The normal form theorem—another proof

By D. Wood*

An alternative proof of the normal form theorem for context-free languages is given. A series of simple constructions lead to the result. The proof makes use of a practical method of removing left cycles due to Foster.

(Received September 1968, revised November 1968)

1. Introduction

The normal form theorem was proved by Sheila Greibach (1965). The proof given in that paper is not the most transparent, so the following proof was developed (Wood, 1968a) which leads the reader to the result by a sequence of simple steps. Foster (1968) in his syntax improvement device uses a method of removing left cycles from a grammar, described informally in Woodward (1966), which is the basis of Theorem 2. Section 2 describes the notation of this paper. Section 3 contains the main results, and Section 4 contains miscellaneous results and discussion.

2. Terminology and basic results

The presentation follows that of Ginsburg (1966) and McKeeman (1966).

An alphabet A, is a finite set of symbols. The number of elements in a set A, is written *Card* (A), the cardinal of A. A string (or word) S, is a concatenation of symbols over an alphabet. The length of a string S is the number of symbols in S, denoted by $|S|$. The empty (or null) string denoted by ϵ has length 0 (this is the semi-group unity, as $x\epsilon = \epsilon x = x$, where x is a word). The free semi-group over an alphabet A, denoted by A^*, is the set of all strings over A, including the empty string. The semi-group zero denoted by ϕ is defined by $\phi x = x\phi = \phi$ where $x \in A^*$. The concatenation of n copies of $x \in A^*$ is usually written x^n.

A production (or rule) $l \to r$ over an alphabet A, is an ordered pair with both $l, r \in A^*$. We call l the left of the production, r the right of the production and read the production as l produces r. Each production is regarded as a rewriting rule allowing the substitution of the right of the production for any occurrence of the left of the production in any string.

A *context-free grammar* G, is a 4-tuple

$$G = (I, T, S, P),$$

where I is a finite set of intermediate symbols which appear on the left of productions in P, T is a finite set of terminal symbols, which appear only on the right of productions in P (excluding ϵ), S is the sentence symbol, $S \in I, P$ is a finite set of productions of the form $X \to q$, $X \in I$, $q \in (I \cup T)^*$. We will henceforth write grammar for context-free grammar.

y is a derivation of x, with respect to a grammar $G, x, y \in (I \cup T)^*$, denoted by

$$x \underset{G}{\Rightarrow} y,$$

if there exists a sequence w_i, $0 \leqslant i \leqslant n, n > 0$ such that $w_0 = x$, $w_n = y$, and there exists u_i, v_i, X_i, q_i such that

$$X_i \to q_i \in P,\ w_i = u_i X_i v_i,\ w_{i+1} = u_i q_i v_i,$$
$$u_i, v_i, q_i \in (I \cup T)^*,\ X_i \in I,\ 0 \leqslant i \leqslant n-1.$$

The w_i form a derivation sequence. We write $x \Rightarrow y$ if G is understood.

A left derivation (right derivation) with respect to a grammar G, is a derivation in which each step of the derivation sequence replaces the leftmost (rightmost) intermediate symbol, written

$$x \overset{L}{\underset{G}{\Rightarrow}} y \quad (x \overset{R}{\underset{G}{\Rightarrow}} y) \quad \text{or simply}$$

$$x \overset{L}{\Rightarrow} y \quad (x \overset{R}{\Rightarrow} y) \quad \text{if } G \text{ is understood.}$$

We write $x \overset{*}{\Rightarrow} y$ if either $x \Rightarrow y$ or $x = y$. Similarly we write $x \overset{L*}{\Rightarrow} y$ and $x \overset{R*}{\Rightarrow} y$.

We write $x \not\Rightarrow y$ if there exists no derivation of y from x. Similarly we write $x \overset{*}{\not\Rightarrow} y$. A sentential derivation is any derivation for which $w_0 = S$.

A sentential form is any string $x \in (I \cup T)^*$ such that $S \Rightarrow x$. A *language* generated by a grammar G, is the set of strings,

$$L(G) = \{x \colon x \in T^* \text{ and } S \Rightarrow x\}.$$

Two grammars G_1 and G_2 are equivalent if $L(G_1) = L(G_2)$, written $G_1 \equiv G_2$. A grammar G is ambiguous if there exists a word $x \in L(G)$ for which at least two distinct left sentential derivations exist. Conversely a grammar is unambiguous if for each $x \in L(G)$ there exists a unique left sentential derivation.†

A cycle occurs in a grammar G, if there exists a derivation $X \Rightarrow uXv$, $X \in I$, $u, v \in (I \cup T)^*$. X is said to be cyclic. If $X \to uXv$ then X is directly cyclic, otherwise indirectly cyclic. X is left cyclic (right cyclic) if there exists a left derivation (right derivation)

$$X \overset{L}{\Rightarrow} Xv \ (X \overset{R}{\Rightarrow} uX).$$

An intermediate symbol X, is non-terminating if there exists no derivation $X \Rightarrow x$, $x \in T^*$. A rule $X \to w$, is non-terminating if X is non-terminating. An intermediate symbol X, is terminating if it is not non-terminating. The set $A_X = \{w \colon \text{all } w,\ X \to w \in P\}$ is

† This definition of ambiguity is equivalent to that usually given (cf. Ginsburg, 1966).

* Work carried out at the *University of Leeds, England*
Present address: *Courant Institute*, 251 *Mercer Street, New York, New York* 10012 *U.S.A.*

called the rule alternative set. Each w (or $X \to w$) is a rule alternative of X.

A rule is left literal if each rule alternative is of the form

$$X \to AX_1 \ldots X_n, A \in T, n \geqslant 0, X_i \in I \cup T.$$

A grammar is said to be admissible if either P is empty (and therefore $L(G)$ is empty, or for each $X \in I \cup T$ there exists at least one sentential derivation containing X and each $X \in I$ is terminating. Henceforth 'grammar' will usually mean admissible grammar.

Having introduced a basic terminology we now define normal and quasi-normal form grammars.

A grammar is in *normal form* if every production is of the form

$$X \to AX_1 \ldots X_n, n \geqslant 0, A \in T,$$
$$X, X_1, \ldots, X_n \in I.$$

A grammar is in *quasi-normal form* if $X_1, \ldots, X_n \in I \cup T$ in the above. This gives us our first trivial result, which we do not prove.

Lemma 1

To each quasi-normal form grammar there exists an equivalent normal form grammar.

We now define substitution, a process of which we will make much use. Given a grammar $G = (I, T, S, P)$ and a rule $Y \to Y_1 \ldots Y_n$, where $Y_i = X$ for some i, then we can construct the grammar $G_1 = (I, T, S, P_1)$ where

$$P_1 = (P - \{Y \to Y_1 \ldots Y_n\}) \cup$$
$$\{Y \to Y_1 \ldots Y_{i-1}wY_{i+1} \ldots Y_n\text{: all } w, X \to w \in P\}.$$

This process is called a substitution.

