

# TCT, fragments of Datalog and the fine-grained complexity of CSP

Benoit Larose <sup>1</sup> <sup>2</sup>

<sup>1</sup>Department of Mathematics and Statistics  
Concordia University, Montréal

<sup>2</sup>Department of Mathematics  
Champlain Regional College, St-Lambert

UA and CSP, Nashville June 2007

# Abstract

We investigate possible refinements of the algebraic dichotomy conjecture of Bulatov, Jeavons, Krokhin in an attempt to classify tractable CSP's within standard complexity classes via

- (i) tame congruence theory (TCT) and
- (ii) (non-)expressibility in fragments of Datalog.

The connection might also shed light on the decidability issue for various fragments of Datalog.

# The BJK algebraic dichotomy conjecture

- A finite relational core structure  $\Gamma$ ; what is the complexity of  $CSP(\Gamma)$  ?
- the algebra  $\mathbb{A} = \mathbb{A}(\Gamma)$ : its terms are all those *idempotent* operations that preserve the basic relations of  $\Gamma$ ;
- if the variety  $\mathcal{V}(\mathbb{A})$  generated by  $\mathbb{A}$  admits the unary type, then  $CSP(\Gamma)$  is **NP**-complete (BJK 2000)
- The “converse” is conjectured: if  $\mathcal{V}(\mathbb{A})$  omits the unary type, then  $CSP(\Gamma)$  is in **P**.

# The BJK algebraic dichotomy conjecture

- A finite relational core structure  $\Gamma$ ; what is the complexity of  $CSP(\Gamma)$  ?
- the algebra  $\mathbb{A} = \mathbb{A}(\Gamma)$ : its terms are all those *idempotent* operations that preserve the basic relations of  $\Gamma$ ;
- if the variety  $\mathcal{V}(\mathbb{A})$  generated by  $\mathbb{A}$  admits the unary type, then  $CSP(\Gamma)$  is **NP**-complete (BJK 2000)
- The “converse” is conjectured: if  $\mathcal{V}(\mathbb{A})$  omits the unary type, then  $CSP(\Gamma)$  is in **P**.

# The BJK algebraic dichotomy conjecture

- A finite relational core structure  $\Gamma$ ; what is the complexity of  $CSP(\Gamma)$  ?
- the algebra  $\mathbb{A} = \mathbb{A}(\Gamma)$ : its terms are all those *idempotent* operations that preserve the basic relations of  $\Gamma$ ;
- if the variety  $\mathcal{V}(\mathbb{A})$  generated by  $\mathbb{A}$  admits the unary type, then  $CSP(\Gamma)$  is **NP**-complete (BJK 2000)
- The “converse” is conjectured: if  $\mathcal{V}(\mathbb{A})$  omits the unary type, then  $CSP(\Gamma)$  is in **P**.

# The BJK algebraic dichotomy conjecture

- A finite relational core structure  $\Gamma$ ; what is the complexity of  $CSP(\Gamma)$  ?
- the algebra  $\mathbb{A} = \mathbb{A}(\Gamma)$ : its terms are all those *idempotent* operations that preserve the basic relations of  $\Gamma$ ;
- if the variety  $\mathcal{V}(\mathbb{A})$  generated by  $\mathbb{A}$  admits the unary type, then  $CSP(\Gamma)$  is **NP**-complete (BJK 2000)
- The “converse” is conjectured: if  $\mathcal{V}(\mathbb{A})$  omits the unary type, then  $CSP(\Gamma)$  is in **P**.

# The classification of Boolean CSP's

- In Allender, Bauland, Immerman, Schnoor, Vollmer (2005), a complete classification of the complexity of Boolean CSP's is obtained;
- all Boolean CSP's satisfy one of the following conditions:
  - in  $AC^0$ ;
  - L-complete;
  - NL-complete;
  - $\oplus$ L-complete;
  - P-complete;
  - NP-complete.
- the above are all standard complexity classes.

# The classification of Boolean CSP's

- In Allender, Bauland, Immerman, Schnoor, Vollmer (2005), a complete classification of the complexity of Boolean CSP's is obtained;
- all Boolean CSP's satisfy one of the following conditions:
  - in  $AC^0$ ;
  - **L**-complete;
  - **NL**-complete;
  - $\oplus$ **L**-complete;
  - **P**-complete;
  - **NP**-complete.
- the above are all standard complexity classes.





# The classification of Boolean CSP's, cont'd

- Remarkably, the classification of the complexity of Boolean CSP's lines up perfectly with
  - the typeset of the variety of the associated algebra
  - the (non-)expressibility of the CSP in various logics

## The classification of Boolean CSP's, cont'd

- Remarkably, the classification of the complexity of Boolean CSP's lines up perfectly with
  - the typeset of the variety of the associated algebra
  - the (non-)expressibility of the CSP in various logics

## The classification of Boolean CSP's, cont'd

- Remarkably, the classification of the complexity of Boolean CSP's lines up perfectly with
  - the typeset of the variety of the associated algebra
  - the (non-)expressibility of the CSP in various logics

# Outline of Part I: Complexity

- 2 Some Complexity Classes
- 3 Relationship between Classes

# Outline of Part II: Datalog and some of its fragments

- 4 Datalog
- 5 Linear Datalog
- 6 Symmetric Datalog
- 7 Non-expressibility Results

# Outline of Part III: Some Tame Congruence Theory

- 8 Types
- 9 Strictly Simple Algebras
- 10 Valeriote's Lemma
- 11 A second key lemma

# Outline of Part IV: Hardness and non-expressibility Results

- 12 A Reduction Lemma
- 13 Hardness and non-expressibility
- 14 Preliminary Evidence for Conjectures



# Part I

## Complexity

# Outline of Part I: Complexity

- We describe 5 important complexity classes
- for each class we describe a problem that somehow captures its essence (*complete* problems);
- we give a CSP form of each problem:  
some of these will be used as running examples.

# Outline of Part I: Complexity

- We describe 5 important complexity classes
- for each class we describe a problem that somehow captures its essence (*complete* problems);
- we give a CSP form of each problem:  
some of these will be used as running examples.

# Outline of Part I: Complexity

- We describe 5 important complexity classes
- for each class we describe a problem that somehow captures its essence (*complete* problems);
- we give a CSP form of each problem:  
some of these will be used as running examples.

# Reductions, hardness, completeness

## Reductions

All reductions are *first-order reductions*  
(unless otherwise specified)

- A problem  $P$  is *hard* for the complexity class  $\mathcal{C}$  if every problem in  $\mathcal{C}$  reduces to  $P$ ;
- the problem  $P$  is  $\mathcal{C}$ -*complete* if it is hard for  $\mathcal{C}$  and belongs to the class  $\mathcal{C}$ .

