MATHS CSP

International Workshop on Mathematics of Constraint Satisfaction: Algebra, Logic and Graph Theory

20-24 March 2006 St Anne's College, University of Oxford

www.comlab.ox.ac.uk/mathscsp

Constraints & Algebra

What's the connection?

Peter Jeavons

Oxford University Computing Laboratory

Outline

- Constraint satisfaction problems
- Constraint languages
- Complexity of different languages
- Algebraic properties of constraint languages
- Polymorphisms and clones
- Clones and complexity
- Soft constraints and their complexity

Further reading…

- For more mathematics see:

Classifying the complexity of constraints using finite algebras
Andrei Bulatov, Peter Jeavons and Andrei Krokhin
Appears in: SIAM Journal on Computing **34**, (2005), pp. 720-742

- For more constraint satisfaction see:

The Complexity of Constraint Languages
David Cohen and Peter Jeavons
To appear in: Handbook of Constraint Programming

http://web.comlab.ox.ac.uk/oucl/research/areas/constraints/publications

Question

What do these problems have in common?

- Drawing up a timetable for a conference
- Choosing frequencies for a mobile-phone network
- Checking the satisfiability of a logical formula
- Fitting a protein structure to measurements
- Laying out components on a circuit board
- Finding a DNA sequence from a set of contigs
- Scheduling a construction project
- Solving a system of linear equations

Answer

What do these problems have in common?

- Drawing up a timetable for a conference
- Choosing frequencies for a mobile-phone network
- Checking the satisfiability of a logical formula
- Fitting a protein structure to measurements
- Laying out components on a circuit board
- Finding a DNA sequence from a set of contigs
- Scheduling a construction project
- Solving a system of linear equations

They all involve searching for a solution which satisfies a set of constraints

What is a constraint?

- A constraint has two parts:
  - A list of variables that are constrained, which we call the scope
  - A relation
    this relation specifies the
    allowed combinations of values
    for the variables in the scope

Examples

“Variables X and Y must be different colours”

Scope is <X,Y>

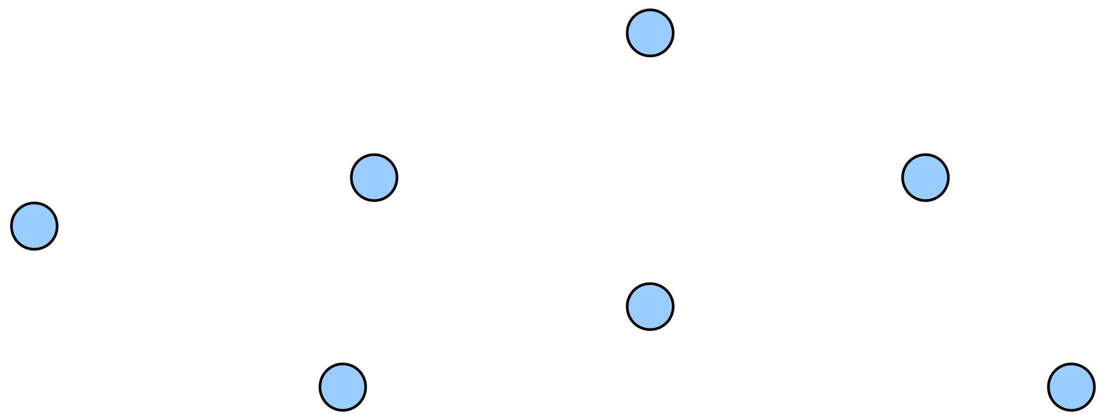
Relation is ≠

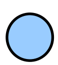
“I fell asleep during that tutorial”

Scope is <S,T>

Relation is “during”

An Abstraction



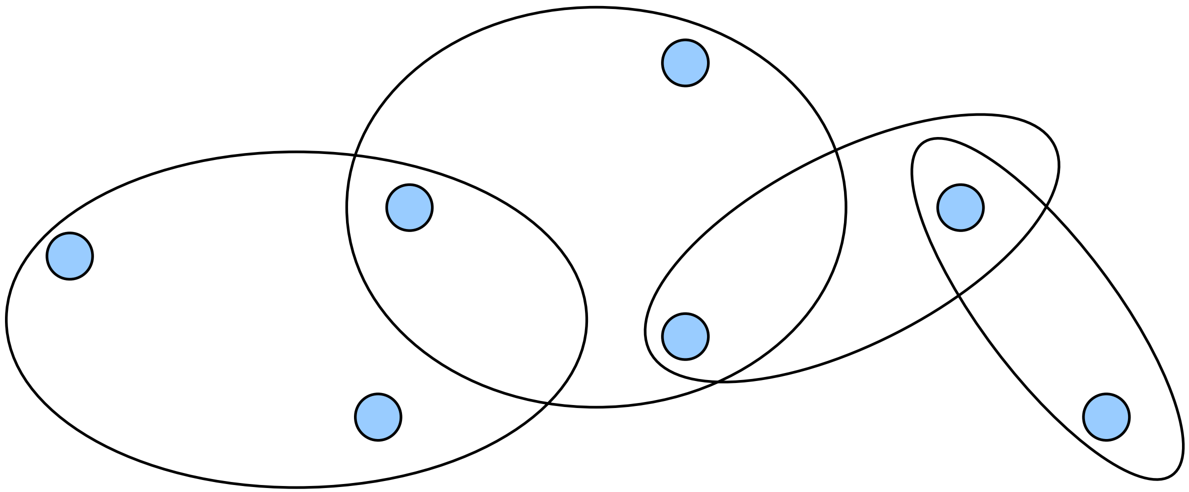
Variables  = Talks to be scheduled at conference

Transmitters to be assigned frequencies

Amino acids to be located in space

Circuit components to be placed on a chip

An Abstraction



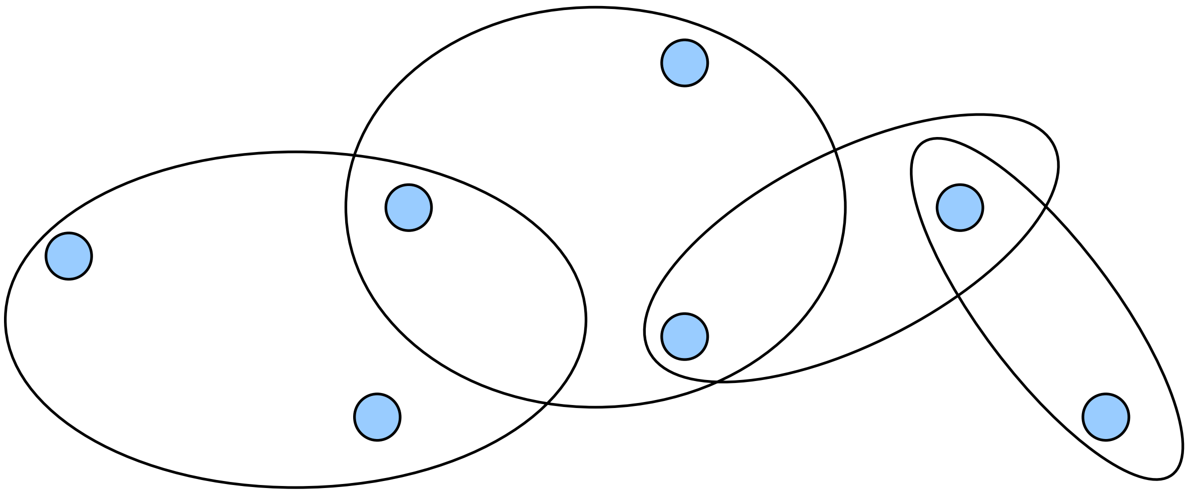
Constraints = All talks on logic on different days

No interference between near transmitters

$x + y + z > 0$

Foundations dug before walls built

The CSP



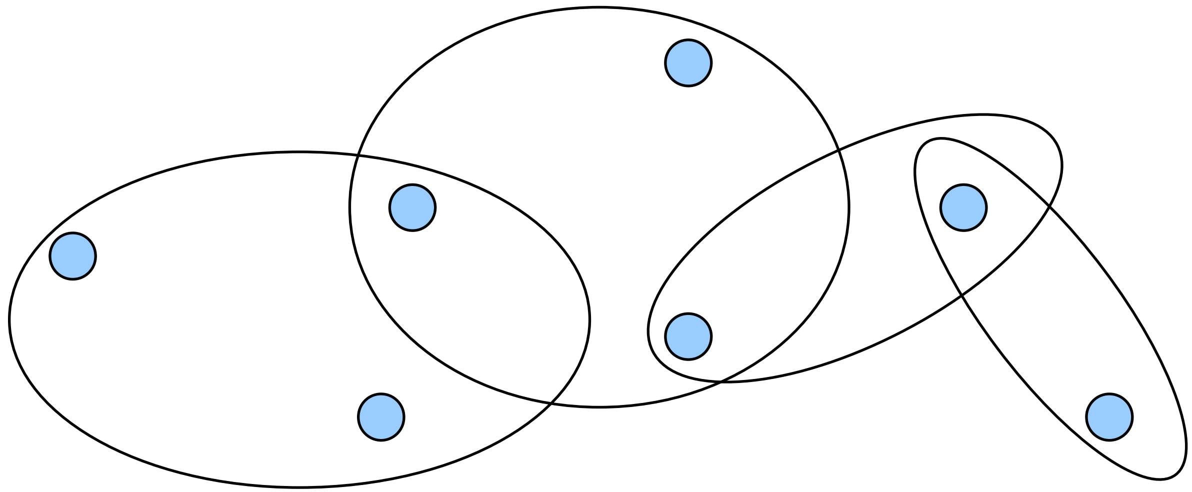
A set of constraints defines an instance of a *constraint satisfaction problem* (CSP)

General Question

- Having a general formulation for this kind of problem allows us to ask general structural questions:

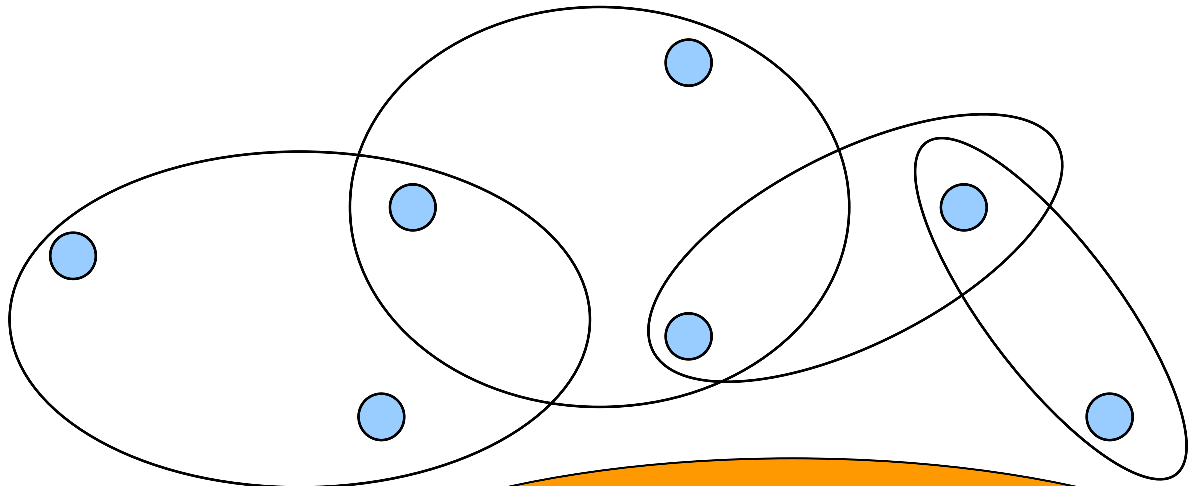
When is the CSP tractable?

Half of the Story...



- This picture illustrates the constraint *scopes*
- The set of scopes is sometimes called the *constraint hypergraph*, or the *scheme*
- A lot of work has been done on CSPs with restricted schemes (such as trees)

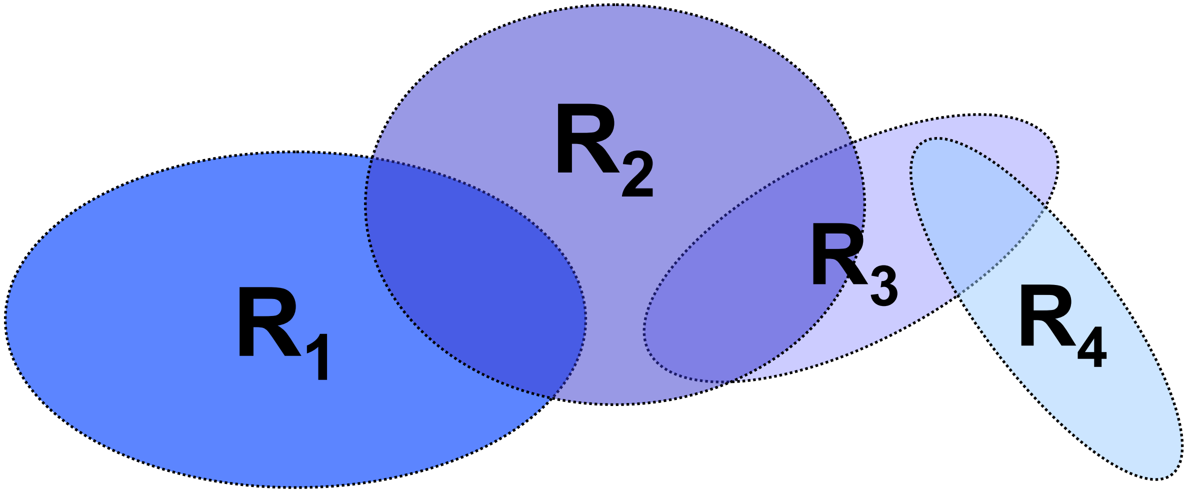
Half of the Story...



- This pictur … *apes*
- The set of … ne *constraint* hyp… *heme*
- A lot of work has been done on CSPs with restricted schemes (such as trees)

For more on this see talks by Georg Gottlob and Daniel Marx

...The Other Half



- The picture now emphasises the constraint *relations*

What do we call the set of constraint relations?

Constraint Languages

Definition: A *constraint language* is a set of relations over a finite set D.