Lemma 2

In the above definition $G_1 \equiv G$ and substitution preserves unambiguity.

Proof: Consider a sentential form containing Y.

$$uYv \underset{G}{\Rightarrow} uY_1 \ldots Y_nv \underset{G}{\Rightarrow} uY_1 \ldots Y_{i-1}wY_{i+1} \ldots Y_nv,$$

say, where

$$u, v \in (I \cup T)^*, \quad \text{and}$$

w is a rule alternative of $X = Y_i$.

In G_1 we have, equivalently,

$$uYv \underset{G_1}{\Rightarrow} uY_1 \ldots Y_{i-1}wY_{i+1} \ldots Y_nv.$$

Only the order of the derivation has been changed, thus $L(G) \subseteq L(G_1)$. By a reverse argument $L(G_1) \subseteq L(G)$, therefore $G_1 \equiv G$. The preservation of unambiguity follows by a similar argument.

3. The normal form theorem

If G is a grammar then we construct a normal form grammar F, such that $L(F) = L(G) - \{\epsilon\}$ and if G is unambiguous then F is also. Using the well known result (Lemma 3) we only need consider ϵ-free grammars. Theorem 1 together with Lemmata 4 and 5 prove that if G is an ϵ-free non-left cyclic grammar then F exists and a construction is provided. Theorem 2 proves that an ϵ-free grammar can be transformed into an ϵ-free non-left cyclic grammar. Thus Theorems 1 and 2 together with Lemma 3 gives us Theorem 3, the normal form theorem.

Lemma 3

Given a grammar G then there exists an ϵ-free grammar G_1 (G_1 contains no rule of the form $X \to \epsilon$) such that $L(G) - \{\epsilon\} = L(G_1)$, and if G is unambiguous then G_1 is unambiguous.

Proof: See Ginsburg (1966, p. 38). The proof makes use of an iterative construction of which we give an informal treatment. Let $G = G^0$. Construct G^{i+1} from G^i as follows. If G_i contains no rule of the form $X \to \epsilon$ then $G^i = G_1$, otherwise for all $X, X \to \epsilon$ wherever X occurs on the right side of a production, replace that production with productions that omit X in all possible combinations. For example

$$Y \to uXvXw \text{ gives } Y \to uXvXw$$
$$Y \to uvXw$$
$$Y \to uXvw$$
$$Y \to uvw.$$

Then delete the rule $X \to \epsilon$; if X no longer appears on the left of any production, delete all productions that contain X. This construction is used below in example 1. Because of Lemma 3 we deal only with ϵ-free grammars in what follows. Theorem 1 gives us a method of obtaining an equivalent normal form grammar from an ϵ-free non-left cyclic grammar.

Lemma 4

Given G, an ϵ-free non-left cyclic grammar then there exists at least one $X \in I$ such that each rule alternative of X is left literal (such an X is also said to be left literal).

Proof: Let *Card* $(I) = n$ and assume the lemma to be false. This implies that for each $X \in I$ there exists at least one rule alternative of the form

$$X \to X_1w, X, X_1 \in I, w \in (I \cup T)^*.$$

It follows that we can construct the left derivation sequence

$$w_0 = X_0 \overset{L}{\Rightarrow} w_n,$$

where for each $i < n$

$$w_i = X_iv_i, w_{i+1} = X_{i+1}u_{i+1}v_i$$
$$X_i \to X_{i+1}u_{i+1} \in P; X_i \in I, w_i, u_i, v_i \in (I \cup T)^* \text{ all } i.$$

Because G is non-left cyclic $X_i \neq X_j$, $i \neq j$, $i, j \leqslant n$. Therefore $n + 1 = Card(\{X_i\}) = Card\ (I) = n$ is a contradiction. The lemma is true. Q.E.D.

Define a partition on I as follows

$$H = \{X\text{: } X \text{ is left literal, } X \in I\}$$
$$J = \{X\text{: } X \text{ is non-left literal, } X \in I\} \text{ then}$$
$$I = H \cup J, H \text{ and } J \text{ are disjoint and by Lemma 4 } H$$

is not empty.

Lemma 5

Given an ϵ-free non-left cyclic grammar G then either there exists at least one $X \in J$ such that each rule alternative is of the form

$$X \rightarrow X_1 w,\ X_1 \in H \cup T,\ w \in (I \cup T)^*, \text{ or } J \text{ is empty.}$$

Proof: If J is empty the lemma follows immediately, otherwise let $Card\,(J) = n$ and assume the lemma to be false. Using the same construction as in Lemma 4 the sequence $X_0, \ldots, X_n$ can always be found, such that $X_i \in J$, $X_i \neq X_j$, $i \neq j$, $i, j \leqslant n$. Otherwise the grammar would be left cyclic.
Then

$$n + 1 = Card\,(\{X_i\}) = Card\,(J) = n$$

as before. Q.E.D.

Theorem 1

Given G an ϵ-free non-left cyclic grammar then there exists a grammar G_1 which is in normal form and $G_1 \equiv G$. Further, if G is unambiguous then G_1 is unambiguous.

Proof: Using the partition previously defined, let $P_1 = P$, $H_1 = H$, $J_1 = J$ and $i = 1$ initially.
Define the algorithm.

Step 1. For each $X \in J_i$ where $X \rightarrow X_1 \ldots X_n \in P_i$, and $X_1 \in H_i$ perform the substitution for X_1 in X. This gives P_{i+1}.

Step 2. Form $H_{i+1} = H_i \cup \{X : X \in J_i, X \text{ is left literal}\}$, $J_{i+1} = J_i - H_{i+1}$ and $i = i + 1$.

Step 3. If J_i is empty then stop, otherwise repeat steps 1, 2 and 3.

By Lemma 4, H_1 exists and is not empty.
By Lemma 5, steps 1 and 2 always produce $H_{i+1} \supset H_i$, and in the case J is empty, $H = I$ and steps 1 and 2 do nothing.

The algorithm must terminate because I is a finite set, moreover it terminates in at most $Card\,(I) - 1$ steps because $Card\,(H_1) \geqslant 1$ and $Card\,(H_{i+1}) - Card\,(H_i) \geqslant 1$. By Lemma 1 there exists a normal form grammar equivalent to the quasi-normal form grammar produced above, let this be G_1. Having only used the process of substitution, by Lemma 2 G_1 is unambiguous if G is unambiguous. Q.E.D.

Corollary 1.1

Given G a non-left cyclic grammar then there exists a grammar G_1 in normal form such that $L(G) - \{\epsilon\} = L(G_1)$ and G_1 is unambiguous if G is unambiguous.

Proof: By Lemma 3 and the theorem. It is possible that the grammar G_1 obtained above is not an admissible grammar; it is simply a matter of deleting unused intermediate symbols from I and unused productions from P.

