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?

Based on slides by D. Figueira, G. Puppis, A.Dawar

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** ∈ **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**) ≠ ∅ ?

⇝ 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

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

Domino

H's 10th

PCP

K

$\vdots$

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

Domino, H's 10th, PCP, K, ⋮

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

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 $f\colon \mathbb{N} \longrightarrow \mathbb{N}$ if
Alg(*input*) uses less than $f(|input|)$ units of TIME.

Complexity theory

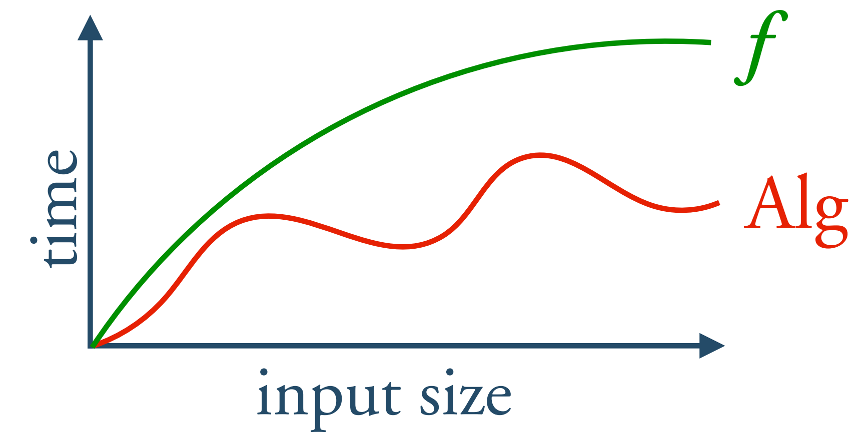
Domino H's 10th PCP K ⋮

What can be **mechanized?** ⇝ decidable/undecidable

How **hard** is it to mechanise? ⇝ complexity classes

usage of resources:
- time
- memory

Algorithm Alg is TIME-bounded by a function f: N ⟶ N if Alg(*input*) uses less than f(|*input*|) units of TIME.



Complexity theory

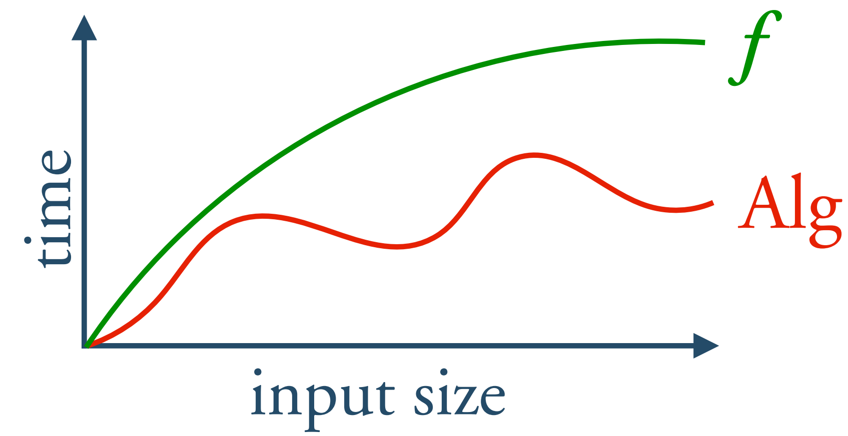
Domino, H's 10th, PCP, K, ⋮

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: N ⟶ N if **Alg**(*input*) uses less than $f(|input|)$ units of ~~TIME~~ SPACE.



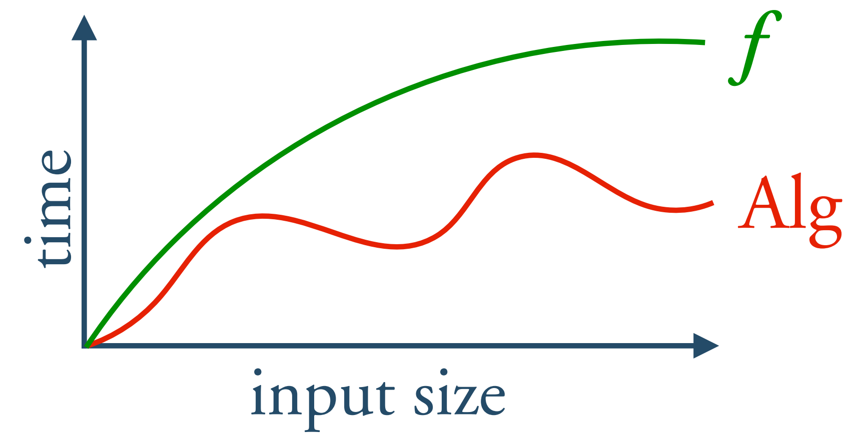
Complexity theory

What can be **mechanized**? $\rightsquigarrow$ decidable/undecidable → Domino, H's 10th, PCP, K, ⋮

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

→ usage of resources:
- time
- memory

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



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

Complexity theory

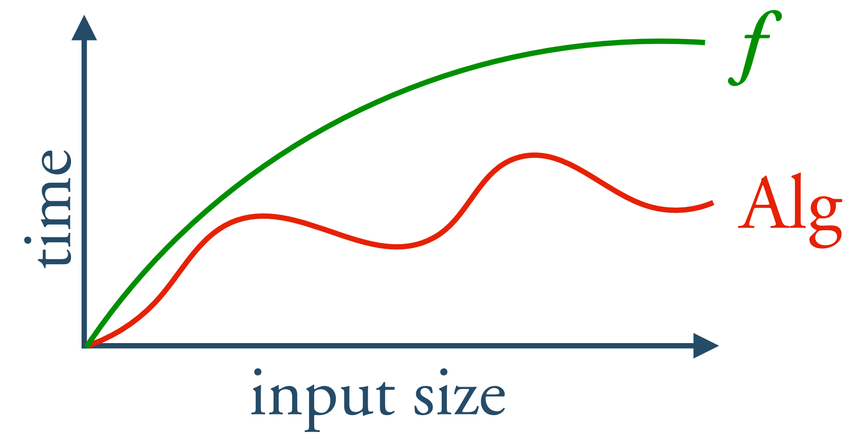
Domino, H's 10th, PCP, K, ⋮

What can be **mechanized**? ⇝ decidable/undecidable

How **hard** is it to mechanise? ⇝ complexity classes

usage of resources:
- time
- memory

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



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

- 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 $\phi(x_1, ..., x_n)$, 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 $\phi(x_1, ..., x_n)$, 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 $\models_{finite}$

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 $G \vDash_\alpha \phi$?

DECIDABLE $\rightsquigarrow$ foundations of the database industry

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

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

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

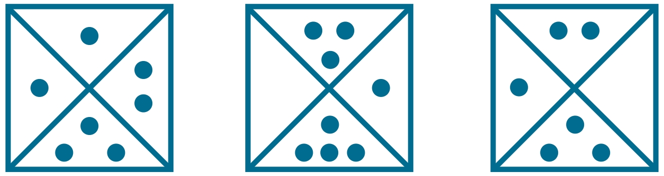
💀 UNDECIDABLE ⇝ both for $\models$ and $\models_{finite}$ [Trakhtenbrot '50]

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).

