Direct RE to DFA Construction

Prepared By: Sukarna Sarker

DFA (Deterministic Finite Automata)

- A deterministic finite automata(DFA) is a special case of NFA where:
  1. There are no moves on input ε
  2. for each state, and for each input symbol there is only one state

Prepared By: Sukarna Sarker

Direct Regular Expression to DFA

Input: A regular expression r
Output: A DFA D that recognizes L(r)

Step 1: Construct Augmented Regular Expression from the given expression.

Step 2: Construct the syntax tree for the augmented regular expression.

Step 3: Construct the functions nullable,firstpos,lastpos and followpos for syntax tree.

Step 4: Construct Dstates -> set if states of DFA D
Construct Dtrans -> transitions functions for DFA D

Step 5: Draw the DFA for the transition trable.

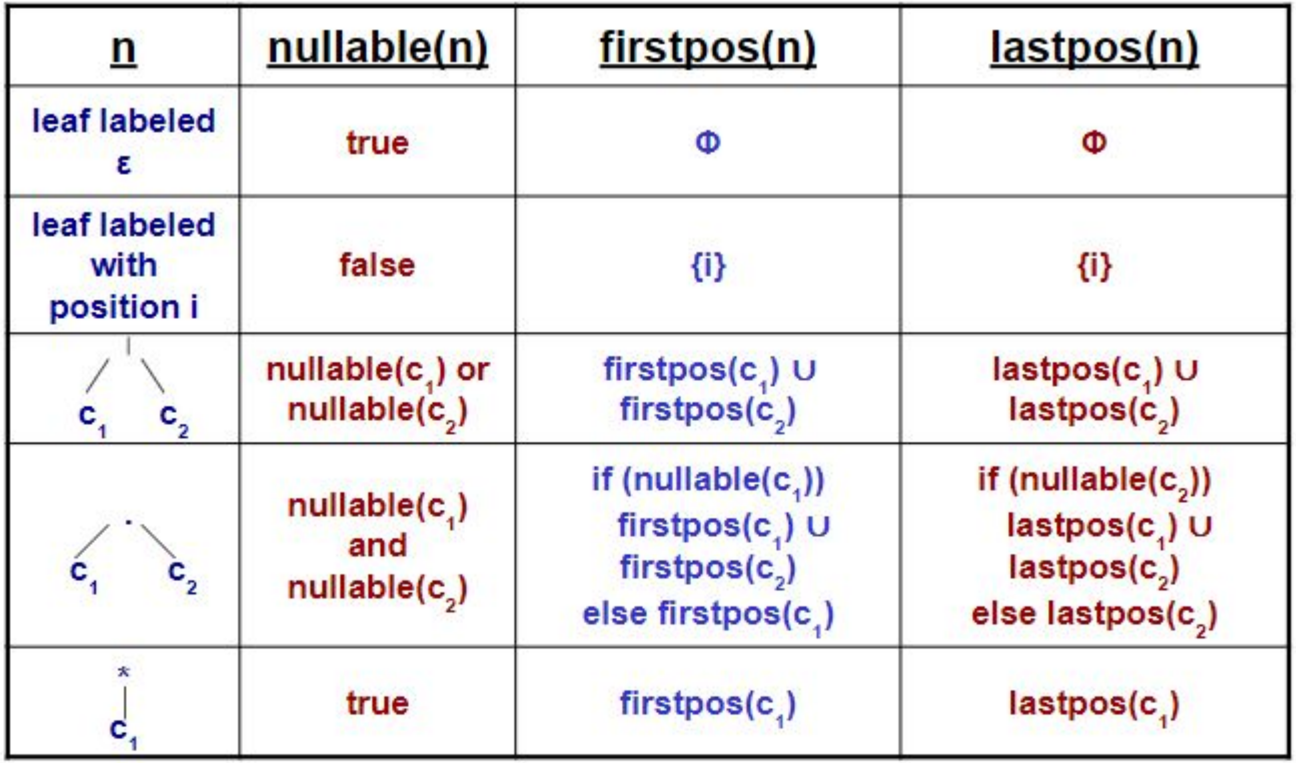
Prepared By: Sukarna Sarker

In the Syntax tree:

- The Leaf node contains alphabet symbols or ε.
- Each non-ε leaf is associated with a unique number- position of the leaf and position of the symbol
- The Internal node contains operators(also known as cat-node, or-node or star-node)

Prepared By: Sukarna Sarker

How to evaluate firstpos, lastpos, nullable



n	nullable(n)	firstpos(n)	lastpos(n)
leaf labeled ε	true	Φ	Φ
leaf labeled with position i	false	{i}	{i}
| c_1 c_2	nullable(c_1) or nullable(c_2)	firstpos(c_1) ∪ firstpos(c_2)	lastpos(c_1) ∪ lastpos(c_2)
· c_1 c_2	nullable(c_1) and nullable(c_2)	if (nullable(c_1)) firstpos(c_1) ∪ firstpos(c_2) else firstpos(c_1)	if (nullable(c_2)) lastpos(c_1) ∪ lastpos(c_2) else lastpos(c_2)
* c_1	true	firstpos(c_1)	lastpos(c_1)

Prepared By: Sukarna Sarker

Example 01

Prepared By: Sukarna Sarker

Step 1: Construct an Augmented regular expression

Suppose, given regular expression is:

$$r = (a \mid b)^{*} abb$$

Concat a unique right end marker # to a regular expression r:

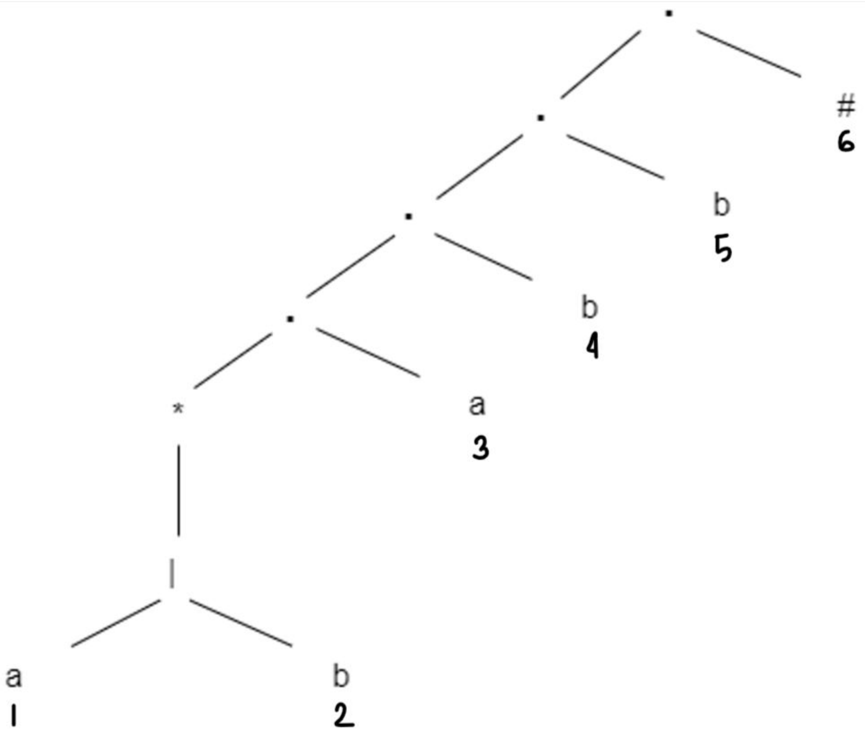
$$r\# = (a \mid b)^{*} abb\#$$

Step 2: Construct the syntax tree for the augmented regular expression.

Syntax tree for r# = (a | b)* abb#

Here,

1. **a**, **b**, **#** are in **leaf nodes**
2. **|**, ***** , **.** are in **internal nodes**
3. Each leaf node has **unique numbers.**



Prepared By: Sukarna Sarker

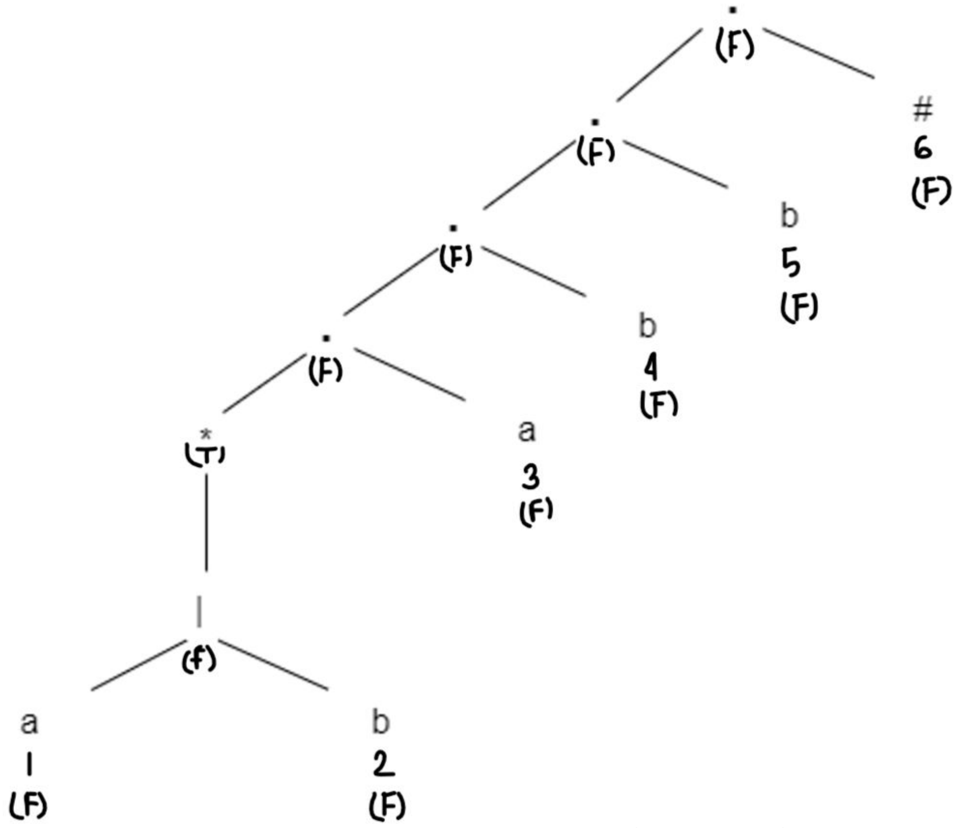
Step 3: Construct the functions nullable,firstpos,lastpos and followpos

1. Nullable

True = Nullable(can produce empty string ε)

False = Not Nullable(can not produce empty string ε)

- If n is a leaf labeled by ε then nullable (n) = true
- If n is a leaf labeled with position i then nullable (n) = false
- If n is an or-node (|) with children c1 and c2 then nullable(n) = nullable(c1) or nullable(c2)
- If n is an cat-node (·) with children c1 and c2 then nullable(n) = nullable(c1) and nullable(c2)
- If n is an star-node (*) with children c1 then nullable (n) = true

