Global Grammar Constraints

CLAUDE-GUY QUIMPER[a] AND TOBY WALSH[b]

[a]University of Waterloo, Canada, cquimper@uwaterloo.ca

[b]NICTA and UNSW, Sydney, Australia, toby.walsh@nicta.com.au

ABSTRACT

We consider global constraints over a sequence of variables which restrict the values assigned to be a string within a given language defined by a grammar or automaton. Such constraints are useful in a wide range of scheduling, rostering and sequencing problems. For regular languages, we gave a simple encoding into ternary constraints that can be used to enforce GAC in time linear in the number of variables. We study a number of extensions including regular languages specified by non-deterministic automata, and soft and cyclic versions of the global constraint. For context-free languages, we give two propagators which enforce GAC based on the CYK and Earley parsers.

1 Introduction

Consider a language $\mathcal{L}$, we want to maintain GAC on the constraint.

$$X_1 X_2 \dots X_n \in \mathcal{L}$$

We propose some propagators specialized for two different classes of languages.

1. Regular languages
2. Context-free languages

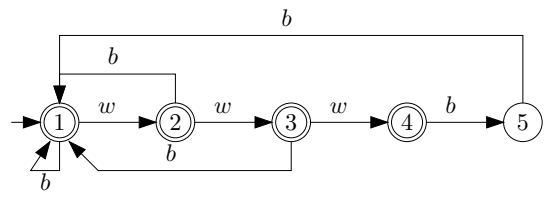
2 Regular Languages

2.1 Regular Languages

Regular languages are those accepted by an automaton. An automaton is fully defined by an alphabet Σ, a set of states Q, an initial state $q_0 \in Q$, a set of final states $F \subseteq Q$, and a transition table $\delta \subseteq Q \times \Sigma \times Q$.

$$\mathcal{A} = \langle \Sigma, Q, q_0, F, \delta \rangle$$

The following automaton ensures that an employee works for a maximum of three consecutive shifts. A sequence of three consecutive shifts must be followed by two periods of break.



$$\delta = \left\{ \begin{array}{lll} (1,b,1), & (1,w,2), & (2,b,1), \quad (2,w,3), \\ (3,b,1), & (3,w,4), & (4,b,5), \quad (5,b,1) \end{array} \right\}$$

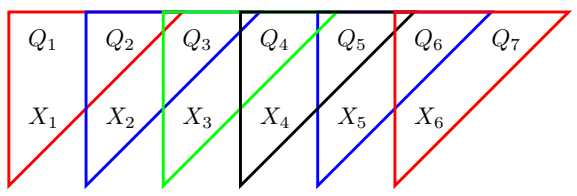
2.2 Decomposition

One can decompose the REGULAR constraint into ternary constraints. Similarly to Beldiceanu et al. for DFAs with counters, we declare the variables $Q_1, \dots, Q_{n+1}$ subject to the following constraints.

$$Q_1 = q_0 \qquad Q_i \in Q \qquad Q_{n+1} \in F$$

$$(Q_i, X_i, Q_{i+1}) \in \delta$$

The constraint graph is Berge-acyclic. Enforcing GAC on the individual ternary constraints achieves GAC on the original REGULAR constraint.



2.3 Running Time Analysis

One can propagate the TABLE constraint in $O(|\delta|)$ steps resulting in an overall time complexity of $O(n|\delta|)$.

2.4 Advantages of the Decomposition

1. The REGULAR constraint can be implemented with n simple and well studied TABLE constraints.
2. The propagation is incremental.
3. The time complexity is the same as the propagator based on dynamic programming [Pesant 04].
4. We have access to the state variables. This can be useful, to specify objective functions (e.g. number of triple shifts).

2.5 NFA vs DFA

A theoretical observation suggests that DFAs have smaller transition tables.

	Maximum number of transitions
DFA	$Q\Sigma$
NFA	$Q^2\Sigma$

However, any DFA specifying the following regular expression has at least 2^k states while a NFA only requires $O(k)$ states.

$$a^*(b|c)^*c(b|c)^{k-1}a^*$$

Conclusion: DFAs are not necessarily faster to propagate. The only factor that affects the running time complexity is the size of the transition table δ.

3 Context-Free Grammars

3.1 Definition

A context-free grammar is a set of productions whose left-hand side is a single non-terminal symbol. For instance, the following productions generate the language of arithmetic expressions such as $(12-6) \times 7$.

