

CS536

Building a Predictive Parser

Last Time: Intro LL(1) Predictive Parser

- “predict” the parse tree top-down
- Parser structure
 - 1 token of lookahead
 - A stack tracking parse tree frontier
 - Selector/parse table
- Necessary conditions
 - Left-factored
 - Free of left-recursion



Today: Building the Parse Table

- Review Grammar transformations
 - Why they are necessary
 - How they work
- Build the selector table
 - $\text{FIRST}(X)$: Set of terminals that can begin at a subtree rooted at X
 - $\text{FOLLOW}(X)$: Set of terminals that can appear after X

Review LL(1) Grammar Transformations

- Necessary (but not sufficient conditions) for LL(1) Parsing:
 - Left factored
 - No rules with common prefix
 - Why? We'd need to look past the prefix to pick rule
 - Free of left recursion
 - No nonterminal loops for a production
 - Why? Need to look past list to know when to cap it

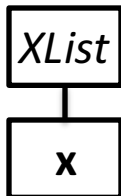
Why Left Recursion is a Problem (Blackbox View)

CFG snippet: $XList \rightarrow XList\ x \mid x$

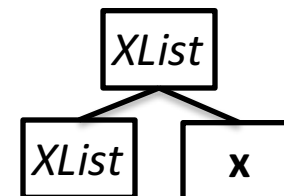
Current parse tree: *XList*

Current token: **x**

How should we grow the tree top-down?



(OR)



Correct if there are no more **xs**

Correct if there are more **xs**

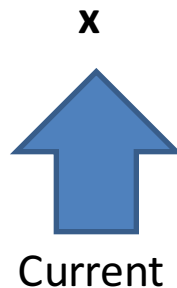
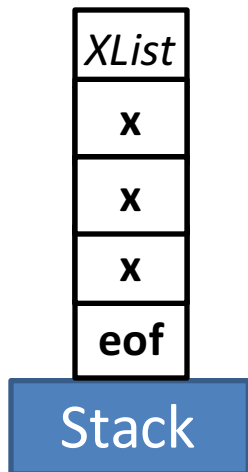
We don't know which without more lookahead

Why Left Recursion is a Problem (Whitebox View)

CFG snippet: $XList \rightarrow XList\ x \mid \epsilon$

Current parse tree: $XList$ x **eof** Current token: x

Parse table: $XList$ $XList\ x$ ϵ



(Stack overflow)

Left Recursion Elimination: Review

Replace

$$A \rightarrow A \alpha \mid \beta$$

With

$$A \rightarrow \beta A'$$

$$A' \rightarrow \alpha A' \mid \varepsilon$$

Head of the list

Where β does not start with A and may not be present

Preserve order (a list of α starting with β) but use right recursion

Left Recursion Elimination: Ex1

$$A \rightarrow A \alpha \mid \beta \quad \Rightarrow \quad \begin{array}{l} A \rightarrow \beta A' \\ A' \rightarrow \alpha A' \mid \varepsilon \end{array}$$

$$\begin{array}{l} E \rightarrow E \text{ cross id} \mid \text{id} \\ \quad \underbrace{\hspace{1.5cm}}_{\alpha} \quad \underbrace{\hspace{1cm}}_{\beta} \end{array} \quad \Rightarrow \quad \begin{array}{l} E \rightarrow \text{id } E' \\ E' \rightarrow \underbrace{\text{cross id}}_{\alpha} E' \mid \varepsilon \\ \quad \underbrace{\hspace{1.5cm}}_{\beta} \end{array}$$

Left Recursion Elimination: Ex2

$$A \rightarrow A \alpha \mid \beta \quad \Rightarrow \quad \begin{array}{l} A \rightarrow \beta A' \\ A' \rightarrow \alpha A' \mid \varepsilon \end{array}$$

$$\begin{array}{l} E \rightarrow E + T \mid T \\ T \rightarrow T * F \mid F \\ F \rightarrow (E) \mid \text{id} \end{array} \quad \Rightarrow \quad \begin{array}{l} E \rightarrow TE' \\ E' \rightarrow + TE' \mid \varepsilon \\ T \rightarrow FT' \\ T' \rightarrow * FT' \mid \varepsilon \\ F \rightarrow (E) \mid \text{id} \end{array}$$

