Bounded Length UCFG Equivalence

B. Litow

Dept. of Computer Science, James Cook University

Abstract. A randomised polylog time algorithm is given for deciding whether or not the sets of words of a given length generated by two unambiguous context-free grammars coincide. The algorithm is in randomised NC4 in terms of the product of the grammar size and the length.

1 Introduction

1.1 Statement of results

Deciding whether or not two unambiguous context-free grammars (UCFG) generate the same language is not known to be decidable. In this paper we investigate the complexity of the bounded length version of this problem UCFGBEQ. Let G_1, G_2 be UCFG with the same terminal alphabet A and let n be a positive integer. $L(G_1), L(G_2)$ are the languages generated by G_1, G_2, respectively. UCFGBEQ is the problem of deciding whether or not $L(G_1) \cap A^n = L(G_2) \cap A^n$.

Note that an hypothesis like unambiguousness appears to be essential because the complementary problem, bounded length inequivalence is NP-complete for right linear grammars [3]. An easy way to see this is by reduction from Hamiltonian circuit. Sketching the reduction, the set of length n cycles in a given graph can be presented by a right linear grammar computable in PTIME in n and the graph size. The same can be done for the length n cycles containing repeated nodes. This grammar is probably going to be highly ambiguous, since guessing is used to see whether a node symbol occurs twice. The graph has a Hamiltonian circuit iff the two grammars generate distinct languages.

There has been some effort to investigate the complexity of a variety of language problems. Results are reported in [10], and an extensive study centering on monoid problems and ranking is carried out by Huynh in [6, 5, 7]. More recent work on ranking for UCFG has been carried out in [2] and using a different method in [11]. Both approaches show that UCFG ranking is in DIV, the class of problems NC1 reducible to integer division.

Although several language problems admit NC type algorithms it appears that randomised techniques have not hitherto been used to obtain polylog parallel time solutions to language problems. In this paper, we present a randomised NC (RNC) algorithm for UCFGBEQ.

Theorem 1. *UCFGBEQ is in RNC4 in terms of the product of n, given in unary notation and the grammar size.*

In general randomised NC algorithms are relatively uncommon. However, the maximum matching algorithm presented in [12] is an example of a randomised NC (RNC2 in fact) algorithm.

1.2 Outline of the paper

Section 2 contains preliminaries from context-free language theory and a few facts about related formal series. Section 3 presents the randomisation result, which can be viewed as a kind of sieve. The randomisation is based on an elementary fact from Galois theory, and may have other algorithmic applications. Section 4 gives the proof of Theorem 1.

2 Context-free grammar preliminaries

A context-free grammar (CFG) G is a 4-tuple, $G = (A, X, x_0, P)$, where A is the terminal set, X is the set of nonterminals, $x_0 \in X$ is the start symbol, and P is the set of productions. The size of a grammar will be the sum of the lengths of the right hand sides of all its productions. The reader should refer to a standard text, e.g., [4] for other common CFG related terms and definitions.

Lemma 2. *Every UCFG can be converted in NC2 (in terms of the grammar size) into a UCFG in Chomsky Normal Form.*

Proof Sketch. It is a straightforward that the standard conversion into CNF (e.g. [4]) does not introduce new ambiguity. This conversion can easily be carried out in DLOG, hence certainly in NC2.

Henceforth we will assume that G_1, G_2 are UCFG in CNF.

For our purpose a *formal series* f is a mapping from A^* into into the nonnegative integers. f can be viewed as the formal sum $\sum_{w \in A^*} f(w) \cdot w$. A detailed account of the formal series viewpoint in language theory is presented in [8]. A *solution* of a CFG is a list of formal series, $f_0, \ldots, f_r$ corresponding to the nonterminals, $x_0, \ldots, x_r$, such that for each x_i the equation $x_i = \beta_1 + \cdots + \beta_s$ becomes a formal series identity when each nonterminal is replaced by its corresponding series throughout all the equations. $\beta_1, \ldots, \beta_s$ are the right hand sides of the x_i productions.

