Descriptive complexity of linear algebra

Bjarki Holm

UNIVERSITY OF CAMBRIDGE

Logical Approaches to Barriers in Computing & Complexity II

Isaac Newton Institute

2012

Overview

Study definability of natural problems in linear algebra and expressiveness of logics with algebraic operators.

- Background & motivation
- Descriptive complexity of problems in linear algebra
- Logics with matrix-rank operators
- Pebble games for rank logics & the Weisfeiler-Lehman method

Overview

Study definability of natural problems in linear algebra and expressiveness of logics with algebraic operators.

- Background & motivation
- Descriptive complexity of problems in linear algebra
- Logics with matrix-rank operators
- Pebble games for rank logics & the Weisfeiler-Lehman method

A logic for **NP**

ESO — Existential second-order logic

Second-order variables existentially quantified, followed by a first-order formula:

$$\exists R_1, \ldots, R_k \,.\, \varphi(R_1, \ldots, R_k)$$

A logic for **NP**

A decision problem is in **NP** if and only if it can be defined in ESO.

Fagin (1974)

ESO — Existential second-order logic

Second-order variables existentially quantified, followed by a first-order formula:

$$\exists R_1, \dots, R_k \,.\, \varphi(R_1, \dots, R_k)$$

A logic for **NP**

A decision problem is in **NP** if and only if it can be defined in ESO.

Fagin (1974)

ESO — Existential second-order logic

Second-order variables existentially quantified, followed by a first-order formula:

$$\underbrace{\exists R_1, \ldots, R_k}_{\text{“guess”}} . \, \varphi(R_1, \ldots, R_k)$$

A logic for **NP**

A decision problem is in **NP** if and only if it can be defined in ESO.

Fagin (1974)

ESO — Existential second-order logic

Second-order variables existentially quantified, followed by a first-order formula:

$$\underbrace{\exists R_1, \ldots, R_k}_{\text{“guess”}} \,.\, \underbrace{\varphi(R_1, \ldots, R_k)}_{\text{“verify”}}$$

A logic for **NP**

A decision problem is in **NP** if and only if it can be defined in ESO.

Fagin (1974)

ESO — Existential second-order logic

Second-order variables existentially quantified, followed by a first-order formula:

$$\underbrace{\exists R_1, \ldots, R_k}_{\text{“guess”}} \,.\, \underbrace{\varphi(R_1, \ldots, R_k)}_{\text{“verify”}}$$

Is there a logic for PTIME?

A logic for **PTIME**?

Fixed-point logic captures PTIME on ordered structures

FP is first-order logic with an inflationary fixed-point operator.

A property P of ordered structures can be *decided* in PTIME if and only if P can be *defined* by a sentence of FP.

Immerman-Vardi (1982)

Fixed-point logic captures PTIME on ordered structures

FP is first-order logic with an inflationary fixed-point operator.

A property P of ordered structures can be *decided* in PTIME if and only if P can be *defined* by a sentence of FP.

Immerman-Vardi (1982)

Ordered structure: Vocabulary contains a binary symbol "$\leqslant$" interpreted as a total ordering of the vertices.

Fixed-point logic captures PTIME on ordered structures

FP is first-order logic with an inflationary fixed-point operator.

A property P of ordered structures can be *decided* in PTIME if and only if P can be *defined* by a sentence of FP.

Immerman-Vardi (1982)

Fixed-point logic captures PTIME on ordered structures

FP is first-order logic with an inflationary fixed-point operator.

A property P of ordered structures can be *decided* in PTIME if and only if P can be *defined* by a sentence of FP.

Immerman-Vardi (1982)

- On unordered structures, FP cannot even express if a graph has an even or odd number of vertices.

Fixed-point logic captures PTIME on ordered structures

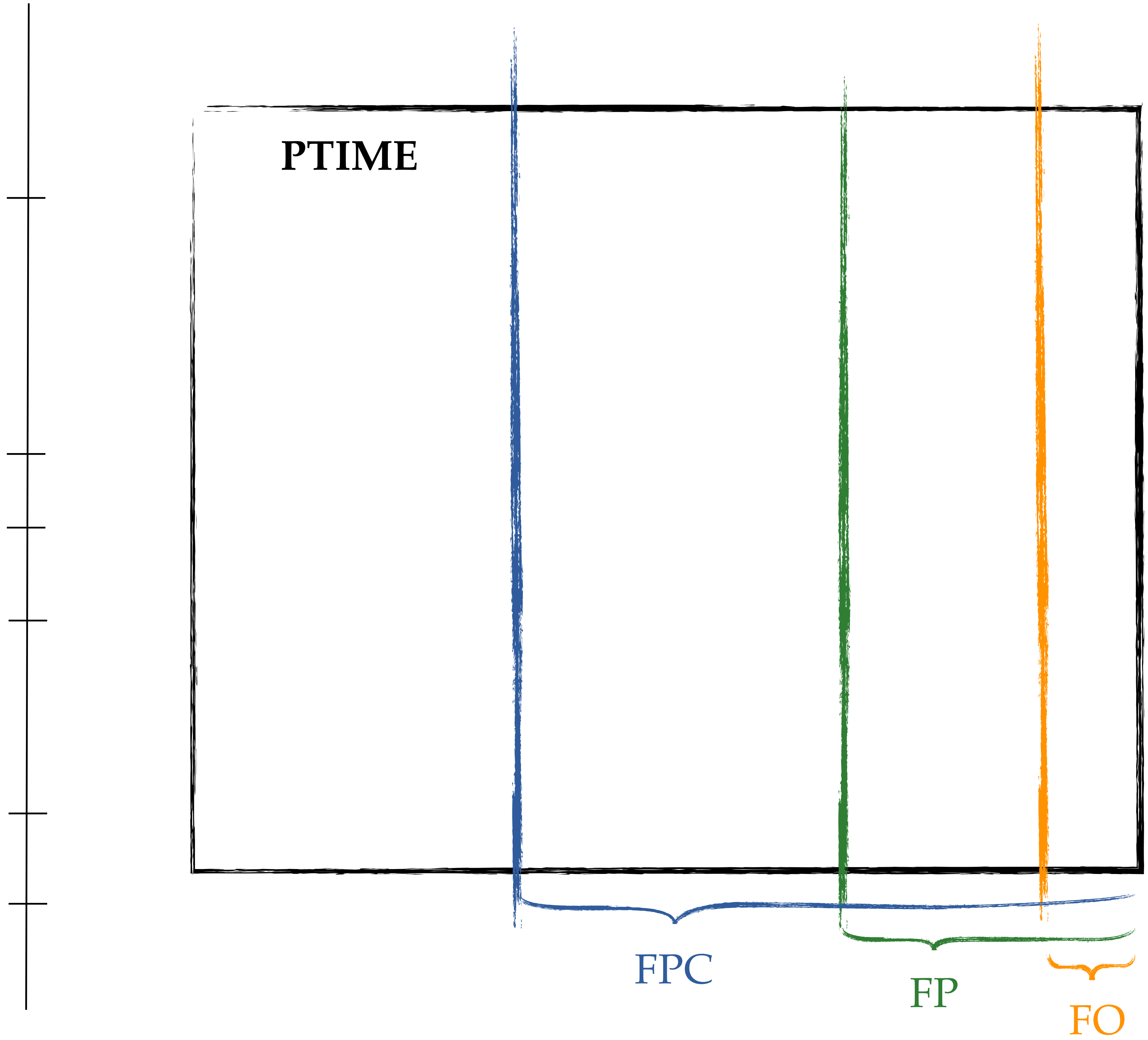
FP is first-order logic with an inflationary fixed-point operator.

A property P of ordered structures can be *decided* in PTIME if and only if P can be *defined* by a sentence of FP.

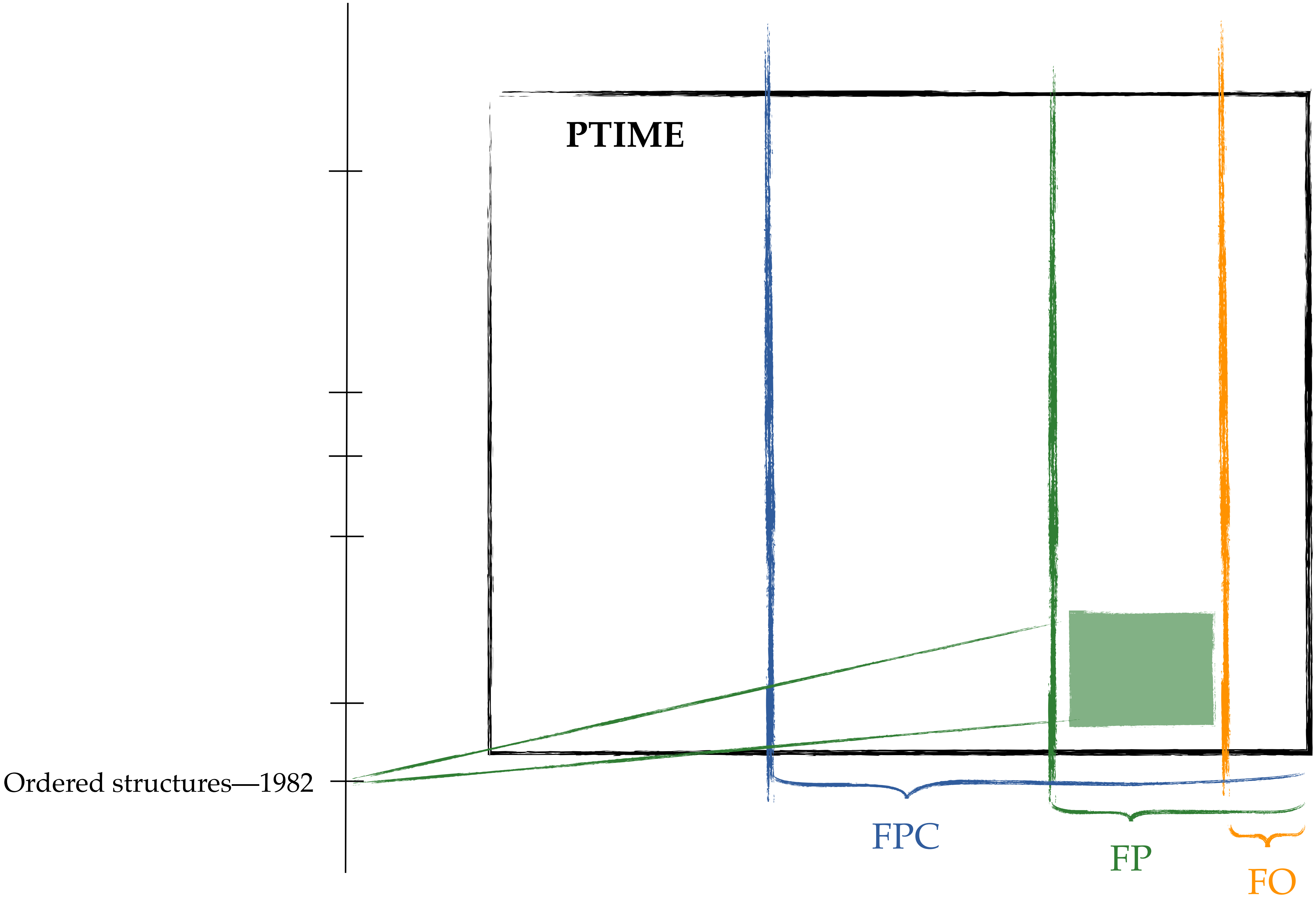
Immerman-Vardi (1982)

- On unordered structures, FP cannot even express if a graph has an even or odd number of vertices.
- *Fixed-point logic with counting* (FPC) is FP together with terms that count the number of solutions to formulas.

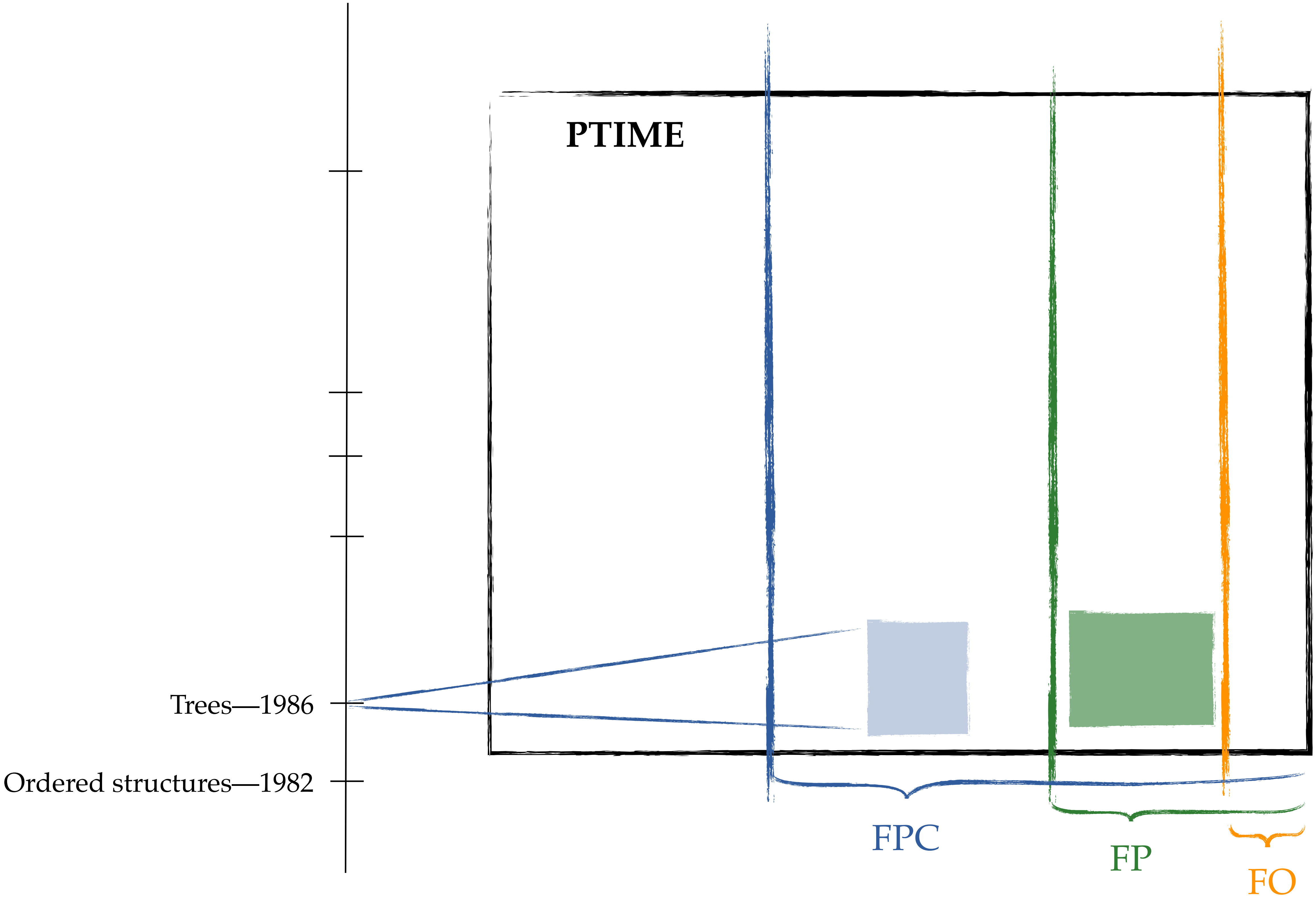
FPC captures **PTIME** on...



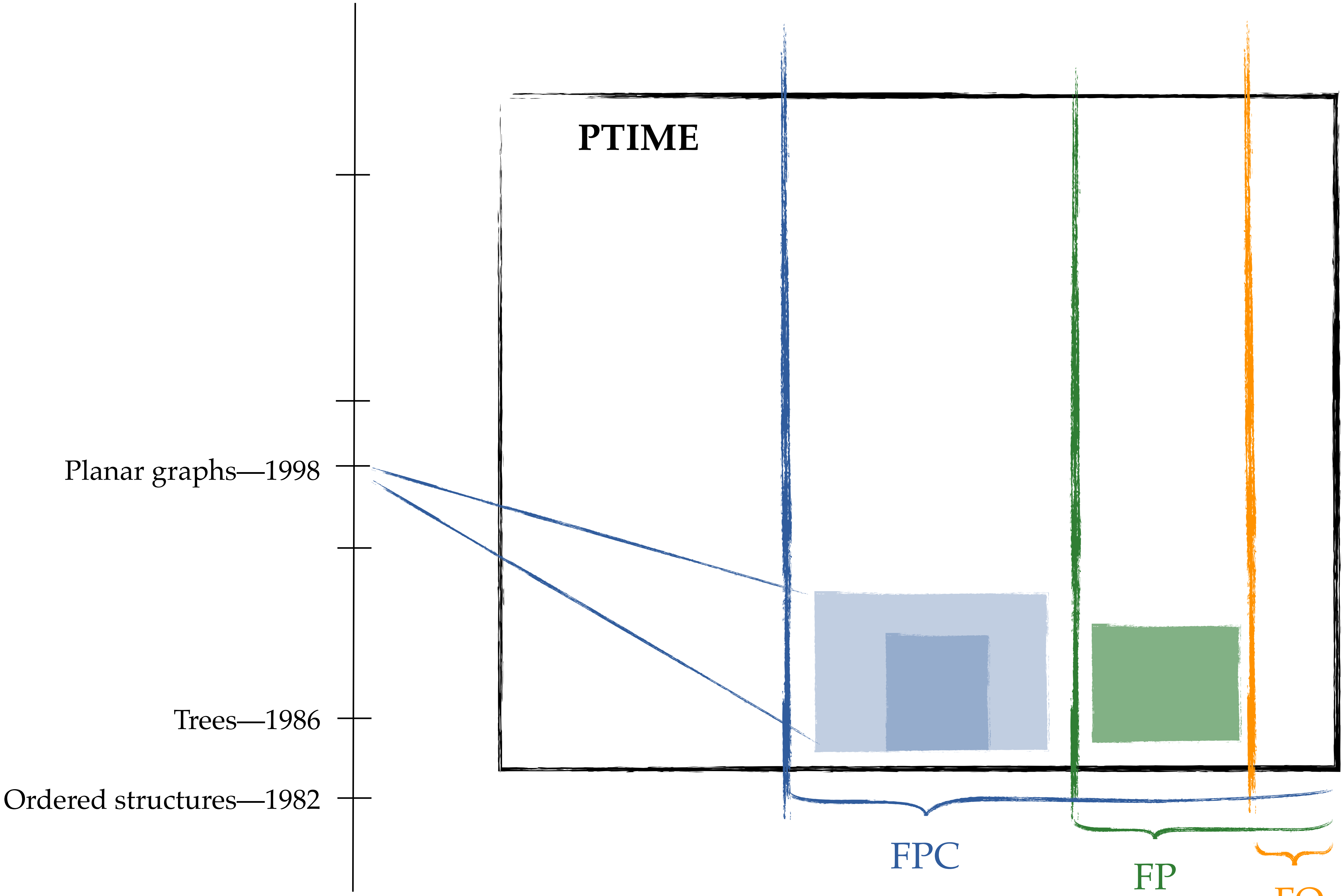
FPC captures **PTIME** on...



