The past 100 days...

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

- 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...
  
“Basic” Problems

- 3D-Matching
- 3-SAT
- Vertex Cover
- Clique
- Colouring
- Hamiltonian Circuit
- Partition

CSP
A General Framework

Variables $\bullet$ = 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 $\bigcirc = $ All invited talks on different days
No interference between near transmitters
$x + y + z > 0$
At least $1\mu m$ 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?
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, \( \text{CSP}(L) \)...
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

<table>
<thead>
<tr>
<th>$L$</th>
<th>CSP($L$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Disequality Relation</td>
<td>Graph Colouring Problem</td>
</tr>
<tr>
<td>${ \neq }$</td>
<td></td>
</tr>
<tr>
<td>Clauses</td>
<td>Satisfiability</td>
</tr>
<tr>
<td>Affine relations</td>
<td>Simultaneous Linear Equations</td>
</tr>
<tr>
<td>Temporal Relations</td>
<td>Simple Temporal Problems</td>
</tr>
<tr>
<td>${ (x,y) \mid x-y\leq t }$</td>
<td></td>
</tr>
</tbody>
</table>

NP-complete

Tractable
The Lattice of Languages

- NP-complete
- Disequality
- Affine relations
- Tractable
- $\emptyset$

$R_D$
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 $GF(2)$

Boolean Languages

- $R_{\{0,1\}}$
- NP-complete
- Not-all-equal SAT
- Tractable
- 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 $\langle L \rangle$, 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 $L' \subseteq \langle L \rangle$ then $\text{CSP}(L')$ is polynomial-time reducible to $\text{CSP}(L)$.
Expressive Power and Reduction

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

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

**Corollary**: If $\langle L_1 \rangle = \langle L_2 \rangle$ then $\text{CSP}(L_1)$ is polynomial-time equivalent to $\text{CSP}(L_2)$. 
Expressive Power and Reduction

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

$\langle \mathbf{L} \rangle$ is more important than $\mathbf{L}$
Calculating $\langle L \rangle$

- A relation is in $\langle L \rangle$ 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 $\phi$, if, for any tuples $a_1, a_2, \ldots, a_k \in R$, the tuple obtained by applying $\phi$ co-ordinatewise is a member of $R$.

If $R$ is invariant under $\phi$, then $\phi$ is called a *polymorphism* of $R$. 
We say that this relation $R$ has the polymorphism $\text{Maximum}$.

\[ \forall s, t \quad \text{if } s \text{ and } t \text{ are in } R, \text{ then } \text{Max}(s, t) \text{ is in } R \]
Sets of relations

Compute the polymorphisms of $L$

$\text{Inv}(\text{Pol}(L)) = \langle L \rangle$

Compute the invariant relations of $\text{Pol}(L)$

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

and independently by Bodnarchuk, Kaluzhnin, Kotov and Romov

Corollary: For any finite constraint language $L$, over a finite domain, the complexity of $\text{CSP}(L)$ is determined by $\text{Pol}(L)$
Sets of relations

Sets of operations

\[ \text{Inv}(\text{Pol}(L)) = \langle L \rangle \]

\[ \emptyset \]

\[ \text{Pol}(L) \]

\[ R_D \]

\[ L \]
Clones

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

Every relational clone is of the form \( \text{Inv}(\Phi) \) for some \( \Phi \).

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

Every clone is of the form \( \text{Pol}(L) \) for some \( L \).
Boolean Operations

Relational Clones
- Not-all-equal satisfiability
- Schaefers 6 maximal tractable classes

Clones of Operations
- Constant 0
- Constant 1
- Max
- Min
- Majority
- Minority

Permutation

Schaefers 6 maximal tractable classes

- Not
- All
- Equal
Boolean Operations

Relational Clones

Boolean Relational Clones

Dichotomy Theorem for Boolean CSP

Post's Lattice

Clones of 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 if and only if it has a polymorphism $f$ such that

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

Travelling Salesperson

Scheduling

Min-
Cut

Max-
SAT

Max-
Flow

Linear Programming

Max-
Cut

ILP

Max-
Clique

3D-
Matching

3-SAT

Colouring

Vertex Cover

Clique

Hamiltonian Circuit

Partition

CSP

x-Clique

3-SAT
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 \((V, D, C, \Omega)\), where
  – \(V\) is a set of variables
  – \(D\) is a single domain of possible values
  – \(C\) is a set of constraints
  – \(\Omega\) is a set of *costs*