For every constraint language, L, we have a corresponding class of problems, CSP(L)…

Definition of CSP(L)

Definition 1a:

- An *instance* of CSP(L) is a 3-tuple (**V**,**D**,**C**), where
  - **V** is a set of variables
  - **D** is a single domain of possible values
  - **C** is a set of constraints

    Each constraint in **C** is a pair (**s**,**R**) where
    - **s** is a *list of variables* defining the scope
    - **R** is a *relation* from L defining the allowed combinations of values
- The *question* is whether each variable in **V** can be assigned a value in **D** so that all constraints in **C** are satisfied

Definition of CSP(Γ)

Definition 1a:

- An *instance* of CSP(Γ) is a 3-tuple (**V**,**D**,**C**), where
  - **V** is a set of variables
  - **D** is a single domain of possible values
  - **C** is a set of constraints

    Each constraint in **C** is a pair (**s**,**R**) where
    - **s** is a *list of variables* defining the scope
    - **R** is a *relation* from Γ defining the allowed combinations of values
- The *question* is whether each variable in **V** can be assigned a value in **D** so that all constraints in **C** are satisfied

Alternative Definition of CSP(L)

Definition 1b:

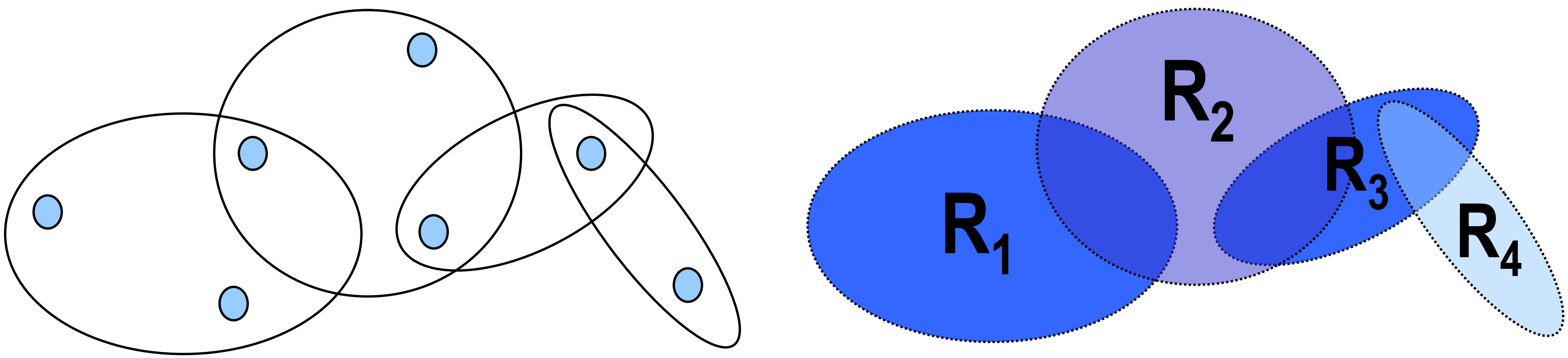
- An *instance* of CSP(L) is defined to be a first order formula:

$$\mathbf{F} = R_1(\underline{s}_1) \wedge R_2(\underline{s}_2) \wedge \ldots \wedge R_m(\underline{s}_m)$$

where each $R_i \in$ L

- The *question* is whether the formula can be satisfied by finding an assignment of values to the variables

Another Alternative View



$$(V,E_1,E_2,E_3,E_4) \longrightarrow (D,R_1,R_2,R_3,R_4)$$

Solution = Mapping from V to D such that
the image of each tuple related by E_i is related by R_i

Alternative Definition of CSP(L)

Definition 1c:

- An instance of CSP(L) is defined to be a pair of similar relational structures:

  $$(V,E_1,\ldots,E_m)\ ,\ (D,R_1,\ldots,R_m)$$

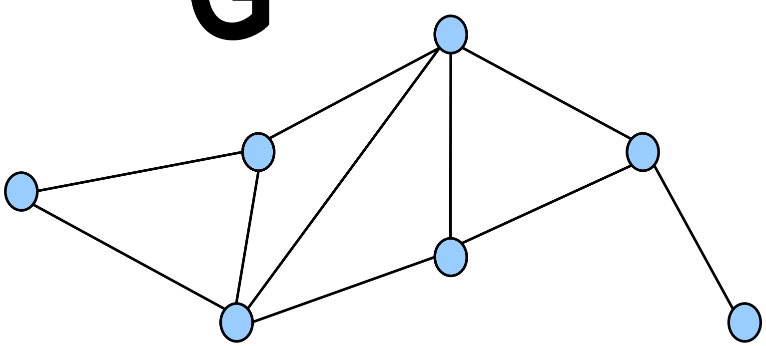
  where each $R_i \in L$

- The *question* is whether there exists a homomorphism from V to D

Example

- If L contains just the binary disequality relation, over the set D = {1,2,…,k}, then CSP(L) is the graph k-colouring problem

G



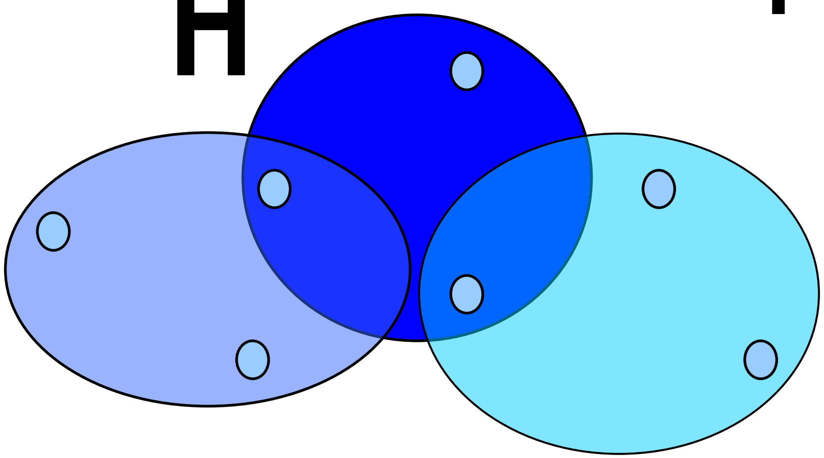
$$\mathbf{F} = (v_1 \neq v_2) \wedge (v_2 \neq v_5) \wedge \ldots \wedge (v_4 \neq v_7)$$

$$\underset{(V,E)}{\mathbf{G}} \xrightarrow{\ ?\ } \underset{(D,\neq)}{\mathbf{K_k}}$$

Example

- If L contains all Boolean relations defined by ternary clauses, then CSP(L) is the 3-satisfiability problem

H



$$\mathbf{F} = (v_1 \vee v_2 \vee \neg v_3) \wedge \ldots \wedge (\neg v_3 \vee \neg v_4 \vee v_7)$$

$$\begin{array}{ccc} \mathbf{H} & & \mathbf{C_3} \\ (V, E_1, \ldots, E_8) & \xrightarrow{\mathbf{?}} & (D, R_1, \ldots, R_8) \end{array}$$

Examples

- If L contains all relations defined by linear equations over some field K, then CSP(L) is the problem of solving simultaneous linear equations over K

- If L contains all relations R(t) over the reals, where $R(t) = \{(x,y) | x-y<t\}$, then CSP(L) is the problem of solving a "simple temporal problem"

Summary of Examples

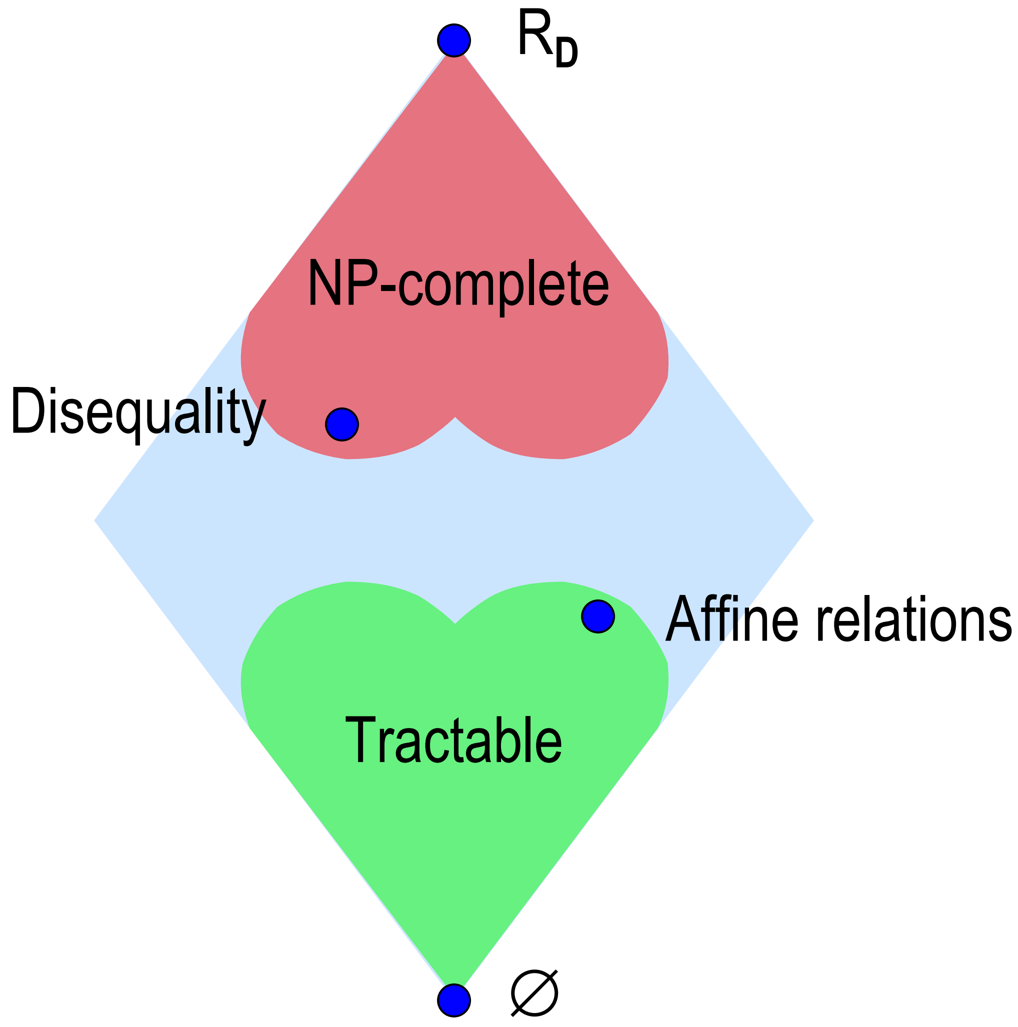
L	CSP(L)
Disequality Relation $\{\neq\}$	Graph Colouring Problem
Clauses	Satisfiability
Affine relations	Simultaneous Linear Equations
Temporal Relations $\{ (x,y) \mid x-y<t \}$	Simple Temporal Problems

NP-complete

Tractable

Complexity of CSP(L)

Languages



R_D
NP-complete
Disequality
Affine relations
Tractable
$\varnothing$

Complexity of CSP(L)

There have been a lot of papers investigating the complexity of CSP(L) for different sets of relations L:

- Schaefer, STOC'78, 2-valued domains
- Hell and Nesetril, JCTB, 1990, binary symmetric rels
- Feder and Vardi, STOC'93, 3 families – logic, groups
- Kirousis, AIJ, 1993, implicative constraints
- Cooper et al, AIJ, 1994, 0/1/all constraints
- Jeavons et al, JACM, 1997, algebraic theory
- Dalmau and Pearson, CP'99, set functions
- Cohen et al, JACM, 2000, disjunctive constraints
- Bulatov et al, STOC'01, maximal tractable sets
- Bulatov, FOCS'02, 3-valued domains

2-Valued Domain

$\boxed{\text{SAT}}$

$x + y + z = 0$

$$(x \vee y \vee \neg z) \wedge (x \vee \neg y \vee z) \wedge (\neg x \vee y \vee z) \wedge (\neg x \vee \neg y \vee \neg z) \equiv$$

x	y	z	
0	0	0	✓
0	0	1	X
0	1	0	X
0	1	1	✓
1	0	0	X
1	0	1	✓
1	1	0	✓
1	1	1	X

Complexity – Boolean Case

Schaefer (1978) showed that when L is a set of Boolean relations, CSP(L) is tractable in exactly the following 6 cases:

- Every R in L contains (0,0,…,0)
- Every R in L is definable by a CNF formula in which each conjunct has at most one un-negated literal (Horn clauses)
- Every R in L is definable by a CNF formula in which each conjunct has at most 2 literals
- Every R in L contains (1,1,…,1)
- Every R in L is definable by a CNF formula in which each conjunct has at most one negated literal (dual Horn)
- Every R in L holds over an affine set in GF(2)

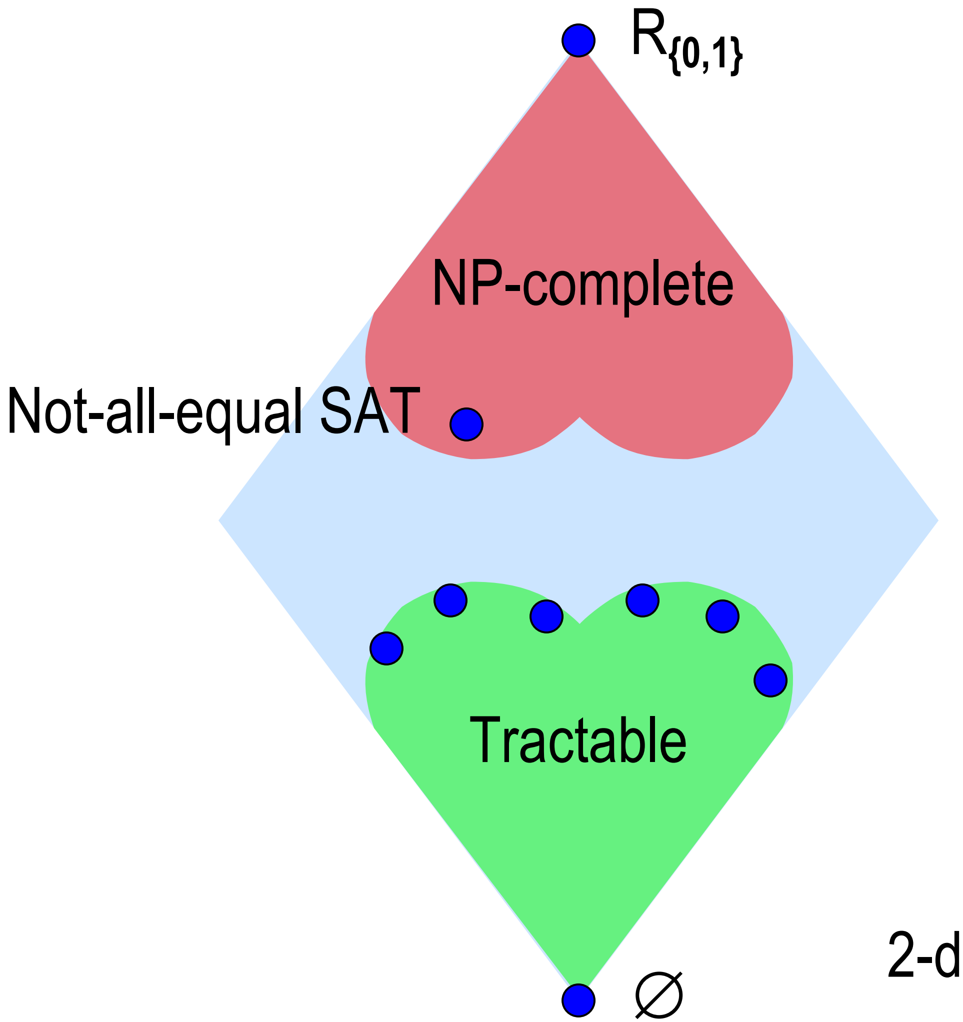
Complexity – Boolean Case

Schaefer (1978) showed that when L is a set of Boolean relations, CSP(L) is tractable in exactly th… …llowing 6 cases:

- Every R in L contains (0,0,… 0…
- Every R i…
  CNF fo…
  conjunct has at…
  un-negated literal (Horn clauses)
- Every R in L is definable by a CNF formula in which each conjunct has at most 2 literals
- Every R in L contains (1,1,…,1)
- … L is definable by a … which each … has at most one negated literal (dual Horn)
- Every R in L holds over an affine set in GF(2)

