PARSING STRATEGIES: A CONCISE SURVEY†

-preliminary report-

Anton Nijholt
Koninginneweg 179
1075 CP Amsterdam The Netherlands

1. INTRODUCTION

After the introduction of context-free grammars and the use of BNF rules, compilers have been built in which we can distinguish methods of syntax-analysis. While initially sometimes many different ideas were used to do syntax-analysis for a given programming language and grammar, later formalizations of these ideas have led to many different parsing methods. Each of these methods can be shown to be suitable for a certain subclass of the context-free grammars.

If w is a sentence generated by a context-free grammar G, then a parsing strategy tells us in which way the productions of G, which are used in the generation of w, will be recognized by the parsing method. Two global strategies can be distinguished, the top-down and the bottom-up strategy. The class of deterministically bottom-up parsable grammars was introduced by Knuth/28/ in 1965. A few years later definitions for the class of deterministically top-down parsable grammars were introduced. Before that, parsing techniques such as precedence analysis and bounded context methods were used and formalized.

In the beginning of the seventies the use of precedence techniques was still advocated. The top-down or LL(k) method was not considered to be powerful enough, although it could be used for parts of programming languages. The bottom-up or LR(k) method seemed too difficult to use in practice. Efficient implementations were not yet known. However, especially after the work of DeRemer and LaLonde LR-methods (and in particular the SLR(1) and LALR(1) method) became well-known. Moreover, the use of parser generators made it worthwile to invest time and efforts in the further development of LR-methods, their error-correcting capabilities and their optimization. Further, both for LL- and LR-methods theory is being developed which makes it possible, by using semantic information, to parse non-LL- and non-LR-grammars with these

† The preparation of this paper was partially supported by a Natural Sciences and Engineering Research Council of Canada Grant No.A-7700.

methods. To illustrate the importance of these two methods we cite from Fisher and Weber/10/: "Also the parser generator is now such an important tool in language implementation that languages are now designed to be LALR(1) or LL(1) parsable." Two examples of such parser generators are described in /27,37/. Recently an LALR(1) grammar has been given for the (Revised) Ada language (cf. /48/).

Now that in parsing these two main techniques have become so popular, it is useful to survey the area of parsing strategies in order to see how other strategies can be defined as more restricted or more general cases of these LL- and LR-strategies. Then we can say more about the classes of languages which can be handled by certain strategies or combinations of these strategies and it becomes possible to give relations between parsing strategies and the possibility of certain parser optimizations and ways of code generation. Apart from this, the purpose of this paper is threefold. Firstly, we want to show that it is possible to discuss the many different strategies in such a way that they can be distinguished by certain basic characteristics. Secondly, we want to show that in the literature many useful concepts and ideas can be found which have not yet obtained sufficient attention in practice. Within the framework of this presentation these ideas can be discussed in a natural way. As a third point we want to mention that there exist several problems, e.g. equivalence problems and problems which deal with the possibility to obtain normal forms, which can be introduced and discussed within the present framework.

In this paper we discuss parsing strategies. This preliminary presentation is informal and apart from a few definitions we will not be concerned with definitions of the associated subclasses of the context-free grammars. Many of these definitions can be found in /41/, others will appear in forthcoming papers.

PRELIMINARIES

A context-free grammar (CFG) G is denoted by the quadruple (N,Σ,P,S), denoting nonterminals, terminals, productions and the start symbol, respectively. Roman capitals A, B, C, ... will usually stand for elements of N; a, b, c, ... will usually denote elements of Σ; w, x, y, z will denote strings over Σ and α, β, γ, δ, ... will denote strings over $V = N \cup \Sigma$. The empty string is denoted by ε. We have the usual notation $\Rightarrow$, $\underset{L}{\Rightarrow}$ and $\underset{R}{\Rightarrow}$ for derivations, leftmost derivations and rightmost derivations, respectively. The language of a CFG G is the set $L(G) = \{w \mid S \overset{*}{\Rightarrow} w \text{ and } w \text{ in } \Sigma^*\}$. If α in V^* and k is a non-negative integer, then $k : \alpha$ denotes α if $|\alpha|$ (the length of α) is less than or equal to k; otherwise it denotes the prefix of α with length k. Similarly, $\alpha : k$ is used for the suffix of α, and $FIRST_k(\alpha) = \{k : w \mid \alpha \overset{*}{\Rightarrow} w, w \text{ in } \Sigma^*\}$. A production $A \to \varepsilon$ is called an ε-production.

It is useful to distinguish positions in the productions. If $A \to X_1X_2\ldots X_n$ is a non-ε-production in P, then X_1 is the first symbol of the righthand side of this prod-

uction; A is called the lefthand side, X_1 will be referred to as the left corner of the production. The ith position of this production is the position after the ith symbol in the righthand side. The position before X_1 is called the zero position. In the formal definitions of the parsing strategies it is sometimes necessary to distinguish productions $A \to w$ with w in Σ^* from the other productions. Here we will not go into these details.

Let $G = (N,\Sigma,P,S)$ be a CFG. Grammar G is said to be left-recursive if there exists a derivation $A \overset{+}{\Rightarrow} A\alpha$ for some A in N and α in V^*. G is said to be ε-free if P does not have ε-productions; G is in Greibach normal form (GNF) if P is a subset of $N \times \Sigma N^*$, G is in Chomsky normal form (CNF) if P is a subset of $N \times (N^2 \cup \Sigma)$. Grammar G is in canonical two form if P is a subset of $N \times (N^2 \cup V)$; G is said to be uniquely invertible (u.i.) if $A \to \alpha$ and $B \to \alpha$ in P implies that A = B; G is said to be left factored if for any $\alpha \neq \varepsilon$, $A \to \alpha\beta$ and $A \to \alpha\gamma$ in P implies that $\beta = \gamma$. Finally, G is said to be in operator form if P is a subset of $N \times (V^* - V^*N^2V^*)$.