Corollary 1.2

Given G a non-left cyclic grammar then there exists a grammar G_1 which is admissible and in normal form, as above.

Example 1 Let $G = (I, T, S, P)$ where

$$I = \{Q, R, S\}$$
$$T = \{a, b, c, d\}$$
$$S = S$$
$$P = \{S \rightarrow b, S \rightarrow RQ, R \rightarrow cQ, R \rightarrow aR, Q \rightarrow d, Q \rightarrow \epsilon, Q \rightarrow R\}.$$

Step 1. Remove rule $Q \rightarrow \epsilon$, giving

$$P = \{S \rightarrow b, S \rightarrow RQ, S \rightarrow R, R \rightarrow cQ, R \rightarrow c, R \rightarrow aR, Q \rightarrow d, Q \rightarrow R\}$$

Step 2. Form H and J, $H = \{R\}$, $J = \{Q, S\}$

Step 3. Using the algorithm given in Theorem 1 generate a quasi-normal form grammar.

$$P_1 = P,\ H_1 = \{R\},\ J_1 = \{Q, S\},\ i = 1$$
$$P_2 = \{S \rightarrow b, S \rightarrow cQQ, S \rightarrow cQ, S \rightarrow cQ, S \rightarrow c, S \rightarrow aRQ, S \rightarrow aR, R \rightarrow cQ, R \rightarrow c, R \rightarrow aR, Q \rightarrow d, Q \rightarrow c, Q \rightarrow aR, Q \rightarrow cQ\}$$
$$H_2 = \{R\} \cup \{Q, S\}$$

J_2 is empty.

This grammar $G = (I, T, S, P_2)$, is in normal form; it only remains to remove one instance of the duplicated rule $S \rightarrow cQ$ whence the grammar is in admissible normal form. Having developed the normal form theorem for non-left cyclic grammars it remains to show that every grammar can be transformed into non-left cyclic form with preservation of unambiguity.

We use a formal version of the method used by Foster (1968) in his syntax improvement device. We will first discuss the method of removing left cycles by example. The exposition follows that of Woodward (1966) when discussing the work of Foster (1968).

A simple example is

$X \rightarrow Xa|b$, we use '|' to separate rule alternatives as in BNF (Naur, 1963). This implies

$$X \rightarrow b|ba|baa| \ldots$$
$$\underline{X \rightarrow bY}, \text{ say, where}$$
$$Y \rightarrow \epsilon|a|aa|aaa| \ldots \text{ giving}$$
$$\underline{Y \rightarrow aY|\,\epsilon.}$$

Woodward (1966) compares this problem to that of solving simultaneous equations in a non-commutative algebra. Cycles can embrace more than one rule, for example

$$\left.\begin{aligned} X_1 &\rightarrow X_1 a_{11} | X_2 a_{21} | b_1 \\ X_2 &\rightarrow X_1 a_{12} | X_2 a_{22} | b_2 \end{aligned}\right\} \qquad (1)$$

in 'matrix' notation, this becomes $\underline{X} \rightarrow \underline{X}\underline{\underline{a}}|\underline{b}$ giving $\underline{X} \rightarrow \underline{b}\,\underline{Y}$ and $\underline{Y} \rightarrow \underline{\underline{a}}\,\underline{Y}|\underline{\underline{E}}$, where a single underscore means a vector and double underscoring implies a matrix. We have

$$\underline{\underline{E}} = [\epsilon_{ij}],\ \epsilon_{ij} = \phi,\ i \neq j,$$
$$\epsilon,\ i = j,$$

(The semi-group zero is being used in the following manner:

$X \to a|\phi$ gives $X \to a$

$X \to \phi|a$ gives $X \to a$

$X \to \phi$ gives a redundant intermediate symbol.)

$$\underline{X} = [X_1\ X_2],$$

$$\underline{b} = [b_1\ b_2],$$

$$\underline{\underline{a}} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix},$$

and

$$\underline{\underline{Y}} = \begin{bmatrix} Y_{11} & Y_{12} \\ Y_{21} & Y_{22} \end{bmatrix}.$$

The solution in expanded form is

$$\left.\begin{array}{l} X_1 \to b_1 Y_{11} | b_2 Y_{21} \\ X_2 \to b_1 Y_{12} | b_2 Y_{22} \end{array}\right\} \qquad (2)$$

where

$$\left.\begin{array}{l} Y_{11} \to a_{11} Y_{11} | a_{12} Y_{21} | \epsilon \\ Y_{21} \to a_{21} Y_{11} | a_{22} Y_{21} \\ Y_{12} \to a_{11} Y_{12} | a_{12} Y_{22} \\ Y_{22} \to a_{21} Y_{12} | a_{22} Y_{22} | \epsilon \end{array}\right\}. \qquad (3)$$

It is not immediately obvious that this is a valid generalisation of the simple example. Consider the possible terminal strings over $\underline{a}$ and $\underline{b}$ that X_1 can have. X_1 can begin with b_1 or b_2.

$$X_1 \to X_1 a_{11} \Rightarrow X_1 a_{11}^m \Rightarrow b_1 a_{11}^m$$
$$\Rightarrow X_2 a_{21} a_{11}^m$$
$$X_2 a_{21} a_{11}^m \Rightarrow X_2 a_{22}^n a_{21} a_{11}^m \Rightarrow b_2 a_{22}^n a_{21} a_{11}^m$$
$$\Rightarrow X_1 a_{12} a_{22}^n a_{21} a_{11}^m.$$

It is fairly easily seen that the matrix solution produces identical strings over $\underline{a}$ and $\underline{b}$. Now consider the situation in which $a_{ij} = \phi$ for some i and j. For example let $a_{21} = \phi$. Then we have

$$X_1 \to X_1 a_{11} | b_1$$
$$X_2 \to X_1 a_{12} | X_2 a_{22} | b_2.$$

We get

$\underline{X} \to \underline{b}\,\underline{\underline{Y}}$ and

$\underline{\underline{Y}} \to \underline{\underline{a}}\,\underline{\underline{Y}} | \underline{\underline{E}}$ as before, giving

$$X_1 \to b_1 Y_{11} | b_2 Y_{21}$$
$$X_2 \to b_1 Y_{12} | b_2 Y_{22}.$$

Now we know that X_1 can only begin with b_1. Therefore we have the equation

$$b_2 Y_{21} = \phi$$

implying that as b_2 is not necessarily empty that $Y_{21} = \phi$. This appears to give the following general rule: if $a_{ij} = \phi$ then $Y_{ij} = \phi$ under the construction. However, if we consider the third-order case (see Appendix 3) we see that the following rule is the more general one. Assume the left cycle transformation has been carried out.

Reduction rule: for any Y_{ij} which is non-terminating define $Y_{ij} = \phi$ and repeat the reduction until no further non-terminating Y_{ij} is found.

