ON THE RELATIVE COMPLEXITY OF SOME LANGUAGES IN NC^1

David Mix Barrington, James Corbett

Computer and Information Science Department
University of Massachusetts

COINS Technical Report 89-22

On the relative complexity of some languages in NC^1

David Mix Barrington[1]
James Corbett[2]
COINS Dept., U. of Massachusetts
Amherst, MA 01003, U.S.A.
March 10, 1989

1. Abstract

We consider the relative complexity of a number of languages known to be in uniform NC^1, using the descriptive framework of Barrington, Immerman, and Straubing [4]. In particular we sharpen several results of Ibarra, Jiang, and Ravikumar [9]. We show that the one-sided Dyck languages, structured CFL's, and bracketed CFL's are recognizable by very uniform families of threshold circuits (are in DLOGTIME-uniform TC^0). We show that a large class of deterministic linear CFL's are in uniform TC^0, but that some are complete for uniform NC^1(and thus not in uniform TC^0 unless TC^0=NC^1 in the uniform setting).

2. Introduction

The parallel complexity class (non-uniform) NC^1 consists of Boolean functions (mappings from $\{0,1\}^*$ to $\{0,1\}$) which are computable by a fan-in two circuit family of $O(\log\ n)$ depth where n is the number of input bits. In other words, there is some constant c such that for each n there exists a circuit C_n, of depth at most $c\log\ n$, computing f on inputs of length n.

Various uniformity conditions can be placed on NC^1. We will use the DLOGTIME-uniformity examined by Barrington, Immerman, and Straubing [4], which requires that a deterministic Turing machine can answer simple questions about the n^{th} circuit in the family in $O(\log\ n)$ time. This type of uniformity turns out to be

[1]Former name David A. Barrington. Supported by NSF Computer and Computation Theory grant CCR-8714714.
[2]Supported by NSF grant CCR-8812567.

very robust, yielding equivalent definitions of several uniform subclasses of NC^1 under a variety of computational models including circuits and first order formulas. For NC^1, DLOGTIME-uniformity is equivalent to Ruzzo's U_{E^*}-uniformity [13] which has the consequence [13] that uniform-NC^1 equals ALOGTIME (languages recognized by an alternating Turing machine with a random access input tape in $O(\log n)$ time).

Ibarra, Jiang, and Ravikumar showed a number of problems to be in ALOGTIME or uniform-NC^1 by exhibiting alternating log time Turing machines. These include the one-sided Dyck language over two letters, structured CFL's, bracketed CFL's, and deterministic linear CFL's [9]. We will make stronger statements about the relative complexity of these languages using the structure theory of NC^1 developed in [2,4,5].

Contained in (non-uniform) NC^1 are three interesting subclasses. The class (non-uniform) AC^0 consists of Boolean functions computable by circuit families of polynomial size, constant depth, and unbounded fan-in. It was shown that counting modulo a constant cannot be done in AC^0[1,7], which motivated Barrington [2] to define the class (non-uniform) ACC. This is defined like AC^0, except that in addition to AND, OR, and NOT gates, we allow MOD-m gates (for some fixed m) which output 1 if and only if the number of inputs that are 1 is congruent to zero mod m. It was also shown that majority cannot be done in AC^0 with MOD-p gates where p is prime [11,14] (note this is not all of ACC). Hence the class (non-uniform) TC^0 [8,10] was defined like AC^0 but allowing MAJORITY gates which output a 1 if and only if a majority (more than half) of their inputs are 1. This is equivalent to allowing arbitrary threshold gates (which output 1 if and only if at least some number of their inputs are 1). Since modulo counting gates can be built from threshold gates and since majority is in NC^1, we have the sequence of subclasses $AC^0 \subset ACC \subseteq TC^0 \subseteq NC^1$, where the last two inclusions are not known to be strict. The DLOGTIME-uniform versions of these classes have equivalent characterizations in terms of constant depth circuits with special gates, expressions with special operators, and first order formulas with special quantifiers, as shown in [4].