2. PARSING STRATEGIES, PART I

2.1. BASIC IDEAS

It is usual to distinguish between top-down and bottom-up strategies. Both notions say something about the order in which the productions are recognized. It is possible to use other strategies which can be considered as restricted bottom-up strategies. That is, although these strategies can be implemented as a shift/reduce parsing algorithm, the productions or parts of the productions have already been recognized in steps of the parsing algorithm which precede the reduce step. A systematic approach of these restrictions is not only useful from the point of view of the theory of parsing but, since productions will be provided with semantic information, also from the point of view of translation and code generation. Moreover, since semantic information can be used in the parsing process it is useful to formalize strategies in which it is known where and when parts of productions have been recognized.

There are several ways to discuss parsing strategies and the associated subclasses of the context-free grammars for which these strategies are suitable. E.g., it is possible to have conditions on:

a. productions or derivations of a grammar; consider e.g. the definitions of simple deterministic, LL(k), LR(k) and simple precedence grammars.

b. (LR-) state sets; see e.g. DeRemer/8/, Hammer/19/, Beatty/3/ and Kral and Demner/30/.

c. transition diagrams; see e.g. Aho and Ullman/2/ (recursive descent), Conway/6/ and Friede/11/.

d. parsers, pushdown transducers and syntax directed translation schemes;

see e.g. Aho and Ullman/1/, Soisalon-Soininen/54/, Brosgol/4/ and Moll/38/.

In Räihä and Ukkonen/50/ a distinction is made between recursive descent and recursive ascent parsing. In this informal approach we will mainly be concerned with ideas which will deal with approach a. However, any strategy and each class of grammars can be defined in any of these ways.

In Figure 1 we have displayed the situation that we have read and processed w and that we are going to read the yield of production $A \rightarrow X_1X_2\ldots X_n$.

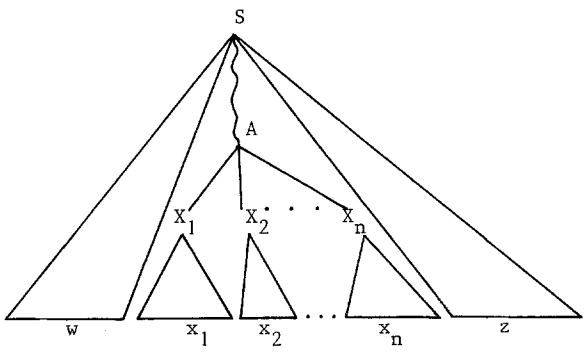

Figure 1.

The most obvious distinction we can make between parsing methods is that we can choose between the following two possibilities:

I. Each production will be recognized in the same way.

II. Each production has its own way in which it will be recognized.

Examples. LL(k) grammars are defined in such a way that for each production $A \rightarrow X_1X_2\ldots X_n$, as displayed in Figure 1, the production is recognized after having seen $k : x_1x_2\ldots x_nz$. On the other hand, Demers/7/ defines generalized left corner parsing, where each production can have a different position, say i, such that after having seen $k : x_{i+1}\ldots x_nz$ the production has to be recognized.

For all the strategies to be mentioned we can always distinguish between I and II. Further on we will not mention this distinction.

We now consider one production, $A \rightarrow X_1X_2\ldots X_n$, and we define a strategy by saying when this production and its component parts have to be recognized in the parsing process.

2.2. PRODUCTION ORIENTED STRATEGIES

In the production oriented strategies each production has a fixed position in its righthand side. Once the part of the righthand side which is to the left of this position has been recognized, the complete production has to be recognized.

2.2.1. BASIC STRATEGIES

In the basic strategies we do not make a distinction between the terminal and the nonterminal symbols in the righthand sides of the productions. That is, the position i is chosen independent of the occurrences of terminal and nonterminal symbols to the left of this position. Well-known examples are:

LL(k): i = 0 for each production
LC(k): i = 1 for each production
LR(k): i = n for each production

2.2.2. NONTERMINAL BASED STRATEGIES

In these strategies we do not demand that the production is recognized after the ith position has been reached (hence, after the ith symbol), but after the ith nonterminal in the righthand side has been recognized. An example is the extended left corner strategy (cf. Brosgol/3,4/), where i = 1 and which is suitable for extended left corner (ELC(k)) grammars.

2.2.3. TERMINAL BASED STRATEGIES

There exist a few strategies where the recognition of the productions is based on the recognition of the terminal symbols in the righthand sides. We mention the simple deterministic grammars and the real-time strict deterministic grammars of degree 1 (cf. Harrison and Havel/24/).

Since in generalized left corner parsing (Demers/7/) it is allowed that for each production we have a different position in its righthand side, the above mentioned strategies can be considered as special cases of generalized left corner parsing.

2.3. LEFTHAND SIDE PREDICTIVE STRATEGIES

In the lefthand side predictive strategies we distinguish between the recognition of the lefthand side A of a production $A \rightarrow X_1X_2...X_n$ and recognition of the complete

production.

