PRECEDENCE PARSERS FOR PROGRAMMING LANGUAGES

Ph.D. Thesis

James Gray

Department of Computer Science
University of California
Berkeley, California 94720

September 1969

ACKNOWLEDGMENT

Thesis acknowledgments usually try to express in veiled terms the answer to the following question:

Pick one:

(1) The advisor wrote this thesis.

* (2) the advisor helped write this thesis.

* (3) The advisor helped.

* (4) The advisor let me go my own way.

(5) I never saw my advisor.

(6) The advisor gave me lots of trouble.

(7) I wish I had never seen my advisor.

Mike has fallen pleasantly into three of these categories. He has also provided the moral and financial support which has made the unpleasant process of passing exams and writing a thesis almost bearable.

I should also like to thank Jay Earley and Lenney Haines for their comments on this thesis.

Lastly I am grateful to NSF Grant GJ-474 for financial support.

Abstract

A theoretical study of precedence schemes used in parsing programming languages is presented. It unifies and generalizes the schemes of Floyd, Pair, Wirth and Weber, and Colmerauer by defining the concepts of token sets and operator sets and then defining precedence relations on them. The case in which a precedence scheme yields the canonical parse is constructively characterized.

It is often necessary to transform a given grammar into a similar one with some special property. A concept of a cover is introduced to formalize the word "similar". Grammar G_1 is said to cover G_2 if one can obtain all parses in G_2 by a simple homomorphism of the parses in G_1. It is shown that every grammar is covered by a normal form grammar and by an operator normal form grammar but that certain grammars are not covered by any grammar in Greibach normal form.

Precedence schemes simply detect the presence of a phrase containing an operator, they do not detect phrases not containing operators nor do they explain what production was used to obtain this phrase. Almost all previous work assumed that productions have unique right sides. A construction is presented which shows that every context free grammar may be transformed to an equivalent grammar with unique right-hand sides. However the resulting grammar is not a cover of the original. Further, this construction destroys the precedence relations. In fact precedence grammars with unique right-hand sides are a proper subset of the set of (1,1) bounded right context grammars.

A central conclusion of this thesis is that the restriction that the productions have unique right-hand sides should be relaxed. It is

shown for example that every LR(k) grammar is covered by a simple precedence LR(k) grammar and every BRC(n,m) grammar is covered by a simple precedence BRC(n',m) grammar.

Introduction

Among the many methods for parsing context-free languages, precedence analysis is especially attractive. It has been studied and has been applied to practical programming languages [1, 2, 3, 5, 12, 14, 18].

Floyd [5] first formalized the idea of precedence analysis by defining relations on terminal characters. In his theory, the grammars were not allowed to have Λ-rules and were required to be in operator form. Pair [16] and independently Wirth and Weber [18] generalized the method by relaxing some restrictions and by defining their precedence relations on the entire vocabulary. McKeeman [14] simplified the work of Wirth and Weber and considered schemes for reducing the size of precedence tables. Colmerauer [2] has generalized this notion by allowing more complex parsers and he began to consider the structure of precedence relations.

The present paper presents a general theory of precedence analysis which includes all other known schemes. We characterize how the schemes of Floyd and of Wirth and Weber fit into our theory. We characterize when Colmerauer's method produces a canonical left to right parse.

During the course of the thesis some of the results presented by Fischer [4] were independently obtained. They deal with the relative power of the schemes of Floyd and Wirth and Weber. We have omitted these results but strongly recommend that his paper be read in conjunction with this thesis.

This thesis is concerned with the use of precedence analyzers in syntax directed translation of programming languages [18]. It argues that a mechanically generated parser must produce a canonical parse although it may be allowed to ignore productions without semantic significance. This yields the idea of a "sparse parse". One grammar is said to "cover" another if it is possible to construct the sparse parses in the second from the sparse parses of the first. From the point of view of a syntax directed translator, being able to parse a covering grammar is as good as being able to parse the original grammar.

It is shown that every grammar is covered by a normal form grammar and by an operator normal form grammar but that there are grammars which are not covered by any Greibach normal form grammar.

Precedence parsers detect only those phrases which contain an operator. They ignore phrases not containing an operator and give no hint how to reduce the phrase. The problem of reduction is quite a complex one. Almost all published work has ignored the problem by requiring that the productions of the grammar have unique right-hand sides. Fischer has shown that if this restriction is relaxed any context-free language can be parsed with a simple precedence analyzer. Conversely, this thesis contains a construction which transforms any grammar into an equivalent grammar in which every production has a unique right-hand side. However these two constructions are incompatable, since the grammars with both properties are a proper subset of the (1,1) bounded right context grammars.

A central conclusion of this thesis is that every LR(k) grammar is covered by a simple precedence LR(k) grammar and every BRC(n,m) grammar is covered by a simple precedence BRC(n',m) grammar. This provides an economical reduction scheme in these cases. These facts are implicitly used by Cheatham [1] and Ichbiah and Morse [12] in constructing LR(k) parsers.

Section 1

Basic Definitions

We begin with the definition of a context-free grammar with a delimiter.

Definition. A context-free grammar with delimiter is a 4-tuple $G = (V, \Sigma, P, \perp S \perp)$ where

(i) V is a finite non-empty set (vocabulary)

(ii) $\Sigma \subseteq V$ is a finite non-empty set (terminal symbols)

(iii) $\perp \in \Sigma$. $\perp$ is called a delimiter.

(iv) $S \in V - \Sigma$. We call $\perp S \perp$ the start string.

(v) P is a finite subset[1] of $(V - \Sigma) \times (V - \{\perp\})^*$ and we write $u \to v$ instead of $(u,v) \in P$. P is the set of productions.

This family of grammars is a variant of the usual definition of context-free grammars since we need delimiters surrounding strings generated by the grammars.

Definition. Let ρ be a binary relation on a set X, i.e., $\rho \subseteq X \times X$. Define

$$\rho^0 = \{(a,a) \mid a \in X\}$$

---

[1] Let X and Y be sets of words. Write $XY = \{xy \mid x \in X, y \in Y\}$ where xy is the concatenation of x and y. Define $X^0 = \{\Lambda\}$ where Λ is the null word. For each $i \geq 0$, define $X^{i+1} = X^i X$ and $X^* = \bigcup_{i \geq 0} X^i$. Let $X^+ = X^* X$ and let $\emptyset$ denote the empty set. Finally, if x is a string, let $lg(x)$ denote the number of occurrences of symbols in x. Further for any $i \geq 0$ and any $x = a_1 a_2 \ldots a_n$, $a_1, \ldots, a_n \in \Sigma$ if $i \geq n$ then $x^{(i)} = x^{(i)} = x$. If $i < n$ then $x^{(i)} = a_1 \ldots a_i$ and $x^{(i)} = a_{n-i+1} \ldots a_n$.

and for each[2] $i \geq 0$

$$\rho^{i+1} = \rho^{i}\rho.$$

Lastly,

$$\rho^{*} = \bigcup_{i \geq 0} \rho^{i}$$

and

$$\rho^{+} = \rho^{*}\rho.$$

Next, we can define the rules for rewriting strings.

Definition. Let $G = (V, \Sigma, P, \perp S \perp)$ be a context-free grammar and let $u, v \in V^{*}$. Define $u \Longrightarrow v$ if there exist words, $x, y, w \in V^{*}$ and $A \in V - \Sigma$ so that $u = xAy$, $v = xwy$ and $A \rightarrow w$ is in P. If $y \in \Sigma^{*}$, we write $u \underset{R}{\Longrightarrow} v$. Furthermore define

$$\overset{*}{\Longrightarrow} = (\Longrightarrow)^{*}$$

and

$$\underset{R}{\overset{*}{\Longrightarrow}} = (\underset{R}{\Longrightarrow})^{*}.$$

The string $x \in V^{*}$ is said to be a sentential form if $\perp S \perp \overset{*}{\Longrightarrow} x$. It is a canonical sentential form if $\perp S \perp \underset{R}{\Longrightarrow} x$. Not every sentential form is canonical. Consider the grammar with productions

$$S \longrightarrow AA$$

$$A \longrightarrow a$$

The sentential form $\perp aA \perp$ is not canonical.

The set $L(G) = \{x \in \Sigma^{*} \mid \perp S \perp \underset{R}{\overset{*}{\Longrightarrow}} x\}$ is said to be the language generated by G.

---

[2] The operation is composition of relations which is defined as follows: if $\rho \subseteq X \times Y$ and $\sigma \subseteq Y \times Z$, define $\rho\sigma = \{(x,z) \mid x\rho y$ and $y\sigma z$ for some $y \in Y\}$. Observe that

$$\rho\sigma \subseteq X \times Z.$$

We now adopt four equivalent but notationally different definitions of derivations. Each notation has its own advantages and we shall use the most suitable one in any particular case.

If $u_o \Longrightarrow u_1 \Longrightarrow \ldots \Longrightarrow u_r$ then we say that the sequence $(u_o,\ldots,u_r)$ is a derivation of u_r from u_o. If $u_o \underset{R}{\Longrightarrow} u_1 \underset{R}{\Longrightarrow} \ldots \underset{R}{\Longrightarrow} u_r$ the derivation is said to be a canonical derivation. If for each $0 < i < r$, if $u_i = v_i A_i w_i$ and $u_{i+1} = v_i y_i w_i$ and u_{i+1} is obtained from u_i by using production $\pi_i = A_i \to y_i$ we say that the sequence of rule integer pairs $((A_o \to y_o, lg(v_o\ y_o)),\ldots,(A_{r-1} \to y_{r-1}, lg(v_{r-1}\ y_{r-1})))$ is a derivation of u_r from u_o. If the derivation is canonical then the sequence $((\pi_o,n_o);\ldots,(\pi_{r-1},n_{r-1}))$ is said to be a canonical derivation of u_r from u_o. In this case the n_i are superfluous so we also let $(\pi_o,\ldots,\pi_{r-1})$ denote the canonical derivation of u_r from u_o. If u_o is not mentioned it is assumed that $u_o = \perp S \perp$. Any particular derivation also corresponds to a labeled directed tree, called the parse tree. Two derivations are said to be equivalent if they have the same parse tree. The canonical derivation of any canonical sentential form is the canonical representative of its equivalence class.

If the sequence $(x_1,x_2,\ldots,x_n)$ is a derivation, of y from z then $(x_n,x_{n-1},\ldots,x_1)$ is said to be a parse of y to z. If the derivation is canonical the parse is said to be canonical. Further if z is not mentioned $\perp S \perp$ is assumed.

We shall sometimes denote the sequence $(\pi_o, \pi_1,\ldots, \pi_n)$ by $(\pi_i)_{i=0}^{n}$.

In a particular derivation of canonical sentential form x, denoted by a sequence of (rule, integer) pairs $((\pi_o,n_o),\ldots,(\pi_r,n_r))$, if $\pi_r = (A \to y)$ then the occurrence of the substring y in x at position n_r

is called a phrase of x, and the pair (π_r, n_r) is called a reduction of x. If the derivation is canonical then (π_r, n_r) is called a handle of x.

G is said to be unambiguous iff each canonical sentential form has a unique canonical derivation. This is clearly equivalent to each canonical sentential form having a unique handle.

Two fundamental problems associated with context-free grammars are:

Recognition Problem. Given a context free grammar $G = (V, \Sigma, P, \perp S \perp)$ give an algorithm which given any $x \in \Sigma^*$ decides whether or not $x \in L(G)$.

Parsing Problem. Given a context-free grammar $G = (V, \Sigma, P, \perp S \perp)$ and a string $x \in \Sigma^*$, produce all the canonical parses of x in G.

Of course, a solution to the parsing problem implies a solution to the recognition problem since if $x \notin L(G)$, there will be no canonical parses of x.

The motivations behind our interest in the parsing problem are the connections with syntax directed language translations. In these schemes, a parser enumerates the nodes of a parse tree of a source language string. With each such node is associated a (possibly null) semantic in some object language. The resulting sequence of semantics is the translated program [18].

The parsing schemes we will be dealing with are collectively known as "bottom up" schemes. They can be thought of as the iteration of the following three steps applied to a string $x \in V^*$. Initially x will be in Σ^*.

(1) find a leftmost phrase, y, of x (detection),

(2) determine the production involved A → y, and output it,

(3) replace y by A to obtain a new x. (reduction) and go to (1).

We shall devote almost all of this thesis to considering canonical parse. This decision is motivated by the following argument. If one is to construct a translator by having the parser enumerate the nodes of the parse tree and then associate a semantic with each node it is vital that the person writing the semantics have a clear idea of the order in which the nodes are going to be enumerated. The canonical parse corresponds to suffix polish or in the terminology of Knuth, post order. If the parser does reductions in a random way or a way depending on the grammar then it may be very difficult to write the semantics. Specifically if the parser is mechanically generated it is vital that it generate the nodes in a uniform way. The following example illustrates the difficulties which may arise.

Example: Consider the grammar

⟨program⟩ → ⟨statement⟩; | ⟨statement⟩; ⟨program⟩

⟨statement⟩ → GO TO A | GO TO B

Suppose that the semantics of "GO TO A" is "JP A" and similarly for B. Thus the program

GO TO A;

GO TO B;

has a canonical parse which generates

JP A

JP B

while the noncanonical parse generates

JP B

JP A

Thus a naive choice of grammar and precedence relations can in fact yield the latter parse. This cannot be tolerated in a fully automatic generator of parsers.

One alternative is possible, build up the parse tree in any order and then traverse it in canonical order. This is especially attractive in some situations (e.g., optimizing compilers). However this is quite an expensive way to build a compiler and is appropriate in most situations.

The following are common types of detection and reduction. Fix $G = (V, \Sigma, P, \perp S \perp)$.

Definition. G is LR(k) detectable iff

if $\perp S \perp \underset{R}{\overset{*}{\Longrightarrow}} xyz$ has handle $(A \to y, lg(xy))$

and $\perp S \perp \underset{R}{\overset{*}{\Longrightarrow}} xyz'$ has handle $(A' \to y', j)$

and ${}^{(k)}z = {}^{(k)}z'$ then $j = lg(xy)$ and $y' = y$.

Note that A' need not equal A.

Definition. G is LR(k) reducible iff

if $\perp S \perp \underset{R}{\overset{*}{\Longrightarrow}} xyz$ has handle $(A \to y, lg(xy))$

$\perp S \perp \underset{R}{\overset{*}{\Longrightarrow}} xyz'$ has handle $(A' \to y, lg(xy))$

and ${}^{(k)}z = {}^{(k)}z'$ then $A = A'$.

Definition. G is LR(k) iff it is LR(k) detectable and reducible. That is iff

if $\perp S \perp \underset{R}{\overset{*}{\Longrightarrow}} xyz$ has handle $(A \to y, lg(xy))$ and

if $\vdash S \dashv \underset{R}{\overset{*}{\Longrightarrow}} xyz'$ has handle $(A' \to y', j)$

and ${}^{(k)}z = {}^{(k)}z'$ then $(A' \to y', j) = (A \to y, lg(xy))$.

Next we deal with the bounded right context property.

Definition. G is bounded right context(n,m), (BRC(n,m)) detectable

iff if $\vdash S \dashv \underset{R}{\overset{*}{\Longrightarrow}} xyz$ has handle $(A \to y, lg(xy))$

and $\vdash S \dashv \underset{R}{\overset{*}{\Longrightarrow}} x'yz'$ has handle $(A' \to y', j)$

and $x^{(n)} = x'^{(n)}$ and ${}^{(m)}z = {}^{(m)}z'$ and $lg(x'y) \leq j$ then $j = lg(x'y)$ and $y' = y$.

Definition. G is BRC(n,m) reducible iff

if $\vdash S \dashv \underset{R}{\overset{*}{\Longrightarrow}} xyz$ has handle $(A \to y, lg(xy))$

and $\vdash S \dashv \underset{R}{\overset{*}{\Longrightarrow}} x'yz'$ has handle $(A' \to y, lg(x'y))$

and $x^{(n)} = x'^{(n)}$ and ${}^{(m)}z = {}^{(m)}z'$ then $A' = A$.

Definition. G is BRC(n,m) iff it is BRC(n,m) detectable and BRC(n,m) reducible. That is iff

if $\vdash S \dashv \underset{R}{\overset{*}{\Longrightarrow}} xyz$ has handle $(A \to y, lg(xy))$

$\vdash S \dashv \underset{R}{\overset{*}{\Longrightarrow}} x'yz'$ has handle $(A' \to y', j')$

and $x^{(n)} = x'^{(n)}$ and ${}^{(m)}z = {}^{(m)}z'$ and $lg(x'y) \leq j'$ then $(A' \to y', j') = (A \to y, lg(x'y))$.

These definitions are variants of the usual ones. Inspection of Knuths [13] paper shows that the above definition of LR(k) coincides with his. However our definition of BRC(nm) does not. We claim his definition is too restrictive since he leaves out the clause "$lg(x'y) \leq j'$" in the hypothesis. The following grammar is BRC(0,0) by our definition but not by his since $(A \to a, 3)$ is not a handle of $\vdash aa \dashv$.

$$S \rightarrow AA$$

$$A \rightarrow a.$$

Similarly Feldman and Gries [3] present a definition of bounded right context grammars with the same error.

Floyd [5] originally defined BRC(n,m) for Λ-free grammars[3]. He left out end markers so his definition fails on the grammar

$$G = (V, \Sigma, P, S) \qquad (\text{not } \perp S \perp)$$

$$S \rightarrow A \mid ab$$

$$A \rightarrow a$$

since by his definition $A \rightarrow a$ has context set (see below) $\{(\Lambda,\Lambda)\}$ so $(A \rightarrow a,1)$ must be a handle of ab.

The following presentation of his original definition corrects this. Floyd's definition proceeds as follows.

Let $G = (V, \Sigma, P, \perp S \perp)$ be a Λ-free grammar and let $m, n \geq 0$ Consider any $(A \rightarrow y) \in P$. The set $C \subseteq V^* \times \Sigma^*$ is said to be an <u>(n,m) canonical context set for $A \rightarrow y$</u> iff

(1) for each $(x,z) \in C$ $lg(x) \leq n$ and $lg(z) \leq m$,

and (2) if $\perp S \perp \underset{R}{\overset{*}{\Longrightarrow}} xAz$ then for some $(x_1,z_1) \in C$, $x_2, z_2 \in V^*$, $x = x_2x_1$ and $z = z_1z_2$.

The <u>production</u> $A \rightarrow y$ will be said to be of <u>bounded right context (n,m)</u> iff $A \rightarrow y$ has an (n,m) canonical context set C such that for any $(x,z) \in C$ and any $x_1, x_2, y_1, y_2, z_1, z_2 \in V^*$ such that $x = x_1x_2$, $y = y_1y_2$, $z = z_1z_2$. <u>None</u> of the following fourteen predicates are satisfied for any $s, t, u, v, w \in V^*$, $B \in V \div \Sigma$:

(12') (1) $\perp S \perp \underset{R}{\overset{*}{\Longrightarrow}} sBzvw \underset{R}{\Longrightarrow} stxyzvw$ and $(tx) \neq \Lambda$

---

[3] A grammar $G = (V, \Sigma, P, \perp S \perp)$ is <u>Λ-free</u> if $P \subseteq (V \times V^+)$.

(22') (2) $\perp s \perp \underset{R}{\overset{*}{\Longrightarrow}} stx_1Bzvw \underset{R}{\Longrightarrow} stx_1x_2yzvw$ and $x_2 \neq \Lambda$

(32') (3) $\perp s \perp \underset{R}{\overset{*}{\Longrightarrow}} stxBzvw \underset{R}{\Longrightarrow} stxyzvw$ and $B \neq A$

(42') (4) $\perp s \perp \underset{R}{\overset{*}{\Longrightarrow}} stxy_1Bzvw \Longrightarrow stxy_1y_2zvw$ and $y_1 \neq \Lambda$

(13') (5) $\perp s \perp \underset{R}{\overset{*}{\Longrightarrow}} sBz_2vw \underset{R}{\Longrightarrow} stxyuz_2vw \underset{R}{\overset{*}{\Longrightarrow}} stxyz_1z_2vw$

and $z_1 \neq \Lambda$

(23') (6) $\perp s \perp \underset{R}{\overset{*}{\Longrightarrow}} stx_1Bz_2vw \underset{R}{\Longrightarrow} stx_1x_2yuz_2vw \underset{R}{\overset{*}{\Longrightarrow}}$

$stxyz_1z_2vw$ and $z_1 \neq \Lambda$

(33')(43') (7) $\perp s \perp \underset{R}{\overset{*}{\Longrightarrow}} stxy_1Bz_2vw \underset{R}{\Longrightarrow} stxy_1y_2uz_2vw \Longrightarrow$

$stxyz_1z_2vw$ and $z_1 \neq \Lambda$

(14') (8) $\perp s \perp \underset{R}{\overset{*}{\Longrightarrow}} sBw \underset{R}{\Longrightarrow} stxyzvw$ and $v \neq \Lambda$

(24') (9) $\perp s \perp \underset{R}{\overset{*}{\Longrightarrow}} stx_1Bw \underset{R}{\Longrightarrow} stx_1x_2yzvw$ and $v \neq \Lambda$

(34')(44') (10) $\perp s \perp \underset{R}{\overset{*}{\Longrightarrow}} stxy_1Bw \underset{R}{\Longrightarrow} stxy_1y_2zvw$ and $v \neq \Lambda$.

The numbers in the left margin are the ones given by Floyd in his paper. Inspection of these predicates shows that in any case the resulting string is stxyzvw = r.

Cases (1), (2), (3), (4) characterize the case

(a) r has handle $(D \rightarrow q, lg(stxy))$ and $A \neq D$ or $q \neq y$

Cases (5), (6), (7) characterize the case

(b) r has handle $(D \rightarrow q,j)$ and $lg(stxy) < j \leq lg(stxyz)$

Cases (8), (9), (10) characterize the case

(c) r has handle $(D \rightarrow q,j)$ and $lg(stxyz) < j$

a, b, c may be abstracted as

r has handle $((D \rightarrow q),j)$ and $j > lg(stxy)$ or $q \neq y$ or $A \neq D$.