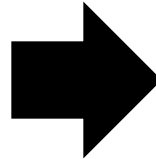
Left Recursion Elimination: Ex3

$$A \rightarrow A \alpha \mid \beta \quad \Rightarrow \quad \begin{array}{l} A \rightarrow \beta A' \\ A' \rightarrow \alpha A' \mid \varepsilon \end{array}$$

$SList \rightarrow SList D \mid \varepsilon$

$D \rightarrow Type \text{ id semi}$

$Type \rightarrow \text{bool} \mid \text{int}$

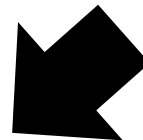


$SList \rightarrow \varepsilon SList'$

$SList' \rightarrow D SList' \mid \varepsilon$

$D \rightarrow Type \text{ id semi}$

$Type \rightarrow \text{bool} \mid \text{int}$



$SList \rightarrow D SList \mid \varepsilon$

$D \rightarrow Type \text{ id semi}$

$Type \rightarrow \text{bool} \mid \text{int}$

Left Factoring: Review

Removing common prefix from grammar

Replace $A \rightarrow \alpha \beta_1 \mid \dots \mid \alpha \beta_m \mid y_1 \mid \dots \mid y_n$

With $A \rightarrow \alpha A' \mid y_1 \mid \dots \mid y_n$
 $A' \rightarrow \beta_1 \mid \dots \mid \beta_m$

Where β_i and y_i are sequence of symbols with no common prefix
 y_i May not be present, one of the β may be ϵ

Squash all “problem” rules starting with α together into one rule $\alpha A'$
Now A' represents the suffix of the “problem” rules

Left Factoring: Example 1

$$A \rightarrow \alpha \beta_1 \mid \dots \mid \alpha \beta_m \mid \gamma_1 \mid \dots \mid \gamma_n \quad \Rightarrow \quad \begin{array}{l} A \rightarrow \alpha A' \mid \gamma_1 \mid \dots \mid \gamma_n \\ A' \rightarrow \beta_1 \mid \dots \mid \beta_m \end{array}$$

$$X \rightarrow \overset{\alpha}{\underbrace{\hspace{1cm}}} \overset{\beta_1}{\underbrace{\hspace{1cm}}} \mid \overset{\alpha}{\underbrace{\hspace{1cm}}} \overset{\beta_2}{\underbrace{\hspace{1cm}}} \mid \overset{\alpha}{\underbrace{\hspace{1cm}}} \overset{\beta_3}{\underbrace{\hspace{1cm}}} \mid \overset{\gamma_1}{\underbrace{\hspace{1cm}}} d$$

$\langle a \rangle \mid \langle b \rangle \mid \langle c \rangle \mid d$

$$X \rightarrow \overset{\alpha}{\underbrace{\hspace{1cm}}} \overset{\gamma_1}{\underbrace{\hspace{1cm}}} \mid d$$

$$X' \rightarrow \underbrace{\hspace{1cm}}_{\beta_1} \mid \underbrace{\hspace{1cm}}_{\beta_2} \mid \underbrace{\hspace{1cm}}_{\beta_3}$$

$a \rangle \mid b \rangle \mid c \rangle$

Left Factoring: Example 2

$$A \rightarrow \alpha \beta_1 \mid \dots \mid \alpha \beta_m \mid \gamma_1 \mid \dots \mid \gamma_n \quad \Rightarrow \quad \begin{array}{l} A \rightarrow \alpha A' \mid \gamma_1 \mid \dots \mid \gamma_n \\ A' \rightarrow \beta_1 \mid \dots \mid \beta_m \end{array}$$

