Compiler Construction
Lent Term 2021
Lecture 4: Table-driven top-down (LL) parsing

1. LL(k) vs LR(k) parsing
2. Automating left-most derivations?
3. FIRST, FOLLOW, and the LL(1) parsing table.
4. LL(1) table-based parsing
5. Computing FIRST and FOLLOW

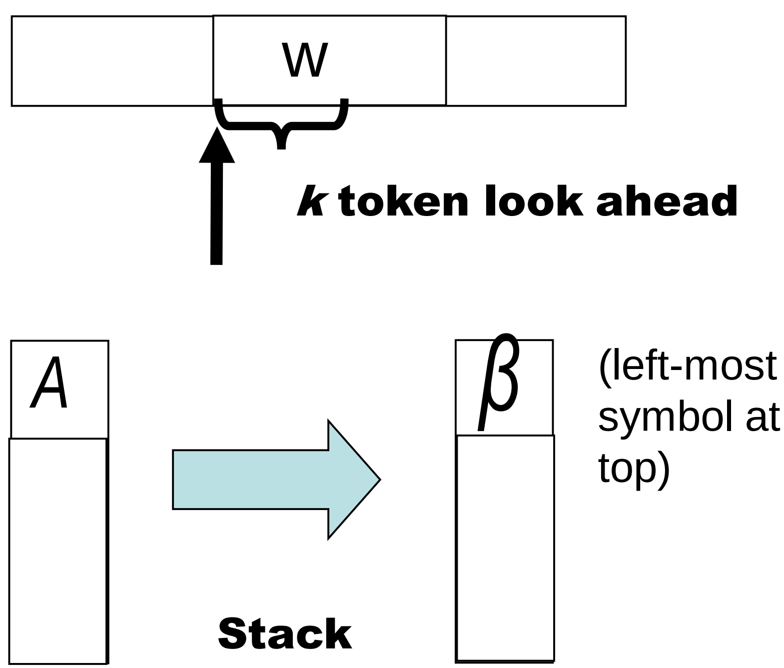
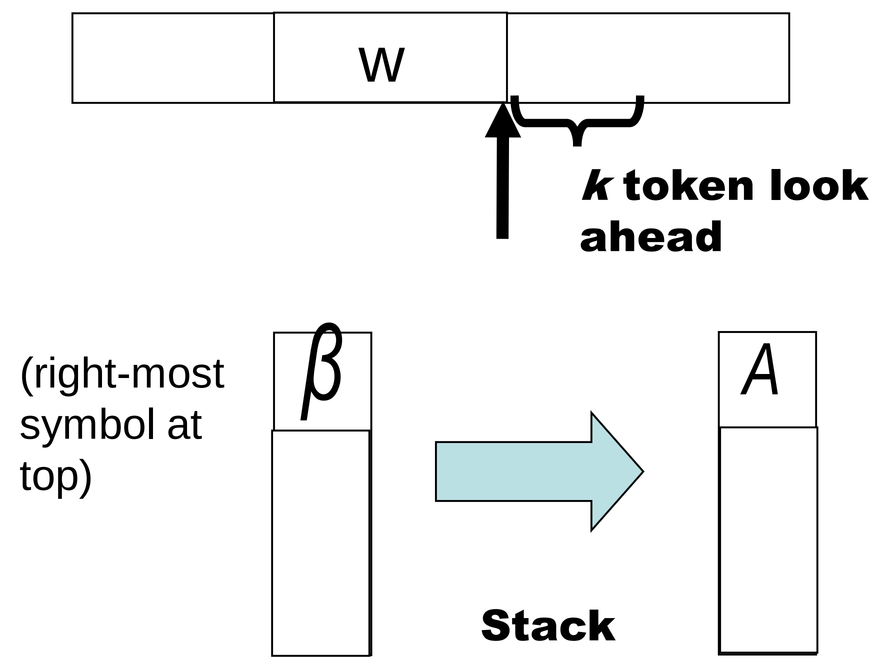
Timothy G. Griffin
tgg22@cam.ac.uk
Computer Laboratory
University of Cambridge

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 <u>backwards</u>!

LL(k) vs. LR(k) reductions (SLR(1) as well)

$$A \rightarrow \beta \Rightarrow^{+} w \quad \beta \in (T \cup N)^{*} \quad w \in T^{*}$$

$LL(k)$	$LR(k)$
	

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

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

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

P_3' :

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

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

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

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

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

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

Leftmost derivations

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

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

a leftmost derivation step is written as

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

A left-most derivation of *(x+y)*

$$
\begin{aligned}
S &\Rightarrow_{lm} E\$ \\
&\Rightarrow_{lm} TE'\$ \\
&\Rightarrow_{lm} FT'E'\$ \\
&\Rightarrow_{lm} (E)T'E'\$ \\
&\Rightarrow_{lm} (TE')T'E'\$ \\
&\Rightarrow_{lm} (FT'E')T'E'\$ \\
&\Rightarrow_{lm} (xT'E')T'E'\$ \\
&\Rightarrow_{lm} (xE')T'E'\$ \\
&\Rightarrow_{lm} (x+TE')T'E'\$ \\
&\Rightarrow_{lm} (x+FT'E')T'E'\$ \\
&\Rightarrow_{lm} (x+yT'E')T'E'\$ \\
&\Rightarrow_{lm} (x+yE')T'E'\$ \\
&\Rightarrow_{lm} (x+y)T'E'\$ \\
&\Rightarrow_{lm} (x+y)E'\$ \\
&\Rightarrow_{lm} (x+y)\$
\end{aligned}
$$

Idea: Can we turn left-most derivations into a stack machine (a PDA)? Perhaps this will work: 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 can we make it work?

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
$+y)\$$	$T'E')T'E'\$$	$T' \to \varepsilon$

But 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)

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

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

FOLLOW (we will see how to compute later)

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

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

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

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

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 $A \to \alpha$

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

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

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

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

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

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 $X = a$ (* a match *)

  then pop; $a := LexNextToken()$

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

    then pop; push α (leftmost symbol on top)

  $X :=$ TopOfStack()

Now use *M* to parse *(x+y)* …

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

... kachunk, kachunk, kachunk ...

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

NULLABLE

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

$$\text{if and only if } \alpha \Rightarrow^{*} \varepsilon.$$

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

$$\text{NULLABLE}(c) = false \qquad (c \in T)$$

$$\text{NULLABLE}(A) = \qquad (A \in N)$$

$$\bigvee_{A \to \alpha} \text{NULLABLE}(\alpha)$$

$$\text{NULLABLE}(X\beta) = \qquad (X \in T \cup N)$$

$$\text{NULLABLE}(X) \wedge \text{NULLABLE}(\beta)$$

Computing FIRST

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

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

while FIRST changes

  if $A \rightarrow \varepsilon$ is a production

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

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

  then $\mathrm{j} = 1$; done := false

    while not done and $\mathrm{j} \leq \mathrm{k}$

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

      if $\text{NULLABLE}(\mathrm{X_j})$

      then $\mathrm{j} := \mathrm{j} + 1$

      else done := *true*

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

Computing FOLLOW

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)

$$S \to d \mid XYS$$
$$Y \to c \mid \varepsilon$$
$$X \to Y \mid a$$

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

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

This is ambiguity!

Grammar is not LL(1)!

Bottom-up (LR) parsing to the rescue!

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

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

$\text{E} \rightarrow \text{E} + \text{T} \mid \text{T} \qquad \text{T} \rightarrow \text{T} * \text{F} \mid \text{F} \qquad \text{F} \rightarrow (\text{E}) \mid \text{id}$

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