Lemma 3. *A CNF grammar G has a unique solution $(f_0, \ldots, f_r)$ and $f_i(w)$ is the number of leftmost derivations of w from x_i.*

Proof. This follows from Theorems 14.9 and 14.11 of [8].

Remark Note that if G is a UCFG, then $f_i(w) \leq 1$.

Given any CFG G in CNF we define a CFG G^n in CNF such that $L(G^n) = L(G) \cap A^n$. $G^n = (A, X^n, (x_0, 1, n), P^n)$, where

- $X^n = \{(x, i, j) \mid x \in X, 1 \leq i \leq j \leq n\}$.
- • $(x, i, i) \to a \in P^n$ iff $x \to a \in P$.
  • If $i < j$, then $(x, i, j) \to (y, i, k)(z, k+1, j) \in P^n$ iff $x \to yz \in P$.

It is possible that the above construction could introduce useless symbols. The standard algorithm for useless symbol removal, e.g., [4] can be carried out in NC2. We will assume this for G^n in what follows.

Lemma 4. *G^n is in CNF and has a unique solution $(f_0^n, \ldots)$ such that if $|w| \neq n$, then $f_0^n(w) = 0$, and if $|w| = n$, then $f_0^n(w) = f_0(w)$, where f_0 is the unique solution element corresponding to x_0.*

Proof Sketch. It is obvious from its definition that G^n is in CNF. By Lemma 3 both G and G^n have unique solutions. An induction on the difference $j-i$ entirely similar to the proof of correctness of the Cocke-Younger-Kasami algorithm can be used to establish the claims about f_0^n.

3 A randomised sieve

Let $\mathrm{e}(z) = \exp(2\pi\sqrt{-1} \cdot z)$. If $w = w_1 \cdots w_n \in \{0,1\}^n$, then for any odd integer g define $\langle w \rangle_g = \prod_{j=1}^{n} \mathrm{e}(g \cdot w_j / 2^{j+1})$.

Lemma 5. *If c_w is rational for each $w \in \{0,1\}^n$, and $c_w \neq 0$ for at least one w, then*

$$\sum_{w \in \{0,1\}^n} c_w \cdot \langle w \rangle_g \neq 0$$

Proof. It is classical [9] that if α is a primitive complex m-th root of unity, then α cannot be a zero of any nonzero rational polynomial of degree below $\phi(m)$. It is clear that $\langle w \rangle_g$ can be written as

$$\langle w \rangle_g = \mathrm{e}(g \cdot D / 2^{n+1})$$

such that $0 \leq D \leq 2^n - 1 < \phi(2^{n+1}) = 2^n$. Since g is odd, $\mathrm{e}(g/2^{n+1})$ is a primitive $n+1$-st root of unity. By uniqueness of binary notation, and the odd parity of g, if $w \neq v$, then $\langle w \rangle_g \neq \langle v \rangle_g$. This means that the above sum can be regarded as the evaluation of a nonzero rational polynomial at $\mathrm{e}(g/2^{n+1})$ of degree below $\phi(2^{n+1})$, and the lemma follows.

We need a fact from Galois theory. Let $P(x) \in \mathbb{Z}[x]$, of degree, d, and let α be a primitive m-th root of unity where $d < \phi(m)$. The norm, N, of $P(x)$ is the sum of the absolute values of its coefficients.

Lemma 6. *If $S = \{1 \leq j < m \mid GCD(j,m) = 1\}$, then for every $\ell > 0$, for at least $\frac{1}{\ell+1}$ of the $j \in S$, $|P(\alpha^j)| \geq 1/N^{1/\ell}$.*

Proof Define

$$\prod = \prod_{j \in S} P(\alpha^j)$$

Recall from elementary Galois theory that

$$\prod \in \mathbb{Z} - \{0\}$$

Indeed this follows from three facts;

- $P(\alpha^j) \neq 0$
- $\prod \in \mathbb{Q}$ since it is fixed under all elements of the Galois group of the field generated by the α^j over $\mathbb{Q}$.
- The α^j are algebraic integers so, by the second item, $\prod$ is in the intersection of the algebraic integers and $\mathbb{Q}$, i.e., $\mathbb{Z}$.