β_1 β_2

$Stmt \rightarrow \text{id assign } E \mid \text{id } (EList) \mid \text{return}$

$E \rightarrow \text{intlit} \mid \text{id}$

$EList \rightarrow E \mid E \text{ comma } EList$

$Stmt \rightarrow \text{id } Stmt' \mid \text{return}$


$Stmt' \rightarrow \text{assign } E \mid (EList)$

$E \rightarrow \text{intlit} \mid \text{id}$

$EList \rightarrow E \mid E \text{ comma } EList$

Left Factoring: Example 3

$$A \rightarrow \alpha \beta_1 \mid \dots \mid \alpha \beta_m \mid \gamma_1 \mid \dots \mid \gamma_n \quad \Rightarrow \quad \begin{aligned} A &\rightarrow \alpha A' \mid \gamma_1 \mid \dots \mid \gamma_n \\ A' &\rightarrow \beta_1 \mid \dots \mid \beta_m \end{aligned}$$

α $B_1 = \epsilon$ α β_2

 $S \rightarrow \text{if } E \text{ then } S \mid \text{if } E \text{ then } S \text{ else } S \mid \text{semi}$
 $E \rightarrow \text{boollit}$

$S \rightarrow \text{if } E \text{ then } S S' \mid \text{semi}$

$S' \rightarrow \text{else } S \mid \epsilon$

$E \rightarrow \text{boollit}$

Left Factoring: Not Always Immediate

$$A \rightarrow \alpha \beta_1 \mid \dots \mid \alpha \beta_m \mid \gamma_1 \mid \dots \mid \gamma_n \quad \Rightarrow \quad \begin{array}{l} A \rightarrow \alpha A' \mid \gamma_1 \mid \dots \mid \gamma_n \\ A' \rightarrow \beta_1 \mid \dots \mid \beta_m \end{array}$$

This snippet yearns for left-factoring

$S \rightarrow A \mid C \mid \text{return}$

$A \rightarrow \text{id assign } E$

$C \rightarrow \text{id } (\text{EList})$

but we cannot! At least without *inlining*

$S \rightarrow \text{id assign } E \mid \text{id } (\text{EList}) \mid \text{return}$

Let's be more constructive

- So far, we've only talked about what precludes us from building a predictive parser
- It's time to actually build the parse table

Building the Parse Table

- What do we actually need to ensure arbitrary production $A \rightarrow \alpha$ is the correct one to apply?
(assume α is an arbitrary symbol string)
 1. What terminals could possibly start α (we call this the FIRST set)
 2. What terminal could possibly come after A (we call this the FOLLOW set)

FIRST Sets

- $\text{FIRST}(\alpha)$ is the set of terminals that begin the strings derivable from α , and also, if α can derive ε , then ε is in $\text{FIRST}(\alpha)$.
- Formally, $\text{FIRST}(\alpha) =$

$$\left\{ t \mid \left(t \in \Sigma \wedge \alpha \xRightarrow{*} t\beta \right) \vee \left(t = \varepsilon \wedge \alpha \xRightarrow{*} \varepsilon \right) \right\}$$

Why is FIRST Important?

- Assume the top-of-stack symbol is A and current token is a
 - Production 1: $A \rightarrow \alpha$
 - Production 2: $A \rightarrow \beta$
- FIRST let us disambiguate:
 - If a is in $\text{FIRST}(\alpha)$, it tells us that Production 1 is a viable choice
 - If a is in $\text{FIRST}(\beta)$, it tells us that Production 2 is a viable choice
 - If a is in only $\text{FIRST}(\alpha)$ xor $\text{FIRST}(\beta)$, we can predict the rule we need.