The left cycle removal construction

Given a grammar $G = (I, T, S, P)$, define a partition of I into H and J, where

$H = \{X\colon X$ is left cyclic$\}$

$J = I - H$. If H is empty let G_1 be G.

Let the elements of H be $X_1, \ldots, X_n$, $n = Card\ (H)$, and re-order each rule alternative so that each X_i is in the form

$$X_i \to X_1 a_{1i} | X_2 a_{2i} | \ldots | X_n a_{ni} | b_i$$

collecting terms and introducing new productions as necessary. Where $a_{ji} \overset{*}{\not\Rightarrow} \epsilon$ for all i and j and possibly $a_{ji} = \phi$ for some i and j. Let the productions introduced during the collecting together of terms be $\{Z_k \to w_{kl}\}$. Let $G_1 = (I_1, T, S, P_1)$ where in the notation used above

$$I_1 = I \cup \{Y_{ij}\colon i, j \leqslant n\} \cup \{Z_k\}$$
$$P_1 = (P - \{X \to w : X \in H\})$$
$$\cup \{X_i \to b_j Y_{ji} : 1 \leqslant i, j \leqslant n\}$$
$$\cup \{Y_{ij} \to a_{ik} Y_{kj} : 1 \leqslant i, j, k \leqslant n\}$$
$$\cup \{Y_{ii} \to \epsilon : 1 \leqslant i \leqslant n\}$$
$$\cup \{Z_k \to w_{kl}\},$$

and where we apply the reduction rule to remove non-terminating Y_{ij}'s.

Notes: If $a_{ij} \overset{*}{\Rightarrow} \epsilon$ is allowed, we could have the following situation: $S \to S|a$, which under the transformation would still have a left cycle, because $a_{11} = \epsilon$. We therefore exclude this possibility. If $X \to Xb$ and $X \to Xaa \in P$ then this would be rewritten $X \to XZ$, $Z \to b$, $Z \to aa$, when collecting terms.

Theorem 2. The left cycle removal theorem

Let $G = (I, T, S, P)$ be an admissable ϵ-free grammar; using the construction defined above we obtain G_1. Then G_1 is non-left cyclic, $G_1 \equiv G$ and if G is unambiguous then G_1 is unambiguous.

Proof: If G_1 is the same as G we have nothing to prove; otherwise:

Part 1. G_1 is non-left cyclic. If $X \in J$ then the transformation has no effect and by definition X is non-left cyclic.

Consider $X_i \in H$, then after the transformation we have

$$X_i \to b_1 Y_{1i} | \ldots | b_i Y_{ii} | \ldots,$$

therefore when accepting (or generating) an X_i we first accept (or generate) a b_j. Now at least one $b_j \neq \phi$ exists (otherwise G would be non-terminating), and since G is ϵ-free, $b_j \overset{*}{\not\Rightarrow} \epsilon$.

By the construction the b_j are non-left cyclic, therefore the X_i are non-left cyclic in the grammar G_1.

Consider $Y_{ij} \in I_1$. The $Y_{ij} \notin I$; therefore since $a_{ik} \overset{*}{\not\Rightarrow} \epsilon$ does not contain Y_{ij}, each Y_{ij} is non-left cyclic. Similarly each Z_i is non-left cyclic.

Therefore G_1 is non-left cyclic.

Part 2. $G \equiv G_1$. Let $Card\ (H) = n$ and let

$$F = \{(i, j) : a_{ij} \neq \phi,\ 1 \leqslant i, j \leqslant n\},$$
$$E = \{i : b_i \neq \phi,\ 1 \leqslant i \leqslant n\}.$$

(*a*) $L(G) \subseteq L(G_1)$.

It suffices to show that a sentential form uX_iv gives rise to the same derivation in G_1 as in G. If $X_i \in J$ this is trivial, because we have not altered these productions. Let $X_i \in H$, then $X_i \to X_i a_{1i}| \ldots |X_i a_{ii}| \ldots |b_i$, giving

$$uX_iv \underset{G}{\Rightarrow} uX_{j_1}a_{j_1i}v, \text{ where } (j_1, i) \in F$$

$$\underset{G}{\Rightarrow} uX_{j_2}a_{j_2j_1}a_{j_1i}v, \text{ where } (j_2, j_1) \in F$$

$$\underset{G}{\Rightarrow} uX_{j_m}a_{j_mj_{m-1}} \ldots a_{j_1i}v,$$

where

$$(j_k, j_{k-1}) \in F, 2 \leqslant k \leqslant m.$$

This derivation can be terminated at any time by replacing X_{j_m} by $b_{j_m} \in E$. We can construct the same derivation in G_1,

$$uX_iv \underset{G_1}{\Rightarrow} ub_{j_m}Y_{j_mi}v, j_m \in E$$

$$\underset{G_1}{\Rightarrow} ub_{j_m}a_{j_mj_{m-1}}Y_{j_{m-1}i}v$$

$$\underset{G_1}{\Rightarrow} ub_{j_m} \ldots a_{j_2j_1}a_{j_1i}Y_{ii}v$$

Choosing $Y_{ii} \to \epsilon$ as the next production to be applied, we then have the same derivation in G_1 as in G, therefore $L(G) \subseteq L(G_1)$. We have excluded from consideration the introduction of the $Z_k \to w_{kl}$ productions and we have also assumed the productions of G have been reordered and terms collected. It is fairly easy to see that this simplification does not affect the result in any way. Note that Y_{ij} can terminate if and only if $i = j$.

(*b*) $L(G_1) \subseteq L(G)$.

Using the reverse of the above argument we can show that any derivation of uXv in G_1 can be constructed in G. Note that Y_{ij} and Z_i can only appear in derivations from an $X \in H$. By the use of the reduction rule during the construction all putative non-terminating derivations in G_1 have been eliminated.

Therefore $G_1 \equiv G$.

Part 3. G is unambiguous implies G_1 is unambiguous. Assume G_1 is ambiguous. It must be ambiguous within the transformed productions, otherwise G would be ambiguous. Treat the b_j and a_{ij} as if they were terminal strings, without loss of generality; we can then use left derivations. We can only enter the transformed productions through an X_i. Therefore there exists an X_i which gives rise to two left derivations over G_1.

Consider the derivation

$$uX_iv \overset{L}{\underset{G_1}{\Rightarrow}} ub_{j_1}a_{j_1j_2} \ldots a_{j_mi}v \text{ (by Part 2)}$$
$$u \in T^*, v \in (I \cup T)^*.$$

Then either

(i) $b_{j_1} \ldots a_{j_pi_{p+1}} = b_{l_1} \ldots a_{l_pi_{p+1}}, 0 \leqslant p \leqslant m,$
(if $p = m, j_{p+1} = i$)

or

(ii) $a_{j_kj_{k+1}} \ldots a_{j_pl_{p+1}} = a_{j_kl_{k+1}} \ldots a_{l_pl_{p+1}}, 1 \leqslant k < p.$

Both cases imply that G is ambiguous (by Part 2). Therefore G_1 is unambiguous if G is unambiguous. We are now in a position to state and prove the normal form theorem.