# Reductions, hardness, completeness

## Reductions

All reductions are *first-order reductions*  
(unless otherwise specified)

- A problem  $P$  is *hard* for the complexity class  $\mathcal{C}$  if every problem in  $\mathcal{C}$  reduces to  $P$ ;
- the problem  $P$  is  *$\mathcal{C}$ -complete* if it is hard for  $\mathcal{C}$  and belongs to the class  $\mathcal{C}$ .

# Reductions, hardness, completeness

## Reductions

All reductions are *first-order reductions*  
(unless otherwise specified)

- A problem  $P$  is *hard* for the complexity class  $\mathcal{C}$  if every problem in  $\mathcal{C}$  reduces to  $P$ ;
- the problem  $P$  is  *$\mathcal{C}$ -complete* if it is hard for  $\mathcal{C}$  and belongs to the class  $\mathcal{C}$ .

# The class **NP**

- **NP** is the class of problems recognised by a polynomial time bounded non-deterministic Turing machine
- equivalently: **NP** is the class of polynomially verifiable problems
- for any structure  $\Gamma$  the problem  $CSP(\Gamma)$  is in **NP**:  
given a solution to  $CSP(\Gamma)$ , one may verify it in polynomial time.



# The class **NP**

- **NP** is the class of problems recognised by a polynomial time bounded non-deterministic Turing machine
- equivalently: **NP** is the class of polynomially verifiable problems
- for any structure  $\Gamma$  the problem  $CSP(\Gamma)$  is in **NP**:  
given a solution to  $CSP(\Gamma)$ , one may verify it in polynomial time.

# The class **NP**

- **NP** is the class of problems recognised by a polynomial time bounded non-deterministic Turing machine
- equivalently: **NP** is the class of polynomially verifiable problems
- for any structure  $\Gamma$  the problem  $CSP(\Gamma)$  is in **NP**:  
given a solution to  $CSP(\Gamma)$ , one may verify it in polynomial time.

# A complete problem for NP

## (positive) NOT ALL EQUAL 3-SAT

- Input: Sets  $S_1, \dots, S_m$  with at most three elements;
- Question: can one colour the elements so that no set gets only one colour ?

# A complete problem for **NP**, CSP form

## CSP form of positive NOT ALL EQUAL 3-SAT

$CSP(\Gamma)$ , where  $\Gamma$  is the structure  $\Gamma = \langle \{0, 1\}; \theta \rangle$  where

$$\theta = \{0, 1\}^3 \setminus \{(0, 0, 0), (1, 1, 1)\}.$$

# The class **P**

- **P** is the class of problems recognised by a polynomial time bounded (deterministic) Turing machine

# A complete problem for $P$

## HORN-3-SAT

- Input: A conjunction of Horn 3-clauses
- Question: is there a satisfying assignment ?

# A complete problem for $\mathbf{P}$ , CSP form

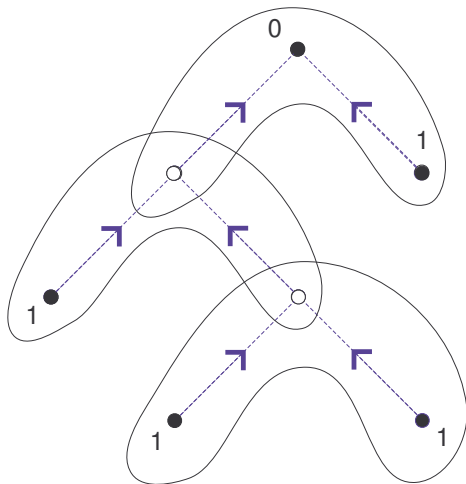
## CSP form of HORN-3-SAT

CSP( $\Gamma$ ), where  $\Gamma$  is the structure  $\Gamma = \langle \{0, 1\}; \{0\}, \{1\}, \rho \rangle$  where

$$\begin{aligned}\rho &= \{(x, y, z) : (y \wedge z) \rightarrow x\} \\ &= \{0, 1\}^3 \setminus \{(0, 1, 1)\}\end{aligned}$$

# A complete problem for $\mathbf{P}$ , cont'd

An unsatisfiable instance:





# The class **NL**

- **NL** is the class of problems recognised by a logarithmic space bounded non-deterministic Turing machine

# A complete problem for **NL**

## Directed Reachability

- Input: a directed graph and two specified nodes  $s$  and  $t$ ;
- Question: is there a directed path from  $s$  to  $t$  ?

# A complete problem for **NL**, CSP form

## CSP form of *Directed Reachability*

CSP( $\Gamma$ ), where  $\Gamma$  is the structure  $\Gamma = \langle \{0, 1\}; \{0\}, \{1\}, \leq \rangle$

- Note: this is actually *Unreachability*, but **NL** is closed under complementation (Immerman 1988; Szelepcsényi 1987)

# A complete problem for **NL**, CSP form

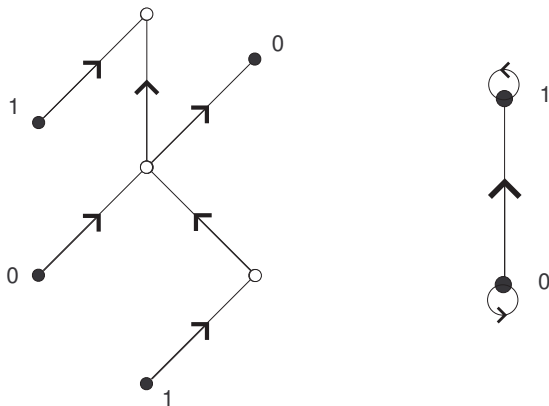
## CSP form of *Directed Reachability*

$CSP(\Gamma)$ , where  $\Gamma$  is the structure  $\Gamma = \langle \{0, 1\}; \{0\}, \{1\}, \leq \rangle$

- Note: this is actually *Unreachability*, but **NL** is closed under complementation (Immerman 1988; Szelepcsényi 1987)

# A complete problem for NL, cont'd

An unsatisfiable instance (and target): there exists a directed path from a node coloured 1 to a node coloured 0.



