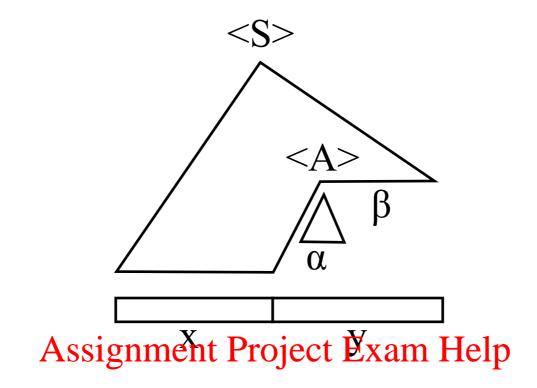
Review: Top-Down Parsing - LL(1)



Basic Idea:

https://powcoder.com

- The parse tree is constanted from the tree's frontier following a **leftmost** derivation.
- The input program is read from **left** to right, and input tokens are read (consumed) as the program is parsed.
- The next non-terminal symbol is replaced using one of its rules. The particular choice <u>has to be unique</u> and uses parts of the input (partially parsed program), for instance the first token of the remaining input.

Review: Predictive Parsing

Basic idea:

For any two productions $A := \alpha$ and $A := \beta$, we would like a distinct way of choosing the correct production to expand.

Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder

```
id_list ::= id id_list_tail
id_list_tail ::= , id id_list_tail
id_list_tail ::= ;
```

Remaining Input: , B , C ;



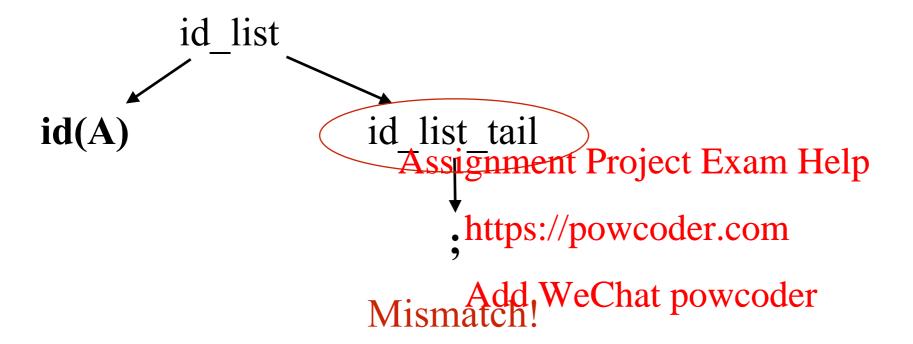
https://powcoder.com

Add WeChat powcoder

Applied Production:

```
id_list ::= id id_list_tail
id_list_tail ::= , id id_list_tail
id_list_tail ::= ;
```

Remaining Input: ,B,C;



Applied Production: id_list_tail ::=;

```
id_list ::= id id_list_tail
id_list_tail ::= , id id_list_tail
id_list_tail ::= ;
```

Remaining Input: , B , C ;



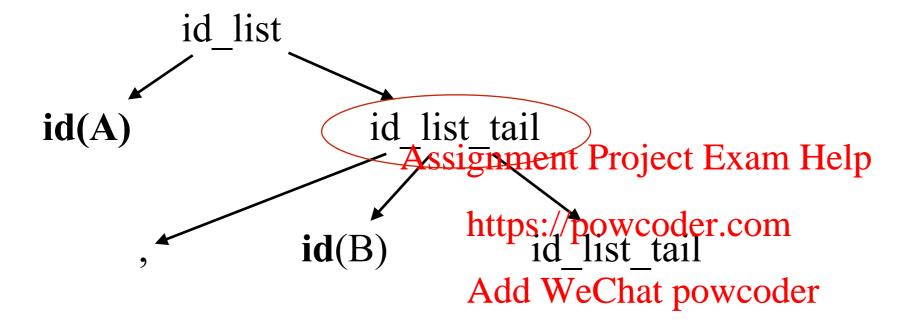
https://powcoder.com

Add WeChat powcoder

Applied Production:

```
id_list ::= id id_list_tail
id_list_tail ::= , id id_list_tail
id_list_tail ::= ;
```

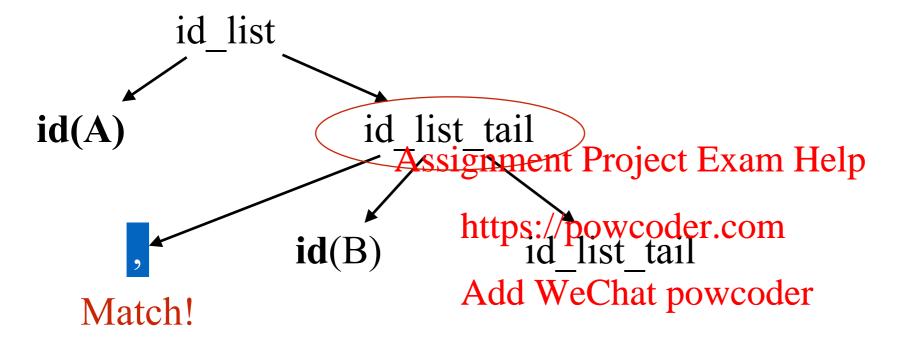
Remaining Input: , B , C ;



Applied Production: id list tail ::=, id id list tail

```
id_list ::= id id_list_tail
id_list_tail ::= , id id_list_tail
id_list_tail ::= ;
```

Remaining Input: ,B,C;



Applied Production: id_list_tail ::= , id id_list_tail

Review: First Set

For some string α , define **FIRST**(α) as the set of tokens that appear as the first symbol in some string derived from α .

