Compiler Construction
Lent Term 2020
Lecture 15
Parsing Part II : Table-based Methods

1. Top-down parsing (LL grammars)
    1. Predictive parsing
2. Bottom-up parsing (LR grammars)
    1. SLR(1) (Simple LR)
    2. LR(1)

We will NOT cover LR(1) this year!

Timothy G. Griffin
tgg22@cam.ac.uk
Computer Laboratory
University of Cambridge

Leftmost, rightmost derivations

$$w \in T^*, \alpha, \beta \in (N \cup T)^*$$

Given : $wA\beta$ and a production $A \to \gamma$

a leftmost derivation step is written as

$$wA\beta \Rightarrow_{lm} w\gamma\beta$$

Given : αAw and a production $A \to \gamma$

a rightmost derivation step is written as

$$\alpha Aw \Rightarrow_{rm} \alpha\gamma w$$

LL(k) and LR(k)

- **LL(k)** : (**L**)eft-to-right parse, (**L**)eft-most derivation, k-symbol lookahead. Based on looking at the next k tokens, an LL(k) parser must *predict* the next production. We have been looking at LL(1).
- **LR(k)** : (**L**)eft-to-right parse, (**R**)ight-most derivation, k-symbol lookahead. Postpone production selection until *the entire* right-hand-side has been seen (and as many as k symbols beyond). LR parsers perform a rightmost derivation backwards!

For LL(1), augment Grammar with end-of-input

$$G_3' = (N_3', T_3, P_3', S)$$

$N_3' = \{\mathrm{E}, \mathrm{E}', \mathrm{T}, \mathrm{T}', \mathrm{F}, \mathrm{S}\}$ $T_3 = \{+, *, (,), \mathrm{id}, \$\}$

P_3' :

$S \rightarrow E\$$ ($ is end of input marker)

$\mathrm{E} \rightarrow \mathrm{T}\ \mathrm{E}'$

$E' \rightarrow +T\ E' \mid \varepsilon$

$\mathrm{T} \rightarrow \mathrm{F}\ \mathrm{T}'$

$T' \rightarrow\ *F\ T' \mid \varepsilon$

$\mathrm{F} \rightarrow (\mathrm{E}) \mid \mathrm{id}$

Can top-down parsing be automated?

$S \Rightarrow_{lm} E\$$ (derivation of $(x+y)$)

$\Rightarrow_{lm} TE'\$$

$\Rightarrow_{lm} FT'E'\$$

$\Rightarrow_{lm} (E)T'E'\$$

$\Rightarrow_{lm} (TE')T'E'\$$

$\Rightarrow_{lm} (FT'E')T'E'\$$

$\Rightarrow_{lm} (xT'E')T'E'\$$

$\Rightarrow_{lm} (xE')T'E'\$$

$\Rightarrow_{lm} (x+TE')T'E'\$$

$\Rightarrow_{lm} (x+FT'E')T'E'\$$

$\Rightarrow_{lm} (x+yT'E')T'E'\$$

$\Rightarrow_{lm} (x+yE')T'E'\$$

$\Rightarrow_{lm} (x+y)T'E'\$$

$\Rightarrow_{lm} (x+y)E'\$$

$\Rightarrow_{lm} (x+y)\$$

Idea : If $S \Rightarrow_{lm}^{+} w\alpha\$$ then w has been read from the input and α is on on the stack.

This looks promising. But ....

input	stack	via production
$(x+y)\$$	S	$S \to E\$$
$(x+y)\$$	$E\$$	$E \to TE'$
$(x+y)\$$	$TE'\$$	$T \to FT'$
$(x+y)\$$	$FT'E'\$$	$F \to (E)$
$(x+y)\$$	$(E)T'E'\$$	match
$x+y)\$$	$E)T'E'\$$	$E \to TE'$
$x+y)\$$	$TE')T'E'\$$	$T \to FT'$
$x+y)\$$	$FT'E')T'E'\$$	$F \to id$
$x+y)\$$	$idT'E')T'E'\$$	match
$+y)\$$	$T'E')T'E'\$$	$T' \to \varepsilon$

.... how do we automate selection of the production to use at each step?

input	stack	via production
$+y)\$$	$E')T'E'\$$	$E' \to +TE'$
$+y)\$$	$+TE')T'E'\$$	match
$y)\$$	$TE')T'E'\$$	$T \to FT'$
$y)\$$	$FT'E')T'E'\$$	$F \to id$
$y)\$$	$idT'E')T'E'\$$	match
$)\$$	$T'E')T'E'\$$	$T' \to \varepsilon$
$)\$$	$E')T'E'\$$	$E' \to \varepsilon$
$)\$$	$)T'E'\$$	match
$\$$	$T'E'\$$	$T' \to \varepsilon$
$\$$	$E'\$$	$E' \to \varepsilon$
$\$$	$\$$	accept!