Now if $|P(\alpha^j)| < 1/N^{1/\ell}$ for more than $\frac{1}{\ell+1}$ of the j, then since $|P(\alpha^j)| \leq N$ for any j, we would have $|\prod| < 1$, which is impossible. □

If $L \subseteq \{0,1\}^n$, then define $\langle L \rangle_g$ to be

$$\langle L \rangle = \sum_{w \in L} \langle w \rangle_g$$

The next lemma is the key technical result.

Lemma 7. *If $L_1, L_2 \subseteq \{0,1\}^n$, and $L_1 \neq L_2$, then for at least half of the odd g, $1 \leq g < 2^{n+1}$,*

$$|\langle L_1 \rangle_g - \langle L_2 \rangle_g| \geq 1/2^n$$

Proof. If $L_1 \neq L_2$, then $\langle L_1 \rangle_g - \langle L_2 \rangle_g$ can be regarded as some $P(\mathrm{e}(g/2^{n+1}))$, where $P(x)$ is a nonzero polynomial, and with nonzero coefficients in $\{1,-1\}$. In particular, positive terms will correspond to the words in $L_1 - L_2$, and negative terms to those in $L_2 - L_1$. The norm N of $P(x)$ is bounded above by 2^n. This follows from the fact that the number of terms in $\langle L_1 \rangle_g - \langle L_2 \rangle_g$ is the size of the symmetric difference of L_1 and L_2, which cannot exceed 2^n. The result now follows from Lemma 5 and Lemma 6.

4 Proof of Theorem 1

4.1 Basic error bounds

It will be convenient to work over the alphabet $\{0,1\}$. If UCFG G_1, G_2 have terminal alphabet A such that $2^{k-1} < \mathrm{card}(A) \leq 2^k$, then code the symbols of A by using length k binary strings. We will assume for the rest of the paper that the grammars G_1, G_2 to be tested for equivalence have been modified to reflect this coding, which will not materially affect the grammar sizes so far as the complexity bounds are concerned.

Let G be a UCFG in CNF with terminal alphabet $\{0,1\}$. Fix an odd integer g with $1 \leq g < 2^{n+1}$. Define β_i to be the sum of the first $3n$ terms of the Taylor series of $\mathrm{e}((g \bmod 2^{i+1})/2^{i+1})$. Note that

$$\mathrm{e}((g \bmod 2^{i+1})/2^{i+1}) = \mathrm{e}((g/2^{i+1})$$

However, $0 \leq \frac{g \mathrm{mod} 2^{i+1}}{2^{i+1}} < 1$, so that we get rapid convergence for the Taylor series.

We are going to associate a system of equations over the complex numbers $[G^n]_g$ with the CFG G^n. The set of variables is $\{[(x,i,j)]_g \mid (x,i,j) \in P^n\}$. Next we specify the equations of the system.

- $[(x,i,i)]_g = b_0 + b_1 \cdot \beta_i$ such that $b_0, b_1 \in \{0,1\}$ and $b_0 = 1$ iff $x \to 0 \in P$, and $b_1 = 1$ iff $x \to 1 \in P$.
- If $i < j$, then $[(x,i,j)]_g = \sum [(y,i,k)]_g \cdot [(z,k+1,j)]_g$ where the sum is over all pairs y, z such that $x \to yz \in P$, and over all $i \leq k < j$.

