CONTEXT-FREE GRAMMARS FOR THE BALANCED, OVERBALANCED AND UNBALANCED BINARY LANGUAGES

by

George Epstein
Casper Martin
Computer Science Department
Indiana University
Bloomington, Indiana 47405

TECHNICAL REPORT NO. 103

CONTEXT-FREE GRAMMARS FOR THE
BALANCED, OVERBALANCED AND UNBALANCED
BINARY LANGUAGES

GEORGE EPSTEIN

CASPER MARTIN

MARCH, 1981

1. Introduction

The exercise of giving grammars for the balanced and overbalanced binary languages was performed approximately 20 years ago by S. Ginsburg, the first author of this technical report, and others. The feature of the work given in this technical report is its transparent structure.

First, a candidate grammar is hypothesized which preserves key features of the language to be generated. Second, there is an attempt to show that an arbitrary sentence of the language can be generated by this grammar (if the attempt fails, the candidate grammar must be widened, then sharpened as needed). Third, to obtain an extension or generalization of the language, add appropriate production rule(s) to the grammar as it stands. Examples of extension or generalization are provided as the last part of this technical report.

CONTEXT-FREE GRAMMARS FOR THE
BALANCED AND OVERBALANCED
BINARY LANGUAGES

George Epstein

Casper Martin

Balanced Binary Language

The balanced binary language is defined as the set of all strings over the symbols 0 and 1 for which the number of 0's in any string is equal to the number of 1's in that string. This language is a context-free language given by the following rewriting rules:

(1) $S \rightarrow 0S1$

(2) $S \rightarrow 1S0$

(3) $S \rightarrow 01$

(4) $S \rightarrow 10$

(5) $S \rightarrow SS.$

Proof

It is obvious that this grammar generates only those strings for which the number of 0's is equal to the number of 1's. To prove that this grammar generates all such strings, note that

I. Rule(5) followed by rules (3) and (4) produce the rule
$S \rightarrow 0110$

II. Rule(5) followed by rules (1) and (4) produce the rule
$S \rightarrow 0S110$

III. Rule(5) followed by rules (3) and (2) produce the rule
$S \rightarrow 011S0$

IV. Rule(5) followed by rules (1) and (2) produce the rule
$S \rightarrow 0S11S0$

In similar fashion, one obtains these rules with 1's and 0's interchanged:

Ia. $S \to 1001$

IIa. $S \to 1S001$

IIIa. $S \to 100S1$

IVa. $S \to 1S00S1$.

Now the rules 1-4, I-IV, and Ia-IVa exhaust all possible rewriting rules for a string with the number of 0's equal to the number of 1's. For if the string has a 0 at one end and a 1 at the other, then apply one of the rules (1)-(4) to obtain it; if the string has 0's at both ends, then apply one of the rules I-IV to obtian it; if the string has 1's at both ends, then apply one of the rules Ia-IVa to obtain it.

The first of these instructions is clearly applicable. Suppose that the string has 0's at both ends. Then the inner string must have an excess of two 1's and be one of the forms

(i) 11

(ii) S11

(iii) 11S

(iv) S11S

where in each of these forms S is again a string with the number of 0's equal to the number of 1's. To prove this statement, consider the string $\alpha = \alpha(1)\ \alpha(2)\ldots\alpha(n)$ which has two more 1's than 0's, where each $\alpha(i)$, $i = 1, 2,\ldots n$ is either 0 or 1. Let $N(i)$ be the excess in the number of 1's over the number of 0's in the string $\alpha(1)\ \alpha(2)\ldots\alpha(i)$, $i = 1, 2,\ldots n$. It is obvious that $|N(j + 1) - N(j)| = 1$ for $j = 1, 2, \ldots n - 1$; i.e., the function $N(i)$ must either increase or decrease by 1 as i moves from j to $j + 1$. In the case when $N(i) = 0$ for some i, there must be a maximum $i = i_o$ such that $N(i_o) = 0$ and $N(i) > N(i_o)$ for all $i > i_o$. Then it is clear that $\alpha(i_o + 1) = \alpha(i_o + 2) = 1$, for otherwise $i = i_o$ is not maximum. Hence, in this case the inner string is of the form (ii) or (iv) and either rule II or rule IV is applicable. Since $2 - N(i)$ is the excess in the number of 1's over the number of 0's in the string $\alpha(i + 1)\ \alpha(i + 2) \ldots \alpha(n)$, $i = 1, 2, \ldots n - 1$, if there is an i such that $2 - N(i) = 0$, then there is a minimum $i = i_p$ such that $2 - N(i_p) = 0$ and $2 - N(i) > 2 - N(i_p)$ for all $i < i_p$. It is clear that $\alpha(i_p - 1) = \alpha(i_p - 2) = 1$, for otherwise $i = i_p$ is not minimum. In this instance the inner string is of the form (iii) or (iv) and either rule III or rule IV is applicable. The only remaining case is when $N(i) \neq 0$ and $2 - N(i) \neq 0$ for all i. Since $|N(j + 1) - N(j)| = 1$ for all j and $N(n) = 2$, it follows that in this last case $N(i) > 0$ for all i. Similarly, since $|[2 - N(j)] - [2 - N(j + 1)]| = 1$ for all j and $2 - N(1)$ is either 1 or 3, it follows that $2 - N(i) > 0$ for all i. Hence, the only permissible value of $N(i)$ is $N(i) = 1$; that is, the string is of the form (i). Hence, in this last case rule I is applicable.

