

A Galois Connection for Valued Constraints

Peter Jeavons
University of Oxford

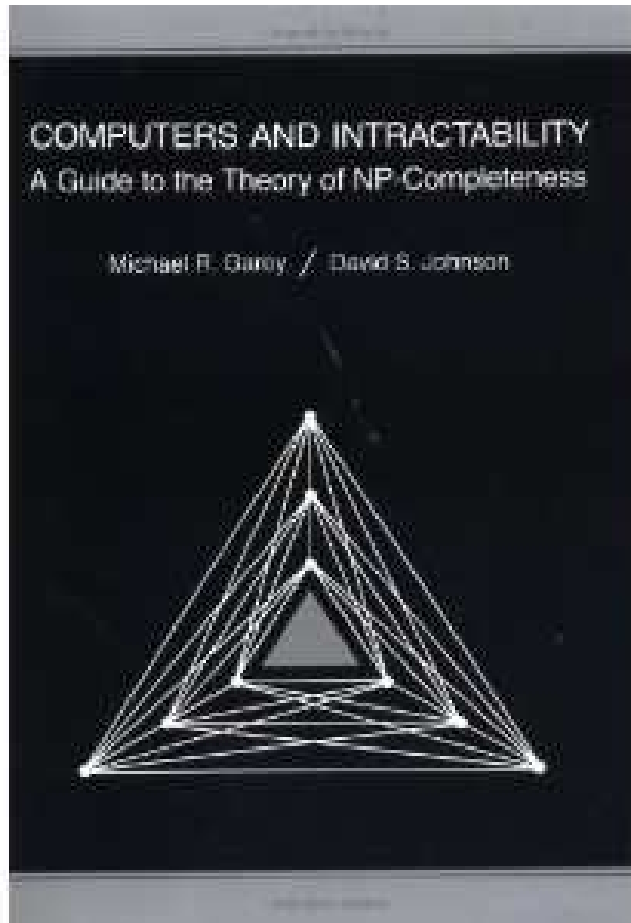
Joint work with
David Cohen, Martin Cooper, Páidí Creed, Standa Živný

Outline

- Background *CSP is good, but...*
- Generalizing *CSP \rightarrow VCSP*
- Algebra *Polymorphisms \rightarrow
Multimorphisms*
- Generalizing again *Multimorphisms \rightarrow
Weighted Polymorphisms*
- Galois connection *and where it takes us...*

Background

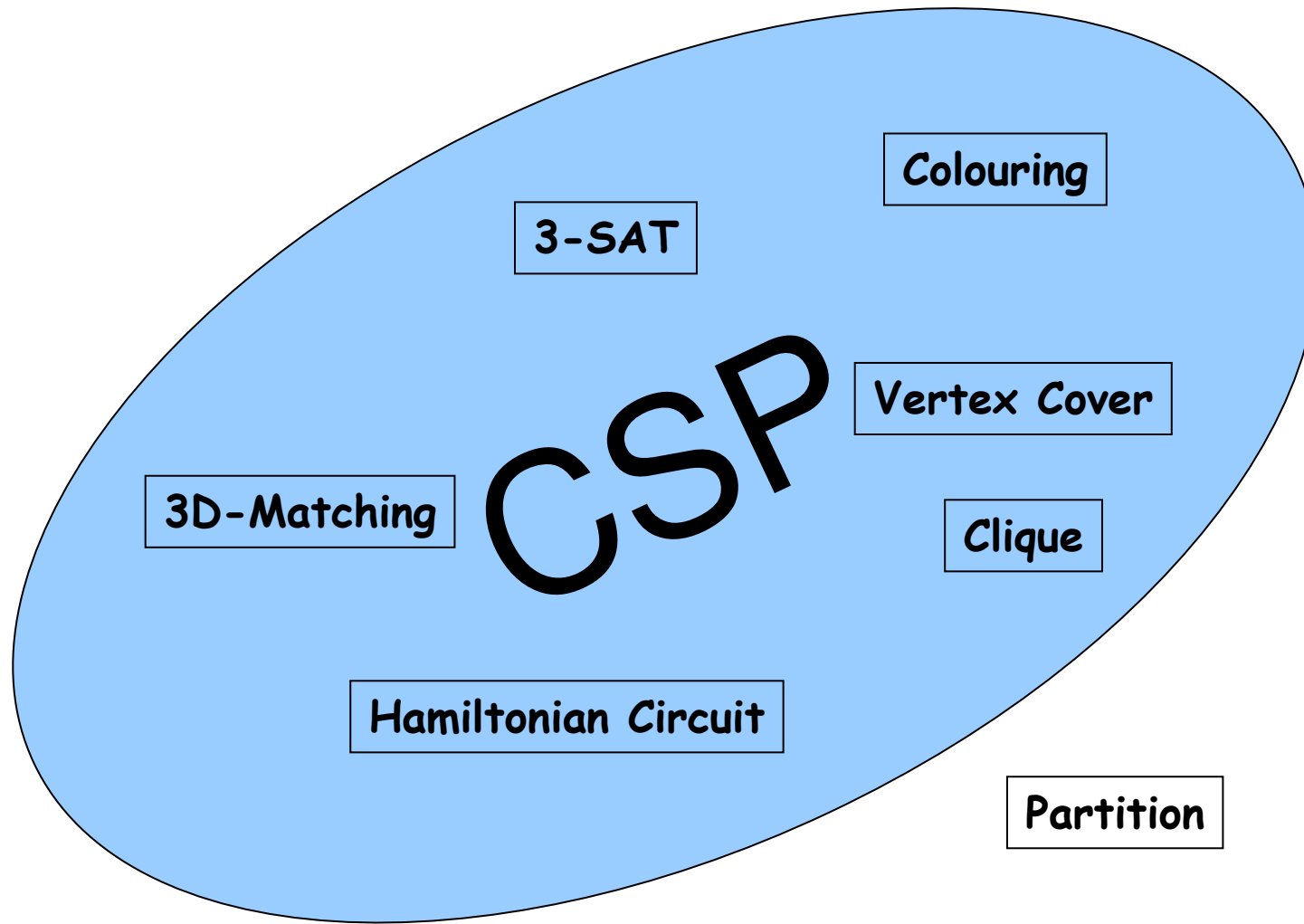
Birth of Complexity Theory



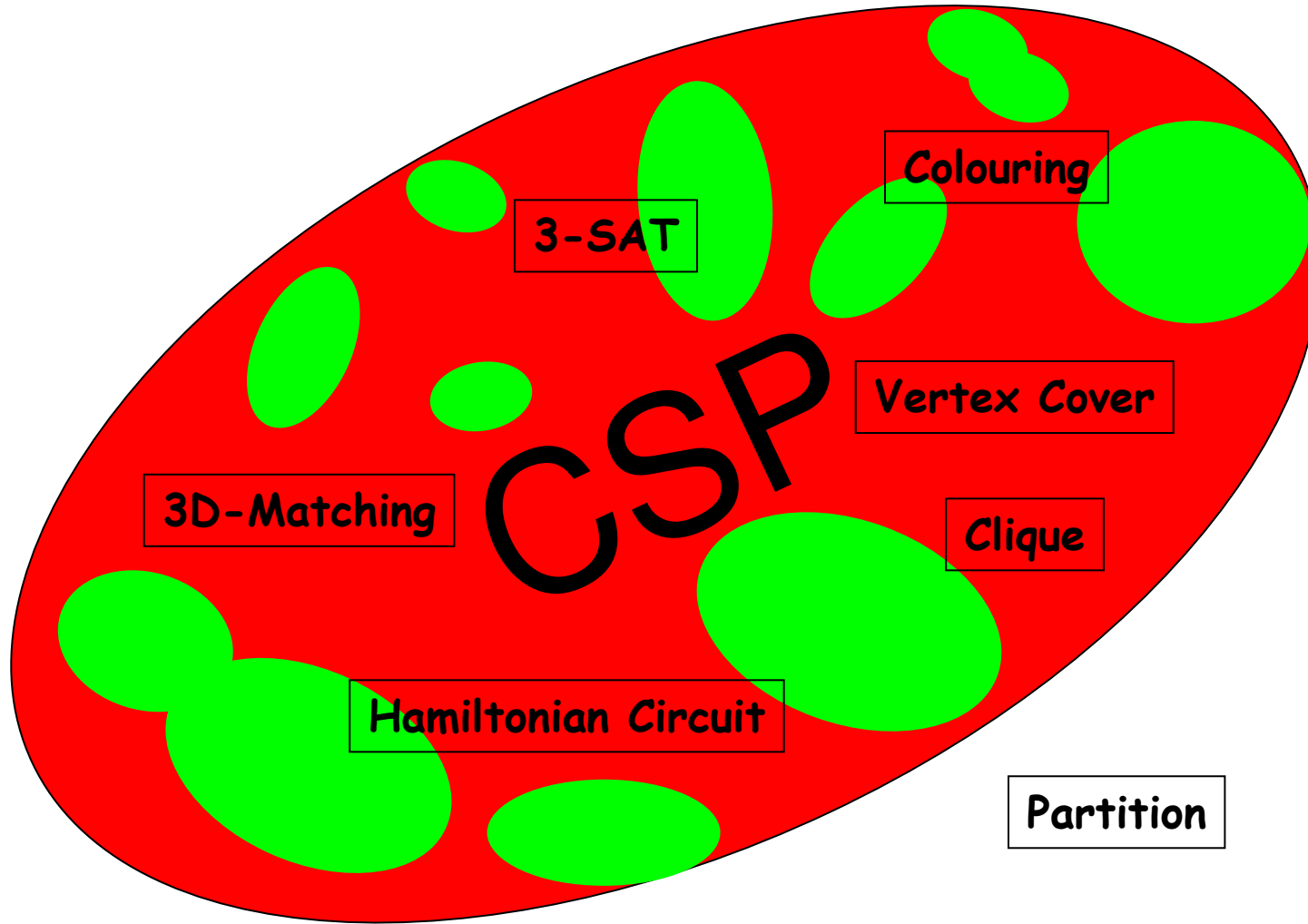
"The progress of science is the discovery at each step of a new order which gives unity to what had seemed unlike"

Jacob Bronowski

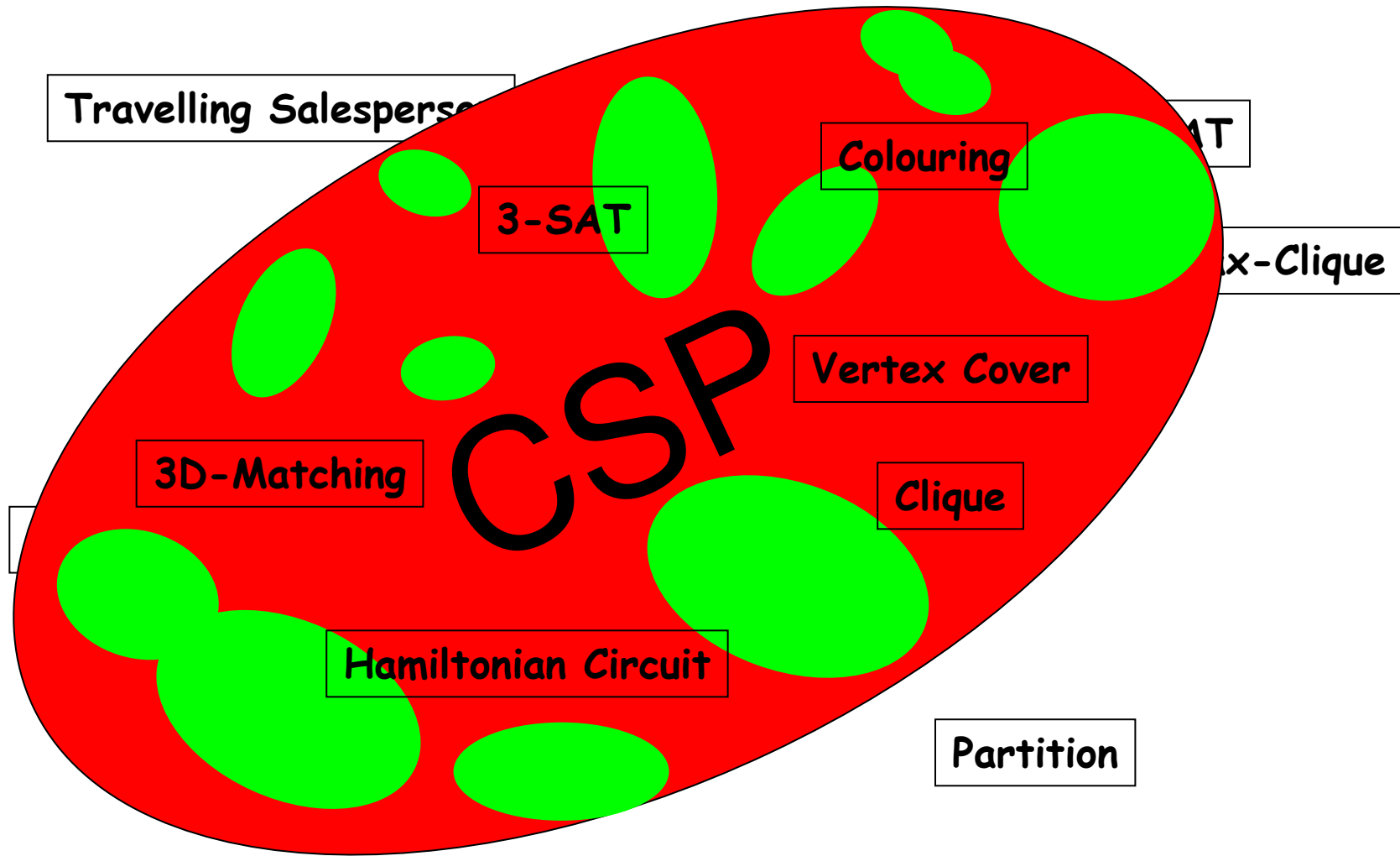
“Basic” Problems



Complexity Classification



A Bigger Picture



Generalization

So let's generalize...

