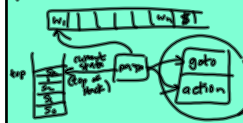


LR parser is a simple table-driven algorithm:



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



Driver algorithm:

```

a = first token // initial lookahead
push S0 onto stack
// "start state" S0

repeat forever
    s = top of stack // current state
    if action[s, a] = "shift"
        advance lookahead
        push s' onto stack
    else if action[s, a] = "reduce A -> B"
        length of B
        pop |B| 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) parser:

1. SLR(1) (Simple LR(1))
 - simple
 - small # of states
 - inadequate for most practical purposes
2. LR(1) ("canonical" LR(1))
 - Full power of LR(1) lookahead
 - adequate for essentially all practical purposes
 - large & unwieldy (hundreds 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) - parsing of grammar symbols
"possible" set of all first tokens of something derivable from β , or ϵ if $\beta \Rightarrow \epsilon$

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

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

$X \Rightarrow aBc$ $X \Rightarrow aB|aB$
 $A \Rightarrow \epsilon$
 Note: if $A \Rightarrow \epsilon$ is a production then $\epsilon \in \text{FIRST}(A)$

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

$FIRST(\beta)$ contains a token a

if and only if there is

so i ($1 \leq i \leq k$)

such that $a \in FIRST(X_i)$

and $\epsilon \in FIRST(X_j)$

for all j with $1 \leq j < i$.

$\epsilon \in FIRST(\beta)$ if and only if

$\epsilon \in FIRST(X_i)$ for all i

with $1 \leq i \leq k$.

Example:

$S \rightarrow E$

$E \rightarrow E + T \mid E - T \mid T$

$T \rightarrow T * F \mid T / F \mid F$

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

X	$FIRST(X)$	X	$FIRST(X)$
E	ϵ	S	$c, v, ($
c	c	E	$c, v, ($
v	v	T	$c, v, ($
$($	$($	F	$c, v, ($
$)$	$)$		
$+$	$+$		
$-$	$-$		
$*$	$*$		
$/$	$/$		

X	$FIRST(X)$
$S \rightarrow E$	$\epsilon, v, ($
$E \rightarrow T T'$	$\epsilon, v, ($
$T' \rightarrow \epsilon \mid + T T'$	$\epsilon, +$
$T \rightarrow F F'$	$\epsilon, v, ($
$F' \rightarrow \epsilon \mid * F F'$	$\epsilon, *$
$F \rightarrow c \mid v \mid (E)$	$c, v, ($

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

$S \rightarrow \dots \Rightarrow acB+De \rightarrow \dots$

S start symbol

$\$ \in FOLLOW(S)$

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

if $\epsilon \in FIRST(\beta)$ then
put all ϵ into $FOLLOW(B)$

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