Review: 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

First Set Example

Start ::= **S eof**

 $S := \mathbf{a} S \mathbf{b} \mid \varepsilon$

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

S can be rewritten as the following:

ab

aaabbb

 $FIRST(S) = \{a, \epsilon\}$ aab Assignment Project Exam Help

https://powcoder.com

Add WeChat poweoder

aSb can be rewritten as the following:

ab aabb

 $FIRST(aSb) = \{a\}$

Computing FIRST Sets

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_{po}) ε if $B_{e}B_{e}$... 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 $\varepsilon \in FIRST(X)$
- For each X as a non-terminal, initialize FIRST(X) to \emptyset
- Iterate until no more terminals or ϵ can be added to any FIRST(X): For each rule in the grant Pariof the toring $::=Y_1Y_2...Y_k$ add a to FIRST(X) if $a \in FIRST(Y_i)$ and $\epsilon \in FIRST(Y_j)$ and we chat powcoder for all $1 \le j \le i-1$ and $i \ge 2$ add ϵ to FIRST(X) if $\epsilon \in FIRST(Y_i)$ for all $1 \le i \le k$ EndFor End iterate

```
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 \leftarrow \textit{FIRST}(Y_1) - \{ \varepsilon \} Resignment Project Exam Help
            i \leftarrow 1
            while ( i \le k-1 and hepsiles of weak der.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
```

```
for each x \in (T \cup EOF \cup \varepsilon)
    FIRST(x) \leftarrow \{x\}
                                                              Initialize FIRST of each non-
for each A \in NT, FIRST(A) \leftarrow \emptyset
                                                              terminal symbol as empty set
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) - \{ \varepsilon \}
Assignment Project Exam Help
           i \leftarrow 1
           while (i \le k-1 and k \ne p \le IRp T  we der.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
```

```
for each x \in (T \cup EOF \cup \varepsilon)
    FIRST(x) \leftarrow \{x\}
                                                              If any FIRST set changes, it might
for each A \in NT, FIRST(A) \leftarrow \emptyset
                                                              affect other FIRST set(s) due to
                                                              the inter-dependence relationship.
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) - \{ \varepsilon \}
Assignment Project Exam Help
           i \leftarrow 1
           while (i \le k-1 and k \ne p \le IRp T  we der.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
```

```
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
                                                                 Check each rule in the
    for each p \in P, of the form X \to Y_1 Y_2 ... Y_k do
                                                                 grammar, see if any other
          temp \leftarrow FIRST(Y_1) - \{\varepsilon\} Project Exam Helpst set needs to be updated.
           while ( i \le k-1 and hepsiles of weakler.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

```
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_1Y_2...Y_k do
           temp \leftarrow \textit{FIRST}(Y_1) - \{ \varepsilon \} Resignment 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
```

```
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_1Y_2...Y_k do
            temp \leftarrow \textit{FIRST}(Y_1) - \{ \varepsilon \} Resignment Project Exam Help
            i \leftarrow 1
            while (i \le k-1 and k \ne p \le IRp T  we der.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)
                                                                            If all the rhs symbols can go to
                 then temp \leftarrow temp \cup { \epsilon }
                                                                            \epsilon, then we add \epsilon to FIRST(lhs)
            \overline{\textit{FIRST}(X)} \leftarrow \textit{FIRST}(X) \cup \text{temp}
         end // if - then
      end // for loop
end // while loop
```

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
while (FIRST sets are still changing) do
     for each p \in P, of the form X \to Y_1Y_2...Y_k do
           temp \leftarrow \textit{FIRST}(Y_1) - \{ \varepsilon \} Project Exam Help
           i \leftarrow 1
            while (i \le k-1 and https://kpower.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)
                                                                           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
```

Consider the simplest parentheses grammar

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

1st means first "while" iteration

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

while (<i>FIRST</i> sets are still changing) do for each $p \in P$, of the form $X \to Y_1Y_2$ https:// temp \leftarrow <i>FIRST</i> (Y_1) - { ε } i \leftarrow 1	Symoon	11000000	1 st	2 nd
	Goal	Ø		
	eChat pow List	coder Ø		
	Pair	Ø		į
	LP	<u>LP</u>	<u>LP</u>	<u>LP</u>
	RP	<u>RP</u>	<u>RP</u>	<u>RP</u>
	EOF	EOF	EOF	EOF

