FIRST and FOLLOW

Course Code: CSC3220 Course Title: Compiler Design

Dept. of Computer Science
Faculty of Science and Technology

Lecturer No:	9	**Week No:**	9	**Semester:**	
Lecturer:	*Md Masum Billah, billah.masumcu@aiub.edu*				

Lecture Outline

1. Review of Subset Construction Rule (NFA to DFA conversion)
2. Overview of First and Follow
3. First and Follow set Rules
4. Examples
5. Exercises

Objective and Outcome

Objective:

- To Explain the necessity or requirement of FIRST and FOLLOW set calculation.
- To elaborate the method/algorithm of FIRST and FOLLOW calculation from a given CFG.
- To provide necessary example and exercise of FIRST and FOLLOW calculation from a given CFG

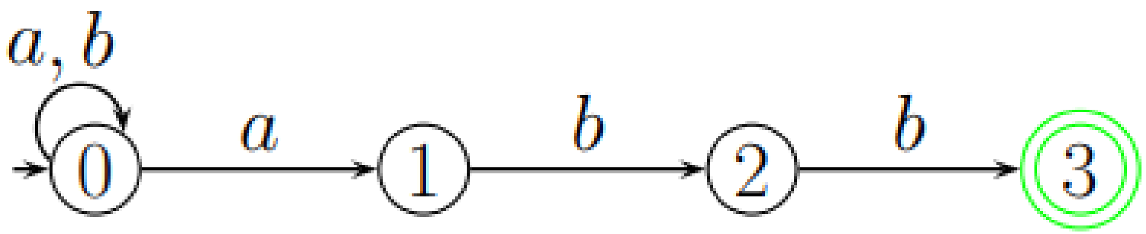
Outcome:

- After this class the students will know the necessity of FIRST and FOLLOW calculation
- After this class the students will be able to demonstrate the FIRST and FOLLOW calculation method.
- The students will also be capable of calculating FIRST and FOLLOW set from a given CFG

Review on NFA to DFA

Example

A NFA for the language, L3 = {a, b}∗{abb}.

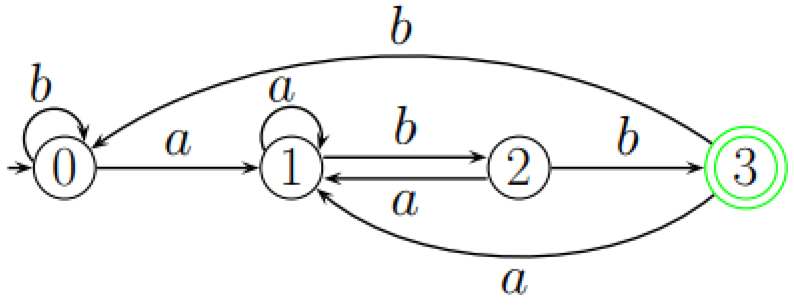


Given NFA

Review on NFA to DFA

Example

names	states	a	b
A	$\{0\}$	B	A
B	$\{0,1\}$	B	C
C	$\{0,2\}$	B	D
D	$\{0,3\}$	B	A



Converted DFA

FIRST and FOLLOW Overview

The basic problem in parsing is choosing which production rule to use at any stage during a derivation.

- Lookahead

Means attempting to analyze the possible production rules which can be applied, in order to pick the one most likely to derive the current symbol(s) on the input.

- FIRST and FOLLOW

We formalize the task of picking a production rule using two functions, FIRST and FOLLOW. we need to find FIRST and FOLLOW sets for a given grammar, so that the parser can properly apply the needed rule at the correct position.

FIRST Set Calculation

Rules

1. If X is terminal, FIRST(X) = {X}.
2. If X → ε is a production, then add ε to FIRST(X).
3. If X is a non-terminal, and X → Y1 Y2 … Yk is a production, and ε is in all of FIRST(Y1), …, FIRST(Yk), then add ε to FIRST(X).
4. If X is a non-terminal, and X → Y1 Y2 … Yk is a production, then add a to FIRST(X) if for some i, a is in FIRST(Yi), and ε is in all
of FIRST(Y1), …, FIRST(Yi-1).

Applying rules 1 and 2 is obvious. Applying rules 3 and 4 for FIRST(Y1 Y2 … Yk) can be done as follows:

Add all the non-ε symbols of FIRST(Y1) to FIRST(Y1 Y2 … Yk). If ε ∈ FIRST(Y1), add all the non-ε symbols of FIRST(Y2). If ε ∈ FIRST(Y1) and ε ∈ FIRST(Y2), add all the non-ε symbols of FIRST(Y3), and so on. Finally, add ε to FIRST(Y1 Y2 … Yk) if ε ∈ FIRST(Yi), for all 1 ≤ i ≤ k.