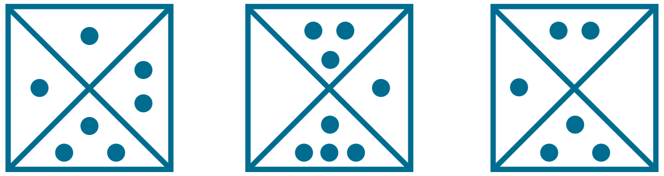
The (undecidable) Domino problem

Domino

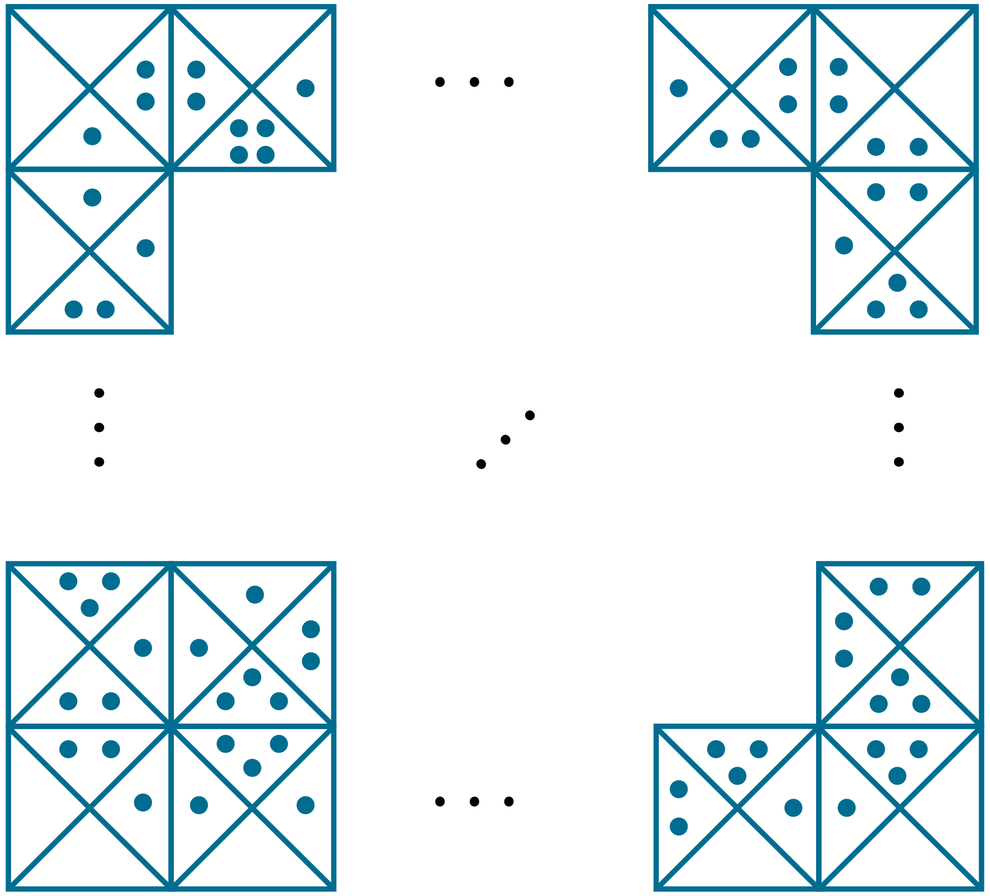
Input: 4-sided dominos: 

The (undecidable) Domino problem

Domino

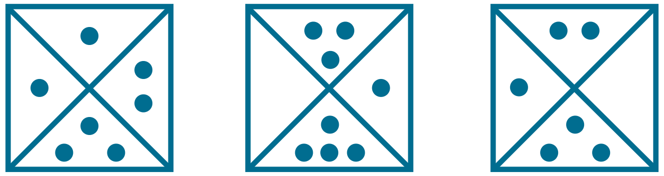
Input: 4-sided dominos: 

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

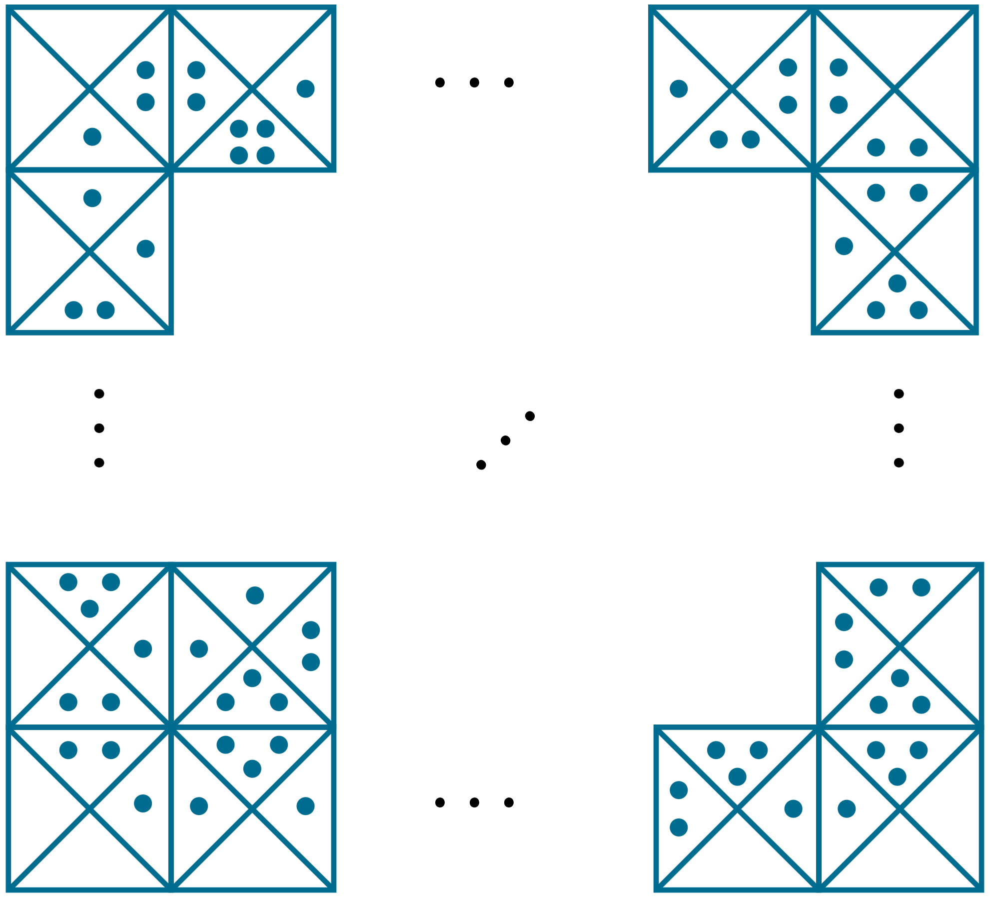


The (undecidable) Domino problem

Domino

Input: 4-sided dominos: 