2.3.1. COMBINATIONS WITH THE BASIC STRATEGIES

In this case we demand that A is recgnized at the ith position of the production and $A \rightarrow X_1X_2...X_n$ is recognized at the jth position ($i \leq j$).

Examples of classes of grammars for which this type of strategy can be used are the PLR(k) grammars (cf. Soisalon-Soininen and Ukkonen/55/), with i = 1 and j = n, and the LP(k) grammars (cf. /40,42/), with i = 0 and j = n. See Nijholt/40/ for other examples. In section 2.4 we will return to these strategies.

2.3.2. COMBINATIONS WITH THE NONTERMINAL BASED STRATEGIES

In this case we demand that A is recognized after the ith nonterminal symbol of the righthand side has been recognized, while $A \rightarrow X_1X_2...X_n$ is recognized after the jth nonterminal symbol has been recognized ($i \leq j$).

2.3.3. OTHER COMBINATIONS

Consider e.g. the strategy in which we demand that A is recognized after the ith nonterminal symbol has been recognized, while $A \rightarrow X_1X_2...X_n$ is recognized at position n. This is, for i = 1, the straightforward generalization from ELC(k) grammars (see section 2.2.2) to extended PLR(k) grammars, similar to the generalization from LC(k) grammars to PLR(k) grammars.

2.4. PARTITION PREDICTIVE STRATEGIES

Let us first consider the situation where π denotes a partition of $V = N \cup \Sigma$, such that $\Sigma \in \pi$. If $A \in N$, then $[A]$ denotes the block of π to which A belongs. We now can distinguish between recognition of:

$[A]$: at position i or after the ith nonterminal symbol

A : at position j ($i \leq j$) or after the jth nonterminal symbol

$A \rightarrow X_1X_2...X_n$: at position k ($j \leq k$) or after the kth nonterminal symbol

An example of such a strategy is the parsing method for strict deterministic grammars (with look-ahead). Here, i = 0 and $j = k = n$ for each production $A \rightarrow X_1...X_n$. Another interesting example is the method which can be used for weak PLR(k) grammars (cf. Ukkonen/58/), which can be defined in this way. Since this class of grammars has some properties which are essential for the parsing strategies which will be dealt with in the forthcoming sections, we will consider it in more detail. Therefore we

need the following definitions.

A production $A \to \beta$ is said to satisfy the LR(k) condition if for any pair of derivations

$$S \overset{*}{\underset{R}{\Rightarrow}} \alpha Aw \underset{R}{\Rightarrow} \alpha\beta w = \gamma w, \text{ and}$$

$$S \overset{*}{\underset{R}{\Rightarrow}} \alpha'A'x \underset{R}{\Rightarrow} \alpha'\beta'x = \gamma w', \text{ with}$$

$$k : w = k : w',$$

we may conclude that $\alpha = \alpha'$, $A = A'$ and $\beta = \beta'$.

Now consider the following conditions for a CFG G and a production $A \to X\beta$, with $X \in V$,

(i) $S \overset{*}{\underset{R}{\Rightarrow}} \alpha Aw \underset{R}{\Rightarrow} \alpha X\beta w$

(ii) $S \overset{*}{\underset{R}{\Rightarrow}} \alpha'A'w' \underset{R}{\Rightarrow} \alpha'\alpha''X\beta'w'$

(iii) $\alpha'\alpha'' = \alpha$ and $\text{FIRST}_k(\beta w) \cap \text{FIRST}_k(\beta' w') \neq \emptyset$

The production $A \to X\beta$ is said to satisfy:

a. the LC(k) condition if $\alpha A = \alpha'A'$, $\alpha'' = \varepsilon$ and $X\beta = X\beta'$
b. the PLR(k) condition if it satisfies the LR(k) condition and, moreover, $\alpha'' = \varepsilon$ and $A = A'$
c. the weak PLR(k) condition if it satisfies the LR(k) condition and, moreover, $\alpha'' = \varepsilon$
d. the left corner condition if $\alpha'' = \varepsilon$

Suppose that instead of b. we demand

b'. the partitioned PLR(k) condition if it satisfies the LR(k) condition and, moreover, $\alpha'' = \varepsilon$ and $[A] = [A']$

However, notice that condition b' is satisfied by any production which satisfies the weak PLR(k) condition if we consider the partition $\pi = \{\Sigma, N\}$. It follows that, from a purely theoretical point of view, the weak PLR(k) and the partitioned PLR(k) strategies coincide.

The following example will be illustrative for the properties of a weak PLR(k) grammar. Let G_0 be a CFG with productions

$S \to aAc \mid abBd$

$A \to bC$

$B \to C$

$C \to bC \mid b$

CFG G_0 is LR(1) (and ELC(1)). However, when we consider the right sentential form abCc then we can not determine where the righthand side of the production to be reduced starts, until we have seen terminal symbol c. That is, the left corner condition is not satisfied.

Let $A \to X\beta$ be a production in P, then X is the left corner of this production. It follows that it is useful to distinguish the recognition of the left corner of a production from the recognition of the other component parts of the production, since not every production of an arbitrary LR-grammar has the left corner condition. In the following sections we will consider other parsing strategies where this distinction is made.

Instead of using a partition of V it is also possible to use a weak partition of V. In that case the blocks of the "partition" are not necessarily disjoint. Nevertheless the recognition of a block (e.g. in condition b') gives information about the lefthand side of the production being recognized. In Pittl/49/ a generalization of strict deterministic grammars is given. One of the characterizations of this generalization uses weak partitions.

3. PARSING STRATEGIES, PART II

Let us consider the bottom-up parsing problem from the point of view of rightmost derivations. Notice that the formal definitions of the classes of grammars mentioned in the previous section are not always given in terms of rightmost derivations. Consider a rightmost derivation

$$\omega_n \overset{P_n}{\underset{R}{\Longrightarrow}} \omega_{n-1} \underset{R}{\Longrightarrow} \cdots \underset{R}{\Longrightarrow} \omega_2 \overset{P_2}{\underset{R}{\Longrightarrow}} \omega_1 \overset{P_1}{\underset{R}{\Longrightarrow}} \omega_0$$

where $\omega_n = S$ (the start symbol) and $\omega_0 \in \Sigma^*$. The goal is to find the string $P_1P_2\ldots P_n$ of productions. If we write $\omega_j = \alpha Aw$ and $\omega_{j-1} = \alpha\beta w$, then the problem reduces to the determination of β, $|\alpha\beta|$ and by which symbol the substring β at position $|\alpha\beta|$ in ω_{j-1} should be replaced in order to obtain ω_j. The pair $(A \to \beta, |\alpha\beta|)$ or, equivalently, the pair $(A \to \beta, |\alpha|)$ is called the handle of ω_{j-1} and β is called the phrase of this handle. If for each ω_i, $0 < i < n$, we can determine the handle, then we can go back from ω_0 to ω_n.

3.1. LR(k) AND BOUNDED RIGHT CONTEXT METHODS

For LR(k) grammars we are able to recognize the handle of $\omega_{j-1} = \alpha\beta w$ once we have seen k : w. From the examples and the definitions in section 2.4 we know that we can distinguish strategies in which the lefthand of the phrase is located before seeing k : w. E.g., the LC-, PLR- and weak PLR conditions are such that this lefthand is

located once we have recognized X, that is, the left corner of the production. For LR(k) and ELC(k) grammars this is not necessarily the case, as is shown by grammar G_0. Hence, it is possible to introduce strategies in which we distinguish between recognition of

(i) $|\alpha|$
(ii) $[A]$
(iii) A
(iv) $A \rightarrow X_1X_2\ldots X_n$

LR(k) grammars can be considered as grammars where the handle is determined by using k symbols to the right of the phrase and all the context to the left of the phrase. For (ℓ,k) bounded right context grammars (or, (ℓ,k) BRC grammars) the handle is uniquely determined by looking ahead k symbols and looking behind ℓ symbols. A production of a (ℓ,k) BRC grammar does not necessarily satisfy the left corner condition. Consider again example grammar G_0 which is (1,1) BRC and not weak PLR(k) for any $k \geq 0$. Clearly, many of the strategies which have been defined in section 2 as restrictions of the LR- or deterministic bottom-up strategy, can now be used to define restrictions of the (ℓ,k) BRC strategy. In section 4 we will return to LR(k) and (ℓ,k) BRC techniques in combination with precedence techniques.

3.2. PRECEDENCE METHODS

Instead of looking at the left and right context of the phrase of a handle, we can consider relations between (strings of) symbols in order to determine the handle of a right sentential form. In analogy with the LR(k) strategy where the complete context to the left of the phrase, together with k symbols of look-ahead, is used, we now can introduce a precedence based strategy where the elements of the relation are pairs consisting of a regular set and a string of length k. In this case we ought to talk about regular precedence relations. An adapted and more restricted version of this idea has been used in Shyamasundar/53/. Moreover, it is possible to introduce the analogue of the (ℓ,k) BRC strategy. Then we have the u.i. (ℓ,k) precedence or the u.i. extended precedence technique. Here we have extended precedence relations between strings of length ℓ and k, respectively. The (1,1) precedence relations are usually referred to as simple precedence or Wirth-Weber precedence relations. For these three cases, i.e. regular, extended and simple precedence it is possible to introduce strategies which use the restrictions mentioned in section 2.

A bibliography on precedence relations can be found in Nijholt/47/.

3.2.1. SIMPLE PRECEDENCE RELATIONS

We spend a few notes on the left corner condition (see section 2) in connection

with precedence relations. Here we will only give the simple precedence relations $\lessdot$, $\doteq$ and $\gtrdot$. These relations on $N \cup \Sigma$ are defined as follows:

(i) $X \lessdot Y$, if there exists $A \rightarrow \alpha X B \beta$, such that $B \overset{+}{\Rightarrow} Y\gamma$, for some $\gamma \in V^*$

(ii) $X \doteq Y$, if there exists $A \rightarrow \alpha XY\beta$ in P

(iii) $X \gtrdot a$, with $a \in \Sigma$, if there exists $A \rightarrow \alpha BY\beta$ such that $B \overset{+}{\Rightarrow} \gamma X$ and $Y \overset{*}{\Rightarrow} a\delta$, for some $\gamma, \delta \in V^*$.

A CFG without ε-productions is now called a precedence grammar if at most one simple precedence relation exists between any pair of symbols in $N \cup \Sigma$. Hence, if G is a precedence grammar then we can uniquely determine the phrase which has to be reduced (cf. Aho and Ullman/1/). However, unless the grammar is u.i., we do not know to which symbol this phrase has to be reduced. In Shyamasundar/53/ another method is given to determine the reduction which has to be made.

Notice that if a CFG G has unique precedence relations, then the left corner condition is satisfied. In fact, we only have to demand that $\lessdot$ is disjoint from the union of $\doteq$ and $\gtrdot$ to make sure that this condition is satisfied. Obviously, the same remark holds for regular and extended precedence relations.

3.2.2. WEAK PRECEDENCE

In the case of weak precedence the relation $\gtrdot$ is disjoint from the union of $\lessdot$ and $\doteq$. The relations $\lessdot$ and $\doteq$ are not necessarily disjoint. However, it is always possible to determine the left corner of the handle since there is an extra condition. This condition says that if $A \rightarrow \alpha X \beta$ and $B \rightarrow \beta$ are productions, $X \in V$, then neither of the relations $X \lessdot B$ and $X \doteq B$ are valid. However, this condition is not strong enough to guarantee that the grammar satisfies the left corner condition. We give a counter-example. Grammar G_1 with productions

$S \rightarrow aA \mid D$

$A \rightarrow cB$

$D \rightarrow acC$

$B \rightarrow bB \mid b$

$C \rightarrow bC \mid c$

is an example of a u.i. weak precedence grammar for which the left corner condition is not satisfied. Notice, that therefore grammar G_1 is also a counter-example to the result suggested in excercise 5.3.22 of Aho and Ullman/1/.

3.2.3. OPERATOR PRECEDENCE RELATIONS

Operator precedence relations (cf. Aho and Ullman/1/) are defined between the terminal symbols of an operator grammar. That is, a grammar in which no production has a righthand side with two adjacent nonterminals. Whenever an operator grammar has

unique precedence relations then the left corner condition is satisfied. Operator precedence parsing is "skeletal" parsing. The productions are determined up to their nonterminal symbols.

Analogous to the case of simple precedence relations it is possible to define weak operator precedence relations (cf. Sudborough/57/). Notice that grammar G_1 is an example of a grammar which is a weak operator precedence grammar which does not satisfy the left corner condition.

3.2.4. CANONICAL PRECEDENCE RELATIONS

In this preliminary report we confine ourselves to the remark that in Gray and Harrison/17/ a general theory of precedence relations is presented which includes the simple precedence and the operator precedence techniques.

Note. It should be mentioned that the restrictions of the LR-strategies (cf. section 2) do not necessarily lead to grammars which satisfy the condition that $\lessdot$ is disjoint from the union of $\doteq$ and $\gtrdot$. The following simple deterministic grammar G_2 with productions

$S \rightarrow aAb \mid bAB$

$A \rightarrow a$

$B \rightarrow b$

is an example of a grammar with $\lessdot$ not disjoint from $\doteq$. Hence, the strategies which are mentioned in section 2 can further be refined by introducing conditions for the precedence relations of the grammar.

3.3. OTHER STRATEGIES WITH PRECEDENCE RELATIONS

As discussed before, also with precedence relations we can demand that certain characteristics of the productions will be recognized before the recognition of the complete production. Therefore, some of the ideas of section 2 can be used for precedence based strategies. Moreover, it is possible to introduce strategies in which both the idea of relations between strings as the idea of looking at the context of a phrase are used. Here we confine ourselves to the following notes (cf. also section 4.3.2).

Precedence relations can be used to determine the handle. In the case of weak precedence relations we had $\gtrdot$ disjoint from the union of $\doteq$ and $\lessdot$. Therefore we needed another condition to determine the left corner of the handle. Now, instead, we can demand that $\lessdot$ is disjoint from $\doteq$ and have a special condition which makes it possible to locate the righthand of the phrase and to determine the reduction which has to be made. This condition can be such that it is not necessary to demand that the grammar is u.i. We mention a few examples. It should be noted that in the formal definitions

of these classes of grammars the precedence relations are not always mentioned.

An example is the class of production prefix grammars (cf. Geller, Graham and Harrison/13/) where $\lessdot$ is disjoint from $\doteq$ and the handle is determined with a SLR(1) condition. The production prefix grammars satisfy the left corner condition.

For left local precedence grammars (cf. Lomet/32/ and Pittl/49/) the complete string to the left of the phrase is used to distinguish the precedence relations. Also in this case the left corner condition is always satisfied (cf. /41/ for a proof).

For precedence regular grammars (cf. Shyamasundar/53/) we have unique precedence relations. The complete context to the left of the phrase is used to determine the reduction.

Mixed strategy precedence grammars (cf. Aho and Ullman/1/) have (extended) precedence relations such that $\gtrdot$ is disjoint from the union of $\doteq$ and $\lessdot$. The complete handle is recognized by considering the context of a phrase in a bounded right context way.

As a last example we mention the left context precedence grammars (cf. Moll/38/). They constitute a subclass of the (not necessarily u.i.) precedence grammars.

4. PARSING STRATEGIES AND GRAMMATICAL TRANSFORMATIONS

Whenever we speak of transformations in combination with parsing, then it is useful to have transformations which preserve some of the structure of the original grammar. In this way it is possible to use the newly obtained grammar for parsing while the semantic actions of the original grammar can be evoked. This notion of preserving the structure has been formalized in various ways (cf. Soisalon-Soininen and Wood/56/, Nijholt/40/ and Hunt and Rosenkrantz/26/). Most of the transformations which we mention below are such that this preservation of structure can be described by a grammatical cover.

4.1. TRANSFORMATIONS TO LL(k) GRAMMARS

We distinguish two types of transformations:

(i) Transformations that are guaranteed to yield LL(k) grammars when they are applied to a specific subclass of the context-free grammars. Cf. e.g. Hammer/19/ and Soisalon-Soininen and Ukkonen/55/.

(ii) Other transformations. That is, there exist several transformations (e.g. the transformation to non-left-recursive grammars) which are useful when a top-down parsing method is desired. However, not for all these transformations the domain which is converted into the LL(k) grammars has been characterized.

Once we have obtained an LL(k) grammar there are still several useful transformations which can be applied in order to obtain better parsing properties. The most well-known is the transformation to strong LL(k) grammars. If an LL(k) grammar G generates an LL(1) language L, then there exists an equivalent LL(1) grammar G'. However, transformations from G to G' are not known. There are also transformations for LL(k) grammars which deal with desirable error-recovery properties of the grammar (cf. e.g. Ghezzi/14/).

4.2. TRANSFORMATIONS TO PRECEDENCE GRAMMARS

There exist transformations which convert ε-free grammars into precedence grammars. See McAfee and Presser/35/ and the references which are given there. However, it is not necessarily the case that the newly obtained grammar is u.i. or that one of the other desirable parsing properties (cf. sections 3.2 and 3.3) is satisfied. It is known (cf. Aho and Ullman/1/) that each u.i. weak precedence grammar can be converted into an equivalent u.i. precedence grammar. A similar result can not be given for weak operator precedence grammars since their class of languages properly contains the operator precedence languages (cf. Sudborough/57/). In Graham/15/ it is shown that each LR(k) grammar (and therefore also each u.i. extended precedence grammar) can be transformed to a u.i. (2,1) precedence grammar.

4.3. TRANSFORMATIONS TO LR(k) AND (ℓ,k) BRC GRAMMARS

There exist transformations which convert non-LR-grammars into LR-grammars. However, hardly any research has been done on the characterization of classes of grammars which can be converted into LR-grammars. Once we have an LR(k) grammar then there exist many transformations which can be applied to obtain a grammar with desirable parsing properties. Each LR(k) grammar can be transformed to an LR(1) grammar and to a (1,1) BRC grammar (cf., e.g., Mickunas/36/, Graham/15/ and Aho and Ullman/1/). Some other transformations will be mentioned below.

4.3.1. LR(k) AND (ℓ,k) BRC GRAMMARS WITH THE LEFT CORNER CONDITION

LR(k) grammars in canonical two form or in Greibach normal form (GNF) satisfy the left corner condition (cf. Ukkonen/56/). Therefore, every LR(k) grammar can be transformed into an LR(k) grammar which satisfies this condition. In fact, every LR(k) grammar in GNF is a left local precedence grammar (see section 3.3, Theorem 5.3 in Moura/39/, Nijholt/45/ and Pittl/49/). Moreover, every left local precedence grammar can be transformed to a strict deterministic grammar (with look-ahead). See again Moura/39/.

In Harrison/20/ various transformations can be found which can be applied to

strict deterministic grammars (e.g. a transformation from strict deterministic grammars to grammars which are both strict deterministic and $(\ell,0)$ BRC, $\ell \geq 0$).

4.3.2. LR(k) AND (ℓ,k) BRC GRAMMARS WITH DISJOINT PRECEDENCE RELATIONS

Instead of transformations from LR(k) grammars to LR(k) grammars with the left corner condition, it is also possible to consider transformations which convert LR(k) grammars into LR(k) grammars with precedence relations which are such that the left corner condition is satisfied. Such transformations have been studied in Graham/15/ and in Gray and Harrison/18/. An important result in these papers is that every LR(k) grammar can be transformed to an LR(k) grammar which has the property that, except for the lefthand side of the production, the handle can be recognized with simple precedence relations. For the determination of the reduction which has to be made it is sufficient to consider k symbols of look-ahead. A similar result is obtained for (ℓ,k) BRC grammars.

4.4. TRANSFORMATIONS TO NORMAL FORMS

Especially in section 4.3.1 we already mentioned parsing properties of LR(k) grammars in certain normal forms. Some other properties can be found in section 5.1. It should be noted that any transformation to GNF yields a grammar which has the property that $\lessdot$ is disjoint from $\doteq$. Cf. Geller, Harrison and Havel/12/ and Nijholt /43/ for such transformations for strict deterministic grammars. It will be clear from the previous sections that grammars which are u.i., in GNF, in operator form, left factored etc. all have special parsing properties.

5. PARSING STRATEGIES AND LANGUAGES

Also from the point of view of languages it is interesting to compare parsing strategies. In this extended abstract we can only point out some of the interesting problems. They deal with language classes, equivalence problems and language hierarchies.

5.1. LANGUAGE CLASSES

The class of LL(k) languages is a proper subclass of the class of LR(1) or deterministic languages. Since each LL(k) grammar is an LR(k) grammar it is interesting to consider classes of grammars between the LL(k) and LR(k) grammars and to try to understand the realations between the associated parsing strategies and the classes of languages which are generated. For example, the class of extended LC(k) languages

lies properly between the LL(k) and deterministic languages. On the other hand, the class of PLR(k) languages coincides with the class of LL(k) languages. It is an open problem whether the class of extended PLR(k) languages (cf. section 2.3.3) coincides with the extended LC(k) languages. Other classes of languages which constitute proper subclasses of the deterministic languages are the simple and operator precedence languages and the real-time strict deterministic languages.

There exist interesting relations between certain normal forms and the languages which can be generated. E.g., every extended LC(k) grammar in Chomsky normal form or in canonical two form is an LC(k) grammar. Therefore it can only generate an LL(k) language. Similarly, each strict deterministic grammar in Chomsky normal form can only generate a real-time strict determonistic language.

5.2. LANGUAGE HIERARCHIES

For LL(k) languages there exists the well-known Kurki-Suonio hierarchy. That is, for each $k > 0$ there exists an LL(k) language which can not be generated by an LL(k-1) grammar. For strict deterministic grammars and languages such a hierarchy can be given in terms of the sizes of the blocks of the partition (cf. Harrison and Havel /22/). Hierarchies for (real-time) deterministic languages can also be found in Harrison and Yehudai/25/ and Yehudai/59/. An obvious open problem is the existence of a hierarchy for extended LC(k) languages.

5.3. EQUIVALENCE PROBLEMS

The equivalence problems are closely related to the problems mentioned in section 5.1. Once we are able to characterize the properties of grammars between the LL(k) and LR(k) grammars which lead to different classes of languages in a sufficiently detailed way, then we are able to say more about the equivalence problems for these classes of languages.

The equivalence problems for LL(k) and real-time strict deterministic grammars with look-ahead are decidable (cf. Nijholt/44/). The problems for ELC(k), simple precedence and operator precedence grammars have not yet been considered in the literature.

6. MISCELLANEOUS

We want to mention a few parsing strategies which are not yet included in the foregoing sections. Schlichtiger/51,52/ has introduced the partitioned chain grammars. Kral and Demner/29/ and Kretinsky/31/ have introduced "semi-top-down" strategies. Hammer/19/ obtains a new class of grammars by introducing restrictions on the state sets of the LR(k) parsing algorithm. Finally we mention that each parsing method can

be generalized by using semantic information or by using "regular" instead of finite look-ahead.

REFERENCES

1. Aho, A.V. and J.D. Ullman. The Theory of Parsing, Translation and Compiling, Vols. 1 and 2. Prentice Hall, Englewoods Cliffs, N.J., 1972 and 1973.

2. Aho, A.V. and J.D. Ullman. Principles of Compiler Design. Addison Wesley, Reading, Mass., 1977.

3. Beatty, J.C. On the relationship between the LL(1) and LR(1) grammars. CS-79-36, University of Waterloo, Waterloo, 1979.

4. Brosgol, B.M. Deterministic translation grammars. TR3-74, Harvard University, Cambridge, Mass., 1974.

5. Brosgol, B.M. Deterministic translation grammars. Proc. Eight Princeton Conf. on Information Sciences and Systems 1974, 300-306.

6. Conway, M. Design of a separable transition diagram compiler. Comm. ACM 6 (1963), 396-408.

7. Demers, A.J. Generalized left corner parsing. Conf. Record of the Fourth ACM Symp. on Principles of Programming Languages 1977, 170-182.

8. DeRemer, F.L. Simple LR(k) grammars. Comm. ACM 14 (1971), 453-460.

9. Deussen, P. One abstract parsing algorithm for all kinds of parsers. In: Automata, Languages and Programming, H.A. Maurer (ed.), Lect. Notes in Comp. Sci. 71 (Springer, Berlin, 1979), 203-217.

10. Fisher, G.A. and M. Weber. LALR(1) parsing for languages without reserved words. SIGPLAN Notices 14, November 1979, 26-30.

11. Friede, D. Transition diagrams and strict deterministic grammars. In: 4th GI Conf. on Theoretical Computer Science, K. Weihrauch (ed.), Lect. Notes in Comp. Sci. 14 (Springer, Berlin, 1979), 113-123.

12. Geller, M.M., M.A. Harrison and I.M. Havel. Normal forms of deterministic grammars. Discrete Mathematics 16 (1976), 313-321.

13. Geller, M.M., S.L. Graham and M.A. Harrison. Production prefix parsing. In: Automata, Languages and Programming, J. Loeckx (ed.), Lect. Notes in Comp. Sci. 14 (Springer, Berlin, 1974), 232-241.

14. Ghezzi, C. LL(1) grammars supporting an efficient error handling. Information Processing Letters 3 (1975), 174-176.

15. Graham, S.L. Precedence languages and bounded right context languages. Ph. D. Thesis, Dept. of Computer Science, Stanford University, California, 1971.

16. Graham, S.L. On bounded right context languages and grammars. SIAM J. of Comput. 3 (1974), 224-254.

17. Gray, J.N. and M.A. Harrison. Canonical precedence schemes. J. Assoc. Comput. Mach. 20 (1973), 214-234.

18. Gray, J.N. and M.A. Harrison. On the covering and reduction problems for context-free grammars. J. Assoc. Comput. Mach. 19 (1972), 675-698.

19. Hammer, M.A. A new grammatical transformation into deterministic top-down form. Mac TR-119, Ph. D. Thesis, Massachusetts Institute of Technology, 1974.

20. Harrison, M.A. Introduction to Formal Language Theory. Addison Wesley, Reading, Mass. 1978.

21. Harrison, M.A. On covers and precedence analysis. In: GI-3. Jahrestagung, W. Brauer (ed.), Lect. Notes in Comp. Sci. 1 (Springer, Berlin, 1973), 2-17.

22. Harrison, M.A. and I.M. Havel. Strict deterministic grammars. J. Comput. System Sci. 7 (1973), 237-277.

23. Harrison, M.A. and I.M. Havel. On the parsing of deterministic languages. J. Assoc. Comput. Mach. 21 (1974), 525-548.