That is

 $x \in FIRST(\alpha)$ iff $\alpha \Rightarrow^* x\gamma$ for some string γ Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder

Review: Predictive Parsing

Key Property:

Whenever two productions $A := \alpha$ and $A := \beta$ both appear in the grammar, we would like

• $FIRST(\alpha) \cap FIRST(\beta) = \emptyset$ Project Exam Help

https://powcoder.com

Add WeChat powcoder

```
id_list ::= id id_list_tail
id_list_tail ::= , id id_list_tail
id_list_tail ::= ;
```

Remaining Input: , B , C ;

```
id_list_tail FIRST(, id id_list_tail) = \{,\}

Assignment Project Exam Help
FIRST(;) = \{;\}

https://powcoder.com

FIRST(, id id_list_tail) = \{,\}

Add WeChat-powcoder
```

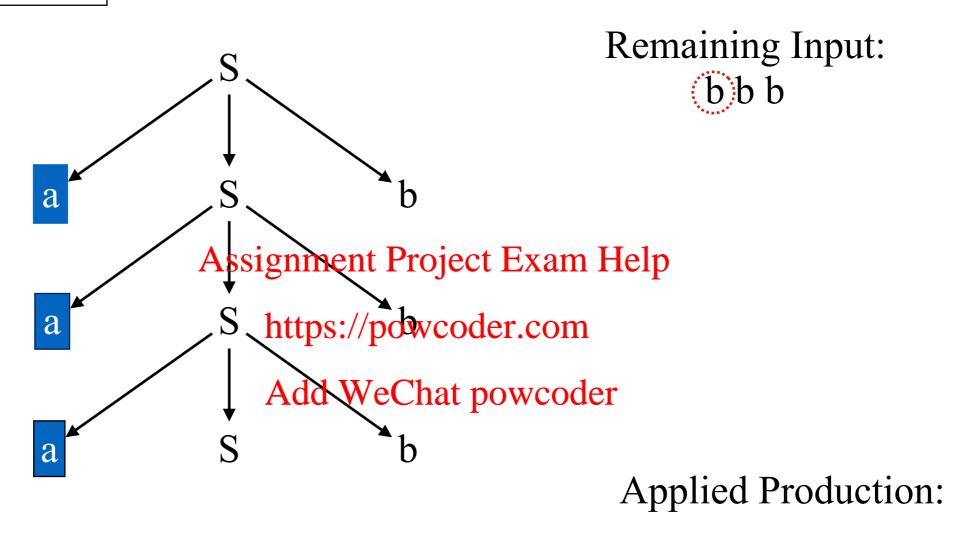
Given id_list_tail as the first **non-terminal** to expand in the tree:

If the first token of remaining input is, we choose the rue id_list_tail ::=, id id_list_tail

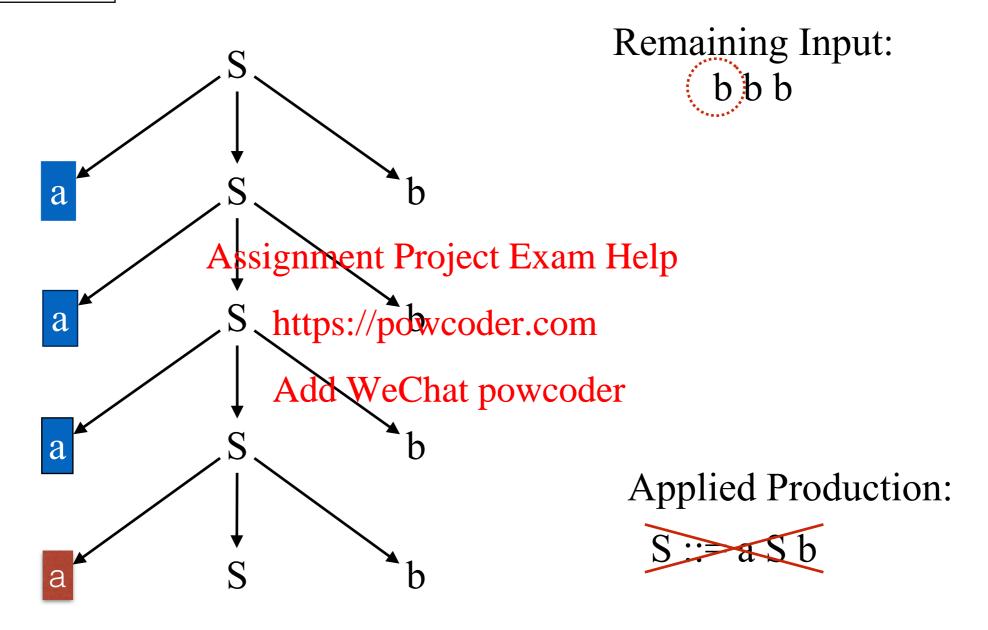
If the first token of remaining input is; we choose the rule

id_list_tail ::= ;