# The class L

- **L** is the class of problems recognised by a logarithmic space bounded (deterministic) Turing machine

# A complete problem for $L$

## Undirected Reachability

- Input: an undirected graph and specified nodes  $s$  and  $t$ ;
  - Question: is there a path from  $s$  to  $t$  ?
- The fact that this problem is in  $L$  follows from a deep result of Reingold (2005)

# A complete problem for $\mathbf{L}$

## Undirected Reachability

- Input: an undirected graph and specified nodes  $s$  and  $t$ ;
  - Question: is there a path from  $s$  to  $t$  ?
- 
- The fact that this problem is in  $\mathbf{L}$  follows from a deep result of Reingold (2005)



# A complete problem for L, CSP form

## CSP form of *Undirected Reachability*

CSP( $\Gamma$ ), where  $\Gamma$  is the structure  $\Gamma = \langle \{0, 1\}; \{0\}, \{1\}, = \rangle$

- Note: the CSP actually encodes *Unreachability*.

# A complete problem for L, CSP form

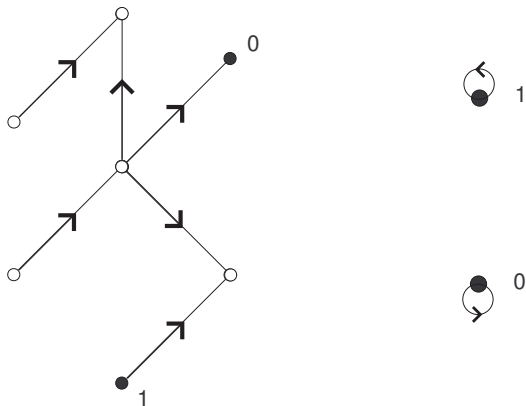
## CSP form of *Undirected Reachability*

CSP( $\Gamma$ ), where  $\Gamma$  is the structure  $\Gamma = \langle \{0, 1\}; \{0\}, \{1\}, = \rangle$

- Note: the CSP actually encodes *Unreachability*.

## A complete problem for L, cont'd

An unsatisfiable instance (and target): there exists an undirected path from a node coloured 1 to a node coloured 0.



# The classes $\text{mod}_p\mathbf{L}$

Let  $p \geq 2$  be a prime.

- A language  $L$  is in  $\text{mod}_p\mathbf{L}$  if there exists a logarithmic space-bounded non-deterministic Turing machine  $M$  such that  $w \in L$  precisely if the number of accepting paths on input  $w$  is  $0 \pmod p$ .
- If  $p = 2$ ,  $\text{mod}_2\mathbf{L}$  is denoted  $\oplus\mathbf{L}$  and is called *parity L*.

# The classes $\text{mod}_p\mathbf{L}$

Let  $p \geq 2$  be a prime.

- A language  $L$  is in  $\text{mod}_p\mathbf{L}$  if there exists a logarithmic space-bounded non-deterministic Turing machine  $M$  such that  $w \in L$  precisely if the number of accepting paths on input  $w$  is  $0 \pmod p$ .
- If  $p = 2$ ,  $\text{mod}_2\mathbf{L}$  is denoted  $\oplus\mathbf{L}$  and is called *parity L*.

# A complete problem for $\text{mod}_p\text{L}$

## Linear equations $\text{mod } p$

- Input: a system of linear equations  $\text{mod } p$ ;
- Question: is there a solution ?

## Some hard problems for $\text{mod}_p\mathbf{L}$ , CSP form

- let  $\mathbb{A} = \langle A; +, 0 \rangle$  be a finite Abelian group and let  $b$  be any non-zero element of  $\mathbb{A}$  such that  $pb = 0$  for some prime  $p$ ;
- the following problem is  $\text{mod}_p\mathbf{L}$ -hard  
(BL, Tesson, 2007)

$\text{mod}_p\mathbf{L}$ -complete CSP form

$\text{CSP}(\Gamma)$ , where  $\Gamma$  is the structure  $\langle A; \mu, \{b\}, \{0\} \rangle$  with

$$\mu = \{(x, y, z) : x + y = z\}.$$

## Some hard problems for $\text{mod}_p\mathbf{L}$ , CSP form

- let  $\mathbb{A} = \langle A; +, 0 \rangle$  be a finite Abelian group and let  $b$  be any non-zero element of  $\mathbb{A}$  such that  $pb = 0$  for some prime  $p$ ;
- the following problem is  $\text{mod}_p\mathbf{L}$ -hard (BL, Tesson, 2007)

$\text{mod}_p\mathbf{L}$ -complete CSP form

$\text{CSP}(\Gamma)$ , where  $\Gamma$  is the structure  $\langle A; \mu, \{b\}, \{0\} \rangle$  with

$$\mu = \{(x, y, z) : x + y = z\}.$$



## Some hard problems for $\text{mod}_p\mathbf{L}$ , CSP form

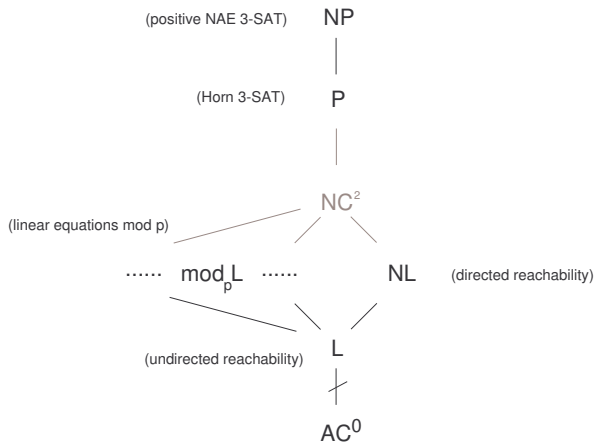
- let  $\mathbb{A} = \langle A; +, 0 \rangle$  be a finite Abelian group and let  $b$  be any non-zero element of  $\mathbb{A}$  such that  $pb = 0$  for some prime  $p$ ;
- the following problem is  $\text{mod}_p\mathbf{L}$ -hard (BL, Tesson, 2007)

### $\text{mod}_p\mathbf{L}$ -complete CSP form

$\text{CSP}(\Gamma)$ , where  $\Gamma$  is the structure  $\langle A; \mu, \{b\}, \{0\} \rangle$  with

$$\mu = \{(x, y, z) : x + y = z\}.$$

# Containments of these complexity classes



## Part II

# Datalog and some Fragments

## Outline of Part II: Datalog and some Fragments

- We define the notion of *Datalog Program*, a means of describing certain sets of structures;
- we define 2 *fragments* of Datalog, i.e. special restricted versions;
- we describe, for each fragment, a CSP which is definable in it;
- each of these CSP's somehow captures the essence of each fragment

