Discrete Optimisation

The Big Picture

Peter Jeavons
Department of Computer Science
University of Oxford

Joint work with
David Cohen, Martin Cooper, Paidi Creed, Standa Zivny

The past 100 days…

- **A new Galois connection for classifying the complexity of a wide range of discrete optimisation problems…**

  An Algebraic Theory of Complexity for Discrete Optimisation,
  David Cohen, Martin Cooper, Paidi Creed, PJ, Stanislav Zivny,
  arxiv.org/abs/1207.6692 (submitted on 28 Jul, 2012)

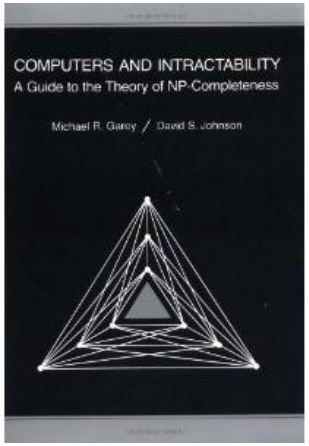
- **A new dichotomy for 3-valued problems identifying infinitely many distinct tractable cases…**

  Skew Bisubmodularity and Valued CSPs,
  Anna Huber, Andrei Krokhin, and Robert Powell
  to appear in SODA 2013 (accepted on Sep 15, 2012)

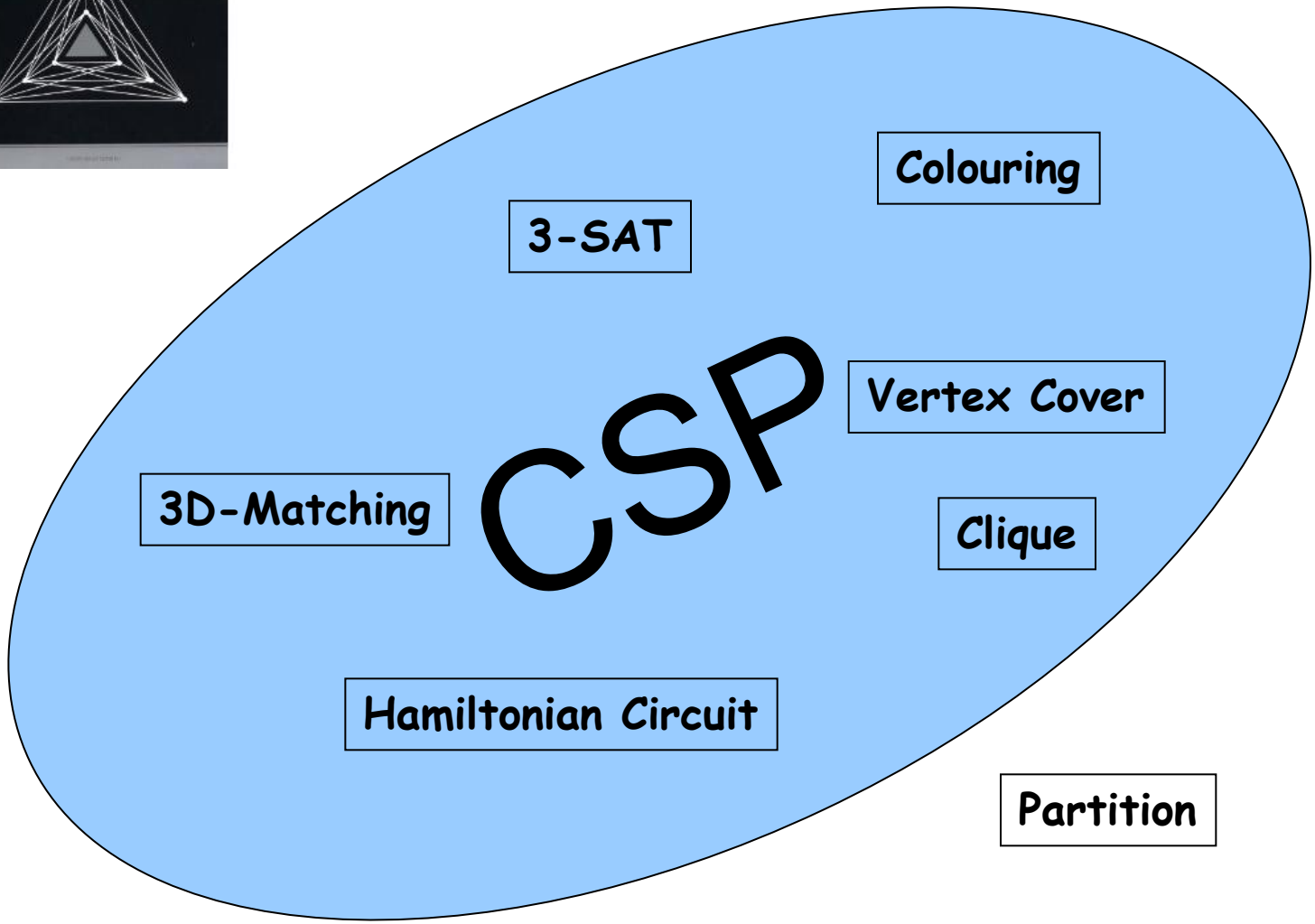
- **A complete classification of finite-valued cases…**

  The complexity of finite-valued CSPs,
  Johan Thapper, Stanislav Zivny
  arxiv.org/abs/1210.2987 (submitted on 10 Oct, 2012)

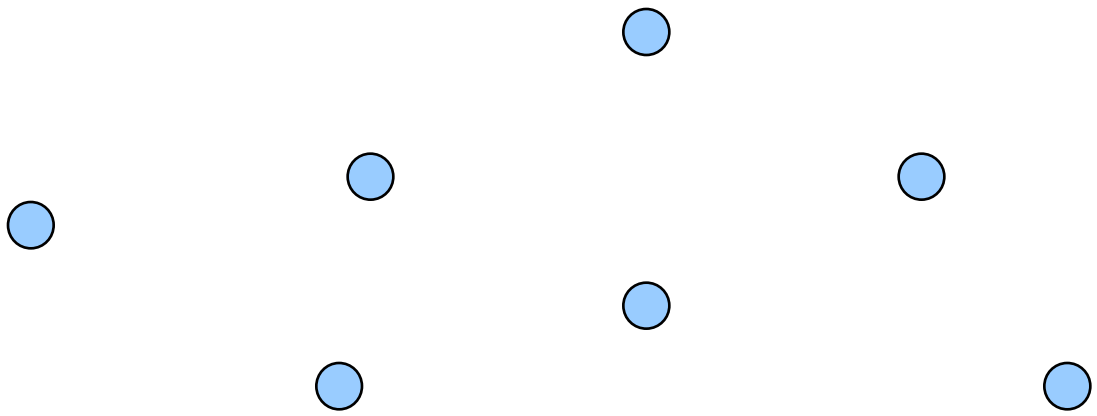
Background



“Basic” Problems



A General Framework



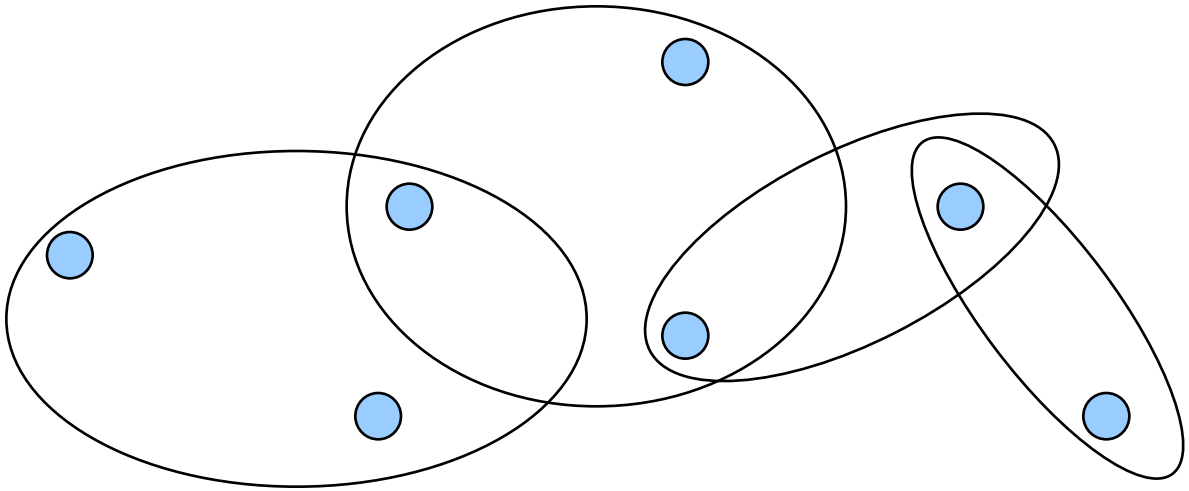
Variables = Talks to be scheduled at conference

Transmitters to be assigned frequencies

Amino acids to be located in space

Circuit components to be placed on a chip

A General Framework



Constraints = All invited talks on different days

No interference between near transmitters

$x + y + z > 0$

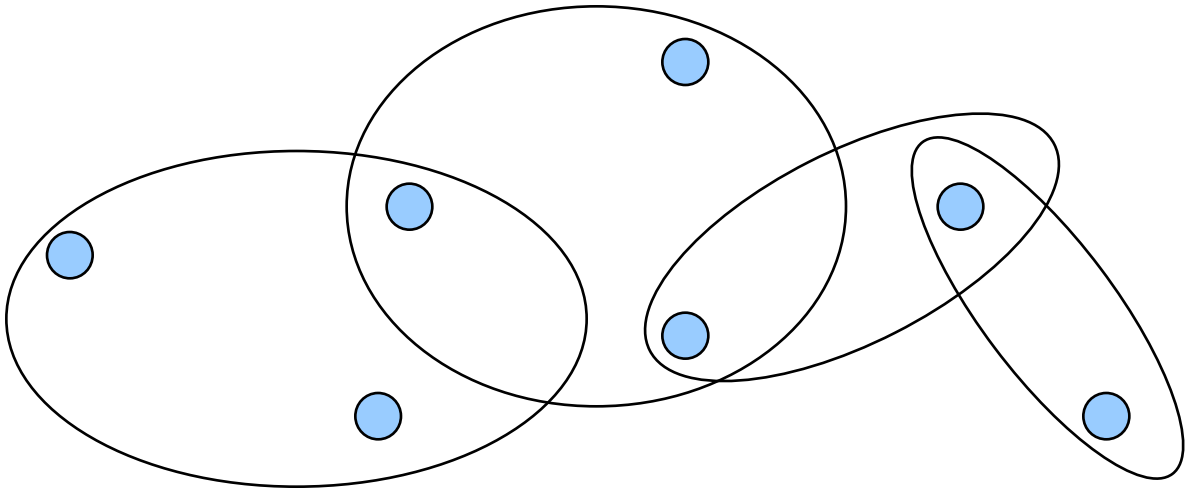
At least 1μm between wires

Outline

- Constraint languages
- Complexity of different languages
- Expressive power
- Algebraic properties of constraint languages
- Generalizing – the bigger picture
- Valued constraint languages

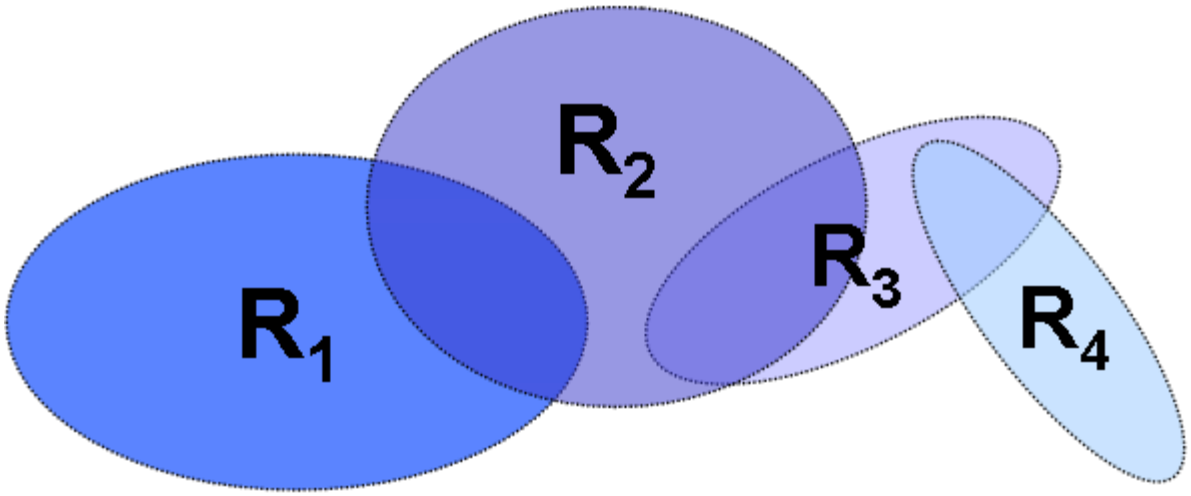
Constraint Languages

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)

...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 some fixed 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

