One Abstract Accepting Algorithm
for all Kinds of Parsers

P. Deussen
Fakultät für Informatik
Universität Karlsruhe

Introduction.

The techniques of syntactical analysis fill a vast amount of literature, their commonalities however are darkened by details (e.g. items, local look-ahead etc.) which are important for ultimate refinements of parsing algorithms but which should be hidden as long as possible if one is concerned with the principles of syntax analysis. Syntax analysis has three main aspects:

Semi-Thue aspect. Formal languages are generated by Semi-Thue systems (e.g. contextfree grammars) and they are accepted by Semi-Thue systems (e.g. push-down acceptors). To each generating Semi-Thue system there is at least one accepting Semi-Thue system whose accepting sequences are in one-to-one correspondence with left- (or right-)most derivations of the generating system; the accepting system yields a parse of the accepted word.

Algorithmic aspect. Special accepting Semi-Thue systems directly give rise to an abstract accepting algorithm $\alpha$. This algorithm when started from an initial situation selects productions from the Semi-Thue system according to a predicate $P$, applies one of them and then continues. Depending on properties of $P$, $\alpha$ turns out to be partially correct or deterministic, respectively. A recursive formulation of $\alpha$ after one small change directly yields the offspring of all backtracking algorithms used in syntax analysis including those of recursive descent.

Finite automata aspect. Efficiency of $\alpha$ solely depends on that of the predicate $P$. In practically important cases, e.g. contextfree languages, $P$ is decidable by means of a finite state acceptor (which is again a special accepting Semi-Thue system). Subset construction and state reduction are consequently the main tools for improving efficiency.

This paper is mainly concerned with the algorithmic and the finite automata aspect whereas the first one is only sketched. It points out that on this level of abstraction the proof of partial correctness of $\alpha$ is almost trivial; furthermore, that almost all parsable language
classes - e.g. LL(k), SLL(k), LR(k), SLR(k), LALR(k), LC(k), SLR(k) grammars - are characterized by requiring $\varepsilon$ to be deterministic.

1. Semi-Thue system aspect.

For any contextfree production system $\Pi$ with typical productions $A::=r$ one obtains two different types of push-down acceptors which expressed in terms of accepting Semi-Thue systems are obtained according to the table:

<table>
<thead>
<tr>
<th>$\Pi$</th>
<th>$\Pi_{LR}$</th>
<th>$\Pi_{LL}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A::=r$</td>
<td>$rq::=Aq$</td>
<td>$Aq::=r^Cq$ for all $t \in T$</td>
</tr>
<tr>
<td>$qt::=tq$</td>
<td>$tqt::=q$</td>
<td></td>
</tr>
<tr>
<td>$Z \rightarrow z$</td>
<td>$qz \rightarrow zq$</td>
<td>$Zqz \rightarrow q$ (for all $t \in T^*$)</td>
</tr>
</tbody>
</table>

$\Pi_{LR}$ essentially applies $A::=r$ in reverse direction but only if $r$ is to the left of the marker (state) $q$; the second production type of $\Pi_{LR}$ serves for shifting terminals from the right side of $q$ to its left (into the stack). $\Pi_{LR}$ accepts $z$ from left to right and the sequence of reduce steps $rq::=Aq$ in reverse order yields a Rightmost derivation $Z \rightarrow z$; $\Pi_{LR}$ is a bottom-up acceptor.

$\Pi_{LL}$, as opposed to the previous case, is a top-down acceptor. To the left of the marker (state) $q$ (in the stack) a word is derived via $Aq::=r^Cq$ ($r^C$ denotes the converse of $r$) which is compared via $tqt::=q$ with the yet uninspected terminal word to the right of $q$. $\Pi_{LL}$ accepts $z$ from left to right and the sequence of steps $Aq::=r^Cq$ directly yields a Leftmost derivation $Z \rightarrow z$.

Both types of Semi-Thue systems with set of states $Q = \{q\}$ are special cases of the general type $\Pi_{Q}$ with productions of type $\lambda qx::=\lambda q'y$ where $V$ and $Q$ are finite sets of symbols and states, respectively, $V \cap Q = \emptyset$ and $\kappa, x, \lambda, y \in V^*$, $q, q' \in Q$. For such $\Pi_{Q}$, the accepted language $L$ is defined by

$$L = \{z \in T^* : aq_oZ \rightarrow f\}$$

for some $a \in V^*$, $q_o \in Q$ and final situation $f \in V^*QV^*$. (see [4,13,15,16])

2. Algorithmic aspect.

2.1 The repetitive accepting algorithm for $L$.

All following considerations hold for general production systems $\Pi_{Q}$,