Reverse Polish notation in constructing the algorithm for polygon triangulation

Predrag V. Krtolica,
Predrag S. Stanimirović and Rade Stanojević

Abstract

The reverse Polish notation properties are used in the construction of the algorithms for the polygon triangulation. The formal grammar method is "translated" to the arithmetic expression field enabling application of the reverse Polish notation method. The result of this approach is a relatively simple algorithm for polygon triangulation[1].

1 Introduction and Preliminaries

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

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

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

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, named *potential*.

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

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

[1]Presented at the IMC "Filomat 2001", Niš,August 26–30, 2001
2000 Mathematics Subject Classification: 68Q40
Keywords: Reverse Polish notation, polygon triangulation, formal grammars

If we apply $n-2$ times the first production in (1) and then we apply $n-1$ times the second production in (1), we get one triangulation of the n-gon.

However, in [4] there is no algorithm suggested for implementation of described approach. Algorithm proposed here presents detailed implementation of the formal grammar approach. Being inspired by the method described in [4], we replace the first grammar rule in (1) and use the following rules generating the arithmetic expression in the reverse Polish notation.

$$S \to SS+ \qquad (1.1) \qquad\qquad S \to b \qquad (1.2)$$

In this way, we switch on arithmetic expressions. Similar idea is 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. As we shall see, our algorithm needs no more than one string of characters, one array of integers and one matrix to store particular triangulation, and corresponding processing is relatively simple.

Before we start with the algorithm construction, let us expose some of the notations 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 $GR(postfix[i])$.*

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.*

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 bijection $F_n : P_n \mapsto T_n$ from the 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 an algorithm for the polygon triangulation, using the properties of the reverse Polish notation and the results of the second section.

2 Arithmetical Expressions and Triangulations

Since we replace the production $S \to aSS$ by production (1.1), in the corresponding triangle we replace the real side a by the real side $+$, keeping the orientation. Sign $+$ 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. It is not difficult to verify that some elements from R_n could be derived in multiple ways. In the following lemma we get unique characterization for each element in R_n.

Beside the known notations (see, e.g. [6]), 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 $+$. In the boundary case, we have $\{b,+\}^*_{0,0} = \{\epsilon\}$.

Lemma 2.1 *An arbitrary element $r_n \in R_n$, corresponding to a particular triangulation of n-sided 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

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

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

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

which satisfy condition $(C2)$ and correspond to any k-gon triangulation. Each of these expressions satisfies $(C1)$. Then, the 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

$$\begin{array}{c} bbb++ = bb\gamma+,\ \gamma = b+ \in \{b,+\}^*_{1,1} \\ bb+b+ = bb\delta+,\ \delta = +b \in \{b,+\}^*_{1,1} \\ \{bbb+\alpha b\beta+,\ bb+b\alpha b\beta+,\ bb\alpha bb+\beta+,\ \alpha \in \{b,+\}^*_{p_\alpha,q_\alpha},\ \beta \in \{b,+\}^*_{p_\beta,q_\beta}, \\ p_\alpha + p_\beta = n-4,\ q_\alpha + q_\beta = n-3\} = \{bb\gamma+,\ \gamma \in \{b,+\}^*_{n-2,n-2}\} \end{array}$$

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

Corolarry 2.1 *The grasp of the head element in the expression corresponding to a particular triangulation of an n-sided 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*

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

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 its initial part is of the form $bb\{b,+\}^*_{p,q}$, $p>q$, $p=1,\ldots,n-3$, $q=1,\ldots,n-3$. 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 the mappings F_n, $n \geq 3$, are well defined. For $n=3$, the unique expression $bb+$ corresponds to the unique and trivial triangulation of a triangle.

In the case $n = k+1$, consider an arbitrary expression from P_{k+1} corresponding to an expression form (2.2). 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 one of the expressions from (2.1) belonging to P_k. By the inductive hypothesis it corresponds to the unique triangulation of the k-gon. But, returning $bb+$ on the previous place, we get the starting expression (2.2) making an 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 an expression from P_n starting from a 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 on it. When we complete the recursion, transform the obtained expression to the reverse Polish form which is obviously the element of the set P_n. ■

3 The Construction of the Algorithm

Definition 3.1 *By the non-commutative sum of two integers a and b, denoted by $a \oplus b$, we assume the usual sum $a + b$, but not $b + a$.*

Definition 3.2 *By the multilevel decomposition of an even integer n we assume its presentation in the form of the non-commutative sum of odd summands not higher than 3, made by using the following rules:*

(1) Express the even integer n as a non-commutative sum of two odd integers.