For more on this see
talks by Nadia Creignou
and Heribert Vollmer

Boolean Languages



0…0 relations

1…1 relations

Horn relations

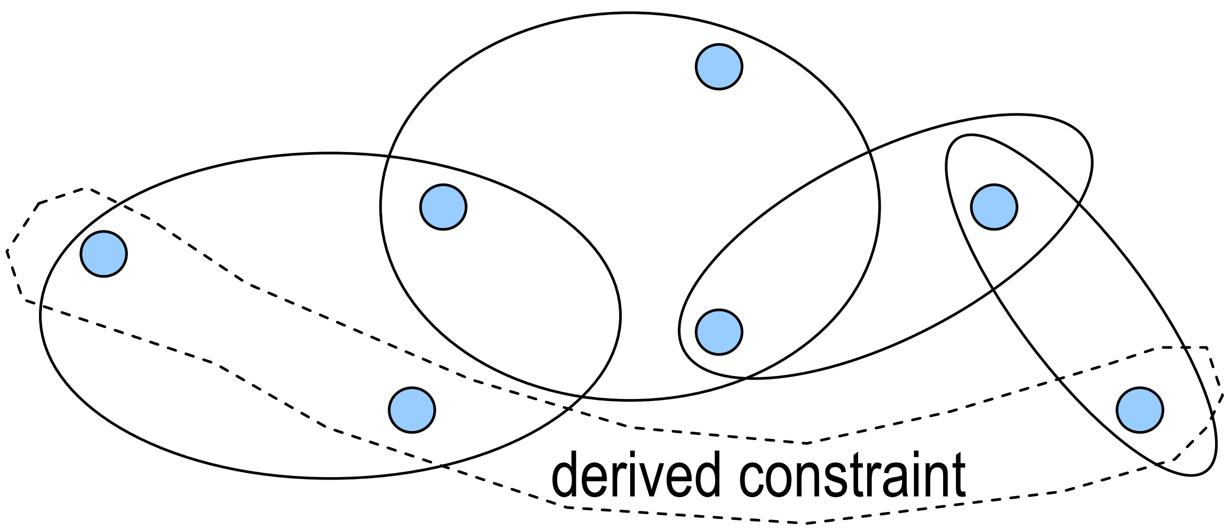
Dual Horn relations

2-decomposable relations

Affine relations

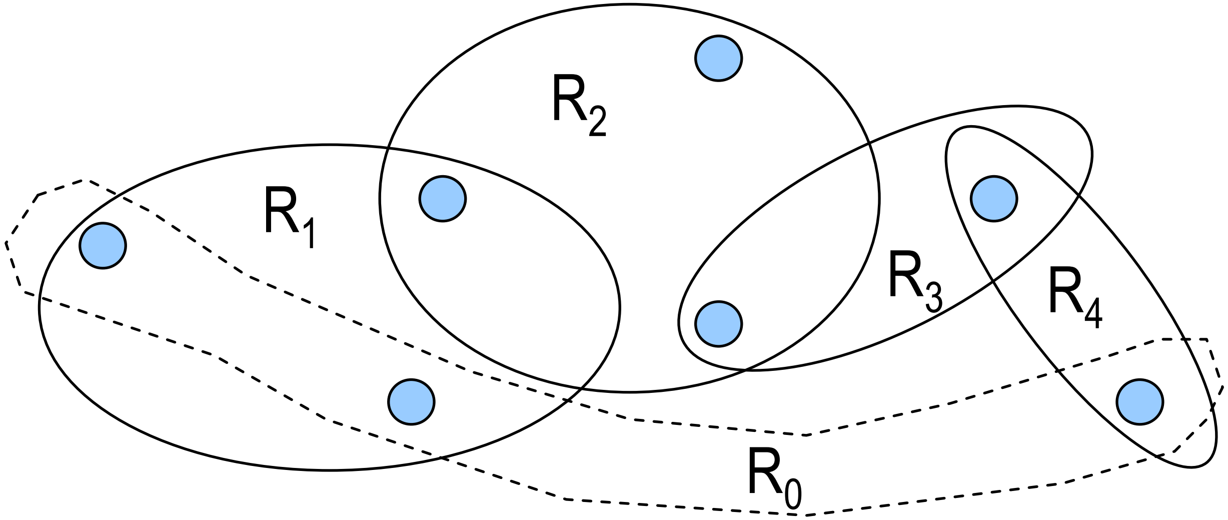
Expressive Power

- The idea of Schaefer's proof was to consider what relations are "expressible" using relations from L
- This makes use of the fact that new constraints can be derived from the combined effect of specified constraints

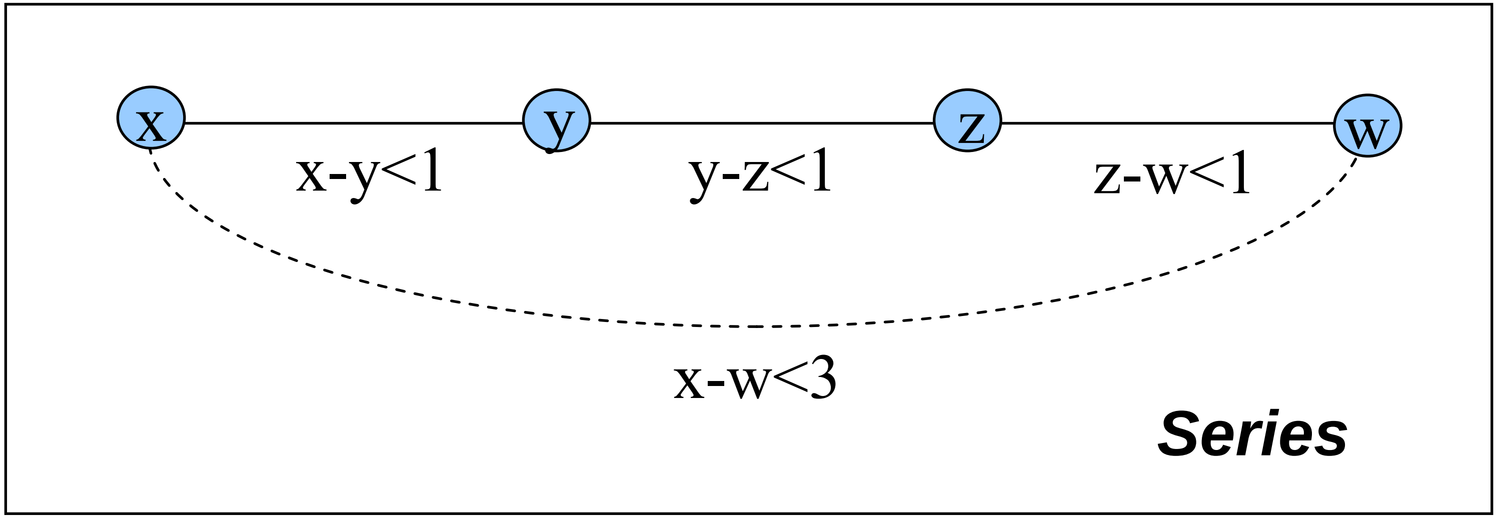


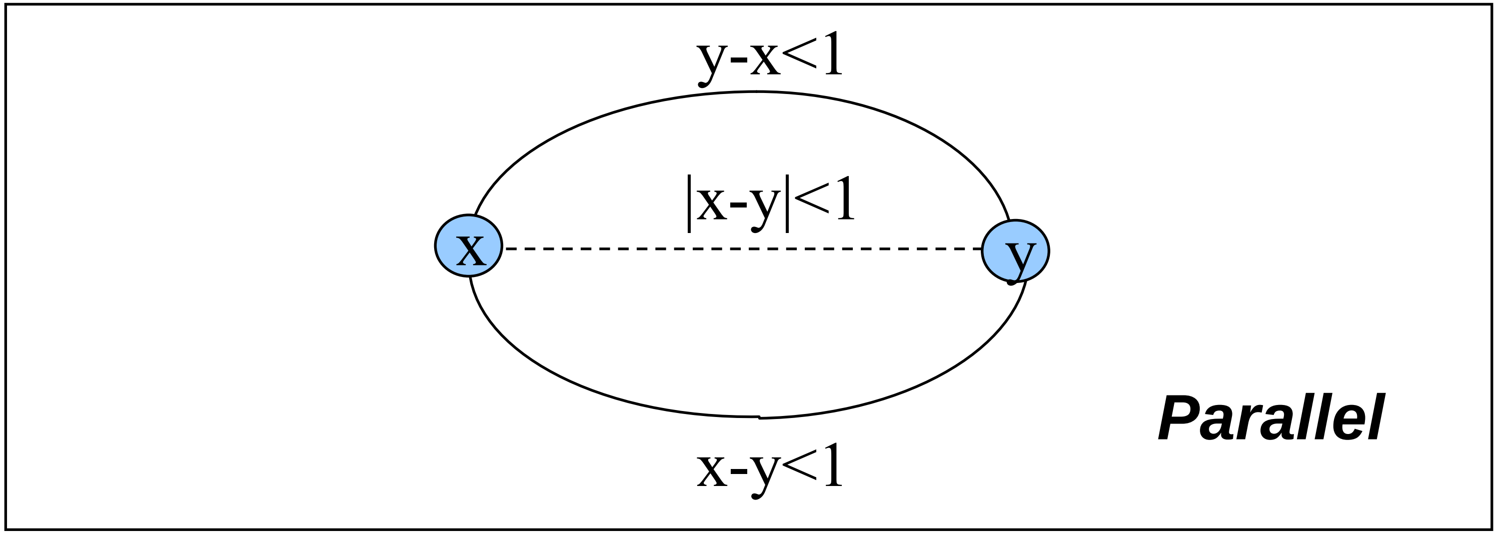
Expressive Power

- If we can combine the relations $R_1, R_2, \ldots, R_k$ to obtain a derived constraint relation R_0, then we say that R_0 can be *expressed* using $R_1, R_2, \ldots, R_k$



Two (Binary) Constructions

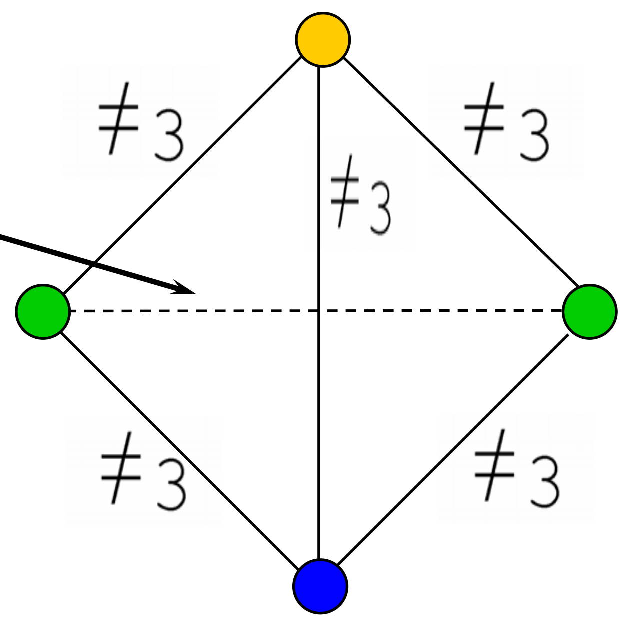




Example

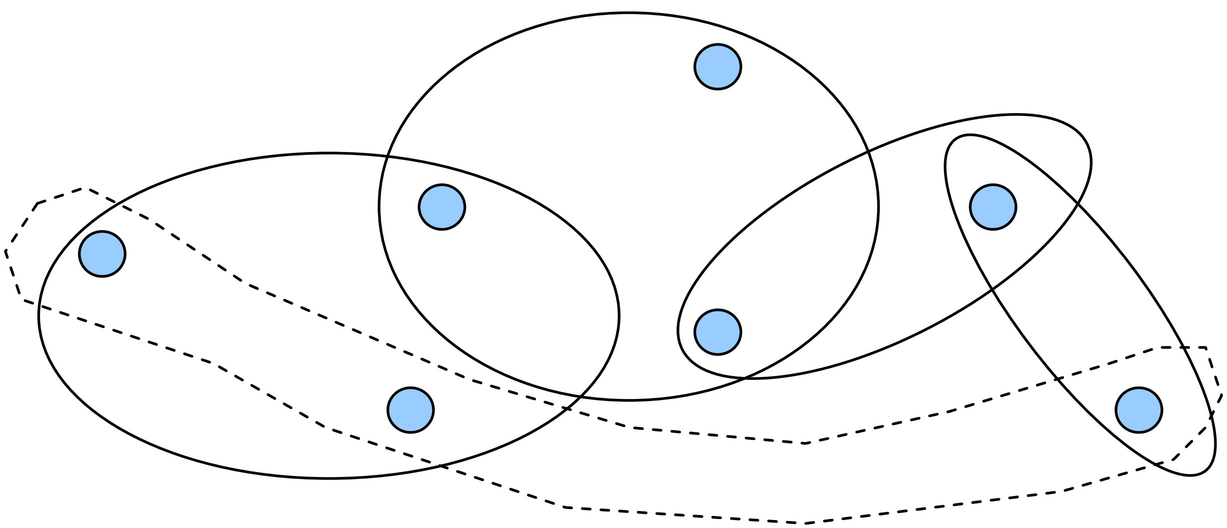
- What constraints can be expressed using just the disequality relation on a 3 element domain?

What is the derived constraint here?



General Constructions

- To include all possible constructions we need to allow arbitrary *join* operations, followed by a *projection* operation
- Then we can construct *any* derived constraint



Expressive Power

Definition 2a:

The "expressive power" of a constraint language L, denoted ⟨L⟩, is defined to be the set of relations that can be expressed using:

- Relations in L
- **Relational join** operations
- **Projection** onto some subset of variables

Expressive Power

Definition 2b:

The "expressive power" of a constraint language L, denoted $\langle L \rangle$, is defined to be the set of relations that can be expressed using:

– Relations in L

– The equality relation over the domain of L

– **Conjunction**

– **Existential quantification**

Properties of Languages

- If we have very few relations in our language, L, then it may be impossible to *describe* the problem we want to solve
- If we have too many relations in our language, L, then CSP(L) may be *intractable*
- There is a trade-off between expressive power and tractability

Expressive Power and Reduction

Theorem: For any constraint language L,
and any finite constraint language L', if $L' \subseteq \langle L \rangle$
then $CSP(L')$ is polynomial-time reducible to $CSP(L)$

Expressive Power and Reduction

Theorem: For any constraint language L,
and any finite constraint language L', if $L' \subseteq \langle L \rangle$
then $CSP(L')$ is polynomial-time reducible to $CSP(L)$

Corollary: We can add any of the relations in $\langle L \rangle$ to L without changing the complexity of $CSP(L)$.

Corollary: If $\langle L_1 \rangle = \langle L_2 \rangle$ then
$CSP(L_1)$ is polynomial-time equivalent to $CSP(L_2)$.

Expressive Power and Reduction

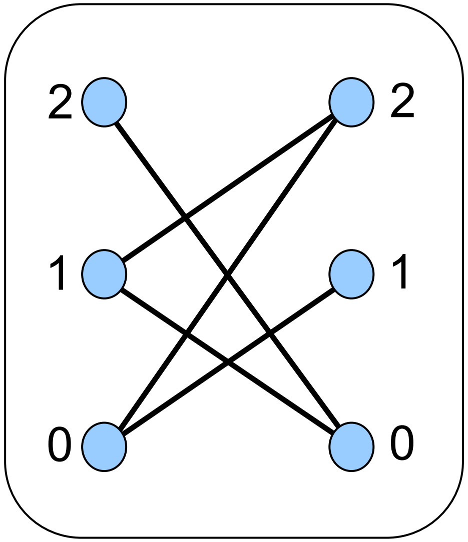
Theorem: For any constraint language L,
and any finite constraint language L', if $L' \subseteq \langle L \rangle$
then $CSP(L')$ is polynomial-time reducible to $CSP(L)$

$\langle L \rangle$ is more important than L

Example

Consider the language consisting of a single binary relation ρ_0 over D = $\{0,1,2,\ldots,n\}$, where ρ_0 contains the following tuples:

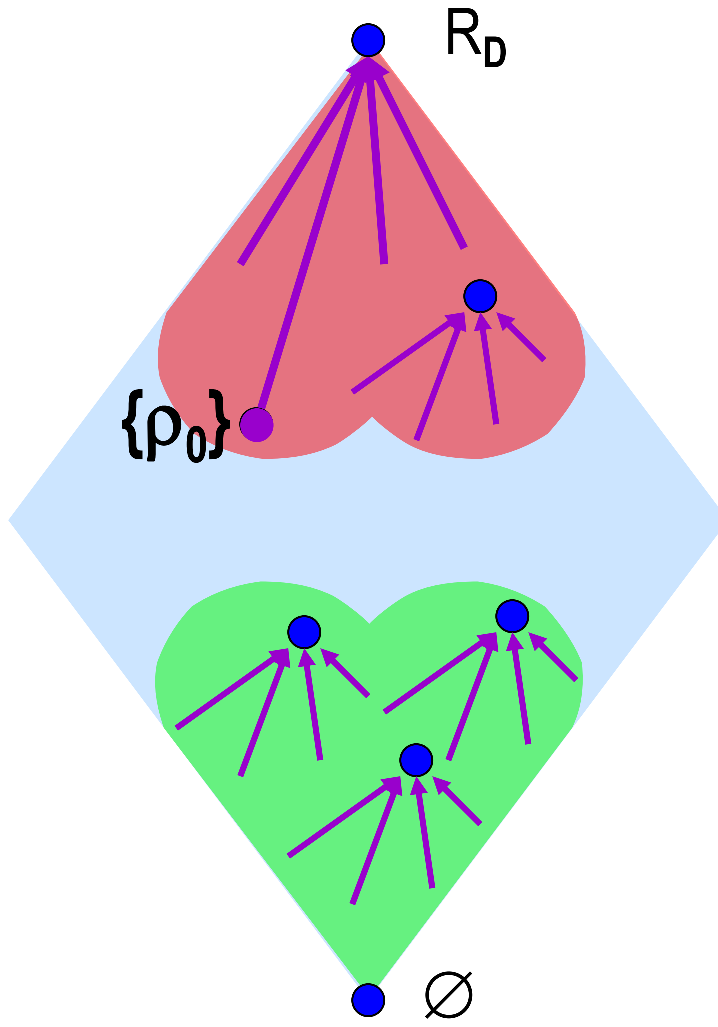
- $(i,i+1)$ for all $1 \leq i \leq n-1$
- $(0,i)$, $(i,0)$ for all $1 \leq i \leq n$