## Outline of Part II: Datalog and some Fragments

- We define the notion of *Datalog Program*, a means of describing certain sets of structures;
- we define 2 *fragments* of Datalog, i.e. special restricted versions;
- we describe, for each fragment, a CSP which is definable in it;
- each of these CSP's somehow captures the essence of each fragment

## Outline of Part II: Datalog and some Fragments

- We define the notion of *Datalog Program*, a means of describing certain sets of structures;
- we define 2 *fragments* of Datalog, i.e. special restricted versions;
- we describe, for each fragment, a CSP which is definable in it;
- each of these CSP's somehow captures the essence of each fragment

## Outline of Part II: Datalog and some Fragments

- We define the notion of *Datalog Program*, a means of describing certain sets of structures;
- we define 2 *fragments* of Datalog, i.e. special restricted versions;
- we describe, for each fragment, a CSP which is definable in it;
- each of these CSP's somehow captures the essence of each fragment

# Datalog

- A *Datalog Program* consists of *rules*, and takes as input a relational structure.
- a typical Datalog rule might look like this one:

$$I(x, y) \leftarrow J(w, u, x), K(x), \theta_1(x, y, z), \theta_2(x, w)$$

- the relations  $\theta_1$  and  $\theta_2$  are basic relations of the input structures (EDB's);
- the relations  $I, J, K$  are auxiliary relations used by the program (IDB's);
- the rule stipulates that if the condition on the righthand side (the *body* of the rule) holds, then the condition of the left (the *head*) should also hold.



# Datalog

- A *Datalog Program* consists of *rules*, and takes as input a relational structure.
- a typical Datalog rule might look like this one:

$$I(x, y) \leftarrow J(w, u, x), K(x), \theta_1(x, y, z), \theta_2(x, w)$$

- the relations  $\theta_1$  and  $\theta_2$  are basic relations of the input structures (EDB's);
- the relations  $I, J, K$  are auxiliary relations used by the program (IDB's);
- the rule stipulates that if the condition on the righthand side (the *body* of the rule) holds, then the condition of the left (the *head*) should also hold.

# Datalog

- A *Datalog Program* consists of *rules*, and takes as input a relational structure.
- a typical Datalog rule might look like this one:

$$I(x, y) \leftarrow J(w, u, x), K(x), \theta_1(x, y, z), \theta_2(x, w)$$

- the relations  $\theta_1$  and  $\theta_2$  are basic relations of the input structures (EDB's);
- the relations  $I, J, K$  are auxiliary relations used by the program (IDB's);
- the rule stipulates that if the condition on the righthand side (the *body* of the rule) holds, then the condition of the left (the *head*) should also hold.

# Datalog

- A *Datalog Program* consists of *rules*, and takes as input a relational structure.
- a typical Datalog rule might look like this one:

$$I(x, y) \leftarrow J(w, u, x), K(x), \theta_1(x, y, z), \theta_2(x, w)$$

- the relations  $\theta_1$  and  $\theta_2$  are basic relations of the input structures (EDB's);
- the relations  $I, J, K$  are auxiliary relations used by the program (IDB's);
- the rule stipulates that if the condition on the righthand side (the *body* of the rule) holds, then the condition of the left (the *head*) should also hold.

# Datalog

- A *Datalog Program* consists of *rules*, and takes as input a relational structure.
- a typical Datalog rule might look like this one:

$$I(x, y) \leftarrow J(w, u, x), K(x), \theta_1(x, y, z), \theta_2(x, w)$$

- the relations  $\theta_1$  and  $\theta_2$  are basic relations of the input structures (EDB's);
- the relations  $I, J, K$  are auxiliary relations used by the program (IDB's);
- the rule stipulates that if the condition on the righthand side (the *body* of the rule) holds, then the condition of the left (the *head*) should also hold.

# An example

Recall:

## HORN-3-SAT

$CSP(\Gamma)$  where  $\Gamma = \langle \{0, 1\}; \{0\}, \{1\}, \rho \rangle$  with

$$\rho = \{(x, y, z) : (y \wedge z) \rightarrow x\}$$

- Here is a *Datalog program* that accepts precisely those structures that are NOT in  $CSP(\Gamma)$ , i.e. that do not admit a homomorphism to  $\Gamma$ :

# An example

Recall:

## HORN-3-SAT

$CSP(\Gamma)$  where  $\Gamma = \langle \{0, 1\}; \{0\}, \{1\}, \rho \rangle$  with

$$\rho = \{(x, y, z) : (y \wedge z) \rightarrow x\}$$

- Here is a *Datalog program* that accepts precisely those structures that are NOT in  $CSP(\Gamma)$ , i.e. that do not admit a homomorphism to  $\Gamma$ :

# A Datalog program for HORN-3-SAT

## A Datalog program

$$W(x) \leftarrow 1(x)$$

$$W(x) \leftarrow W(y), W(z), \rho(x, y, z)$$

$$G \leftarrow W(x), 0(x)$$

- the 0-ary relation  $G$  is the *goal predicate* of the program: it "lights up" precisely if the input structure admits NO homomorphism to the target structure  $\Gamma$ .

### Definition (Definability in Datalog)

We say that  $\neg\text{CSP}(\Gamma)$  is *definable in Datalog* if there exists a Datalog program that accepts precisely those structures that do not admit a homomorphism to  $\Gamma$ .

### Theorem

*If  $\neg\text{CSP}(\Gamma)$  is definable in Datalog then  $\text{CSP}(\Gamma)$  is in **P**.*

- Idea: IDB's have bounded arity, so the program can do only polynomially many steps before stabilising



### Definition (Definability in Datalog)

We say that  $\neg\text{CSP}(\Gamma)$  is *definable in Datalog* if there exists a Datalog program that accepts precisely those structures that do not admit a homomorphism to  $\Gamma$ .

### Theorem

If  $\neg\text{CSP}(\Gamma)$  is definable in Datalog then  $\text{CSP}(\Gamma)$  is in **P**.

- Idea: IDB's have bounded arity, so the program can do only polynomially many steps before stabilising

# A first fragment: Linear Datalog

## Definition (Linear Datalog)

A Datalog program is said to be *linear* if each rule contains at most one occurrence of an IDB in the body.