Theorem 3. *The normal form theorem*

Given an admissible context-free grammar G there exists an admissible grammar G_1, whose rules are of the form

$$X \to AX_1 \ldots X_n, A \in T, X, X_1, \ldots, X_n \in I, n \geqslant 0,$$

such that $L(G) - \{\epsilon\} = L(G_1)$ and G_1 is unambiguous if G is unambiguous.

Proof: By Lemma 3 we can remove all rules $X \to \epsilon$, giving $L(G_2) = L(G) - \{\epsilon\}$.

By Theorem 2 we can remove left cycles giving $G_3 \equiv G_2$.

By Corollary 1.2 we can construct a grammar G_1, in normal form, such that $G_1 \equiv G_3$, and G_1 is admissible.

Therefore $L(G_1) = L(G) - \{\epsilon\}$ and G_1 is in normal form. Each step preserves unambiguity, therefore G_1 is unambiguous if G is unambiguous.

A grammar is in *reverse normal form* if its rules are of the form $X \to X_1 \ldots X_nA$, $A \in T$, $X, X_i \in I$ and $n \geqslant 0$. The proof of Theorem 3 is symmetric, so we obtain the result.

Corollary 3.1

Given a grammar G there exists a grammar G_1 in reverse standard form, $L(G_1) = L(G) - \{\epsilon\}$ and if G is unambiguous G_1 is unambiguous also.

We now give a worked example.

Example 2

This is the example used by Sheila Greibach (1965).

Let $\quad G = (I, T, S, P)$, where
$\quad I = \{S, Y\}$,
$\quad T = \{a, b, c, d\}$,
$\quad S = S$

and

$$P = \{S \to a, S \to Sb, S \to Ya, Y \to c, Y \to YYd\}.$$

Rewrite P as

$$S \to Sb|Ya|a$$
$$Y \to YYd|c, \text{ then } H = \{S, Y\}.$$

Using the construction of Theorem 2 we can rewrite this as

$$S \to aY_{11}|cY_{21}$$
$$Y \to \quad cY_{22}$$
$$Y_{11} \to bY_{11}|\epsilon$$
$$Y_{21} \to aY_{11}|YdY_{21}, Y_{12} = \phi \text{ as } a_{12} = \phi,$$
$$Y_{22} \to YdY_{22}|\epsilon.$$

If we now remove the ϵ-rules we obtain

$$S \to a|aY_{11}|cY_{21}$$
$$S \to c|cY_{22}$$
$$Y_{11} \to b|bY_{11}$$
$$Y_{21} \to a|aY_{11}|YdY_{21}$$
$$Y_{22} \to Yd|YdY_{22}.$$

Having removed left cycles we now substitute for Y in rules Y_{21} and Y_{22}.

$$Y_{21} \to a|aY_{11}|cdY_{21}|cY_{22}dY_{21}$$
$$Y_{22} \to cd|cY_{22}d|cdY_{22}|cY_{22}dY_{22}.$$

We can remove the rule $Y \rightarrow c|cY_{22}$ and letting $Y_{11} = A$, $Y_{21} = B$, $Y_{22} = C$ we obtain

$$G_1 = (I_1, T, S, P_1)$$

where

$$I_1 = \{S, A, B, C\}$$

and

$$\begin{aligned} P_1 = \{&S \rightarrow a,\ S \rightarrow aA,\ S \rightarrow cB,\\ &A \rightarrow b,\ A \rightarrow bA,\\ &B \rightarrow a,\ B \rightarrow aA,\ B \rightarrow cdB,\ B \rightarrow cCdB,\\ &C \rightarrow cd,\ C \rightarrow cCd,\ C \rightarrow cdC,\ C \rightarrow cCdC\}. \end{aligned}$$

This is in admissible quasi-normal form. To obtain normal form introduce the production $D \rightarrow d$, and substitute D for d everywhere in P_1. This can be reduced to the same form as Greibach (1965) by noting that $S \rightarrow cB$ can be replaced by

$$S \rightarrow ca,\ S \rightarrow caA,\ S \rightarrow ccdB,\ S \rightarrow ccCdB$$

and that $S \rightarrow ccdB|ccCdB$ can be replaced by

$$S \rightarrow cCa|cCaA.$$

There is a small error in the example in Greibach (1965, p. 48) because of a mistake in the substitution of Y in one of the rules.

4. Discussion

Sheila Greibach (1965) makes two apparently arbitrary restrictions on context-free grammars. These are

(1) they should be ϵ-free,

(2) their rules should be of the form

$$X \rightarrow AX_1 \dots X_n, \quad n \geqslant 1$$

or $$X \rightarrow a,\ a \in T,\ A,\ X_i \in (I \cup T),\ X \in I.$$

(1) We have made the same restriction by use of Lemma 3.

(2) Theorem 1 does not depend on these assumptions but Theorem 2 does. For example, consider equations (1) in which $a_{11} = \epsilon$, say, then we have an example of infinite recursion and ambiguity, because $X_1 \rightarrow X_1$ is obtained. Also in equations (3) we have

$$Y_{11} \rightarrow Y_{11}|a_{12}Y_{21}|\epsilon.$$

This is equivalent to allowing $n \geqslant 0$ rather than $n \geqslant 1$ in the above. We see that the two assumptions are necessary (we use the condition $a_{ij} \overset{*}{\not\Rightarrow} \epsilon$ instead of restriction 2). One further point of difference is that Greibach removes left cycles and maintains the ϵ-free property. This is because she uses a transformation similar to

$$\underline{X} \rightarrow \underline{b}|\underline{b}\,\underline{\underline{Y}} \dots \text{(Wood)}$$

and $$\underline{\underline{Y}} \rightarrow \underline{\underline{a}}\,\underline{\underline{Y}}|\underline{\underline{a}}, \text{ rather than}$$

$$\underline{X} \rightarrow \underline{b}\,\underline{\underline{Y}} \dots \text{(Foster)}$$

and $$\underline{\underline{Y}} \rightarrow \underline{\underline{a}}\,\underline{\underline{Y}}|\underline{\underline{E}}$$

as we do in the left cycle removal construction. The reason for our choice is that it increases the 'left-factoredness' of a grammar (see Wood, 1968b; Woodward, 1966), this is important in the syntax improvement device (Foster, 1968). In Appendix 1 it is shown that the two transformations are equivalent. However because the Foster transformation introduces rules of the form $X \rightarrow \epsilon$ into the grammar, it is better to use the Wood transformation if it is required that the non-left cyclic grammar should also be ϵ-free. In Wood (1968b) motivation is given for choosing the Foster transformation.