We say one problem A is DLOGTIME-AC^0 reducible to another problem B if and only if there is a DLOGTIME-uniform AC^0 circuit, with oracles for B, solving A. A language is said to be NC^1-complete if and only if it is in uniform-NC^1 and all other languages in uniform-NC^1 are DLOGTIME-AC^0 reducible to it. Both the Boolean formula value problem [6] and the word problem for nonsolvable groups [2] are known to be NC^1-complete. The existence of these complete languages suggests that $TC^0 \neq NC^1$ for the following reason. The constant depth circuits of TC^0 are naturally parameterized by depth: TC^0_k is defined as the class of languages recognized by threshold circuits of depth at most k. Then $TC^0_1 \subset TC^0_2 \subset \ldots \subset TC^0_k \ldots$ and $TC^0 = \cup_{k=1}^{\infty} TC^0_k$. It is known that the first three levels of this hierarchy are distinct [8] and that the analogous hierarchy for monotone threshold circuits does not collapse [16]. If $TC^0 = NC^1$, then the two complete languages would be in TC^0_k for some k, thus collapsing the hierarchy above that arbitrary depth [3].

In section 3 we show that the one-sided Dyck languages are in DLOGTIME-uniform TC^0. In section 4 we show that structured and bracketed CFL's are both in DLOGTIME-uniform TC^0. In section 5 we examine which deterministic linear CFL's are in DLOGTIME-uniform TC^0. We conclude with some open problems.

3. One-sided Dyck Languages in TC^0

The one-sided Dyck language on two letters, which we shall denote D_2, is the balanced parenthesis language with two types of parentheses defined by the following grammar.

$$\begin{aligned} S &\rightarrow SS \\ S &\rightarrow [_1\, S\,]_1 \\ S &\rightarrow [_2\, S\,]_2 \\ S &\rightarrow \epsilon \end{aligned}$$

D_2 can be parsed using the well known level trick [12]. Assign a level to each parenthesis, starting with one and ignoring the differences in parenthesis type. The level of a parenthesis equals the number of opening parentheses to its left (including it) minus the number of closing parentheses strictly to its left (not including it). A right parenthesis is said to match a left parenthesis if it is the closest parenthesis to the right on the same level. A string is in D_2 if and only if all parentheses have a positive level and each left parenthesis has a matching right parenthesis of the same type (this is easily proved by induction on the length of the strings).

The only part of this which requires nonconstant depth is determining the level of each parenthesis (which requires counting). However, arbitrary threshold gates can count in constant depth, hence the problem is in (non-uniform) TC^0. To show uniformity, we will express membership in D_2 with a first order formula [4] with majority quantifiers.

In our formulas, variables range over positions in the input string (from 1 to n), which is read with the predicate $\pi_a(i) \equiv$ "the i^{th} input is a". Variables can also be compared. For example, the regular language 0^*1^+ can be expressed as

$$\exists x \forall y[((y < x) \rightarrow \pi_0(y)) \wedge ((y \geq x) \rightarrow \pi_1(y))]$$

Majority quantifiers are a generalization of existential and universal quantifiers defined as follows: $(Mx)\psi(x)$ is true if and only if $\psi(x)$ is true for more than half of the possible x's. It is easy to see how such an expression can be converted into a TC^0 circuit by replacing universal, existential, and majority quantifiers with AND, OR, and MAJORITY gates respectively. The resulting family of circuits will be DLOGTIME-uniform by the results of [4].

