day 2

ESSLLI 2016
Bolzano, Italy

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

$$\begin{aligned} \text{Tables} &= \text{Relations} \\ \text{Rows} &= \text{Tuples} \\ \text{Queries} &= \text{Formulas} \end{aligned}$$

[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

$$\mathrm{RA} =^{*} \mathrm{FO}$$

$$\mathit{How} = \mathit{What}$$

[E.F. Codd 1972]

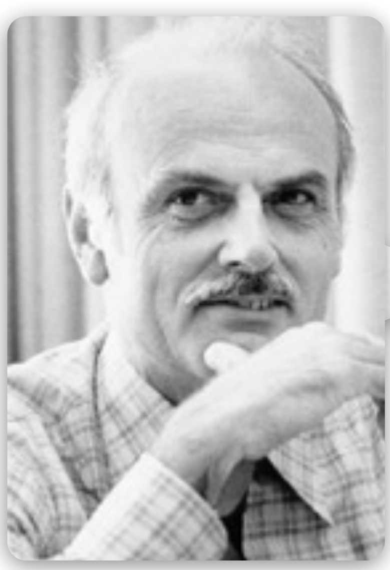
Formulas as queries

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

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

$\boxed{\mathrm{RA} \subseteq \mathrm{FO}}$

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

Formulas as queries

$\boxed{\mathrm{FO} \subseteq \mathrm{RA}}$ does not hold in general!

Formulas as queries

$\boxed{\mathrm{FO} \subseteq \mathrm{RA}}$ does not hold in general!

"the complement of R" $\begin{array}{l} \notin \mathrm{RA} \\ \in \mathrm{FO} : \neg R(x) \end{array}$

Formulas as queries

$\boxed{\text{FO} \not\subseteq \text{RA}}$

$$\text{“the complement of R”} \begin{array}{l} \notin \ \text{RA} \\ \in \ \text{FO} : \ \neg R(x) \end{array}$$

Formulas as queries

$\boxed{\text{FO} \not\subseteq \text{RA}}$

$$\text{“the complement of R”} \begin{array}{l} \notin \ \text{RA} \\ \in \ \text{FO}: \ \neg R(x) \end{array}$$

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

Formulas as queries

$\boxed{\text{FO} \not\subseteq \text{RA}}$

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

elements in the relations

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

FO$^{\text{act}}$

=

FO restricted
to active domain

Formulas as queries

$\boxed{\text{FO} \not\subseteq \text{RA}}$

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

elements in the relations

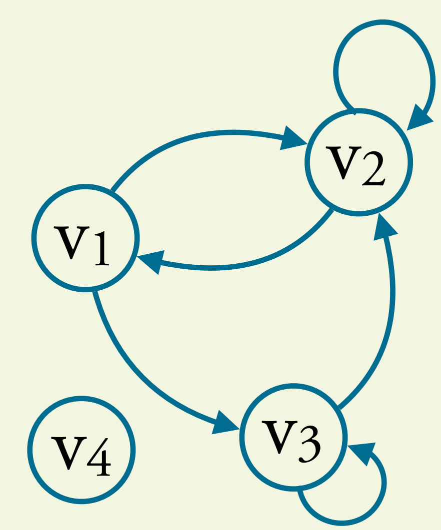
⇝ We restrict variables to range over **active domain**

FO^{act}

=

FO restricted
to active domain

$\phi_1(x) = \forall y\, E(y,x)$
$\phi_1(G) = \{v_2\}$

G = 

$\phi_2(x,y) = \neg E(x,y)$
$\phi_2(G) = \{(v_1,v_1),(v_3,v_1),(v_2,v_3)\}$

First-order logic restricted to active domain

Formal Semantics of FO^{act}

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

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

$G \models_\alpha \phi \wedge \psi$ iff $G \models_\alpha \phi$ and $G \models_\alpha \psi$

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

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

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

$ACT(G) = \{v \mid$ **for some** v': $(v,v') \in E$ or $(v',v) \in E\}$

First-order logic restricted to active domain

$\boxed{\mathrm{FO}^{\mathrm{act}} \subseteq \mathrm{RA}}$

First-order logic restricted to active domain

$FO^{act} \subseteq RA$

Assume:

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

$$\exists x_1\, \exists x_2\, \neg\exists x_3\, \exists x_4\, .\, (\, E(x_1,x_3) \wedge \neg E(x_4,x_2)\,) \vee (x_1 = x_3)$$

First-order logic restricted to active domain

$\mathrm{FO}^{\mathrm{act}} \subseteq \mathrm{RA}$

Assume:

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

$$\exists x_1\, \exists x_2\, \neg\exists x_3\, \exists x_4 \,.\, (\, E(x_1,x_3) \wedge \neg E(x_4,x_2)\,) \vee (x_1{=}x_3)$$

Adom = RA expression for active domain = "$\pi_1(E) \cup \pi_2(E)$"

Translation

- $(R(x_{i_1},\ldots,x_{i_t}))^+ \rightsquigarrow R$
- $(x_i = x_j)^+ \rightsquigarrow \sigma_{\{i=j\}}(\,\mathbf{Adom} \times \cdots \times \mathbf{Adom}\,)$
- $(\psi_1 \wedge \psi_2)^+ \rightsquigarrow \psi_1^+ \cap \psi_2^+$
- $(\neg\psi)^+ \rightsquigarrow \mathbf{Adom} \times \cdots \times \mathbf{Adom} \setminus \psi^+$
- $(\,\exists x_i\, \phi(x_{i_1},\ldots,x_{i_t})\,)^+ \rightsquigarrow \pi_{\{i_1,\ldots,i_t\}\setminus\{i\}}(\,\phi^+\,)$

First-order logic restricted to active domain

$\mathrm{FO}^{act} \subseteq \mathrm{RA}$

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

$\pi_{\{1,\ldots,n\}}(\sigma_{\{i_1=n+1,\ldots,i_t=n+t\}}(\mathrm{Adom}^n \times R))$

$\exists x_1 \exists x_2 \neg\exists x_3 \exists x_4 \,.\, (E(x_1,$

Adom = RA expression for acti ... $m = \pi_1(E) \cup \pi_2(E)$

Translation

- $(R(x_{i_1},\ldots,x_{i_t}))^+ \rightsquigarrow R$
- $(x_i = x_j)^+ \rightsquigarrow \sigma_{\{i=j\}}(\mathrm{Adom} \times \cdots \times \mathrm{Adom})$
- $(\psi_1 \wedge \psi_2)^+ \rightsquigarrow \psi_1^+ \cap \psi_2^+$
- $(\neg\psi)^+ \rightsquigarrow \mathrm{Adom} \times \cdots \times \mathrm{Adom} \setminus \psi^+$
- $(\exists x_i\, \phi(x_{i_1},\ldots,x_{i_t}))^+ \rightsquigarrow \pi_{\{i_1,\ldots,i_t\}\setminus\{i\}}(\phi^+)$

First-order logic restricted to active domain

$\mathrm{FO}^{act} \subseteq \mathrm{RA}$

Assume:

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

$$\exists x_1\, \exists x_2\, \neg\exists x_3\, \exists x_4\, .\, (\, E(x_1,x_3) \wedge \neg E(x_4,x_2)\,) \vee (x_1 = x_3)$$

Adom = RA expression for active domain = “π (

Adom^n

Translation

- $(R(x_{i_1},\ldots,x_{i_t}))^+ \rightsquigarrow R$
- $(x_i = x_j)^+ \rightsquigarrow \sigma_{\{i=j\}}(\,\mathbf{Adom} \times \cdots \times \mathbf{Adom}\,)$
- $(\psi_1 \wedge \psi_2)^+ \rightsquigarrow \psi_1^+ \cap \psi_2^+$
- $(\neg\psi)^+ \rightsquigarrow \mathbf{Adom} \times \cdots \times \mathbf{Adom} \setminus \psi^+$
- $(\,\exists x_i\, \phi(x_{i_1},\ldots,x_{i_t})\,)^+ \rightsquigarrow \pi_{\{i_1,\ldots,i_t\}\setminus\{i\}}(\,\phi^+\,)$

First-order logic restricted to active domain

$\mathrm{FO}^{\mathrm{act}} \subseteq \mathrm{RA}$

Assume:

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

$$\exists x_1\, \exists x_2\, \neg\exists x_3\, \exists x_4 \,.\, (\, E(x_1,x_3) \wedge \neg E(x_4,x_2)\,) \vee (x_1=x_3)$$

Adom = RA expression for active domain = "$\pi_1(E) \cup \pi_2(E)$"

Translation

- $(R(x_{i_1},\dots,x_{i_t}))^+ \rightsquigarrow R$
- $(x_i = x_j)^+ \rightsquigarrow \sigma_{\{i=j\}}(\,\textbf{Adom} \times$
- $(\psi_1 \wedge \psi_2)^+ \rightsquigarrow \psi_1^+ \cap \psi_2^+$
- $(\neg\psi)^+ \rightsquigarrow \textbf{Adom} \times \cdots \times \textbf{Adom} \setminus \psi^+$
- $(\,\exists x_i\, \phi(x_{i_1},\dots,x_{i_t})\,)^+ \rightsquigarrow \pi_{\{i_1,\dots,i_t\}\setminus\{i\}}(\,\phi^+\,)$

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

First-order logic restricted to active domain

$\mathbf{FO}^{act} \subseteq \mathrm{RA}$

Assume:

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

$$\exists x_1\, \exists x_2\, \neg\exists x_3\, \exists x_4\, .\, (\, E(x_1,x_3) \wedge \neg E(x_4,x_2)\,) \vee (x_1=x_3)$$

Adom = RA expression for active domain = "$\pi_1(E) \cup \pi_2(E)$"

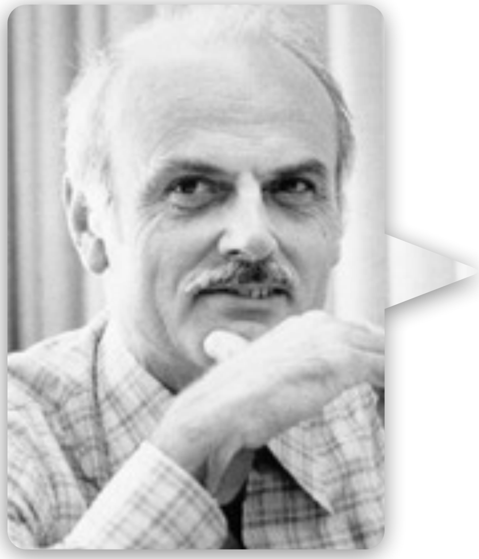
Translation

- $(R(x_{i_1},\ldots,x_{i_t}))^+ \rightsquigarrow R$
- $(x_i = x_j)^+ \rightsquigarrow \sigma_{\{i=j\}}(\, \mathbf{Adom} \times \cdots \times \mathbf{Ador}$

Adom^t if t is the arity of ψ^+

- $(\psi_1 \wedge \psi_2)^+ \rightsquigarrow \psi_1^+ \cap \psi_2^+$
- $(\neg\psi)^+ \rightsquigarrow \mathbf{Adom} \times \cdots \times \mathbf{Adom} \setminus \psi^+$
- $(\, \exists x_i\, \phi(x_{i_1},\ldots,x_{i_t})\,)^+ \rightsquigarrow \pi_{\{i_1,\ldots,i_t\}\setminus\{i\}}(\, \phi^+\,)$

Corollary



$\mathbf{FO}^{\mathrm{act}}$ 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 $\exists y,z \,.\, (R_1(x,y) \wedge R_1(y,z) \wedge\ x \neq z\,)$ expressed in RA?
Remember: The signature is the same as before (R_1,R_2 binary)

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

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

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

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

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

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

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

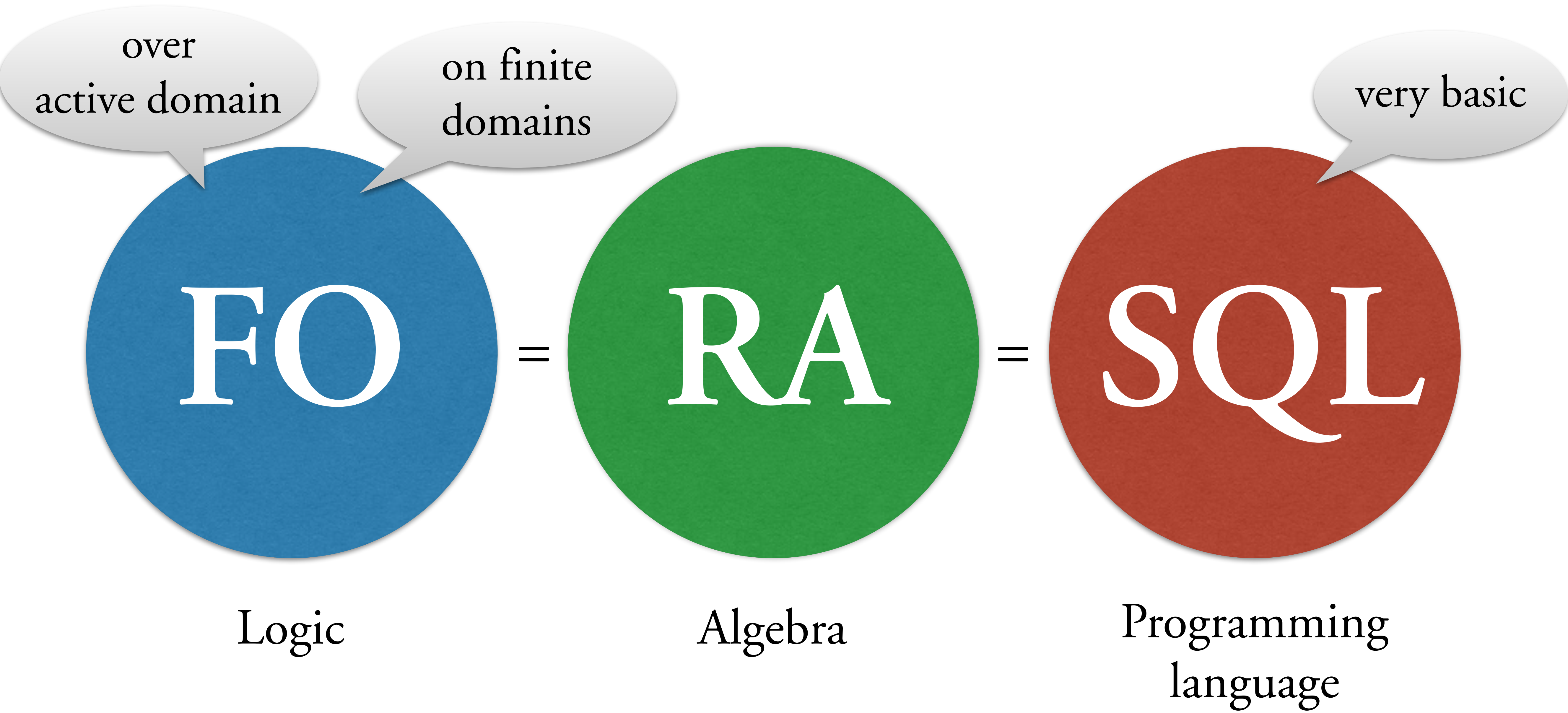
Answer: $\pi_1(\sigma_{\{2=3,1\neq 4\}}(R_1 \times R_1))$

FO = RA = SQL

Logic

Algebra

Programming language


over
active domain
on finite
domains
very basic
FO
=
RA
=
SQL
Logic
Algebra
Programming
language

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 $\mathbf{t} \in \mathbf{Q}(\mathbf{db})$?