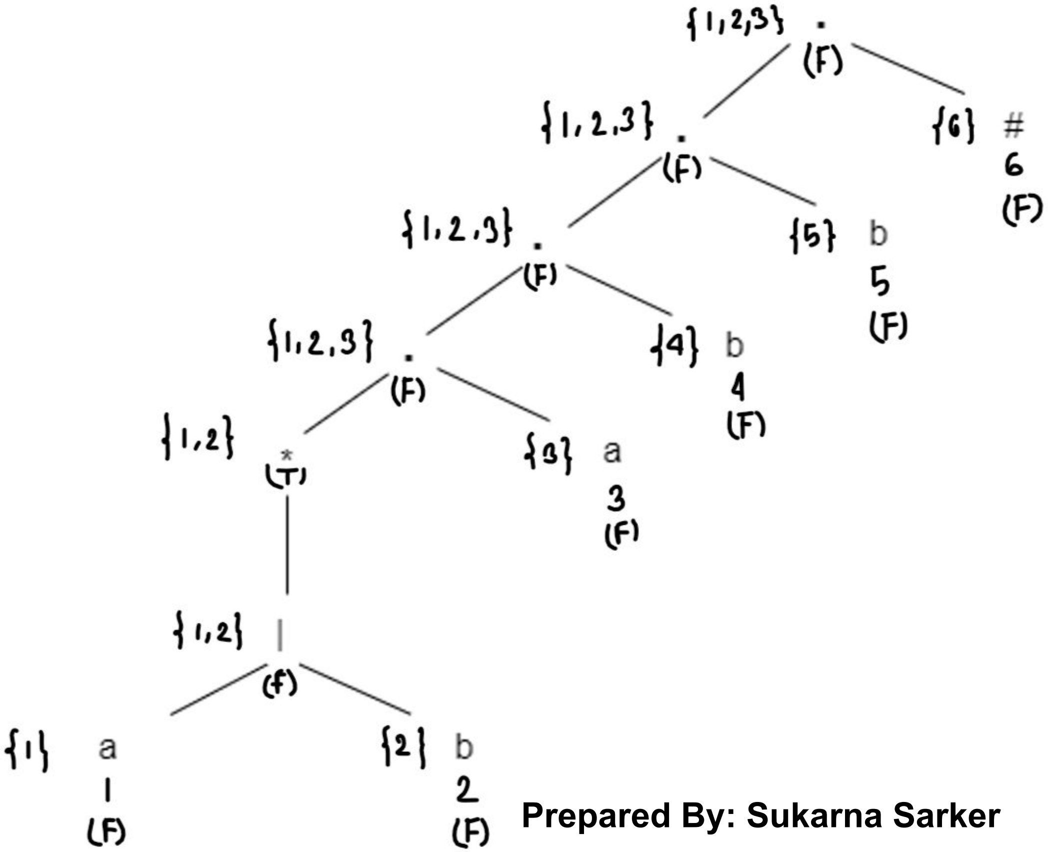


Prepared By: Sukarna Sarker

Step 3: Construct the functions nullable,firstpos,lastpos and followpos

2. Firstpos(n)

- If n is a leaf labeled by ε then firstpos (n) = ∅
- If n is a leaf labeled with position i firstpos (n) = {i}
- If n is an or-node (|) with children c1 and c2 then firstpos(n) = firstpos(c1) ∪ firstpos(c2)
- If n is a cat-node (·) with children c1 and c2 then
```
firstpos(n) = If nullable(c1)
                  then
                  firstpos(c1) ∪ firstpos (c2)
                  else
                  firstpos(c1)
```
- If n is an star-node (*) with children c1 firstpos (n) = firstpos(c1)

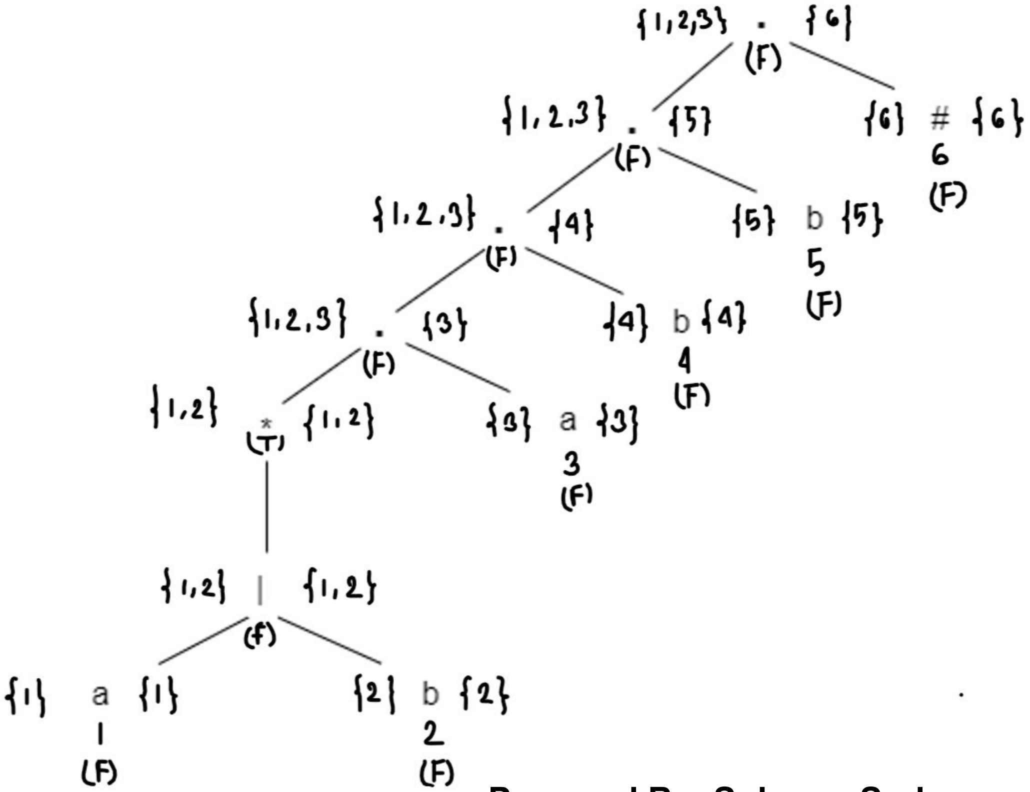


Prepared By: Sukarna Sarker

Step 3: Construct the functions nullable,firstpos,lastpos and followpos

3. Lastpos(n)

- If n is a leaf labeled by ε then
  lastpos (n) = ∅
- If n is a leaf labeled with position *i*
  lastpos (n) = {i}
- If n is an or-node (|) with children c1 and c2
  lastpos (n) = lastpos(c1) ∪ lastpos (c2)
- If n is an cat-node (·) with children c1 and c2
  lastpos(n) = If nullable (c2)
  then
  lastpos(c1) ∪ lastpos (c2)
  else
  lastpos(c2)
- If n is an star-node (*) with children c1
  lastpos (n) = lastpos(c1)



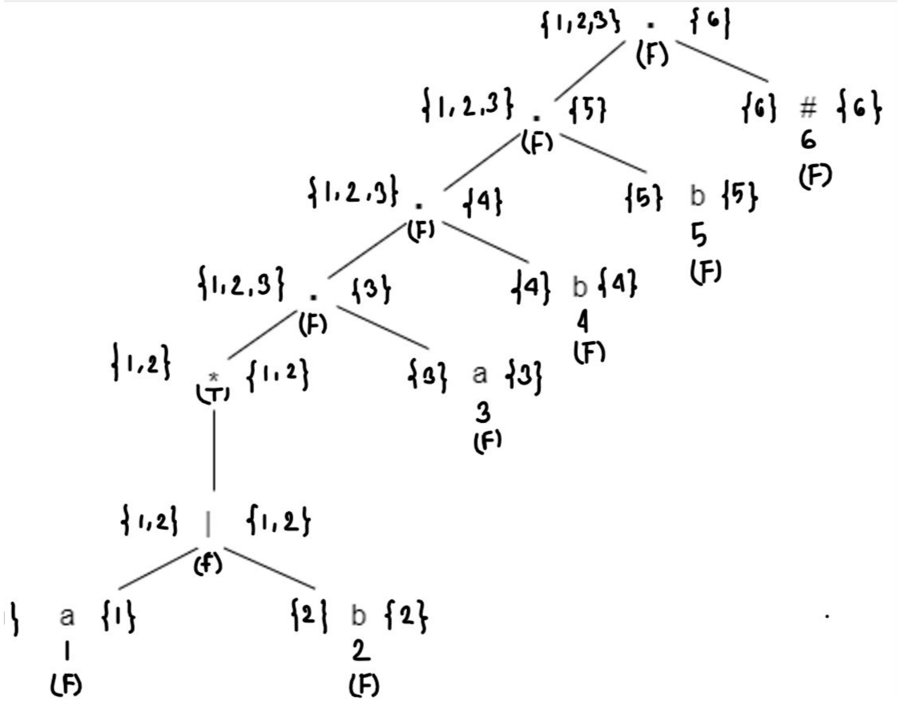
Prepared By: Sukarna Sarker

Step 3: Construct the functions nullable,firstpos,lastpos and followpos

4. Followpos(n)

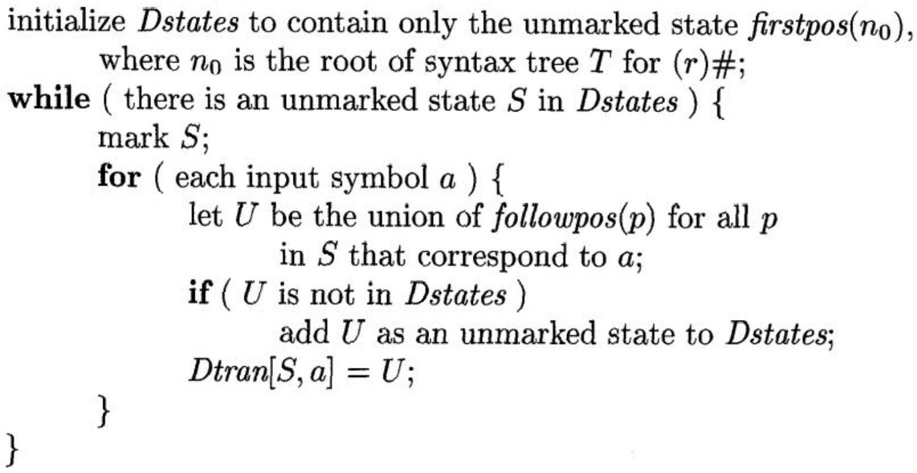
- If *n* is a cat-node with left child *c1* and right child *c2* and *i* is a position in lastpos (*c1*), then all positions in firstpos(*c2*) are in followpos(*i*)
- If *n* is a star node, and *i* is a position in lastpos(*n*), then all positions in firstpos(*n*) are in followpos(*i*)