Definition 1a:

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

$$R_1(\underline{s}_1) \wedge R_2(\underline{s}_2) \wedge \dots \wedge R_m(\underline{s}_m)$$

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

So let's generalize...

Definition 1b:

- An instance of **CSP** is defined to be a pair of similar relational structures:

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

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

So let's generalize...

Definition 1c:

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

Each constraint in \mathbf{C} is a pair (\mathbf{s}, \mathbf{R}) where

- \mathbf{s} is a list of variables defining the scope
- \mathbf{R} is a relation defining the allowed combinations of values

Definition of CSP

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

Each constraint in \mathbf{C} is a pair (\mathbf{s}, \mathbf{R}) where

- \mathbf{s} is a list of variables defining the scope
- \mathbf{R} is a relation defining the allowed combinations of values

Definition of VCSP

- An *instance* of VCSP 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* (ordered, can be added)

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

- s is a list of variables defining the scope
- φ is a *function* 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

VSAT

Very general discrete
optimization problem

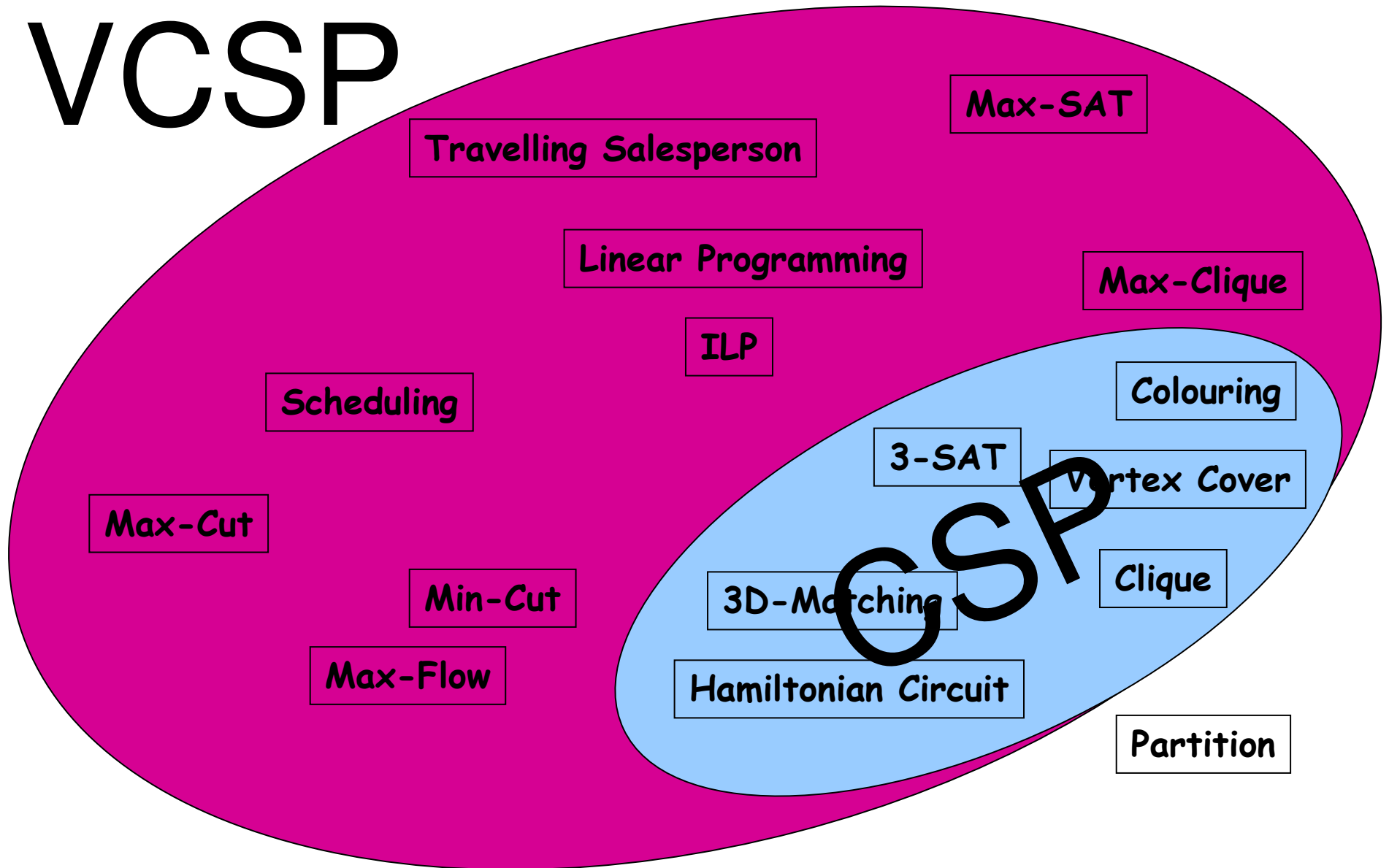
NP-hard



x	y	z	
0	0	0	0
0	0	1	$\frac{5}{8}$
0	1	0	7
0	1	1	1
1	0	0	∞
1	0	1	$\frac{1}{3}$
1	1	0	∞
1	1	1	0

A Bigger Picture

VCSP



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, \mathbf{L} , we have a corresponding class of instances, $\text{VCSP}(\mathbf{L})$...

Where the *question* is to find an assignment with *minimal total cost*.

Valued Constraint Languages

Definition: A *valued constraint language* is a set of functions from D^n to $\mathbb{Q}_+ \cup \{\infty\}$ for some fixed finite set D .

For every valued constraint language, \mathbf{L} , we have a corresponding class of instances, $\text{VCSP}(\mathbf{L})$...

Where the *question* is to find an assignment with *minimal total cost*.

General Question

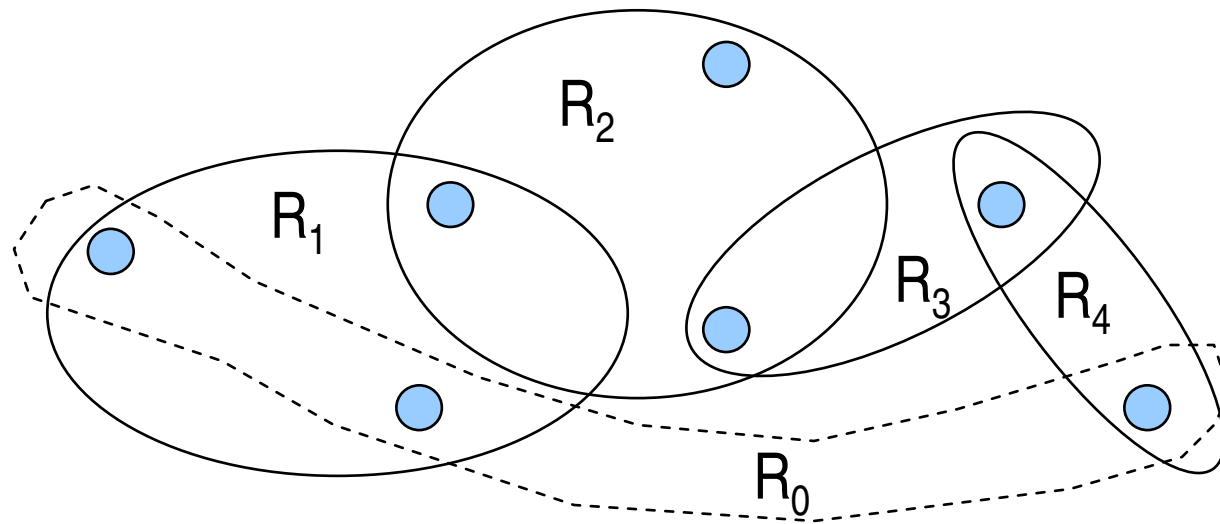
- Having a *general* formulation for all of these problems allows us to ask *general* complexity questions:

When is VCSP(L)
tractable?

Reductions

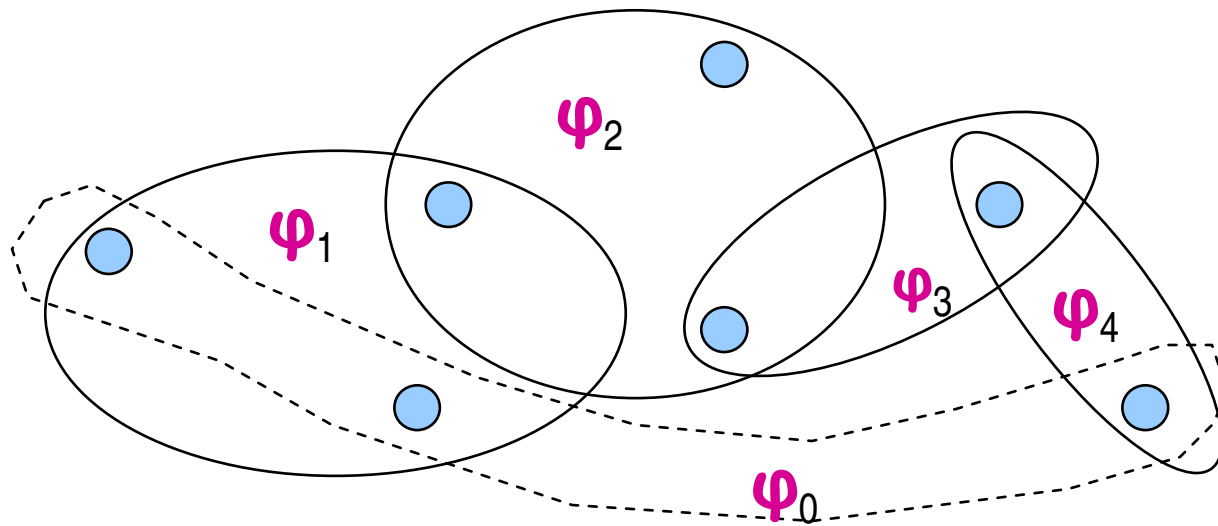
Expressive Power

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



Expressive Power

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



Expressive Power

Definition:

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

- Conjunction
- Existential quantification

Expressive Power

Definition:

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

- Summation
- Minimisation

Expressive Power and Reduction

Theorem: (J. 98) For any constraint languages L, L' ,
if L' finite, and $L' \subseteq \langle L \rangle$
then $\text{CSP}(L')$ is polynomial-time reducible to $\text{CSP}(L)$

Expressive Power and Reduction

Theorem: (J. 98) For any constraint languages L, L' ,
if L' finite, and $L' \subseteq \langle L \rangle$
then $CSP(L')$ is polynomial-time reducible to $CSP(L)$

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

Closure

Definition:

The “closure” of a valued constraint language L , denoted $\langle\langle L \rangle\rangle$, 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

