ON STRINGS, BRACELETS AND BRACKETINGS

MARCUS KRACHT

1. The Problem

The problem is this: given a string of length n, how many binary constituent structures exist for this string? Put differently: suppose you want to insert brackets in such a way that a pair of bracket encloses exactly two constituents, how many ways are there to insert brackets?

We suppose that the string is $\vec{x} = x_0 x_1 \cdots x_{n-1}$. For $n = 1$ we set the number to 1, even though no brackets can be added. If $n = 2$ there is again just one solution, $(x_0 x_1)$. If $n = 3$ there are two solutions: $(x_0(x_1 x_2))$ and $((x_0 x_1)x_2)$. For $n = 4$ we have five bracketings.

$$(1)\quad (((x_0x_1)x_2)x_3),\quad ((x_0x_1)(x_2x_3)),\quad (x_0(x_1(x_2x_3))),\quad (x_0((x_1x_2)x_3)),\quad ((x_0(x_1x_2))x_3)$$

Here is how the series develops:

(2)

length of string	1	2	3	4	5	6	7
number of bracketings	1	1	2	5	14	42	132

Call the numbers κ_n. There is a general solution to this sequence. We cut the string in two parts, of length k and $k-n$, where $0 < k < n$. We know that there κ_k ways to analyse the first part and κ_{n-k} to analyze the second. Thus we get

$$\kappa_n = \sum_{k=1}^{n-1} \kappa_k \kappa_{n-k} \tag{3}$$

Notice that the right hand side has only occurrences of κ_k with $k < n$. This recursion has a known solution, the so-called **Catalan numbers**. These numbers are as follows.

$$C_n = \binom{2n}{n}/(n+1) \tag{4}$$

More exactly, we have

$$\kappa_n = C_{n-1} \tag{5}$$

It is certainly possible to ascertain the correctness by showing that the Catalan numbers satisfy the recursion. But there is another way to

show this which teaches us much more about strings and their analysis. The Calatalan number C_n describes the number of different bracelets you can make from n red and $n+1$ blue pearls. We shall see that that this has a lot to do with our problem.

2. Polish Notation

We shall make a first step of transforming the problem. First, obviously the choice of the letters does not affect the problem, so we might as well assume that the symbol is just the letter `x`, repeated n times. The next simplification is this: instead of inserting brackets, we insert the symbol `o` where the opening bracket was; the closing bracket is omitted. Thus,

$$(((x_0x_1)x_2)x_3) \mapsto \texttt{oooxxxx} \tag{6}$$

$$((x_0x_1)(x_2x_3)) \mapsto \texttt{ooxxoxx} \tag{7}$$

$$(x_0(x_1(x_2x_3))) \mapsto \texttt{oxoxoxx} \tag{8}$$

$$(x_0((x_1x_2)x_3)) \mapsto \texttt{oxooxxx} \tag{9}$$

$$((x_0(x_1x_2))x_3) \mapsto \texttt{ooxoxxx} \tag{10}$$

In this way we transform the bracketed string of length n into a string of length $2n-1$ consisting of exactly n occurrences `x` and $n-1$ ocurrences of `o`. However, not all such strings qualify, for example `xxxxooo`. So our task is to count the strings that do.

One thing to note about these strings is that they correspond to the terms in Polish Notation formed by using only the letter `x` denoting a unary symbol (a constant or a variable) and the binary operation symbol `o`. In general, Polish Notation is defined as follows. Given some operation symbols f_i, where f_i has arity $n(i)$, terms are nonzero strings over these symbols, which have the form $f_i\vec{x}_0\vec{x}_1\cdots\vec{x}_{n(i)}$, where for all $j < n(i)$, $\vec{x}_j$ is a term. Now, if in particular $n(i) = 0$ then f_i alone is a term. In out case, a string is a term iff (i) it is of the form `x` or (ii) it is of the form $\texttt{o}\vec{x}\vec{y}$, where $\vec{x}$ and $\vec{y}$ are terms.

Strings in Polish Notation can be generated using context free grammars. In our case, the terms are exactly the strings which are generated by the following grammar:

$$\texttt{S} \to \texttt{x} \mid \texttt{oSS} \tag{11}$$

Now, we need to see why writing just the opening bracket gives us Polish Notation of some sort. There is a way to see this: think of the opening bracket as a binary function symbol (so it needs two terms). Of course, if we do this, we have to get rid of the closing brackets. You can also think of it this way: keep the brackets, and insert the operator

between the opening bracket and the next symbol. Finally, erase the brackets.

$$((\mathtt{x}(\mathtt{xx}))\mathtt{x}) \mapsto (\mathtt{o}(\mathtt{ox}(\mathtt{oxx}))\mathtt{x}) \mapsto \mathtt{ooxoxxx} \tag{12}$$

The reason why one may erase the brackets without generating confusion lies in a general property of Polish Notation that we now turn to.