Node	Followpos
1	{1,2,3}
2	{1,2,3}
3	{4}
4	{5}
5	{6}
6	{}



Prepared By: Sukarna Sarker

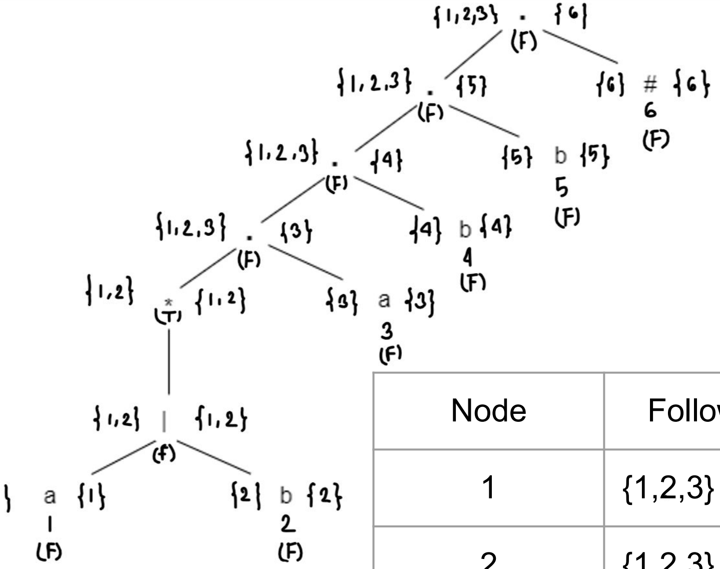
Step 4: Construct Dstates,Construct Dtrans



```
initialize $Dstates$ to contain only the unmarked state $firstpos(n_0)$,
        where $n_0$ is the root of syntax tree $T$ for $(r)\#$;
while ( there is an unmarked state $S$ in $Dstates$ ) {
        mark $S$;
        for ( each input symbol $a$ ) {
                let $U$ be the union of $followpos(p)$ for all $p$
                        in $S$ that correspond to $a$;
                if ( $U$ is not in $Dstates$ )
                        add $U$ as an unmarked state to $Dstates$;
                $Dtran[S, a] = U$;
        }
}
```

State	a	b
A		

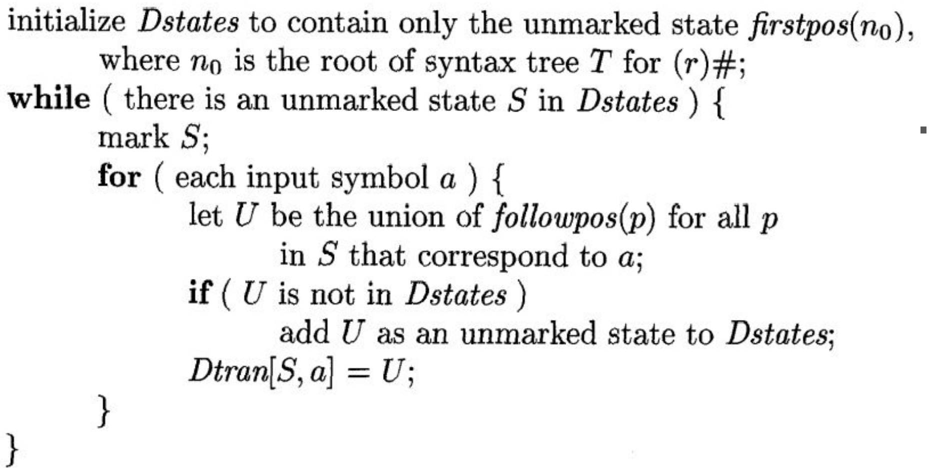
firstpos(n0) = {1,2,3} ->A



Node	Followpos
1	{1,2,3}
2	{1,2,3}
3	{4}
4	{5}
5	{6}
6	{}

Prepared By: Sukarna Sarker

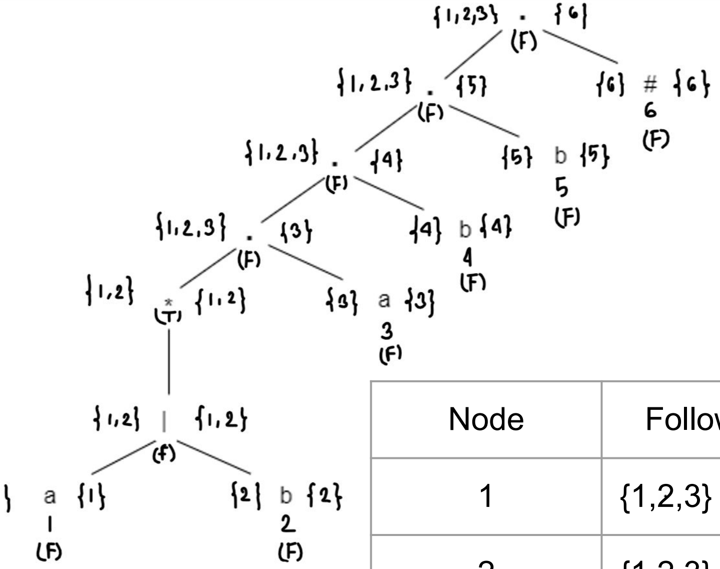
Step 4: Construct Dstates,Construct Dtrans



```
initialize $Dstates$ to contain only the unmarked state $firstpos(n_0)$,
        where $n_0$ is the root of syntax tree $T$ for $(r)\#$;
while ( there is an unmarked state $S$ in $Dstates$ ) {
        mark $S$;
        for ( each input symbol $a$ ) {
                let $U$ be the union of $followpos(p)$ for all $p$
                        in $S$ that correspond to $a$;
                if ( $U$ is not in $Dstates$ )
                        add $U$ as an unmarked state to $Dstates$;
                $Dtran[S, a] = U$;
        }
}
```

State	a	b
A	B	
B		

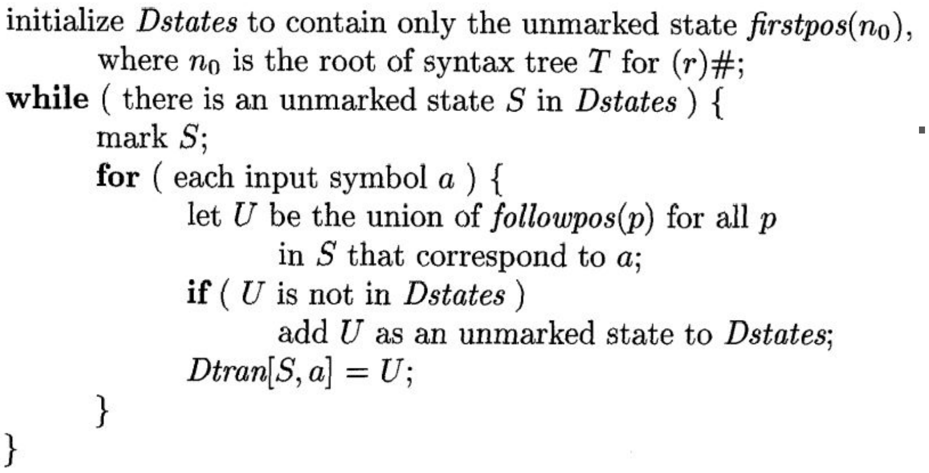
Dtran[A,a] = followpos(1) U followpos(3)
= {1,2,3} U {4}
= {1,2,3,4} -> B



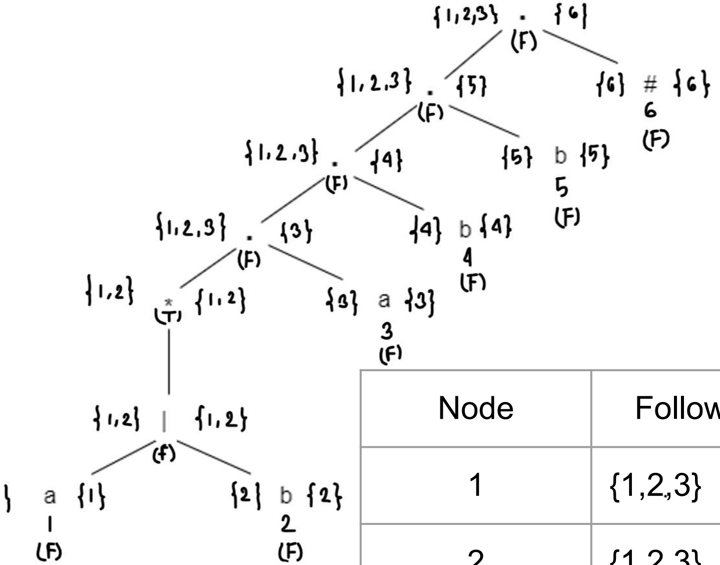
Node	Followpos
1	{1,2,3}
2	{1,2,3}
3	{4}
4	{5}
5	{6}
6	{}

Prepared By: Sukarna Sarker

Step 4: Construct Dstates,Construct Dtrans



```
initialize $Dstates$ to contain only the unmarked state $firstpos(n_0)$,
        where $n_0$ is the root of syntax tree $T$ for $(r)\#$;
while ( there is an unmarked state $S$ in $Dstates$ ) {
        mark $S$;
        for ( each input symbol $a$ ) {
                let $U$ be the union of $followpos(p)$ for all $p$
                        in $S$ that correspond to $a$;
                if ( $U$ is not in $Dstates$ )
                        add $U$ as an unmarked state to $Dstates$;
                $Dtran[S, a] = U$;
        }
}
```