It is known that $\langle\{\rho_0\}\rangle$ contains all possible relations over D

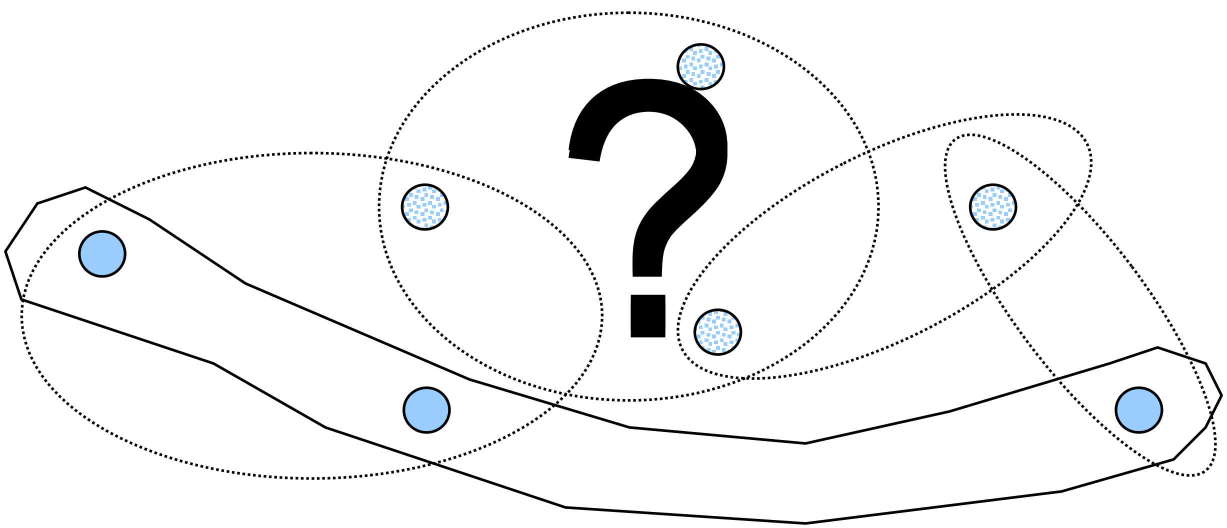
Hence CSP($\{\rho_0\}$) is NP-complete

Closure



Calculating ⟨L⟩

- A relation is in ⟨L⟩ if and only if it can be expressed *somehow* using the relations in L
- For a given relation, how can we decide if it can or cannot be expressed in L?



Polymorphisms and Clones

Algebraic Invariance

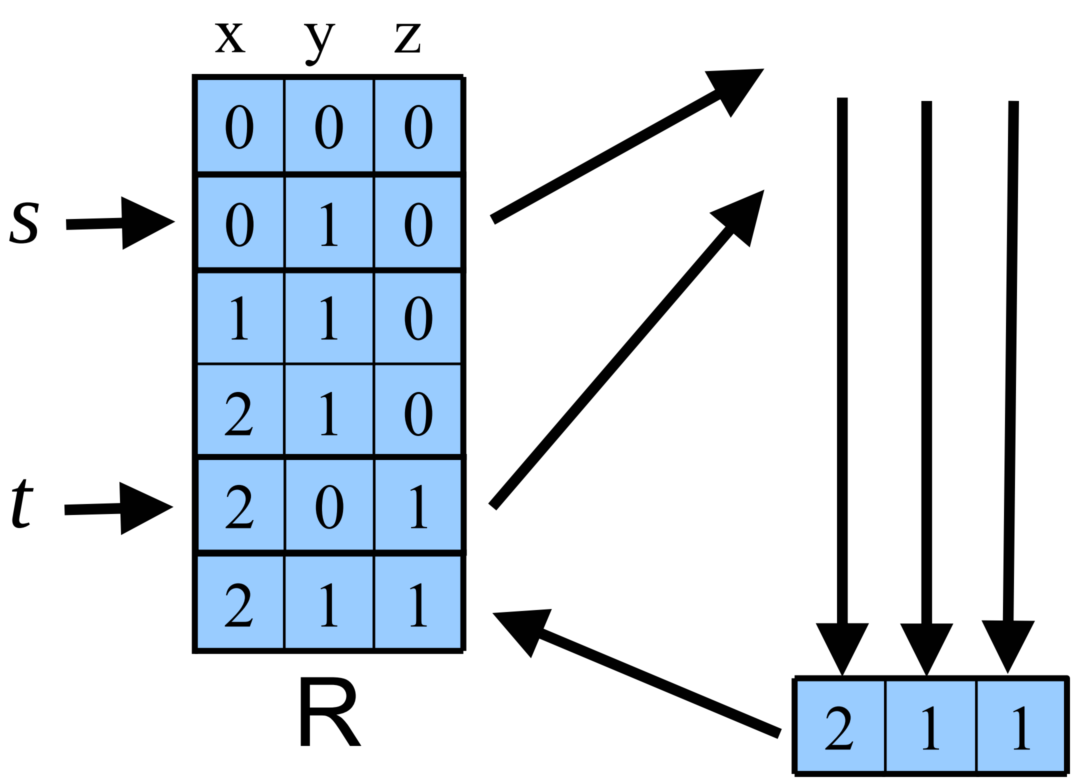
Definition: A relation R is *invariant* under a k-ary operation ϕ, if, for any tuples $\underline{a}_1,\underline{a}_2,\ldots,\underline{a}_k \in$ R, the tuple obtained by applying ϕ co-ordinatewise is a member of R .

If R is invariant under ϕ,

then ϕ is called a *polymorphism* of R.

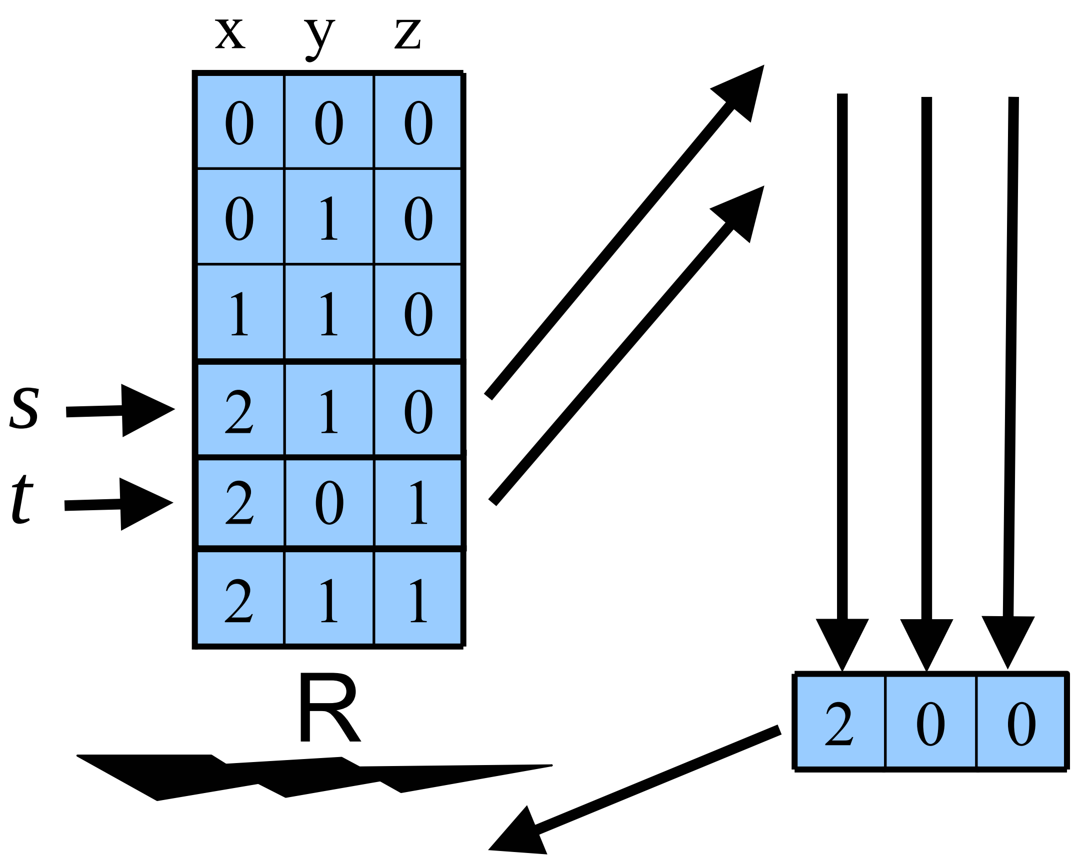
Example of Polymorphism

$\forall s,t$ if s and t are in R, then Max(s,t) is in R



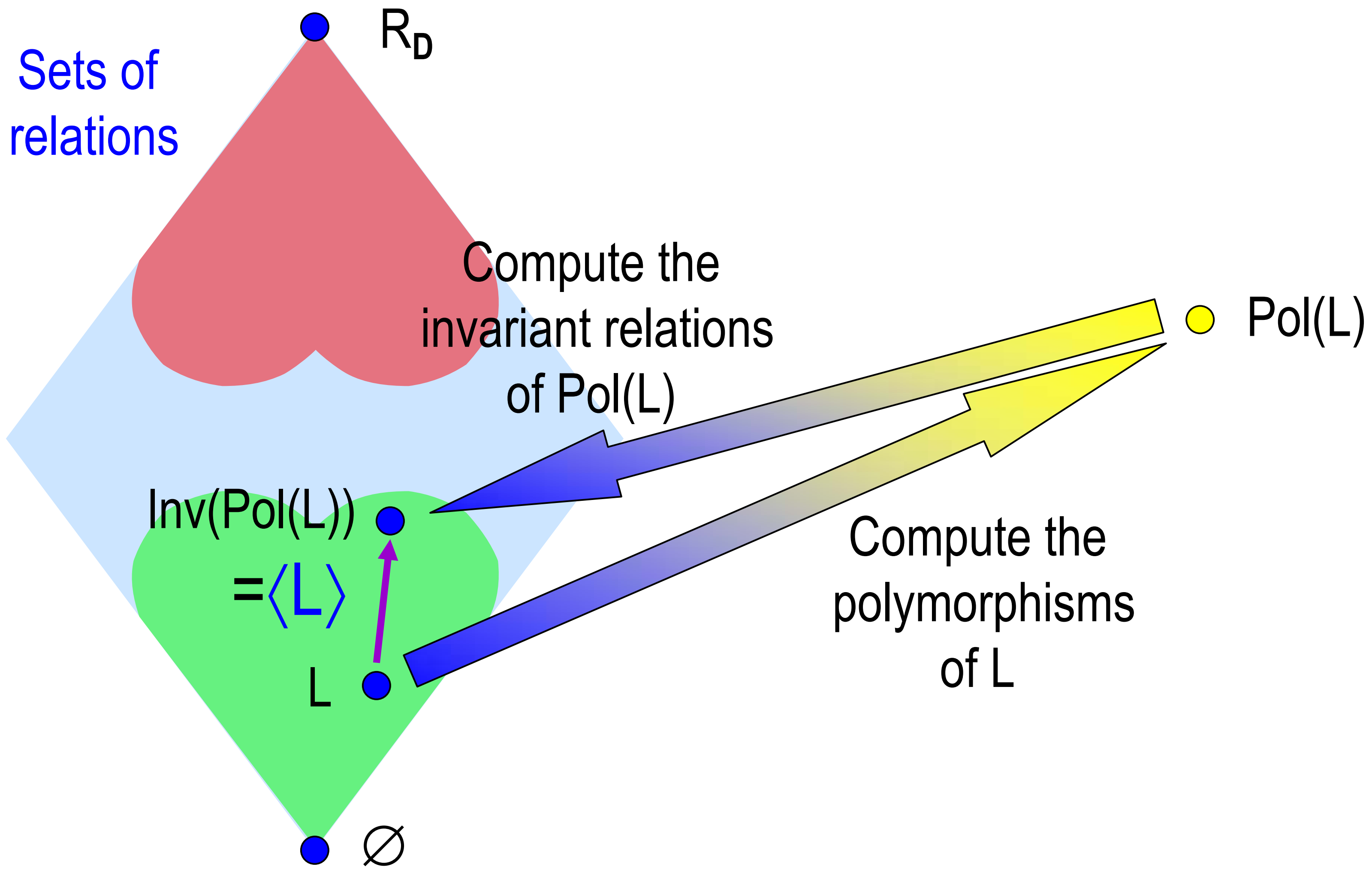
We say that this relation R has the **polymorphism** Maximum

Example of non-Polymorphism



We say that this relation R ***doesn’t*** have the **polymorphism** Minimum

Pol and Inv



R_D
Sets of relations
Compute the invariant relations of Pol(L)
Pol(L)
Inv(Pol(L))
$=\langle L \rangle$
Compute the polymorphisms of L
L
$\varnothing$

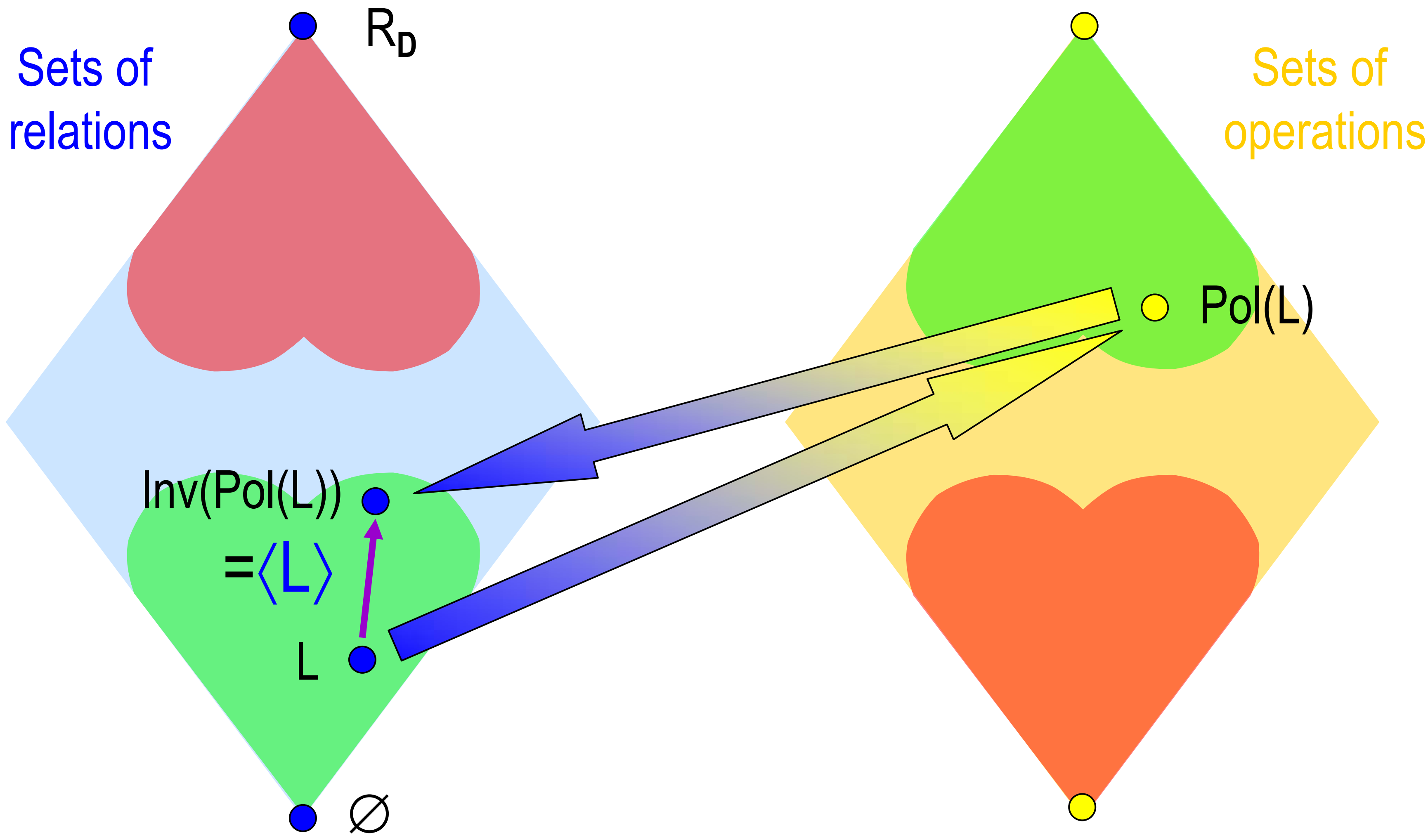
Expressive Power

Theorem (Geiger): For any constraint language L, over a finite domain, $\langle L \rangle$ = Inv(Pol(L))