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

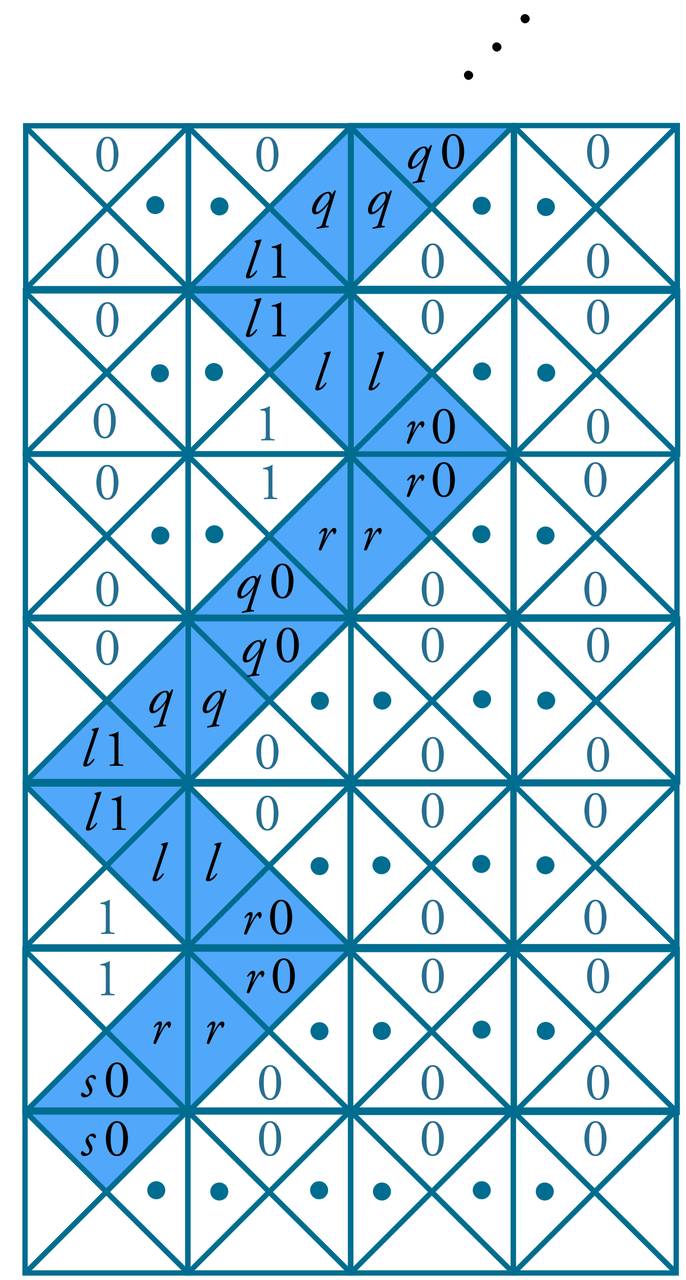


Rules: sides must match,
you can't rotate the dominos, but you can 'clone' them.

The (undecidable) Domino problem

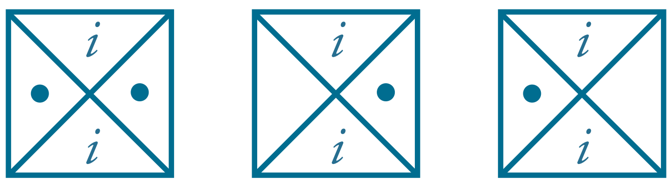
Domino - Why is it undecidable?

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

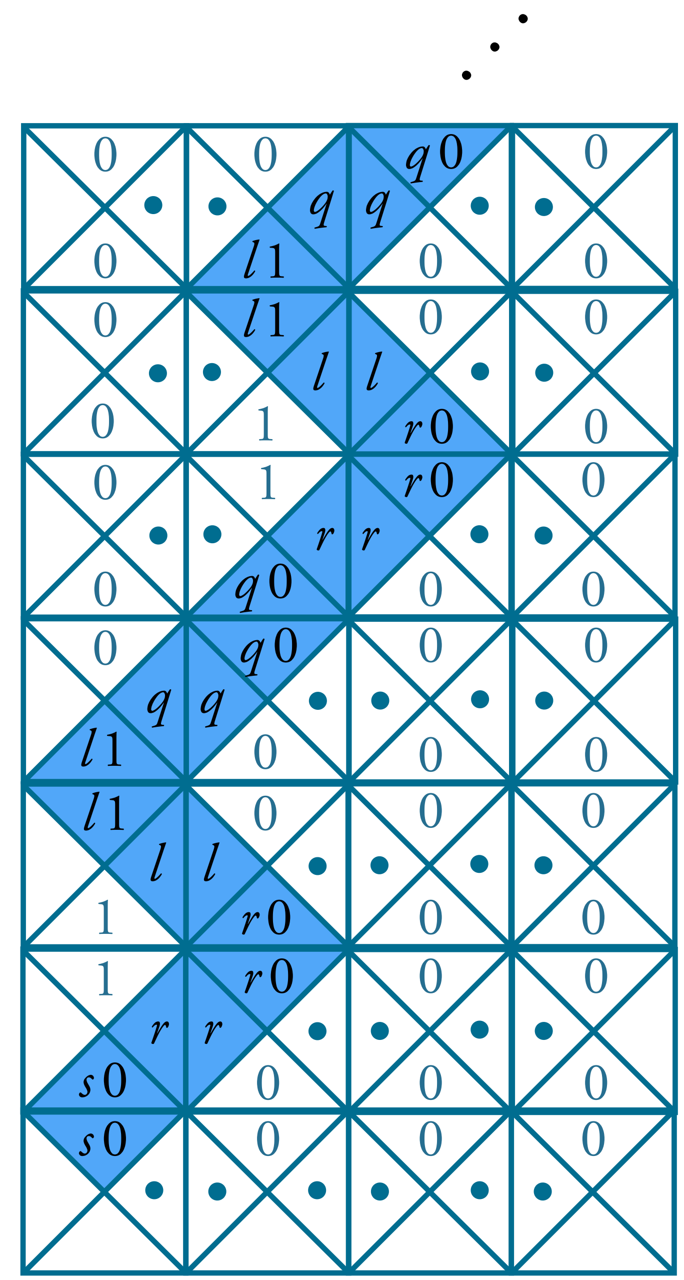


Domino - Why is it undecidable?

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



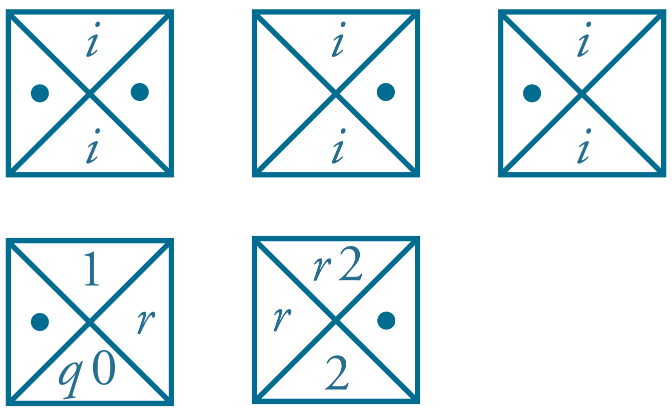
(head is elsewhere,
symbol is not modified)



The (undecidable) Domino problem

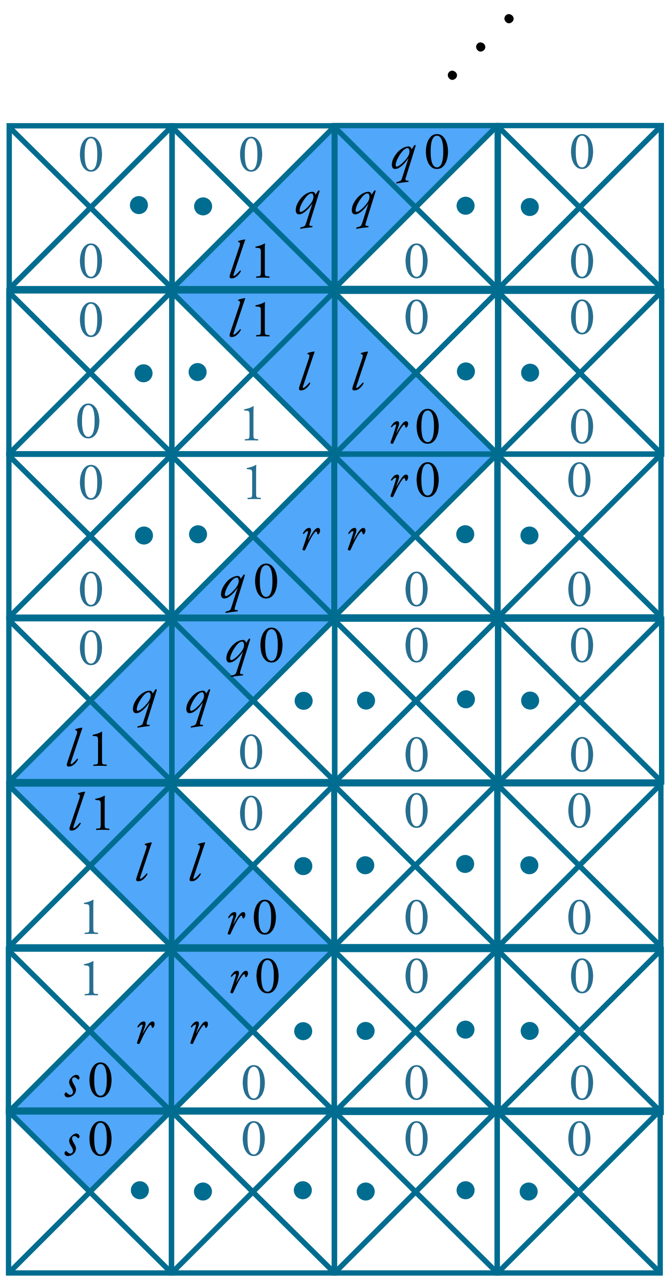
Domino - Why is it undecidable?