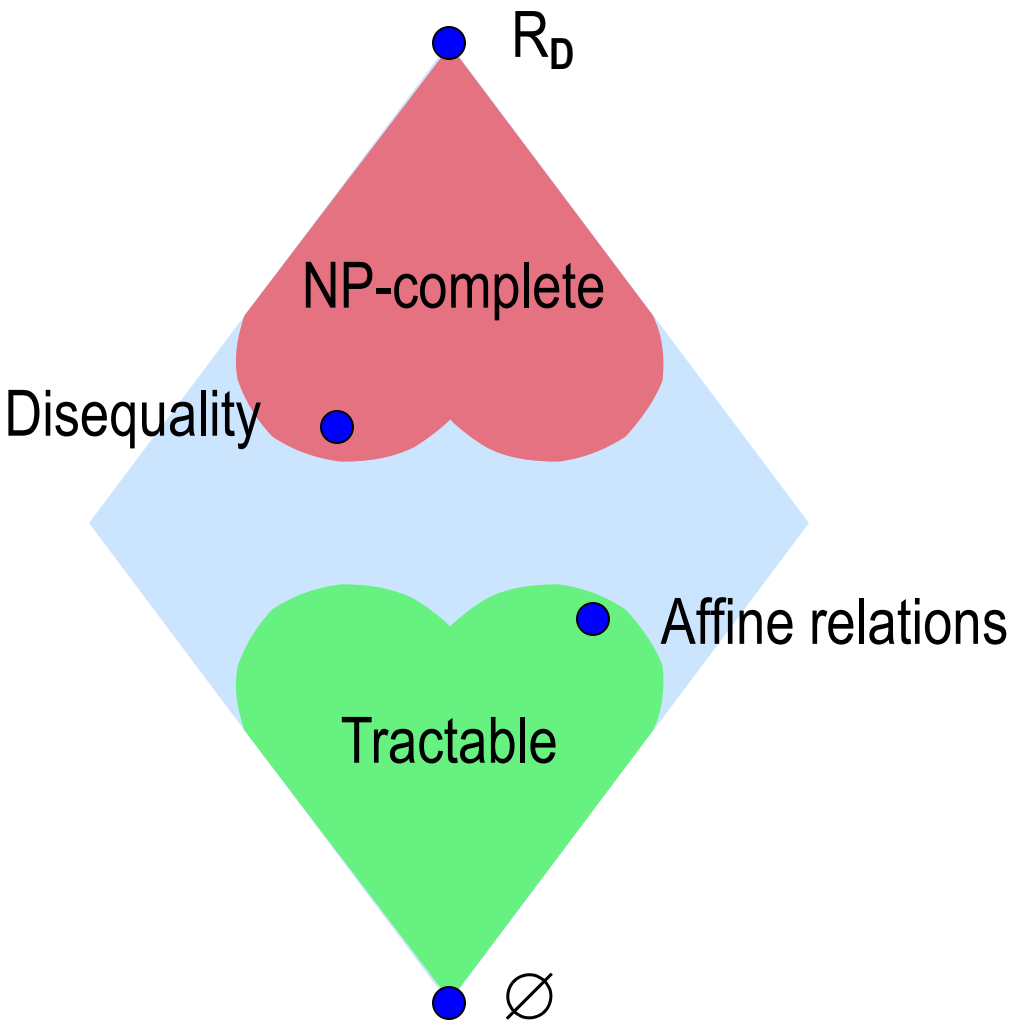
Examples

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

The Lattice of Languages



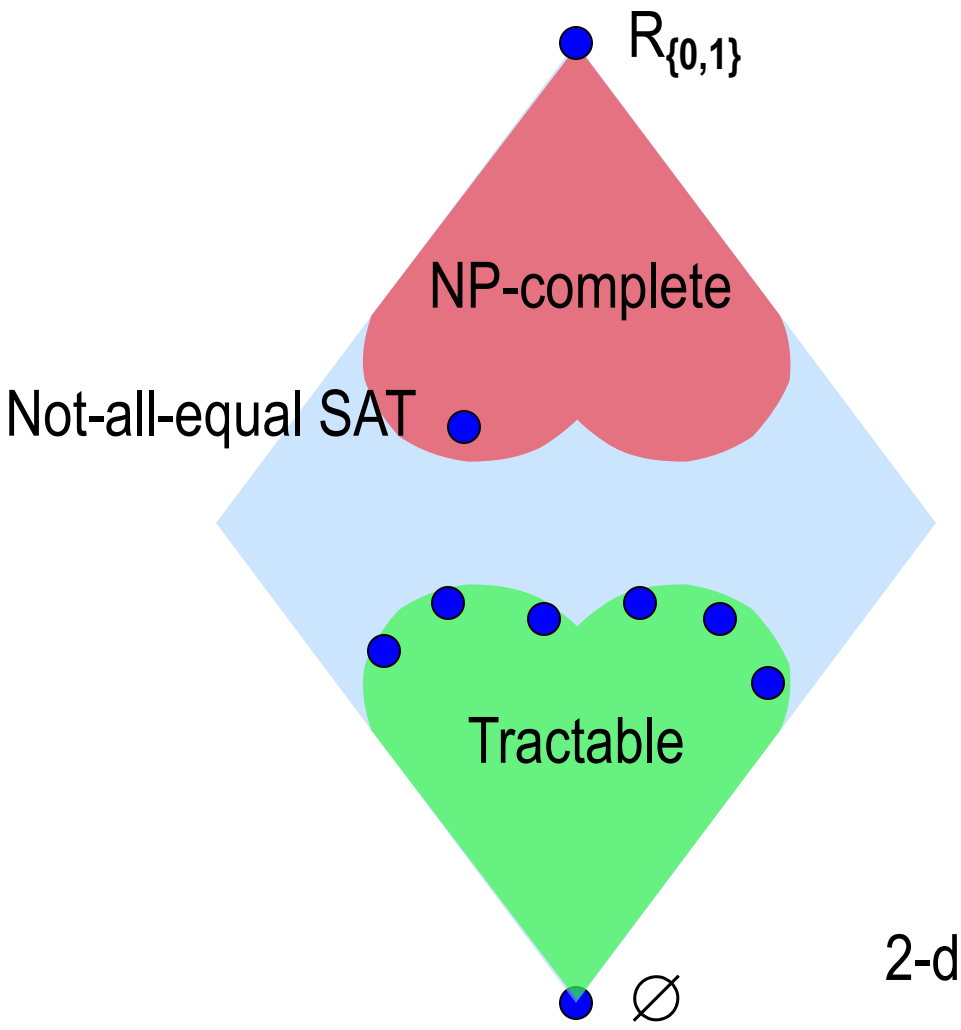
Complexity – Boolean Case

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)

see Schaefer, T.J., "The complexity of satisfiability problems", Proc 10th ACM Symposium on Theory of Computing (STOC), (1978) pp.216-226.

Boolean Languages



0…0 relations

1…1 relations

Horn relations

Dual Horn relations

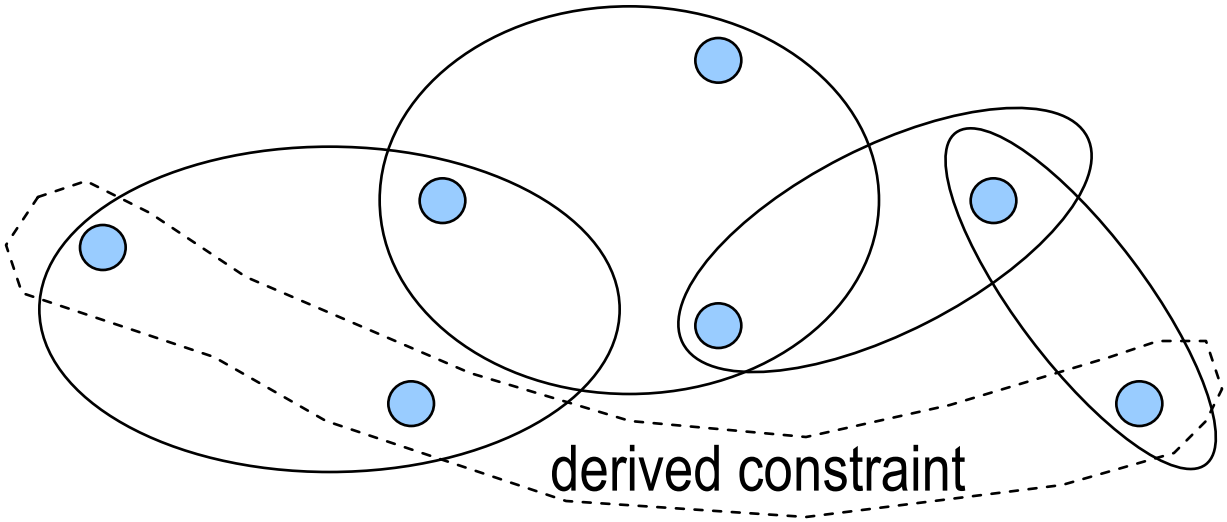
2-decomposable relations

Affine relations

Expressive Power

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

Definition 2:

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 and Reduction

Theorem (Jeavons 98) **:** For any constraint language **L**, and any finite constraint language **L′**, if $\mathbf{L'} \subseteq \langle \mathbf{L} \rangle$ then CSP(**L′**) is polynomial-time reducible to CSP(**L**)

Expressive Power and Reduction

Theorem (Jeavons 98) **:** For any constraint language **L**, and any finite constraint language **L′**, if $\mathbf{L}' \subseteq \langle \mathbf{L} \rangle$ then CSP(**L′**) is polynomial-time reducible to CSP(**L**)

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

Corollary: If $\langle \mathbf{L}_1 \rangle = \langle \mathbf{L}_2 \rangle$ then CSP($\mathbf{L}_1$) is polynomial-time equivalent to CSP($\mathbf{L}_2$)

.

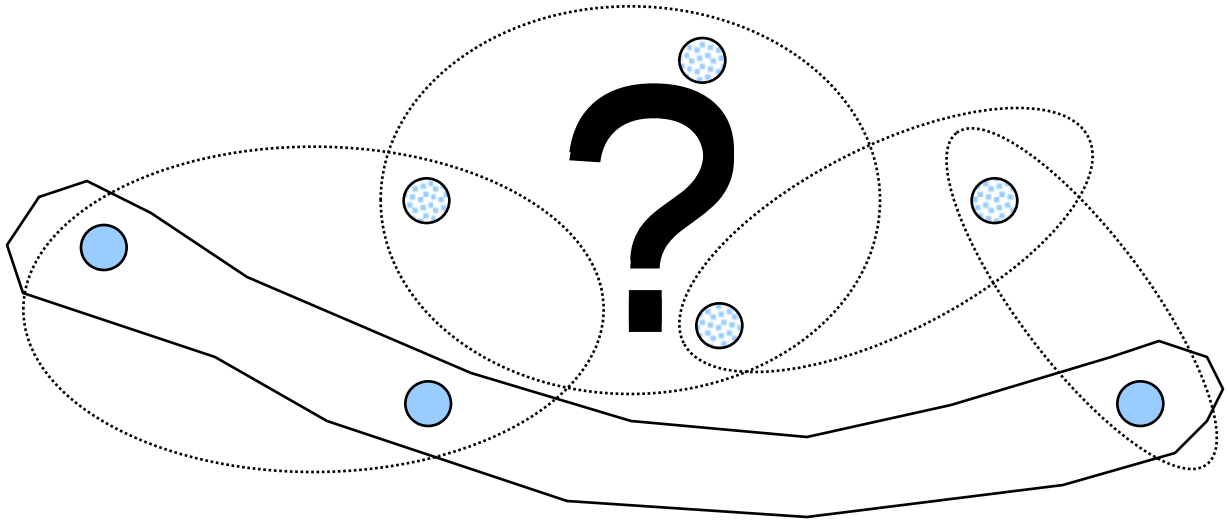
Expressive Power and Reduction

Theorem (Jeavons 98) **:** For any constraint language **L**, and any finite constraint language **L′**, if $\mathbf{L}' \subseteq \langle \mathbf{L} \rangle$ then CSP(**L′**) is polynomial-time reducible to CSP(**L**)

$\langle \mathrm{L} \rangle$ is more important than L

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?



Algebraic Properties

Algebraic Invariance

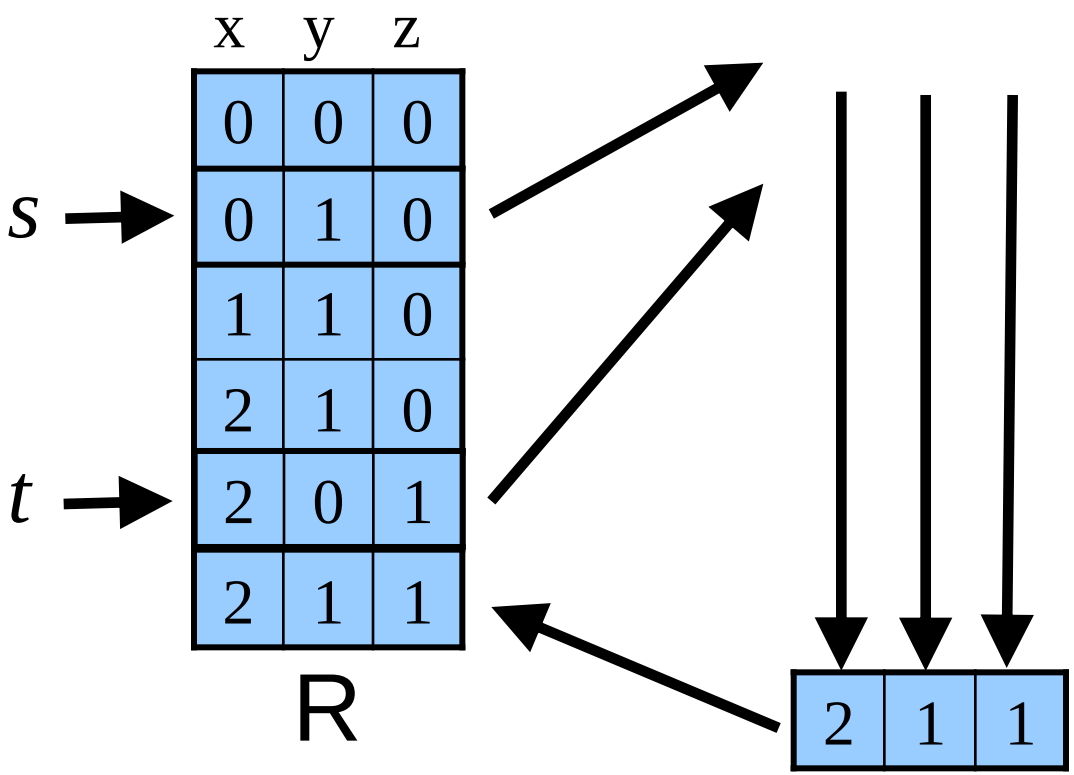
Definition: A relation R is ***invariant*** under a k-ary operation ϕ, if, for any tuples $\underline{a}_1, \underline{a}_2, \ldots, \underline{a}_k \in R$, the tuple obtained by applying ϕ co-ordinatewise is a member of R .

If R is invariant under ϕ,

then ϕ is called a ***polymorphism*** of R.