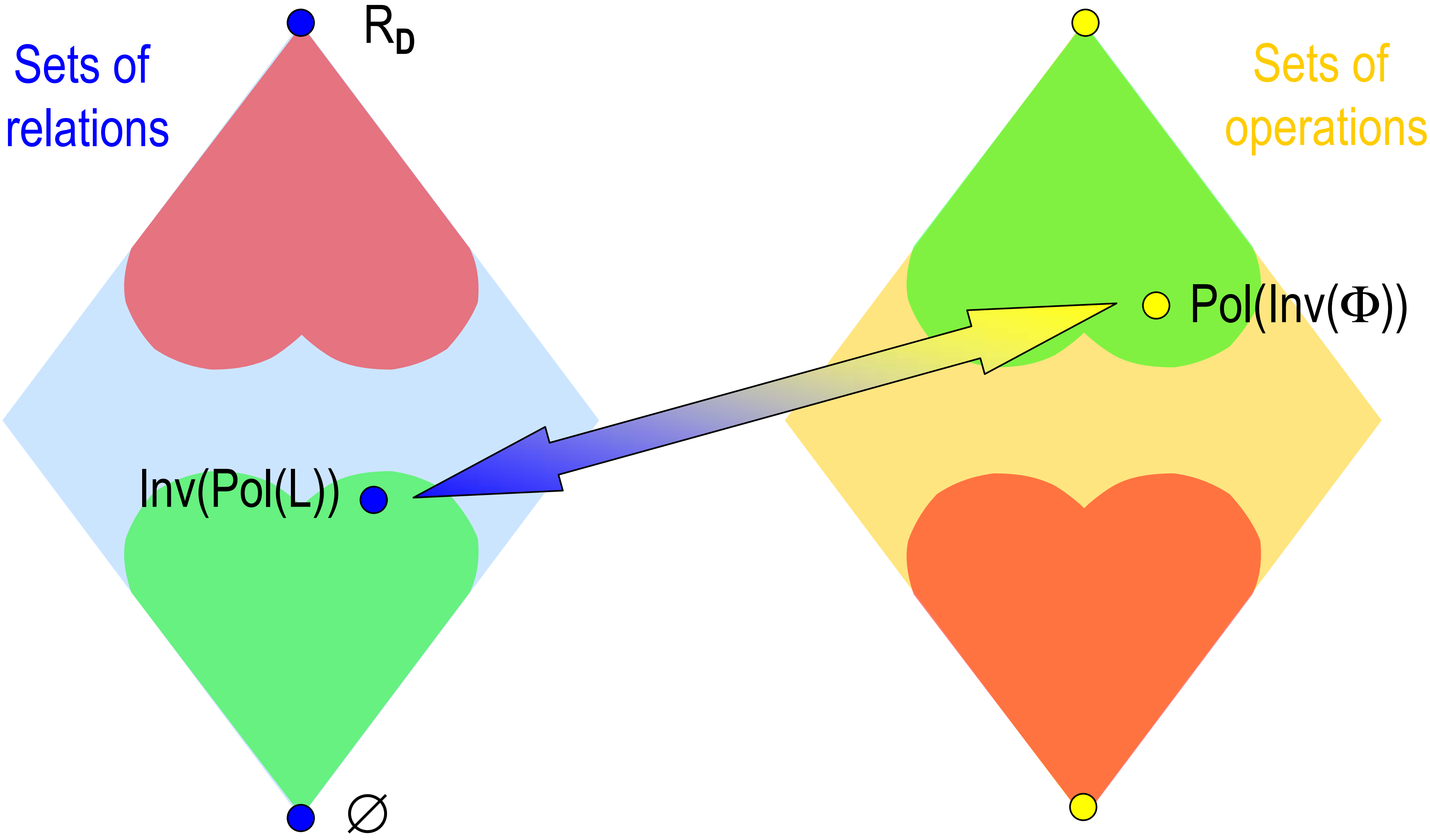
and independently by **Bodnarchuk, Kaluzhnin, Kotov** and **Romov**

Corollary: For any finite constraint language L, over a finite domain, the complexity of CSP(L) is determined by Pol(L)

Galois Connection



Galois Connection



Clones

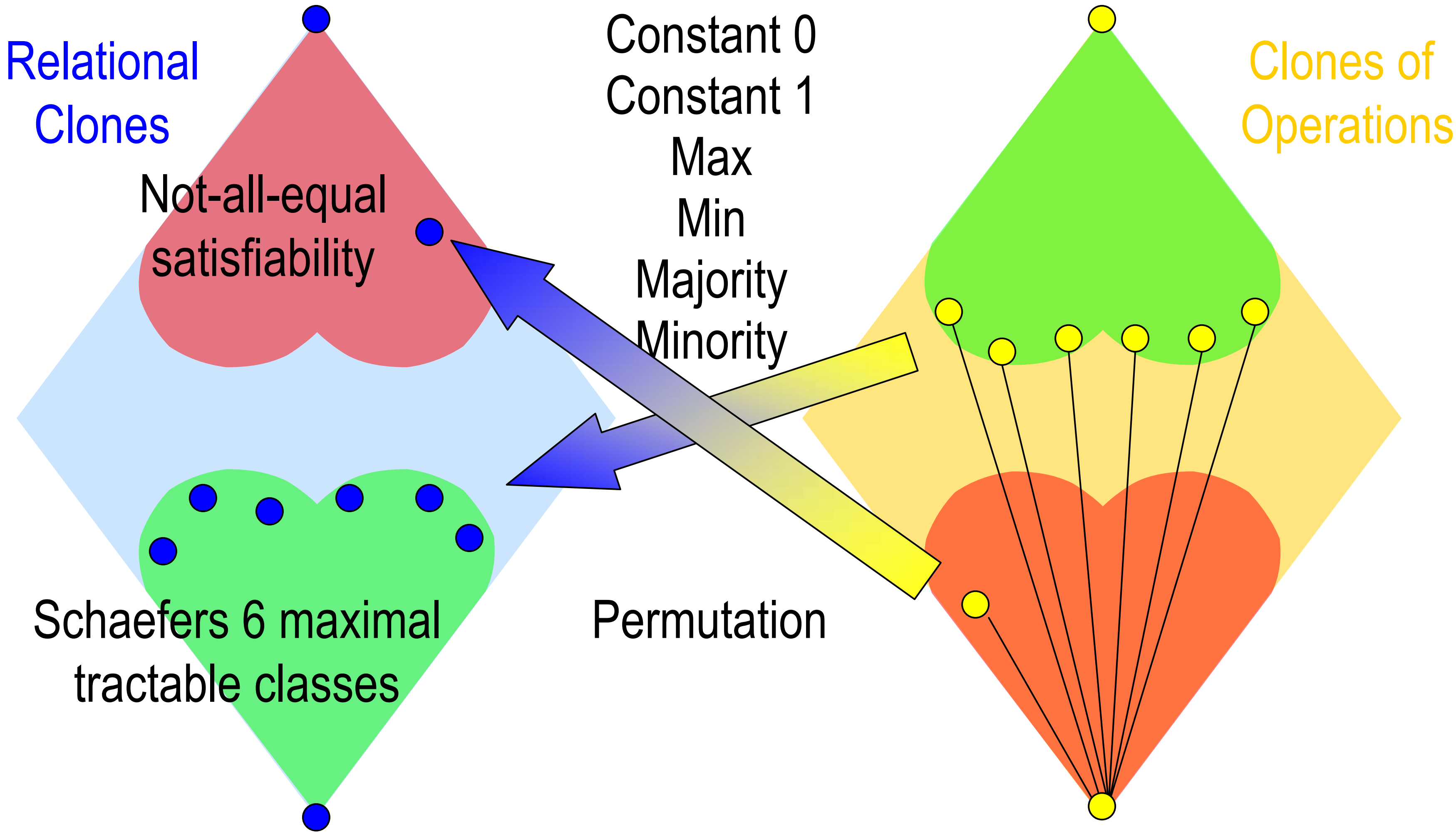
Definition: A *relational clone* is a set of relations which is closed under relational join and projection.

Definition: A *clone* is a set of operations which is closed under composition and contains all projection operations.

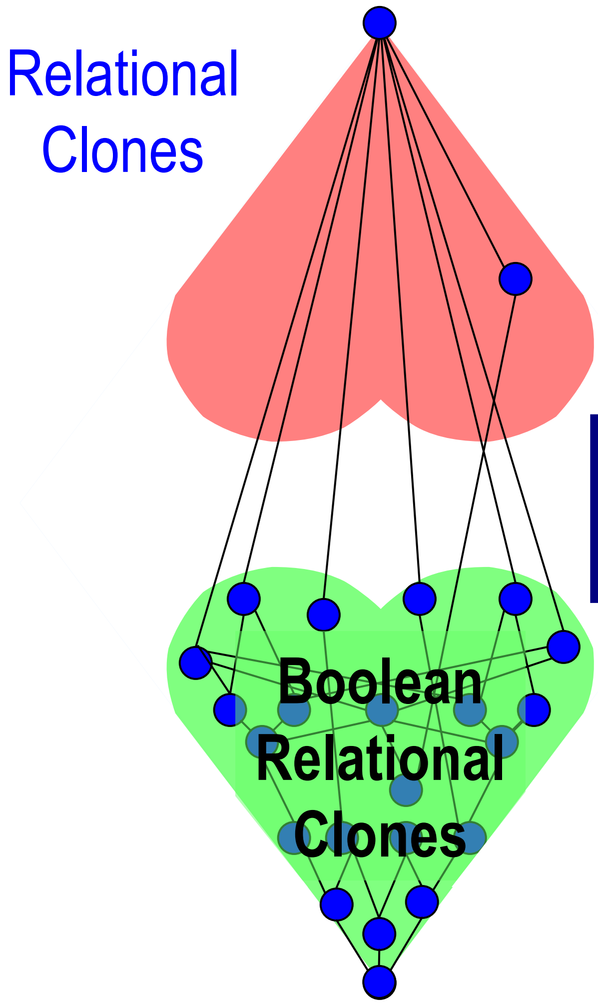
Every relational clone is of the form Inv(Φ) for some Φ

Every clone is of the form Pol(L) for some L

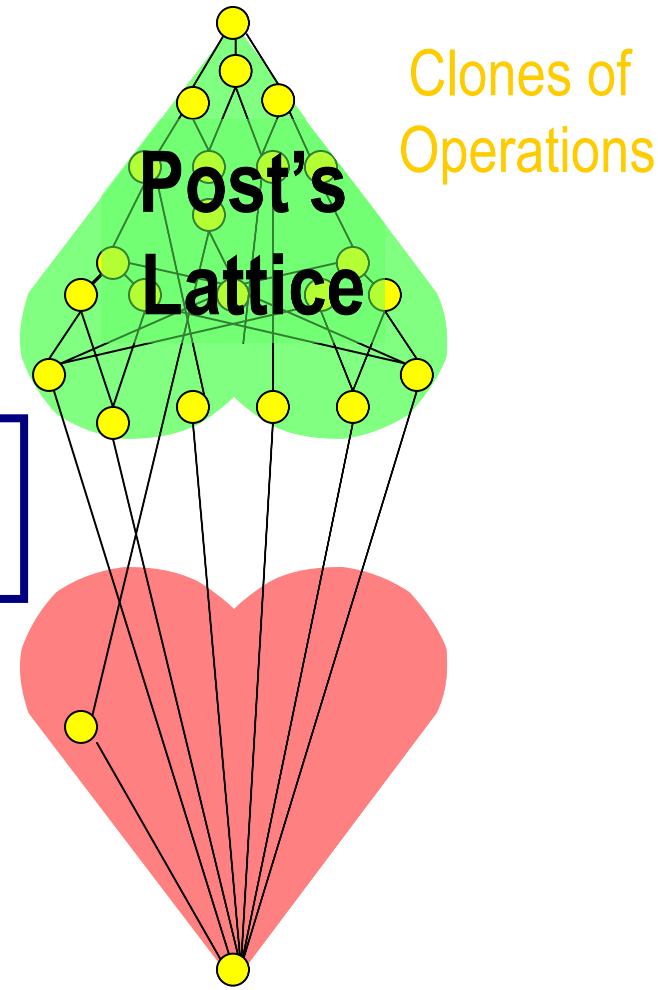
Boolean Operations



Boolean Operations



Dichotomy Theorem
for Boolean CSP



General Case

For domains larger than 2, the lattice of clones is uncountable, and not yet fully characterised…

Theorem (Rosenberg): Every *minimal* clone over a finite set is generated by one of the following:

- A constant operation;
- A ternary affine operation;
- A ternary majority operation;
- A non-identical unary operation;
- A semiprojection;
- A binary idempotent operation.

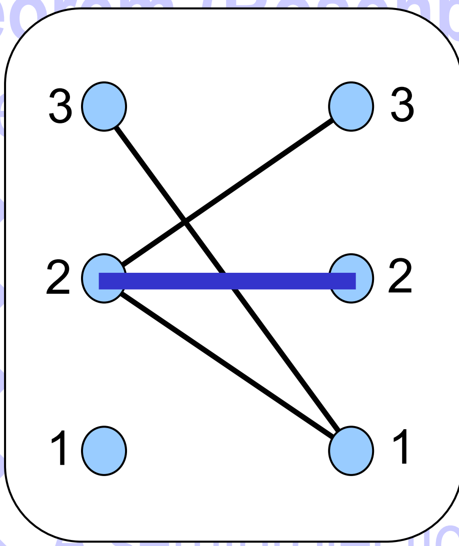
Constant Operations

What kinds of relations are invariant under

e.g. $c(x,y,z) = 2$?

Theorem (Rosenberg): Every *minimal* clone over a finite ... ted by one of

- ...ration;
- ... operation;
- ...ity operation;
- ... unary operation
- A semiprojection;
- A binary idempotent operation.



c c

{ (3, 1),
(2, 3),
(2, 2),
(2, 1) }

(2, 2)

Such relations are tractable – just assign constant value to all variables

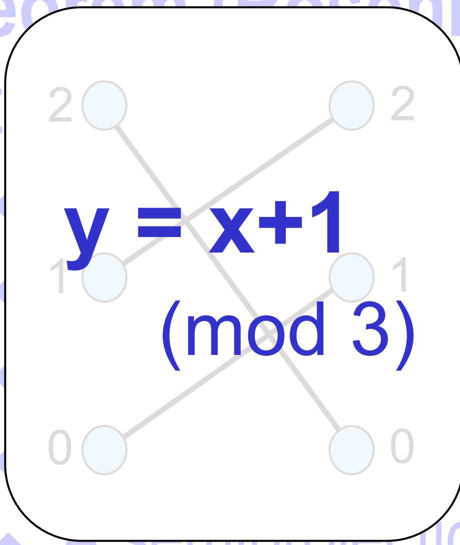
Affine Operations

What kinds of relations are invariant under

e.g. f(x,y,z) = x-y+z ?
(mod 3)

Theorem (Rosenberg): Every *minimal* clone over a finit... ...ted by one of

- ...ration;
- ...e operation;
- ...rity operation;
- ... unary operation
- A semiprojection;
- A binary idempotent operation.



f f

{ (2, 0),
(1, 2),
(0, 1) }

(1, 2)

Such relations are tractable – use Gaussian elimination on linear equations

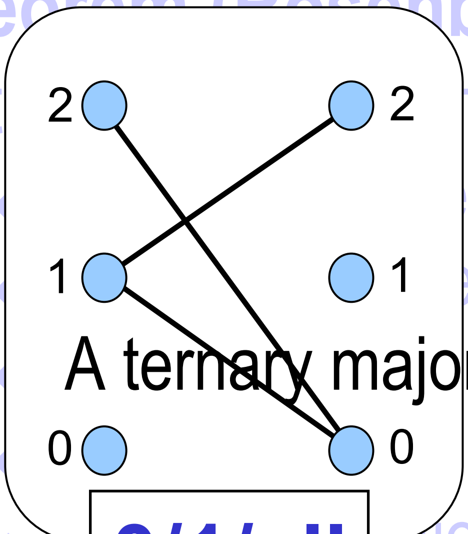
Majority Operations

What kinds of relations
are invariant under

e.g. $m(x,y,z) = \begin{cases} x & \text{if } x = y \\ z & \text{otherwise} \end{cases}$?