The Greibach transformation can be described as follows: If there exists no mutual cycle between X_i and X_j then there is no necessity to treat X_i and X_j as if a mutual cycle does exist. This means we can group the term involving X_i (in the rule $X_j \rightarrow X_i a_{ij}$) with b_j, and vice versa, whilst we perform the transformation on X_i and X_j. Thus while performing the remainder of the transformation we resume the *status quo*. In Appendix 1 we show the equality of a particular example of the Greibach and Wood transformation. Rather than perform the Foster transformation simultaneously on all elements of H we could split H into mutually disjoint sets, where

$$H_i(X_i) = \{Y : X_i \overset{L}{\Rightarrow} Yw,\ Y \overset{L}{\Rightarrow} X_i v,\ w, v \in (I \cup T)^*,\ X_i,\ Y \in H\}. \text{ Then}$$

$$H = \bigcup_{i=1}^{k} H_i(X_i), \quad 1 \leqslant k \leqslant Card(H).$$ The left cycle removal construction could then be applied independently to each $H_i(X_i)$.

By *ϵ-normal form* we mean a grammar whose rules are of the form

$$X \rightarrow aX_1 \dots X_n, \quad n \geqslant 0$$

or $$X \rightarrow \epsilon,\ X,\ X_1 \in I,\ a \in T.$$

Lemmata 4 and 5 and Theorem 2 can be proved by similar methods if the 'ϵ-free' restriction is removed. We can then obtain the ϵ-normal form theorems equivalent to Theorems 1 and 3 (see Appendix 2).

Acknowledgements

I wish to thank Dr. M. Wells, my supervisor, for his constant encouragement during the period this work was performed. I also thank the referee for discovering an error in an earlier draft of Theorem 2 and for suggesting the correction.

References

FOSTER, J. M. (1968). A syntax improving program, *Computer Journal*, Vol. 11, p. 31.

GINSBURG, S. (1966). *The mathematical theory of context-free languages*, London: McGraw-Hill Book Co.

GREIBACH, SHEILA A. (1965). A new normal-form theorem for context-free phrase structure grammars, *J. Assoc. Comp. Mach.*, Vol. 12, p. 42.

MCKEEMAN, W. M. (1966). *An approach to computer language design*, Technical Report CS48, Computer Science Dept., Stanford University.

NAUR, P., *et al.* (1963). Revised report on the algorithmic language ALGOL 60, *Computer Journal*, Vol. 5, p. 349.

WOOD, D. (1968a). *On generalized interpretive schemes for programming languages*, Doctoral dissertation, Leeds University.

WOOD, D. (1968b). The theory of left factored languages (in preparation).

WOODWARD, P. M. (1966). *A note on Foster's syntax improving device*, RRE Memorandum No. 2352, Royal Radar Establishment, Malvern.

Appendix 1

We illustrate the derivation of both the Wood and Greibach transformations by example. The Foster transformation on

$$\underline{X} \to \underline{X}\,\underline{\underline{a}}|\underline{b}$$

is $\quad \underline{X} \to \underline{b}\,\underline{\underline{Y}}$ and $\underline{\underline{Y}} \to \underline{\underline{a}}\,\underline{\underline{Y}}|\underline{\underline{E}}$.

In particular we have

$$\begin{aligned} X_1 &\to X_1a_{11}|X_2a_{21}|b_1 \\ X_2 &\to X_1a_{12}|X_2a_{22}|b_2 \end{aligned}$$

giving

$$\begin{aligned} X_1 &\to b_1Y_{11}|b_2Y_{21} \\ X_2 &\to b_1Y_{12}|b_2Y_{22} \\ Y_{11} &\to a_{11}Y_{11}|a_{12}Y_{21}|\epsilon \\ Y_{21} &\to a_{21}Y_{11}|a_{22}Y_{21} \\ Y_{12} &\to a_{11}Y_{12}|a_{12}Y_{22} \\ Y_{22} &\to a_{21}Y_{12}|a_{22}Y_{22}|\epsilon. \end{aligned}$$

Removing the ϵ-rules this gives

$$\begin{aligned} X_1 &\to b_1|b_1Y_{11}|b_2Y_{21} \\ X_2 &\to b_1Y_{12}|b_2|b_2Y_{22} \\ Y_{11} &\to a_{11}|a_{11}Y_{11}|a_{12}Y_{21} \\ Y_{21} &\to a_{21}|a_{21}Y_{11}|a_{22}Y_{21} \\ Y_{12} &\to a_{11}Y_{12}|a_{12}|a_{12}Y_{22} \\ Y_{22} &\to a_{21}Y_{12}|a_{22}|a_{22}Y_{22}, \end{aligned}$$

which in the notation above is

$$\underline{X} \to \underline{b}|\underline{b}\,\underline{\underline{Y}} \text{ and } \underline{\underline{Y}} \to \underline{\underline{a}}|\underline{\underline{a}}\,\underline{\underline{Y}},$$

i.e. the Wood transformation. Assume $a_{21} = \phi$, we then have

$$\begin{aligned} X_1 &\to X_1a_{11}|b_1 \\ X_2 &\to X_2a_{22}|(b_2|X_1a_{12}) \end{aligned}$$

giving

$$\left.\begin{aligned} X_1 &\to b_1|b_1Y_{11} \\ X_2 &\to b_2|X_1a_{12}|b_2Y_{22}|X_1a_{12}Y_{22} \\ Y_{11} &\to a_{11}|a_{11}Y_{11} \\ Y_{22} &\to a_{22}|a_{22}Y_{22} \end{aligned}\right\} \quad (1)$$

under the Greibach transformation. Under the Wood transformation we have

$$\left.\begin{aligned} X_1 &\to b_1|b_1Y_{11} \\ X_2 &\to b_1Y_{12}|b_2|b_2Y_{22} \\ Y_{11} &\to a_{11}|a_{11}Y_{11} \\ Y_{12} &\to a_{11}Y_{12}|a_{12}|a_{12}Y_{22} \\ Y_{22} &\to a_{22}|a_{22}Y_{22}. \end{aligned}\right\} \quad (2)$$

Comparing (1) and (2) we see that they differ as follows

(1) $\quad X_2 \to X_1a_{12}|X_1a_{12}Y_{22}$

(2) $\quad X_2 \to b_1Y_{12}$ and $Y_{12} \to a_{11}Y_{12}|a_{12}|a_{12}Y_{22}$.

Considering the strings generated by X_2 in each case we have

(1) $\quad b_1a_{11}^na_{12}$ or $b_1a_{11}^na_{12}Y_{22}$, $n \geqslant 1$

and

(2) $\quad b_1a_{11}^na_{12}$ or $b_1a_{11}^na_{12}Y_{22}$, $n \geqslant 1$.

Therefore the two transformations are equivalent.

Appendix 2

An intermediate symbol X, is *ϵ-left literal* if at most one rule alternative is $X \to \epsilon$ and the remaining rule alternatives are left literal.