⇝ How hard is it to retrieve data?

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

⇝ 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 $\mathbf{t} \in \mathbf{Q}(\mathbf{db})$?

⇝ How hard is it to retrieve data?

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

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

Equivalence problem: Given queries $\mathbf{Q}_1$, $\mathbf{Q}_2$, is

$$\mathbf{Q}_1(\mathbf{db}) = \mathbf{Q}_2(\mathbf{db})$$

for all database instances **db**?

⇝ Can we safely replace a query with another? (Query optimization)

Complexity theory

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

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

Complexity theory

H's 10th PCP
Domino K
$\vdots$

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

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

Complexity theory

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

Domino

H's 10th

PCP

K

$\vdots$

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

usage of resources:
- time
- memory

Complexity theory

Domino H's 10th PCP K ⋮

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

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

usage of resources:
- time
- memory

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

Complexity theory

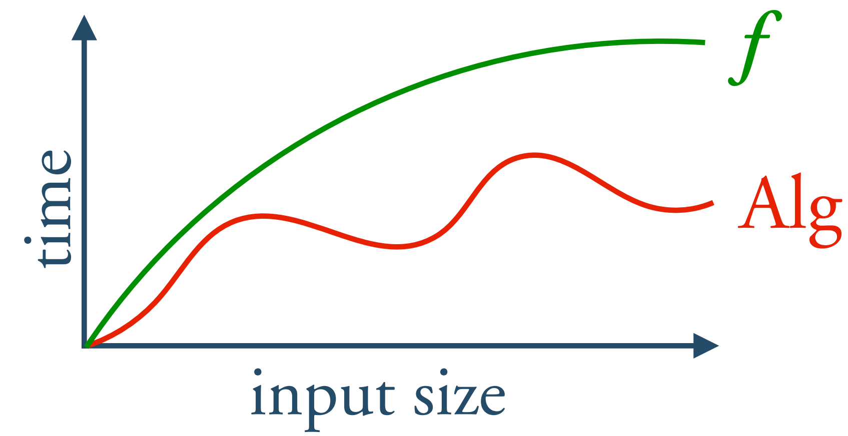
Domino H's 10th PCP K ⋮

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

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

usage of resources:
- time
- memory

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



Complexity theory

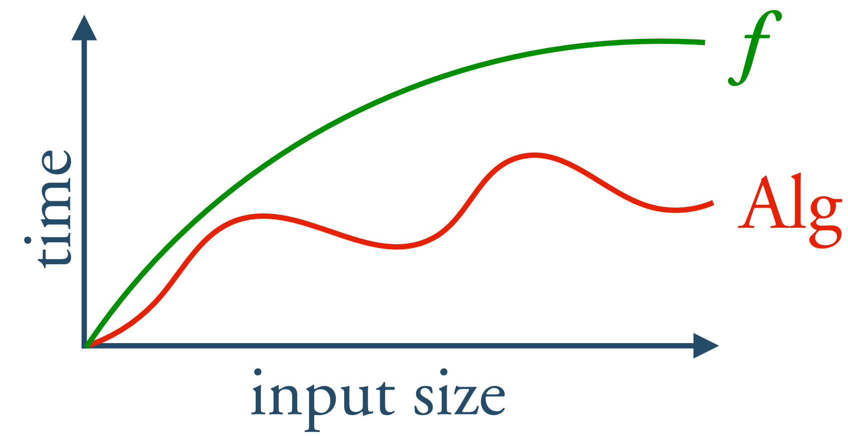
Domino H's 10th PCP K ⋮

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

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

usage of resources:
- time
- memory

Algorithm **Alg** is ~~TIME~~ SPACE-bounded
by a function $f : \mathrm{N} \longrightarrow \mathrm{N}$ if
Alg(*input*) uses less than $f(|\mathit{input}|)$ units of ~~TIME~~ SPACE.



Complexity theory

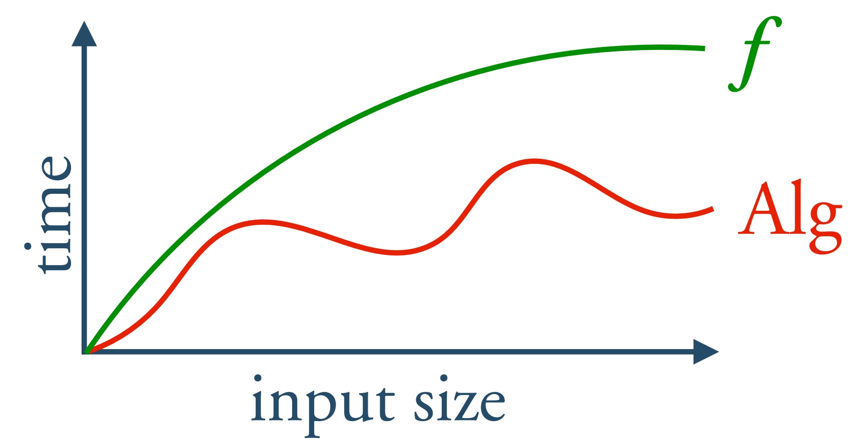
Domino H's 10th PCP K ⋮

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

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

usage of resources:
- time
- memory

Algorithm **Alg** is ~~TIME~~ SPACE-bounded by a function $f\colon \mathrm{N} \longrightarrow \mathrm{N}$ if **Alg**(input) uses less than $f(|\mathit{input}|)$ units of ~~TIME.~~ SPACE.



LOGSPACE $\subseteq$ PTIME $\subseteq$ PSPACE $\subseteq$ EXPTIME $\subseteq \cdots$

Complexity theory

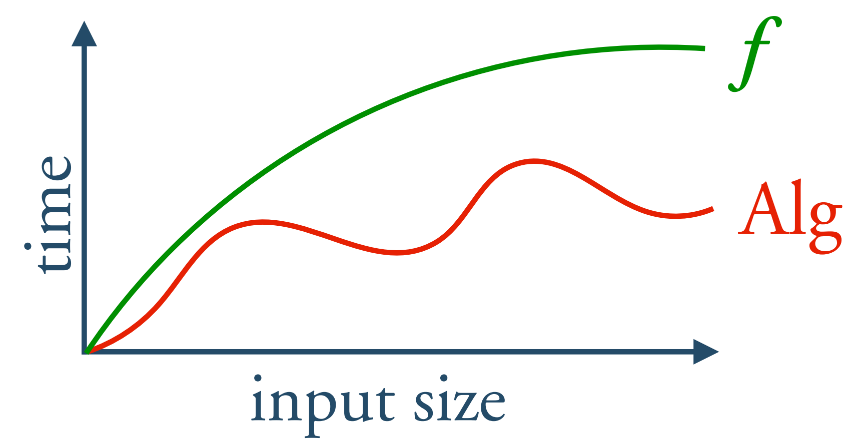
Domino H's 10th PCP K ⋮

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

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

usage of resources:
- time
- memory

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



LOGSPACE ⊆ PTIME ⊆ PSPACE ⊆ EXPTIME ⊆ · · ·

PTIME: TIME-bounded by a polynomial

PSPACE: SPACE-bounded by a polynomial

LOGSPACE: SPACE-bounded by log(n)

Algorithmic problems for FO

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

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

Equivalence problem: Given FO formulae ϕ,ψ, is

$$G \models_\alpha \phi \quad \text{iff} \quad G \models_\alpha \psi$$

for all graphs G and bindings α?

Algorithmic problems for FO

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

DECIDABLE ⇝ foundations of the database industry

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

Equivalence problem: Given FO formulae ϕ, ψ, is

$$G \models_\alpha \phi \quad \text{iff} \quad G \models_\alpha \psi$$

for all graphs G and bindings α?

Algorithmic problems for FO

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

DECIDABLE ⇝ foundations of the database industry

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

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

Equivalence problem: Given FO formulae ϕ,ψ, is

$$G \models_\alpha \phi \quad \text{iff} \quad G \models_\alpha \psi$$

for all graphs G and bindings α?

Algorithmic problems for FO

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

DECIDABLE $\rightsquigarrow$ foundations of the database industry

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

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

Equivalence problem: Given FO formulae ϕ,ψ, is

$$G \models_\alpha \phi \quad \text{iff} \quad G \models_\alpha \psi$$

for all graphs G and bindings α?

💀 UNDECIDABLE $\rightsquigarrow$ by reduction to the satisfiability problem

Algorithmic problems for FO

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

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

Algorithmic problems for FO

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

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

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

Algorithmic problems for FO

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

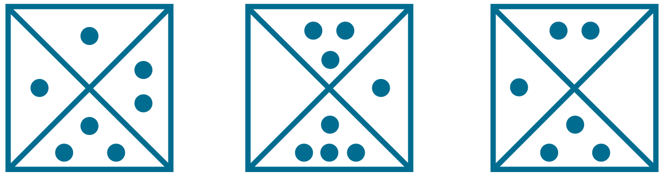
💀 UNDECIDABLE $\rightsquigarrow$ both for $\models$ and $\models_{finite}$ [Trakhtenbrot '50]

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).

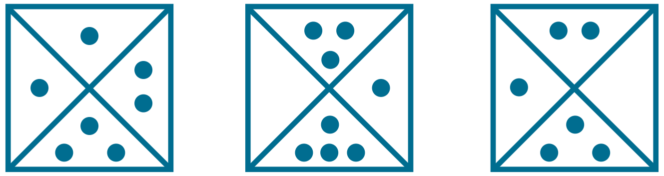
The (undecidable) Domino problem

Domino

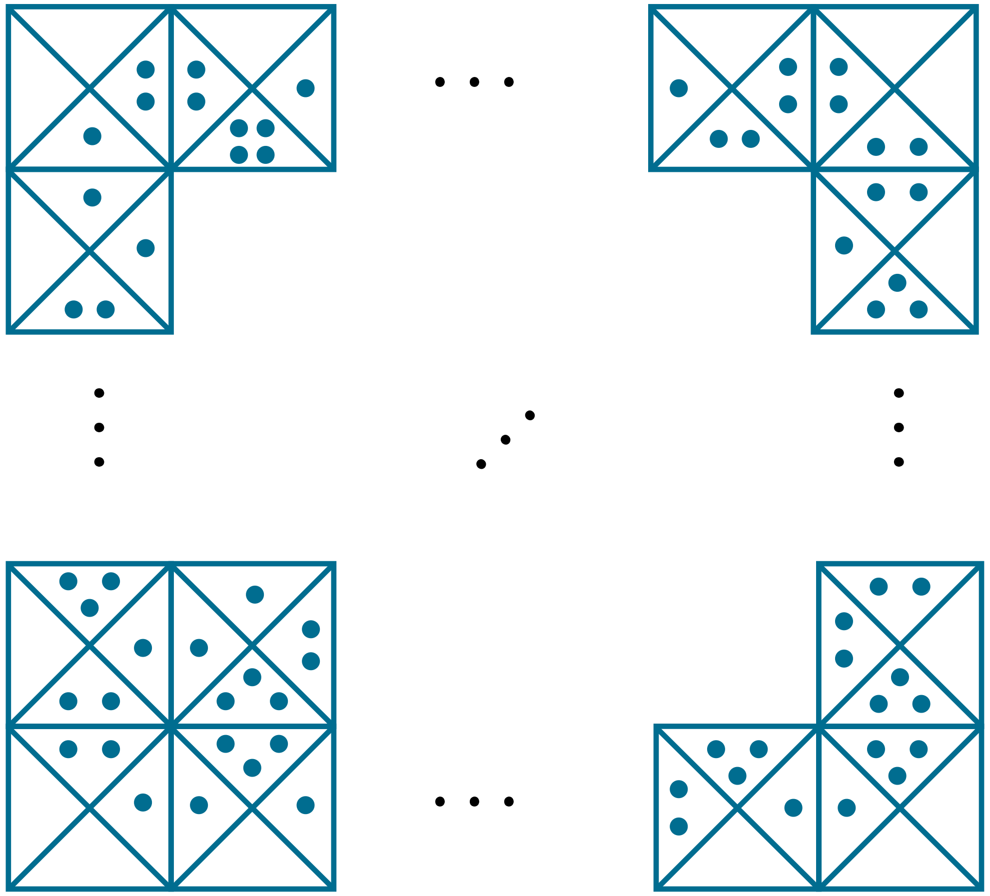
Input: 4-sided dominos: 

The (undecidable) Domino problem

Domino

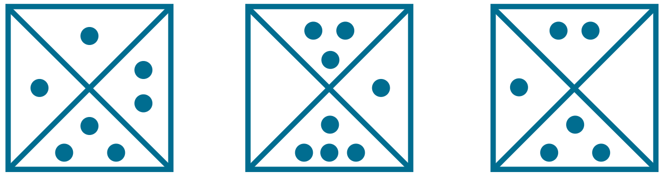
Input: 4-sided dominos: 