FIRST (we will see how to compute later)

$$\text{FIRST}(A) = \left\{ a \in T \mid \exists \beta \in (N \cup T)^*, A \Rightarrow^+ a\beta \right\} \cup \left\{ \varepsilon \mid A \Rightarrow^+ \varepsilon \right\}$$

$S \rightarrow E\$$	$\text{FIRST}(S) = \{(, id\}$
$\text{E} \rightarrow \text{T E}'$	$\text{FIRST}(E) = \{(, id\}$
$E' \rightarrow +T\ E' \mid \varepsilon$	$\text{FIRST}(E') = \{+, \varepsilon\}$
$\text{T} \rightarrow \text{F T}'$	$\text{FIRST}(T) = \{(, id\}$
$T' \rightarrow\ *F\ T' \mid \varepsilon$	$\text{FIRST}(T') = \{*, \varepsilon\}$
$\text{F} \rightarrow (\text{E}) \mid \text{id}$	$\text{FIRST}(T) = \{(, id\}$

FOLLOW (we will see how to compute later)

$$\text{FOLLOW}(A) = \left\{ a \mid \exists \alpha\ \beta, S \Rightarrow^{+} \alpha A a \beta \right\}$$

$S \rightarrow E\$$	
$\text{E} \rightarrow \text{T E}'$	$\text{FOLLOW}(E) = \{), \$ \}$
$E' \rightarrow +T\ E' \mid \varepsilon$	$\text{FOLLOW}(E') = \{), \$ \}$
$\text{T} \rightarrow \text{F T}'$	$\text{FOLLOW}(T) = \{ +,), \$ \}$
$T' \rightarrow\ *F\ T' \mid \varepsilon$	$\text{FOLLOW}(T') = \{ +,), \$ \}$
$\text{F} \rightarrow (\text{E}) \mid \text{id}$	$\text{FOLLOW}(F) = \{ +, *,), \$ \}$

$")" \in \text{FOLLOW}(E)?$

$$S \Rightarrow E\$ \Rightarrow TE'\$ \Rightarrow FT'E'\$ \Rightarrow (E)T'E'\$$$

The LL(1) Parsing table *M*

for all $A \in N, a \in T, M[A, a] = \{\}$

for each $A \in N$

for each production $A \rightarrow \alpha$

if $a \in \text{FIRST}(\alpha)$ and $a \in T$

then $\text{M}[A, a] = M[A, a] \cup \{A \rightarrow \alpha\}$

else if $\varepsilon \in \text{FIRST}(\alpha)$

then for each $b \in \text{FOLLOW}(A)$

$\text{M}[A, b] = M[A, b] \cup \{A \rightarrow \alpha\}$

See slide 22 for definition of FIRST(α)

Table M for grammar G_3'

	id	$+$	$*$	$($	$)$	$\$$
E	$E \to TE'$			$E \to TE'$		
E'		$E' \to +TE'$			$E' \to \varepsilon$	$E' \to \varepsilon$
T	$T \to FT'$			$T \to FT'$		
T'		$T' \to \varepsilon$	$T' \to *FT'$		$T' \to \varepsilon$	$T' \to \varepsilon$
F	$F \to id$			$F \to (E)$		

The LL(1) Parsing Algorithm

$a := LexNextToken()$

$X := TopOfStack()$

while $(X \neq \$)$

if $X = a$

then pop; $a := LexNextToken()$

else if $M[X,a] = \{X \rightarrow \alpha\}$

then pop; push α (leftmost symbol on top)

$X :=$ TopOfStack ()

Now use *M* to parse *(x+y)* …

input	stack	action
$(x+y)\$$	S	$M[S,(] = \{S \to E\$\}$
$(x+y)\$$	$E\$$	$M[E,(] = \{E \to TE'\}$
$(x+y)\$$	$TE'\$$	$M[T,(] = \{T \to FT'\}$
$(x+y)\$$	$FT'E'\$$	$M[F,(] = \{F \to (E)\}$
$(x+y)\$$	$(E)T'E'\$$	*match*
$x+y)\$$	$E)T'E'\$$	$M[E,id] = \{E \to TE'\}$
$x+y)\$$	$TE')T'E'\$$	$M[T,id] = \{T \to FT'\}$
$x+y)\$$	$FT'E')T'E'\$$	$M[F,id] = \{F \to id\}$
$x+y)\$$	$idT'E')T'E'\$$	*match*
$+y)\$$	$T'E')T'E'\$$	$M[T',+] = \{T' \to \varepsilon\}$

… kachunk, kachunk, kachunk …

input	stack	action
$+\,y)\$$	$E')T'E'\$$	$M[E',+]=\{E'\rightarrow +TE'\}$
$+\,y)\$$	$+TE')T'E'\$$	*match*
$y)\$$	$TE')T'E'\$$	$M[T,id]=\{T\rightarrow FT'\}$
$y)\$$	$FT'E')T'E'\$$	$M[F,id]=\{F\rightarrow id\}$
$y)\$$	$idT'E')T'E'\$$	*match*
$)\$$	$T'E')T'E'\$$	$M[T',)]=\{T'\rightarrow \varepsilon\}$
$)\$$	$E')T'E'\$$	$M[E',)]=\{E'\rightarrow \varepsilon\}$
$)\$$	$)T'E'\$$	*match*
$\$$	$T'E'\$$	$M[T',\$]=\{T'\rightarrow \varepsilon\}$
$\$$	$E'\$$	$M[E',\$]=\{E'\rightarrow \varepsilon\}$
$\$$	$\$$	*accept*

