Yugoslav Journal of Operations Research
13 (2003), Number 1, 61-67

A NON-RECURSIVE ALGORITHM FOR POLYGON TRIANGULATION

Predrag S. STANIMIROVI], Predrag V. KRTOLICA,
Rade STANOJEVI]

Faculty of Science and Mathematics,
University of Ni{, Ni{, Serbia
pecko@pmf.pmf.ni.ac.yu, krca@pmf.pmf.ni.ac.yu, rrr@eunet.yu

Abstract: In this paper an algorithm for the convex polygon triangulation based on the reverse Polish notation is proposed. The formal grammar method is used as the starting point in the investigation. This idea is "translated" to the arithmetic expression field enabling application of the reverse Polish notation method.

Keywords: Reverse Polish notation, convex polygon triangulation, contex-free grammar.

1. INTRODUCTION AND PRELIMINARIES

The triangulation of the convex polygon is the following problem. For the given polygon find the number of the possible splitting on triangles by its diagonals without gaps and overlaps of these splitting. This is the classical problem solved so far in a few ways.

One solution from [4] uses the context-free grammar with productions:

$$S \to aSS, \quad S \to b, \tag{1}$$

where a and b are terminals, and S a non-terminal symbol.

The triangulation of polygon is based on the following principles:

(a) The non-terminal S represents an oriented topological segment. This segment is named potential.

(b) The chain aSS corresponds to the oriented topological triangle with one real edge a and two potential edges S and S.

(c) The production $S \to aSS$ means the replacement of the potential segment S by the triangle aSS, consisting of its real edge a defined orientation and potential edges S, being the edge a with defined orientation and potential edges S, being the edge of the polygon which is about to be defined.

(d) The replacement $S \to b$ means replacing the potential edge S with the real edge b.

If we apply $n-2$ times the first production in (1) (in other words, if we produce a word with $n-2$ $a's$ in it) and then we apply, the corresponding number of times, the second production in (1), we get one triangulation of the polygon with n edges.

In [4] it is shown that the number of all possible triangulations of the n-edges polygon $f(n)$ is given by the recurrent formula

$$f(2) = f(3) = 1,$$
$$f(n) = \sum_{i=2}^{n-1} f(i) f(n-i+1)$$

Of course, implementation of this approach (especially if we want to list all triangles by their nodes) is an additional problem.

Being inspired by the method described above, we replace the first grammar rule in (1) and use the following rules generating the arithmetic expression in the reverse Polish notation.

$$S \to SS+ \tag{1.1}$$

$$S \to b \tag{1.2}$$

Before we start with the algorithm construction, let us expose some of the results concerning the reverse Polish notation method based on the properties investigated in [5]. Note that the expression in the reverse Polish notation is stored in the array postfix, where $postfix[i]$, for each $i \geq 0$, is a string which denotes an expression element, i.e. a variable, a constant, or an operator.

Definition 1.1. The grasp of the element $postfix[i]$ is the number of its preceding elements which form operand(s) of the element $postfix[i]$. We denote the grasp of the element $postfix[i]$ by $GR(postfix[i])$. Integer i is called the index of the element $postfix[i]$. Index i of the element $postfix[i]$ will be alternatively denoted by $IND(postfix[i])$.

Definition 1.2. The grasped elements of the operator $postfix[i]$ are the grasp left preceding elements in the array postfix which form operand(s) of the operator $postfix[i]$. The index of the most left element among them is called the left grasp bound. The left grasp bound of the operator $postfix[i]$ is denoted by $LGB(postfix[i])$.

Definition 1.3. The element $postfix[i]$ is called the main element or head for the expression formed by $postfix[i]$ and its grasped elements.

In the second section we investigate a 1-1 and onto mapping $F_n : P_n \mapsto T_n$ from a subset of the expressions, made by consecutive application of the rule (1.1) $n-2$ times and the rule (1.2) $n-1$ times, to the set of an n-gon triangulations.