Goal ::= List

Consider the simplest parentheses grammar

```
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
                                                             \varnothing
                                                            LP
Applying
     Pair ::= \underline{LP} List \underline{RP}
                                              RP
                                                            RP
                                                                           RP
                                                                                         RP
                                             EOF
                                                                         EOF
                                                                                        EOF
                                                           EOF
```

1st means first "while" iteration

Goal ::= List

Consider the simplest parentheses grammar

```
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
Applying
                                          LP
                                                        LP
     List ::= Pair List
                                          RP
                                                       RP
                                                                     RP
                                                                                  RP
                                         EOF
                                                                    EOF
                                                                                 EOF
                                                      EOF
```

1st means first "while" iteration

Goal ::= List

Consider the simplest parentheses grammar

```
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
                                                                   LP, ε
                             Add WeChat powcoder
If we visit the rules
                                                                   LP, ε
                                         List
in order 4, 3, 2, 1
                                         Pair
                                                        Ø
                                         LP
                                                       LP
 Applying
     Goal ::= List
```

1st means first "while" iteration

RP

EOF

RP

EOF

RP

EOF

RP

EOF

Consider the simplest parentheses grammar

```
1 | Goal ::= List
2 | List ::= Pair List
3 | \epsilon
4 | Pair ::= LP List RP

Assignment Project Exam Help
Symbol Frame | \epsilon

Add WeChat powcoder | \epsilon
in order 4, 3, 2, 1
```

Pair

Applying

Pair ::= \underline{LP} List \underline{RP}

LP	<u>LP</u>	<u>LP</u>	<u>LP</u>
RP	<u>RP</u>	<u>RP</u>	<u>RP</u>
EOF	EOF	EOF	EOF

Ø

Consider the simplest parentheses grammar

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

1st means first "while" iteration

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

Ass	ignment Project Ex Symbol	kam Help Initiat	1 st	2 nd
	https://powcoder.c		<u>LP</u> , ε	
	Add WeChat 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

Consider the simplest parentheses grammar

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

1st means first "while" iteration

Where <u>LP</u> is (and <u>RP</u> is)

	Assignmen	t Project Ex Symbol	xam Help Initial	1 st	2 nd
		powcoder. Goal		<u>LP</u> , ε	<u>LP</u> , ε
		VeChat pow List	coder Ø	<u>LP</u> , ε	<u>LP</u> , ε
		Pair	Ø	<u>LP</u>	<u>LP</u>
Applying		LP	<u>LP</u>	<u>LP</u>	<u>LP</u>
Goal ::= List		RP	<u>RP</u>	<u>RP</u>	<u>RP</u>
		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>
```

1st means first "while" iteration

FIRST Sets

_		ignment Project E	xam Help Initial	1 st	2 nd
	FIRST (Pair) and {LP.	https://powcoder.	~	<u>LP</u> , ε	<u>LP</u> , ε
ε} to <i>FIRST</i> (List) and ^A	Add WeChat pow List	coder Ø	<u>LP</u> , ε	<u>LP</u> , ε	
	$FIRST$ (Goal) \Rightarrow If we take them in	Pair	Ø	<u>LP</u>	<u>LP</u>
rule order 4, 3, 2, 1Algorithm reaches fixed point	LP	<u>LP</u>	<u>LP</u>	<u>LP</u>	
	RP	<u>RP</u>	<u>RP</u>	<u>RP</u>	
	EOF	EOF	EOF	EOF	

FOLLOW Sets

FOLLOW(A):

For $A \in NT$, define 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 to be the set of the s

Add WeChat powcoder

```
Start ::= S = eof

S ::= a S = b
```

One possible derivation process from the start symbol:

```
Start \Rightarrow Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder

FOLLOW(S) = \{
```

```
Start ::= S = a \cdot S = b  |
```

One possible derivation process from the start symbol:

```
Start \Rightarrow S eof

Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder

FOLLOW(S) = \{
```

Start ::=
$$S = eof$$

 $S ::= a S = b$

One possible derivation process from the start symbol:

Start
$$\Rightarrow$$
 S eof \Rightarrow a S b eof

https://powcoder.com

Add WeChat powcoder

FOLLOW(S) = {

Start ::=
$$S = eof$$

 $S ::= a S = b$

One possible derivation process from the start symbol:

Start
$$\Rightarrow$$
 S eof \Rightarrow a S b eof \Rightarrow a b eof https://powcoder.com

Add WeChat powcoder

$$FOLLOW(S) = \{$$

Start ::=
$$S = eof$$

 $S ::= a S = b$

One possible derivation process from the start symbol:

Start
$$\Rightarrow$$
 S eof \Rightarrow a S b eof \Rightarrow a b eof https://powcoder.com

Add WeChat powcoder

$$FOLLOW(S) = \{ eof, b \}$$

For a production $A \rightarrow B_1B_2 \dots B_{k-1}B_k$:

```
• FOLLOW(B<sub>k</sub>) includes FOLLOW(A)
```

```
• FOLLOW(B_{k-1}) includes FIRST(B_k) - \epsilon, and FOLLOW(A) if B_k can derive \epsilon Assignment Project Exam Help
```

• FOLLOW(B_{k-2}) includes FIRST($B_{k-1}B_k$) - ehttps://powcoder.com and FOLLOW(A) if $B_{k-1}B_k$ can derive a result of the second of t

• . . .

Follow Set Construction

Given a rule *p* in the grammar:

$$A \rightarrow B_1B_2...B_iB_{i+1}...B_k$$

If B_i is a non-terminal, FOLLOW(B_i) includes

- FIRST($B_{i+1}...B_k$) { ϵ } U FOLLOW(A), if ϵ \in FIRST($B_{i+1}...B_k$) Assignment Project Exam Help

Add WeChat powcoder

Follow Set Construction

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

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

```
For each rule gignthen granted Exam Help

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

if \epsilon \in FIRTP(\beta) powered LOW(A)} 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

```
Initially, set FOLLOW for
for each A \in NT
    FOLLOW(A) \leftarrow \emptyset
                                                               each non-terminal symbol
FOLLOW(S) \leftarrow \{ EOF \}
while (FOLLOW sets are still changing) do
     for each p \in P, of the form A \to B_1B_2...B_k do
        TRAILER \leftarrow FOLLOW(A)
for i \leftarrow k down to signment Project Exam Help
             if B_i \in \mathbf{NT} then
                 B_i \in \mathbf{NT} then https://powcoder.com

FOLLOW(B_i) \leftarrow FOLLOW(B_i) \cup TRAILER
                 if \varepsilon \in FIRST(BAdd WeChat powcoder)
                     TRAILER \leftarrow TRAILER \cup (FIRST(B<sub>i</sub>) - { \epsilon })
                 else TRAILER \leftarrow FIRST(B_i)
             else TRAILER \leftarrow \{B_i\}
```

```
for each A \in NT
                                                                 Set FOLLOW for start
     FOLLOW(A) \leftarrow \emptyset
FOLLOW(S) \leftarrow \{ EOF \}
                                                                 symbol S as {EOF}
while (FOLLOW sets are still changing) do
     for each p \in P, of the form A \to B_1B_2...B_k do
        TRAILER \leftarrow FOLLOW(A) for i \leftarrow k down to signment Project Exam Help
             if B_i \in \mathbf{NT} then
                 B_i \in \mathbf{NT} then https://powcoder.com

FOLLOW(B_i) \leftarrow FOLLOW(B_i) \cup TRAILER
                 if \varepsilon \in FIRST(BAdd WeChat powcoder)
                     TRAILER \leftarrow TRAILER \cup (FIRST(B<sub>i</sub>) - { \epsilon })
                 else TRAILER \leftarrow FIRST(B_i)
              else TRAILER \leftarrow \{B_i\}
```

```
for each A \in NT

FOLLOW(A) \leftarrow \emptyset

FOLLOW(S) \leftarrow \{ EOF \}
```

```
while (FOLLOW sets are still changing) do for each p \in P, of the form A \to B_1B_2...B_k do TRAILER \leftarrow FOLLOW(A) for i \leftarrow k down to signment Project Exam Help if B_i \in NT then https://powcoder.com FOLLOW(B_i) \leftarrow FOLLOW(B_i) \cup TRAILER if \epsilon \in FIRST(B_i) dd WeChat powcoder TRAILER \leftarrow TRAILER \leftarrow TRAILER \cup (FIRST(B_i) - \{\epsilon\}) else TRAILER \leftarrow \{B_i\}
```