$$S := a S b | \varepsilon$$

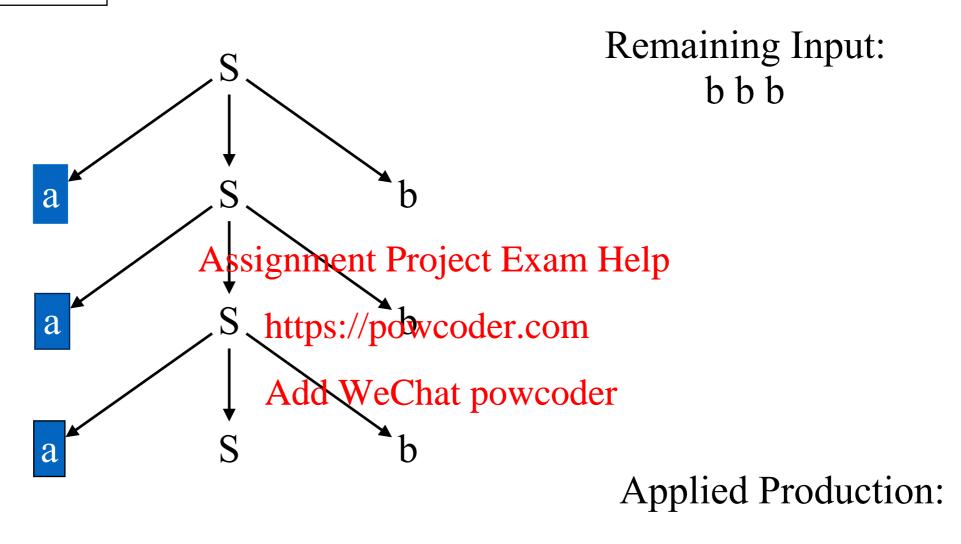


 $S := a S b | \varepsilon$

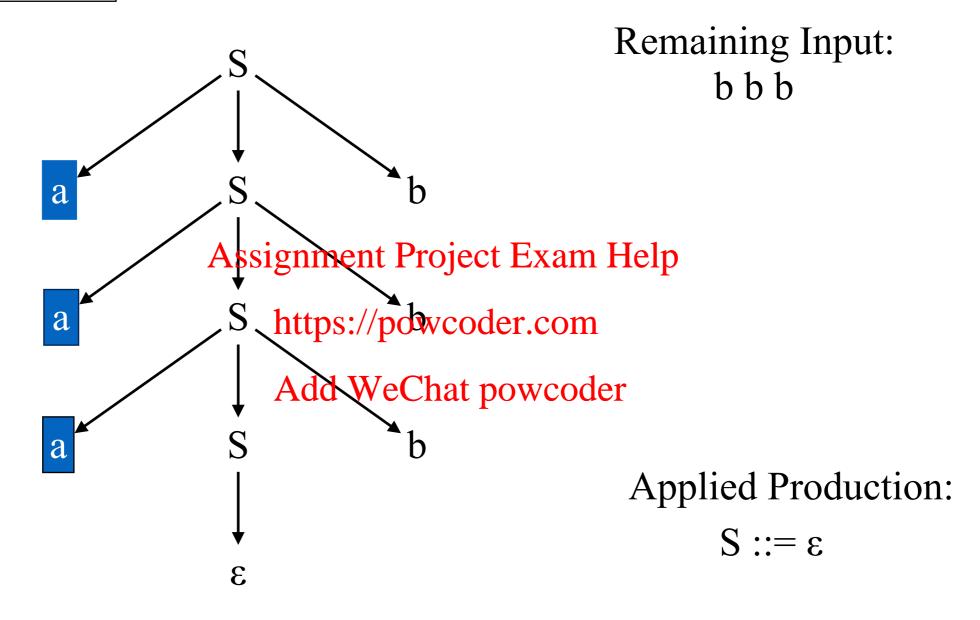


Mismatch!
It only means S ::= aSb is not the right production rule to use!

$$S := a S b | \varepsilon$$



$$S := a S b | \varepsilon$$



 $S := \varepsilon$ turns out to be the right rule later.

However, at this point, ε does not match "b" either!

Review: Follow Set

For a non-terminal A, define **FOLLOW**(A) as the set of terminals that can appear immediately to the right of A in some sentential form.

Thus, a non-terminal's **FOLLOW** set specifies the tokens that can legally appear after it. A terminal symbol has no **FOLLOW** set.

Assignment Project Exam Help

https://powcoder.com

FIRST and FOLLOW sets can be constructed automatically

Review: Predictive Parsing

Key Property:

Whenever two productions $A := \alpha$ and $A := \beta$ both appear in the grammar, we would like

- $FIRST(\alpha) \cap FIRST(\beta) = FrojectExam Help$
- if $\alpha \Rightarrow * \epsilon$, then $FIRST(\beta) \cap FOLLOW(A) = \emptyset$

Analogue case for $\beta \stackrel{\text{Add}}{\Rightarrow} \text{WeChat powcoder}$

Note: due to first condition, at most one of α and β can derive ϵ .

This would allow the parser to make a correct choice with a lookahead of only one symbol!

Review: LL(1) Grammar

Define $PREDICT(A := \delta)$ for rule $A := \delta$

- $FIRST(\delta)$ { ε } U Follow (A), if $\varepsilon \in FIRST(\delta)$
- $FIRST(\delta)$ otherwise

Assignment Project Exam Help

```
A Grammar is LL(1) ifftps://powcoder.com (A ::= \alpha \text{ and } A ::= \beta) \text{ implies} \\ \text{Add WeChat powcoder} \\ \text{PREDICT}(A ::= \alpha) \cap \text{PREDICT}(A ::= \beta) = \emptyset
```

Table Driven LL(1) Parsing

Example:

Predict Sets

$$S := a S b | \varepsilon$$

$$PREDICT(S := aSb) = \{a\}$$

$$PREDICT(S := aSb) = \{a\}$$

 $PREDICT(S := \epsilon) = \{b, eof\}$

Assignment Project Exam Help

LL(1) parse table

https://powcoder.com

	Add We	Chatbpowe	ode e of	other
S	S := aSb	$S ::= \varepsilon$	S ::= ε	error

Table Driven LL(1) Parsing

Example:

Predict Sets

$$S := a S b | \epsilon$$

$$PREDICT(S := aSb) = \{a\}$$

$$PREDICT(S := aSb) = \{a\}$$

 $PREDICT(S := \epsilon) = \{b, eof\}$

Assignment Project Exam Help

LL(1) parse table

https://powcoder.com

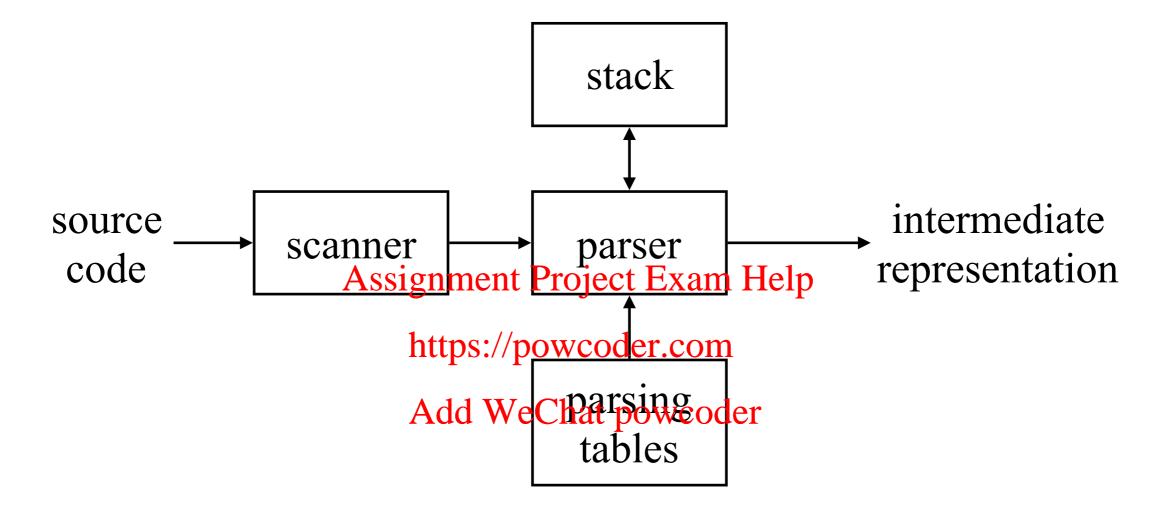
	Add We	Chatlpowe	ode c of	other
S	S := aSb	S ::= ε	S ::= ε	error