Thus we may rephrase Floyd's definition as A → y is BRC(nm) iff there is a context set C for A → y such that $\forall(x_0 z_1) \in C$, $x_2, z_2 \in V^*$ if $\perp S \perp \underset{R}{\overset{*}{\Longrightarrow}} x_2 x_1 y z_1 z_2$ has handle $(B \to q, j)$ and if $j \geq \lg(x_2 x_1 y)$ then $(B \to q, j) = (A \to y, \lg(x_2 x_1 y))$..

We shall adopt this modified version of Floyd's definition in what follows. Floyd went on to define G to be BRC(n,m) iff each A → y in P is BRC(n,m).

Floyd adopted this form of the definition because it made completely clear that the question "Is G BRC(n,m)?" is decidable. Since there is a great deal of confusion about briefer forms of the definition we prove that in the case of Λ-free grammars our definition coincides with Floyd's.

Proposition. Suppose $G = (V, \Sigma, P, \perp S \perp)$ is a Λ-free grammar (Floyd did not define BRC(n,m) on grammars with Λ rules). Then for any $n, m \geq 0$, G is BRC(n,m) in the Floyd sense iff it is BRC(n,m) in our sense.

Proof: Suppose G is BRC(n,m) in the Floyd sense. Then suppose

$$\perp S \perp \underset{R}{\overset{*}{\Longrightarrow}} xyz \quad \text{has handle } (A \to y, \lg(xy))$$

$$\perp S \perp \underset{R}{\overset{*}{\Longrightarrow}} x'yz' \text{ has handle } (A' \to y', j')$$

and suppose $j' \geq \lg(x'y)$ and $x^{(n)} = x'^{(n)}$ and ${}^{(m)}z = {}^{(m)}z'$. Now A → y has an (n,m) canonical context set C. So for some $(x_1, z_1) \in C$, $x_2, z_2 \in V^*$ $xyz = x_2 x_1 y z_1 z_2$. Since $\lg(x_1) \leq n$, $x^{(n)} = x'^{(n)}$ and ${}^{(m)}z = {}^{(m)}z'$. We have $x'yz' = x_3 x_1 y z_1 z_3$ for some $x_3, z_3 \in V^*$. Further $j' \geq \lg(x'y)$ so x'yz' has handle $(A \to y, \lg(x'y)) = (A' \to y', j')$ since A → y is BRC(n,m) in Floyd's sense. So G is BRC(n,m) in our sense.

Conversely if G is BRC(nm) in our sense and G is Λ free consider any (A → y) in P. If it is not the case that for some u, v ∈ V^* $\perp S \perp \underset{R}{\Longrightarrow} uAv$. Then A → y vacuously satisfies Floyd's definition

of BRC(n,m). So assume that

$$C = \{(x^{(n)}, {}^{(m)}z) \mid \perp S \perp \underset{R}{\overset{*}{\Longrightarrow}} xAz\}$$

is nonempty. Clearly it is an (n,m) canonical context set for $A \to y$. To verify that $A \to y$ is BRC(n,m) in the Floyd sense consider any $(x_1, z_1) \in C$. Then for some $x_2, z_2 \in V^*$

$$\perp S \perp \underset{R}{\overset{*}{\Longrightarrow}} x_2 x_1 A z_1 z_2 \underset{R}{\Longrightarrow} x_2 x_1 y z_1 z_2$$

Now suppose that for some x_3, $z_3 \in V^*$

$\perp S \perp \underset{R}{\overset{*}{\Longrightarrow}} x_3 x_1 y z_1 z_3$ has handle $(A' \to y', j')$ and $j \geq \lg(x_3 x_1 y)$.

Then since G is BRC(n,m) in our sense $(A' \to y', j') = (A \to y, \lg(x_3 x_1 y))$. So $A \to y$ satisfies Floyd's definition of a BRC(n,m) production. Thus all productions are BRC(n,m) in Floyd's sense so G is BRC(n,m) in Floyd's sense.

In passing we mention that one can define Floyd's concept of bounded context (n,m) (BC(nm)) as:

For any n, $m \geq 0$ and any grammar $G = (V, \Sigma, P, \perp S \perp)$, G is BC(n,m) iff

(a) every sentential form has a unique handle (i.e., G is unambiguous),

and (b) for all x, z, x', z' in V^*, $A \to y$ in P, if

$\perp S \perp \underset{}{\overset{*}{\Longrightarrow}} xAz \Longrightarrow xyz$, and

$\perp S \perp \underset{}{\overset{*}{\Longrightarrow}} x'yz'$ and if

$x^{(n)} = x'^{(n)}$ and ${}^{(m)}z = {}^{(m)}z'$

Then x'Az' is a sentential form.

We have not found BC(n,m) to be a useful concept in parsing programming languages since it does not take advantage of the information gained by previous reductions. Also does not correspond to the canonical parse.

These definitions are distinct from those given in the literature. We have chosen them for their relative simplicity. For examply they allow the following simple proof that BRC(n,m) implies LR(m).

Remark: If G is BRC(n,m) it is LR(m).

Proof: Suppose

$$\perp S \perp \underset{R}{\overset{*}{\Longrightarrow}} xyz \quad \text{has handle } (A \to y,\ lg(xy))$$

$$\perp S \perp \underset{R}{\overset{*}{\Longrightarrow}} xyz' \quad \text{has handle } (A' \to y', j')$$

where ${}^{(m)}z = {}^{(m)}z'$. There is no loss in generality in assuming that $j' \geq lg(xy)$ by the symmetry of the problem. Now $x^{(n)} = x^{(n)}$ and ${}^{(m)}z = {}^{(m)}z'$ so $((A \to y),\ lg(xy)) = ((A' \to y'), j')$ by the definition of BRC(n,m). Therefore G is LR(m). ∎

Similarly we can show that if G is LR(k) it is G unambiguous.

Proof: First we recall that a grammar is unambiguous iff each sentential form has a unique handle. The definition of LR(k) implies that every sentential form has a unique handle (take x = y). So LR(k) grammars are unambiguous. ∎

Section 2

Normal Forms and Covers

When presented with a particular grammar most parsing algorithms give one of the following answers:

(1) I can parse this grammar,

(2) I can't parse this grammar but, I can parse one "similar" to it,

(3) I can't parse this grammar.

This section proposes a formal definition of the word "similar". The development is predicated on the idea that the parser is intended to drive a syntax directed compiler.

The parsers described so far enumerate all nodes of the parse tree. For applications, we would be content with a parser which enumerated only those nodes in the parse tree which have semantic significance. For example, consider the generation tree of Fig. 1 which occurs in Euler [16].

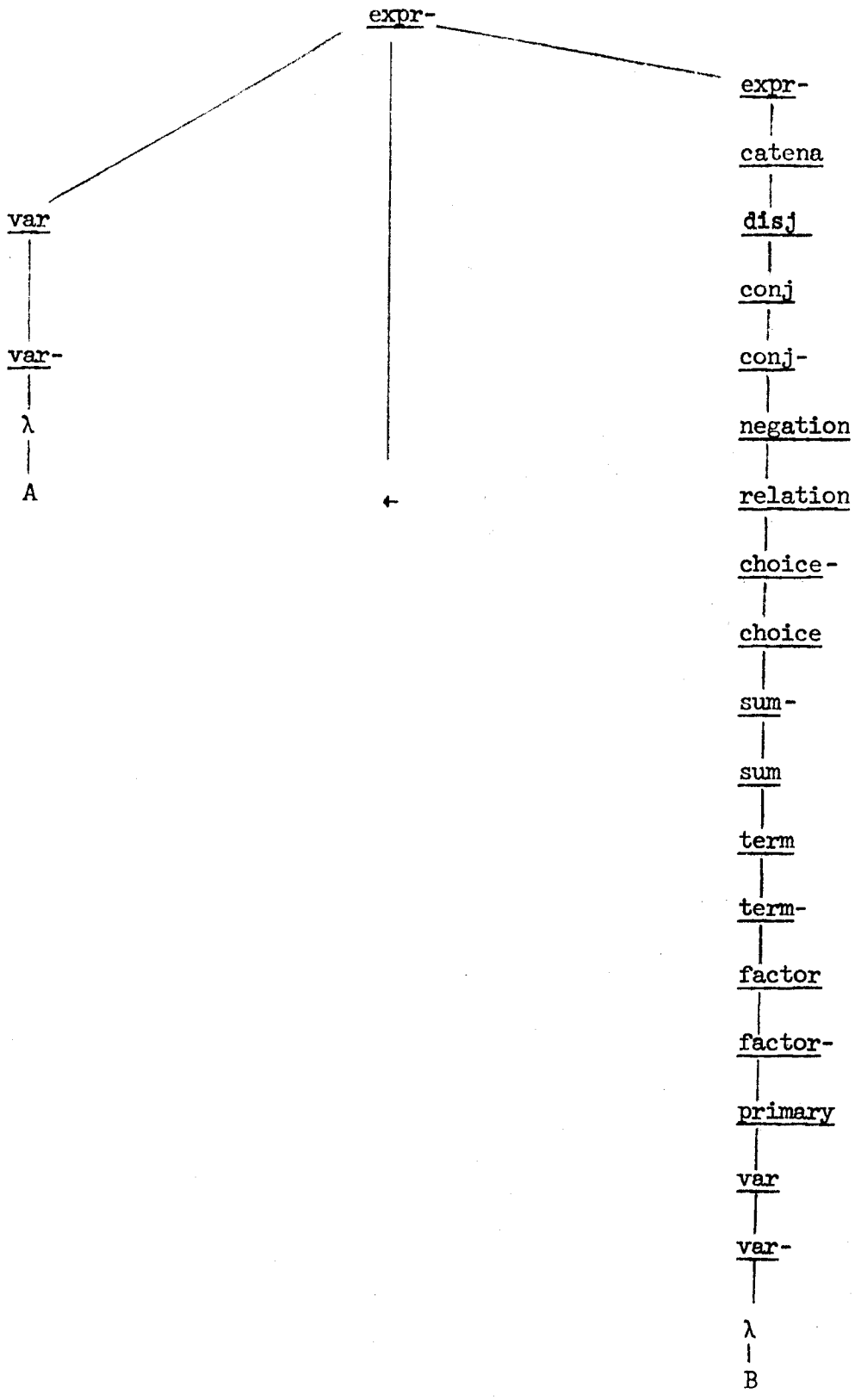
The chain[4] $\underline{\text{expr}} \overset{*}{\Longrightarrow} \lambda$ is typical of what happens in grammars for programming languages. Chains exist to enforce precedence of operators and to collect several categories of syntactic types.

Example. In ALGOL, one has the BNF statement

⟨statement⟩::=⟨unconditional statement⟩ |
⟨conditional statement⟩ |
⟨for statement⟩

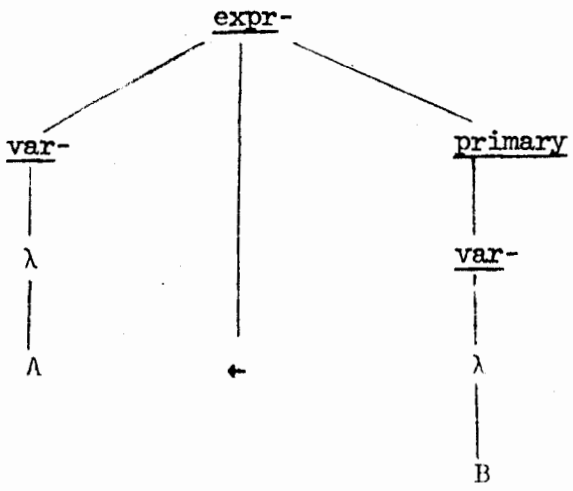
---

[4] A derivation $z_0 \Longrightarrow \ldots \Longrightarrow z_r$ is said to be a chain if $r > 0$ and $z_i \in V - \Sigma$ for each i.

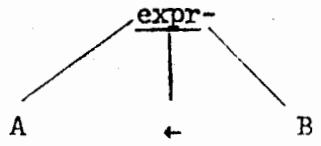


A generation from EULER

Fig. 1



(a) A sparse parse of the generation in Fig. 1.



(b) A non-sparse parse of Fig. 1.

Fig. 2

Chain productions rarely have semantic significance as our running example shows since in Euler, only the following productions have nontrivial semantics.

$$\underline{\text{expr-}} \rightarrow \underline{\text{var}} \leftarrow \underline{\text{expr-}}$$

$$\underline{\text{var-}} \rightarrow \lambda$$

$$\underline{\text{primary}} \rightarrow \underline{\text{var}}$$

$$\lambda \rightarrow A$$

$$\lambda \rightarrow B$$

For the purposes of code generation, the "sparse generation tree" of Fig. 2a is as satisfactory as the tree in Fig. 1. The tree shown in Fig. 2b would not be a satisfactory replacement for the original tree because some nodes with semantic significance have been omitted.

We must formalize this notion of restricting attention to nodes with semantic significance.

Definition. Let $G = (V, \Sigma, P, \perp S \perp)$ be a context-free grammar and H be a subset of P. Let $\pi_1,\ldots,\pi_k$ be distinct symbols representing the productions in P. Suppose x is a sentential form and the string $\pi_{i_1}\ldots\pi_{i_r}$ denotes a canonical derivation of x. The H-sparse generation of x corresponding to $\pi_{i_1}\ldots\pi_{i_r}$ is $\varphi(\pi_{i_1}\ldots\pi_{i_r})$ where φ is the homomorphism generated by

$$\varphi(\pi_i) = \begin{cases} \pi_i & \text{if } \pi_i \in H \\ \Lambda & \text{otherwise} \end{cases}$$

Thus $\varphi(\pi_{i_1}\ldots\pi_{i_r})$ denotes the subsequence of the productions of $\pi_{i_1}\ldots\pi_{i_r}$ which are in H. Note that we may speak of sparse parses as

well as sparse generations. Non-canonical sparse generations are defined analogously.

It is often convenient to transform a given grammar G into a "similar" grammar G' which possesses some special form. For our purposes, "similar" must mean, at the very least, equivalence [4]. We also would like to be able to construct a parse for x in G if we were given a parse for x in G' because we would like to be able to generate the same code for x in both grammars.

Definition. Let $G = (V, \Sigma, P, \perp S \perp)$ and $G' = (V', \Sigma', P', \perp S' \perp)$ be context-free grammars with $H \subseteq P$ and $H' \subseteq P'$. We say that (G,H) covers (G',H') under a map φ from H' into H if

(1) $L(G') = L(G)$

and (2) for each $x \in L(G)$,

(a) if π is an H-sparse generation of x in G there is an H'-sparse generation π' of x in G' so that $\varphi(\pi') = \pi$.

and (b) if π' is an H'-sparse generation of x in G' then $\varphi(\pi')$ is an H-sparse generation of x in G.

G' is said to cover (G,H) if there exist H' and φ so that (G',H') covers (G,H) under φ. Finally, if $H = P$, we say that G' completely covers G.

The basic idea behind this definition is that if (G',H') covers (G,H) by φ and if one can construct the H' sparse parse of x in G' then, using a simple table look up method (with the table of φ) one can construct the H-sparse parse of x in G.

If (G_{i+1}, H_{i+1}) covers (G_i, H_i) under map φ_i for $i = 1, 2$, then (G_3, H_3) covers (G_1, H_1) under map $\varphi_1\varphi_2$. In other words, covering is a transitive relation. The relation is reflexive but not symmetric.

Note that G' is equivalent to G iff G covers $\langle G', \varphi \rangle$.

The spirit of our definition of covers is very much like that of Reynolds [17]. It differs in that we allow sparse generations and in that Reynolds is more concerned with embeddings so he does not require $L(G) = L(G')$.

Covers are also related to isomorphism of grammars, a concept introduced in a more restrictive way in [7]. We need this concept in its full generality.

Definition. Let $G = (V, \Sigma, P, \perp S \perp)$ and $G' = (V', \Sigma', P', \perp S' \perp)$ be context-free grammars. G is isomorphic to G' if there exists a map ψ so that (G', P') covers (G, P) under ψ and

(1) there is a one-to-one map φ from V' into V so that $\varphi A = A$ if $A \in \Sigma'$ or $A = S'$,

and (2) Extending φ to a homomorphism on V^* we require that

$$\psi(A \to w) = \varphi A \to \varphi w.$$

Thus isomorphism requires that G'-parse trees map onto G-parse trees by a functional relabelling of non-terminal nodes [7,18].

We will also need some familiar definitions from the theory of context-free languages [7,11].

Definition. A context-free grammar $G = (V, \Sigma, P, \perp S \perp)$ is said to be

(i) Λ-free if $P \subseteq ((V - \Sigma) \times V^+) \cup \{(S, \Lambda)\}$

(ii) chain free if $P \cap ((V - \Sigma) \times (V - \Sigma)) = \emptyset$

(iii) reduced if (a) for each $A \in V$, there exist x, y in V^* so that $S \overset{*}{\Longrightarrow} xAy$,

and (b) for each $A \neq S$, there exists $x \in \Sigma^*$ so that $A \overset{*}{\Longrightarrow} x$.

(iv) invertible if $A \to x$ and $B \to x$ in P implies $A = B$

(v) in normal form if $P \subseteq (V - \Sigma) \times (\{\Lambda\} \cup V \cup (V - \Sigma)^2)$

(vi) in operator form if $P \subseteq (V - \Sigma) \times (V^* - V^*(V - \Sigma)^2 V^*)$

(vii) in Greibach form if $P \subseteq ((V - \Sigma) \times \Sigma V^*)$

The following results all appear in the literature (cf. [7,11] unless otherwise referenced.)

(a) G is equivalent to a Λ-free grammar.

(b) G is equivalent to a chain-free grammar.

(c) G is isomorphic to a reduced grammar.

(d) If G is a parenthesis grammar then G is equivalent to an invertible parenthesis grammar [15].

(e) G is equivalent to a grammar in normal form.

(f) G is equivalent to a grammar in operator form [10].

(g) G is equivalent to a grammar in Greibach form [10,12].

These properties may be combined into pairs (i.e., a grammar may be assumed to satisfy an arbitrary pair of the properties) except that the pairs (d,e), (e,f) and (e,g) are incompatible.

Our first result extends (d) to the class of all context-free languages. This result has the following interesting interpretation. Bottom-up parsing may be thought of as involving two processes, (1) detecting a phrase and (2) reducing the phrase. It is possible to do (1) easily for any context-free language using precedence techniques [4]. If this is done, reduction becomes difficult. On the other hand, Theorem 2.1 shows that reduction can be made trivial but detection becomes more difficult.

Theorem 2.1. For each context-free grammar $G = (V, \Sigma, P, \perp S \perp)$ there is an invertible context-free grammar $G' = (V', \Sigma, P', \perp S' \perp)$ so that $L(G') = L(G)$.

Proof: We may assume, without loss of generality, that G has no Λ-rules and that G has no rules of the form $A \to B$. (The case where $\Lambda \in L(G)$ may be easily handled.)

Let $G' = (V', \Sigma, P', \perp S' \perp)$ where $V' - \Sigma = \{U \subseteq V - \Sigma \mid U \neq \emptyset\} \cup \{S'\}$ where S' is a new symbol not in V.

Thus the variables of G' (except S') will be nonempty subsets of the variables of G.

P' is defined as follows:

1. $S' \to A$ where $S \in A \subseteq V - \Sigma$ is in P'
2. For each production $B \to x_0 B_1 x_1 \ldots B_n x_n$ in P with $B_1, \ldots, B_n \in V - \Sigma$ and $x_0, \ldots, x_n \in \Sigma^*$, then for each $A_1, \ldots, A_n \in V' - (\Sigma \cup \{S'\})$, P' contains

$$A \to x_0 A_1 x_1 \ldots A_n x_n$$

where

$$A = \{C \mid C \to x_0 C_1 x_1 \ldots C_n x_n \text{ is in } P \text{ for some } C_1, \ldots, C_n \text{ with each } C_i \in A_i\}$$

Thus if $C \to y_0 C_1 y_1 \ldots C_n y_n$ with $y_0, \ldots, y_n \in \Sigma^*$, $C_i \in V - \Sigma$, we call the string $y_0 - y_1 \ldots - y_n$ the stencil of the production (variables replaced by dashes).

Note that P and P' have the same set of stencils and that G' is invertible. Assume without loss of generality that G' is reduced.

Before embarking on a proof that $L(G') = L(G)$, we given an example of the construction.

Example: Consider the following grammar

$$S \to 0A \mid 1B$$

$$A \to 0A \mid 0S \mid 1B$$

$$B \to 1 \mid 0$$

Applying the construction of the theorem leads to the following grammar.

$\{B\} \to 1 \mid 0$

$\{A\} \to 0\{S\} \mid 0\{S,B\}$

$\{A,S\} \to 0\{A\} \mid 0\{A,B\} \mid 0\{A,S\} \mid 0\{A,S,B\} \mid 1\{B\} \mid 1\{B,A\} \mid 1\{B,A,S\} \mid 1\{B,S\}$

$S' \to \{S\} \mid \{A,S\} \mid \{B,S\} \mid \{A,B,S\}$

Reducing the grammar leads to

$$S' \to \{A,S\}$$

$$\{B\} \to 1 \mid 0$$

$$\{A,S\} \to 0\{A,S\} \mid 1\{B\}$$

Now we begin the proof that $L(G') = L(G)$.

Claim 1. For each $A \in V' - \Sigma$ and each $x \in \Sigma^*$

$$A \overset{*}{\Longrightarrow} x \text{ in } G'$$

implies

$$B \overset{*}{\Longrightarrow} x \text{ in } G \text{ for each } B \in A .$$

Proof: The argument is an induction on ℓ, the length of a derivation in G'.

Basis. Suppose $\ell = 1$. Then $A \Longrightarrow x \in \Sigma^*$ in G' and $A \to x$ is in P'. By the construction $A = \{C \in V - \Sigma \mid C \to x \text{ is in } P\}$. Clearly this holds if and only if $B \to x$ is in P for each $B \in A$.