FIRST Construction: Single Symbol

- We begin by doing FIRST sets for a single, arbitrary symbol X
 - If X is a terminal: $\text{FIRST}(X) = \{ X \}$
 - If X is ϵ : $\text{FIRST}(\epsilon) = \{ \epsilon \}$
 - If X is a nonterminal, for each $X \rightarrow Y_1 Y_2 \dots Y_k$
 - Put $\text{FIRST}(Y_1) - \{\epsilon\}$ into $\text{FIRST}(X)$
 - If ϵ is in $\text{FIRST}(Y_1)$, put $\text{FIRST}(Y_2) - \{\epsilon\}$ into $\text{FIRST}(X)$
 - If ϵ is also in $\text{FIRST}(Y_2)$, put $\text{FIRST}(Y_3) - \{\epsilon\}$ into $\text{FIRST}(X)$
 - ...
 - If ϵ is in FIRST of all Y_i symbols, put ϵ into $\text{FIRST}(X)$

FIRST(X) Example

Building FIRST(X) for nonterm X

for each $X \rightarrow Y_1 Y_2 \dots Y_k$

- Add $\text{FIRST}(Y_1) - \{\epsilon\}$
- If ϵ is in $\text{FIRST}(Y_{1 \text{ to } i-1})$: add $\text{FIRST}(Y_i) - \{\epsilon\}$
- If ϵ is in all RHS symbols, add ϵ

$Exp \rightarrow Term\ Exp'$

$Exp' \rightarrow \text{minus}\ Term\ Exp' \mid \epsilon$

$Term \rightarrow Factor\ Term'$

$Term' \rightarrow \text{divide}\ Factor\ Term' \mid \epsilon$

$Factor \rightarrow \text{intlit} \mid \text{lparens}\ Exp\ \text{rparens}$

$\text{FIRST}(Factor) = \{ \text{intlit}, \text{lparens} \}$

$\text{FIRST}(Term') = \{ \text{divide}, \epsilon \}$

$\text{FIRST}(Term) = \{ \text{intlit}, \text{lparens} \}$

$\text{FIRST}(Exp') = \{ \text{minus}, \epsilon \}$

$\text{FIRST}(Exp) = \{ \text{intlit}, \text{lparens} \}$

FIRST(α)

- We now extend FIRST to strings of symbols α
 - We want to define FIRST for all RHS
- Looks very similar to the procedure for single symbols
- Let $\alpha = Y_1 Y_2 \dots Y_k$
 - Put $\text{FIRST}(Y_1) - \{\epsilon\}$ in $\text{FIRST}(\alpha)$
 - If ϵ is in $\text{FIRST}(Y_1)$: add $\text{FIRST}(Y_2) - \{\epsilon\}$ to $\text{FIRST}(\alpha)$
 - If ϵ is in $\text{FIRST}(Y_2)$: add $\text{FIRST}(Y_3) - \{\epsilon\}$ to $\text{FIRST}(\alpha)$
 - ...
 - If ϵ is in FIRST of all Y_i symbols, put ϵ into $\text{FIRST}(\alpha)$

Building $\text{FIRST}(\alpha)$ from $\text{FIRST}(X)$

Building $\text{FIRST}(X)$ for nonterm X

for each $X \rightarrow Y_1 Y_2 \dots Y_k$

- Add $\text{FIRST}(Y_1) - \{\epsilon\}$
- If ϵ is in $\text{FIRST}(Y_{1 \text{ to } i-1})$: add $\text{FIRST}(Y_i) - \{\epsilon\}$
- If ϵ is in all RHS symbols, add ϵ

Building $\text{FIRST}(\alpha)$

Let $\alpha = Y_1 Y_2 \dots Y_k$

- Add $\text{FIRST}(Y_1) - \{\epsilon\}$
- If ϵ is in $\text{FIRST}(Y_{1 \text{ to } i-1})$: add $\text{FIRST}(Y_i) - \{\epsilon\}$
- If ϵ is in all RHS symbols, add ϵ

FIRST(α) Example

Building FIRST(α)

Let $\alpha = Y_1 Y_2 \dots Y_k$

- Add FIRST(Y_1) - $\{\epsilon\}$
- If ϵ is in FIRST($Y_{1 \text{ to } i-1}$): add FIRST(Y_i) - $\{\epsilon\}$
- If ϵ is in all RHS symbols, add ϵ