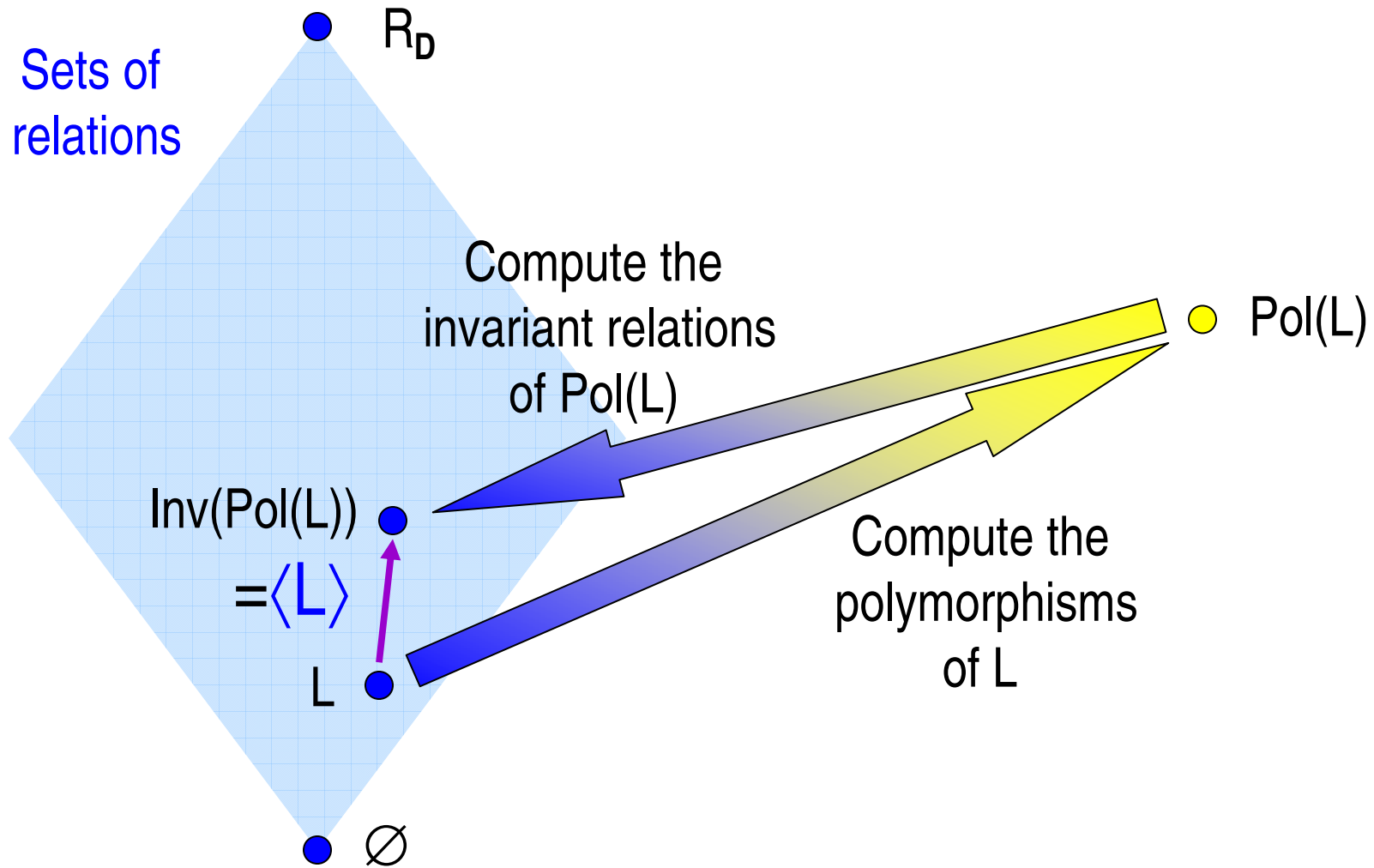
Closure and Reduction

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 $\text{VCSP}(\mathbf{L}')$ is polynomial-time reducible to $\text{VCSP}(\mathbf{L})$

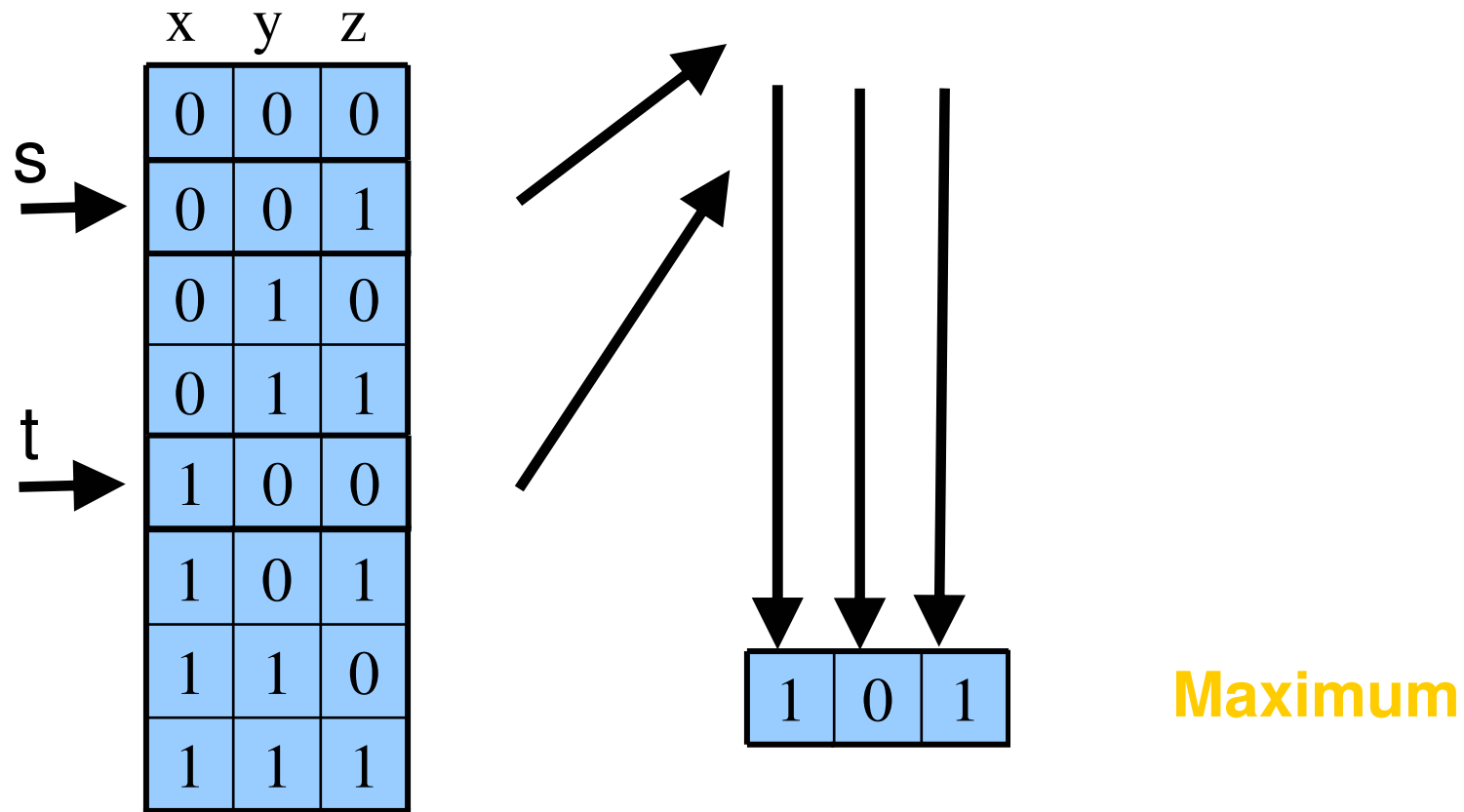
Corollary: A **valued** constraint language \mathbf{L} is (locally) **tractable** if and only if $\langle\langle \mathbf{L} \rangle\rangle$ is tractable;
similarly, \mathbf{L} is **NP-hard** if and only if $\langle\langle \mathbf{L} \rangle\rangle$ is **NP-hard**.

Algebra?

