Research report 75

ON THE COMPLEXITY OF PARALLEL PARSING OF GENERAL CONTEXT-FREE LANGUAGES

by

Wojciech Rytter

(RR75)

Abstract

Let T(n) be the time to recognize context-free languages on a parallel random access machine without write conflicts (P-RAM) using a polynomial number of processors. We assume that $T(n) = \Omega(\log n)$. Let P(n) be the time to compute a representation of a parsing tree for strings of length n using a polynomial number of processors. Then we prove: $P(n) = O(T(n))$. A related result is a parallel time log n computation of the transitive closure of directed graphs having special structure.

Department of Computer Science
University of Warwick
Coventry, CV4 7AL, England

April 1986

ON THE COMPLEXITY OF PARALLEL PARSING OF GENERAL CONTEXT-FREE LANGUAGES

Wojciech Rytter,
University of Warwick ,Department of Comp.Science
Coventry CV4 7AL
and
Warsaw University, Institute of Informatics

Abstract

Let T(n) be the time to recognize context-free languages on a parallel random access machine without write conflicts (P-RAM) using a polynomial number of processors. We assume that $T(n) = \Omega(\log n)$.
Let P(n) be the time to compute a representation of a parsing tree for strings of length n using a polynomial number of processors. Then we prove: $P(n) = O(T(n))$.
A related result is a parallel time log n computation of the transitive closure of directed graphs having special structure.

The problem of parsing for context-free languages (cfl's,shortly) seems to be harder than the problem of recognition. It was proved by Ruzzo [1] that if T'(n) is the time to recognize cfl's on a RAM (sequentially) then the time needed to parse cfl's on a RAM is O(T'(n)log n)). We show that when one considers parallel time then parsing is not harder than recognition (however the number of processors can grow considerably, though polynomially).

Our model of parallel computation is a parallel random access machine without write conflicts (known also as a CREW P-RAM). Such a machine consists of a number of synchronously working processors (RAM's) which are using a common memory . No two processors can attempt to write in the same step into the same location, however many processors can read from the same location. Such a model corresponds to bounded fan-in circuits.

The best algorithms for parallel general context-free recognition on a P-RAM work in $\log^2(n)$ time using $O(n^6)$ processors, see [2,3] (such complexity can even be achieved on much weaker models of parallel computations, cube connected computers and perfect shuffle computers, see [4]). It is a hard open problem whether general context-free recognition can be done in parallel time log(n) on a P-RAM. For unambiguous languages log n time is enough,see [5], and if the language is a bracket cfl then the number of processors can be linear,see [6]. Optimal (log n time and n/log(n) processors) parallel parsing and recognition algorithms can be constructed for one-sided Dyck languages.

The algorithms for general context-free recognition use the parsing matrix (to be defined later). We show that using this matrix a parsing tree can be constructed in log (n) time using a cubic number of processors. Even if the recognition does not construct the parsing matrix then we can construct it by executing simultaneously $O(n^2)$ parallel recognizing algorithms. The parallel time does not increase, however the number of processors should be multiplied by n^2 in this case.

Throughout the paper we assume (for ease of exposition) that the grammars are in Chomsky normal form. Let G be a context-free grammar in Chomsky normal form, let N , Ter, denote the set of nonterminal, terminal symbols, respectively. We write A - >B if the grammar has such a production, and A->*w iff the string w can be derived from A. Let S be the

starting symbol of the grammar.

Let w=a[1]a[2]....a[n] be a given input string of length n. The recognition problem is: decide whether A->*w, where A ε N

The parsing problem is: construct a parsing tree PT.

This tree has 2n-1 nodes numbered 1..2n-1. With each node x there is associated the following information:

Father[x], Left[x],Right[x], (left and right sons, if x is not a leaf) and Label[x] (an element of N).

The tree should satisfy locally the rules of the grammar:

Label[root]=S;

Label[x]->Label[Left[x]]Label[Right[x]] if x is not a leaf;

Label[x]->a[i] if x is the i-th leaf, (from the left).

Father[Left[x]]=Father[Right[x]]=x.

It is enough to compute only the tables Father and Label. Left and Right can be computed then easily on a P-RAM in log(n) time using a small number of processors. On the other hand, if we have Left and Right then this does not determine Father, since PT might be any directed acyclic graph whose nonleaf nodes have outdegree 2.

The parsing table Tab is of the type array[0..n,0..n] of subsets of N,

Tab[i,j]=if i<j then { A : A->*a[i+1]...a[j]} else ø.

Example.

Let G be the following grammar:

S->CS S->AS S->CA S->DD S->AC

C->AA C->BB

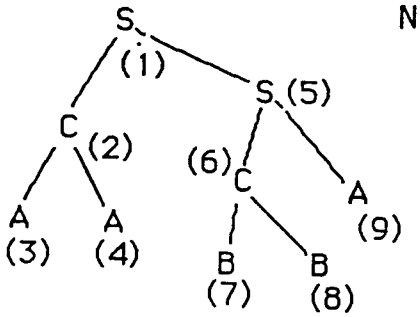
D->AA D->DC

A->a B->b.

