Restricted Global Grammar Constraints.*

George Katsirelos[1], Sebastian Maneth[2], Nina Narodytska[2], and Toby Walsh[2]

[1] NICTA, Sydney, Australia, email: george.katsirelos@nicta.com.au
[2] NICTA and University of NSW, Sydney, Australia, email: sebastian.maneth@nicta.com.au, ninan@cse.unsw.edu.au, toby.walsh@nicta.com.au

Abstract. We investigate the global GRAMMAR constraint over restricted classes of context free grammars like deterministic and unambiguous context-free grammars. We show that detecting disentailment for the GRAMMAR constraint in these cases is as hard as parsing an unrestricted context free grammar. We also consider the class of linear grammars and give a propagator that runs in quadratic time. Finally, to demonstrate the use of linear grammars, we show that a weighted linear GRAMMAR constraint can efficiently encode the EDITDISTANCE constraint, and a conjunction of the EDITDISTANCE constraint and the REGULAR constraint.

1 Introduction

In domains like staff scheduling, regulations can often be naturally expressed using formal grammars. Pesant [9] introduced the global REGULAR constraint to express problems using finite automaton. Sellmann [14] and Quimper and Walsh [11] then introduced the global GRAMMAR constraint for context-free grammars. Unlike parsing which only finds a single solution, propagating such a constraint essentially considers all solutions. Nevertheless, a propagator for the REGULAR constraint runs in linear time and a propagator for the GRAMMAR constraint runs in cubic time just like the corresponding parsers. Subsequently, there has been research on more efficient propagators for these global constraints [12, 10, 4, 7, 6]. Whilst research has focused on regular and unrestricted context-free languages, a large body of work in formal language theory considers grammars between regular and context-free. Several restricted forms of context-free grammars have been proposed that permit linear parsing algorithms whilst being more expressive than regular grammars. Examples of such grammars are LL(k), LR(1), and LALR. Such grammars play an important role in compiler theory. For instance, yacc generates parsers that accept LALR languages.

In this paper we explore the gap between the second and third levels of the Chomsky hierarchy for classes of grammars which can be propagated more efficiently than context-free grammars. These classes of grammar are attractive because either they have a linear or quadratic time membership test (e.g., LR(k) and linear grammars, respectively) or they permit counting of strings of given length in polynomial time (e.g., unambiguous grammars). The latter may be useful in branching heuristics. One of our main contributions is a lower bound on the time complexity for propagating grammar

* NICTA is funded by the Australian Government's Department of Broadband, Communications, and the Digital Economy and the Australian Research Council.

constraints using deterministic or unambiguous grammars. We prove that detecting disentailment for such constraints has the same time complexity as the best parsing algorithm for an arbitrary context-free grammar. Using LL(k) languages or unambiguous grammars does not therefore improve the efficiency of propagation. Another contribution is to show that linearity of the grammar permits quadratic time propagation. We show that we can encode an EDITDISTANCE constraint and a combination of an EDITDISTANCE constraint and a REGULAR constraint using such a linear grammar. Experimental results show that this encoding is very efficient in practice.

2 Background

A context-free grammar is a tuple $G = \langle N, T, P, S \rangle$, where N is a finite set of *non-terminal* symbols, T is a finite set of *terminal* symbols, P is a set of *productions*, and $S \in N$ is the *start symbol*. A production is of the form $A \rightarrow \alpha$ where $A \in N$ and $\alpha \in (N \cup T)^+$. The derivation relation $\Rightarrow_G$ induced by G is defined as follows: for any $u, v \in (N \cup T)^*$, $uAv \Rightarrow_G u\alpha v$ if there exists a production $A \rightarrow \alpha$ in P. Sometimes we additionally index $\Rightarrow_G$ by the production that is applied. The transitive, reflexive closure of $\Rightarrow_G$ is denoted by $\Rightarrow_G^*$. A string in $s \in T^*$ is *generated* by G if $S \Rightarrow_G^* s$. The set of all strings generated by G is denoted $L(G)$. Note that this does not allow the *empty string* ε as right-hand side of a production. Hence, $\varepsilon \notin L(G)$. This is not a restriction: we can add a new start symbol Z with productions $Z \rightarrow \varepsilon \mid S$ to our grammars. Our results can easily be generalized to such ε-enriched grammars. We denote the length of a string s by $|s|$. The *size* of G, denoted by $|G|$, is $\sum_{A \rightarrow \alpha \in P} |A\alpha|$.