Pol and Inv

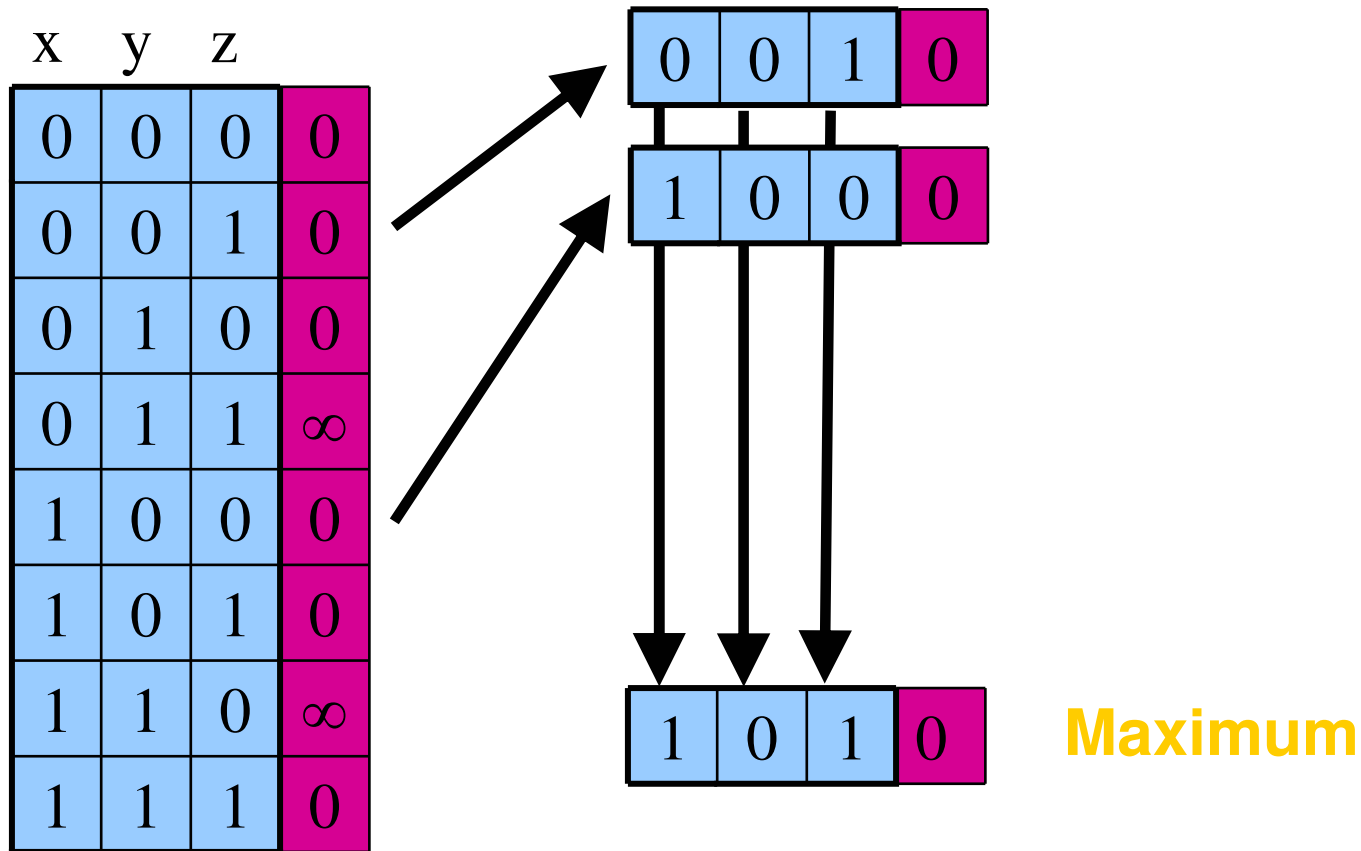


Defining Pol



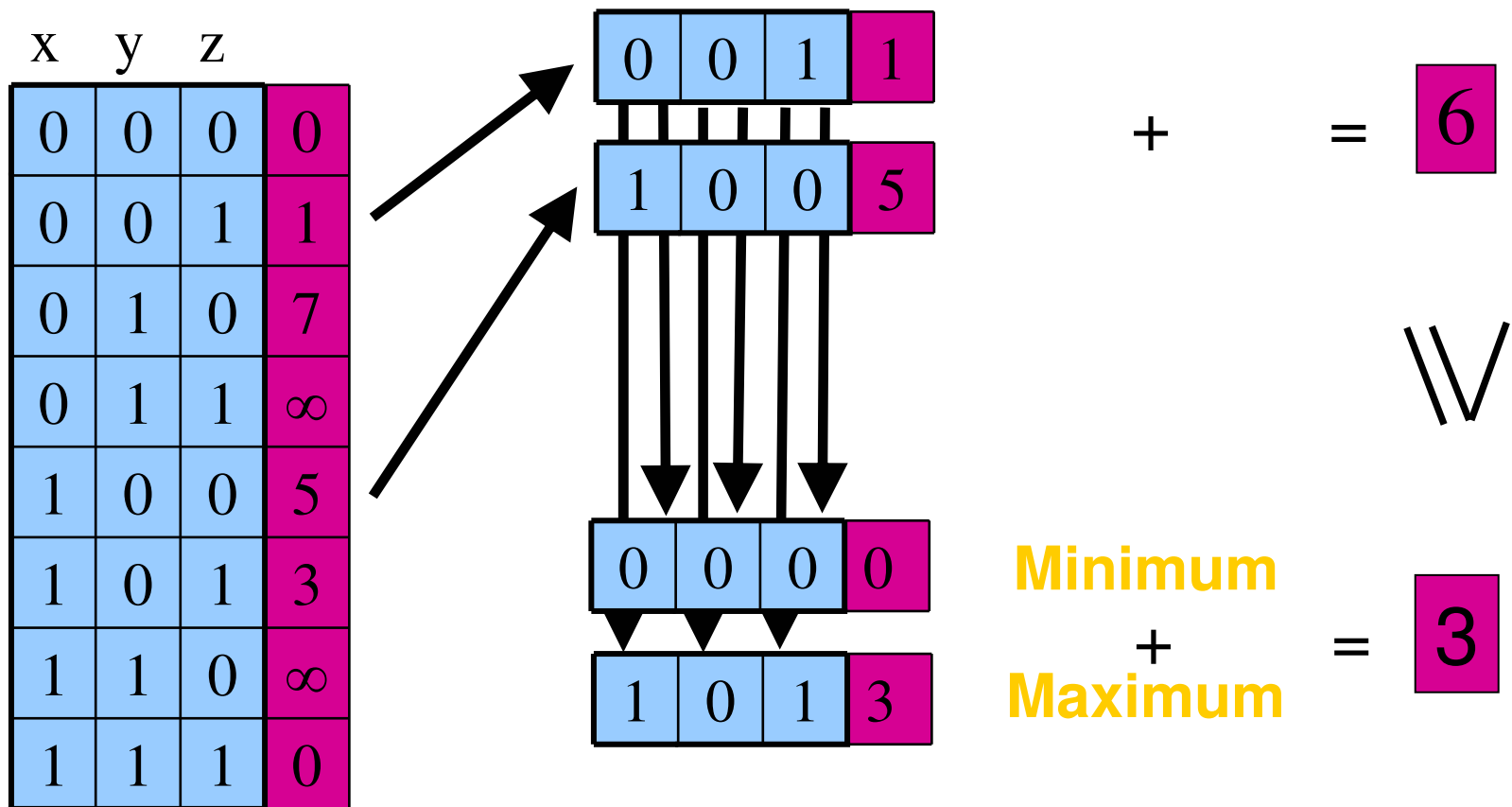
Generalizing Pol

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



Generalizing Pol

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



Generalizing Pol

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

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

We say that the cost function has the *multimorphism* $\langle \text{Min}, \text{Max} \rangle$

(any cost function with this particular multimorphism is called *submodular*)

Tractable Cases

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

- 1) $\langle \text{Min}, \text{Max} \rangle$
- 2) $\langle \text{Max}, \text{Max} \rangle$
- 3) $\langle \text{Min}, \text{Min} \rangle$
- 4) $\langle \text{Majority}, \text{Majority}, \text{Majority} \rangle$
- 5) $\langle \text{Minority}, \text{Minority}, \text{Minority} \rangle$
- 6) $\langle \text{Majority}, \text{Majority}, \text{Minority} \rangle$
- 7) $\langle \text{Constant } 0 \rangle$
- 8) $\langle \text{Constant } 1 \rangle$

Note: These are tractable cases for all finite domains

(Cohen et al, CP'03)

Boolean Dichotomy Theorem

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

- 1) $\langle \text{Min}, \text{Max} \rangle$
- 2) $\langle \text{Max}, \text{Max} \rangle$
- 3) $\langle \text{Min}, \text{Min} \rangle$
- 4) $\langle \text{Majority}, \text{Majority}, \text{Majority} \rangle$
- 5) $\langle \text{Minority}, \text{Minority}, \text{Minority} \rangle$
- 6) $\langle \text{Majority}, \text{Majority}, \text{Minority} \rangle$
- 7) $\langle \text{Constant } 0 \rangle$
- 8) $\langle \text{Constant } 1 \rangle$

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

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

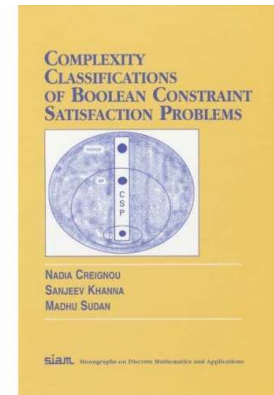
- 1) $\langle \text{Min}, \text{Max} \rangle$
- 2) $\langle \text{Max}, \text{Max} \rangle$
- 3) $\langle \text{Min}, \text{Min} \rangle$
- 4) $\langle \text{Majority}, \text{Majority}, \text{Majority} \rangle$
- 5) $\langle \text{Minority}, \text{Minority}, \text{Minority} \rangle$
- 6) $\langle \text{Majority}, \text{Majority}, \text{Minority} \rangle$
- 7) $\langle \text{Constant } 0 \rangle$
- 8) $\langle \text{Constant } 1 \rangle$