Theorem (Rosenberg): Every *minimal* clone over a finit… ted by one of

- … ration;
- … e ope…;
- A ternary majority operation;
- … unary operation…
- A … ction;
- A binary idempotent operation.



0/1/all

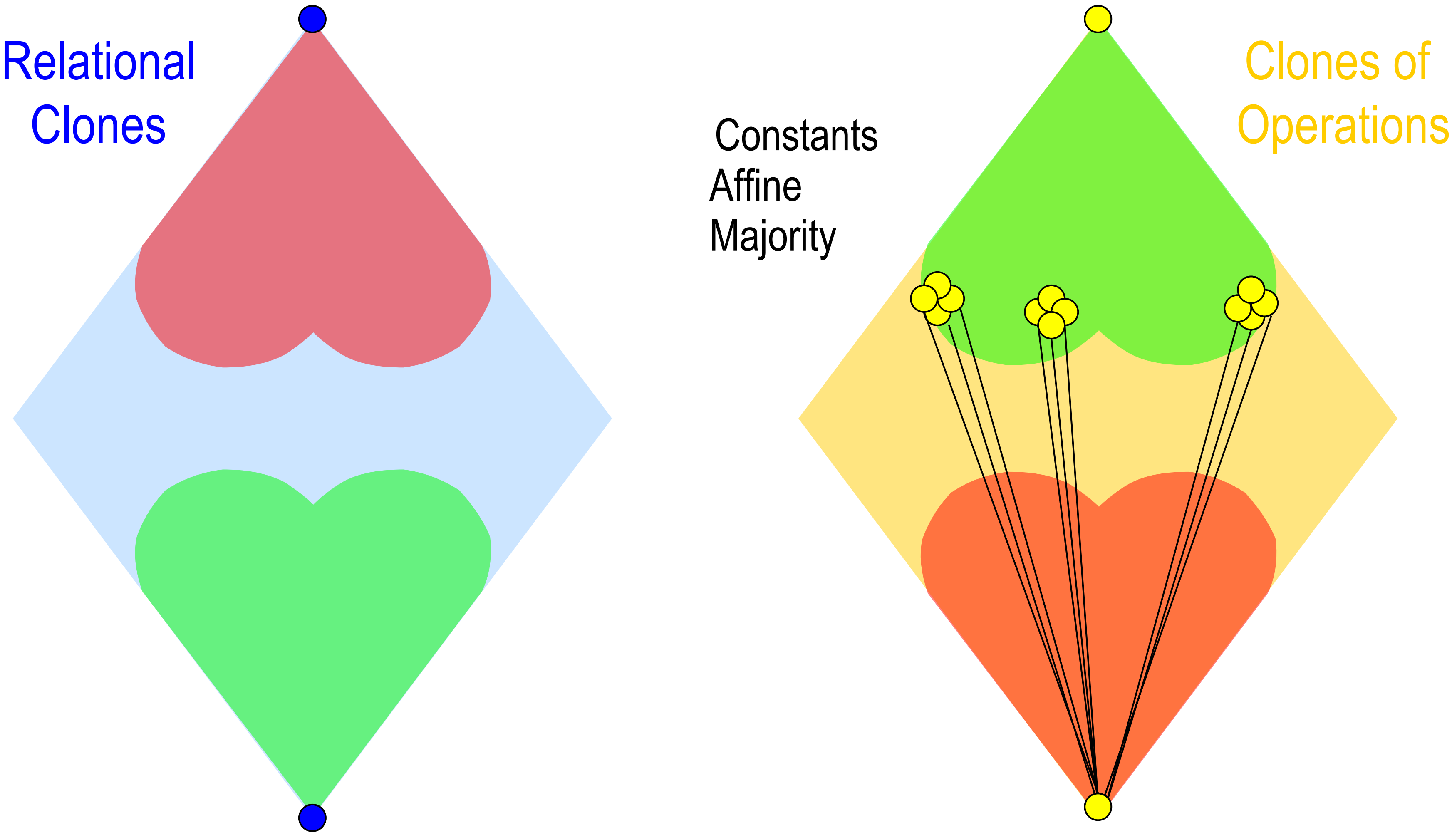
m m

{ (2, 0),
(1, 0),
(1, 2) }

(1, 0)

Such relations are tractable – use local consistency techniques

Galois Connection



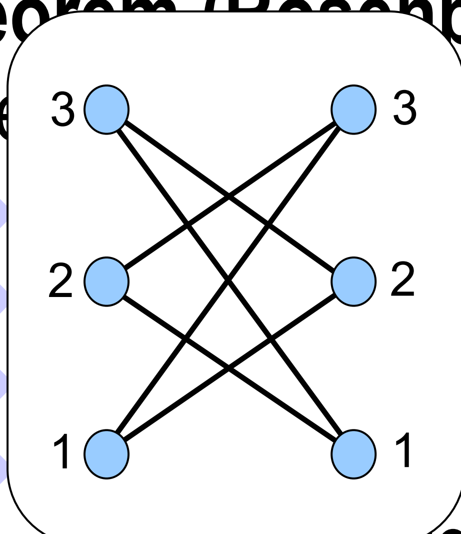
Unary Operations

What kinds of relations
are invariant under

e.g. $h(x) = 4 - x$?

Theorem (Rosenberg): Every r… over a
finite… ted by one of

- …ration;
- … operation;
- …rity operation;
- … **unary operation**
- A semiprojection;
- A binary idempotent operation.

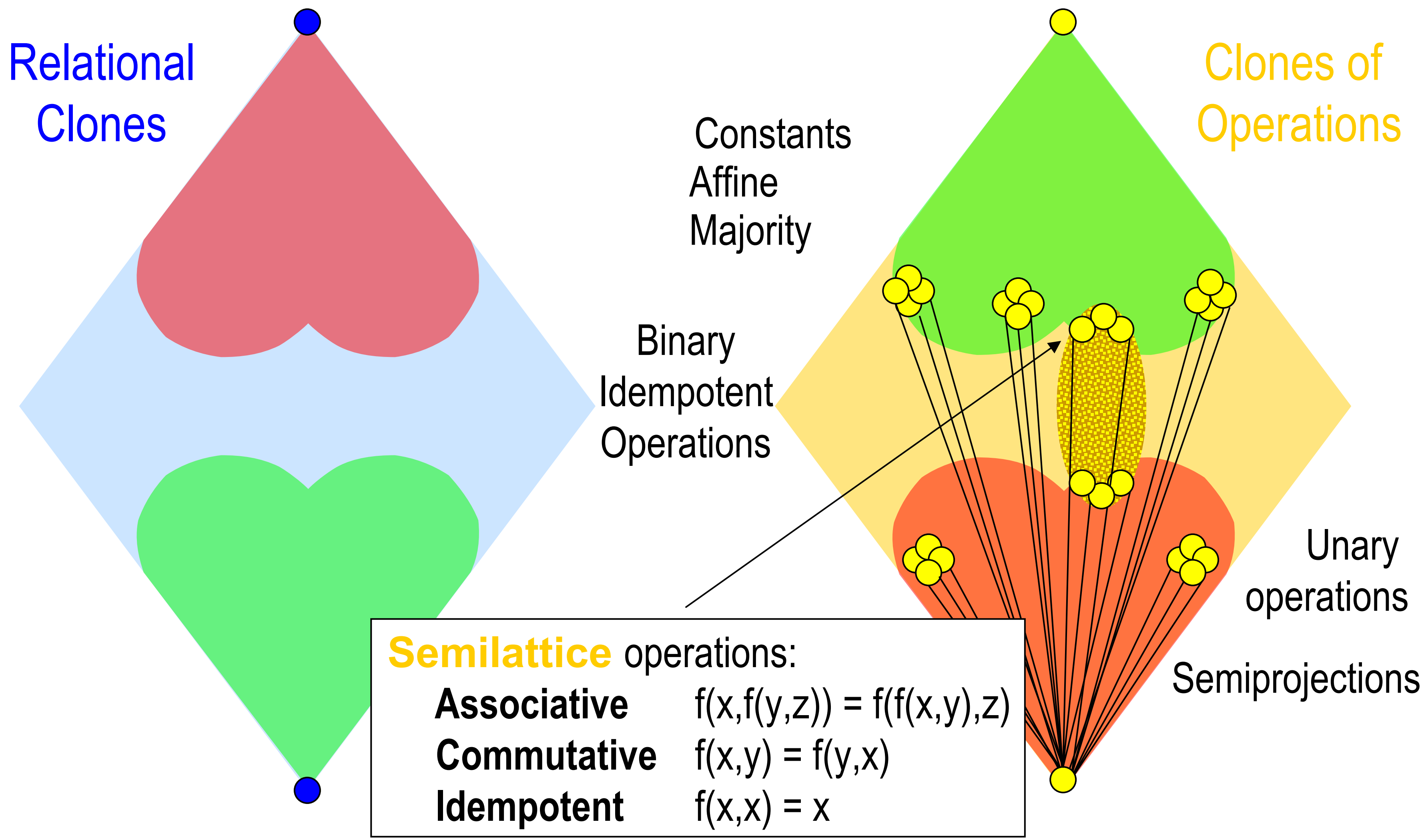



```
  h   h
{(3, 2),
 (3, 1),
 (2, 3),
 (2, 1),
 (1, 3),
 (1, 2) }

 (2, 1)
```

Such relations are NP-complete – equivalent to graph colouring

Galois Connection



Examples

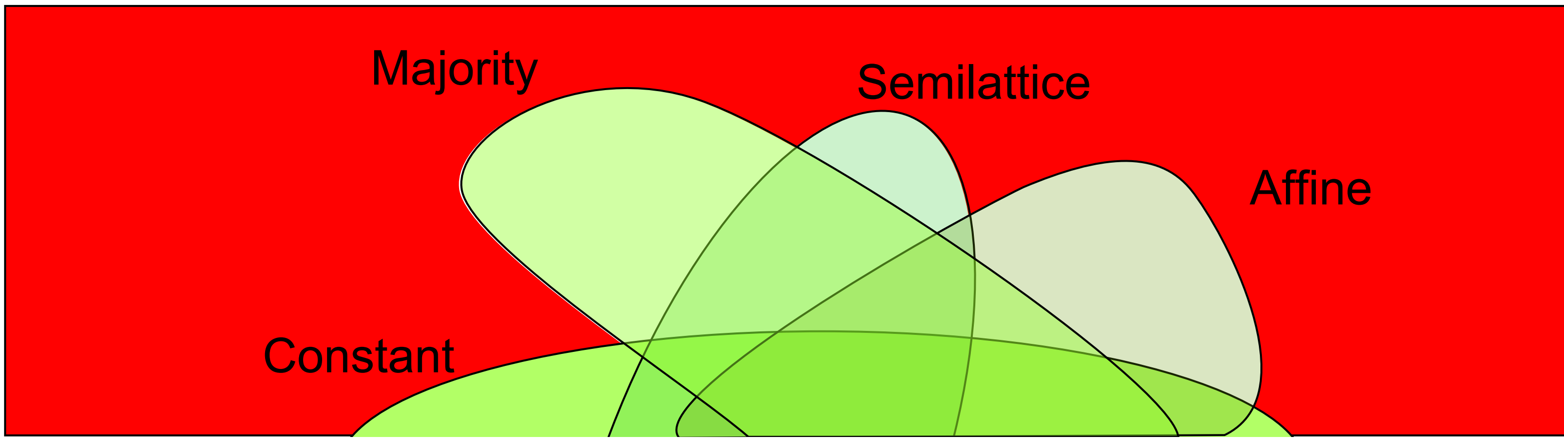
Any constraint language (over a finite ordered set) whose relations can be expressed as *a conjunction of pairwise disjunctions of upper/lower bounds* is invariant under a **majority operation** (median), and hence tractable.

For example, $\rho_2 = \{\, (x,y,z) \mid (x<3) \vee (y>5) \wedge (x>2) \vee (z<4) \,\}$

Any constraint language (over a finite ordered set) whose relations can be expressed as *a disjunction of upper bounds together with at most one lower bound* is invariant under a **semilattice operation** (maximum), and hence tractable.

For example, $\rho_3 = \{\, (x,y,z) \mid (x<3) \vee (y<5) \vee (z>4) \,\}$

Islands of tractability



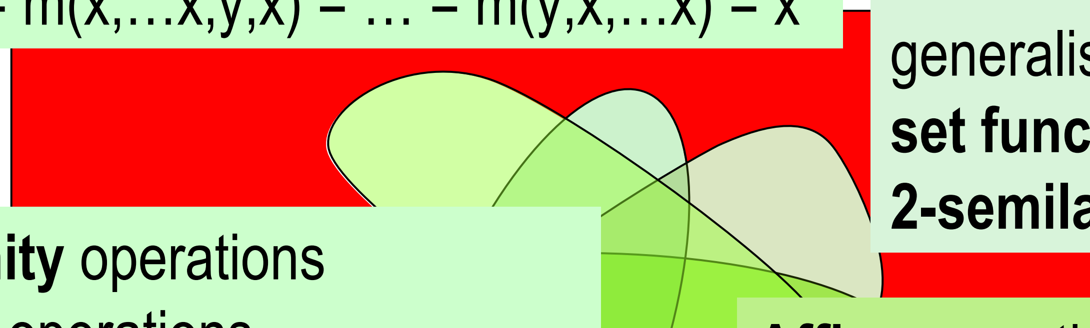
- In the Boolean case this is a complete description
  (2 constants, 1 majority, 2 semilattice, 1 affine)

- For larger domains this is **not** a complete description…

Islands of tractability

Majority operations have been generalised to **near-unanimity operations** where $m(x,x,\ldots,x,y) = m(x,\ldots x,y,x) = \ldots = m(y,x,\ldots x) = x$

Semilattice operations have been generalised to **set functions** and **2-semilattices**



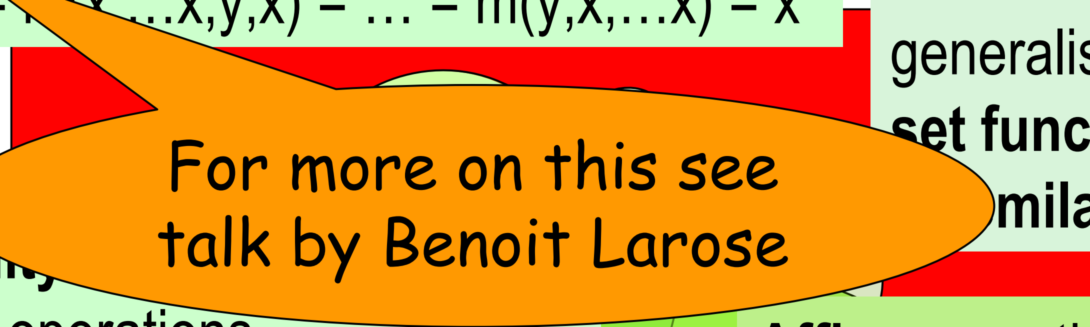
Near-unanimity operations and **Mal'tsev** operations have both been generalised further to **majority/minority operations** where *on each 2-element subset* the operation is either majority or Mal'tsev (Dalmau, LICS 2005)

Affine operations have been generalised to **Mal'tsev** operations where $f(y, y, x) = f(x, y, y) = x$

Islands of tractability

Majority operations have been generalised to **near-unanimity operations** where m(x,x,…,x,y) = m(x,…x,y,x) = … = m(y,x,…x) = x

Semilattice operations have been generalised to **set functions** and **…milattices**



For more on this see talk by Benoit Larose

Near-unanim… and **Mal'tsev** operations have both been generalised further to **majority/minority operations** where *on each 2-element subset* the operation is either majority or Mal'tsev (Dalmau, LICS 2005)

Affine operations have been generalised to **Mal'tsev** operations where f(y, y, x) = f(x, y, y) = x

From Clones to Algebras

From Clones to Algebras

For every constraint language L over D there is an associated algebra

$$\mathcal{A} = (D, \mathrm{Pol}(L))$$

For every algebra $\mathcal{A} = (D, \boldsymbol{\Phi})$ there is an associated constraint problem

$$\mathrm{CSP}(\mathcal{A}) = \mathrm{CSP}(\mathrm{Inv}(\boldsymbol{\Phi}))$$

Hence we can classify constraint problems by classifying algebras…

Unary Relations and Subalgebras

Let L be a constraint language over a set D,
and let $\mathcal{A}_L$ be the algebra (D,Pol(L)) .

