose solutions can be witnessed by a *certificate* to be guessed and checked in *polynomial time* (e.g. a colouring)
The class NP

\[ \text{LOGSPACE} \subseteq \text{PTIME} \subseteq \text{NP} \subseteq \text{PSPACE} \subseteq \text{EXPTIME} \]

\text{NP} = \text{Problems whose solutions can be witnessed by a certificate to be guessed and checked in polynomial time} \quad (\text{e.g. a colouring})
The class \( \text{NP} \)

\[
\text{LOGSPACE} \subseteq \text{PTIME} \subseteq \text{NP} \subseteq \text{PSPACE} \subseteq \text{EXPTIME}
\]

\( \text{NP} = \) Problems whose solutions can be witnessed by a certificate to be guessed and checked in polynomial time (e.g. a colouring)

Non-deterministic transitions

Many paths, each has length bounded by a polynomial
The class NP

\[
\text{LOGSPACE} \subseteq \text{PTIME} \subseteq \text{NP} \subseteq \text{PSPACE} \subseteq \text{EXPTIME}
\]

**NP** = Problems whose solutions can be witnessed by a *certificate* to be guessed and checked in *polynomial time* (e.g. a colouring)

Non-deterministic transitions

Many paths, each has length bounded by a *polynomial*

A solution exists if there is at least a *successful* path.
Consider: \textbf{Positive FO} = FO without \(\forall, \neg\)

E.g. \(\phi = \exists x \exists y \exists z \cdot (E(x, y) \lor E(y, z)) \land (y=z \lor E(x, z))\)

What is the complexity of evaluating Positive FO on graphs?
Question

Consider:  \textbf{Positive FO = FO without }\forall,\neg

E.g.  \( \phi = \exists x \exists y \exists z \cdot (E(x, y) \lor E(y, z)) \land (y=z \lor E(x, z)) \)

What is the complexity of evaluating Positive FO on graphs?

Solution

This is in NP:  Given \( \phi \) and \( G=(V, E) \) it suffices to guess a binding \( \alpha : \{x, y, z, \ldots \} \to V \) and then verify that the formula holds.
Conjunctive Queries

Def. \[ CQ = FO \text{ without } \forall, \neg, \lor \]

Eg: \[ \phi(x, y) = \exists z. (\text{Parent}(x, z) \land \text{Parent}(z, y)) \]

Usual notation: “Grandparent(X,Y) : – Parent(X,Z), Parent(Z,Y)”
Conjunctive Queries

Def.

CQ = FO without ∀, ¬, ∨

Normal form: “∀x1, ..., xn. φ(x1, ..., xn)"

quantifier-free and no equalities!

Eg: φ(x, y) = ∃z. (Parent(x, z) ∧ Parent(z, y))

Usual notation: “Grandparent(X,Y) : – Parent(X,Z), Parent(Z,Y)”
Conjunctive Queries

Def. \[ CQ = \text{FO without } \forall, \neg, \lor \]

Normal form: \( \exists x_1, \ldots, x_n. \phi(x_1, \ldots, x_n) \)
quantifier-free and no equalities!

Eg:\( \phi(x, y) = \exists z. (\text{Parent}(x, z) \land \text{Parent}(z, y)) \)

Usual notation: “Grandparent(X,Y) : – Parent(X,Z), Parent(Z,Y)”

It corresponds to positive
“SELECT-FROM-WHERE” SQL queries
Select ...
From ...
Where Z
no negation or disjunction

It corresponds to “\( \pi-\sigma-\times \)” RA queries
\[ \pi_X(\sigma_Z(R_1 \times \cdots \times R_n)) \]
no negation

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

Chapters 1, 2, 3