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

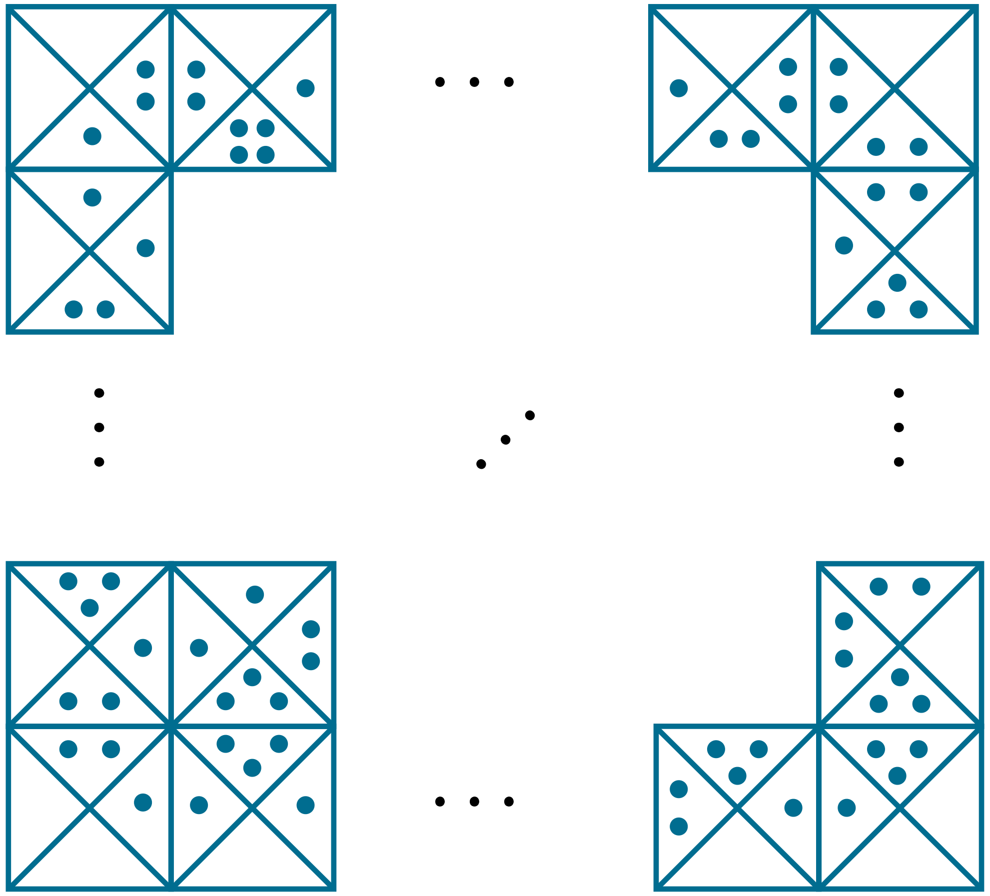


The (undecidable) Domino problem

Domino

Input: 4-sided dominos: 

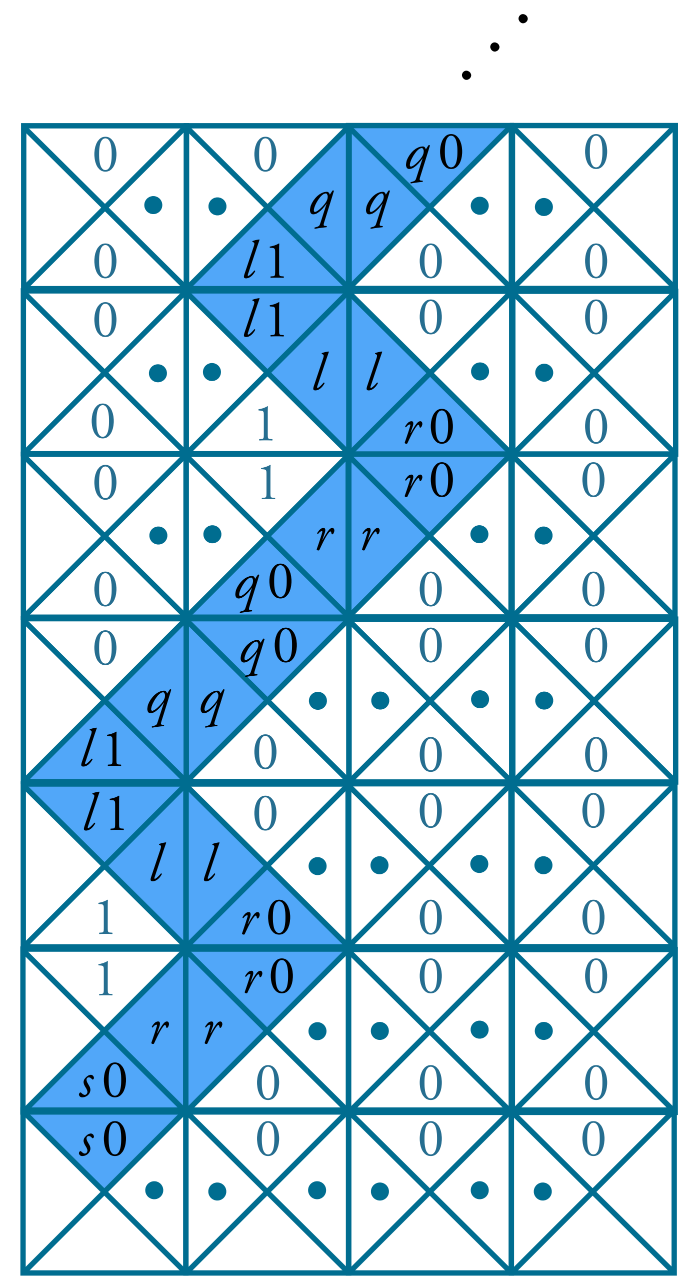
Output: Is it possible to form a white-bordered rectangle? (of any size)



Rules: sides must match,
you can't rotate the dominos, but you can 'clone' them.

Domino - Why is it undecidable?

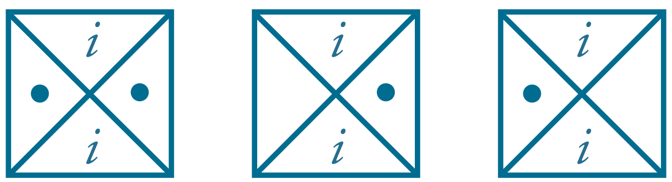
It can easily encode *halting* computations of Turing machines:



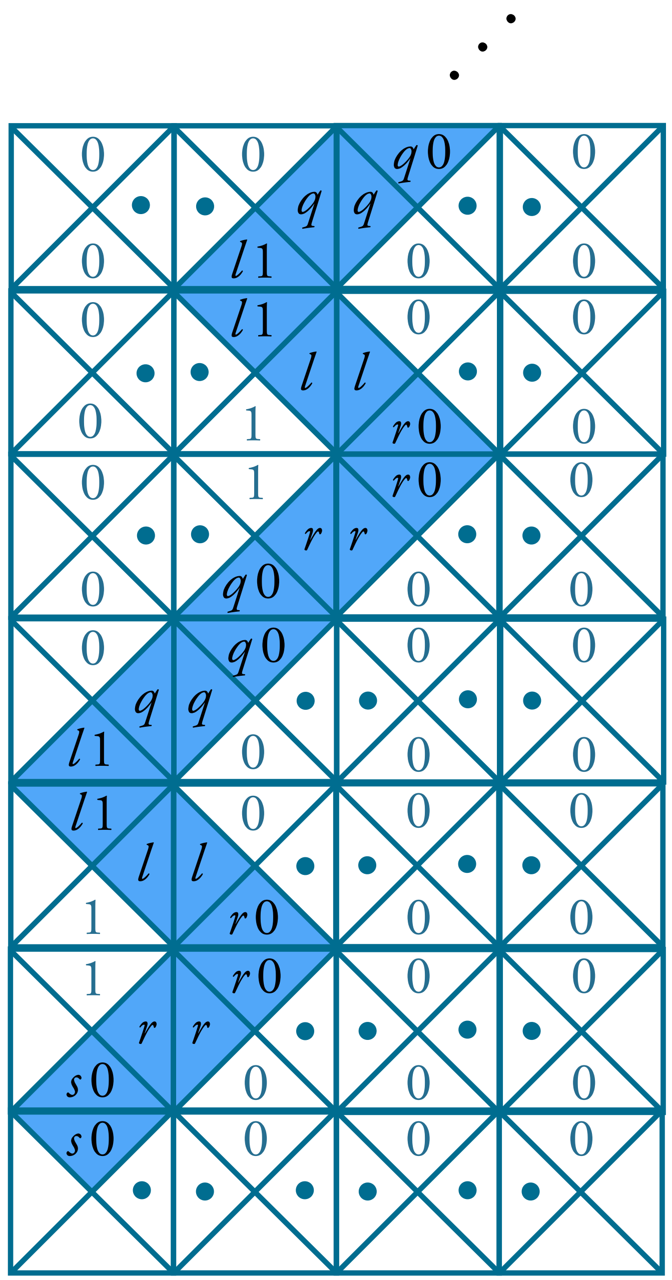
The (undecidable) Domino problem

Domino - Why is it undecidable?

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



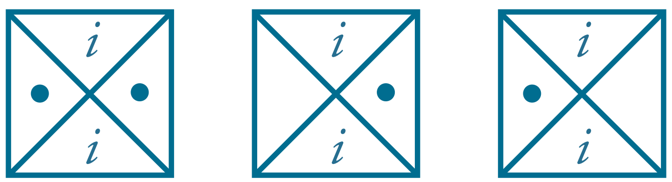
(head is elsewhere,
symbol is not modified)



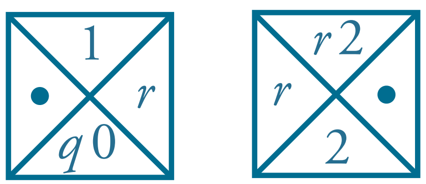
The (undecidable) Domino problem

Domino - Why is it undecidable?

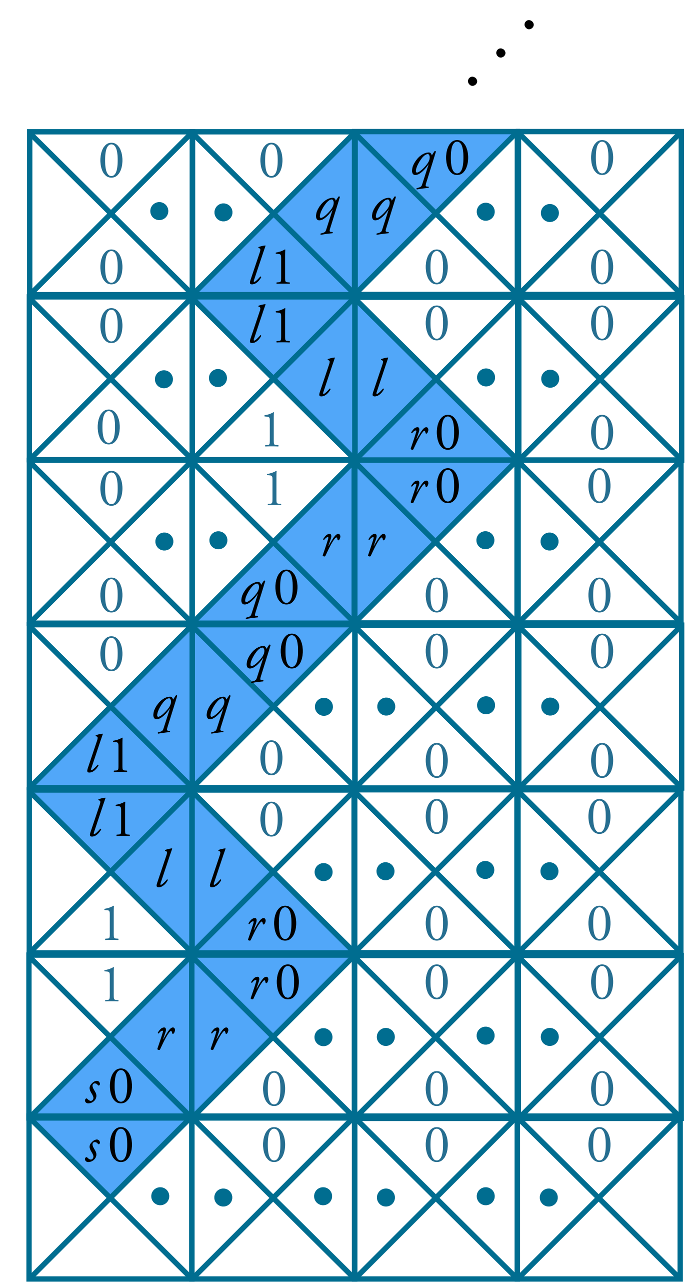
It can easily encode *halting* computations of Turing machines:



(head is elsewhere, symbol is not modified)



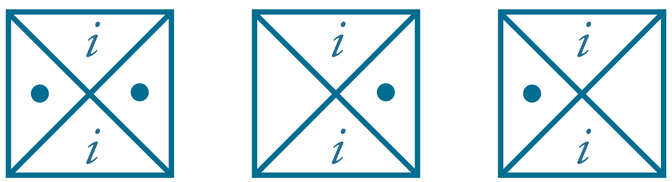
(head is here, symbol is rewritten, head moves right)



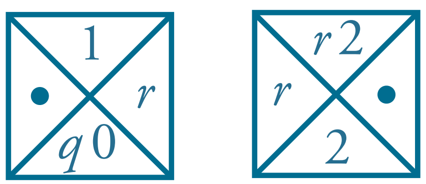
The (undecidable) Domino problem

Domino - Why is it undecidable?