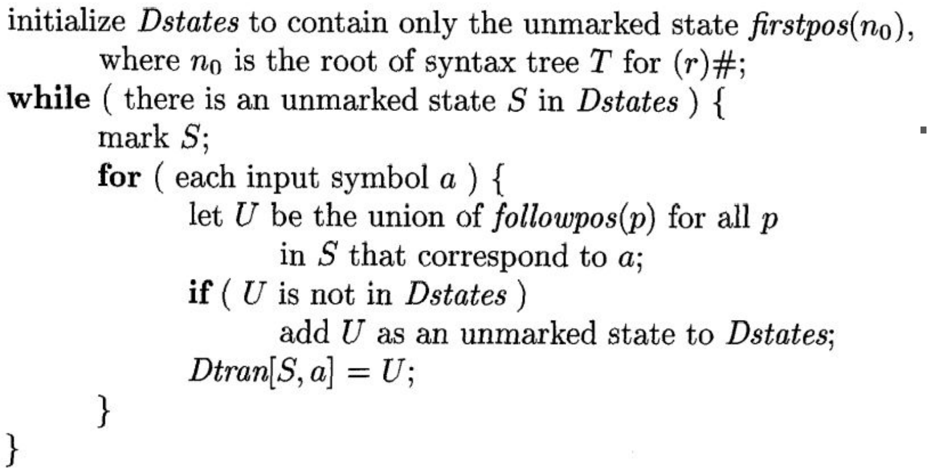
State	a	b
A	B	A
B		

Dtran[A,b] = followpos(2)
= {1,2,3} -> A

Node	Followpos
1	{1,2,3}
2	{1,2,3}
3	{4}
4	{5}
5	{6}
6	{}

Prepared By: Sukarna Sarker

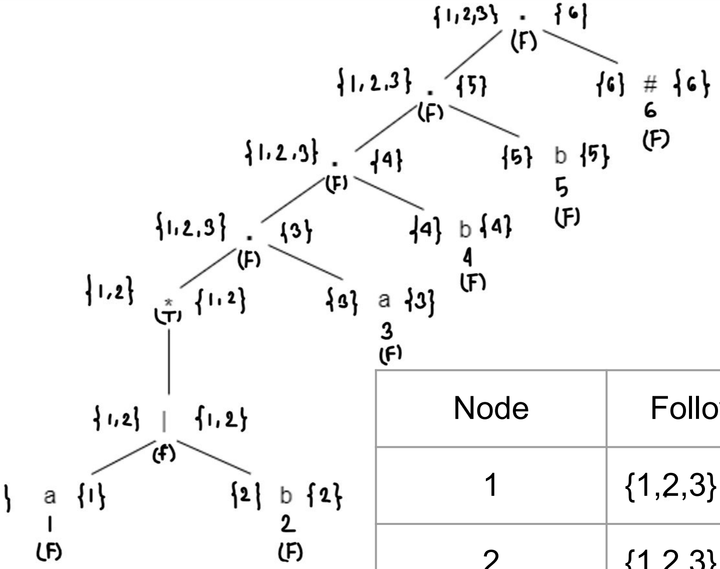
Step 4: Construct Dstates,Construct Dtrans



```
initialize $Dstates$ to contain only the unmarked state $firstpos(n_0)$,
        where $n_0$ is the root of syntax tree $T$ for $(r)\#$;
while ( there is an unmarked state $S$ in $Dstates$ ) {
        mark $S$;
        for ( each input symbol $a$ ) {
                let $U$ be the union of $followpos(p)$ for all $p$
                        in $S$ that correspond to $a$;
                if ( $U$ is not in $Dstates$ )
                        add $U$ as an unmarked state to $Dstates$;
                $Dtran[S, a] = U$;
        }
}
```

State	a	b
A	B	A
B	B	

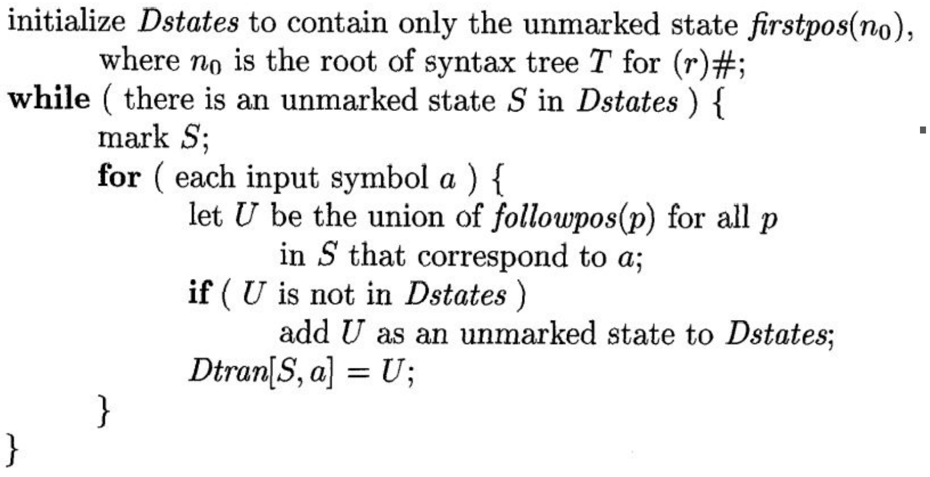
Dtran[B,a] = followpos(1) U followpos(3)
= {1,2,3} U {4}
= {1,2,3,4} -> B



Node	Followpos
1	{1,2,3}
2	{1,2,3}
3	{4}
4	{5}
5	{6}
6	{}

Prepared By: Sukarna Sarker

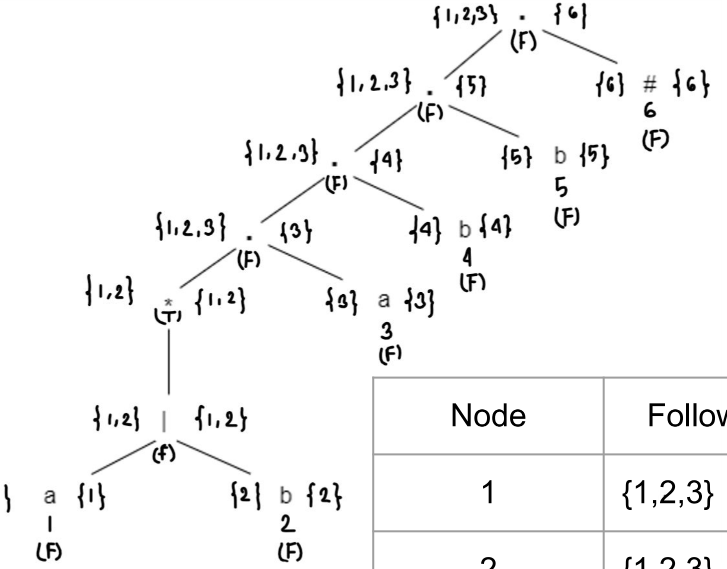
Step 4: Construct Dstates,Construct Dtrans



```
initialize $Dstates$ to contain only the unmarked state $firstpos(n_0)$,
        where $n_0$ is the root of syntax tree $T$ for $(r)\#$;
while ( there is an unmarked state $S$ in $Dstates$ ) {
        mark $S$;
        for ( each input symbol $a$ ) {
                let $U$ be the union of $followpos(p)$ for all $p$
                        in $S$ that correspond to $a$;
                if ( $U$ is not in $Dstates$ )
                        add $U$ as an unmarked state to $Dstates$;
                $Dtran[S, a] = U$;
        }
}
```

State	a	b
A	B	A
B	B	C
C		

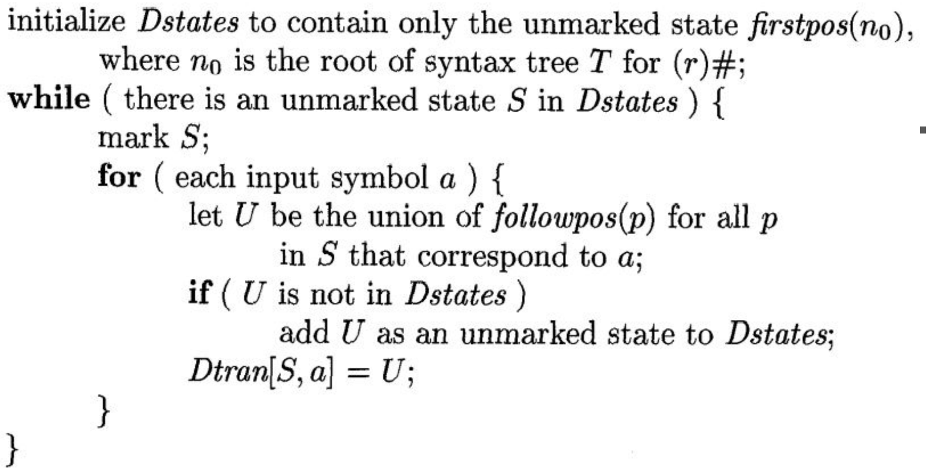
Dtran[B,b] = followpos(2) U followpos(4)
= {1,2,3} U {5}
= {1,2,3,5} -> C



Node	Followpos
1	{1,2,3}
2	{1,2,3}
3	{4}
4	{5}
5	{6}
6	{}

Prepared By: Sukarna Sarker

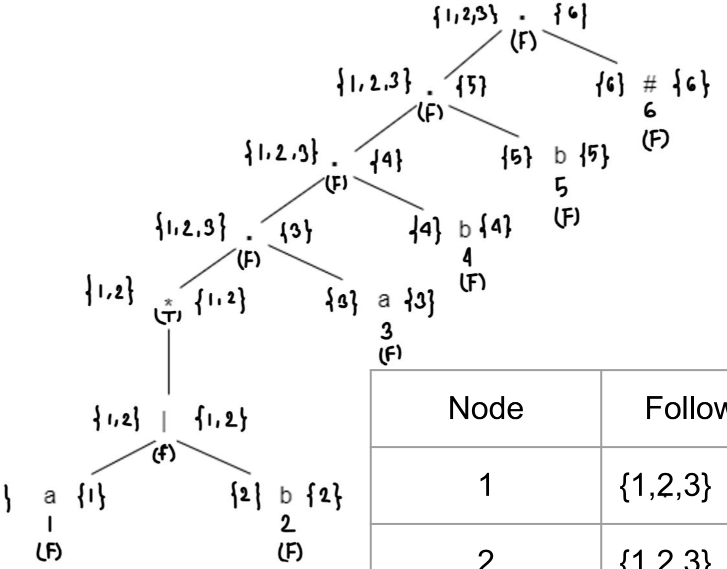
Step 4: Construct Dstates,Construct Dtrans



```
initialize $Dstates$ to contain only the unmarked state $firstpos(n_0)$,
        where $n_0$ is the root of syntax tree $T$ for $(r)\#$;
while ( there is an unmarked state $S$ in $Dstates$ ) {
        mark $S$;
        for ( each input symbol $a$ ) {
                let $U$ be the union of $followpos(p)$ for all $p$
                        in $S$ that correspond to $a$;
                if ( $U$ is not in $Dstates$ )
                        add $U$ as an unmarked state to $Dstates$;
                $Dtran[S, a] = U$;
        }
}
```

State	a	b
A	B	A
B	B	C
C	B	

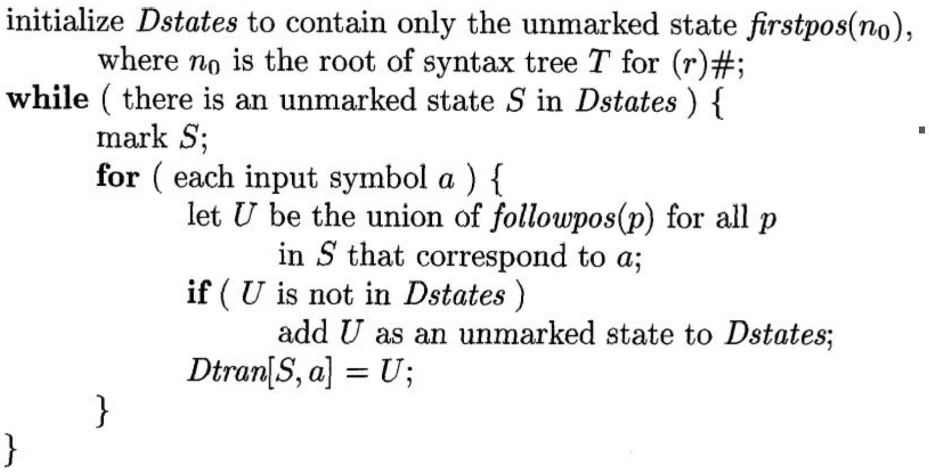
Dtran[C,a] = followpos(1) U followpos(3)
= {1,2,3} U {4}
= {1,2,3,4} -> B