In the third section, we construct the algorithms for the polygon triangulation using the results of the second section.

2. ARITHMETICAL EXPRESSIONS AND TRIANGULATIONS

Since the production $S \to aSS$ means the construction of the triangle aSS with the real oriented side a and two potential oriented sides S, one can assume that the production $S \to SS+$ means the construction of the triangle obtained by replacing the real side a of the triangle aSS by the real side $+$, keeping the orientation. In this way, we replace the terminal a by the sign $+$ which can be considered as the arithmetic operator, with two S's as its operands.

Using this correspondence, for every $n \geq 3$ we can consider the mapping $F_n : R_n \mapsto T_n$, whose domain is the set of expressions made by the consecutive application of the rule (1.1) $n-2$ times and the rule (1.2) $n-1$ times, and the range is the set of an n-gon triangulations. But, it is not difficult to verify that the set R_n contains multiple elements. In the following lemma we got unique characterization for each element in R_n.

Using known notations from [6], by V^n we designate all strings of the length n on V, and by V^* we denote the closure set $\{\varepsilon\} \cup V \cup V^2 \cup \cdots$, where ε is the empty string. Also, by $\{b,+\}^*_{p,q}$ we denote the subset of the closure set $\{b,+\}^*$ consisting of p appearances of the sign b and q appearances of the sign $+$. It is easy to see that $\{b,+\}^*_{0,0} = \{\varepsilon\}$.

Lemma 2.1. An arbitrary element $r_n \in R_n$, corresponding to the particular triangulation of n-angle polygon, is uniquely determined by the following two conditions:
(C1) It possesses the form

$$r_n = bb\alpha+, \quad \alpha \in \{b,+\}^*_{n-3,n-3},$$

(C2) Each initial part of the expression r_n (the substring of consecutive characters which start from the first character) must be of the form

$$\{b,+\}^*_{p,q} \quad p > q, \quad p = 1,...,n-1, \; q = 1,...,n-2.$$

Proof: We use the induction by n. For $n = 3$ the expression $bb+$ corresponds to the triangle, and satisfies the conditions (C1) and (C2). An arbitrary $(k+1)$-gon triangulation is derived replacing by triangle an adequate side of a triangle in the corresponding k-gon triangulation. Consider the expressions

$$\begin{aligned} &bb+, \quad bb\alpha b\beta+, \\ &\alpha \in \{b,+\}^*_{p_\alpha,q_\alpha}, \; \beta \in \{b,+\}^*_{p_\beta,q_\beta}, \; p_\alpha + q_\alpha = n-4, \; p_\beta + q_\beta = n-3 \end{aligned} \tag{2.1}$$

which satisfy condition (C2) and correspond to any k-gon triangulation. Each of these expression satisfies (C1). Then, corresponding $(k+1)$-gon triangulation is determined by one of the following expressions

$$bbb++, \; bb+b+, \; bbb+\alpha b\beta+, \; bb+b\alpha b\beta+, \; bb\alpha bb+\beta+, \tag{2.2}$$

which satisfy (C2) and

$$\alpha \in \{b,+\}^*_{p_\alpha,q_\alpha},\ \beta \in \{b,+\}^*_{p_\beta,q_\beta},\ p_\alpha + p_\beta = n-4,\ q_\alpha + q_\beta = n-3 .$$

Each of the expressions from (2.2) is uniquely determined. Since the following transformations are valid

$$bbb++ = bb\gamma+,\ \gamma = b+ \in \{b,+\}^*_{1,1}$$

$$bb+b+ = bb\delta+,\ \delta = +b \in \{b,+\}^*_{1,1}$$

$$\{bbb+\alpha b\beta+,\ bb+b\alpha b\beta+,\ bb\alpha bb+\beta+,\ \alpha \in \{b,+\}^*_{p_\alpha,q_\alpha},\ \beta \in \{b,+\}^*_{p_\beta,q_\beta},$$

$$p_\alpha + p_\beta = n-4,\ q_\alpha + q_\beta = n-3\} = \{bb\gamma+,\ \gamma \in \{b,+\}^*_{n-2,n-2}\}$$

we conclude that these expressions satisfy the condition (C1), too. Using the same transformations and the inductive hypothesis we can prove that these expressions also satisfy the condition (C2). ♦

By P_n we denote the set of expressions generated by the grammar rules (1.1), (1.2) which satisfy conditions (C1) and (C2).

