(**Note**)

Context-free recognition via shortest paths computation: a version of Valiant's algorithm

Wojciech Rytter

Institute of Informatics, Warsaw University,

ul. Banacha 2, 02-097 Warszawa, Poland

e-mail: rytter@mimuw.edu.pl

Abstract.

We present a new algorithm recognizing general context-free languages in O(BM(n)) time, where BM(n) is the time to multiply two n by n boolean matrices. The only known algorithm for this problem with the same asymptotic complexity is the Valiant's algorithm [Va75], which is quite sofisticated. The probem related to texts is reduced to arithmetics of matrices whose elements are in a semiring of a constant size. The main difficulty in Valiant's algorthm is nonassociativity of considered semirings, the main point in our algorithm is that nonassociative semirings are replaced by associative ones. This simplifies the algorithm considerably. Our algorithm is more structured, its main part is a computation of shortest paths in a special graph called here the *lattice graph.* The Valiant's lemma, see [Ha78], is replaced here by a shortest paths lemma. The shortest paths problem for lattice graphs is also interested in its own.

1. Introduction

Our main result is a new version and a simplification of an important classical algorithm. The Valiant's algorithm for context-free recognition in less than cubic time is probably the most interesting algorithm related to formal languages. However it is too complicated, especially from the point of view of correctness, thus it deserves further study. We refer the reader to [Ha78] for the excellent exposition of Valiant's algorithm.

The recognition problem for context-free languages is:

for an input word $w = a_1a_2...a_n$ of length n we have to decide whether w can be generated by a given grammar G, in other words if $w \in L(G)$, where $L(G)$ is the context-free language generated by the grammar G. The size of the grammar is constant. The size of the whole problem is n.

The Valiant's algorithm is of a *divide and conquere* type. Our approach is similar. The recurrences for time complexities of our main and auxiliary algorithms are:

$$(*) \quad T(n) = 2\, T(n/2) + O(T1(n)),$$

$$(**) \quad T1(n) = 4\, T1(n/2) + O(BM(n)).$$

where *T1(n)* is the complexity to compute (defined later) the procedure *ShortestPaths*..

If *BM(n)* = $\Omega(n^{2+\varepsilon})$, then it is very easy to see that *T(n)* and *T1(n)* are O(*BM(n)*). In this case the recurrences (*) and (**) can be computed trivially using the general techniques for divide and conquere recurrences, see [AHU74]. For other functions *BM(n),* for example for *BM(n)* = n^2 *log(n)*, essentially the same analysis as in [Va75] can be done, we refer to [Va75] for details.

Our terminology is rather abstract since we use *semirings* but it allows to eliminate some obscuring details. We refer the reader to [AHU74] for the definition of the *semiring*. We consider the semiring of binary relations. The *syntactic semiring* used by Valiant corresponded to composition of nonterminals according to rules of the grammar. Nonassocitivity of this semiring is inherent and without associativity the weight of a path (whose edge weights are in the semiring) depends on the bracket structure of multiplication. However the bracket structure coresponds to a parsing tree, which is unknown. On the other hand associativity allows to compute weights of paths edge by edge.

Our algorithm uses new terminology but in fact it does not differ significantly from the known algorithms. The auxiliary algorithm is similar to the reduction of the transitive closure to boolean matrix multiplication, see [AHU74], and to the parallel algorithm for *grid graphs* in [Ry88]. The graph-theoretic approach to context-free recognition is also used in [BKR91]. The main algorithm is also similar to Valiant's algorithm The main difference is done by associativity.

It is convenient to assume that *BM(n)* = $\Omega(n^{2+\varepsilon})$, however our algorithm has in general the same asymptotic complexity with respect to Boolean matrix multiplication as the Valiant's algorithm.

2. Lattice graphs

The crucial role in our algorithm is played by special graphs. We define the n by n *lattice graph* of size n as the weighted directed acyclic graph $L= (V,E,s)$ whose set of nodes is $V = \{(i,j)\colon 1 \le i, j \le n\}$ and s is the *source node*. The set of edges is $E = E_H \cup E_V$, where

$E_H = \{(i,j)\to (i,j+k) : 1 \le i, j , j+k \le n, k > 0 \}$ and

$E_V= \{(i,j)\to (i-k,j) : 1 \le i, j , i - k \le n, k > 0 \}$.