A context-free grammar is in *Chomsky form* if all productions are of the form $A \rightarrow BC$ where B and C are non-terminals or $A \rightarrow a$ where a is a terminal. Any ε-free context-free grammar G can be converted to an equivalent grammar G' in Chomsky form with at most a linear increase in its size; in fact, $|G'| \leq 3|G|$, see Section 4.5 in [3]. A context-free grammar is in *Greibach form* if all productions are of the form $A \rightarrow a\alpha$ where a is a terminal and α is a (possibly empty) sequence of non-terminals. Any context-free grammar G can be converted to an equivalent grammar G' in Greibach form with at most a polynomial increase in its size; in fact, the size of G' is in $O(|G|^4)$ in general, and is in $O(|G|^3)$ if G has no chain productions of the form $A \rightarrow B$ for nonterminals A, B, see [1]. A context-free grammar is *regular* if all productions are of the forms $A \rightarrow w$ or $A \rightarrow wB$ for non-terminals A, B and $w \in T^+$.

3 Simple Context-Free Grammars

In this section we show that propagating a *simple* context-free grammar constraint is at least as hard as parsing an (unrestricted) context-free grammar. A grammar G is *simple* if it is in Greibach form, and for every non-terminal A and terminal a there is at most one production of the form $A \rightarrow a\alpha$. Hence, restricting ourselves to languages recognized by simple context-free grammars does not improve the complexity of propagating a global grammar constraint. Simple context-free languages are included in the deterministic context-free languages (characterized by deterministic push-down automata), and also in the $LL(1)$ languages [13], so this result also holds for propagating these

classes of languages. Given finite sets $D_1, \ldots, D_n$, their *Cartesian product language* $L(R_{D_1,\ldots,D_n})$ is the cross product of the domains $\{a_1 a_2 \cdots a_n \mid a_1 \in D_1, \ldots, a_n \in D_n\}$. Following [14], we define the global GRAMMAR constraint:

Definition 1. *The* GRAMMAR$([X_1, \ldots, X_n], G)$ *constraint is true for an assignment variables X iff a string formed by this assignment belongs to $L(G)$.*

From Definition 1, we observe that finding a support for the grammar constraint is equivalent to intersecting the context-free language with the Cartesian product language of the domains.

Proposition 1. *Let G be a context-free grammar, $X_1, \ldots, X_n$ be a sequence of variables with domains $D(X_1), \ldots, D(X_n)$. Then $L(G) \cap L(R_{D(X_1),\ldots,D(X_n)}) \neq 0$ iff* GRAMMAR$([X_1, \ldots, X_n], G)$ *has a support.*

Context-free grammars are effectively closed under intersection with regular grammars. To see this, consider a context-free grammar G in Chomsky form and a regular grammar R. Following the "triple construction", the intersection grammar has non-terminals of the form $\langle F, A, F' \rangle$ where F, F' are non-terminals of R and A is a non-terminal of G. Intuitively, $\langle F, A, F' \rangle$ generates strings w that are generated by A and also by F, through a derivation from F to wF'. If $A \rightarrow BC$ is a production of G, then we add, for all non-terminals F, F', F'' of R, the production $\langle F, A, F'' \rangle \rightarrow \langle F, B, F' \rangle \langle F', C, F'' \rangle$. The resulting grammar is $O(|G|n^3)$ in size where n is the number of non-terminals of R. This is similar to the construction of Theorem 6.5 in [3] which uses push-down automata instead of grammars. Since emptiness of context-free grammars takes linear time (cf. [3]) we obtain through Proposition 1 a cubic time algorithm to check whether a global constraint GRAMMAR$([X_1, \ldots, X_n], G)$ has support. In fact, this shows that we can efficiently propagate more complex constraints, such as the conjunction of a context-free with a regular constraint. Note that if R is a Cartesian product language then the triple construction generates the same result as the CYK based propagator for the GRAMMAR constraint [14, 11].