First Set

The algorithm to compute the firsts set of a symbol X:

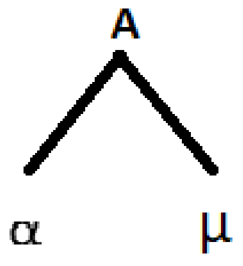
```
if(X is a terminal symbol):
 first(X) = X;
 break;
if (X -> Ɛ ∈ productions of the grammar):
 first(X).add({ Ɛ });
foreach(X -> Y1....Yn ∈ productions of the grammar):
 j = 1;
 while (j <= n):
  first(X).add({ b }), ∀ b ∈ first(Yj) ;
  if ( Ɛ ∈ first(Yj)):
    j ++;
  else:
    break;
if(j = n+1):
 first(X).add({ Ɛ });
```

First Set (Case 1)

- For a Production, if the first things is terminals that terminal (left most) would be considered as a 'First'
- If the Left most thing is a terminals then that terminals will be 'First'
- Don't worry about the rest of the things residing on the right side of the first terminals

Case 1

A -> αμ



FIRST (A) = { α }

First Set (Case 2)

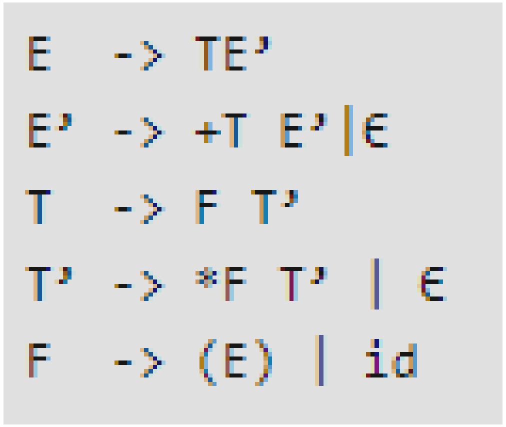
- For a Production, if the first things is epsilon (ε) then ‘FIRST’ is epsilon (ε)

First Set (Case 3)

- For a Production, if the first things is Non-Terminals, then we should continue until we found a terminals.
- Look for the next production and next until we encounter a terminals

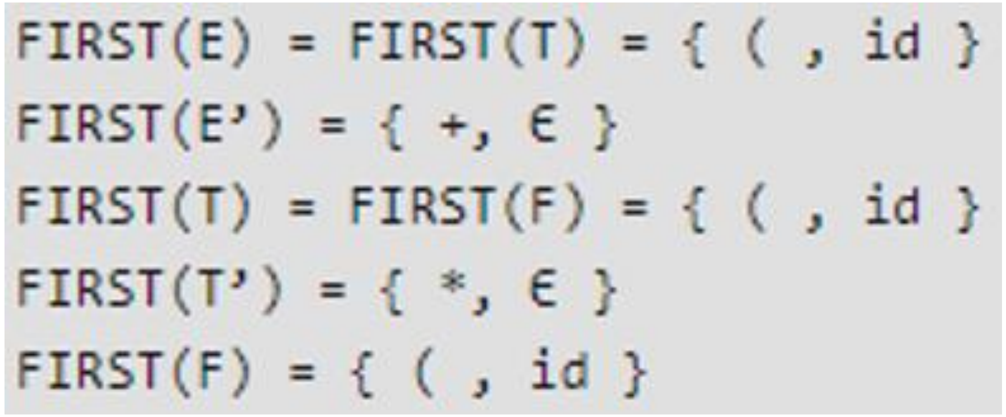
First Set (Example 1)

Problem



```
E   -> TE’
E’  -> +T E’|Є
T   -> F T’
T’  -> *F T’ | Є
F   -> (E) | id
```

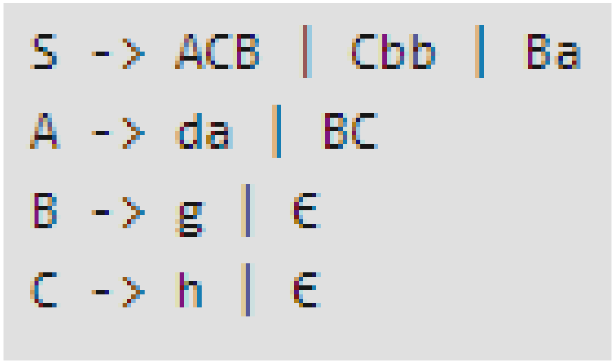
Solution



```
FIRST(E) = FIRST(T) = { ( , id }
FIRST(E’) = { +, Є }
FIRST(T) = FIRST(F) = { ( , id }
FIRST(T’) = { *, Є }
FIRST(F) = { ( , id }
```