3. Unique Readability

Polish Notation needs no brackets. To see this, assign the following **weight** to symbols: a variable is assigned -1. The weight of x_i is denoted by $w(x_i)$. An operator symbol of arity p is assigned the weight $p-1$. We write the weights under each symbol (second line) and add them up (third line).

o	o	x	o	x	x	x
1	1	-1	1	-1	-1	-1
1	2	1	2	1	0	-1

(13)

Thus, given $\vec{x}$, put

$$\gamma(\vec{x}) := \sum_{i<n} w(x_i) \tag{14}$$

$\gamma(\vec{x})$ is the weight of $\vec{x}$. A **prefix** of $\vec{x}$ is a string of the form $x_0x_1\cdots x_k$, $k \leq n$. A **suffix** is a string of the form $x_ix_{i+1}\cdots x_{n-1}$, $i \leq n$.

Theorem 1. *A string is a term iff (a) its weight is -1, and (b) the weight of every proper prefix is ≥ 0.*

Proof. By induction on the length of the string. Let $\vec{x}$ have length 1. Then (b) is trivially satisfied, so only (a) is relevant. But clealry, it is a term iff it is of the form x_i, where x_i has weight -1. Now let $\vec{x}$ have length > 1. Suppose it is a term. Then it begins with an operational symbol of arity $n > 0$,say f. Thus it has the form $ft_0t_1\cdots t_{n-1}$. Then $\gamma(\vec{x}) = w(f)+\sum_{i<n}\gamma(t_i) = (n-1)-n = -1$. Furthermore, suppose you take a proper prefix $\vec{y}$ of this string. It has the form $\vec{y} = ft_0t_1\cdots t_{j-1}\vec{u}$, where $\vec{u}$ is either empty or a propoer prefix of t_j, $j < n$. Then

$$\gamma(\vec{y}) = (n-1) + j(-1) + \gamma(\vec{u}) = (n-1-j) + \gamma(\vec{u}) \geq \gamma(\vec{u}) \geq 0 \tag{15}$$

This shows (b). Now, assume conversely that $\vec{x}$ satisfies (a) and (b). Then its first symbol has weight ≥ 0, so it is a function symbol f of arity > 0. Let us divide $\vec{x}$ as

$$\vec{x} = f\vec{y}_0\vec{y}_1\cdots\vec{y}_{m-1} \tag{16}$$

where $\vec{y}_0$ is the smallest string starting after f of weight -1, $\vec{y}_1$ is the smallest string starting after $f\vec{y}_0$ having weight -1, and so on. (It is always possible to decompose $\vec{x}$ in this way; notice that $\gamma(x_1x_2\cdots x_{n-1}) = -w(f)-1 < 0$. Because the accumulated weight can jump up any number, it can only go down by 1; thus, there is a j such that $\gamma(x_1x_2\cdots x_j) = -1$. In general, any string with negative weight has a prefix that is a term.) By construction, $\vec{y}_i$ all satisfy (a) and (b), so they are terms. Moreover, since the weight of $\vec{x}$ is -1, we have $m = w(f)+1$, so $\vec{x}$ is a term. □

This characterization is used to show unique readability.

Corollary 2. *Let $\vec{x}$ be a term. Then it has a unique decomposition $\vec{x} = f\vec{y}_0\vec{y}_1\cdots\vec{y}_{n-1}$ with $n = w(f)$.*

Proof. Clearly, f is unique, being the first symbol. Then n is fixed, too. Now, suppose that we have a decomposition

$$\vec{x} = f\vec{y}_0\vec{y}_1\cdots\vec{y}_{n-1} = f\vec{z}_0\vec{z}_1\cdots\vec{z}_{n-1} \tag{17}$$

Then $\vec{y}_0$ and $\vec{z}_0$ are both terms, and they are prefixes of each other. Hence they are equal. Inductively one sees that $\vec{y} = \vec{z}_1$ and so on. □

4. Cyclic Transpositions

Let $\vec{x} = x_0x_1\cdots x_{n-1}$. Normally, we think of this as being written on paper. Now however think of it as being written letter by letter on the pearls of a bracelet. Then the string `oxoxx` represents the same bracelet as does `xoxxo`, because the first letter of the string is thought to follow the last. For reasons that will become clear we are interested not in the strings but in the bracelets that can be formed from them. Let $\vec{x} = x_ix_{i+1}\cdots x_{n-1}x_0x_1\cdots x_{i-1}$. Then put

$$T(\vec{x}) = x_1x_2\cdots x_{n-1}x_0 \tag{18}$$

For example, $T(\texttt{fish}) = \texttt{ishf}$. If $\vec{x}$ has length n then $T^n(\vec{x}) = \vec{x}$. It may happen, though, that $T^k(\vec{x}) = \vec{x}$ even if $k < n$, for example $T^2(\texttt{abab}) = \texttt{abab}$. As we shall see, this is not case for terms. Call a **cyclic transposition** of $\vec{x}$ a string of the form $T^k(\vec{x})$. For example, the cyclic transpositions of `abca` are `abca`, `bcaa`, `caab` and `aabc`. We shall use Theorem 1 to derive the following.