We now show that for *simple* context-free grammars G, detecting disentailment of the constraint GRAMMAR$([X_1, \ldots, X_n], G)$, i.e. testing whether it has a solution, is at least as hard as parsing an arbitrary context-free grammar.

Theorem 1. *Let G be a context-free grammar in Greibach form and s a string of length n. One can construct in $O(|G|)$ time a simple context-free grammar G' and in $O(|G|n)$ time a Cartesian product language $L(R_{D(X_1),\ldots,D(X_n)})$ such that $L(G') \cap L(R_{D(X_1),\ldots,D(X_n)}) \neq \emptyset$ iff $s \in L(G)$.*

Proof. The idea behind the proof is to determinize an unrestricted context free grammar G by mapping each terminal in G to a set of pairs – the terminal and a production that can consume this terminal. This allows us to carry information about the derivation inside a string in G'. Then, we construct a Cartesian product language $L(R_{D(X_1),\ldots,D(X_n)})$ over these pairs so that all strings from this language map only to the string s. Let $G = \langle N, T, P, S \rangle$ and fix an arbitrary order of the productions in P. We now construct the grammar $G' = \langle N, T', P', S \rangle$. For every $1 \leq j \leq |P|$, if the j-th production of P is $A \rightarrow a\alpha$ then let (a, j) be a new symbol in T' and let the production

$A \rightarrow (a, j)\alpha$ be in P'. Next, we construct the Cartesian product language. We define $D(X_i) = \{(a, j) | (s_i = a) \wedge (a, j) \in T'\}$, $i = 1, \ldots, n$ and s_i is the i-th letter of s. Clearly, G' is constructed in $O(|G|)$ time and $L(R_{D(X_1),\ldots,D(X_n)})$ in $O(|P|n)$ time.

($\Rightarrow$) Let $L(G') \cap L(R_{D(X_1),\ldots,D(X_n)})$ be non empty. Then there exits a string s' that belongs to the intersection. Let $s' = (a_1, i_1) \cdots (a_n, i_n)$. By the definition of $L(R_{D(X_1),\ldots,D(X_n)})$, the string $a_1 a_2 \cdots a_n$ must equal s. Since $s' \in L(G')$, there must be a derivation by G of the form

$$S \Rightarrow_{G,p_1} a_1\alpha \Rightarrow_{G,p_2} a_1 a_2 \alpha' \cdots \Rightarrow_{G,p_n} a_1 \cdots a_n$$

where p_j is the j-th production in P. Hence, $s \in L(G)$.

($\Leftarrow$) Let $s \in L(G)$. Consider a derivation sequence of the string s. We replace every symbol a in s that was derived by the i-th production of G by (a, i). By the construction of G', the string s' is in $L(G')$. Moreover, s' is also in $L(R_{D(X_1),\ldots,D(X_n)})$. □

Note that context-free parsing has a quadratic time lower bound, due to its connection to matrix multiplication [8]. Given this lower bound and the fact that the construction of Theorem 1 requires only linear time, we can deduce the following.

Corollary 1. *Let G be a context-free grammar. If G is simple (or deterministic or $LL(1)$) then detecting disentailment of* GRAMMAR$([X_1, \ldots, X_n], G)$ *is at least as hard as context-free parsing of a string of length n.*

We now show the converse to Theorem 1 which reduces intersection emptiness of a context-free with a regular grammar, to the membership problem of context-free languages. This shows that the time complexity of detecting disentailment for the GRAMMAR constraint is the same as the time complexity of the best parsing algorithm for an arbitrary context free grammar. Therefore, our result shows that detecting disentailment takes $O(n^{2.4})$ time [2], as in the best known algorithm for Boolean matrix multiplication. It does not, however, improve the asymptotic complexity of a domain consistency propagator for the GRAMMAR constraint [14, 11].