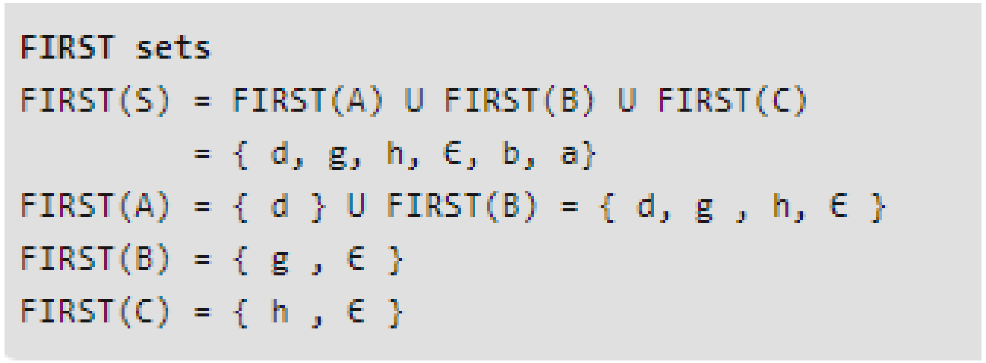
First Set (Example 2)

Problem



```
S -> ACB | Cbb | Ba
A -> da | BC
B -> g | ∈
C -> h | ∈
```

Solution



```
FIRST sets
FIRST(S) = FIRST(A) U FIRST(B) U FIRST(C)
         = { d, g, h, ∈, b, a}
FIRST(A) = { d } U FIRST(B) = { d, g , h, ∈ }
FIRST(B) = { g , ∈ }
FIRST(C) = { h , ∈ }
```

Follow Set

Rules

- Follow should be look for right side of anything
- Follow always starts with $
- **Follow(X)** to be the set of terminals that can appear immediately to the right of Non-Terminal X in some sentential form.
- FOLLOW (S) = { S } // where S is the starting Non-Terminal
- If A -> pBq is a production, where p, B and q are any grammar symbols, then everything in FIRST (q) except ε is in FOLLOW (B)
- If A->pB is a production, then everything in FOLLOW(A) is in FOLLOW (B)
- If A->pBq is a production and FIRST(q) contains ε, then FOLLOW (B) contains { FIRST(q) - ε} U FOLLOW (A)

Follow Set

Rules

Apply the following rules:

1. If \$ is the input end-marker, and S is the start symbol, $\$ \in \text{FOLLOW}(S)$.
2. If there is a production, $A \rightarrow \alpha B \beta$, then $(\text{FIRST}(\beta) - \varepsilon) \subseteq \text{FOLLOW}(B)$.
3. If there is a production, $A \rightarrow \alpha B$, or a production $A \rightarrow \alpha B \beta$, where $\varepsilon \in \text{FIRST}(\beta)$, then $\text{FOLLOW}(A) \subseteq \text{FOLLOW}(B)$.

Note that unlike the computation of FIRST sets for non-terminals, where the focus is on*what a non-terminal generates*, the computation of FOLLOW sets depends upon *where the non-terminal appears on the RHS of a production*

Follow Set (Case 1-a)

- Follow means something right behind of it.
- Follow means the next one
- If the next of a thing (whos Follow should be calculated) **terminal**/nonterminal then we must find the 'FIRST' of that terminal/nonterminal
- That particular 'FIRST' would be the designated 'FOLLOW' of the things (whos Follow should be calculated)

Follow Set (Case 1-b)

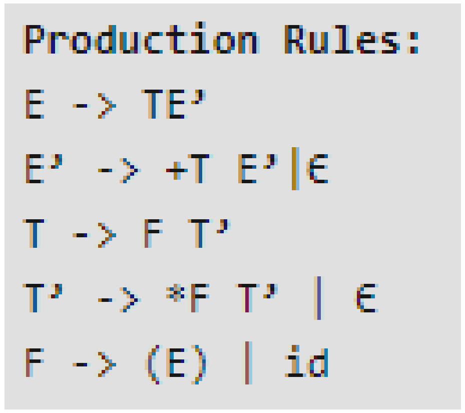
- Follow means something right behind of it.
- Follow means the next one
- If the next of a thing (whos Follow should be calculated) terminal/nonterminal then we must find the ‘FIRST’ of that terminal/nonterminal
- That particular ‘FIRST’ would be the designated ‘FOLLOW’ of the things (whos Follow should be calculated)