Review: Table Driven LL(1) Parsing

```
Input: a string w and a parsing table M for G
     push eof
     push Start Symbol
     token \leftarrow next \ token()
     X \leftarrow \text{top-of-stack}
                                                                  b
                                                                                   other
                                                                          eof
     repeat
                                                       a
         if X is a terminal then
                                               S \mid S ::= aSb \mid S ::= \epsilon \mid S ::= \epsilon
                                                                                    error
           if X == tokenstlæment Project Exam Help
              pop X
              token \leftarrow next\_token() https://powcoder.com
                              Add WeChat powcodeM is the parse table
           else error()
          else /* X is a non-terminal */
               if M[X, token] == X \rightarrow Y_1Y_2 \dots Y_k then
                  pop X
                  push Y_k, Y_{k-1}, \ldots, Y_1
                else error()
           X \leftarrow \text{top-of-stack}
     until X = EOF
     if token != EOF then error()
```

Predictive Parsing

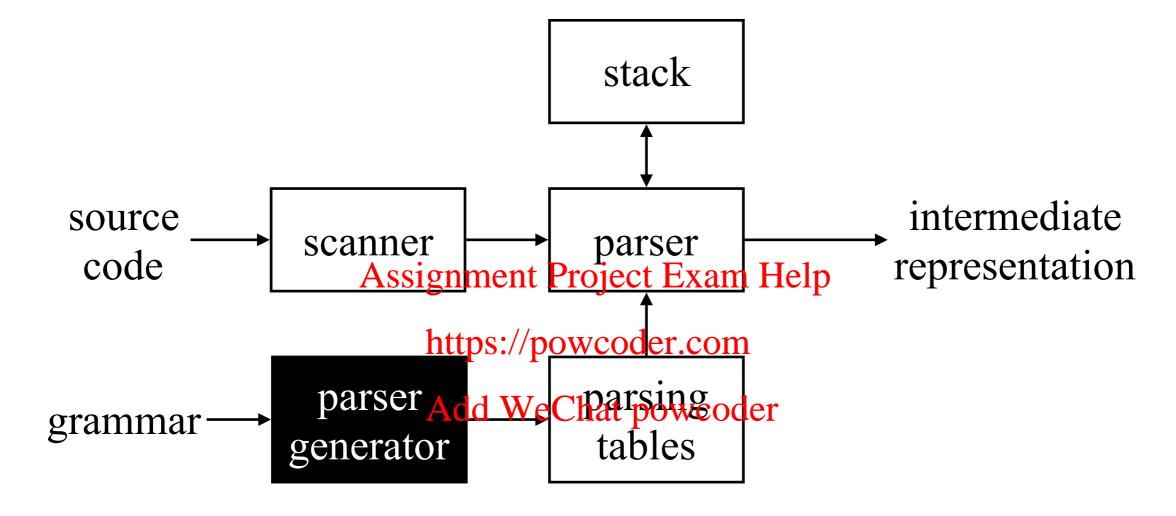
Now, a predictive parser looks like:



Rather than writing code, we build tables.

Predictive Parsing

Now, a predictive parser looks like:



Rather than writing code, we build tables. Building tables can be automated!

Predictive Parsing

So far:

- Introduced FIRST, FOLLOW, and PREDICT sets
- Introduced LL(1) condition:
 - A grammar G can be parsed predictively with one symbol of lookahead if for all pairs of productions $A := \alpha$ and $A := \beta$ that satisfy:

PREDICT(A ::= α) \cap PREDICT(A ::= β) = \emptyset

• Introduced a recursive descent parser for an LL(1) grammar https://powcoder.com

-----Add-WeChat-powcoder-----

How to automatically construct *FIRST* and *FOLLOW* sets?

FIRST and FOLLOW Sets

FIRST(α):

For some $\alpha \in (T \cup NT \cup EOF \cup \epsilon)^*$, define **FIRST** (α) as the set of tokens that appear as the first symbol in some string that derives from α .

Assignment Project Exam Help That is, $\mathbf{x} \in FIRST(\alpha)$ iff $\alpha \Rightarrow^* \mathbf{x} \gamma$ for some γ https://powcoder.com

Add WeChat powcoder

FIRST set is defined over the strings of grammar symbols $(T \cup NT \cup EOF \cup \epsilon)^*$

T: terminals NT: non-terminals

For a production $A \rightarrow B_1B_2 \dots B_k$:

- FIRST(A) includes FIRST(B_1) ε
- FIRST(A) includes FIRST(B_2) ε if B_1 can be rewritten as ε
- FIRST(A) includes FIRST(B_3) ε if both B_1 and B_2 can derive ε
- ... Assignment Project Exam Help
- FIRST(A) includes FIRST(B_{p0}) \overline{v}_{c} ε if $B_{1}B_{2}$... B_{m-1} can derive ε

Add WeChat powcoder

```
FIRST(A) includes FIRST(B_1) ... FIRST(B_m) not including \varepsilon iff \varepsilon \in \text{FIRST}(B_1), FIRST(B_2), FIRST(B_3), ..., FIRST(B_{m-1})
```

```
FIRST(A) includes \varepsilon iff \varepsilon \in FIRST(B_1), FIRST(B_2), FIRST(B_3), ..., FIRST(B_k)
```

First Set Construction

Build FIRST(X) for all grammar symbols X:

- For each X as a terminal, then FIRST(X) is {X}
- If $X := \varepsilon$, then $\varepsilon \in FIRST(X)$
- 1. For each X as a non-terminal, initialize FIRST(X) to \emptyset
 - 2. Iterate until no more terminals or ϵ can be added to any FIRST(X): For each rule in the grain project than the project

28

Filling in the Details: Computing FIRST sets

```
for each x \in (T \cup EOF \cup \varepsilon)

FIRST(x) \leftarrow \{x\}

for each A \in NT, FIRST(A) \leftarrow \emptyset
```