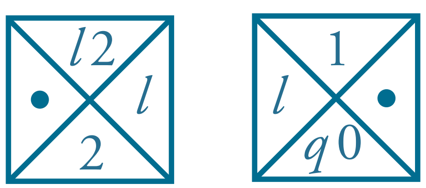
It can easily encode *halting* computations of Turing machines:



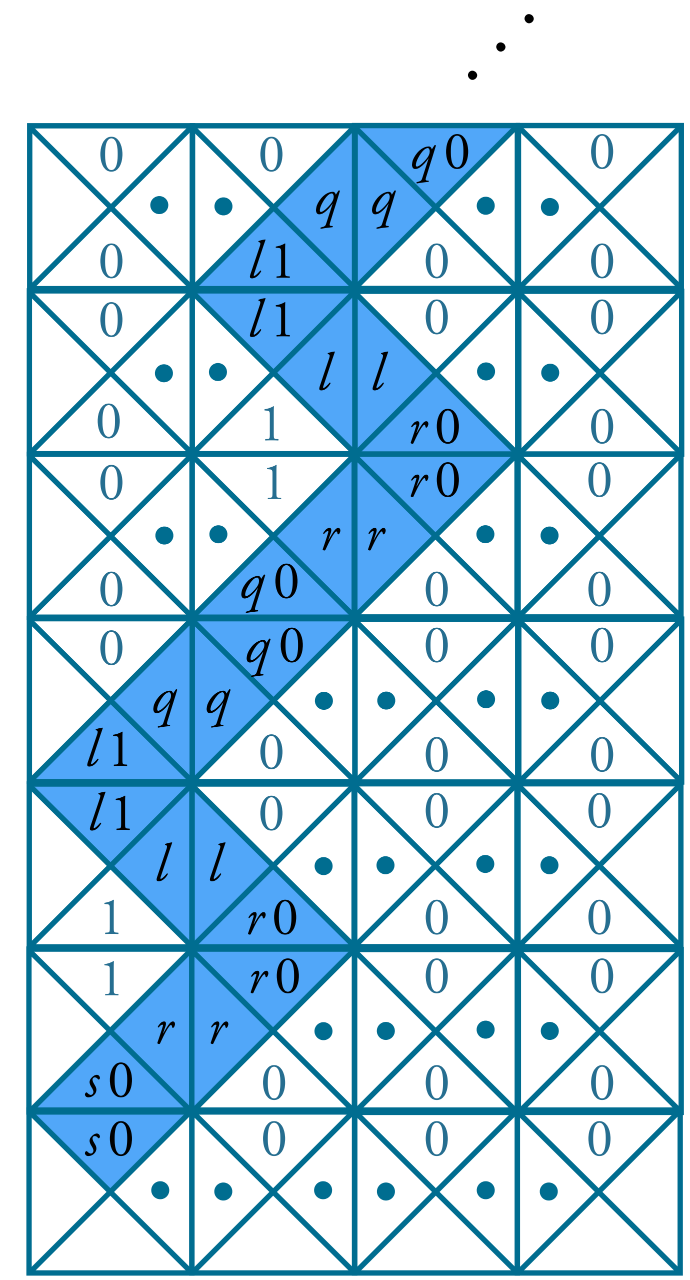
(head is elsewhere,
symbol is not modified)



(head is here, symbol is
rewritten, head moves right)



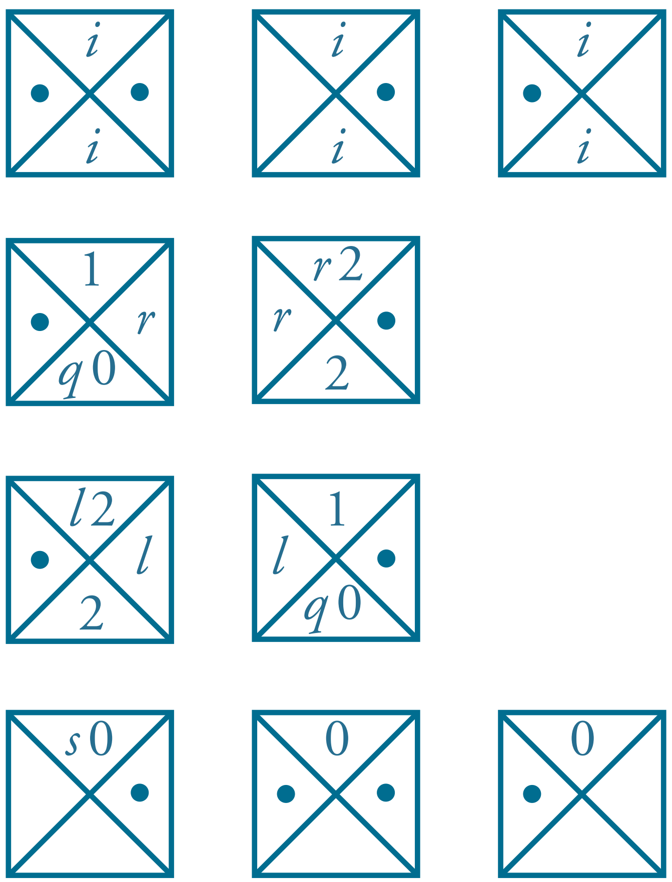
(head is here, symbol is
rewritten, head moves left)



The (undecidable) Domino problem

Domino - Why is it undecidable?

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

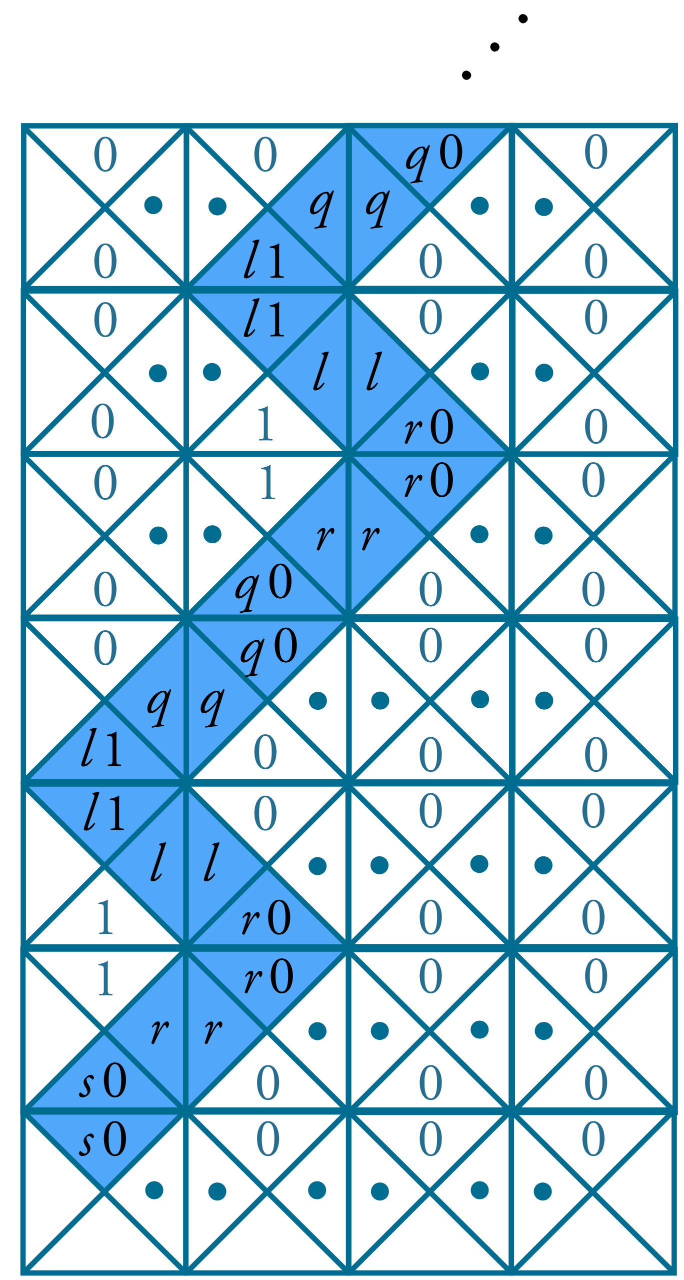


(head is elsewhere, symbol is not modified)

(head is here, symbol is rewritten, head moves right)

(head is here, symbol is rewritten, head moves left)

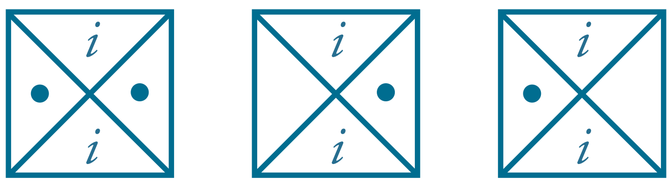
(initial configuration)



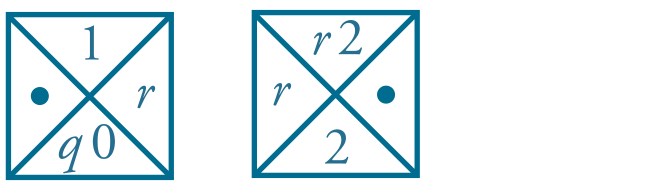
The (undecidable) Domino problem

Domino - Why is it undecidable?

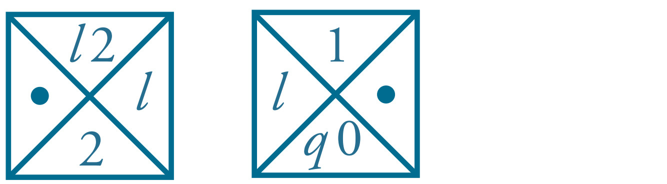
It can easily encode *halting* computations of Turing machines:



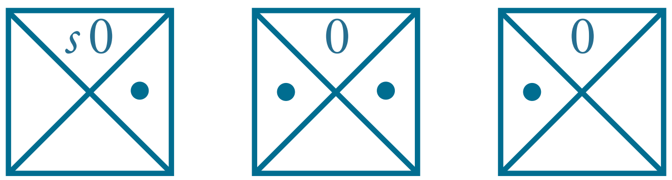
(head is elsewhere,
symbol is not modified)



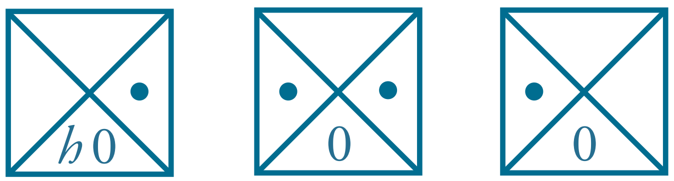
(head is here, symbol is
rewritten, head moves right)



(head is here, symbol is
rewritten, head moves left)

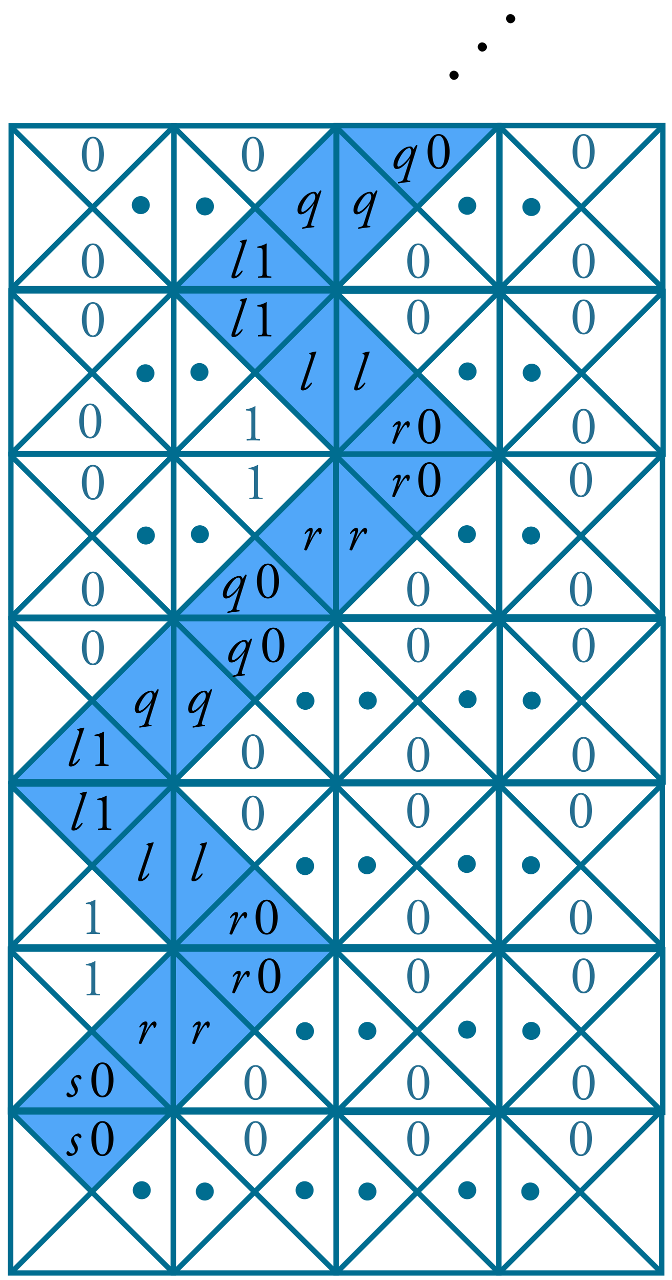


(initial configuration)



(halting configuration)

. . .

