

# Modelling GRAMMAR Constraints with ASP

Christian Drescher

NICTA and University of New South Wales

17 February 2011



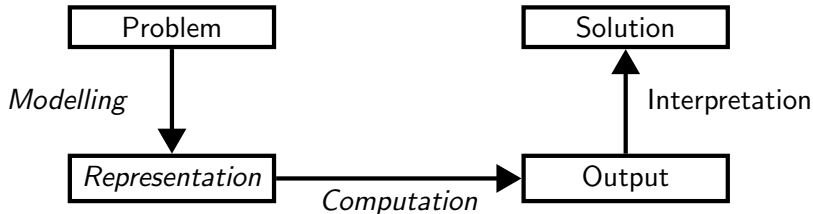
Australian Government  
Department of Broadband, Communications  
and the Digital Economy  
Australian Research Council

## NICTA Funding and Supporting Members and Partners



# Declarative Problem Solving

- **What is the problem?**  
instead of
- How to solve the problem?



As a declarative programming paradigm, ASP combines

- an expressive but simple modelling language, viz. logic
- high-performance solving capacities.

# Introduction

- CSP** combinatorial problems  $(V, D, C)$ , from various areas
- Log. Program** set of rules  $a_0 \leftarrow a_1, \dots, a_m, \text{not } a_{m+1}, \dots, \text{not } b_n$   
over alphabet  $\mathcal{A}$ , and extensions like  
choice rules  $\{a_0\} \leftarrow a_1, \dots, a_m, \text{not } a_{m+1}, \dots, \text{not } b_n$
- Answer Set  $X$**   $\subseteq$ -minimal model of the GL-reduct  $P^X$   
 $\{head(r) \leftarrow body(r)^+ \mid r \in P, body(r)^- \cap X = \emptyset\}$
- Algorithm** Conflict-Driven Nogood Learning, Unit Propagation,  
Heuristic Search, Conflict Resolution, Extended Propagators
- Aspect** breaking symmetry to eliminate symmetric parts of  
the search space

# Topics on Symmetry Breaking for ASP

- EMCL Student Workshop 2010  
A Translational Approach to Constraint Answer Set Solving  
Theory and Practice of Logic Programming
- EMCL Student Workshop Summer 2010  
Symmetry-breaking Answer Set Solving  
Proceedings of 26th ICLP' Workshop on ASPOCP;  
AI Communications Journal, To Appear  
Symmetry Breaking for Distributed Multi-Context Systems  
Proceedings of 11th LPNMR, To Appear
- EMCL Student Workshop 2011  
Modelling GRAMMAR Constraints with ASP

# Outline

- 1 Modelling the GRAMMAR Constraint
- 2 Modelling the REGULAR Constraint
- 3 Modelling the PRECEDENCE Constraint
- 4 Empirical Evaluation
- 5 Conclusions

# Context-Free Languages and GRAMMAR Constraints

## A Context-Free Language $\mathcal{L}$

- is produced by a **Context-Free Grammar**  $\mathcal{G} = (N, \Sigma, P, S)$  with production rules  $P \subseteq N \times (N \cup \Sigma)^*$  of the form  $A ::= \omega$ .  
 $\mathcal{G}$  is in **Chomsky normal form** iff all productions are of the form  $A ::= t$  or  $A ::= BC$ .
- is **linear** iff every production in  $\mathcal{G}$  contains at most one nonterminal in its right-hand side.
- is **regular** iff all productions in  $\mathcal{G}$  are of the form  $A ::= t$  or  $A ::= tB$ .

## Definition

GRAMMAR( $\mathcal{G}, [v_1, \dots, v_n]$ ) is satisfied by just those assignments to the sequence of variables  $(v_1, \dots, v_n)$  which belong to the language produced by  $\mathcal{G}$ .

# Modelling GRAMMAR Constraints with ASP

based on the famous CYK algorithm

- 1 for each production of the form  $A ::= t$  and  $1 \leq i \leq n$

$$\{A(i, 1)\} \leftarrow \llbracket v_i \mapsto t \rrbracket ,$$

- 2 for each production of the form  $A ::= BC$  and  $1 \leq i \leq n$

$$\begin{aligned} \{A(i, j)\} &\leftarrow \omega_{BC}(i, j) \\ \{\omega_{BC}(i, j)\} &\leftarrow B(i, k), C(i+k, j-k) \end{aligned}$$

- 3 encode support for  $v_i \mapsto t$  (takes part of a production from  $S$ )

$$\perp \leftarrow \llbracket v_i \mapsto t \rrbracket, \text{not } A_1(i, 1), \text{not } A_2(i, 1), \dots, \text{not } A_m(i, 1)$$

$$\perp \leftarrow A(i, j), \text{not } \omega_1(i, j), \text{not } \omega_2(i, j), \dots, \text{not } \omega_m(i, j)$$

$$\perp \leftarrow \text{not } S(1, n)$$

## Modelling GRAMMAR Constraints with ASP (ctd)

### Theorem

GRAMMAR *is satisfiable iff our encoding has an answer set.*

### Theorem

*Unit-propagation on our encoding enforces domain consistency on GRAMMAR in  $\mathcal{O}(|P|n^3)$  down any branch of the search.*