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



(head is elsewhere, symbol is not modified)

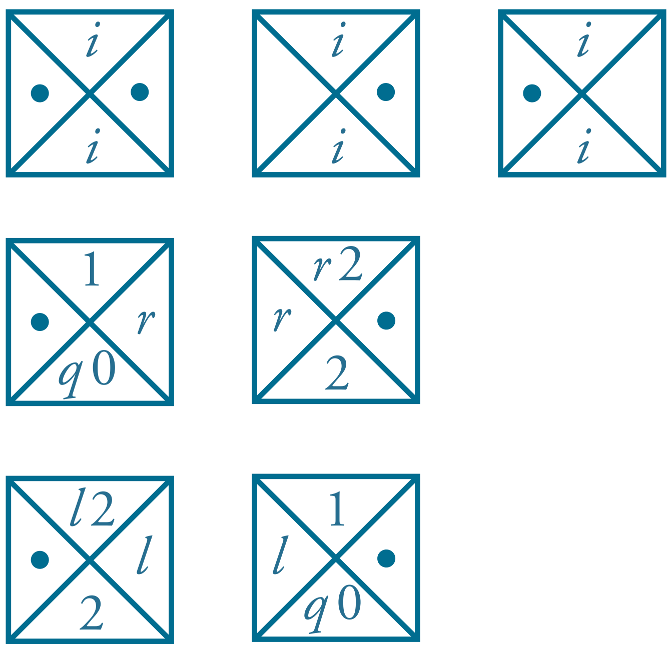
(head is here, symbol is rewritten, head moves right)



The (undecidable) Domino problem

Domino - Why is it undecidable?

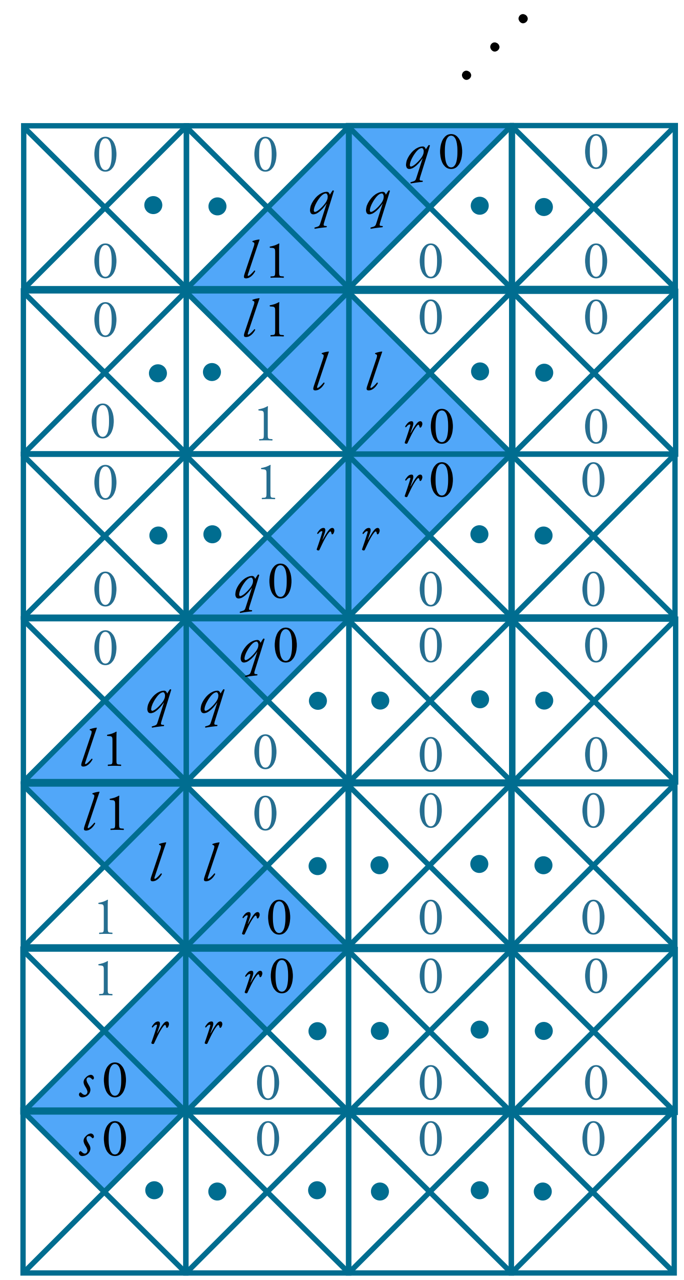
It can easily encode *halting* computations of Turing machines:



(head is elsewhere, symbol is not modified)

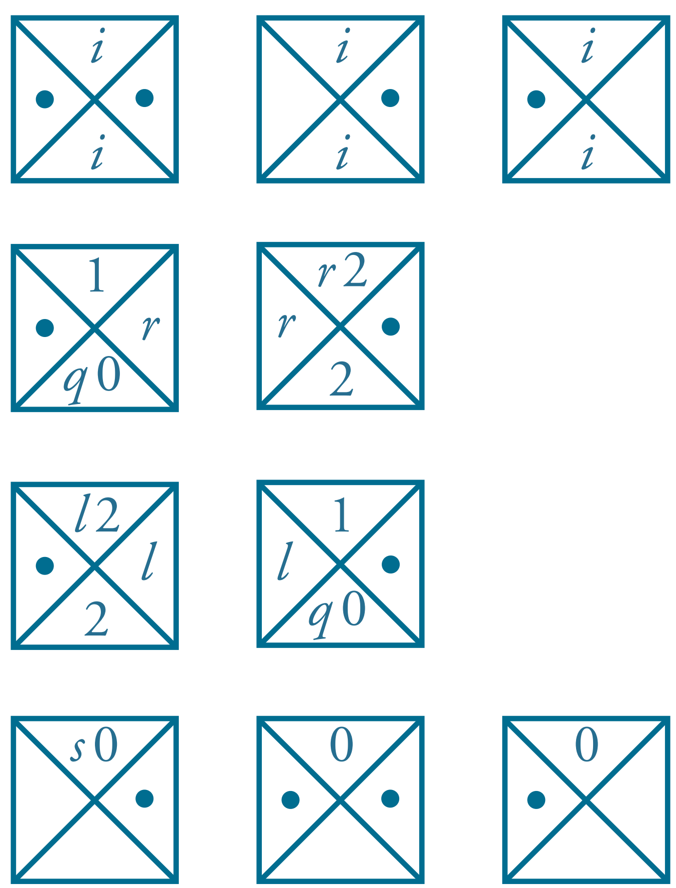
(head is here, symbol is rewritten, head moves right)

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



Domino - Why is it undecidable?