The following are equivalent:

- The unary relation R is invariant under Pol(L);
- The unary relation R belongs to $\langle L \rangle$;
- The set of elements of R is a subalgebra of $\mathcal{A}_L$

If L contains all unary relations, then $\mathcal{A}_L$ has every subset as a subalgebra and is called conservative

Conservative Algebras

Bulatov (2003) showed that when L is a set of relations containing all unary relations, then CSP(L) is tractable precisely when every 2-element subalgebra of $\mathcal{A}_L$ = (D,Pol(L)) is tractable.

Theorem (Bulatov): A conservative algebra $\mathcal{A}$ = (D, $\boldsymbol{\Phi}$) is tractable if and only if, for every 2-element subset B of D, there exists f in Pol(Inv($\boldsymbol{\Phi}$)) such that $f|_B$ is either:

- A semilattice operation;
- A ternary affine operation;
- A ternary majority operation.

Classifying Algebras

Theorem: Tractability of an algebra $\boldsymbol{A}$ is preserved by:

- taking subalgebras
- taking homomorphic images
- taking finite powers

} taking factors

(By reduction to CSP($\boldsymbol{A}$*))*

Every finite algebra whose operations are permutations is NP-complete. *(By reduction from COLOURING)*

Conjecture: An idempotent algebra is NP-complete if it has a factor containing only permutations. *Otherwise it is tractable*.

Classifying Algebras

Theorem: Tractability of an algebra $\mathcal{A}$ is preserved by:

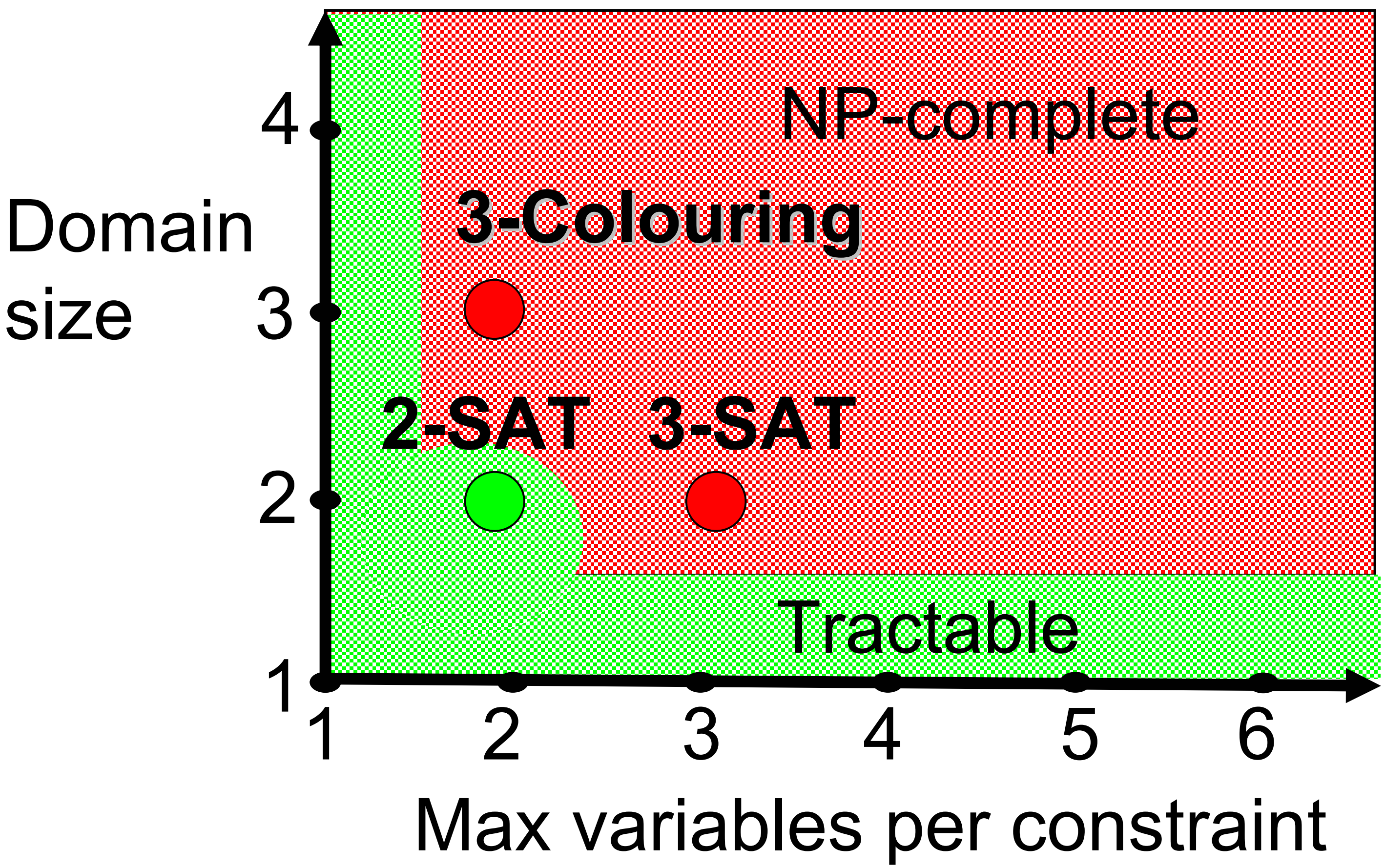
- taking subalgebras
- taking homomorphic images
- taking finite direct products
- taking factors

For more on this see
talk by Andrei Bulatov

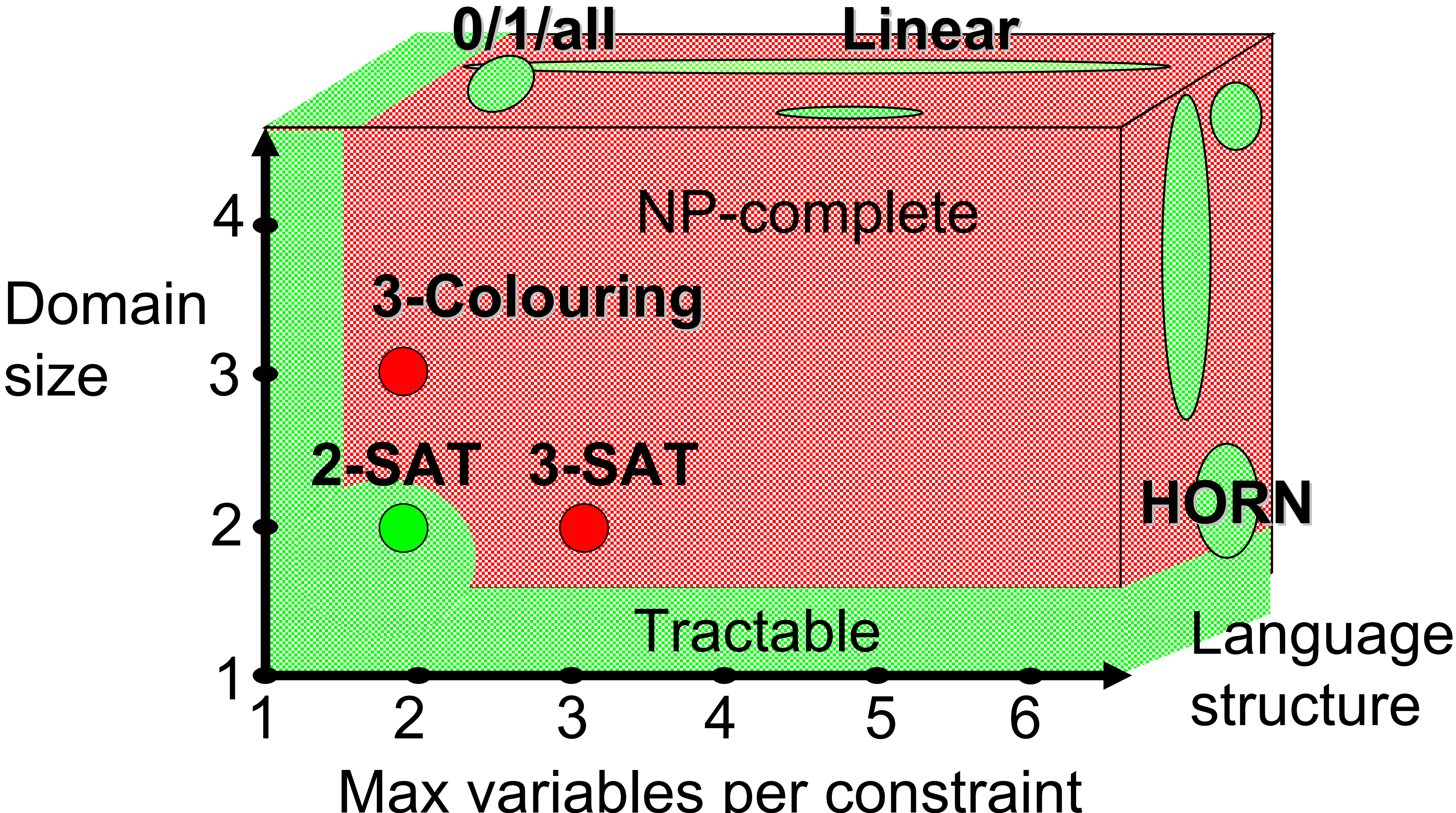
Every finite algebra whose operations are permutations is NP-complete. *(By reduction from COLOURING)*

Conjecture: An algebra is NP-complete if it has a factor containing only permutations.*Otherwise it is tractable*.

Complexity of Languages



Complexity of Languages



Constraints & Algebra

What are the advantages of this approach?

- Gives a unified way to characterise sets of structures/classes of problems
- Links each efficient algorithm to a structural property/polymorphism
- Gives hardness proofs without reductions
- Brings a rich algebraic theory to bear
- Suggests new approaches for infinite domains and soft constraints…

Soft Constraints

Definition of CSP(L)

Definition 1a:

- An *instance* of CSP(L) is a 3-tuple (**V,D,C**), where
  - **V** is a set of variables
  - **D** is a single domain of possible values
  - **C** is a set of constraints

Each constraint in **C** is a pair (**s,R**) where

- **s** is a list of variables defining the scope
- **R** is a relation from L defining the allowed combinations of values

Definition of VCSP(L)

Definition 1a:

- An *instance* of VCSP(L) is a 4-tuple (**V**,**D**,**C**,**Ω**), where
  - **V** is a set of variables
  - **D** is a single domain of possible values
  - **C** is a set of constraints
  - **Ω** is a set of *costs*

Each constraint in **C** is a pair (**s**,**φ**) where

- **s** is a list of variables defining the scope
- **φ** is a function from L defining the cost associated with each combination of values

valued Boolean constraints

SAT

$$x + y + z = 0$$

$$(x \vee y \vee \neg z) \wedge (x \vee \neg y \vee z) \wedge (\neg x \vee y \vee z) \wedge (\neg x \vee \neg y \vee \neg z)$$

$\equiv$

x	y	z	
0	0	0	✓
0	0	1	X
0	1	0	X
0	1	1	✓
1	0	0	X
1	0	1	✓
1	1	0	✓
1	1	1	X

valued Boolean constraints

MAX-SAT

$$x + y + z = 0$$

$$(x \vee y \vee \neg z) \wedge (x \vee \neg y \vee z) \wedge (\neg x \vee y \vee z) \wedge (\neg x \vee \neg y \vee \neg z) \equiv$$

x	y	z	
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

valued Boolean constraints

MAX-SAT

$$x + y + z = 0$$

$$(x \vee y \vee \neg z) \wedge (x \vee \neg \quad \vee z) \wedge$$
$$(\neg x \vee y \vee z) \wedge (\neg x \quad y \vee \neg z)$$

$$\equiv$$

x	y	z	
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

For more on this see
talk by Peter Jonsson

valued Boolean constraints

VSAT

x	y	z	
0	0	0	0
0	0	1	1
0	1	0	7
0	1	1	1
1	0	0	∞
1	0	1	3
1	1	0	∞
1	1	1	0

- Very general discrete optimization problem
- NP-hard

Tractable cases

x	y	z	
0	0	0	0
0	0	1	1
0	1	0	7
0	1	1	1
1	0	0	∞
1	0	1	3
1	1	0	∞
1	1	1	0

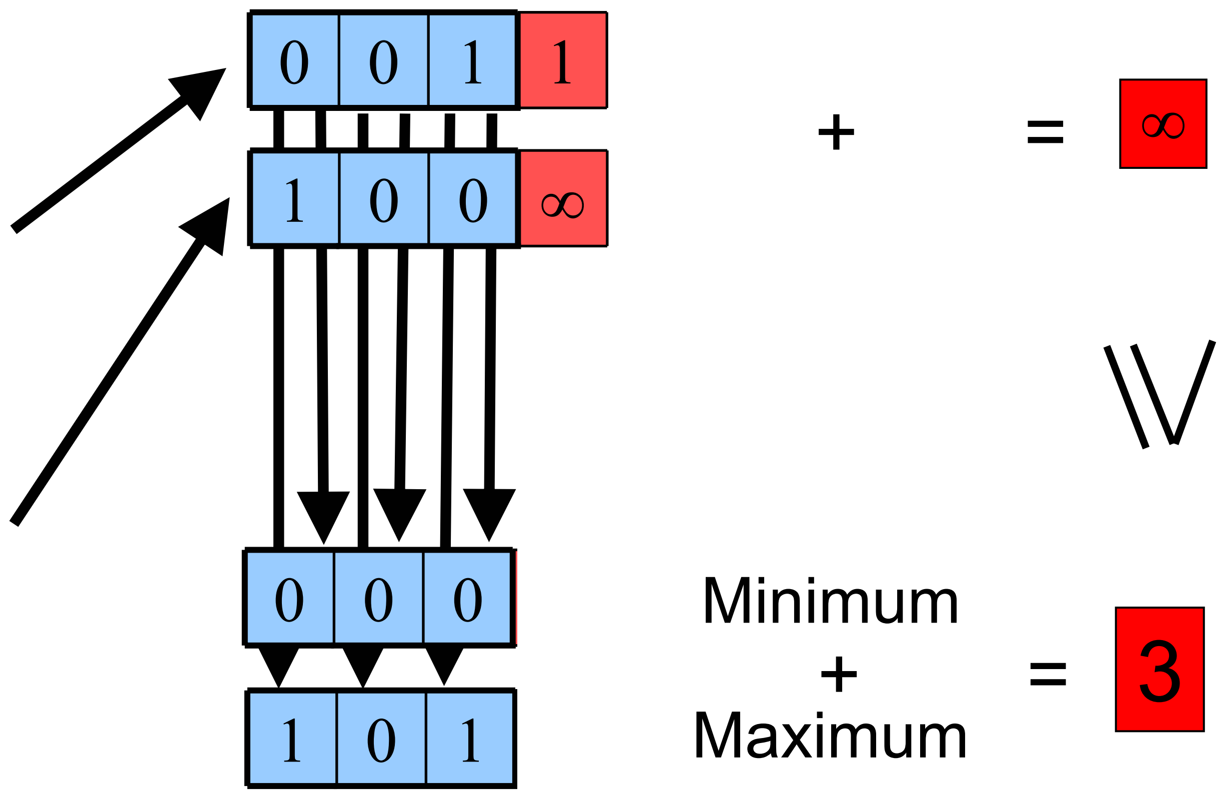
1	0	1

Maximum

Tractable cases

$$\forall s,t \quad \mathrm{Cost}(\mathrm{Min}(s,t)) + \mathrm{Cost}(\mathrm{Max}(s,t)) \le \mathrm{Cost}(s) + \mathrm{Cost}(t)$$

x	y	z	
0	0	0	0
0	0	1	1
0	1	0	7
0	1	1	1
1	0	0	∞
1	0	1	3
1	1	0	∞
1	1	1	0



Tractable cases

$\forall s,t \;\; \mathrm{Cost}(\mathrm{Min}(s,t)) + \mathrm{Cost}(\mathrm{Max}(s,t)) \leq \mathrm{Cost}(s) + \mathrm{Cost}(t)$

x	y	z	
0	0	0	0
0	0	1	1
0	1	0	7
0	1	1	1
1	0	0	∞
1	0	1	3
1	1	0	∞
1	1	1	0

We say that the cost function has the ***multimorphism*** (Min,Max)