$E \rightarrow TX$

$X \rightarrow +TX \mid \epsilon$

$T \rightarrow FY$

$Y \rightarrow *FY \mid \epsilon$

$F \rightarrow (E) \mid id$

$FIRST(E) = \{ (, id \}$

$FIRST(T) = \{ (, id \}$

$FIRST(F) = \{ (, id \}$

$FIRST(X) = \{ +, \epsilon \}$

$FIRST(Y) = \{ *, \epsilon \}$

$FIRST(TX) = \{ (, id \}$

$FIRST(+TX) = \{ + \}$

$FIRST(FY) = \{ (, id \}$

$FIRST(*FY) = \{ * \}$

$FIRST((E)) = \{ (\}$

$FIRST(id) = \{ id \}$

FIRST Sets aren't enough for Parse Tables

- If a rule can derive ϵ , we need to know what comes next
 - Obviously, some productions won't work

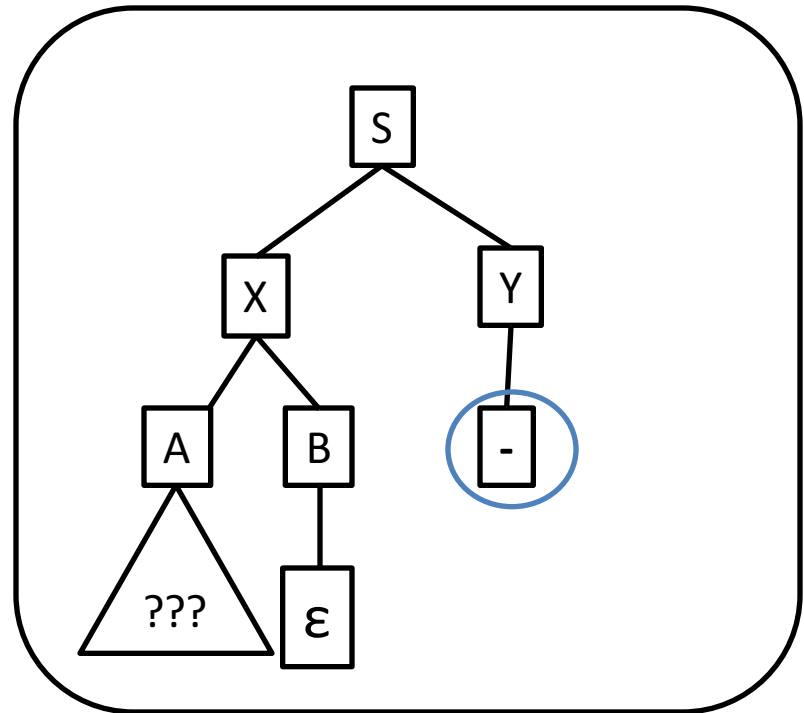
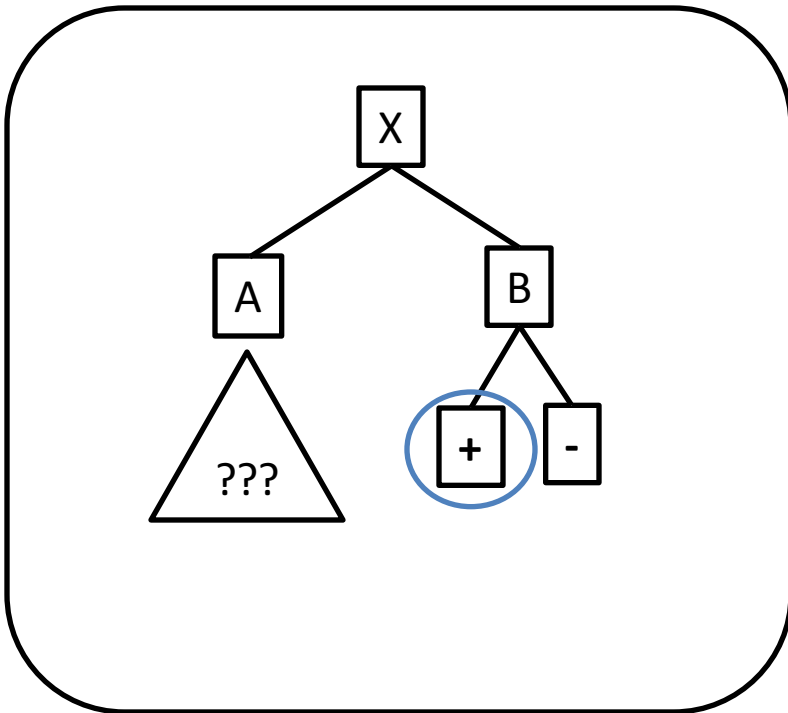
FOLLOW Sets