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

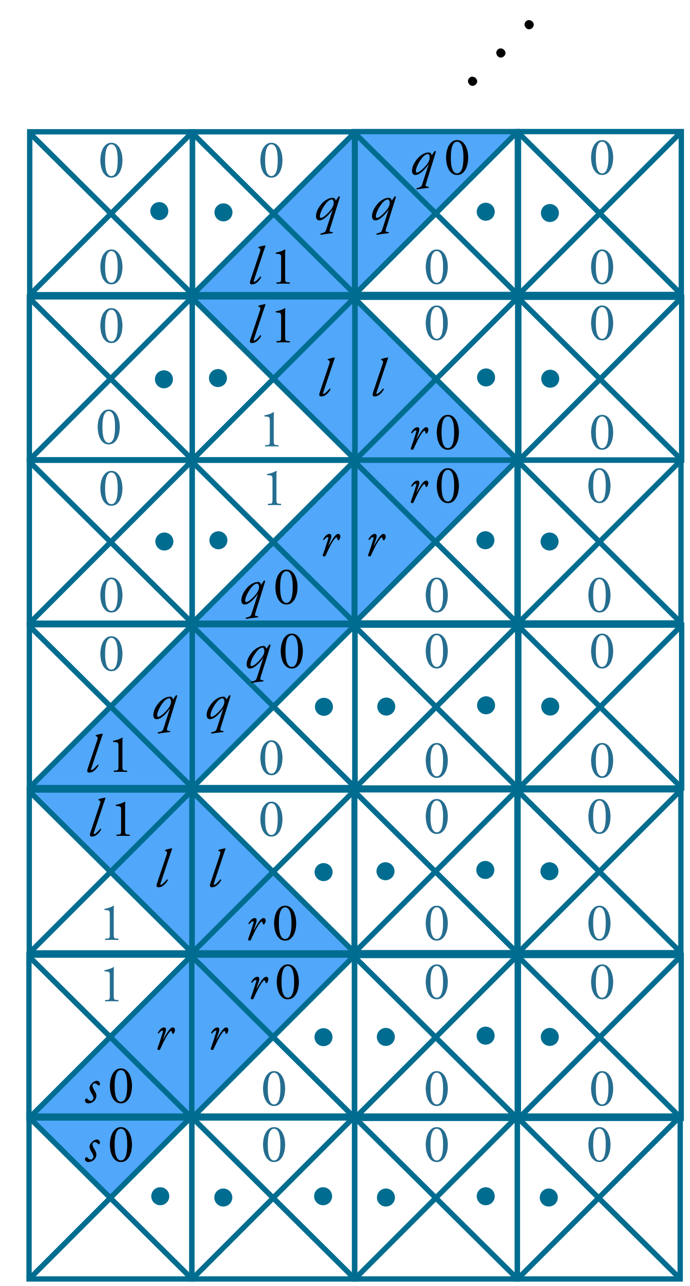


(head is elsewhere, symbol is not modified)

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

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

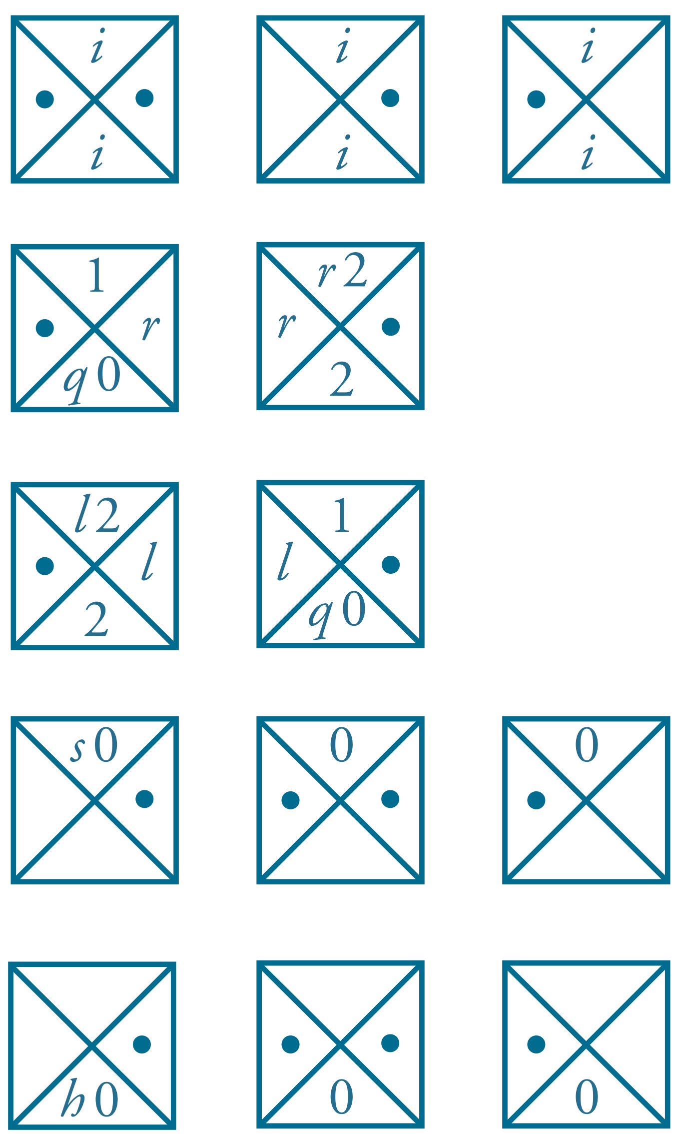
(initial configuration)



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)

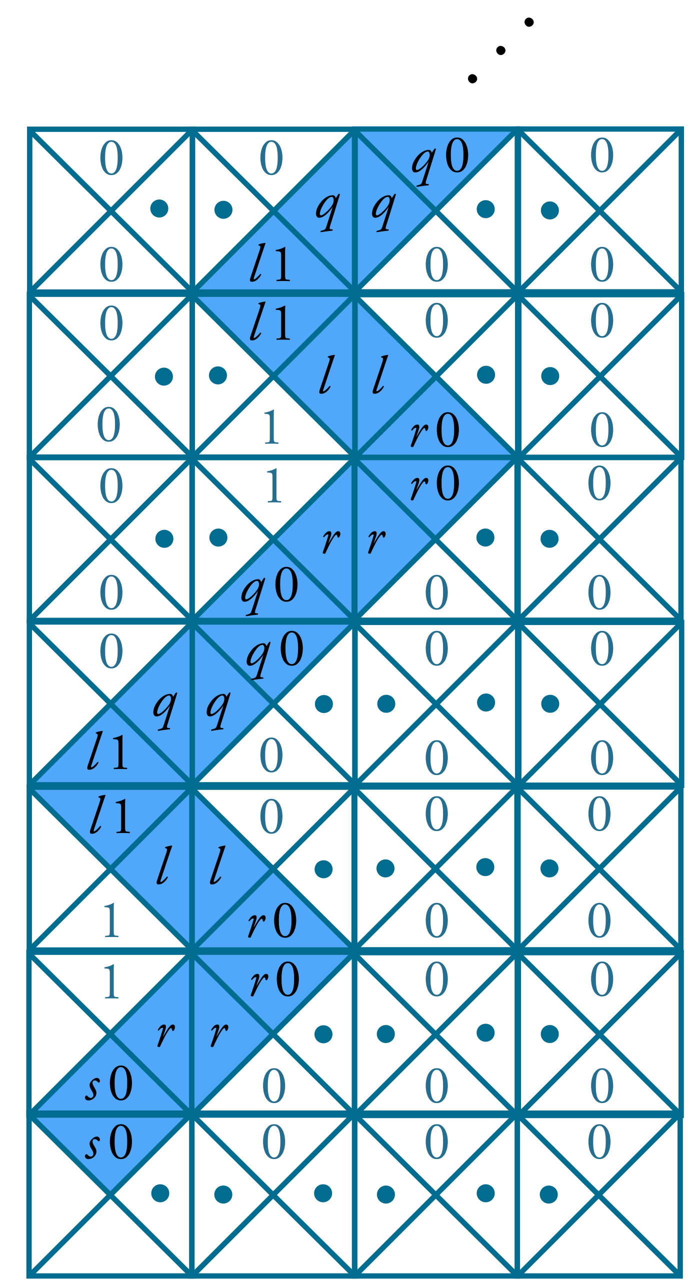
(head is here, symbol is rewritten, head moves right)

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

(initial configuration)

(halting configuration)

. . .