Initially, set *FIRST* for each terminal symbol, EOF and ε

```
while (FIRST sets are still changing) do
    for each p \in P, of the form X \to Y_1 Y_2 ... Y_k do
     temp ← FIRST(Y<sub>1</sub>)Assignment Project Exam Help
           i \leftarrow 1
           while ( i \le k-1 and https://kpa.com
              temp \leftarrow temp \cup (FIRST(Y<sub>i+1</sub>) - { \varepsilon })
                                  Add WeChat powcoder
              i \leftarrow i + 1
           end // while loop
           if i == k and \varepsilon \in FIRST(Y_k)
           then temp \leftarrow temp \cup { \varepsilon }
           FIRST(X) \leftarrow FIRST(X) \cup temp
        end // if - then
     end // for loop
end // while loop
```

Filling in the Details: Computing FIRST sets

```
ε complicates matters
for each x \in (T \cup EOF \cup \varepsilon)
    FIRST(x) \leftarrow \{x\}
for each A \in NT, FIRST(A) \leftarrow \emptyset
while (FIRST sets are still changing) do
     for each p \in P, of the form X \to Y_1 Y_2 ... Y_k do
     temp \leftarrow FIRST(Y_1)_{Assignment Project Exam Help}
           i \leftarrow 1
           while ( i ≤ k-1 and htepsiksowcoder.com
                                                                    If FIRST(Y_1) contains \varepsilon, then
              temp \leftarrow temp \cup (FIRST(Y<sub>i+1</sub>) - { \varepsilon })
                                                                    we need to add FIRST(Y_2) to
                                Add WeChat powcoder
              i \leftarrow i + 1
                                                                    rhs, and ...
           end // while loop
           if i == k and \varepsilon \in FIRST(Y_k)
               then temp \leftarrow temp \cup \{ \epsilon \}
           FIRST(X) \leftarrow FIRST(X) \cup temp
        end // if - then
     end // for loop
end // while loop
```

Filling in the Details: Computing FIRST sets

```
ε complicates matters
for each x \in (T \cup EOF \cup \varepsilon)
    FIRST(x) \leftarrow \{x\}
for each A \in NT, FIRST(A) \leftarrow \emptyset
while (FIRST sets are still changing) do
     for each p \in P, of the form X \to Y_1 Y_2 ... Y_k do
     temp \leftarrow FIRST(Y_1)_{Assignment Project Exam Help}
           i \leftarrow 1
            while ( i ≤ k-1 and htepsiksoder.com
               temp \leftarrow temp \cup (FIRST(Y<sub>i+1</sub>) - { \varepsilon })
                                     Add WeChat powcoder
               i \leftarrow i + 1
            end // while loop
           if i == k and \epsilon \in FIRST(Y_k)
                                                                        If the entire rhs can go to \varepsilon,
                then temp \leftarrow temp \cup { \varepsilon }
                                                                        then we add \varepsilon to FIRST(lhs)
            \overline{\textit{FIRST}(X)} \leftarrow \textit{FIRST}(X) \cup \text{temp}
        end // if - then
     end // for loop
end // while loop
```

```
for each x \in (T \cup EOF \cup \varepsilon)
    FIRST(x) \leftarrow \{x\}
for each A \in NT, FIRST(A) \leftarrow \emptyset
while (FIRST sets are still changing) do
     for each p \in P, of the form X \to Y_1 Y_2 ... Y_k do
     temp \leftarrow FIRST(Y_1)_{Assignment Project Exam Help}
           i \leftarrow 1
            while ( i \le k-1 and k \in \mathbb{R} while ( i \le k-1 and k \in \mathbb{R}
               temp \leftarrow temp \cup (FIRST(Y<sub>i+1</sub>) - { \varepsilon })
                                     Add WeChat powcoder
               i \leftarrow i + 1
            end // while loop
            if i == k and \varepsilon \in FIRST(Y_k)
                                                                             Outer loop is monotone
                 then temp \leftarrow temp \cup { \varepsilon }
                                                                             increasing for FIRST sets
            FIRST(X) \leftarrow FIRST(X) \cup temp
                                                                             \Rightarrow | T \cup NT \cup EOF \cup \epsilon | is
         end // if - then
                                                                             bounded, so it terminates
     end // for loop
end // while loop
```

Example

Consider the SheepNoise grammar and its *FIRST* sets

Goal ::= SheepNoise

SheepNoise ::= SheepNoise baa |

baa

baa is a terminal symbol

Clearly, $FIRST(x) = \{baa\}, \forall x \in (T \cup NT)\}$ https://powcoder.com

Symbodd WeC	nat powerer Set
Goal	baa
SheepNoise	baa
baa	baa

```
for each x \in (T \cup EOF \cup \varepsilon)

FIRST(x) \leftarrow \{x\}

for each A \in NT, FIRST(A) \leftarrow \emptyset
```