- For nonterminal A , $\text{FOLLOW}(A)$ is the set of terminals that can appear immediately to the right of A
- Formally, $\text{FOLLOW}(A) =$

$$\{t \mid (t \in \Sigma \wedge S \xRightarrow{+} \alpha A t \beta) \vee (t = \mathbf{eof} \wedge S \xRightarrow{+} \alpha A)\}$$

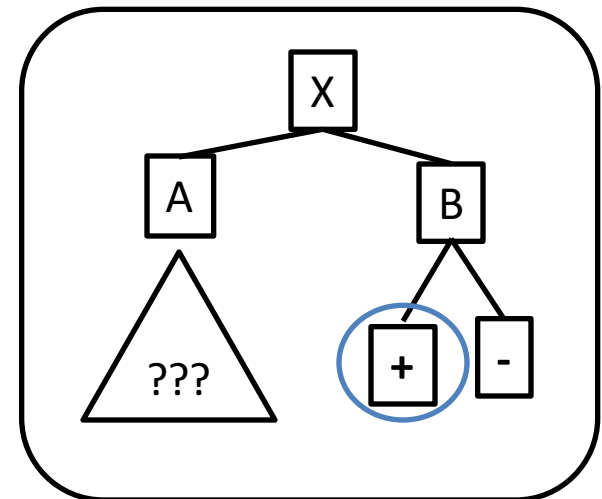
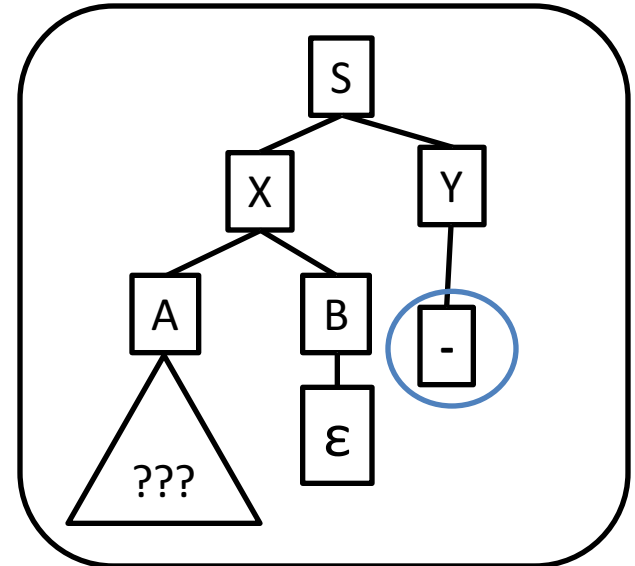
FOLLOW Sets: Pictorially

- For nonterminal A , $\text{FOLLOW}(A)$ is the set of terminals that can appear immediately to the right of A



FOLLOW Sets: Construction

- To build FOLLOW(A)
 - If A is the start nonterminal, add **eof** Where α, β may be empty
 - For rules $X \rightarrow \alpha A \beta$
 - Add $\text{FIRST}(\beta) - \{\epsilon\}$
 - If ϵ is in $\text{FIRST}(\beta)$ or β is empty, add FOLLOW(X)
- Continue building FOLLOW sets until saturation