Corollary 2.1. The grasp of the head element in the expression corresponding to the particular triangulation of n-angle polygon is $2n-4$. The head element has $n-1$ signs b and $n-3$ signs $+$ as its grasped elements, and the initial part of the grasped elements is of the form

$$bb\{b,+\}^*_{p,q},\ p > q,\ p = 1,...,n-3,\ q = 1,...,n-3 .$$

Proof: From Lemma 2.1 it follows that the length of the expression from P_n corresponding to any triangulation of the polygon with n angles is $2n-3$, possesses the form $bb\{b,+\}^*_{n-3,n-3}$, and each of its initial parts is of the form $bb\{b,+\}^+_{p,q}$, $p > q$, $p = 1,...,n-3$, $q = 1,...,n-3$. Then, the proof is obvious from Definition 1.1 and Definition 1.3. ♦

Lemma 2.2. The mapping $F_n : P_n \mapsto T_n$ is well defined, one-to-one and onto for any $n \geq 3$.

Proof: Firstly, we shall prove by the induction that mappings F_n, $n \geq 3$, are well defined. The claim will be proved using induction by n, $n \geq 3$. For $n = 3$, the unique expression $bb+$, obtained by one application of the rule (1.1) and two applications of the rule (1.2), corresponds to the unique and trivial triangulation of the triangle.

Suppose that the claim is valid for $n = k$, $k \geq 3$.

In the case $n = k+1$, consider an arbitrary expression (2.2) generated by rule (1.1) applied $k-1$ times and by rule (1.2) applied k times. If in the expression under the consideration we replace the appearance of the substring $bb+$ by b (using the opposite directions in the rules (1.2) and (1.1)), then we get the expression of the form (2.1) generated by applying the rule (1.1) $k-2$ times and rule (1.2) $(k-1)$ times. According to the inductive hypothesis it corresponds to the unique triangulation of the k-gon. But, returning $bb+$ to the previous place, and transforming it into $bb+$, we get the starting

expression (2.2) making the additional triangle on the edge of the k-gon which corresponds to the transformed b. In this way we get one unique triangulation of $(k+1)$-gon.

Similarly, we can prove by induction that the mapping F_n is one-to-one.

In order to prove that the mapping F_n is onto, we will construct an effective procedure to get the expression from P_n starting from the given triangulation. Suppose that we have one triangulation with oriented edges and diagonals. To every diagonal and to edge AB assign the $+$ sign and to the rest of edges assign the b sign. Observe two edges which define one triangle with edge AB. Denote the incoming edge sign by L and outgoing edge sign by R. Generally, the needed expression has the form $L+R$. If L or R is equal to $+$ sign, it should be surrounded by the parenthesis and the algorithm recursively applied to it. When we complete the recursion, transform the begotten expression to the reverse Polish form which is obviously the element of the set P_n. ♦

3. THE ALGORITHM

We should generate all postfix expressions corresponding to the particular triangulations, and then we shall get one triangulation for each of these expressions.

All expressions from the set R_n could be generated using the result of Lemma 2.1 and backtracking method. We start from the highest allowed expression in the lexicographic sense. This is the expression

$$S^{n-1}+^{n-2}.$$

In every further step we find the first allowed expression lexicographically less than the previously found expression. In this way we could find all allowed expressions form R_n.