A grammar is in *quasi-ϵ-normal form* if each rule is ϵ-left literal; ϵ-normal form can be defined in a similar manner.

Lemma A4

Given G a non-left cyclic grammar then there exists at least one $X \in I$ such that each rule alternative of X is ϵ-left literal.

Define a partition on I as follows

$$H = \{X : X \text{ is } \epsilon\text{-left literal, } X \in I\}$$

$$J = I - H, \ H \text{ is not empty by Lemma A4.}$$

Lemma A5

Given G a non-left cyclic grammar then either there exists at least one $X \in J$ such that each rule alternative of X is of the form

$$X \to X_1w \text{ or } X \to \epsilon,\ X_1 \in H \cup T,\ w \in (I \cup T)^*,$$

or J is empty.

The proofs of Lemmata A4 and A5 follow those of Lemmata 4 and 5 exactly.

Theorem A1

Given a non-left cyclic grammar G, then there exists a grammar G_1 which is in ϵ-normal form, and $G_1 \equiv G$. Further if G is unambiguous G_1 is unambiguous.

Proof: Using the algorithm defined in Theorem 1, the only major difference in the proof is that step 2 may not always generate an $H_{i+1} \supset H_i$. For example, let $Y \in J_i$ such that

$$Y \to Y_1 \ldots Y_m,\ Y_1 \in H_1 \cup T.$$

If Y_1 is left literal then on substitution Y is at least ϵ-left literal. If, however, Y_1 is ϵ-left literal then on substitution we get

$$Y \to wY_2 \ldots Y_m$$

$$Y \to Y_2 \ldots Y_m$$

or $\quad Y \to \epsilon.$

It can be proved by methods similar to those used in Lemmata 4 and 5 that there exists a $k > 0$ such that

$$H_i \subset H_{i+k}$$

and $\quad H_i = H_{i+1} = \ldots = H_{i+k-1}$

where $\quad k \leqslant \max\{|w| : Q \to w \in P\}.$

An upper bound on the number of iterations needed to complete the process is

$$\sum_{\text{all } w,\ X \to w \in P} |w|.$$

Theorem A2. *The left cycle removal theorem*

Let $G = (I, T, S, P)$ be an admissible grammar; using the construction defined in Section 3 we obtain G_1. Then G_1 is non-left cyclic, $G_1 \equiv G$ and if G is unambiguous then G_1 is unambiguous.

Proof: In the proof of Theorem 2 we make use of $b_i \overset{*}{\not\Rightarrow} \epsilon$ only in Part 1. Consider $X_j \in H$, then after the transformation we have

$$X_j \rightarrow b_1 Y_{1j} | \ldots | b_i Y_{ij} | \ldots,$$

therefore when accepting (or generating) an X_j we first accept (or generate) a b_i. Now at least one $b_i \neq \phi$ exists, if $b_i \overset{*}{\Rightarrow} \epsilon$, then

$$X_j \rightarrow Y_{ij} \text{ for all } j,\ 1 \leqslant j \leqslant n,$$

which could lead to a left cycle unless we restrict each a_{ij} as follows.

Restriction: if $b_i \overset{*}{\Rightarrow} \epsilon$ then for each a_{ik}, $1 \leqslant k \leqslant n$,

$$a_{ik} \overset{*}{\not\Rightarrow} X_m u,\ u \in (I \cup T)^*,\ 1 \leqslant m \leqslant n.$$

With this restriction the proof of Part 1 of the theorem follows immediately.

Consider the situation in which the above restriction no longer holds.

For example, with the grammar $G = (I, T, S, P)$, where

$$I = \{X_1, X_2\},$$
$$T = \{a, b, p\},$$
$$S = X_1,$$

and $P = \{X_1 \rightarrow X_1 X_2 a | b | \epsilon,\ X_2 \rightarrow X_1 p\}$.

$$X_1 \rightarrow b Y_{11} | Y_{11}$$
$$X_2 \rightarrow b Y_{12} | Y_{12}$$
$$Y_{11} \rightarrow X_2 a Y_{11} | \epsilon$$
$$Y_{12} \rightarrow X_2 a Y_{12} | p,$$

we can see that X_2 still has a left cycle. The reason it was not removed is that although this left cycle existed in the original grammar, it was hidden. Apply the construction again, first substituting for Y_{12} in $X_2 \rightarrow Y_{12}$. We obtain

$$X_2 \rightarrow X_2 a Y_{12} | (p | b Y_{12})$$

giving $X_2 \rightarrow pZ | b Y_{12} Z$

$$Z \rightarrow a Y_{12} Z | \epsilon.$$

We have now removed the left cycle from X_2, thus the grammar is now completely non-left cyclic. This leads us to propose the following conjecture: for every grammar G there exists a finite integer k such that after k successive applications of the left cycle removal construction the grammar so obtained is non-left cyclic, and this is the least such k. However, consider the following grammar:

$$G = (\{S\}, \{a\}, S, \{S \rightarrow SSS | a | \epsilon\})$$

which generates the set $\{a\}^*$. There does not exist any finite k which satisfies the above conjecture. This grammar is ambiguous ('infinitely' ambiguous). We could exclude such cases by the addition of 'an unambiguous grammar G' to the above conjecture. Note that the above grammar would also be excluded from Theorem 2 because $SS = a_{11} \overset{*}{\not\Rightarrow} \epsilon$ exists. Thus amend the conjecture to emphasize the need for each $a_{ij} \overset{*}{\not\Rightarrow} \epsilon$.

Lemma 5

All intermediate symbols which are left cyclic in a grammar G will, whether open or hidden, be included in the set H under the left cycle removal construction.

Proof: This immediately follows from the definition of left cycles.

We define the *left cycle order* LCO, of a grammar as the integer k specified in the conjecture above, where each $a_{ij} \overset{*}{\not\Rightarrow} \epsilon$. That the LCO of a grammar (in which the restriction $a_{ij} \overset{*}{\not\Rightarrow} \epsilon$ holds) is finite could perhaps be derived intuitively from the application of Lemma 3 and Theorem 2 to the grammar G (i.e. first remove ϵ-rules, then do the left cycle removal construction).

Theorem 4

If $G = (I, T, S, P)$ is an admissible grammar and under the construction no $a_{ij} \overset{*}{\Rightarrow} \epsilon$ then there exists a grammar G_1 which is non-left cyclic, $G_1 \equiv G$ and if G is unambiguous G_1 is unambiguous.

Proof: We need to show that such a grammar has an LCO; if it has the proof follows easily. By Lemma 5 both hidden and open cyclic intermediate symbols will be included in H. This means that each a_{ij} is of the form:

$$X_{l_1} X_{l_2} \ldots X_{l_p} w,\ 1 \leqslant l_k \leqslant n,\ 1 \leqslant k \leqslant p,$$
$$p \geqslant 0,\ w \in (I \cup T)^*.$$

If $a_{ij} = Xw$, $X \in I - H$, $w \in (I \cup T)^*$ then X cannot be a member of a left cycle, because if it were it would be in H. Therefore only leading intermediate symbols which are in H are of interest. Consider the maximum of p over all a_{ij}. We define $m = p + 1$ to be the left cycle order of G.