N={S,C,D,A}, Ter={a,b}.

Let w=aabba.

The parsing table is presented in Fig.2, and the parsing tree in Fig.1.



Node	Label	Father	Left	Right
1	S	-	2	5
2	C	1	3	4
3	A	2	-	-
4	A	2	-	-
5	S	1	6	9
6	C	5	7	8
7	B	6	-	-
8	B	6	-	-
9	A	5	-	-

Fig.1.Parsing tree and its representation

	0	1	2	3	4	5
0		A	C D		D S	S
1			A		S	S
2				B	C	S
3					B	
4						A
5						

Fig.2. The parsing table for G and w=aabba.

Let G be a directed acyclic graph given by the relation R, where R(u,v) holds whenever (u,v) is an edge of G. We say that G satisfies the unique path condition (UPC, shortly) iff for every two nodes v, u there is at most -

one path from u to v. (Equivalently, one can say that G is weakly acyclic,after removing orientation the undirected graph is acyclic). The best known upper bound to compute the transitive closure R^* of directed graphs is $\log^2 n$, however if the graph satisfies UPC then it can be improved.

Lemma.

If the directed graph G with m nodes satisfies UPC then the transitive closure of G can be computed in log(m) parallel time on a P-RAM using m^3 processors.

Proof.

Let R be the relation corresponding to a directed acyclic graph G satisfying UPC, and V be the set of nodes. Assume that the nodes are numbered 1..m. We say that a node is a sink iff it has outdegree zero. Let s= log m . First we compute the tables $R_k[v]$ (corresponding to some relations), for $0 \le k \le s$.

R_0:=R;

for each sink v do in parallel $R_0[v,v]$:=true;

for k:=1 to s do

for each v1,v2,v3 such that $R_{k-1}[v1,v2]$ and $R_{k-1}[v2,v3]$

do in parallel $R_k[v1,v3]$:=true;

We claim that there are no write conflicts in the above algorithm and $R_s[v1,v2]$ holds iff there is a path from v1 to v2 and v2 is a sink.

The first fact follows from the following (easy to prove) invariant:

(*) if $R_k[v,v1]$ and $R_k[v,v2]$ and v, v1, v2 are lying on the same path in G then v1=v2, for k=0..s.

This invariant implies that whenever we have $R_{k-1}[v1,v2]$ and $R_{k-1}[v2,v3]$ and $R_{k-1}[v1,v2']$ and $R_{k-1}[v2',v3]$ then v2=v2', because UPC guarantees that all the nodes involved lie on the same path.

The second fact follows from our doubling technique. We are doubling the distances between v1, v2 for which $R_k[v1,v2]$ holds , until ultimately v2 becomes a sink. The following invariant can be easily proved:

(**) If $R_k[x,y]$ and y is not a sink then $dist(x,y)=2^k$

(dist is the length of the path from x to y in the graph G).

We have computed a part of R*, if y is a sink then $R^*(x,y)=R_S[x,y]$. Now we compute R* for all nonsink nodes.

We introduce two relations R'_k and D_k (k=0..s), represented by two-dimensional tables with the same names.

$R'_k[x,y]$ holds iff $R_k[x,y]$ holds and y is not a sink.

$D_k[x,y]$ holds iff $dist(x,y)<2^{k+1}$, for nonsink nodes x,y .

Let ID denote the identity relation and • denote the composition of relations, we consider only the nodes which are not sinks. The relations D_k can be computed using the following recurrence formula (following from invariant (**)):

$D_0= R+ID$; $D_{k+1}=D_k+D_k•R'_{k+1}$.

We can easily compute R'_0 and D_0 , next we apply the recurrence equation log(n) times.

```
for k:=1 to s do
begin
```

for each x,z do in parallel if $D_{k-1}[x,z]$ then $D_k[x,z]$:=true;

for each x,y,z such that $D_{k-1}[x,y]$ and $R'_k[y,z]$ do in parallel $D_k[x,z]$:=true

end

There are no write conflicts here because if x,y,z are lying on the same path and $R'_k[y,z]$ holds then y is uniquely determined by x, z (as a node lying on the path from x to z, whose distance to z is 2^k). Observe that in this algorithm D_k could be replaced by D (in fact the subscript k is not needed, though it helps to apply the recurrence formula directly).

Now we can compute $R^*= R_S+D_S$ in one parallel step. This completes the proof.

Theorem

Assume that context-free recognition can be done in the parallel time T(n) using R(n) processors of a P-RAM, and $T(n)=\Omega(\log n)$.

Then the representation of a parsing tree (if there is any) can be constructed in parallel time O(T(n)) using $O(R(n)\ n^2+n^3)$ processors.

If the recognition procedure constructs the parsing table then $O(R(n)+n^3)$ processors are enough.

Proof.

First we construct the parsing table Tab for a given input string w=a[1]a[2]...a[n], and a given grammar G in Chomsky normal form. This can be done in parallel time T(n) using n^2 R(n) processors. For each A,i<j, simultaneously we check whether A->*a[i+1]...a[j].