Theorem 1 *D_2 is in DLOGTIME-uniform TC^0.*

Proof: In [4] it is shown that we can express $y = \sharp x : \phi(x)$ (i.e. y is the exact number of x's such that $\phi(x)$ is true), in first order logic with majority quantifiers. Hence we can express $LEVEL(i,l)$ (i.e. parenthesis i is at level l), $MATCH(i,j)$ (i.e. parenthesis i matches parenthesis j), and $D2$ (i.e. the input is in D_2), as follows:

$$\begin{aligned}
OPEN(i) &\equiv \pi_{[_1}(i) \vee \pi_{[_2}(i) \\
CLOSE(i) &\equiv \pi_{]_1}(i) \vee \pi_{]_2}(i) \\
LEVEL(i,l) &\equiv \exists y(y = \sharp x : x \leq i \wedge OPEN(x)) \wedge \\
&\quad\ \exists z(z = \sharp x : x < i \wedge CLOSE(x)) \wedge \\
&\quad\ y \geq z \wedge l = y - z \\
MATCH(i,j) &\equiv i < j \wedge OPEN(i) \wedge CLOSE(j) \wedge \\
&\quad\ \exists l(LEVEL(i,l) \wedge LEVEL(j,l) \wedge \forall k(i < k < j \rightarrow \neg LEVEL(k,l))) \\
D2 &\equiv \forall i \exists l\ LEVEL(i,l) \wedge \\
&\quad\ \forall i(OPEN(i) \rightarrow \exists j(MATCH(i,j) \wedge (\pi_{[_1}(i) \equiv \pi_{]_1}(j))))
\end{aligned}$$

□

In general, a one-sided Dyck language may have any fixed number k of parenthesis types. Such a language is denoted D_k.

Corollary 1 *D_k is in DLOGTIME-uniform TC^0 for all $k \geq 1$.*

Proof: We can easily adapt the above predicates to any fixed number of parenthesis types. □

4. Structured and Bracketed CFL's in TC^0

Ibarra, Jiang, and Ravikumar proved that two subclasses of CFL's are in NC^1 [9]. Their techniques suffice to prove the sharper result that these subclasses are actually in TC^0. As defined in [9], a structured context-free grammar (CFG) $G' = (\Sigma', V, P', S)$ is induced by an arbitrary CFG $G = (\Sigma, V, P, S)$ as follows:

$$\begin{aligned} \Sigma' &= \Sigma \cup \{[_A,]_A | A \in V\} \\ P' &= \{A \rightarrow [_A\alpha]_A | A \rightarrow \alpha \in P\} \end{aligned}$$

Structured CFL's are those generated by structured CFG's. A bracketed CFG is similar but the productions are numbered arbitrarily and the parentheses are labeled with the number of the production rather than the nonterminal. We denote the substring of the input from position i to position j by $w_{i:j}$.

Theorem 2 *Structured CFL's and bracketed CFL's are in DLOGTIME-uniform* TC^0.

Proof: We adapt the basic proof method of [9] and show that it yields a uniform-TC^0 solution. Let G be a structured CFG. Then we can check membership in $L(G)$ as follows. First make sure the parentheses are balanced correctly just as we did in the proof of Theorem 1 (i.e. all parentheses have positive level, each left parenthesis has a matching right parenthesis of the same type). Then verify that what is contained in each pair of matched parentheses could have been produced by the nonterminal which labels the parentheses. For example, if we see

$$[_A x_0 [_{A_1} \alpha_1]_{A_1} x_1 \ldots [_{A_k} \alpha_k]_{A_k} x_k]_A$$

with $x_i \in \Sigma^*, \alpha_i \in (\Sigma')^*$, then there must be a production

$$A \rightarrow [_A x_0 A_1 x_1 \ldots A_k x_k]_A$$

which produced it. The OPEN, CLOSE, LEVEL, and D2 predicates defined above are easily modified to check the first condition. The second condition can be checked by constructing predicates $FROM_k(i, j)$ for each k which are true if and only if $w_{i:j}$ could have been produced by rule k (number the rules arbitrarily). By OR-ing a fixed number of these we can easily construct predicates $FROM_A(i, j)$ which are true if and only if $w_{i:j}$ could have come from nonterminal A. Below is an example for a CFG in which

$$\begin{aligned} (1)\ A &\rightarrow [_A aaaBbcCcdDddd]_A \\ (2)\ A &\rightarrow [_A bcd]_A \end{aligned}$$

are the only productions for A. We define the predicate $\Pi_{a_0 \ldots a_m}(i)$ as a generalization of the $\pi_a(i)$ in the natural way: it tests for the presence of a string $a_0 \ldots a_m$ starting at position i in the input.