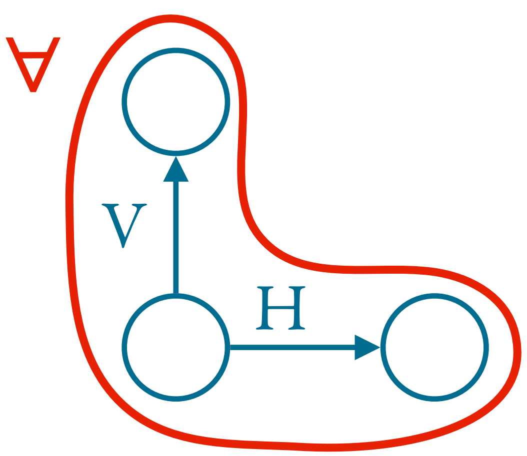


Domino $\rightsquigarrow$ Sat-FO (domino has a solution iff ϕ satisfiable)

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

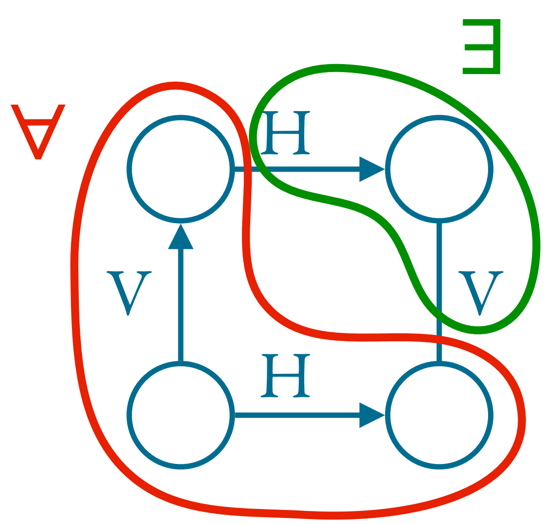
Domino ⇝ Sat-FO (domino has a solution iff ϕ satisfiable)

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



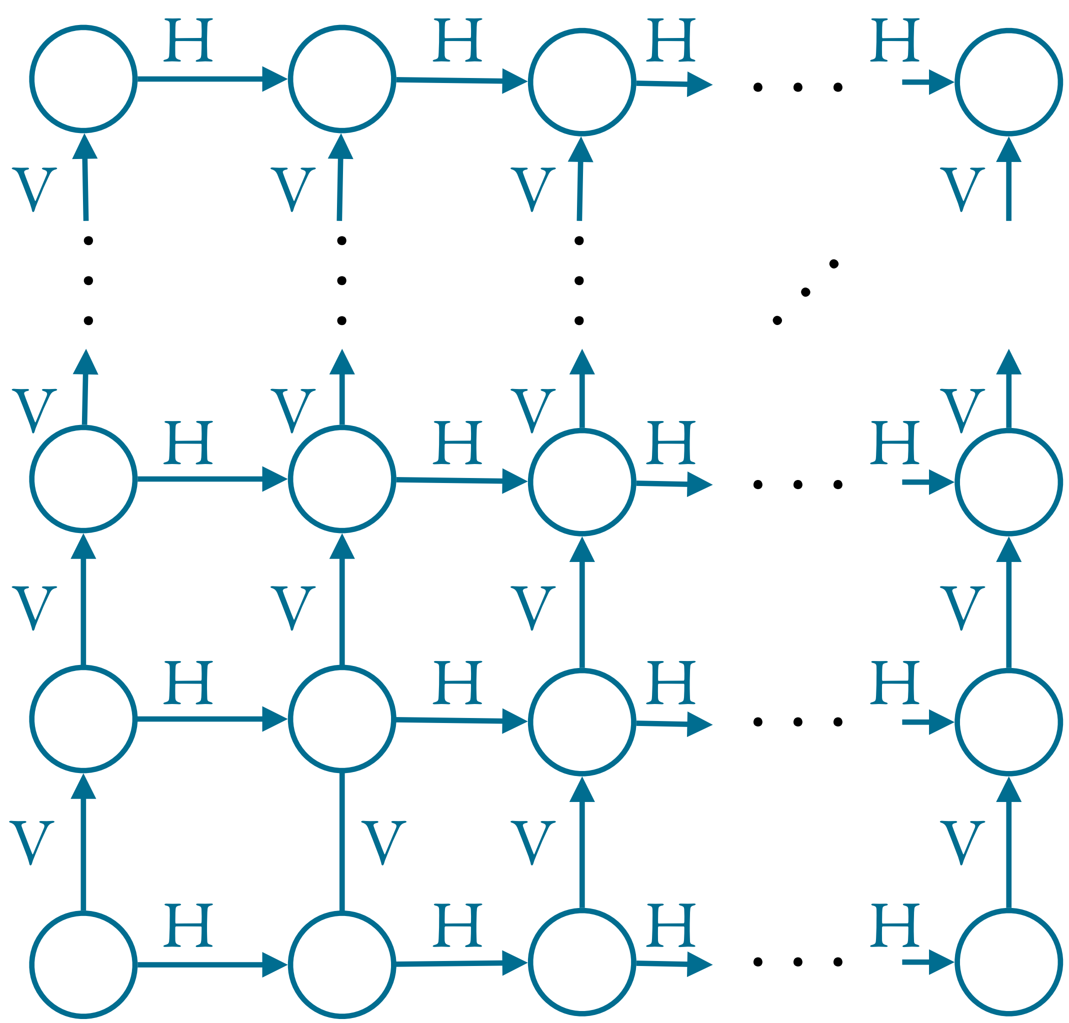
Domino ⇝ Sat-FO (domino has a solution iff ϕ satisfiable)

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



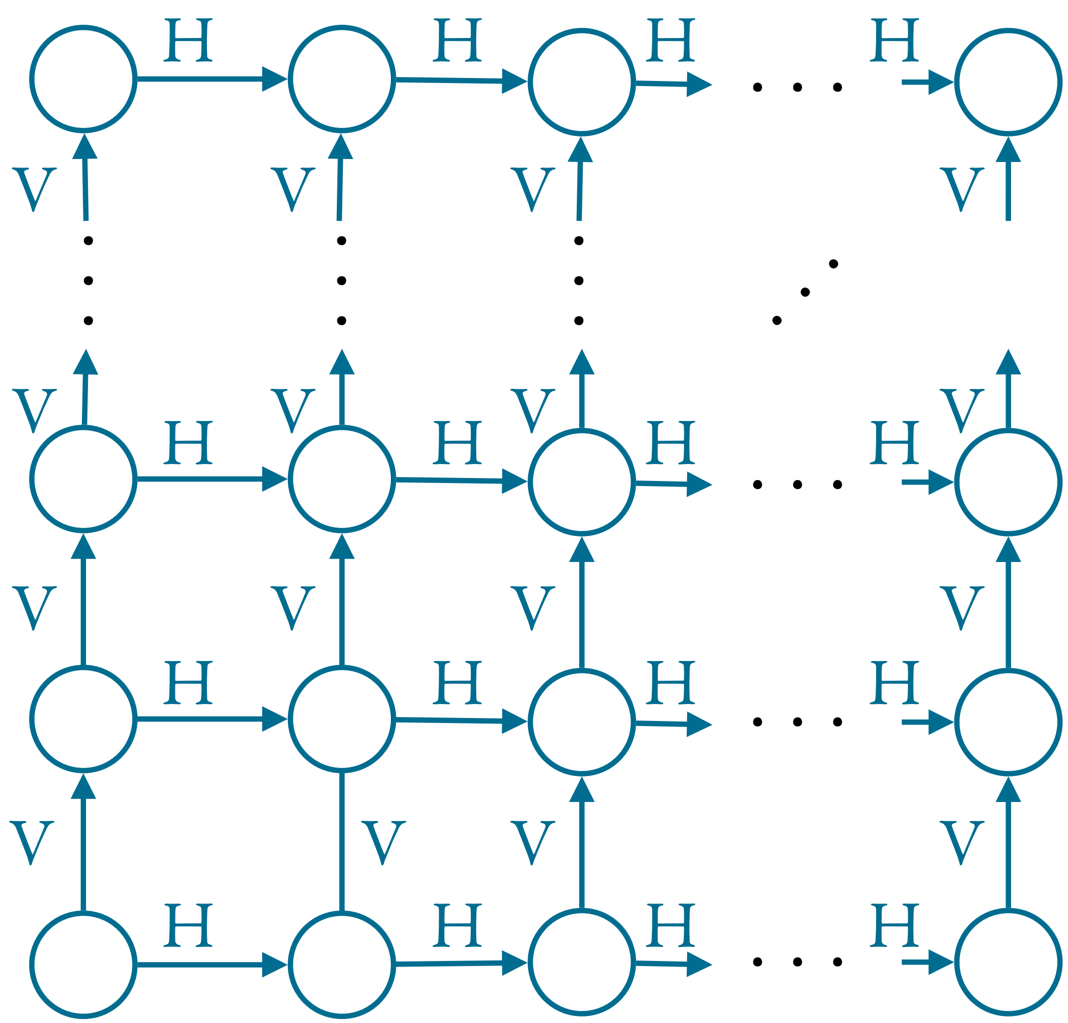
Domino ⇝ Sat-FO (domino has a solution iff ϕ satisfiable)

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



Domino ⇝ Sat-FO (domino has a solution iff ϕ satisfiable)

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



2. **Assign one domino to each node:**

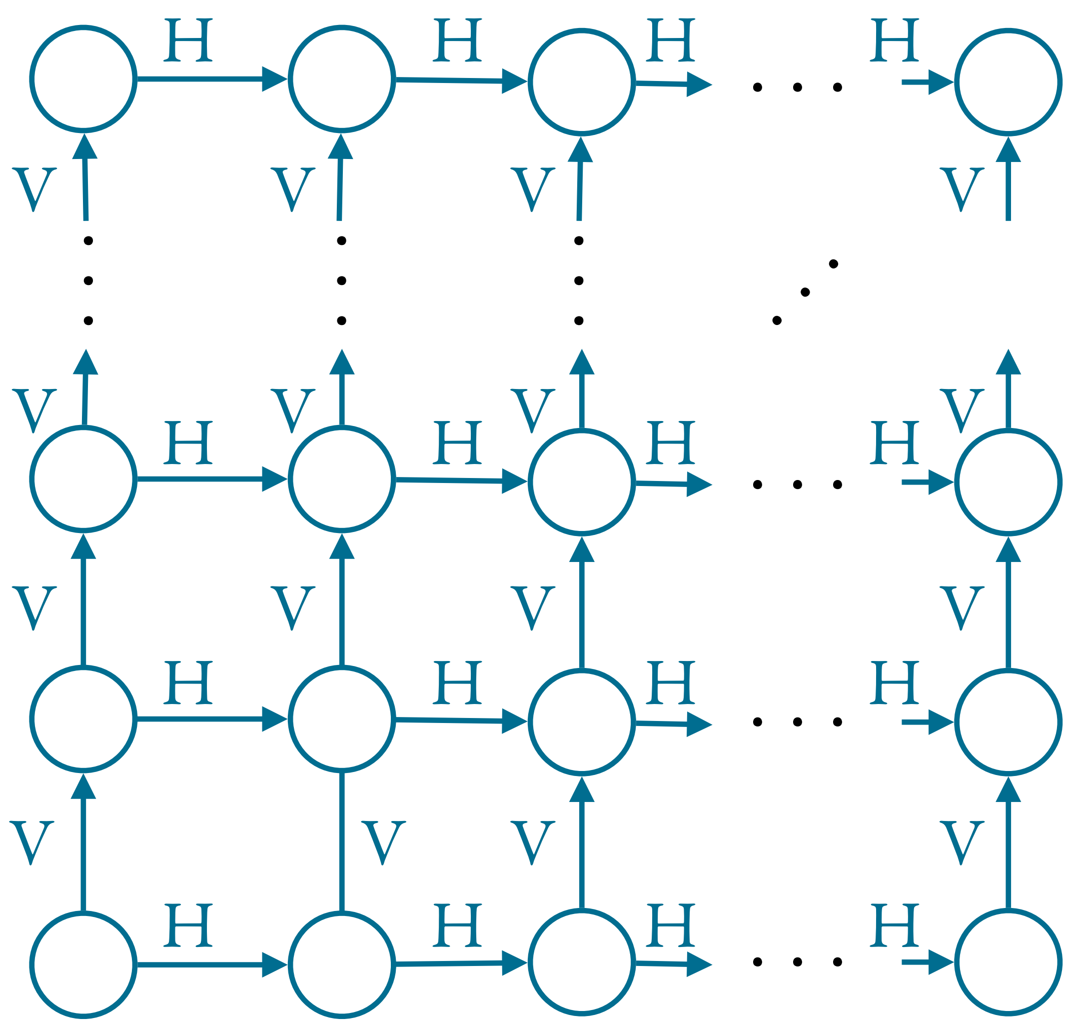
a unary relation

$$\mathbf{D}_{\boxtimes}(\mathbf{x})$$

for each domino $\boxtimes$

Domino ⇝ Sat-FO (domino has a solution iff ϕ 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

$$D_{\boxtimes}(x)$$

for each domino $\boxtimes$

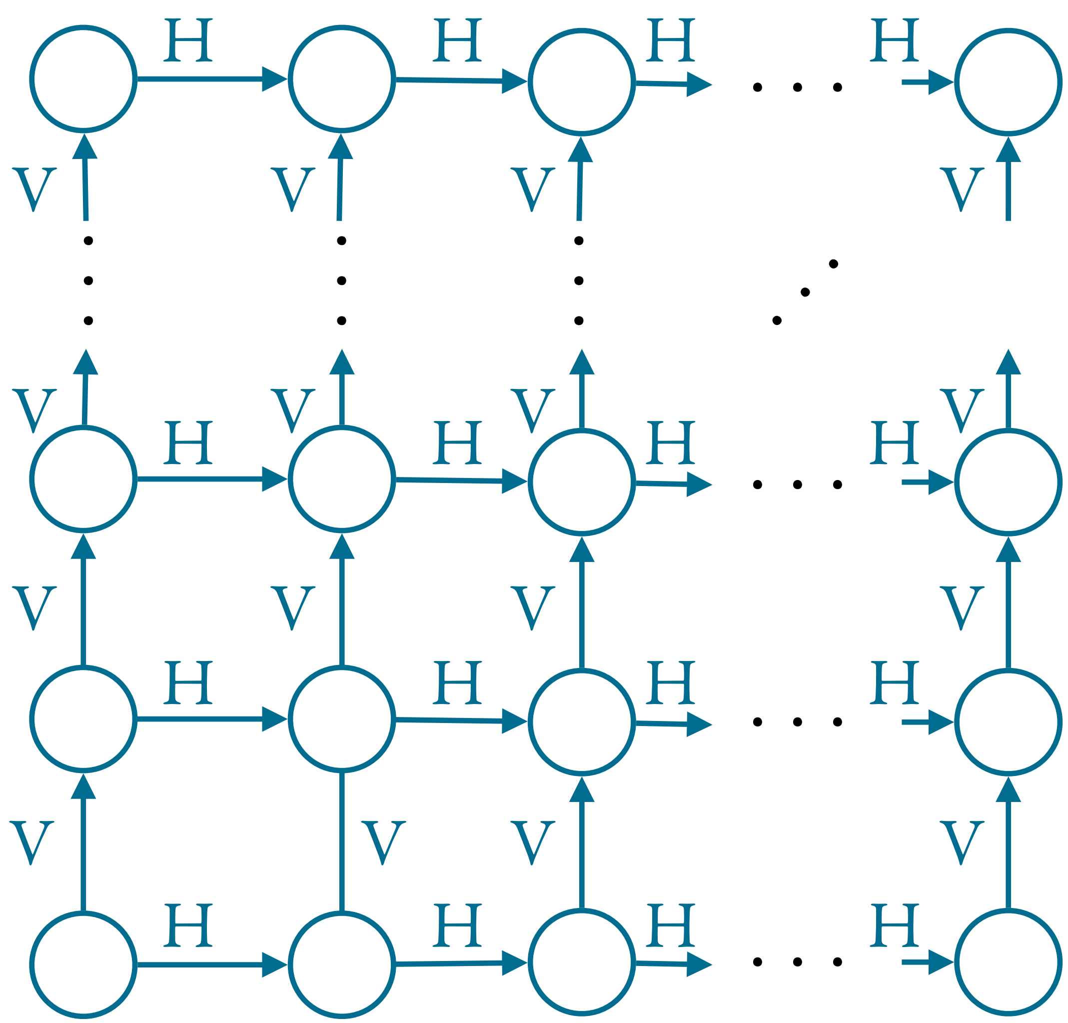
3. **Match the sides** $\forall x,y$

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

for some dominos **a,b** that 'match' horizontally (Idem vertically)