SAT

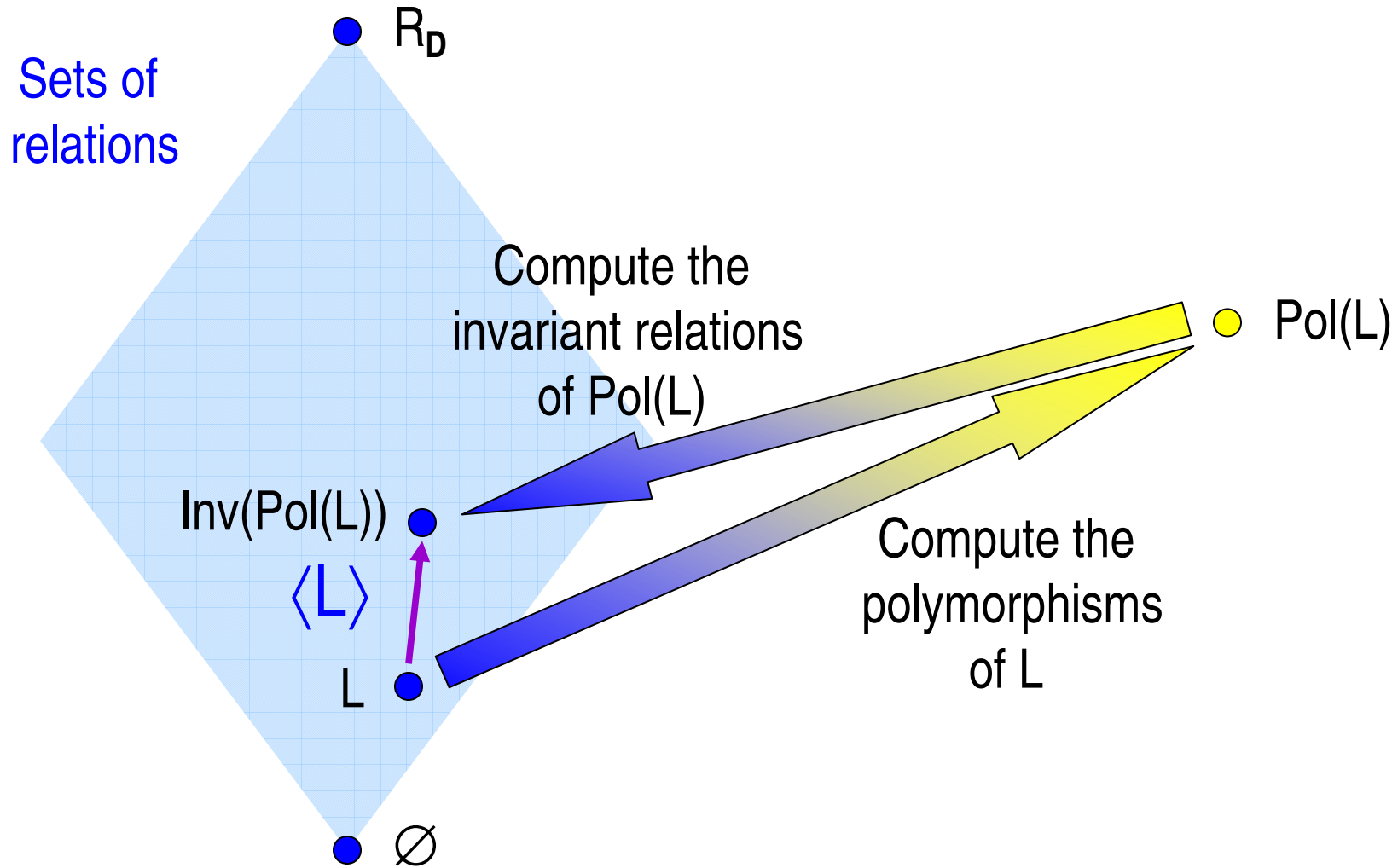
Max-SAT

Max-Ones SAT

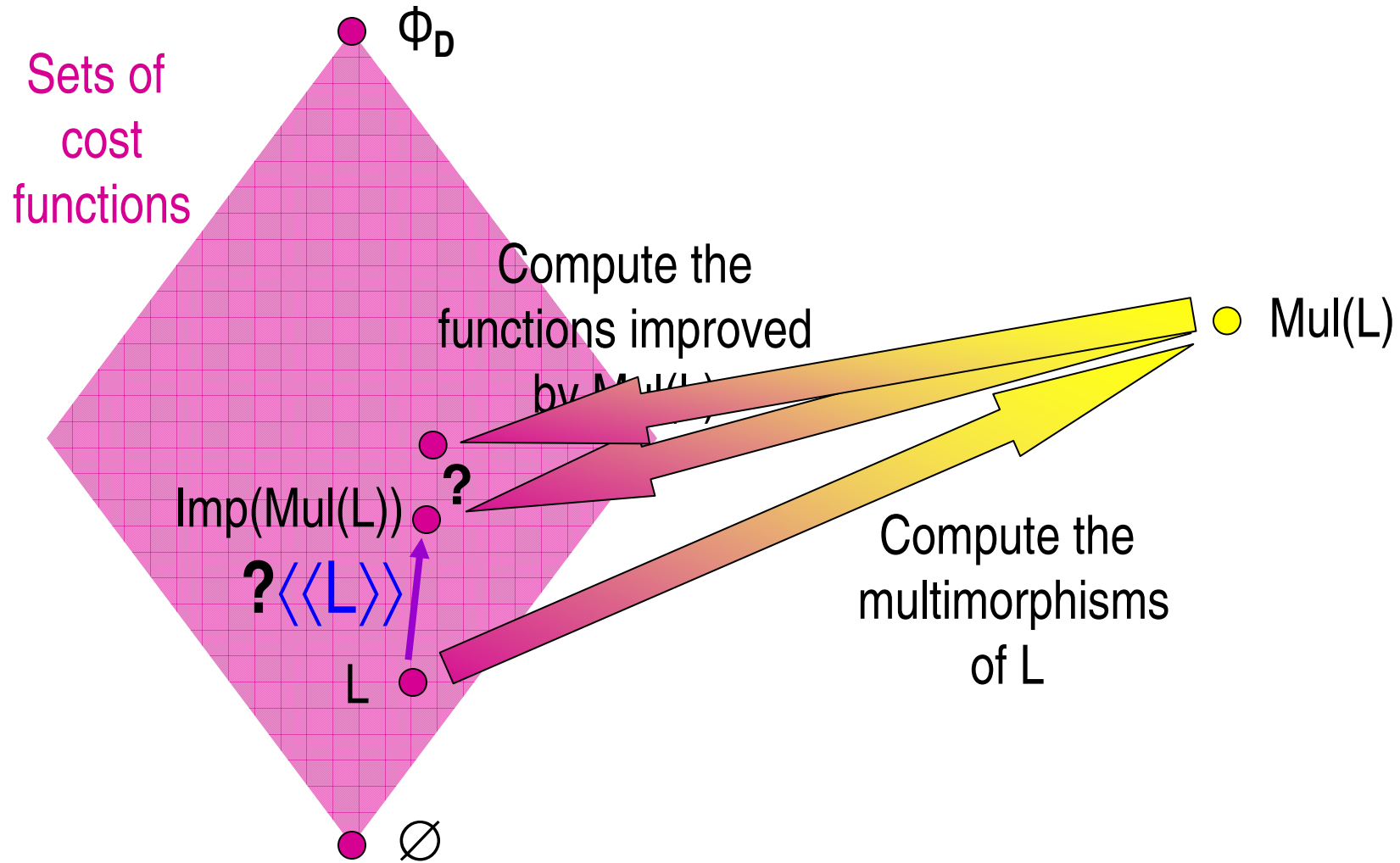
Min-Ones SAT



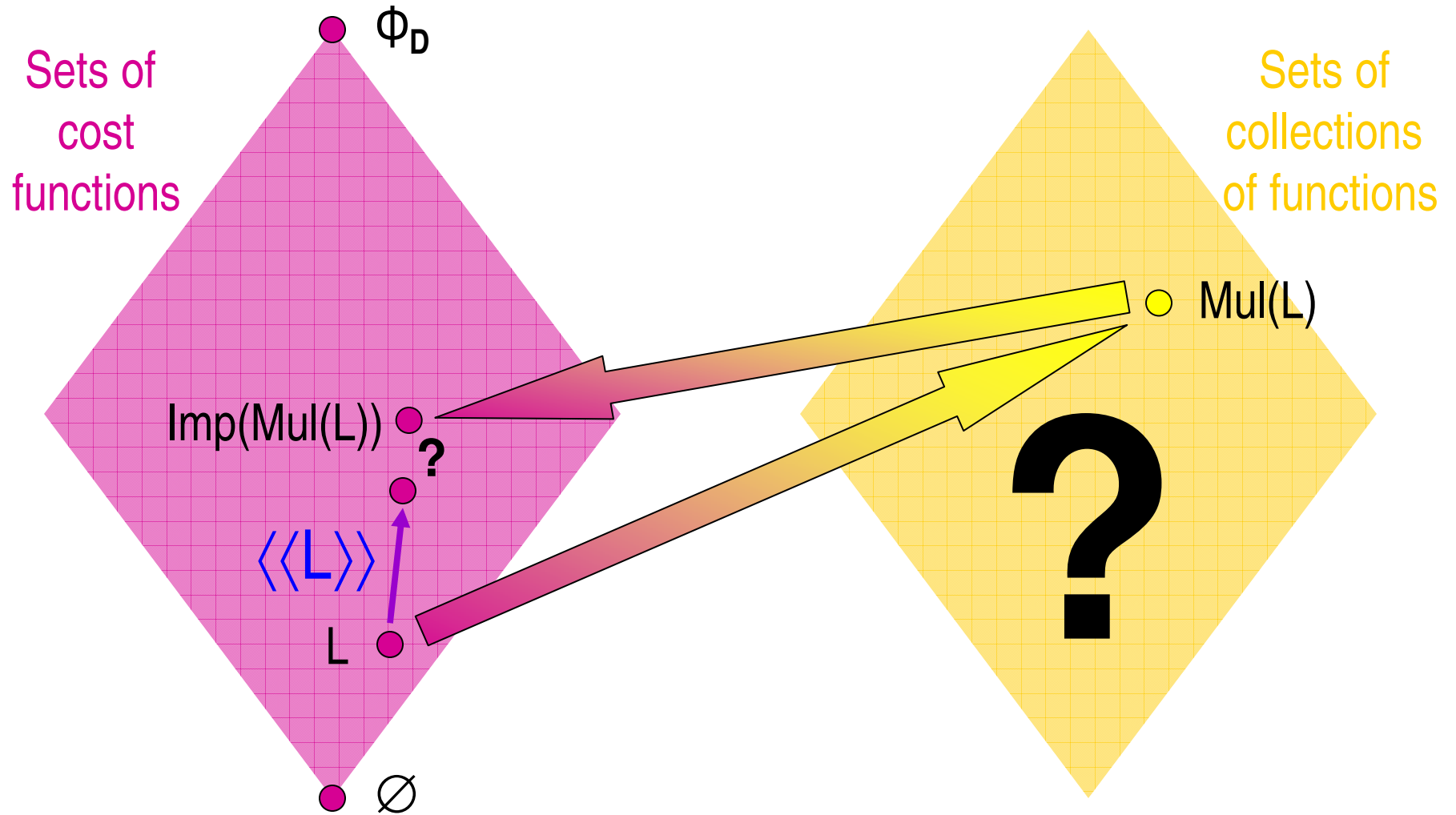
Pol and Inv



Mul and Imp



Mul and Imp



Generalization (again)

Generalizing Mul

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

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

0	0	1	1
1	0	0	5
0	0	0	0
1	0	1	3

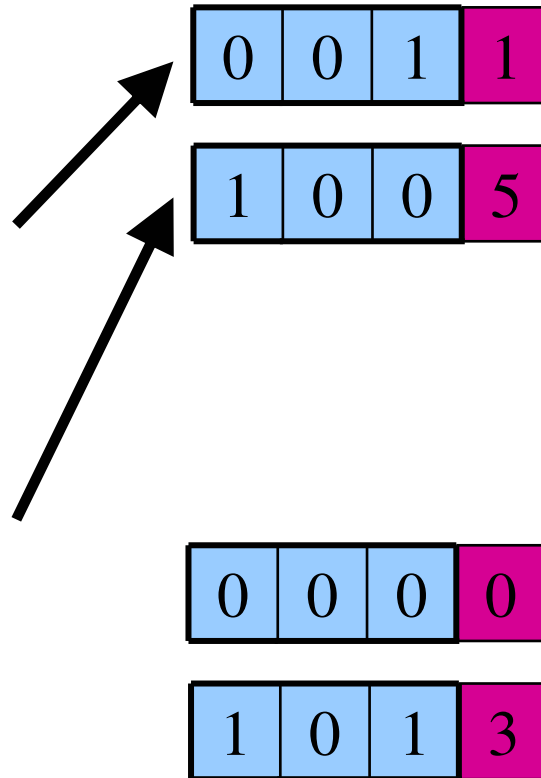
$$\text{Project}_1 + \text{Project}_2 = 6$$

\searrow

$$\text{Minimum} + \text{Maximum} = 3$$

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



Project₁
+
Project₂

$$= \begin{matrix} 0 \\ 6 \end{matrix}$$

Minimum
+
Maximum

$$= \begin{matrix} -3 \\ 3 \end{matrix}$$

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	Project₁
1	0	0	5	-1	Project₂
0	0	0	0	+1	Minimum
1	0	1	3	+1	Maximum

0

$\searrow \swarrow$

-3

Weighted Operations

Definition: A k-ary **weighted operation**, ω , is a (partial) *function* from 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	Project₁
----	----------------------------

-1	Project₂
----	----------------------------

+1	Minimum
----	----------------

+1	Maximum
----	----------------

Weighted Polymorphism

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

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

0	0	1	1	-1	Project₁
---	---	---	---	----	----------------------------

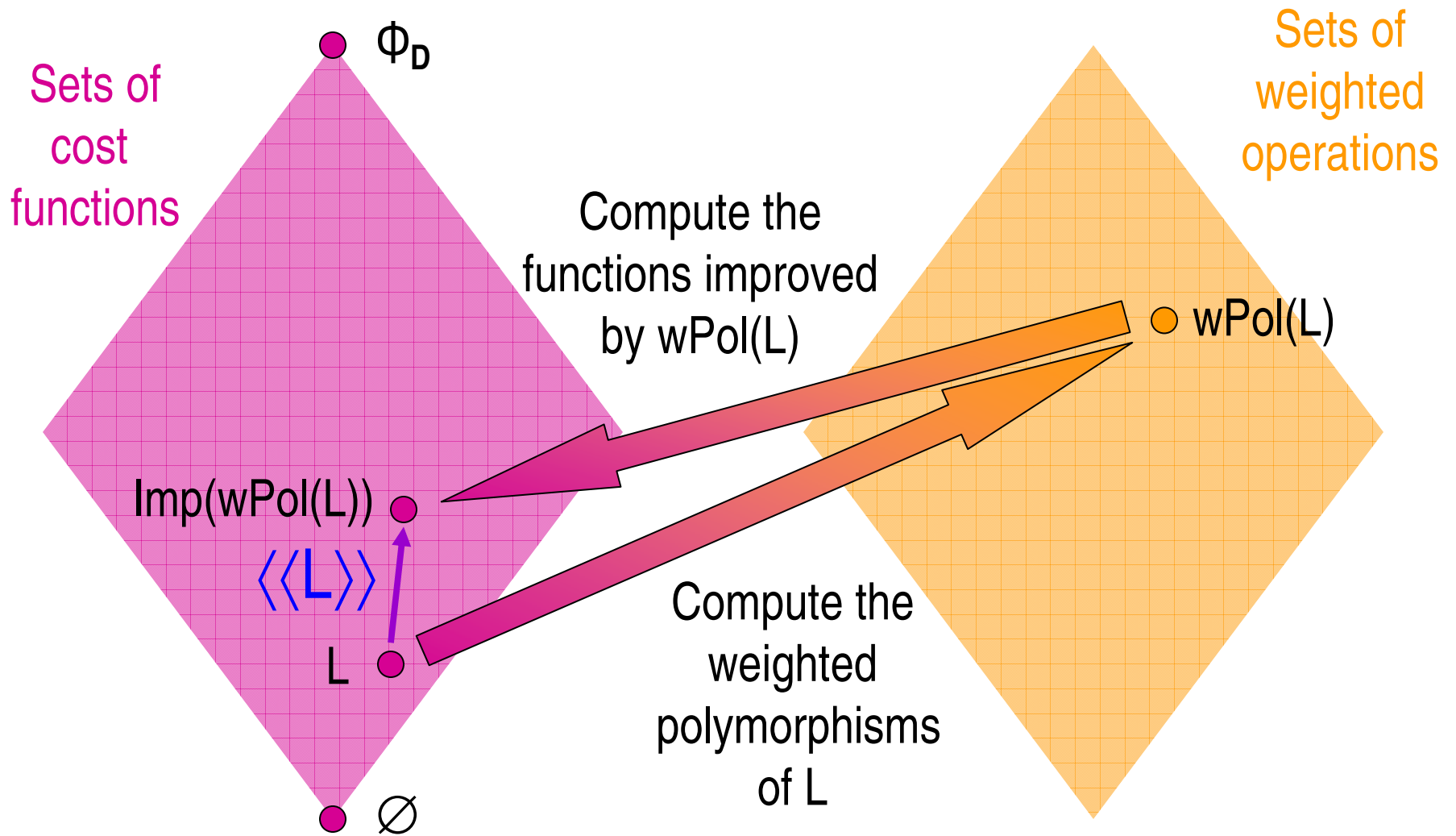
1	0	0	5	-1	Project₂
---	---	---	---	----	----------------------------

0	0	0	0	+1	Minimum
---	---	---	---	----	----------------

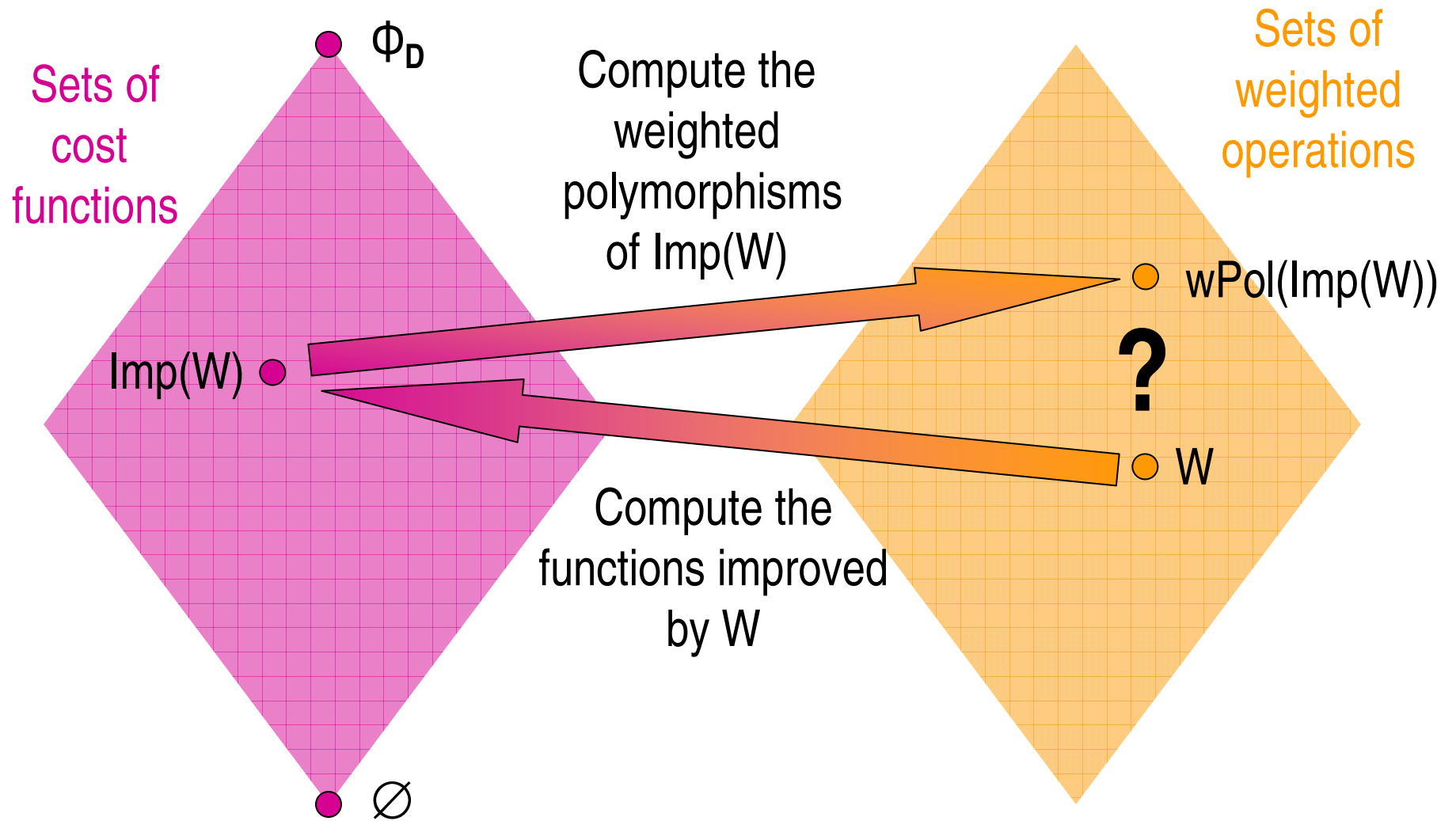
1	0	1	3	+1	Maximum
---	---	---	---	----	----------------

$$-3 \leq 0$$

wPol and Imp



wPol and Imp



Clones

Definition: Given a fixed set D , a **clone** on D is a set of operations that *contains all projections*, and is *closed under composition*.