$$\begin{aligned}
\Pi_{a_0 \ldots a_m}(i) &\equiv i + m \leq n \wedge \pi_{a_0}(i) \wedge \pi_{a_1}(i+1) \wedge \ldots \pi_{a_m}(i+m) \\
FROM_1(i,j) &\equiv \pi_{[_A}(i) \wedge \Pi_{aaa}(i+1) \wedge \pi_{[_B}(i+4) \wedge \\
&\quad \exists x (MATCH(i+4,x) \wedge \pi_{]_B}(x) \wedge \Pi_{bc}(x+1) \wedge \pi_{[_C}(x+3) \wedge \\
&\quad \exists y (MATCH(x+3,y) \wedge \pi_{]_C}(y) \wedge \Pi_{cd}(y+1) \wedge \pi_{[_D}(y+3) \\
&\quad \wedge MATCH(y+3,j-4))) \wedge \pi_{]_D}(j-4) \wedge \Pi_{ddd}(j-3) \wedge \pi_{]_A}(j) \\
FROM_2(i,j) &\equiv \pi_{[_A}(i) \wedge \Pi_{bcd}(i+1) \wedge i+4 = j \wedge \pi_{]_A}(j) \\
FROM_A(i,j) &\equiv FROM_1(i,j) \vee FROM_2(i,j)
\end{aligned}$$

The proof for bracketed CFL's (where the parentheses are labeled with the production number) is similar. The $FROM_A(i,j)$ predicates are not necessary. Instead the $FROM_k(i,j)$ predicates must test disjunctively for all the parenthesis types which could produce a given nonterminal. For example, given the following bracketed CFG

$$\begin{aligned}
(1)\ A &\rightarrow [_1 a]_1 \\
(2)\ A &\rightarrow [_2 aa]_2 \\
(3)\ B &\rightarrow [_3 bA]_3
\end{aligned}$$

the predicate $FROM_3(i,j)$ would be expressed as

$$\begin{aligned}
FROM_3(i,j) \equiv\ & (\pi_{[_3}(i) \wedge \pi_b(i+1) \wedge \pi_{[_1}(i+2) \wedge \pi_{]_1}(j-1) \wedge \\
& MATCH(i+2,j-1) \wedge \pi_{]_3}(j)) \vee \\
& (\pi_{[_3}(i) \wedge \pi_b(i+1) \wedge \pi_{[_2}(i+2) \wedge \pi_{]_2}(j-1) \wedge \\
& MATCH(i+2,j-1) \wedge \pi_{]_3}(j))
\end{aligned}$$

□

5. Deterministic Linear CFL's

Ibarra, Jiang, and Ravikumar also showed that deterministic linear CFL's are in NC^1. A deterministic linear CFL is one generated by a context-free grammar in which (1) at most one nonterminal appears in the right of any production, and (2) if any two of the rules $A \rightarrow x_1 B_1 y_1$, $A \rightarrow x_2 B_2 y_2$, $A \rightarrow x_3$ occur in the grammar, then x_i is not a prefix of x_j for $i \neq j$ [9].

Their construction contains a number of steps, all of which are clearly in TC^0 except one: the recognition of a regular "prefix language". The prefix language of a nonterminal A, denoted L_A, is defined by

$$L_A = \{x \mid S \overset{*}{\Rightarrow} xAy \text{ for some } y \in \Sigma^*\}$$

This is easily seen to be a regular language, though its syntactic monoid could be nonsolvable (in which case its recognition would be NC^1-complete [2] and thus probably not in TC^0). Since the word problem for solvable monoids is in TC^0 [5], if all the prefix languages are recognized by solvable monoids, then the deterministic linear CFL is in TC^0 by the proof in [9].