We need to generate triangulations corresponding to every allowed word from R_n. Let w be an arbitrary word from R_n. Hence, w is the postfix notation of the expression containing $n-1$ operands $(S's)$. It is clear that there is bijection between R_n and the set of binary trees with $n(n-1)/2$ nodes and depth $n-2$. We could observe every node as a set of consecutive operands which are descendants of a given node. If we numerate the vertices of a given polygon by integers $1,...,n$, then we could wee every vertex designed by number for $1,...,n-1$ as an operand from w. Henceforth, every tree node in the tree corresponding to some polygon represents the set of consecutive polygon vertices. This set is defined by the first and the last vertex, i.e. by its last vertex and by the last vertex of the previous node in the same tree level. So, it is enough to remember in every tree level the biggest integer numerating some of the node descendants. Thus, we conclude that accompanying the node v with the first vertex p and the node u with the last vertex q is the separation of the vertices $p, p+1,...,q$ from the rest of the vertices. In geometrical sense, it corresponds to drawing the diagonal $(p,q+1)$. We are ready to present the algorithm for the n-gon triangulation corresponding to the word w.

1. Assign to every S from w an ordinal number from left to right.
2. Let the pointer curr be the first element index of word w.

3. If the value of the element pointed by curr is S, then curr gets the index value of the next element and repeat this step. If its value is equal to $+$ go to step 4, or if it is the end of string go to step 5.
4. In this moment pointer curr points to a $+$ from word w. By Lemma 2.1 it is assured that to this $+$ at least two S precede (which represents nodes of the binary tree). Our goal is a transition to the higher level. We achieve this by joining two S nodes which proceed to $+$, and printing the diagonal $(pred(pred(pred(curr))), pred(curr)+1)$ (the function pred designates the predecessor of its argument). Further, in this step we eliminate current $+$ and preceding node S. The pointer curr gets the index value of $(pred(pred(curr)))$, while we remember the last vertex of the node $pred(curr)$ as the last vertex of the node $(pred(pred(curr)))$. Go to step 3.
5. At this point, all diagonals corresponding to the word w, making particular triangulation of the given polygon, are generated and algorithm is stopped.

In the following table we present behavior of this algorithm.

Table 3.1.

n	# of different triangulations	Processing time [s]
13	58786	2
14	208012	6
15	742900	22
16	2674440	91
17	9694845	342
18	35357670	1455

4. CONCLUSION

As we saw, starting from the triangulation strategy based on the formal grammar from [4] and using the arithmetic expressions in the reverse Polish notation, we get the relatively simple algorithm for polygon triangulation. Similar idea could be found in [3], where an algorithm, based on the matrix multiplication and corresponding parse trees, is presented. But, this algorithm needs to operate with data structures which are more complex than in the case of our algorithm.

The reader should be aware that our intention was not to present revolutionary better algorithms for polygon triangulation. The problem of convex polygon triangulation is an old one and has been solved so far in many ways (see, for example, [1, 2]) and it is known that it could be done in linear time. Our main goal is to show how to successfully apply reverse Polish method for the symbolic computation in this area.

REFERENCES

[1] Chazelle, B., "Triangulation a simple polygon in linear time", Discrete Comput. Geom., 6 (1991) 485-524.

[2] Chazelle, B., and Palios, L., "Decomposition algorithms in geometry", in: Ch.L. Bajaj (ed.), Algebraic Geometry and Its Application, Springer-Verlag, New York, Inc., 1994, 419-447.

[3] Corman, T.H., Leiserson, C.E., and Rivest, R.L., Introduction to Algorithms, MIT Press,

Cambridge, Massachusetts, London, England, 1990.
[4] Kross, M., and Lentin, A., Notions sur les grammaries formelles, Gauthier-Villars, 1967.
[5] Krtolica, P.V., and Stanimirovi}, P.S., "On some properties of reverse Polish notation", FILOMAT, 13 (1999) 157-172.
[6] Tremblay, J.P., and Sorenson, P.G., The Theory and Practice of Compiler Writing, McGraw-Hill Book Company, New York, 1985.