Domino ⇝ Sat-FO (domino has a solution iff ϕ 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

$$D_{\boxtimes}(x)$$

for each domino 

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

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

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 α, does $G \models_\alpha \phi$?

DECIDABLE ⇝ foundations of the database industry

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

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

Equivalence problem: Given FO formulae ϕ,ψ, is

$$G \models_\alpha \phi \quad \text{iff} \quad G \models_\alpha \psi$$

for all graphs G and bindings α?

💀 UNDECIDABLE ⇝ by reduction to the satisfiability problem

Algorithmic problems for FO

Equivalence problem: Given FO formulae ϕ,ψ, is

$$G \models_\alpha \phi \quad \text{iff} \quad G \models_\alpha \psi$$

for all graphs G and bindings α?

💀 UNDECIDABLE ⇝ by reduction from the satisfiability problem

Algorithmic problems for FO

ϕ is **satisfiable** iff ϕ is **not equivalent** to $\bot$

Satisfiability problem undecidable $\rightsquigarrow$ Equivalence problem undecidable

Equivalence problem: Given FO formulae ϕ,ψ, is

$$G \models_\alpha \phi \quad \text{iff} \quad G \models_\alpha \psi$$

for all graphs G and bindings α?

💀 UNDECIDABLE $\rightsquigarrow$ by reduction from the satisfiability problem

Algorithmic problems for FO

ϕ is **satisfiable** iff ϕ is **not equivalent** to $\bot$

Satisfiability problem undecidable ⇝ 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

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

Equivalence problem: Given FO formulae ϕ,ψ, is

$$G \models_\alpha \phi \quad \text{iff} \quad G \models_\alpha \psi$$

for all graphs G and bindings α?

💀 UNDECIDABLE ⇝ 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 α, does $G \vDash_\alpha \phi$?

DECIDABLE ⇝ foundations of the database industry

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

💀 UNDECIDABLE ⇝ both for $\vDash$ and $\vDash_{\text{finite}}$

Equivalence problem: Given FO formulae ϕ,ψ, is

$$G \vDash_\alpha \phi \quad \text{iff} \quad G \vDash_\alpha \psi$$

for all graphs G and bindings α?

💀 UNDECIDABLE ⇝ by reduction to the satisfiability problem

Evaluation problem for FO

Input:

$$\left(\begin{array}{l} \phi(x_1,\ldots,x_n) \\ G = (V,E) \\ \alpha = \{x_1,\ldots,x_n\} \longrightarrow V \end{array} \right.$$

Output: $G \models_\alpha \phi$?

Evaluation problem for FO

Input: $\left(\begin{array}{l} \phi(x_1,\ldots,x_n) \\ G = (V,E) \\ \alpha = \{x_1,\ldots,x_n\} \longrightarrow V \end{array} \right.$ **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), …

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

Evaluation problem for FO

Input: $\left(\begin{array}{l} \phi(x_1,\ldots,x_n) \\ G = (V,E) \\ \alpha = \{x_1,\ldots,x_n\} \longrightarrow V \end{array}\right.$ **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), …

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

Encoding of $\alpha = \{x_1,\ldots,x_n\} \longrightarrow V$

- 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: $\begin{cases} \phi(x_1,\ldots,x_n) \\ G = (V,E) \\ \alpha = \{x_1,\ldots,x_n\} \longrightarrow V \end{cases}$

Output: $G \models_\alpha \phi$?

Evaluation problem for FO

Input: $\begin{cases} \phi(x_1,\ldots,x_n) \\ G = (V,E) \\ \alpha = \{x_1,\ldots,x_n\} \longrightarrow V \end{cases}$ 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 $\alpha' = \alpha \cup \{y \mapsto v\}$
  we have $G \models_{\alpha'} \psi$.

Evaluation problem for FO

Input: $\left(\begin{array}{l} \phi(x_1,\dots,x_n) \\ G = (V,E) \\ \alpha = \{x_1,\dots,x_n\} \longrightarrow V \end{array} \right.$

Output: $G \models_\alpha \phi$?

- If $\phi(x_1,\dots,x_n) = E(x_i,x_j)$:
  answer YES iff $(\alpha(x_i),\alpha(x_j)) \in E$
- If $\phi(x_1,\dots,x_n) = \psi(x_1,\dots,x_n) \wedge \psi'(x_1,\dots,x_n)$:
  answer YES iff $G \models_\alpha \psi$ and $G \models_\alpha \psi'$
- If $\phi(x_1,\dots,x_n) = \neg\psi(x_1,\dots,x_n)$:
  answer NO iff $G \models_\alpha \psi$
- If $\phi(x_1,\dots,x_n) = \exists y . \psi(x_1,\dots,x_n,y)$:
  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: $\begin{cases} \phi(x_1,\ldots,x_n) \\ G = (V,E) \\ \alpha = \{x_1,\ldots,x_n\} \longrightarrow V \end{cases}$

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 ⇝ LOGSPACE

- 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 $\alpha' = \alpha \cup \{y \mapsto v\}$ we have $G \models_{\alpha'} \psi$.

Question:
How much space does it take?

Evaluation problem for FO

Input: $\begin{cases} \phi(x_1,\ldots,x_n) \\ G = (V,E) \\ \alpha = \{x_1,\ldots,x_n\} \longrightarrow V \end{cases}$

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 $\rightsquigarrow$ LOGSPACE

- 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 $G \models_\alpha \psi'$

  $\rightsquigarrow$ 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 $G \models_\alpha \psi$

- 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: $\begin{cases} \phi(x_1,\ldots,x_n) \\ G = (V,E) \\ \alpha = \{x_1,\ldots,x_n\} \longrightarrow V \end{cases}$ 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 $\rightsquigarrow$ LOGSPACE

- 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 $G \models_\alpha \psi'$
  $\rightsquigarrow$ 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 $G \models_\alpha \psi$
  $\rightsquigarrow$ 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 $G \models_{\alpha'} \psi$.

Question:
How much space
does it take?

Evaluation problem for FO

Input: $\begin{cases} \phi(x_1,\ldots,x_n) \\ G = (V,E) \\ \alpha = \{x_1,\ldots,x_n\} \longrightarrow V \end{cases}$ Output: $G \vDash_\alpha \phi$?

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

- If $\phi(x_1,\ldots,x_n) = \psi(x_1,\ldots,x_n) \wedge \psi'(x_1,\ldots,x_n)$:
  answer YES iff $G \vDash_\alpha \psi$ and $G \vDash_\alpha \psi'$
  $\rightsquigarrow$ MAX(SPACE($G \vDash_\alpha \psi$)), SPACE($G \vDash_\alpha \psi'$)))

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

- 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 $G \vDash_{\alpha'} \psi$.
  $\rightsquigarrow$ $2 \cdot \log(|G|)$ + SPACE($G \vDash_{\alpha'} \psi$)

Question:
How much space
does it take?

Evaluation problem for FO

Input: $\left(\begin{array}{l} \phi(x_1,\ldots,x_n) \\ G = (V,E) \\ \alpha = \{x_1,\ldots,x_n\} \longrightarrow V \end{array} \right.$ 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 $\rightsquigarrow$ 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 $G \models_\alpha \psi'$
  $\rightsquigarrow$ 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 $G \models_\alpha \psi$
  $\rightsquigarrow$ 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 $G \models_{\alpha'} \psi$.
  $\rightsquigarrow$ $2 \cdot \log(|G|)$ + SPACE($G \models_{\alpha'} \psi$)

Question:
How much space does it take?

$$\underbrace{2\cdot\log(|G|) + \cdots + 2\cdot\log(|G|)}_{\leq |\phi| \text{ times}} + k\cdot\log(|\alpha|+|G|) \text{ space}$$

Evaluation problem for FO in PSPACE

Input: $\begin{cases} \phi(x_1,\ldots,x_n) \\ G = (V,E) \\ \alpha = \{x_1,\ldots,x_n\} \longrightarrow V \end{cases}$

Output: $G \vDash_\alpha \phi$?

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

  use 4 pointers $\rightsquigarrow$ LOGSPACE

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

  $\rightsquigarrow$ MAX(SPACE($G \vDash_\alpha \psi$)), SPACE($G \vDash_\alpha \psi'$)))

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

  $\rightsquigarrow$ SPACE($G \vDash_\alpha \psi$))

- 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 $G \vDash_{\alpha'} \psi$.

  $\rightsquigarrow$ $2 \cdot \log(|G|)$ + SPACE($G \vDash_{\alpha'} \psi$)

Question:
How much space does it take?

$$\underbrace{2 \cdot \log(|G|) + \cdots + 2 \cdot \log(|G|)}_{\leq |\phi| \text{ times}} + k \cdot \log(|\alpha|+|G|) \text{ space}$$

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: **QBF**

(satisfaction of Quantified Boolean Formulas)

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

$\exists p\ \forall q\ .\ (p \vee \neg q)$ 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)

$\exists p \; \forall q \, . \, (p \vee \neg q)$ where p,q range over {T,F}

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

Polynomial reduction **QBF** $\rightsquigarrow$ FO :

1. Given $\psi \in$ QBF,
let $\psi'(x)$ be the replacement
of each 'p' with 'p=x' in ψ.

2. Note: $\exists x \, \psi'$ holds in a 2-element
graph iff ψ is QBF-satisfiable

3. Test if $G \models_{\varnothing} \psi'$ 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)

$\exists p\, \forall q\, .\, (p \vee \neg q)$ where p,q range over {T,F}

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

Polynomial reduction **QBF** $\rightsquigarrow$ FO :

$\psi'(x) = \exists p\, \forall q\, .\, (\, (p=x) \vee \neg(q=x)\,)$

1. Given $\psi \in$ QBF, let $\psi'(x)$ be the replacement of each 'p' with 'p=x' in ψ.

2. Note: $\exists x\, \psi'$ holds in a 2-element graph iff ψ is QBF-satisfiable

3. Test if $G \models_{\emptyset} \psi'$ 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)

$\exists p\ \forall q\ .\ (p \vee \neg q)$ where p,q range over {T,F}

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

Polynomial reduction **QBF** $\rightsquigarrow$ FO :

$\psi'(x) = \exists p\ \forall q\ .\ (\ (p=x) \vee \neg(q=x)\ )$

$\exists x\ \exists p\ \forall q\ .\ (\ (p=x) \vee \neg(q=x)\ )$

1. Given $\psi \in$ QBF, let $\psi'(x)$ be the replacement of each 'p' with 'p=x' in ψ.

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

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

Combined, Query, and Data complexities

[Vardi, 1982]

Problem: Usual scenario in database

A **database** of size 10^6

A **query** of size 100

Input:

Combined, Query, and Data complexities

[Vardi, 1982]

Problem: Usual scenario in database

A **database** of size 10^6

A **query** of size 100

Input: • **query** +

Combined, Query, and Data comp

Problem: Usual scen

Input: • **query** +

database

Combined, Query, and Data comp

Problem: Usual scen

Input: • **query** +

database

But we don't distinguish this in the analysis:

$$\mathrm{TIME}(2^{|\text{query}|} + |\text{data}|) = \mathrm{TIME}(|\text{query}| + 2^{|\text{data}|})$$

Combined, Query, and Data complexities
[Vardi, 1982]