Induction Step. Suppose $\ell \geq 2$ and Claim 1 holds for all derivations of length less than ℓ. Then suppose $A \Longrightarrow x_0 A_1 x_1 \ldots A_n x_n \overset{*}{\Longrightarrow} x$ in G' by a derivation of length ℓ. This implies that for each i, $1 \leq i \leq n$, $A_i \overset{*}{\Longrightarrow} y_i \in \Sigma^*$ and $x_0 y_1 x_1 \ldots y_n x_n = x$. By the construction, for each $B \in A$ there exist $B_i \in A_i$ so that $B \to x_0 B_1 x_1 \ldots B_n x_n$ is in P. Moreover, the induction hypothesis implies that $B_i \overset{*}{\Longrightarrow} y_i$ in G and therefore

$$B \Longrightarrow x_0 B_1 x_1 \ldots B_n x_n \Longrightarrow x_0 y_1 x_1 \ldots y_n x_n = x .$$

Note that Claim 1 implies that

$$L(G') \subseteq L(G)$$

To complete the proof, the following result is needed.

Claim 2. For each $x \in \Sigma^*$, let $X = \{C \in V - \Sigma \mid C \overset{*}{\Longrightarrow} x \text{ in } G\}$. If $B \overset{*}{\Longrightarrow} x$ in G then $A \overset{*}{\Longrightarrow} x$ in G' for some A such that $B \in A \subseteq X$.

Proof: The argument is an induction on ℓ, the length of a derivation in G.

Basis. $\ell = 1$. Suppose $B \Longrightarrow x \in \Sigma^*$ so $B \to x$ is in P. Then $A \to x$ is in P' with $B \in A = \{C \in V - \Sigma \mid C \to x \text{ is in } P\}$.

Induction Step. Suppose $B \Longrightarrow x_0 B_1 x_1 \ldots B_n x_n \overset{*}{\Longrightarrow} x_0 y_1 x_1 \ldots y_n x_n = x \in \Sigma^*$ in G is a derivation of length ℓ.

There are derivations $B_i \overset{*}{\Longrightarrow} y_i$, all of which have length $< \ell$. By the induction hypothesis, there are $A_i \in V' - \Sigma$ so that

$$A_i \overset{*}{\Longrightarrow} y_i \text{ in } G'$$

and $B_i \in A_i$. By the construction $A \to x_0 A_1 x_1 \ldots A_n x_n$ is in P' with $B \in A$. Thus $A \Longrightarrow x_0 A_1 x_1 \ldots A_n x_n \overset{*}{\Longrightarrow} x_0 y_1 x_1 \ldots y_n x_n = x$ in G'.

By Claim 2, $L(G') \supseteq L(G)$ and hence $L(G') = L(G)$. ∎

It is easy to see that this condition is compatible with conditions (a) through (f) and not compatible with (g). It is interesting to note that for any grammar G, one can find an equivalent grammar G' which is invertible and chain free. On the other hand, there are grammars G for which there do not exist equivalent grammars which are invertible, chain free and Λ-free. An example of such a grammar is

$$S \to \Lambda \mid b$$
$$A \to aA \mid a$$

This first result provides an opportunity to exercise the definition of cover. The grammar G' of Theorem 2.1 does not necessarily cover G. For example if G is the grammar

$$S \to A \mid B$$
$$A \to a$$
$$B \to a$$

Then G' is

$$\{S\} \to \{A,B\}$$
$$\{A,B\} \to a$$

which cannot cover G since it is unambiguous. However the grammar

$$S \to A \mid B$$
$$A \to a$$
$$B \to aL$$
$$L \to \Lambda$$

does completely cover G. Generalizing this result we obtain the following theorem:

Theorem 2.1A: Let $G = (N, \Sigma, P, \lfloor S \rfloor)$ be a context-free grammar. Then G is completely covered by an invertible grammar G'.

Proof: We simply present the construction. Index the elements of N by the integers $1, 2, \ldots, |N|$. Let the index of $A \in N$ be denoted $I(A)$. Let L be a new symbol and construct $G' = (N', T, P', \lfloor S \rfloor)$ as follows:

$$N' = N \cup \{L\}$$

$$P' = \{A \to x\, L^i \mid A \to x \in P \text{ and } I(A) = i\} \cup \{L \to \Lambda\}$$

Then $\langle G', H\rangle$ covers $\langle G, P\rangle$ under φ where $H = P' - \{L \to \Lambda\}$ and φ: $H \to P$ is defined by $\varphi(A \to x\, L^i) = (A \to x)$ for each $A \in N$, $i = I(A)$, $(A \to x\, L^i) \in H$. ∎

This result shows an application of the concept of covers as a research tool. Perhaps it is useful to explain the genesis of these results. 2.1 was a generalization of McNaughton's result on parenthesis grammars [15]. The work on covers came later, intuitively it was felt that the construction should not produce a cover. This followed from the intuition provided by the construction of Theorem 2.1. However close investigation yielded the result of 2.1A. It is true that the grammar G in the example above is not completely covered by any Λ-free grammar.

Next we turn to a consideration of normal forms and a short exposition of covers.

Theorem 2.2. For each context-free grammar G there is a context free grammar G' which is in normal form and which completely covers G.

Proof: Let $G = (V, \Sigma, P, \lfloor S \rfloor)$ be a context-free grammar and suppose that [and] are two new symbols not in V. We define a new

context-free grammar $G' = (V', \Sigma, P', [S])$ where

$$V' = \Sigma \cup \{[y] \mid A \to xy \text{ is in } P \text{ for some } y \in V^2V^*\}$$
$$\cup \{[A] \mid A \in V\}$$

The variables of the new grammar are of the form [x] and there are only a finite number of them. Define

$$P_1 = \{[a] \to a \mid a \in \Sigma - \{\perp\}\}$$
$$P_2 = \{[A] \to \Lambda \mid A \to \Lambda \text{ is in } P\}$$
$$P_3 = \{[A] \to [B] \mid A \to B \text{ is in } P \text{ for } A \in V - \Sigma,\ B \in V\}$$
$$P_4 = \{[A] \to [B][x] \mid A \to Bx \text{ is in } P \text{ for } B \in V,\ x \in V^+\}$$

and $P_5 = \{[Ax] \to [A][x] \mid A \in V,\ x \in V^+,\ [Ax] \in V' - \Sigma\}$

Combining, let $P' = \bigcup_{i=1}^{5} P_i$ and define $H' = P_2 \cup P_3 \cup P_4$. We define φ from H' into P by

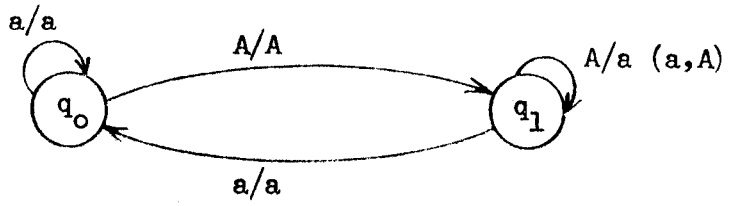
$$\varphi([A] \to \Lambda) = A \to \Lambda \quad \text{for each } [A] \to \Lambda \text{ in } P_2$$
$$\varphi([A] \to [B]) = A \to B \text{ for each } [A] \to [B] \text{ in } P_3$$
$$\varphi([A] \to [B][x]) = A \to Bx \text{ for each } [A] \to [B][x] \text{ in } P_4$$

Clearly φ is a one-to-one map from H' onto P. Consider the (phrase structure) grammar G'' obtained by dropping the brackets in G'. Note that the rules corresponding to P_1 and P_5 are identity maps on V^* while all other rules come from P. It is now easy to verify that $L(G') = L(G) = L(G'')$ and that φ maps the sparse parses into each other as required. ∎

Another type of grammar which can be a cover is the operator form.

Theorem 2.3. If G is a reduced Λ-free context-free grammar then there is an operator grammar G' which completely covers G.

Proof: Let $G = (V, \Sigma, P, \perp S \perp)$ be a Λ-free grammar. First, we define a sequential transducer[5] $\mathfrak{T} = (\{q_o, q_1\}, V, V', H, q_o)$ where $V' = V \cup (\Sigma \times (V - \Sigma))$ and $H = \{(q_o, a, a, q_o), (q_1, a, a, q_o), (q_o, A, A, q_1), (q_1, A, a(a,A), q_1) \mid a \in \Sigma - \{\perp\}, A \in V - \Sigma\}$. Or graphically:



for each a in $\Sigma - \{\perp\}$ and A in $V - \Sigma$.

Fig. 3

It is clear that S has the following property:

Claim 1. For each x, y in V^*, $\mathfrak{T}(q_o,x) \cap \mathfrak{T}(q_o,y) \neq \emptyset$ if and only if $x = y$. The (trivial) proof of Claim 1 is omitted.

Thus, Claim 1 implies that for each $z \in \mathfrak{T}(V^*)$ there is a unique z' such that $z \in \mathfrak{T}(z')$. We call this correspondence φ and note that φ maps from $(V')^*$ into V^*.

Define $G' = (V', \Sigma, P', \perp S \perp)$ where $P' = P_1 \cup P_2 \cup P_3$ and $P_1 = \{A \to y \mid A \to x \text{ is in } P \text{ and } \varphi y = x\}$

---

[5] Cf. [8] for a definition of sequential transducers and their properties.

$P_2 = \{(a,A) \to (a,B)y \mid A \to x \text{ is in } P,\ a \in \Sigma,\ \varphi(By) = x,\ B \in V - \Sigma\}$

$P_3 = \{(a,A) \to y \mid A \to x \text{ is in } P,\ a \in \Sigma,\ \varphi(ay) = x\}$

It is clear from the construction of φ (i.e., from) that $\varphi^{-1}x$ is finite and thus P' is a finite set of productions. Also note that G is an operator grammar because no y in $\varphi(x)$ can contain two adjacent occurrences of variables.

Define a map ψ from P' into P as follows:

(i) For each $A \to y$ in P_1, $\psi(A \to y) = A \to \varphi(y)$

(ii) For each $(a,A) \to (a,B)y$ in P_2

$$\psi((a,A) \to (a,B)y) = A \to \varphi(By)$$

(iii) For each $(a,A) \to y$ in P_3

$$\psi((a,A) \to y) = A \to \varphi(ay)$$

Now, we can start our verification that we have a complete cover.

Claim 2. For each $a \in \Sigma$, $A \in V - \Sigma$, and $x \in \Sigma^*$,

(a) if $A \underset{R}{\overset{*}{\Longrightarrow}} ax$ in G by canonical derivation $(\pi_1,\ldots,\pi_n)$, $\pi_i \in P$

(b) then $A \underset{R}{\overset{*}{\Longrightarrow}} ax$ in G' by canonical derivation $(\pi'_1,\ldots,\pi'_n)$, $\pi'_i \in P'$

where $\psi\pi'_i = \pi_i$ for each i, $1 \le i \le n$ and $(a,A) \underset{R}{\overset{*}{\Longrightarrow}} x$ in G' by canonical derivation $(\pi''_1,\ldots,\pi''_n)$, each $\pi''_i \in P'$ and $\psi\pi''_i = \pi_i$ for each i, $1 \le i \le n$.

Proof: The argument is an induction on n and the trivial basis is omitted.

Let $\pi_i = A_i \to x_i$ for $1 \le i \le n$ and let $x_1 = u_0B_1u_1\ldots B_mu_m$ where $u_i \in \Sigma^*$, $B_i \in V - \Sigma$. Let $x = u_0v_1u_1\ldots v_mu_m$ where $B_i \overset{*}{\Longrightarrow} v_i \in \Sigma^*$ for

each i, $1 \leq i \leq m$. Thus the sequence $(\pi_i,\ldots,\pi_n)$ factors into $(\pi_1,(\pi_{1,1},\ldots,\pi_{1,p_1}),\ldots(\pi_{m,1},\ldots,\pi_{m,p_n}))$ where $(\pi_{i,1},\ldots,\pi_{i,p_i})$ is the generation sequence of $B_{m-i+1} \xRightarrow[R]{*} v_{m-i+1}$ for $1 \leq i \leq m$.

Consider the production $\pi'_1 = A_1 \to u'_o C_1 u'_1 \ldots C_m u'_m$ where $u'_o = u_o$, $C_1 = B_1$, and for each i so that $1 \leq i \leq m$:

If $u_i = \Lambda$ then $u'_i = {}^{(1)}v_i$ and $C_i = ({}^{(1)}v_i, B_i)$.

If $u_i \neq \Lambda$ then $u'_i = u_i$ and $C_i = B_i$.

Clearly $u'_o C_1 u'_1 \ldots C_m u'_m$ is in $\Psi(x_1)$ so $\psi(\pi'_1) = \pi_2$. By the induction hypothesis(b), there is a canonical derivation $(\pi'_{i,1},\ldots,\pi'_{i,p_i})$ of $C_{m-i+1} \xRightarrow[R]{*} v_{m-i+1}$ if C_{m-i+1} is in $V - \Sigma$ for $1 \leq i \leq m$. If C_{m-i+1} is in $\Sigma \times (V - \Sigma)$, there is a canonical derivation $(\pi'_{i,1},\ldots,\pi'_{i,p_i})$ of $C_{m-i+1} \xRightarrow[R]{*} v_{m-i+1}^{(\lg(v_{m-i+1})-1)}$. Further $\psi(\pi'_{i,1},\ldots,\pi'_{i,p_i}) = (\pi_{i,1},\ldots,\pi_{i,p_i})$ for each i. Thus the sequence $\pi' = (\pi'_1,\pi'_{1,1},\ldots,\pi'_{1,p_1},\ldots,\pi'_{m,1},\ldots,\pi'_{m,p_m})$ is a derivation of x in G' from A_1. Note that $\psi(\pi'_1) = (A_1 \to x_1)$ and $\psi(\pi'_{i,j}) = \pi_{i,j}$ by the induction hypothesis. Thus $\psi(\pi') = (\pi_1,\ldots,\pi_n)$

The second part of the statement follows analogously with the modification that $\pi'_1 = (a,A) \to u'_o C_1 \ldots C_m u'_m$ where $a = {}^{(1)}x$. If $u_o = ay$, $a \in \Sigma$ then $u'_o = y$ while $u_o = \Lambda$ implies $u'_o = \Lambda$ and $C_1 = (a,B_1)$. All other quantities remain the same in the proof of Claim 2.

Next, we must turn to the converse.

Claim 3. For each $A \in V - \Sigma$, $x \in \Sigma^*$ and $a \in \Sigma$, if $(a,A) \xRightarrow[R]{*} x$ in G' by canonical derivation $(\pi_1,\ldots,\pi_n)$ or if $A \xRightarrow{*} ax$ in G' by canonical derivation $(\pi_1,\ldots,\pi_n)$ then $A \xRightarrow[R]{*} ax$ in G by canonical derivation $(\psi\pi_1,\ldots,\psi\pi_n)$.

The proof of Claim 3 is again an induction. Since it is similar to the proof of Claim 2, it is omitted.

To complete the proof, we only need to note that $L(G) = L(G')$ but this follows from Claims 2 and 3. ∎

As an example of the construction, consider the following grammar:

$$I \to D \mid ID$$

$$D \to 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9$$

with $V - \Sigma = \{I,D\}$. Using the construction, we obtain

$$I \to D \mid I0(0,D) \mid I1(1,D) \mid \cdots \mid I9(9,D)$$

$$D \to 0 \mid 1 \mid 2 \mid \cdots \mid 9$$

and for each i, $0 \leq i \leq 9$,

$$(i,D) \to \Lambda$$

Note that the resulting grammar is not necessarily Λ-free even if the original one is.

Theorem 2.3 is surprising. It was known [10] that every grammar was equivalent to one in operator form but the use of the Greibach form in the construction of the operator form destroyed the structure of the trees in an essential way as the next result will show.

The previous results are typical of the kind of positive theorem which can be obtained concerning covers. We can show that not every transformation yields a cover. For instance, we will now show that parsing the Greibach normal form G' of a grammar G is of no great help in parsing G since one must unravel the left recursions.

Theorem 2.4. Let G be the following context-free grammar:

$$S \rightarrow S0 \mid S1 \mid 0 \mid 1$$

There is no grammar $G' = (V', \Sigma', P', \perp S' \perp)$ in Greibach normal form such that (G',H') covers (G,P) under φ for any $H' \subseteq P'$ and φ mapping H' into P.

Proof: Suppose there is a grammar $G' = (V', \Sigma', P', \perp S' \perp)$ in Greibach form such that G' is reduced and there exist $H' \subseteq P'$ and φ so that (G',H') covers (G,P) under φ.

Claim 1. $H' = P' \subseteq (V' - \Sigma) \times (\Sigma(V' - \Sigma)^*)$.

Proof: Suppose $x \in \Sigma^+$ and $\perp S' \perp \xrightarrow[R]{*} \perp x \perp$ in G' with canonical derivation $\pi = (\pi_1,\ldots,\pi_n)$. The H'-sparse generation of π, π' has the property that[5] $\varphi(\pi')$ is a P-sparse generation of $\perp x \perp$. But then $\varphi(\pi')$ is a generation of $\perp x \perp$ in G. Since each rule of P contributes exactly one terminal character to x thus $n \geq lg(x)$. Since each π_i in P' contributes at least one terminal character to x it follows that $lg(x) \geq n$. Thus $n = lg(x)$ and each π_i contributes exactly one terminal character to x, and each $\pi_i \in H'$. Since G' is reduced it follows that $H' = P'$ and each $\pi_i \in P'$ contains exactly one terminal character. So since G' is in Greibach normal form $P' \subseteq N' \times (\Sigma N'^*)$.

Since every production in P' contains exactly one terminal character, the following result holds.

---

[5] If $\pi = (\pi_1,\ldots,\pi_n)$ is a sequence of elements chosen from set Q and φ is a partial function from Q into Q'. Then if $(\pi_{i_1},\ldots,\pi_{i_j})$ is the subsequence of members of the domain of φ, $\varphi(\pi) = (\varphi(\pi_{i_1}),\ldots, \varphi(\pi_{i_j}))$.

Claim 2. For any A in $V' - \Sigma$, x in $(V')^*$, $A \overset{n}{\underset{R}{\Longrightarrow}} x$ in G' ~~if and only if~~[6] $\#_\Sigma(x) = n$.

Next, we partition the variables of V' according to their "self embedding" properties. Define

$L = \{A \in V' - \Sigma \mid A \overset{+}{\underset{R}{\Longrightarrow}} xAy \text{ for some } x,y \in (V')^*\}$ and $M = (V' - \Sigma) - L$.

Finally, define Q as the set of all sentential forms which may be generated without using productions whose left-hand side is in L, i.e.,

$$Q = \{x \in (V')^* \mid \perp S' \perp \overset{*}{\underset{\blacksquare}{\Longrightarrow}} x \text{ by } (A_1 \to x_1, \ldots, A_n \to x_n)$$

$$\text{and} \qquad A_i \notin M \text{ for each } i,\ 1 \le i \le n\}.$$

Since the words in Q come from trees of bounded height and hence of bounded width, we have established the following claim.

Claim 3. Q is finite.

Next, we must obtain more detailed information about derivations in G'.

Claim 4. For each $x \in (V')^*$, $A \in V' - \Sigma$, and $z \in \Sigma^*$; if $\perp S' \perp \overset{*}{\underset{R}{\Longrightarrow}} \perp xAz \perp$ in G' and $\#_\Sigma(x) = k$ then there is a $y_A \in \Sigma^k$ so that

$$\Sigma^*\{u \in \Sigma^* \mid A \overset{*}{\underset{G'}{\Longrightarrow}} u\} \subset \Sigma^*\{y_A\}$$

Proof: Let $\perp S' \perp \overset{*}{\underset{R}{\Longrightarrow}} \perp xAz \perp$ in G' by canonical derivation $(\pi_i)_{i=1}^n$. Now $\varphi(\pi_i)_{i=1}^n$ is a generation in G of $\perp Sw \perp$ for some $w \in \Sigma^*$.

---

[6] For each $a \in V$ and x in V^*, let $\#_a(x)$ denote the number of occurrences of a in x. Let $\#_\Sigma(x) = \sum_{a \in \Sigma} \#_a(x)$ so $\#_\Sigma(x)$ is the number of occurrences terminals in x.

First note that $lg(w) = n$ since each production $\varphi(\pi_i)$ contributes exactly one character to w. Also note by Claim 2 that $n = \#_\Sigma(xAz) = \#_\Sigma(x) + \#_\Sigma(A) + \#_\Sigma(z) = k + lg(z)$. So $lg(w) = lg(z) + k$. Now suppose $xA \underset{R}{\overset{*}{\Longrightarrow}} u$ in G' by $(\pi_i)_{i=n+1}^{m}$. Then $\perp S' \perp \underset{R}{\overset{*}{\Longrightarrow}} \perp uz \perp$ in G' by $(\pi_i)_{i=1}^{m}$ and $\varphi(\pi_i)_{i=1}^{m}$ is a derivation $\perp S \perp \underset{R}{\overset{*}{\Longrightarrow}} \perp u'w \perp = \perp uz \perp$ in G. So since $lg(w) = lg(z) + k$, $w = (u^{(k)})z$. Thus w uniquely determines $u^{(k)}$. So Claim 4 is established.

Claim 4 immediately yields:

Claim 5. For each $A \in L$, $L(A)$[7] has the property that for each $y_1, y_2 \in L(A)$, $lg(y_1) < lg(y_2)$ implies $zy_1 = y_2$ for some $z \in \Sigma^*$.

Proof: Since A is in L, there exist $x, z \in (V')^*$ so that $A \underset{R}{\overset{+}{\Longrightarrow}} xAz$, and because G' is in Greibach form, $\#_\Sigma(x) \geq 1$. Thus the number of occurrences of terminals which may preceed A in a sentential form is unbounded.

By Claim 4, we have shown that for any $n > 0$, there exists $y_A \in \Sigma^*$, $lg(y_A) = n$ and $L(A) \subset \Sigma^*\{y_A\}$. Note that all strings of length less than n in L(A) are suffixes of y_A.

To see that the claim is satisfied, let y_1, y_2 be in L(A) and let $n = lg(y_2) \geq lg(y_1)$. By the above there exists $y \in \Sigma^*$, $lg(y) = n$ so that $y = x_1y_1 = x_2y_2$ for some $x_i \in \Sigma^*$. Since $lg(y) = lg(y_1)$ it follows that $x_1 = \Lambda$ and $y_1 = x_2y_2$ for some $x_2 \in \Sigma^*$.

Our last formal claim is set theoretic in nature.

Claim 6. $\Sigma^+ = \bigcup_{x \in Q} \{x_0y_1x_1 \ldots y_nx_n \mid n \geq 0,\ x = x_0A_1x_1 \ldots A_nx_n,$
$x_i \in \Sigma^*,\ A_i \in L,\ y_i \in L(A_i)\}$

---

[7] For each $A \in V$, $L(A) = \{x \in \Sigma^* \mid A \overset{*}{\Longrightarrow} x\}$.

Proof: Recalling that $L(S') = L(S) = \Sigma^+$ we have that

$$\Sigma^+ = L(S') = \bigcup_{\substack{x \in Q \\ x = x_o A_1 \ldots A_n x_n}} \{x_o\} L(A_1)\{x_1\} \ldots L(A_n)\{x_n\}$$

$$= \bigcup_{x \in Q} \{x_o y_1 x_1 \ldots y_n x_n \mid n \geq 0,\ x_i \in \Sigma^*,\ A_i \in L,\ y_i \in L(A_i)\}$$

Let x be in Q, $x = x_o A_1 x_1 \ldots A_n x_{n_x}$, $n \geq 0$, $x_i \in \Sigma^*$, $A_i \in L$, and let $k_x = \#_\Sigma(x)$. Let j be any positive integer and define $m = k_x + j$. Denote by[8] Part(j), the set of all partitions of j. Let

$$T_{x,j} = \{z \in \Sigma^m \mid x \xrightarrow{*} z\} = \bigcup_{(i_1,\ldots,i_{n_x}) \in \text{Part}(j)} \{x_o\}(L(A_1) \cap \Sigma^{i_1}) \ldots (L(A_n) \cap \Sigma^{i_{n_x}})\{x_{n_x}\}.$$

Since $|L(A_i) \cap \Sigma^i| \leq 1$ by Claim 5.

$$|T_{x,j}| \leq \sum_{(i_1,\ldots,i_{n_x}) \in \text{Part}_{n_x}(j)} 1 = |\text{Part}_{n_x}(j)| < cj^{n_x - 1}$$

for some constant c.

Let $k = \max\{k_x \mid x \in Q\}$ and $n = \max\{n_x \mid x \in Q\}$. Clearly k and n exist since Q is finite by Claim 3. Now by Claim 6,

---

[8] If $j \geq 0$ we say that $(i_1,\ldots,i_r)$ is a partition of j if $i_1 + \ldots + i_r = y$ and each $i_k \geq 0$. A partition $(i_1,\ldots,i_r)$ with each $i_j > 0$ is said to have r parts. Let $\text{Part}_n(j)$ be the set of partitions of j into exactly n parts. It is well known that $|\text{Part}_n(j)| < cj^{n-1}$ for some $c > 0$.

$$\Sigma^+ = \{y \in \Sigma^* \mid x \overset{*}{\Longrightarrow} y \text{ for some } x \in Q\}$$

$$= \bigcup_{x \in Q} \{y \in \Sigma^* \mid x \overset{*}{\Longrightarrow} y\}$$

$$= \bigcup_{x \in Q} \bigcup_{i=0}^{\infty} \{y \in \Sigma^i \mid x \overset{*}{\Longrightarrow} y\}$$

$$= \bigcup_{x \in Q} \bigcup_{i=k}^{\infty} T_{x,i-k_x} \cup \bigcup_{i=1}^{k} \Sigma^i$$

So for $i > k$ we conclude

$$\Sigma^i = \bigcup_{x \in Q} T_{x,i-k_x}$$

so $|\Sigma^i| = 2^i \leq \sum_{x \in Q} c(i-k_x)^{n_x} \leq \sum_{x \in Q} ci^n \leq c'i^n$. But for large i, $2^i > i^n$. This contradiction shows $\bigcup_{x \in Q} \{y \mid x \overset{*}{\Longrightarrow} y\} \neq \Sigma^*$ so $L(G') \neq L(G)$. Thus the theorem has been established. ∎

Section 3

Phrase Detection

The key to giving a generalized theory of phrase detection through precedence analysis is in the following definition.

Definition. Let $G = (V, \Sigma, P, \perp S \perp)$ be a context-free grammar. A set $T \subseteq V$ is said to be a token set for G if

(1) $\Sigma \subseteq T$

(2) For each $A \in T$, it is not the case that $A \overset{*}{\Longrightarrow} \Lambda$

(3) For each $A \in T$, $B \in V$, $A \overset{*}{\Longrightarrow} B$ implies $B \in T$.

The definition of token set is quite weak. Tokens must never completely disappear (i.e., generate a string which does not contain a token) and every terminal character must be a token.

It is clear that the token sets of a grammar G form a sublattice of the lattice of all subsets of V. This lattice has minimal element Σ, maximal element $M = \{A \in V \mid \text{not } A \overset{*}{\Longrightarrow} \Lambda\}$, and for any $R \subseteq V$, R may be extended to a token set by adjunction of more elements if and only if $\Sigma \subseteq R \subseteq M$. For the latter case, the minimum token set over R will be $\{A \in V \mid B \overset{*}{\Longrightarrow} A \text{ for some } B \in R\}$.

It will turn out that a token set must satisfy a stronger property (i.e., be an operator set) for precedence analysis to succeed. This is related to the "binding" between operators and operands which is enforced by precedence relations.

Definition. Let $G = (V, \Sigma, P, \perp S \perp)$ be a context-free grammar and let T be a token set for G.

(1) T is a Colmerauer operator set (COP for short) if for each x, $y \in V^*$; A, B, C $\in$ V; $A \rightarrow xBCy$ in P implies B $\in$ T or C $\in$ T.

(2) T is a Floyd operator set (FOP for short) if for each x, $y \in V^*$; A, B, C $\in$ V; A $\rightarrow$ xBCy in P implies B $\in$ T or $\mathcal{L}(C) \subseteq T$ where

$$\mathcal{L}(C) = \{D \in V \mid C \xrightarrow[R]{*} Dx \text{ for some } x \in V^*\}.$$

(3) T is a strong operator set (SOP for short) if for each x, $y \in V^*$, A, B, C $\in$ V; A $\rightarrow$ xBCy in P implies B $\in$ T or C $\in \Sigma$.

If T is an operator set (of any type) then the elements of $V - T = \overline{T}$ are called operands.

The following proposition is immediate.

Proposition 3.1. Let $G = (V, \Sigma, P, \perp S \perp)$ be a context-free grammar and T a token set for G. If T is a SOP for G then T is a FOP for G. If T is a FOP for G then T is a COP for G.

It is the concept of a SOP which plays the key role in what follows. It will be seen that a "canonical precedence scheme" yields a canonical sparse parse if and only if the underlying operator set is a SOP. This will characterize the case in which Colmerauer's precedence scheme [2] yields the canonical parse and also will unify the precedence methods of Floyd [4] and of Wirth and Weber [19].

Colmerauer [2] restricted attention to Λ-free grammars and considered a form of a COP. From this, he obtained a necessary condition for the existence of a canonical precedence parser. Our approach yields necessary and sufficient conditions for the existence of such a parser. When $T = \Sigma$ or $T = V$, the definitions of COP and SOP coincide but they differ at other points in the semilattice of operator sets.

It is important to observe that one can easily decide if a token set T is a SOP (FOP) (COP) for a grammar G by examining the productions.

Proposition 3.2 characterizes the difference between these types of token sets. The following example may be useful in understanding these definitions.

Example:

$$S \rightarrow ASA \mid a$$

$$A \rightarrow a$$

The token sets are:

$$\{a,S\} \quad \{a,A\} \quad \{a,S\} \quad \{a,A,S\}$$

$\{a,S\}$ is a COP but not a FOP or SOP

$\{a,A\}$ is a FOP but not a SOP

$\{a,A,S\}$ is a SOP

At the poles of $T = V$ or $T = \Sigma$ the concepts of FOP, COP and SOP coincide as the following result shows.

Theorem 3.1. Let $G = (V, \Sigma, P, \perp S \perp)$ be a context-free grammar.

(a) G is an operator grammar

(i) if and only if Σ is a SOP.

(ii) if and only if Σ is a FOP.

(iii) if and only if Σ is a COP.

(b) V is a token set for G

(i) if and only if V is a COP.

(ii) if and only if V is a FOP.

(iii) if and only if V is a SOP.

(iv) if and only if G is Λ-free.

Proof: (a) If G is an operator grammar then Σ is clearly a SOP and hence a FOP and COP by Proposition 3.1. To complete the proof, it suffices to show that if Σ is a COP then G is an operator grammar. To do this, note that if Σ is a COP, then $A \to xBCy$ implies $B \in \Sigma$ or $C \in \Sigma$ which implies that G is an operator grammar.

(b) If V is a COP then it must be a token set. By Proposition 3.1, it suffices to show that if V is a token set then V is a SOP. To do this note that $A \to xBCy$ implies $B \in V$ or $C \in V$ is a tautology. Note that V is a token set if and only if G is Λ-free. ∎

Our next result characterizes the distinction between FOP's, COP's and SOP's. First a new definition is required.

Definition. Let V be any finite set and $T \subseteq V$. A string $x \in V^*$ is a T-infix string if for each $y, z \in V^*$; $A, B \in V$, $x = yABz$ implies $A \in T$ or $B \in T$.

In our applications, T will be a token set of the vocabulary V of some grammar. Note that T-infix strings are substring closed, i.e., if uvw is a T-infix string then v is a T-infix string.

Theorem 3.2. Let $G = (V, \Sigma, P, \perp S \perp)$ be a reduced context-free grammar and T a token set.

(a) T is a SOP if and only if for each $x, y \in V^*$; $A, B \in V$;

$$\perp S \perp \underset{R}{\overset{*}{\Longrightarrow}} xABy \text{ implies } A \in T \text{ or } B \in \Sigma.$$

(b) Let G be reduced. T is a FOP if and only if for each $x \in V^*$,

$$\perp S \perp \underset{R}{\overset{*}{\Longrightarrow}} x \quad \text{implies} \quad x \text{ is a T-infix string.}$$

(c) T is a COP if and only if for each $z_n \in \Sigma^*$ such that there exist strings $z_1, \ldots, z_{n-1}$ so that

$$\perp S \perp = z_o \Longrightarrow \ldots \Longrightarrow z_n \qquad (*)$$

then there exist strings $y_i \in V^*$ so that

(i) $\perp S \perp = y_o \Longrightarrow \ldots \Longrightarrow y_n = z_n$ (**)

(ii) (*) and (**) have the same canonical[9] generation and

(iii) each y_i is a T-infix string.

Proof of (a): The result is trivial in one direction and a straightforward induction in the other direction.

Proof of (b): Let T be a FOP for G and suppose that $\perp S \perp \underset{R}{\overset{*}{\Longrightarrow}} z$ where z is a T-infix string. It suffices to show that if $z \underset{R}{\Longrightarrow} z'$ then z' is also a T-infix string. If $z \in \Sigma^*$, we are done so suppose $z = \mu A w$, $A \in V - \Sigma$, $w \in \Sigma^*$ and $z' = \mu v w$. Since T is a FOP, we must have that v is a T-infix string and then vw is a T-infix string. If $\mu = \Lambda$, we are done so suppose $\mu = xB$ with $x \in V^*$, $B \in V$.

---

[9] Condition (ii) means that the parse tree of (*) is preserved so that (**) is a reordered generation of (*). The generation is reordered in that the variables in z_o are expanded in a different order though the same productions are used at the same places.

Suppose, for the sake of contradiction, that $z' = \mu vw$ is not a T-infix string. We must have that $B \notin T$, $vw = Cy$ for some $C \in V$ with $C \notin T$ because μ is a T-infix string (it is a substring of z) and also vw is a T-infix string. Now C cannot be the leftmost character of w since $C \in \Sigma$ and $w \in \Sigma^*$. Thus C is the leftmost character of v and $C \in \mathcal{L}(A)$.

Let the canonical derivation of $\perp S \perp \underset{R}{\overset{*}{\Longrightarrow}} \mu Aw$ be $(z_o,\ldots,z_n)$. Let i be the least integer so that there exist $y_o,\ldots,y_{n-i}$ so that $z_i = \mu y_o$ and $z_{i+j} = \mu y_j$ for $j = 0,\ldots,n-i$. Let E be the leftmost character of y_o, that is $y_o = Ey_o'$. [Exists since $vw \neq \Lambda$]. Then $z_i = xBEy_o'$. Surely $E \in V - \Sigma$ because $A \in \mathcal{L}(E)$ and $A \in V - \Sigma$. By the minimality of i, $z_{i-1} = w_1Fw_2$, with $w_2 \in \Sigma^*$ and F in $V - \Sigma$, $F \rightarrow w_3BEw_4$ is in P, $x = w_1w_3$ and $y_o' = w_4w_2$. Since $B \notin T$ and T is a FOP, $\mathcal{L}(E) \subseteq T$. But $A \in \mathcal{L}(E)$ and $C \in \mathcal{L}(A)$; so $C \in \mathcal{L}(E) \subseteq T$. Thus $C \in T$ which contradicts that $C \notin T$. Therefore $\mu vw = z'$ is a T-infix string.

Conversely, suppose that G satisfies the condition of (b) and suppose that T is not a FOP for G. That is, there is a production $A \rightarrow xBCy$ in P with $B \notin T$ and there is $z \in V^*$ so that $C \underset{R}{\overset{*}{\Longrightarrow}} Dz$ with $D \notin T$. Since G is reduced, there is y' in Σ^* $y \overset{*}{\Longrightarrow} y'$ and hence $A \underset{R}{\Longrightarrow} xBCy \underset{R}{\overset{*}{\Longrightarrow}} xBDzy'$. Note that $xBDzy'$ is not a T-infix string. Since G is reduced, $\perp S \perp \overset{*}{\Longrightarrow} \mu Av$ for some $\mu,v \in V^*$, so that $\perp S \perp \overset{*}{\Longrightarrow} \mu xBDzy'v$ and $\mu xBDzy'v$ is not a T-infix string. This contradiction establishes that T is a FOP for G.

Proof of (c): Let T be a COP for G. The argument is an induction on n where we assume that z_o is any T-infix string so that

$$z_o \Longrightarrow \ldots \Longrightarrow a_n$$

with z_n a T-infix string. We must show that there exist $y_o,\ldots,y_n$ so that

$$z_o = y_o \Longrightarrow \ldots \Longrightarrow y_n = z.$$

with the same parse tree as (1) and each y_i is a T-infix string.

If $n \leq 1$, the result is immediate. If z_1 is a T-infix string, the induction hypothesis applies to $z_1 \overset{*}{\Longrightarrow} z_n$ and so the induction extends to $z_o \overset{*}{\Longrightarrow} z_n$ and finally we take $z_o = \perp S \perp$. Thus we are done unless z_1 is not a T-infix string, i.e., $z_o = \mu Aw$, $z_1 = \mu vw$ where z_o is a T-infix string and z_1 is not. Thus μ and w are T-infix strings. Since T is a COP for G, we know that v is a T-infix string.

Observe that z_i is a T-infix string if and only if $\perp z_i \perp$ is a T-infix string. Thus, we may assume, without loss of generality that $\mu \neq \Lambda \neq w$. Thus we can write $\mu = \mu'U$ and $w = Ww'$ with $U, W \in V$. If $v = \Lambda$, then $A \rightarrow \Lambda$ is in P which implies that $A \notin T$ since T is a token set. Thus $U \in T$ and $W \in T$ since μAw is a T-infix string. Thus if $v = \Lambda$, then $\mu vw = \mu w$ is a T-infix string and we are done. So assume $v \neq \Lambda$.

Let $z_o = \mu Aw = X_o \ldots X_n$ and $z_1 = X_1 \ldots X_{j-1}\ Y_1 \ldots Y_p\ X_{j+1} \ldots X_m = Z_1 \ldots Z_{m+p-1}$ where the X_i, Y_i, and Z_i are in V. Note that $v = Y_1 \ldots Y_p$. If z_1 is not a T-infix string then there exists i so that Z_{i-1} and $Z_i \in V - T$. If $i < j$ or $i > j+p$ or $j < i < j+p-1$ then $Z_{i-1} \in T$ or $Z_i \in T$ since μ, v, and w are T-infix strings. Thus, if Z_{i-1} and Z_i are not in T, we must have that $i = j$ or $i = j+p$. The two cases are similar so we only deal with the case where $i = j$. Then we have $Z_{j-1} = X_{j-1} \notin T$ so that $X_j = A \in T$ since z_o is a T-infix string.

The derivation $uvw \overset{*}{\Longrightarrow} z_n$ gives rise to the factorization $z_n = u'v'w'$ where $u \overset{*}{\Longrightarrow} u'$, $v \overset{*}{\Longrightarrow} v'$, and $w \overset{*}{\Longrightarrow} w'$. Let these derivations be

$$\begin{aligned} u = u_o &\Longrightarrow \ldots \Longrightarrow u_q = u' \\ v = v_o &\Longrightarrow \ldots \Longrightarrow v_r = v' \\ w = w_o &\Longrightarrow \ldots \Longrightarrow w_s = w' \end{aligned} \tag{1}$$

Since $q, r, s \leq n$ and u, v, w, u', v', and w' are T-infix strings the induction hypothesis applies and the derivations may be reordered as

$$\begin{aligned} u = u'_o &\Longrightarrow \ldots \Longrightarrow u'_q = u' \\ v = v'_o &\Longrightarrow \ldots \Longrightarrow v'_r = v' \\ w = w'_o &\Longrightarrow \ldots \Longrightarrow w'_s = w' \end{aligned} \tag{2}$$

and

where each u'_i, v'_i and w'_i is a T-infix string.

Consider the derivation

$$\begin{aligned} uAw \quad & uAw'_1 \Longrightarrow \ldots \Longrightarrow uAw' \\ & u_1Aw' \Longrightarrow \ldots \Longrightarrow u'Aw' \\ & u'vw' \Longrightarrow \ldots \Longrightarrow u'v'w' \end{aligned} \tag{3}$$

Claim. Each string in derivation (3) is a T-infix string.

Proof: Note that if $B \in T$ and y, z are T-infix strings then yBz is a T-infix string. From this fact and the fact that $A \in T$, we observe that the following are all T-infix strings:

uAw'_i for each i such that $0 \leq i \leq s$;

$u'Aw'$ for each i such that $0 \leq i \leq q$.

Thus $u'Aw'$ is a T-infix string. Also recall that $u', w' \in \Sigma^+$ so write $u' = u'_1b$ and $w = cw'_1$ for $b, c \in \Sigma$.

By a previous remark, $u_1'bv_i'$ is a T-infix string for each i such that $0 \leq i \leq r$ since $b \in \Sigma \subseteq T$. Therefore $u_1'bv_i'cw_1'$ is a T-infix string for each i such that $0 \leq i \leq r$ and this completes the proof of the claim.

The result now follows from the fact that we have only reordered the derivation and not changed the generation tree.

Conversely, note that if T is a token set for G and if the condition holds, then if $A \rightarrow x$ is in P x must be a T-infix string. If $x = x'BCy$ then $B \in T$ or $C \in T$ since x is a T-infix string but this means that T is a COP for G. ∎

The above theorem characterizes the relationships between the types of operator sets. The COP sets are the direct generalization of the definition of operator grammar as presented by Floyd [5]. Colmerauer adopted this definition in his work [2]. The FOP are the generalization of the definition of operator sets which satisfies the constraint that every canonical sentential form be an infix string. This constraint is not strong enough for the definition of canonical precedence scheme. The definition of SOP is exactly what is needed to satisfy the first part of clause (iv) (b) of the definition of canonical precedence scheme. In other words it may be stated as: The operands in the "neighborhood" of an operator are reduced with that operator. This is the relationship between binding of operators to operands and the relative precedence of operators. The following results make this explicit.

Theorem 3.2A: Let $G = (V, \Sigma, P, \perp S \perp)$ be a reduced context free grammar with token set T. Then T is an SOP iff

(*) for all $x, y, z \in V^*$, $A \in N$, if

$$\dashv S \vdash \underset{R}{\overset{*}{\Longrightarrow}} x A z \underset{R}{\Longrightarrow} x y z \quad \text{then}$$

$$x \in V^* T \quad \text{and} \quad z \in \Sigma^*$$

Proof: In the forward direction note that $z \in \Sigma^*$ trivially since the derivation is rightmost. Further $A \notin \Sigma$ since $A \to y$ is in P. Theorem 3.2(a) showed that if T is a SOP then for any $x', z \in V^*$, $A, B \in V$, if $\dashv S \vdash \underset{R}{\overset{*}{\Longrightarrow}} x'B A z$ then $B \in T$ or $A \in \Sigma$. The previous two remarks combine to prove that $x = x'B$ for some $B \in T$. So $x \in V^*T$.

Conversely, suppose T satisfies (*). Proceed by contradiction. If T isn't an SOP then for some $C \in V$; $B \in \overline{T}$; $A \in N$; $u, v \in V^*$, $C \to u B A v$ is in P. Since G is reduced there exist $x, y, z \in V^*$; $v' \in \Sigma^*$, such that:

$$\dashv S \vdash \underset{R}{\overset{*}{\Longrightarrow}} x C z \underset{R}{\Longrightarrow} x u B A v z \underset{R}{\overset{*}{\Longrightarrow}} x u B A v' z$$

$$\underset{R}{\Longrightarrow} x u B y v' z.$$

But $x u B \notin V^*T$ so (*) is contradicted. Thus T must be an SOP if (*) is satisfied. ∎

Corollary 3.2A: Let G be as above. Then the token set T is a SOP for G iff

(*') for every $xyz = A_o x_1 A_1 \ldots x_n A_n$ where $A_i \in T$, $x_i \in \overline{T}^*$, if

$$\dashv S \vdash \underset{R}{\overset{*}{\Longrightarrow}} xAz \underset{R}{\Longrightarrow} xyz$$

for some $0 \leq i \leq j \leq n$

then $x = A_o \ldots A_{i-1}$

$y = x_i A_i \ldots A_{j-1} x_j$

$z = A_j x_{j+1} \ldots x_n A_n$

In order to simplify the presentation in this section, we will only deal with precedence detection methods which yield the canonical parse. More precisely, this means that we will detect (using precedence relations) the leftmost phrase which contains an operator. Repeated application of this detection, followed by reduction, leads to an H-sparse parse where $H = \{A \to x \text{ in } P \mid x \in (V - T)^*\}$.

In Section 4, an algorithm for general parsing is given and the restriction to the canonical case is relaxed.

We now formalize canonical precedence schemes.

Definition. A canonical precedence scheme is a 5-tuple $(G, T, <\cdot, \doteq, \cdot>)$ where

(i) $G = (V, \Sigma, P, \perp S \perp)$ is a context-free grammar.

(ii) T is a token set for G.

(iii) $<\cdot$, $\doteq$, and $\cdot>$ are binary relations on T which are pairwise disjoint.

(iv) for any $n \geq 0$; $x_1,\ldots,x_n \in (V - T)^*$; $A,A_o,\ldots,A_n \in T$; $x, z \in V^*$; if $\perp S \perp \xrightarrow[R]{*} xAz \xrightarrow[R]{} A_o x_1 A_1 \ldots x_n A_n$ by the rule $A \to y$ then either

(a) $y \in (V - T)^*$

or

(b) there exist $0 < i \leq j < n$ so that

$$x = A_o x_1 A_1 \ldots A_{i-1}$$

$$y = x_i A_i \ldots A_j x_{j+1}$$

$$z = A_{j+1} x_{j+2} A_{j+2} \ldots x_n A_n$$

with

$$A_{i-1} \lessdot A_i,$$

$$A_k \doteq A_{k+1} \quad \text{for each } k \text{ such that } i \leq k < j,$$

and

$$A_j \gtrdot A_{j+1}.$$

Further if G is a grammar with SOP T we say G is <u>T-precedence detectable</u> if there exists relations $\lessdot$, $\doteq$, $\gtrdot$ such that $(G, T, \lessdot, \doteq, \gtrdot)$ is a canonical precedence scheme. If $T = V$ we say G is <u>simple precedence detectable</u> and if $T = \Sigma$ we say G is <u>Floyd precedence detectable</u>.

It should be clear (and we shall prove) that this definition is an abstraction of the known precedence methods. Phrase detection will proceed from left to right by searching for a string of tokens bracketed by the relations as

$$A_{i-1} \lessdot A_i \doteq \ldots \doteq A_j \gtrdot A_{j+1}.$$

Note that operands are ignored. Also note that <u>all</u> of x_i and x_{j+1} must

be included in y. That is all operands adjacent to an operator are "bound" to that operator when it is reduced. This is a fundamental assumption of precedence analysis. We have not investigated other binding schemes.

Section 4 will explain more precisely how the scheme is used.

Definition. A context-free grammar $G = (V, \Sigma, P, \perp S \perp)$ is said to be P-reduced if G is reduced and for each $A \neq S$ there exists $x \in \Sigma^+$ so that $A \overset{*}{\Longrightarrow} x$.

In addition to being reduced, every variable different from S can produce at least one non-null word in a P-reduced grammar. It is easily seen that every grammar has an equivalent P-reduced grammar.

Our next result justifies the definition of SOP's.

Theorem 3.3. Let $G = (V, \Sigma, P, \perp S \perp)$ be a P-reduced context free grammar and let T be a token set for G. If $(G, T, <\cdot, \doteq, \cdot>)$ is a canonical precedence scheme then T is a SOP.

Proof: Suppose, for the sake of contradiction that T is not a SOP for G. Then there is $A \in \overline{T}$, $B, C \in V - \Sigma$; $x, z \in V^*$ so that $C \to xABz$ is in P. Let $xABz = U_1 \ldots U_{n_1}$ with $A = U_{n_A}$ and $B = U_{n_B}$. Since G is P-reduced, there is $y \in \Sigma^+$ and $y_0, \ldots, y_m \in V^*$ so that $B = y_0 \underset{R}{\Longrightarrow} y_1 \Longrightarrow \ldots \Longrightarrow y_m = y$. Let $y_i = Y_{i_1} \ldots Y_{i_m}$ for each i. Let i be the least positive integer such that $Y_{i_j} \in T$ for some j. Clearly i exists since $y_m = y \in T^+$ and hence $Y_{n_1} \in \Sigma \subseteq T$. Since G is P-reduced, there exist $V_1, \ldots, V_{n_2}, X_1, \ldots, X_{n_5} \in V$; $W_1, \ldots, W_{n_3}$, $Z_1, \ldots, Z_{n_4} \in \Sigma$ so that $n_2 > 0$, and

$$\perp S \perp \underset{R}{\overset{*}{\Longrightarrow}} V_1 \cdots V_{n_2} C W_1 \cdots W_{n_3} \underset{R}{\overset{*}{\Longrightarrow}} V_1 \cdots V_{n_2} U_1 \cdots U_{n_1} W_1 \cdots W_{n_3}$$

$$\underset{R}{\overset{*}{\Longrightarrow}} V_1 \cdots V_{n_2} U_1 \cdots U_{n_B} Z_1 \cdots Z_{n_4} = X_1 \cdots X_{n_5}$$

We have $U_{n_B-1} = U_{n_A} = A \in T$. Let k be the largest integer less than $n_A + n_2$ such that $X_k \in T$. Since $X_1 = \perp \in T$, we know that k exists. Now $U_{n_B} = B \underset{R}{\overset{*}{\Longrightarrow}} y_{i-1}$ so that

$$X_1 \cdots X_{n_5} \underset{R}{\overset{*}{\Longrightarrow}} X_1 \cdots X_{n_2 + n_A} Y_{i-1,1} \cdots Y_{i-1,m_{i-1}} Z_1 \cdots Z_{n_4}$$

The generation $y_{i-1} \underset{R}{\Longrightarrow} y_i$ is by production. $Y_{i-1,q} \rightarrow Y_{i,q} \cdots Y_{i,q+r}$ where $Y_{i-1,q}$ is the rightmost variable of y_{i-1}. We chose i so that y_{i-1} contained no tokens and y_i was chosen to contain a token. Let $Y_{i,t}$ be the leftmost token in y_i and note that $q \leq t \leq q+r$. Thus we have

$$X_1 \cdots X_{n_2 + n_A} Y_{i-1,1} \cdots Y_{i-1,m_{i-1}} Z_1 \cdots Z_{n_4}$$

$$\Longrightarrow X_1 \cdots X_{n_2 + n_A} Y_{i-1,1} \cdots Y_{i-1,q-1} Y_{i,q} \cdots Y_{i,q+r} Y_{i-1,q+1} \cdots Y_{i-1,m_{i-1}}$$

$$Z_1 \cdots Z_{n_4} = w$$

We claim that

$$X_{k+1} \cdots X_{n_2 + n_A} Y_{i-1,1} \cdots Y_{i-1,q-1} \in (V - T)^+ \qquad (*)$$

because $k < n_A + n_2$ implies $X_{k+1} \cdots X_{n_2 + n_A} \in (V - T)^*$ and i was chosen so that $Y_{i-1} \cdots Y_{i-1,q-1} \in (V - T)^+$.

Furthermore, $Y_{i,q}\cdots Y_{i,q+r}$ contains the leftmost token $Y_{i,t}$. Since $(G, T, \lessdot, \doteq, \gtrdot)$ is a canonical precedence scheme, we have

$$X_k \lessdot Y_{i,t}$$

and

$$X_{k+1}\cdots X_{n_A} + n_2 Y_{i-1,1}\cdots Y_{i-1,q-1} Y_{i,q}\cdots Y_{i,t-1} = Y_{i,q}\cdots Y_{i,t-1}$$

[because from the definition of a canonical precedence scheme $y = Y_{i,q}\cdots Y_{i,q+r} = x_j A_j \cdots A_p x_{p+1}$ with $A_j = Y_{j,t}$ and hence $x_j = Y_{i,q}\cdots Y_{i,t-1}$]. Hence $X_{k+1}\cdots X_{n_A} + n_2 Y_{i-1,1}\cdots Y_{i,q-1} = \Lambda$ which contradicts (*). Therefore T must be a SOP for G. ∎

It is also possible to obtain a converse and we shall do this shortly. Towards this end, we wish to characterize which families of triples of binary relations on T yield a canonical precedence scheme. We begin by finding minimal relations which must be contained in any relations which are part of a precedence detection scheme.

Definition. Let $G = (V, \Sigma, P, \perp S \perp)$ be a context-free grammar, let T be a token set for G, and let $H = \{A \in V \mid A \overset{*}{\Longrightarrow} \Lambda\}$. We define binary relations on V as follows:

$\lambda = \{(A,B) \mid A \to xBy$ is in P for some $y \in V^*$, $x \in (V - T)^*\}$

$\delta = \{(A,B) \mid A \to xBy$ is in P for some $y \in V^*$, $x \in H^*\}$

$\rho = \{(A,B) \mid B \to xAy$ is in P for some $x \in V^*$, $y \in (V - T)^*\}$

$\alpha = \{(A,B) \mid C \to xAyBz$ is in P for some $x, z \in V^*$, $y \in (V - T)^*\} \cup \{(\perp,S), (S,\perp)\} \cup \{(\perp,\perp) \mid S \notin T\}$

$\gamma = \{(A,B) \mid C \to xAyBz$ is in P, $x, z \in V^*$, $y \in H^*\}$ $\cup \{(\perp,S), (S,\perp)\} \cup \{(\perp,\perp) \mid S \in H\}$.

In order to obtain a converse to the previous result, an intermediate lemma is required.

Lemma 3.1. Let $G = (V, \Sigma, P, \perp S \perp)$ be a context-free grammar, let T be a token set for G, and let $H = \{A \in V - \Sigma \mid A \overset{*}{\Longrightarrow} \Lambda\}$. For any $n \geq 0$; $A_0,\ldots,A_n \in V$; if $\perp S \perp \overset{*}{\Longrightarrow} A_0 \ldots A_n$ and if i is the least integer so that $A_i \ldots A_n \in \Sigma^*$ then for any $0 \leq j < k \leq n$, we have

(a) if $k < i$ and $A_{j+1} \ldots A_{k-1} \in (V - T)^*$ then $(A_j, A_k) \in \alpha\lambda^*$

(b) if $k < i$ and $A_{j+1} \ldots A_{k-1} \in H^*$ then $(A_j, A_k) \in \gamma\delta^*$

(c) if $i \leq k$, $j > 0$, and $A_{j+1} \ldots A_{k-1} \in H^*$ then either $(A_j, A_k) \in \rho^*\gamma\delta^*$ or $A_0 \ldots A_n = \perp S \perp$.

Proof: Let m be the number of steps in $\perp S \perp \overset{*}{\Longrightarrow} A_0 \ldots A_n$ and the argument will be an induction on m.

Basis: $m = 0$. Then we have $\perp S \perp = A_0 A_1 A_2$ so that $n = i = 2$. If $k = 1$ then $j = 0$ and we are in cases (a) and (b) and we have $(\perp, S) \in (\alpha\lambda^*) \cap (\gamma\delta^*)$. If $k = 2$, then (a) and (b) are satisfied vacuously and (c) follows since $A_0 A_1 A_2 = \perp S \perp$.

Induction Step. Suppose that

$$\perp S \perp \underset{R}{\overset{m}{\Longrightarrow}} A_0 \ldots A_n$$

$$\underset{R}{\Longrightarrow} A_0 \ldots A_{i-2} B_0 \ldots B_p A_i \ldots A_n = C_0 \ldots C_q$$

Let i' be the least integer such that $C_{i'} \ldots C_q \in \Sigma^*$ and assume the induction hypothesis holds for $A_0 \ldots A_n$. Note $i' \leq i$. We establish (a) and (b) simultaneously. If $k < i-1$ or $i+p \leq j$ then (a) and (b) follow from the induction hypothesis. Suppose that $i-2 < j < k < i+p$ (i.e., both operators in $B_0 \ldots B_p$), then since $(A_{i-1} \rightarrow B_0 \ldots B_p) \in P$

$C_{j+1}\cdots C_{k-1} \in (V - T)^*$ implies $(C_j, C_k) \in \alpha \subseteq \alpha\lambda^*$

while $C_{j+1}\cdots C_{k-1} \in H^*$ implies $(C_j, C_k) \in \gamma \subseteq \gamma\delta^*$.

Furthermore, since $i' \leq i \leq i+p$, we know that if $i+p < k$ then $i' \leq k$ so that $k \not< i'$ and so (a) and (b) are vacuous in this case. Thus the only remaining case is where $j \leq i-2 < k < i+p$. If $C_{j+1}\cdots C_{k-1} \in (V - T)^*$, we see that (a) of the induction hypothesis implies that $(C_j, A_{i-1}) \in \alpha\lambda^*$. By the definition of λ, we have $(A_{i-1}, C_k) \in \lambda$. But this implies $(C_j, C_k) \in \alpha\lambda^*$. If $C_{j+1}\cdots C_{k-1} \in H^*$, then (b) of the induction hypothesis implies that $(C_j, A_{i-1}) \in \gamma\delta^*$. Moreover, the definition of δ yields $(A_{i-1}, C_k) \in \delta$ and hence $(C_j, C_k) \in \gamma\delta^*$. Thus the induction has been extended.

Next, we consider (c). The induction hypothesis holds if $j \geq i+p$. If $i-2 < j < i+p \leq k$ then $C_{j+1}\cdots C_{k-1} \in H^*$ implies $C_j \rho A_{i-1}$ because $H \subseteq (V - T)$. By the induction hypothesis, either $(A_{i-1}, C_k) \in \rho^*\gamma\delta^*$ which implies $(C_j, C_k) \in \rho^*\gamma\delta^*$, or $\perp S \perp = A_o A_1 A_2$ in which case the result also follows easily.

If $i-2 < j < k < i+p$ then (c) follows from the definition of γ. If $j \leq i-2 < k < i+p$ then $k \geq i'$ and $C_{j+1}\cdots C_{k-1} \in H^*$ implies $(C_j, A_{i-1}) \in \gamma\delta^*$ by part (b). Also $(A_{i-1}, C_k) \in \delta$ so that $(C_j, C_k) \in \gamma\delta^* \subseteq \rho^*\gamma\delta^*$.

The last case is where $0 < j \leq i-2 < i+p < k$. In this case since $C_{j+1}\cdots C_{k-1} \in H^*$, we have that $k = i+p+1 = i'$. For $C_{k-1} = A_{k-p-1} \in H$ and $H \cap \Sigma = \emptyset$ so $i' \geq k$ and we know $K \geq i+p+1 \geq i'$.

Next we claim that $A_{i-1} \in H$ since $A_{i-1} \rightarrow B_o \cdots B_p$ with each $B_r \in H$. This implies $B_r \overset{*}{\Longrightarrow} \Lambda$ for each $0 \leq r \leq p$ so that $A_{i-1} \overset{*}{\Longrightarrow} \Lambda$. Thus $A_{j+1}\cdots A_{i-1} \in H^*$ and by the induction hypothesis $(A_j, A_i) = (C_j, C_k) \in \rho^*\gamma\delta^*$

unless $A_o A_1 A_2 = \perp S \perp$. In this case, $C_o \ldots C_q = \perp B_o \ldots B_p \perp$ so $0 < j \leq i-2 = 0$ is vacuous. Thus the induction has been extended for case (c). ∎

We now constructively define the canonical precedence relations.

Definition. Let $G = (V, \Sigma, P, \perp S \perp)$ be a context-free grammar with token set T. The T-canonical precedence relations for G are defined as

$$<\cdot_T = \alpha\lambda^+ \cap (T \times T)$$
$$\doteq_T = \alpha \cap (T \times T)$$
$$\cdot>_T = (\rho^+ \gamma \delta^*) \cap (T \times \Sigma)$$

The following result is the converse to the previous result and indicates another application of SOP's.

Theorem 3.4. Let $G = (V, \Sigma, P, \perp S \perp)$ be a context-free grammar and T a token set. Let $<\cdot = <\cdot_T$, $\doteq = \doteq_T$, and $\cdot> = \cdot>_T$. Then $(G, T, <\cdot, \doteq, \cdot>)$ is a canonical precedence scheme if and only if

(a) $<\cdot$, $\doteq$, and $\cdot>$ are pairwise disjoint

and (b) T is a SOP.

Proof: Suppose $(G, T, <\cdot, \doteq, \cdot>)$ is a canonical precedence scheme. By Theorem 3.3, we see that T is a SOP for G and the relations must be pairwise disjoint by definition.

Conversely, suppose that T is a SOP for $G = (V, \Sigma, P, \perp S \perp)$ and let $<\cdot_T$, $\doteq_T$ and $\cdot>_T$ be pairwise disjoint. Assume that

$$\perp S \perp \underset{R}{\overset{*}{\Longrightarrow}} xAz \underset{R}{\Longrightarrow} xyz \qquad (1)$$

and there exist $n \geq 0$, $x \in V^*$, $z \in \Sigma^*$, $A_o, \ldots, A_n \in T$; $x_1, \ldots, x_n \in (V - T)^*$ so that $xyz = A_o x_1 A_1 \ldots x_n A_n$. We must show that either $y \in (V - T)^*$ or

there exist i and j with $0 < i \leq j < n$ so that $x = A_o x_1 A_1 \cdots A_{1-1}$, $y = x_i A_i \cdots A_j x_{j+1}$, $z = A_{j+1} x_{j+1} A_{j+2} \cdots x_n A_n$, $A_{i-1} <\cdot_T A_i$, for each k with $i \leq k < j$ we have $A_k \doteq A_{k+1}$, and $A_j \cdot>_T A_{j+1}$.

Involve 3.2.A.

Suppose that $y \notin (V - T)^*$. Then there exist $0 < i \leq j < n$ so that

$$y = w_i A_i x_{i+1} \cdots A_j w_{j+1}$$

where w_i is a suffix of x_i while w_{j+1} is a prefix of x_{j+1}. We first claim that $w_{j+1} = x_{j+1}$. To see that, note that $x_{j+1} = w_{j+1} w'$ for some w'. Note $w' \in (V - T)^*$ since x_{j+1} is, also $w' \in \Sigma^*$ since w' is a prefix of z in the rightmost derivation (1). Thus $w' \in (V - T)^* \cap \Sigma^* = \{\Lambda\}$ since $\Sigma \subseteq T$. Thus $w' = \Lambda$ and $w_{j+1} = x_{j+1}$.

Since $A \to y$ is the last production used in (1), it is immediately clear that $A_k \doteq_T A_{k+1}$ for each k such that $i \leq k < j$. By Lemma 3.1, part (a), we have that $(A_{i-1}, A) \in \alpha\lambda^*$. From the definition of λ, we know $(A, A_i) \in \lambda$ so that $(A_{i-1}, A_i) \in \alpha\lambda^*$. Thus $A_{i-1} <\cdot_T A_i$.

In a similar fashion, $(A_j, A) \in \rho$. Since $w_{j+1} = x_{j+1}$, we see that $z = A_{j+1} z'$ for some $z' \in \Sigma^*$. Thus, Lemma 3.1(c) implies that $(A, A_{j+1}) \in \rho^* \gamma \delta^*$. Therefore $(A_j, A_{j+1}) \in \rho^+ \gamma \delta^*$. So $A_j \cdot>_T A_{j+1}$ and thus $A_j \cdot> A_{j+1}$ since $\cdot>_T \subseteq \cdot>$.

To complete the proof, we must prove that $w_i = x_i$. Suppose that $x_i = v w_i$. Note that if $lg(x_i) \geq 2$ then part (a) of Theorem 3.2 is contradicted. Thus $lg(x_i) < 2$. If $x_i = \Lambda$ then $x_i = w_i = v = \Lambda$ and we are done. Suppose that $lg(x_i) = 1$. Then if $v = \Lambda$, we have $w_i = x_i$ and we are through. Suppose $v = x_i$ and $w_i = \Lambda$, then $x' = A_o x_1 \cdots A_{i-1} x_i$ and xAz is a sentential form. Thus $\perp S \perp \overset{*}{\Longrightarrow} x'vAz$ with $v \in V - T$ and $\Lambda \notin \Sigma$. This contradicts part (a) of Theorem 3.2 and shows that $w_i \neq \Lambda$ which implies $x_i = w_i$. The argument is complete. ∎

The previous result justified calling $<\cdot_T$, $\doteq_T$, and $\cdot>_T$ precedence relations. Next, we justify calling these relations canonical

Theorem 3.5. Let $G = (V, \Sigma, P, \perp S \perp)$ be a P-reduced context free grammar with token set T and let $<\cdot$, $\doteq$, and $\cdot>$ be any three binary relations on T. $(G, T, <\cdot, \doteq, \cdot>)$ is a canonical precedence scheme if and only if

(a) $<\cdot$, $\doteq$, and $\cdot>$ are pairwise disjoint,

and (b) $<\cdot_T \subseteq <\cdot$, $\doteq_T \subseteq \doteq$, and $\cdot>_T \subseteq \cdot>$,

and (c) T is a SOP.

If $(G, T, <\cdot, \doteq, \cdot>)$ is a canonical precedence scheme then so is $(G, T, <\cdot_T, \doteq_T, \cdot>_T)$.

Proof: Let $(G, T, <\cdot, \doteq, \cdot>)$ be a canonical precedence scheme. T must be a SOP from Theorem 3.3. By definition, the relations are pairwise disjoint so it suffices to prove (b).

Let $A, B \in T$ and suppose $A \doteq_T B$. This implies $(A,B) \in \alpha$ so that there are $C \in V - \Sigma$; $x, z \in V^*$, $y \in (V - T)^*$ so that $C \to xAyBz$ is in P. Since G is reduced, there exist $u, v \in V^*$ so that

$$\perp S \perp \overset{*}{\underset{R}{\Longrightarrow}} vCw \underset{R}{\Longrightarrow} vxAyBzw$$

By the definition of a canonical precedence scheme, $A \doteq B$ and we have shown $\doteq_T \subseteq \doteq$.

Suppose that $A <\cdot_T B$. Then there exist $C, D \in V - \Sigma$ so that $(A,C) \in \alpha$, $(C,D) \in \lambda^*$, and $(D,B) \in \lambda$. Note that $(A,C) \in \alpha$ implies that $E \to xAyCz$ is in P where[10] $y \in (V - T)^*$, while $(D,B) \in \lambda$ implies $D \to uBv$ is in P where $u \in (V - T)^*$. We also have that $(C,D) \in \lambda^*$

---

[10] We omit consideration of the other possibilities $(C,A) = (\perp,S)$, etc., as those cases are trivial.

implies $C \overset{*}{\Longrightarrow} sDt$ with $s \in (V - T)^*$. Since G is reduced, we have that there exist z', $z'' \in V^*$

$$\perp S \perp \overset{*}{\underset{R}{\Longrightarrow}} z'Ez'' \underset{R}{\Longrightarrow} z'xAyCzz'' \overset{*}{\underset{R}{\Longrightarrow}} z'xAysDtzz''$$

$$\underset{R}{\Longrightarrow} z'xAysuBvtzz''$$

Since y, s, and u are in $(V - T)^*$, we have that $ysu \in (V - T)^*$. Thus $A <\cdot B$ and we have shown that $<\cdot_T \subseteq <\cdot$.

Suppose that $A \cdot>_T B$ with $A \in T$ and $B \in \Sigma$. Then $(A,B) \in \rho^+\gamma\delta^*$. There exist C, $D \in V - \Sigma$; $E \in V$ so that $(A,C) \in \rho$, $(C,D) \in \rho^*$, $(D,E) \in \gamma$, and $(E,B) \in \delta^*$. First note that $(A,C) \in \rho$ implies $C \rightarrow xAy$ where $y \in (V - T)^*$ while $(C,D) \in \rho^*$ implies $D \overset{*}{\underset{R}{\Longrightarrow}} uCv$ where $v \in (V - T)^*$. Since $(D,E) \in \gamma$, we have $F \rightarrow x_1Dy_1Ez_1$ with $y_1 \in H^*$ (which implies $y_1 \overset{*}{\underset{R}{\Longrightarrow}} \Lambda$). Lastly, $(E,B) \in \delta^*$ implies $E \overset{*}{\underset{R}{\Longrightarrow}} u_1Bu_2$ with $u_1 \in H^*$ (so that $u_1 \overset{*}{\underset{R}{\Longrightarrow}} \Lambda$).

Combining these results with the fact that G is reduced, we see that there exist z', z'', z_1' and u_2' so that

$$\perp S \perp \overset{*}{\underset{R}{\Longrightarrow}} z'Fz'' \underset{R}{\Longrightarrow} z'x_1Dy_1Ez_1z''$$

$$\overset{*}{\underset{R}{\Longrightarrow}} z'x_1Dy_1u_1Bu_2z_1'z''$$

$$\overset{*}{\underset{R}{\Longrightarrow}} z'x_1DBu_2'z_1'z'' \overset{*}{\underset{R}{\Longrightarrow}} z'x_1uCvBu_2'z_1'z''$$

$$\underset{R}{\Longrightarrow} z'x_1uxAyvBu_2'z_1'z''$$

Thus it follows from the definition of a canonical precedence scheme that $A \cdot> B$.

Conversely, suppose that (a), (b), and (c) hold for a system $(G, T, <\cdot, \doteq, \cdot>)$. By (a), these relations are pairwise disjoint. Furthermore, Theorem 3.4 and (c) and (a) imply that $(G, T, <\cdot_T, \doteq_T, \cdot>_T)$ is a canonical precedence scheme. To show that $(G, T, <\cdot, \doteq, \cdot>)$ is a canonical precedence scheme we argue as follows. Suppose $\perp S \perp \overset{*}{\underset{R}{\Longrightarrow}} xAz \Longrightarrow xyz = A_0 x_1 A_1 \ldots x_n A_n$ as above. Let i,j be as above. Since $(G, T, <\cdot_T, \doteq_T, \cdot>_T)$ is a canonical precedence system, we know that

$$A_{i-1} <\cdot_T A_i$$

$$A_k \doteq_T A_{k+1} \qquad \text{for each } k \text{ so that } i \leq k < j$$

and $$A_j \cdot>_T A_{j+1}.$$

By (b) we have $A_{i-1} <\cdot A_i$, $A_k \doteq A_{k+1}$ for each $i \leq k < j$, and $A_j \cdot> A_{j+1}$. Therefore $(G, T, <\cdot, \doteq, \cdot>)$ is also a canonical precedence scheme.

Note that if $(G, T, <\cdot, \doteq, \cdot>)$ is a canonical precedence scheme, then (a), (b) and (c) hold. So $<\cdot_T$, $\doteq_T$, and $\cdot>_T$ are also pairwise disjoint. Hence $(G, T, <\cdot_T, \doteq_T, \cdot>_T)$ is also a canonical precedence scheme by Theorem 3.4. Finally, the proof is complete. ∎

Corollary. G has a canonical precedence scheme based on token set T if and only if $(G, T, <\cdot_T, \doteq_T, \cdot>_T)$ is a canonical precedence scheme.

It may be of interest to show that condition (c) cannot be removed in Theorem 3.5. Consider the grammar with productions

$$S \rightarrow AB$$

$$A \rightarrow a$$

$$B \rightarrow b$$

Pick token set $T=\{\perp, a, b, B\}$. T is a COP and a FOP but not a SOP since $A \in V - T$ and $B \in \Sigma$. The T canonical precedence matrix is

	$\perp$	a	b	B
$\perp$	$\doteq$	$<\cdot$	$<\cdot$	$<\cdot$
a			$\cdot>$	
b	$\cdot>$			
B	$\cdot>$			

We will show that this is not a canonical precedence scheme. Consider

$$\perp S \perp \underset{R}{\overset{*}{\Longrightarrow}} \perp AB \perp \underset{R}{\Longrightarrow} \perp Ab \perp$$

We find that $\perp <\cdot b$ and $b \cdot> \perp$. Therefore y = Ab which is supposed to be the right-hand side of a production but is not.

It is now clear that one can effectively decide if a grammar has a precedence detection scheme based on T. It is known that each context free language has a precedence detection scheme [4].

We now relate our study to the special cases given in the literature. First, it should be noted that the theory developed by Floyd [5] can be easily extended to include Λ-rules. Floyd's precedence relations are defined (using our notation), with token set Σ, as follows:

$$<\cdot_F = \alpha\lambda^+ \cap (\Sigma \times \Sigma)$$

$$\doteq_F = \alpha \cap (\Sigma \times \Sigma)$$

$$\cdot>_F = \rho^+\alpha \cap (\Sigma \times \Sigma)$$

Now, we summarize the manner in which Floyd precedence fits into our scheme.

Theorem 3.6. $(G, \Sigma, <\cdot_F, \doteq_F, \cdot>_F)$ is a canonical precedence scheme if and only if $(G, \Sigma, <\cdot_\Sigma, \doteq_\Sigma, \cdot>_\Sigma)$ is a canonical precedence scheme. In this case

$$<\cdot_F = <\cdot_\Sigma, \quad \doteq_F = \doteq_\Sigma, \text{ and } \quad \cdot>_F = \cdot>_\Sigma .$$

Proof: Suppose $(G, \Sigma, <\cdot_F, \doteq_F, \cdot>_F)$ is a canonical precedence scheme. By Theorem 3.5, $(G, \Sigma, <\cdot_\Sigma, \doteq_\Sigma, \cdot>_\Sigma)$ is also and we have that $<\cdot_\Sigma \subseteq <\cdot_F$, $\doteq_\Sigma \subseteq \doteq_F$, and $\cdot>_\Sigma = \cdot>_F$. By the definitions, it is clear that $<\cdot_\Sigma = <\cdot_F$ and $\doteq_\Sigma = \doteq_F$ so that we need only show that $\cdot>_F \subseteq \cdot>_\Sigma$ (for this, together with $\cdot>_\Sigma \subseteq \cdot>_F$, implies $\cdot>_\Sigma = \cdot>_F$). Let $A, B \in \Sigma$ and suppose that $A \cdot>_F B$ (i.e., $(A,B) \in \rho^+\alpha$). Thus there exists $C \in V$ so that $(A,C) \in \rho^+$ and $(C,B) \in \alpha$. By the definition of ρ^+, we have $C \in V - \Sigma$. Since $(C,B) \in \alpha$, there exists a production $E \to xCyBz$ with $y \in (V - \Sigma)^*$. Σ is a SOP by Theorem 3.3 so G must be an operator grammar by Theorem 3.1. Thus $y = \Lambda$ which implies $(C,B) \in \gamma$. Therefore $(A,B) \in \rho^+\gamma \subseteq \rho^+\gamma\delta^*$ and so $A \cdot>_\Sigma B$.

Conversely, if $(G, \Sigma, <\cdot_\Sigma, \doteq_\Sigma, \cdot>_\Sigma)$ is a canonical precedence scheme, Σ must be a SOP and the previous argument holds. Therefore $<\cdot_\Sigma = <\cdot_F$, $\doteq_\Sigma = \doteq_F$, and $\cdot>_\Sigma = \cdot>_F$ so that $(G, \Sigma, <\cdot_F, \doteq_F, \cdot>_F)$ is a canonical precedence scheme.

The Floyd precedence relations are not exactly the same as the Σ-canonical precedence relations since the former are only defined if Σ is a SOP for G, i.e., G is an operator grammar. If Σ is a SOP then the two are identical. We chose the rather weak definition of a token set and from it derived the fact that G must be an operator grammar if Σ yields a canonical precedence scheme. In this sense Floyd's restriction that G be an operator grammar was superfluous.

Corollary. If $G = (V, \Sigma, P, \perp S \perp)$ has a Σ-canonical precedence scheme, then G is an operator grammar.

We also are able to show that

Corollary (Fischer, Manacher). If G is a Floyd precedence detectable grammar, then G is equivalent to an invertible Floyd precedence detectable grammar.

Proof: One need only observe that the standard construction [7], [11], which deletes the Λ rules from G leaves the relations ρ, α, λ invariant and the standard construction which deletes chain rules from G, [7], [11], leaves $<\cdot_F$, $\doteq_F$, and $>_F$ invariant. The resulting grammar is Λ-free and chain-free so Proposition 2.1 can be invoked. Inspection of the construction of 2.1 shows that it leaves ρ, α, λ invariant. So the resulting grammar is an invertible Floyd precedence detectable grammar. ∎

This is in distinction to simple precedence. Fischer's results show that Floyd precedence is weaker than simple precedence. Proposition 2.3 shows that this is not because G must be an operator grammar since any grammar may be turned into an operator grammar. Rather the weakness of the Floyd relations lies in ignoring the non-terminal characters.

Next, we consider the simple precedence methods developed by Pair [16] and by Wirth and Weber [18] and embed these into our theory. Take the token set to be[11] V and then define the simple precedence relations after Wirth and Weber as:

---

[11] Recall that this implies that G is Λ-free.

$$<\cdot_S = \alpha\lambda^+$$

$$\doteq_S = \alpha$$

$$\cdot>_S = \rho^+\alpha\lambda$$

It should be noted that the simple precedence relations do not enjoy the same (left-to-right) symmetry properties as the Floyd relations.

The simple precedence relations are distinct from the V-canonical relations. The following grammar exhibits this difference.

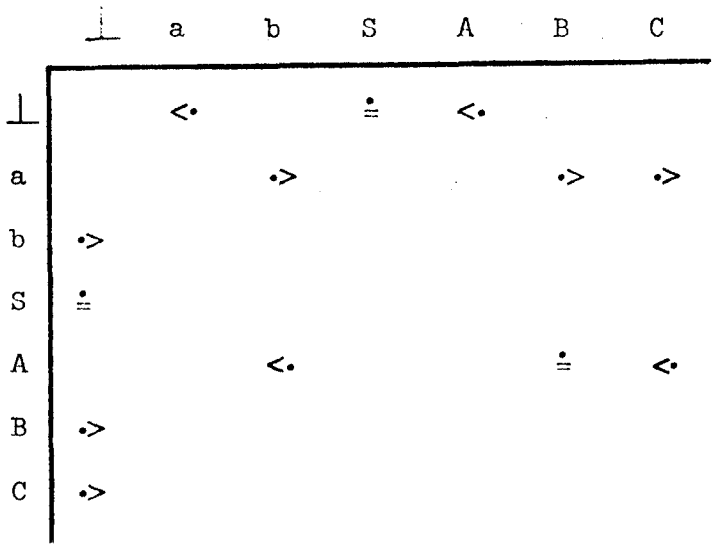
$S \rightarrow AB$

$A \rightarrow a$

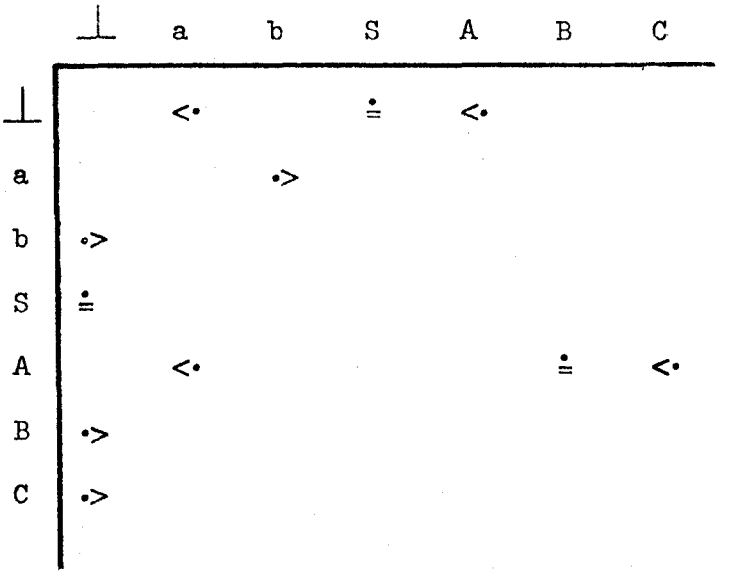
$B \rightarrow C$

$C \rightarrow c$

The precedence matrices are:



The Simple Relations for G



The V-Canonical Relations for G

Thus the simple precedence relations are not minimal in the sense of Theorem 3.5. We shall now show that these differences are not critical.

Theorem 3.7. If $G = (V, \Sigma, P, \perp S \perp)$ is reduced then $(G, V, <\cdot_V, \doteq_V, \cdot>_V)$ is a canonical precedence scheme if and only if $(G, V, <\cdot_S, \doteq_S, \cdot>_S)$ is a canonical precedence scheme.

Proof: By Theorem 3.5, if $(G, V, <\cdot_S, \doteq_S, \cdot>_S)$ is a canonical precedence scheme, then so is $(G, V, <\cdot_V, \doteq_V, \cdot>_V)$.

Suppose that $(G, V, <\cdot_V, \doteq_V, \cdot>_V)$ is a canonical precedence scheme. By Theorem 3.3, V is a SOP. It suffices to show that $<\cdot_S$, $\doteq_S$, and $\cdot>_S$ are pairwise disjoint because $<\cdot_V = <\cdot_S$, $\doteq_V = \doteq_S$, and $\cdot>_V \subseteq \cdot>_S$ implies (by Theorem 3.5) that $(G, V, <\cdot_S, \doteq_S, \cdot>_S)$ must be a canonical precedence scheme.

Since $<\cdot_V = <\cdot_S$ and $\doteq_V = \doteq_S$, it follows that $<\cdot_S \cap \doteq_S = \emptyset$. Thus it suffices to show that $(<\cdot_S \cup \doteq_S) \cap \cdot>_S = \emptyset$. Suppose there exists $(A,B) \in \rho^+\alpha\lambda^*$ and $(A,B) \in \alpha\lambda^*$ since $\alpha\lambda^* = \alpha\lambda^+ \cup \alpha = <\cdot_S \cup \doteq_S$. Since G is Λ-free (because V is a token set) and reduced, there exist $b \in \Sigma$, $z \in \Sigma^*$ so that $B \overset{*}{\Longrightarrow} bz$. Clearly $(B,b) \in \lambda^*$ and hence $(A,b) \in \rho^+\alpha\lambda^*$ and $(A,b) \in \alpha\lambda^*$. Since the token set is V, note that $\lambda = \delta$ and $\alpha = \gamma$. Thus $(A,b) \in \alpha\lambda^*$ and $(A,b) \in \rho^+\alpha\lambda^* = \rho^+\gamma\delta^*$. Since $b \in \Sigma$, we have shown that $A \cdot>_V b$ and either

(i) $A <\cdot_V b$

or (ii) $A \doteq b$

Therefore

$$(<\cdot \cup \doteq_V) \cap \cdot>_V \neq \emptyset$$

which implies that $<\cdot_V$, $\doteq_V$, and $\cdot>_V$ are not pairwise disjoint. But, $(G, V, <\cdot_V, \doteq_V, \cdot>)$ is a canonical precedence scheme and therefore (*) contradicts part (a) of Theorem 3.4. So we conclude that the simple precedence relations are disjoint if the V-canonical relations are. ∎

Pair defined the V-canonical relations in his paper [16].

The assumption that G is reduced in Theorem 3.7 is essential as the following example shows. G is the following grammar

$$S \to AB \mid A$$

$$A \to a \mid aA$$

$(G, V, <\cdot_V, \doteq_V, \cdot>_V)$ is a canonical precedence scheme whose precedence matrix is given below

	⊥	a	S	A	B
⊥		<·	≐	<·	
a		<·			
S	≐				
A	·>				
B	·>				

Next we compute the simple precedence relations for G.

	⊥	a	S	A	B
⊥		<·	≐	<·	
a		<·		≐	·>
S	≐				
A	·>				
B	·>				

Since $A \doteq_S B$ and $A \cdot\!>_S B$, we know that $(G, V, <\!\cdot_S, \doteq_S, \cdot\!>_S)$ is not a canonical precedence scheme.

Section 4

Reduction

We now present a general definition of precedence analysis by relaxing the restriction that the parser must detect the leftmost phrase. We will require only that it detect some phrase which may be reduced. We also discuss how the precedence relations may be used to construct a parser for the grammar. We do this in the spirit of [2]. Colmerauer introduces an automaton which operates on two push-down stacks using only the precedence relations between the topmost operators of the stacks to decide the next step.

Our presentation is made more complex by our desire to define the relations on any COP and our desire to have the parser explicitly generate the sparse parse of the input string. In order to do that we need the following preliminary results. Fix grammar $G = (V, \Sigma, P, \perp S \perp)$ and let T be a COP for G. Define

$$P(T) = \{(A,x,y) \mid (A \to x) \in H \text{ and } x \overset{*}{\Longrightarrow} y \text{ by productions in } P - H\}$$

where $H = \{(A \to x) \in P \mid x \notin (V - T)^*\}$.

Consider the grammar $G_T = (V, \Sigma, P_T, \perp S \perp)$ where

$P_T = \{(A,y) \mid (A,x,y) \in P(T) \text{ for some } x\} \cup \{(S,A) \mid S \overset{*}{\Longrightarrow} A \in (V - T)\}$

This is the standard construction for eliminating chains of operands and for eliminating Λ rules from the grammar.

First observe that

Lemma 4.1. Let G, T and H be as above. Then for any T-infix string $x = x_0A_1x_1 \ldots x_nA_n$ where $x_i \in (V - T)^* \cup \{\Lambda\}$ and $A_i \in T$, if $x \overset{*}{\Longrightarrow} y$ by

a generation $((B_i \to z_i, n_i))_{i=1}^{m}$ where each $B_i \to z_i \in$ (P-H) then

(a) $y = y_0 A_1 y_1 \ldots A_n y_n$ where each $y_i \in (V-T) \cup \{\Lambda\}$

and (b) $x_i \overset{*}{\Longrightarrow} y_i$ by canonical derivation $(B_{i_j} \to z_{i_j})_{j=1}^{m_j}$ for each i, $0 \leq i \leq n$.

Proof: Induct an m. If m = 0 the result is trivial. If true for m, consider the case m+1. Let $x \overset{*}{\Longrightarrow} y_0 A_1 y_1 \ldots A_n y_n$ under $((B_i \to z_i), n_i)_{i=1}^{m}$. By the induction hypothesis $y_i \in (\overline{T})^*$ exist. Now consider $(B_{m+1} \to z_{m+1}, n_{m+1})$. Since $(B_{m+1} \to z_{m+1}) \notin H$, $z_{m+1} \in (\overline{T})^*$. Since T is a COP $z_{m+1} = \Lambda$ or $z_{m+1} \in \overline{T}$. Now $B_{m+1} \notin T$ since $z_{m+1} \in (\overline{T})^*$ and T is a token set. So $B_{m+1} = y_j$ for some j. Thus $(B_i \to z_i)_{i=1}^{n}$ is the derivation $x \overset{*}{\Longrightarrow} y_0 A_1 \ldots A_j\, y_j A_{j+1} \ldots y_n \Longrightarrow y_0 A_1 \ldots A_j z_{m+1} A_{j+1} \ldots y_n$.

Clearly z_{m+1} is a T-infix string since T is a COP. Also A_j, $A_{j+1} \in T$ so $y_0 A_1 \ldots A_j z_{m+1} A_{j+1} \ldots y_n$ is an infix string.

This new string satisfies (a) and (b). So the induction is extended. ∎

Observe that this implies that a T-infix string generates at most a finite number of distinct T-infix strings using rules chosen from P-H. In particular note that for any $(A \to x) \in P$, $\{y \mid x \overset{*}{\Longrightarrow} y$ by rules chosen from P-H$\}$ is finite. So P(T) is finite and we conclude that G_T is a grammar.

Our interest in G_T is that a precedence analyzer working with operator set T will produce a parse of x in G_T. However G_T covers $\langle G,H \rangle$ so it follows that one can view the parser as producing an H-sparse parse of x in G.

Define a two-stack automaton[11] after Griffiths and Petrick: $\mathcal{A} = \langle V, \vdash, F \rangle$ will be said to be a <u>two-stack automaton</u> if

---

[11] This automaton bears a strong similarity to the list automaton of Ginsburg and Harrison [10].

(1) V is a finite alphabet,

and (2) $\vdash \subset (V^*)^4$ is finite. We write

$$(x,y) \vdash (x',y') \quad \text{if} \quad (x,y,x',y') \in \vdash ,$$

and (3) $F \subseteq V^2$ is a finite set of final states.

The relation $\vdash$ is extended in the natural way:

for each $v, w \in V^*$ if $(x,y) \vdash (x',y')$ then

$$(vx,yw) \vdash (vx',y'w).$$

A sequence $((x_i,y_i))_{i=1}^{n}$ such that $(x_i,y_i) \vdash (x_{i+1},y_{i+1})$ for $i = 1,\ldots,n-1$ is called a computation of $\mathcal{A}$ on (x_1,y_1). It is successful if $(x_n,y_n) \in F$. Let G, T, P(T), G_T and H be as above and suppose $\lessdot$, $\doteq$, $\gtrdot$ are three disjoint binary relations on T. Then $(G, T, \lessdot, \doteq, \gtrdot)$ is said to induce the two-stack automaton

$$\mathcal{A} = \langle V, \vdash, F \rangle \quad \text{where}$$

$$F = \{(\perp x, \perp) \mid x = S \text{ or } (S,y,x) \in P(T) \text{ and } y \in (V-T)^*\}$$

and $\vdash$ is defined by cases:

(i) for each $C \in \overline{T}$, $(\Lambda,C) \vdash (C,\Lambda)$,

and (ii) for each $A, B \in T$, $x \in (\overline{T}) \cup \{\Lambda\}$

$$(Ax,B) \vdash (AxB,\Lambda) \quad \text{if} \quad A \lessdot B \text{ or } A \doteq B$$

and

(iii) for each $n > 0$, $A,A_0,\ldots,A_{n+1} \in T$, $x_0,\ldots,x_n \in (V-T)^*$

$$(A_0x_0A_1\ldots A_nx_n,A_{n+1}) \vdash (A_0,AA_{n+1})$$

$$\text{iff} \quad (A,x,x_0A_1x_1\ldots A_nx_n) \in P(T)$$

and $A_o <\cdot A_1$

and $A_i \doteq A_{i+1}$ for $1 \leq i < n$

and $A_n \cdot> A_{n+1}$

Note that (i), (ii), and (iii) are mutually exclusive since $<\cdot$, $\doteq$, and $\cdot>$ are pairwise disjoint. If $(x,y) \vdash (x',y')$ by virtue of (iii) we say that this is a reduction step.

For each $\perp x \perp \in \Sigma^*$ and each successful computation, $((x_i,y_i))_{i=1}^n$ of $\mathcal{M}$ started on $(\perp,x\perp)$, we consider the subsequence of reduction steps

$$(x_{i_1},y_{i_1}) \vdash (x_{i_1+1},y_{i_1+1}) \vdash^* (x_{i_2},y_{i_2}) \vdash (x_{i_2+1},y_{i_2+1}) \vdash^* \cdots$$

$$(x_{i_k},y_{i_k}) \vdash (x_{i_k+1},y_{i_k+1})$$

We say $\mathcal{M}$ outputs $((A_j \rightarrow z_j, lg(x_{i_j}))_{j=1}^k$ on $(\perp,x\perp)$ if

$(x_{i_j},y_{i_j}) \vdash (x_{i_j+1},y_{i_j+1})$ by virtue of $(A_j,z_j,w_j) \in P(T)$ for each j.

Note that $\mathcal{M}$ may output several sequences on $(\perp,x\perp)$.

We say that $(G, Q, <\cdot, \doteq, \cdot>)$ is a precedence scheme if for any $\perp x \perp \in L(G)$, $\mathcal{M}$ started on $(\perp,x\perp)$ outputs the H-sparse parses corresponding to all non-equivalent parses[13] of $\perp x \perp$ in G.

This definition produces a two-stack automaton. For $|F| \leq |P(T)|$ and we know $P(T)$ is finite so F is finite. Similarly $\vdash$ is finite since (i) and (ii) contribute at most $2 \cdot |V|^3$ elements to $\vdash$ and (iii) contributes at most $|V|^2 \cdot |P_T|$ elements to $\vdash$.

---

[13] Recall that we defined two parses (derivations) to be equivalent if they have the same canonical derivation (i.e., same parse tree).

First we show that the length of any particular successful computation of $\mathcal{A}$ started on a string of length n requires a number of steps linearly proportional to n.

Theorem 4.1. Let $\perp x \perp \in L(G)$. Then if $(G, T, <\cdot, \doteq, \cdot>)$ is a precedence scheme inducing automaton $\mathcal{A}$, then $\mathcal{A}$ outputs at least one H-sparse parse of x in less than $(4|T - \Sigma| + 5) \cdot lg(x) + 1$ steps.

Proof: Let $((x_i,y_i))_{i=1}^{n}$ be a successful computation of $\mathcal{A}$ on $\perp x \perp$. Suppose it outputs sparse parse $((A_i \rightarrow z_i, n_i))_{i=1}^{p}$. And suppose the i^{th} reduction uses $(A_i, z_i, w_i) \in P(T)$.

First we establish the bound $p \leq 2(|T - \Sigma| + 1)\, lg(x)$. Observe that since the computation is successful,

$$lg(x) \geq lg(x_1y_1) - lg(x_ny_n) = \sum_{i=1}^{n-1} (lg(x_iy_i) - lg(x_{i+1}y_{i+1}))$$

Note that $lg(x_iy_i) = lg(x_{i+1}y_{i+1})$ if $(x_i,y_i) \vdash (x_{i+1},y_{i+1})$ by clauses (i) or (ii) of the definition of $\mathcal{A}$ or by clause (iii) if by virtue of $(B,u_1,u_2) \in P(T)$ where $lg(u_2) = 1$ for some $B,u_1,u, \in V^*$, otherwise $lg(x_iy_i) - lg(x_{i+1}y_{i+1}) = lg(u_2) - 1$. So $lg(x_1y_1) - lg(x_ny_n) = \sum_{i=1}^{n-1} (lg(w_i) - 1)$ and $lg(x) \geq |\{lg(w_i) - 1 | lg(w_i) \geq 2,\ 1 \leq i \leq p\}|$.

Also we note that since the parse tree contains lg(x) occurrences of elements of Σ,

$$lg(x) \geq |\{w_i | w_i \in (V - \Sigma)^*,\ 1 \leq i \leq p\}|$$

So now we count $\{w_i | w_i \in (V - \Sigma),\ 1 \leq i \leq p\}$. Consider the complete parse tree associated with the sparse parse. We may assume without loss of generality that there is no chain $A \overset{+}{\Longrightarrow} A$ in the tree. So any

chain contains at most $|T - \Sigma|$ operators. Further every (A_i, z_i, w_i) where $w_i \in (V - \Sigma)$ corresponds to a chain $A_i \overset{*}{\Longrightarrow} w_i$. Each chain in the tree terminates with a non-chain production, i.e., (A_i, z_i, w_i) where $w_i \in \Sigma$ or $lg(w_i) \geq 2$. By the above there are at most $2 \cdot lg(x)$ such chains. So there at most $2 \cdot lg(x)$ chains each containing at most $|T - \Sigma|$ operators. So $|\{w_i | w_i \in (T - \Sigma)\}_{i=1}^{p}| \leq 2|T - \Sigma| \ lg(x)$. Thus

$$p = |\{w_i | 1 \leq i \leq p\}| \leq 2(|T - \Sigma| + 1) \ lg(x).$$

Now consider $lg(y_i) - lg(y_{i-1})$. On a reduction step $lg(y_i) - lg(y_{i-1}) = +1$ and on other steps $lg(y_i) - lg(y_{i-1}) = -1$. Also $lg(y_n) - lg(y_1) = -lg(x)$ since that computation is successful. But $lg(y_i)$ increases by one p times and decreases by one (n-1)-p times. The total change is $-lg(x)$ so

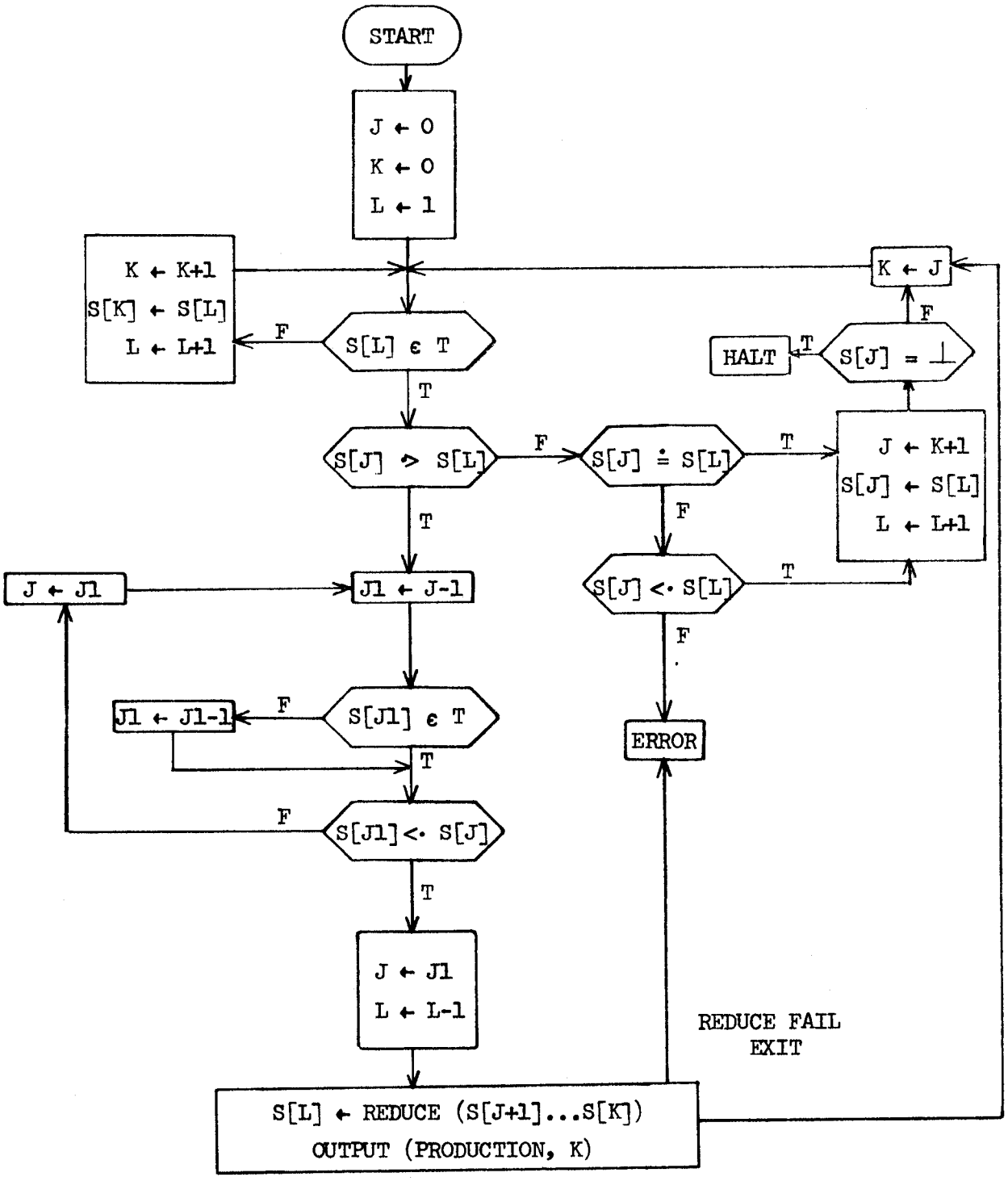
$$-lg(x) = p - (n-1-p) = 2p-n+1$$

So $n \leq 2p + lg(x) + 1$. But $p \leq 2(|T - \Sigma| + 1) \ lg(x)$

so $n \leq (4|T - \Sigma| + 5) \ lg(x) + 1$. ∎

If $(G, T, <\cdot, \doteq, \cdot>)$ is a precedence scheme the following flow chart models the automaton $\mathcal{A}$. Initally the string $x \in \{\perp\}(\Sigma - \{\perp\})^*\{\perp\}$, $x = x_1x_2\ldots x_n$ is stored in character array S[1:N]. S[I] is the topmost operator of the left stack S[J] the topmost symbol of the left stack and S[K] the topmost symbol of the right stack. The algorithm succeeds if it halts with $S[1] = S[J] = \perp$ and either $J = 2$ in which case we must have $S \overset{*}{\Longrightarrow} \Lambda$

or $J = 3$ in which case we must have $S \overset{*}{\Longrightarrow} S[2]$. Since the three relations are disjoint the only nondeterminancy is introduced by the REDUCE function.



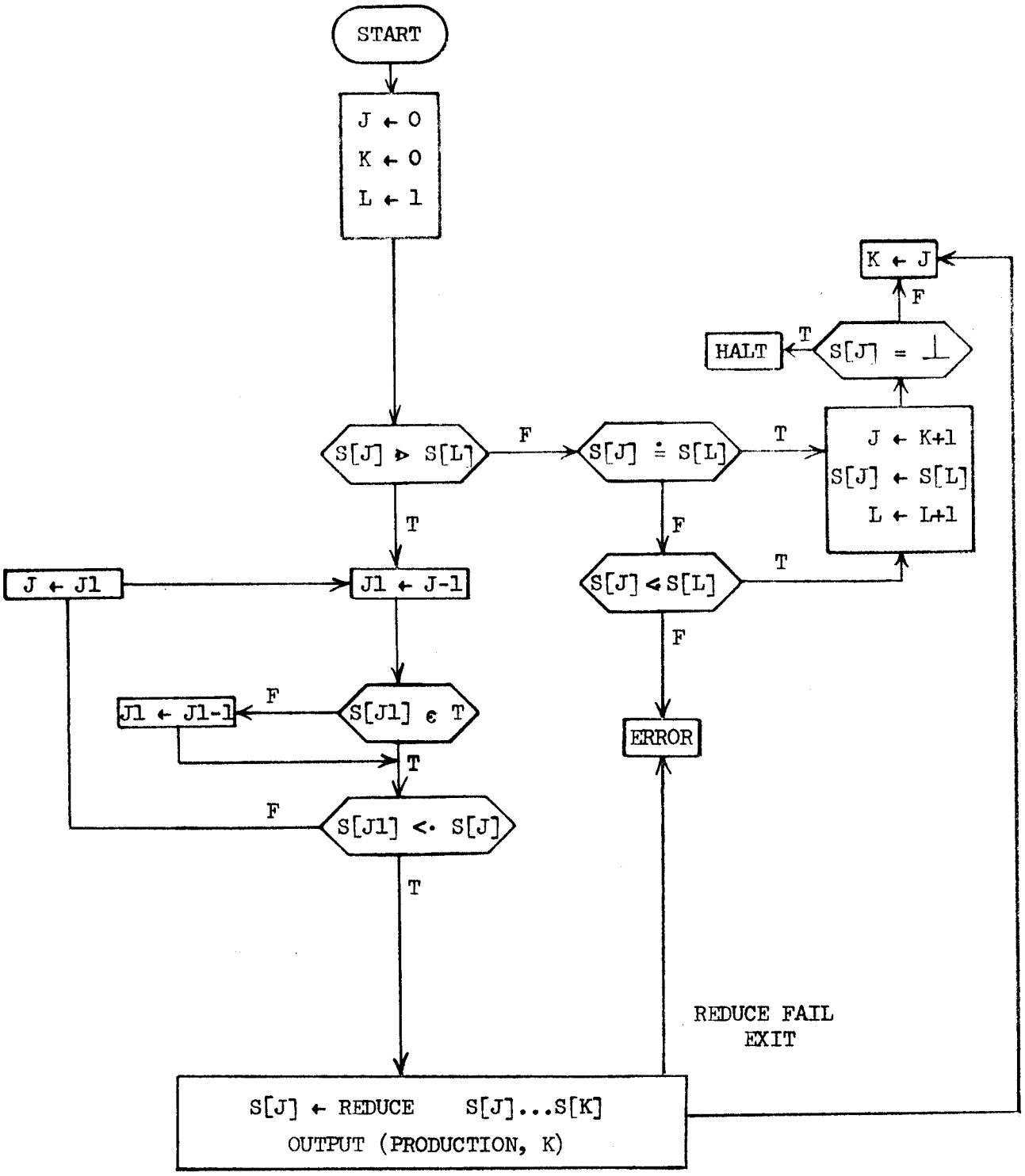
The algorithm parses S[0]...S[N] if and only if it halts with S[0] ...S[J] = y with $y = \perp S \perp$ or $y \in \perp(V - T)\perp \cup \{\perp\ \perp\}$ and $\perp S \perp \overset{*}{\Longrightarrow} y$.

Fig. 3.

We next remark that $(G, T, <\cdot, \doteq, \cdot>)$ is a canonical precedence scheme iff it is a precedence scheme and its induced automaton outputs only canonical H-sparse parses.

In the case of canonical precedence Fig. 3 simplifies to Fig. 4. In this case the input stream is read only and there is only one stack. It corresponds to the classical one-way PDA.

This algorithm will produce canonical H-sparse parses of strings in L(G).



The algorithm parses S[0]...S[X] if and only if it halts with S[0]...S[J] = y with $y = \perp S \perp$ or $y \in \perp (V - T) \perp \cup \{\perp \perp\}$ and $\perp S \perp \overset{*}{\Longrightarrow} y$.

Fig. 4.

Note that in any case the precedence relations do not detect Λ rules or chain rules not involving operators. However it is a reasonably straightforward process to ignore these rules using the chain table. The advantage of this was discussed with the definition of sparse parse.

One must discuss possible schemes for performing the reduce function in order to make the discussion of precedence detection complete.

First we discuss a way of storing the set P(T) compactly. Suppose we construct an $|V-T| \times (|V-T| + 1)$ boolean matrix, CHAIN. CHAIN is indexed by $x \in (V-T)$, $y \in (V-T) \cup \{\Lambda\}$ and CHAIN (x,y) = TRUE iff $x \overset{*}{\Longrightarrow} y$. Then by Lemma 4.1 given any $(A \to x_o A_o x_1 A_1 \ldots A_n x_n) \in P$ where $A_i \in T$, $x_i \in (V-T) \cup \{\Lambda\}$.

$$(A,x,y) \in P(T) \text{ iff } y = y_o A_o y_1 A_1 \ldots A_n y_n \text{ and CHAIN } (x_i,y_i) \text{ for } i = 1,\ldots, n.$$

So given any $(A \to x) \in P$ and $y \in V^*$ if $x = X_1 \ldots X_M$ and $y = Q_o \ldots Q_M$ where X_i and $Q_j \in V$, then the following algorithm answers yes or no to the question $(A,x,y) \in P(T)$. This provides a quick way of enumerating all relevant $(A,x,y) \in P(T)$ given y (hash coding of the $A \to x$ table will greatly reduce the number of candidates to be examined).

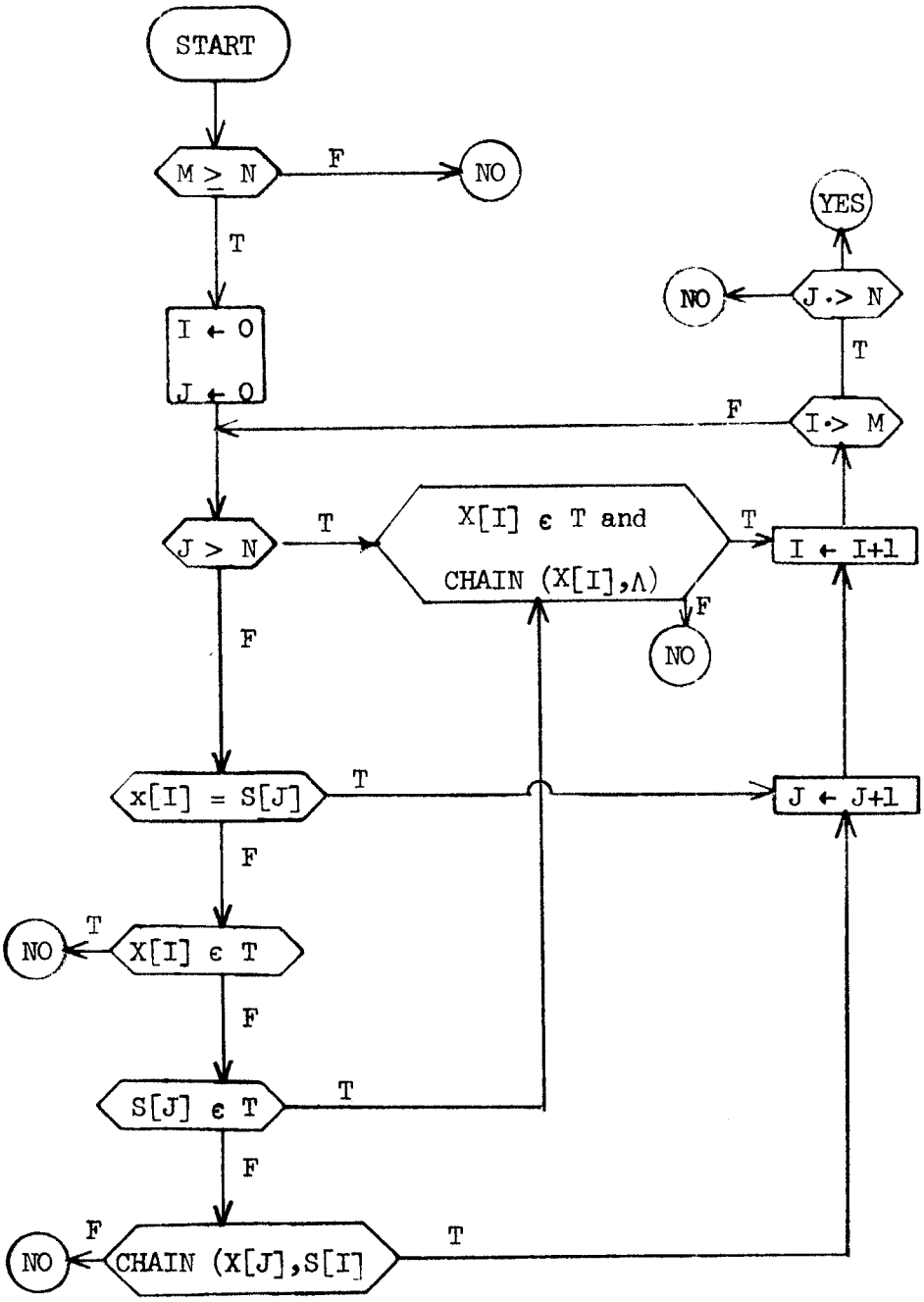
Now even if G is invertible the above algorithm may find more than one $(A,x,y) \in P(T)$ for a particular y. Consider the example

$$S \to A = B \mid A = C$$

$$B \to C$$

Then $(S, A = B, A = C)$, $(S, A = C, A = C) \in P(T)$ if $= \in T$.