Theorem 2. *Let $G = \langle N, T, P, S\rangle$ be a context-free grammar and $L(R_{D(X_1),\ldots,D(X_n)})$ be Cartesian product language. One can construct in time $O(|G| + |T|^2)$ a context-free grammar G' and in time $O(n|T|)$ a string s such that $s \in L(G')$ iff $L(G) \cap L(R_{D(X_1),\ldots,D(X_n)}) \neq \emptyset$.*

Proof. (Sketch) We assign an index to each terminal in T. For each position i of the strings of R, we create a bitmap of the alphabet that describes the terminals that may appear in that position. The j-th bit of the bitmap is 1 iff the symbol with index j may appear at position i. The string s is the concatenation of the bitmaps for each position and has size $n|T|$. First, we add $B \rightarrow 0$ and $B \rightarrow 1$ to G'. For each terminal in T with index j, we introduce $T_j \rightarrow B^{j-1}1B^{|T|-j}$ into G' to accept any bitmap with 1 at the j-th position. Then, for each production in G of the form $A \rightarrow a\alpha$ such that the index of a is j, we add $A \rightarrow T_j\alpha$ to G'. In this construction, every production in G' except for those with T_i on the left hand side can be uniquely mapped to a production in G. It can be shown that $s \in L(G')$ iff $L(G) \cap L(R_{D(X_1),\ldots,D(X_n)}) \neq \emptyset$. □

4 Linear Context-Free Grammars

A context-free grammar is *linear* if every production contains at most one non-terminal in its right-hand side. The linear languages are a proper superset of the regular languages and are a strict subset of the context-free languages. Linear context-free grammars possess two important properties: (1) membership of a given string of length n can be checked in time $O(n^2)$ (see Theorem 12.3 in [15]), and (2) the class is closed under intersection with regular grammars (to see this, apply the "triple construction" as explained after Proposition 1). The second property opens the possibility of constructing a polynomial time propagator for a conjunction of the the linear GRAMMAR and the REGULAR constraints. Interestingly, we can show that a CYK-based propagator for this type of grammars runs in quadratic time. This is then the third example of a grammar, besides regular and context-free grammars, where the asymptotic time complexity of the parsing algorithm and that of the corresponding propagator are equal.

Theorem 3. *Let G be a linear grammar and* GRAMMAR$([X_1, \ldots, X_n], G)$ *be the corresponding global constraint. There exists a domain consistency propagator for this constraint that runs in $O(n^2|G|)$ time.*

Proof. We convert $G = \langle N, T, P, S\rangle$ into CNF. Every linear grammar can be converted into the form $A \to aB$, $A \to Ba$ and $A \to a$, where $a, b \in T$ and $A, B \in N$ (see Theorem 12.3 of [15]) in $O(|G|)$ time. To obtain CNF we replace every terminal $a \in T$ that occurs in a production on the right hand side with a new non-terminal Y_a and introduce a production $Y_a \to a$.

Consider the CYK-based domain consistency propagator for an arbitrary context-free grammar constraint [11, 14]. The algorithm runs in two stages. In the first stage, it constructs in a bottom-up fashion a dynamic programing table $V_{n\times n}$, where an element A of $V_{i,j}$ is a potential non-terminal that generates a substring from the domains of variables $[X_i, \ldots, X_{i+j}]$. In the second stage, it performs a top-down traversal of V and marks an element A of $V_{i,j}$ iff it is reachable from the starting non-terminal S using productions of the grammar and elements of V. It then removes unmarked elements, including terminals. If it removes a terminal at column i of the table, it prunes the corresponding value of variable X_i.