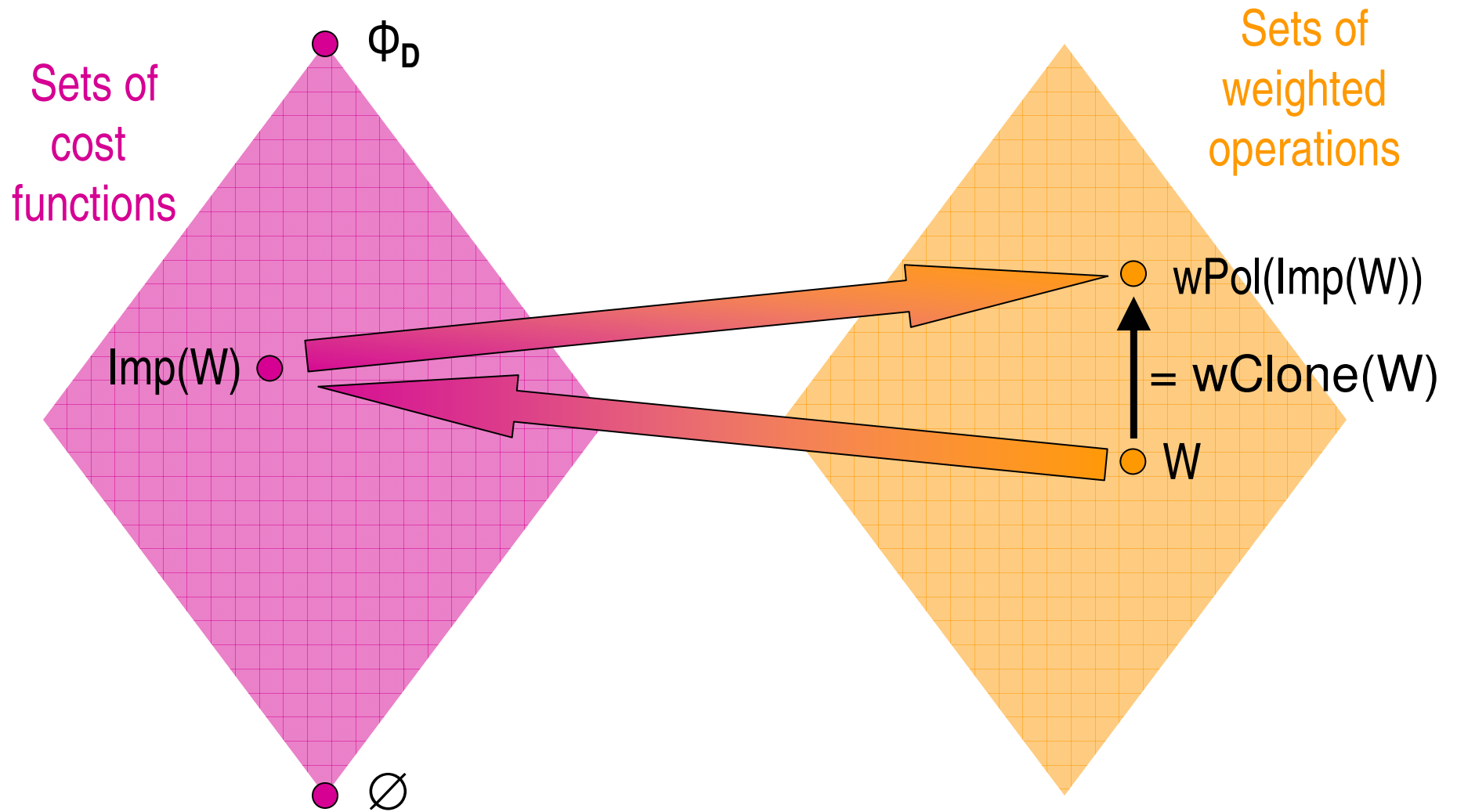
Weighted Clones

Definition: Given a fixed clone \mathcal{C} , a **weighted clone** on \mathcal{C} is a set of weighted operations that contains all weighted operations $\omega: \mathcal{C} \rightarrow \{0\}$, and is closed under:

1. Addition: $\omega_1 + \omega_2$
2. Scaling: $c \omega$ ($c \in \mathbb{Q}_+$)
3. Translation: $\omega[g_1, \dots, g_k]$

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

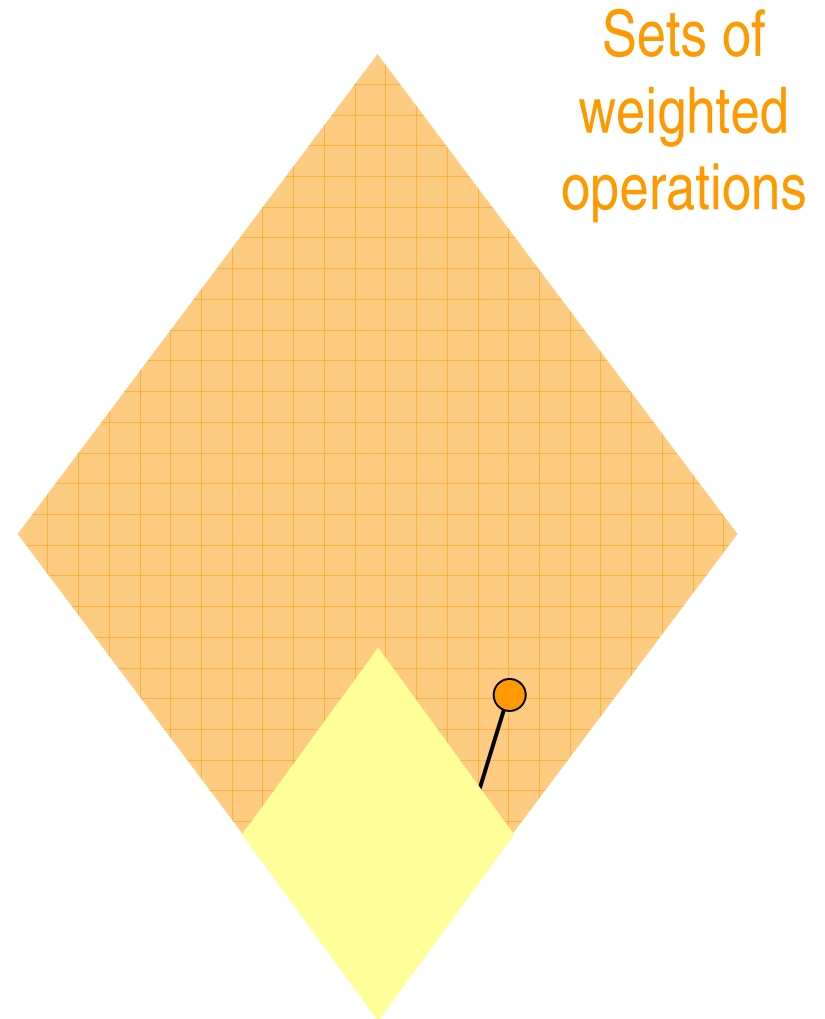
wPol and Imp



Minimal Weighted Clones

Definition: A weighted clone is *crisp* if every weighted operation in it has range $\{0\}$

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



Minimal Weighted Clones

Theorem: (Rosenberg)

Every minimal clone is generated by:

1. A unary retraction or cyclic permutation; or
2. A binary idempotent operation; or
3. A ternary minority operation; or
4. A ternary majority operation; or
5. A semiprojection.

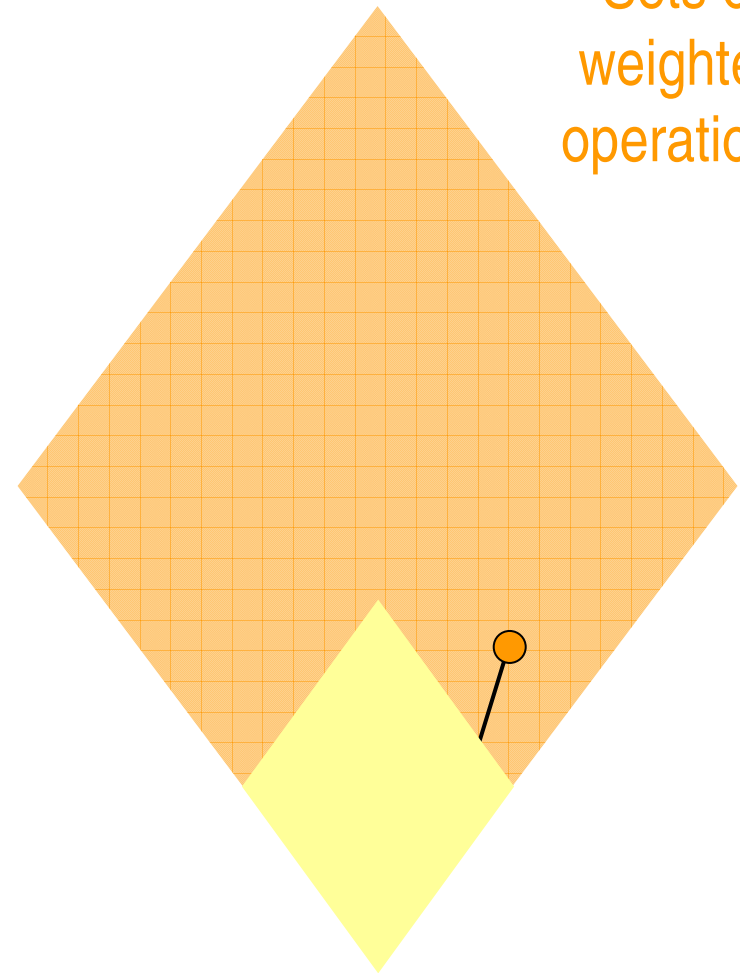
Minimal Weighted Clones

Theorem: (Creed & Živný, CP'11)

Every minimal weighted clone is generated by a weighted operation where the operations with positive weight are:

1. Unary; or
2. Binary idempotent; or
3. Ternary sharp; or
4. Semiprojections of arity > 3 .

Sets of
weighted
operations



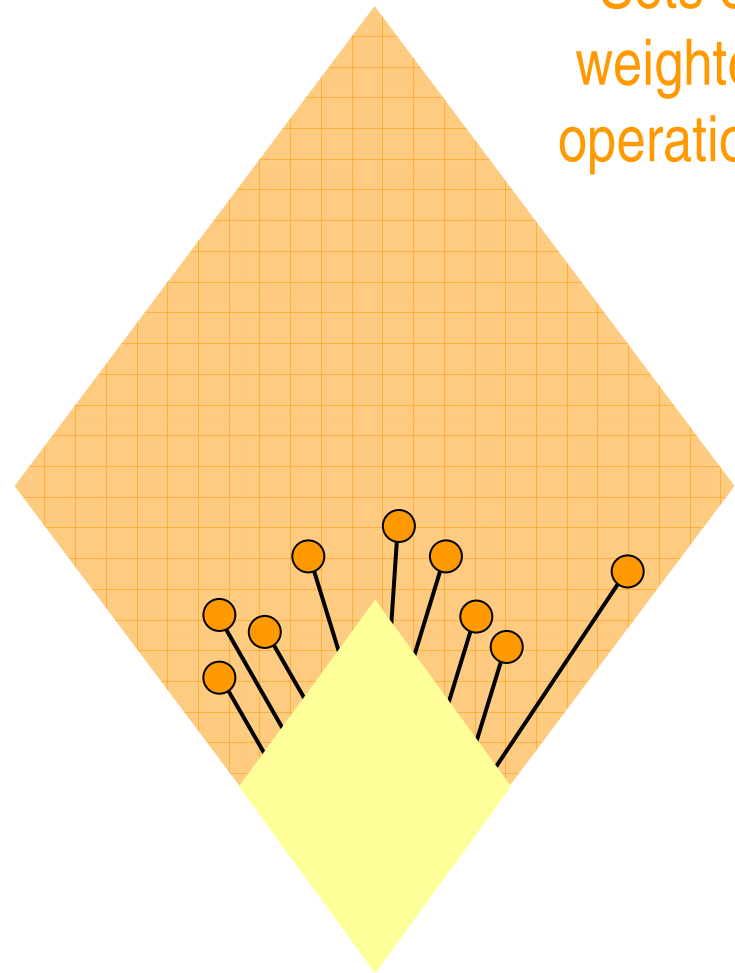
Minimal Weighted Clones

Theorem: (Creed & Živný, CP'11)