24. Harrison, M.A. and I.M. Havel. Real-time strict deterministic languages. SIAM J. of Comput. 1 (1972), 333-349.

25. Harrison, M.A. and A. Yehudai. A hierarchy of deterministic languages. J. Comput. System Sci. 19 (1979), 63-78.

26. Hunt III, H.B. and D.J. Rosenkrantz. Complexity of grammatical similarity relations. Proc. of the Conf. on Theoretical Computer Science, Waterloo, 1977, 139-145.

27. Johnson, S.C. YACC - yet another compiler-compiler. CSTR 32, Bell Laboratories, Murray Hill, New Jersey.

28. Knuth, D.E. On the translation of languages from left to right. Information and Control 8 (1965), 607-639.

29. Kral, J. and J. Demner. Semi-top-down syntax analysis. Research report UVT 6/73, Tecnnical University of Prague, 1973.

30. Kral, J. and J. Demner. A note on the number of states of DeRemer's recognizer. Information Processing Letters 2 (1973), 22-23.

31. Kretinsky, M. Semi-top-down syntax analysis of precedence grammars. Scripta Fac. Sci. Natur. UJEP Brunensis Math. 8 (1978), 1-11.

32. Lomet, D.B. Automatic generation of multiple exit parsing strategies. In: Automata, Languages and Programming, J. Loeckx (ed.), Lect. Notes in Comp. Sci. 14 (Springer, Berlin, 1974), 214-231.

33. Lomet, D.B. A formalization of transition diagram systems. J. Assoc. Comput. Mach. 20 (1973), 235-257.

34. Mayer, O. A framework for producing deterministic canonical bottom-up parsers. In: Mathematical Foundations of Computer Science, Lect. Notes in Comput. Sci. 64 (Springer, Berlin, 1978), 355-363.

35. McAfee, J. and L. Presser. An algorithm for the design of simple precedence grammars. J. Assoc. Comput. Mach. 19 (1972), 385-395.

36. Mickunas, M.D. On the complete covering problem for LR(k) grammars. J. Assoc. Comput. Mach. 23 (1976), 17-30.

37. Milton, D.R., L.W. Kirchhoff and B.R. Rowland. An ALL(1) compiler generator. SIGPLAN Notices 14, August 1979, 152-157.

38. Moll, K.R. Left context precedence grammars. Acta Informatica 14 (1980), 317-336.

39. Moura, A. Syntactic equivalence of grammar classes. Ph. D. Thesis, 1980, Berkeley.

40. Nijholt, A. Context-Free Grammars: Covers, Normal Forms, and Parsing. Lect. Notes in Comp. Sci. 93 (Springer, Berlin, 1980).

41. Nijholt, A. and J. Pittl. A framework for classes of grammars between the LL(k) and LR(k) grammars. In preparation (see also CSTR-80-25, McMaster University).

42. Nijholt, A, and E. Soisalon-Soininen. Ch(k) grammars: A characterization of LL(k) languages. In: Mathematical Foundations of Computer Science. J. Becvár (ed.), Lect. Notes in Comp. Sci. 74 (Springer, Berlin, 1979), 390-397.

43. Nijholt, A. Strict deterministic grammars and Greibach normal form. Elektr. Informationsverarbeitung und Kybernetik (EIK) 15 (1979), 395-401.

44. Nijholt, A. The equivalence problem for LL- and LR-regular grammars. In: Fundamentals of Computation Theory. Proceedings of the 3rd Conference, 1981.

45. Nijholt, A. On the equivalence problem for extended left corner grammars, manuscript, 1981.

46. Nijholt, A. On the relationship between the LL(k) and LR(k) grammars. submitted for publication.

47. Nijholt, A. Precedence relations: A bibliography. manuscript, 1981.

48. Persch, G., G. Winterstein, S. Drossopoulou and M. Dausmann. An LALR(1) grammar for (Revised) Ada. SIGPLAN Notices 16, No.3, March 1981, 85-98.

49. Pittl, J. On LLP(k) grammars and languages. Theoret. Comput. Sci. 16 (1981).

50. Räihä, K.-J. and E. Ukkonen. Balancing syntactic and semantic power in compiler specification. In: Information Processing 80, North Holland, 1980, 65-70.

51. Schlichtiger, P. Kettengrammatiken: Ein Konzept zur Definition handhabbarer Grammatikklassen mit effizientem Analyseverhalten. Ph. D. Thesis, Kaiserslautern, 1979.

52. Schlichtiger, P. Partitioned chain grammars. In: Automata, Languages and Programming, J.W. de Bakker and J. v. Leeuwen (eds.), Lect. Notes in Comp. Sci. 85 (Springer, Berlin, 1980), 555-568.

53. Shyamasundar, R.K. Precedence regular grammars. Int. J. Comput. Math. 7 (1979), 173-186.

54. Soisalon-Soininen, E. Characterization of LL(k) languages by restricted LR(k) grammars. Ph. D. Thesis, Report A-1977-3, University of Helsinki.

55. Soisalon-Soininen, E. and E. Ukkonen. A method for transforming grammars into LL(k) form. Acta Informatica 12 (1979), 339-369.

56. Soisalon-Soininen, E. and D. Wood. On a covering relation for context-free grammars. CSTR 80-CS-21, McMaster University, Hamilton.

57. Sudborough, I.H. A note on weak operator precedence grammars. IPL7 (1978) 213-218.

58. Ukkonen, E. A modification of the LR(k) method for constructing compact bottom-up parsers. Automata, Languages and Programming, LNCS 71, 1979, 646-658.

59. Yehudai, A. A hierarchy of real-time deterministic languages and their equivalence. manuscript, 1980.