The notation $x\to y$ means the directed edge from x to y. E_H ,E_V are the set of *horizontal* (left-to-right) edges and and the set of *vertical* (top-down) edges, respectively. The rows are numbered (top-down) in a decreasing order. (i,j) is the element in the i-th row and j-th column.

The weights of edges are elements of an associative semiring S with operations of summation '+' and multiplication $\otimes$.

Two edges $\pi1$, $\pi2$ are said to be *strongly congruent* iff they are both horizontal or both vertical and there is a vector π, such that the endpoints of $\pi1$ can be obtained by shifting $\pi2$ by the vector π. We require here that π is vertical if $\pi1$, $\pi2$ are horizontal and π is horizontal if $\pi1$, $\pi2$ are vertical.

In other words $\pi1$, $\pi2$ are both horizontal and start in the same column, or they are both vertical and start in the same row. Also strongly congruent edges have the same length (in horizontal or vertical direction). The length of (i,j) $\to$ (i+k,j+l) is the sum k+l.

Example

The vectors (4,7)$\to$ (4,15) and (4,3)$\to$ (4,12) are strongly congruent. They start in the same row (4th row) and have length 8. Similarly (3,5) $\to$ (10,5) (8,5) $\to$ (15,5) are strongly congruent.

We assume that the considered lattice graphs satisfy the following crucial condition

congruency condition : the weights of any two strongly congruent edges are the same.

The congruency condition implies that there exist matrices H and V such that the weights satisfy the following conditions for each i, j, k (see Figure 1) :

$$weight(\ (i,k)\rightarrow (i,j)\ ) = H(k, j);\quad weight(\ (k,j)\rightarrow (i,j)\ ) = V(k\ i).$$

H and V are called respectively the matrix of horizontal and the matrix of vertical weights.

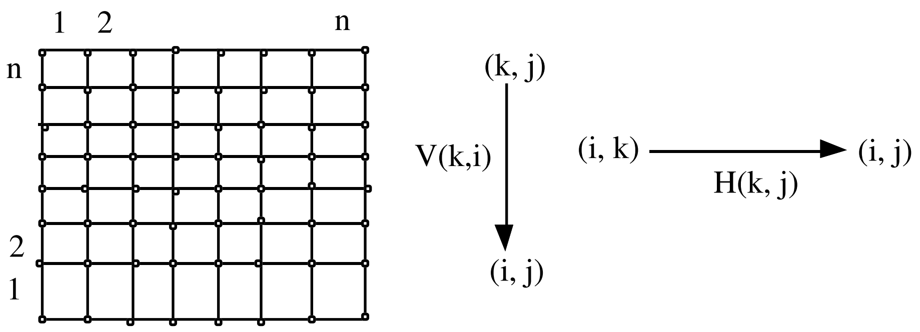

Fig.1. The edges of L are vertical (directed top-down) or horizontal (directed left-to-right) lines of any natural length smaller than n.

Observation 1. The triple (H, V, s) is a *succinct* representation of the lattice graph. A graph L has O(n^3) edges, but its representation (H, V, s) has size only O(n^2).

Given the weights of edges, we introduce the terminology related to *shortest paths* with respect to a given semiring.

We define the weight of a path as the multiplication (under $\otimes$) of weights of consecutive edges of this path. The weight of the empty path is defined to be the *unit element* of the semiring.

The weight Φ(x) of the *shortest path* from the source s to a node x (in the semiring S) is defined as

$$\Phi(x) = \sum \text{weight}(\mu),$$

where the summation is over all all nonempty paths μ from s t o x

The most intuitive meaning of $\Phi(x)$ is when '+' is the *minimum* and $\otimes$ is the usual summation. Then $\Phi(x)$ is the weight of shortest paths from the the source. For other semirings it is not the shortest path n the usual meaning, however we adopt generally the terminology *shortest paths* for any semiring.