Initialization assigns each *FIRST* set a value

FIRST Set

while (<i>FIRST</i> sets are still changing) do for each $p \in P$, of the form $X \to Y_1Y_2Y_k$ do temp \leftarrow <i>FIRST</i> (Y_1) Assignment Project Exam Help $i \leftarrow 1$ while ($i \le k-1$ and https://ps.com temp \leftarrow temp \cup (<i>FIRST</i> (Y_{i+1}) - { ϵ }) $i \leftarrow i+1$ Add WeChat powcoder end // while loop		
if $i == k$ and $\varepsilon \in FIRST(Y_k)$	~ 1 1	:
then temp \leftarrow temp $\cup \{ \epsilon \}$	Symbol	
$FIRST(X) \leftarrow FIRST(X) \cup temp$	Goal	
end // if - then	~1 > 7 ·	
end // for loop	SheepNoise	: : :
end // while loop	baa	

```
for each x \in (T \cup EOF \cup \varepsilon)
    FIRST(x) \leftarrow \{x\}
for each A \in NT, FIRST(A) \leftarrow \emptyset
                                                                    Goal
                                                                                 ::= SheepNoise
while (FIRST sets are still changing) do
                                                                   SheepNoise ::= SheepNoise baa
    for each p \in P, of the form X \to Y_1 Y_2 ... Y_k do
    temp \leftarrow FIRST(Y_1)_{Assignment Project Exam} HelpheepNoise ::= baa
          i \leftarrow 1
           while (i \le k-1 and https://kpower.com
                                                                         If we visit the rule
             temp \leftarrow temp \cup (FIRST(Y_{i+1}) - { \epsilon })
                                                                         in the order 3, 2, 1
                                  Add WeChat powcoder
             i \leftarrow i + 1
           end // while loop
           if i == k and \varepsilon \in FIRST(Y_k)
               then temp \leftarrow temp \cup { \varepsilon }
                                                                          Symbol
                                                                                          FIRST Set
          FIRST(X) \leftarrow FIRST(X) \cup temp
                                                                           Goal
        end // if - then
                                                                       SheepNoise
     end // for loop
end // while loop
                                                                            baa
                                                                                             {baa}
```

```
for each x \in (T \cup EOF \cup \varepsilon)
    FIRST(x) \leftarrow \{x\}
for each A \in NT, FIRST(A) \leftarrow \emptyset
                                                                     Goal
                                                                                  ::= SheepNoise
while (FIRST sets are still changing) do
                                                                     SheepNoise ::= SheepNoise baa
    for each p \in P, of the form X \to Y_1 Y_2 ... Y_k do
     temp \leftarrow FIRST(Y_1)_{Assignment Project Exam} HelpheepNoise ::= baa
           i \leftarrow 1
           while (i \le k-1 and hepsiles of weather.com
                                                                          If we visit the rule
              temp \leftarrow temp \cup (FIRST(Y<sub>i+1</sub>) - { \varepsilon })
                                                                          in the order 3, 2, 1
                                  Add WeChat powcoder
              i \leftarrow i + 1
           end // while loop
           if i == k and \varepsilon \in FIRST(Y_k)
                                                                                           FIRST Set
               then temp \leftarrow temp \cup \{ \epsilon \}
                                                                           Symbol
           FIRST(X) \leftarrow FIRST(X) \cup temp
                                                                             Goal
        end // if - then
                                                                         SheepNoise
                                                                                               {baa}
     end // for loop
end // while loop
                                                                              baa
                                                                                               {baa}
```

An Example

Consider the simplest parentheses grammar

```
1 Goal ::= List
2 List ::= Pair List
3 | ε
4 Pair ::= <u>LP</u> List <u>RP</u>
```

A	Assignmen	it Project Ex Symbol	kam Help Initiat	1 st	2^{nd}
If we visit the rules in order 4, 3, 2, 1		powcoder. Goal			
	Add V ⇒	VeChat pow List	coder Ø		
		Pair	Ø		
		LP	<u>LP</u>	<u>LP</u>	<u>LP</u>
		RP	<u>RP</u>	<u>RP</u>	<u>RP</u>
	•	EOF	EOF	EOF	EOF

An Example

Consider the simplest parentheses grammar

```
Goal ::= List
List ::= Pair List
                                          Where \underline{LP} is ( and \underline{RP} is )
Pair ::= \underline{LP} List \underline{RP}
                        Assignment Project Exam Help
                                                                     1 st
                                                                                  2nd
                             https://powcoder.com
                             Add WeChat powcoder
If we visit the rules
                                         List
in order 4, 3, 2, 1
                                        Pair
                                                        Ø
                                         LP
                                                       LP
                                         RP
                                                      RP
                                                                    RP
                                                                                 RP
```

EOF

EOF

EOF

EOF

```
Goal ::= List
List ::= Pair List
                                         Where \underline{LP} is ( and \underline{RP} is )
Pair ::= \underline{LP} List \underline{RP}
                       Assignment Project Exam Help
                                                                   1 st
                                                                               2nd
                            https://powcoder.com
                            Add WeChat powcoder
If we visit the rules
                                        List
in order 4, 3, 2, 1
                                       Pair
                                                      Ø
                                        LP
                                                     LP
                                        RP
                                                     RP
                                                                  RP
                                                                               RP
                                       EOF
                                                    EOF
                                                                 EOF
                                                                              EOF
```

```
Goal ::= List
List ::= Pair List
                                          Where \underline{LP} is ( and \underline{RP} is )
Pair ::= \underline{LP} List \underline{RP}
                        Assignment Project Exam Help
                                                                    1 st
                                                                                 2nd
                             https://powcoder.com
                             Add WeChat powcoder
If we visit the rules
                                         List
                                                                  <u>LP</u>, ε
in order 4, 3, 2, 1
                                        Pair
                                                       Ø
                                         LP
                                                      LP
                                         RP
                                                      RP
                                                                   RP
                                                                                 RP
                                        EOF
                                                     EOF
                                                                  EOF
                                                                                EOF
```

```
Goal ::= List
List ::= Pair List
                                          Where \underline{LP} is ( and \underline{RP} is )
Pair ::= \underline{LP} List \underline{RP}
                        Assignment Project Exam Help
                                                                    1 st
                                                                                 2nd
                             https://powcoder.com
                             Add WeChat powcoder
If we visit the rules
                                         List
                                                                  <u>LP</u>, ε
in order 4, 3, 2, 1
                                        Pair
                                                       Ø
                                         LP
                                                      LP
                                         RP
                                                      RP
                                                                   RP
                                                                                 RP
                                        EOF
                                                     EOF
                                                                  EOF
                                                                                EOF
```

```
Goal ::= List
List ::= Pair List
                                           Where \underline{LP} is ( and \underline{RP} is )
Pair ::= \underline{LP} List \underline{RP}
                        Assignment Project Exam Help
                                                                      1 st
                                                                                   2<sup>nd</sup>
                             https://powcoder.com
                                                                   LP, ε
                             Add WeChat powcoder
If we visit the rules
                                                                   LP, ε
                                         List
in order 4, 3, 2, 1
                                         Pair
                                                        Ø
                                          LP
                                                       LP
                                         RP
                                                       RP
                                                                     RP
                                                                                  RP
                                                                                 EOF
                                        EOF
                                                      EOF
                                                                   EOF
```

Consider the simplest parentheses grammar

```
1 Goal ::= List
2 List ::= Pair List
3 | ε
4 Pair ::= <u>LP</u> List <u>RP</u>
```

Assignment Project Exam Help 1 st 2nd https://powcoder.com <u>LP</u>, ε Add WeChat powcoder If we visit the rules LP, ε List in order 4, 3, 2, 1 Pair Ø LP LP RP RP RP RP **EOF EOF EOF EOF**

Consider the simplest parentheses grammar

```
1 Goal ::= List
2 List ::= Pair List
3 | ε
4 Pair ::= <u>LP</u> List <u>RP</u>
```