Node	Followpos
1	{1,2,3}
2	{1,2,3}
3	{4}
4	{5}
5	{6}
6	{}

Prepared By: Sukarna Sarker

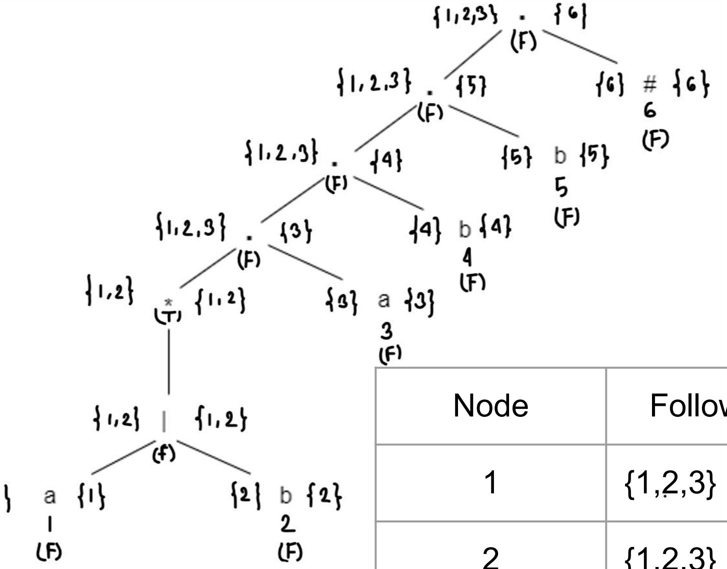
Step 4: Construct Dstates,Construct Dtrans



```
initialize $Dstates$ to contain only the unmarked state $firstpos(n_0)$,
        where $n_0$ is the root of syntax tree $T$ for $(r)\#$;
while ( there is an unmarked state $S$ in $Dstates$ ) {
        mark $S$;
        for ( each input symbol $a$ ) {
                let $U$ be the union of $followpos(p)$ for all $p$
                        in $S$ that correspond to $a$;
                if ( $U$ is not in $Dstates$ )
                        add $U$ as an unmarked state to $Dstates$;
                $Dtran[S, a] = U$;
        }
}
```

State	a	b
A	B	A
B	B	C
C	B	D

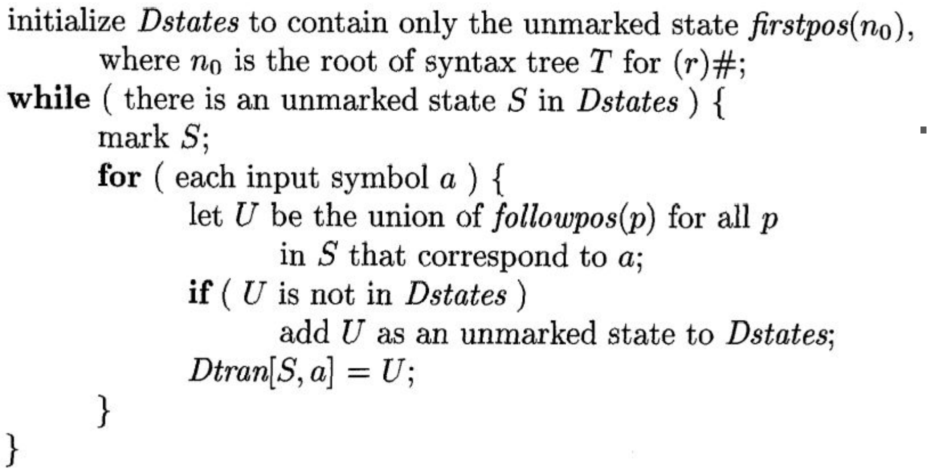
Dtran[C,b] = followpos(2) U followpos(5)
= {1,2,3} U {6}
= {1,2,3,6} -> D



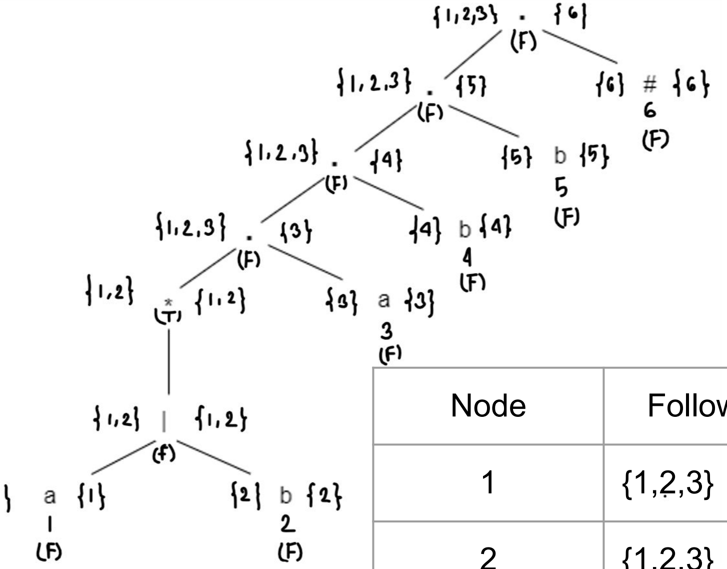
Node	Followpos
1	{1,2,3}
2	{1,2,3}
3	{4}
4	{5}
5	{6}
6	{}

Prepared By: Sukarna Sarker

Step 4: Construct Dstates,Construct Dtrans



```
initialize $Dstates$ to contain only the unmarked state $firstpos(n_0)$,
        where $n_0$ is the root of syntax tree $T$ for $(r)\#$;
while ( there is an unmarked state $S$ in $Dstates$ ) {
        mark $S$;
        for ( each input symbol $a$ ) {
                let $U$ be the union of $followpos(p)$ for all $p$
                        in $S$ that correspond to $a$;
                if ( $U$ is not in $Dstates$ )
                        add $U$ as an unmarked state to $Dstates$;
                $Dtran[S, a] = U$;
        }
}
```



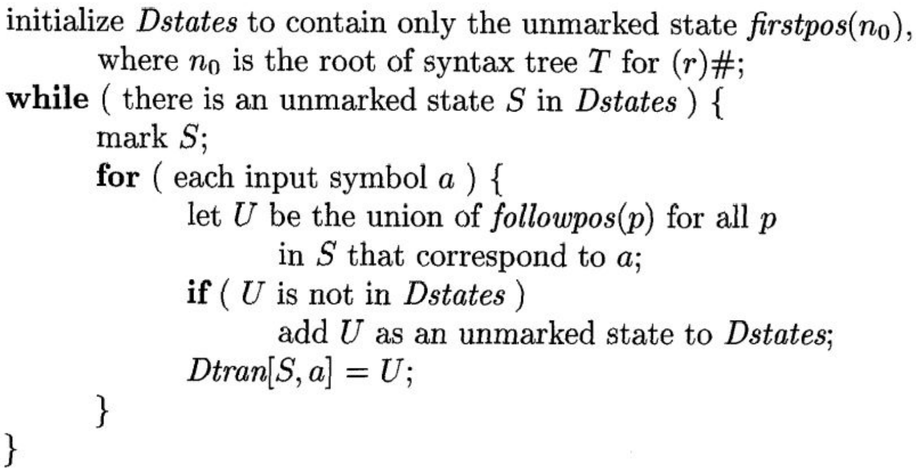
State	a	b
A	B	A
B	B	C
C	B	D
D	B	

Dtran[D,a] = followpos(1) U followpos(3)
= {1,2,3} U {4}
= {1,2,3,4} -> B

Node	Followpos
1	{1,2,3}
2	{1,2,3}
3	{4}
4	{5}
5	{6}
6	{}

Prepared By: Sukarna Sarker

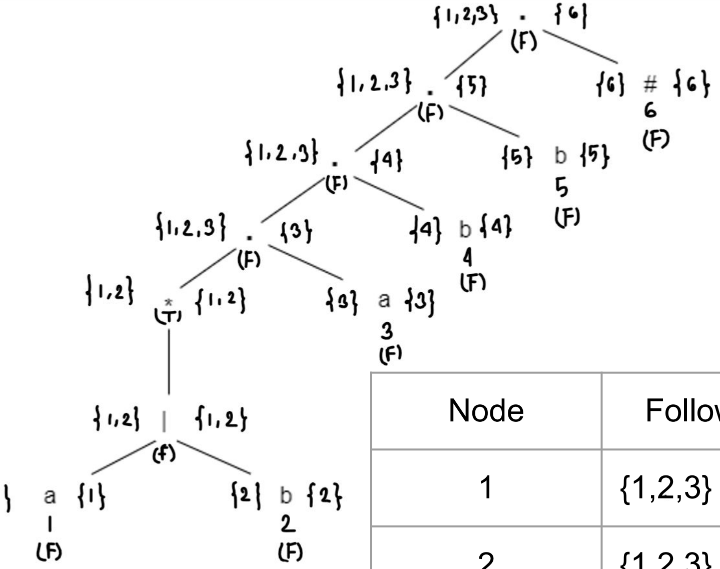
Step 4: Construct Dstates,Construct Dtrans



```
initialize $Dstates$ to contain only the unmarked state $firstpos(n_0)$,
        where $n_0$ is the root of syntax tree $T$ for $(r)\#$;
while ( there is an unmarked state $S$ in $Dstates$ ) {
        mark $S$;
        for ( each input symbol $a$ ) {
                let $U$ be the union of $followpos(p)$ for all $p$
                        in $S$ that correspond to $a$;
                if ( $U$ is not in $Dstates$ )
                        add $U$ as an unmarked state to $Dstates$;
                $Dtran[S, a] = U$;
        }
}
```

State	a	b
A	B	A
B	B	C
C	B	D
D	B	A

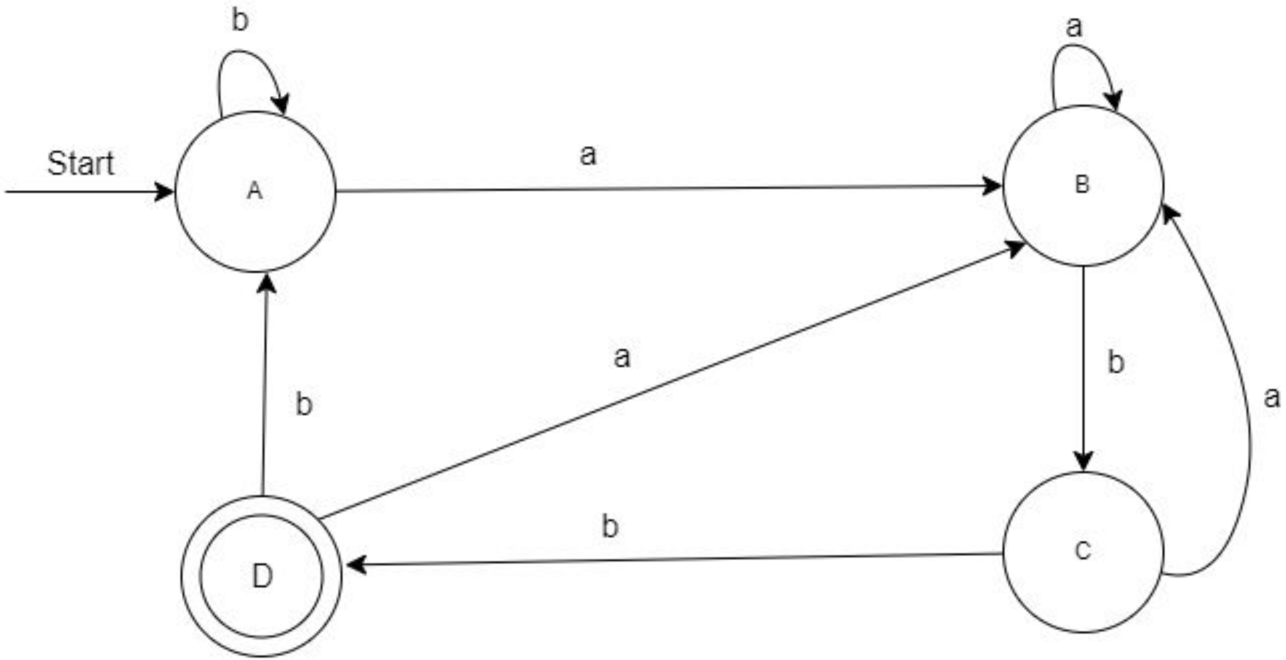
Dtran[D,b] = followpos(2)
= {1,2,3}   -> A



Node	Followpos
1	{1,2,3}
2	{1,2,3}
3	{4}
4	{5}
5	{6}
6	{}

Prepared By: Sukarna Sarker

Step 5: Draw the DFA for the transition trable.



Prepared By: Sukarna Sarker

Example 02

Prepared By: Sukarna Sarker

Step 1: Construct an Augmented regular expression

Suppose, given regular expression is:

$$r = (a+b)^* + (ac)^*$$

Concat a unique right end marker # to a regular expression r:

$$r\# = (a+b)^* + (ac)^*\#$$

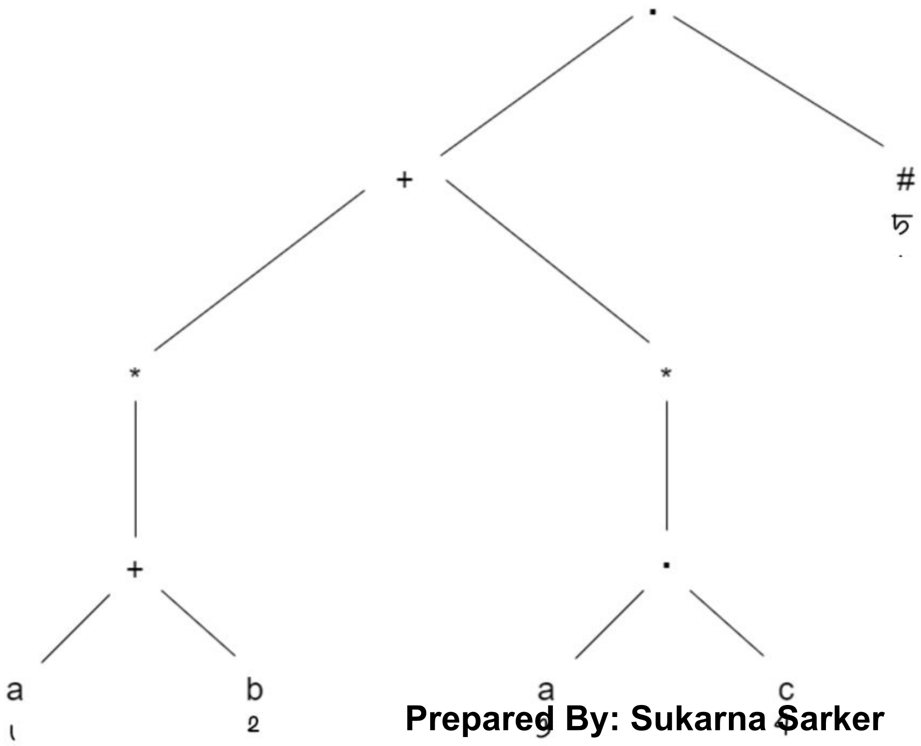
Prepared By: Sukarna Sarker

Step 2: Construct the syntax tree for the augmented regular expression.

Syntax tree for r# = (a+b)* + (ac)*#

Here,

1. **a**, **b**, **c, #** are in **leaf nodes**
2. **+**, ***** , **.** are in **internal nodes**
3. Each leaf node has **unique numbers.**



Prepared By: Sukarna Sarker

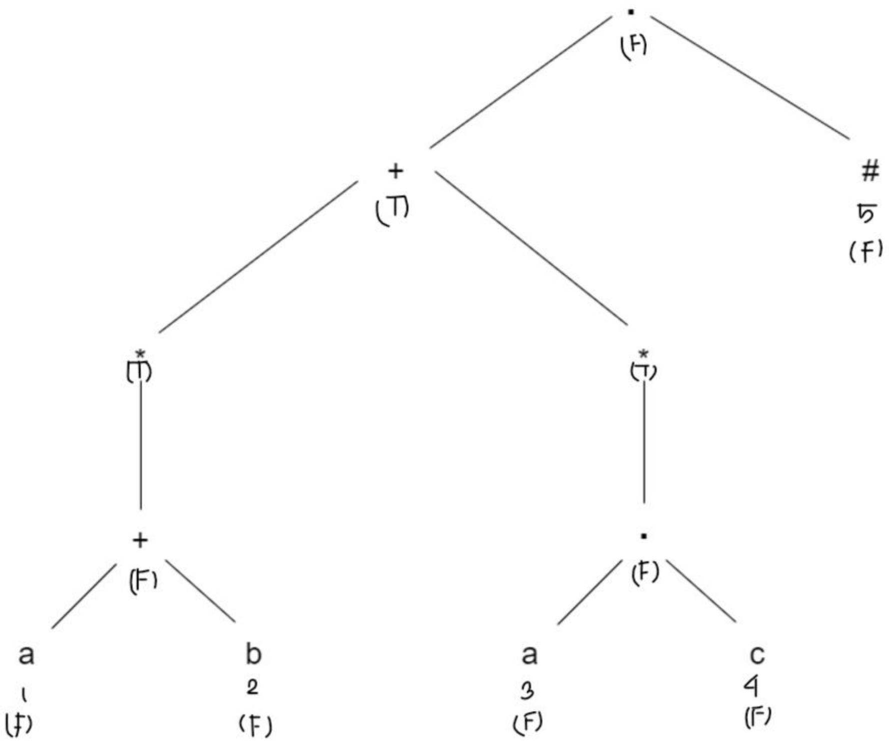
Step 3: Construct the functions nullable,firstpos,lastpos and followpos

1. Nullable

True = Nullable(can produce empty string ε)

False = Not Nullable(can not produce empty string ε)

- If n is a leaf labeled by ε then
  nullable (n) = true
- If n is a leaf labeled with position i then
  nullable (n) = false
- If n is an or-node (|) with children c1
  and c2 then
  nullable(n) = nullable(c1) or nullable(c2)
- If n is an cat-node (·) with children c1 and c2
  then nullable(n) = nullable(c1) and nullable(c2)
- If n is an star-node (*) with children c1 then
  nullable (n) = true

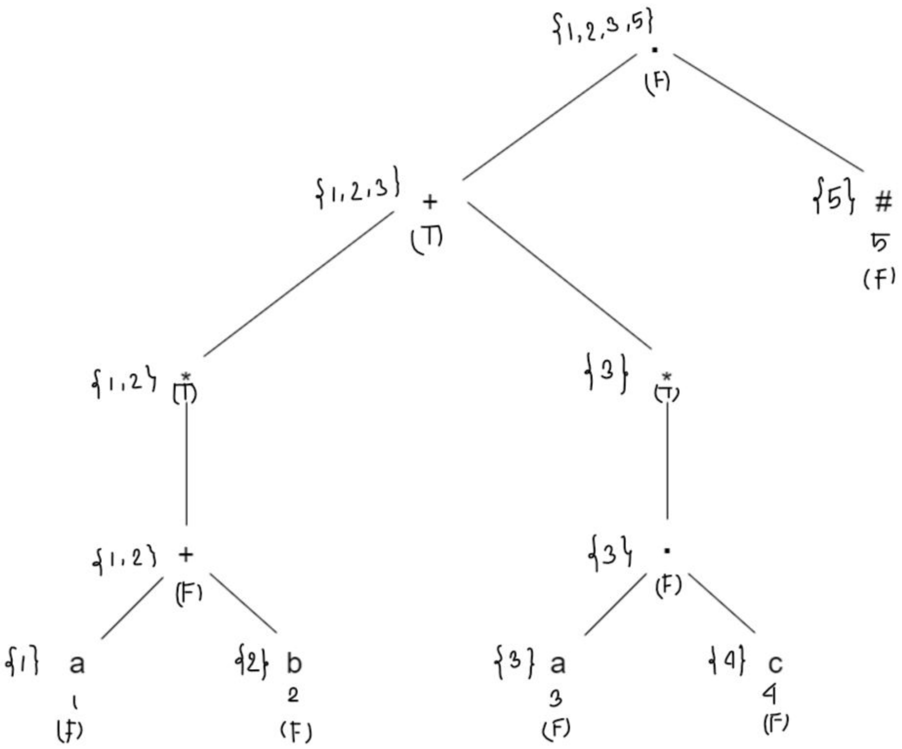


Prepared By: Sukarna Sarker

Step 3: Construct the functions nullable,firstpos,lastpos and followpos

2. Firstpos(n)

- If n is a leaf labeled by ε then
  firstpos (n) = ∅
- If n is a leaf labeled with position i
  firstpos (n) = {i}
- If n is an or-node (|) with children
  c1 and c2 then
  firstpos(n) = firstpos(c1) ∪ firstpos(c2)
- If n is a cat-node (·) with children c1
  and c2 then
  firstpos(n) = If nullable(c1)
  then
  firstpos(c1) ∪ firstpos (c2)
  else
  firstpos(c1)
- If n is an star-node (*) with children c1
  firstpos (n) = firstpos(c1)

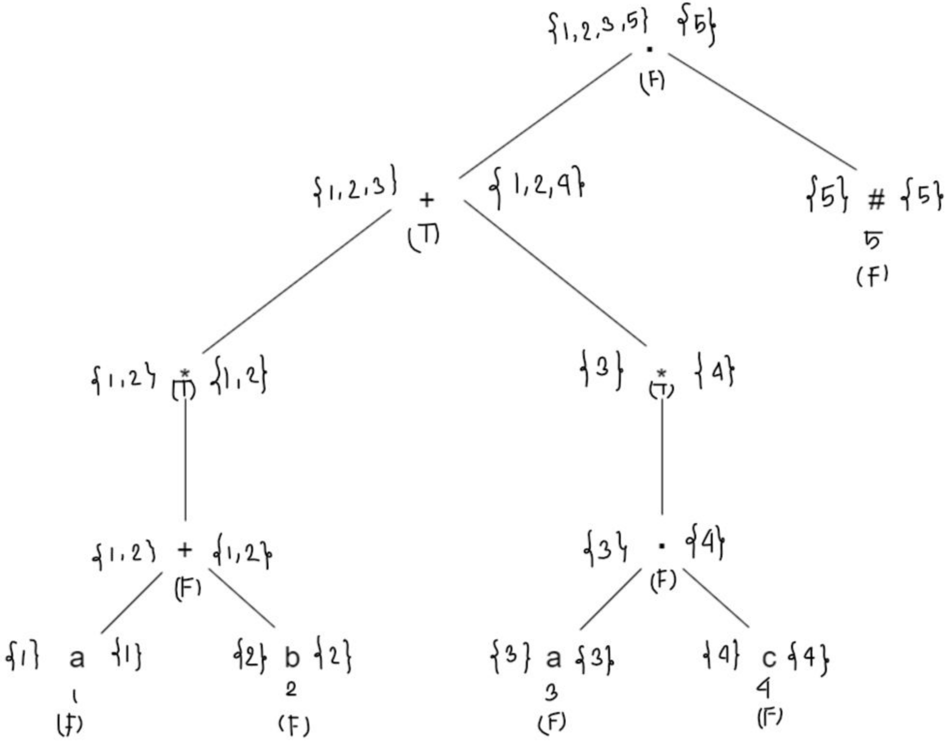


Prepared By: Sukarna Sarker

Step 3: Construct the functions nullable,firstpos,lastpos and followpos

3. Lastpos(n)

- If n is a leaf labeled by ε then
  lastpos (n) = ∅
- If n is a leaf labeled with position *i*
  lastpos (n) = {i}
- If n is an or-node (|) with children c1 and c2
  lastpos (n) = lastpos(c1) ∪ lastpos (c2)
- If n is an cat-node (·) with children c1 and c2
  lastpos(n) = If nullable (c2)
  then
  lastpos(c1) ∪ lastpos (c2)
  else
  lastpos(c2)
- If n is an star-node (*) with children c1
  lastpos (n) = lastpos(c1)



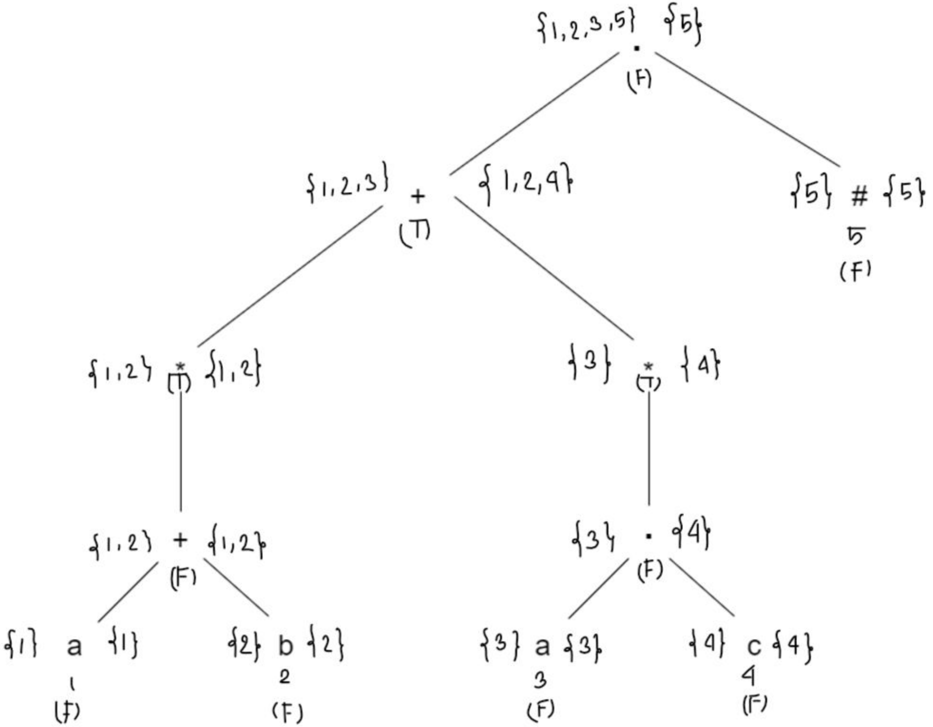
Prepared By: Sukarna Sarker

Step 3: Construct the functions nullable,firstpos,lastpos and followpos

4. Followpos(n)

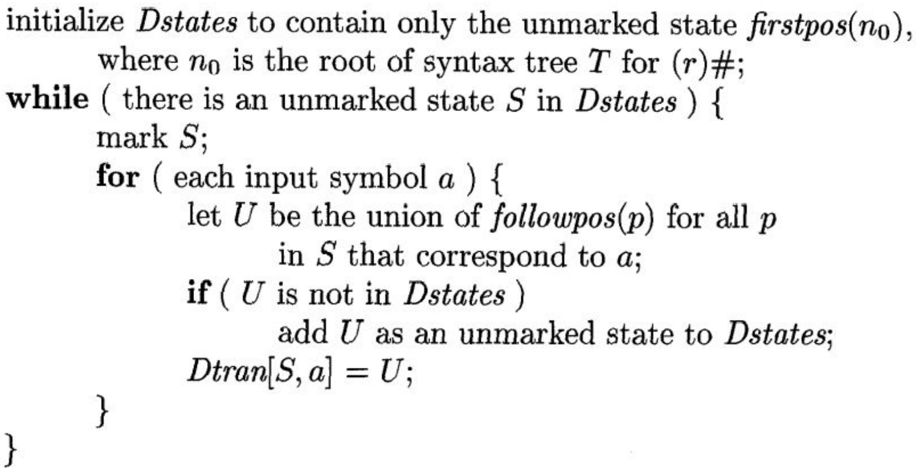
- If *n* is a cat-node with left child *c1* and right child *c2* and *i* is a position in lastpos (*c1*), then all positions in firstpos(*c2*) are in followpos(*i*)
- If *n* is a star node, and *i* is a position in lastpos(*n*), then all positions in firstpos(*n*) are in followpos(*i*)

Node	Followpos
1	{1,2,5}
2	{1,2,5}
3	{4}
4	{3,5}
5	{}



Prepared By: Sukarna Sarker

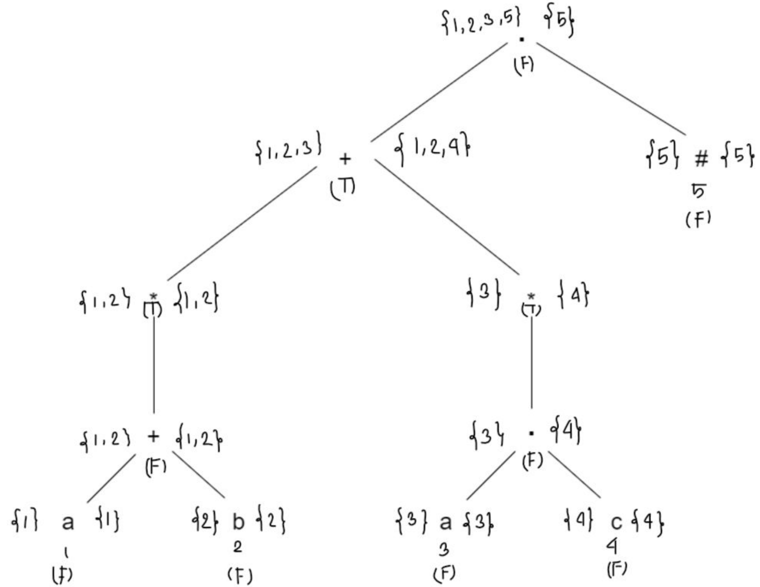
Step 4: Construct Dstates,Construct Dtrans



```
initialize $Dstates$ to contain only the unmarked state $firstpos(n_0)$,
        where $n_0$ is the root of syntax tree $T$ for $(r)\#$;
while ( there is an unmarked state $S$ in $Dstates$ ) {
        mark $S$;
        for ( each input symbol $a$ ) {
                let $U$ be the union of $followpos(p)$ for all $p$
                        in $S$ that correspond to $a$;
                if ( $U$ is not in $Dstates$ )
                        add $U$ as an unmarked state to $Dstates$;
                $Dtran[S, a] = U$;
        }
}
```

State	0	1
A		

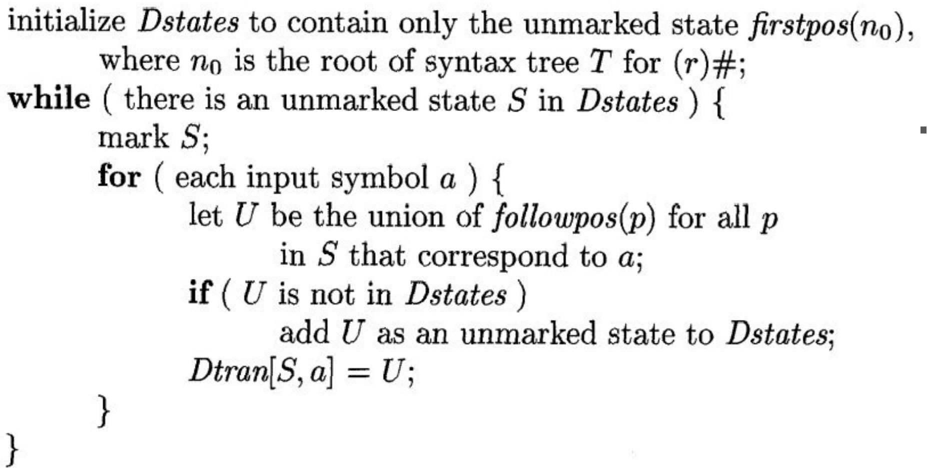
firstpos(n0) = {1,2,3,5} ->A



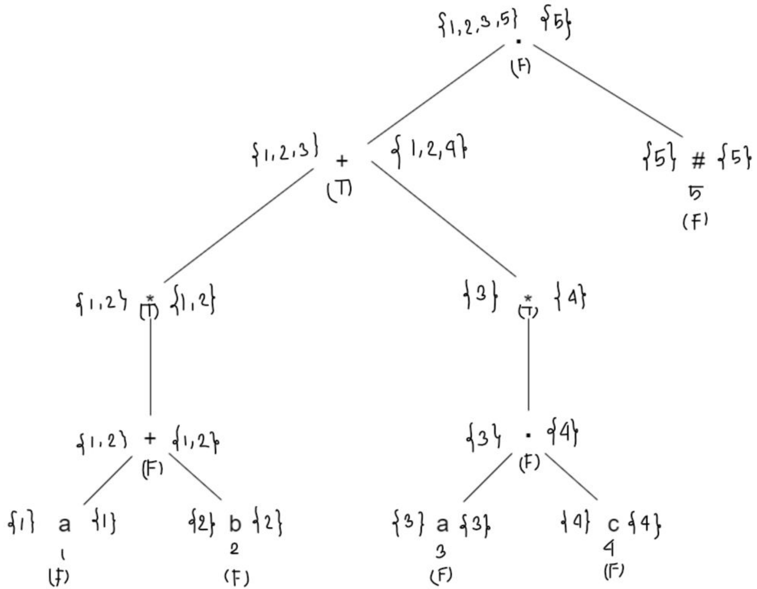
Node	Followpos	Node	Followpos
1	{1,2,5}	4	{3,5}
2	{1,2,5}	5	{}
3	{4}		