Each constraint in \(C\) is a pair \((s, \phi)\) where
  • \(s\) is a list of variables defining the scope
  • \(\phi\) is a function from \(L\) defining the cost associated with each combination of values
Boolean constraints

\( x + y + z = 0 \)

\[
(x \lor y \lor \neg z) \land (x \lor \neg y \lor z) \land \\
(\neg x \lor y \lor z) \land (\neg x \lor \neg y \lor \neg z)
\]

SAT

\[
\begin{array}{ccc|c}
  x & y & z & \text{valued} \\
  \hline
  0 & 0 & 0 & \checkmark \\
  0 & 0 & 1 & \times \\
  0 & 1 & 0 & \times \\
  0 & 1 & 1 & \checkmark \\
  1 & 0 & 0 & \times \\
  1 & 0 & 1 & \checkmark \\
  1 & 1 & 0 & \checkmark \\
  1 & 1 & 1 & \times \\
\end{array}
\]
Valued Boolean constraints

SAT

$x + y + z = 0$

$(x \lor y \lor \neg z) \land (x \lor \neg y \lor z) \land (\neg x \lor y \lor z) \land (\neg x \lor \neg y \lor \neg z)$

<table>
<thead>
<tr>
<th>x</th>
<th>y</th>
<th>z</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>✓</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>X</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>✓</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>X</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>✓</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>✓</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>X</td>
</tr>
</tbody>
</table>
Valued Boolean constraints

\[ x + y + z = 0 \]

\[ (x \lor y \lor \neg z) \land (x \lor \neg y \lor z) \land (\neg x \lor y \lor z) \land (\neg x \lor \neg y \lor \neg z) \]

\[
\begin{array}{ccc|c}
0 & 0 & 0 & 0 \\
0 & 0 & 1 & 1 \\
0 & 1 & 0 & 1 \\
0 & 1 & 1 & 0 \\
1 & 0 & 0 & 1 \\
1 & 0 & 1 & 0 \\
1 & 1 & 0 & 0 \\
1 & 1 & 1 & 1 \\
\end{array}
\]
**Valued Boolean constraints**

**VSAT**

- Very general discrete optimization problem
- NP-hard
Valued constraints

VCSP

- Very general discrete optimization problem
- NP-hard
A Bigger Picture

VCSP

Travelling Salesperson
Linear Programming
Max-Clique
Max-SAT
Max-Cut
Scheduling
MinMax
Max-Flow
Min-Cut
3D-Matching
Hamiltonian Circuit
Partition
Colouring
Vertex Cover
Clique
3-SAT
Valued Constraint Languages
Valued Constraint Languages

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

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 $\varphi_0$, then we say that $\varphi_0$ can be expressed using $\varphi_1, \varphi_2, \ldots, \varphi_k$.
Expressive Power

Definition:

The “expressive power” of a valued constraint language $L$, denoted $\langle L \rangle$, 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 $\langle\langle L \rangle\rangle$, 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 $L, L'$, if $L'$ finite, and $L' \subseteq \langle \langle L \rangle \rangle$, then VCSP($L'$) is polynomial-time reducible to VCSP($L$).

**Corollary:** A valued constraint language $L$ is (locally) tractable if and only if $\langle \langle L \rangle \rangle$ is tractable; similarly, $L$ is NP-hard if and only if $\langle \langle L \rangle \rangle$ is NP-hard.
Algebra?
Pol and Inv

Sets of relations

Compute the polymorphisms of $L$

$\text{Inv}(\text{Pol}(L)) = \langle L \rangle$

Compute the invariant relations of Pol($L$)

$\varnothing$

Pol($L$)
We say that this relation $R$ has the polymorphism $\text{Maximum}$ if $s$ and $t$ are in $R$, then $\text{Max}(s, t)$ is in $R$. 

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

Generalizing Pol

∀s,t  Cost(Max(s,t)) ≤ Cost(s) + Cost(t)
Generalizing Pol

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

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>7</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>∞</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Minimum + Maximum = 3

\[ + = 6 \]
Generalizing Pol

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

We say that the cost function has the multimorphism \([\text{Min}, \text{Max}]\)

(any cost function with this particular multimorphism is called \textit{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

Constant  
Majority  
Semilattice  
Affine

Constant
Tractable cases of VCSP

Essentially Crisp Languages

\( \langle \text{Mjrty, Mjrty, Mjrty} \rangle \)

\( \langle \text{Max, Max} \rangle \)

\( \langle \text{Mnrt, Mnrt, Mnrt} \rangle \)

\( \langle \text{Constant} \rangle \)

\( \langle \text{Mjrty, Mjrty, Mnrt} \rangle \)

\( \langle \text{Min, Max} \rangle \)
The past 100 days...