Follow Set (Case 2)

- We never write epsilon (ε) in ‘FOLLOW’
- If we do not have anything on right side
- That is, if we do not have an ‘FOLLOW’ then we will take the ‘FOLLOW’ (all FOLLOW) of its parent (non-terminal) (from which the production came)

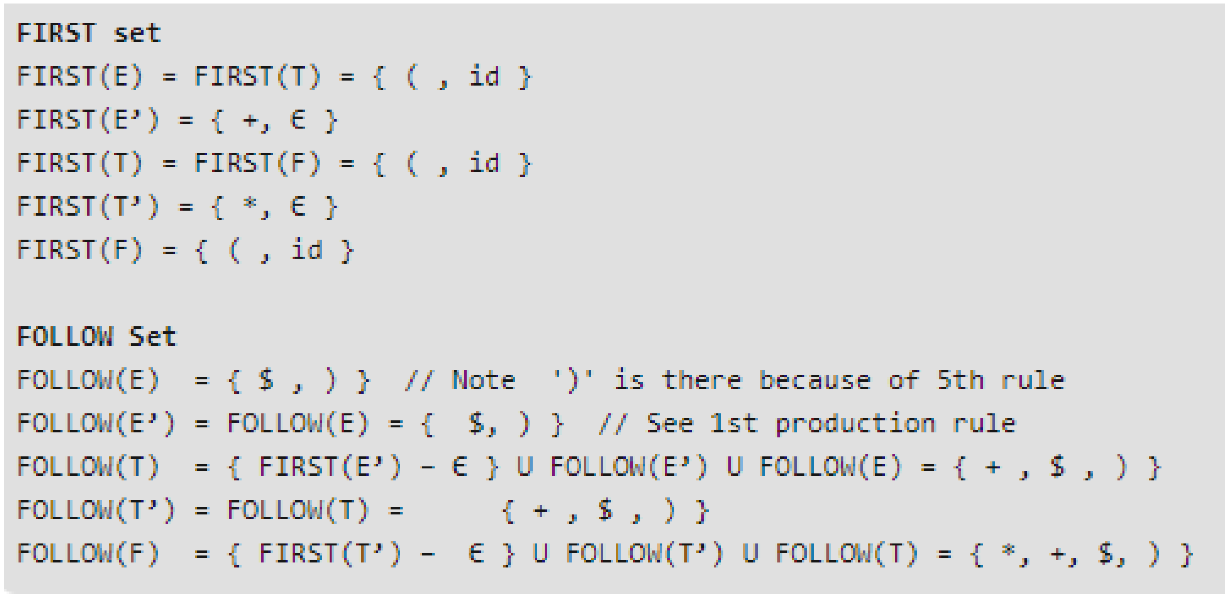
Follow Set (Example 1)

Problem



```
Production Rules:
E -> TE’
E’ -> +T E’|Є
T -> F T’
T’ -> *F T’ | Є
F -> (E) | id
```

Solution

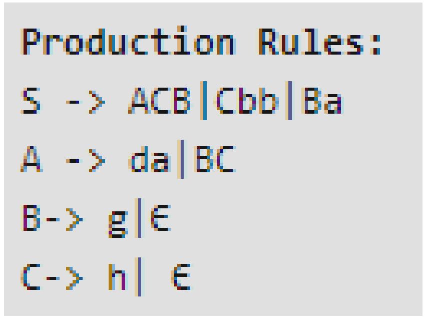


```
FIRST set
FIRST(E) = FIRST(T) = { ( , id }
FIRST(E’) = { +, Є }
FIRST(T) = FIRST(F) = { ( , id }
FIRST(T’) = { *, Є }
FIRST(F) = { ( , id }

FOLLOW Set
FOLLOW(E)  = { $ , ) }  // Note  ')' is there because of 5th rule
FOLLOW(E’) = FOLLOW(E) = {  $, ) }  // See 1st production rule
FOLLOW(T)  = { FIRST(E’) - Є } U FOLLOW(E’) U FOLLOW(E) = { + , $ , ) }
FOLLOW(T’) = FOLLOW(T) =      { + , $ , ) }
FOLLOW(F)  = { FIRST(T’) -  Є } U FOLLOW(T’) U FOLLOW(T) = { *, +, $, ) }
```