Any cost function with this property is called ***submodular***

If all cost functions are submodular
the problem is tractable

Tractable cases

$$\forall s,t \;\; \mathrm{Cost}(\mathrm{Min}(s,t)) + \mathrm{Cost}(\mathrm{Max}(s,t)) \leq \mathrm{Cost}(s) + \mathrm{Cost}(t)$$

We say that the cost function has the ***multimorphism*** (Min,Max)

Any cost function with this property is called ***submodular***

If all cost functions are submodular
the problem is tractable

Tractable cases

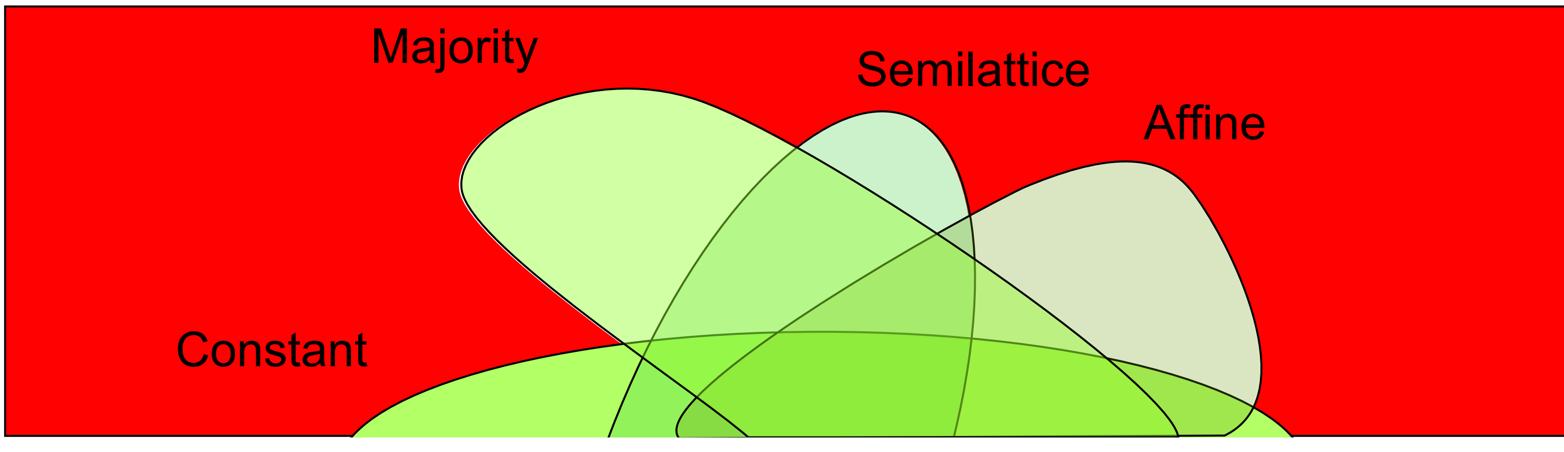
If the cost functions all have one of these eight multimorphisms, then the problem is tractable:

1) (Min,Max)
2) (Max,Max)
3) (Min,Min)
4) (Majority,Majority,Majority)
5) (Minority,Minority,Minority)
6) (Majority,Majority,Minority)
7) (Constant 0)
8) (Constant 1)

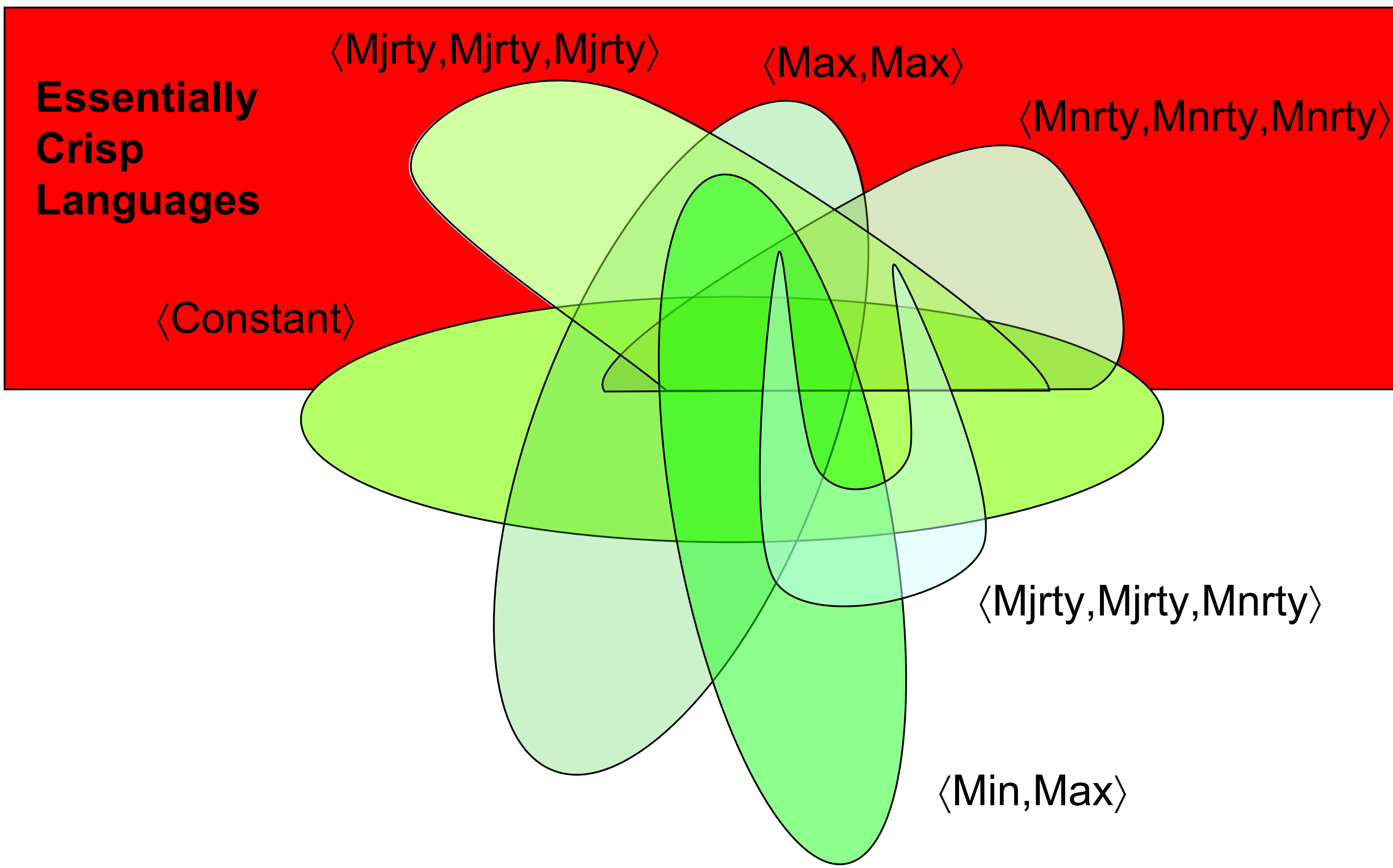
Note: These are tractable cases for all finite domains

Cohen et al CP'03

Tractable cases



Tractable cases



Intractable cases

$$\boxed{\text{MAX-SAT}}$$

This constraint is known to be NP-hard

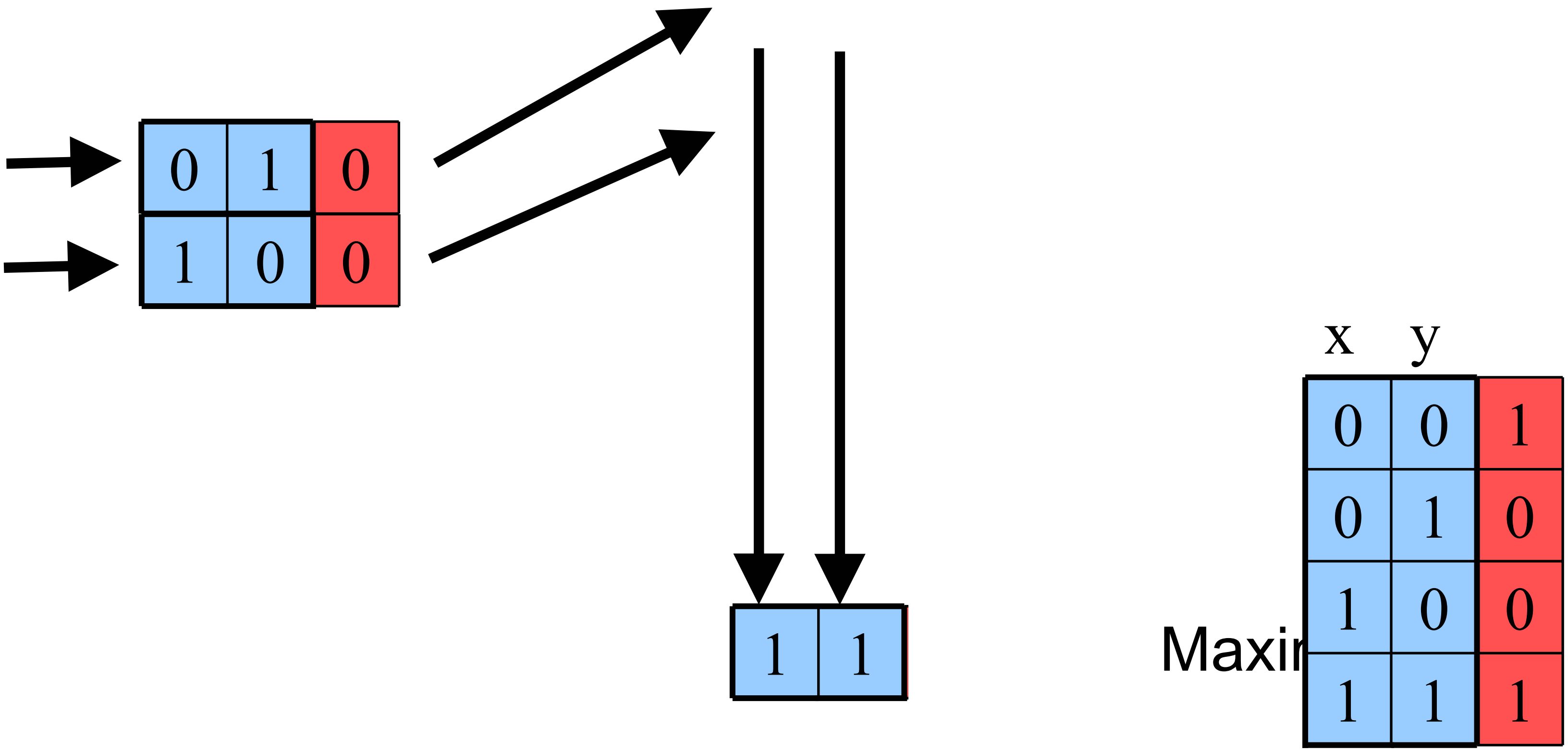
$x \neq y$

$(x \wedge y) \vee (\neg x \wedge \neg y)$

$\equiv$

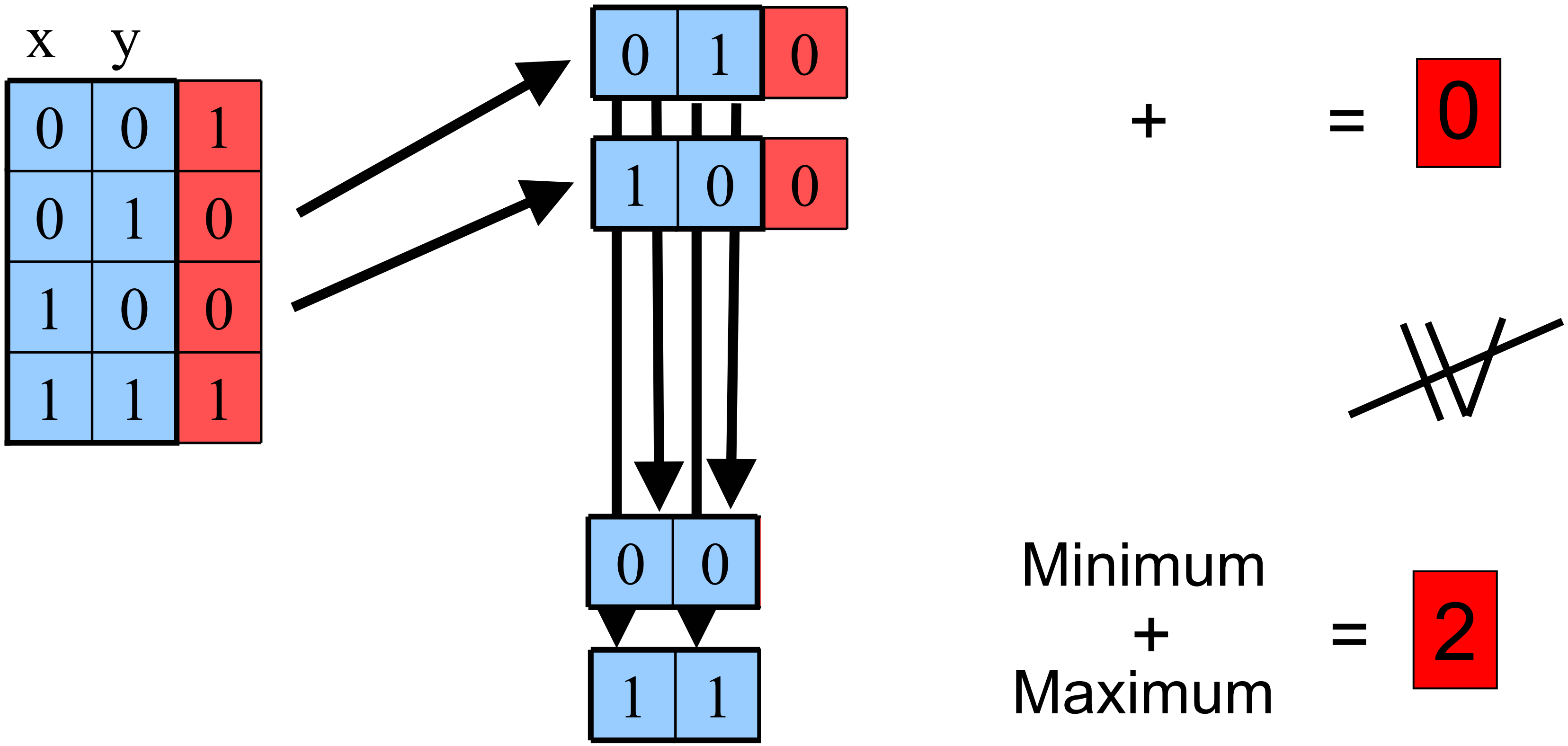
x	y	
0	0	1
0	1	0
1	0	0
1	1	1

Intractable cases



Intractable cases

This cost function has **no** significant multimorphisms



Intractable cases

This cost function has **no** significant multimorphisms

x	y	
0	0	b
0	1	a
1	0	a
1	1	b

For some
$a < b < \infty$

Any set of Boolean cost functions
which doesn't have a multimorphism
from the list of 8
can be combined to express
this form of cost function

and hence is NP-hard

Cohen, Cooper, Jeavons CP'04

Dichotomy Theorem

If the cost functions all have one of these eight multimorphisms, then the problem is tractable:

1) (Min,Max)
2) (Max,Max)
3) (Min,Min)
4) (Majority,Majority,Majority)
5) (Minority,Minority,Minority)
6) (Majority,Majority,Minority)
7) (Constant 0)
8) (Constant 1)

In all other (Boolean) cases the cost functions have **no** significant common multimorphisms and the problem is **NP-hard**.

Summary on soft constraints

- Valued constraints are a general framework for discrete optimization including SAT, MAX-SAT and VSAT.
- These problems are NP-hard in general.
- In the Boolean case:
  - there are exactly 8 tractable cases, each characterized by a multimorphism.
  - Any set of cost functions which **doesn’t** have one of these 8 has no significant multimorphisms and is NP-hard.

Current challenges/Open problems

- Complete the classification of constraint languages over a finite domain
- Combine analysis of constraint languages with structural aspects of constraint problems to identify broader tractable classes
- Extend the algebraic theory to infinite domains
- Show that the expressive power of valued constraints is determined by their multimorphisms
- Link to practical constraint programming systems ("global constraints")

Complexity of CSP