As long as any *FOLLOW* set changes

for each $A \in NT$

```
FOLLOW(A) \leftarrow \emptyset
FOLLOW(S) \leftarrow \{ EOF \}
while (FOLLOW sets are still changing) do
      for each p \in P, of the form A \to B_1B_2...B_k do
          TRAILER \leftarrow FOLLOW(A)
for i \leftarrow k down to signment Project Exam Help
               if B_i \in \mathbf{NT} then
                   B_i \in \mathbb{N} I then <a href="https://powcoder.com">https://powcoder.com</a>
FOLLOW(B_i) \leftarrow FOLLOW(B_i) \cup TRAILER
                   if \varepsilon \in FIRST(BAdd WeChat powcoder)
                        TRAILER \leftarrow TRAILER \cup (FIRST(B<sub>i</sub>) - { \varepsilon })
                   else TRAILER \leftarrow FIRST(B<sub>i</sub>)
                else TRAILER \leftarrow \{B_i\}
```

As long as any *FOLLOW* set changes, check all the rules

```
for each A \in NT
    FOLLOW(A) \leftarrow \emptyset
FOLLOW(S) \leftarrow \{ EOF \}
while (FOLLOW sets are still changing) do
     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 project Exam Help}
             if B_i \in \mathbf{NT} then
                 B_i \in \mathbf{NT} then https://powcoder.com

FOLLOW(B_i) \leftarrow FOLLOW(B_i) \cup TRAILER
                 if \varepsilon \in FIRST(BAdd WeChat powcoder)
                      TRAILER \leftarrow TRAILER \cup (FIRST(B<sub>i</sub>) - { \epsilon })
                 else TRAILER \leftarrow FIRST(B_i)
              else TRAILER \leftarrow \{B_i\}
```

set trailing context to FOLLOW(A)

```
for each A \in NT
    FOLLOW(A) \leftarrow \emptyset
FOLLOW(S) \leftarrow \{ EOF \}
while (FOLLOW sets are still changing) do
     for each p \in P, of the form A \to B_1B_2...B_k do
        TRAILER \leftarrow FOLLOW(A)
for i \leftarrow k down to signment Project Exam Help
                                                                                     it goes
                                                                                     backwards
            if B_i \in \mathbf{NT} then
                Follow(B_i) \leftarrow Follow(B_i) \cup TRAILER
                if \varepsilon \in FIRST(BAdd WeChat powcoder)
                    TRAILER \leftarrow TRAILER \cup (FIRST(B<sub>i</sub>) - { \varepsilon })
                else TRAILER \leftarrow FIRST(B_i)
             else TRAILER \leftarrow \{B_i\}
```

else TRAILER $\leftarrow \{B_i\}$

```
for each A \in NT

FOLLOW(A) \leftarrow \emptyset

FOLLOW(S) \leftarrow \{ EOF \}

while (FOLLOW \text{ sets are still changing}) do

for each p \in P, of the form A \rightarrow B_1B_2...B_k do

TRAILER \leftarrow FOLLOW(A)

for i \leftarrow k down to fSignment Project Exam Help

if B_i \in NT \text{ then } \text{https://powcoder.com}
FOLLOW(B_i) \leftarrow FOLLOW(B_i) \cup TRAILER
if \varepsilon \in FIRST(B_iAdd WeChat powcoder)
TRAILER \leftarrow TRAILER \cup (FIRST(B_i) - \{ \varepsilon \})
else TRAILER \leftarrow FIRST(B_i)
```

if the symbol is non-terminal, need to check if it derives ε

```
for each A \in NT
    FOLLOW(A) \leftarrow \emptyset
FOLLOW(S) \leftarrow \{ EOF \}
while (FOLLOW sets are still changing) do
     for each p \in P, of the form A \to B_1B_2...B_k do
        TRAILER \leftarrow FOLLOW(A)
for i \leftarrow k down to signment Project Exam Help
            if B_i \in \mathbf{NT} then
                FOLLOW(B_i) \leftarrow FOLLOW(B_i) \cup TRAILER
                                                                                  Consecutive non-
               if \varepsilon \in FIRST(BAdd WeChat powcoder)
                                                                                  terminals that derive
                   TRAILER \leftarrow TRAILER \cup (FIRST(B<sub>i</sub>) - { \varepsilon })
                                                                                  ε in trailing context
               else TRAILER \leftarrow FIRST(B_i)
             else TRAILER \leftarrow \{B_i\}
```