FOLLOW Sets Example

FOLLOW(A) for $X \rightarrow \alpha A \beta$

If A is the start, add **eof**

Add FIRST(β) – $\{\epsilon\}$

Add FOLLOW(X) if ϵ in FIRST(β) or β is empty

$S \rightarrow B c | D B$

$B \rightarrow a b | c S$

$D \rightarrow d | \epsilon$

FIRST (S) = { **a, c, d** }

FIRST (B) = { **a, c** }

FIRST (D) = { **d, ϵ** }

FIRST (B c) = { **a, c** }

FIRST (D B) = { **d, a, c** }

FIRST (**a b**) = { **a** }

FIRST (**c S**) = { **c** }

FOLLOW (S) = { **eof** }

FOLLOW (B) = { **c, eof** }

FOLLOW (D) = { **a, c** }

FOLLOW (S) = { **eof, c** }

FOLLOW (B) = { **c, eof** }

FOLLOW (D) = { **a, c** }

FOLLOW (S) = { **eof, c** }

FOLLOW (B) = { **c, eof** }

FOLLOW (D) = { **a, c** }

Building the Parse Table

```
for each production  $X \rightarrow \alpha$  {  
    for each terminal  $\mathbf{t}$  in  $\text{FIRST}(\alpha)$  {  
        put  $\alpha$  in  $\text{Table}[X][\mathbf{t}]$   
    }  
    if  $\epsilon$  is in  $\text{FIRST}(\alpha)$  {  
        for each terminal  $\mathbf{t}$  in  $\text{FOLLOW}(X)$  {  
            put  $\alpha$  in  $\text{Table}[X][\mathbf{t}]$   
        }  
    }  
}
```

Table collision \Leftrightarrow Grammar is not LL(1)

Putting it all together

- Build FIRST sets for each nonterminal
- Build FIRST sets for each production's RHS
- Build FOLLOW sets for each nonterminal
- Use FIRST and FOLLOW to fill parse table for each production

Tips n' Tricks

- FIRST sets
 - Only contain alphabet terminals and ε
 - Defined for arbitrary RHS and nonterminals
 - Constructed by starting at the beginning of a production
- FOLLOW sets
 - Only contain alphabet terminals and **eof**
 - Defined for nonterminals only
 - Constructed by jumping into production

FIRST(α) for $\alpha = Y_1 Y_2 \dots Y_k$

Add FIRST(Y_1) - $\{\epsilon\}$

If ϵ is in FIRST($Y_{1 \text{ to } i-1}$): add FIRST(Y_i) - $\{\epsilon\}$

If ϵ is in all RHS symbols, add ϵ

FOLLOW(A) for $X \rightarrow \alpha A \beta$

If A is the start, add **eof**

Add FIRST(β) - $\{\epsilon\}$

Add FOLLOW(X) if ϵ in FIRST(β) or β empty

Table[X][t]

for each production $X \rightarrow \alpha$

for each terminal **t** in FIRST(α)

put α in Table[X][**t**]

if ϵ is in FIRST(α) {

for each terminal **t** in FOLLOW(X) {

put α in Table[X][**t**]

FIRST(S) = { **a, c, d** }

FIRST(B) = { **a, c** }

FIRST(D) = { **d, ϵ** }

FIRST(B c) = { **a, c** }

FIRST(D B) = { **d, a, c** }

FIRST(**a b**) = { **a** }

FIRST(**c S**) = { **c** }

FOLLOW(S) = { **eof, c** }

FOLLOW(B) = { **c, eof** }

FOLLOW(D) = { **a, c** }

CFG

S \rightarrow B c l D B

B \rightarrow a b l c S

D \rightarrow d l ϵ



	a	b	c	d	eof
S	B c D B		B c D B	D B	
B	a b		c S		
D	ϵ		ϵ		