A note on left factored languages

T. Komor

INFELOR Systems Engineering Institute, 1281 Budapest, P.O.B. 10, Hungary

In the papers of Wood (1969, 1970) left factored (LF) grammars and languages are defined. In this paper we show that Lemma 8 in Wood (1970) does not hold, and therefore the proof of Lemma 6 is incorrect. We prove a theorem which gives a necessary and sufficient condition for an unambiguous grammar to be an LF grammar. On the basis of the theorem we give an example of a language, which can be used in the proof of Lemma 6. In Section 5 we given an extension to Foster's SID (see Foster, 1968), which transforms context-free grammars to LF form.

(Received September 1972)

Terminology

A context-free grammar G is a 4-tuple $G = (M, T, S, P)$, where M is a finite set of metasymbols, T is a finite set of terminal symbols, $S \in M$ is the sentence symbol and P is a finite set of rules of the form $A \to \phi$, $A \in M$, $\phi \in V^*$, $V = T \cup M$. Further, by grammar we will understand context-free grammar and the use of G as a grammar assumes implicitly $G = (M, T, S, P)$.

Usually we denote terminal symbols by $a, b, c, \ldots$; strings of terminals by x, y; metasymbols by $S, A, B, C, \ldots$; symbols from V by α; finally, strings from V^* by other Greek letters.

If G is a grammar, then by $G_\#$ will denote the grammar $G_\# = (M_\#, T_\#, S_\#, P_\#)$, where $M_\# = M \cup \{S_\#\}$, $T_\# = T \cup \{\#\}$, $P_\# = P \cup \{S_\# \to S\#\}$ (it is assumed, that $S_\# \notin M$ and $\# \notin T$).

Occasionally we use BNF style notation for rule alternatives, i.e. $A \to \phi_1|\phi_2|\ldots|\phi_n$. The set $R_A = \{\phi | A \to \phi \in p\}$ is called the rule alternative set.

We write $\phi \underset{G}{\Rightarrow} \psi$ if $\phi = xA\eta$, $\psi = x\omega\eta$ and $A \to \omega$ is a rule of the grammar G. We say ψ is derived from ϕ, denoted $\phi \overset{*}{\underset{G}{\Rightarrow}} \psi$, if a sequence of η_i exists, $0 \leqslant i \leqslant n$, such that $\psi = \eta_0 \underset{G}{\Rightarrow} \eta_1 \underset{G}{\Rightarrow} \ldots \underset{G}{\Rightarrow} \eta_n = \psi$, the strings η_i form a derivation. We write $\phi \Rightarrow \psi$ if G is understood. Note that we consider only left-derivations. A language generated by a grammar G is $L(G) = \{x | x \in T^*, S \overset{*}{\Rightarrow} x\}$.

We assume that the notions of admissible and unambiguous grammar are known (they are defined for example in Ginsburg (1966)).

The following notation is needed. Let $\phi = \alpha_1\alpha_2 \ldots \alpha_k$, where α_i are symbols in some alphabet, then $|\phi| = k$. For $\wedge$—the empty string—$|\wedge| = 0$. By LM_j we denote the first j symbols of a string, that is $LM_j(\phi) = \alpha_1\alpha_2 \ldots \alpha_j$ if $j \leqslant k$ and $LM_j(\phi) = \phi$ for $j > k$.

Let T be an alphabet, $T(\phi) = k$ if $LM_k(\phi) \in T^*$ but $LM_{k+1}(\phi) \notin T^*$. We denote the empty set by ϕ.

The left terminal set of a rule $A \to \omega$, denoted $[A, \omega]$ with respect to a grammar G we describe with the help of the corresponding grammar $G_\# = (M_\#, T_\#, S_\#, P_\#)$

$[A, \omega] = \{a | S_\# \overset{*}{\Rightarrow} xA\phi\,,\ A\phi \Rightarrow \omega\phi \overset{*}{\Rightarrow} ay,\ a \in T_\#,\ x, y \in T_\#,\ \phi \in (M_\# \cup T_\#)^*\}$.