Computing FOLLOW Sets

```
for each A \in NT
    FOLLOW(A) \leftarrow \emptyset
FOLLOW(S) \leftarrow \{ EOF \}
while (FOLLOW sets are still changing) do
     for each p \in P, of the form A \to B_1B_2...B_k do
        TRAILER \leftarrow FOLLOW(A)
for i \leftarrow k down to signment Project Exam Help
            if B_i \in \mathbf{NT} then
                FOLLOW(B_i) \leftarrow FOLLOW(B_i) \cup TRAILER
                if \varepsilon \in FIRST(BAdd WeChat powcoder)
                    TRAILER \leftarrow TRAILER \cup (FIRST(B<sub>i</sub>) - { \varepsilon })
                                                                                   Trailing context
                else TRAILER \leftarrow FIRST(B_i)
                                                                                   needs to be reset
             else TRAILER \leftarrow \{B_i\}
```

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

Computing FOLLOW Sets

```
for each A \in NT
     FOLLOW(A) \leftarrow \emptyset
FOLLOW(S) \leftarrow \{ EOF \}
while (FOLLOW sets are still changing) do
      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 Project Exam Help}
              if B_i \in \mathbf{NT} then
                  B_i \in \mathbf{NT} then https://powcoder.com

FOLLOW(B_i) \leftarrow FOLLOW(B_i) \cup TRAILER
                  if \varepsilon \in FIRST(BAdd WeChat powcoder)
                       TRAILER \leftarrow TRAILER \cup (FIRST(B<sub>i</sub>) - { \varepsilon })
                  else TRAILER \leftarrow FIRST(B<sub>i</sub>)
                                                                                                when B<sub>i</sub> does not
               else \overline{TRAILER} \leftarrow \{B_i\}
                                                                                                derive \varepsilon
```

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

Consider the simplest parentheses grammar

while (<i>FOLLOW</i> sets are still changing) do Add WeChat p	owsoderol	FIRST Set
for each $p \in P$, of the form $A \to B_1B_2B_k$ do TRAILER $\leftarrow FOLLOW(A)$	Goal	<u>LP</u> , ε
for $i \leftarrow k$ down to 1 if $B_i \in \mathbf{NT}$ then	List	<u>LP</u> , ε
$FOLLOW(B_i) \leftarrow FOLLOW(B_i) \cup TRAILER$	Pair	<u>LP</u>
if $\varepsilon \in \textit{FIRST}(B_i)$ TRAILER \leftarrow TRAILER \cup ($\textit{FIRST}(B_i)$ - $\{\varepsilon\}$)	LP	<u>LP</u>
else TRAILER \leftarrow <i>FIRST</i> (B _i) else TRAILER \leftarrow { B _i }	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} \text{ List } \underline{RP}$

Symbol	Initial	1 st
Goal	EOF	EOF
List	Ø	

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

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

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

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>
Goal ::= List	LP	<u>LP</u>
	RP	<u>RP</u>
Add FOLLOW(Goal) to FOLLOW(List)	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
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

List ::= Pair List

Add WeChat powcoder Symbol	FIRST Set
Goal	<u>LP</u> , ε
List	$(\underline{LP}, \epsilon)$
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} \text{ 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	
List ::= Pair List	
A 11 PID OT (I ' 4) 4	

- Add FIRST(List) to FOLLOW(Pair)
- Add FOLLOW(List) to FOLLOW(Pair)

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

EOF, LP

Consider the simplest parentheses grammar

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

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

Iteration 1:

https://powcoder.com

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

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

Pair ::= \underline{LP} List \underline{RP}

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 ::= \underline{LP} List \underline{RP}	Pair	<u>LP</u>
	LP	<u>LP</u>
• Add FIRST(<u>RP</u>) to	RP	<u>RP</u>
FOLLOW(List)	EOF	EOF

Consider the simplest parentheses grammar

1	Goal ::= List List ::= Pair List	Symbo
		Goal
3		Goar
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	List ::= Pair List
3	3
4	Goal ::= List List ::= Pair List ε Pair ::= <u>LP</u> List <u>RP</u>

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

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
Assignn	nent Project E Pair	xam Help	EOF, LP	EOF, LP

Iteration 2:

https://powcoder.com

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

Goal ::= List

Add FOLLOW(Goal) to FOLLOW(List)

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

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

Iteration 2:

https://powcoder.com

If	we	vis	sit	the	e r	ules	5
in	ord	ler	1,	2,	3,	4	

List ::= Pair List

- Add FIRST(List) to FOLLOW(Pair)
- Add FOLLOW(List) to FOLLOW(Pair)

Add WeChat J	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
7	List := Pair List
2	List— Lan List
3	3
4	Goal ::= List List ::= Pair List

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

Iteration 2:

https://powcoder.com

If we vis	sit	the	e r	ule	S
in order	1,	2,	3,	4	

Pair ::= \underline{LP} List \underline{RP}

Add FIRST(<u>RP</u>) toFOLLOW(List)

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.	ment Project E Pair		EOF, LP	EOF, RP, LP
		ps://powcoder.			