## Modelling GRAMMAR Constraints with ASP (ctd)

### Theorem

GRAMMAR *is satisfiable iff our encoding has an answer set.*

### Theorem

*Unit-propagation on our encoding enforces domain consistency on GRAMMAR in  $\mathcal{O}(|P|n^3)$  down any branch of the search.*

### Theorem

*If  $\mathcal{G}$  is **linear** then unit-propagation on our encoding enforces domain consistency on GRAMMAR in  $\mathcal{O}(|P|n^2)$  down any branch of the search.*

### Theorem

*If  $\mathcal{G}$  is **regular** then unit-propagation on our encoding enforces domain consistency on GRAMMAR in  $\mathcal{O}(|P|n)$  down any branch of the search.*

# Regular Languages and REGULAR Constraints

A Regular Language  $\mathcal{L}$

- is recognized by a **Deterministic Finite Automaton** (DFA)  $M = (Q, \Sigma, \delta, q_0, F)$  with transition function  $\delta : Q \times \Sigma \rightarrow Q$ .

## Definition

$\text{REGULAR}(M, [v_1, \dots, v_n])$  is satisfied on just those assignments to the sequence of variables  $(v_1, \dots, v_n)$  which belong to the language recognised by  $M$ .

The  $\text{REGULAR}$  constraint is a special case of the  $\text{GRAMMAR}$  constraint.

## Modelling REGULAR Constraints with ASP

- 1 new atoms  $q_k(i)$  for each state  $q_k$  and  $0 \leq i \leq n$
- 2 for each transition  $\delta(q_j, t) = q_k$  and  $0 \leq i \leq n$

$$q_k(i) \leftarrow q_j(i-1), \llbracket v_i \mapsto t \rrbracket$$

$$d(q_j, q_k, i) \leftarrow q_j(i-1), q_k(i)$$

- 3 encode support for  $v_i \mapsto t$  (being part of  $M$ 's processing)

$$q_0(0) \leftarrow$$

$$\perp \leftarrow q_{rej}(n) \qquad \forall q_{rej} \in Q \setminus F$$

$$\perp \leftarrow \llbracket v_i \mapsto t \rrbracket, \text{not } d(q_{j_1}, q_{k_1}, i), \dots, d(q_{j_m}, q_{k_m}, i) .$$

# Modelling REGULAR Constraints with ASP

- 1 new atoms  $q_k(i)$  for each state  $q_k$  and  $0 \leq i \leq n$
- 2 for each transition  $\delta(q_j, t) = q_k$  and  $0 \leq i \leq n$

$$q_k(i) \leftarrow q_j(i-1), \llbracket v_i \mapsto t \rrbracket$$

$$d(q_j, q_k, i) \leftarrow q_j(i-1), q_k(i)$$

- 3 encode support for  $v_i \mapsto t$  (being part of  $M$ 's processing)

$$q_0(0) \leftarrow$$

$$\perp \leftarrow q_{rej}(n) \qquad \forall q_{rej} \in Q \setminus F$$

$$\perp \leftarrow \llbracket v_i \mapsto t \rrbracket, \text{not } d(q_{j_1}, q_{k_1}, i), \dots, d(q_{j_m}, q_{k_m}, i) .$$

## Theorem

REGULAR *is satisfiable iff our encoding has an answer set.*

## Theorem

*Unit-propagation on our encoding enforces domain consistency on REGULAR in  $\mathcal{O}(|Q|nd)$  down any branch of the search.*

# The PRECEDENCE Constraint

The PRECEDENCE constraint is a special case of the REGULAR constraint.

- is used for breaking symmetries of interchangeable values in a CSP (values are **interchangeable** if we can swap them in any solution).

## Definition

PRECEDENCE( $[t_1, \dots, t_m], [v_1, \dots, v_n]$ ) holds iff  
 $\min(\{i \mid v_i = t_k\} \cup \{n + 1\}) < \min(\{i \mid v_i = t_\ell\} \cup \{n + 2\})$   
for all  $1 \leq k < \ell < m$ .