In the case of an ambiguous grammar (like the one above) there is no hope of eliminating the nondeterminancy. REDUCE must return both possibilities. So we shall restrict attention to unambiguous grammars. In particular we shall examine the possibility of having REDUCE choose between the various members of P(T) on the basis of the context of y.



Part of The REDUCE Function

Fig. 5.

We now discuss the possibility of making reduce a deterministic function. We mentioned earlier that precedence analyzers are commonly required to be invertible. We will argue from a theoretical point of view that this is a very strong restriction. In particular it is known (Fischer [4]) that invertible simple precedence grammars are a subset of the 1-state deterministic languages and of BRC(1,1) languages. We will show that if more powerful reduction algorithms are allowed one can parse correspondingly more powerful grammars. We shall do this only for simple precedence since it allows one to ignore Λ rules. These results also have implications for more general classes of token sets.

The appendix contains the arduous proof of the following two results:

Theorem 4.1. Every Λ-free grammar is completely covered by a Λ free normal form grammar. If the original grammar is LR(k) or BRC(n,m) or unambiguous, then the resulting grammar is LR(k), BRC(n+r-1,m) or unambiguous respectively, where $r = \max \{lg(x), 1 | A \to x \text{ is in } P\}$.

Theorem 4.2.[14] Every Λ-free normal form grammar is completely covered by a simple precedence grammar. If the original grammar is unambiguous, LR(k) or BRC(n,m) then the new grammar will be unambiguous, LR(k) reducible and BRC(n+1,m) reducible respectively.

The exact constructions used in these proofs are of limited practical interest since they cause an explosion in the number of nonterminals. However they do yield the following result.

Theorem 4.3. If G is a P-reduced grammar then G is covered by a simple precedence grammar. If the original grammar is unambiguous, or LR(k) or BRC(n,m) then the resulting simple precedence

---

[14] This result is a generalization of Theorem 3.4 of Fischer [4].

grammar will also be unambiguous, LR(k) reducible or BRC(n+r,m) reducible where $r = \max \{lg(x) \mid A \rightarrow x \in P\}$.

Proof: Follows immediately from 4.1, 4.2 and the transitivity of covers.

The theoretical and practical implications of this result are quite interesting. First if each reduction step can be done in a number of elementary steps which is bounded by some constant then it follows from Proposition 4.1 that the entire parsing algorithm runs in linear time.

If one accepts the definition of cover as an abstraction of the idea of equivalence or grammars with respect to parsing, then we conclude that any Λ-free LR(k) or BRC(n,m) grammar may be parsed using V-canonical precedence relations and LR(k) or BRC(n,m) reduction respectively. There is reason to believe that this is a technique for drastically reducing the number of "state sets" of LR(k) parsers [1], [13], [14].

Section 5

Summary and Conclusions

Section 1 presented a short and relatively simple definition of BRC(n,m) closely related to the definition of LR(k). Section 2 introduced the concepts of sparse parse and of cover. It developed several results about covers which showed their usefulness. The most interesting result was that the Greibach normal form is a very weak form of cover but the operator form gives a strong cover. It is interesting to ask whether every LR(k) grammar is covered by a BRC(1,0) grammar. Knuth showed that every LR(k) grammar is equivalent to a BRC(1,0) grammar and we conjecture that this can be strengthened to be a strong form of cover.

Section 3 gave a complete characterization of canonical precedence detection schemes. It did this in the spirit of Colmerauer. However it is done in greater generality giving both necessary and sufficient conditions for the existence of canonical precedence relations.

We have not treated the "higher order" schemes of Wirth and Weber and of McKeeman which define the relations on pairs of strings of operators rather than simply on pairs of operators. We mention in passing that as defined in [18] and elsewhere the (2,2)-precedence relations for a normal form grammar are vacuous! (The fact that this error has gone unnoticed for five years is eloquent testimony to the usefulness of higher-order precedence). The introduction of token sets smaller than V solves many of the problems which higher-order schemes also solve. One may wonder at this. Especially since Theorem 4.3 shows that if one relaxes the invertibility constraint then every grammar is covered by a simple precedence grammar. The

advantages of small token sets are:

(a) They allow Λ-rules,

(b) They produce a sparse parse,

(c) They require smaller precedence tables,

and (d) They may eliminate stratification.[10]

The advantage of (b) was described in Fig. 1. The extra work required by reducing phrases which contain operands (see REDUCE of Section 4) is compensated for by the elimination or long chains of operands with no semantic significance. Of course this requires that the grammar and operator set be chosen so that any production with semantic significance contain an operator. Stratification can always be eliminated by using the construction of Theorem 4.2. But this may destroy invertibility. It also multiplies the size of the grammar. The grammar EULER shows the effects of this transformation.

This points out a bane of precedence analysis. Theorem 2.1A shows that any grammar is completely covered by an invertible grammar and Theorem 4.2 shows that every grammar is completely covered by a simple precedence grammar. Fischer [4] shows that requiring both of these properties yields a proper sub-family of the family of BRC(1,1) languages. Given even a simple language (e.g., $\{x \in \{a,b\}^* \mid \#_a(x) \neq \#_b(x)\}$) it is quite difficult to construct a simple precedence grammar which is also invertible.

This naturally suggests the possibility of using more general reduction schemes. Section 4 explored this possibility. It demonstrated that all LR(k) grammars can be done if LR(k) reduction is

---

(10) An operator B is stratified if there is some A so that

(i) $A \doteq B$ and $A <\cdot B$

or (ii) $B \doteq A$ and $A \cdot> B$

allowed and, all that BRC(k,m) grammars can be done if BRC reduction is allowed. Section 4 made heavy use of the concept of a covering grammar.

It also made recourse to the idea of the sparse parse consisting of those productions of the parse which contain operators.

Section 4 also dealt with non-canonical precedence schemes. It showed that in all cases these schemes run in linear time. This Thesis fails to characterize the non-canonical precedence relations. Colmerauer has had partial success in this direction but it still remains as an open and very difficult problem.

REFERENCES

1. Cheatham, T. E., Jr., "The Theory and Construction of Compilers", Notes for AM 219 R, Harvard University, Spring, 1967.

2. Colmerauer, A., Precedence, Analyse Syntaxique et Langages de Programmation, Thesis, University of Grenoble, 1967.

3. Feldman, J. and Gries, D., "Translator Writing Systems", Communications of the Association for Computing Machinery, Vol. 11, pp. 77-113, 1968.

4. Fischer, M. J., "Some Properties of Precedence Languages", Proceedings of the Symposium on Theory of Computing, pp. 181-190, May, 1969.

5. Floyd, R. W., "Syntactic Analysis and Operator Precedence", Journal of the Association for Computing Machinery, Vol. 10, pp. 316-333, 1963.

6. Floyd, R. W., "Bounded Context Syntactic Analysis", Communications of the Association for Computing Machinery, Vol. 7, pp. 62-66, 1964.

7. Ginsburg, S., The Mathematical Theory of Context-Free Languages, McGraw-Hill Book Company, New York, 1966

8. Ginsburg, S. and Harrison, M. A., "Bracketed Context-Free Languages", Journal of Computer and System Sciences, Vol. 1, pp. 1-23, 1967.

9. Ginsburg, S. and Harrison, M. A., "One-Way Nondeterministic Real-Time List-Storage Languages", Journal of the Association for Computing Machinery, Vol. 15, pp. 428-446, 1968.

10. Greibach, S. A., "A New Normal Form Theorem for Context-Free Phrase Structure Grammars", Journal of the Association for Computing Machinery, Vol. 12, pp. 42-52, 1965.

11. Hopcroft, J. E. and Ullman, J. D., Formal Languages and Their Relation to Automata, Addison Wesley Publishing Company, Reading, Massachusetts, 1969.

12. Ichbiah, J. D. and Morse, S. P., "Optimal Generation of Floyd-Evans Productions for Precedence Grammars", DR.S. 69.65. ND., Compagnie Internationale Pour L'informatique Les Clayes-Sous-Bois, France, 1969.

13. Knuth, D. E., "On the Translation of Languages From Left to Right", Information and Control, Vol. 8, pp. 607-639, 1965.

14. McKeeman, W. M., "An Approach to Computer Language Design", Technical Report CS 48, Computer Science Department, Stanford University, Stanford, California, 1966.

15. McNaughton, R., "Parenthesis Grammars", Journal of the Association for Computing Machinery, Vol. 14, pp. 490-500, 1967.

16. Pair, C., "Abre, piles et compilation", RFTI 7, 3 (1964), 199-216.

17. Reynolds, J. C., "Grammatical Covering", Argonne National Laboratory Technical Memo 96, March, 1968.

18. Wirth, N. and Weber, H., "Euler: A Generalization of ALGOL and its Formal Definition", Communications of the Association for Computing Machinery, Vol. 9, pp. 11-23, 89-99, 1966.

Appendix

Theorem 4.1. Let $G = (V, \Sigma, P, \perp S \perp)$ be a Λ-free context-free grammar. There exists a grammar $G' = (V', \Sigma, P', \perp S' \perp)$ and a production set $H' \subseteq P'$ so that $\langle G', H' \rangle$ completely covers G where

(i) G' is a Λ-free normal grammar

(ii) If G is unambiguous then so is G'

(iii) If G is LR(k) then so is G'

(iv) If G is BRC(n,m) then G' is BRC(n+r-1,m) where

$r = \max \{1, lg(x) | A \to x \text{ is in } P\}$.

Proof: Define $G' = (V', \Sigma, P', \perp [S] \perp)$ as follows for each $x \in V^*$, let $[x]$ be a new character. Let

$$V' = \Sigma \cup \{[A] | A \in V\} \cup \{[x] | A \to xy \text{ is in } P \text{ for some } x \in V^+, y \in V^*\}$$

and define

$$P_1 = \{[A] \to [B] | A \to B \text{ is in } P\}$$

$$P_2 = \{[A] \to [B][x] | A \to Bx \text{ is in } P, B \in V, \text{ and } x \in V^+\}$$

$$P_3 = \{[Ax] \to [A][x] | A \in V, x \in V^+, [Ax] \in V' - \Sigma\}$$

$$P_4 = \{[a] \to a | a \in \Sigma - \{\perp\}\}$$

We take $H' = P_1 \cup P_2$ and $P' = H' \cup P_3 \cup P_4$. A canonical sentential form x is said to be of

type 0 if $x \in (\perp \Sigma^* \perp) \cup (\perp \{[A] | A \in V\}^* \{[A] | A \in V - \Sigma\} \Sigma^* \perp)$

type 1 if $(x \in \perp \Sigma^* \perp) \cup (\perp \{[A] | A \in V\}^* \Sigma^* \perp)$

type 2 if $x \in \perp \{[A] | A \in V\} \{[x] | x \in V^2 V^*\} \Sigma^*$

Claim 1. Every canonical sentential form of G' is either of type 1 or type 2. If $(A \to w, j)$ is a handle of a type 2 sentential form G', then $A \to w$ is in $P_2 \cup P_3$.

Proof: The result follows from a straightforward induction on the length of a generation.

Next, define a homomorphism ψ which maps $(V')^*$ into $(V)^*$ by

$$\psi(\Lambda) = \Lambda$$

$$\psi(a) = a \quad \text{for each } a \in \Sigma$$

$$\psi(\lceil y \rceil) = y \quad \text{for each } [y] \in V' - \Sigma$$

$$\psi(a_1 \ldots a_n) = \psi(a_1) \ldots \psi(a_n) \quad \text{for all } a_1, \ldots, a_n \in V'$$

Remark 1. Let x be a canonical sentential form of type 1 and let i be such that $0 \leq i \leq lg(x)$. Then

$$lg(\psi(^{(i)}x)) = i$$

Remark 2. If x and x' are canonical sentential forms and $\psi(x) = \psi(x')$ then $x \underset{R}{\overset{*}{\Longrightarrow}} x'$ or $x' \underset{R}{\overset{*}{\Longrightarrow}} x$.

Now we define the map φ

$$\varphi([A] \to [B]) = A \to B \qquad \text{for each } [A] \to [B] \text{ in } P_1$$

$$\varphi([A] \to [B][x]) = A \to Bx \qquad \text{for each } [A] \to [B][x] \text{ in } P_2$$

Claim 2. Let $(A_i \to x_i)_{i=1}^n$ be the canonical production sequence in G' of the derivation

$$\perp [S] \perp \underset{R}{\overset{*}{\Longrightarrow}} xA_n y \underset{R}{\Longrightarrow} xx_n y = X_1 \ldots X_k \quad \text{with each } X_i \in V'$$

If j is the least positive integer so that $X_{j+1} \ldots X_k \in \Sigma^*$, then

(a) $(\varphi(A_i \to x_i))_{i=1}^n$ is a canonical generation in G of $\psi(xx_n y)$.

(b) If $A_n \to x_n$ is not in P_4 then a handle of $\psi(xx_n y)$ is $(B \to z;\ lg(\psi(X_1 \ldots X_j)) = lg(\psi(xx_n))$.

(c) $A_n \to x_n$ is in P_4 if and only if xx_ny is a type 1 sentential form and $\psi(xx_ny)$ has handle $(B \to z, q)$ where $j < q = \lg \psi(xx_n) = \lg(xx_n)$.

Proof: First we observe that

(*) if $A \to x$ is not in H' then $\psi(A) = \psi(x)$.

(**) if $A \to x$ is in H' then $\varphi(A \to x) = (\psi(A) \to \psi(x))$.

We now perform an induction on n. For $n = 1$, the result is straightforward. For the induction step, assume (a), (b), and (c) for $n < m$ and suppose $n = m$.

If $A_n \to x_n$ is not in H' then (*) implies that $\psi(xA_ny) = \psi(xx_ny)$ so (a) follows immediately from the induction hypothesis. If $A_n \to x_n$ is in H', then (**) implies $\varphi(A_n \to x_n) = (\psi(A_n) \to \psi(x_n))$ is in P. Thus $\psi(xA_ny) \underset{R}{\Longrightarrow} \psi(xx_ny)$. Therefore $\varphi(A_i \to x_i)_{i=1}^{n}$ is a canonical derivation of $\psi(xx_ny)$ and (a) has been established.

To prove (b), suppose that $A_n \to x_n$ is not in P_4. If $A_n \to x_n$ is in H then $\psi(xx_ny)$ has handle $(A_n \to x_n, \lg(\psi(xx_n)))$ by (a). By the choice of H', x_n contains no elements of Σ. Thus, $\lg(xx_n) \leq j$. Since the generation is right most, $\lg(xx_n) \geq j$, and hence $j = \lg(xx_n)$ or $xx_n = X_1 \ldots X_j$. Therefore $(A_n \to x_n, \lg(\psi(X_1 \ldots X_j)))$ is a handle of $\psi(xx_ny)$. On the other hand, suppose $A_n \to x_n$ is not in H which means that $A_n \to x_n$ is in P_3. We must have that xA_ny is of type 2. By Claim 1, xA_ny has a handle $(B \to z, k)$ where $B \to z$ is not in P_4 so that $\psi(xA_ny) = \psi(xx_ny)$ has handle $(B \to z, \lg(\psi(xA_n))) = (B \to z, \lg(\psi(X_1 \ldots X_j)))$ by the induction hypothesis. Therefore (b) has been established.

To consider (c), suppose that $A_n \to x_n$ is in P_4. Since $A_n = [x_n]$ and $y \in \Sigma^*$ we must have that xA_ny is of type 1. Thus $x \in \perp\{[A] \mid A \in V\}^*$ so that xx_n is of type 1. Now, consider $xA_ny = X_1 \ldots X_j[x_n]X_{j+2} \ldots X_k$.

(The string factors this way since $xx_ny = X_1 \ldots X_k$). The least integer i such that $X_{i+1} \ldots X_k$ is in Σ^* must be j+1. Further $\psi(xA_ny) = \psi(xx_ny)$ since $\psi(A_n) = \psi([x_n]) = \psi(x_n) = x_n \in \Sigma$. Thus $\psi(xA_ny)$ has handle $(B \to z, q)$. By (b), if $xA_ny = x'x_{n-1}y'$ and $A_{n-1} \to x_{n-1}$ is not in P_4, then $q = lg(\psi(X_1 \ldots X_j[x_n])) > j$. By the induction hypothesis, if $A_{n-1} \to x_{n-1}$ is in P_4 then $q > j+1 > j$. In either event, we have $q > j$.

Conversely, suppose that xx_ny is of type 1 and $j < q$. If $A_n \to x_n$ is not in P_4, then (b) yields that $lg(\psi(xx_n)) = q$. Since $A_n \to x_n$ is not in P_4, $x_n \notin \Sigma - \{\perp\}$. Thus, $xx_n = X_1 \ldots X_j$, and since xx_ny is of type 1, $lg(\psi(X_1 \ldots X_j)) = j$. Therefore $q = j$ but this is a contradiction which establishes (c) and completes the proof of Claim 2.

Claim 3. For each canonical sentential form x, there is a unique type 0 canonical sentential form x' so that $\psi(x) = \psi(x')$ and $x \overset{*}{\Longrightarrow} x'$. If G is unambiguous, then there is exactly one canonical derivation of $x \underset{R}{\overset{*}{\Longrightarrow}} x'$.

Proof: In view of Remark 2, it suffices to show that $\psi(x) = \psi(x')$. Let $x = X_1 \ldots X_n$ and let j be the least positive integer such that $X_{j+1} \ldots X_n$ is in Σ^*. By Claim 1, x must be of type 1 or of type 2.

Case 1. x is a type 1 canonical sentential form. Let k be the least integer such that $X_k \ldots X_{j-1} \in \{[a] \mid a \in \Sigma\}^*$ and note that $k \leq j$. Define $x' = X_1 \ldots X_{k-1}\ \psi(X_k \ldots X_{j-1})\ X_j \ldots X_n$ and note (by inspection of P', especially P_4) that $x \underset{R}{\overset{*}{\Longrightarrow}} x'$ by derivation $(X_i \to \psi(X_i))_{i=j-1}^{k}$. Of course $\psi(x) = \psi(x')$. Since, either $k = 1$ (which means $X_1 = \perp$) or $X_k \in V' - \Sigma$, note that x' is of type 0.

We claim that x' is the unique type 0 string such that $\psi(x') = \psi(x)$. To see this, suppose that $\psi(x'') = \psi(x)$ and x'' is type 0. Then

$lg(x') = lg(x'') = n$ by Remark 1 and we have $x'' = Y_1 \ldots Y_n$ and $\psi(Y_i) = \psi(X_i)$ for each i, $1 \leq i \leq n$. Clearly $X_{k+1} \ldots X_n = Y_{k+1} \ldots Y_n$ because each such X_i is either of the form [a] or a. If $k = 1$, we have $x' = x''$ and we would be done. If $k > 1$, then $X_k = [A]$ with $A \in V' - \Sigma$. Since $\psi(X_k) = \psi(Y_k)$, we have $Y_k = X_k = \psi(X_k)$ and because x' and x'' are type 0, it follows that $X_1 \ldots X_k = Y_1 \ldots Y_k$ and hence $x' = X_1 \ldots X_k \, \psi(X_k) X_{k+1} \ldots X_n = x''$ is unique.

Suppose that there were two canonical derivations of $x \overset{*}{\underset{R}{\Longrightarrow}} x'$. We already know that if $x \overset{*}{\underset{R}{\Longrightarrow}} y$ and y is type 0 with $\psi(y) = \psi(x)$ then $y = x$. Thus the only way that there can be two such derivations is if $x \overset{*}{\underset{R}{\Longrightarrow}} x' \overset{+}{\underset{R}{\Longrightarrow}} x'$. But $x' \overset{*}{\underset{R}{\Longrightarrow}} x'$ implies $X_k \overset{+}{\underset{R}{\Longrightarrow}} X_k$ in G'. By Claim 2, we have $\psi(X_k) \overset{+}{\Longrightarrow} \psi(X_k)$ in G which implies that G is ambiguous, a contradiction. Therefore the derivation of $x \overset{+}{\Longrightarrow} x'$ must be unique.

Case 2. x is a type 2 canonical sentential form. By inspection of P', particularly P_3, there is a unique string y so that $x \Longrightarrow y$. If y is of type 2, repeat this production. After at most $r = \max \{lg(y) \mid A \to y \text{ is in } P\} - 2$ steps, a unique string y' of type 1 is obtained. Note that $y \overset{*}{\underset{R}{\Longrightarrow}} y'$ and $\psi(y') = \psi(x)$. Now we use Case 1 and the claim follows.

Since Case 2 reduces to Case 1 and uniqueness has been shown for Case 1, then we are done.

Claim 4. Let $(A_i \to x_i)_{i=1}^{n}$ be a canonical derivation of $\perp S \perp \overset{*}{\underset{R}{\Longrightarrow}} xz$ in G where $z \in \Sigma^*$ and $x \in (V^*(V - \Sigma)) \cup \{\Lambda\}$. Then there is a unique type 0 sentential form yz in G' where $y \in (V')^*(V' - \Sigma)) \cup \{\Lambda\}$. Furthermore, there is a canonical generation $(B_i \to z_i)_{i=1}^{m}$ in G'