A grammar G is an *LF grammar*, if for all $A \in M$ it is true for all $\phi, \psi \in R_A$, $\phi \neq \psi$ that $[A, \phi] \cap [A, \psi] = \phi$ (Wood, 1969). A language L is an *LF language*, if $L = L(G)$ for some *LF* grammar G.

LF languages and power property

Let us consider the following grammars.

$G_1 = \{\{S_1\}, \{a, b, c\}, S_1, P_1\}$, where $P_1 = \{S_1 \to aS_1a|b|c\}$,

$G_2 = \{\{S_2\}, \{a, b, c\}, S_2, P_2\}$, where $P_2 = \{S_2 \to aS_2b|c|\wedge\}$, and

$G_3 = \{M_3, T_3, S_3, P_3\}$, where $M_3 = \{S_3, A, B, C\}$, $T_3 = \{a, b, c, d, e, f, g, h\}$ and

$$P_3 = \{S_3 \to aAB,\ A \to bAC|c|d,\ B \to g|h,\ C \to e|f\}\,.$$

The languages, generated by G_1, G_2 and G_3 are denoted by L_1, L_2 and L_3 respectively. Grammars G_1, G_2 and G_3 are left factored, therefore L_1, L_2 and L_3 are LF languages.

In Wood (1970) the power property is defined and Lemma 8 asserts that every LF language has the power property. Nevertheless, for L_1, L_2 and L_3 we can form the sets

$$\Omega_1 = \{a^iba^i | i > 0\},\ R_1 = \{a^ica^i | i > 0\};$$
$$\Omega_2 = \{a^icb^i | i > 0\},\ R_2 = \{a^ib^i | i > 0\};$$
$$\Omega_3 = \{ab^ice^ig | i > 0\},\ R_3 = \{ab^idf^ih | i > 0\};$$

and these show that L_1, L_2 and L_3 do not have the power property.

LF grammars and k-derivations

In this section by grammar we will understand admissible grammar.

Definition 1

Let G be a grammar and $\phi \overset{*}{\underset{G}{\Rightarrow}} \psi$, be a derivation

$$\phi = \gamma_0 \Rightarrow \gamma_1 \Rightarrow \ldots \Rightarrow \gamma_l = \psi\,,$$

for $l \geqslant 1$. If there exists $k \geqslant 1$ such that $T(\psi) \geqslant k$ but $T(\gamma_{l-1}) < k$ and for some $y \in T^*$, $S \overset{*}{\Rightarrow} y\phi$, then we write $\phi \overset{k}{\Rightarrow} \psi$ and the derivation above is a *k-derivation*.

In other words, $\phi \overset{k}{\Rightarrow} \psi$ means that the first k symbols of ψ are terminals and the kth terminal of ψ appears on the last step of the derivation.

Definition 2

A grammar G is a *deterministic derivation grammar*, if for all $k > 0$ in the grammar $G_\#$ for all ϕ, ψ_1 and ψ_2 (ϕ, ψ_1, $\psi_2 \in \{M_\# \cup T_\#\}^*$) such that $\phi \overset{k}{\Rightarrow} \psi_1$, $\phi \overset{k}{\Rightarrow} \psi_2$ and $LM_k(\psi_1) = LM_k(\psi_2)$ then $\psi_1 = \psi_2$.

Lemma

If in some grammar G, $S \Rightarrow x\phi$, $T(\phi) = 0$ and $\phi \overset{*}{\underset{1}{\Rightarrow}} a\psi$, $a \in T$, then there is a word η such that $\phi \overset{*}{\Rightarrow} a\eta \Rightarrow a\psi$.

Proof

Let us consider the derivation

$$\phi = \gamma_0 \Rightarrow \gamma_1 \Rightarrow \ldots \Rightarrow \gamma_l = a\psi.T(\gamma_0) = T(\phi) = 0\,,$$

$T(\gamma_l) = T(a\psi) \geqslant 1$. Since we consider only left derivations

Downloaded from https://academic.oup.com/comjnl/article/17/3/242/389323 by guest on 19 April 2024

$T(\gamma_{j+1}) \geqslant T(\gamma_j)$ holds. Therefore $p = \min(j|T(\gamma_j \geqslant 1)$ exists, $0 < p \leqslant l$. Obviously $LM_1(\gamma_P) = a$, therefore for some η, $\gamma_p = a\eta$. The derivation $\phi = \gamma_0 \Rightarrow \gamma_1 \Rightarrow \ldots \Rightarrow \gamma_p = a\eta$ fulfills all the conditions of Definition 1 for $k = 1$, thus $\phi \overset{1}{\Rightarrow} a\eta$. From $\gamma_p \overset{*}{\Rightarrow} \gamma_l$ it follows that $a\eta \overset{*}{\Rightarrow} a\psi$.

Theorem
(*a*) Each LF grammar is a deterministic derivation grammar.
(*b*) An unambiguous deterministic derivation grammar is an LF grammar.

Proof
We will prove both statements by indirect proof.

(*a*) Let G be an LF grammar, assume that G is not a deterministic derivation grammar. In this case there are $k > 0$, ϕ, $\psi_1, \psi_2 \in \{T_\# \cup M_\#\}^*$ such that in the grammar $G_\#$ $\phi \overset{k}{\Rightarrow} \psi_1$, $\phi \overset{k}{\Rightarrow} \psi_2$, $LM_k(\psi_1) = LM_k(\psi_2)$ but $\psi_1 \neq \psi_2$. By Definition 1 $LM_k(\psi_1) \in T_\#^*$ and thus we can write $\psi_1 = y\psi'_1$, $\psi_2 = y\psi'_2$, where $y \in T_\#^*$ and $|y| = k$. Let us consider the derivations $\phi = \gamma_0 \Rightarrow \gamma_1 \Rightarrow \ldots \Rightarrow \gamma_l = y\psi'_1$ and $\phi = \delta_0 \Rightarrow \delta_1 \Rightarrow \ldots \Rightarrow \delta_j = y\psi'_2$. For $p = \max(s|\gamma_s = \delta_s)$ we have $p < l$ and $p < j$, therefore γ_p and δ_p are of the form $\gamma_p = \delta_p = xA\eta$, where $|x| < k$. Furthermore, $\gamma_{p+1} = x\omega_1\eta$ and $\delta_{p+1} = x\omega_2\eta$, where ω_1 and ω_2 are different alternatives of A and we have $x\omega_1\eta \overset{*}{\Rightarrow} y\psi'_1$, $x\omega_2\eta \overset{*}{\Rightarrow} y\psi'_2$. Since $|x| < k$ and $|y| = k$, then $y = xx'$ where $|x'| > 0$. Thus we have $LM_1(x') \in [A, \omega_1]$ and $LM_1(x') \in [A, \omega_2]$, but this is in contradiction with the definition of LF grammars.

(*b*) Let G be an unambiguous deterministic derivation grammar. Assume that G is not an LF grammar, then there are two different alternatives ω_1 and ω_2 for some meta-symbol A in the grammar G for which $[A, \omega_1] \cap [A, \omega_2] \neq \phi$. This means, that for some $x \in T_\#^*$ and $a \in T_\#$ we have in the grammar $G_\#$ $S_\# \overset{*}{\Rightarrow} xA\phi$, $A\phi \Rightarrow \omega_1\phi \overset{*}{\Rightarrow} ay_1$, $A\phi \Rightarrow \omega_2\phi \overset{*}{\Rightarrow} ay_2$. By the Lemma there exist η_1 and η_2, such that $A\phi \overset{1}{\Rightarrow} a\eta_1 \overset{*}{\Rightarrow} ay_1$, $A\phi \overset{1}{\Rightarrow} a\eta_2 \overset{*}{\Rightarrow} ay_2$. Since G is unambiguous, $\eta_1 \neq \eta_2$ (the derivations of $a\eta_1$ and $a\eta_2$ from $A\phi$ begin with different rules $A \to \omega_1$ and $A \to \omega_2$); this, however, is impossible in a deterministic derivation grammar.

We show now that the language $L_4 = \{a^idba^ib, a^idca^ic|i > 0\}$ is not an LF language. Assume the contrary, that there would be an LF grammar $G_4 = \{M_4, T_4, S_4, P_4\}$, such that $L(G_4) = L_4$. Let us consider the words a^idba^ib and a^idca^ic for a fixed i. Similar to the proof of the Lemma it can be shown that there are $\eta_{i,1}$ and $\eta_{i,2}$ such that $S_4 \overset{i+1}{\Rightarrow} a^id\eta_{i,1} \overset{*}{\Rightarrow} a^idba^ib$ and $S_4 \overset{i+1}{\Rightarrow} a^id\eta_{i,2} \overset{*}{\Rightarrow} a^idca^ic$. By the Theorem, $\eta_{i,1} = \eta_{i,2}$, we denote it by η_i. We can assume that all metasymbols of G have more than one terminal derivation (a metasymbol, which has a unique terminal derivation, can be removed from an LF grammar by substituting for this metasymbol in the rules of all other metasymbols with its unique terminal derivation). In this case it is easy to see that $|\eta_i| = 1$, otherwise from η_i would be derived not only ba^ib and ca^ic. Thus $\eta_i \in M_4$ and for $i \neq j$, $\eta_i \neq \eta_j$ which implies that M_4 is an infinite set, which is a contradiction. Therefore L_4 is not an LF language.

Note that L_4 can be generated by grammar

$$G_5 = (M_5, T_5, S_5, P_5),$$

where

$$M_5 = \{S_5, B, C\}, T_5 = \{a, b, c, d\}$$

and

$$P_5 = \{S_5 \to Bb|Cc, B \to aBa|db, C \to aCa|dc\}.$$

G_5 is an $LR(0)$-grammar, thus in Wood's notation L_4 is an E language (Knuth, 1965), accordingly L_4 can be used in the proof of Lemma 6 of Wood (1970).

In a similar manner it can be shown that languages $L_5 = \{a^iba^ib, a^ica^ic|i > 0\}$ and $L_6 = \{a^ibc, a^i|i > 0\}$ (see Wood, 1970) are not LF languages, but it is more complicated to show for these grammars that $|\eta_i| = 1$.

An open problem 1

In this section under Open Problem 1 we will understand the Open Problem 1 of Wood (1969).

Foster's SID (Foster, 1968) can be extended for the case in which one of two intersecting left terminal sets result from a rule of the form $A \to \wedge$.

In a grammar G, let there be a rule $A \to \phi_1|\phi_2|\ldots|\phi_k$, where $\phi_1 = \wedge$ and $[A, \phi_1] \cap [A, \phi_s] \neq \varnothing$ for some $1 < s < k$ (In such cases SID reports a failure.)

This is possible, for example, if in the grammar G there is a rule $B \to \psi_1|\psi_2|\ldots|\psi_l$, $\psi_1 = \eta A\alpha\zeta$, $\alpha\zeta \Rightarrow ax$, and $a \in [A, \phi_s]$. To eliminate this 'bad' appearance of A we can change the rule of B to a rule of the form $B \to \eta A'|\psi_2|\ldots|\psi_l$, where A' is a new metasymbol, and we include a rule for A':

$$A' \to f(\phi_1)|f(\phi_2)|\ldots|f(\phi_k)$$

where

$$f(\phi) = \begin{cases} \phi'A' & \text{if } \phi = \phi'A \\ \phi\alpha\zeta & \text{otherwise.} \end{cases}$$

Of course for A' we have $[A', f(\phi_1)] \cap [A', f(\phi_s)] \neq \phi$ and A' must be further processed in the manner described by Foster (1968).

Another case is when $\psi_1 = \eta A$ and is close up G there is a rule

$$C \to \xi_1|\xi_2|\ldots|\xi_m, \xi_1 = \tau B\alpha\zeta.$$

In this case we first process C and obtain

$$C \to \tau B'|\xi_2|\ldots|\xi_m,$$
$$B' \to g(\psi_1)|g(\psi_2)|\ldots|g(\psi_l),$$

where

$$g(\phi) = \begin{cases} \phi'B' & \text{if } \phi = \phi'B, \\ \phi\alpha\zeta & \text{otherwise.} \end{cases}$$

$g(\psi_1) = \eta A\alpha\zeta$ will be further processed as in the first case.

Note that if $B = A$ in the first case or $C = B$ in the second one, this algorithm does not eliminate the 'bad' appearance of A.

This method was realised in a Grammar Transformer Program in the form of a recursive ALGOL-procedure. The procedure processes 'bad' appearances of the metasymbol A one after the other in all rules. Either this appearance is of the form $B \to \eta A\alpha\zeta$, and then this appearance is transformed in the manner described above (the new rule for A' becomes the last rule in the grammar), except in the case $B = A$, when the procedure reports a failure; or this appearance is of the form $B \to \eta A$, and then the procedure begins to process metasymbol B if $B \neq A$ and continues to process A if $B = A$. The procedure also reports a failure if it begins to process a metasymbol which is already being processed, that is, if there is a deep right recursion of 'bad' metasymbols.

Let us consider the grammar $G_7 = (\{S_7, A, B\}, \{a, b\}, S_7, P_7)$, where $P_7 = \{S_7 \to Bbb, B \to aA, A \to \wedge|aA|baA\}$, (this is the example from Foster (1968), where A is a number continuation, B is a number, S is a word, a is a digit and b is a space).

For A we have $[A, \wedge] \cap [A, baA] \neq \phi$. The algorithm finds the appearance of A in the rule of B, and begins to process B. It finds B in the rule of S_7, the transformation of this appearance results in $S_7 \to B'$, $B' \to aAbb$. After this the procedure continues to process metasymbol A. On finding the appearances of A on the right end of the rule alternatives of A—they do not alter the grammar—the procedure goes on to find an appearance of A in the rule of B' and the transformation of this appearance results $B' \to aA'$, $A' \to bb|aA'|baA'$. A' is further processed in the manner described by Foster, which gives $A' \to bA''|aA'$, $A'' \to b|aA'$.

Downloaded from https://academic.oup.com/comjnl/article/17/3/243/383363 by guest on 19 April 2024

The algorithm excludes metasymbols B and A from the grammar to become an admissible grammar. Finally, we have $P_7 = \{S_7 \to B', B' \to aA', A' \to bA''|aA', A'' \to b|aA'\}$. This grammar is similar to the grammar given by Wood in Open Problem 1 (S_7-word, B'-number, A'-number tail, A''-space tail) however Wood gives the rules of A as $A' \to aA'|bA''|\wedge|$, which is incorrect, because after the last digit of the number there must be two spaces, (see grammar G_7).

Note that GTP does not solve Open Problem 1. For example, it loops for the grammar $G_8 = (\{S_8, A, B\}, \{a\}, S_8, P_8)$, where $P_8 = \{S_8 \to A, A \to aA|aB|a, B \to aB|aA\}$. However, $L_8 = L(G_8)$ is obviously an LF language.

References

FOSTER, J. M. (1968). A syntax improving program, *The Computer Journal*, Vol. 11, pp. 31-34.

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

KNUTH, D. (1965). On the Translation of Languages from Left to Right. *Information and Control*, Vol. 8, pp. 607-639.

WOOD, D. (1969). The theory of left factored languages: Part 1, *The Computer Journal*, Vol. 12, pp. 349-356.

WOOD, D. (1970). The theory of left factored languages: Part 2, *The Computer Journal*, Vol. 13, pp. 55-62.

Book reviews

Principles of interactive computer graphics, by W. M. Newman and R. F. Sproull, 1973; 607 pages. (*McGraw-Hill*, £7·85)

At least a useful and complete book on interactive computer graphics. This book, representing a collection of teaching material in interactive graphics which can be used at any university level, is also a great help to a professional in the field who uses the book to get more insight to a newly encountered aspect of the use of graphics or uses the book as a handbook.