Corollary 3. *Let $\vec{x}$ be a term and of length n. Then for no $0 < i < n$, is $T^i(\vec{x})$ a term.*

Proof. $T^i(\vec{x}) = x_ix_{i+1}\cdots x_{n-1}x_0x_1\cdots x_{i-1}$. Let $\vec{y} = x_0x_1\cdots x_{i-1}$ and $\vec{z} = x_ix_{i+1}\cdots x_{n-1}$. Then $\gamma(\vec{y}) + \gamma(\vec{z}) = -1$ since $\vec{x} = \vec{y}\vec{z}$. Also,

$\gamma(\vec{y}) \geq 0$, by Theorem 1, and so $\gamma(\vec{z}) < 0$. Now $T^i(\vec{x}) = \vec{z}\vec{y}$, and it has a proper prefix of weight < 0. Hence it is not a term, by Theorem 1. □

Moreover, here is a surprising fact:

Lemma 4. *Every string with weight -1 has a cyclic transposition which is a term.*

Proof. Let $x_0x_1\cdots x_{n-1}$ be given. The sum of weights is -1, and this is the case with all cyclic transpositions. Define $\mu(\vec{x}, j) := \sum_{i=0}^{j} w(x_i)$. This is a function from the set of numbers $< n$ into the integers, which assumes a minimum $\mu_* < 0$. Let j be the least number such that $\mu(\vec{x}, j) = \mu_*$. We claim that the desired string is

$$\vec{y} = T^{j+1}(\vec{x}) = x_{j+1}x_{j+2}\cdots x_{n-1}x_0x_1\cdots x_j \tag{19}$$

To this end note that its weight is -1. We need to show therefore that all proper prefixes have weight ≥ 0, that is, that $\mu(\vec{y}, i) \geq 0$ for all $i < n$. (Case 1.) $i \leq n - j$. Then by choice of j, $\mu(\vec{y}, i) = \mu(\vec{x}, j + i) - \mu(\vec{x}, j) = \mu(\vec{x}, j + i) - \mu_* \geq 0$. (Case 2.) $n > i > n - j$. Then

$$\begin{aligned} &\gamma(x_{j+1}x_{j+2}\cdots x_0x_1\cdots x_{i-(n-j)}) \\ >&\gamma(x_{j+1}x_{j+2}\cdots x_0x_1\cdots x_{n-(n-j)}) \\ =&\gamma(\vec{x}) \\ =&-1 \end{aligned} \tag{20}$$

This is because the accumulated weight reaches its minimum first at $j = n - (n - j)$ so that the accumulated weight of the strings that are shorter is $> \mu_* = \gamma(x_0x_1\cdots x_j)$. This shows the claim. □

For example, take the string `xoxxoox`. Here is the sequence of accumulated weights.

	x	o	x	x	o	o	x
	-1	1	-1	-1	1	1	-1
	-1	0	-1	-2	1	1	0
				*			

(21)

So, we choose $j = 3$. Now, T^4(`xoxxoox`) = `ooxxoxx`, which is a term.

Theorem 5. *For given n there are exactly $\binom{2n-1}{n-1}/n$ terms of length $2n - 1$.*

Proof. First we count the number of strings. These are of length $2n-1$ and contain `o` exactly $n - 1$ times. There are $\binom{2n-1}{n-1}$ many strings of this form. To see this, notice that each string is uniquely characterized by the set of positions which contain `o`. There are $2n - 1$ available

positions of which we choose $n-1$. The symbol $\binom{2n-1}{n-1}$ denotes exactly that number.

Now, take an arbitrary string $\vec{y}$. By Lemma 4 there is a j such that $\vec{x} = T^j(\vec{y})$ is a term. Also, we know that for all $i < k < 2n-1$, $T^i(\vec{x}) = T^k(\vec{x})$. (Otherwise, $T^{k-j}(\vec{x}) = \vec{x}$, so $k-j$ must be a multiple of $2n-1$, by Corollary 3. Contradiction.) Thus, the set of strings falls into sets of $2n-1$ strings which are cyclic transpositions of each other. Hence, there must be $\binom{2n-1}{n-1}/(2n-1)$ many terms. Finally, observe that

$$\begin{aligned}\binom{2n-1}{n-1}/(2n-1) &= \frac{(2n-1)!}{(n-1)!n!(2n-1)} \\ &= \frac{(2n-2)!}{(n-1)!(n-1)!n} \\ &= \binom{2n-2}{n-1}/n \\ &= C_{n-1}\end{aligned} \tag{22}$$

□

5. Conclusion

Our method has shown a surprising connection between strings in Polish Notation and bracelets. Also, it allowed a rather painless solution to the counting of bracketed strings. Finally, let us briefly see whether the results can be generalized somewhat. First, Lemma 4 is completely general; we had to make no assumptions on the symbols we use. Second, the result can be generalized to (exactly) ternary branching, in general k–ary branching trees. However, generalizations to flexible branching are not immediate.

Department of Linguistics, UCLA, 3125 Campbell Hall, Los Angeles, CA 90095-1543