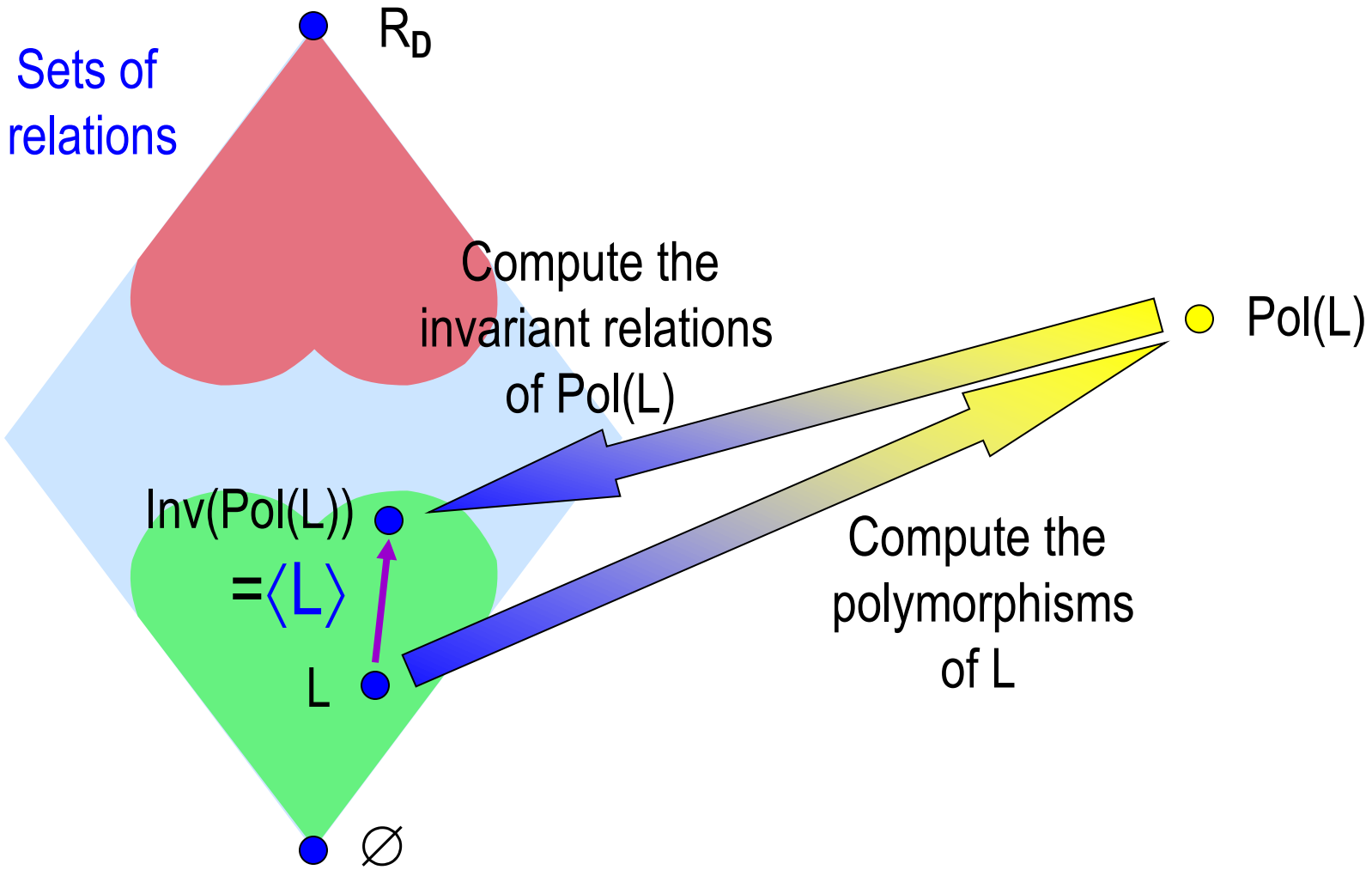
Example of Polymorphism

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



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

Pol and Inv


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

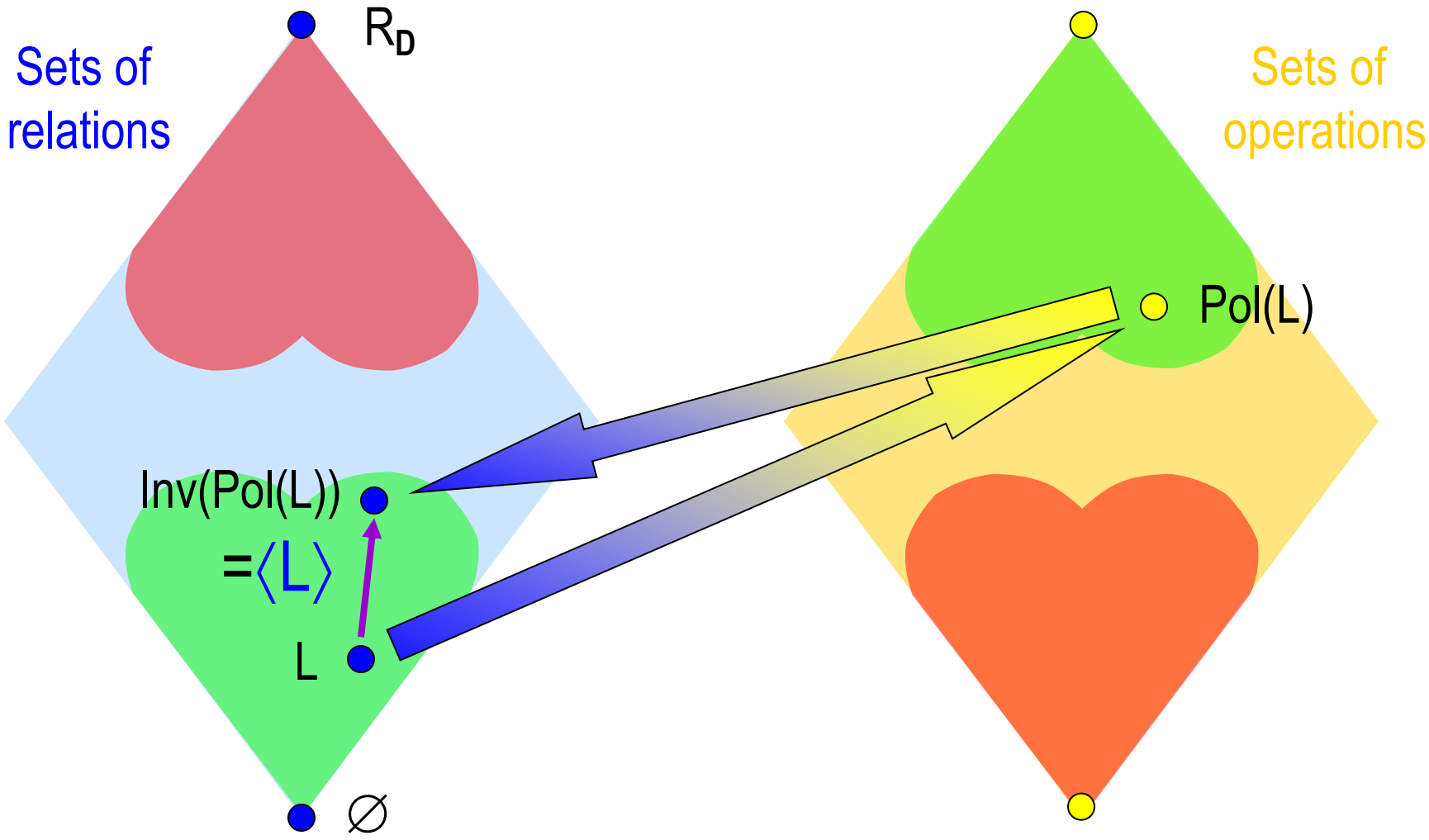
Expressive Power

Theorem (Geiger 68) **:** For any constraint language L, over a finite domain, $\langle L \rangle = \mathrm{Inv}(\mathrm{Pol}(L))$

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



Clones

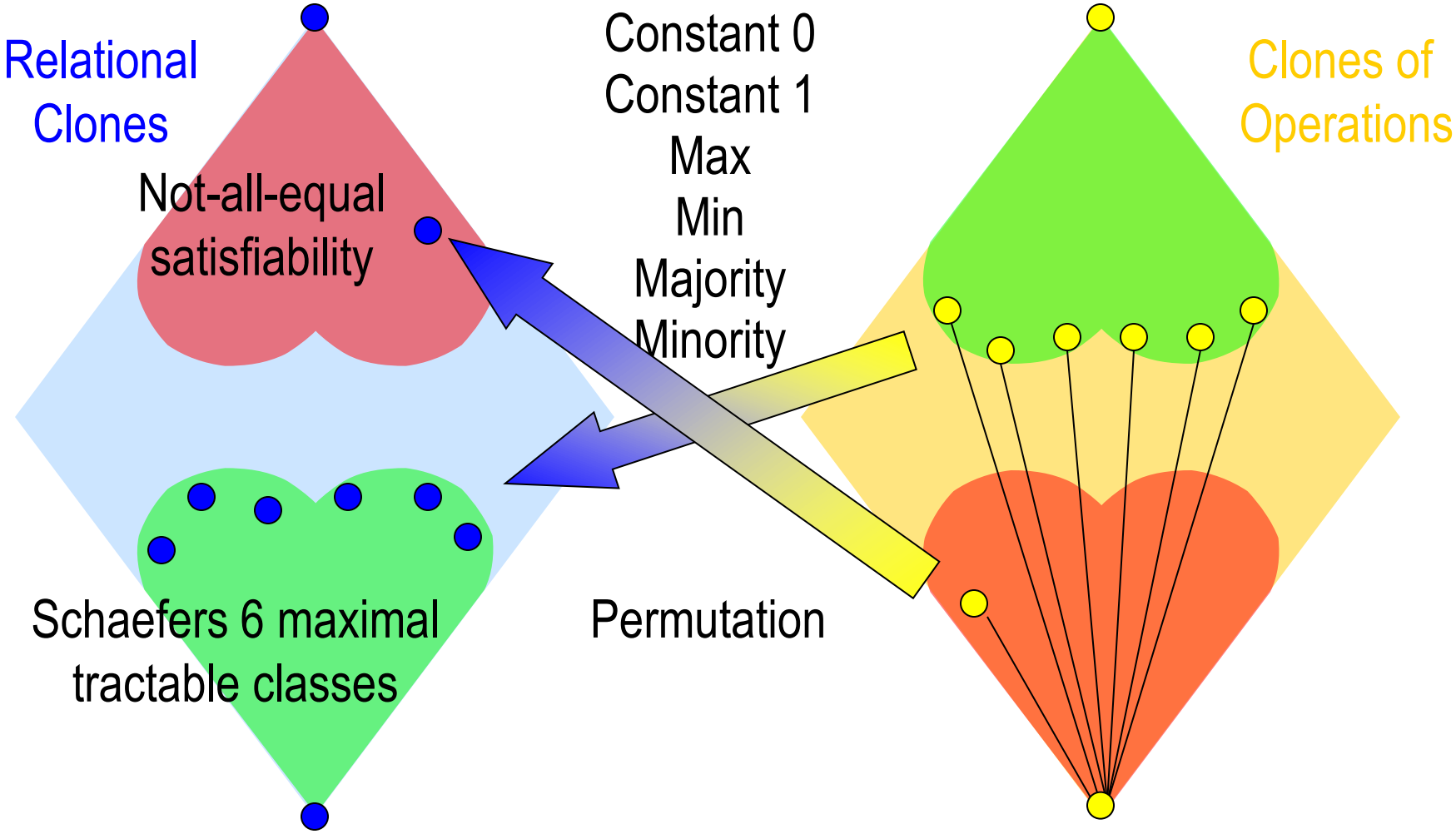
Definition: A *relational clone* is a set of relations which is closed under relational join and projection.

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

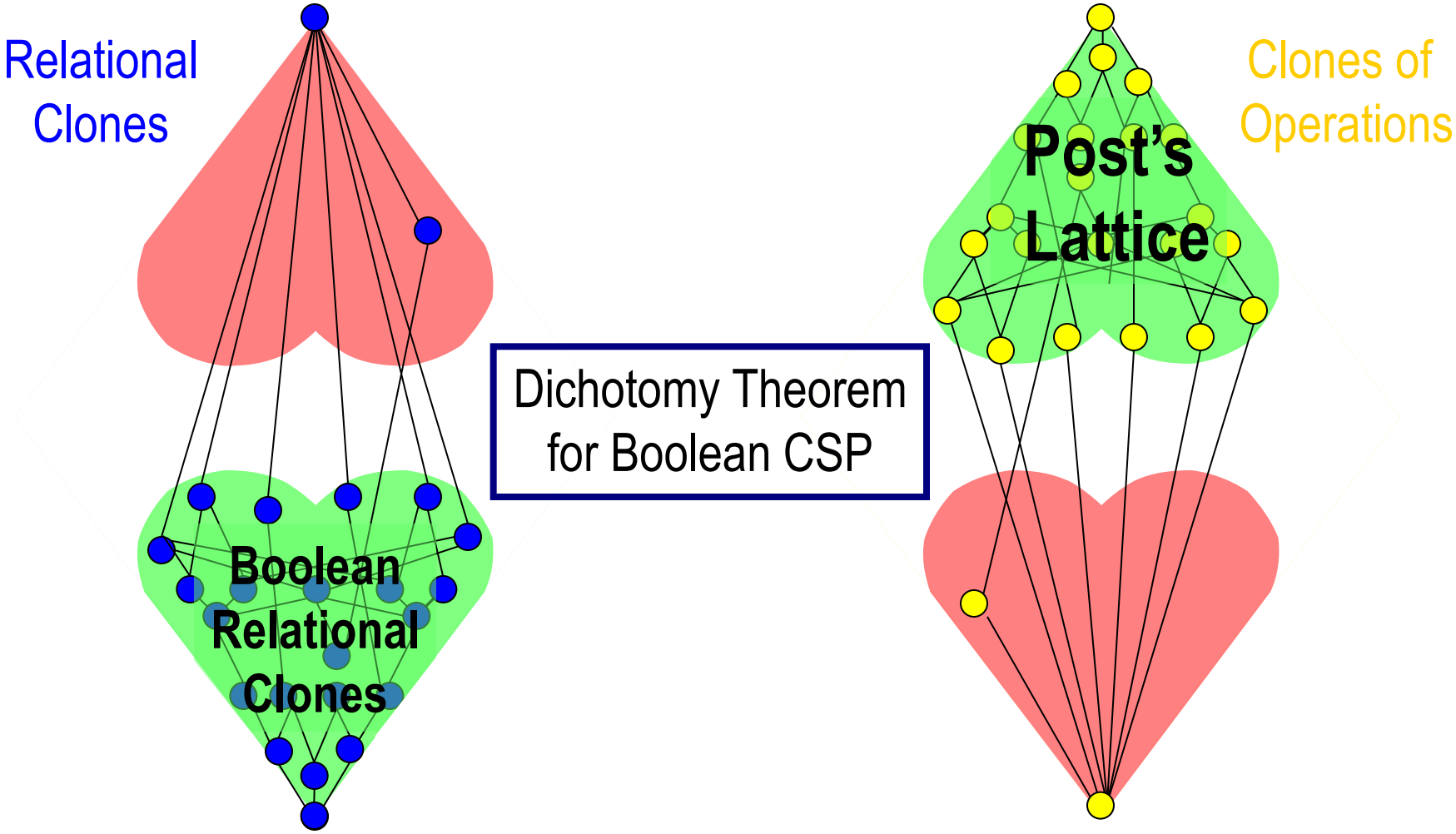
Every relational clone is of the form Inv(Φ) for some Φ