The book is practically divided into five parts. The first part presents the principles of computer graphics, and describes the common types of graphics in use today. It also briefly illustrates the way a graphical processor is controlled by graphic instructions.

Part II introduces the concepts of programs used to drive a display, i.e. display file and structures. It describes the mechanisms used for 2D picture transformations (translation and rotation). It also introduces the concepts and related mechanisms for clipping and windowing the picture.

The third part discusses aspects related to the interactive use of computer graphics. It describes the use of various means for interaction between graphics users and the computer through graphics (light pen, mouse, etc.). This part also illustrates the interrupt and attention handling techniques, as well as various methods used to draw and track figures across the screen.

Part IV is concerned with 3D graphics and related aspects, such as hidden-lines, hidden-surfaces and shading, while Part V deals with advanced features in the use of graphics in graphical systems, graphical command and graphical programming languages. It also discusses the problem of the design of a graphical system.

One of the appendices, that concerned with the aspects of choosing a display system, is of particular importance.

All in all an excellent book. My only criticism is that the authors did not devote more time and space to aspects of the display files and structures, and the related mechanisms of creation and processing of such structures, which I regard as very fundamental. I would also like to have seen the author's opinion and a review of the display files and structures as used up to date.

On the other hand I found some unnecessary details slightly distracting, for example formulae on the electrical field and forces on the electron.

Furthermore I feel that the book is more inclined to the pictorial and geometrical use of graphics than to CAD applications.

These small criticisms do not alter my conclusion that this book belongs on the desk of any person involved in computer graphics.

N. MAROVAC (London)

A Handbook of Systems Analysis, by J. E. Bingham and G. W. P. Davies, 1972; 191 pages. (*Macmillan*, £4·95)

The structure of the book is in four parts; Part I: The steps of system analysis, Part II: Techniques, Part III: General Systems Considerations, Part IV: Project Control. In this form the book does give a useful overview of the pragmatic approach to systems work in Part I with general considerations and extensions dealt with in its later parts. Part II covers specific techniques of Fact Gathering, Charting, Simulation and Decision Tables. Unfortunately these topics are dealt with at an appreciation level only and do not meet the authors' introductory claims of 'working' guidelines'. Necessarily these topics and indeed the whole text provide 'a framework which further study and experience can expand' (the authors' own words in the introduction). The authors thus achieve their objective of a 'primary text for newcomers'.

A 'guidebook to practising analysts' is another claim. The text is laid out to emphasise by special markings, the activities or factors to be carried out or considered at successive phases of systems designer work. These do provide simple checklists which should help to re-assure a practitioner that he is following similar procedures. Obviously a very useful checklist for the newcomer about to undertake his first investigation! Here, though, the reviewer was conscious of the word 'handbook' in the title. To a one-time design engineer this conjured up images of fact sheets, design data sheets, standards on filing and reference systems, etc. Here I thought I might find some useful data for the systems designer distilled from the authors' practical experiences, perhaps some figures for task estimates, program size, run-times, key-punch loads, timing, etc. Unfortunately this was not so—the checklist of steps in analysis was useful, as were some standard documentation aids but I feel that without some practical data sheets, and the whole gathered into a loose-leaf format or useful appendices, a book of this kind is of limited usefulness to the practitioner.

Another laudable aim was 'the analysis of business systems—not about computers' but here the authors pay only lip-service. The chapters on techniques serve to illustrate this; e.g. simulation, after a general discussion simple illustrative examples follow on computer selection, benchmarks, a communications network. What an opportunity missed here to demonstrate the use of simulation in the analysis of business systems.

In conclusion this is a book of limited usefulness because of its lack of in-depth treatment of techniques and rather an expensive text for one of the overview variety.

A. H. WISE (Leicester)

Downloaded from https://academic.oup.com/comjnl/article/17/3/242/389323 by guest on 19 April 2021