The complexity of this algorithm is bounded by the number of possible 1-step derivations from each non-terminal in the table. Let $G' = \langle N', T', P', S'\rangle$ be an arbitrary context free grammar. There are $O(|N'|n^2)$ non-terminals in the table and each non-terminal can be expanded in $O(F'(A)n)$ possible ways, where $F'(A)$ is the number of productions in G' with non-terminal A on the left-hand side. Therefore, the total time complexity of the propagator for unrestricted context-free grammars is $n^2 \sum_{A\in N'} nF'(A) = O(n^3|G'|)$. In contrast, the number of possible 1-step derivations from each non-terminal in linear grammars is bounded by $O(F(A))$. Therefore, the propagator runs in $O(n^2|G|)$ for a linear grammar G. □

Theorem 3 can be extended to the weighted form of the linear GRAMMAR constraint, WEIGHTEDCFG [5]. A weighted grammar is annotated with a weight for each production and the weight of a derivation is the sum of all weights used in it. The linear WEIGHTEDCFG$(G, Z, [X_1, \ldots, X_n])$ constraint holds iff an assignment X

forms a string belonging to the weighted linear grammar G and the minimal weight derivation of X is less than or equal to Z. The domain consistency propagator for the WEIGHTEDCFG constraint is an extension of the propagator for GRAMMAR that computes additional information for each non-terminal $A \in V_{i,j}$—the minimum and the maximum weight derivations from A. Therefore, this algorithm has the same time and space asymptotic complexity as the propagator for GRAMMAR, so the complexity analysis for the linear WEIGHTEDCFG constraint is identical to the non-weighted case.

It is possible to restrict linear grammars further, so that the resulting global constraint problem is solvable in *linear time*. As an example, consider "fixed-growth" grammars in which there exists l and r with $l + r \geq 1$ such that every production is of the form either $A \rightarrow w \in T^+$ or $A \rightarrow uBw$ where the length of $u \in T^*$ equals l and the length of $w \in T^*$ equals r. In this case, the triple construction (explained below Proposition 1) generates $O(|G|n)$ new non-terminals implying linear time propagation (similarly, CYK runs in linear time as it only generates non-terminals on the diagonal of the dynamic program). A special case of fixed-growth grammars are regular grammars which have $l = 1$ and $r = 0$ (or vice versa).

5 The EDITDISTANCE Constraint

To illustrate linear context-free grammars, we show how to encode an edit distance constraint into such a grammar. EDITDISTANCE$([X_1, \ldots, X_n, Y_1, \ldots, Y_m], N)$ holds iff the edit distance between assignments of two sequences of variables $\mathbf{X}$ and $\mathbf{Y}$ is less than or equal to N. The edit distance is the minimum number of deletion, insertion and substitution operations required to convert one string into another. Each of these operations can change one symbol in a string. W.L.O.G. we assume that $n = m$. We will show that the EDITDISTANCE constraint can be encoded as a linear WEIGHTEDCFG constraint. The idea of the encoding is to parse matching substrings using productions of weight 0 and to parse edits using productions of weight 1.

We convert EDITDISTANCE$([\mathbf{X}, \mathbf{Y}], N)$ into a linear WEIGHTEDCFG$([\mathbf{Z}_{2n+1}, N, G_{ed})$ constraint. The first n variables in the sequence $\mathbf{Z}$ are equal to the sequence $\mathbf{X}$, the variable Z_{n+1} is ground to the sentinel symbol $\#$ so that the grammar can distinguish the sequences $\mathbf{X}$ and $\mathbf{Y}$, and the last n variables of the sequence $\mathbf{Z}$ are equal to the reverse of the sequence $\mathbf{Y}$. We define the linear weighted grammar G_{ed} as follows. Rules $S \rightarrow dSd$ with weight $w = 0$, $\forall d \in D(X) \cup D(Y)$, capture matching terminals, rules $S \rightarrow d_1 S d_2$ with $w = 1$, $\forall d_1 \in D(X), d_2 \in D(Y), d_1 \neq d_2$, capture replacement, rules $S \rightarrow dS|Sd$ with $w = 1$, $\forall d \in D(X)$, capture insertions and deletions. Finally, the rule $S \rightarrow \#$ with weight $w = 0$ generates the sentinel symbol. As discussed in the previous section, the propagator for the linear WEIGHTEDCFG constraint takes $O(n^2|G|)$ time. Down a branch of the search tree, the time complexity is $O(n^2|G|ub(N))$.

