Syntax Analysis
Part I

Chapter 4

Left Factoring

- When a nonterminal has two or more productions whose right-hand sides start with the same grammar symbols, the grammar is not LL(1) and cannot be used for predictive parsing
- Replace productions

  $$A \to \alpha\ \beta_1 \mid \alpha\ \beta_2 \mid \ldots \mid \alpha\ \beta_n \mid \gamma$$

  with

  $$A \to \alpha\ A_R \mid \gamma$$
  $$A_R \to \beta_1 \mid \beta_2 \mid \ldots \mid \beta_n$$

Predictive Parsing

- Eliminate left recursion from grammar
- Left factor the grammar
- Compute FIRST and FOLLOW
- Two variants:
  - Recursive (recursive-descent parsing)
  - Non-recursive (table-driven parsing)

FIRST (Revisited)

- FIRST(α) = { *the set of terminals that begin all strings derived from* α }
  FIRST(a) = {a} if $a \in T$
  FIRST(ε) = {ε}
  FIRST(A) = $\cup_{A\to\alpha}$ FIRST(α) for $A\to\alpha \in P$
  FIRST($X_1X_2\ldots X_k$) =

- **if** for all j = 1, …, i-1 : $\varepsilon \in$ FIRST(X_j) **then**
  add non-ε in FIRST(X_i) to FIRST($X_1X_2\ldots X_k$)

- **if** for all j = 1, …, k : $\varepsilon \in$ FIRST(X_j) **then**
  add ε to FIRST($X_1X_2\ldots X_k$)

Calculate FIRST

- $E \rightarrow TE_R$
  $E_R \rightarrow +TE_R \mid \varepsilon$
  $T \rightarrow FT_R$
  $T_R \rightarrow *FT_R \mid \varepsilon$
  $F \rightarrow (E) \mid id$

Calculate FIRST

- S→ ACB | CbB | Ba
  A → da | BC
  B → g | ε
  C → h | ε

Calculate FIRST

- S→ aBDh
- B → cC
- C → bC | ε
- D → EF
- E → g | ε
- F → f | ε

FOLLOW

- FOLLOW(A) = { *the set of terminals that can immediately follow nonterminal A* }

  FOLLOW(A) =
  for all $(B \to \alpha\ A\ \beta) \in P$ **do**
  add FIRST(β)\{ε} to FOLLOW(A)

- **for** all $(B \to \alpha\ A\ \beta) \in P$ and $\varepsilon \in$ FIRST(β) **do**
  add FOLLOW(B) to FOLLOW(A)

- **for** all $(B \to \alpha\ A) \in P$ **or** $(B \to \alpha\ A\ \beta)$ when β produces null
  do add FOLLOW(B) to FOLLOW(A)

  if A is the start symbol S **then**
  add **$** to FOLLOW($A$)

Calculate Follow

- $E \rightarrow TE_R$

  $E_R \rightarrow +TE_R \mid \varepsilon$

  $T \rightarrow FT_R$

  $T_R \rightarrow *FT_R \mid \varepsilon$

  $F \rightarrow (E) \mid id$

Calculate Follow

- S→ ACB | CbB | Ba

  A → da | BC

  B → g | ε

  C → h | ε

Calculate Follow

- S→ aBDh
- B → cC
- C → bC | ε
- D → EF
- E → g | ε
- F → f | ε

LL(1) Grammar

- A grammar G is LL(1) if it is not left recursive and for each collection of productions

  $A \to \alpha_1 \mid \alpha_2 \mid \ldots \mid \alpha_n$

  for nonterminal A the following holds:

  1. FIRST(α_i) $\cap$ FIRST(α_j) = $\varnothing$ for all $i \neq j$
  2. if $\alpha_i \Rightarrow^* \varepsilon$ then

     2.a. $\alpha_j \not\Rightarrow^* \varepsilon$ for all $i \neq j$

     2.b. FIRST(α_j) $\cap$ FOLLOW(A) = $\varnothing$
     for all $i \neq j$

Non-LL(1) Examples

Grammar	*Not LL(1) because:*
$S \rightarrow S\ \mathbf{a} \mid \mathbf{a}$	Left recursive
$S \rightarrow \mathbf{a}\ S \mid \mathbf{a}$	FIRST($\mathbf{a}\ S$) $\cap$ FIRST($\mathbf{a}$) $\neq \emptyset$
$S \rightarrow \mathbf{a}\ R \mid \varepsilon$ $R \rightarrow S \mid \varepsilon$	For *R*: $S \Rightarrow^* \varepsilon$ and $\mathrm{R} \Rightarrow^* \varepsilon$
$S \rightarrow \mathbf{a}\ R\ \mathbf{a}$ $R \rightarrow S \mid \varepsilon$	For *R*: FIRST(S) $\cap$ FOLLOW(R) $\neq \emptyset$