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

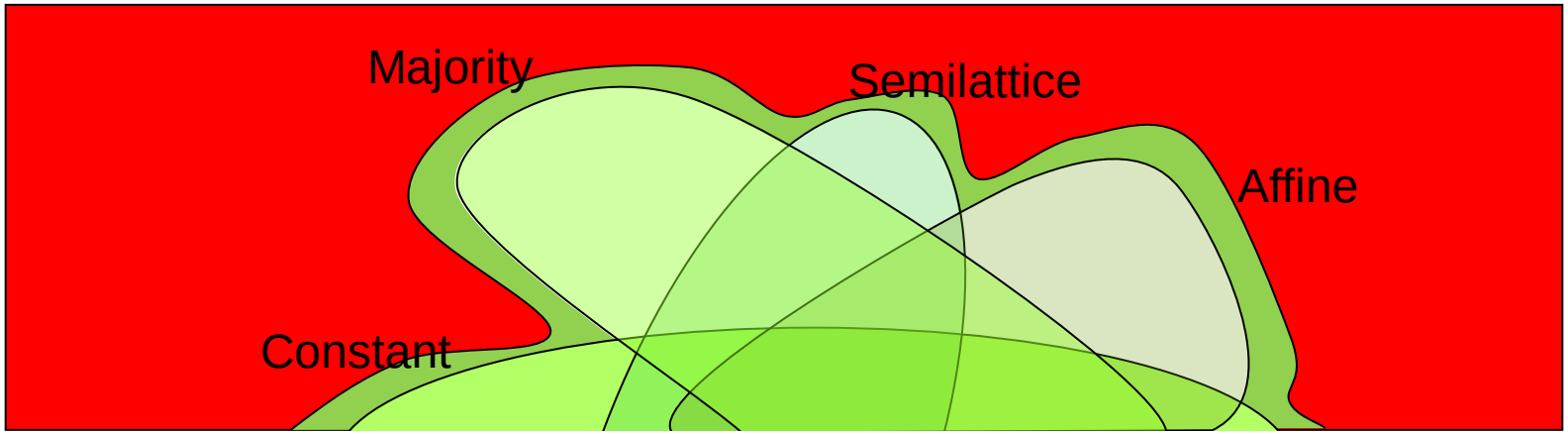
Boolean Operations



Boolean Operations



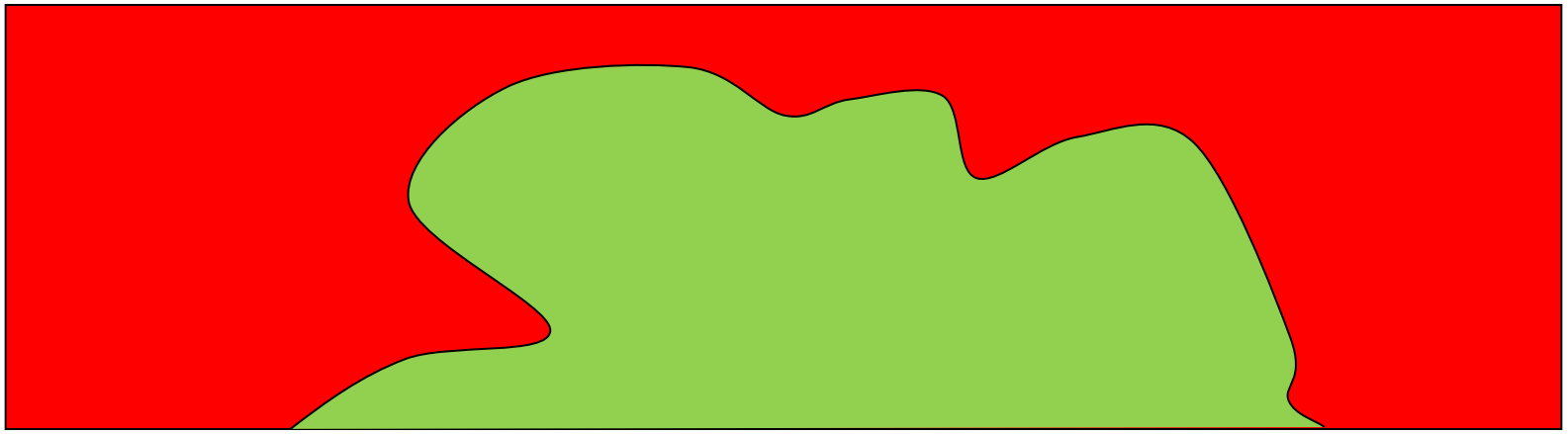
Islands of tractability



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

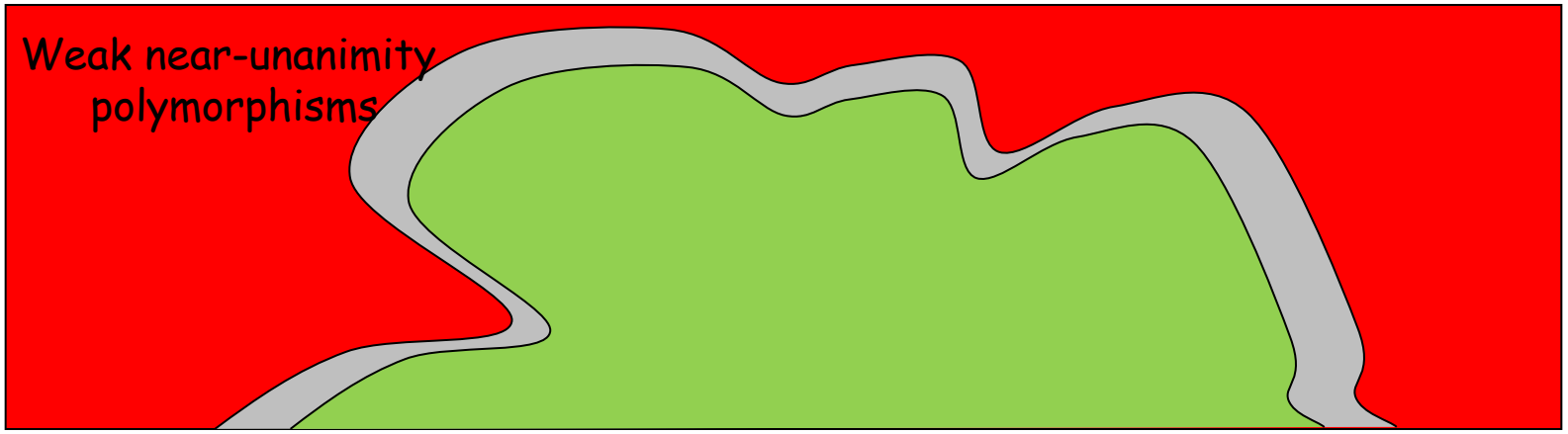
- For larger domains this is **not** a complete description…

Towards a Dichotomy



By investigating the *algebras* associated with the clone of polymorphisms it may be possible to identify *precisely* which polymorphisms lead to tractability on any finite domain…

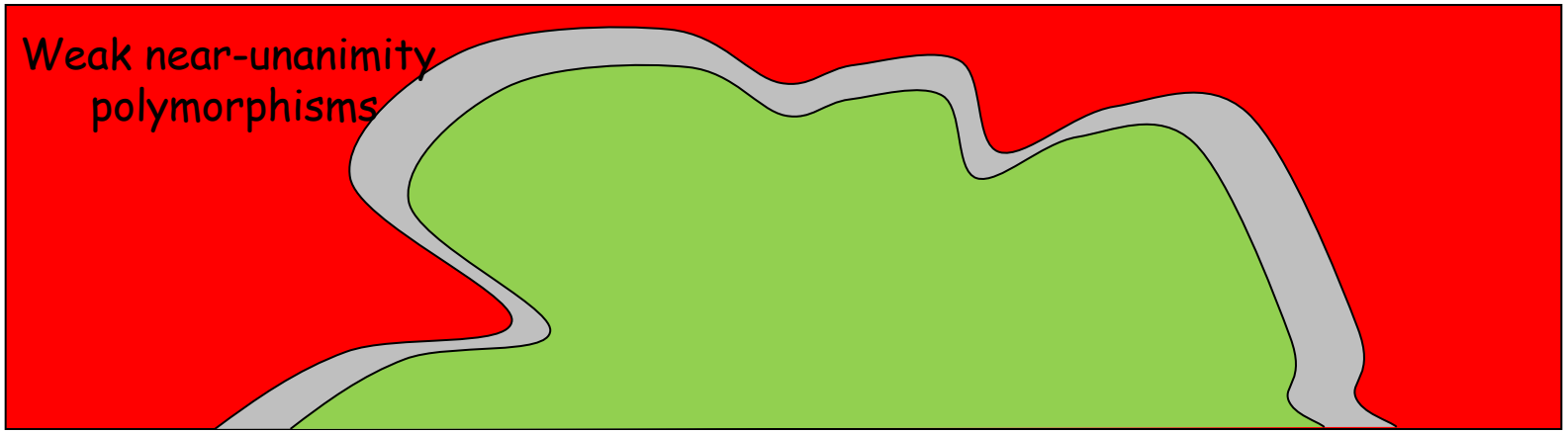
Towards a Dichotomy



Theorem: A constraint language over a finite domain that includes all constants is tractable only if it has a polymorphism f such that

$$f(x,x,\ldots,x,y) = f(x,\ldots x,y,x) = \ldots = f(y,x,\ldots x)$$

Towards a Dichotomy

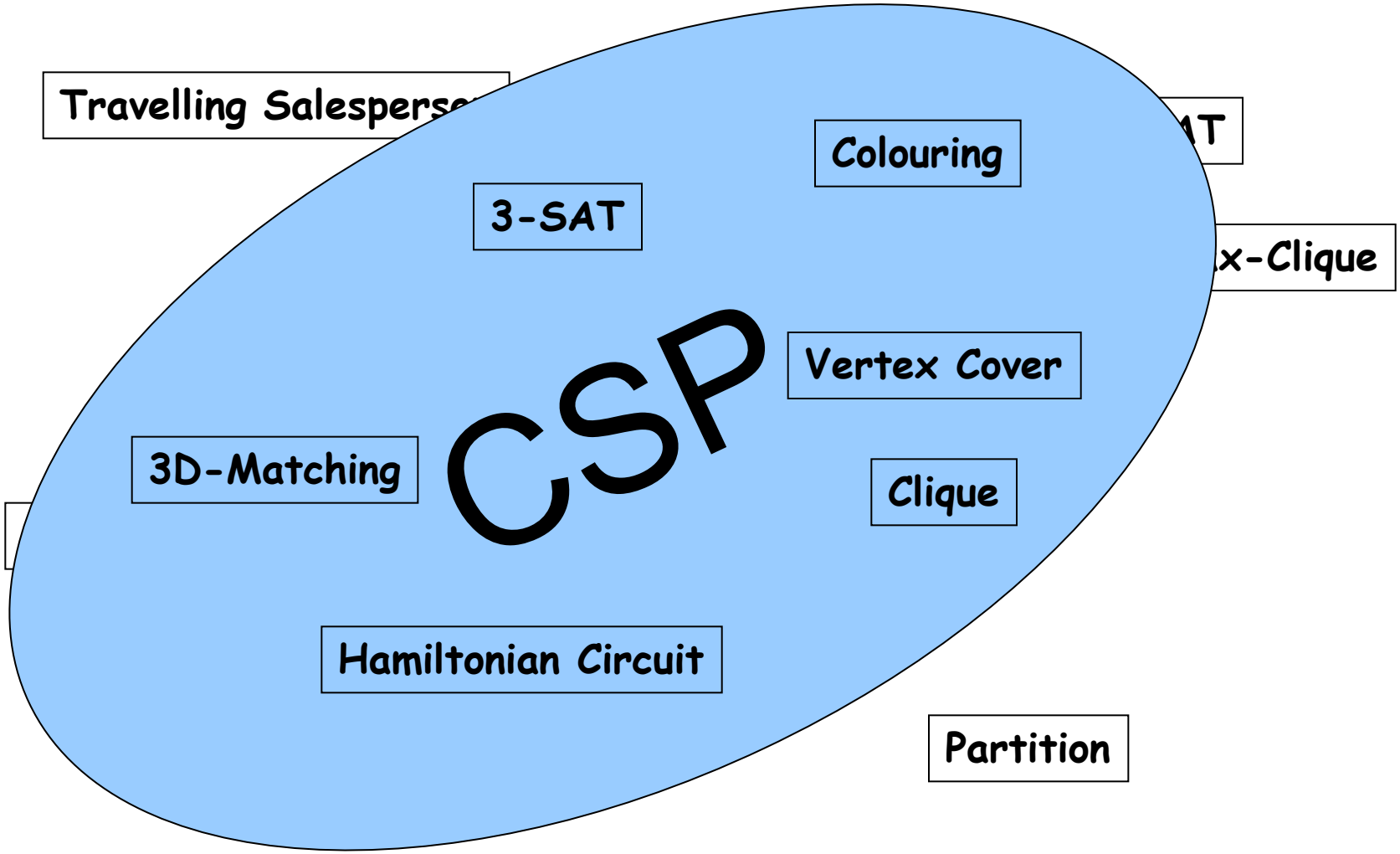


Conjecture: A constraint language over a finite domain that includes all constants is tractable if and only if it has a polymorphism f such that

$$f(x,x,\ldots,x,y) = f(x,\ldots x,y,x) = \ldots = f(y,x,\ldots x)$$

Generalizing the CSP

A Bigger Picture



Fragmentation

- COP
- Max-CSP
- Max-SAT
- WCSP
- FCSP
- HCLP
- Pseudo-Boolean Optimisation
- Bayesian Networks
- Random Markov Fields
- Integer Programming
- …

Definition of CSP(L)

- 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)

- 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

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

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

VSAT

- Very general discrete optimization problem
- NP-hard

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

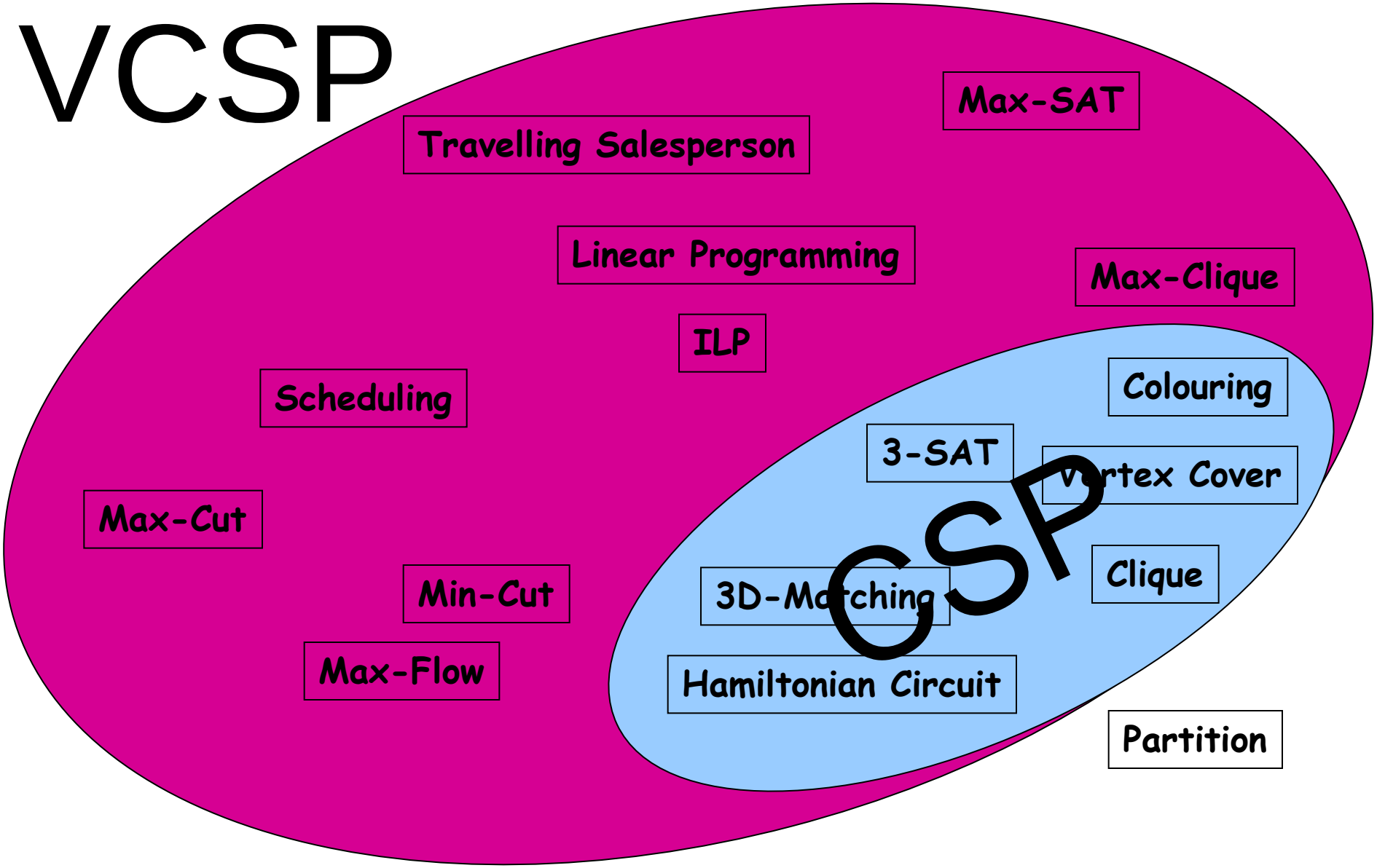
Valued constraints

VCSP

- Very general discrete optimization problem
- NP-hard

x_1	x_2	…	x_k	
0	0	…	0	3
0	0	…	1	0
0	0	…	2	∞
.	.	…	.	.
.	.	…	.	.
.	.	…	.	.
d	d	…	d	6