We can use this encoding of the EDITDISTANCE constraint into a linear WEIGHTEDCFG constraint to construct propagators for more complex constraints. For instance, we can exploit the fact that linear grammars are closed under intersection with regular grammars to propagate efficiently the conjunction of an EDITDISTANCE constraint and REGULAR constraints on each of the sequences $\mathbf{X}$,$\mathbf{Y}$. More formally,

let $\mathbf{X}$ and $\mathbf{Y}$ be two sequences of variables of length n subject to the constraints REGULAR$(\mathbf{X}, R_1)$, REGULAR$(\mathbf{Y}, R_2)$ and EDITDISTANCE$(\mathbf{X}, \mathbf{Y}, N)$. We construct a domain consistency propagator for the conjunction of these three constraints, by computing a grammar that generates strings of length $2n+1$ which satisfy the conjunction. First, we construct an automaton that accepts $\mathcal{L}(R_1)\#\mathcal{L}(R_2)^R$. This language is regular and requires an automaton of size $O(|R_1|+|R_2|)$. Second, we intersect this with the linear weighted grammar that encodes the EDITDISTANCE constraint using the "triple construction". The size of the obtained grammar is $G_\wedge = |G_{ed}|(|R_1|+|R_2|)^2$ and this grammar is a weighted linear grammar. Therefore, we can use the linear WEIGHTEDCFG$(\mathbf{Z}, N, G_\wedge)$ constraint to encode the conjunction. Note that the size of $G_\wedge$ is only quadratic in $|R_1|+|R_2|$, because G_{ed} is a linear grammar. The time complexity to enforce domain consistency on this conjunction of constraints is $O(n^2|G_\wedge|) = O(n^2d^2(|R_1|+|R_2|)^2)$ for each invocation and $O(n^2d^2(|R_1|+|R_2|)^2 ub(N))$ down a branch of the search tree.

Table 1. Performance of the encoding into WEIGHTEDCFG constraints shown in: number of instances solved in 60 sec / average number of choice points / average time to solve.

n	N	ED_{Dec}			$ED_\wedge$		
		#solved	#choice points	time	#solved	#choice points	time
15	2	**100**	**29**	**0.025**	**100**	6	0.048
20	2	**100**	661	0.337	**100**	**6**	**0.104**
25	3	93	2892	2.013	**100**	**10**	**0.226**
30	3	71	6001	4.987	**100**	**12**	**0.377**
35	4	58	5654	6.300	**100**	**17**	**0.667**
40	4	40	3140	4.690	**100**	**17**	**0.985**
45	5	36	1040	2.313	**100**	**19**	**1.460**
50	5	26	1180	4.848	**100**	**24**	**1.989**
TOTALS							
solved/total			524 /800			**800** /800	
avg time for solved			2.557			**0.732**	
avg choice points for solved			2454			**14**	