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

Combined, Query, and Data complexities

Combined complexity: input size is |query| + |data|

Query complexity (|data| fixed): input size is |query|

Data complexity (|query| fixed): input size is |data|

Combined, Query, and Data complexities

Combined complexity: input size is |query| + |data|

Query complexity (|data| fixed): input size is |query|

Data complexity (|query| fixed): input size is |data|

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

$O(|query| + 2^{|data|})$ is
exponential in **combined** complexity
linear 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: |data|

query complexity, input size: |query|

combined complexity, input size: |data| + |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: |data|

query complexity, input size: |query|

combined complexity, input size: |data| + |query|

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

query

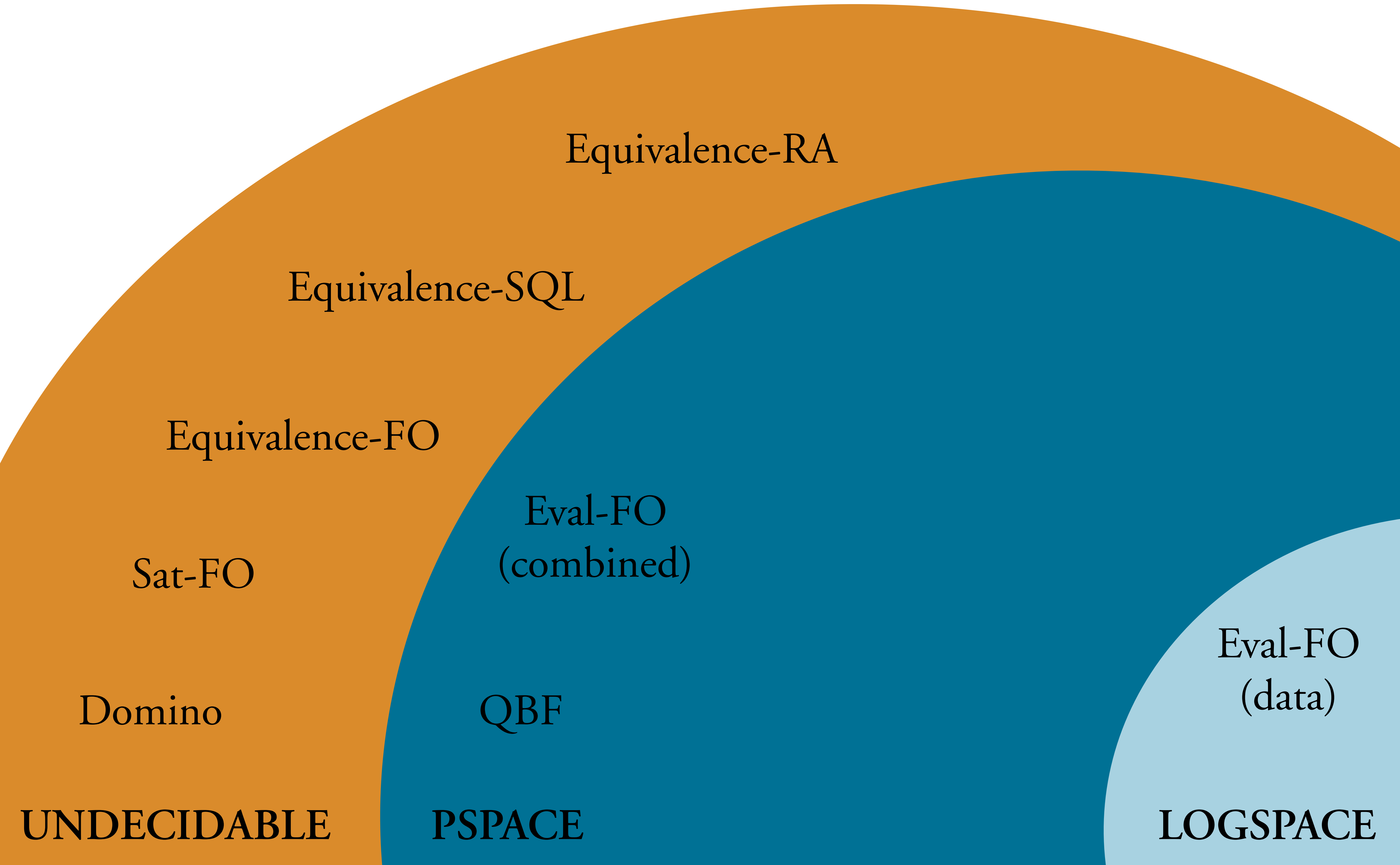
data

O(log(|data|)·|query|) space

PSPACE combined and query complexity

LOGSPACE data complexity

Recap


Equivalence-RA
Equivalence-SQL
Equivalence-FO
Sat-FO
Domino
UNDECIDABLE
Eval-FO
(combined)
QBF
PSPACE
Eval-FO
(data)
LOGSPACE

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

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

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

LOGSPACE $\subseteq$ PTIME $\subseteq$ **NP** $\subseteq$ PSPACE $\subseteq$ EXPTIME

NP = 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. $(p \vee \neg q \vee r) \wedge (\neg p \vee s) \wedge (\neg s \vee \neg p)$, 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 $\{B_1, \ldots, B_n\}$, can we find a subset $S \subseteq \{B_1, \ldots, B_n\}$ such that $\sum S = A$?

The class NP

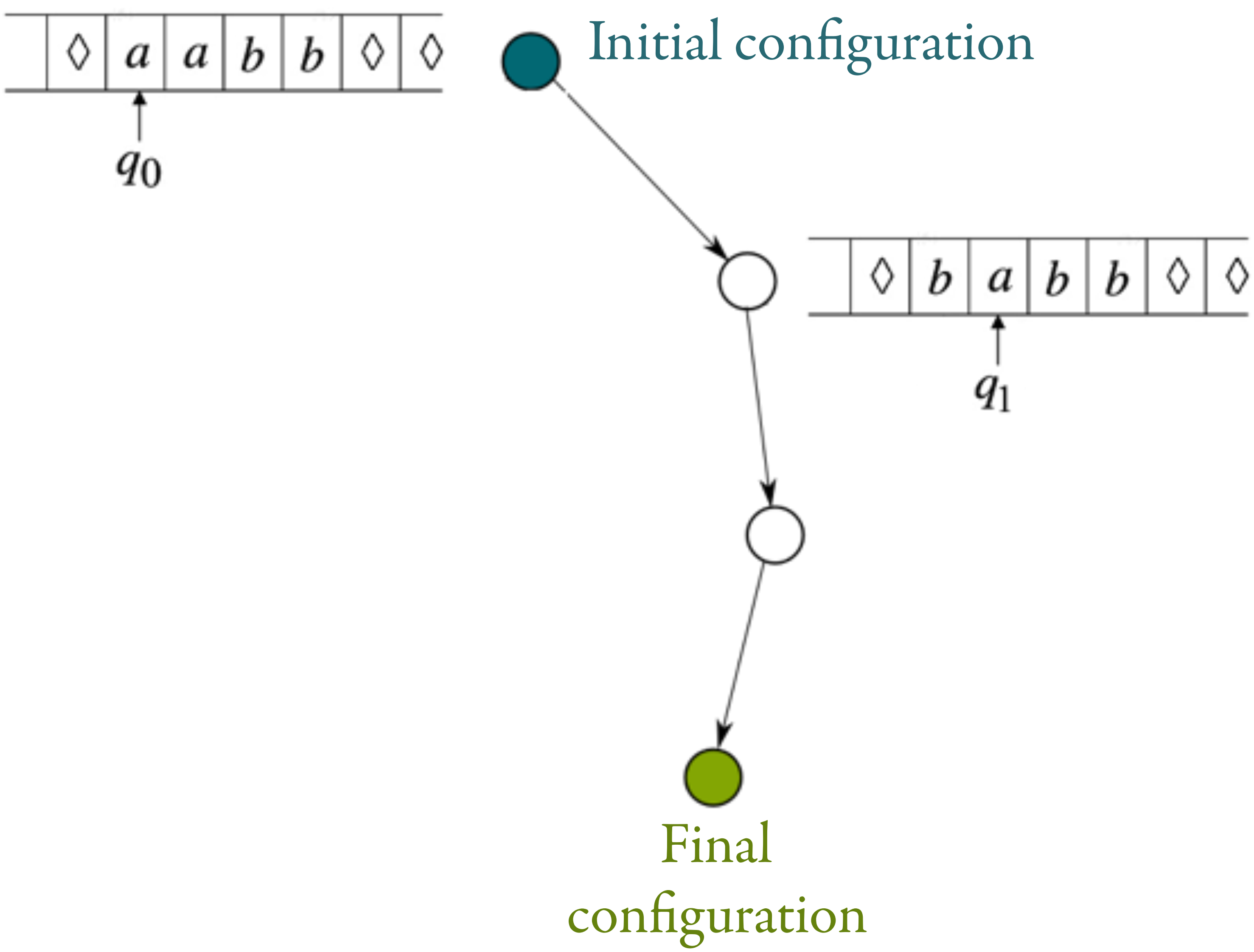
$$\text{LOGSPACE} \quad \subseteq \quad \text{PTIME} \subseteq \textbf{NP} \subseteq \text{PSPACE} \quad \subseteq \quad \text{EXPTIME}$$

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

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

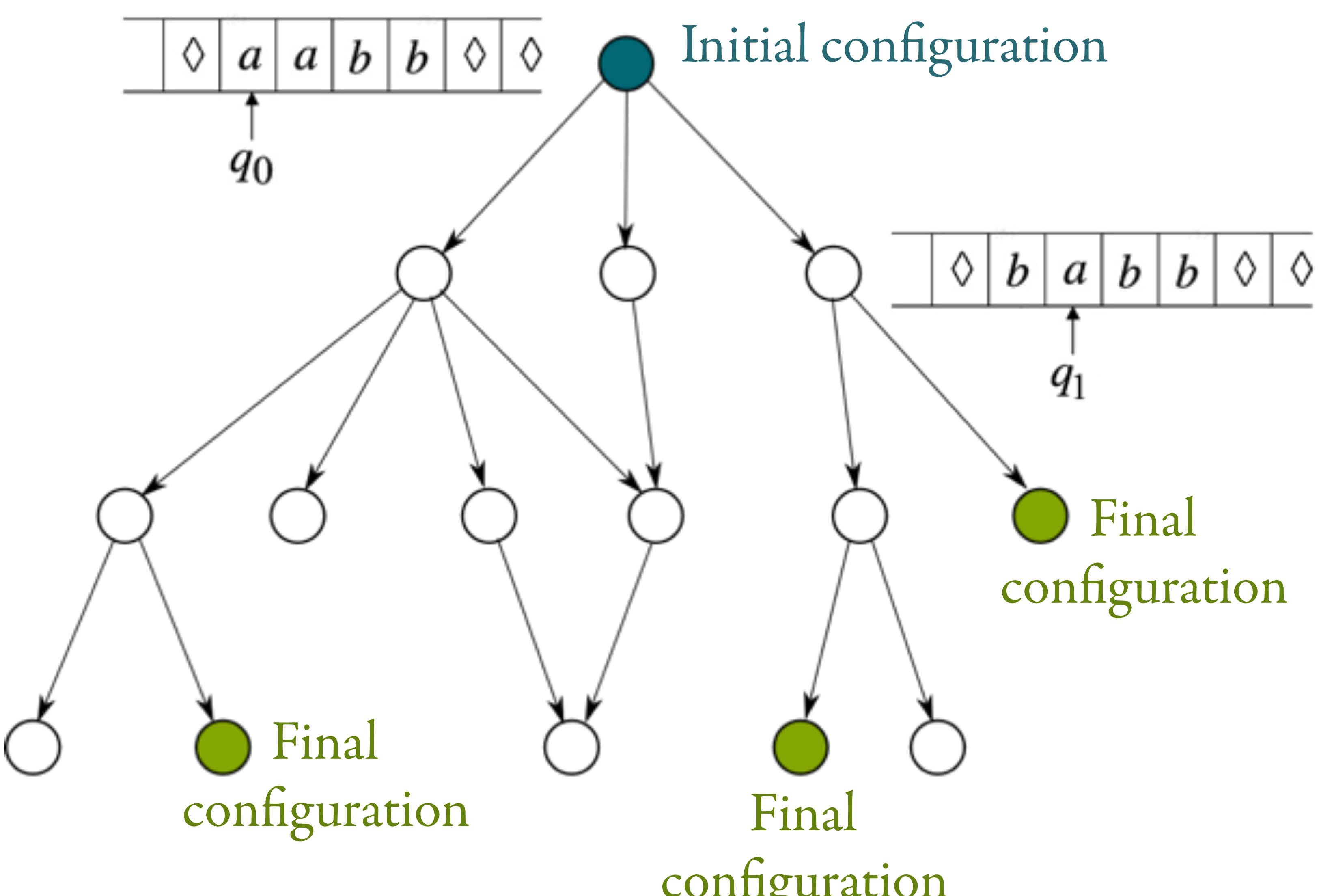
NP = Problems whose solutions can be witnessed by a ***certificate*** to be guessed and checked in *polynomial time* (e.g. a colouring)



The class NP

LOGSPACE $\subseteq$ PTIME $\subseteq$ **NP** $\subseteq$ PSPACE $\subseteq$ EXPTIME