A Bigger Picture



Valued Constraint Languages

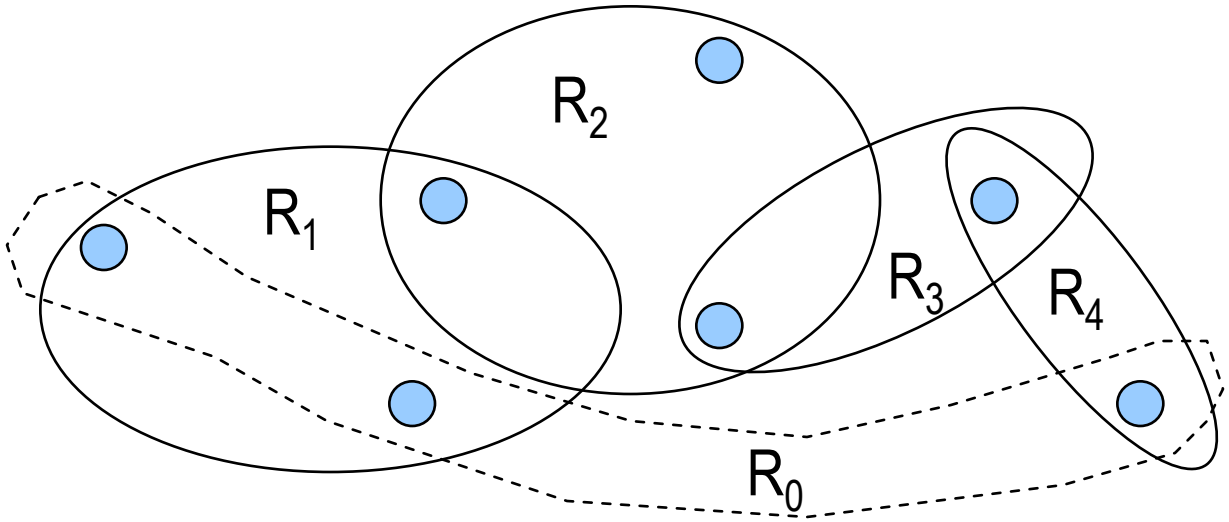
Valued Constraint Languages

Definition: A *valued constraint language* is a set of *functions* from D^n to Ω, for some fixed finite set D and some set of costs Ω.

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

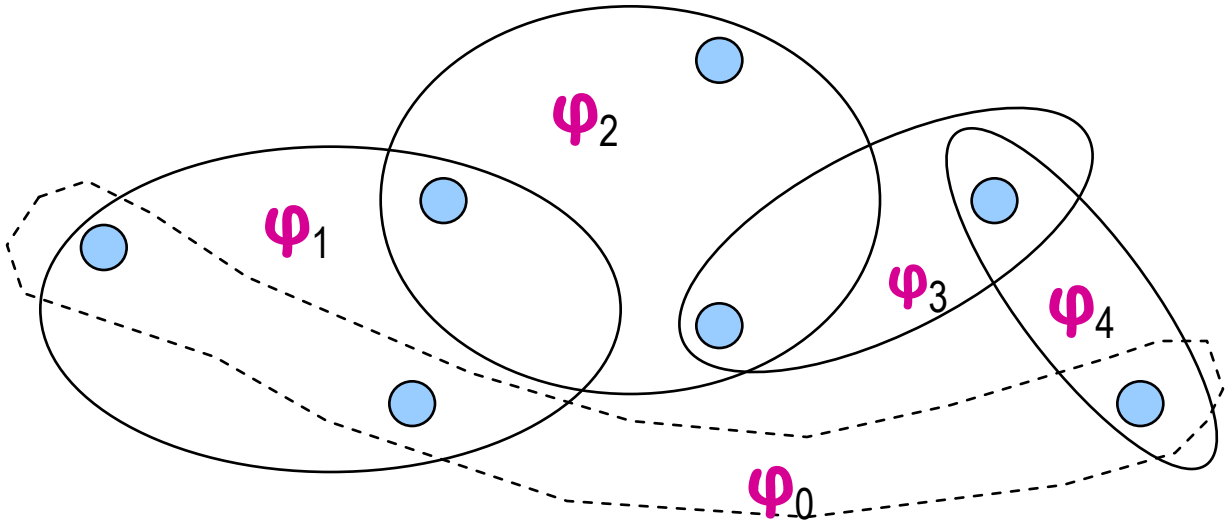
Expressive Power

- 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 $R_1, R_2, \ldots, R_k$



Expressive Power

- If we can combine the functions $\varphi_1, \varphi_2, \ldots, \varphi_k$ to obtain a derived cost function φ_0, then we say that φ_0 can be *expressed* using $\varphi_1, \varphi_2, \ldots, \varphi_k$



Expressive Power

Definition:

The "expressive power" of a valued constraint language **L**, denoted $\langle \mathbf{L} \rangle$, is defined to be the set of functions that can be obtained from functions in **L** using

- Summation
- Minimisation

Closure

Definition:

The “closure” of a valued constraint language **L**, denoted $\langle\langle \mathbf{L} \rangle\rangle$, is defined to be the set of functions that can be obtained from functions in **L** using

- Summation
- Minimisation
- Multiplication by a non-negative rational
- Addition of a constant

Closure and Reduction

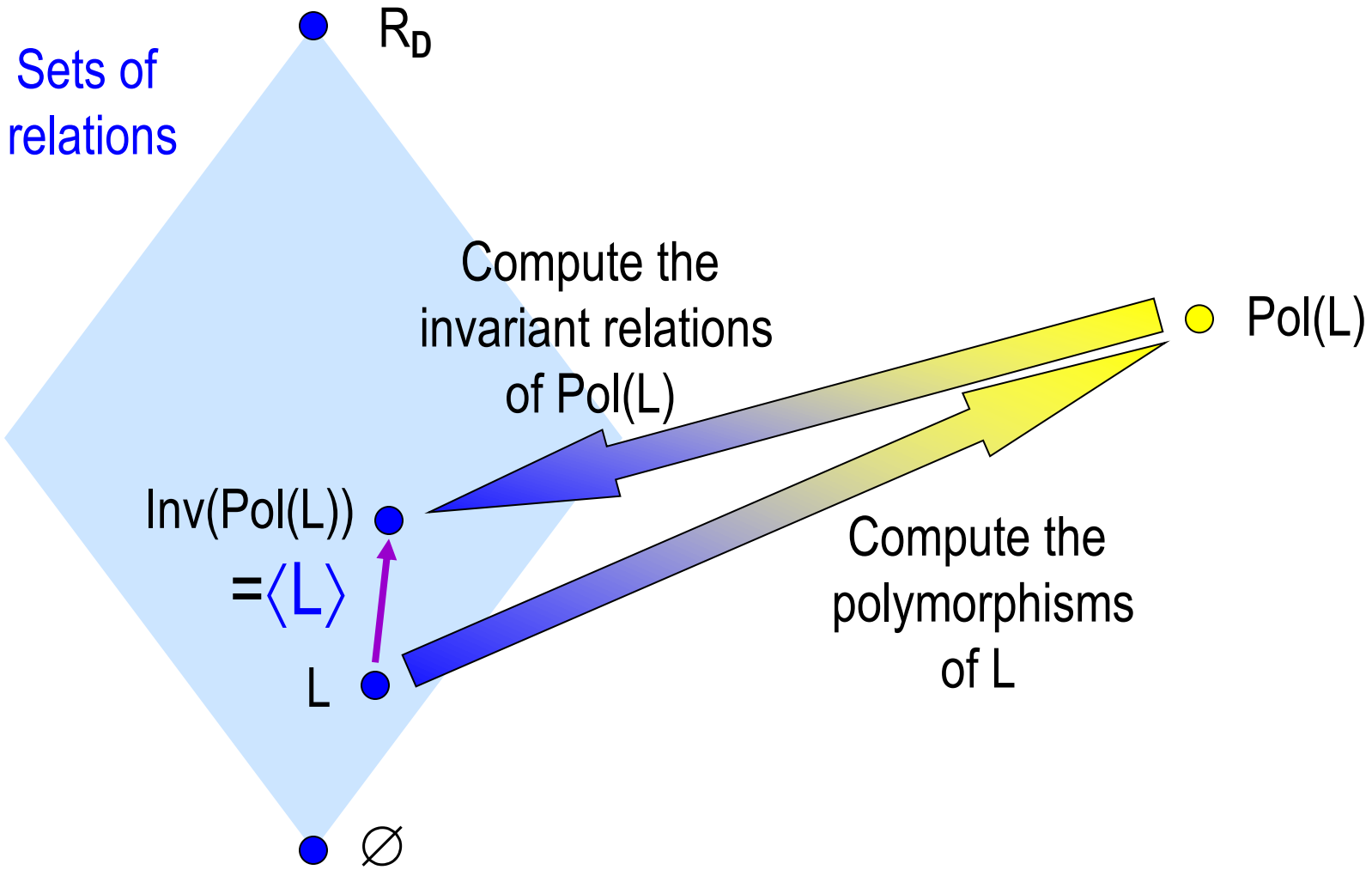
Theorem: (Cohen, Cooper, J. 06) For any valued constraint languages $\mathbf{L}$, $\mathbf{L}'$, if $\mathbf{L}'$ finite, and $\mathbf{L}' \subseteq \langle\langle \mathbf{L} \rangle\rangle$ then VCSP($\mathbf{L}'$) is polynomial-time reducible to VCSP($\mathbf{L}$)

Corollary: A valued constraint language $\mathbf{L}$ is (locally) tractable if and only if $\langle\langle \mathbf{L} \rangle\rangle$ is tractable;

similarly, $\mathbf{L}$ is NP-hard if and only if $\langle\langle \mathbf{L} \rangle\rangle$ is NP-hard.

Algebra?

Pol and Inv


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

Defining Pol

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

	x	y	z
	0	0	0
s →	0	0	1
	0	1	0
	0	1	1
t →	1	0	0
	1	0	1
	1	1	0
	1	1	1

0	0	1
1	0	0

1	0	1

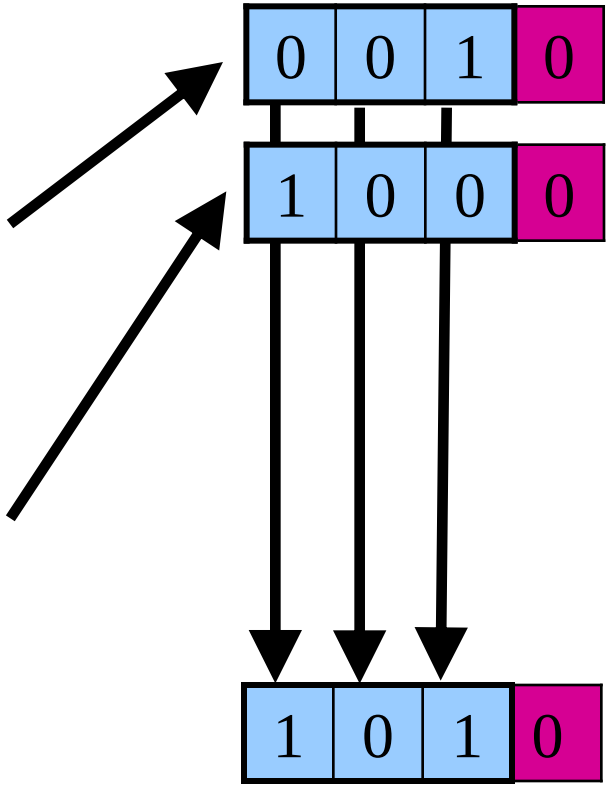
We say that this relation R has the **polymorphism** Maximum

Maximum

Generalizing Pol

$\forall s,t \;\; Cost(Max(s,t)) \leq Cost(s) + Cost(t)$

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

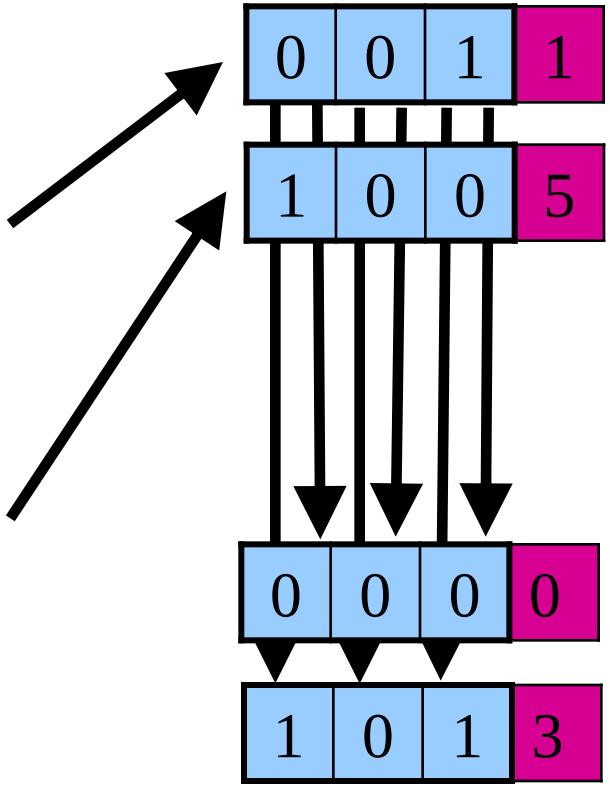


Maximum

Generalizing Pol

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

x	y	z	
0	0	0	0
0	0	1	1
0	1	0	7
0	1	1	1
1	0	0	5
1	0	1	3
1	1	0	∞
1	1	1	0



+ = 6

$\mathbb{V}$

Minimum
+
Maximum
= 3

Generalizing Pol

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

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 particular multimorphism is called ***submodular***)

If all cost functions are submodular
the problem is tractable

Tractable Cases

Any set of Boolean cost functions which all have one of these eight multimorphisms, 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

(see Cohen, Cooper, Jeavons, Krokhin, "Soft Constraints: Complexity and Multimorphisms" CP2003, pp.244-258)

Boolean Dichotomy Theorem

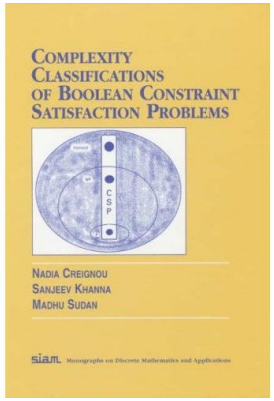
Any set of Boolean cost functions which all have one of these eight multimorphisms, is tractable:

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**.