• A new Galois connection to characterise the expressive power and complexity of valued constraint languages...

Sets of functions

\( \Phi_D \)

\( \text{Imp(Mul(L))} \)

\( \langle \langle L \rangle \rangle \)

\( L \)

\( \emptyset \)

\( \text{Mul(L)} \)

\( \text{Multimorphisms of L} \)

\( \text{Compute the functions improved by Mul(L)} \)

\( \text{Compute the multimorphisms of L} \)
Sets of sets of operations:

- $\Phi_D$
- $\langle\langle L \rangle\rangle$
- $\emptyset$

Sets of functions:

- $\text{Imp} (\text{Mul}(L))$

Mul and Imp

$\text{Mul}(L)$
Sets of weightings $w_{Pol}$ and $Imp$.

- Compute the weighted polymorphisms $w_{Pol}(L)$.
- Compute the functions improved by $w_{Pol}(L)$.

Sets of functions $\Phi_D$ and $\emptyset$.

- $\langle\langle L \rangle\rangle$.
- $L$.

Diagram:
- $w_{Pol}(L)$.
- $\Phi_D$.
- $\emptyset$.
- $L$.
- $\langle\langle L \rangle\rangle$.

Compute the weighted polymorphisms of $L$. 

Sets of weightings.
Generalizing Mul

∀s,t  Cost(Min(s,t)) + Cost(Max(s,t)) ≤ Cost(s) + Cost(t)

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>7</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>∞</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Project₁ + Project₂ = 6

Minimum + Maximum = 3
# Generalizing Mul

A table with columns labeled $x$, $y$, and $z$, and rows containing binary values. The table is used to illustrate different operations:

- **Project $P_1$**: The result is $6$.
- **Project $P_2$**: The result is $-3$.
- **Minimum**: The result is $3$.
- **Maximum**: The result is $5$.

The operations are performed on the rows of the table, with the results shaded in blue and the operations in red.
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 0 0 5
0 0 0 0
1 0 1 3

-1 Project₁
-1 Project₂
+1 Minimum
+1 Maximum
Weightings

Definition: A k-ary \textit{weighting}, $\omega$, is a (partial) \textit{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 \text{ \textbf{Project}_1}
-1 \text{ \textbf{Project}_2}
+1 \text{ \textbf{Minimum}}
+1 \text{ \textbf{Maximum}}
**Weighted Polymorphism**

**Definition:** A $k$-ary weighting, $\omega$, is a *weighted polymorphism* of a cost function $\phi$, if, for all $x_1, x_2, \ldots, x_k$, $\sum_f \omega(f) \phi(f(x_1, x_2, \ldots, x_k)) \leq 0$
Sets of weightings $\{w_{\text{Pol}} \}$ and $\{\text{Imp} \}$

Sets of functions $w_{\text{Pol}}(L)$ and $\text{Imp}(w_{\text{Pol}}(L))$

Compute the weighted polymorphisms of $L$

$\langle \langle L \rangle \rangle$

$\Phi_D$

Compute the functions improved by $w_{\text{Pol}}(L)$

$\emptyset$

$w_{\text{Pol}}(L)$
Sets of weightings \( w_{Pol} \) and \( \Phi \)

Compute the weighted polymorphisms of \( \text{Imp}(W) \)

Sets of functions

Compute the functions improved by \( W \)

\( \text{Imp}(W) \)

\( \Phi_D \)

\( \emptyset \)

\( \text{wPol} \left( \text{Imp}(W) \right) \)

\( W \)
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:** \(c\omega\) (\(c \in \mathbb{Q}^{\geq 0}\))
3. **Superposition:** \(\omega[g_1, \ldots g_k]\)

where \(\omega[g_1, \ldots g_k](f) = \sum \omega(f')\) \(\{ f' \mid f = f'[g_1, \ldots g_k] \}\)
Sets of weightings $w_{\text{Pol}}$ and $\text{Imp}$.

Sets of functions $w_{\text{Pol}}(\text{Imp}(W))$.

$\text{Imp}(W) = w_{\text{Clone}}(W)$.
Minimal Weighted Clones

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

**Definition:** A weighted clone is zero-valued if every weighting in it is zero-valued.
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 $\omega$ 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 $\omega(\text{Majority}) = 2\omega(\text{Minority})$. 
The past 100 days…

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


• 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*…

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…