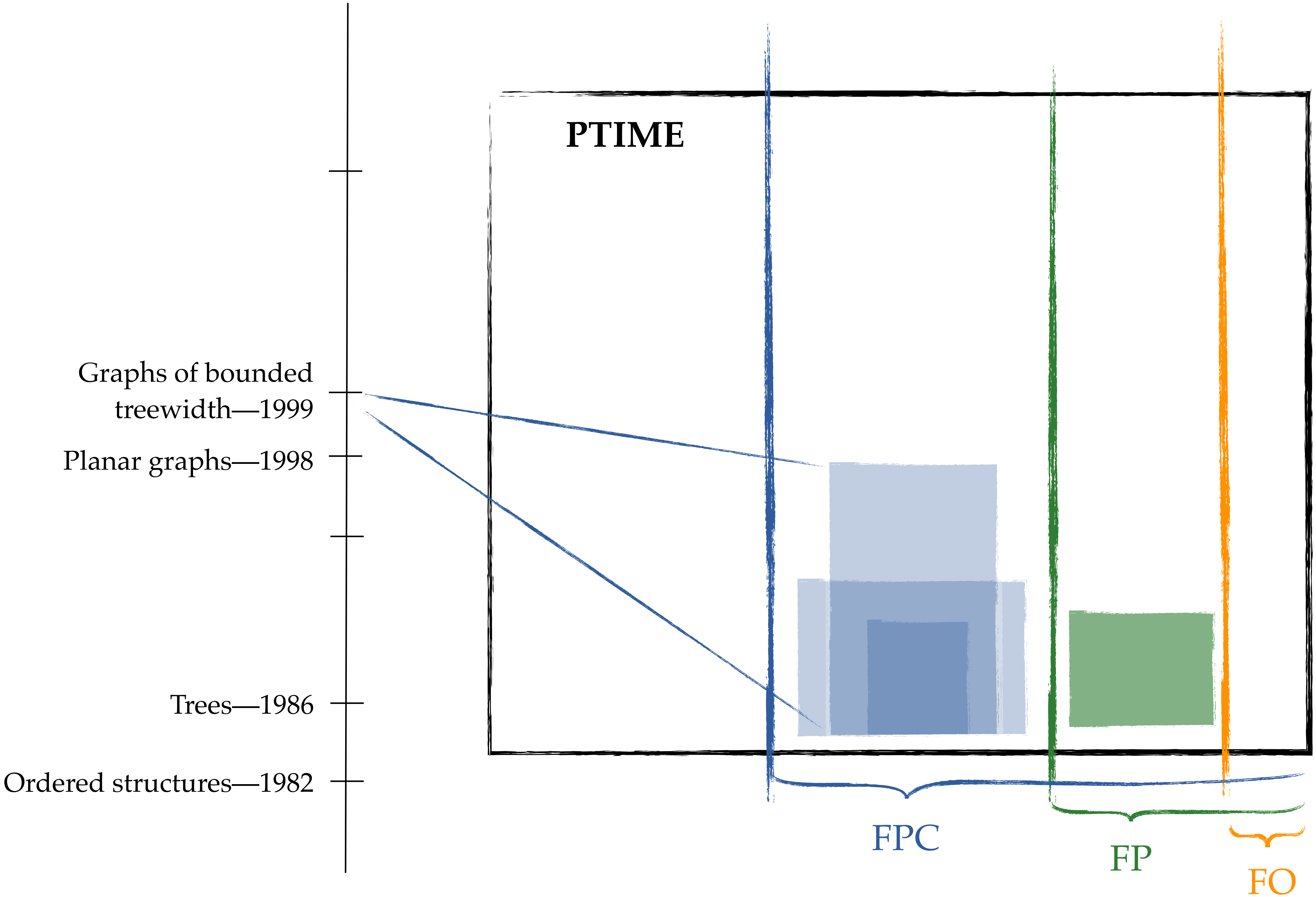
FPC captures **PTIME** on...



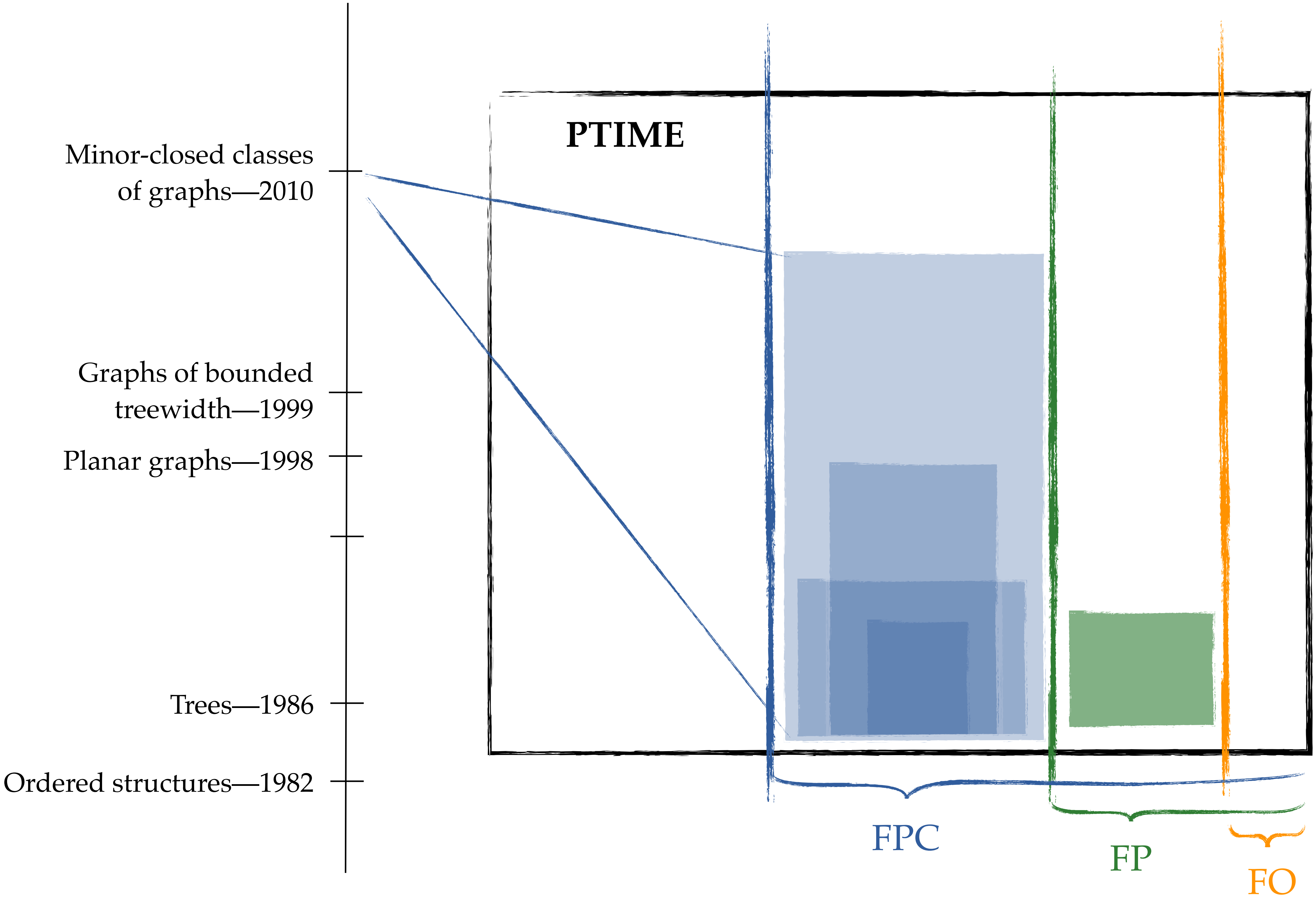
FPC captures **PTIME** on...



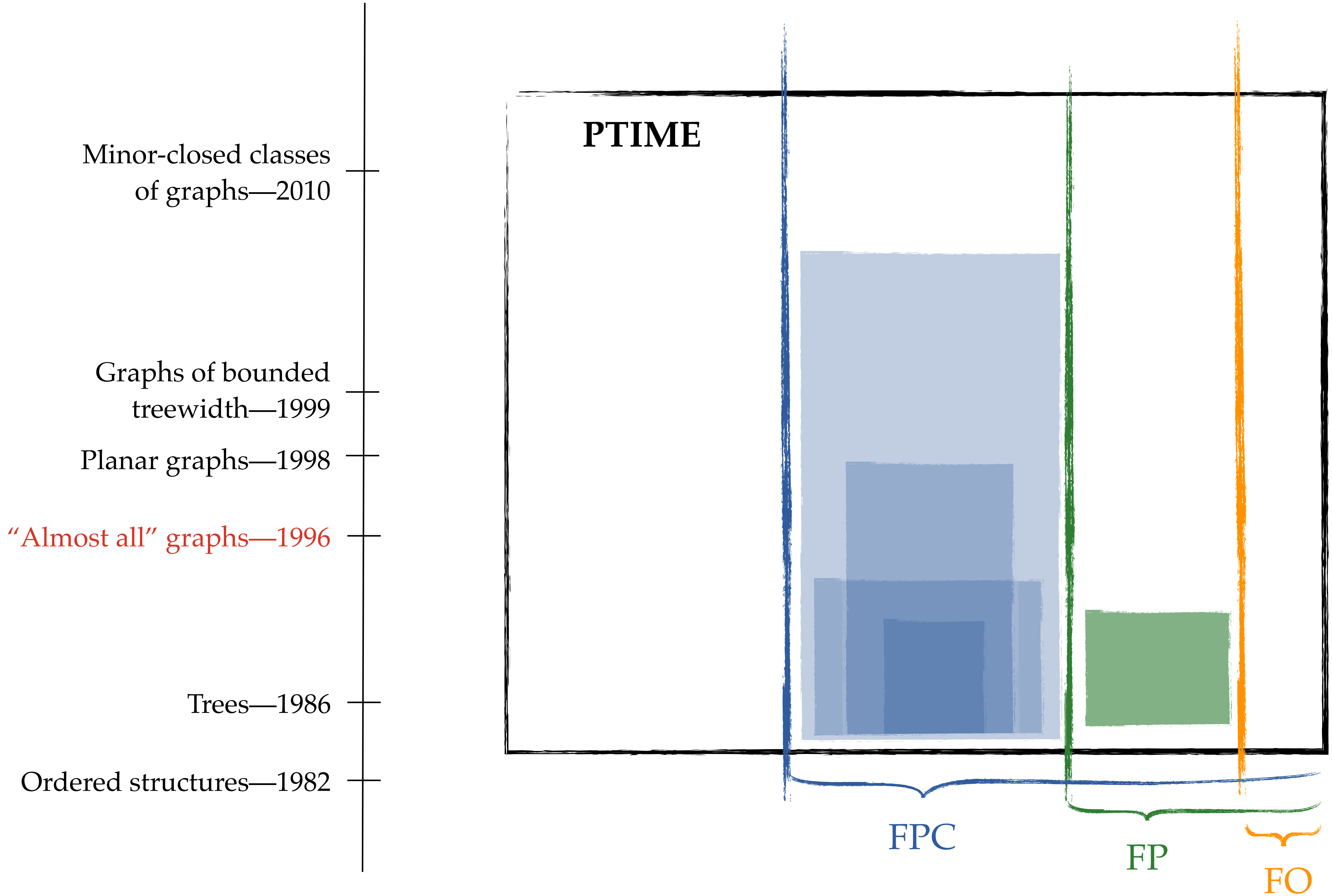
FPC captures **PTIME** on...



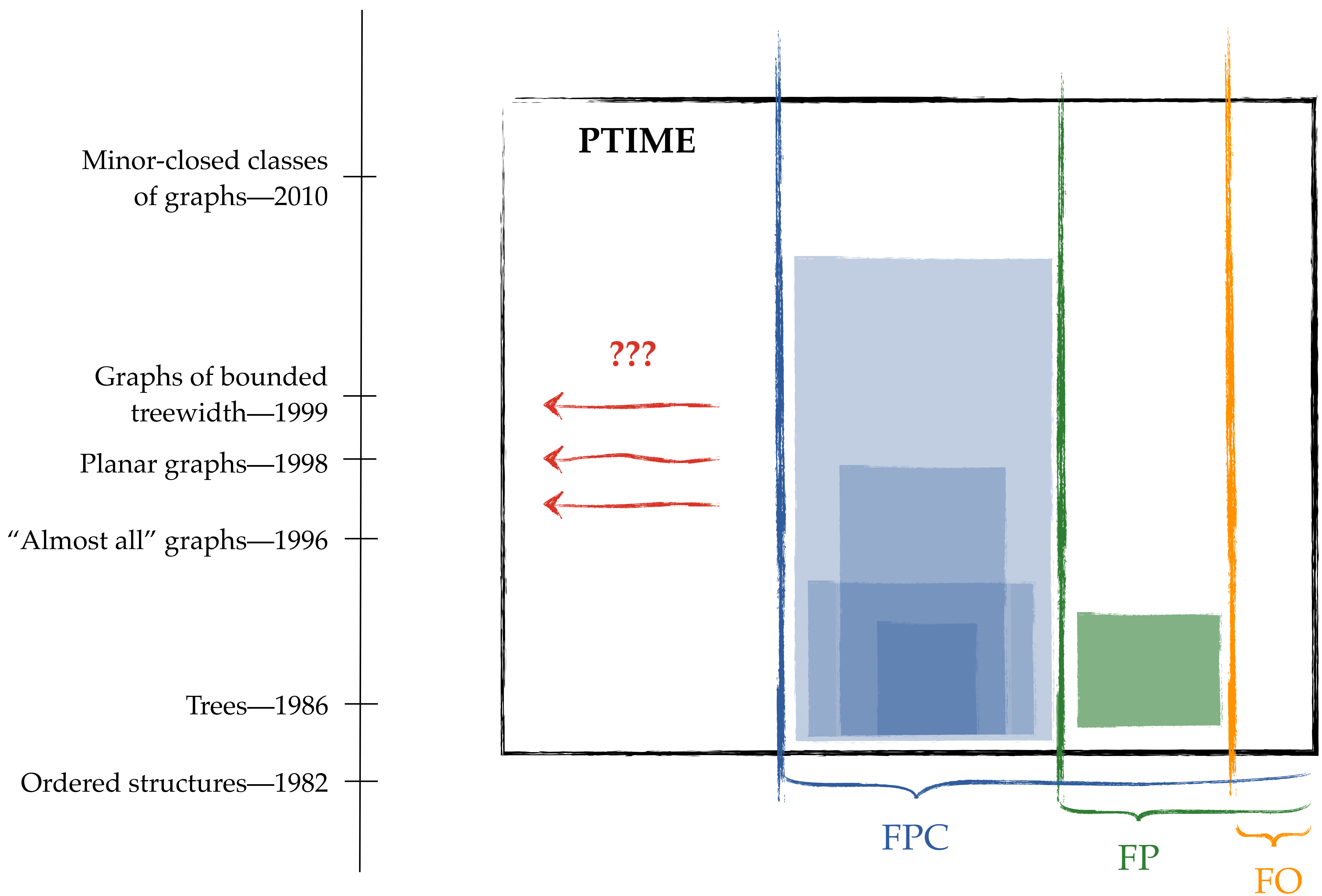
FPC captures **PTIME** on...



FPC captures **PTIME** on...



FPC captures **PTIME** on... all graphs?



Proving non-definability in FPC

C^k — first-order logic with variables x_1, ..., x_k and counting quantifiers of the form $\exists^{\geq i} x \,.\, \varphi$

Proving non-definability in FPC

C^k — first-order logic with variables $x_1, \ldots, x_k$ and counting quantifiers of the form $\exists^{\geq i} x \,.\, \varphi$

1. Every formula of FPC is invariant under C^k-equivalence, for some k.

Proving non-definability in FPC

C^k — first-order logic with variables $x_1, \ldots, x_k$ and counting quantifiers of the form $\exists^{\geq i} x \,.\, \varphi$

1. Every formula of FPC is invariant under C^k-equivalence, for some k.
2. C^k-equivalence can be characterised by a k-pebble bijection game (a variant of Ehrenfeucht–Fraïsse)

Proving non-definability in FPC

C^k — first-order logic with variables $x_1, \ldots, x_k$ and counting quantifiers of the form $\exists^{\geq i} x \,.\, \varphi$

1. Every formula of FPC is invariant under C^k-equivalence, for some k.
2. C^k-equivalence can be characterised by a k-pebble bijection game (a variant of Ehrenfeucht–Fraïsse)

G and H agree on all sentences of C^k	iff	Duplicator has a winning strategy in the k-pebble bijection game on G and H

Proving non-definability in FPC

C^k — first-order logic with variables $x_1, \ldots, x_k$ and counting quantifiers of the form $\exists^{\geq i} x \,.\, \varphi$

Proving non-definability in FPC

C^k — first-order logic with variables x_1, ..., x_k and counting quantifiers of the form $\exists^{\geq i} x \, . \, \varphi$

To show that a property **P** is not definable in FPC:

Proving non-definability in FPC

C^k — first-order logic with variables $x_1, \ldots, x_k$ and counting quantifiers of the form $\exists^{\geq i} x \,.\, \varphi$

To show that a property **P** is not definable in FPC:

For each k, exhibit a pair of graphs G_k and H_k for which

Proving non-definability in FPC

C^k — first-order logic with variables $x_1, \ldots, x_k$ and counting quantifiers of the form $\exists^{\geq i} x \,.\, \varphi$

To show that a property **P** is not definable in FPC:

For each k, exhibit a pair of graphs G_k and H_k for which

- G_k has property **P** but H_k does not; and

Proving non-definability in FPC

C^k — first-order logic with variables $x_1, \ldots, x_k$ and counting quantifiers of the form $\exists^{\geq i} x \,.\, \varphi$

To show that a property **P** is not definable in FPC:

For each k, exhibit a pair of graphs G_k and H_k for which

- G_k has property **P** but H_k does not; and
- Duplicator wins the k-pebble game on G_k and H_k.

Proving non-definability in FPC

C^k — first-order logic with variables x_1, ..., x_k and counting quantifiers of the form $\exists^{\geq i} x \, . \, \varphi$

1. Every formula of FPC is invariant under C^k-equivalence, for some k.
2. C^k-equivalence can be characterised by a k-pebble bijection game (a variant of Ehrenfeucht–Fraïsse)

Proving non-definability in FPC

C^k — first-order logic with variables $x_1, \ldots, x_k$ and counting quantifiers of the form $\exists^{\geq i} x \, . \, \varphi$

1. Every formula of FPC is invariant under C^k-equivalence, for some k.
2. C^k-equivalence can be characterised by a k-pebble bijection game (a variant of Ehrenfeucht–Fraïsse)

Facts

Proving non-definability in FPC

C^k — first-order logic with variables $x_1, \ldots, x_k$ and counting quantifiers of the form $\exists^{\geq i} x \,.\, \varphi$

1. Every formula of FPC is invariant under C^k-equivalence, for some k.
2. C^k-equivalence can be characterised by a k-pebble bijection game (a variant of Ehrenfeucht–Fraïsse)

Facts

- For each k, we can decide the winner of the k-pebble game in polynomial time.

Proving non-definability in FPC

C^k — first-order logic with variables $x_1, \ldots, x_k$ and counting quantifiers of the form $\exists^{\geq i} x \,.\, \varphi$

1. Every formula of FPC is invariant under C^k-equivalence, for some k.
2. C^k-equivalence can be characterised by a k-pebble bijection game (a variant of Ehrenfeucht–Fraïsse)

Facts

- For each k, we can decide the winner of the k-pebble game in polynomial time.
- Close connection with a family of algorithms for graph isomorphism: Weisfeiler-Lehman method.

Non-definability result for FPC

There is a polynomial-time decidable property of finite graphs that is not definable in FPC.

Cai, Fürer and Immerman (1992)

Non-definability result for FPC

There is a polynomial-time decidable property of finite graphs that is not definable in FPC. “CFI property”

Cai, Fürer and Immerman (1992)

Non-definability result for FPC

There is a polynomial-time decidable property of finite graphs that is not definable in FPC.

"CFI property"

Cai, Fürer and Immerman (1992)

Corollary

FPC does not capture PTIME on

Non-definability result for FPC

There is a polynomial-time decidable property of finite graphs that is not definable in FPC. "CFI property"

Cai, Fürer and Immerman (1992)

Corollary

FPC does not capture PTIME on

- graphs of bounded degree

Non-definability result for FPC

There is a polynomial-time decidable property of finite graphs that is not definable in FPC. "CFI property"

Cai, Fürer and Immerman (1992)

Corollary

FPC does not capture PTIME on

- graphs of bounded degree (not even degree 3)

Non-definability result for FPC

There is a polynomial-time decidable property of finite graphs that is not definable in FPC. "CFI property"

Cai, Fürer and Immerman (1992)

Corollary

FPC does not capture PTIME on

- graphs of bounded degree (not even degree 3)
- graphs of bounded colour-class size

Non-definability result for FPC

There is a polynomial-time decidable property of finite graphs that is not definable in FPC. "CFI property"

Cai, Fürer and Immerman (1992)

Corollary

FPC does not capture PTIME on

- graphs of bounded degree (not even degree 3)
- graphs of bounded colour-class size (not even size 4)

Non-definability result for FPC

There is a polynomial-time decidable property of finite graphs that is not definable in FPC. "CFI property"

Cai, Fürer and Immerman (1992)

Corollary

FPC does not capture PTIME on

- graphs of bounded degree (not even degree 3)
- graphs of bounded colour-class size (not even size 4)

Still, the CFI query is hardly a *natural* graph property...

Non-definability result for FPC

There is a polynomial-time decidable property of finite graphs that is not definable in FPC.

"CFI property"

Cai, Fürer and Immerman (1992)

Corollary

FPC does not capture PTIME on

- graphs of bounded degree (not even degree 3)
- graphs of bounded colour-class size (not even size 4)

Still, the CFI query is hardly a *natural* graph property...

More recently: See which problems in linear algebra can be expressed in FPC

Descriptive complexity of problems in linear algebra

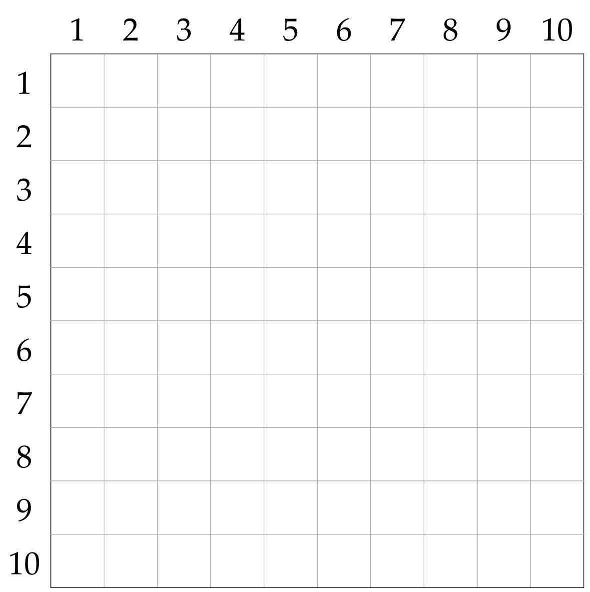
The usual notion of a matrix

$A = (a_{ij})$ — an m-by-n rectangular array of elements

	1	2	3	4	5	6	7	8	9	10
1										
2										
3										
4										
5										
6										
7										
8										
9										
10										

The usual notion of a matrix

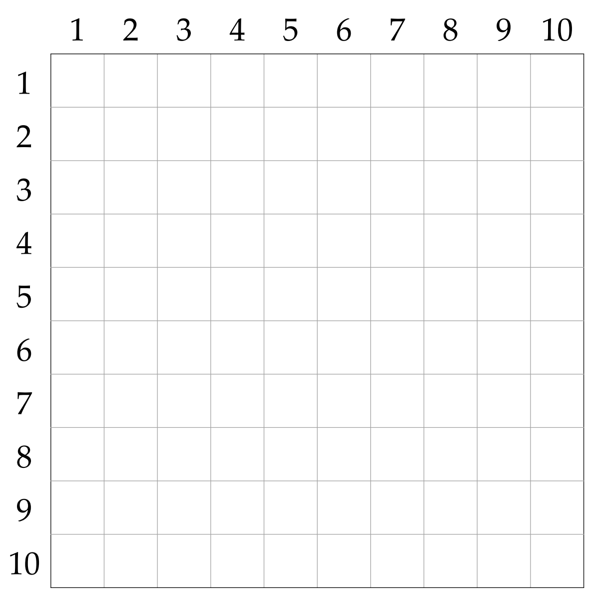
$A = (a_{ij})$ — an m-by-n rectangular array of elements



Recall: Over ordered structures FP (and hence FPC) can define *all* polynomial-time properties.

The usual notion of a matrix

$A = (a_{ij})$ — an m-by-n rectangular array of elements



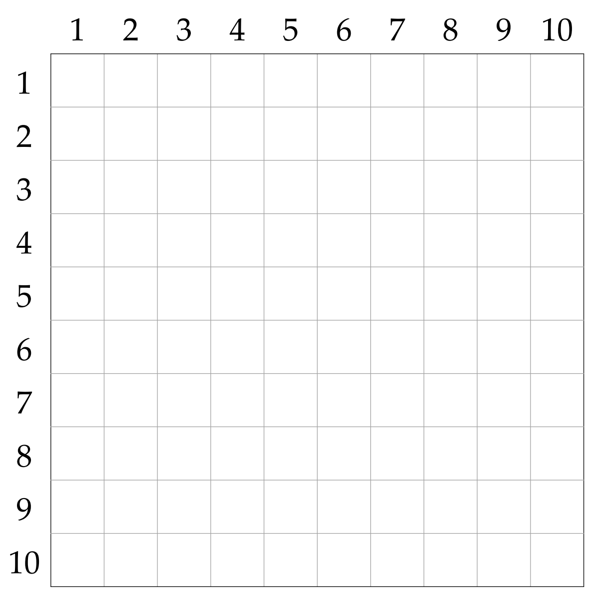
Recall: Over ordered structures FP (and hence FPC) can define *all* polynomial-time properties.

rows and columns ordered

all PTIME matrix properties can be defined in FP

The usual notion of a matrix

$A = (a_{ij})$ — an m-by-n rectangular array of elements



Recall: Over ordered structures FP (and hence FPC) can define *all* polynomial-time properties.

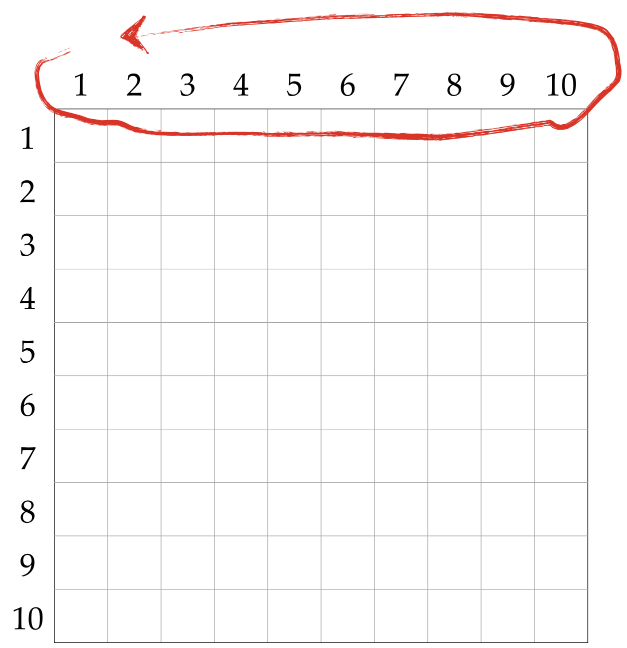
rows and columns ordered

all PTIME matrix properties can be defined in FP

Many natural matrix properties invariant under permutation of rows and columns

The usual notion of a matrix

$A = (a_{ij})$ — an m-by-n rectangular array of elements



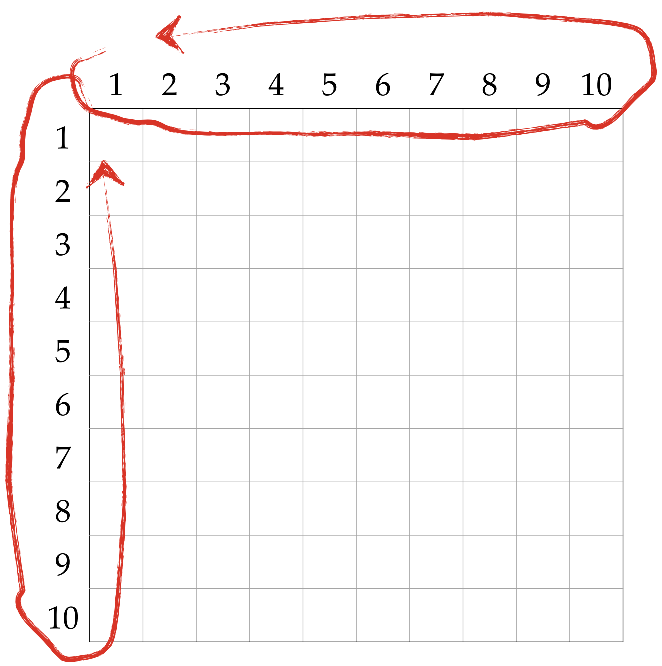
Recall: Over ordered structures FP (and hence FPC) can define *all* polynomial-time properties.

rows and columns ordered $\rightsquigarrow$ all PTIME matrix properties can be defined in FP

Many natural matrix properties invariant under permutation of rows and columns

The usual notion of a matrix

$A = (a_{ij})$ — an m-by-n rectangular array of elements



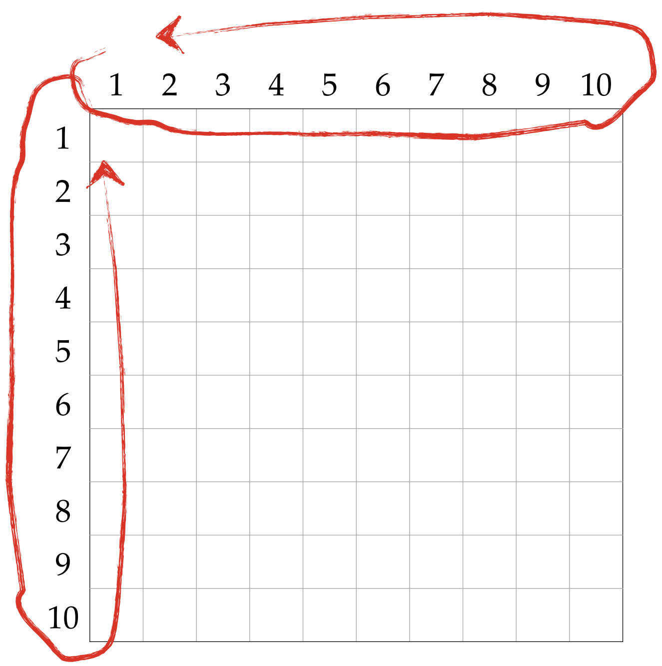
Recall: Over ordered structures FP (and hence FPC) can define *all* polynomial-time properties.

rows and columns ordered → all PTIME matrix properties can be defined in FP

Many natural matrix properties invariant under permutation of rows and columns

The usual notion of a matrix

$A = (a_{ij})$ — an *m*-by-*n* rectangular array of elements



Recall: Over ordered structures FP (and hence FPC) can define *all* polynomial-time properties.

rows and columns ordered

all PTIME matrix properties can be defined in FP

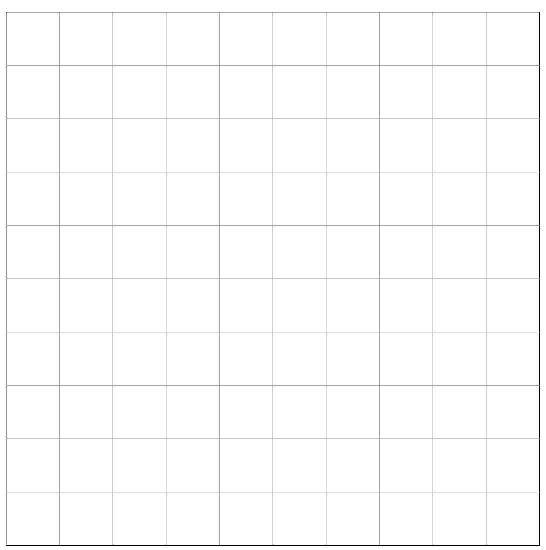
Many natural matrix properties invariant under permutation of rows and columns

(rank, determinant, etc.)

Unordered matrices

I, J — finite and non-empty sets

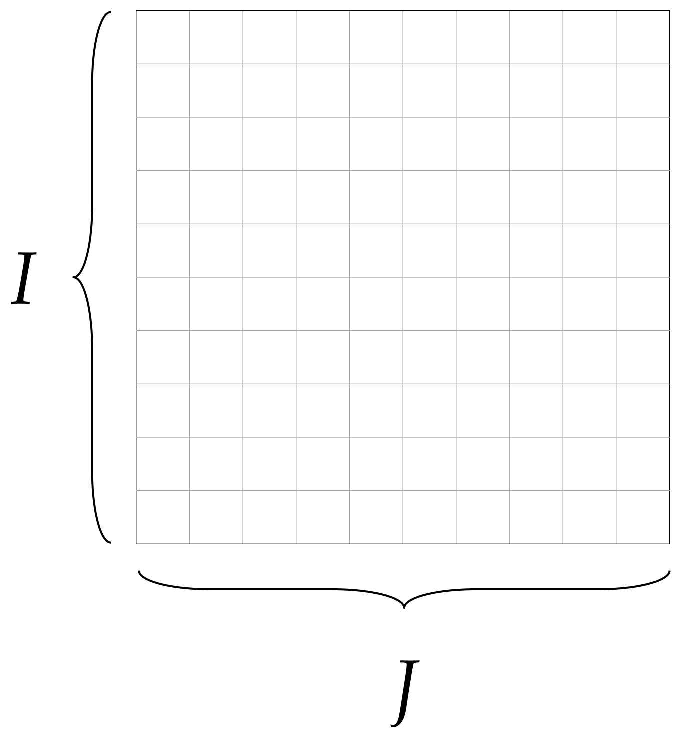
D — a group, a ring or a field



Unordered matrices

I, J — finite and non-empty sets

D — a group, a ring or a field

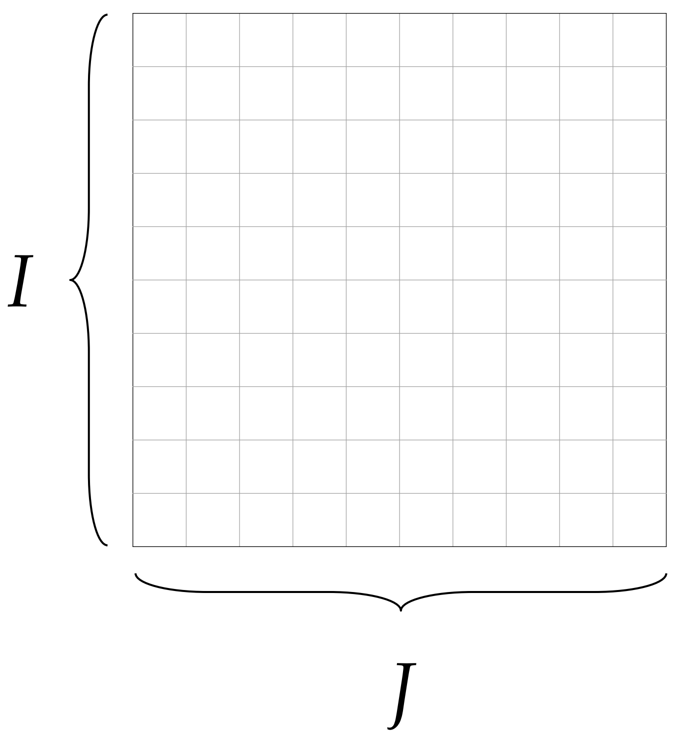


$$A : I \times J \to D$$

Unordered matrices

I, *J* — finite and non-empty sets

D — a group, a ring or a field



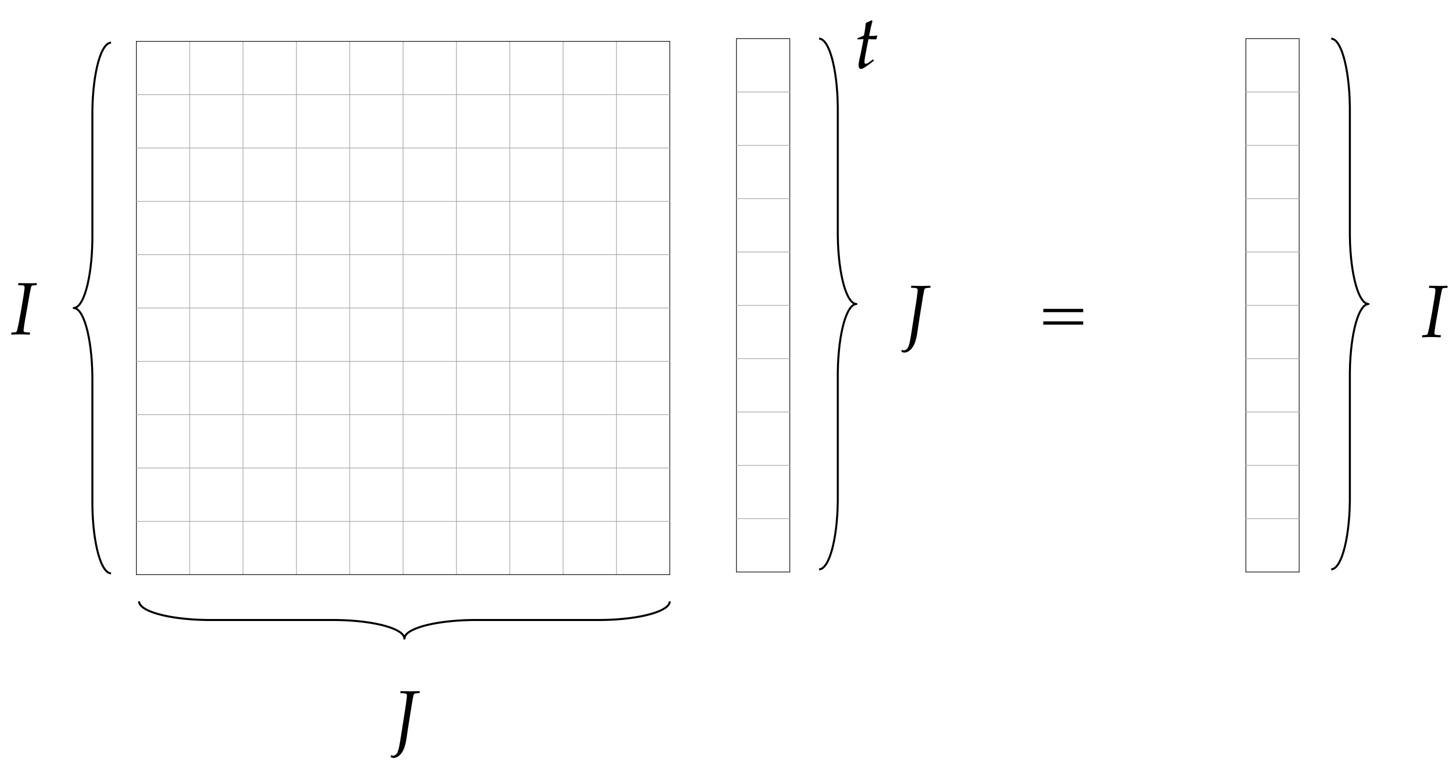
$$A : I \times J \to D$$

"an *I*-by-*J* matrix over *D*"

Unordered systems of linear equations

I, J — finite and non-empty sets

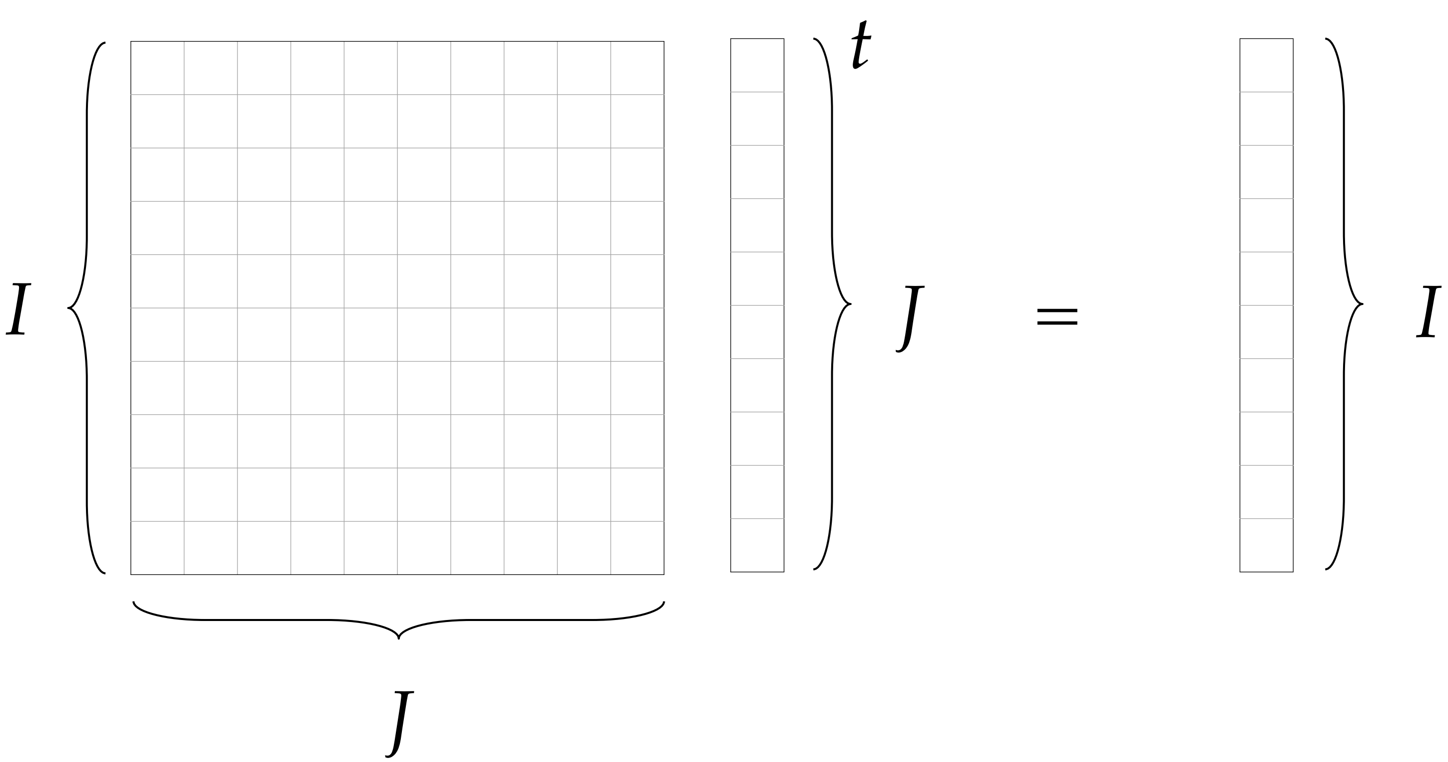
D — a group, a ring or a field



Unordered systems of linear equations

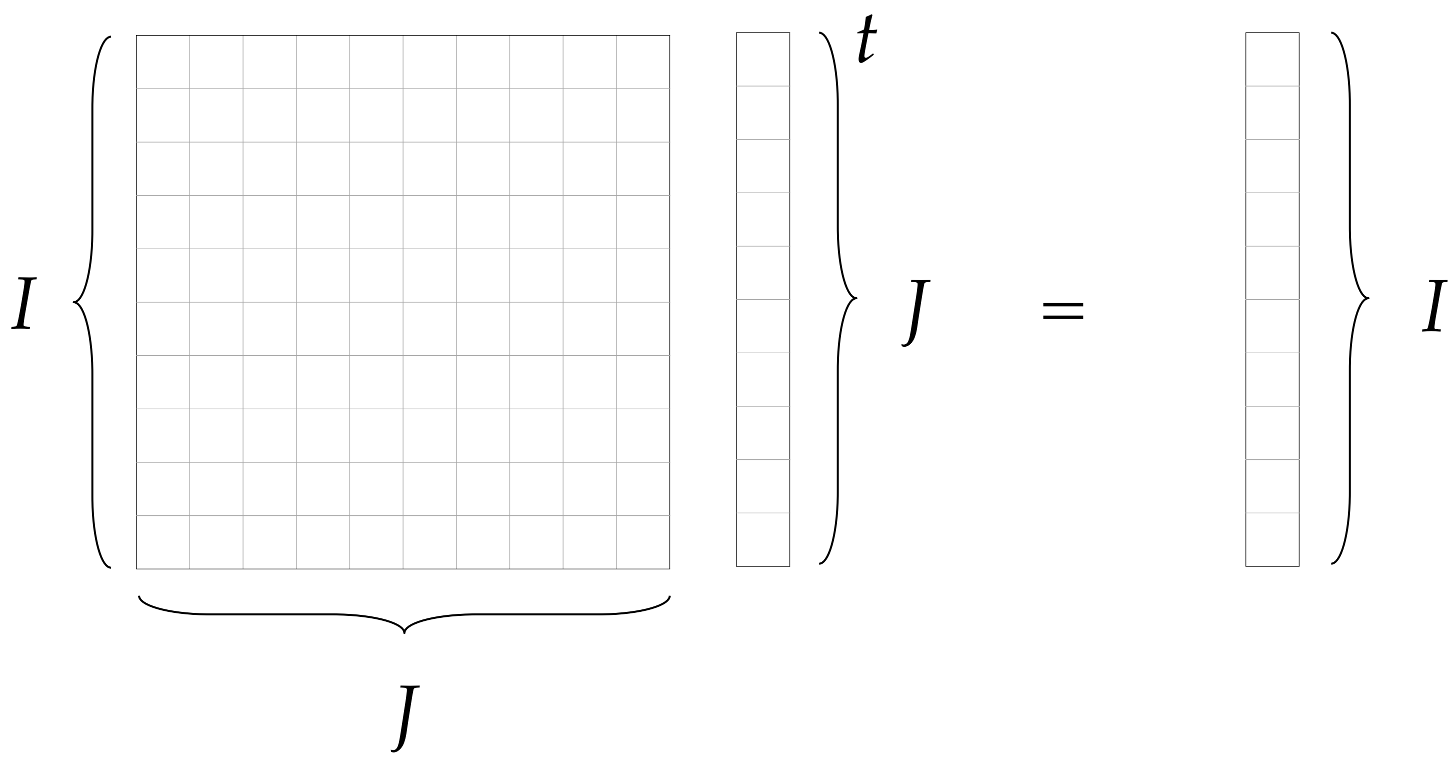
I, J — finite and non-empty sets

D — a group, a ring or a field



$$A\,\mathbf{x} = \mathbf{b}$$

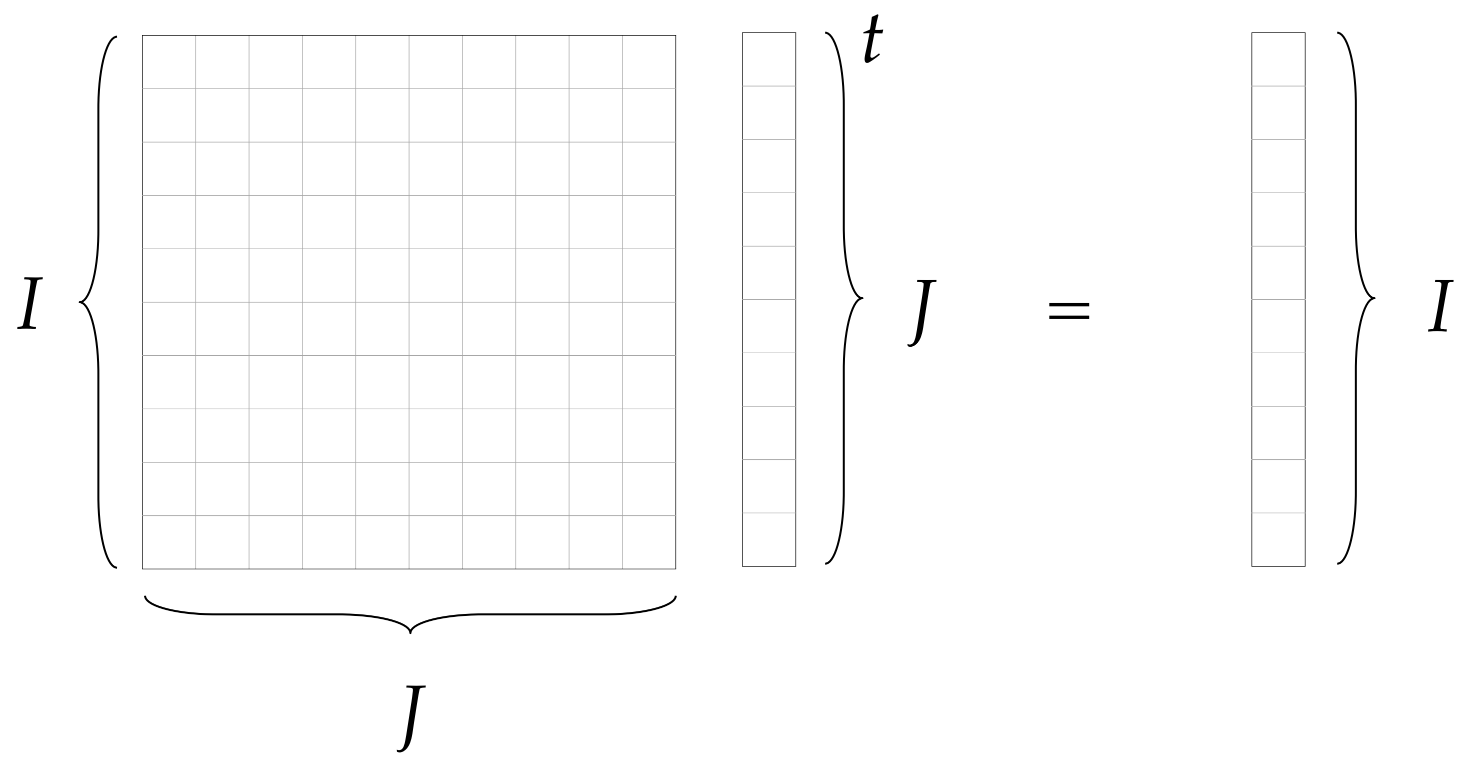
Unordered systems of linear equations



$$A\,\mathbf{x} = \mathbf{b}$$

Unordered systems of linear equations

As a relational structure over a fixed domain D:

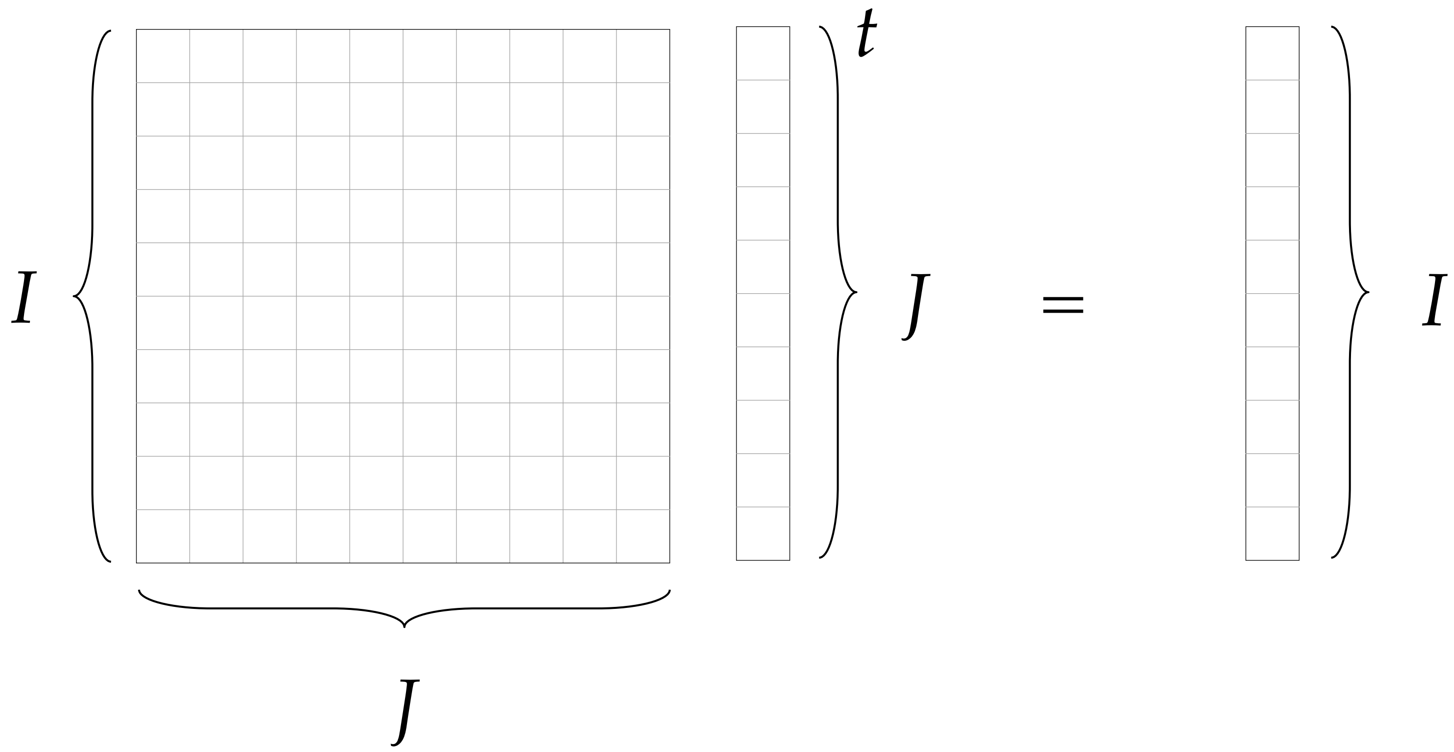


$$A\,\mathbf{x} = \mathbf{b}$$

Unordered systems of linear equations

As a relational structure over a fixed domain D:

$$\mathfrak{S} = (I, J; (A_d)_{d\in D}, (b_d)_{d\in D}) \quad \text{where} \quad A_d \subseteq I \times J \ \text{ and } \ b_d \subseteq I$$

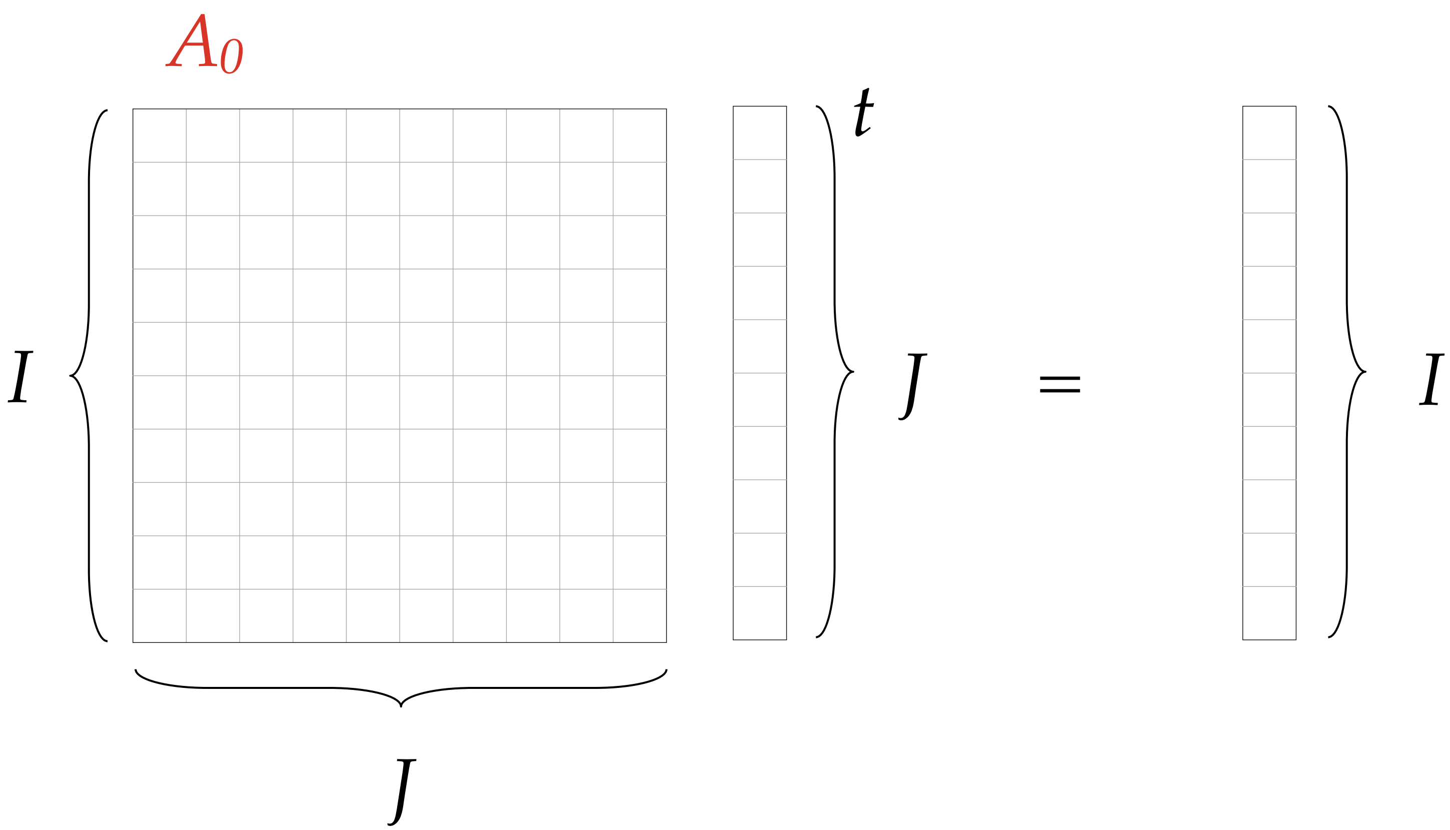


$$A\,\mathbf{x} \quad = \quad \mathbf{b}$$

Unordered systems of linear equations

As a relational structure over a fixed domain D:

$$\mathfrak{S} = (I, J; (A_d)_{d \in D}, (b_d)_{d \in D}) \quad \text{where} \quad A_d \subseteq I \times J \ \text{ and } \ b_d \subseteq I$$

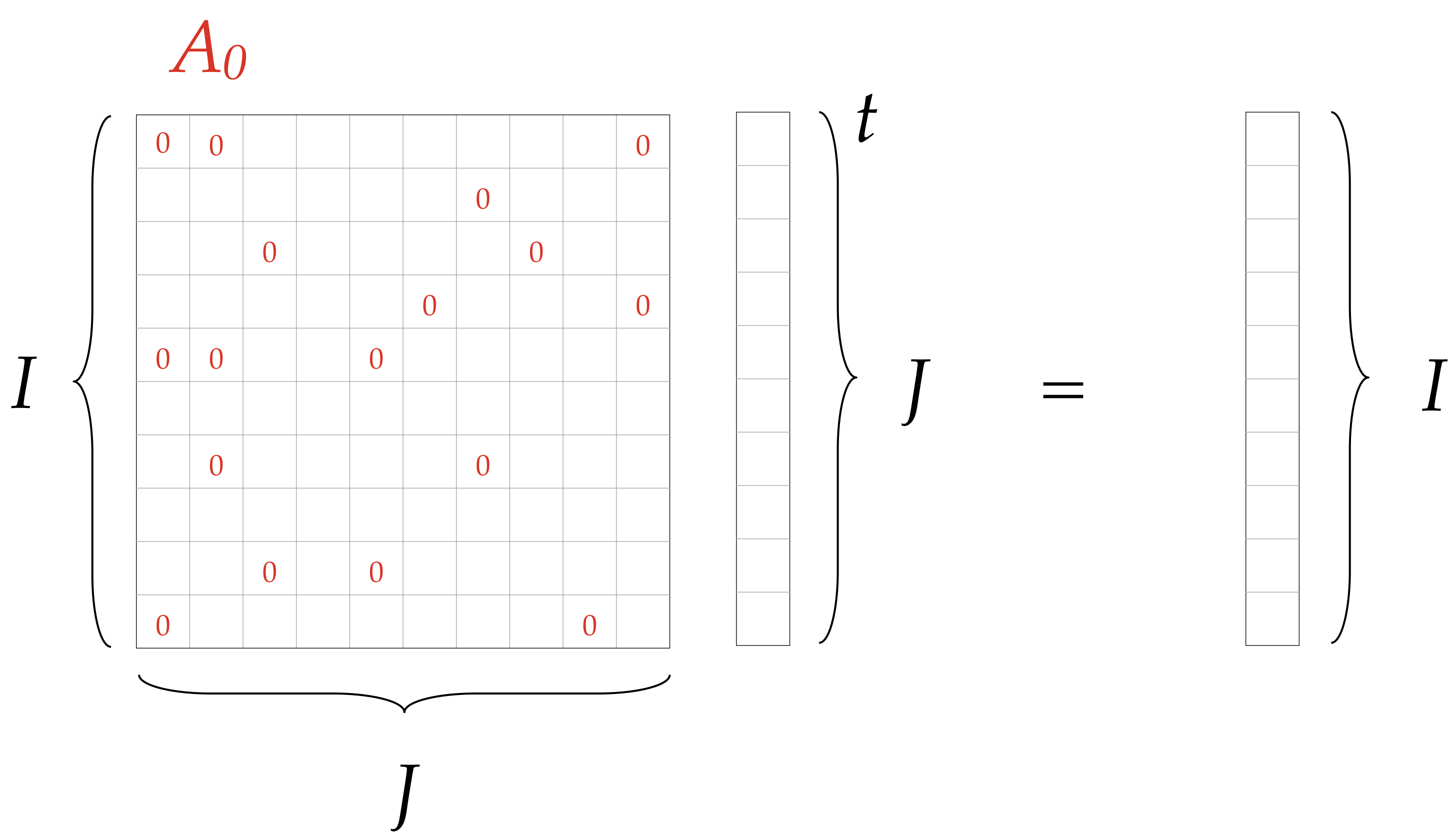


$$A\,\mathbf{x} \qquad = \qquad \mathbf{b}$$

Unordered systems of linear equations

As a relational structure over a fixed domain D:

$$\mathfrak{S} = (I, J; (A_d)_{d \in D}, (b_d)_{d \in D}) \quad \text{where} \quad A_d \subseteq I \times J \ \text{ and } \ b_d \subseteq I$$

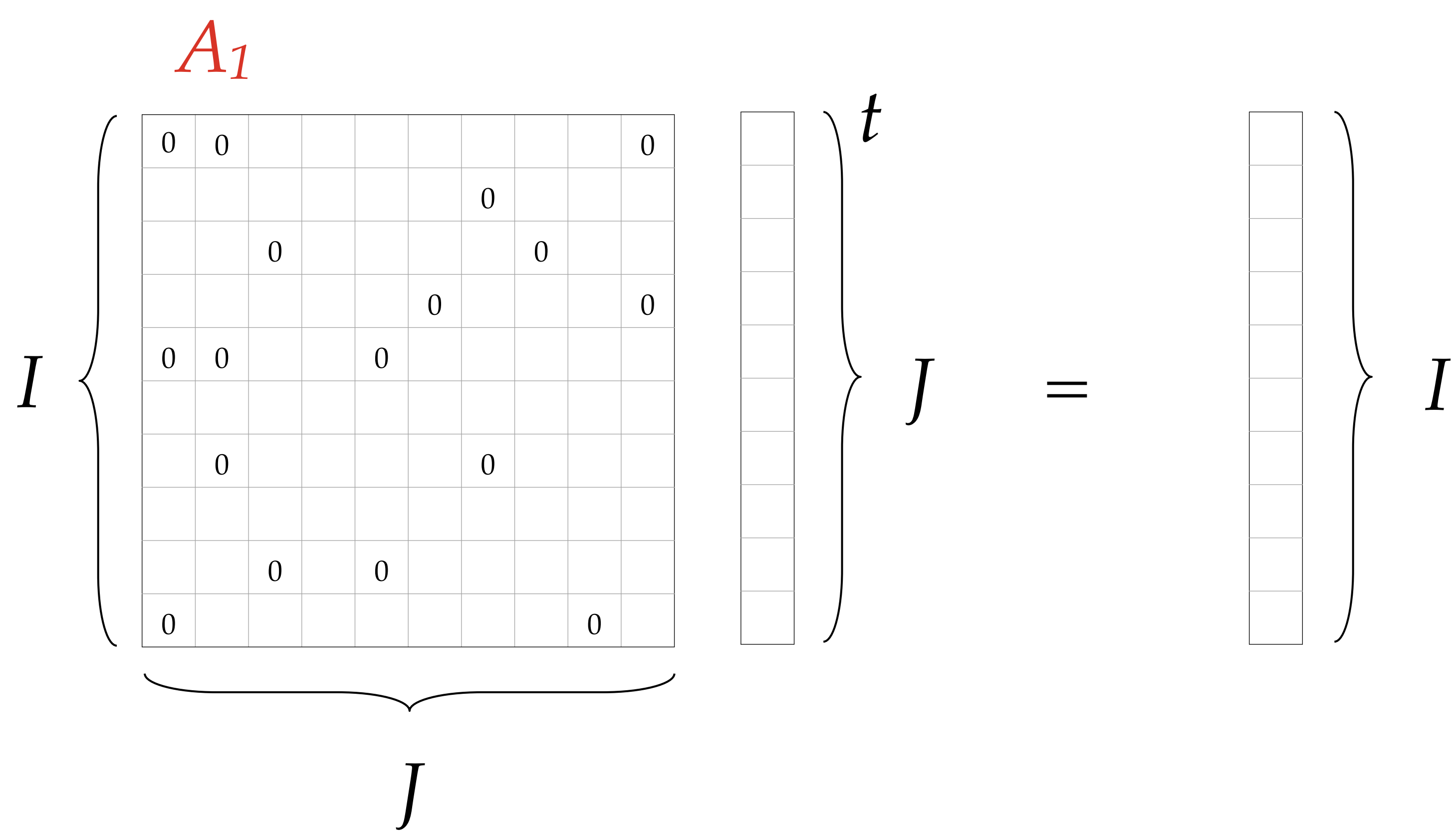


$$A\,\mathbf{x} = \mathbf{b}$$

Unordered systems of linear equations

As a relational structure over a fixed domain D:

$$\mathfrak{S} = (I, J; (A_d)_{d \in D}, (b_d)_{d \in D}) \quad \text{where} \quad A_d \subseteq I \times J \;\; \text{and} \;\; b_d \subseteq I$$



$$A\,\mathbf{x} \quad = \quad \mathbf{b}$$

Unordered systems of linear equations

As a relational structure over a fixed domain D:

$$\mathfrak{S} = (I, J; (A_d)_{d \in D}, (b_d)_{d \in D}) \quad \text{where} \quad A_d \subseteq I \times J \;\text{ and }\; b_d \subseteq I$$

$$A_1$$

$$I\left\{\underbrace{\begin{array}{|c|c|c|c|c|c|c|c|c|c|}\hline 0 & 0 & 1 & & 1 & & & & & 0 \\ \hline & & & 1 & & & 0 & & & \\ \hline & & 0 & & 1 & & & 0 & & \\ \hline & & & 1 & 1 & 0 & & & & 0 \\ \hline 0 & 0 & & & 0 & & & & & \\ \hline & & & 1 & & 1 & & & & \\ \hline & 0 & & & & 1 & 0 & & & \\ \hline & & & 1 & & & & 1 & & \\ \hline & & 0 & & 0 & & 1 & 1 & & \\ \hline 0 & & & & & & & & 0 & \\ \hline \end{array}}_{J}\right. \quad \left.\begin{array}{|c|}\hline \\ \hline \\ \hline \\ \hline \\ \hline \\ \hline \\ \hline \\ \hline \\ \hline \\ \hline \\ \hline \end{array}\right\}^{t} J \quad = \quad \left.\begin{array}{|c|}\hline \\ \hline \\ \hline \\ \hline \\ \hline \\ \hline \\ \hline \\ \hline \\ \hline \\ \hline \\ \hline \end{array}\right\} I$$

$$A\,\mathbf{x} \quad = \quad \mathbf{b}$$

Unordered systems of linear equations

As a relational structure over a fixed domain D:

$$\mathfrak{S} = (I, J; (A_d)_{d \in D}, (b_d)_{d \in D}) \quad \text{where} \quad A_d \subseteq I \times J \quad \text{and} \quad b_d \subseteq I$$

$$\begin{array}{c} \\ A_1 \\ I\left\{\underbrace{\begin{array}{|c|c|c|c|c|c|c|c|c|c|}\hline 0 & 0 & 1 & & 1 & & & & & 0 \\ \hline & & & 1 & & & 0 & & & \\ \hline & & 0 & & 1 & & & 0 & & \\ \hline & & & 1 & 1 & 0 & & & & 0 \\ \hline 0 & 0 & & & 0 & & & & & \\ \hline & & & 1 & & 1 & & & & \\ \hline & 0 & & & & 1 & 0 & & & \\ \hline & & & 1 & & & & 1 & & \\ \hline & & 0 & & 0 & & 1 & 1 & & \\ \hline 0 & & & & & & & & 0 & \\ \hline \end{array}}_{J}\right. \end{array} \quad \left.\begin{array}{|c|}\hline \\ \hline \\ \hline \\ \hline \\ \hline \\ \hline \\ \hline \\ \hline \\ \hline \\ \hline \\ \hline \end{array}\right\}^{t} J \quad = \quad \left.\begin{array}{|c|}\hline \\ \hline \\ \hline \\ \hline \\ \hline \\ \hline \\ \hline \\ \hline \\ \hline \\ \hline \\ \hline \end{array}\right\} I$$

$$A\,\mathbf{x} \qquad = \qquad \mathbf{b}$$

In this talk: Focus on $I = J$

FPC — more non-definability results

Solvability of systems of linear equations over any fixed finite Abelian group is not definable in FPC.

Atserias, Bulatov and Dawar (2007)

FPC — more non-definability results

Corollary

Solvability of systems of linear equations over any fixed finite field is not definable in FPC.

Atserias, Bulatov and Dawar (2007)

FPC — more non-definability results

Corollary

Solvability of systems of linear equations over any fixed finite field is not definable in FPC.

Atserias, Bulatov and Dawar (2007)

Recall: A linear system $A\mathbf{x} = \mathbf{b}$ over a field k is solvable if and only if the matrices A and $(A \mid \mathbf{b})$ have the same rank over k

FPC — more non-definability results

Corollary

Solvability of systems of linear equations over any fixed finite field is not definable in FPC.

Atserias, Bulatov and Dawar (2007)

Recall: A linear system $A\mathbf{x} = \mathbf{b}$ over a field k is solvable if and only if the matrices A and $(A \mid \mathbf{b})$ have the same rank over k

Corollary

Matrix rank over finite fields is not definable in FPC.

Which matrix properties *can* be defined in FPC?

Which matrix properties *can* be defined in FPC?

1. Characteristic polynomial and determinant of a square matrix over **Z**, **Q** and any finite field.

Dawar, H., Grohe, Laubner (2009)

Which matrix properties *can* be defined in FPC?

1. Characteristic polynomial and determinant of a square matrix over **Z**, **Q** and any finite field.
2. The inverse to any invertible square matrix over **Z**, **Q** and any finite field.

Dawar, H., Grohe, Laubner (2009)

Which matrix properties *can* be defined in FPC?

1. Characteristic polynomial and determinant of a square matrix over **Z**, **Q** and any finite field.
2. The inverse to any invertible square matrix over **Z**, **Q** and any finite field.
3. Rank of a matrix over **Q**.

Dawar, H., Grohe, Laubner (2009)

Which matrix properties *can* be defined in FPC?

1. Characteristic polynomial and determinant of a square matrix over **Z**, **Q** and any finite field.
2. The inverse to any invertible square matrix over **Z**, **Q** and any finite field.
3. Rank of a matrix over **Q**.

Dawar, H., Grohe, Laubner (2009)

4. Minimal polynomial of a square matrix over **Q** and any finite field.

H.-Pakusa (2010)

Which matrix properties *can* be defined in FPC?

1. Characteristic polynomial and determinant of a square matrix over **Z**, **Q** and any finite field.
2. The inverse to any invertible square matrix over **Z**, **Q** and any finite field.
3. Rank of a matrix over **Q**.

Dawar, H., Grohe, Laubner (2009)

4. Minimal polynomial of a square matrix over **Q** and any finite field.

H.-Pakusa (2010)

Fundamental linear-algebraic property over *fields* that separates FPC from PTIME: rank over finite fields

Which matrix properties *can* be defined in FPC?

1. Characteristic polynomial and determinant of a square matrix over **Z**, **Q** and any finite field.
2. The inverse to any invertible square matrix over **Z**, **Q** and any finite field.
3. Rank of a matrix over **Q**.

Dawar, H., Grohe, Laubner (2009)

4. Minimal polynomial of a square matrix over **Q** and any finite field.

H.-Pakusa (2010)

Fundamental linear-algebraic property over *fields* that separates FPC from PTIME: rank over finite fields

(Next talk: solvability problems over groups and rings)

Next step: extend fixed-point logic with ability to define matrix rank

Definable matrix relations

Recall: View any $A \subseteq I \times I$ as a matrix over GF(2).

Definable matrix relations

Recall: View any $A \subseteq I \times I$ as a matrix over GF(2).

formula $\varphi(x, y)$

graph $G = (V, E)$

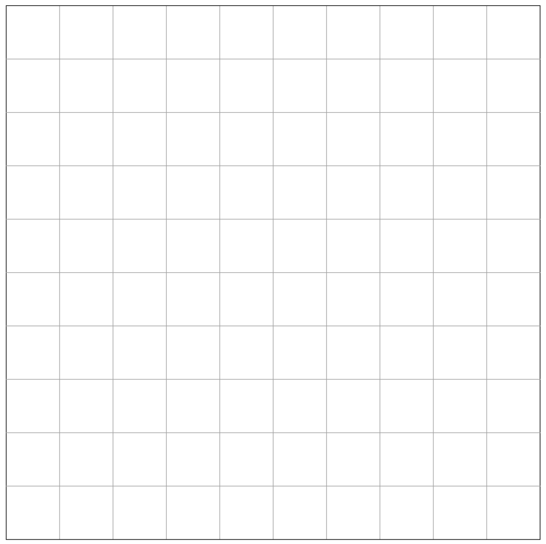
Definable matrix relations

Recall: View any $A \subseteq I \times I$ as a matrix over GF(2).

formula $\varphi(x, y)$

graph $G = (V, E)$ $\rightsquigarrow$ M_φ^G : (over GF(2))

V



V

Definable matrix relations

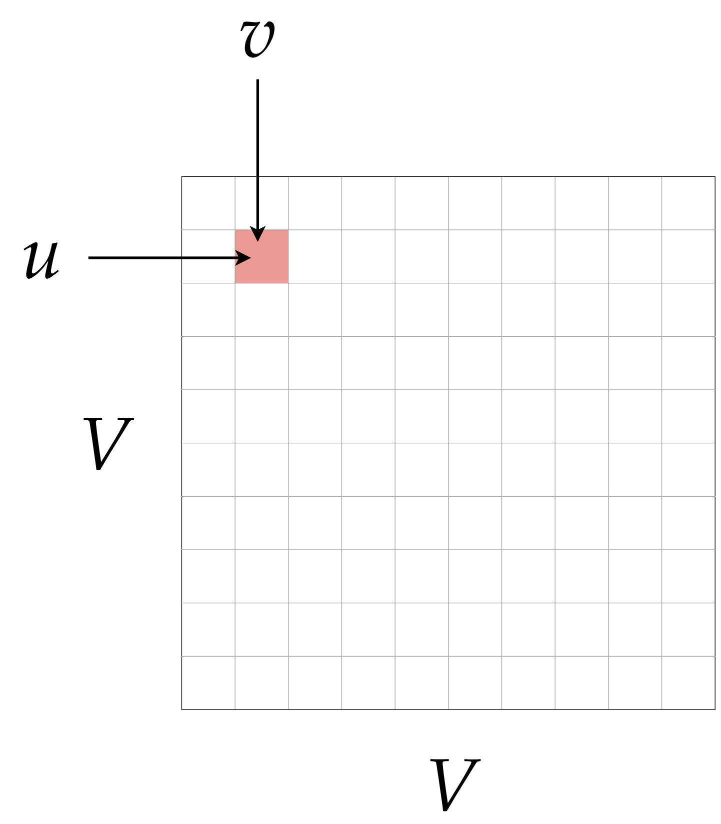
Recall: View any $A \subseteq I \times I$ as a matrix over GF(2).

formula $\varphi(x, y)$

graph $G = (V, E)$

$\rightsquigarrow$ M_{φ}^{G} :

(over GF(2))



Definable matrix relations

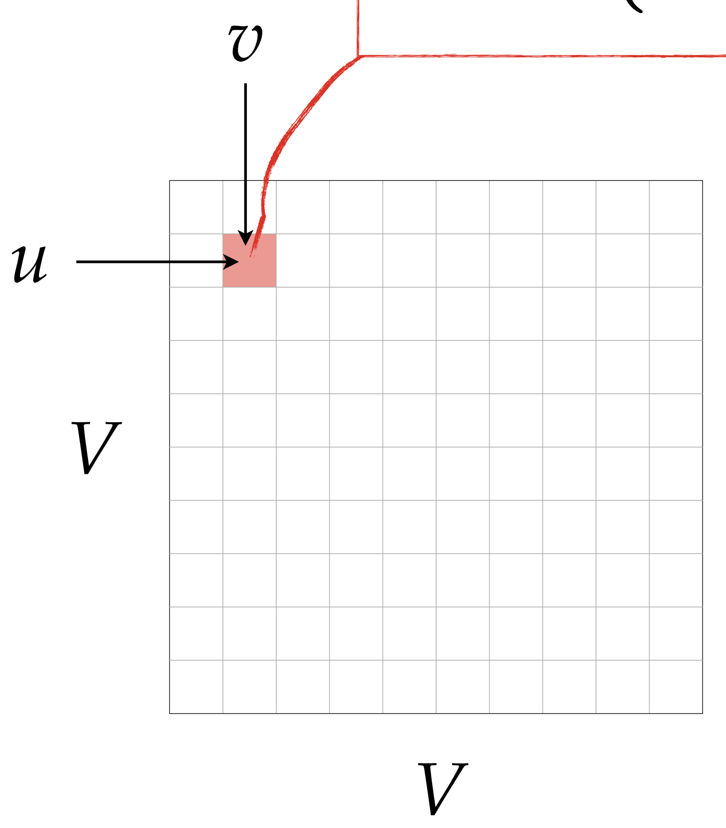
Recall: View any $A \subseteq I \times I$ as a matrix over GF(2).

$$(u,v) \mapsto \begin{cases} 1 & \text{if } G \models \varphi[u,v], \\ 0 & \text{otherwise.} \end{cases}$$

formula $\varphi(x,y)$

graph $G = (V, E)$ $\rightsquigarrow$ M_φ^G :

(over GF(2))



Definable matrix relations

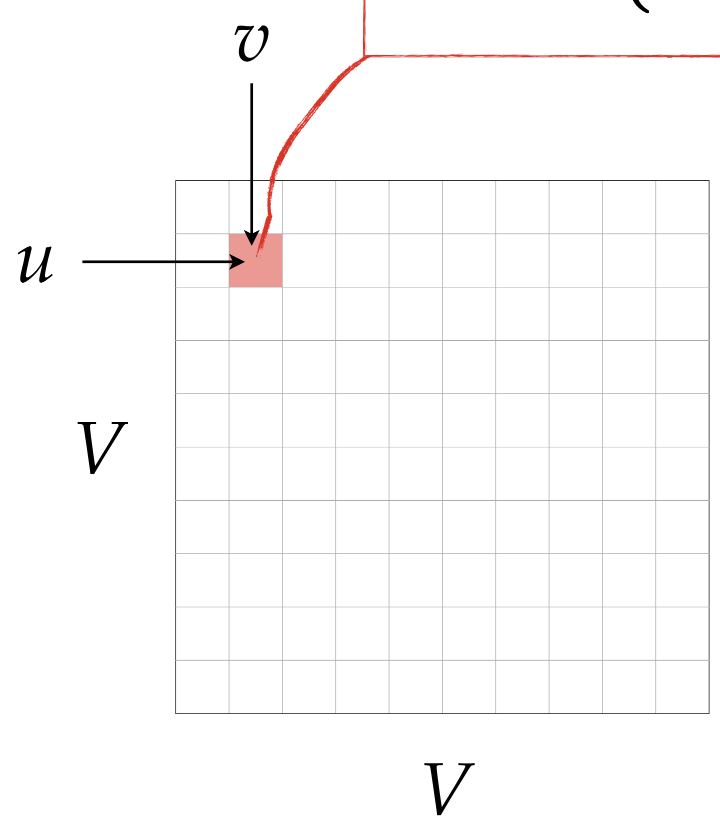
Recall: View any $A \subseteq I \times I$ as a matrix over GF(2).

$$(u, v) \mapsto \begin{cases} 1 & \text{if } G \models \varphi[u, v], \\ 0 & \text{otherwise.} \end{cases}$$

formula $\varphi(x, y)$

graph $G = (V, E)$ $\rightsquigarrow$ M_φ^G :

(over GF(2))



Example: $\varphi(x, y) := E(x, y) \rightsquigarrow M_\varphi^G$ = adjacency matrix of G

Definable matrix relations

Recall: View any $A \subseteq I \times I$ as a matrix over GF(2).

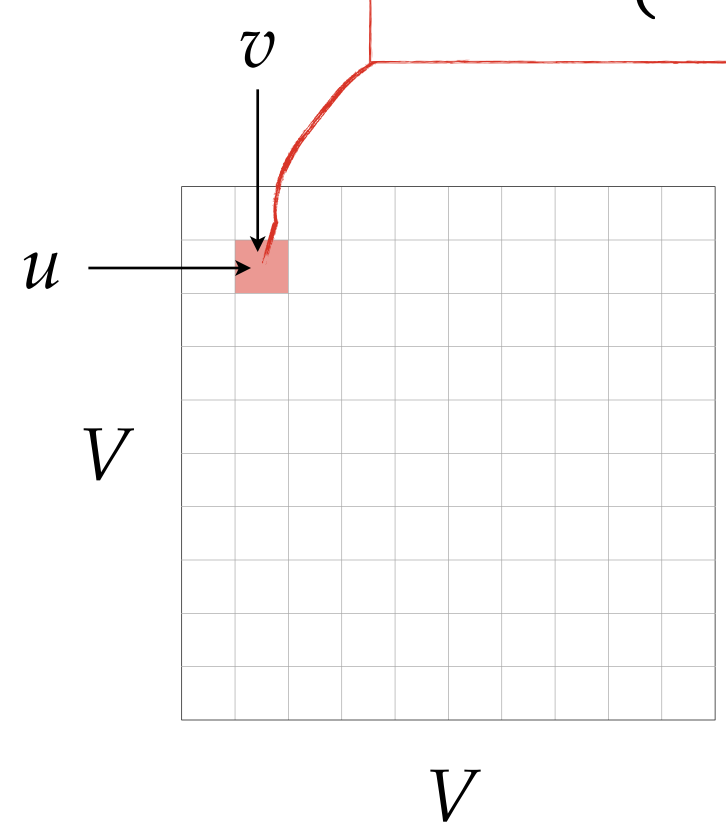
$$(u, v) \mapsto \begin{cases} 1 & \text{if } G \models \varphi[u, v], \\ 0 & \text{otherwise.} \end{cases}$$

formula $\varphi(x, y)$

graph $G = (V, E)$

$\rightsquigarrow$ M^G_φ :

(over GF(2))

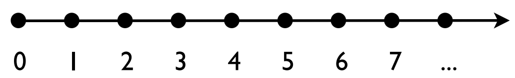


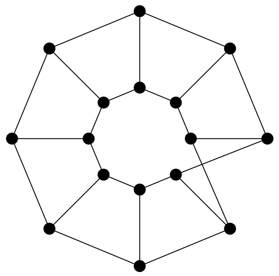
Example: $\varphi(x, y) := E(x, y) \rightsquigarrow M^G_\varphi$ = adjacency matrix of G

More generally: formalise matrices over GF(p), p prime

Fixed-point logic with rank operators

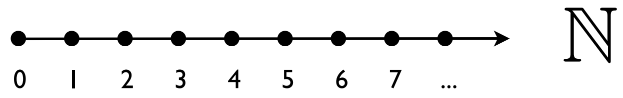
Variables are typed:

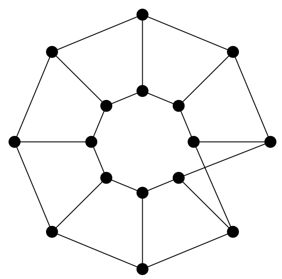




Fixed-point logic with rank operators

Variables are typed:



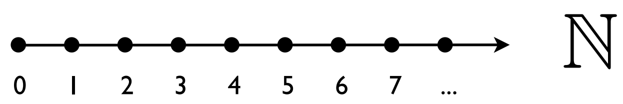


$G = (V, E)$

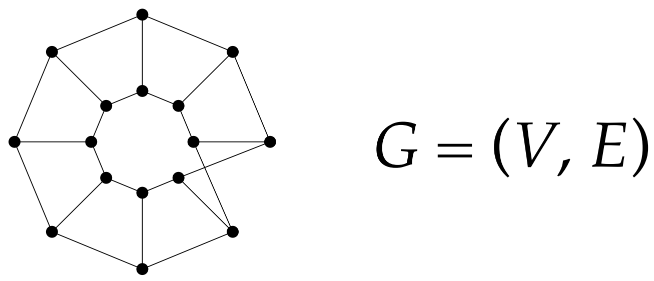
vertex variables: range over the vertices V

Fixed-point logic with rank operators

Variables are typed:



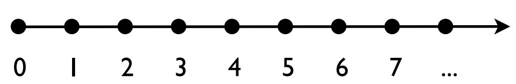
number variables: range over $\mathbb{N}$



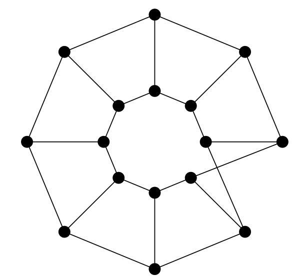
vertex variables: range over the vertices V

Fixed-point logic with rank operators

Variables are typed:



$\mathbb{N}$



$G = (V, E)$

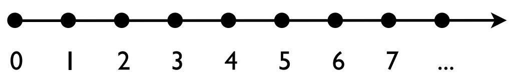
number variables: range over $\mathbb{N}$

vertex variables: range over the vertices V

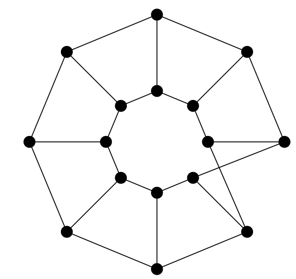
- Bounded quantification over number sort

Fixed-point logic with rank operators

Variables are typed:



$\mathbb{N}$



$G = (V, E)$

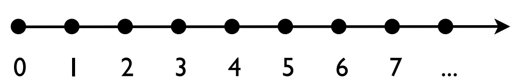
number variables: range over $\mathbb{N}$

vertex variables: range over the vertices V

- Bounded quantification over number sort
- Extend FP with rules for rank terms: $\mathbf{rk}_p(x, y).\varphi$ (p prime)

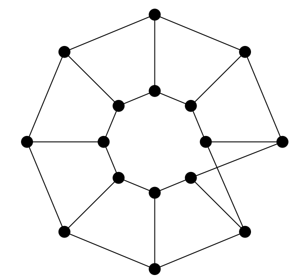
Fixed-point logic with rank operators

Variables are typed:



$\mathbb{N}$

number variables: range over $\mathbb{N}$



$G = (V, E)$

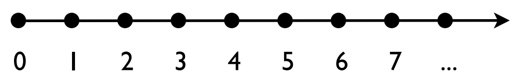
vertex variables: range over the vertices V

- Bounded quantification over number sort
- Extend FP with rules for rank terms: $\mathbf{rk}_p(x, y).\varphi$ (p prime)

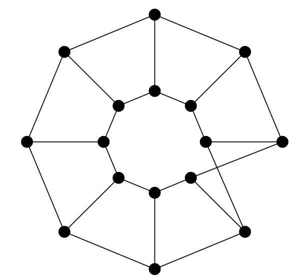
Semantics: $(\mathbf{rk}_p(x, y).\varphi)^G := \mathrm{rank}(M_\varphi^G)$ over GF(p)

Fixed-point logic with rank operators

Variables are typed:



$\mathbb{N}$



$G = (V, E)$

number variables: range over $\mathbb{N}$

vertex variables: range over the vertices V

- Bounded quantification over number sort
- Extend FP with rules for rank terms: $\mathbf{rk}_p(x, y).\varphi$ (p prime)

Semantics: $(\mathbf{rk}_p(x, y).\varphi)^G := \mathrm{rank}(M_\varphi^G)$ over GF(p)

$\longrightarrow$ Logics FPR$_p$, FPR and similarly FOR$_p$, FOR

Expressive power of rank logics

For any prime p, FPR_p can express solvability of linear equations over $\mathrm{GF}(p)$.

Dawar, Grohe, H., Laubner (2009)

Expressive power of rank logics

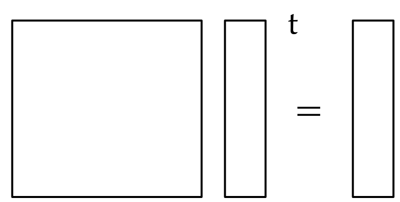
For any prime p, FPR_p can express solvability of linear equations over $\mathrm{GF}(p^m)$ for any m.

H. (2010)

Expressive power of rank logics

For any prime p, FPR_p can express solvability of linear equations over $\mathrm{GF}(p^m)$ for any m.

H. (2010)

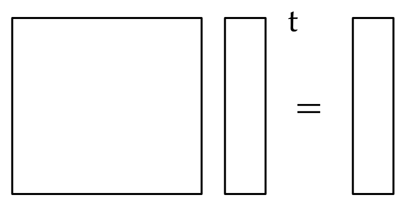


over $\mathrm{GF}(p^m)$

Expressive power of rank logics

For any prime p, FPR_p can express solvability of linear equations over $\mathrm{GF}(p^m)$ for any m.

H. (2010)



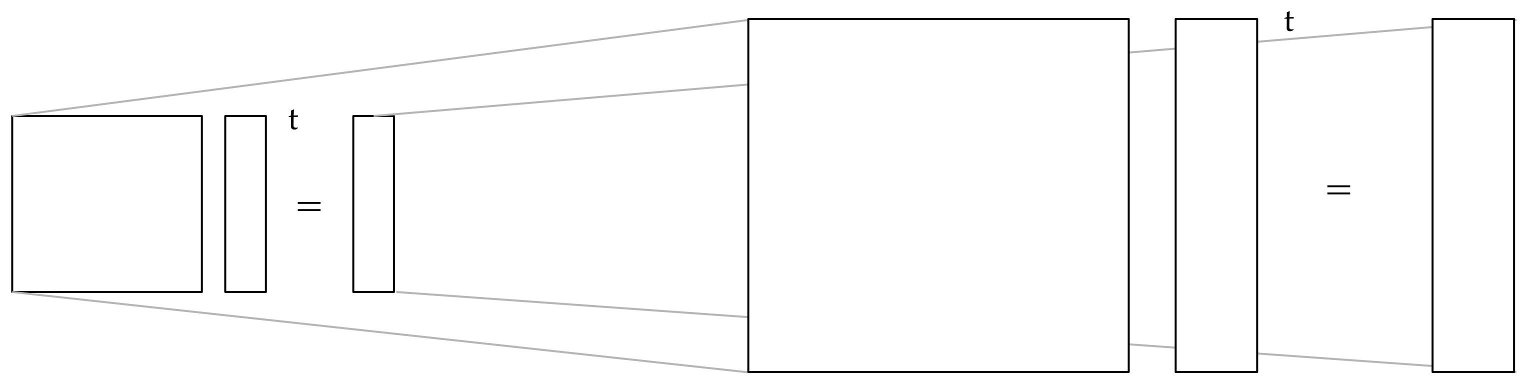
over $\mathrm{GF}(p^m)$

Represent each element of $\mathrm{GF}(p^m)$
as an m-by-m matrix over GF(p)

Expressive power of rank logics

For any prime p, FPR_p can express solvability of linear equations over $\mathrm{GF}(p^m)$ for any m.

H. (2010)



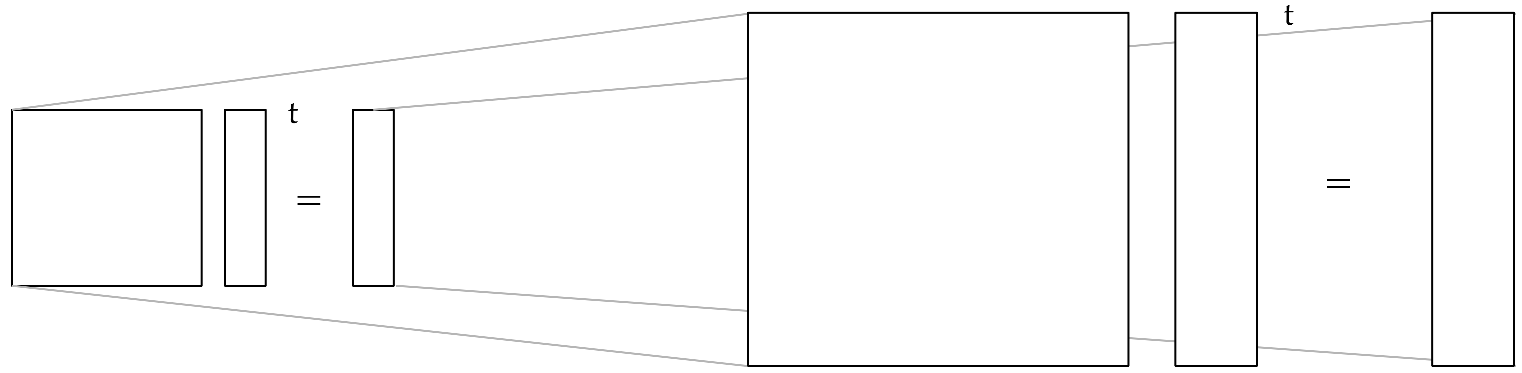
over $\mathrm{GF}(p^m)$ equivalent system over $\mathrm{GF}(p)$

Represent each element of $\mathrm{GF}(p^m)$
as an m-by-m matrix over GF(p)

Expressive power of rank logics

For any prime p, FPR_p can express solvability of linear equations over $\mathrm{GF}(p^m)$ for any m.

H. (2010)



over $\mathrm{GF}(p^m)$

equivalent system over $\mathrm{GF}(p)$

Represent each element of $\mathrm{GF}(p^m)$
as an m-by-m matrix over GF(p)

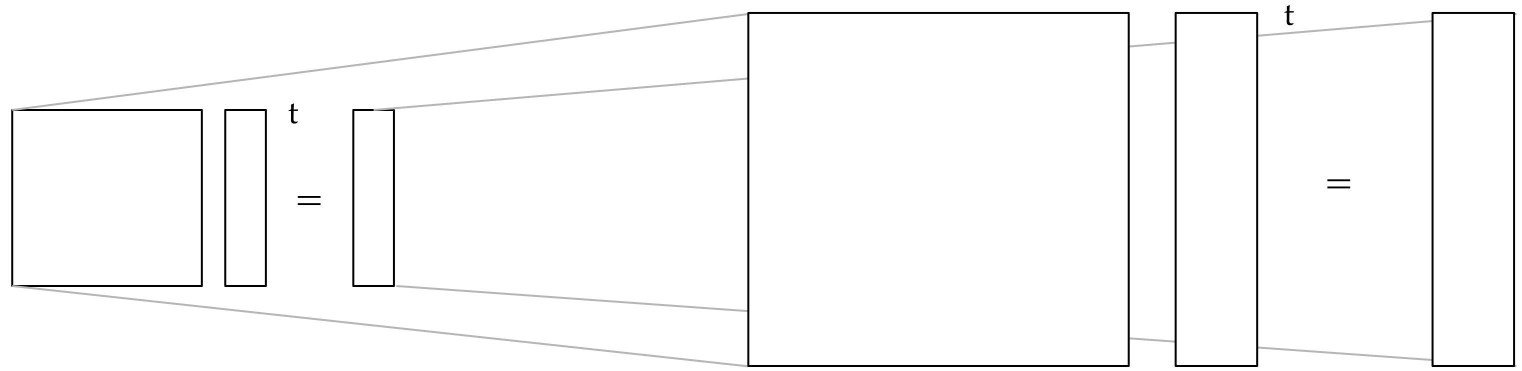
Corollary

For any prime p, $\mathrm{FPC} \subsetneq \mathrm{FPR}_p \subseteq \mathrm{PTIME}$.

Expressive power of rank logics

For any prime p, FPR_p can express solvability of linear equations over $\mathrm{GF}(p^m)$ for any m.

H. (2010)



over $\mathrm{GF}(p^m)$ equivalent system over $\mathrm{GF}(p)$

Represent each element of $\mathrm{GF}(p^m)$
as an m-by-m matrix over GF(p)

(we can simulate counting by
expressing rank of diagonal matrices)

Corollary

For any prime p, $\mathrm{FPC} \subsetneq \mathrm{FPR}_p \subseteq \mathrm{PTIME}$.

CFI graphs revisited

Non-isomorphic CFI graphs can be distinguished by a sentence of FOR_2.

Dawar, Grohe, H., Laubner (2009)

CFI graphs revisited

Non-isomorphic CFI graphs can be distinguished by a sentence of FOR_2.

Dawar, Grohe, H., Laubner (2009)

Recall: FPC does not capture PTIME on graphs of bounded colour-class size $\rightsquigarrow$ not even size 4

CFI graphs revisited

Non-isomorphic CFI graphs can be distinguished by a sentence of FOR_2.

Dawar, Grohe, H., Laubner (2009)

Recall: FPC does not capture PTIME on graphs of bounded colour-class size $\rightsquigarrow$ not even size 4

Isomorphism of graphs of colour class size 4 can be expressed in FOR_2.

Dawar, H. (2011)

Pebble games for rank logics & the Weisfeiler-Lehman method

Proving non-definability in FPR_p

<u>Recall</u>: Proofs of inexpressibility in FPC are generally formulated using a game method.

Proving non-definability in FPR_p

Recall: Proofs of inexpressibility in FPC are generally formulated using a game method.

Our wish list:

Proving non-definability in FPR_p

Recall: Proofs of inexpressibility in FPC are generally formulated using a game method.

Our wish list:

A pebble game for finite-variable rank logics for which...

Proving non-definability in FPR_p

Recall: Proofs of inexpressibility in FPC are generally formulated using a game method.

Our wish list:

A pebble game for finite-variable rank logics for which...

1. we can decide who wins the game in polynomial time, and

Proving non-definability in FPR_p

Recall: Proofs of inexpressibility in FPC are generally formulated using a game method.

Our wish list:

A pebble game for finite-variable rank logics for which...

1. we can decide who wins the game in polynomial time, and
2. there is a corresponding "stable colouring algorithm", like for the counting game on graphs.

Proving non-definability in FPR_p

Recall: Proofs of inexpressibility in FPC are generally formulated using a game method.

Our wish list:

A pebble game for finite-variable rank logics for which...

matrix-rank game

1. we can decide who wins the game in polynomial time, and
2. there is a corresponding "stable colouring algorithm", like for the counting game on graphs.

Proving non-definability in FPR_p

Recall: Proofs of inexpressibility in FPC are generally formulated using a game method.

Our wish list:

A pebble game for finite-variable rank logics for which...

matrix-rank game

1. we can decide who wins the game in polynomial time, and
2. there is a corresponding "stable colouring algorithm", like for the counting game on graphs.

invertible-map game

Proving non-definability in FPR$_p$

R_p^k — first-order logic with variables x_1, ..., x_k and rank quantifiers of the form $\mathrm{rk}_p^{\geq i}(x, y) \, . \, (\varphi)$

Proving non-definability in FPR_p

R_p^k — first-order logic with variables x_1, ..., x_k and rank quantifiers of the form $\mathrm{rk}_p^{\geq i}(x, y) \,.\, (\varphi)$

1. Every formula of FPR_p is invariant under R_p^k -equivalence, for some k.

Proving non-definability in FPR_p

R_p^k — first-order logic with variables x_1, ..., x_k and rank quantifiers of the form $\mathrm{rk}_p^{\geq i}(x, y) \,.\, (\varphi)$

1. Every formula of FPR_p is invariant under R_p^k -equivalence, for some k.
2. R_p^k-equivalence can be characterised by a k-pebble matrix-rank game (over $\mathrm{GF}(p)$)

Proving non-definability in FPR_p

R_p^k — first-order logic with variables x_1, ..., x_k and rank quantifiers of the form $\mathrm{rk}_p^{\geq i}(x, y)\,.\,(\varphi)$

1. Every formula of FPR_p is invariant under R_p^k -equivalence, for some *k*.
2. R_p^k-equivalence can be characterised by a *k*-pebble matrix-rank game (over GF(*p*))

G and *H* agree on all sentences of *k*-variable rank logic over GF(*p*)	iff	Duplicator has a winning strategy in the *k*-pebble matrix-rank game on *G* and *H*

Matrix-rank game over GF(p)

Matrix-rank game over GF(p)

Game played on finite graphs G and H

Matrix-rank game over GF(p)

Game played on finite graphs G and H

- Protocol based on partitioning each game board into disjoint $\{0,1\}$-matrices ("partition matrices").

Matrix-rank game over GF(p)

Game played on finite graphs G and H

- Protocol based on partitioning each game board into disjoint $\{0,1\}$-matrices ("partition matrices").
- Algebraic game rules: At each round, Duplicator has to ensure that every linear combination of partition matrices over G has the same GF(p)-rank as the corresponding linear combination over H.

Matrix-rank game over GF(p)

Problem: Not known if we can decide in polynomial time which player wins the game.

Game played on finite graphs G and H

- Protocol based on partitioning each game board into disjoint {0,1}-matrices ("partition matrices").
- Algebraic game rules: At each round, Duplicator has to ensure that every linear combination of partition matrices over G has the same GF(p)-rank as the corresponding linear combination over H.

Matrix-rank game over GF(p)

Problem: Not known if we can decide in polynomial time which player wins the game.

Game played on finite graphs G and H

- Protocol based on partitioning each game board into disjoint {0,1}-matrices ("partition matrices").
- Algebraic game rules: At each round, Duplicator has to ensure that every linear combination of partition matrices over G has the same GF(p)-rank as the corresponding linear combination over H.

Strengthening the game rules

Two tuples (A_1, A_2, ..., A_m) and (B_1, B_2, ..., B_m) of n-by-n matrices over a field k are simultaneously similar if there is an invertible S such that $S A_i S^{-1} = B_i$ for all i.

Strengthening the game rules

Two tuples $(A_1, A_2, \ldots, A_m)$ and $(B_1, B_2, \ldots, B_m)$ of n-by-n matrices over a field k are simultaneously similar if there is an invertible S such that $S A_i S^{-1} = B_i$ for all i.

There is a deterministic algorithm that, given two m-tuples $\mathbf{A}$ and $\mathbf{B}$ of n-by-n matrices over a finite field GF(q), determines in time poly(n, m, q) whether $\mathbf{A}$ and $\mathbf{B}$ are simultaneously similar.

Chistov, Karpinsky and Ivanyov (1997)

Game based on invertible linear maps

Invertible-map game on G and H over GF(p):

- Protocol based on partitioning each game board into disjoint {0,1}-matrices ("partition matrices").

Game based on invertible linear maps

Invertible-map game on G and H over GF(p):

- Protocol based on partitioning each game board into disjoint {0,1}-matrices ("partition matrices").
- New game rule: At each round, Duplicator has to ensure that the two tuples of partition matrices (over G and H) are simultaneously similar over GF(p).

Game based on invertible linear maps

Invertible-map game on G and H over GF(p):

- Protocol based on partitioning each game board into disjoint {0,1}-matrices ("partition matrices").
- New game rule: At each round, Duplicator has to ensure that the two tuples of partition matrices (over G and H) are simultaneously similar over GF(p).

Facts:

Game based on invertible linear maps

Invertible-map game on G and H over GF(p):

- Protocol based on partitioning each game board into disjoint {0,1}-matrices ("partition matrices").
- New game rule: At each round, Duplicator has to ensure that the two tuples of partition matrices (over G and H) are simultaneously similar over GF(p).

Facts:

- We can decide who wins this game in PTIME.

Game based on invertible linear maps

Invertible-map game on G and H over GF(p):

- Protocol based on partitioning each game board into disjoint {0,1}-matrices ("partition matrices").
- New game rule: At each round, Duplicator has to ensure that the two tuples of partition matrices (over G and H) are simultaneously similar over GF(p).

Facts:

- We can decide who wins this game in PTIME.
- Refines R_p^k-equivalence: If Duplicator wins the k-pebble invertible-map game on G and H then she also wins the k-pebble matrix rank game on G and H.

Connection with stable colouring

Recall:

Our wish list:

A pebble game for finite-variable rank logics for which...

1. we can decide who wins the game in polynomial time, and
2. there is a corresponding "stable colouring algorithm", like for the counting game on graphs.

matrix-rank game

invertible-map game

Weisfeiler-Lehman refinement

Input: Graph $G = (V, E)$

Output: Equivalence relation $\approx$ on V.

Weisfeiler-Lehman refinement

"colour refinement"
or "stable colouring"

Input:	Graph $G = (V, E)$
Output:	Equivalence relation $\approx$ on V.

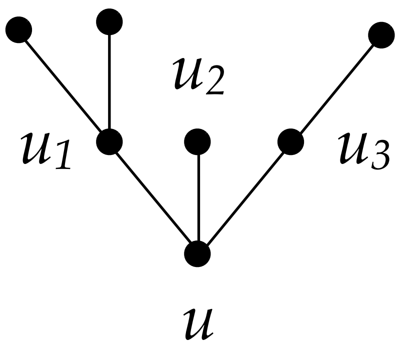
Weisfeiler-Lehman refinement

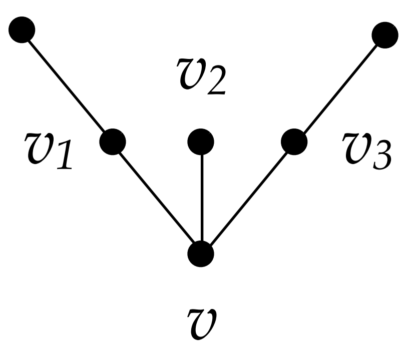
"colour refinement"
or "stable colouring"

Input: Graph $G = (V, E)$

Output: Equivalence relation $\approx$ on V.

Inductively define: $\sim_0 \supseteq \sim_1 \supseteq \ \dots \ \supseteq \sim_m = \sim_{m+1} =: \approx$





Weisfeiler-Lehman refinement

"colour refinement"
or "stable colouring"

Input: Graph $G = (V, E)$

Output: Equivalence relation $\approx$ on V.

Inductively define: $\sim_0 \supseteq \sim_1 \supseteq \ \dots \ \supseteq \sim_m = \sim_{m+1} =: \approx$

Initial: $u \sim_0 v$ iff $\deg(u) = \deg(v)$

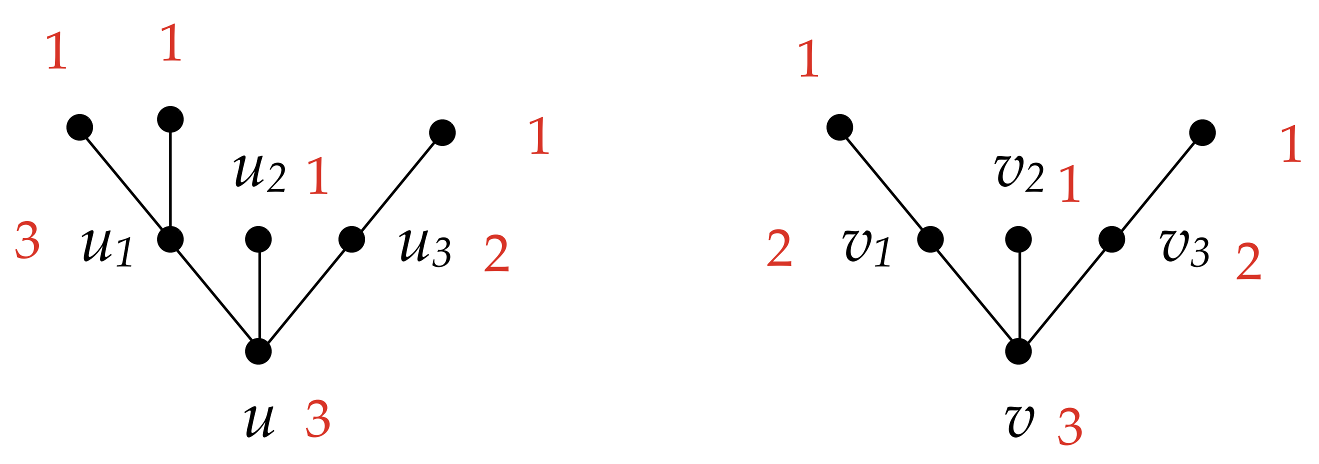
Weisfeiler-Lehman refinement

"colour refinement"
or "stable colouring"

Input: Graph $G = (V, E)$
Output: Equivalence relation $\approx$ on V.

Inductively define: $\sim_0 \supseteq \sim_1 \supseteq \ \dots \ \supseteq \sim_m = \sim_{m+1} =: \approx$

Initial: $u \sim_0 v \quad$ iff $\quad \deg(u) = \deg(v)$



Weisfeiler-Lehman refinement

"colour refinement"
or "stable colouring"

Input: Graph $G = (V, E)$

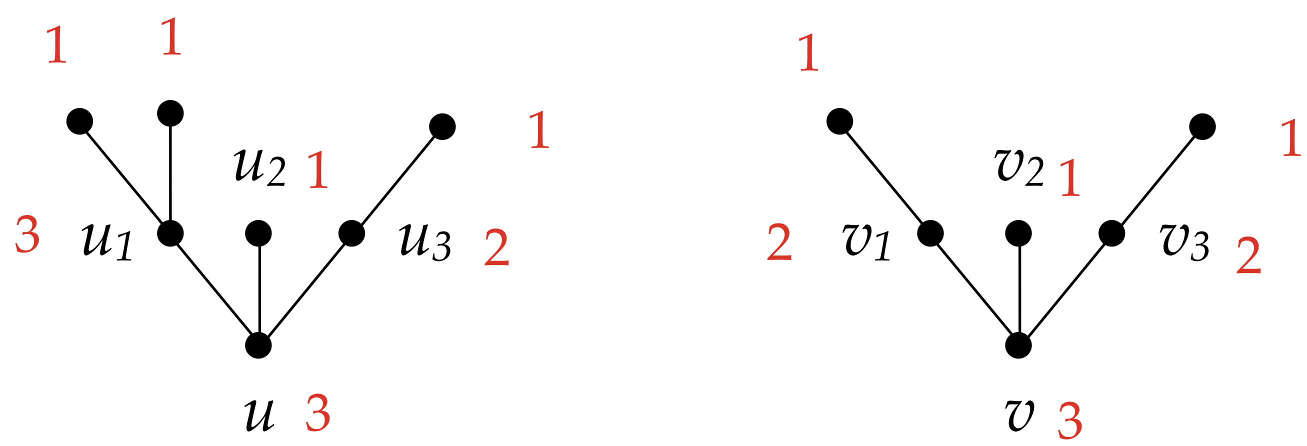
Output: Equivalence relation $\approx$ on V.

Inductively define: $\sim_0 \supseteq \sim_1 \supseteq \ \dots \ \supseteq \sim_m = \sim_{m+1} =: \approx$

Initial: $u \sim_0 v$ iff $\deg(u) = \deg(v)$

Refine: $u \sim_{i+1} v$ iff $u \sim_i v$ and for all $\alpha \in V/\sim_i$:

$$\|N(u) \cap \alpha\| = \|N(v) \cap \alpha\|$$



Weisfeiler-Lehman refinement

"colour refinement"
or "stable colouring"

Input: Graph $G = (V, E)$

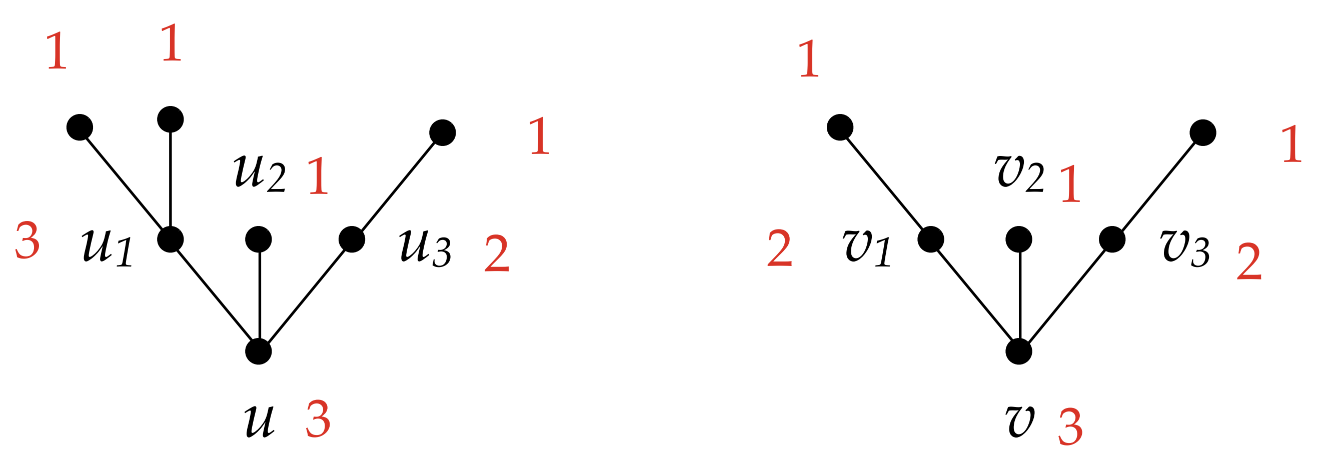
Output: Equivalence relation $\approx$ on V.

Inductively define: $\sim_0 \supseteq \sim_1 \supseteq \; \ldots \; \supseteq \sim_m = \sim_{m+1} =: \approx$

Initial: $u \sim_0 v$ iff $\deg(u) = \deg(v)$

Refine: $u \sim_{i+1} v$ iff $u \sim_i v$ and for all $\alpha \in V/\sim_i$:

$$\|N(u) \cap \alpha\| = \|N(v) \cap \alpha\|$$



$\alpha = \{w \mid \deg(w) = 2\}$

Weisfeiler-Lehman refinement

"colour refinement"
or "stable colouring"

Input: Graph $G = (V, E)$

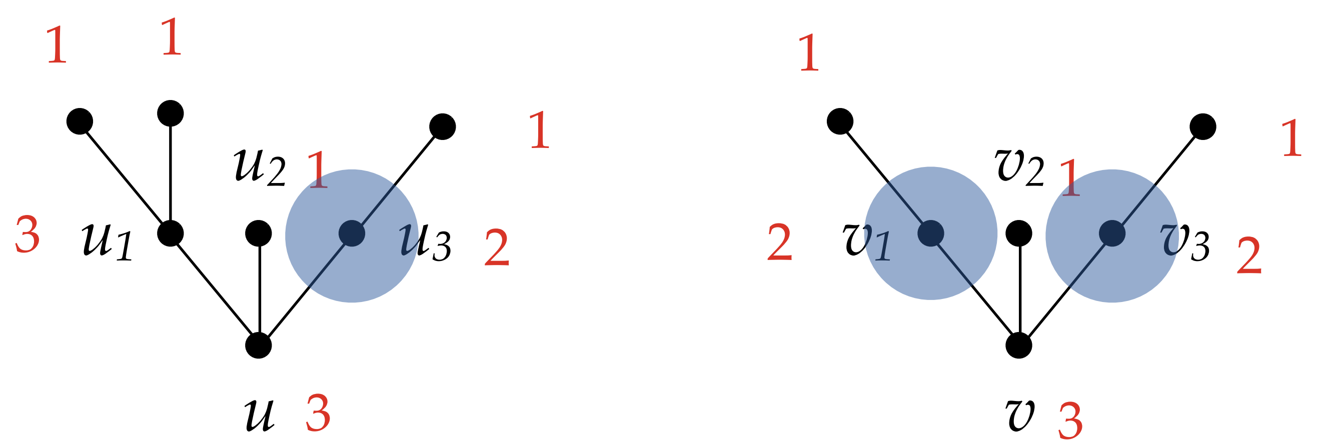
Output: Equivalence relation $\approx$ on V.

Inductively define: $\sim_0 \supseteq \sim_1 \supseteq \; \ldots \; \supseteq \sim_m = \sim_{m+1} =: \approx$

Initial: $u \sim_0 v$ iff $\deg(u) = \deg(v)$

Refine: $u \sim_{i+1} v$ iff $u \sim_i v$ and for all $\alpha \in V/\sim_i$:

$$\|N(u) \cap \alpha\| = \|N(v) \cap \alpha\|$$



$\alpha = \{w \mid \deg(w) = 2\}$

Weisfeiler-Lehman algorithm for GI

Input: Graphs $G = (V_G, E_G)$ and $H = (V_H, E_H)$

Output: "isomorphic" or "not isomorphic"

Weisfeiler-Lehman algorithm for GI

Input: Graphs $G = (V_G, E_G)$ and $H = (V_H, E_H)$
Output: "isomorphic" or "not isomorphic"

1. Compute the WL refinement $\approx$ on $G \dot\cup H$
2. Output "not isomorphic" if there is some $\alpha \in G \dot\cup H / \approx$ such that $\|\alpha \cap V_G\| \neq \|\alpha \cap V_H\|$; else "isomorphic".

Weisfeiler-Lehman algorithm for GI

Input: Graphs $G = (V_G, E_G)$ and $H = (V_H, E_H)$

Output: "isomorphic" or "not isomorphic"

1. Compute the WL refinement $\approx$ on $G \mathbin{\dot\cup} H$
2. Output "not isomorphic" if there is some $\alpha \in G \mathbin{\dot\cup} H / \approx$ such that $\|\alpha \cap V_G\| \neq \|\alpha \cap V_H\|$; else "isomorphic".

Some facts:

Weisfeiler-Lehman algorithm for GI

Input:	Graphs $G = (V_G, E_G)$ and $H = (V_H, E_H)$
Output:	"isomorphic" or "not isomorphic"

1. Compute the WL refinement $\approx$ on $G \dot\cup H$
2. Output "not isomorphic" if there is some $\alpha \in G \dot\cup H / \approx$ such that $\|\alpha \cap V_G\| \neq \|\alpha \cap V_H\|$; else "isomorphic".

Some facts:

1. WL runs in time $O(n^2 \log(n))$

Weisfeiler-Lehman algorithm for GI

Input: Graphs $G = (V_G, E_G)$ and $H = (V_H, E_H)$
Output: "isomorphic" or "not isomorphic"

1. Compute the WL refinement $\approx$ on $G \,\dot{\cup}\, H$
2. Output "not isomorphic" if there is some $\alpha \in G \,\dot{\cup}\, H / \approx$ such that $\|\alpha \cap V_G\| \neq \|\alpha \cap V_H\|$; else "isomorphic".

Some facts:

1. WL runs in time $O(n^2 \log(n))$
2. WL is correct almost surely

Babai, Erdös and Selkow (1980)

Weisfeiler-Lehman algorithm for GI

Input:	Graphs $G = (V_G, E_G)$ and $H = (V_H, E_H)$
Output:	"isomorphic" or "not isomorphic"

1. Compute the WL refinement $\approx$ on $G \dot{\cup} H$
2. Output "not isomorphic" if there is some $\alpha \in G \dot{\cup} H / \approx$ such that $\|\alpha \cap V_G\| \neq \|\alpha \cap V_H\|$; else "isomorphic".

Some facts:

1. WL runs in time $O(n^2 \log(n))$
2. WL is correct almost surely   Babai, Erdös and Selkow (1980)
3. WL fails on non-isomorphic regular graphs

k-dimensional WL* refinement

One-element extensions in $G = (V, E)$

For $\alpha \subseteq V^k$, a k-tuple $\vec{u} \in V^k$ and $0 \leq i < k$, let:

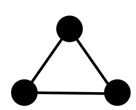
$$\Gamma_i(\vec{u}, \alpha) := \{w \in V \mid \vec{u}\frac{w}{i} \in \alpha\}$$

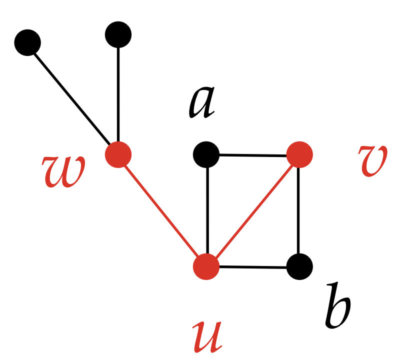
k-dimensional WL* refinement

One-element extensions in $G = (V, E)$

For $\alpha \subseteq V^k$, a k-tuple $\vec{u} \in V^k$ and $0 \leq i < k$, let:

$$\Gamma_i(\vec{u}, \alpha) := \{w \in V \mid \vec{u}\frac{w}{i} \in \alpha\}$$

Example: Let $k = 3$ and $\alpha := \{(x, y, z) \in V^3 \mid (x, y, z) =$  $\}$

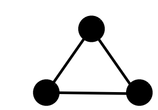


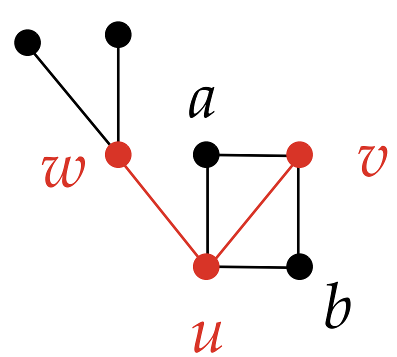
k-dimensional WL* refinement

One-element extensions in $G = (V, E)$

For $\alpha \subseteq V^k$, a k-tuple $\vec{u} \in V^k$ and $0 \leq i < k$, let:

$$\Gamma_i(\vec{u}, \alpha) := \{w \in V \mid \vec{u}\tfrac{w}{i} \in \alpha\}$$

Example: Let $k = 3$ and $\alpha := \{(x, y, z) \in V^3 \mid (x, y, z) =$  $\}$



$$\Gamma_0(uvw, \alpha) = \{a, b\}$$
$$\Gamma_1(uvw, \alpha) = \emptyset$$

k-dimensional WL* refinement

Input: Graph $G = (V, E)$

Output: Equivalence relation $\approx$ on V^k.

k-dimensional WL* refinement

Input: Graph $G = (V, E)$

Output: Equivalence relation $\approx$ on V^k.

Initial: $\vec{u} \sim_0 \vec{v}$ iff $\mathrm{atp}_G(\vec{u}) = \mathrm{atp}_G(\vec{v})$

k-dimensional WL* refinement

Input:	Graph $G = (V, E)$
Output:	Equivalence relation $\approx$ on V^k.

Initial:	$\vec{u} \sim_0 \vec{v}$	iff	$\mathrm{atp}_G(\vec{u}) = \mathrm{atp}_G(\vec{v})$
Refine:	$\vec{u} \sim_{m+1} \vec{v}$	iff	$\vec{u} \sim_m \vec{v}$ and for all $0 \leq i < k$ there is a bijection $f : V \to V$ s.t.

$$f : \Gamma_i(\vec{u}, \alpha) \mapsto \Gamma_i(\vec{v}, \alpha)$$

for all $\alpha \in V^k / \sim_m$

k-dimensional WL* refinement

Input: Graph $G = (V, E)$

Output: Equivalence relation $\approx$ on V^k.

Initial: $\vec{u} \sim_0 \vec{v}$ iff $\mathrm{atp}_G(\vec{u}) = \mathrm{atp}_G(\vec{v})$

Refine: $\vec{u} \sim_{m+1} \vec{v}$ iff $\vec{u} \sim_m \vec{v}$ and for all $0 \leq i < k$ there is a bijection $f : V \to V$ s.t.

$$\boxed{\Gamma_i(\vec{u}, \alpha) := \{w \in V \mid \vec{u}\tfrac{w}{i} \in \alpha\}} \longrightarrow \quad f : \Gamma_i(\vec{u}, \alpha) \mapsto \Gamma_i(\vec{v}, \alpha)$$

for all $\alpha \in V^k / \sim_m$

k-dimensional WL* refinement

Input: Graph $G = (V, E)$

Output: Equivalence relation $\approx$ on V^k.

Initial: $\vec{u} \sim_0 \vec{v}$ iff $\mathrm{atp}_G(\vec{u}) = \mathrm{atp}_G(\vec{v})$

Refine: $\vec{u} \sim_{m+1} \vec{v}$ iff $\vec{u} \sim_m \vec{v}$ and for all $0 \leq i < k$ there is a bijection $f : V \to V$ s.t.

$\boxed{\Gamma_i(\vec{u}, \alpha) := \{w \in V \mid \vec{u}\tfrac{w}{i} \in \alpha\}}$

$$f : \Gamma_i(\vec{u}, \alpha) \mapsto \Gamma_i(\vec{v}, \alpha)$$

for all $\alpha \in V^k / \sim_m$

Theorem: $\vec{u} \approx \vec{v}$ iff they agree on all C^k-formulas in G.

k-dimensional WL* algorithm for GI

As before: compute k-dimensional WL* refinement and compare across the two graphs.

PTIME for fixed k: k-dim WL* runs in time $O(n^{k+1}\log(n))$.

k-dimensional WL* algorithm for GI

As before: compute k-dimensional WL* refinement and compare across the two graphs.

PTIME for fixed k: k-dim WL* runs in time $O(n^{k+1}\log(n))$.

There exists a sequence of pairs $\{(G_n, H_n)\}_n$ of non-isomorphic graphs for which it holds that:

- G_n and H_n have O(n) vertices but
- G_n and H_n are not distinguished by the n-dim WL* algorithm.

Cai, Fürer and Immerman (1992)

Refinement by invertible linear maps

Two-element extensions in $G = (V, E)$

For $\alpha \subseteq V^k$, a k-tuple $\vec{u} \in V^k$ and $0 \leq i \neq j < k$, let:

$$\Gamma_{ij}(\vec{u}, \alpha) := \{(a, b) \in V \times V \mid \vec{u}\frac{a}{i}\frac{b}{j} \in \alpha\} \subseteq V \times V$$

Refinement by invertible linear maps

Two-element extensions in $G = (V, E)$

For $\alpha \subseteq V^k$, a k-tuple $\vec{u} \in V^k$ and $0 \leq i \neq j < k$, let:

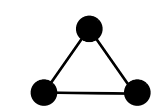
$$\Gamma_{ij}(\vec{u}, \alpha) := \{(a, b) \in V \times V \mid \vec{u}\tfrac{a}{i}\tfrac{b}{j} \in \alpha\} \subseteq V \times V$$ {0,1}-matrix

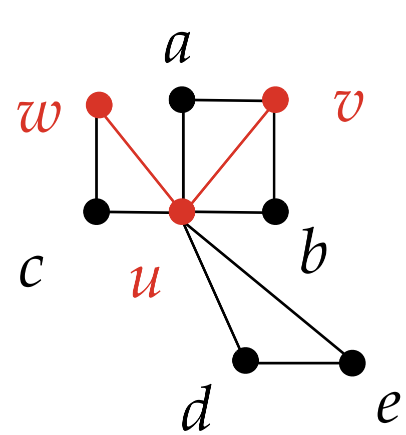
Refinement by invertible linear maps

Two-element extensions in $G = (V, E)$

For $\alpha \subseteq V^k$, a k-tuple $\vec{u} \in V^k$ and $0 \leq i \neq j < k$, let:

$$\Gamma_{ij}(\vec{u}, \alpha) := \{(a,b) \in V \times V \mid \vec{u}\tfrac{a}{i}\tfrac{b}{j} \in \alpha\} \subseteq V \times V$$ {0,1}-matrix

Example: Let $k = 3$ and $\alpha := \{(x,y,z) \in V^3 \mid (x,y,z) =$  $\}$

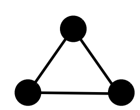


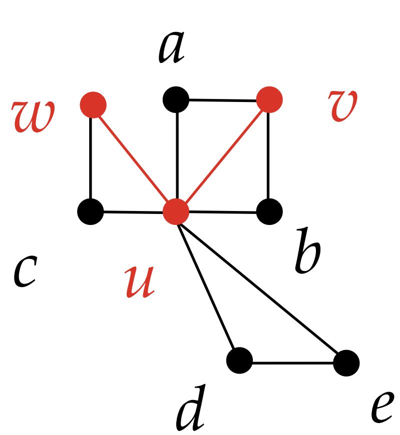
Refinement by invertible linear maps

Two-element extensions in $G = (V, E)$

For $\alpha \subseteq V^k$, a k-tuple $\vec{u} \in V^k$ and $0 \leq i \neq j < k$, let:

$$\Gamma_{ij}(\vec{u}, \alpha) := \{(a, b) \in V \times V \mid \vec{u}\frac{a}{i}\frac{b}{j} \in \alpha\} \subseteq V \times V$$ ⟵ {0,1}-matrix

Example: Let $k = 3$ and $\alpha := \{(x, y, z) \in V^3 \mid (x, y, z) =$  $\}$



Γ_{12}:

	u	v	w	a	b	c	d	e
u								
v				1	1			
w						1		
a		1						
b		1						
c			1					
d								1
e							1	

k-dimensional IM refinement over GF(p)

Input: Graph $G = (V, E)$

Output: Equivalence relation $\approx$ on V^k.

k-dimensional IM refinement over GF(p)

Input: Graph $G = (V, E)$

Output: Equivalence relation $\approx$ on V^k.

Initial: $\vec{u} \sim_0 \vec{v} \quad \text{iff} \quad \mathrm{atp}_G(\vec{u}) = \mathrm{atp}_G(\vec{v})$

k-dimensional IM refinement over GF(p)

Input: Graph $G = (V, E)$

Output: Equivalence relation $\approx$ on V^k.

Initial: $\vec{u} \sim_0 \vec{v}$ iff $\mathrm{atp}_G(\vec{u}) = \mathrm{atp}_G(\vec{v})$

Refine: $\vec{u} \sim_{m+1} \vec{v}$ iff $\vec{u} \sim_m \vec{v}$ and for all $0 \leq i \neq j < k$ there is $S \in \mathrm{GL}_V(\mathsf{GF}(p))$ s.t.

$$S \cdot \Gamma_{ij}(\vec{u}, \alpha) \cdot S^{-1} = \Gamma_{ij}(\vec{v}, \alpha)$$

for all $\alpha \in V^k / \sim_m$

k-dimensional IM_p algorithm for GI

Similar to WL: compute k-dimensional IM refinement and compare across the two graphs (here over $\mathrm{GF}(p)$)

k-dimensional IM_p algorithm for GI

Similar to WL: compute k-dimensional IM refinement and compare across the two graphs (here over GF(p))

- For each k, k-dim IM_p runs in polynomial time for all p.
- Refinement: k-dim WL* $\supseteq$ (k+1)-dim IM_p $\supseteq$ (k+2)-dim IM_p

k-dimensional IM_p algorithm for GI

Similar to WL: compute k-dimensional IM refinement and compare across the two graphs (here over GF(p))

- For each k, k-dim IM_p runs in polynomial time for all p.
- Refinement: k-dim WL* $\supseteq$ (k+1)-dim IM_p $\supseteq$ (k+2)-dim IM_p

For each k and prime p, there is a pair of non-isomorphic graphs that can be distinguished by 3-dim IM_p but not by k-dim WL*.

Dawar and H. (2012)

k-dimensional IM_p algorithm for GI

Similar to WL: compute k-dimensional IM refinement and compare across the two graphs (here over GF(p))

- For each k, k-dim IM_p runs in polynomial time for all p.
- Refinement: k-dim WL* $\supseteq$ (k+1)-dim IM_p $\supseteq$ (k+2)-dim IM_p

For each k and prime p, there is a pair of non-isomorphic graphs that can be distinguished by 3-dim IM_p but not by k-dim WL*.

Dawar and H. (2012)

For each k and distinct primes p and q, there is a pair of non-isomorphic graphs that can be distinguished by 3-dim IM_p but not by k-dim IM_q.

H. (2010)

k-dimensional IM_p more generally

Consider the invertible-map algorithm for larger matrices (higher arity) and finite sets of primes.

Can we give instances where the general algorithm fails to express graph isomorphism?

Some open problems

Problem 1: Separate FOR_p and FOR_q over empty signatures

For formula $\varphi(x, y)$, integer n and prime p, let $r^p_\varphi(n)$ denote the GF(p)-rank of the matrix defined by $\varphi(x, y)$ over an n-element set.

Problem 1: Separate FOR_p and FOR_q over empty signatures

For formula $\varphi(x, y)$, integer n and prime p, let $r^p_\varphi(n)$ denote the GF(p)-rank of the matrix defined by $\varphi(x, y)$ over an n-element set.

Polynomial-rank conjecture

For each $\varphi(x, y)$ and each prime p, there are unary polynomials $f_0, \ldots, f_{p\text{-}1}$ such that $r^p_\varphi(n) = f_i(n)$ for all (sufficiently large) n congruent to i modulo p.

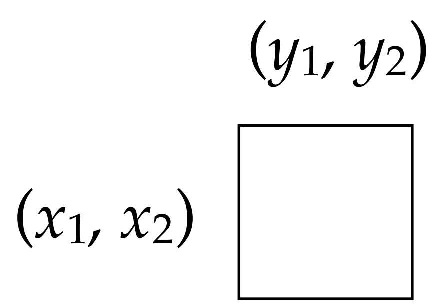
Problem 1: Separate FOR_p and FOR_q over empty signatures

For formula $\varphi(x, y)$, integer n and prime p, let $r^p_\varphi(n)$ denote the GF(p)-rank of the matrix defined by $\varphi(x, y)$ over an n-element set.

Polynomial-rank conjecture

For each $\varphi(x, y)$ and each prime p, there are unary polynomials $f_0, \ldots, f_{p\text{-}1}$ such that $r^p_\varphi(n) = f_i(n)$ for all (sufficiently large) n congruent to i modulo p.

True for:



H. and Laubner (2010)

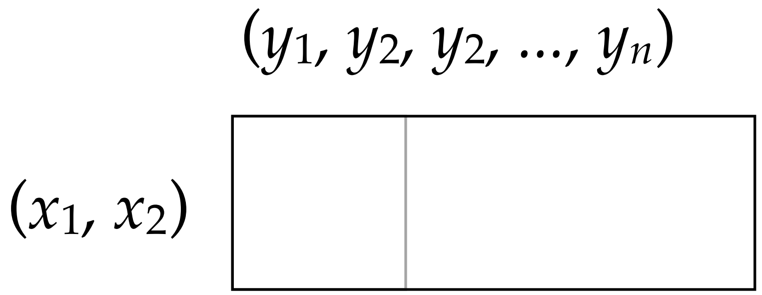
Problem 1: Separate FOR_p and FOR_q over empty signatures

For formula $\varphi(x, y)$, integer *n* and prime *p*, let $r^p_\varphi(n)$ denote the GF(*p*)-rank of the matrix defined by $\varphi(x, y)$ over an *n*-element set.

Polynomial-rank conjecture

For each $\varphi(x, y)$ and each prime *p*, there are unary polynomials $f_0, \ldots, f_{p-1}$ such that $r^p_\varphi(n) = f_i(n)$ for all (sufficiently large) *n* congruent to *i* modulo *p*.

True for:



H. and Laubner (2010)

Kirsten (2012)

Problem 1: Separate FOR_p and FOR_q over empty signatures

For formula $\varphi(x,y)$, integer n and prime p, let $r^p_\varphi(n)$ denote the GF(p)-rank of the matrix defined by $\varphi(x,y)$ over an n-element set.

Polynomial-rank conjecture

For each $\varphi(x,y)$ and each prime p, there are unary polynomials $f_0, \ldots, f_{p\text{-}1}$ such that $r^p_\varphi(n) = f_i(n)$ for all (sufficiently large) n congruent to i modulo p.

???

$(y_1, y_2, y_2, \ldots, y_n)$

$(x_1, x_2, \ldots, x_n)$

H. and Laubner (2010)

Kirsten (2012)

Problem 2: Give capturing results for FPR on natural classes of graphs

Consider classes on which we know that FPC does *not* capture PTIME:

- graphs of bounded degree
- graphs of bounded colour-class size

Further questions

- Can FPR express matching in arbitrary graphs?
- Does the “simultaneous similarity game” correspond to a natural logic?

More open problems to come in the next talk!