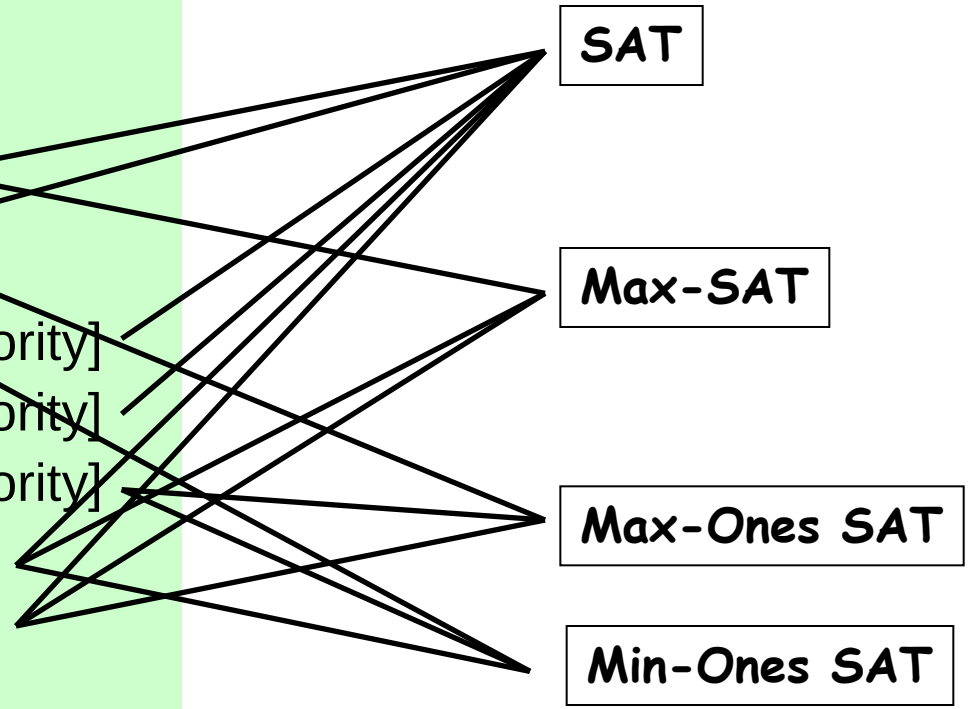
(Cohen, Cooper, J. CP'04)

Special Cases

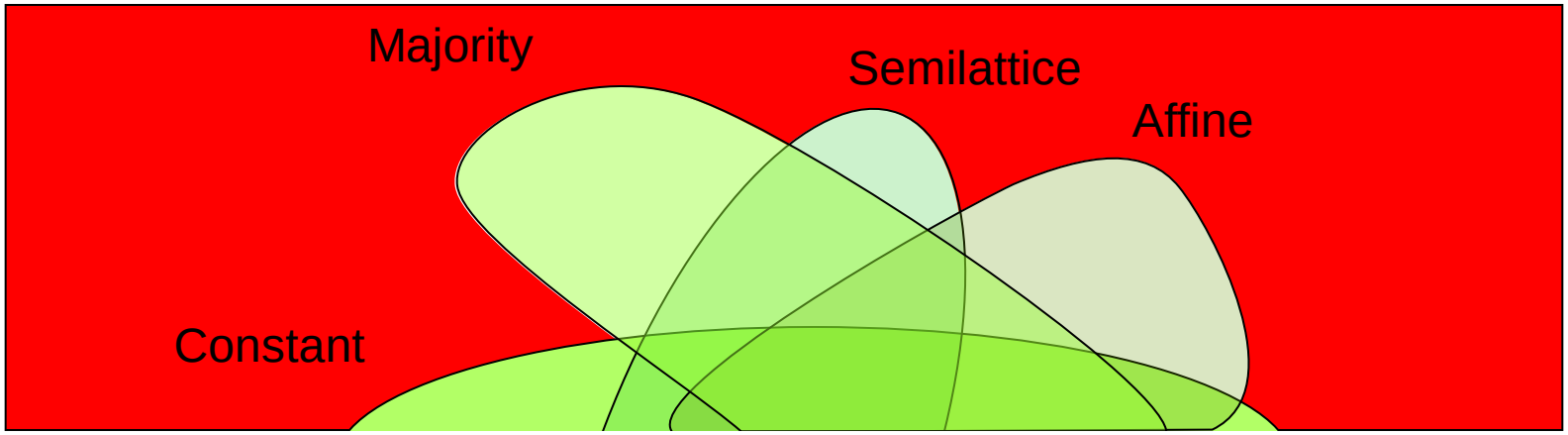


Any set of Boolean cost functions which all have one of these eight multimorphisms, is tractable:

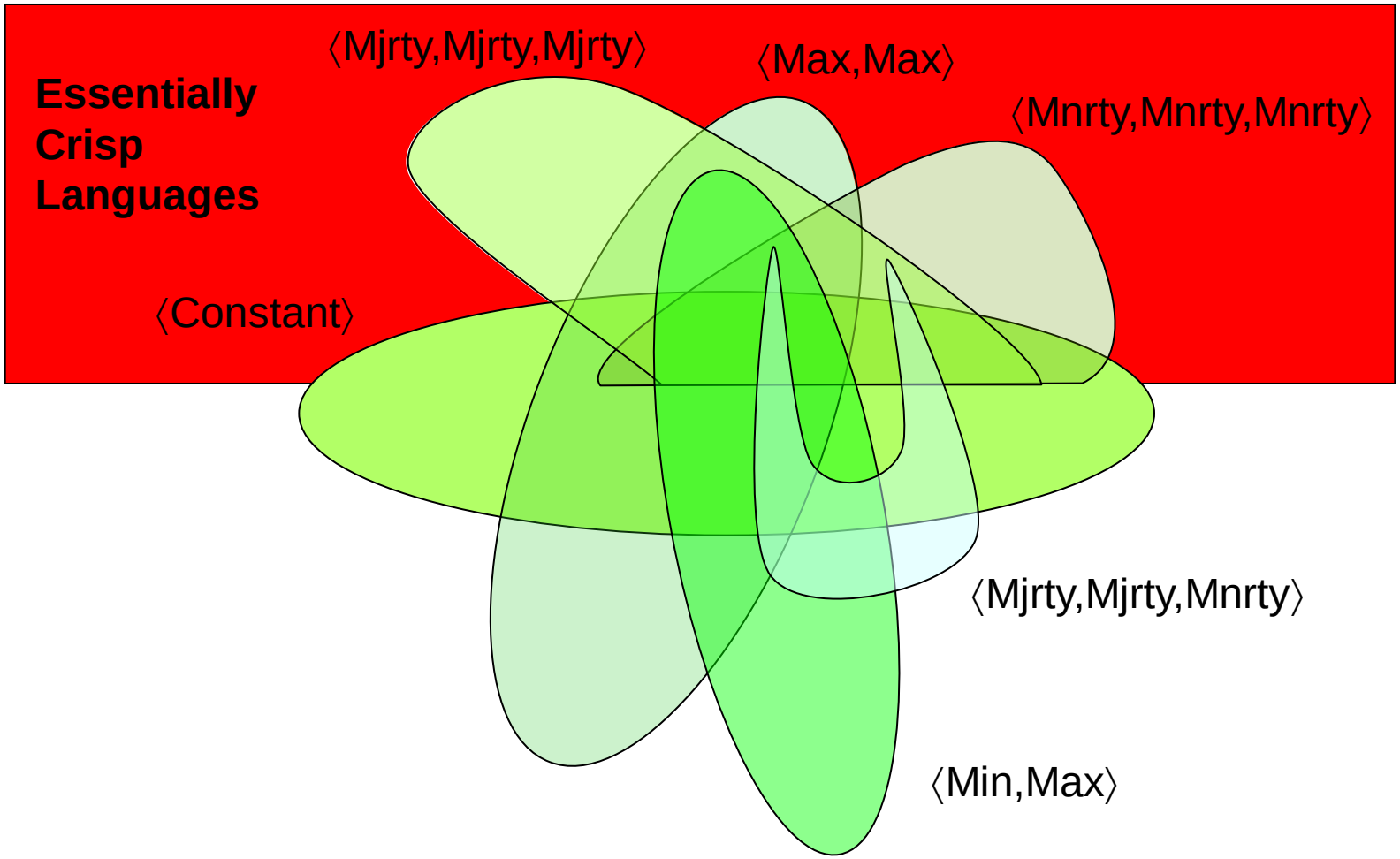
1) [Min,Max]
2) [Max,Max]
3) [Min,Min]
4) [Majority,Majority,Majority]
5) [Minority,Minority,Minority]
6) [Majority,Majority,Minority]
7) [Constant 0]
8) [Constant 1]