The ***single source shortest path*** problem is to compute $\Phi(x)$ for all $x \in N$.

Lemma 1 (***shortest paths lemma***)
Assume $M_S(n) = O(n^{2+\varepsilon})$ and the *lattice graph L* is given by a triple (H, V, s). Then the *single source shortest path* problem for L can be computed in time $T1(n) = O(M_S(n))$, if $M_S(n) = O(n^{2+\varepsilon})$. The function $T1(n)$ satisfies the recurrence (**).
Proof (postponed till Section 3).

2. Main algorithm

Let us fix a context-free grammar $G=(V_N, V_T, P, S)$ in Chomsky normal form, see [AHU79]. V_N, V_T are sets of, respectively, nonterminal and terminal symbols. P is the set of production rules and S is the axiom.

Let R be the semiring of binary relations over the set V_N of nonterminals. The operation '+' is the set-theoretical union of relations and $\otimes$ is the composition of relations.

The following lemma corresponds to an analogous lemma stated in [Va75] for *syntactic semirings*. Recall that $M_S(n)$ is the time to multiply two n by n matrices over the semiring S and *BM(n)* is the time to multiply Boolean matrices.

Lemma 2.
$M_R(n) = O(BM(n))$.
Proof.
Assume M is a matrix over the semiring R. Each element M[i,j] is a relation, we can assume it is represented by a boolean matrix. Assume $|V_N|=s$, identify (in this proof) nonterminals with integers 1..s. Then the element M[i,j] is an s by s boolean matrix.

Hence it makes sense to write M[i,j][p,q]. A matrix M over R can be represented by s^2 boolean matrices $M^{p,q}$, where $M^{p,q}[i,j] = M[i,j][p,q]$.

Assume we have two matrices $M1$ and $M2$ over the semiring R. Let $C = M1 \otimes M2$. Then

$$C^{p,q} = \sum_{r\in [1..s]} M1^{p,r} \times M2^{r,q} ,$$

where $\times$ is the Boolean matrix multiplication and '+' is the boolean 'or'.

Hence we have a constant number of multiplications to compute $M1 \otimes M2$. This completes the proof. ♦

Denote $I = \{ (A, i, j)$: $A \in V_N$, $1 \le i < j \le n$ }. Elements of I are called *items*. We say that a pair (A,i,j) is a *valid item* iff the substring $a_i a_{i+1} ... a_j$ is derivable from the nonterminal A in the gramar G.

The *item* (A, i, j) can be interpreted as a *potential* derivation tree whose root is labelled A and leaves correspond to consecutive symbols of the subword $a_i a_{i+1} ... a_j$.

The *item* is said to be *valid* iff its derivation tree can be realized according to the grammar.

We use the relation $\Rightarrow$ of implication: if one item is valid then other item is also valid We define it formally later.

The general informal structure of the **main algorithm** is:

compute recursively a partial set $\prod$ of valid items and a relation $\Rightarrow$.

Then close $\prod$ with respect to $\Rightarrow^*$. The resulting set is the set of all valid items.

Denote by *VALID(k,l)* the set of all *valid items* (A,i,j) such that $k \le i < j \le l$. Then

$$w \in L(G) \text{ iff } (S,1,n) \in VALID(1,n).$$

Hence it is enough to compute the set *VALID(1,n)*. We can assume that the length of w is a power of two (by appending several dummy symbols and changing slightly the grammar). Assume that the set

$$\prod = VALID(1,n/2) \cup VALID(n/2+1,n)$$

is already computed. It is the partial set of *valid items*.

Define formally the following relation $\Rightarrow$ of "implication".