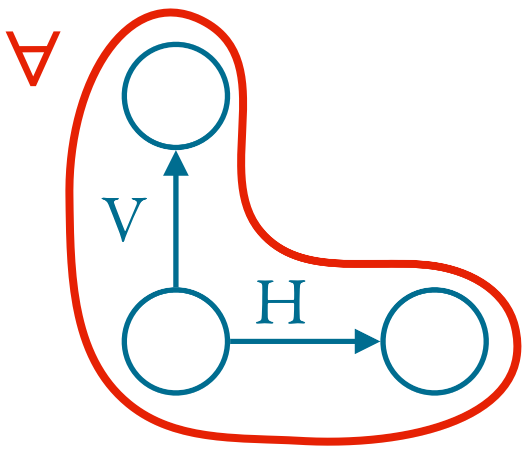


Domino ⇝ Sat-FO (domino has a solution iff ϕ satisfiable)

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

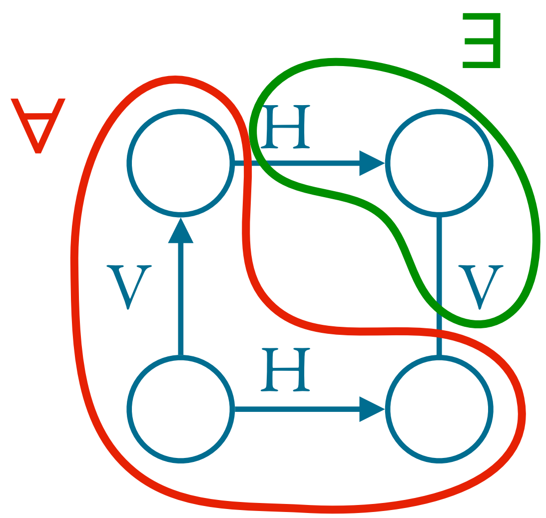
Domino ⇝ Sat-FO (domino has a solution iff ϕ satisfiable)

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



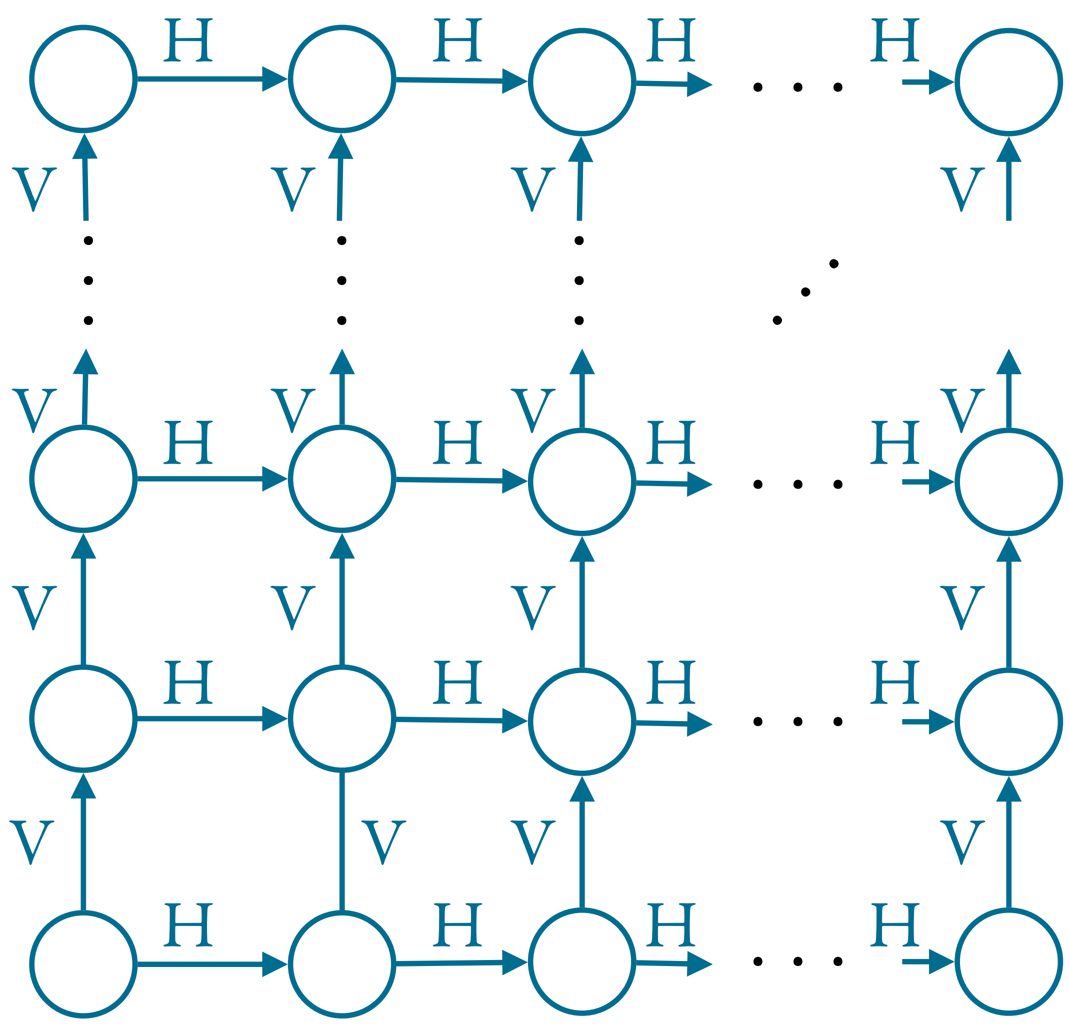
Domino ⇝ Sat-FO (domino has a solution iff ϕ satisfiable)