In similar fashion rules Ia-IVa can be proven to be applicable when there are 1's at the ends of the string. Continued application of these rules will yield any arbitrary string in which the number of 0's is equal to the number of 1's.

Overbalanced Binary Language, An Extension

The 0- overbalanced binary language is defined as the set of all strings over the symbols 0 and 1 for which the number of 0's in any string is greater than or equal to the number of 1's in that string. This unbalanced binary language is given by the rules (1)-(5) above and the additional rule

(6) $S \to 0$.

Proof

These rules obviously produce only strings with this property. Let $Z(i)$ be the excess in the number of 0's over the number of 1's in the string $\alpha(1)\ \alpha(2)\ \alpha(3)\ldots\alpha(n)$. Then, if $Z(i) = 0$ for some i, there is a maximum value $i = i_o$ such that $Z(i_o) = 0$. If $i_o < n$, then $\alpha(i_o + 1) = 0$ and there is a maximum value $i = i_1$, such that $Z(i_1) = 1$. Continuing in this fashion, there is a string

$$\alpha(1)\ \alpha(2)\ldots\alpha(i_o)0\ \alpha(i_o + 2)\ldots\alpha(i_1)0\ \alpha(i_1 + 2)\ldots\alpha(i_2)\ldots\alpha(n)$$

with the property that the 0's are markers for strings (possibly null) which have as many 0's as 1's. A similar string is produced in the case that $Z(i) \neq 0$ for all i by starting with 0 markers in the first m positions, where $Z(i) \neq m - 1$ for all $i > m$ and m is maximum, and then proceeding in the same manner as above, finding a maximum $i = i_m$ such that $Z(i_m) = m$, and so forth.

If the number of non-null strings between markers is A, then application of rule (5), $A + Z(n) - 1$ times, followed by application of rule (6), $Z(n)$ times will produce a string in which the symbol S represents only strings for which the number of 0's is equal to the number of 1's. Rules (1) - (5) will then yield the final string.

Unbalanced Binary Language, A Generalization

The following context free grammar will generate precisely those strings of ones and zeroes where the number of zeroes in the string does not equal the number of ones in the string. From now on these will be called unbalanced strings. As before, such strings are not ordered.

1. $S \to A$
2. $S \to B$
3. $A \to 1$
4. $A \to AA$
5. $A \to CA$
6. $A \to AC$
7. $B \to 0$
8. $B \to BB$
9. $B \to CB$
10. $B \to BC$
11. $C \to 10$
12. $C \to 01$
13. $C \to 1C0$
14. $C \to 0C1$

Proof

Clearly, this grammar can not generate a balanced string of ones and zeroes. Therefore, it remains to show that it can generate all unbalanced strings of ones and zeroes. Since productions three through six are analogous to productions seven through ten, we can, without loss of generality, assume that we will use production one and prove that we can generate any string with more ones than zeroes.

Consider the addition of a fifteenth production of the form C → CC. We know that productions eleven through fifteen would generate all balanced strings of ones and zeroes, due to the first result above. We will use this fact in the proof and then demonstrate that the addition of this fifteenth production is not required.

The proof will be by induction on the length of the string. There is only one string with more ones than zeroes of length one and it is 1. This string is generated by A → 1. Assume that the grammar can generate all strings with more ones than zeroes of length n or less. We now must show that any string with more ones than zeroes of length n + 1 can be generated by the grammar.

Assuming that w is a string of length n + 1 with more ones than zeroes, there are three cases we must check out. They are w = 1w' (w has a 1 on its left end), w = 0w'1 (w has a 0 on its left end and a 1 on its right end), and w = 0w'0 (w has a 0 on both ends). Note that in each case w' is the remaining substring of w. In the first case, we know that w' is either balanced or it has more ones than zeroes. If w' is balanced, we can use the productions A → AC and A → 1. The fact that we can generate all balanced strings from C takes care of the rest. If w' has more ones than zeroes we can use the productions A → AA and A → 1. In this case the induction hypothesis handles the generation of w'. It is clear that the case where w = 0w'1 can be handled in a similar manner. We now have to deal with the case where w = 0w'0. We can scan from left to right across w keeping track of the count of ones and zeroes. When the ones and zeroes counts are equal (they must be equal sometime before we reach the end of w) we will have divided w into two substrings, w = xy, where x is balanced and y has more ones than zeroes. We can now use the production A → CA. From the first result of this report, we know that x can be generated from C and by the induction hypothesis, y can be generated from A. This concludes the proof for the 15 production grammar.

Assume that the fifteenth production is required to allow this grammar to generate all unbalanced strings of ones and zeroes. This means that, at some point, the only way to add C adjacent to an existing C is by using production fifteen. Therefore, at no time did that existing C ever have a B or an A on either side of it, else production five, six, nine or ten would have served the same purpose. Note that the only way to generate a C initially is through one of these same productions, and each of them leaves an A or a B adjacent to the C. This is a contradiction. Therefore, the fifteenth production is redundant.