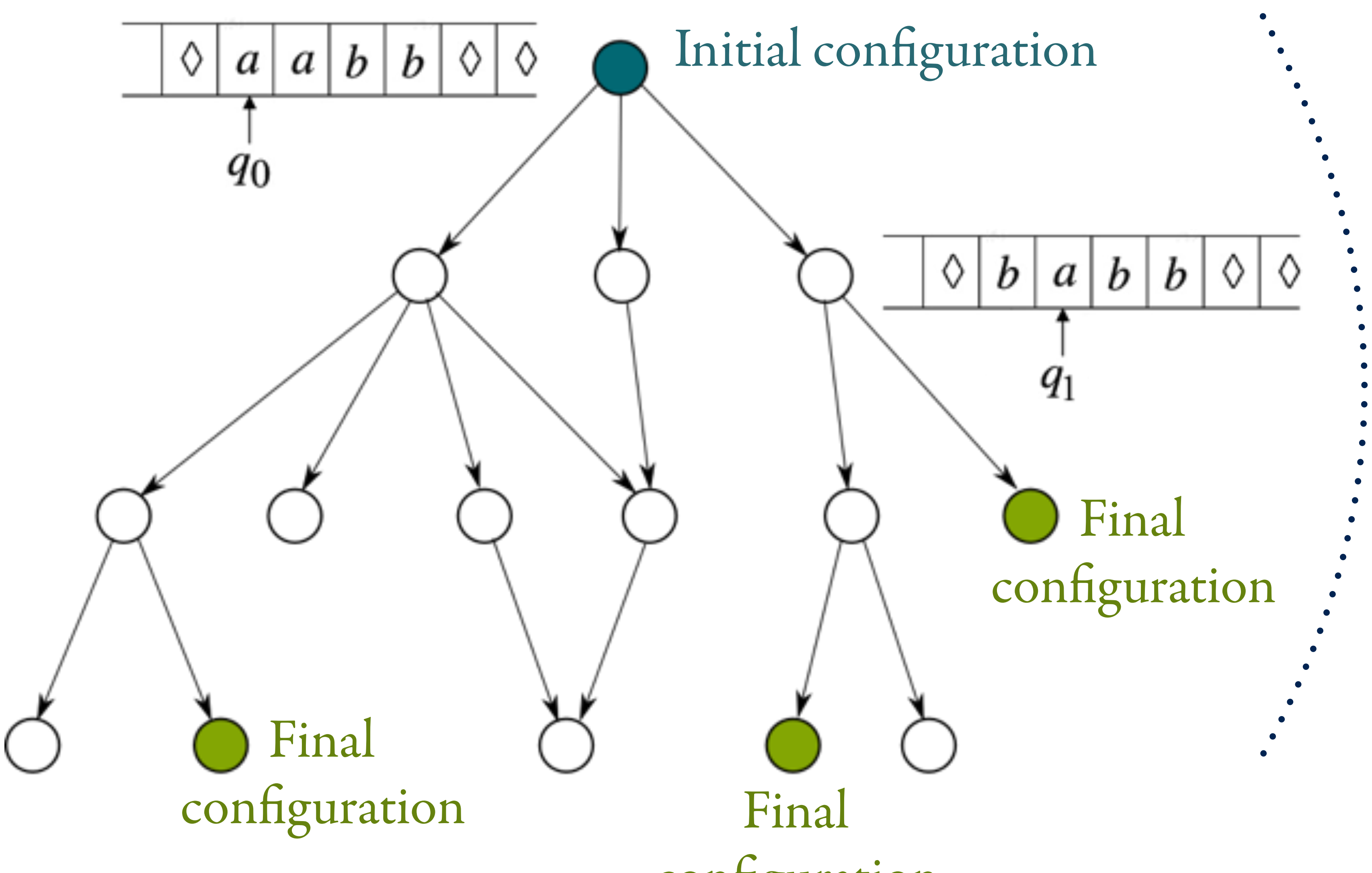
NP = Problems whose solutions can be witnessed by a ***certificate*** to be guessed and checked in *polynomial time* (e.g. a colouring)



The class NP

LOGSPACE $\subseteq$ PTIME $\subseteq$ **NP** $\subseteq$ PSPACE $\subseteq$ EXPTIME

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

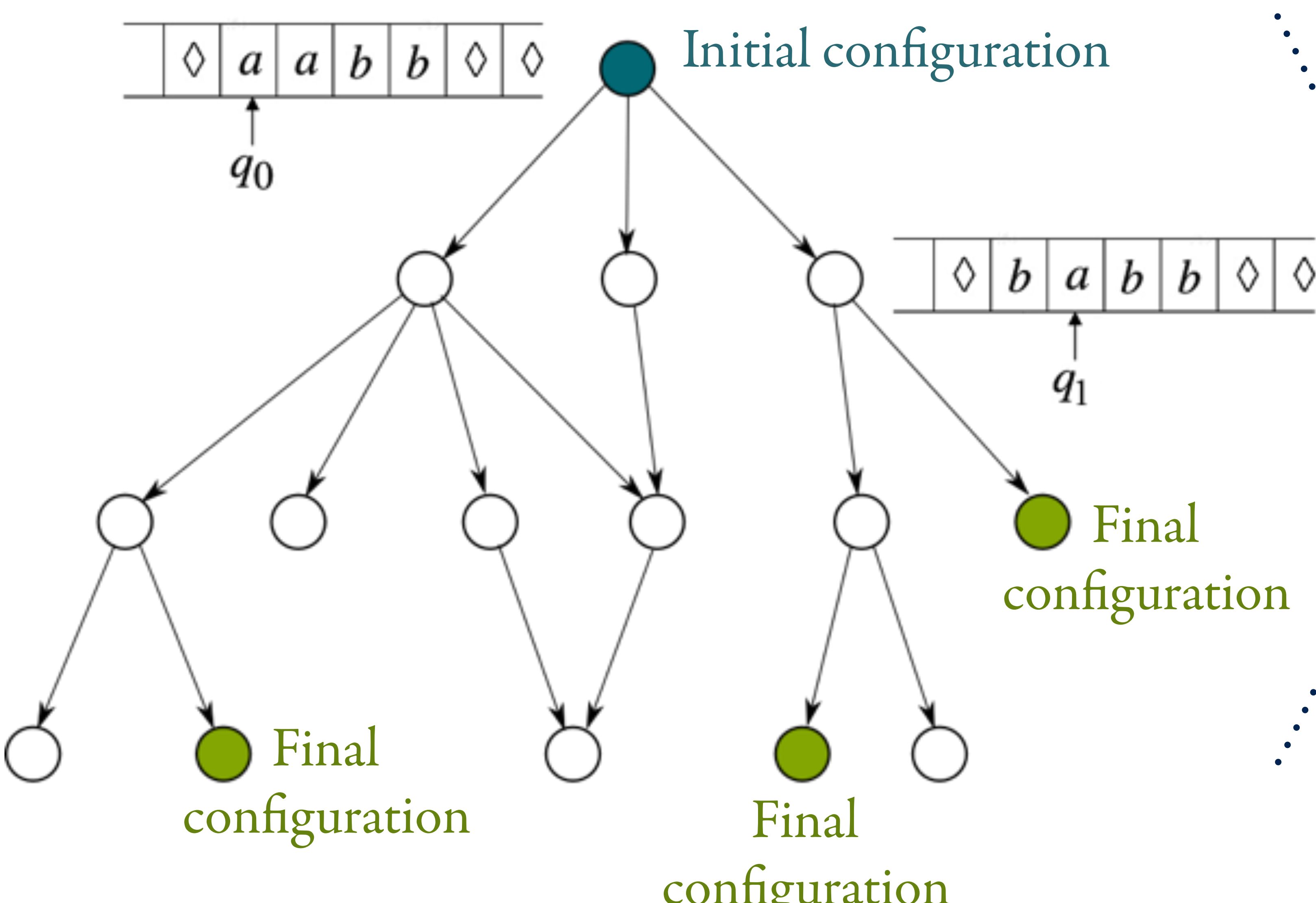


Non-deterministic transitions

The class NP

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)



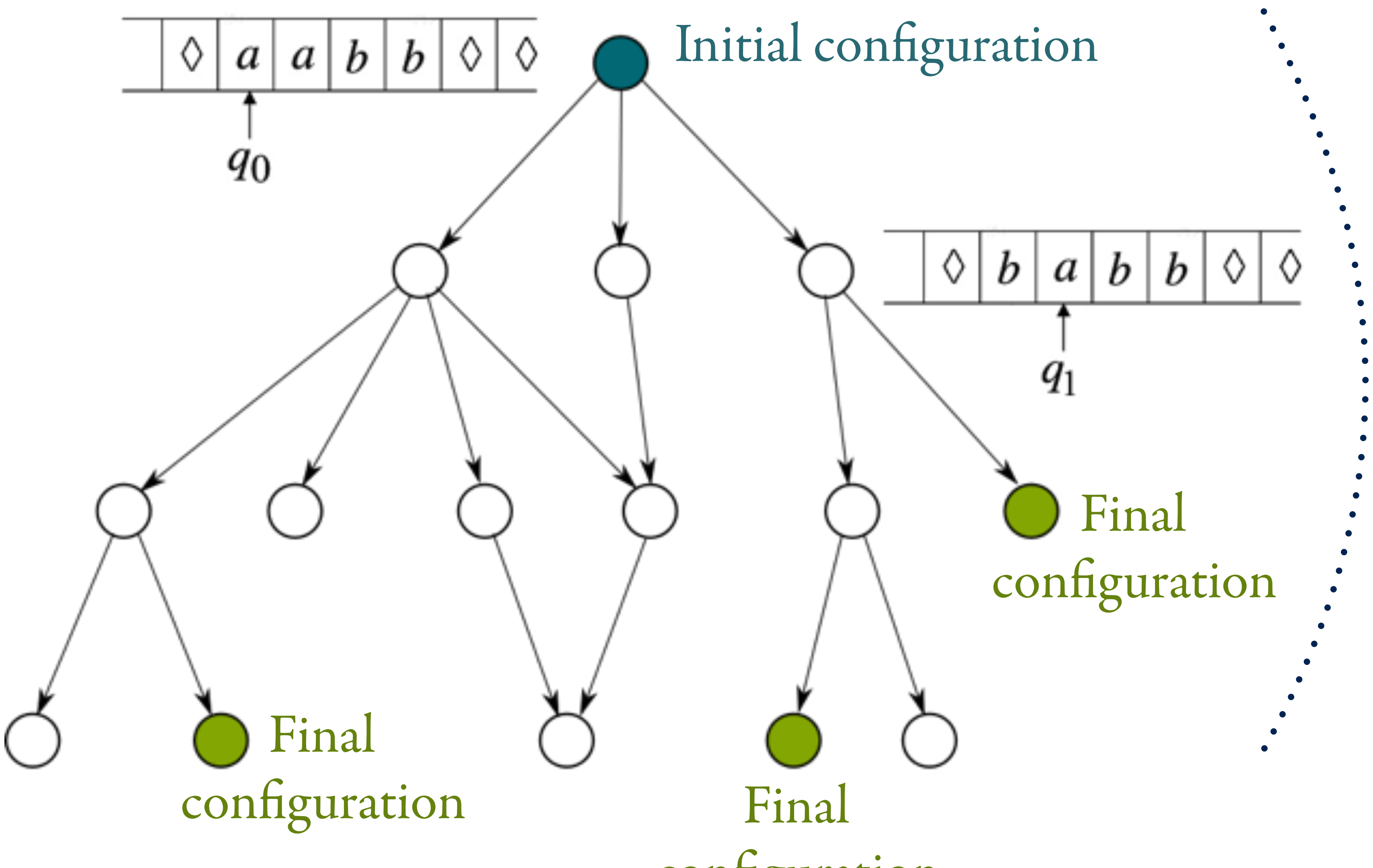
Non-deterministic transitions

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

The class NP

LOGSPACE $\subseteq$ PTIME $\subseteq$ **NP** $\subseteq$ PSPACE $\subseteq$ 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.

Question

Consider: **Positive FO = FO without ∀,¬**

E.g. $\phi = \exists x \, \exists y \, \exists z \, . \, \big(E(x, y) \lor E(y, z)\big) \land \big(y{=}z \lor E(x, z)\big)$

What is the complexity of evaluating Positive FO on graphs ?

Question

Consider: **Positive FO = FO without ∀,¬**

E.g. $\phi = \exists x\, \exists y\, \exists z\, .\, \big(E(x, y) \vee E(y, z)\big) \wedge \big(y{=}z \vee E(x, z)\big)$

What is the complexity of evaluating Positive FO on graphs ?

Solution

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

Conjunctive Queries

Def.

CQ = FO without $\forall, \neg, \vee$

Eg: $\phi(x, y) = \exists z \,.\, (\mathrm{Parent}(x, z) \wedge \mathrm{Parent}(z, y))$

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

Conjunctive Queries

Def.

CQ = FO without $\forall, \neg, \vee$

Normal form: " $\exists\, x_1, \ldots, x_n \,.\, \phi(x_1, \ldots, x_n)$ "

quantifier-free and no equalities!

Eg: $\phi(x, y) = \exists\, z \,.\, (\mathrm{Parent}(x, z) \wedge \mathrm{Parent}(z, y))$

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

Conjunctive Queries

Def.

CQ = FO without $\forall, \neg, \vee$

Normal form: " $\exists x_1, \ldots, x_n . \phi(x_1, \ldots, x_n)$ "

quantifier-free and no equalities!

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

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 "π-σ-×" RA queries

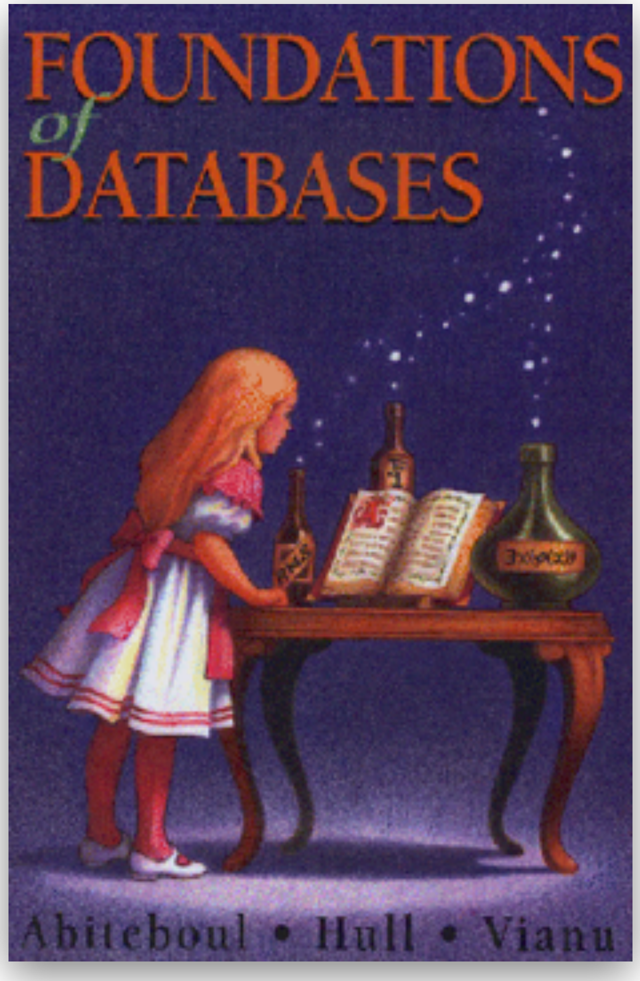
$\pi_X(\sigma_Z(R_1 \times \cdots \times R_n))$

no negation

Bibliography

Abiteboul, Hull, Vianu, "Foundations of Databases", Addison-Wesley, 1995.

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



Chapters 1, 2, 3