In other words, each rule looks like this

$$I(x, y) \leftarrow J(w, u, x), \theta_1(x, y, z), \theta_2(x, w)$$

where  $I$  and  $J$  are the only IDB's, or like this

$$I(x, y) \leftarrow \theta_1(x, y, z), \theta_2(x, w).$$

# A non-linear Datalog program

Our program for HORN-3-SAT is *not* linear, since the IDB  $W$  occurs twice in the body of the second rule:

## A non-linear program

$$W(x) \leftarrow 1(x)$$

$$W(x) \leftarrow W(y), W(z), \rho(x, y, z)$$

$$G \leftarrow W(x), 0(x)$$

# A linear Datalog program for *Directed Reachability*

## A linear Datalog program

$$W(x) \leftarrow 1(x)$$
$$W(y) \leftarrow W(x), \theta_{\leq}(x, y)$$
$$G \leftarrow W(x), 0(x)$$

# Expressibility in Linear Datalog

## Theorem

If  $\neg\text{CSP}(\Gamma)$  is definable in Linear Datalog then  $\text{CSP}(\Gamma)$  is in **NL**.

- Idea: the program rejects if and only if there is a *derivation path* that ends in the goal predicate: this amounts to directed reachability

## Another fragment: Symmetric Datalog

### Definition (Symmetric Datalog)

A Datalog program is said to be *symmetric* if (i) it is linear and (ii) it is invariant under symmetry of rules.

In other words, if the program contains the rule

$$I(x, y) \leftarrow J(w, u, x), \theta_1(x, y, z), \theta_2(x, w)$$

then it must also contain its *symmetric*:

$$J(w, u, x) \leftarrow I(x, y), \theta_1(x, y, z), \theta_2(x, w).$$

# A non-symmetric (linear) Datalog program

Our program for *Directed Reachability* is *not* symmetric:

## A non-symmetric linear program

$$W(x) \leftarrow 1(x)$$

$$W(y) \leftarrow W(x), \theta_{\leq}(x, y)$$

$$G \leftarrow W(x), 0(x)$$

# A symmetric Datalog program for *Undirected Reachability*

## A symmetric Datalog program

$$W(x) \leftarrow 1(x)$$
$$W(y) \leftarrow W(x), \theta_=(x, y)$$
$$W(x) \leftarrow W(y), \theta_=(x, y)$$
$$G \leftarrow W(x), 0(x)$$



# Expressibility in Symmetric Datalog

## Theorem (Egri, BL, Tesson, 2007)

*If  $\neg\text{CSP}(\Gamma)$  is definable in Symmetric Datalog then  $\text{CSP}(\Gamma)$  is in  $\mathbf{L}$ .*

- Idea: The program rejects if and only if there is a derivation path that ends in the goal predicate: since the rules are symmetric this amounts to undirected reachability

# Non-expressibility Results

The problems we described above which are complete for  $mod_p\mathbf{L}$ ,  $\mathbf{P}$  and  $\mathbf{NL}$  also have “extremal” properties with respect to expressibility in fragments of Datalog:

## Non-expressibility Results, cont'd

### Theorem (Feder, Vardi, 1993)

Let  $\mu = \{(x, y, z) : x + y = z\}$  and let  $b \neq 0$ . Then  $\neg\text{CSP}(A; \mu, \{b\})$  is not expressible in Datalog.

- This result has been recently extended to more general logics by Atserias, Bulatov, Dawar (2007).

### Theorem (Dalmau, Egri, BL, Tesson, 2007)

- *HORN-3-SAT is not expressible in Linear Datalog.*
- *Directed Reachability is not expressible in Symmetric Datalog.*
- The result on linear Datalog is implicit in Cook, Sethi (1976).

## Non-expressibility Results, cont'd

### Theorem (Feder, Vardi, 1993)

Let  $\mu = \{(x, y, z) : x + y = z\}$  and let  $b \neq 0$ . Then  $\neg\text{CSP}(A; \mu, \{b\})$  is not expressible in Datalog.

- This result has been recently extended to more general logics by Atserias, Bulatov, Dawar (2007).

### Theorem (Dalmau, Egri, BL, Tesson, 2007)

- *HORN-3-SAT* is not expressible in Linear Datalog.
- Directed Reachability is not expressible in Symmetric Datalog.
- The result on linear Datalog is implicit in Cook, Sethi (1976).

## A remark about expressibility

It is still unknown whether the following problems are decidable:

### Decidability for Datalog fragments

- Input: a finite relational structure  $\Gamma$ ;
- Question:  
is  $\neg\text{CSP}(\Gamma)$  expressible in (linear, symmetric) Datalog ?

## Part III

# Some Tame Congruence Theory

## Outline of Part III: TCT

- to every CSP is associated an idempotent algebra  $\mathbb{A}$ ;
- we present a lemma correlating the existence of certain “minimal” algebras in  $\mathcal{V}(\mathbb{A})$  with the *typeset* of  $\mathcal{V}(\mathbb{A})$ ;
- we describe key properties of these “minimal” algebras, connecting them to the problems described in Parts I and II.

## Outline of Part III: TCT

- to every CSP is associated an idempotent algebra  $\mathbb{A}$ ;
- we present a lemma correlating the existence of certain “minimal” algebras in  $\mathcal{V}(\mathbb{A})$  with the *typeset* of  $\mathcal{V}(\mathbb{A})$ ;
- we describe key properties of these “minimal” algebras, connecting them to the problems described in Parts I and II.



## Outline of Part III: TCT

- to every CSP is associated an idempotent algebra  $\mathbb{A}$ ;
- we present a lemma correlating the existence of certain “minimal” algebras in  $\mathcal{V}(\mathbb{A})$  with the *typeset* of  $\mathcal{V}(\mathbb{A})$ ;
- we describe key properties of these “minimal” algebras, connecting them to the problems described in Parts I and II.

# A very vague overview of types

- to each (finite) algebra  $\mathbb{A}$  is associated a set of *types*;
- the possible types are:
  - the *unary type*, or type 1;
  - the *affine type*, or type 2;
  - the *Boolean type*, or type 3;
  - the *lattice type*, or type 4;
  - the *semilattice type*, or type 5.
- the *typeset* of the variety  $\mathcal{V}(\mathbb{A})$  is the union of all typesets of all finite algebras in it.

# A very vague overview of types

- to each (finite) algebra  $\mathbb{A}$  is associated a set of *types*;
- the possible types are:
  - the *unary type*, or type 1;
  - the *affine type*, or type 2;
  - the *Boolean type*, or type 3;
  - the *lattice type*, or type 4;
  - the *semilattice type*, or type 5.
- the *typeset* of the variety  $\mathcal{V}(\mathbb{A})$  is the union of all typesets of all finite algebras in it.