The idea behind this system of equations is to generate for each $w = w_1 \cdots w_n \in L(G^n)$ the number $[w]_g = \prod_{i=1}^{n} [w_i]_g$ where $[w_i]_g = 1$ if $w_i = 0$, and $[w_i]_g = \beta_i$ if $w_i = 1$. Notice the similarity to the bracket notation of Section 3, with the only difference being the appearance of the approximation β_i. We make this idea precise with the next lemma.

Lemma 8. *$[G^n]_g$ has a unique solution namely $\{[(x,i,j)]_g \mid (x,i,j) \in P^n\}$ and*

$$|[(x_0,1,n)]_g - \langle L(G^n)\rangle_g| < 1/2^{n+3}$$

Proof. Existence and uniqueness of the solution follow directly from Lemma 4. In particular, $[(x_0,1,n)]_g$ is obtained from the formal series f_0^n for $(x_0,1,n)$ by regarding the terminal alphabet A as $\{c_1,\ldots,c_n,d\}$, instantiating these symbols as $c_i = \beta_i$, and $d = 1$ (d stands in for occurrences of the symbol 0 at any position), then evaluating f_0^n.

The error of approximation $\delta = |\mathrm{e}(g/2^{i+1}) - \beta_i|$ is certainly bounded above by $1/2^{3n}$. If we let $[w]_g$ designate the value of any $w \in \{0,1\}^n$ after instantiating all occurrences of 0 by 1 and the occurrence of 1 in the i-th position by β_i, and letting $\delta_w = |[w]_g - \langle w\rangle_g|$, we get $\delta_w < 2n\delta$. This is obtained by noting since $|\mathrm{e}(z)| = 1$ for real z, that

$$|(\mathrm{e}(z)+\delta)(\mathrm{e}(z')+\delta) - \mathrm{e}(z+z')| < 2\delta + \delta^2 < 3\delta$$

and then iterating this fact $\log n$ times. We conclude from this and the triangle inequality applied at most 2^n times (upper bound on card$(L(G^n))$) that

$$|\langle L(G^n)\rangle_g - [(x_0,1,n)]_g| \leq 2^n \cdot \delta_w < n \cdot 2^{n+1} \cdot \delta < 1/2^{n+3}$$

Remark The righmost bound in the last inequality of the proof of Lemma 8 is not intended to be sharp, but it is enough to prove the main result of this subsection.

Lemma 9. *If G_1 and G_2 are UCFG in CNF with start symbols $x_{1,0}$ and $x_{2,0}$, respectively, then*

- *If $L(G_1) \cap \{0,1\}^n = L(G_2) \cap \{0,1\}^n$, then for any g,*
$$|[(x_{1,0},1,n)]_g - [(x_{2,0},1,n)]_g| < 1/2^{n+2}$$
- *If $L(G_1) \cap \{0,1,\}^n \neq L(G_2) \cap \{0,1\}^n$, then for at least half the odd g, $1 \leq g < 2^{n+1}$*
$$|[(x_{1,0},1,n)]_g - [(x_{2,0},1,n)]_g| > 1/2^{n+1}$$

Proof. The first claim follows from Lemma 8 and the triangle inequality. The second claim follows from Lemma 8, the triangle inequality and Lemma 7.

4.2 Complexity

We now describe an algorithm which produces the solution to the equation system $[G^n]_g$.

Lemma 10. *The solution to $[G^n]_g$ can be computed in NC4 in terms of the problem size which is the product of n and the grammar size.*

Proof. It is straightforward that the system $[G^n]_g$ can be constructed in NC4 (in fact certainly NC2 suffices) from G and the binary notation for n. This includes computing the complex rational approximations β_i. If s is the size of G, then $[G^n]_g$ has at most $s \cdot n^2$ equations. The solution will be obtained in $\log n$ stages, with each stage involving the solution of a system of linear equations. Each succeeding system is smaller than its predecessor.