It would be nice to show that if the prefix language for any nonterminal were nonsolvable then the deterministic linear CFL was NC^1-hard, but this turns out to be false. One can easily construct a context-free grammar which generates the trivial language Σ^* all of whose prefix languages are nonsolvable. Take any nonsolvable group G and create a terminal and a nonterminal for each element of G. Then create productions $A \rightarrow bC$ for each $a, b, c \in G$ such that $c = a * b$, and finally a production $A \rightarrow \epsilon$ for each nonterminal. Let the start symbol be the nonterminal corresponding to the identity of the group. Then the prefix language L_A is $\{g_1 g_2 \ldots g_m \mid g_1 * g_2 * \cdots * g_n = a\}$, which has syntactic monoid G. We do note that there are some special deterministic linear CFL's which are NC^1-complete since this class includes all regular languages.

6. Conclusion and Open Problems

A number of fundamental questions in this area remain open. Which CFL's are in NC^1? It is unlikely that $CFL \subset NC^1$ since this would imply NSPACE($\log n$) = DSPACE($\log n$) by Sudborough's result [15]. Which CFL's are NC^1-hard? Which parentheses languages are NC^1-hard (Buss [6] showed all of them to be in NC^1)? Examples, such as the Boolean formula value problem, are known, but there is yet no characterization of what makes them hard.

References

[1] M. Ajtai, "Σ_1^1 formulae on finite structures," *Annals of Pure and Applied Logic* **24** (1983), 1-48.

[2] D. A. Barrington, "Bounded-width polynomial-size branching programs recognize exactly those languages in NC^1," *J. Comp. Syst. Sci.* **38:1** (1989), 150-164.

[3] P. Berman, personal communication.

[4] D. A. M. Barrington, N. Immerman, and H. Straubing, "On uniformity within NC^1," *Structure in Complexity Theory: Third Annual Conference*(Washington: IEEE Computer Society Press, 1988), 47-59.

[5] D. A. M. Barrington and D. Thérien, "Finite Monoids and the fine structure of NC^1," *J. ACM* **35:4** (1988), 941-952.

[6] S. R. Buss, 'The Boolean formula value problem is in ALOGTIME," *18th ACM STOC Symp.* (1987), 123-131.

[7] M. Furst, J. B. Saxe, and M. Sipser, "Parity, circuits, and the polynomial-time hierarchy," *Math. Syst. Theory* **17** (1984), 13-27.

[8] A. Hajnal, W. Maass, P. Pudlák, M. Szegedy, and G. Turán, "Threshold circuits of bounded depth," *28th IEEE FOCS Symp.* (1987), 99-110.

[9] O. H. Ibarra, T. Jiang, and B. Ravikumar, "Some subclasses of context-free languages in NC^1," *Information Processing Letters* **29** (1988), 111-117.

[10] I. Parberry and G. Schnitger, "Parallel computation with threshold functions," in *Structure in Complexity Theory*, Lecture Notes in Computer Science No. 223 (New York, Springer Verlag, 1986), 272-290.

[11] A. A. Razborov, "Lower bounds for the size of circuits of bounded depth with basis $\{\&, \oplus\}$," *Mathematicheskie Zametki* **41:4** (April 1987), 598-607 (in Russian). English translation *Math. Notes Acad. Sci. USSR* **41:4** (Sept. 1987), 333-338.

[12] R. W. Ritchie and F. N. Springsteel, "Language recognition by marking automata," *Inform. and Control* **20** (1972), 313-330.

[13] W. L. Ruzzo, "On uniform circuit complexity," *J. Comp. Syst. Sci.* **21:2** (1981), 365-383.

[14] R. Smolensky, "Algebraic methods in the theory of lower bounds for Boolean circuit complexity," *19th ACM STOC Symp.* (1987), 77-82.

[15] I. H. Sudborough, "On the tape complexity of deterministic context-free languages," *J. ACM* **25:3** (1978), 405-414.

[16] A. Yao, indirect personal communication.