(2) In every sum continue to decompose those summands greater than 3, but decreasing by 1 every such summand before the further decomposition. The summand a which is decreased by one before the further decomposition will be denoted in the multilevel decomposition by a_+. The value of this expression is $a + 1$.

Theorem 3.1 *There is a bijection between the set T_n of different triangulations of the given n-gon and the set M_n of different multilevel decompositions of the number $2n - 4$.*

Proof. We prove the existence of the bijection between the sets P_n and M_n. Consider the expression $bb\alpha+$, $\alpha \in \{b, +\}^*$ from P_n corresponding to any n-gon triangulation. From the Corollary 2.1 the grasp of its head $+$ is $2n-4$. This is an even number, so the lengths of its arguments arg_1 and arg_2 must be both even or both odd. But, they cannot be even numbers, because the least length of an argument is always equal to 1 (for argument S, i. e. b after the transformation $S \to b$), and any longer argument is made by the rule (1.1) which increases its length by two.

Observe the heads op_1 and op_2 of arguments arg_1 and arg_2. In the case $GR(op_1) = 1$ or $GR(op_1) = 3$ and $GR(op_2) > 3$, the grasp of the head can be expressed as

$$2n - 4 = GR(op_1) \oplus (1 + GR(op_2)) = GR(op_1) \oplus (GR(op_2))_+.$$

This gives us multilevel decompositions

$$2n - 4 = 1 \oplus (2n - 6)_+ \text{ and } 2n - 4 = 3 \oplus (2n - 8)_+,$$

respectively.

Similarly, for $GR(op_1) > 3$ and $GR(op_2) = 1$ or $GR(op_2) = 3$, we get

$$2n - 4 = (1 + GR(op_1)) \oplus GR(op_2) = (GR(op_1))_+ \oplus GR(op_2).$$

Respectively, we can write

$$2n - 4 = (2n - 6)_+ \oplus 1 \text{ and } 2n - 4 = (2n - 8)_+ \oplus 3.$$

In the case $GR(op_1) = k_1$ and $GR(op_2) = k_2$, $k_1, k_2 \in \{5, 7, \ldots\}$, $k_1 + k_2 = 2n - 4 - 2$, we use

$$2n - 4 = (GR(op_1))_+ \oplus (GR(op_2))_+ = (k_1)_+ \oplus (k_2)_+$$

and apply further decompositions on k_1 and k_2.

Finally, it is clear that we must go on with the decomposition until we get arguments no longer than 3, because they correspond to one triangle ($bb+$).

Conversely, when we have a multilevel decomposition of $2n - 4$, it is trivial to get the expression from the set P_n, corresponding to the n-gon triangulation.

The proof can be completed using Lemma 2.2. ∎

Example 3.1 *In the case of the pentagon triangulation, the number* $2n-4=6$ *can be expressed as a non-commutative sum of two odd integers in three different ways:* $1 \oplus 5 = 6$, $3 \oplus 3 = 6$, $5 \oplus 1 = 6$. *In the first and third sum there is one summand greater than 3. We should decompose them decreasing their values for 1. All possibilities are summarized in Table 3.1.*

Table 3.1.

Multilevel decomposition	*Derivation of corresponding expression from* P_5
$1 \oplus (1 \oplus 3)_+ = 6$	$S \to SS+ \to SSS++ \to SSSS+++ \to bbbb+++$
$1 \oplus (3 \oplus 1)_+ = 6$	$S \to SS+ \to SSS++ \to SSS+S++ \to bbb+b++$
$3 \oplus 3 = 6$	$S \to SS+ \to SSS++ \to SS+SS++ \to bb+bb++$
$(1 \oplus 3)_+ \oplus 1 = 6$	$S \to SS+ \to SS+S+ \to SS+S+S+ \to bb+b+b+$
$(3 \oplus 1)_+ \oplus 1 = 6$	$S \to SS+ \to SS+S+ \to SSS++S+ \to bbb++b+$

Algorithm 3.1.

Regarding to Theorem 3.1, on the basis of one multilevel decomposition we could get one triangulation. At the first level we decompose $2n-4$ on two odd summands. For every summand higher than 3 we should continue with its decomposition.