$(B,i,j) \Rightarrow (A,i,j+k)$ iff there is a rule $A \to BC$ and $(C,j+1,j+k) \in \prod$ for some $C \in V_N$;

$(C,i,j) \Rightarrow (A,i-k,j)$ iff there is a rule $A \to BC$ and $(A,i-k,i-1) \in \prod$ for some $A \in V_N$

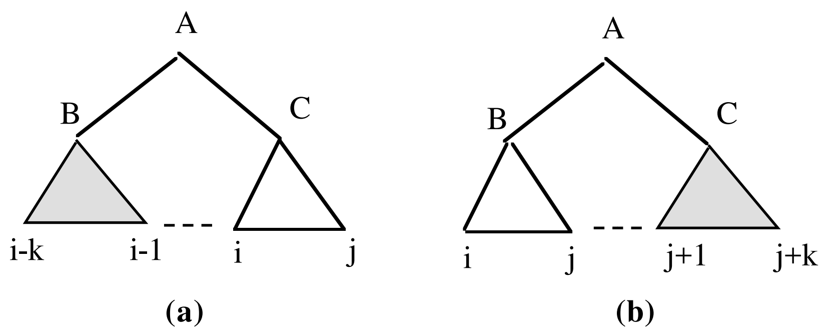

Fig. 2. **(a)** $(C,i,j) \Rightarrow (A,i-k,j)$, the *shadowed* item $(B,i-k,i-1) \in \prod$ is known to be valid. If (C,i,j) is valid then $(A,i-k,j)$ is also valid. **(b)** (B,i,j) $\Rightarrow$ (A,i,j+k), the *shadowed* item (C,j+1,j+k) $\in \prod$ is known to be valid. If (B,i,j) is valid then (A,i,j+k) is also valid.

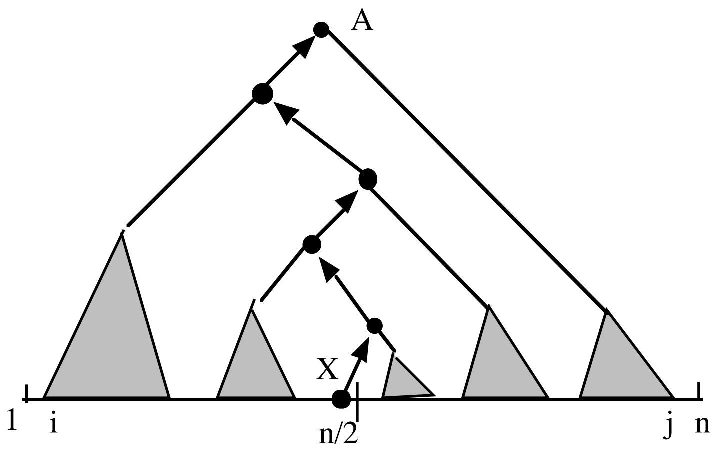

Figure 3. A path of implications $(X,n/2,n/2) \Rightarrow^* (A,i,j)$ from an item $(X,n/2,n/2)$ corresponding to the segment $[n/2, n/2]$ to an item corresponding to the segment $[i..j]$. The items in $\prod$ are shaded. The final weight of (i,j) is the relation containing (X,A).

Observation 2. The derivation tree corresponding to a valid item (A,i,j) has the structure presented in Figure 3. There is a path from leaf $n/2$ to the root such that each outgoing subtree corresponds to a valid item in $\prod$.

Let $\Rightarrow^*$ be the transitive closure of $\Rightarrow$. Define the set

$$IMPLIED(k,l) = \{\ (A,i,j) :\ (X,k,l) \Rightarrow^* (A,i,j), \text{ and } (X,k,l) \in \prod \text{ for some } X \in V_N\}.$$

In other words *IMPLIED(k,l)* is the set of all items whose validity is implied (transitively) by the validity of items corresponding to the segment [k..l]. All items in $\prod$ are valid, hence all items in IMPLIED(k,l) are also valid. Due to Observation 2 if $i \le n/2$, $j > n/2$ then the following equivalence holds:

$$(A,i,j) \in VALID(1,n) \Leftrightarrow (A,i,j) \in IMPLIED(n/2,n/2).$$

This implies directly the following lemma.

Lemma 3.
$VALID(1,n) = \prod \cup IMPLIED(n/2,n/2)$.

Lemma 4.
The set *IMPLIED*(n/2,n/2) can be computed in $O(BM(n))$ time.

Proof.

Construct the lattice graph *L* of size n with the weights in the semiring *R* of binary relations over V_N. The *source* is the pair (n/2,n/2) .The weight of a (horizontal or vertical) edge (i,j)→ (i',j') is the binary relation R such that $(X,A) \in R$ iff $(X,i,j) \Rightarrow (A,i',j')$.