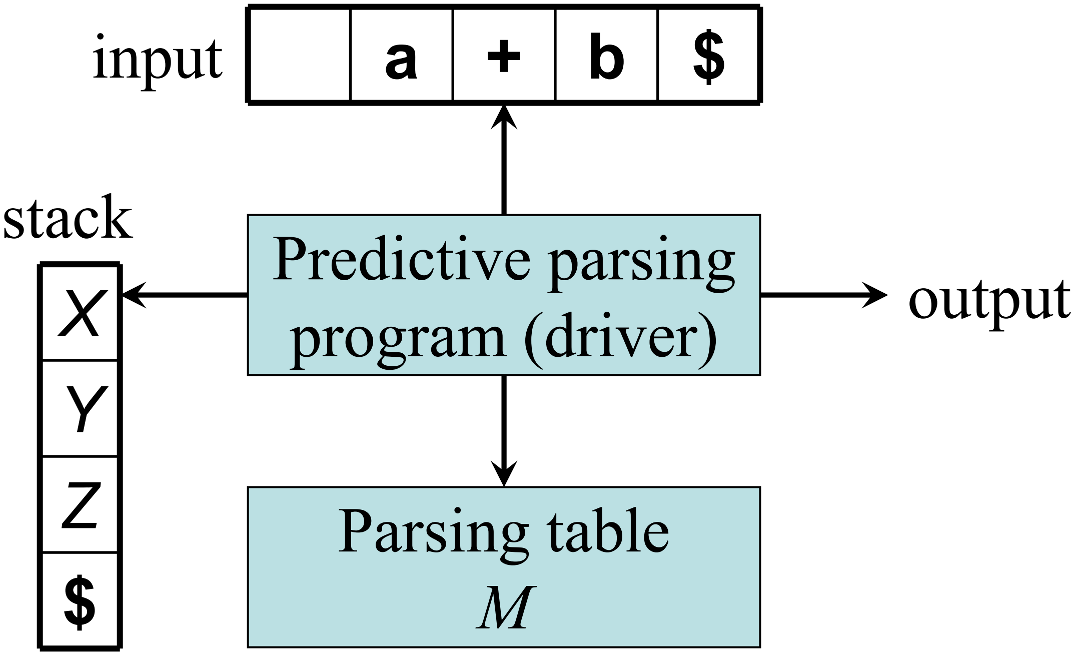
Recursive-Descent Parsing (Recap)

- Grammar must be LL(1)
- Every nonterminal has one (recursive) procedure responsible for parsing the nonterminal’s syntactic category of input tokens

- When a nonterminal has multiple productions, each production is implemented in a branch of a selection statement based on input look-ahead information

Non-Recursive Predictive Parsing: Table-Driven Parsing

- Given an LL(1) grammar *G* = (*N*, *T*, *P*, *S*) construct a table *M*[*A*,*a*] for *A* ∈ *N*, *a* ∈ *T* and use a *driver program* with a *stack*



Constructing an LL(1) Predictive Parsing Table

```
for each production $A \rightarrow \alpha$ do
    for each $a \in$ FIRST($\alpha$) do
        add $A \rightarrow \alpha$ to $M[A,a]$
    enddo
    if $\varepsilon \in$ FIRST($\alpha$) then
        for each $b \in$ FOLLOW($A$) do
            add $A \rightarrow \alpha$ to $M[A,b]$
        enddo
    endif
enddo
Mark each undefined entry in $M$ error
```

Example Table

$$
\begin{aligned}
&E \rightarrow T\,E_R \\
&E_R \rightarrow + \,T\,E_R \mid \varepsilon \\
&T \rightarrow F\,T_R \\
&T_R \rightarrow \mathbf{*}\,F\,T_R \mid \varepsilon \\
&F \rightarrow \mathbf{(}\,E\,\mathbf{)} \mid \mathbf{id}
\end{aligned}
$$

$A \rightarrow \alpha$	FIRST(α)	FOLLOW(A)
$E \rightarrow T\,E_R$	**(id**	**$)**
$E_R \rightarrow$ **+** $T\,E_R$	**+**	**$)**
$E_R \rightarrow \varepsilon$	ε	**$)**
$T \rightarrow F\,T_R$	**(id**	**+ $)**
$T_R \rightarrow$ ***** $F\,T_R$	*****	**+ $)**
$T_R \rightarrow \varepsilon$	ε	**+ $)**
$F \rightarrow$ **(** E **)**	**(**	*** + $)**
$F \rightarrow$ **id**	**id**	*** + $)**

	id	**+**	*****	**(**	**)**	**$**
E	$E \rightarrow T\,E_R$			$E \rightarrow T\,E_R$		
E_R		$E_R \rightarrow$ **+** $T\,E_R$			$E_R \rightarrow \varepsilon$	$E_R \rightarrow \varepsilon$
T	$T \rightarrow F\,T_R$			$T \rightarrow F\,T_R$		
T_R		$T_R \rightarrow \varepsilon$	$T_R \rightarrow$ ***** $F\,T_R$		$T_R \rightarrow \varepsilon$	$T_R \rightarrow \varepsilon$
F	$F \rightarrow$ **id**			$F \rightarrow$ **(** E **)**		