There are precisely 9 *Boolean* minimal weighted clones, generated by:

1. $\{ (-1, e_1), (1, \text{Const}_0) \}$
2. $\{ (-1, e_1), (1, \text{Const}_1) \}$
3. $\{ (-1, e_1), (1, 1-x) \}$
4. $\{ (-1, e_1), (-1, e_2), (2, \text{Min}) \}$
5. $\{ (-1, e_1), (-1, e_2), (2, \text{Max}) \}$
6. $\{ (-1, e_1), (-1, e_2), (-1, e_3), (3, \text{Mnrty}) \}$
7. $\{ (-1, e_1), (-1, e_2), (-1, e_3), (3, \text{Mjrty}) \}$
8. $\{ (-1, e_1), (-1, e_2), (1, \text{Min}), (1, \text{Max}) \}$
9. $\{ (-1, e_1), (-1, e_2), (-1, e_3), (1, \text{Mnrty}), (2, \text{Mjrty}) \}$

Sets of
weighted
operations



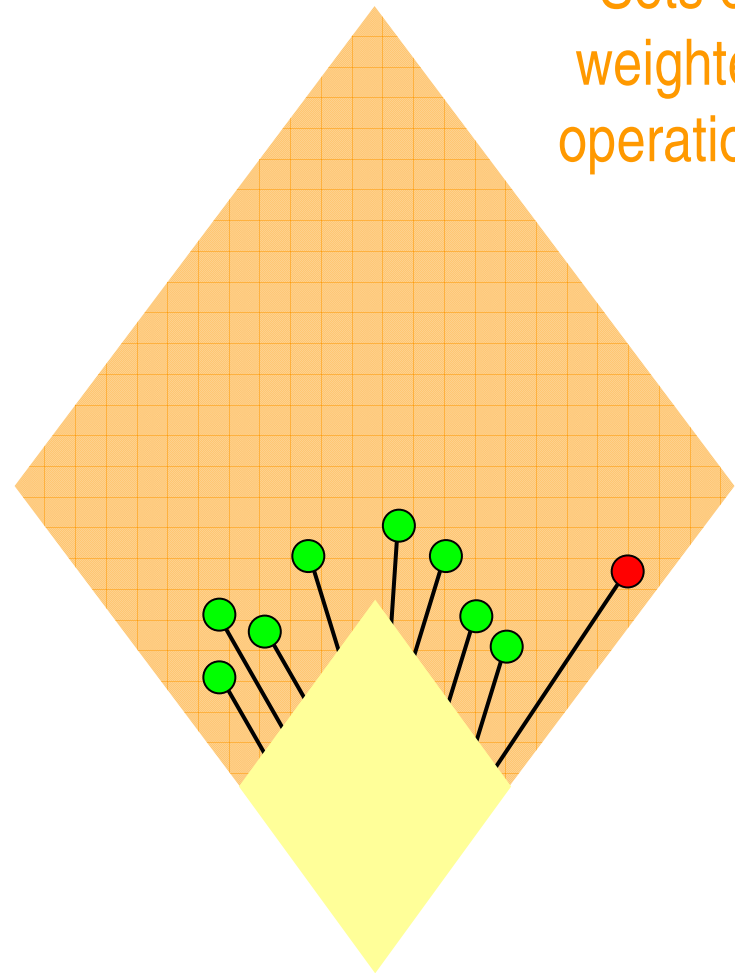
Minimal Weighted Clones

Theorem: (Creed & Živný, CP'11)

There are precisely 9 *Boolean* minimal weighted clones, generated by:

1. $\{ (-1, e_1), (1, \text{Const}_0) \}$
2. $\{ (-1, e_1), (1, \text{Const}_1) \}$
3. $\{ (-1, e_1), (1, 1-x) \}$
4. $\{ (-1, e_1), (-1, e_2), (2, \text{Min}) \}$
5. $\{ (-1, e_1), (-1, e_2), (2, \text{Max}) \}$
6. $\{ (-1, e_1), (-1, e_2), (-1, e_3), (3, \text{Mnrty}) \}$
7. $\{ (-1, e_1), (-1, e_2), (-1, e_3), (3, \text{Mjrty}) \}$
8. $\{ (-1, e_1), (-1, e_2), (1, \text{Min}), (1, \text{Max}) \}$
9. $\{ (-1, e_1), (-1, e_2), (-1, e_3), (1, \text{Mnrty}), (2, \text{Mjrty}) \}$

Sets of
weighted
operations



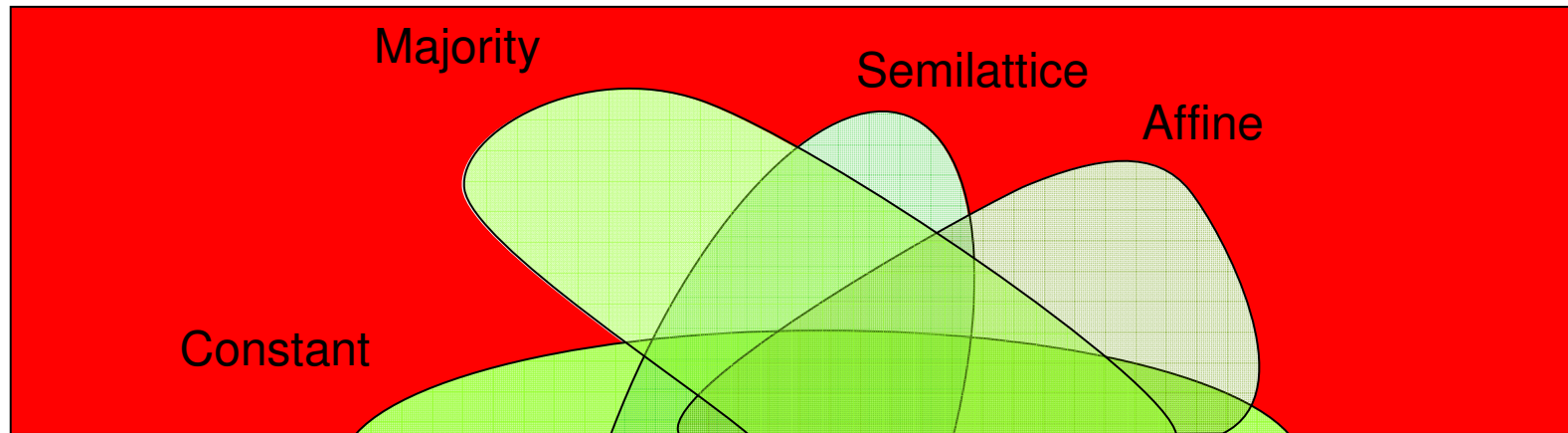
Open Problems

- What does the rest of the Boolean weighted clone lattice look like?
- What happens over larger domains?