Assignment Project Exam Help 1 st 2nd https://powcoder.com <u>LP</u>, ε Add WeChat powcoder If we visit the rules LP, ε List <u>LP</u>, ε in order 4, 3, 2, 1 Pair Ø LP LP RP RP RP RP **EOF EOF EOF EOF**

Consider the simplest parentheses grammar

```
1 Goal ::= List
2 List ::= Pair List
3 | ε
4 Pair ::= <u>LP</u> List <u>RP</u>
```

Where \underline{LP} is (and \underline{RP} is)

	Assignmer	symbol	kam Help Initial	1 st	2 nd
If we visit the rules in order 4, 3, 2, 1		/powcoder.coal	, —	<u>LP</u> , ε	<u>LP</u> , ε
	Add V ⇒	VeChat pow List	coder Ø	<u>LP</u> , ε	<u>LP</u> , ε
		Pair	Ø	<u>LP</u>	<u>LP</u>
		LP	<u>LP</u>	<u>LP</u>	<u>LP</u>
		RP	<u>RP</u>	<u>RP</u>	<u>RP</u>
	•	EOF	EOF	EOF	EOF

```
1 Goal ::= List
2 List ::= Pair List
3 | ε
4 Pair ::= <u>LP</u> List <u>RP</u>
```

	Assignmen	nt <mark>Project E</mark> z Symbol	kam Help Initial	1 st	2 nd
•	Iteration 1 adds LP to https:/ FIRST(Pair) and LP, \varepsilon to Add V FIRST(List) and	/powcoder. Goal	com Ø	<u>LP</u> , ε	<u>LP</u> , ε
	FIRST(List) and Add V	veChat pow List	coder Ø	<u>LP</u> , ε	<u>LP</u> , ε
	FIRST(Goal) ⇒ If we take them in rule	Pair	Ø	<u>LP</u>	<u>LP</u>
	order 4, 3, 2, 1	LP	<u>LP</u>	<u>LP</u>	<u>LP</u>
•	Algorithm reaches fixed point at Iteration 2	RP	<u>RP</u>	<u>RP</u>	<u>RP</u>
	point at iteration 2	EOF	EOF	EOF	EOF

FOLLOW Sets

FOLLOW(A):

For $A \in \mathbf{NT}$, define $\mathbf{FOLLOW}(A)$ as the set of tokens that can occur immediately after A in a valid sentential form.

Assignment Project Exam Help

FOLLOW set is defined by expressed of fiben-terminal symbols, NT.

Add WeChat powcoder

Follow Set Construction

To Build FOLLOW(X) for non-terminal X:

- Place EOF in FOLLOW(<start>)
- 1. For each X as a non-terminal, initialize FOLLOW(X) to \emptyset
 - 2. *Iterate until* no more terminals can be added to any FOLLOW(X):

```
For each rule pinthing Paring Exam Help

If p is of the form A := \alpha B \beta, then

if \epsilon \in FIRST(\beta)

Place \{FIRST(\beta)\} in FOLLOW(B)

else

Place \{FIRST(\beta)\} in FOLLOW(B)

If p is of the form A := \alpha B, then

Place FOLLOW(A) in FOLLOW(B)
```

End iterate

Computing FOLLOW Sets

```
for each A \in NT
    FOLLOW(A) \leftarrow \emptyset
FOLLOW(S) \leftarrow \{ EOF \}
while (FOLLOW sets are still changing) do
                                                                             Don't add ε
    for each p \in P, of the form A \to B_1B_2...B_k do
        TRAILER \leftarrow FOLLOW(A)
for i \leftarrow k \text{ down to signment Project Exam Help}
           if B_i \in NT then https://powcoder.commain checking
               FOLLOW(B_i) \leftarrow FOLLOW(B_i) \cup TRAILER
                if \varepsilon \in FIRST(B_1)^d WeChat poweder right context
                   TRAILER \leftarrow TRAILER \cup (FIRST(B<sub>i</sub>) - { \varepsilon })
                else TRAILER \leftarrow FIRST(B<sub>i</sub>) // no \varepsilon => truncate the right context
             else TRAILER \leftarrow { B_i } // B_i \in T \Rightarrow only 1 symbol
```

To build *FOLLOW* sets, we need *FIRST* sets

Computing *FOLLOW* Sets

For a production $A \rightarrow B_1B_2 \dots B_k$:

- It works its way backward through the production: $B_k, B_{k-1}, \dots B_1$
- It builds the *FOLLOW* sets for the rhs symbols, $B_1, B_2, \dots B_k, \text{not}_{Assignment Project Exam Help}$
- In the absence of ϵ , $FOLLOW(B_i)$ is just $FIRST(B_{i+1})$ https://powcoder.com As always, ϵ makes the algorithm more complex
 - Add WeChat powcoder

To handle ε, the algorithm keeps track of the first word in the trailing right context as it works its way back through rhs: B_k , B_{k-1} , ... B_1

Computing FOLLOW Sets

Consider the simplest parentheses grammar

	Goal ::= List	Symbol	Initial
3		Goal	EOF
4	$Pair ::= \underline{LP} List \underline{RP}$	List	Ø

Assignment Project Exam Help Pair

https://powcoder.com

Initial Values:

Add WeChat powcoder

- Goal, List and Pair are set to Ø
- Goal is then set to { **EOF** }

	Goal ::= List		Symbol	Initial	1 st
2 3	List ::= Pair List ε		Goal	EOF	
4	$Pair ::= \underline{LP} \text{ List } \underline{RP}$		List	Ø	
		Assignn	nent Project E Pair	xam Help	
	Iteration 1:	http	s://powcoder.	com	

Add V	VeChat powcoder Symbol	FIRST Set
If we visit the rules	Goal	<u>LP</u> , ε
in order 1, 2, 3, 4	List	<u>LP</u> , ε
	Pair	<u>LP</u>
	LP	<u>LP</u>
Assume FIRST Sets are	RP	<u>RP</u>
obtained using the algorithm we—	EOF	EOF
discussed in previous slides.		•

Consider the simplest parentheses grammar

1	Goal ::= List List ::= Pair List
2	List ::= Pair List
3	3
4	$Pair ::= \underline{LP} List \underline{RP}$

Symbol	Initial	1 st
Goal	EOF	EOF
List	Ø	EOF

Iteration 1:

https://powcoder.com

Assignment Project Exam Help Pair

If we visit the rules in order 1, 2, 3, 4

Add WeChat powcoder Symbol	FIRST Set
Goal	<u>LP</u> , ε
List	<u>LP</u> , ε
Pair	<u>LP</u>
LP	<u>LP</u>
RP	<u>RP</u>
EOF	EOF

Consider the simplest parentheses grammar

	Goal ::= List	Symbol	Initial
3	List ::= Pair List ε	Goal	EOF
4	$Pair := \underline{LP} List \underline{RP}$	List	Ø
	Assignm Iteration 1:	nent Project E Pair	xam Help

Iteration 1:

If we visit the rules in order 1, 2, 3, 4

Add WeChat powcoder	FIRST Set
Goal	<u>LP</u> , ε
List	<u>LP</u> , ε
Pair	<u>LP</u>
LP	<u>LP</u>
RP	<u>RP</u>
EOF	EOF

1st

EOF

EOF, LP

EOF

Consider the simplest parentheses grammar

	Goal ::= List	Symbol	Initial
2 3	List ::= Pair List ε	Goal	EOF
4	$Pair ::= \underline{LP} \text{ List } \underline{RP}$	List	Ø

Iteration 1:

If we visit the rules in order 1, 2, 3, 4

Add WeChat powcoder Symbol	FIRST Set
Goal	<u>LP</u> , ε
List	<u>LP</u> , ε
Pair	<u>LP</u>
LP	<u>LP</u>
RP	<u>RP</u>
EOF	EOF

1st

EOF

EOF, RP

EOF, LP

Assignment Project Exam Help Pair

Consider the simplest parentheses grammar

	Goal ::= List	Symbol	
2 3	List ::= Pair List ε	Goal	
4	$Pair ::= \underline{LP} List \underline{RP}$	List	

	Symbol	Initial	1 st
	Goal	EOF	EOF
	List	Ø	EOF, RP
Assignn	nent Project E Pair	xam Help	EOF, LP

Iteration 1:

	Add WeChat powcoder Symbol	FIRST Set
If we visit the rules	Goal	<u>LP</u> , ε
in order 1, 2, 3, 4	List	<u>LP</u> , ε
	Pair	<u>LP</u>
	LP	<u>LP</u>
	RP	<u>RP</u>
	EOF	EOF

Consider the simplest parentheses grammar

1	Goal ::= List
2	Goal ::= List List ::= Pair List
3	•
4	$Pair ::= \underline{LP} \text{ List } \underline{RP}$

	Symbol	Initial	1 st	2 nd
	Goal	EOF	EOF	
	List	Ø	EOF , RP	
Assignn	nent Project E Pair	xam Help	EOF, LP	

Iteration 2:

If	we	vis	sit	the	e ri	ıles
in	ord	ler	1,	2,	3,	4

Add WeChat powcoder Symbol	FIRST Set
Goal	<u>LP</u> , ε
List	<u>LP</u> , ε
Pair	<u>LP</u>
LP	<u>LP</u>
RP	<u>RP</u>
EOF	EOF

Consider the simplest parentheses grammar

1	Goal ::= List
2	Goal ::= List List ::= Pair List
3	·
4	Pair ::= \underline{LP} List \underline{RP}

Symbol	Initial	1 st	2 nd
Goal	EOF	EOF	EOF
List	Ø	EOF , RP	EOF, RP
nt Project E Pair	xam Help	EOF , LP	

Iteration 2:

If	we	visit	t the	e ri	ules
in	ord	er 1	, 2,	3,	4

Add WeChat powcoder Symbol	FIRST Set
Goal	<u>LP</u> , ε
List	<u>LP</u> , ε
Pair	<u>LP</u>
LP	<u>LP</u>
RP	<u>RP</u>
EOF	EOF

Consider the simplest parentheses grammar

1	Goal ::= List List ::= Pair List
2	List ::= Pair List
3	3
4	$Pair ::= \underline{LP} List \underline{RP}$

	Symbol	Initial	1st	2 nd
	Goal	EOF	EOF	EOF
	List	Ø	EOF, RP	EOF, RP
	nent Project E Pair	, -	EOF, LP	EOF , RP, LP
https://powcoder.com				

Iteration 2:

If we visit the rules in order 1, 2, 3, 4

Add WeChat powcoder Symbol	FIRST Set
Goal	<u>LP</u> , ε
List	<u>LP</u> , ε
Pair	<u>LP</u>
LP	<u>LP</u>
RP	<u>RP</u>
EOF	EOF

Consider the simplest parentheses grammar

1	Goal ::= List List ::= Pair List
2	List ::= Pair List
3	3
4	$Pair ::= \underline{LP} List \underline{RP}$

Symbol	Initial	1 st	2 nd
Goal	EOF	EOF	EOF
List	Ø	EOF , RP	EOF, RP
nent Project E Pair	xam Help	EOF, LP	EOF , RP, LP

Iteration 2:

If we visit the rules in order 1, 2, 3, 4

Add WeChat powcoder Symbol	FIRST Set
Goal	<u>LP</u> , ε
List	<u>LP</u> , ε
Pair	<u>LP</u>
LP	<u>LP</u>
RP	<u>RP</u>
EOF	EOF

Consider the simplest parentheses grammar

	Goal ::= List	Symbol	Initial	1 st	2 nd
2 3	List ::= Pair List ε	Goal	EOF	EOF	EOF
4	$Pair := \underline{LP} \text{ List } \underline{RP}$	List	Ø	EOF , RP	EOF , RP
	Iteration 2:	nent Project E Pair s://powcoder.		EOF, LP	EOF, RP, LP

 Production 1 adds nothing navd WeCha 	it powcoder	FIRST Set
• Production 2 adds RP to FOLLOW(Pair)	Goal	<u>LP</u> , ε
from $FOLLOW(List)$, $\varepsilon \in FIRST(List)$	List	<u>LP</u> , ε
 Production 3 does nothing Production 4 adds nothing new 	Pair	<u>LP</u>
	LP	<u>LP</u>
	RP	<u>RP</u>
	EOF	EOF

Iteration 3 produces the same result \Rightarrow reached a fixed point (omitted in the table)

Next Lecture

Things to do:

• Read Scott, Chapter 2.1 - 2.3.3; ALSU 2.4

Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder