# On the complexity of \#CSP 

Martin Dyer

University of Leeds

Fields Institute
Toronto
Wednesday, 3rd August, 2011
(joint work with David Richerby)

# (1) Introduction 

(2) Rectangularity
(3) Frames
(4) Counting
(5) Decidability

6 Conclusion

## Definitions and notation

A constraint language $\Gamma$ is a collection of named relations over a fixed finite set $D$, the domain.

An instance has a set of variables $V=\left\{v_{1}, v_{2}, \ldots, v_{n}\right\}$ and a finite collection of constraints, $\mathcal{C}$.

A constraint has the form $R\left(v_{i_{1}}, \ldots, v_{i_{k}}\right)$, where $R \in \Gamma$ has arity $k$, and $v_{i_{1}}, \ldots, v_{i_{k}} \in V$, not necessarily distinct.

An assignment is a mapping $\sigma: V \rightarrow D$. It is satisfying if $\left(\sigma\left(v_{1}\right), \ldots, \sigma\left(v_{k}\right)\right) \in R$, for every constraint in $\mathcal{C}$

We write $\operatorname{CSP}(\Gamma)$ for CSP with all constraints from $\Gamma$.
In non-uniform CSP, we regard $D$ and $\Gamma$ as being fixed constants. We measure the size of the input by the number of variables, $n$.

## Definitions and notation

A constraint language $\Gamma$ is a collection of named relations over a fixed finite set $D$, the domain.

An instance has a set of variables $V=\left\{v_{1}, v_{2}, \ldots, v_{n}\right\}$ and a finite collection of constraints, $\mathcal{C}$.

A constraint has the form $R\left(v_{i_{1}}, \ldots, v_{i_{k}}\right)$, where $R \in \Gamma$ has arity $k$, and $v_{i_{1}}, \ldots, v_{i_{k}} \in V$, not necessarily distinct.

An assignment is a mapping $\sigma: V \rightarrow D$. It is satisfying if $\left(\sigma\left(v_{1}\right), \ldots, \sigma\left(v_{k}\right)\right) \in R$, for every constraint in $\mathcal{C}$.

We write $\operatorname{CSP}(\Gamma)$ for CSP with all constraints from $\Gamma$.
In non-uniform CSP, we regard $D$ and $\Gamma$ as being fixed constants. We measure the size of the input by the number of variables, $n$.

## Definitions and notation

A constraint language $\Gamma$ is a collection of named relations over a fixed finite set $D$, the domain.

An instance has a set of variables $V=\left\{v_{1}, v_{2}, \ldots, v_{n}\right\}$ and a finite collection of constraints, $\mathcal{C}$.

A constraint has the form $R\left(v_{i_{1}}, \ldots, v_{i_{k}}\right)$, where $R \in \Gamma$ has arity $k$, and $v_{i_{1}}, \ldots, v_{i_{k}} \in V$, not necessarily distinct.

An assignment is a mapping $\sigma: V \rightarrow D$. It is satisfying if $\left(\sigma\left(v_{1}\right), \ldots, \sigma\left(v_{k}\right)\right) \in R$, for every constraint in $\mathcal{C}$.

We write CSP(Г) for CSP with all constraints from Г.
In non-uniform CSP, we regard $D$ and $\Gamma$ as being fixed constants. We measure the size of the input by the number of variables, $n$.

## Definitions and notation

A constraint language $\Gamma$ is a collection of named relations over a fixed finite set $D$, the domain.

An instance has a set of variables $V=\left\{v_{1}, v_{2}, \ldots, v_{n}\right\}$ and a finite collection of constraints, $\mathcal{C}$.

A constraint has the form $R\left(v_{i_{1}}, \ldots, v_{i_{k}}\right)$, where $R \in \Gamma$ has arity $k$, and $v_{i_{1}}, \ldots, v_{i_{k}} \in V$, not necessarily distinct.

An assignment is a mapping $\sigma: V \rightarrow D$. It is satisfying if $\left(\sigma\left(v_{1}\right), \ldots, \sigma\left(v_{k}\right)\right) \in R$, for every constraint in $\mathcal{C}$.

We write CSP(Г) for CSP with all constraints from Г.
In non-uniform CSP, we regard $D$ and $\Gamma$ as being fixed constants. We measure the size of the input by the number of variables, $n$.

## Definitions and notation

A constraint language $\Gamma$ is a collection of named relations over a fixed finite set $D$, the domain.

An instance has a set of variables $V=\left\{v_{1}, v_{2}, \ldots, v_{n}\right\}$ and a finite collection of constraints, $\mathcal{C}$.

A constraint has the form $R\left(v_{i_{1}}, \ldots, v_{i_{k}}\right)$, where $R \in \Gamma$ has arity $k$, and $v_{i_{1}}, \ldots, v_{i_{k}} \in V$, not necessarily distinct.

An assignment is a mapping $\sigma: V \rightarrow D$. It is satisfying if $\left(\sigma\left(v_{1}\right), \ldots, \sigma\left(v_{k}\right)\right) \in R$, for every constraint in $\mathcal{C}$.

We write $\operatorname{CSP}(\Gamma)$ for CSP with all constraints from $\Gamma$.
In non-uniform CSP, we regard $D$ and $\Gamma$ as being fixed constants. We measure the size of the input by the number of variables, $n$.

## Definitions and notation

A constraint language $\Gamma$ is a collection of named relations over a fixed finite set $D$, the domain.

An instance has a set of variables $V=\left\{v_{1}, v_{2}, \ldots, v_{n}\right\}$ and a finite collection of constraints, $\mathcal{C}$.

A constraint has the form $R\left(v_{i_{1}}, \ldots, v_{i_{k}}\right)$, where $R \in \Gamma$ has arity $k$, and $v_{i_{1}}, \ldots, v_{i_{k}} \in V$, not necessarily distinct.

An assignment is a mapping $\sigma: V \rightarrow D$. It is satisfying if $\left(\sigma\left(v_{1}\right), \ldots, \sigma\left(v_{k}\right)\right) \in R$, for every constraint in $\mathcal{C}$.

We write $\operatorname{CSP}(\Gamma)$ for CSP with all constraints from $Г$.
In non-uniform CSP, we regard $D$ and $\Gamma$ as being fixed constants. We measure the size of the input by the number of variables, $n$.

## Decision v. counting

For a given input, there are (at least) two questions we can ask:

- Decision: is there any satisfying assignment for the given instance?
- Counting: how many satisfying assignments are there?

For a given $\Gamma$, we can generalise these questions as follows:

- CSP(Г): what is the complexity of determining any satisfying assignment for an arbitrary instance?
- \#CSP $(\Gamma)$ : what is the complexity of determining how many satisfying assignments there are for an arbitrary instance?

The computational complexity is a function of $n$, the number of variables. Here we will be concerned mostly with \# $\operatorname{HSP}(\Gamma)$

## Decision v. counting

For a given input, there are (at least) two questions we can ask:

- Decision: is there any satisfying assignment for the given instance?
- Counting: how many satisfying assignments are there?

> For a given $\Gamma$, we can generalise these questions as follows:
> - $\operatorname{CSP}(\Gamma)$ : what is the complexity of determining any satisfying assignment for an arbitrary instance?
> - \#CSP(Г): what is the complexity of determining how many satisfying assignments there are for an arbitrary instance?

The computational complexity is a function of $n$, the number of variables. Here we will be concerned mostly with \#CSP( $\Gamma$ ).

## Decision v. counting

For a given input, there are (at least) two questions we can ask:

- Decision: is there any satisfying assignment for the given instance?
- Counting: how many satisfying assignments are there?

> For a given $\Gamma$, we can generalise these questions as follows:
> - $\operatorname{CSP}(\Gamma)$ : what is the complexity of determining any satisfying assignment for an arbitrary instance?
> - \#CSP(Г): what is the complexity of determining how many satisfying assignments there are for an arbitrary instance?

The computational complexity is a function of $n$, the number of variables. Here we will be concerned mostly with \# $\operatorname{CSP}(\Gamma)$

## Decision v. counting

For a given input, there are (at least) two questions we can ask:

- Decision: is there any satisfying assignment for the given instance?
- Counting: how many satisfying assignments are there?

For a given $\Gamma$, we can generalise these questions as follows:

- $\operatorname{CSP}(\Gamma)$ : what is the complexity of determining any satisfying assignment for an arbitrary instance?
- \#CSP(Г): what is the complexity of determining how many satisfying assignments there are for an arbitrary instance?

The computational complexity is a function of $n$, the number of variables. Here we will be concerned mostly with \# $\operatorname{CSP}(\Gamma)$

## Decision v. counting

For a given input, there are (at least) two questions we can ask:

- Decision: is there any satisfying assignment for the given instance?
- Counting: how many satisfying assignments are there?

For a given $\Gamma$, we can generalise these questions as follows:

- $\operatorname{CSP}(\Gamma)$ : what is the complexity of determining any satisfying assignment for an arbitrary instance?
- \#CSP $(\Gamma)$ : what is the complexity of determining how many satisfying assignments there are for an arbitrary instance?

The computational complexity is a function of $n$, the number of variables. Here we will be concerned mostly with \# $\#$ CSP( $(\Gamma)$

## Decision v. counting

For a given input, there are (at least) two questions we can ask:

- Decision: is there any satisfying assignment for the given instance?
- Counting: how many satisfying assignments are there?

For a given $\Gamma$, we can generalise these questions as follows:

- $\operatorname{CSP}(\Gamma)$ : what is the complexity of determining any satisfying assignment for an arbitrary instance?
- \#CSP(Г): what is the complexity of determining how many satisfying assignments there are for an arbitrary instance?

The computational complexity is a function of $n$, the number of variables. Here we will be concerned mostly with \#CSP( $(\top)$

## Decision v. counting

For a given input, there are (at least) two questions we can ask:

- Decision: is there any satisfying assignment for the given instance?
- Counting: how many satisfying assignments are there?

For a given $\Gamma$, we can generalise these questions as follows:

- $\operatorname{CSP}(Г)$ : what is the complexity of determining any satisfying assignment for an arbitrary instance?
- \#CSP(Г): what is the complexity of determining how many satisfying assignments there are for an arbitrary instance?

The computational complexity is a function of $n$, the number of variables.
Here we will be concerned mostly with \#CSP(Г)

## Decision v. counting

For a given input, there are (at least) two questions we can ask:

- Decision: is there any satisfying assignment for the given instance?
- Counting: how many satisfying assignments are there?

For a given $\Gamma$, we can generalise these questions as follows:

- $\operatorname{CSP}(Г)$ : what is the complexity of determining any satisfying assignment for an arbitrary instance?
- \#CSP(Г): what is the complexity of determining how many satisfying assignments there are for an arbitrary instance?

The computational complexity is a function of $n$, the number of variables.
Here we will be concerned mostly with $\# \mathrm{CSP}(\Gamma)$.

## Homomorphisms

An well-known alternative view is to regard a satisfying assignment as a homomorphism from a finite structure determined by the variables and constraints to a finite structure determined by the domain and constraint language.

Then problems like graph homomorphisms can be viewed within the CSP framework. This view is often convenient and, in fact, predates constraint satisfaction

Any use of the word "homomorphism" in this context can always be replaced by "satisfying assignment"

So counting homomorphisms and \#CSP are essentially the same.

## Homomorphisms

An well-known alternative view is to regard a satisfying assignment as a homomorphism from a finite structure determined by the variables and constraints to a finite structure determined by the domain and constraint language.

Then problems like graph homomorphisms can be viewed within the CSP framework. This view is often convenient and, in fact, predates constraint satisfaction.

Any use of the word "homomorphism" in this context can always be replaced by "satisfying assignment"

So counting homomorphisms and \#CSP are essentially the same.

## Homomorphisms

An well-known alternative view is to regard a satisfying assignment as a homomorphism from a finite structure determined by the variables and constraints to a finite structure determined by the domain and constraint language.

Then problems like graph homomorphisms can be viewed within the CSP framework. This view is often convenient and, in fact, predates constraint satisfaction.

Any use of the word "homomorphism" in this context can always be replaced by "satisfying assignment".

So counting homomorphisms and \#CSP are essentially the same.

## Homomorphisms

An well-known alternative view is to regard a satisfying assignment as a homomorphism from a finite structure determined by the variables and constraints to a finite structure determined by the domain and constraint language.

Then problems like graph homomorphisms can be viewed within the CSP framework. This view is often convenient and, in fact, predates constraint satisfaction.

Any use of the word "homomorphism" in this context can always be replaced by "satisfying assignment".

So counting homomorphisms and \#CSP are essentially the same.

## Weighted counting

A natural generalisation of \#CSP is to assign a (non-negative) numerical weight $w_{R}(\mathbf{u})$ to each tuple $\mathbf{u} \in R$ for the relations $R \in \Gamma$.

Then a relation can be regarded as a weight function that takes only the values 1 and 0 , where $1 / 0$ signifies in/not in the relation.

If $\mathcal{C}$ is the set of constraints, then the weight of an assignment $\sigma$ is:

and the partition function is the sum of these weights:


The weighted counting problem is that of evaluating this partition function for a given instance.

## Weighted counting

A natural generalisation of \#CSP is to assign a (non-negative) numerical weight $w_{R}(\mathbf{u})$ to each tuple $\mathbf{u} \in R$ for the relations $R \in \Gamma$.

Then a relation can be regarded as a weight function that takes only the values 1 and 0 , where $1 / 0$ signifies in/not in the relation.

If $\mathcal{C}$ is the set of constraints, then the weight of an assignment $\sigma$ is:

and the partition function is the sum of these weights:


The weighted counting problem is that of evaluating this partition function for a given instance.

## Weighted counting

A natural generalisation of \#CSP is to assign a (non-negative) numerical weight $w_{R}(\mathbf{u})$ to each tuple $\mathbf{u} \in R$ for the relations $R \in \Gamma$.

Then a relation can be regarded as a weight function that takes only the values 1 and 0 , where $1 / 0$ signifies in/not in the relation.

If $\mathcal{C}$ is the set of constraints, then the weight of an assignment $\sigma$ is:

$$
W(\sigma)=\prod_{R\left(v_{i_{1}}, \ldots, v_{i_{k}}\right) \in \mathcal{C}} w_{R}\left(\sigma\left(v_{i_{1}}\right), \ldots, \sigma\left(v_{i_{k}}\right)\right),
$$

and the partition function is the sum of these weights:

The weighted counting problem is that of evaluating this partition function for a given instance.

## Weighted counting

A natural generalisation of \#CSP is to assign a (non-negative) numerical weight $w_{R}(\mathbf{u})$ to each tuple $\mathbf{u} \in R$ for the relations $R \in \Gamma$.

Then a relation can be regarded as a weight function that takes only the values 1 and 0 , where $1 / 0$ signifies in/not in the relation.

If $\mathcal{C}$ is the set of constraints, then the weight of an assignment $\sigma$ is:

$$
W(\sigma)=\prod_{R\left(v_{i_{1}}, \ldots, v_{i_{k}}\right) \in \mathcal{C}} w_{R}\left(\sigma\left(v_{i_{1}}\right), \ldots, \sigma\left(v_{i_{k}}\right)\right),
$$

and the partition function is the sum of these weights:

$$
\sum_{\sigma: V \rightarrow D} W(\sigma)
$$

The weighted counting problem is that of evaluating this partition function for a given instance.

## Weighted counting

A natural generalisation of \#CSP is to assign a (non-negative) numerical weight $w_{R}(\mathbf{u})$ to each tuple $\mathbf{u} \in R$ for the relations $R \in \Gamma$.

Then a relation can be regarded as a weight function that takes only the values 1 and 0 , where $1 / 0$ signifies in/not in the relation.

If $\mathcal{C}$ is the set of constraints, then the weight of an assignment $\sigma$ is:

$$
W(\sigma)=\prod_{R\left(v_{i_{1}}, \ldots, v_{i_{k}}\right) \in \mathcal{C}} w_{R}\left(\sigma\left(v_{i_{1}}\right), \ldots, \sigma\left(v_{i_{k}}\right)\right),
$$

and the partition function is the sum of these weights:

$$
\sum_{\sigma: V \rightarrow D} W(\sigma)
$$

The weighted counting problem is that of evaluating this partition function for a given instance.

## Complexity classes

The classes P and NP relate to decision problems.
The reductions between problems are (usually) many-one reductions.

> For counting, the appropriate complexity classes are
> FD: the set of functions $\sum^{*} \rightarrow \mathbb{N}$ computed by deterministic poly-time Turing machines.
> \#P: the set of functions $f$ for which there is a nondeterministic poly-time Turing machine with $f(x)$ accepting paths for input $x$ The reductions between problems are (usually) Turing reductions For weighted counting, the oracle class FP\#P is appropriate TonA (1901) showed \#P-complete is much harder than ND-complete. LADNER (1975) showed that (if $P \neq N P$ ) there is an infinite number of complexity classes between P and NP-complete, and a straightforward modification of his proof shows that (if $F P \neq \# P$ ) there is an infinite number of complexity classes between FP and \#P-complete.

## Complexity classes

The classes P and NP relate to decision problems.
The reductions between problems are (usually) many-one reductions.
For counting, the appropriate complexity classes are:


## Complexity classes

The classes P and NP relate to decision problems.
The reductions between problems are (usually) many-one reductions.
For counting, the appropriate complexity classes are:
FP: the set of functions $\Sigma^{*} \rightarrow \mathbb{N}$ computed by deterministic poly-time Turing machines.


## Complexity classes

The classes P and NP relate to decision problems.
The reductions between problems are (usually) many-one reductions.
For counting, the appropriate complexity classes are:
FP: the set of functions $\Sigma^{*} \rightarrow \mathbb{N}$ computed by deterministic poly-time Turing machines.
\#P: the set of functions $f$ for which there is a nondeterministic poly-time Turing machine with $f(x)$ accepting paths for input $x$. The reductions between problems are (usually) Turing reductions.


## Complexity classes

The classes P and NP relate to decision problems.
The reductions between problems are (usually) many-one reductions.
For counting, the appropriate complexity classes are:
FP: the set of functions $\Sigma^{*} \rightarrow \mathbb{N}$ computed by deterministic poly-time Turing machines.
\#P: the set of functions $f$ for which there is a nondeterministic poly-time Turing machine with $f(x)$ accepting paths for input $x$. The reductions between problems are (usually) Turing reductions. For weighted counting, the oracle class FP\#P is appropriate.


## Complexity classes

The classes P and NP relate to decision problems.
The reductions between problems are (usually) many-one reductions.
For counting, the appropriate complexity classes are:
FP: the set of functions $\Sigma^{*} \rightarrow \mathbb{N}$ computed by deterministic poly-time Turing machines.
\#P: the set of functions $f$ for which there is a nondeterministic poly-time Turing machine with $f(x)$ accepting paths for input $x$. The reductions between problems are (usually) Turing reductions.
For weighted counting, the oracle class FP\#P is appropriate.
ToDA (1991) showed \#P-complete is much harder than NP-complete.
LADNER (1975) showed that (if $P \neq N P$ ) there is an infinite number of complexity classes between P and NP-complete, and a straightforward modification of his proof shows that (if $\mathrm{FP} \neq \# \mathrm{P}$ ) there is an infinite number of complexity classes between FP and \#P-complete.

## Complexity classes

The classes P and NP relate to decision problems.
The reductions between problems are (usually) many-one reductions.
For counting, the appropriate complexity classes are:
FP: the set of functions $\Sigma^{*} \rightarrow \mathbb{N}$ computed by deterministic poly-time Turing machines.
\#P: the set of functions $f$ for which there is a nondeterministic poly-time Turing machine with $f(x)$ accepting paths for input $x$. The reductions between problems are (usually) Turing reductions.
For weighted counting, the oracle class FP\#P is appropriate.
TodA (1991) showed \#P-complete is much harder than NP-complete.
LADNER (1975) showed that (if $P \neq N P$ ) there is an infinite number of complexity classes between P and NP-complete, and a straightforward modification of his proof shows that (if $\mathrm{FP} \neq \# \mathrm{P}$ ) there is an infinite number of complexity classes between FP and \#P-complete.

## Counting dichotomy

Corresponding to the dichotomy conjecture for $\operatorname{CSP}(\Gamma)$, we have

## Conjecture

 \#CSP $(\Gamma)$ is either in FP or is \#P-complete, for all $Г$.NB: If $\operatorname{CSP}(\Gamma)$ is NP-complete then \#CSP $(\Gamma)$ is obviously hard, but it is not known that this implies \#P-completeness. If $\operatorname{CSP}(\Gamma) \in P$, this certainly does not imply that $\# C S P(\Gamma) \in F P($ e.g. 2SAT $)$

The conjecture was known to be true in special cases, e.g.

- the Boolean (2-element) domain (Creignou \& Hermann, 1996).
- the edge relation of undirected graph (Dyer \& Greenhill, 2000)
- the edge relation of of a directed acyclic graph (DAG) (Dyer, Goldberg \& Paterson, 2007)


## Counting dichotomy

Corresponding to the dichotomy conjecture for $\operatorname{CSP}(\Gamma)$, we have

## Conjecture

\#CSP $(\Gamma)$ is either in FP or is \#P-complete, for all $Г$.

NB: If $\operatorname{CSP}(\Gamma)$ is NP-complete then \#CSP $(\Gamma)$ is obviously hard, but it is not known that this implies \#P-completeness. If $\operatorname{CSP}(\Gamma) \in P$, this certainly does not imply that $\# C S P(\Gamma) \in F P$ (e.g. 2SAT).

The conjecture was known to be true in special cases, e.g.

- the Boolean (2-element) domain (Creignou \& Hermann, 1996)
- the edge relation of undirected graph (Dyer \& Greenhill, 2000)
- the edge relation of of a directed acyclic graph (DAG) (Dyer, Goldberg \& Paterson, 2007)


## Counting dichotomy

Corresponding to the dichotomy conjecture for $\operatorname{CSP}(\Gamma)$, we have

## Conjecture

 \#CSP $(\Gamma)$ is either in FP or is \#P-complete, for all $Г$.NB: If $\operatorname{CSP}(\Gamma)$ is NP-complete then \#CSP $(\Gamma)$ is obviously hard, but it is not known that this implies \#P-completeness. If $\operatorname{CSP}(\Gamma) \in P$, this certainly does not imply that $\# C S P(\Gamma) \in F P$ (e.g. 2SAT).

The conjecture was known to be true in special cases, e.g.

- the Boolean (2-element) domain (Creignou \& Hermann, 1996)
- the edge relation of undirected graph (Dyer \& Greenhill, 2000)
- the edge relation of of a directed acyclic graph (DAG) (Dyer, Goldberg \& Paterson, 2007).


## Counting dichotomy

Corresponding to the dichotomy conjecture for $\operatorname{CSP}(\Gamma)$, we have

## Conjecture

 \#CSP $(\Gamma)$ is either in FP or is \#P-complete, for all $Г$.NB: If $\operatorname{CSP}(\Gamma)$ is NP-complete then $\# \operatorname{CSP}(\Gamma)$ is obviously hard, but it is not known that this implies \#P-completeness. If $\operatorname{CSP}(\Gamma) \in P$, this certainly does not imply that $\# C S P(\Gamma) \in F P$ (e.g. 2SAT).

The conjecture was known to be true in special cases, e.g.

- the Boolean (2-element) domain (Creignou \& Hermann, 1996).
- the edge relation of undirected graph (Dyer \& Greenhill, 2000)
- the edge relation of of a directed acyclic graph (DAG) (Dyer, Goldberg \& Paterson, 2007)


## Counting dichotomy

Corresponding to the dichotomy conjecture for $\operatorname{CSP}(\Gamma)$, we have

## Conjecture

 \#CSP $(\Gamma)$ is either in FP or is \#P-complete, for all $Г$.NB: If $\operatorname{CSP}(\Gamma)$ is NP-complete then $\# \operatorname{CSP}(\Gamma)$ is obviously hard, but it is not known that this implies \#P-completeness. If $\operatorname{CSP}(\Gamma) \in P$, this certainly does not imply that $\# C S P(\Gamma) \in F P$ (e.g. 2SAT).

The conjecture was known to be true in special cases, e.g.

- the Boolean (2-element) domain (Creignou \& Hermann, 1996).
- the edge relation of undirected graph (Dyer \& Greenhill, 2000).
- the edge relation of of a directed acyclic graph (DAG) (Dyer, Goldberg \& Paterson, 2007)


## Counting dichotomy

Corresponding to the dichotomy conjecture for $\operatorname{CSP}(\Gamma)$, we have

## Conjecture

 \#CSP $(\Gamma)$ is either in FP or is \#P-complete, for all $Г$.NB: If $\operatorname{CSP}(\Gamma)$ is NP-complete then $\# \operatorname{CSP}(\Gamma)$ is obviously hard, but it is not known that this implies \#P-completeness. If $\operatorname{CSP}(\Gamma) \in P$, this certainly does not imply that $\# C S P(\Gamma) \in F P$ (e.g. 2SAT).

The conjecture was known to be true in special cases, e.g.

- the Boolean (2-element) domain (Creignou \& Hermann, 1996).
- the edge relation of undirected graph (DyER \& Greenhill, 2000).
- the edge relation of of a directed acyclic graph (DAG) (Dyer, Goldberg \& Paterson, 2007).


## Counting dichotomy

Corresponding to the dichotomy conjecture for $\operatorname{CSP}(\Gamma)$, we have

## Conjecture

 \#CSP $(\Gamma)$ is either in FP or is \#P-complete, for all $Г$.NB: If $\operatorname{CSP}(\Gamma)$ is NP-complete then $\# \operatorname{CSP}(\Gamma)$ is obviously hard, but it is not known that this implies \#P-completeness. If $\operatorname{CSP}(\Gamma) \in P$, this certainly does not imply that $\# C S P(\Gamma) \in F P$ (e.g. 2SAT).

The conjecture was known to be true in special cases, e.g.

- the Boolean (2-element) domain (Creignou \& Hermann, 1996).
- the edge relation of undirected graph (DyER \& Greenhill, 2000).
- the edge relation of of a directed acyclic graph (DAG) (Dyer, Goldberg \& Paterson, 2007).


## Breakthrough

However, unlike the decision case, this conjecture has been settled.

## Theorem (Bulatov, 2008) <br> 

- the proof is long, and requires a good understanding of universal algebra, including lattice theory, tame congruence theory and commutator theory.
- the FP algorithm requires first transforming an instance to a much larger subdirect product form, and its overall time complexity is far from clear
- the criterion for the dichotomy (congruence singularity) isn't shown to be decidable.


## Breakthrough

However, unlike the decision case, this conjecture has been settled.

```
Theorem (Bulatov, 2008)
#CSP(\Gamma) is either in FP or is #P-complete, for all Г.
```

- the proof is long, and requires a good understanding of universal algebra, including lattice theory, tame congruence theory and commutator theory.
- the FP algorithm requires first transforming an instance to a much larger subdirect product form, and its overall time complexity is far from clear
- the criterion for the dichotomy (congruence singularity) isn't shown to be decidable.


## Breakthrough

However, unlike the decision case, this conjecture has been settled.

```
Theorem (Bulatov, 2008)
#CSP(Г) is either in FP or is #P-complete, for all Г.
```

But ...

- the proof is long, and requires a good understanding of universal algebra, including lattice theory, tame congruence theory and commutator theory.
- the FP algorithm requires first transforming an instance to a much larger subdirect product form, and its overall time complexity is far from clear
- the criterion for the dichotomy (congruence singularity) isn't shown to be decidable.


## Breakthrough

However, unlike the decision case, this conjecture has been settled.

```
Theorem (Bulatov, 2008)
#CSP(Г) is either in FP or is #P-complete, for all Г.
```

But ...

- the proof is long, and requires a good understanding of universal algebra, including lattice theory, tame congruence theory and commutator theory.
- the FP algorithm requires first transforming an instance to a much larger subdirect product form, and its overall time complexity is far from clear.
- the criterion for the dichotomy (congruence singularity) isn't shown
to be decidable.


## Breakthrough

However, unlike the decision case, this conjecture has been settled.

```
Theorem (Bulatov, 2008)
#CSP(\Gamma) is either in FP or is #P-complete, for all }\Gamma\mathrm{ .
```

But ...

- the proof is long, and requires a good understanding of universal algebra, including lattice theory, tame congruence theory and commutator theory.
- the FP algorithm requires first transforming an instance to a much larger subdirect product form, and its overall time complexity is far from clear.
- the criterion for the dichotomy (congruence singularity) isn't shown to be decidable.


## Breakthrough

However, unlike the decision case, this conjecture has been settled.

```
Theorem (Bulatov, 2008)
#CSP(\Gamma) is either in FP or is #P-complete, for all }\Gamma\mathrm{ .
```

But ...

- the proof is long, and requires a good understanding of universal algebra, including lattice theory, tame congruence theory and commutator theory.
- the FP algorithm requires first transforming an instance to a much larger subdirect product form, and its overall time complexity is far from clear.
- the criterion for the dichotomy (congruence singularity) isn't shown to be decidable.


## Our results

- An elementary, and relatively short proof of Bulatov's dichotomy for \#CSP $(\Gamma)$, using a new criterion.
- A natural algorithm, with proven time complexity, for the class of problems in FP
- And, most importantly,
decidability of the new criterion.


## Our results

- An elementary, and relatively short proof of Bulatov's dichotomy for \#CSP $(\Gamma)$, using a new criterion.
- A natural algorithm, with proven time complexity, for the class of problems in FP.
By-product: an improved algorithm for $\operatorname{CSP}(\Gamma)$ when $\Gamma$ is "strongly rectangular"
- And, most importantly,
decidability of the new criterion.


## Our results

- An elementary, and relatively short proof of Bulatov's dichotomy for \#CSP $(\Gamma)$, using a new criterion.
- A natural algorithm, with proven time complexity, for the class of problems in FP.
By-product: an improved algorithm for $\operatorname{CSP}(\Gamma)$ when $\Gamma$ is "strongly rectangular".
- And, most importantly,
decidability of the new criterion.


## Our results

- An elementary, and relatively short proof of Bulatov's dichotomy for \#CSP $(\Gamma)$, using a new criterion.
- A natural algorithm, with proven time complexity, for the class of problems in FP.
By-product: an improved algorithm for $\operatorname{CSP}(\Gamma)$ when $\Gamma$ is "strongly rectangular".
- And, most importantly, decidability of the new criterion.


## (1) Introduction

## (2) Rectangularity

(4) Counting
(6) Conclusion

## Rectangularity

A relation $R$ defined on $A \subseteq D^{r}$, for some $r$, is rectangular if


## Rectangularity

A relation $R$ defined on $A \subseteq D^{r}$, for some $r$, is rectangular if

$$
\left.\begin{array}{l}
(\mathbf{a}, \mathbf{c}) \\
(\mathbf{a}, \mathbf{d}) \\
(\mathbf{b}, \mathbf{c})
\end{array}\right\} \in R \Rightarrow(\mathbf{b}, \mathbf{d}) \in R
$$

## Rectangularity

A relation $R$ defined on $A \subseteq D^{r}$, for some $r$, is rectangular if

$$
\left.\begin{array}{l}
(\mathbf{a}, \mathbf{c}) \\
(\mathbf{a}, \mathbf{d}) \\
(\mathbf{b}, \mathbf{c})
\end{array}\right\} \in R \Rightarrow(\mathbf{b}, \mathbf{d}) \in R
$$

$$
(\mathbf{a}, \mathbf{c}) \quad(\mathbf{a}, \mathbf{d})
$$

(b, c)

## Rectangularity

A relation $R$ defined on $A \subseteq D^{r}$, for some $r$, is rectangular if

$$
\left.\begin{array}{l}
(\mathbf{a}, \mathbf{c}) \\
(\mathbf{a}, \mathbf{d}) \\
(\mathbf{b}, \mathbf{c})
\end{array}\right\} \in R \Rightarrow(\mathbf{b}, \mathbf{d}) \in R
$$

$$
\begin{array}{cc}
(\mathbf{a}, \mathbf{c}) & (\mathbf{a}, \mathbf{d}) \\
& \\
& \downarrow \\
(\mathbf{b}, \mathbf{c}) \rightarrow & (\mathbf{b}, \mathbf{d})
\end{array}
$$

## Strong rectangularity

A relation is pp－definable in $\Gamma$ if it uses only
$\exists$（existential quantifier），$\wedge$（logical＂and＂）and the relations in $\Gamma$ ．
This adds $\exists$ to the operations permissible in CSP（Г）．
「 is strongly rectangular if every relation pp－definable in「 is rectangular
It＇s not clear that this is decidable，but we have the well known

## Lemma

「 is strongly rectangular if，and only if，it has a Mal＇tsev polymorphism

In view of this，Bulatov \＆Dalmau（2006）used＂relations invariant under a Mal＇tsev operation＂for what we call＂strongly rectangular＂

This directly implies an algorithm for testing the strong rectangularity of $\Gamma$

## Strong rectangularity

A relation is pp-definable in 「 if it uses only
$\exists$ (existential quantifier), $\wedge$ (logical "and") and the relations in $\Gamma$.
This adds $\exists$ to the operations permissible in $\operatorname{CSP}(\Gamma)$.
> $\Gamma$ is strongly rectangular if every relation pp-definable in $\Gamma$ is rectangular It's not clear that this is decidable, but we have the well known

## Lemma

$\Gamma$ is strongly rectangular if, and only if, it has a Mal'tsev polymorphism

In view of this, Bulatov \& Dalmau (2006) used "relations invariant under a Mal'tsev operation" for what we call "strongly rectangular" This directly implies an algorithm for testing the strong rectangularity of $\Gamma$

## Strong rectangularity

A relation is pp－definable in 「 if it uses only
$\exists$（existential quantifier），$\wedge$（logical＂and＂）and the relations in $\Gamma$ ．
This adds $\exists$ to the operations permissible in $\operatorname{CSP}(\Gamma)$ ．
「 is strongly rectangular if every relation pp－definable in 「 is rectangular．
It＇s not clear that this is decidable，but we have the well known

## Lemma

「 is strongly rectangular if，and only if，it has a Mal＇tsev polymorphism

In view of this，Bulatov \＆Dalmau（2006）used＂relations invariant under a Mal＇tsev operation＂for what we call＂strongly rectangular＇

This directly implies an algorithm for testing the strong rectangularity of $\mid$

## Strong rectangularity

A relation is pp－definable in 「 if it uses only
$\exists$（existential quantifier），$\wedge$（logical＂and＂）and the relations in $\Gamma$ ．
This adds $\exists$ to the operations permissible in $\operatorname{CSP}(\Gamma)$ ．
「 is strongly rectangular if every relation pp－definable in 「 is rectangular．
It＇s not clear that this is decidable，but we have the well known


## Strong rectangularity

A relation is pp-definable in $\Gamma$ if it uses only
$\exists$ (existential quantifier), $\wedge$ (logical "and") and the relations in $\Gamma$.
This adds $\exists$ to the operations permissible in $\operatorname{CSP}(\Gamma)$.
「 is strongly rectangular if every relation pp-definable in $\Gamma$ is rectangular.
It's not clear that this is decidable, but we have the well known

## Lemma

「 is strongly rectangular if, and only if, it has a Mal'tsev polymorphism.

In view of this, Bulatov \& Dalmau (2006) used "relations invariant under a Mal'tsev operation" for what we call "strongly rectangular" This directly implies an algorithm for testing the strong rectangularity of $\Gamma$

## Strong rectangularity

A relation is pp-definable in $\Gamma$ if it uses only
$\exists$ (existential quantifier), $\wedge$ (logical "and") and the relations in $\Gamma$.
This adds $\exists$ to the operations permissible in $\operatorname{CSP}(\Gamma)$.
$\Gamma$ is strongly rectangular if every relation pp-definable in $\Gamma$ is rectangular.
It's not clear that this is decidable, but we have the well known

## Lemma

「 is strongly rectangular if, and only if, it has a Mal'tsev polymorphism.

In view of this, Bulatov \& Dalmau (2006) used "relations invariant under a Mal'tsev operation" for what we call "strongly rectangular".

This directly implies an algorithm for testing the strong rectangularity of $\Gamma$.

## Strong rectangularity

A relation is pp-definable in $\Gamma$ if it uses only
$\exists$ (existential quantifier), $\wedge$ (logical "and") and the relations in $\Gamma$.
This adds $\exists$ to the operations permissible in $\operatorname{CSP}(\Gamma)$.
「 is strongly rectangular if every relation pp-definable in $\Gamma$ is rectangular.
It's not clear that this is decidable, but we have the well known

## Lemma

「 is strongly rectangular if, and only if, it has a Mal'tsev polymorphism.

In view of this, Bulatov \& Dalmau (2006) used "relations invariant under a Mal'tsev operation" for what we call "strongly rectangular".

This directly implies an algorithm for testing the strong rectangularity of $\Gamma$.

## (1) Introduction

(2) Rectangularity
(3) Frames
(4) Counting
(5) Decidability

6 Conclusion

## Notation

We use the following notation. Let $[n]$ denote $\{1,2, \ldots, n\}$.

## If $J \subseteq[n]$, then $\mathrm{pr}_{\mathrm{J}} \mathrm{R}$ is the relation $R$ restricted to the positions in $J$.

Example: Suppose $D=\{0,1\}$ and $R$ is the ternary relation with 3-tuples:

then $\operatorname{pr}_{\{1,2\}} R$ is the binary relation with 2-tuples:

$(1,1)$
We may omit brackets in the subscrint, e. $g$. $\mathrm{pr}_{1,2} R$.

## Notation

We use the following notation. Let $[n]$ denote $\{1,2, \ldots, n\}$.
If $J \subseteq[n]$, then $\operatorname{pr}_{J} R$ is the relation $R$ restricted to the positions in $J$.
Example: Suppose $D=\{0,1\}$ and $R$ is the ternary relation with 3-tuples:
then $\operatorname{pr}_{\{1,2\}} R$ is the binary relation with 2-tuples:


We may omit brackets in the subscript, e.g. $\mathrm{pr}_{1,2} R$.

## Notation

We use the following notation. Let $[n]$ denote $\{1,2, \ldots, n\}$.
If $J \subseteq[n]$, then $\operatorname{pr}_{J} R$ is the relation $R$ restricted to the positions in $J$.
Example: Suppose $D=\{0,1\}$ and $R$ is the ternary relation with 3-tuples:

$$
\begin{aligned}
& (0,1,0) \\
& (0,1,1) \\
& (1,0,1) \\
& (1,1,0) \\
& (1,1,1)
\end{aligned}
$$

then $\operatorname{pr}_{\{1,2\}} R$ is the binary relation with 2-tuples:


We may omit brackets in the subscript, e.g. $\operatorname{pr}_{1,2} R$.

## Notation

We use the following notation. Let $[n]$ denote $\{1,2, \ldots, n\}$.
If $J \subseteq[n]$, then $\operatorname{pr}_{J} R$ is the relation $R$ restricted to the positions in $J$.
Example: Suppose $D=\{0,1\}$ and $R$ is the ternary relation with 3-tuples:

$$
\begin{aligned}
& (0,1,0) \\
& (0,1,1) \\
& (1,0,1) \\
& (1,1,0) \\
& (1,1,1)
\end{aligned}
$$

then $\operatorname{pr}_{\{1,2\}} R$ is the binary relation with 2-tuples:

$$
\begin{aligned}
& (0,1) \\
& (1,0) \\
& (1,1)
\end{aligned}
$$

## Notation

We use the following notation. Let $[n]$ denote $\{1,2, \ldots, n\}$.
If $J \subseteq[n]$, then $\mathrm{pr}_{J} R$ is the relation $R$ restricted to the positions in $J$.
Example: Suppose $D=\{0,1\}$ and $R$ is the ternary relation with 3-tuples:

$$
\begin{aligned}
& (0,1,0) \\
& (0,1,1) \\
& (1,0,1) \\
& (1,1,0) \\
& (1,1,1)
\end{aligned}
$$

then $\operatorname{pr}_{\{1,2\}} R$ is the binary relation with 2-tuples:

$$
\begin{aligned}
& (0,1) \\
& (1,0) \\
& (1,1)
\end{aligned}
$$

We may omit brackets in the subscript, e.g. $\mathrm{pr}_{1,2} R$.

## Frames

Frames are our concise representations for strongly rectangular relations. They are similar to, but generally somewhat smaller than, the "compact representations" introduced by Bulatov and Dalmau (2006).

A frame for a relation $R \subseteq D^{n}$ is any relation $F \subseteq R$ such that: If, for any $0 \leq i<n$, $R$ contains a pair of tuples ( $u_{1}$ then $F$ contains a pair of tuples $\left(v_{1}, \ldots, v_{i}, a, \ldots\right),\left(v_{1}, \ldots, v_{i}, b, \ldots\right)$
$R$ is a frame for itself so every relation has a frame However, to be useful they should be much smaller than $R$ The union of the set of witness pairs in Bulatov and Dalmau's algorithm is also a frame, but provably smaller frames exist.

## Frames

Frames are our concise representations for strongly rectangular relations. They are similar to, but generally somewhat smaller than, the "compact representations" introduced by Bulatov and Dalmau (2006).

A frame for a relation $R \subseteq D^{n}$ is any relation $F \subseteq R$ such that:

> If, for any $0 \leq i<n$,
> $R$ contains a pair of tuples ( $u_{1}$ then $F$ contains a pair of tuples ( $v_{1}$,
> $R$ is a frame for itself so every relation has a frame However, to be useful they should be much smaller than $R$ The union of the set of witness pairs in Bulatov and Dalmau's algorithm is also a frame, but provably smaller frames exist.

## Frames

Frames are our concise representations for strongly rectangular relations. They are similar to, but generally somewhat smaller than, the "compact representations" introduced by Bulatov and Dalmau (2006).

A frame for a relation $R \subseteq D^{n}$ is any relation $F \subseteq R$ such that:
If, for any $0 \leq i<n$,
$R$ contains a pair of tuples $\left(u_{1}, \ldots, u_{i}, a, \ldots\right),\left(u_{1}, \ldots, u_{i}, b, \ldots\right)$, then $F$ contains a pair of tuples $\left(v_{1}, \ldots, v_{i}, a, \ldots\right),\left(v_{1}, \ldots, v_{i}, b, \ldots\right)$.
$R$ is a frame for itself so every relation has a frame.
However, to be useful they should be much smaller than $R$
The union of the set of witness nairs in Bulatov and Darmau's algorithm is also a frame, but provably smaller frames exist.

## Frames

Frames are our concise representations for strongly rectangular relations. They are similar to, but generally somewhat smaller than, the "compact representations" introduced by Bulatov and Dalmau (2006).

A frame for a relation $R \subseteq D^{n}$ is any relation $F \subseteq R$ such that:
If, for any $0 \leq i<n$,
$R$ contains a pair of tuples $\left(u_{1}, \ldots, u_{i}, a, \ldots\right),\left(u_{1}, \ldots, u_{i}, b, \ldots\right)$, then $F$ contains a pair of tuples $\left(v_{1}, \ldots, v_{i}, a, \ldots\right),\left(v_{1}, \ldots, v_{i}, b, \ldots\right)$.
$R$ is a frame for itself so every relation has a frame.
However, to be useful they should be much smaller than $R$
The union of the set of witness pairs in Bulatov and Dalmau's algorithm is also a frame, but provably smaller frames exist.

## Frames

Frames are our concise representations for strongly rectangular relations. They are similar to, but generally somewhat smaller than, the "compact representations" introduced by Bulatov and Dalmau (2006).

A frame for a relation $R \subseteq D^{n}$ is any relation $F \subseteq R$ such that:
If, for any $0 \leq i<n$,
$R$ contains a pair of tuples $\left(u_{1}, \ldots, u_{i}, a, \ldots\right),\left(u_{1}, \ldots, u_{i}, b, \ldots\right)$,
then $F$ contains a pair of tuples $\left(v_{1}, \ldots, v_{i}, a, \ldots\right),\left(v_{1}, \ldots, v_{i}, b, \ldots\right)$.
$R$ is a frame for itself so every relation has a frame. However, to be useful they should be much smaller than $R$.

> The union of the set of witness pairs in Bulatov and Dalmau's algorithm is also a frame, but provably smaller frames exist.

## Frames

Frames are our concise representations for strongly rectangular relations. They are similar to, but generally somewhat smaller than, the "compact representations" introduced by Bulatov and Dalmau (2006).

A frame for a relation $R \subseteq D^{n}$ is any relation $F \subseteq R$ such that:
If, for any $0 \leq i<n$,
$R$ contains a pair of tuples $\left(u_{1}, \ldots, u_{i}, a, \ldots\right),\left(u_{1}, \ldots, u_{i}, b, \ldots\right)$,
then $F$ contains a pair of tuples $\left(v_{1}, \ldots, v_{i}, a, \ldots\right),\left(v_{1}, \ldots, v_{i}, b, \ldots\right)$.
$R$ is a frame for itself so every relation has a frame. However, to be useful they should be much smaller than $R$.

The union of the set of witness pairs in Bulatov and Dalmau's algorithm is also a frame, but provably smaller frames exist.

## Example

Here is a frame for the complete relation $\{0,1,2\}^{3}$.


It contains only 7 of the 27 3-tuples in the relation.
Similarly, there is a frame with less than $n|D| n$-tuples for any complete relation $D^{n}$ (which has $|D|^{n} n$-tuples).

The complete relation $D^{n}$ is trivially strongly rectangular.
For example, any function $\varphi: D^{3} \rightarrow D$ satisfying
$\varphi(a, b, b)=\varphi(b, b, a)=a$ is a Mal'tsev polymorphism of $D^{n}$.

## Example

Here is a frame for the complete relation $\{0,1,2\}^{3}$.

$$
\begin{array}{lll}
(0,0,0) & & \\
(1,0,0) & (0,1,0) & (0,0,1) \\
(2,0,0) & (0,2,0) & (0,0,2)
\end{array}
$$

It contains only 7 of the 27 3-tuples in the relation.
Similarly, there is a frame with less than $n|D| n$-tuples for any complete relation $D^{n}$ (which has $|D|^{n} n$-tuples).

The complete relation $D^{n}$ is trivially strongly rectangular.
For example, any function $\varphi: D^{3} \rightarrow D$ satisfying
$\varphi(a, b, b)=\varphi(b, b, a)=a$ is a Mal'tsev polymorphism of $D^{n}$.

## Example

Here is a frame for the complete relation $\{0,1,2\}^{3}$.

$$
\begin{array}{lll}
(0,0,0) & & \\
(1,0,0) & (0,1,0) & (0,0,1) \\
(2,0,0) & (0,2,0) & (0,0,2)
\end{array}
$$

It contains only 7 of the 27 3-tuples in the relation.
Similarly, there is a frame with less than $n|D|$ n-tuples for any complete relation $D^{n}$ (which has $|D|^{n} n$-tuples).

The complete relation $D^{n}$ is trivially strongly rectangular
For example, any function $\varphi: D^{3} \rightarrow D$ satisfying
$\varphi(a, b, b)=\varphi(b, b, a)=a$ is a Mal'tsev polymorphism of $D^{n}$.

## Example

Here is a frame for the complete relation $\{0,1,2\}^{3}$.

$$
\begin{array}{lll}
(0,0,0) & & \\
(1,0,0) & (0,1,0) & (0,0,1) \\
(2,0,0) & (0,2,0) & (0,0,2)
\end{array}
$$

It contains only 7 of the 27 3-tuples in the relation.
Similarly, there is a frame with less than $n|D| n$-tuples for any complete relation $D^{n}$ (which has $|D|^{n} n$-tuples).

The complete relation $D^{n}$ is trivially strongly rectangular.
For example, any function $\varphi: D^{3} \rightarrow D$ satisfying
$\varphi(a, b, b)=\varphi(b, b, a)=a$ is a Mal'tsev polymorphism of $D^{n}$.

## Example

Here is a frame for the complete relation $\{0,1,2\}^{3}$.

$$
\begin{array}{lll}
(0,0,0) & & \\
(1,0,0) & (0,1,0) & (0,0,1) \\
(2,0,0) & (0,2,0) & (0,0,2)
\end{array}
$$

It contains only 7 of the 27 3-tuples in the relation.
Similarly, there is a frame with less than $n|D| n$-tuples for any complete relation $D^{n}$ (which has $|D|^{n} n$-tuples).

The complete relation $D^{n}$ is trivially strongly rectangular.


## Example

Here is a frame for the complete relation $\{0,1,2\}^{3}$.

$$
\begin{array}{lll}
(0,0,0) & & \\
(1,0,0) & (0,1,0) & (0,0,1) \\
(2,0,0) & (0,2,0) & (0,0,2)
\end{array}
$$

It contains only 7 of the 27 3-tuples in the relation.
Similarly, there is a frame with less than $n|D| n$-tuples for any complete relation $D^{n}$ (which has $|D|^{n} n$-tuples).

The complete relation $D^{n}$ is trivially strongly rectangular.
For example, any function $\varphi: D^{3} \rightarrow D$ satisfying $\varphi(a, b, b)=\varphi(b, b, a)=a$ is a Mal'tsev polymorphism of $D^{n}$.

## Properties of frames

If $F$ is a frame for a strongly rectangular $n$-ary relation $R$, with Mal'tsev polymorphism $\varphi$ :

- $F=\emptyset$ if, and only if, $R=\emptyset$.
- We can recover $R$ from $F$ and $\varphi$, by taking the closure of $F$ under $\varphi$. However, this will take exponential time if $R$ has exponential size.
- In time $\mathcal{O}\left(n^{2}|F|^{2}\right)$, we can construct a small frame for $R$, which means a frame with at most $n|D| n$-tuples, if one exists.
- If $F$ is a small frame for $R$, and $\mathrm{a} \in D^{n}$, we can test whether or not $\mathbf{a} \in R$, in time $\mathcal{O}\left(n^{2}\right)$.


## Properties of frames

If $F$ is a frame for a strongly rectangular $n$-ary relation $R$, with Mal'tsev polymorphism $\varphi$ :

- $F=\emptyset$ if, and only if, $R=\emptyset$.
- We can recover $R$ from $F$ and $\varphi$, by taking the closure of $F$ under $\varphi$ However, this will take exponential time if $R$ has exponential size.
- In time $\left.\mathcal{O}^{( } n^{2 \mid} \mid \Gamma^{2}\right)$, we can construct a small frame for $R$, which means a frame with at most n $D$ | $n$-tuples, if one exists.
- If $F$ is a small frame for $R$, and $\mathbf{a} \in D^{n}$, we can test whether or not $\mathbf{a} \in R$, in time $\mathcal{O}\left(n^{2}\right)$.


## Properties of frames

If $F$ is a frame for a strongly rectangular $n$-ary relation $R$, with Mal'tsev polymorphism $\varphi$ :

- $F=\emptyset$ if, and only if, $R=\emptyset$.
- We can recover $R$ from $F$ and $\varphi$, by taking the closure of $F$ under $\varphi$. However, this will take exponential time if $R$ has exponential size.
- In time $\mathcal{O}\left(n^{2}|F|^{2}\right)$, we can construct a small frame for $R$, which means a frame with at most $n|D| n$-tuples, if one exists.
- If $F$ is a small frame for $R$, and $a \in D^{n}$, we can test whether or not $\mathbf{a} \in R$, in time $\mathcal{O}\left(n^{2}\right)$


## Properties of frames

If $F$ is a frame for a strongly rectangular $n$-ary relation $R$, with Mal'tsev polymorphism $\varphi$ :

- $F=\emptyset$ if, and only if, $R=\emptyset$.
- We can recover $R$ from $F$ and $\varphi$, by taking the closure of $F$ under $\varphi$. However, this will take exponential time if $R$ has exponential size.
- In time $\mathcal{O}\left(n^{2}|F|^{2}\right)$, we can construct a small frame for $R$, which means a frame with at most $n|D| n$-tuples, if one exists.
- If $F$ is a small frame for $R$, and $\mathrm{a} \in D^{n}$, we can test whether or not $\mathbf{a} \in R$, in time $\mathcal{O}\left(n^{2}\right)$


## Properties of frames

If $F$ is a frame for a strongly rectangular $n$-ary relation $R$, with Mal'tsev polymorphism $\varphi$ :

- $F=\emptyset$ if, and only if, $R=\emptyset$.
- We can recover $R$ from $F$ and $\varphi$, by taking the closure of $F$ under $\varphi$. However, this will take exponential time if $R$ has exponential size.
- In time $\mathcal{O}\left(n^{2}|F|^{2}\right)$, we can construct a small frame for $R$, which means a frame with at most $n|D| n$-tuples, if one exists.
- If $F$ is a small frame for $R$, and $\mathbf{a} \in D^{n}$, we can test whether or not $\mathbf{a} \in R$, in time $\mathcal{O}\left(n^{2}\right)$.


## Frames for strongly rectangular constraints

Let $\Gamma$ be a strongly rectangular constraint language over domain $D$.
For an instance / of \#CSP(Г) with $n$ variables, the set of satisfying assignments can be considered to be an $n$-ary relation $\Phi \subseteq D^{n}$.

Then $\Phi$ is pp-definable in $\Gamma$, so is also strongly rectangular (and has the same Mal'tsev polymorphism).

We wish to construct, in time polynomial in $n$, a small frame $F$ for $\Phi$, i.e. one with at most $n|D| n$-tuples.

## Frames for strongly rectangular constraints

Let $\Gamma$ be a strongly rectangular constraint language over domain $D$.
For an instance $/$ of $\# \operatorname{CSP}(\Gamma)$ with $n$ variables, the set of satisfying assignments can be considered to be an $n$-ary relation $\Phi \subseteq D^{n}$.

> Then $\Phi$ is pp-definable in $\Gamma$, so is also strongly rectangular (and has the same Mal'tsev polymorphism)

> We wish to construct, in time polynomial in $n$, a small frame $F$ for $\phi$, i.e. one with at most $n|D| n$-tuples.

## Frames for strongly rectangular constraints

Let $\Gamma$ be a strongly rectangular constraint language over domain $D$.
For an instance $/$ of $\# \operatorname{CSP}(\Gamma)$ with $n$ variables, the set of satisfying assignments can be considered to be an $n$-ary relation $\Phi \subseteq D^{n}$.

Then $\Phi$ is pp-definable in $\Gamma$, so is also strongly rectangular (and has the same Mal'tsev polymorphism).

We wish to construct, in time polynomial in $n$, a small frame $F$ for $\Phi$, i.e. one with at most $n|D| n$-tuples.

## Frames for strongly rectangular constraints

Let $\Gamma$ be a strongly rectangular constraint language over domain $D$.
For an instance $/$ of $\# \operatorname{CSP}(\Gamma)$ with $n$ variables, the set of satisfying assignments can be considered to be an $n$-ary relation $\Phi \subseteq D^{n}$.

Then $\Phi$ is pp-definable in $\Gamma$, so is also strongly rectangular (and has the same Mal'tsev polymorphism).

We wish to construct, in time polynomial in $n$, a small frame $F$ for $\Phi$, i.e. one with at most $n|D| n$-tuples.

## Bulatov \& Dalmau's idea

Let instance I have constraints $C_{1}, \ldots, C_{m}$.
For $0 \leq j \leq m$, let $l_{j}$ be the sub-instance of I with all variables but only constraints $C_{1}, \ldots, C_{j}$, determining relation $\Phi_{j}$

So, $I_{m}=I$ and $I_{0}$ has no constraints.
Therefore $\Phi_{0}=D^{n}$, and has a small frame (as we've seen).
We must construct, efficiently, a frame for $\Phi_{j}$, given $C_{j}$ and a frame $F_{j-1}$ for $\Phi_{j-1}$.

## Bulatov \& Dalmau's idea

Let instance I have constraints $C_{1}, \ldots, C_{m}$.
For $0 \leq j \leq m$, let $l_{j}$ be the sub-instance of $I$ with all variables but only constraints $C_{1}, \ldots, C_{j}$, determining relation $\Phi_{j}$.

So, $I_{m}=I$ and $I_{0}$ has no constraints.
Therefore $\Phi_{0}=D^{n}$, and has a small frame (as we've seen).
We must construct, efficiently, a frame for $\Phi_{j}$, given $C_{j}$ and a frame $F_{j-1}$ for $\Phi_{j-1}$

## Bulatov \& Dalmau's idea

Let instance I have constraints $C_{1}, \ldots, C_{m}$.
For $0 \leq j \leq m$, let $l_{j}$ be the sub-instance of $I$ with all variables but only constraints $C_{1}, \ldots, C_{j}$, determining relation $\Phi_{j}$.

So, $I_{m}=I$ and $I_{0}$ has no constraints.
Therefore $\Phi_{0}=D^{n}$, and has a small frame (as we've seen).
We must construct, efficiently, a frame for $\Phi_{j}$, given $C_{j}$ and a frame $F_{j-1}$ for $\Phi_{j-1}$

## Bulatov \& Dalmau's idea

Let instance I have constraints $C_{1}, \ldots, C_{m}$.
For $0 \leq j \leq m$, let $l_{j}$ be the sub-instance of $I$ with all variables but only constraints $C_{1}, \ldots, C_{j}$, determining relation $\Phi_{j}$.

So, $I_{m}=I$ and $I_{0}$ has no constraints.
Therefore $\Phi_{0}=D^{n}$, and has a small frame (as we've seen).
We must construct, efficiently, a frame for $\Phi_{j}$, given $C_{j}$ and
a frame $F_{j-1}$ for $\Phi_{j-1}$

## Bulatov \& Dalmau's idea

Let instance I have constraints $C_{1}, \ldots, C_{m}$.
For $0 \leq j \leq m$, let $l_{j}$ be the sub-instance of $I$ with all variables but only constraints $C_{1}, \ldots, C_{j}$, determining relation $\Phi_{j}$.

So, $I_{m}=I$ and $I_{0}$ has no constraints.
Therefore $\Phi_{0}=D^{n}$, and has a small frame (as we've seen).
We must construct, efficiently, a frame for $\Phi_{j}$, given $C_{j}$ and a frame $F_{j-1}$ for $\Phi_{j-1}$.

## Inductive step

Let $F$ be a frame for the relation $\Psi=\Phi_{j-1}$ determined by the constraints $C_{1}, C_{2}, \ldots, C_{j-1}$ added so far.

Assume for simplicity that the next constraint $C_{j}$ is $C=R\left(x_{1}, \ldots, x_{k}\right)$.

- For each $i>k$, choose a set $T_{i} \subseteq F$ from which $\operatorname{pr}_{\{1, \ldots, k, i\}} \Psi$ can be reconstructed.
- Remove from each $T_{i}$ anything that is inconsistent with $C$.
- Use the resulting sets sequentially to construct "partial frames" for $\operatorname{pr}_{\{1, \ldots, k+1\}}(\Psi \wedge C), \ldots \ldots, \operatorname{pr}_{\{1, \ldots, n\}}(\Psi \wedge C)=\Phi_{j} \wedge C$.

The total time to construct the frame is $\mathcal{O}\left(n^{5}\right)$, if $n$ is the number of variables in $\Phi_{n}=\Phi$, provided $\Gamma$ has constant size.

## Inductive step

Let $F$ be a frame for the relation $\psi=\Phi_{j-1}$ determined by the constraints $C_{1}, C_{2}, \ldots, C_{j-1}$ added so far.

Assume for simplicity that the next constraint $C_{j}$ is $C=R\left(x_{1}, \ldots, x_{k}\right)$.

- For each $i>k$, choose a set $T_{i} \subseteq F$ from which $\operatorname{pr}_{\{1, \ldots, k, i\}}{ }^{\Psi}$ can be reconstructed.
- Remove from each $T_{i}$ anything that is inconsistent with $C$.
- Use the resulting sets sequentially to construct "partial frames" for $\operatorname{pr}_{\{1, \ldots, k+1\}}(\Psi \wedge C)$


## The total time to construct the frame is $\mathcal{O}\left(n^{5}\right)$, if $n$ is the number of variables in $\Phi_{n}=\Phi$, provided $\Gamma$ has constant size.

## Inductive step

Let $F$ be a frame for the relation $\psi=\Phi_{j-1}$ determined by the constraints $C_{1}, C_{2}, \ldots, C_{j-1}$ added so far.

Assume for simplicity that the next constraint $C_{j}$ is $C=R\left(x_{1}, \ldots, x_{k}\right)$.

- For each $i>k$, choose a set $T_{i} \subseteq F$ from which $\operatorname{pr}_{\{1, \ldots, k, i\}} \Psi$ can be reconstructed.
- Remove from each $T_{i}$ anything that is inconsistent with $C$.
- Use the resulting sets sequentially to construct "partial frames" for $\operatorname{pr}_{\{1, \ldots, k+1\}}(\Psi \wedge C)$


## The total time to construct the frame is $\mathcal{O}\left(n^{5}\right)$, if $n$ is the number of variables in $\Phi_{n}=\Phi$, provided $\Gamma$ has constant size.

## Inductive step

Let $F$ be a frame for the relation $\psi=\Phi_{j-1}$ determined by the constraints $C_{1}, C_{2}, \ldots, C_{j-1}$ added so far.

Assume for simplicity that the next constraint $C_{j}$ is $C=R\left(x_{1}, \ldots, x_{k}\right)$.

- For each $i>k$, choose a set $T_{i} \subseteq F$ from which $\operatorname{pr}_{\{1, \ldots, k, i\}} \Psi$ can be reconstructed.
- Remove from each $T_{i}$ anything that is inconsistent with $C$.
- Use the resulting sets sequentially to construct "partial frames" for $\operatorname{pr}_{\{1, \ldots, k+1\}}(\Psi \wedge C)$


## The total time to construct the frame is $\mathcal{O}\left(n^{5}\right)$, if $n$ is the number of variables in $\Phi_{n}=\Phi$, provided $\Gamma$ has constant size.

## Inductive step

Let $F$ be a frame for the relation $\psi=\Phi_{j-1}$ determined by the constraints $C_{1}, C_{2}, \ldots, C_{j-1}$ added so far.

Assume for simplicity that the next constraint $C_{j}$ is $C=R\left(x_{1}, \ldots, x_{k}\right)$.

- For each $i>k$, choose a set $T_{i} \subseteq F$ from which $\operatorname{pr}_{\{1, \ldots, k, i\}} \Psi$ can be reconstructed.
- Remove from each $T_{i}$ anything that is inconsistent with $C$.
- Use the resulting sets sequentially to construct "partial frames" for $\operatorname{pr}_{\{1, \ldots, k+1\}}(\Psi \wedge C), \ldots \ldots, \operatorname{pr}_{\{1, \ldots, n\}}(\Psi \wedge C)=\Phi_{j} \wedge C$.


## Inductive step

Let $F$ be a frame for the relation $\Psi=\Phi_{j-1}$ determined by the constraints $C_{1}, C_{2}, \ldots, C_{j-1}$ added so far.

Assume for simplicity that the next constraint $C_{j}$ is $C=R\left(x_{1}, \ldots, x_{k}\right)$.

- For each $i>k$, choose a set $T_{i} \subseteq F$ from which $\operatorname{pr}_{\{1, \ldots, k, i\}} \Psi$ can be reconstructed.
- Remove from each $T_{i}$ anything that is inconsistent with $C$.
- Use the resulting sets sequentially to construct "partial frames" for $\operatorname{pr}_{\{1, \ldots, k+1\}}(\Psi \wedge C), \ldots \ldots, \operatorname{pr}_{\{1, \ldots, n\}}(\Psi \wedge C)=\Phi_{j} \wedge C$.

The total time to construct the frame is $\mathcal{O}\left(n^{5}\right)$, if $n$ is the number of variables in $\Phi_{n}=\Phi$, provided $\Gamma$ has constant size.

## (1) Introduction

(2) Rectangularity
(3) Frames
(4) Counting
(5) Decidability

6 Conclusion

## Block matrices

Let $A=\left(a_{i j}\right)$ be a $k \times \ell$ non-negative real-valued matrix.
The matrix $A$ has an underlying relation
$R_{A}=\left\{(i, j): a_{i j}>0\right\} \subseteq[k] \times[\ell]$.
A block of $A$ is a set of rows $K \subset[k]$, and a set of columns $L \subset[\theta]$, such that $a_{i j}=0$ if $i \in K, j \notin L$, or $i \notin K, j \in L$.

Example: The $4 \times 4$ matrix

has the three blocks shown, and underlying relation

$$
R_{A}=\{(1,3),(1,4),(2,3),(2,4),(3,2),(4,1)\}
$$

## Block matrices

Let $A=\left(a_{i j}\right)$ be a $k \times \ell$ non-negative real-valued matrix.
The matrix $A$ has an underlying relation
$R_{A}=\left\{(i, j): a_{i j}>0\right\} \subseteq[k] \times[\ell]$.
A block of $A$ is a set of rows $K \subset[k]$, and a set of columns $L \subset[\ell]$, such that $a_{i j}=0$ if $i \in K, j \notin L$, or $i \notin K, j \in L$

Example: The $4 \times 4$ matrix

has the three blocks shown, and underlying relation

$$
R_{A}=\{(1,3),(1,4),(2,3),(2,4),(3,2),(4,1)\}
$$

## Block matrices

Let $A=\left(a_{i j}\right)$ be a $k \times \ell$ non-negative real-valued matrix.
The matrix $A$ has an underlying relation
$R_{A}=\left\{(i, j): a_{i j}>0\right\} \subseteq[k] \times[\ell]$.
A block of $A$ is a set of rows $K \subset[k]$, and a set of columns $L \subset[\ell]$, such that $a_{i j}=0$ if $i \in K, j \notin L$, or $i \notin K, j \in L$.

Example: The $4 \times 4$ matrix

has the three blocks shown, and underlying relation
$\square$

## Block matrices

Let $A=\left(a_{i j}\right)$ be a $k \times \ell$ non-negative real-valued matrix.
The matrix $A$ has an underlying relation
$R_{A}=\left\{(i, j): a_{i j}>0\right\} \subseteq[k] \times[\ell]$.
A block of $A$ is a set of rows $K \subset[k]$, and a set of columns $L \subset[\ell]$, such that $a_{i j}=0$ if $i \in K, j \notin L$, or $i \notin K, j \in L$.

Example: The $4 \times 4$ matrix

$$
A=\left[\begin{array}{llll}
0 & 0 & 1 & 2 \\
0 & 0 & 2 & 1 \\
0 & 1 & 0 & 0 \\
2 & 0 & 0 & 0
\end{array}\right]
$$

has the three blocks shown, and underlying relation

## Block matrices

Let $A=\left(a_{i j}\right)$ be a $k \times \ell$ non-negative real-valued matrix.
The matrix $A$ has an underlying relation
$R_{A}=\left\{(i, j): a_{i j}>0\right\} \subseteq[k] \times[\ell]$.
A block of $A$ is a set of rows $K \subset[k]$, and a set of columns $L \subset[\ell]$, such that $a_{i j}=0$ if $i \in K, j \notin L$, or $i \notin K, j \in L$.

Example: The $4 \times 4$ matrix

$$
A=\left[\begin{array}{cc|cc}
0 & 0 & 1 & 2 \\
0 & 0 & 2 & 1 \\
0 & 1 & 0 & 0 \\
\hline 2 & 0 & 0 & 0
\end{array}\right]
$$

has the three blocks shown, and underlying relation

## Block matrices

Let $A=\left(a_{i j}\right)$ be a $k \times \ell$ non-negative real-valued matrix.
The matrix $A$ has an underlying relation
$R_{A}=\left\{(i, j): a_{i j}>0\right\} \subseteq[k] \times[\ell]$.
A block of $A$ is a set of rows $K \subset[k]$, and a set of columns $L \subset[\ell]$, such that $a_{i j}=0$ if $i \in K, j \notin L$, or $i \notin K, j \in L$.

Example: The $4 \times 4$ matrix

$$
A=\left[\begin{array}{cc|cc}
0 & 0 & 1 & 2 \\
0 & 0 & 2 & 1 \\
0 & 1 & 0 & 0 \\
\hline 2 & 0 & 0 & 0
\end{array}\right]
$$

has the three blocks shown, and underlying relation

$$
R_{A}=\{(1,3),(1,4),(2,3),(2,4),(3,2),(4,1)\}
$$

## Rank-one block matrix matrices

## Lemma

Suppose A decomposes into blocks of rank 1. Then

- $R_{A}$ is a rectangular relation.
- we can recover $A$ from $R_{A}$ and the row and column sums of $A$.

A decomposition of $A$ into blocks of rank 1 corresponds to the existence of a row function $\alpha:[k] \rightarrow \mathbb{R}$ and a column function $\beta:[\ell] \rightarrow \mathbb{R}$ such that $a_{i j}=\alpha(i) \beta(j)$ for $(i, j) \in R_{A}$ Example: The $4 \times 4$ matrix

is a rank-one block matrix.

## Rank-one block matrix matrices

## Lemma

Suppose A decomposes into blocks of rank 1. Then

- $R_{A}$ is a rectangular relation.
- we can recover $A$ from $R_{A}$ and the row and column sums of $A$.

A decomposition of $A$ into blocks of rank 1 corresponds to the existence of a row function $\alpha:[k] \rightarrow \mathbb{R}$ and a column function $\beta:[\ell] \rightarrow \mathbb{R}$ such that $a_{i j}=\alpha(i) \beta(j)$ for $(i, j) \in R_{A}$.

Example: The $4 \times 4$ matrix

with


## Rank-one block matrix matrices

## Lemma

Suppose A decomposes into blocks of rank 1. Then

- $R_{A}$ is a rectangular relation.
- we can recover $A$ from $R_{A}$ and the row and column sums of $A$.

A decomposition of $A$ into blocks of rank 1 corresponds to the existence of a row function $\alpha:[k] \rightarrow \mathbb{R}$ and a column function $\beta:[\ell] \rightarrow \mathbb{R}$ such that $a_{i j}=\alpha(i) \beta(j)$ for $(i, j) \in R_{A}$.
Example: The $4 \times 4$ matrix

$$
A=\left[\begin{array}{llll}
0 & 0 & 1 & 1 \\
0 & 0 & 2 & 2 \\
0 & 1 & 0 & 0 \\
2 & 0 & 0 & 0
\end{array}\right]
$$


is a rank-one block matrix

## Rank-one block matrix matrices

## Lemma

Suppose A decomposes into blocks of rank 1. Then

- $R_{A}$ is a rectangular relation.
- we can recover $A$ from $R_{A}$ and the row and column sums of $A$.

A decomposition of $A$ into blocks of rank 1 corresponds to the existence of a row function $\alpha:[k] \rightarrow \mathbb{R}$ and a column function $\beta:[\ell] \rightarrow \mathbb{R}$ such that $a_{i j}=\alpha(i) \beta(j)$ for $(i, j) \in R_{A}$.
Example: The $4 \times 4$ matrix

$$
A=\left[\begin{array}{cc|cc}
0 & 0 & 1 & 1 \\
0 & 0 & 2 & 2 \\
0 & 1 & 0 & 0 \\
\hline 2 & 0 & 0 & 0
\end{array}\right], \quad \text { with } \quad \alpha=\left[\begin{array}{c}
1 \\
2 \\
\hline \frac{1}{1}
\end{array}\right], \quad \beta=\left[\begin{array}{ll}
2|1| 1 & 1
\end{array}\right]
$$

is a rank-one block matrix.

## Balance matrices

For a ternary relation $R$, define its balance matrix to be

$$
M(x, y)=|\{z:(x, y, z) \in R\}|
$$

$R$ is balanced if $M$ decomposes into blocks of rank 1
(i.e. if $M(x, y)=\alpha(x) \beta(y)$ for $\left.(x, y) \in \operatorname{pr}_{1,2} R\right)$.

Example: The ternary relation on $\{1,2,3,4\}$, with tuples

$$
\begin{aligned}
& \{(1,3,1),(1,4,1),(1,4,3),(2,3,2),(2,3,4), \\
& (2,4,2),(3,2,2),(4,1,2),(4,1,3)\}
\end{aligned}
$$

has balance matrix


## Balance matrices

For a ternary relation $R$, define its balance matrix to be

$$
M(x, y)=|\{z:(x, y, z) \in R\}|
$$

$R$ is balanced if $M$ decomposes into blocks of rank 1
(i.e. if $M(x, y)=\alpha(x) \beta(y)$ for $\left.(x, y) \in \operatorname{pr}_{1,2} R\right)$.

Example: The ternary relation on $\{1,2,3,4\}$, with tuples

$$
\{(1,3,1),(1,4,1),(1,4,3),(2,3,2),(2,3,4),
$$

$$
(2,4,2),(3,2,2),(4,1,2),(4,1,3)\}
$$

has balance matrix


## Balance matrices

For a ternary relation $R$, define its balance matrix to be

$$
M(x, y)=|\{z:(x, y, z) \in R\}|
$$

$R$ is balanced if $M$ decomposes into blocks of rank 1
(i.e. if $M(x, y)=\alpha(x) \beta(y)$ for $\left.(x, y) \in \operatorname{pr}_{1,2} R\right)$.

Example: The ternary relation on $\{1,2,3,4\}$, with tuples

$$
\begin{aligned}
& \{(1,3,1),(1,4,1),(1,4,3),(2,3,2),(2,3,4) \\
& (2,4,2),(3,2,2),(4,1,2),(4,1,3)\}
\end{aligned}
$$

## has balance matrix

## Balance matrices

For a ternary relation $R$, define its balance matrix to be

$$
M(x, y)=|\{z:(x, y, z) \in R\}|
$$

$R$ is balanced if $M$ decomposes into blocks of rank 1
(i.e. if $M(x, y)=\alpha(x) \beta(y)$ for $\left.(x, y) \in \operatorname{pr}_{1,2} R\right)$.

Example: The ternary relation on $\{1,2,3,4\}$, with tuples

$$
\begin{aligned}
& \{(1,3,1),(1,4,1),(1,4,3),(2,3,2),(2,3,4) \\
& (2,4,2),(3,2,2),(4,1,2),(4,1,3)\}
\end{aligned}
$$

has balance matrix

$$
M=\left[\begin{array}{llll}
0 & 0 & 1 & 2 \\
0 & 0 & 2 & 1 \\
0 & 1 & 0 & 0 \\
2 & 0 & 0 & 0
\end{array}\right]
$$

## Balance matrices

For a ternary relation $R$, define its balance matrix to be

$$
M(x, y)=|\{z:(x, y, z) \in R\}|
$$

$R$ is balanced if $M$ decomposes into blocks of rank 1
(i.e. if $M(x, y)=\alpha(x) \beta(y)$ for $\left.(x, y) \in \operatorname{pr}_{1,2} R\right)$.

Example: The ternary relation on $\{1,2,3,4\}$, with tuples

$$
\begin{aligned}
& \{(1,3,1),(1,4,1),(1,4,3),(2,3,2),(2,3,4) \\
& (2,4,2),(3,2,2),(4,1,2),(4,1,3)\}
\end{aligned}
$$

has balance matrix

$$
M=\left[\begin{array}{ll|ll}
0 & 0 & 1 & 2 \\
0 & 0 & 2 & 1 \\
0 & 1 & 0 & 0 \\
\hline 2 & 0 & 0 & 0
\end{array}\right]
$$

## Balance matrices

For a ternary relation $R$, define its balance matrix to be

$$
M(x, y)=|\{z:(x, y, z) \in R\}|
$$

$R$ is balanced if $M$ decomposes into blocks of rank 1
(i.e. if $M(x, y)=\alpha(x) \beta(y)$ for $\left.(x, y) \in \operatorname{pr}_{1,2} R\right)$.

Example: The ternary relation on $\{1,2,3,4\}$, with tuples

$$
\begin{aligned}
& \{(1,3,1),(1,4,1),(1,4,3),(2,3,2),(2,3,4) \\
& (2,4,2),(3,2,2),(4,1,2),(4,1,3)\}
\end{aligned}
$$

has balance matrix

$$
M=\left[\begin{array}{ll|ll}
0 & 0 & 1 & 2 \\
0 & 0 & 2 & 1 \\
0 & 1 & 0 & 0 \\
\hline 2 & 0 & 0 & 0
\end{array}\right]
$$

which is not a rank-one block matrix.

## Strong balance

A relation of arity $r \geq 3$ can be considered as a collection of ternary relations over $D^{i} \times D^{j} \times D^{k}(i, j, k \geq 1, i+j+k=r)$.

Example: a relation $R \subseteq D^{4}$ can be considered as a ternary relation over $D^{2} \times D \times D$, in 4 ! ways, by permuting the 4 positions in $R$.

「 is strongly balanced if every ternary relation derived from every relation pp-definable from 「 is balanced.

Strong balance clearly implies strong rectangularity.

## Strong balance

A relation of arity $r \geq 3$ can be considered as a collection of ternary relations over $D^{i} \times D^{j} \times D^{k}(i, j, k \geq 1, i+j+k=r)$.

Example: a relation $R \subseteq D^{4}$ can be considered as a ternary relation over $D^{2} \times D \times D$, in 4 ! ways, by permuting the 4 positions in $R$.

> 「 is strongly balanced if every ternary relation derived from every relation pp-definable from 「 is balanced.

Strong balance clearly implies strong rectangularity.

## Strong balance

A relation of arity $r \geq 3$ can be considered as a collection of ternary relations over $D^{i} \times D^{j} \times D^{k}(i, j, k \geq 1, i+j+k=r)$.

Example: a relation $R \subseteq D^{4}$ can be considered as a ternary relation over $D^{2} \times D \times D$, in 4! ways, by permuting the 4 positions in $R$.
$\Gamma$ is strongly balanced if every ternary relation derived from every relation pp-definable from $\Gamma$ is balanced.

Strong balance clearly implies strong rectangularity.

## Strong balance

A relation of arity $r \geq 3$ can be considered as a collection of ternary relations over $D^{i} \times D^{j} \times D^{k}(i, j, k \geq 1, i+j+k=r)$.

Example: a relation $R \subseteq D^{4}$ can be considered as a ternary relation over $D^{2} \times D \times D$, in 4 ! ways, by permuting the 4 positions in $R$.

「 is strongly balanced if every ternary relation derived from every relation pp-definable from $\Gamma$ is balanced.

Strong balance clearly implies strong rectangularity.

## \#P-completeness

We use the following theorem, which strengthens a theorem of Bulatov \& Dalmau (2007) concerning strong rectangularity.

## Theorem

If $\Gamma$ is not strongly balanced, then \#СSP(Г) is \#P-complete.
$\square$
Proof.
Via weighted \#CSP(Г), using a result of Bulatov \& Grohe (2005), for partition functions of graph homomorphisms.

From this, failure of the rank-one block condition for the balance matrix of any ternary relation pp-definable on 「 implies \#P-completeness.

## \#P-completeness

We use the following theorem, which strengthens a theorem of Bulatov \& Dalmau (2007) concerning strong rectangularity.

## Theorem

If $\Gamma$ is not strongly balanced, then $\# \mathrm{CSP}(\Gamma)$ is \# P -complete.
$\square$
Proof.
Via weighted \#CSP(Г), using a result of Bulatov \& Grohe (2005), for partition functions of graph homomorphisms.

From this, failure of the rank-one block condition for the balance matrix of any ternary relation pp-definable on 「 implies \#P-completeness.

## \#P-completeness

We use the following theorem, which strengthens a theorem of Bulatov \& Dalmau (2007) concerning strong rectangularity.

## Theorem

If $\Gamma$ is not strongly balanced, then \#CSP $(\Gamma)$ is \#P-complete.

## Proof.

Via weighted \#CSP(Г), using a result of Bulatov \& Grohe (2005), for partition functions of graph homomorphisms.
From this, failure of the rank-one block condition for the balance matrix of any ternary relation pp-definable on 「 implies \#P-completeness.

## The counting algorithm

Suppose now that $\Gamma$ is strongly balanced, and we have a given instance.
First, we compute a small frame $F$ for set of assignments $\Phi$, using the algorithm outlined above.

Assume there are at least two variables, so $\Phi$ is at least binary.
For $1 \leq i<j \leq n$, let

$$
N_{i, j}(a)=\mid\left\{\left(u_{1}, \ldots, u_{i}\right):\left(u_{1}, \ldots, u_{n}\right) \in \Phi \text { and } u_{j}=a\right\} \mid
$$

## Then the total number of satisfying assignments, $N=|\Phi|$, is



## The counting algorithm

Suppose now that $\Gamma$ is strongly balanced, and we have a given instance.
First, we compute a small frame $F$ for set of assignments $\Phi$, using the algorithm outlined above.

Assume there are at least two variables, so $\Phi$ is at least binary.
For $1 \leq i<j \leq n$, let

$$
N_{i, j}(a)=\mid\left\{\left(u_{1}, \ldots, u_{i}\right):\left(u_{1}, \ldots, u_{n}\right) \in \Phi \text { and } u_{j}=a\right\}
$$

Then the total number of satisfying assignments, $N=|\Phi|$, is


## The counting algorithm

Suppose now that $\Gamma$ is strongly balanced, and we have a given instance.
First, we compute a small frame $F$ for set of assignments $\Phi$, using the algorithm outlined above.

Assume there are at least two variables, so $\Phi$ is at least binary.
For $1 \leq i<j \leq n$, let

$$
N_{i, j}(a)=\mid\left\{\left(u_{1}, \ldots, u_{i}\right):\left(u_{1}, \ldots, u_{n}\right) \in \Phi \text { and } u_{j}=a\right\}
$$

Then the total number of satisfying assignments, $N=|\Phi|$, is


## The counting algorithm

Suppose now that $\Gamma$ is strongly balanced, and we have a given instance.
First, we compute a small frame $F$ for set of assignments $\Phi$, using the algorithm outlined above.

Assume there are at least two variables, so $\Phi$ is at least binary.
For $1 \leq i<j \leq n$, let

$$
N_{i, j}(a)=\mid\left\{\left(u_{1}, \ldots, u_{i}\right):\left(u_{1}, \ldots, u_{n}\right) \in \Phi \text { and } u_{j}=a\right\} \mid .
$$

Then the total number of satisfying assignments, $N=|\Phi|$, is


## The counting algorithm

Suppose now that $\Gamma$ is strongly balanced, and we have a given instance.
First, we compute a small frame $F$ for set of assignments $\Phi$, using the algorithm outlined above.

Assume there are at least two variables, so $\Phi$ is at least binary.
For $1 \leq i<j \leq n$, let

$$
N_{i, j}(a)=\mid\left\{\left(u_{1}, \ldots, u_{i}\right):\left(u_{1}, \ldots, u_{n}\right) \in \Phi \text { and } u_{j}=a\right\} \mid .
$$

Then the total number of satisfying assignments, $N=|\Phi|$, is

$$
N=\sum_{a \in D} N_{n-1, n}(a) .
$$

## What the $N_{i, j}$ count

If $\Phi$ is the relation with tuples in $\mathbf{u} \in D^{n}$ :

$\left(u_{N, 1}, u_{N, 2}, \cdots \cdots, u_{N, i-1}, u_{N, i}, \cdots \cdots, u_{N, j}, \cdots \cdots, u_{N, n}\right)$
then $N_{i, j}(a)=\left|\left\{\mathbf{u} \in \operatorname{pr}_{\{1, \ldots, i-1, j\}} \Phi: u_{j}=a\right\}\right|$.
Note that $\operatorname{pr}_{\{1, \ldots, i-1, j\}} \Phi$ has fewer than $N$ tuples, in general, because many different tuples in $\Phi$ give rise to the same one in $\operatorname{pr}_{\{1, \ldots, i-1, j\}} \Phi$.

## What the $N_{i, j}$ count

If $\Phi$ is the relation with tuples in $\mathbf{u} \in D^{n}$ :

$$
\left.\begin{array}{ccccccc}
\left(u_{1,1}, u_{1,2},\right. & \cdots \cdots, u_{1, i-1}, & u_{1, i}, \cdots \cdots, & u_{1, j}, & \cdots \cdots, & u_{1, n}
\end{array}\right)
$$

then $N_{i, j}(a)=\left|\left\{\mathbf{u} \in \operatorname{pr}_{\{1, \ldots, i-1, j\}} \phi: u_{j}=a\right\}\right|$
Note that $\operatorname{pr}_{\{1, \ldots, i-1, j\}} \Phi$ has fewer than $N$ tuples, in general, because many different tuples in $\Phi$ give rise to the same one in $\mathrm{pr}_{\{1, \ldots, i-1, j\}} \Phi$.

## What the $N_{i, j}$ count

If $\Phi$ is the relation with tuples in $\mathbf{u} \in D^{n}$ :

| $\left(u_{1,1}, u_{1,2}, \cdots \cdots, u_{1, i-1}\right.$, | $u_{1, i}, \cdots \cdots$, | $u_{1, j}$ | $\left., \cdots \cdots, u_{1, n}\right)$ |
| :---: | :---: | :---: | :---: |
| $\left(u_{2,1}, u_{2,2}, \cdots \cdots, u_{2, i-1}\right.$ | $u_{2, i}, \cdots \cdots$, | $u_{2, j}$ | $\left., \cdots \cdots, u_{2, n}\right)$ |
| : ....... |  |  |  |
|  |  |  |  |
| $\left(u_{N, 1}, u_{N, 2}, \cdots \cdots, u_{N, i-1}\right.$, | $u_{N, i}$ | $u_{N, j}$ | $u_{N, n}$ |

then $N_{i, j}(a)=\left|\left\{\mathbf{u} \in \operatorname{pr}_{\{1, \ldots, i-1, j\}} \Phi: u_{j}=a\right\}\right|$.
Note that $\operatorname{pr}_{\{1, \ldots, i-1, j\}} \Phi$ has fewer than $N$ tuples, in general, because many different tuples in $\Phi$ give rise to the same one in $\operatorname{pr}_{\{1, \ldots, i-1, j\}} \Phi$.

## What the $N_{i, j}$ count

If $\Phi$ is the relation with tuples in $\mathbf{u} \in D^{n}$ :

| $\left(u_{1,1}, u_{1,2}, \cdots \cdots, u_{1, i-1}\right.$, | $u_{1, i}, \cdots \cdots \cdots$, | $u_{1, j}$ | , $\cdots \cdots, u_{1, n}$ |
| :---: | :---: | :---: | :---: |
| $\left(u_{2,1}, u_{2,2}, \cdots \cdots, u_{2, i-1}\right.$, | $u_{2, i}, \cdots \cdots$, | $u_{2, j}$ | ) |
|  |  |  |  |
| . |  |  |  |
| $\left(u_{N, 1}, u_{N, 2}, \cdots \cdots, u_{N, i-1}\right.$, | $u_{N, i}$ | $u_{N, j}$ | $\cdots \cdots, u_{N, n}$ |

then $N_{i, j}(a)=\left|\left\{\mathbf{u} \in \operatorname{pr}_{\{1, \ldots, i-1, j\}} \Phi: u_{j}=a\right\}\right|$.
Note that $\operatorname{pr}_{\{1, \ldots, i-1, j\}} \Phi$ has fewer than $N$ tuples, in general, because many different tuples in $\Phi$ give rise to the same one in $\mathrm{pr}_{\{1, \ldots, i-1, j\}} \Phi$.

## Computing the $N_{i, j}$ : rectangularity

Each $N_{1, j}$ can be calculated easily, because $\left|\mathrm{pr}_{1, j} \Phi\right| \leq|D|^{2}=\mathcal{O}(1)$.
Suppose we have computed each $N_{i-1, j}$, for some $i$.
We consider $\Lambda=\operatorname{pr}_{\{1, \ldots, i, j\}} \Phi$ to be a ternary relation on


The crucial observation is that, for different $(x, y) \in \mathrm{pr}_{i} \Phi \times \mathrm{pr}_{j} \Phi$, the sets $\left\{\mathbf{u} \in \operatorname{pr}_{\{1 \ldots, i-1\}} \Phi:(\mathbf{u}, x, y) \in \Lambda\right\}$ are disjoint or identical.

This follows by rectangularity:


## Computing the $N_{i, j}$ : rectangularity

Each $N_{1, j}$ can be calculated easily, because $\left|\mathrm{pr}_{1, j} \Phi\right| \leq|D|^{2}=\mathcal{O}(1)$.
Suppose we have computed each $N_{i-1, j}$, for some $i$.
We consider $\Lambda=\operatorname{pr}_{\{1, \ldots, i, j\}} \Phi$ to be a ternary relation on

The crucial observation is that, for different $(x, y) \in \mathrm{pr}_{j} \Phi \times \mathrm{pr}_{j} \Phi$, the sets $\left\{\mathbf{u} \in \operatorname{pr}_{\{1 \ldots ., i-1\}} \Phi:(\mathbf{u}, x, y) \in \Lambda\right\}$ are disjoint or identical.

This follows by rectangularity:


## Computing the $N_{i, j}$ : rectangularity

Each $N_{1, j}$ can be calculated easily, because $\left|\mathrm{pr}_{1, j} \Phi\right| \leq|D|^{2}=\mathcal{O}(1)$.
Suppose we have computed each $N_{i-1, j}$, for some $i$.
We consider $\Lambda=\operatorname{pr}_{\{1, \ldots, i, j\}} \Phi$ to be a ternary relation on

$$
\operatorname{pr}_{\{1, \ldots, i-1\}} \Phi \times \operatorname{pr}_{i} \Phi \times \operatorname{pr}_{j} \Phi .
$$

The crucial observation is that, for different $(x, y) \in \mathrm{pr}_{i} \Phi \times \mathrm{pr}_{j} \phi$, the sets $\left\{\mathbf{u} \in \operatorname{pr}_{\{1, \ldots, i-1\}} \Phi:(\mathbf{u}, x, y) \in \Lambda\right\}$ are disjoint or identical.

This follows by reciangularity:


## Computing the $N_{i, j}$ : rectangularity

Each $N_{1, j}$ can be calculated easily, because $\left|\mathrm{pr}_{1, j} \Phi\right| \leq|D|^{2}=\mathcal{O}(1)$.
Suppose we have computed each $N_{i-1, j}$, for some $i$.
We consider $\Lambda=\operatorname{pr}_{\{1, \ldots, i, j\}} \Phi$ to be a ternary relation on

$$
\operatorname{pr}_{\{1, \ldots, i-1\}} \Phi \times \operatorname{pr}_{i} \Phi \times \operatorname{pr}_{j} \Phi .
$$

The crucial observation is that, for different $(x, y) \in \operatorname{pr}_{i} \Phi \times \mathrm{pr}_{j} \Phi$, the sets $\left\{\mathbf{u} \in \operatorname{pr}_{\{1, \ldots, i-1\}} \Phi:(\mathbf{u}, x, y) \in \Lambda\right\}$ are disjoint or identical.

This follows by rectangularity:


## Computing the $N_{i, j}$ : rectangularity

Each $N_{1, j}$ can be calculated easily, because $\left|\mathrm{pr}_{1, j} \Phi\right| \leq|D|^{2}=\mathcal{O}(1)$.
Suppose we have computed each $N_{i-1, j}$, for some $i$.
We consider $\Lambda=\operatorname{pr}_{\{1, \ldots, i, j\}} \Phi$ to be a ternary relation on

$$
\operatorname{pr}_{\{1, \ldots, i-1\}} \Phi \times \operatorname{pr}_{i} \Phi \times \operatorname{pr}_{j} \Phi .
$$

The crucial observation is that, for different $(x, y) \in \operatorname{pr}_{i} \Phi \times \mathrm{pr}_{j} \Phi$, the sets $\left\{\mathbf{u} \in \operatorname{pr}_{\{1, \ldots, i-1\}} \Phi:(\mathbf{u}, x, y) \in \Lambda\right\}$ are disjoint or identical.

This follows by rectangularity:


## Computing the $N_{i, j}$ : rectangularity

Each $N_{1, j}$ can be calculated easily, because $\left|\mathrm{pr}_{1, j} \Phi\right| \leq|D|^{2}=\mathcal{O}(1)$.
Suppose we have computed each $N_{i-1, j}$, for some $i$.
We consider $\Lambda=\operatorname{pr}_{\{1, \ldots, i, j\}} \Phi$ to be a ternary relation on

$$
\operatorname{pr}_{\{1, \ldots, i-1\}} \Phi \times \operatorname{pr}_{i} \Phi \times \operatorname{pr}_{j} \Phi .
$$

The crucial observation is that, for different $(x, y) \in \operatorname{pr}_{i} \Phi \times \mathrm{pr}_{j} \Phi$, the sets $\left\{\mathbf{u} \in \operatorname{pr}_{\{1, \ldots, i-1\}} \Phi:(\mathbf{u}, x, y) \in \Lambda\right\}$ are disjoint or identical.

This follows by rectangularity:

$$
\left.\begin{array}{l}
(\mathbf{u}, x, y) \\
\left(\mathbf{u}^{\prime}, x, y\right) \\
\left(\mathbf{u}^{\prime}, x^{\prime}, y^{\prime}\right)
\end{array}\right\} \in \Lambda \Rightarrow\left(\mathbf{u}, x^{\prime}, y^{\prime}\right) \in \Lambda .
$$

## Computing the $N_{i, j}$ : strong balance

Using a frame $F$ for $\Phi$, we can determine an equivalence relation:

$$
\begin{aligned}
(x, y) \equiv\left(x^{\prime}, y^{\prime}\right) & \Leftrightarrow \quad\{\mathbf{u}:(\mathbf{u}, x, y) \in \Lambda\}=\left\{\mathbf{u}:\left(\mathbf{u}, x^{\prime}, y^{\prime}\right) \in \Lambda\right\} \\
(x, y) \not \equiv\left(x^{\prime}, y^{\prime}\right) & \Leftrightarrow \quad\{\mathbf{u}:(\mathbf{u}, x, y) \in \Lambda\} \cap\left\{\mathbf{u}:\left(\mathbf{u}, x^{\prime}, y^{\prime}\right) \in \Lambda\right\}=\emptyset
\end{aligned}
$$

## Now, since $\Lambda$ is pp-definable in $\Gamma$, it is balanced. Therefore, the matrix


is a rank-one block matrix, and $N_{i, j}(a)=\sum_{x \in D} M(x, a)$ are its column totals.

Let matrix $M$ be the quotient of $M$ under the equivalence $\equiv$
Using $F$ again, we can determine the block structure of $\widehat{M}$
Its row and column sums can be determined from $N_{i-1, i}$ and $N_{i-1, j}$
Hence we can determine $\widehat{M}$, and then $M$, and finally $N_{i, j}$

## Computing the $N_{i, j}$ : strong balance

Using a frame $F$ for $\Phi$, we can determine an equivalence relation:

$$
\begin{aligned}
(x, y) \equiv\left(x^{\prime}, y^{\prime}\right) & \Leftrightarrow \quad\{\mathbf{u}:(\mathbf{u}, x, y) \in \Lambda\}=\left\{\mathbf{u}:\left(\mathbf{u}, x^{\prime}, y^{\prime}\right) \in \Lambda\right\} \\
(x, y) \not \equiv\left(x^{\prime}, y^{\prime}\right) & \Leftrightarrow \quad\{\mathbf{u}:(\mathbf{u}, x, y) \in \Lambda\} \cap\left\{\mathbf{u}:\left(\mathbf{u}, x^{\prime}, y^{\prime}\right) \in \Lambda\right\}=\emptyset
\end{aligned}
$$

Now, since $\Lambda$ is pp-definable in $\Gamma$, it is balanced. Therefore, the matrix

$$
M(x, y)=|\{\mathbf{u}:(\mathbf{u}, x, y) \in \Lambda\}|
$$

is a rank-one block matrix, and $N_{i, j}(a)=\sum_{x \in D} M(x, a)$ are its column

## Let matrix $M$ be the quotient of $M$ under the equivalence $\equiv$.

Using $F$ again, we can determine the block structure of $\widehat{M}$
Its row and column sums can be determined from $N_{i-1, i}$ and $N_{i-1, j}$
$\square$

## Computing the $N_{i, j}$ : strong balance

Using a frame $F$ for $\Phi$, we can determine an equivalence relation:

$$
\begin{aligned}
(x, y) \equiv\left(x^{\prime}, y^{\prime}\right) & \Leftrightarrow \quad\{\mathbf{u}:(\mathbf{u}, x, y) \in \Lambda\}=\left\{\mathbf{u}:\left(\mathbf{u}, x^{\prime}, y^{\prime}\right) \in \Lambda\right\} \\
(x, y) \not \equiv\left(x^{\prime}, y^{\prime}\right) & \Leftrightarrow \quad\{\mathbf{u}:(\mathbf{u}, x, y) \in \Lambda\} \cap\left\{\mathbf{u}:\left(\mathbf{u}, x^{\prime}, y^{\prime}\right) \in \Lambda\right\}=\emptyset
\end{aligned}
$$

Now, since $\Lambda$ is pp-definable in $\Gamma$, it is balanced. Therefore, the matrix

$$
M(x, y)=|\{\mathbf{u}:(\mathbf{u}, x, y) \in \Lambda\}|
$$

is a rank-one block matrix, and $N_{i, j}(a)=\sum_{x \in D} M(x, a)$ are its column totals.

Let matrix $M$ be the quotient of $M$ under the equivalence
Using $F$ again, we can determine the block structure of $\widehat{M}$
Its row and column sums can be determined from $N_{i-1, i}$ and $N_{i-1, j}$
Hence we can determine $\widehat{M}$, and then $M$, and finally $N_{i j}$.

## Computing the $N_{i, j}$ : strong balance

Using a frame $F$ for $\Phi$, we can determine an equivalence relation:

$$
\begin{aligned}
& (x, y) \equiv\left(x^{\prime}, y^{\prime}\right) \quad \Leftrightarrow \quad\{\mathbf{u}:(\mathbf{u}, x, y) \in \Lambda\}=\left\{\mathbf{u}:\left(\mathbf{u}, x^{\prime}, y^{\prime}\right) \in \Lambda\right\} \\
& (x, y) \not \equiv\left(x^{\prime}, y^{\prime}\right) \Leftrightarrow
\end{aligned} \Leftrightarrow \quad\{\mathbf{u}:(\mathbf{u}, x, y) \in \Lambda\} \cap\left\{\mathbf{u}:\left(\mathbf{u}, x^{\prime}, y^{\prime}\right) \in \Lambda\right\}=\emptyset .
$$

Now, since $\Lambda$ is pp-definable in $\Gamma$, it is balanced. Therefore, the matrix

$$
M(x, y)=|\{\mathbf{u}:(\mathbf{u}, x, y) \in \Lambda\}|
$$

is a rank-one block matrix, and $N_{i, j}(a)=\sum_{x \in D} M(x, a)$ are its column totals.

Let matrix $\widehat{M}$ be the quotient of $M$ under the equivalence $\equiv$.
Using F again, we can determine the block structure of $\widehat{M}$.
Its row and column sums can be determined from $N_{i-1, i}$ and $N_{i-1, j}$
Hence we can determine $\widehat{M}$, and then $M$, and finally $N_{i j}$

## Computing the $N_{i, j}$ : strong balance

Using a frame $F$ for $\Phi$, we can determine an equivalence relation:

$$
\begin{aligned}
& (x, y) \equiv\left(x^{\prime}, y^{\prime}\right) \quad \Leftrightarrow \quad\{\mathbf{u}:(\mathbf{u}, x, y) \in \Lambda\}=\left\{\mathbf{u}:\left(\mathbf{u}, x^{\prime}, y^{\prime}\right) \in \Lambda\right\} \\
& (x, y) \not \equiv\left(x^{\prime}, y^{\prime}\right) \Leftrightarrow
\end{aligned} \Leftrightarrow \quad\{\mathbf{u}:(\mathbf{u}, x, y) \in \Lambda\} \cap\left\{\mathbf{u}:\left(\mathbf{u}, x^{\prime}, y^{\prime}\right) \in \Lambda\right\}=\emptyset .
$$

Now, since $\Lambda$ is pp-definable in $\Gamma$, it is balanced. Therefore, the matrix

$$
M(x, y)=|\{\mathbf{u}:(\mathbf{u}, x, y) \in \Lambda\}|
$$

is a rank-one block matrix, and $N_{i, j}(a)=\sum_{x \in D} M(x, a)$ are its column totals.

Let matrix $\widehat{M}$ be the quotient of $M$ under the equivalence $\equiv$. Using $F$ again, we can determine the block structure of $\widehat{M}$.

Its row and column sums can be determined from $N_{i-1, i}$ and $N_{i-1, j}$ Hence we can determine $\widehat{M}$, and then $M$, and finally $N_{i, j}$.

## Computing the $N_{i, j}$ : strong balance

Using a frame $F$ for $\Phi$, we can determine an equivalence relation:

$$
\begin{aligned}
& (x, y) \equiv\left(x^{\prime}, y^{\prime}\right) \quad \Leftrightarrow \quad\{\mathbf{u}:(\mathbf{u}, x, y) \in \Lambda\}=\left\{\mathbf{u}:\left(\mathbf{u}, x^{\prime}, y^{\prime}\right) \in \Lambda\right\} \\
& (x, y) \not \equiv\left(x^{\prime}, y^{\prime}\right) \Leftrightarrow
\end{aligned} \Leftrightarrow \quad\{\mathbf{u}:(\mathbf{u}, x, y) \in \Lambda\} \cap\left\{\mathbf{u}:\left(\mathbf{u}, x^{\prime}, y^{\prime}\right) \in \Lambda\right\}=\emptyset .
$$

Now, since $\Lambda$ is pp-definable in $\Gamma$, it is balanced. Therefore, the matrix

$$
M(x, y)=|\{\mathbf{u}:(\mathbf{u}, x, y) \in \Lambda\}|
$$

is a rank-one block matrix, and $N_{i, j}(a)=\sum_{x \in D} M(x, a)$ are its column totals.

Let matrix $\widehat{M}$ be the quotient of $M$ under the equivalence $\equiv$.
Using $F$ again, we can determine the block structure of $\widehat{M}$.
Its row and column sums can be determined from $N_{i-1, i}$ and $N_{i-1, j}$.
Hence we can determine $\widehat{M}$, and then $M$, and finally $N_{i,}$,

## Computing the $N_{i, j}$ : strong balance

Using a frame $F$ for $\Phi$, we can determine an equivalence relation:

$$
\begin{aligned}
& (x, y) \equiv\left(x^{\prime}, y^{\prime}\right) \quad \Leftrightarrow \quad\{\mathbf{u}:(\mathbf{u}, x, y) \in \Lambda\}=\left\{\mathbf{u}:\left(\mathbf{u}, x^{\prime}, y^{\prime}\right) \in \Lambda\right\} \\
& (x, y) \not \equiv\left(x^{\prime}, y^{\prime}\right) \Leftrightarrow
\end{aligned} \Leftrightarrow \quad\{\mathbf{u}:(\mathbf{u}, x, y) \in \Lambda\} \cap\left\{\mathbf{u}:\left(\mathbf{u}, x^{\prime}, y^{\prime}\right) \in \Lambda\right\}=\emptyset .
$$

Now, since $\Lambda$ is pp-definable in $\Gamma$, it is balanced. Therefore, the matrix

$$
M(x, y)=|\{\mathbf{u}:(\mathbf{u}, x, y) \in \Lambda\}|
$$

is a rank-one block matrix, and $N_{i, j}(a)=\sum_{x \in D} M(x, a)$ are its column totals.

Let matrix $\widehat{M}$ be the quotient of $M$ under the equivalence $\equiv$. Using $F$ again, we can determine the block structure of $\widehat{M}$. Its row and column sums can be determined from $N_{i-1, i}$ and $N_{i-1, j}$. Hence we can determine $\widehat{M}$, and then $M$, and finally $N_{i, j}$.

## Dichotomy theorem

Therefore we have

## Theorem

If $\Gamma$ is strongly balanced, then $\# \operatorname{CSP}(\Gamma)$ is computable in time $\mathcal{O}\left(n^{5}\right)$. Otherwise, it is \#P-complete.

We can prove that strong balance is equivalent to the congruence singularity criterion of BULATOV (2008). So the dichotomy is identical, as would be expected

But is the strong balance property decidable, for a given $\Gamma$ ?

The answer to this question is yes and, in fact, it is decidable in NP

## Dichotomy theorem

Therefore we have

## Theorem

If $\Gamma$ is strongly balanced, then $\# \operatorname{CSP}(\Gamma)$ is computable in time $\mathcal{O}\left(n^{5}\right)$. Otherwise, it is \#P-complete.

We can prove that strong balance is equivalent to the congruence singularity criterion of Bulatov (2008). So the dichotomy is identical, as would be expected.

But is the strong balance property decidable, for a given 「?
The answer to this question is yes and, in fact, it is decidable in NP

## Dichotomy theorem

Therefore we have

## Theorem

If $\Gamma$ is strongly balanced, then $\# \operatorname{CSP}(\Gamma)$ is computable in time $\mathcal{O}\left(n^{5}\right)$. Otherwise, it is \#P-complete.

We can prove that strong balance is equivalent to the congruence singularity criterion of Bulatov (2008). So the dichotomy is identical, as would be expected.

But is the strong balance property decidable, for a given 「?

The answer to this question is yes and, in fact, it is decidable in NP

## Dichotomy theorem

Therefore we have

## Theorem

If $\Gamma$ is strongly balanced, then $\# \operatorname{CSP}(\Gamma)$ is computable in time $\mathcal{O}\left(n^{5}\right)$. Otherwise, it is \#P-complete.

We can prove that strong balance is equivalent to the congruence singularity criterion of Bulatov (2008). So the dichotomy is identical, as would be expected.

But is the strong balance property decidable, for a given $\Gamma$ ?
The answer to this question is yes and, in fact, it is decidable in NP.

## (1) Introduction

(2) Rectangularity
(3) Frames
(4) Counting
(5) Decidability

6 Conclusion

## The question

Is the following problem decidable？

Input：a constraint language「
Question：is 「 strongly balanced？

And，if so，what is its computational complexity？
Note that $D$ and 「 are not fixed parameters in this meta－problem． though they were in the dichotomy theorem．

## The question

Is the following problem decidable？

Input：a constraint language「
Question：is 「 strongly balanced？

And，if so，what is its computational complexity？
Note that $D$ and 「 are not fixed parameters in this meta－problem， though they were in the dichotomy theorem．

## The question

Is the following problem decidable?

Input: a constraint language $\Gamma$
Question: is 「 strongly balanced?

And, if so, what is its computational complexity?
Note that $D$ and 「 are not fixed parameters in this meta-problem, though they were in the dichotomy theorem.

## A weaker condition

We can relax the strong balance criterion to a more useful condition which we call almost-strong balance.

An constraint language 「 with domain $D$ is almost-strongly balanced if the balance matrix of every pp-definable ternary relation which is a subset of $D^{k} \times D \times D$, for some $k$, is a rank-one block matrix.

This is sufficient for the algorithm we have described to succeed, and hence is equivalent to strong balance by the chain of implications:


## A weaker condition

We can relax the strong balance criterion to a more useful condition which we call almost-strong balance.

An constraint language $\Gamma$ with domain $D$ is almost-strongly balanced if the balance matrix of every pp-definable ternary relation which is a subset of $D^{k} \times D \times D$, for some $k$, is a rank-one block matrix.

This is sufficient for the algorithm we have described to succeed, and hence is equivalent to strong balance by the chain of implications:


## A weaker condition

We can relax the strong balance criterion to a more useful condition which we call almost-strong balance.

An constraint language $\Gamma$ with domain $D$ is almost-strongly balanced if the balance matrix of every pp-definable ternary relation which is a subset of $D^{k} \times D \times D$, for some $k$, is a rank-one block matrix.

This is sufficient for the algorithm we have described to succeed, and hence is equivalent to strong balance by the chain of implications:


## A weaker condition

We can relax the strong balance criterion to a more useful condition which we call almost-strong balance.

An constraint language $\Gamma$ with domain $D$ is almost-strongly balanced if the balance matrix of every pp-definable ternary relation which is a subset of $D^{k} \times D \times D$, for some $k$, is a rank-one block matrix.

This is sufficient for the algorithm we have described to succeed, and hence is equivalent to strong balance by the chain of implications:
strong balance $\Longrightarrow$ almost strong balance
$\Longrightarrow$ the algorithm works
$\Longrightarrow$ the problem is in FP
$\Longrightarrow$ the problem isn't \#P-complete
$\Longrightarrow$ strong balance,

## A weaker condition

We can relax the strong balance criterion to a more useful condition which we call almost-strong balance.

An constraint language $\Gamma$ with domain $D$ is almost-strongly balanced if the balance matrix of every pp-definable ternary relation which is a subset of $D^{k} \times D \times D$, for some $k$, is a rank-one block matrix.

This is sufficient for the algorithm we have described to succeed, and hence is equivalent to strong balance by the chain of implications:

$$
\begin{aligned}
\text { strong balance } & \Longrightarrow \text { almost strong balance } \\
& \Longrightarrow \text { the algorithm works } \\
& \Longrightarrow \text { the problem is in FP } \\
& \Longrightarrow \text { the problem isn't \#P-complete } \\
& \Longrightarrow \text { strong balance, }
\end{aligned}
$$

## A weaker condition

We can relax the strong balance criterion to a more useful condition which we call almost-strong balance.

An constraint language $\Gamma$ with domain $D$ is almost-strongly balanced if the balance matrix of every pp-definable ternary relation which is a subset of $D^{k} \times D \times D$, for some $k$, is a rank-one block matrix.

This is sufficient for the algorithm we have described to succeed, and hence is equivalent to strong balance by the chain of implications:

$$
\begin{aligned}
\text { strong balance } & \Longrightarrow \text { almost strong balance } \\
& \Longrightarrow \text { the algorithm works } \\
& \Longrightarrow \text { the problem is in FP }
\end{aligned}
$$

$\Longrightarrow$ the problem isn't \#P-complete
$\Longrightarrow$ strong balance,

## A weaker condition

We can relax the strong balance criterion to a more useful condition which we call almost-strong balance.

An constraint language $\Gamma$ with domain $D$ is almost-strongly balanced if the balance matrix of every pp-definable ternary relation which is a subset of $D^{k} \times D \times D$, for some $k$, is a rank-one block matrix.

This is sufficient for the algorithm we have described to succeed, and hence is equivalent to strong balance by the chain of implications:

$$
\begin{aligned}
\text { strong balance } & \Longrightarrow \text { almost strong balance } \\
& \Longrightarrow \text { the algorithm works } \\
& \Longrightarrow \text { the problem is in FP } \\
& \Longrightarrow \text { the problem isn't \#P-complete }
\end{aligned}
$$

$\Longrightarrow$ strong balance,

## A weaker condition

We can relax the strong balance criterion to a more useful condition which we call almost-strong balance.

An constraint language $\Gamma$ with domain $D$ is almost-strongly balanced if the balance matrix of every pp-definable ternary relation which is a subset of $D^{k} \times D \times D$, for some $k$, is a rank-one block matrix.

This is sufficient for the algorithm we have described to succeed, and hence is equivalent to strong balance by the chain of implications:

$$
\begin{aligned}
\text { strong balance } & \Longrightarrow \text { almost strong balance } \\
& \Longrightarrow \text { the algorithm works } \\
& \Longrightarrow \text { the problem is in FP } \\
& \Longrightarrow \text { the problem isn't \#P-complete } \\
& \Longrightarrow \text { strong balance, }
\end{aligned}
$$

## A weaker condition

We can relax the strong balance criterion to a more useful condition which we call almost-strong balance.

An constraint language $\Gamma$ with domain $D$ is almost-strongly balanced if the balance matrix of every pp-definable ternary relation which is a subset of $D^{k} \times D \times D$, for some $k$, is a rank-one block matrix.

This is sufficient for the algorithm we have described to succeed, and hence is equivalent to strong balance by the chain of implications:

```
strong balance \Longrightarrow almost strong balance
    \Longrightarrow ~ t h e ~ a l g o r i t h m ~ w o r k s
    the problem is in FP
    \Longrightarrow ~ t h e ~ p r o b l e m ~ i s n ' t ~ \# P - c o m p l e t e
    \Longrightarrow ~ s t r o n g ~ b a l a n c e ,
```

provided that the dichotomy exists, i.e. $\mathrm{FP} \neq \# \mathrm{P}$.

## A useful characterisation of strong balance

We require a more uniform condition that a matrix is a rank-one block matrix. This is provided by the following lemma:

## Lemma

$M$ is a rank-one block matrix if and only if its underlying relation is rectangular, and
for every 2 2 submatrix $\left(\begin{array}{cc}u \\ w & x \\ x\end{array}\right)$

Strong rectangularity (which we can test via Mal'tsev polymorphism) implies that the underlying relation of any such matrix is rectangular So, for strong balance, we need that
$M(a, c)^{2} M(b, d)^{2} M(a, d) M(b, c)=M(a, d)^{2} M(b, c)^{2} M(a, c) M(b, d)$ for all $a, b$ d


## A useful characterisation of strong balance

We require a more uniform condition that a matrix is a rank-one block matrix. This is provided by the following lemma:

## Lemma

$M$ is a rank-one block matrix if and only if its underlying relation is rectangular, and

$$
u^{2} x^{2} v w=v^{2} w^{2} u x
$$

for every $2 \times 2$ submatrix $\left(\begin{array}{cc}u & v \\ w & x\end{array}\right)$.

Strong rectangularity (which we can test via Mal'tsev polymorphism) implies that the underlying relation of any such matrix is rectangular So, for strong balance, we need that
$M(a, c)^{2} M(b, d)^{2} M(a, d) M(b, c)=M(a, d)^{2} M(b, c)^{2} M(a, c) M(b, d)$


## A useful characterisation of strong balance

We require a more uniform condition that a matrix is a rank-one block matrix. This is provided by the following lemma:

## Lemma

$M$ is a rank-one block matrix if and only if its underlying relation is rectangular, and

$$
u^{2} x^{2} v w=v^{2} w^{2} u x
$$

for every $2 \times 2$ submatrix $\left(\begin{array}{cc}u & v \\ w & x\end{array}\right)$.

Strong rectangularity (which we can test via Mal'tsev polymorphism) implies that the underlying relation of any such matrix is rectangular.

So, for strong balance, we need that
$M(a, c)^{2} M(b, d)^{2} M(a, d) M(b, c)=M(a, d)^{2} M(b, c)^{2} M(a, c) M(b, d)$


## A useful characterisation of strong balance

We require a more uniform condition that a matrix is a rank-one block matrix. This is provided by the following lemma:

## Lemma

$M$ is a rank-one block matrix if and only if its underlying relation is rectangular, and

$$
u^{2} x^{2} v w=v^{2} w^{2} u x
$$

for every $2 \times 2$ submatrix $\left(\begin{array}{cc}u & v \\ w & x\end{array}\right)$.

Strong rectangularity (which we can test via Mal'tsev polymorphism) implies that the underlying relation of any such matrix is rectangular.

So, for strong balance, we need that

$$
M(a, c)^{2} M(b, d)^{2} M(a, d) M(b, c)=M(a, d)^{2} M(b, c)^{2} M(a, c) M(b, d)
$$



## A useful characterisation of strong balance

We require a more uniform condition that a matrix is a rank-one block matrix. This is provided by the following lemma:

## Lemma

$M$ is a rank-one block matrix if and only if its underlying relation is rectangular, and

$$
u^{2} x^{2} v w=v^{2} w^{2} u x
$$

for every $2 \times 2$ submatrix $\left(\begin{array}{cc}u & v \\ w & x\end{array}\right)$.

Strong rectangularity (which we can test via Mal'tsev polymorphism) implies that the underlying relation of any such matrix is rectangular.

So, for strong balance, we need that

$$
M(a, c)^{2} M(b, d)^{2} M(a, d) M(b, c)=M(a, d)^{2} M(b, c)^{2} M(a, c) M(b, d)
$$

for all $a, b, c, d \in D$ and every $M=M(R), R \subseteq D^{k} \times D \times D$.

## Cartesian powers

We will recast this as a problem in $D^{6}$. We abbreviate the sextuple $(a, b, c, d, e, f) \in D^{6}$ to abcdef.

Now, using the usual definition of Cartesian powers of a finite structure, we can define a new constraint language $\Gamma^{\prime}$ over $D^{6}$, and translate the relation $R \subseteq D^{k}$ to $R^{\prime} \subseteq\left(D^{6}\right)^{k}$, with corresponding balance matrix $M^{\prime}$.

Our condition for 「 to be strongly balanced then becomes that

$$
M^{\prime}(a a b b a b, c c d d d c)=M^{\prime}(a a b b a b, d d c c c d)
$$

for all $a, b, c, d \in D$ and all balance matrices $M^{\prime}$ of these translated relations $R^{\prime}$.

We will write $M^{\prime}(\bar{a}, \bar{b})=M^{\prime}(\bar{b}, \bar{c})$ for these identities, where $\bar{a}, \bar{b}, \bar{c} \in D^{6}$

## Cartesian powers

We will recast this as a problem in $D^{6}$. We abbreviate the sextuple $(a, b, c, d, e, f) \in D^{6}$ to abcdef.

Now, using the usual definition of Cartesian powers of a finite structure, we can define a new constraint language $\Gamma^{\prime}$ over $D^{6}$, and translate the relation $R \subseteq D^{k}$ to $R^{\prime} \subseteq\left(D^{6}\right)^{k}$, with corresponding balance matrix $M^{\prime}$.

Our condition for 「 to be strongly balanced then becomes that

## $M^{\prime}(a a b b a b, c c d d d c)=M^{\prime}(a a b b a b, d d c c c d)$

for all $a, b, c, d \in D$ and all balance matrices $M^{\prime}$ of these translated
relations $R^{\prime}$.
We will write $M^{\prime}(\bar{a}, \bar{b})=M^{\prime}(\bar{b}, \bar{c})$ for these identities, where $\bar{a}, \bar{b}, \bar{c} \in D^{6}$

## Cartesian powers

We will recast this as a problem in $D^{6}$. We abbreviate the sextuple $(a, b, c, d, e, f) \in D^{6}$ to abcdef.

Now, using the usual definition of Cartesian powers of a finite structure, we can define a new constraint language $\Gamma^{\prime}$ over $D^{6}$, and translate the relation $R \subseteq D^{k}$ to $R^{\prime} \subseteq\left(D^{6}\right)^{k}$, with corresponding balance matrix $M^{\prime}$.

Our condition for $\Gamma$ to be strongly balanced then becomes that

## $M^{\prime}($ aabbab, ccdddc $)=M^{\prime}($ aabbab, ddcccd $)$

for all $a, b, c, d \in D$ and all balance matrices $M^{\prime}$ of these translated
relations $R^{\prime}$
We will write $M^{\prime}(\bar{a}, \bar{b})=M^{\prime}(\bar{b}, \bar{c})$ for these identities, where $\bar{a}, \bar{b}, \bar{c} \in D^{6}$

## Cartesian powers

We will recast this as a problem in $D^{6}$. We abbreviate the sextuple $(a, b, c, d, e, f) \in D^{6}$ to abcdef.

Now, using the usual definition of Cartesian powers of a finite structure, we can define a new constraint language $\Gamma^{\prime}$ over $D^{6}$, and translate the relation $R \subseteq D^{k}$ to $R^{\prime} \subseteq\left(D^{6}\right)^{k}$, with corresponding balance matrix $M^{\prime}$.

Our condition for 「 to be strongly balanced then becomes that

$$
M^{\prime}(a a b b a b, c c d d d c)=M^{\prime}(a a b b a b, d d c c c d)
$$

for all $a, b, c, d \in D$ and all balance matrices $M^{\prime}$ of these translated relations $R^{\prime}$

We will write $M^{\prime}(\bar{a}, \bar{b})=M^{\prime}(\bar{b}, \bar{c})$ for these identities, where $\bar{a}, \bar{b}, \bar{c} \in D^{6}$

## Cartesian powers

We will recast this as a problem in $D^{6}$. We abbreviate the sextuple $(a, b, c, d, e, f) \in D^{6}$ to abcdef.

Now, using the usual definition of Cartesian powers of a finite structure, we can define a new constraint language $\Gamma^{\prime}$ over $D^{6}$, and translate the relation $R \subseteq D^{k}$ to $R^{\prime} \subseteq\left(D^{6}\right)^{k}$, with corresponding balance matrix $M^{\prime}$.

Our condition for $\Gamma$ to be strongly balanced then becomes that

$$
M^{\prime}(a a b b a b, c c d d d c)=M^{\prime}(a a b b a b, d d c c c d)
$$

for all $a, b, c, d \in D$ and all balance matrices $M^{\prime}$ of these translated relations $R^{\prime}$.

We will write $M^{\prime}(\bar{a}, \bar{b})=M^{\prime}(\bar{b}, \bar{c})$ for these identities, where $\bar{a}, \bar{b}, \bar{c} \in D^{6}$

## Cartesian powers

We will recast this as a problem in $D^{6}$. We abbreviate the sextuple $(a, b, c, d, e, f) \in D^{6}$ to abcdef.

Now, using the usual definition of Cartesian powers of a finite structure, we can define a new constraint language $\Gamma^{\prime}$ over $D^{6}$, and translate the relation $R \subseteq D^{k}$ to $R^{\prime} \subseteq\left(D^{6}\right)^{k}$, with corresponding balance matrix $M^{\prime}$.

Our condition for $\Gamma$ to be strongly balanced then becomes that

$$
M^{\prime}(a a b b a b, c c d d d c)=M^{\prime}(a a b b a b, d d c c c d)
$$

for all $a, b, c, d \in D$ and all balance matrices $M^{\prime}$ of these translated relations $R^{\prime}$.

We will write $M^{\prime}(\bar{a}, \bar{b})=M^{\prime}(\bar{b}, \bar{c})$ for these identities, where $\bar{a}, \bar{b}, \bar{c} \in D^{6}$.

## Automorphisms

Our condition is then that $M^{\prime}(\bar{a}, \bar{b})=M^{\prime}(\bar{b}, \bar{c})$ for all $\bar{a}, \bar{b}, \bar{c}$ (of appropriate form) and all $M^{\prime}=M^{\prime}\left(R^{\prime}\right)$.

This means that, for every $R^{\prime}$, the number of tuples beginning $\bar{a}, \bar{b}$ is always the same as the number beginning $\bar{a}, \bar{c}$.

Using a technique of LovÁsz (1967), we can show that this happens if and only if, for every $\bar{a}, \bar{b}, \bar{c}$ (of appropriate form), there exists an automorphism $\eta$ of $\Gamma^{\prime}$ with $\eta(\bar{a})=\bar{a}$ and $\eta(\bar{b})=\bar{c}$.

We may therefore use the existence of these automorphisms as the criterion for strong balance.

## Automorphisms

Our condition is then that $M^{\prime}(\bar{a}, \bar{b})=M^{\prime}(\bar{b}, \bar{c})$ for all $\bar{a}, \bar{b}, \bar{c}$ (of appropriate form) and all $M^{\prime}=M^{\prime}\left(R^{\prime}\right)$.

This means that, for every $R^{\prime}$, the number of tuples beginning $\bar{a}, \bar{b}$ is always the same as the number beginning $\bar{a}, \bar{c}$.

> Using a technique of Lovász (1967), we can show that this happens if and only if, for every $\bar{a}, \bar{b}, \bar{c}$ (of appropriate form), there exists an automorphism $\eta$ of $\Gamma^{\prime}$ with $\eta(\bar{a})=\bar{a}$ and $\eta(\bar{b})=\bar{c}$.

We may therefore use the existence of these automorphisms as the criterion for strong balance.

## Automorphisms

Our condition is then that $M^{\prime}(\bar{a}, \bar{b})=M^{\prime}(\bar{b}, \bar{c})$ for all $\bar{a}, \bar{b}, \bar{c}$ (of appropriate form) and all $M^{\prime}=M^{\prime}\left(R^{\prime}\right)$.

This means that, for every $R^{\prime}$, the number of tuples beginning $\bar{a}, \bar{b}$ is always the same as the number beginning $\bar{a}, \bar{c}$.

Using a technique of LovÁsz (1967), we can show that this happens if and only if, for every $\bar{a}, \bar{b}, \bar{c}$ (of appropriate form), there exists an automorphism $\eta$ of $\Gamma^{\prime}$ with $\eta(\bar{a})=\bar{a}$ and $\eta(\bar{b})=\bar{c}$.

We may therefore use the existence of these automorphisms as the criterion for strong balance.

## Automorphisms

Our condition is then that $M^{\prime}(\bar{a}, \bar{b})=M^{\prime}(\bar{b}, \bar{c})$ for all $\bar{a}, \bar{b}, \bar{c}$ (of appropriate form) and all $M^{\prime}=M^{\prime}\left(R^{\prime}\right)$.

This means that, for every $R^{\prime}$, the number of tuples beginning $\bar{a}, \bar{b}$ is always the same as the number beginning $\bar{a}, \bar{c}$.

Using a technique of LovÁsz (1967), we can show that this happens if and only if, for every $\bar{a}, \bar{b}, \bar{c}$ (of appropriate form), there exists an automorphism $\eta$ of $\Gamma^{\prime}$ with $\eta(\bar{a})=\bar{a}$ and $\eta(\bar{b})=\bar{c}$.

We may therefore use the existence of these automorphisms as the criterion for strong balance.

## Decidability

## Theorem

Strong balance is decidable in NP.

```
Proof.
First verify that \Gamma is strongly rectangular. If not, answer no. If so:
Construct }\mp@subsup{\Gamma}{}{\prime}\mathrm{ and, for each }\overline{a},\overline{b},\overline{c}\mathrm{ of the required form, nondeterministically
guess a function }\eta:\mp@subsup{D}{}{6}->\mp@subsup{D}{}{6}\mathrm{ . Check that these functions are the required
automorphisms. If so, answer yes, otherwise answer no
```

As a corollary, we have

## Theorem

Congruence singularity is decidable in NP

## Decidability

## Theorem

Strong balance is decidable in NP.

## Proof.

First verify that $\Gamma$ is strongly rectangular. If not, answer no. If so:
Construct $\Gamma^{\prime}$ and, for each $\bar{a}, \bar{b}, \bar{c}$ of the required form, nondeterministically guess a function $\eta: D^{6} \rightarrow D^{6}$. Check that these functions are the required automorphisms. If so, answer yes, otherwise answer no.

As a corollary, we have

## Theorem

Congruence singularity is decidable in NP

## Decidability

## Theorem

Strong balance is decidable in NP.

## Proof.

First verify that $\Gamma$ is strongly rectangular. If not, answer no. If so:
Construct $\Gamma^{\prime}$ and, for each $\bar{a}, \bar{b}, \bar{c}$ of the required form, nondeterministically guess a function $\eta: D^{6} \rightarrow D^{6}$. Check that these functions are the required automorphisms. If so, answer yes, otherwise answer no.

As a corollary, we have

## Theorem <br> Congruence singularity is decidable in NP.

## Complexity of strong balance

It seems unlikely that strong balance is as hard as NP.
It is not difficult to show that strong balance is reducible to the graph isomorphism problem GI.

GI is clearly in NP but, if it is NP-complete then it follows that the polynomial time hierarchy collapses to the second level.

This gives compelling evidence that strong balance is not NP-complete.

## Complexity of strong balance

It seems unlikely that strong balance is as hard as NP.
It is not difficult to show that strong balance is reducible to the graph isomorphism problem GI.

GI is clearly in NP but, if it is NP-complete then it follows that the polynomial time hierarchy collapses to the second level.

This gives compelling evidence that strong balance is not NP-complete.

## Complexity of strong balance

It seems unlikely that strong balance is as hard as NP.
It is not difficult to show that strong balance is reducible to the graph isomorphism problem GI.

GI is clearly in NP but, if it is NP-complete then it follows that the polynomial time hierarchy collapses to the second level.

This gives compelling evidence that strong balance is not NP-complete.

## Complexity of strong balance

It seems unlikely that strong balance is as hard as NP.
It is not difficult to show that strong balance is reducible to the graph isomorphism problem GI.

GI is clearly in NP but, if it is NP-complete then it follows that the polynomial time hierarchy collapses to the second level.

This gives compelling evidence that strong balance is not NP-complete

## Complexity of strong balance

It seems unlikely that strong balance is as hard as NP.
It is not difficult to show that strong balance is reducible to the graph isomorphism problem GI.

GI is clearly in NP but, if it is NP-complete then it follows that the polynomial time hierarchy collapses to the second level.

This gives compelling evidence that strong balance is not NP-complete.

## Complexity of strong balance

It seems unlikely that strong balance is as hard as NP.
It is not difficult to show that strong balance is reducible to the graph isomorphism problem GI.

GI is clearly in NP but, if it is NP-complete then it follows that the polynomial time hierarchy collapses to the second level.

This gives compelling evidence that strong balance is not NP-complete.

## (1) Introduction

(2) Rectangularity
(3) Frames
(4) Counting
(5) Decidability
(6) Conclusion

## Open questions and further work

- Can the algorithm for handling strongly rectangular relations be made more efficient? It is $\mathcal{O}\left(n^{5}\right)$, but there seems no reason why is should be worse than matrix multiplication: $\mathcal{O}\left(n^{2.376}\right)$. This computation dominates the counting algorithm.
- What about weighted \#CSP ?

- What new or known special cases (for restricted classes of Г) can be derived from our results? Can the algorithm be made more efficient in these cases? Most known special cases have $O(n)$ time counting algorithms.


## Open questions and further work

- Can the algorithm for handling strongly rectangular relations be made more efficient? It is $\mathcal{O}\left(n^{5}\right)$, but there seems no reason why is should be worse than matrix multiplication: $\mathcal{O}\left(n^{2.376}\right)$. This computation dominates the counting algorithm.
- What about weighted \#CSP ?

Bulatov, Dyer, Goldberg, Jalsenius, Jerrum \& Richerby (2010) show that the dichotomy extends to rational-weighted \#CSP. Cai, Chen \& Lu (2011) have extended it to algebraic weights by further developing the approach used here.
More generally, negative or complex weights can be considered.
Dichotomies are currently known only in special cases.

- What new or known special cases (for restricted classes of Г) can be derived from our results? Can the algorithm be made more efficient in these cases? Most known special cases have $\mathcal{O}(n)$ time counting algorithms.


## Open questions and further work

- Can the algorithm for handling strongly rectangular relations be made more efficient? It is $\mathcal{O}\left(n^{5}\right)$, but there seems no reason why is should be worse than matrix multiplication: $\mathcal{O}\left(n^{2.376}\right)$. This computation dominates the counting algorithm.
- What about weighted \#CSP ?

Bulatov, Dyer, Goldberg, Jalsenius, Jerrum \& Richerby (2010) show that the dichotomy extends to rational-weighted \#CSP. Cai, Chen \& Lu (2011) have extended it to algebraic weights by further developing the approach used here.
More generally, negative or complex weights can be considered.
Dichotomies are currently known only in special cases.

- What new or known special cases (for restricted classes of Г) can be derived from our results? Can the algorithm be made more efficient in these cases? Most known special cases have $\mathcal{O}(n)$ time counting algorithms.


## Open questions and further work

- What can be said if restrictions are placed on the instance? For example, if any variable can occur only a bounded number of times in the constraints? The two known approaches to the general dichotomy shed no light on this.
- What can be said for approximate counting ? It seems unlikely that a simple dichotomy exists, but D, Goldberg And Jerrum (2010) have given a trichotomy for the Boolean domain.
- Can we be more precise about the complexity of strong balance? Is it equivalent to GI? Is it in $P$ (e.g. via a special case of GI ) ?
- Congruence singularity is decidable by equivalence to strong balance. But is there a direct proof?


## Open questions and further work

- What can be said if restrictions are placed on the instance? For example, if any variable can occur only a bounded number of times in the constraints? The two known approaches to the general dichotomy shed no light on this.
- What can be said for approximate counting ? It seems unlikely that a simple dichotomy exists, but D, Goldberg and Jerrum (2010) have given a trichotomy for the Boolean domain.
- Can we be more precise about the complexity of strong balance? Is it equivalent to GI? Is it in P (e.g. via a special case of GI) ?
- Congruence singularity is decidable by equivalence to strong balance. But is there a direct proof?


## Open questions and further work

- What can be said if restrictions are placed on the instance? For example, if any variable can occur only a bounded number of times in the constraints? The two known approaches to the general dichotomy shed no light on this.
- What can be said for approximate counting? It seems unlikely that a simple dichotomy exists, but D, Goldberg and Jerrum (2010) have given a trichotomy for the Boolean domain.
- Can we be more precise about the complexity of strong balance? Is it equivalent to GI ? Is it in P (e.g. via a special case of GI ) ?
- Congruence singularity is decidable by equivalence to strong balance. But is there a direct proof?


## Open questions and further work

- What can be said if restrictions are placed on the instance? For example, if any variable can occur only a bounded number of times in the constraints? The two known approaches to the general dichotomy shed no light on this.
- What can be said for approximate counting? It seems unlikely that a simple dichotomy exists, but D, Goldberg and Jerrum (2010) have given a trichotomy for the Boolean domain.
- Can we be more precise about the complexity of strong balance? Is it equivalent to GI ? Is it in P (e.g. via a special case of GI ) ?
- Congruence singularity is decidable by equivalence to strong balance. But is there a direct proof?