 Production 1 adds nothing navd WeCha 	t 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	Pair	<u>LP</u>
 Production 4 adds nothing new 	LP	<u>LP</u>
	RP	<u>RP</u>
	EOF	EOF

Iteration 3 produces the same result \Rightarrow reached a fixed point

Review: LL(1) 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 $\beta \stackrel{\text{Add}}{\Rightarrow} e^{\text{WeChat powcoder}}$

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
```

Building the PREDICT set

• Need a *PREDICT set* for every rule

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

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

DICT set	Symbol	FIRST	FOLLOW
T set for every rule	Goal	<u>LP</u> , ε	EOF
- Set for every full	List	<u>LP</u> , ε	EOF, RP
for rule $A := \delta$	Pair	<u>LP</u>	EOF, RP, LP
w (A), if $\varepsilon \in FIRST(\delta)$	LP	<u>LP</u>	_
	RP	<u>RP</u>	<u>-</u>
Assignment Project E	xam Help	EOF	_

https://powender.com

	nttps://powcoder.com						
Add WeChat powcoder PREDICT							
1	Goal ::= List		1	EOF, LP			
2	List ::= Pair List		2	LP			
3	List ::= ε		3	EOF, RP			
4	$Pair ::= \underline{LP} List \underline{RP}$		4	LP			

Building the PREDICT set

• Need a *PREDICT set* for every rule

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

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

Symbol	FIRST	FOLLOW
Goal	<u>LP</u> , ε	EOF
List	<u>LP</u> , ε	EOF , RP
Pair	<u>LP</u>	EOF, RP, LP
LP	<u>LP</u>	_
RP	<u>RP</u>	_
xamEHelp	EOF	-

https://powcoder.com

Assignment Project E

	Add We	Chat powc	oder ^{le}	PREDICT
1	Goal ::= List		1	EOF, LP
2	List ::= Pair List		2	LP ← FIRST(Pair List)
3	List ::= ε		3	EOF , RP← FOLLOW(List)
4	$Pair := \underline{LP} List \underline{RP}$		4	LP

- Need a row for every NT and a column for every T
- Need an interpreter for the table (skeleton parser)

		Rule	PREDICT		LP	R P	EOF
1	Goal ::= List Assi	gnment P	roj EQE kam Help	Goal	1		1
2	List ::= Pair List	https://po	wcodercom	List			
3	List $:= \epsilon$	Add WeC	ha Ept wRøder	Pair			
4	$Pair ::= \underline{LP} List \underline{RP}$	4	LP	Гап			

- Need a row for every NT and a column for every T
- Need an interpreter for the table (skeleton parser)

		Rule	PREDICT		LP	R P	EOF
1	Goal ::= List Assi	gnment P	roj e Q E Ra m Help	Goal	1		1
2	List ::= Pair List	https://po	wcod e rPcom	List	2	3	3
3	List ::= ε	Add W eC	ha Ept w Ro der	Pair			
4	$Pair ::= \underline{LP} \text{ List } \underline{RP}$	4	LP	Гап			

- Need a row for every NT and a column for every T
- Need an interpreter for the table (skeleton parser)

		Rule	PREDICT		LP	R P	EOF
1	Goal ::= List Assi	gnment P	roj e Q E Ra m Help	Goal	1		1
2	List ::= Pair List	https://po	wcod e rPcom	List	2	3	3
3	3	Add WeC	ha E por wRoder	Pair			
4	$Pair ::= \underline{LP} List \underline{RP}$	4	LP	ıan	-		

- Need a row for every NT and a column for every T
- Need an interpreter for the table (skeleton parser)

		Rule	PREDICT		LP	R P	EOF
1	Goal ::= List Assi	gnment P	roj e Q'E Ra m Help	Goal	1		1
2	List ::= Pair List	https://po	wcodercom	List	2	3	3
3	3	Add WeC	ha Ept wRøder	Pair	Δ		
4	Pair ::= <u>LP</u> List <u>RP</u>	4	LP	1 an	7		

Next Lecture

Things to do:

- Start programming in C.
- Read Scott, Chapter 3.1 3.3; ALSU 7.1
- Read Scott, Chapter 8.1 8.2; ALSU 7.1 7.3 Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder