LR parser is a simple table-driven algorithm:

LR(1) parsing: 1 token of lookahead
LR(k) parsing: k tokens of lookahead
Assume one token (LR(1))

Driver algorithm:

```
a = first token   // "virtual lookahead"
push S0 onto stack
   // "start state"
repeat forever
   s = top of stack   // "current state"
   if action[s,a] = "shift s'"
         advance lookahead
         push s' onto stack
   else if action[s,A] = "reduce A→β"
         // length of β / production in grammar
         pop |β| many states off the stack
         s" = top of stack
         push goto[s", A] onto stack
   else if action[s,a] = "accept"
         halt   // successful, complete parse of entire input
   else
         error()
```

Basic LR(1) parsing algo.

Three standard types of LR(1) parsers:

1. SLR(1) (Simple LR(1))
   - simple
   - small # of states
   - inadequate for most practical purposes
2. LR(1) ("canonical" LR(1))
   - full power of 1 lookahead token
   - adequate for essentially all practical purposes
   - large & unwieldy (huge # of states)
3. LALR(1) ("lookahead" LR(1))
   - condenses states of LR(1) together, so not so huge
   - still adequate for most purposes
   - what yacc/bison produce.

FIRST & FOLLOW sets

FIRST(β) β = string of grammar symbols

"possible" set of all first tokens of something derivable from β, or ε if $\beta \Rightarrow^* \varepsilon$

Compute FIRST(X) for every grammar symbol X (or ε):

1. FIRST(ε) = {ε}
2. FIRST(a) = {a}
   for all terminal symbols a
3. set FIRST(A) := {} initially
   repeat
      if $A \to X_1 X_2 \cdots X_k$ is a production, then
         for all i from 1 to k
            and for all terminals a,
               if a ∈ FIRST(X_i)
                  and ε ∈ FIRST(X_j)
                     for all j < i,
               then put a into FIRST(A)
         end for
         ε ∈ FIRST(X_i) for all i from 1 to k,
            put ε into FIRST(A)
   until nothing more can be added to any FIRST-set.

$X \to AbBc$; $X \to Ab \mid ab$; $A \to \varepsilon$

Note: If A→ε is a production, then ε ∈ FIRST(A)

Apr 3-11:03 AM

For any $\beta = X_1 \cdots X_k$ (grammar symbols):

FIRST(β) contains a token a if and only if there is i ($1 \le i \le k$) such that $a \in$ FIRST(X_i) and $\varepsilon \in$ FIRST(X_j) for all j with $1 \le j < i$.

$\varepsilon \in$ FIRST(β) if and only if $\varepsilon \in$ FIRST(X_i) for all i with $1 \le i \le k$.

Example:

$S \to E$

$E \to E + T \mid E - T \mid T$

$T \to T * F \mid T / F \mid F$

$F \to c \mid v \mid (E)$

X	FIRST(X)
ε	ε
c	c
v	v
(	(
)	)
+	+
$	$

X	FIRST(X)
S	c, v, (
E	c, v, (
T	c, v, (
F	c, v, (

$S \to E$

$E \to T\ T'$

$T' \to \varepsilon \mid +\ T\ T'$

$T \to F\ F'$

$F' \to \varepsilon \mid *\ F\ F'$

$F \to c \mid v \mid (E)$

X	FIRST(X)
S	c, v, (
E	c, v, (
T'	ε, +
T	c, v, (
F'	ε, *
F	c, v, (

IF A is a nonterminal, then FOLLOW(A) contains any token (or $) that could immediately follow A in some sentential form.

$S \to \cdots \Rightarrow \underline{acB+De} \to \cdots$

S start symbol

$\$ \in$ FOLLOW(S)

if $A \to \alpha B \beta$ is a production, then put all tokens in FIRST(β) into FOLLOW(B).

if $\varepsilon \in$ FIRST(β) then put all of FOLLOW(A) into FOLLOW(B)

Apply these rules repeatedly until nothing more can be added to any FOLLOW set.

Apr 3-11:52 AM