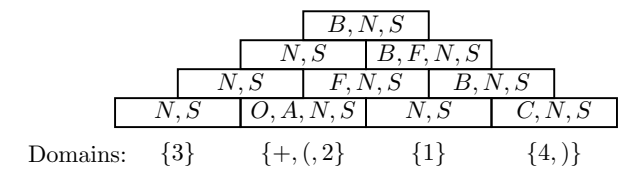
$$\begin{array}{lll} S \to (S) & S \to SOS & S \to N \\ N \to NN & N \to 0|1|\dots|9 & O \to +|-|\times|/ \end{array}$$

3.2 CYK-Style Propagator

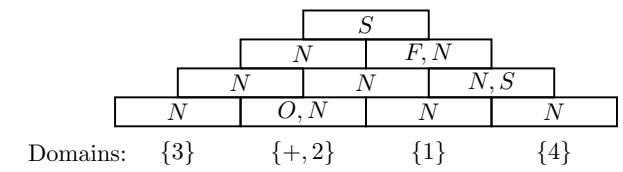
This bottom-up GAC propagator is based on the CYK parser. First, we convert the grammar in its Chomsky form.

$$\begin{array}{lll} S \to AB & S \to SF & S \to NF \\ F \to OS & F \to ON & B \to SC \\ B \to NC & N \to NN & S \to NN \\ A \to (& C \to) & N \to 0|1|\dots|9 \\ O \to +|-|\times| & S \to 0|1|\dots|9 & \end{array}$$

We then fill in a pyramid where each block contains the non-terminals producing the sequence at its basis.



We filter out the non-terminals that do not contribute to the production of the non-terminal S at the summit.



Time and space complexity: $\Theta(|G|n^3)$ where $|G|$ is the size of the grammar and n the number of variables.

3.3 Earley-Style Parser

This top-down GAC propagator iterates through the variables from left to right. The tuple $\langle S \to AbC \bullet D, p, i, S \rangle$ indicates that the production $S \to AbC$ generates the sequence $X_p, \dots, X_{i-1}$ using the supports in S.

1. If $v \in \text{dom}(X_i)$, tuples of the form $\langle w \to \dots \bullet v \dots, p, i, S \rangle$ produce $\langle w \to \dots v \bullet \dots, p, i+1, S \cup \{X_i = v\} \rangle$.
2. Tuples of the form $\langle u \to \dots \bullet v \dots, p, i, S \rangle$ produce $\langle v \to \bullet w, i, i, \emptyset \rangle$ for every rule $v \to w \in G$.
3. In presence of $\langle u \to v \bullet, p, i, S \rangle$, tuples of the form $\langle w \to \dots \bullet u \dots, q, p, T \rangle$ produce $\langle w \to \dots u \bullet \dots, q, i, S \cup T \rangle$.

After processing all variables, the tuple $\langle s \to u \bullet, 0, n+1, S \rangle$ contains the set S of supports.

Time and space complexity: $O(|G|n^3)$ where $|G|$ is the size of the grammar and n the number of variables.

4 Experiments

n	$\|\Sigma\|$	$\|Q\|$	Pesant's REGULAR	Ternary encoding of REGULAR
25	5	20	0.003	**0.003**
		40	0.005	**0.005**
		80	0.008	**0.004**
25	10	20	0.010	**0.006**
		40	0.017	**0.009**
		80	0.028	**0.014**
25	20	20	0.020	**0.008**
		40	0.040	**0.014**
		80	0.081	**0.023**
50	5	20	0.005	**0.004**
		40	0.010	**0.009**
		80	0.017	**0.009**
50	10	20	0.021	**0.013**
		40	0.036	**0.016**
		80	0.063	**0.030**
50	20	20	0.040	**0.018**
		40	0.081	**0.029**
		80	0.166	**0.046**

Table 1. Time in seconds to find a sequence satisfying a randomly generated automaton either using Pesant's propagator for the REGULAR constraint or a ternary encoding using the TABLE constraint

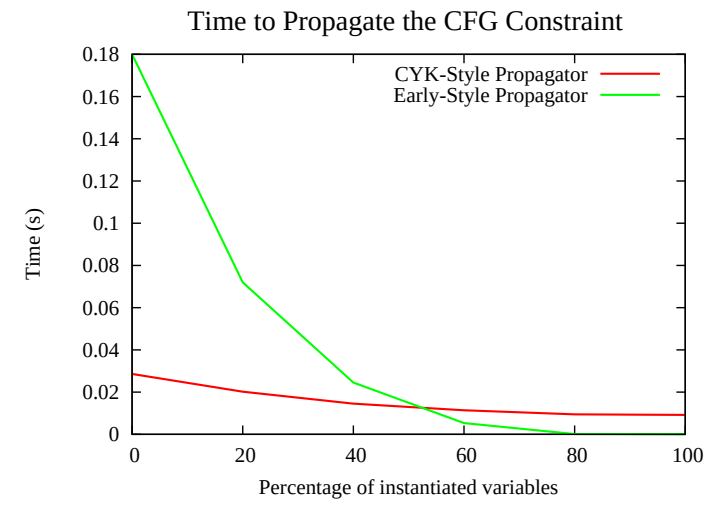


Figure 1. Time in seconds to enforce GAC on a Context-Free Grammar constraint of 30 variables in which the first p percents of the variables are ground. When $p = 100\%$, the problem degenerates to parsing a sequence.

8 Conclusion

- We now have an easy and efficient way to implement the REGULAR constraint with both DFAs and NFAs.
- We have two GAC propagators for the Context-Free Grammar Constraint.