We start from a single triangle denoting and recording its nodes as $A=1$, $B=2$, and $C=3$ (keeping the triangle orientation). During the decomposition the current triangle nodes A, B, and C change their values. Trailing one "path" of decomposition we get the corresponding triangulation in the following way. If we decompose the right summand, then this corresponds to the further transformation of right S from $SS+$ or to the further transformation of edge (B,C). In this case, the node A becomes the former node B, the node B becomes the former node C, and we introduce a new node C with value which is for 1 greater than minimal value among former nodes B and C. Further, any node into so far found triangles with value greater than or equal to the value of newly introduced node should be increased for 1. If we decompose the left summand, then this corresponds to the further transformation of left S from $SS+$ or to the further transformation of edge (C,A). In this case, the node A becomes the former node C, the node B becomes the former node A, and we introduce a new node C in the same manner as in the previous case. But, if some of the nodes C or A have the value 1, we introduce a new node C with the value which is for 1 greater than the value of maximal node among former nodes A and C.

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 P_n could be generated using 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 previous found expression. In this way we could find all allowed expressions from P_n. Having the generated set P_n we use the grasp of the operator to "reconstruct" the multilevel decomposition.

Every grasp greater then two indicates a compound argument, i.e. further decomposition of the integer. We need to know which of the arguments should be decomposed (right or left or both).

For every expression from P_n we must calculate the grasps of its elements. In [5] there is an algorithm for recursive grasp calculation. But, in this case we have only one kind of the operator and it is easy to calculate grasp non-recursively, speeding up our algorithm.

In the following table we present behavior of this algorithm.

Table 3.2.

n	# of different triangulations	Processing time [s]
13	58786	2
14	208012	11
15	742900	48
16	2674440	195
17	9694845	790
18	35357670	3197

From Table 3.2. we can conclude that complexity of Algorithm 3.2 linearly depends on n. This observation is formally expressed in the following theorem.

Theorem 3.2 *Number of needed recursions to generate one particular triangulation on an n-gon is equal to $n-2$.*

Proof. From Corollary 2.1 we have that grasp of the corresponding expression from P_n is $2n-4$. By other words, we want to prove that we need twice less recursions. For $n=3$ grasp of the corresponding expression is two and we need only one function call to generate the unique triangulation of the triangle. Suppose that claim is valid for every polygon with $k<n$ nodes and consider an n-gon. If l_1 and l_2 are the lengths of the arguments for the head of the corresponding expression we have that $l_1+l_2=2n-4$. To complete the needed recursions for these arguments (which correspond to the polygons with less then n nodes) we need $(l_1-1)/2 = gr_1/2$ and $(l_2-1)/2 = gr_2/2$ recursions respectively, where gr_1 and gr_2 are the corresponding grasps. In addition, we need additional call to process these two arguments. So, this gives

$$\frac{gr_1}{2}+\frac{gr_2}{2}+1=\frac{gr_1+gr_2+2}{2}=\frac{l_1+l_2}{2}=\frac{2n-4}{2}=n-2$$

recursive calls to complete one triangulation corresponding to an n-gon. ∎

Example 3.2 *Let us illustrate Algorithm 3.1 for $n=5$ and the decomposition "path" $6=1\oplus 5=1\oplus(1\oplus 3)_+$.*

Fig. 3.1

Fig. 3.2

Fig. 3.3

4 Conclusion

Starting from the triangulation strategy based on the formal grammar from [4] and using the arithmetic expressions in the reverse Polish notation, we get an relatively simple algorithm for polygon triangulation. Similar idea is 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 algorithm for polygon triangulation. The problem of convex polygon triangulation is old and solved so far in many ways (see, for example, [1], [2]) and it is known that it can 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] B. Chazelle, *Triangulation a simple polygon in linear time*, Discrete Comput. Geom. **6** (1991), 485-524.

[2] B. Chazelle and L. Palios, *Decomposition Algorithms in Geometry*, Algebraic Geometry and its Application (ed. Ch. L. Bajaj), Springer-Verlag New York, Inc., (1994), pp. 419–447.

[3] T. H. Corman, C. E. Leiserson and R. L. Rivest, *Introduction to Algorithms*, MIT Press, Cambridge, Massachusetts, London, England, (1990).

[4] M. Kross and A. Lentin, *Notions sur les Grammaries Formelles*, Gauthier-Villars, 1967.

[5] P. V. Krtolica and P. S. Stanimirović, *On some properties of reverse Polish notation*, FILOMAT, **13** (1999), 157-172.

[6] J.-P. Tremblay and P. G. Sorenson, *The Theory and Practice of Compiler Writing*, McGraw-Hill Book Company, New York (1985).

Faculty of Science and Mathematics, University of Niš,
Višegradska 33, 18000 Niš
krca@pmf.pmf.ni.ac.yu, pecko@pmf.pmf.ni.ac.yu, rrr@eunet.yu