If $m = 1$ then, by Theorem A2, G_1 exists and the theorem follows trivially. Otherwise at each stage of the transformation we remove the open left cycles and disclose some hidden left cycles (i.e. each application of the left cycle removal construction). Now by the example above we see that we reduce the LCO of a grammar by one after each transformation. Thus after $m - 1$ transformations the LCO is one, we can then apply Theorem A2. The proof of equivalence and preservation of unambiguity follows by induction on the LCO.

Theorem A3. *The ϵ-normal form theorem*

Given an admissible context-free grammar G there exists an admissible grammar G_1 in ϵ-normal form such that $G_1 \equiv G$ and G_1 is unambiguous if G is unambiguous.

Proof: Under the conditions of Theorem 4 we can remove left cycles giving $G_2 \equiv G$. By Theorem A1 we can construct a grammar G_1, in ϵ-normal form, where G_1 is admissible. Therefore $G_1 \equiv G$, G_1 is in ϵ-normal form and as each step preserves ambiguity G_1 is unambiguous if G is unambiguous. By Corollary 1.2, the result then follows.

Appendix 3

Consider the effect of a particular $a_{ij} = \phi$ in the left cycle removal construction. Without loss of generality choose $a_{31} = \phi$ in the third-order situation. Our original equations are:

$$\begin{aligned} X_1 &\to X_1a_{11}|X_2a_{21}|b_1 \\ X_2 &\to X_1a_{12}|X_2a_{22}|X_3a_{32}|b_2 \\ X_3 &\to X_1a_{13}|X_2a_{23}|X_3a_{33}|b_3. \end{aligned}$$

Under the construction we obtain:

$$\begin{aligned} X_1 &\to b_1Y_{11}|b_2Y_{21}|b_3Y_{31} \\ X_2 &\to b_1Y_{12}|b_2Y_{22}|b_3Y_{32} \\ X_3 &\to b_1Y_{13}|b_2Y_{23}|b_3Y_{33} \end{aligned}$$

where

$$\begin{aligned} Y_{11} &\to a_{11}Y_{11}|a_{12}Y_{21}|a_{13}Y_{31}|\epsilon \\ Y_{12} &\to a_{11}Y_{12}|a_{12}Y_{22}|a_{13}Y_{32} \\ Y_{13} &\to a_{11}Y_{13}|a_{12}Y_{23}|a_{13}Y_{33} \\ Y_{21} &\to a_{21}Y_{11}|a_{22}Y_{21}|a_{23}Y_{31} \\ Y_{22} &\to a_{21}Y_{12}|a_{22}Y_{22}|a_{23}Y_{32}|\epsilon \\ Y_{23} &\to a_{21}Y_{13}|a_{22}Y_{23}|a_{23}Y_{33} \\ Y_{31} &\to a_{31}Y_{11}|a_{32}Y_{21}|a_{33}Y_{31} \\ Y_{32} &\to a_{31}Y_{12}|a_{32}Y_{22}|a_{33}Y_{32} \\ Y_{33} &\to a_{31}Y_{13}|a_{32}Y_{23}|a_{33}Y_{33}|\epsilon. \end{aligned}$$

If we now applied the simple rule $a_{ij} = \phi$ implies $Y_{ij} = \phi$, we find that X_1 could not begin with b_3 after the construction, whereas before the construction it could begin with b_3. Therefore carrying through the consequence of $a_{31} = \phi$ we delete the alternatives $a_{31}Y_{1i}$, $i = 1, 2, 3$. If we now also assume $a_{32} = \phi$, we also delete $a_{32}Y_{2i}$, $i = 1, 2, 3$. However we are then left with two non-terminating rules, namely

$$Y_{31} \to a_{33}Y_{31}$$

and

$$Y_{32} \to a_{33}Y_{32}.$$

In this situation, however, we must delete these rules by defining Y_{31} and Y_{32} to be ϕ. If we did not do this X_1 and X_2 would have non-terminating alternatives, corresponding to the fact that X_1 and X_2 cannot begin with b_3 if $a_{31} = a_{32} = \phi$. This leads to the following rule:

Reduction rule: For any Y_{ij} which is non-terminating define $Y_{ij} = \phi$ and repeat the reduction until no further Y_{ij} is non-terminating.

Book Review

A Mathematical Theory of Systems Engineering, by A. Wayne Wymore, 1967; 353 pages. (London and New York: *John Wiley and Sons Limited*, 150s.)

As a theoretical text on systems engineering this book has a relatively unique feature: it uses set theory rather than classical algebra. It aims at a general theory of systems and succeeds in so far as it treats data processing automata, control and man-machine systems between the same covers. This is a refreshing standpoint for control engineers, but disappointing for those who look for a direct application. Set theory provides a weak mathematical structure which, due to its weakness, embraces many systems while allowing little scope for useful manipulation. Indeed, the author makes no claims in this respect and is happy to provide the tools, perhaps merely a language, whereby systems may be described.

The *introductory chapter* makes the point that rigour is important for systems that involve information and its communication. These cannot be tackled intuitively in the same way as systems that have physically tangible parameters. The chapter also presents an introduction to the set theory that is used in the rest of the book. Chapter 2 defines the *elements of the general theory.* Here control concepts of state space appear amid tools of automata theory such as admissible input sequences, semigroups and Turing machines.

The third chapter is on *modelling* and I consider it to be a highlight of the book. Proceeding from the premise that '. . . modelling is an art . . .' the author asserts his artistry by producing models for a wide variety of systems. These include computing elements, a widget factory, a watershed and a human-tracking experiment. The introduction of a pseudo-computer language is interesting since it underlines the fact that the aim of modelling is, usually, the design of a computer simulation program. Chapter 4 is on the *comparison of systems.* Laws of homomorphic mapping are used to generate equivalence classes as is customary in the minimisation of sequential machines. Chapter 5, *coupling of systems*, deals with cascading and feedback. Here the author is led to the assertion that a system with no input is no system. This is a pity since it excludes autonomous machines which are generally interesting systems.

Chapter 6, *subsystems and components*, uses concepts developed in connection with sequential machines by Gill, Hartmanis and Stearns for the decomposition of a system into elements. The discussion on duality in systems and the meaning of observability and controllability is worth noting. Chapter 7 is on *discrete systems* and provides an interesting link between programming and state transition tables.

The book is well organised. The mathematical and the discussion sections are clearly set out. However, even though the jacket claims that all the necessary mathematical foundations are included, a reader with no knowledge of set jargon may find it difficult to follow.

IGOR ALEKSANDER (Canterbury)