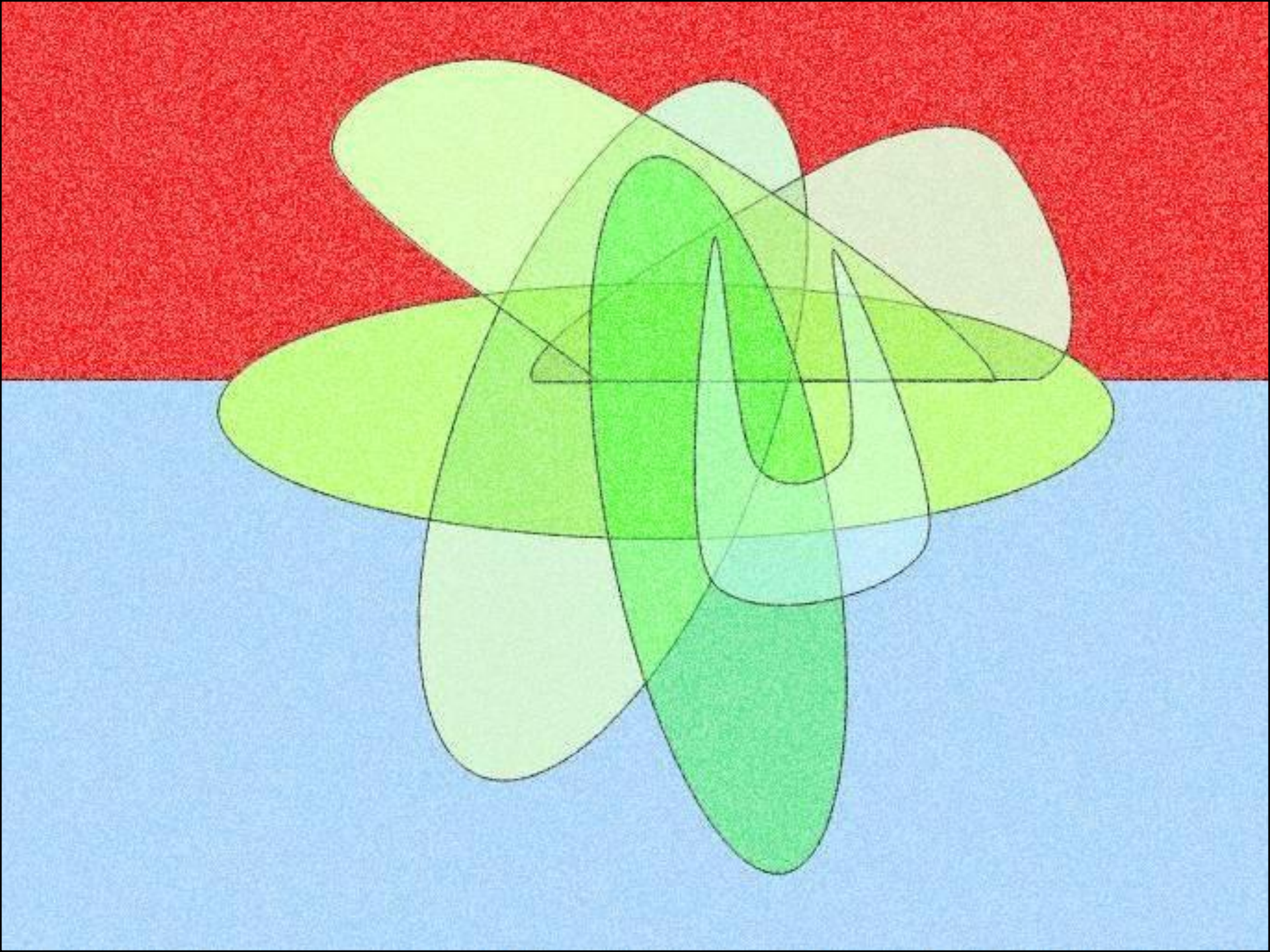


Tractable cases of CSP



Tractable cases of VCSP



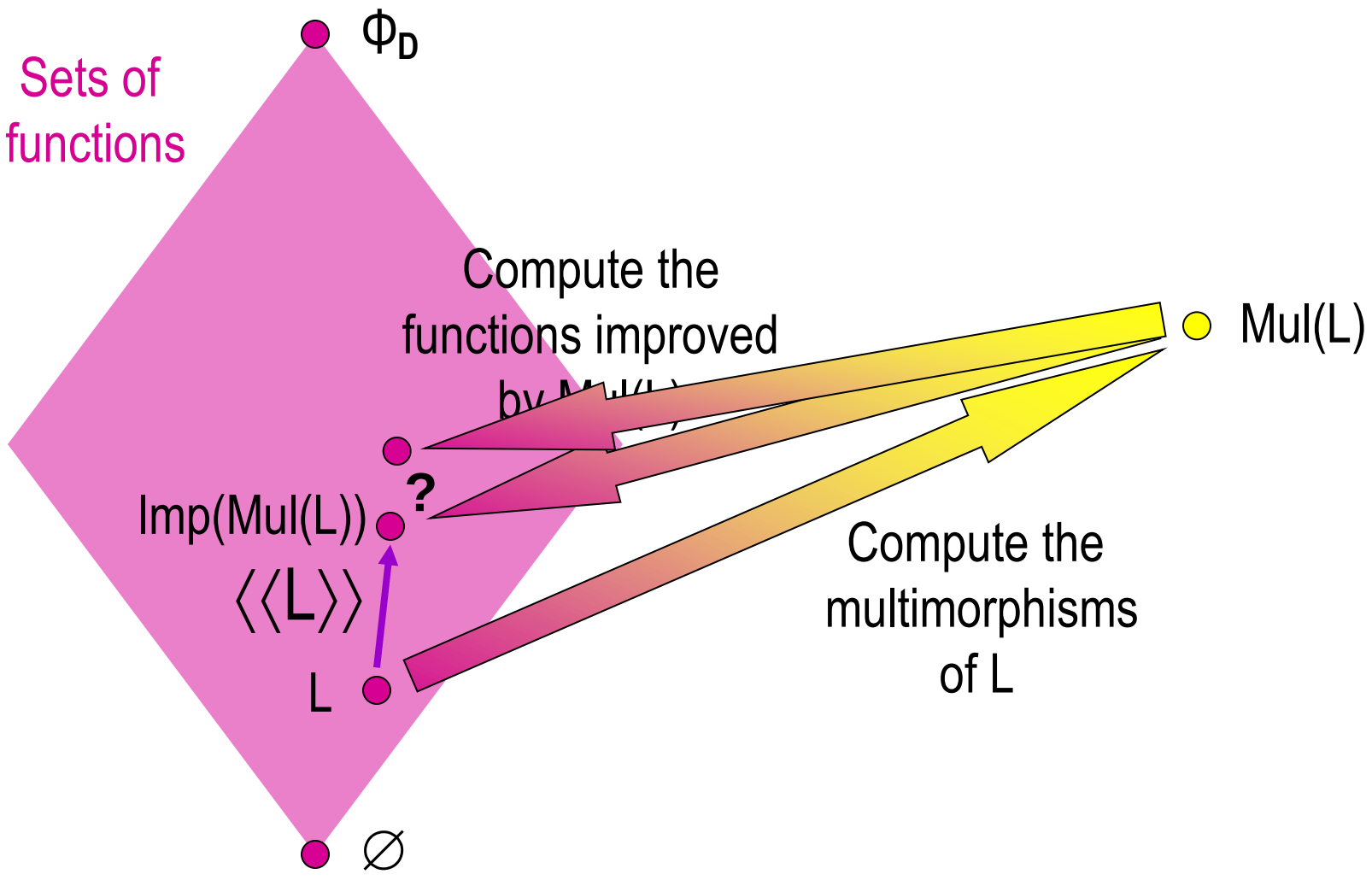


The past 100 days…

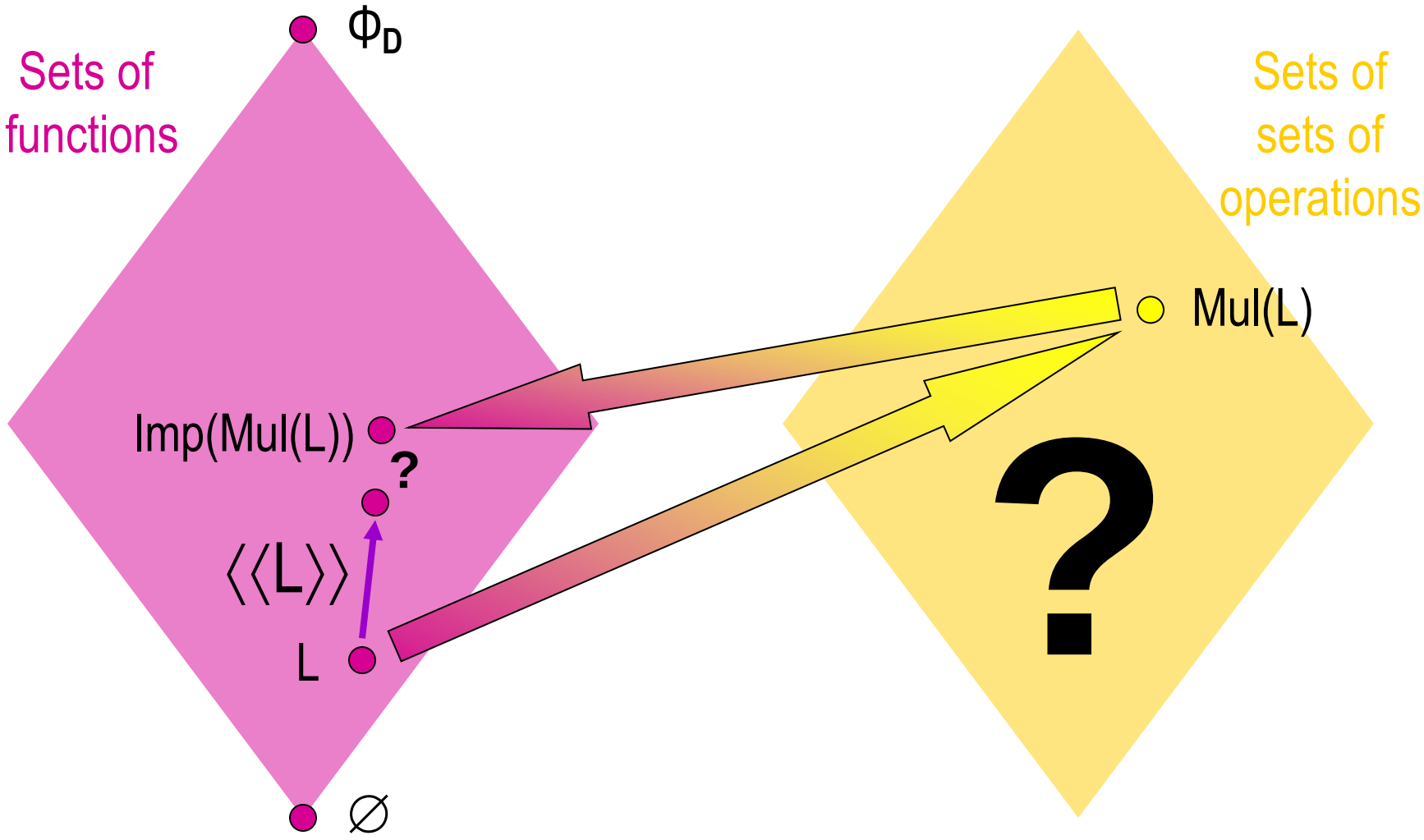
- **A new Galois connection to characterise the expressive power and complexity of valued constraint languages…**

An Algebraic Theory of Complexity for Discrete Optimisation,
David Cohen, Martin Cooper, Paidi Creed, PJ, Stanislav Zivny,
arxiv.org/abs/1207.6692 (submitted on 28 Jul, 2012)

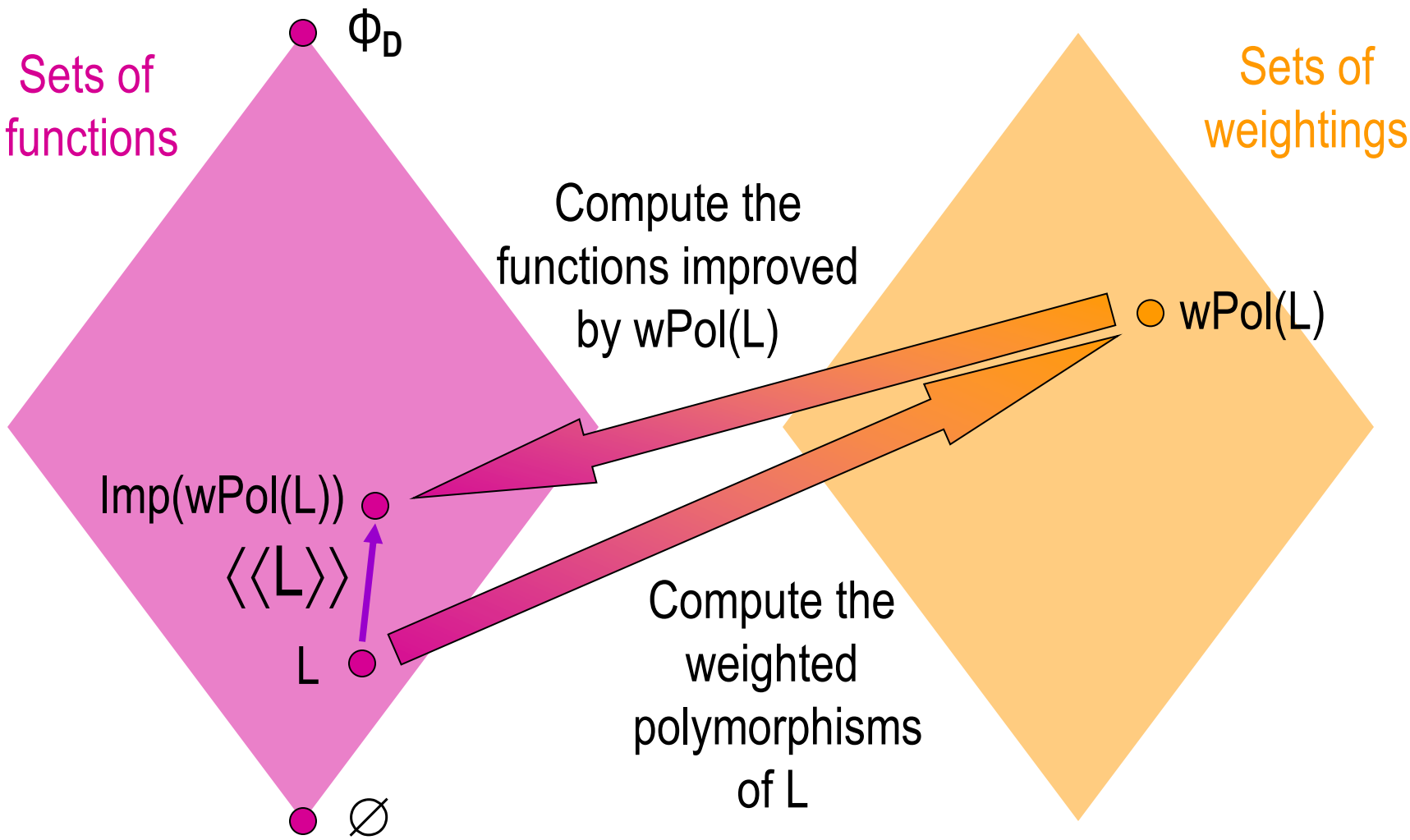
Mul and Imp


Sets of functions
Φ_D
Compute the functions improved by Mul(L)
Mul(L)
?
Imp(Mul(L))
⟨⟨L⟩⟩
L
Compute the multimorphisms of L
∅

Mul and Imp


Sets of functions
Φ_D
Imp(Mul(L))
?
⟨⟨L⟩⟩
L
∅
Sets of sets of operations
Mul(L)
?

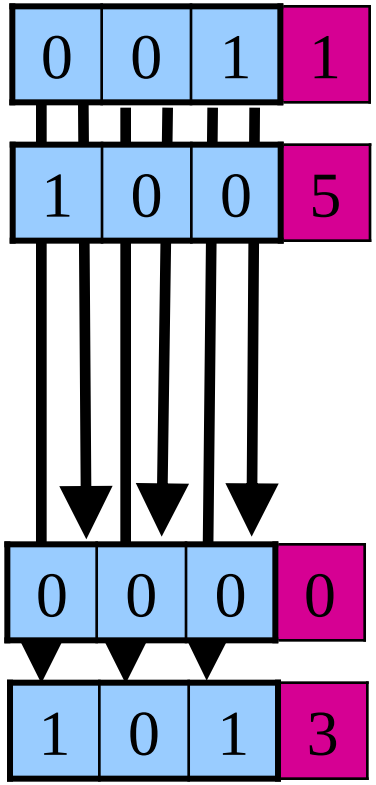
wPol and Imp


Sets of functions
Φ_D
Sets of weightings
Compute the functions improved by wPol(L)
wPol(L)
Imp(wPol(L))
⟨⟨L⟩⟩
L
Compute the weighted polymorphisms of L
∅

Generalizing Mul

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

x	y	z	
0	0	0	0
0	0	1	1
0	1	0	7
0	1	1	1
1	0	0	5
1	0	1	3
1	1	0	∞
1	1	1	0



Project$_1$
+
Project$_2$
= **6**

$\vee\!\!\vee$

Minimum
+
Maximum
= **3**

Generalizing Mul

x	y	z	
0	0	0	0
0	0	1	1
0	1	0	7
0	1	1	1
1	0	0	5
1	0	1	3
1	1	0	∞
1	1	1	0

0	0	1	1
1	0	0	5

Project$_1$
+
Project$_2$
= 0, 6

0	0	0	0
1	0	1	3

Minimum
+
Maximum
= -3, 3

Generalizing Mul

We now have a *function* that weights the operations

x	y	z	
0	0	0	0
0	0	1	1
0	1	0	7
0	1	1	1
1	0	0	5
1	0	1	3
1	1	0	∞
1	1	1	0

0	0	1	1	-1	**Project$_1$**
1	0	0	5	-1	**Project$_2$**
0	0	0	0	+1	**Minimum**
1	0	1	3	+1	**Maximum**

0

$\vee$

-3

Weightings

Definition: A k-ary **weighting**, **ω**, is a (partial) *function* from the k-ary operations on a set D to rational weights, such that:

1. Only projections can have negative weights
2. The sum of all the weights is 0

-1	**Project$_1$**
-1	**Project$_2$**
+1	**Minimum**
+1	**Maximum**

Weighted Polymorphism

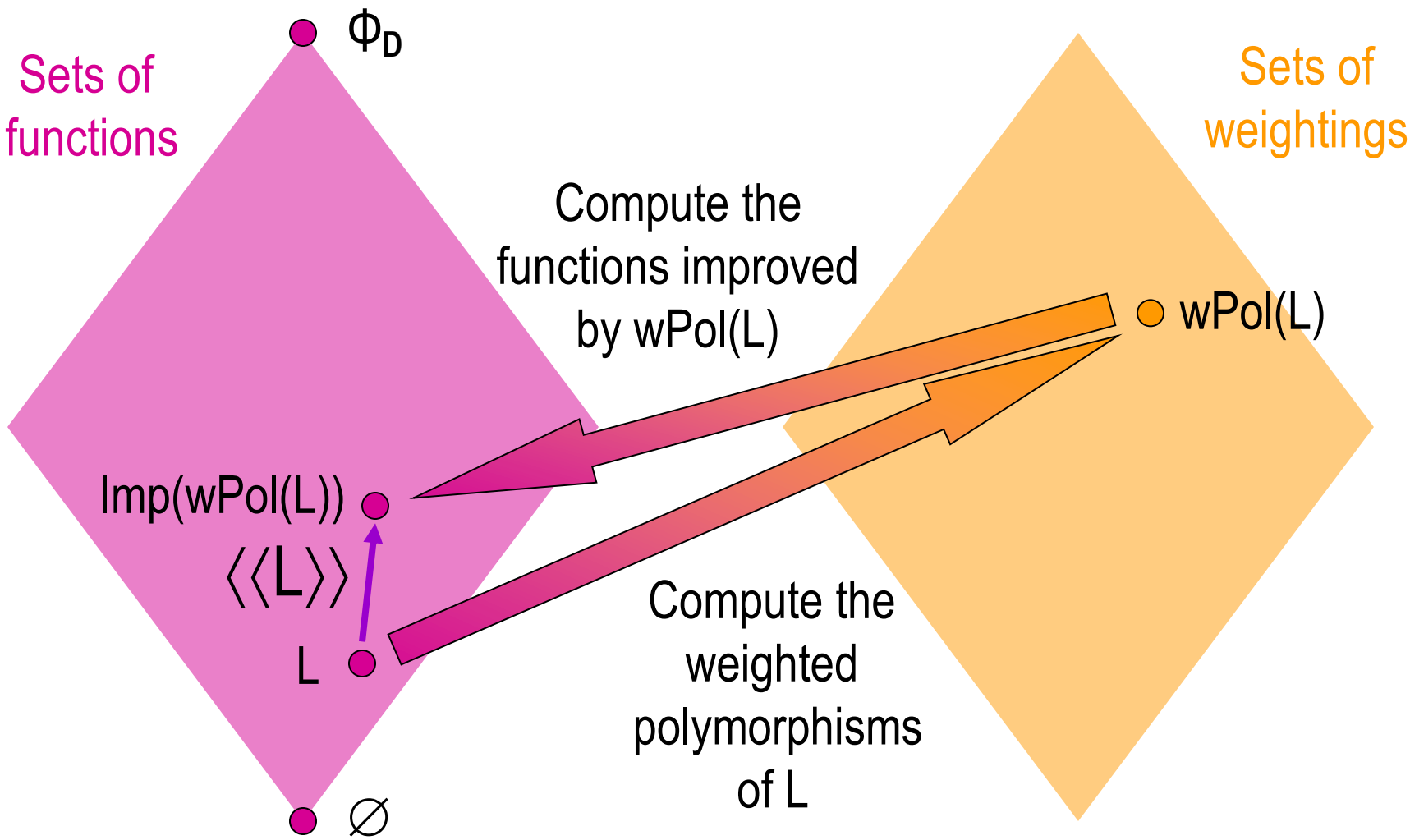
Definition: A k-ary weighting, ω, is a *weighted polymorphism* of a cost function φ, if,

for all $x_1, x_2, \ldots x_k$, $\sum_f \omega(f)\varphi(f(x_1, x_2, \ldots x_k)) \leq 0$