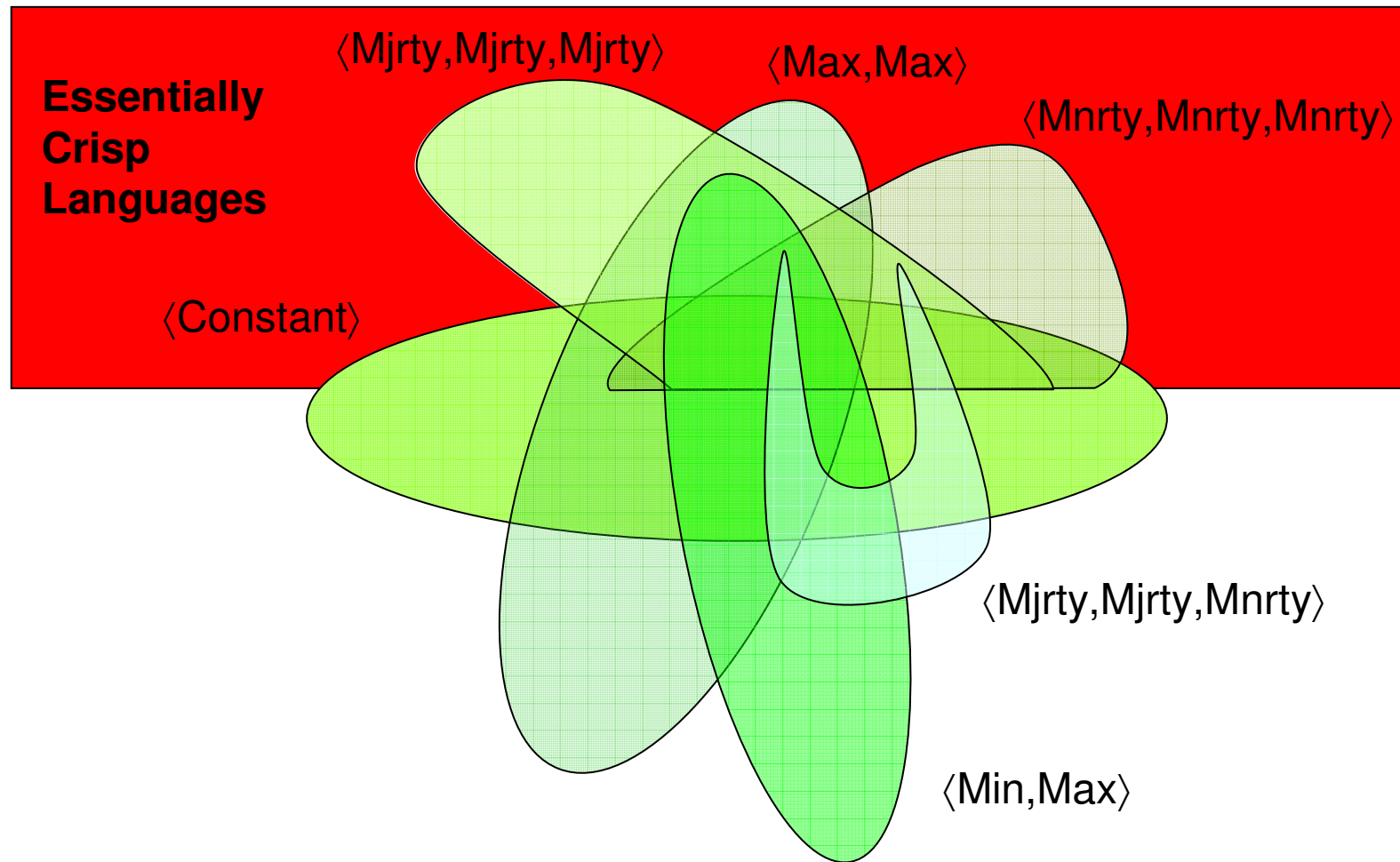


Thank you

Tractable cases



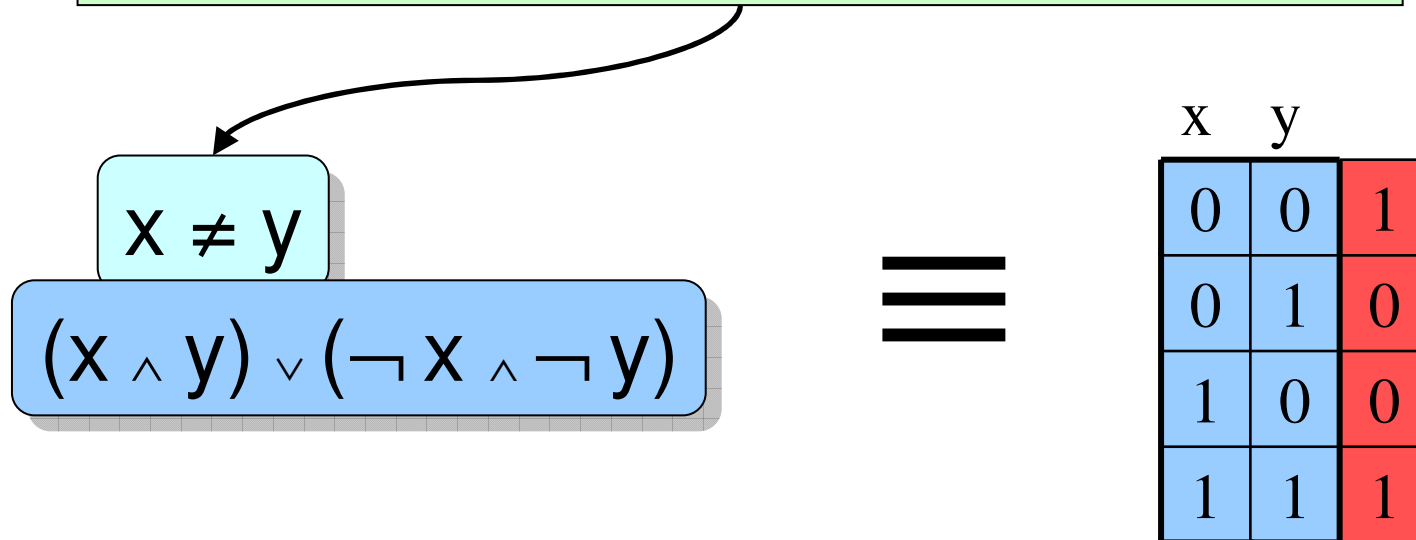
Tractable cases



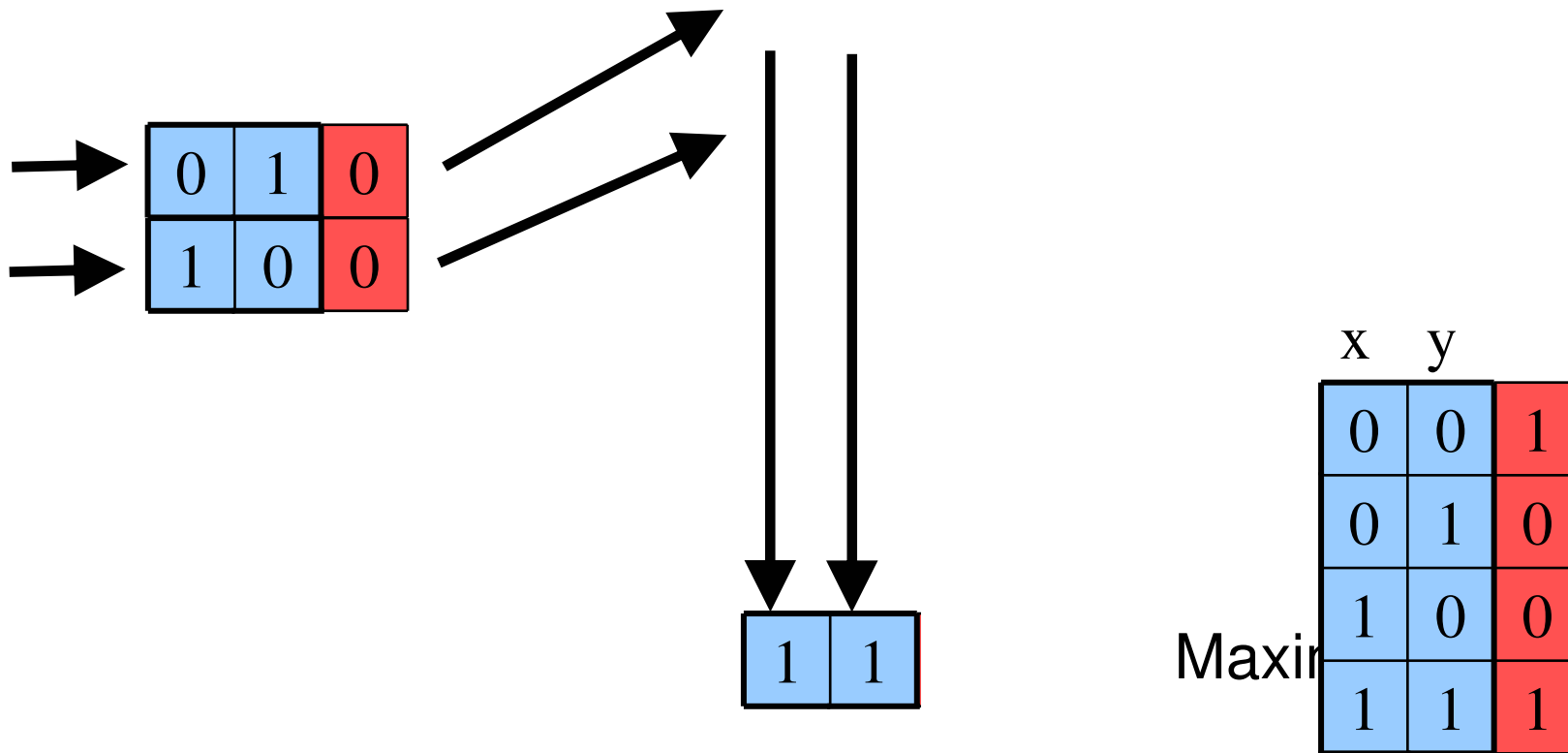
Intractable cases

MAX-SAT

This constraint is known to be NP-hard

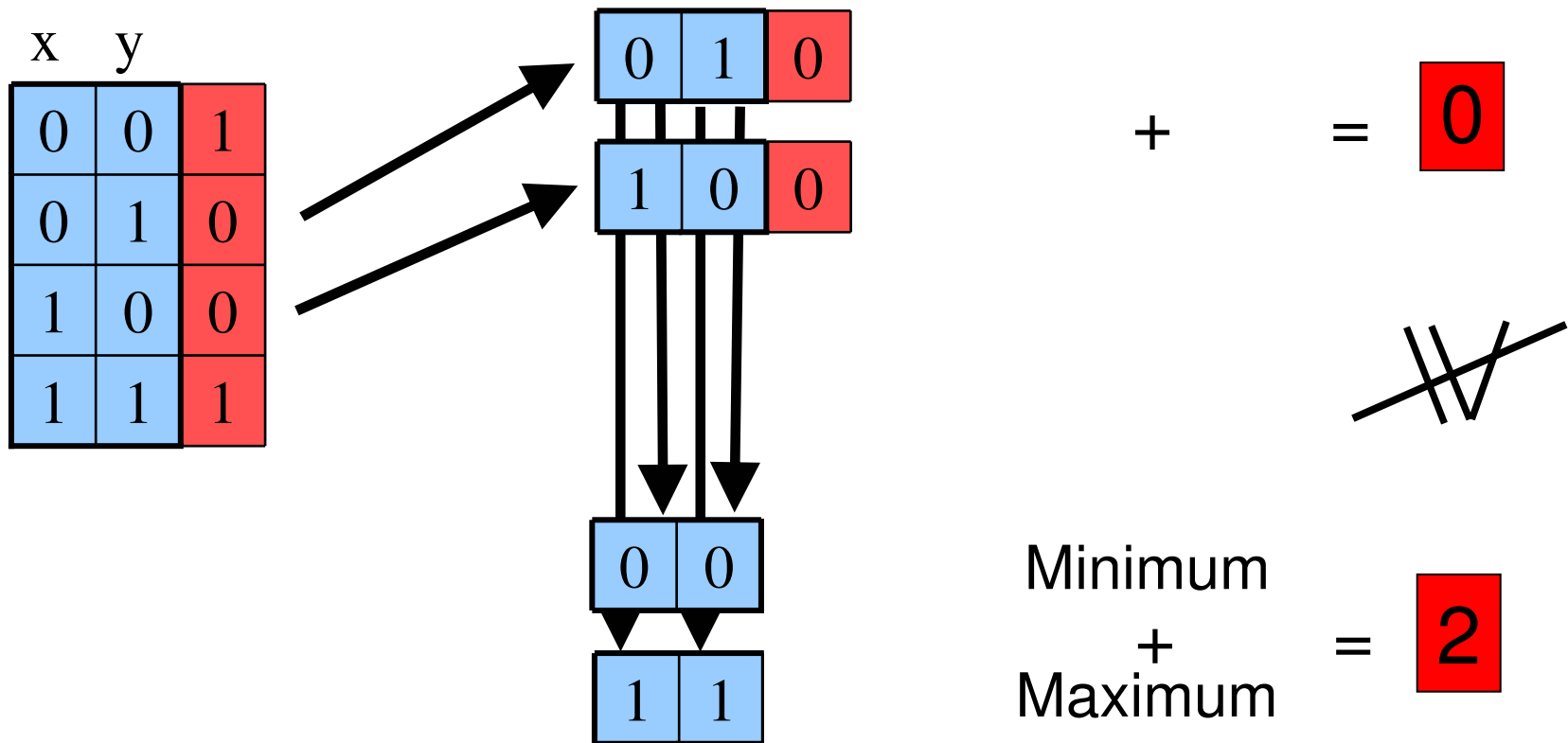


Intractable cases



Intractable cases

This cost function has **no** significant multimorphisms



Intractable cases

This cost function has **no** significant multimorphisms

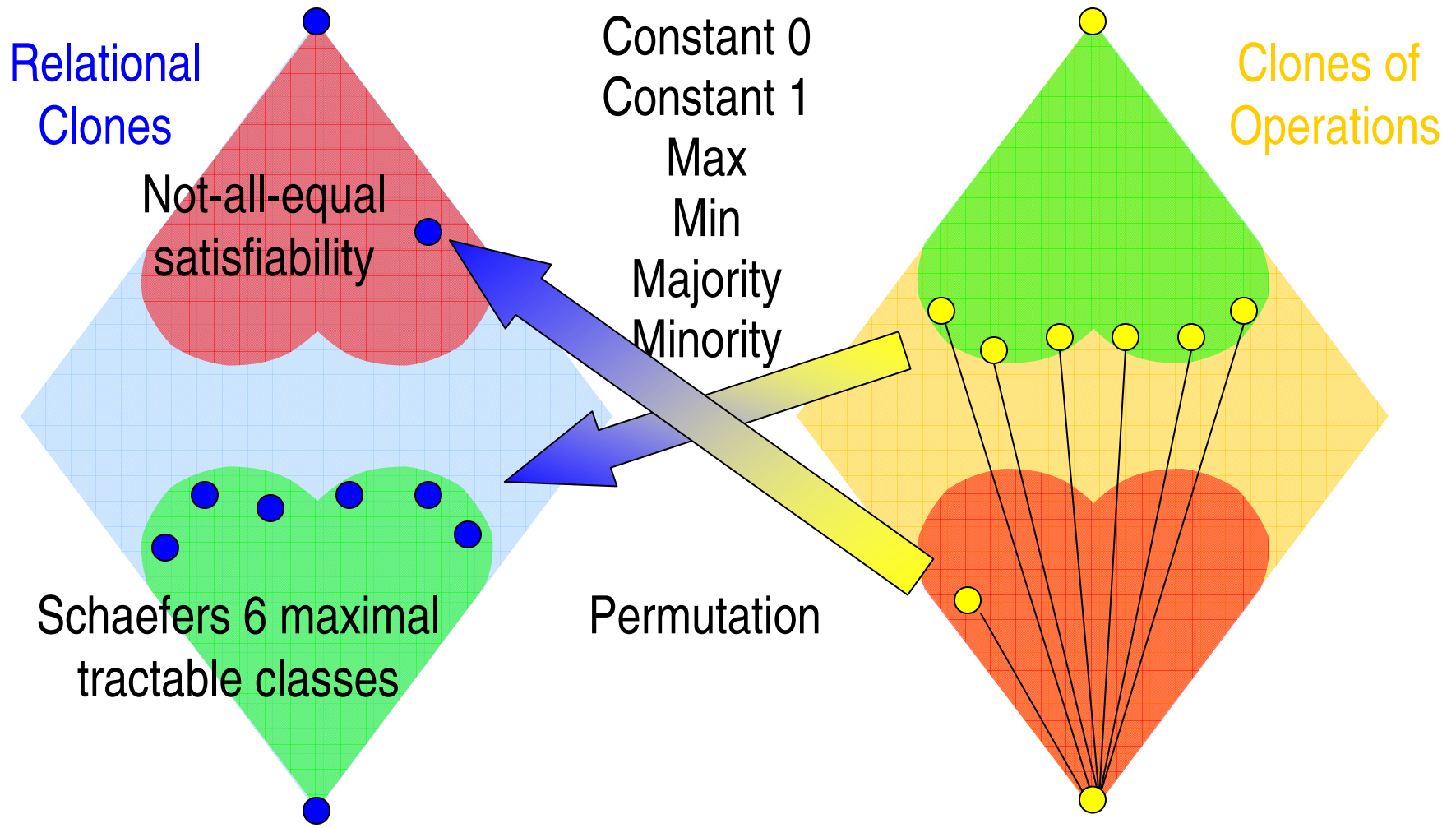
x	y	
0	0	b
0	1	a
1	0	a
1	1	b

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](#)

Boolean Operations



Boolean Operations