Then the definition of " ⇒ " implies directly the following fact.

Claim 1.
The *congruency condition* is satisfied in the lattice *L*.

Let Φ be the matrix of weights of shortest paths from the source in the grah *L*. It is easy to see the following.

Claim 2.

$(A,i,j) \in IMPLIED(n/2,n/2)$ iff $(X,A) \in \Phi(i,j)$ for some $(X,n/2,n/2) \in \Pi$.
Now due to Lemma 2 and Lemma 1 Φ can be computed in O(*BM(n)*) time. ♦

```
function VALID(i,j);
 m:=j-i+1 {m is a power of two}
if m is small then
        compute VALID(i,j) in a constant time else
begin
        Π:= VALID(i,m/2) ∪ VALID(m/2+1,j);
        construct the lattice graph L; ShortestPaths(L);
        compute IMPLIED(m/2,m/2); {see Claim 2}
        return VALID(i,j) = Π ∪ IMPLIED(m/2,m/2);
end;
```

Theorem 1. (***main result***)
We can recognize context-free languages in time $T(n) = O(n^{2+\varepsilon})$ time, if the time to multiply two n by n boolean matrices is $O(n^{2+\varepsilon})$, for a constant $\varepsilon > 0$. *T(n)* satisfies the recurrence (*).
Proof.
The recursive structure of the computation of *VALID*(1,n) is presented above, it is based on Lemmas 3 and 4. The algorithm is correct due to Lemmas 2, 3 ,4.
Let T(m) be the time complexity to perform *VALID(*i,j) for $m = j-i+1$. Then it satisfies the recurrence (*). This implies the claimed time bounds and completes the proof. ♦

3. Proof of the shortest paths lemma.

Assume we have computed partially shortest paths (the summation is over a subset of possible paths) and the actually given weight of paths from the source to x for each node is $\Psi(x)$. Define the operation *SingleEdgeExtend*(L) which extends the paths by single edges. Ψ is changed (by this operation) as follows:

$$\Psi(x) := \Psi(x) + \sum \Psi(x) \otimes \mathit{weight}(x', x),$$

where the summation is over all directed single edges (x', x) of the graph L.

Let $M_S(n)$ be the time two multiply two n by n matrices over the semiring S .

Lemma 5. The operation *SingleEdgeExtend*(L) can be computed in time $O(M_S(n))$.

Proof.

We can "advance" by one edge horizontally or vertically. Advancing by vertical (horizontal) edges corresponds to the computation for each $i.\,j$ of, respectively,

$$\Psi_{\mathrm{v}}(\mathrm{i,j}) = \sum_{\mathrm{k}} \Psi(k, j) \otimes \mathit{weight}(\ (k,j) \rightarrow (i,j)\ ) = \sum_{\mathrm{k}} \Psi(k, j) \otimes V(k,i),$$

$$\Psi_{\mathrm{h}}(\mathrm{i,j}) = \sum_{\mathrm{k}} \Psi(i,k) \otimes \mathit{weight}(\ (i,k) \rightarrow (i,j)\ ) = \sum_{\mathrm{k}} \Psi(i,k) \otimes H(k,j).$$

The total change of Ψ corresponds to the following operation on matrices:

$$\Psi := \Psi + \Psi_{\mathrm{v}} + \Psi_{\mathrm{h}} := \Psi + (\Psi^{\mathrm{T}} \otimes \mathrm{V})^{\mathrm{T}} + \Psi \otimes \mathrm{H},$$

where V, H are the matrices of *vertical weights*, and *horizontal weights*, respectively, and Ψ^{T} is the transposition of the matrix Ψ.

It involves two matrix multiplications over S, so the complexity is $O(M_S(n))$. This completes the proof.♦

Proof of Lemma 1.

Assume X is a square subarray of L. Assume also that we have computed for each $x \in X$ the weight of the shortest path from s to x such that the only node in X on the path is the last node.

We define the *ShortestPaths(X)* which computes $\Phi(x)$ for each $x \in X$ under such assumption.