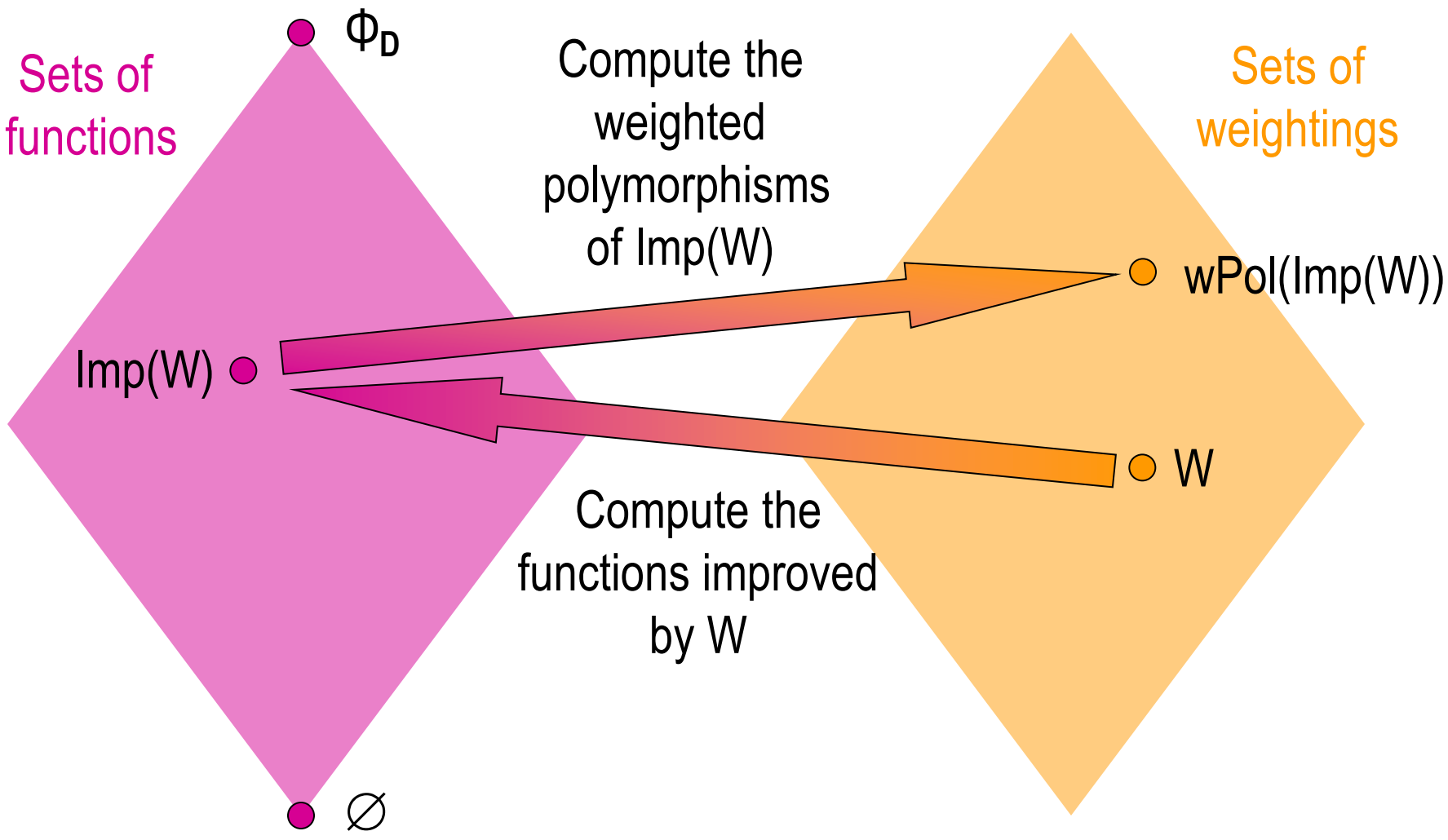
0	0	1	1	-1	**Project$_1$**
1	0	0	5	-1	**Project$_2$**
0	0	0	0	+1	**Minimum**
1	0	1	3	+1	**Maximum**

$$-3 \leqslant 0$$

wPol and Imp


Sets of functions
Φ_D
Sets of weightings
Compute the functions improved by wPol(L)
wPol(L)
Imp(wPol(L))
⟨⟨L⟩⟩
L
Compute the weighted polymorphisms of L
∅

wPol and Imp


Sets of functions
Φ_D
Compute the weighted polymorphisms of Imp(W)
Sets of weightings
wPol(Imp(W))
Imp(W)
W
Compute the functions improved by W
∅

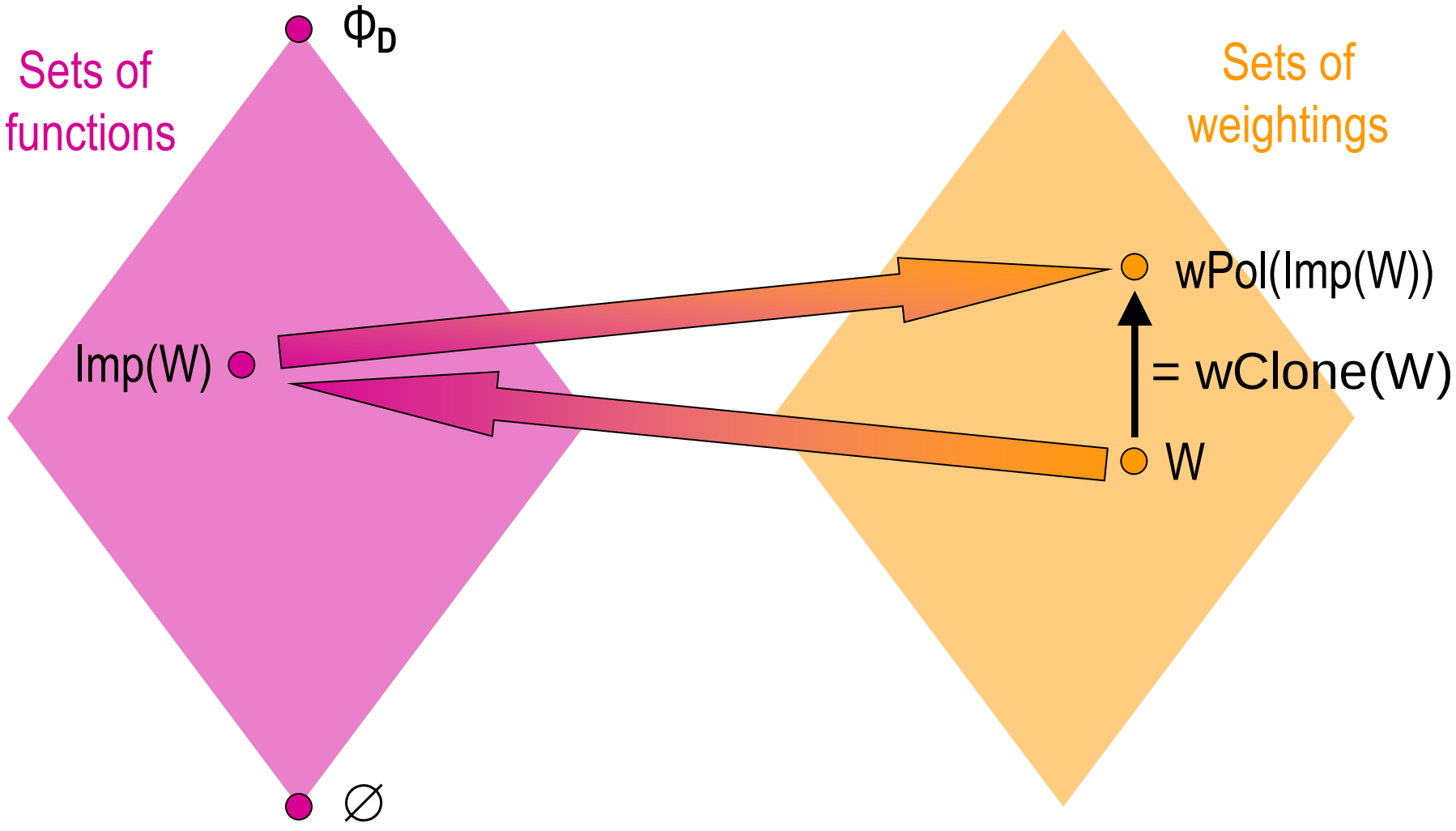
Weighted Clones

Definition: A set of weightings of a clone C is a *weighted clone* if it is closed under:

1. Addition: $\omega_1 + \omega_2$
2. Scaling: $c\,\omega \quad (c \in \mathbb{Q}^{\geq 0})$
3. Superposition: $\omega[g_1,\ldots g_k]$

where $$\omega[g_1,\ldots g_k](f) = \sum_{\{f' \mid f = f'[g_1,\ldots g_k]\}} \omega(f')$$

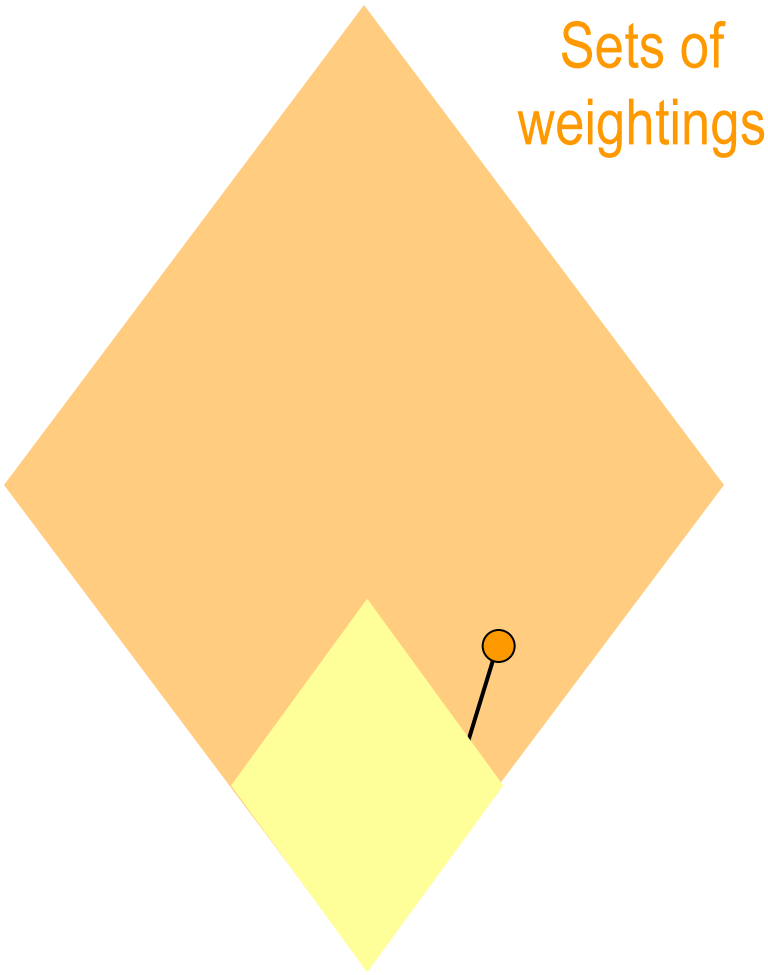
wPol and Imp


Sets of functions
Φ_D
Imp(W)
∅
Sets of weightings
wPol(Imp(W))
= wClone(W)
W

Minimal Weighted Clones

Definition: A weighted clone is zero-valued if every weighting in it is zero-valued

Definition: A non-zero weighted clone is *minimal* if every non-zero weighting it contains is a generator



Minimal Weighted Clones

Theorem: Any non-zero weighted clone W must contain a weighting that assigns positive weight to either:

1. A set of unary operations that are not projections; or
2. A set of binary idempotent operations that are not projections; or
3. A set of ternary sharp operations (majority operations, minority operations, Pixley operations or semiprojections); or
4. A set of k-ary semiprojections (for some k > 3)..

Corollary: Any non-zero weighted clone W *on the Boolean domain* must contain a weighting **ω** that assigns positive weight to either:

1. Exactly one of the unary operations constant 0, constant 1, or inversion;
2. Exactly one of the binary operations min and max, or both of them equally;
3. Exactly one of the ternary operations Majority and Minority, or both of them with **ω**(Majority) = 2**ω**(Minority).

The past 100 days…

- **A new Galois connection to characterise the expressive power and complexity of valued constraint languages using *weighted polymorphisms*…**

  An Algebraic Theory of Complexity for Discrete Optimisation,
  David Cohen, Martin Cooper, Paidi Creed, PJ, Stanislav Zivny,
  arxiv.org/abs/1207.6692 (submitted on 28 Jul, 2012)

- **A new dichotomy for *finite-valued* constraint languages on a *3-element* domain, showing there are infinitely many distinct tractable cases (all characterised by *binary* weighted polymorphisms) …**

  Skew Bisubmodularity and Valued CSPs,
  Anna Huber, Andrei Krokhin, and Robert Powell
  to appear in SODA 2013 (accepted on Sep 15, 2012)

- **A complete characterisation of tractable finite-valued constraint languages over *arbitrary finite domains*…**

  The complexity of finite-valued CSPs,
  Johan Thapper, Stanislav Zivny
  arxiv.org/abs/1210.2987 (submitted on 10 Oct, 2012)

Summary

What do we get from the CSP/VCSP model?

- A unified approach to combinatorial search and optimisation problems
- An understanding of tractable cases that could be exploited by general constraint solving tools
- A link between efficient algorithms and structural properties/polymorphisms
- Generic hardness proofs without reductions
- A bridge into a rich algebraic theory
- New mathematical approach to the whole problem space…