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



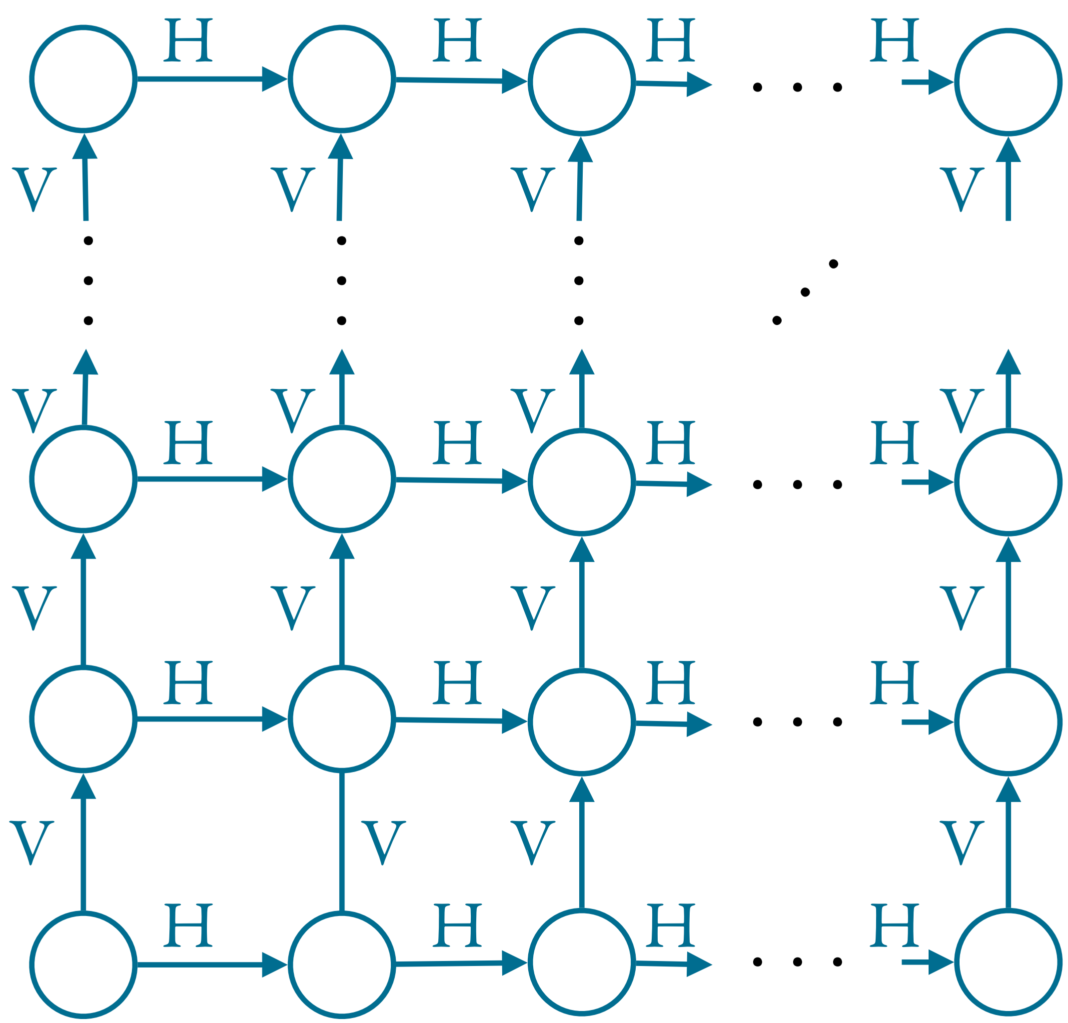
Domino ⇝ Sat-FO (domino has a solution iff ϕ satisfiable)

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



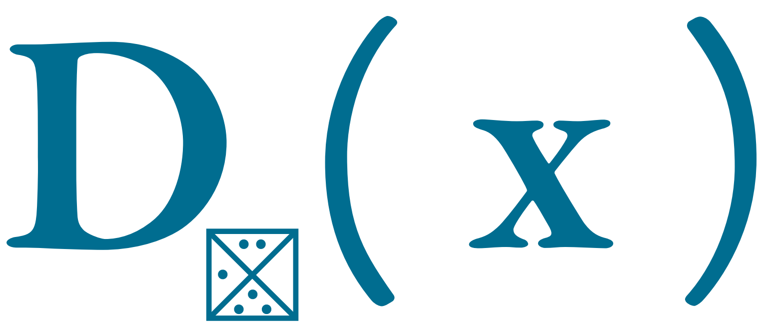
Domino ⇝ Sat-FO (domino has a solution iff ϕ satisfiable)

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



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

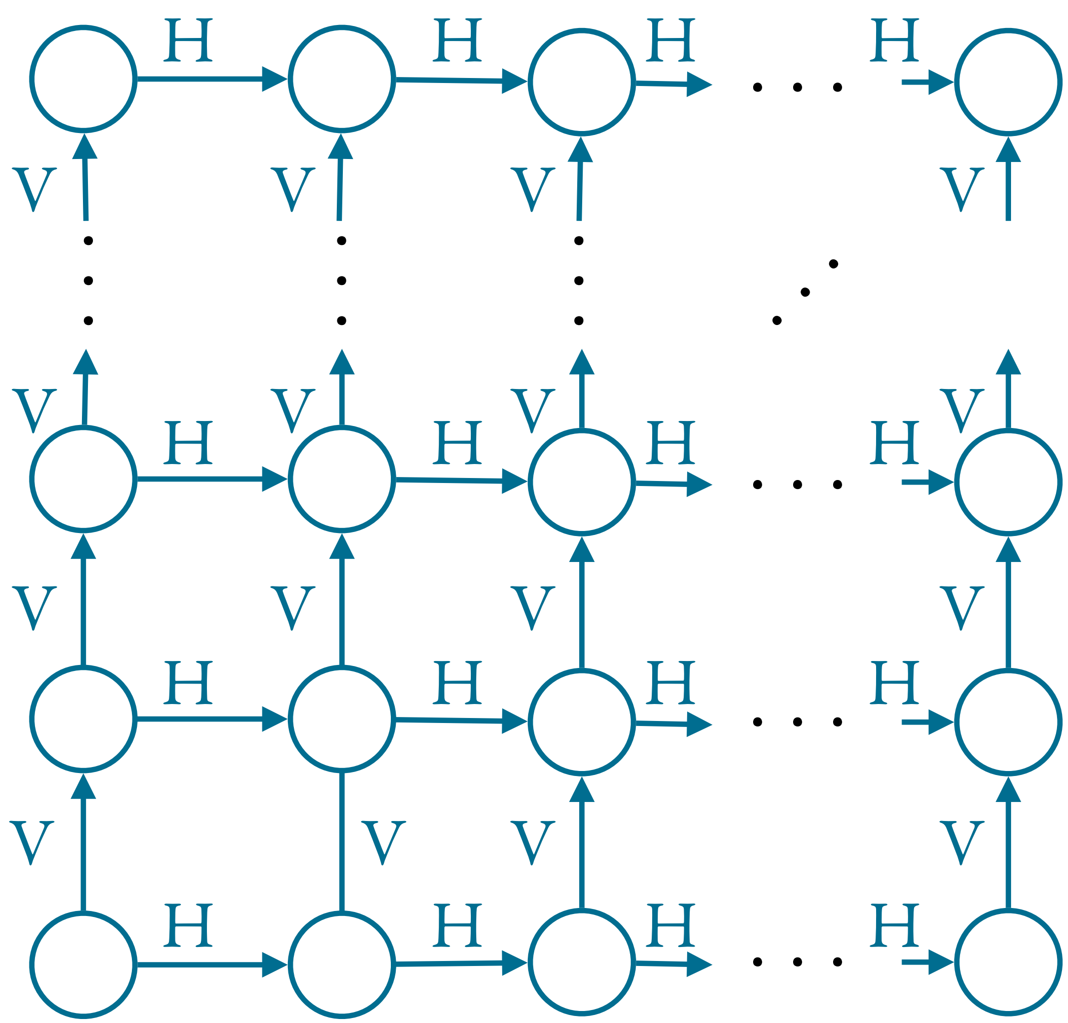
a unary relation



for each domino

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

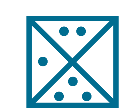
1. **There is a grid:** H(,) and V(,) are relations representing bijections such that…