Computing FIRST: Easy when there are no epsilon productions!

for all $a \in T$, FIRST(a) := {a}

for all $A \in N$, FIRST(A) := {}

while FIRST changes

if $A \rightarrow X\alpha$ is a production $(X \in N \cup T)$

then FIRST(A) := FIRST(A) $\cup$ FIRST(X)

Computing FOLLOW: Easy when there are no epsilon productions!

for all $A \in N$, FOLLOW$(A) := \{\}$

FOLLOW(S) $:= \{\$\}$ (S is the start symbol)

while FOLLOW changes

if $A \rightarrow \alpha B \beta$ is a production $(B \in N, \beta \neq \varepsilon)$

then FOLLOW(B) := FOLLOW(B) $\cup$ FIRST(β)

if $A \rightarrow \alpha B$ is a production $(B \in N)$

then FOLLOW(B) := FOLLOW(B) $\cup$ FOLLOW(A)

With epsilon productions NULLABLE is a useful function

$$\text{NULLABLE} : (N \cup T)^* \rightarrow \{true, false\}$$

$$\text{NULLABLE}(\varepsilon) = true$$

$$\text{NULLABLE}(a\beta) = false$$

$$\text{NULLABLE}(A\beta) = (A \Rightarrow^* \varepsilon) \wedge \text{NULLABLE}(\beta)$$

Computing FIRST with epsilon productions!

for all $a \in T$, FIRST(a) := $\{a\}$

for all $A \in N$, FIRST(A) := $\{\}$

while FIRST changes

  if $A \rightarrow \varepsilon$ is a production

  then FIRST(A) := FIRST(A) $\cup \{\varepsilon\}$

  if $A \rightarrow X_1 X_2 \cdots X_k$ is a production

  then $j = 1$; done := false

    while not done and $j \leq k$

      FIRST(A) := FIRST(A) $\cup$ (FIRST(X_j) - $\{\varepsilon\}$)

      if NULLABLE(X_j)

      then $j := j + 1$

      else done := *true*

    if $j = k + 1$ then FIRST(A) := FIRST(A) $\cup \{\varepsilon\}$

Computing FOLLOW with epsilon productions!

for all $A \in N$, FOLLOW$(A) := \{\}$

FOLLOW(S) $:= \{\$\}$ (S is the start symbol)

while FOLLOW changes

  if $A \rightarrow \alpha B \beta$ is a production $(B \in N, \beta \neq \varepsilon)$

  then FOLLOW(B) := FOLLOW(B) $\cup$ (FIRST(β) - $\{\varepsilon\}$)

  if $A \rightarrow \alpha B \beta$ is a production and $\varepsilon \in$ FIRST(β)

  then FOLLOW(B) := FOLLOW(B) $\cup$ FOLLOW(A)

  if $A \rightarrow \alpha B$ is a production $(B \in N)$

  then FOLLOW(B) := FOLLOW(B) $\cup$ FOLLOW(A)

Many grammars cannot be parsed LL(1)

$S \rightarrow d \mid XYS$

$Y \rightarrow c \mid \varepsilon$

$X \rightarrow Y \mid a$

	FIRST	FOLLOW
S	$\{a, c, d\}$	$\{\}$
Y	$\{c\}$	$\{a, c, d\}$
X	$\{a, c\}$	$\{a, c, d\}$

$M[S, d] = \{ S \rightarrow d, S \rightarrow XYS \}$

This is ambiguity!

Grammar is not LL(1)!

Bottom-up (LR) parsing to the rescue!

$$G_2 = (N_2, T_1, P_2, E)$$

$$N_2 = \{\mathrm{E}, \mathrm{T}, \mathrm{F}\} \qquad T_1 = \{+, *, (,), \mathrm{id}\}$$

$$\mathrm{E} \to \mathrm{E} + \mathrm{T} \mid \mathrm{T} \qquad \mathrm{T} \to \mathrm{T} * \mathrm{F} \mid \mathrm{F} \qquad \mathrm{F} \to (\mathrm{E}) \mid \mathrm{id}$$

With LR parsing we no longer have to eliminate left recursion from the grammar!

Define $\mathrm{FIRST}(\alpha)$ for $\alpha \in (\mathrm{N} \cup \mathrm{T})^*$

$\mathrm{FIRST}'(\varepsilon) = \{\varepsilon\}$

$\mathrm{FIRST}'(X) = \mathrm{FIRST}(X) \qquad (X \in N \cup T)$

$\mathrm{FIRST}'(a\beta) = \{a\} \qquad (a \in T)$

$\mathrm{FIRST}'(A\beta) = \qquad (A \in N)$

if NULLABLE(A)

then $\mathrm{FIRST}'(\beta) \cup (\mathrm{FIRST(A)} - \{\varepsilon\})$

else FIRST(A)

Now, abuse notation and just write

$\mathrm{FIRST}(\alpha)$ instead of $\mathrm{FIRST}'(\alpha)$