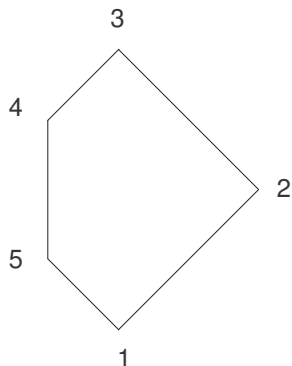
# A very vague overview of types

- to each (finite) algebra  $\mathbb{A}$  is associated a set of *types*;
- the possible types are:
  - the *unary type*, or type 1;
  - the *affine type*, or type 2;
  - the *Boolean type*, or type 3;
  - the *lattice type*, or type 4;
  - the *semilattice type*, or type 5.
- the *typeset* of the variety  $\mathcal{V}(\mathbb{A})$  is the union of all typesets of all finite algebras in it.

# The Ordering of Types

we shall refer later to the following ordering of types:

$$1 < 2 < 3 > 4 > 5 > 1$$



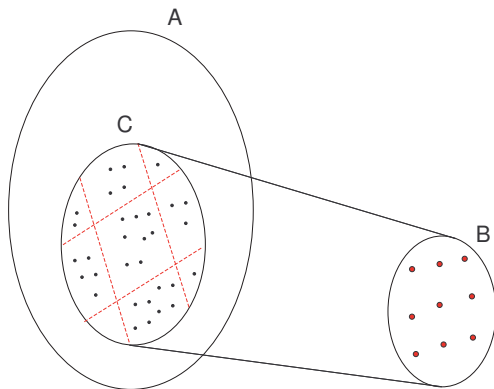
# Factor algebras

## Definition (Factors)

We say that the algebra  $\mathbb{B}$  is a *factor* of the algebra  $\mathbb{A}$  if  $\mathbb{B} \in HS(\mathbb{A})$ , i.e. it is a homomorphic image of a subalgebra of  $\mathbb{A}$ .

## Factor algebras, cont'd

The algebra  $\mathbb{B}$  is a homomorphic image of the subalgebra  $\mathbb{C}$  of  $\mathbb{A}$ , hence  $\mathbb{B}$  is a *factor* of  $\mathbb{A}$ :



# Strictly simple algebras

## Definition (Strictly simple algebra)

An algebra is *strictly simple* if it has no proper factors, i.e. it is simple and has no non-trivial subalgebras.



## A key lemma

- every strictly simple idempotent algebra has a unique type associated to it;
- The next lemma is one of the two key links between typesets and CSP's we shall require:

Lemma (Valeriote, 2007)

*Let  $\mathbb{A}$  be an idempotent algebra, and suppose type  $i$  is in the typeset of  $\mathcal{V}(\mathbb{A})$ . Then  $\mathbb{A}$  has a strictly simple factor of type  $\leq i$ .*

## A key lemma

- every strictly simple idempotent algebra has a unique type associated to it;
- The next lemma is one of the two key links between typesets and CSP's we shall require:

Lemma (Valeriote, 2007)

*Let  $\mathbb{A}$  be an idempotent algebra, and suppose type  $i$  is in the typeset of  $\mathcal{V}(\mathbb{A})$ . Then  $\mathbb{A}$  has a strictly simple factor of type  $\leq i$ .*

## A key lemma

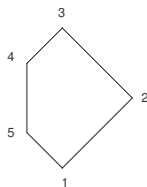
- every strictly simple idempotent algebra has a unique type associated to it;
- The next lemma is one of the two key links between typesets and CSP's we shall require:

### Lemma (Valeriote, 2007)

*Let  $\mathbb{A}$  be an idempotent algebra, and suppose type  $i$  is in the typeset of  $\mathcal{V}(\mathbb{A})$ . Then  $\mathbb{A}$  has a strictly simple factor of type  $\leq i$ .*

## Valeriote's Lemma, cont'd

To illustrate:



- if  $\mathcal{V}(\mathbb{A})$  admits type 1, then  $\mathbb{A}$  has a strictly simple factor of unary type;
- if  $\mathcal{V}(\mathbb{A})$  omits type 1 but admits type 4, then  $\mathbb{A}$  has a strictly simple factor of semilattice type or lattice type.
- Etc.

## A property of strictly simple algebras

- We now have conditions on the existence of strictly simple factors of our algebra  $\mathbb{A}$ ;
- Szendrei (1992) has completely classified these algebras according to their type. We need the following consequences (we split up the result into 4 distinct lemmas):

## A property of strictly simple algebras

- We now have conditions on the existence of strictly simple factors of our algebra  $\mathbb{A}$ ;
- Szendrei (1992) has completely classified these algebras according to their type. We need the following consequences (we split up the result into 4 distinct lemmas):

## A property of strictly simple algebras, cont'd

### Lemma (unary type 1)

*Let  $\mathbb{A}$  be a strictly simple idempotent algebra of unary type. Then it is a 2-element algebra, and its basic operations preserve the relation*

$$\theta = \{0, 1\}^3 \setminus \{(0, 0, 0), (1, 1, 1)\}.$$

### Lemma (affine type 2)

*Let  $\mathbb{A}$  be a strictly simple idempotent algebra of affine type. Then there exists an Abelian group structure on  $A$  such that the basic operations of  $\mathbb{A}$  preserve the relation*

$$\mu = \{(x, y, z) : x + y = z\}.$$

## A property of strictly simple algebras, cont'd

### Lemma (unary type 1)

*Let  $\mathbb{A}$  be a strictly simple idempotent algebra of unary type. Then it is a 2-element algebra, and its basic operations preserve the relation*

$$\theta = \{0, 1\}^3 \setminus \{(0, 0, 0), (1, 1, 1)\}.$$

### Lemma (affine type 2)

*Let  $\mathbb{A}$  be a strictly simple idempotent algebra of affine type. Then there exists an Abelian group structure on  $A$  such that the basic operations of  $\mathbb{A}$  preserve the relation*

$$\mu = \{(x, y, z) : x + y = z\}.$$



## A property of strictly simple algebras, cont'd

### Lemma (lattice type 4)

*Let  $\mathbb{A}$  be a strictly simple idempotent algebra of lattice type. Then it is a 2-element algebra, and its basic operations preserve the usual ordering  $\leq$  on  $\{0, 1\}$ .*

### Lemma (semilattice type 5)

*Let  $\mathbb{A}$  be a strictly simple idempotent algebra of semilattice type. Then it is isomorphic to a 2-element algebra whose basic operations preserve the relation*

$$\rho = \{(x, y, z) : (y \wedge z) \rightarrow x\}.$$

## A property of strictly simple algebras, cont'd

### Lemma (lattice type 4)

*Let  $\mathbb{A}$  be a strictly simple idempotent algebra of lattice type. Then it is a 2-element algebra, and its basic operations preserve the usual ordering  $\leq$  on  $\{0, 1\}$ .*