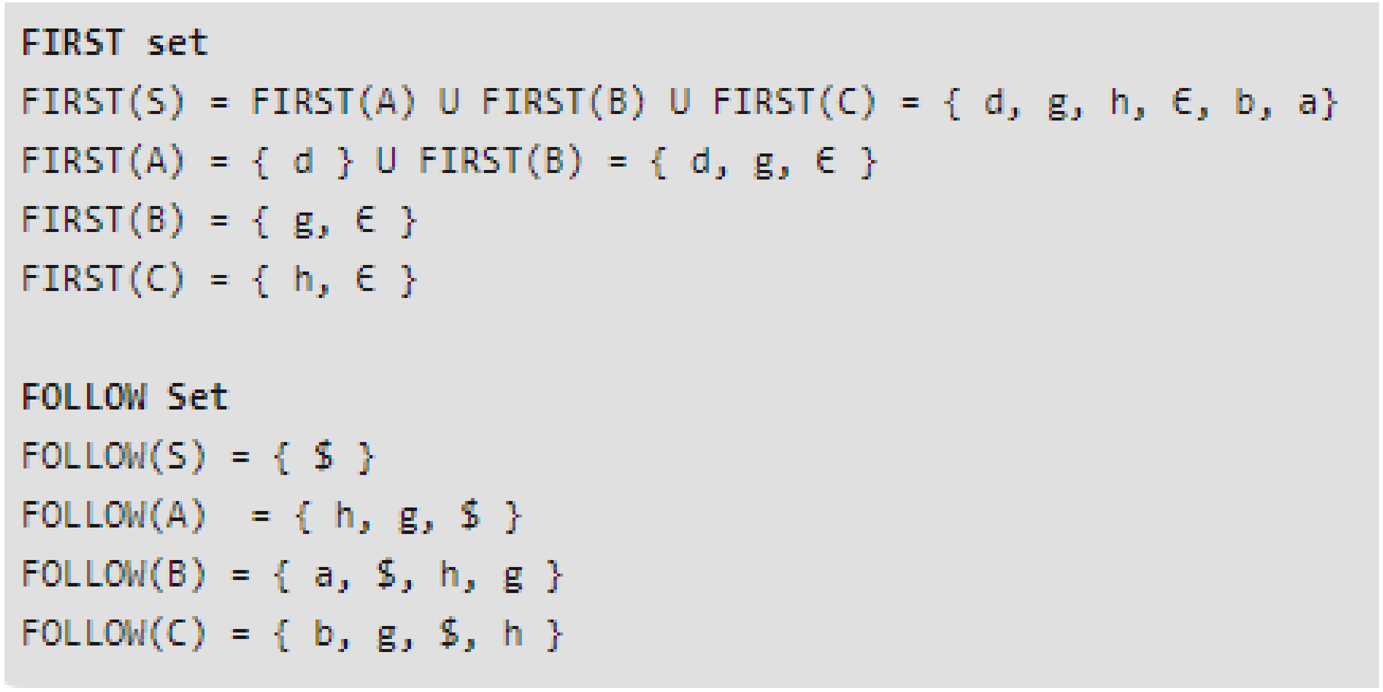
Follow Set (Example 2)

Problem



```
Production Rules:
S -> ACB|Cbb|Ba
A -> da|BC
B-> g|Є
C-> h| Є
```

Solution



```
FIRST set
FIRST(S) = FIRST(A) U FIRST(B) U FIRST(C) = { d, g, h, Є, b, a}
FIRST(A) = { d } U FIRST(B) = { d, g, Є }
FIRST(B) = { g, Є }
FIRST(C) = { h, Є }

FOLLOW Set
FOLLOW(S) = { $ }
FOLLOW(A)  = { h, g, $ }
FOLLOW(B) = { a, $, h, g }
FOLLOW(C) = { b, g, $, h }
```

First and Follow Set

Example

Grammar	First	Follow
S->ABCDE	{a, b, c}	{ $ }
A-a/epsilon	{a, epsilon}	{b, c}
B->b/epsilon	{b, epsilon}	{c}
C->c	{c}	{d, e, $}
D->d/epsilon	{d, epsilon}	{e, $ }
E->e/epsilon	{e, epsilon}	{$}

Lecture References

- Online Tool:

http://jsmachines.sourceforge.net/machines/ll1.html

- Online Tutorial

https://www.geeksforgeeks.org/why-first-and-follow-in-compiler-design/

- Maynooth University Material

http://www.cs.nuim.ie/~jpower/Courses/Previous/parsing/node48.html

- StackOverflow Explanation

https://stackoverflow.com/questions/3720901/what-is-the-precise-definition-of-a-lookahead-set

References/ Books

- 1. Compilers-Principles, techniques and tools (2nd Edition) V. Aho, Sethi and D. Ullman
- 2. Principles of Compiler Design (2nd Revised Edition 2009) A. A. Puntambekar
- 3. Basics of Compiler Design Torben Mogensen