Observe that if X is the whole graph and nothing is precomputed then *ShortestPaths(X)* computes the *shortest path* problem for the whole graph L. Assume that initially we have the table Ψ such that $\Psi(x) = \varnothing$ (the *zero* of the semiring) for each *x*.
Assume, without loss of generality, that *n* is a power of two. Partition the lattice array L into four disjoint lattice graphs A, B, C and D which are quadrants of L listed in anti-clockwise order, see Figure 4.

The implementation of *ShortestPaths(L),* presented below, is based on the structure of paths illustrated in Figure 4.

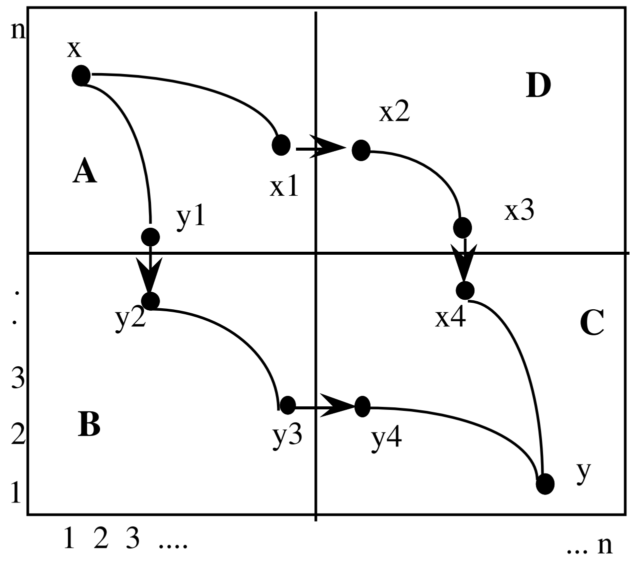

Figure 4. The structure of paths from a point x in quadrant A to a point y in quadrant C.

```
procedure ShortestPaths(L)
  if size(L) is small then
        compute ShortestPaths(L) in a constant time else
begin
        1: ShortestPaths(A);
        2: SingleEdgeExtend(L);
        3: ShortestPaths(B); ShortestPaths(D);
        4: SingleEdgeExtend(L);
        5: ShortestPaths(C);
end;
```

The correctness means that each path μ from a point x to a point y contributes weight(μ) to the new total weight $\Psi(y)$ of y. We show it for the case when a node y is in quadrant C and the path starts at a point x in quadrant A.

We refer to Figure 4. Let us consider one such path μ. It goes through quadrant D or through quadrant B. Assume the latter case. Take the last point *y1* of μ in A, the first point *y2* in B, then the last point *y3* in B and the first point *y4* in C. The structure of the path μ is $x \rightarrow^* y1 \rightarrow y2 \rightarrow^* y3 \rightarrow y4 \rightarrow^* y$. In step 1 the weight of the subpath $x \rightarrow^* y1$ is computed, in step 2 it is computed the weight of $x \rightarrow^* y1 \rightarrow y2$ etc. Eventually the weight of the whole path is computed and contributed to the total new weight Φ of y.
Other cases can be considered analogously.

Assume $M_S(n) = O(n^{2+\varepsilon})$, denote by *T1(n)* the time to compute *ShortestPaths(L)* . Then *T1(n)* satisfies the recurrence (**), with *BM* replaced by M_S . Consequently *T1(n)* = $O(n^{2+\varepsilon})$. This completes the proof. ♦

References

[AHU74] J.Aho, J.Hopcroft, J.Ullman, The design and analysis of algorithms, Addison-Wesley (1974)

[BKR91] L.Banachowski, A.Kreczmar, W.Rytter, Analysis of algorithms and data structures, Addison-Wesley (1991)

[Ha78] J.Harrison, Introduction to formal language theory, Addison-Wesley (1978)

[[Ry 88] W. Rytter, On efficient parallel computation of costs of paths on a grid graph, IPL 29 (1988) 71-74

[Va75] L.Valiant, General context-free recognition in less than cubic time, Journal of Comp. and Systems Sc. 10:2 (1975) 308-315.