The *rank* of the variable $[(x,i,j)]_g$ is defined to be $j - i$. Let ℓ be the least integer such that $2^\ell \leq n < 2^{\ell+1}$. For $1 \leq h \leq \ell + 1$ define R_h to be the set of all variables of rank at least $\lfloor n/2^h \rfloor$, but less than $\lfloor n/2^{h-1} \rfloor$. Note that $R_{\ell+1}$ contains the variables $[(x,i,i)]_g$ of rank 0. Next, define S_h to be the set of all equations whose left hand side variable is in R_h. Observe that if $h \leq \ell$, then each equation in S_h has at least one variable as a right hand side factor whose rank is less than $n/2^h$.

The solution procedure starts with $H_{\ell+1}$. Each left side variable of an equation in $H_{\ell+1}$ is replaced by the complex rational value on the right side. Next, H_ℓ is inspected and for each equation, the right side variables of rank 0 are replaced by the values just determined. By the observation above, at least one of the two right side variables must have rank 0. This modified H_ℓ can now be solved as an ordinary linear system.

The situation after $H_{\ell+1}, \ldots, H_m$ have been solved is that all variables of rank less than $\lfloor n/2^{m-1} \rfloor$ have received complex rational values. This collection of variables is precisely $R_{\ell+1} \cup \cdots \cup R_m$. By the observation each right side of an equation in H_{m-1} involves a variable in $R_{\ell+1} \cup \cdots \cup R_m$, and so H_{m-1} can be solved as a linear system. The process terminates on solving H_1, where in particular $[(x_0, 1, n)]_g$ will receive a value.

The main cost of this procedure is in solving $\ell + 1$ linear systems, each of size bounded above by $s \cdot n^2$. Since $\ell = O(\log n)$, we get an NC3 bound [1]. We have ignored the numerical computations which involve $\log n$ iterated matrix multiplications. Since the matrices are at most order $s \cdot n^2 \times s \cdot n^2$, and the initial numerical data requires $n^{O(1)}$ bits (the truncated Taylor series), it is straightforward that the intermediate numbers are also $n^{O(1)}$ bits size. The overall NC4 bound then follows from this and the well known fact e.g. [13] that addition and multiplication are in NC1.

4.3 Completion of the proof

The correctness of the algorithm follows from Lemma 9 and its complexity follows from Lemma 10.

References

1. S. Berkowitz. On computing the determinant in small parallel time using a small number of processors. *Inf. Proc. Lett.*, 18:147–150, 1984.
2. A. Bertoni, M. Goldwurm, and P. Massazza. Counting problems and algebraic formal power series in noncommuting variables. *Inf. Proc. Lett.*, 34:117–121, 1990.
3. Michael R. Garey and David S. Johnson. *Computers and Intractability.* W.H.Freeman and Company, New York, 1979.
4. J. Hopcroft and J. Ullman. *Introduction to Automata Theory,Languages and Computation.* Addison-Wesley, 1979.
5. D. Huynh. The complexity of ranking. In *3rd Structure in Complexity Theory Conf.*, pages 204–212, 1988.
6. D. Huynh. The complexity of deciding code and monoid properties for regular sets. Technical report, Computer science, U. Texas-Dallas, 1990.
7. D.T. Huynh. The complexity of ranking simple languages. *Mathematical Systems Theory*, 23:1–19, 1990.
8. W. Kuich and A. Salomaa. *Semirings,Automata,Languages.* Springer-Verlag, 1986.
9. S. Lang. *Algebra.* Addison-Wesley, 1965.
10. B. Litow. Parallel complexity of the regular code problem. *Inf. and Comp.*, 86,1:107–114, 1990.

11. B. Litow. Numbering unambiguous context-free languages. In *17th Australian Computer Society Conf.*, pages 373–378. University of Canterbury, New Zealand, 1994.
12. K. Mulmuley, U. Vazirani, and V. Vazirani. Matching is as easy as matrix inversion. In *19th Symp. on Theory of Computing (STOC)*, pages 345–354. ACM, 1987.
13. I. Wegener. *The Complexity of Boolean Functions.* Wiley-Teubner, 1987.