To evaluate the performance of the WEIGHTEDCFG$(\mathbf{Z}, N, G_\wedge)$ constraint we carried out a series of experiments on random problems. In out first model the conjunction of the EDITDISTANCE constraint and two REGULAR constraints was encoded with a single WEIGHTEDCFG$(\mathbf{Z}, N, G_\wedge)$ constraint. We call this model $ED_\wedge$. The second model contains the EDITDISTANCE constraint, encoded as WEIGHTEDCFG$(\mathbf{Z}, N, G_{ed})$, and two REGULAR constraints. The REGULAR constraint for the model ED_{Dec} is implemented using a decomposition into ternary table constraints [11]. The WEIGHTEDCFG constraint is implemented with an incremental monolithic propagator [5]. The first REGULAR constraint ensures that there are at most two consecutive values one in the sequence. The second encodes a randomly generated string of 0s and 1s. To make problems harder, we enforced the EDITDISTANCE con-

straint and the REGULAR constraints on two sequences $\mathbf{X}\#(\mathbf{Y})^R$ and $\mathbf{X}'\#(\mathbf{Y}')^R$ of the same length $2n+1$. The EDITDISTANCE constraint and the first REGULAR constraint are identical for these two sequences, while $\mathbf{Y}$ and $\mathbf{Y}'$ correspond to different randomly generated strings of 0s and 1s. Moreover, $\mathbf{X}$ and $\mathbf{X}'$ overlap on 15% of randomly chosen variables. For each possible value of $n \in \{15, 20, 25, 30, 35, 40, 45, 50\}$, we generated 100 instances. Note that n is the length of each sequence $\mathbf{X}$, $\mathbf{Y}$, $\mathbf{X}'$ and $\mathbf{Y}'$. N is the maximum edit distance between $\mathbf{X}$ and $\mathbf{Y}$ and between $\mathbf{X}'$ and $\mathbf{Y}'$. We used a random value and variable ordering and a time out of 60 sec. Results for different values of n are presented in Table 1. As can be seen from the table, the model $ED_\wedge$ significantly outperforms the model ED_{Dec} for larger problems, but it is slightly slower for smaller problems. Note that the model $ED_\wedge$ solves many more instances compared to ED_{Dec}.

6 Conclusions

Unlike parsing, restrictions on context free grammars such as determinism do not improve the efficiency of propagation of the corresponding global GRAMMAR constraint. On the other hand, one specific syntactic restriction, that of linearity, allows propagation in quadratic time. We demonstrated an application of such a restricted grammar in encoding the EDITDISTANCE constraint and more complex constraints.

References

1. N. Blum and R. Koch. Greibach normal form transformation revisited. *Inf. Comput.*, 150:112–118, 1999.
2. D. Coppersmith and S. Winograd. Matrix multiplication via arithmetic progressions. *J. Symbolic Comput.*, 9:251–280, 1990.
3. J. W. Hopcroft and J. D. Ullman. *Introduction to automata theory, languages, and computation*. Addison-Wesley, 1979.
4. S. Kadioglu and M. Sellmann. Efficient context-free grammar constraints. In *AAAI*, pages 310–316, 2008.
5. G. Katsirelos, N. Narodytska, and T. Walsh. The weighted CFG constraint. In *CPAIOR*, pages 323–327, 2008.
6. G. Katsirelos, N. Narodytska, and T. Walsh. Reformulating global grammar constraints. In *CPAIOR09*, pages 132–147, 2009.
7. M. Lagerkvist. *Techniques for Efficient Constraint Propagation*. PhD thesis, KTH, Sweden, 2008.
8. L. Lee. Fast context-free grammar parsing requires fast boolean matrix multiplication. *J. ACM*, 49:1–15, 2002.
9. G. Pesant. A regular language membership constraint for finite sequences of variables. In *CP*, pages 482–495, 2004.
10. C. Quimper and T. Walsh. Decompositions of grammar constraints. In *AAAI*, pages 1567–1570, 2008.
11. C. G. Quimper and T. Walsh. Global grammar constraints. In *CP*, pages 751–755, 2006.
12. C. G. Quimper and T. Walsh. Decomposing global grammar constraints. In *CP*, pages 590–604, 2007.
13. G. Rozenberg and A. Salomaa. *Handbook of Formal Languages*, volume 1. Springer, 2004.
14. M. Sellmann. The theory of grammar constraints. In *CP*, pages 530–544, 2006.
15. K. Wagner and G. Wechsung. *Computational Complexity*. Springer, 1986.