LL(1) Grammars are Unambiguous

Ambiguous grammar

$S \rightarrow \mathbf{i}\ E\ \mathbf{t}\ S\ S_R \mid \mathbf{a}$

$S_R \rightarrow \mathbf{e}\ S \mid \varepsilon$

$E \rightarrow \mathbf{b}$

$A \rightarrow \alpha$	FIRST(α)	FOLLOW(A)
$S \rightarrow \mathbf{i}\ E\ \mathbf{t}\ S\ S_R$	**i**	**e $**
$S \rightarrow \mathbf{a}$	**a**	**e $**
$S_R \rightarrow \mathbf{e}\ S$	**e**	**e $**
$S_R \rightarrow \varepsilon$	ε	**e $**
$E \rightarrow \mathbf{b}$	**b**	**t**

Error: duplicate table entry

	a	**b**	**e**	**i**	**t**	**$**
S	$S \rightarrow \mathbf{a}$			$S \rightarrow \mathbf{i}\ E\ \mathbf{t}\ S\ S_R$		
S_R			$S_R \rightarrow \varepsilon$ $S_R \rightarrow \mathbf{e}\ S$			$S_R \rightarrow \varepsilon$
E		$E \rightarrow \mathbf{b}$				

Example Table-Driven Parsing

Stack	Input	Production applied
$\$\underline{E}$	$\underline{\mathbf{id}}\mathbf{+id*id\$}$	$E \rightarrow T\ E_R$
$\$E_R\underline{T}$	$\underline{\mathbf{id}}\mathbf{+id*id\$}$	$T \rightarrow F\ T_R$
$\$E_RT_R\underline{F}$	$\underline{\mathbf{id}}\mathbf{+id*id\$}$	$F \rightarrow \mathbf{id}$
$\$E_RT_R\underline{\mathbf{id}}$	$\underline{\mathbf{id}}\mathbf{+id*id\$}$	
$\$E_R\underline{T}_R$	$\underline{\mathbf{+}}\mathbf{id*id\$}$	$T_R \rightarrow \varepsilon$
$\$\underline{E}_R$	$\underline{\mathbf{+}}\mathbf{id*id\$}$	$E_R \rightarrow +\ T\ E_R$
$\$E_RT\underline{+}$	$\underline{\mathbf{+}}\mathbf{id*id\$}$	
$\$E_R\underline{T}$	$\underline{\mathbf{id}}\mathbf{*id\$}$	$T \rightarrow F\ T_R$
$\$E_RT_R\underline{F}$	$\underline{\mathbf{id}}\mathbf{*id\$}$	$F \rightarrow \mathbf{id}$
$\$E_RT_R\underline{\mathbf{id}}$	$\underline{\mathbf{id}}\mathbf{*id\$}$	
$\$E_R\underline{T}_R$	$\underline{\mathbf{*}}\mathbf{id\$}$	$T_R \rightarrow *\ F\ T_R$
$\$E_RT_RF\underline{*}$	$\underline{\mathbf{*}}\mathbf{id\$}$	
$\$E_RT_R\underline{F}$	$\underline{\mathbf{id}}\mathbf{\$}$	$F \rightarrow \mathbf{id}$
$\$E_RT_R\underline{\mathbf{id}}$	$\underline{\mathbf{id}}\mathbf{\$}$	
$\$E_R\underline{T}_R$	$\underline{\mathbf{\$}}$	$T_R \rightarrow \varepsilon$
$\$\underline{E}_R$	$\underline{\mathbf{\$}}$	$E_R \rightarrow \varepsilon$
$\underline{\$}$	$\underline{\mathbf{\$}}$	

Predictive Parsing Program (Driver)

```
push(**$**)
push($S$)
$a$ := *lookahead*
**repeat**
        $X$ := pop()
        **if** $X$ is a terminal or $X$ = **$** **then**
                match($X$) // *moves to next token and a := lookahead*
        **else if** $M[X,a] = X \to Y_1Y_2\ldots Y_k$ **then**
                push($Y_k$, $Y_{k\text{-}1}$, …, $Y_2$, $Y_1$) // *such that* $Y_1$ *is on top*
                ... invoke actions and/or produce IR output …
        **else**    error()
        **endif**
**until** $X$ = **$**
```