### Lemma (semilattice type 5)

*Let  $\mathbb{A}$  be a strictly simple idempotent algebra of semilattice type. Then it is isomorphic to a 2-element algebra whose basic operations preserve the relation*

$$\rho = \{(x, y, z) : (y \wedge z) \rightarrow x\}.$$

## Part IV

# Hardness and non-expressibility Results

# Outline of Part IV: Hardness and non-expressibility

A recap of Parts I, II and III:

- From Part I: some specific CSP's that are hard for the complexity classes **NP**, **P**, **NL** and  $mod_p\mathbf{L}$ ;
- from Part II: some specific CSP's that are not expressible in Datalog, Linear Datalog and Symmetric Datalog;
- from Part III:
  - if the variety generated by the idempotent algebra  $A$  admits type  $i$ , then there exists a factor of  $A$  of type  $\leq i$ ;
  - the basic operations of this factor preserve specific relations related to the problems described in Parts I and II.

# Outline of Part IV: Hardness and non-expressibility

A recap of Parts I, II and III:

- From Part I: some specific CSP's that are hard for the complexity classes **NP**, **P**, **NL** and  $mod_p\mathbf{L}$ ;
- from Part II: some specific CSP's that are not expressible in Datalog, Linear Datalog and Symmetric Datalog;
- from Part III:
  - if the variety generated by the idempotent algebra  $A$  admits type  $i$ , then there exists a factor of  $A$  of type  $\leq i$ ;
  - the basic operations of this factor preserve specific relations related to the problems described in Parts I and II.

# Outline of Part IV: Hardness and non-expressibility

A recap of Parts I, II and III:

- From Part I: some specific CSP's that are hard for the complexity classes **NP**, **P**, **NL** and  $mod_p\mathbf{L}$ ;
- from Part II: some specific CSP's that are not expressible in Datalog, Linear Datalog and Symmetric Datalog;
- from Part III:
  - if the variety generated by the idempotent algebra  $\mathbb{A}$  admits type  $i$ , then there exists a factor of  $\mathbb{A}$  of type  $\leq i$ ;
  - the basic operations of this factor preserve specific relations related to the problems described in Parts I and II.

## Outline of Part IV: cont'd

- We describe a lemma that relates the complexity and expressibility of the “factor CSP” to the CSP associated to the algebra  $\mathbb{A}$ ;
- We deduce hardness and non-expressibility results in terms of the typeset of  $\mathcal{V}(\mathbb{A})$ ;
- We present some preliminary evidence for the natural conjectures associated to the above-mentioned results.

## Outline of Part IV: cont'd

- We describe a lemma that relates the complexity and expressibility of the “factor CSP” to the CSP associated to the algebra  $\mathbb{A}$ ;
- We deduce hardness and non-expressibility results in terms of the typeset of  $\mathcal{V}(\mathbb{A})$ ;
- We present some preliminary evidence for the natural conjectures associated to the above-mentioned results.



## Outline of Part IV: cont'd

- We describe a lemma that relates the complexity and expressibility of the “factor CSP” to the CSP associated to the algebra  $\mathbb{A}$ ;
- We deduce hardness and non-expressibility results in terms of the typeset of  $\mathcal{V}(\mathbb{A})$ ;
- We present some preliminary evidence for the natural conjectures associated to the above-mentioned results.

# A reduction lemma

## Lemma (BL, Tesson, 2007)

*Let  $\Gamma$  be a core and let  $\mathbb{A}$  be the (idempotent) algebra associated to  $\text{CSP}(\Gamma)$ . Let  $\mathbb{B}$  be a factor of  $\mathbb{A}$ , and let  $\Gamma'$  be a structure whose basic relations are irredundant and invariant under the operations of  $\mathbb{B}$ . Then*

- there is a first-order reduction of  $\text{CSP}(\Gamma')$  to  $\text{CSP}(\Gamma)$ ;*
- if  $\neg\text{CSP}(\Gamma)$  is expressible in (Linear, Symmetric) Datalog then so is  $\neg\text{CSP}(\Gamma')$ .*

# A reduction lemma

## Lemma (BL, Tesson, 2007)

*Let  $\Gamma$  be a core and let  $\mathbb{A}$  be the (idempotent) algebra associated to  $\text{CSP}(\Gamma)$ . Let  $\mathbb{B}$  be a factor of  $\mathbb{A}$ , and let  $\Gamma'$  be a structure whose basic relations are irredundant and invariant under the operations of  $\mathbb{B}$ . Then*

- *there is a first-order reduction of  $\text{CSP}(\Gamma')$  to  $\text{CSP}(\Gamma)$ ;*
- *if  $\neg\text{CSP}(\Gamma)$  is expressible in (Linear, Symmetric) Datalog then so is  $\neg\text{CSP}(\Gamma')$ .*

# A reduction lemma

## Lemma (BL, Tesson, 2007)

*Let  $\Gamma$  be a core and let  $\mathbb{A}$  be the (idempotent) algebra associated to  $\text{CSP}(\Gamma)$ . Let  $\mathbb{B}$  be a factor of  $\mathbb{A}$ , and let  $\Gamma'$  be a structure whose basic relations are irredundant and invariant under the operations of  $\mathbb{B}$ . Then*

- *there is a first-order reduction of  $\text{CSP}(\Gamma')$  to  $\text{CSP}(\Gamma)$ ;*
- *if  $\neg\text{CSP}(\Gamma)$  is expressible in (Linear, Symmetric) Datalog then so is  $\neg\text{CSP}(\Gamma')$ .*

# Hardness results

## Corollary (1)

Let  $\Gamma$  be a core and let  $\mathbb{A}$  be the (idempotent) algebra associated to  $\text{CSP}(\Gamma)$ .

- (BJK, 2000) If  $\mathcal{V}(\mathbb{A})$  admits the unary type, then  $\text{CSP}(\Gamma)$  is **NP**-complete;
- if  $\mathcal{V}(\mathbb{A})$  admits the affine type, then  $\text{CSP}(\Gamma)$  is  $\text{mod}_p$ **L**-hard ( $\exists p$ );
- if  $\mathcal{V}(\mathbb{A})$  admits the semilattice type, then  $\text{CSP}(\Gamma)$  is **P**-hard;
- if  $\mathcal{V}(\mathbb{A})$  admits the lattice type, then  $\text{CSP}(\Gamma)$  is **NL**-hard.