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

a unary relation

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

for each domino 

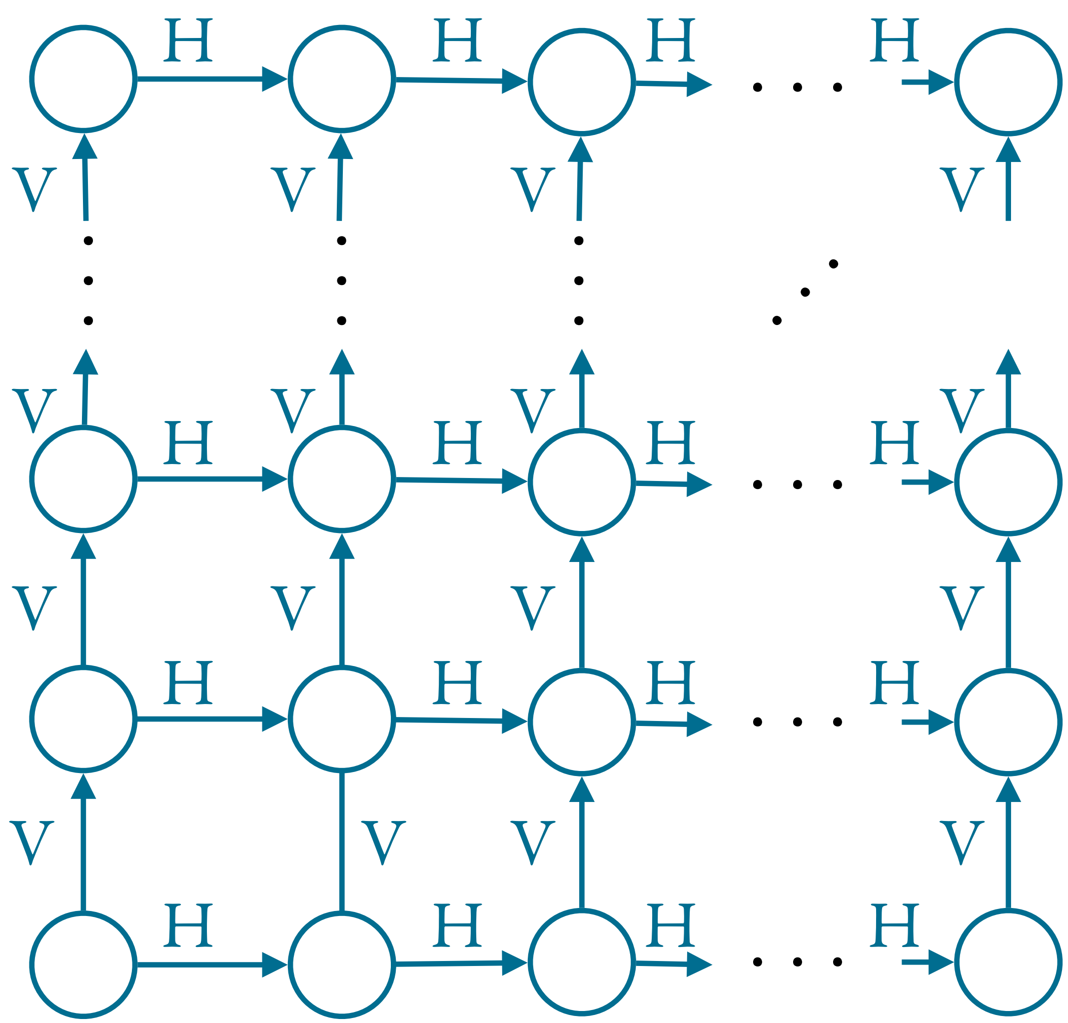
3. **Match the sides** $\forall x,y$

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

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

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)

4. **Borders are white.**

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:

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

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

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 $\alpha = \{x_1,\dots,x_n\} \longrightarrow V$

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

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

Evaluation problem for FO

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

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

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 $(\alpha(x_i),\alpha(x_j)) \in E$

- 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 $G \vDash_\alpha \psi'$

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

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

Question:
How much space
does it take?

Evaluation problem for FO

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

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 ⇝ 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: $\begin{cases} \phi(x_1,\dots,x_n) \\ G = (V,E) \\ \alpha = \{x_1,\dots,x_n\} \longrightarrow V \end{cases}$ Output: $G \models_\alpha \phi$?

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

- 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 $G \models_\alpha \psi'$
  ⇝ MAX(SPACE($G \models_\alpha \psi$)), SPACE($G \models_\alpha \psi'$)))

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

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

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

  use 4 pointers $\rightsquigarrow$ LOGSPACE

- 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 $G \models_\alpha \psi'$

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

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

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

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

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

- If $\phi(x_1,\dots,x_n) = E(x_i,x_j)$:
  answer YES iff $(\alpha(x_i),\alpha(x_j)) \in E$ — use 4 pointers ⇝ LOGSPACE
- 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 $G \models_\alpha \psi'$ — ⇝ MAX(SPACE($G \models_\alpha \psi$)), SPACE($G \models_\alpha \psi'$)))
- If $\phi(x_1,\dots,x_n) = \neg\psi(x_1,\dots,x_n)$:
  answer NO iff $G \models_\alpha \psi$ — ⇝ SPACE($G \models_\alpha \psi$))
- 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 $G \models_{\alpha'} \psi$. — ⇝ $2 \cdot \log(|G|)$ + SPACE($G \models_{\alpha'} \psi$)

Question:
How much space does it take?

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

- 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 $G \models_\alpha \psi'$
  ⇝ $\mathrm{MAX}(\,\mathrm{SPACE}(G \models_\alpha \psi)), \mathrm{SPACE}(G \models_\alpha \psi'))\,)$

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

- 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 $G \models_{\alpha'} \psi$.
  ⇝ $2 \cdot \log(|G|) + \mathrm{SPACE}(G \models_{\alpha'} \psi\,)$

Question:
How much space does it take?

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

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

O(log(|data|)·|query|) space

PSPACE combined and query complexity

LOGSPACE data complexity

Bounded Degree

$\mathcal{D}_k$—the class of structures $\mathbb{A}$ in which every element has at most k neighbours in $G\mathbb{A}$.

$O(f(k) \cdot n)$

Theorem (Seese)

For every sentence φ of FO and every k there is a linear time algorithm which, given a structure $\mathbb{A} \in \mathcal{D}_k$ determines whether $\mathbb{A} \models \varphi$.

Note: this is not true for MSO unless P = NP.

The proof is based on *locality* of first-order logic. Specifically, *Hanf's theorem*.

Hanf Types

For an element a in a structure $\mathbb{A}$, define

> $N_r^{\mathbb{A}}(a)$—*the substructure of* $\mathbb{A}$ *generated by the elements whose distance from* a *(in* $G\mathbb{A}$*) is at most* r*.*

We say $\mathbb{A}$ and $\mathbb{B}$ are *Hanf equivalent* with radius r and threshold q ($\mathbb{A} \simeq_{r,q} \mathbb{B}$) if, for every $a \in A$ the two sets

$$\{a' \in A \mid N_r^{\mathbb{A}}(a) \cong N_r^{\mathbb{A}}(a')\} \quad \text{and} \quad \{b \in B \mid N_r^{\mathbb{A}}(a) \cong N_r^{\mathbb{B}}(b)\}$$

either have the same size or both have size greater than q;
and, similarly for every $b \in B$.

Hanf Locality Theorem

Theorem (Hanf)
For every vocabulary σ and every p there qre r and q such that for any σ-structures $\mathbb{A}$ and $\mathbb{B}$: if $\mathbb{A} \simeq_{r,q} \mathbb{B}$ then $\mathbb{A} \equiv_p \mathbb{B}$.

For $\mathbb{A} \in \mathcal{D}_k$:

$N_r^{\mathbb{A}}(a)$ has at most $k^r + 1$ elements

each $\simeq_{r,q}$ has finite index.

Each $\simeq_{r,q}$-class t can be characterised by a finite table, I_t, giving isomorphism types of neighbourhoods and numbers of their occurrences up to threshold q.

Satisfaction on $\mathcal{D}_k$

For a sentence φ of FO, we can compute a set of tables $\{I_1, \ldots, I_s\}$ describing $\simeq_{r,q}$-classes consistent with it.
This computation is independent of any structure $\mathbb{A}$.

Given a structure $\mathbb{A} \in \mathcal{D}_k$,

for each a, determine the isomorphism type of $N_r^{\mathbb{A}}(a)$

construct the table describing the $\simeq_{r,q}$-class of $\mathbb{A}$.

compare against $\{I_1, \ldots, I_s\}$ to determine whether $\mathbb{A} \models \varphi$.

For fixed k, r, q, this requires time *linear* in the size of $\mathbb{A}$.

Note: evaluation for FO is in $O(f(l, k)n)$.