Syntactic Analysis

- Syntax of PLs is (almost) context-free.

- Semantics of a program must not be ambiguous; PLs are designed not to be ambiguous in this sense.

  The parser for a PL must cope with non-determinism in rule selection and application order, but it cannot allow global ambiguity.

- Development of deterministic parsing algorithms for context-free grammars.

LL parsing: left-to-right scan of input; leftmost derivations.

LR parsing: left-to-right scan of input; rightmost derivations.

- LL(k): an LL grammar that can be parsed deterministically using k-symbol lookahead.

LL parsing : 1. Recursive-descent 2. table-driven (predictive) parsing

- *Lookahead:* looking at next tokens before deciding on the rule application.

  in $S \Rightarrow_* uAv, u \in \Sigma^*$, x is the lookahead string in ux

  $S \overset{L}{\Rightarrow}_* uAv$ and $u \in \Sigma^*$. If u is a prefix of p, which $A \rightarrow w$ rules to apply?

  if $p = uaq, a \in \Sigma$, then 1-symbol lookahead can reveal that A rules *not starting* with a cannot lead to p.

input: $aaba$ G_1 : $S \rightarrow aA \mid abC$
$A \rightarrow aA \mid bA \mid C \mid \epsilon$
$C \rightarrow c$

- LOOKAHEAD SETS OF VARIABLES in grammar $G = (V, \Sigma, P, S)$

  LA$(A) = \{x \mid S \Rightarrow_* uAv \Rightarrow_* ux; ux \in \Sigma^*\}$

  x is a terminal string after prefix u: this gives all terminals from Av when prefix is u

- LOOKAHEAD SETS OF RULES in the grammar:

$\mathrm{LA}(A \to w) = \{x \mid wv \Rightarrow_* x \in \Sigma^*; S \Rightarrow_* uAv, u \in \Sigma^*\}$

subset of LA(A) in which subderivation $Av \Rightarrow_* x$ are done by $A \to w$

$$\mathrm{LA}(A) = \bigcup_{i=0}^{N} \mathrm{LA}(A \to w_i)$$

- if LA sets of rules are disjoint, rule selection can be done deterministically.

ex: $S \to \quad Aabd \mid cAbcd$
$A \to \quad a \mid b \mid \epsilon$

LA($S \rightarrow Aabd$) = $\{aabd, babd, abd\}$

LA($S \rightarrow cAbcd$) = $\{cabcd, cbbcd, cbcd\}$

LA(S) = union of the two LA sets

LA($A \rightarrow a$) = $\{aabd, abcd\}$

LA($A \rightarrow b$) = $\{babd, bbcd\}$

LA($A \rightarrow \epsilon$) = $\{abd, bcd\}$

- How many lookaheads do we need for this grammar?

  $\Rightarrow$ choosing A : a can pick 1st or 3rd A rule

  $\Rightarrow$ choosing A : ab can pick 1st or 3rd A rule $\Rightarrow$ LL(3)

- If there are recursive rules, LA strings can be of arbitrary length; use $trunc_k(x)$ to truncate them to length k

  $\text{LA}_k(A) = trunc_k(LA(A))$

- FIRST, FOLLOW and LA sets: predictive parsing in LL

  Direct construction of LA sets is not straightforward in a large grammar; use FIRST and FOLLOW sets.

  $\text{FIRST}_k(A) =$ prefixes of terminal strings derivable from A

FOLLOW$_k(A) =$ prefixes of terminal strings that can follow the strings derivable from A

for any string $u \in (V \cup \Sigma)^*$

FIRST$_k(u) = trunc_k(\{x \mid u \Rightarrow_* x;\ \ x \in \Sigma*\})$

FOLLOW$_k(A) = trunc_k(\{x \mid S \Rightarrow_* uAv;\ \ x \in \text{FIRST}_k(v)\}$

$\Rightarrow$ LA$_k(A) = trunc_k(\text{FIRST}_k(A)\text{FOLLOW}_k(A))$

$\Rightarrow$ LA$_k(A \to w) = trunc_k(\text{FIRST}_k(w)\text{FOLLOW}_k(A))$

ex: for G_1

$\Rightarrow$ FIRST$_3(S) = \{aab, bab, abd, cab, cbb, cbc\}$

$\Rightarrow$ FOLLOW$_3(S) = \{\epsilon\}$

- STRONG LL(k) GRAMMARS: if the LA$_k(A)$ sets are partitioned by sets LA$_K(A \rightarrow w_i)$, then G is strong LL(k).

  if G is strong LL(k), G is unambiguous.