# Non-expressibility results

## Corollary (2)

Let  $\Gamma$  be a core and let  $\mathbb{A}$  be the (idempotent) algebra associated to  $\text{CSP}(\Gamma)$ .

- (BL, Zádori, 2006) If  $\mathcal{V}(\mathbb{A})$  admits the unary or affine type, then  $\neg\text{CSP}(\Gamma)$  is not expressible in Datalog;
- if  $\mathcal{V}(\mathbb{A})$  admits the semilattice type, then  $\neg\text{CSP}(\Gamma)$  is not expressible in Linear Datalog;
- if  $\mathcal{V}(\mathbb{A})$  admits the lattice type, then  $\neg\text{CSP}(\Gamma)$  is not expressible in Symmetric Datalog.

# Recap

A core  $CSP(\Gamma)$  with associated idempotent algebra  $\mathbb{A}$ .

$\mathcal{V}(\mathbb{A})$			
<i>omits</i>	<i>admits</i>	<i>complexity</i>	<i>expressibility</i>
	1	<b>NP</b> -complete	not Datalog
1	2	$mod_p$ <b>L</b> -hard ( $\exists p$ )	not Datalog
1,2	5	<b>P</b> -hard	not Linear Datalog
1,2,5	4	<b>NL</b> -hard	not Symmetric Datalog

## Some natural conjectures

It is tempting to make the following conjectures: (but I won't)

### Conjecture

*Let  $\Gamma$  be a core and let  $\mathbb{A}$  be the idempotent algebra associated to  $\text{CSP}(\Gamma)$ .*

- *(BJK) If  $\mathcal{V}(\mathbb{A})$  omits type 1 then  $\text{CSP}(\Gamma)$  is in  $\mathbf{P}$ ;*
- *(BL, Z)  $\mathcal{V}(\mathbb{A})$  omits types 1 and 2 if and only if  $\neg\text{CSP}(\Gamma)$  is in Datalog;*
- *$\mathcal{V}(\mathbb{A})$  omits types 1, 2 and 5 if and only if  $\neg\text{CSP}(\Gamma)$  is in Linear Datalog;*
- *$\mathcal{V}(\mathbb{A})$  omits types 1, 2, 4 and 5 if and only if  $\neg\text{CSP}(\Gamma)$  is in Symmetric Datalog.*



# The Boolean Case

$\mathcal{V}(A)$			
<i>omits</i>	<i>admits</i>	<i>complexity</i>	<i>in/not in</i>
	1	<b>NP</b> -complete	-/Datalog
1	2	$\oplus$ <b>L</b> -complete	-/Datalog
1,2	5	<b>P</b> -complete	Datalog/Linear
1,2,5	4	<b>NL</b> -complete	Linear/Symmetric
1,2,4,5		<b>L</b> -complete/FO	Symmetric/-

## More evidence

- $\exists$  majority term (implies congruence-distributive, implies omit 1,2, 5): in Linear Datalog (Dalmau, Krokhin 2007)
- $\exists$  Maltsev term + Datalog (implies omits type 1,2,4,5): in Symmetric Datalog (Dalmau, BL 2007)
- strictly simple algebras of type 3: in Symmetric Datalog (Egri, BL, Tesson, 2007)
- term equivalent to FO (implies only type 3): in Symmetric Datalog (Egri, BL, Tesson, 2007)

## More evidence

- $\exists$  majority term (implies congruence-distributive, implies omit 1,2, 5): in Linear Datalog (Dalmau, Krokhin 2007)
- $\exists$  Maltsev term + Datalog (implies omits type 1,2,4,5): in Symmetric Datalog (Dalmau, BL 2007)
- strictly simple algebras of type 3: in Symmetric Datalog (Egri, BL, Tesson, 2007)
- term equivalent to FO (implies only type 3): in Symmetric Datalog (Egri, BL, Tesson, 2007)

## More evidence

- $\exists$  majority term (implies congruence-distributive, implies omit 1,2, 5): in Linear Datalog (Dalmau, Krokhin 2007)
- $\exists$  Maltsev term + Datalog (implies omits type 1,2,4,5): in Symmetric Datalog (Dalmau, BL 2007)
- strictly simple algebras of type 3: in Symmetric Datalog (Egri, BL, Tesson, 2007)
- term equivalent to FO (implies only type 3): in Symmetric Datalog (Egri, BL, Tesson, 2007)

## More evidence

- $\exists$  majority term (implies congruence-distributive, implies omit 1,2, 5): in Linear Datalog (Dalmau, Krokhin 2007)
- $\exists$  Maltsev term + Datalog (implies omits type 1,2,4,5): in Symmetric Datalog (Dalmau, BL 2007)
- strictly simple algebras of type 3: in Symmetric Datalog (Egri, BL, Tesson, 2007)
- term equivalent to FO (implies only type 3): in Symmetric Datalog (Egri, BL, Tesson, 2007)

## Decidability of types: notes

Let  $\Gamma$  be a core, and let  $\mathbb{A}$  be the idempotent algebra associated to  $CSP(\Gamma)$ .

### Decidability: input is $\Gamma$

- $\mathcal{V}(\mathbb{A})$  omits type 1 ? **NP**-complete (Bul Jea, 2000) ;
- $\mathcal{V}(\mathbb{A})$  omits types 1,2 ? **NP**-complete (Bul);
- $\mathcal{V}(\mathbb{A})$  omits types 1, 2, 5 ? decidable, likely in **NP** (Val);
- $\mathcal{V}(\mathbb{A})$  omits type 1, 2, 4, 5 ? decidable, likely in **NP** (Val).

### Decidability: input is $\mathbb{A}$

All the above: in **P** (Freese, Valeriote, 2007)

## Decidability of types: notes

Let  $\Gamma$  be a core, and let  $\mathbb{A}$  be the idempotent algebra associated to  $CSP(\Gamma)$ .

### Decidability: input is $\Gamma$

- $\mathcal{V}(\mathbb{A})$  omits type 1 ? **NP**-complete (Bul Jea, 2000) ;
- $\mathcal{V}(\mathbb{A})$  omits types 1,2 ? **NP**-complete (Bul);
- $\mathcal{V}(\mathbb{A})$  omits types 1, 2, 5 ? decidable, likely in **NP** (Val);
- $\mathcal{V}(\mathbb{A})$  omits type 1, 2, 4, 5 ? decidable, likely in **NP** (Val).

### Decidability: input is $\mathbb{A}$

All the above: in **P** (Freese, Valeriote, 2007)