## Modelling PRECEDENCE Constraints with ASP

- 1 new atom  $taken(t_\ell, j)$  for each  $t_\ell$  and  $0 \leq j \leq n$  indicates whether  $v_i \mapsto t_\ell$  for some  $i < j$
- 2  $t_j$  has been taken if  $v_i \mapsto t_j$ ,  $1 \leq i \leq n$

$$taken(t_j, i+1) \leftarrow \llbracket v_i \mapsto t_j \rrbracket$$

$$taken(t_j, i+1) \leftarrow taken(t_j, i)$$

- 3 encode precedence condition, for all  $1 \leq k < \ell \leq m$

$$\perp \leftarrow \llbracket v_i \mapsto t_\ell \rrbracket, not\ taken(t_k, i)$$

# Modelling PRECEDENCE Constraints with ASP

- 1 new atom  $taken(t_\ell, j)$  for each  $t_\ell$  and  $0 \leq j \leq n$  indicates whether  $v_i \mapsto t_\ell$  for some  $i < j$
- 2  $t_j$  has been taken if  $v_i \mapsto t_j$ ,  $1 \leq i \leq n$

$$taken(t_j, i+1) \leftarrow \llbracket v_i \mapsto t_j \rrbracket$$

$$taken(t_j, i+1) \leftarrow taken(t_j, i)$$

- 3 encode precedence condition, for all  $1 \leq k < \ell \leq m$

$$\perp \leftarrow \llbracket v_i \mapsto t_\ell \rrbracket, not\ taken(t_k, i)$$

## Theorem

PRECEDENCE is satisfiable iff our encoding has an answer set.

## Theorem

Unit-propagation on our encoding enforces domain consistency on PRECEDENCE in  $\mathcal{O}(m^2n)$  down any branch of the search.

# Experiments on Graph Colouring

Benchmark Domain:

- randomly generated 3-, 4- and 5-colouring instances
- 600 instances around the phase transition density with 400, 150, 75 vertices, respectively

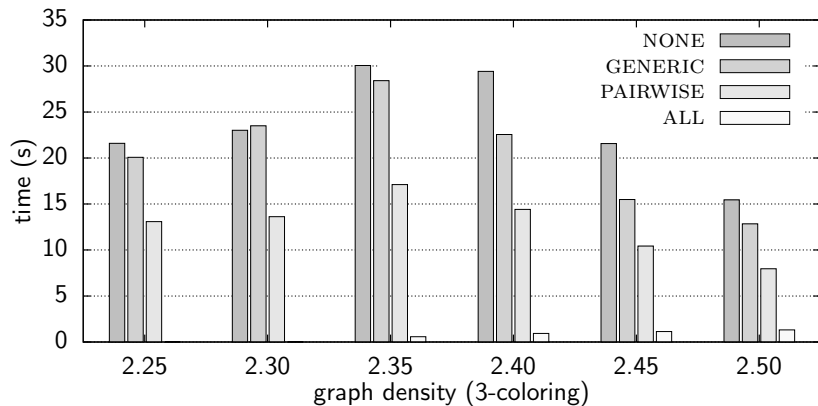
Options:

- ALL uses our ASP encoding for the PRECEDENCE constraint to break all value symmetry
- PAIRWISE posts PRECEDENCE constraints between pairwise interchangeable values
- NONE breaks no symmetry
- GENERIC employs SBASS for symmetry breaking in terms of generators

Setting:

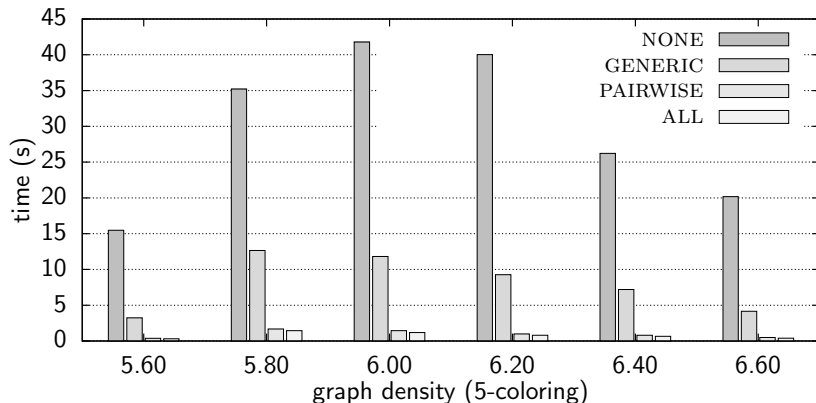
- 2.00 GHz PC/Linux, runs limited to 600s time

## Experiments on Graph Colouring (ctd)



Average time required to solve random 3-colouring instances near the phase transition density

## Experiments on Graph Colouring (ctd)



Average time required to solve random 5-colouring instances near the phase transition density

# Conclusions

We have modelled GRAMMAR, REGULAR, and PRECEDENCE constraints with ASP. Our encodings

- are elaboration tolerant: permit to apply results from formal language theory, e.g., standard techniques for automaton minimisation
- are optimal: the best known constraint propagators have the same runtime complexity
- are experimentally proven to be effective and efficient