If S->*w then we start to compute a parsing tree else we stop here, because we know that in such case there is no parsing tree.

We construct now the following acyclic directed graph G represented by the relation R (the relation:to be a possible father). The set of nodes is $V= \{(A,i,j) : i<j, A \varepsilon Tab[i,j]\}$. We execute:

for each node (A,i,j), i<j, do in parallel

find i<k<j, B,C such that

there is a production A->BC and B ε Tab[i,k] and C ε Tab[k,j];

R[(B,i,k),(A,i,j)]:=R[(C,k,j),(A,i,j)]:=true;

((A,i,j) becomes a possible father of (B,i,k),(C,k,j), the number k and nonterminals B, C can be found using a linear number of processors for fixed i,j,these processors could be organized into the tree to search i-th row and j-th column (simultaneously) of Tab for suitable B,C. There can be many k's possible, but we choose any one of them and it is then fixed). The graph corresponding to our example grammar and the string aabba is shown in Fig.3. Observe that the tree from Fig.1 corresponds to a subgraph of this graph.

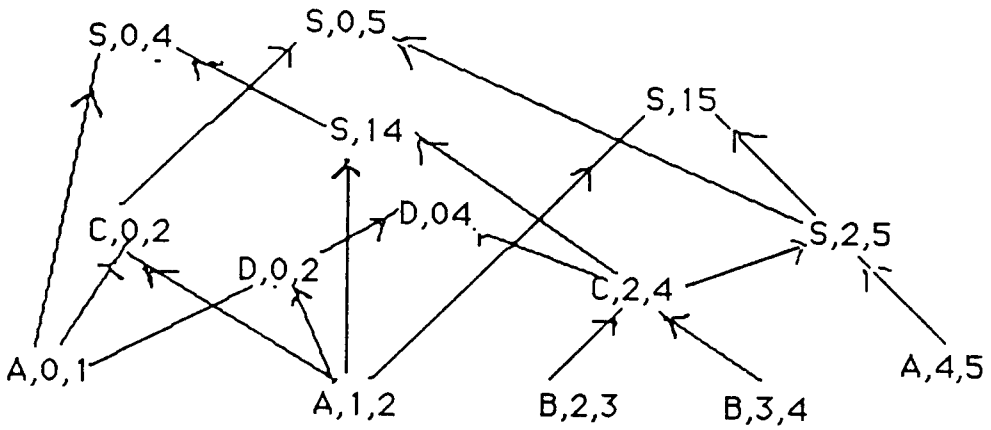


Fig.3. The graph G.

Next we compute the transitive closure R*. The graph G satisfies UPC and we can use algorithm from the lemma. Let v_0 =(S,0,n). The parsing tree PT consists of all nodes v such that R*(v,v_0) holds .The root of PT is v_0 . The

function Father is computed as follows:

for each u, v ε PT do in parallel

if R(u,v) then Father[u]:=v;

The tables Left and Right can be computed in parallel time log n using the table Father.So far the nodes are not numbered (from 1 to 2n-1) as required, each node is a triple of the form (A,i,j) ,and all the tables have entries indexed by such triples. The set of such triples belonging to PT can be numbered from 1 to 2n-1 using the algorithm of Tarjan and Vishkin [7] for preorder (or postorder) numbering of trees.This can be done also by arranging all possible triples in any initial order (e.g. lexicographically), and then the final number of a triple belonging to PT could be obtained by counting the number of preceding triples which are elements of PT (using a prefix computation). If num is the numbering obtained , then Label[num(A,i,j)]:=A.

The constructed tree now satisfies all the requirements. This completes the proof.

It was proved in [5] that every unambiguous cfl can be recognized in log(n) time on a P-RAM using a polynomial number of processors. Now we can strengthen the result of [5].

Corollary.

Every unambiguous cfl can parsed on a P-RAM in log n time using polynomial number of processors.

Acknowledgments.

The author thanks M.Paterson for his comments on this paper.

References.

[1] W. Ruzzo. On the complexity of general context free language parsing and recognition. Automata, languages and programming. Lect.Notes in Computer Science,1979

[2] W.Ruzzo. Tree-size bounded alternation. Journal of Comp. and Syst.Sciences 21, 218-235, 1980

[3] W.Rytter. The complexity of two way pushdown automata and recursive programs. NATO Advanced Research Workshop "Combinatorial algorithms on words" (ed.A.Apostolico,Z.Galil),Springer-Verlag 1985

[4] W.Rytter. On the recognition of context free languages. Computation Theory, Lect.Notes in Comp.Science,Springer Verlag 1985

[5] W.Rytter. Parallel time log n recognition of unambiguous cfl's. Fund.of Computation Theory, Lect.Notes in Computer Science 1985

[6] W.Rytter,R.Giancarlo. Parallel parsing of bracket and recognition of input driven languages. manuscript,1985

[7] R.Tarjan,U.Vishkin. Finding biconnected components and computing tree functions in logarithmic parallel time. Proceedings of IEEE Symp. on Found.of Comp.Science, 1984