Prepared By: Sukarna Sarker

Step 4: Construct Dstates,Construct Dtrans



```
initialize $Dstates$ to contain only the unmarked state $firstpos(n_0)$,
        where $n_0$ is the root of syntax tree $T$ for $(r)\#$;
while ( there is an unmarked state $S$ in $Dstates$ ) {
        mark $S$;
        for ( each input symbol $a$ ) {
                let $U$ be the union of $followpos(p)$ for all $p$
                        in $S$ that correspond to $a$;
                if ( $U$ is not in $Dstates$ )
                        add $U$ as an unmarked state to $Dstates$;
                $Dtran[S, a] = U$;
        }
}
```



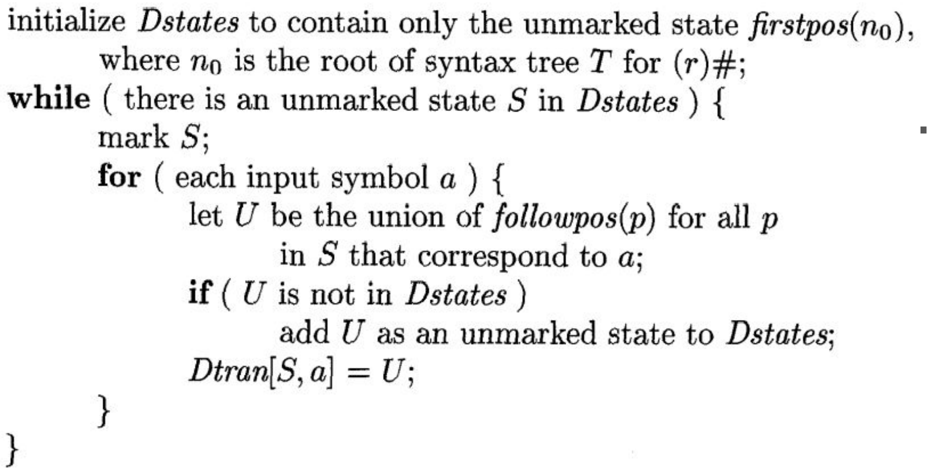
State	a	b	c
A	B		
B			

Dtran[A,a] = followpos(1) U followpos(3)
= {1,2,5} U {4}
= {1,2,5,4} -> B

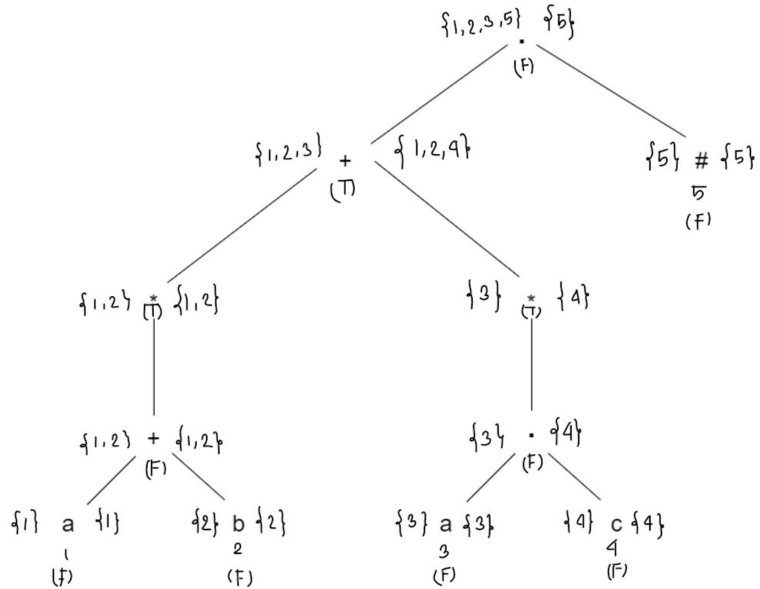
Node	Followpos	Node	Followpos
1	{1,2,5}	4	{3,5}
2	{1,2,5}	5	{}
3	{4}		

Prepared By: Sukarna Sarker

Step 4: Construct Dstates,Construct Dtrans



```
initialize $Dstates$ to contain only the unmarked state $firstpos(n_0)$,
        where $n_0$ is the root of syntax tree $T$ for $(r)\#$;
while ( there is an unmarked state $S$ in $Dstates$ ) {
        mark $S$;
        for ( each input symbol $a$ ) {
                let $U$ be the union of $followpos(p)$ for all $p$
                        in $S$ that correspond to $a$;
                if ( $U$ is not in $Dstates$ )
                        add $U$ as an unmarked state to $Dstates$;
                $Dtran[S,a] = U$;
        }
}
```



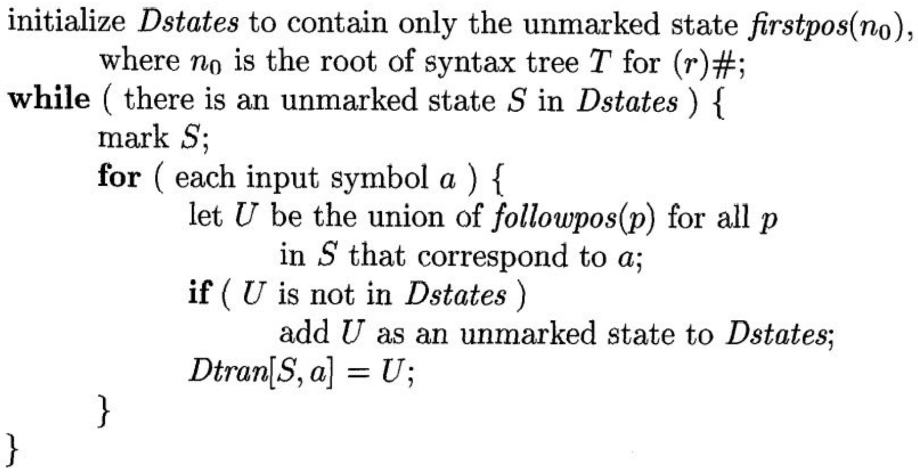
State	a	b	c
A	B	C	
B			
C			

Dtran[A,b] = followpos(2)
= {1,2,5} -> C

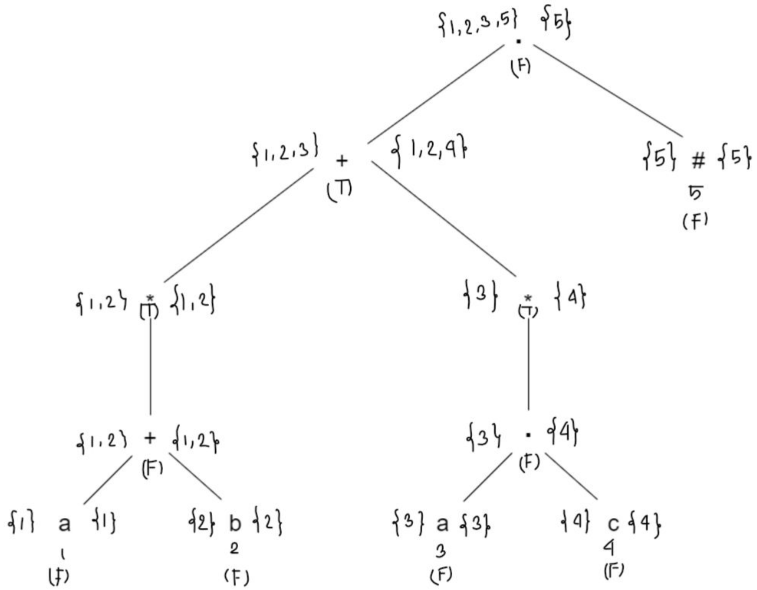
Node	Followpos	Node	Followpos
1	{1,2,5}	4	{3,5}
2	{1,2,5}	5	{}
3	{4}		

Prepared By: Sukarna Sarker

Step 4: Construct Dstates,Construct Dtrans



```
initialize $Dstates$ to contain only the unmarked state $firstpos(n_0)$,
        where $n_0$ is the root of syntax tree $T$ for $(r)\#$;
while ( there is an unmarked state $S$ in $Dstates$ ) {
        mark $S$;
        for ( each input symbol $a$ ) {
                let $U$ be the union of $followpos(p)$ for all $p$
                        in $S$ that correspond to $a$;
                if ( $U$ is not in $Dstates$ )
                        add $U$ as an unmarked state to $Dstates$;
                $Dtran[S, a] = U$;
        }
}
```



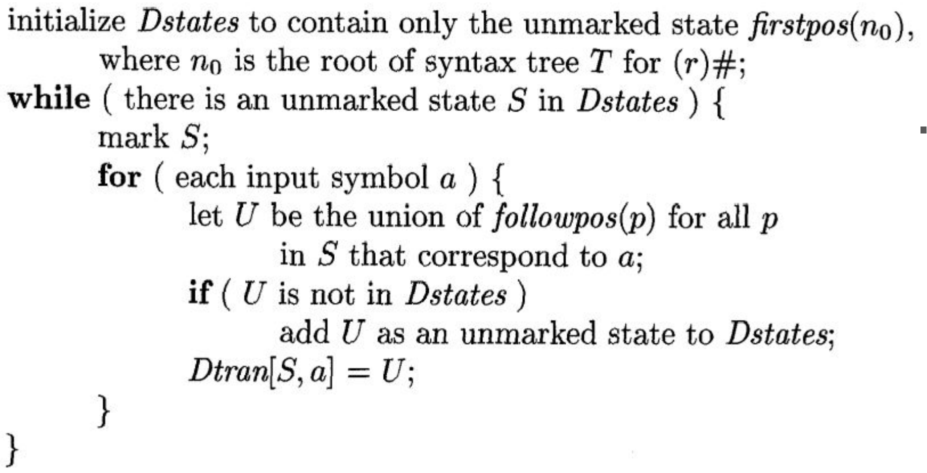
State	a	b	c
A	B	C	-
B			
C			

Dtran[A,c] = c is not present in node 1,2,3 or 5

Node	Followpos	Node	Followpos
1	{1,2,5}	4	{3,5}
2	{1,2,5}	5	{}
3	{4}		

Prepared By: Sukarna Sarker

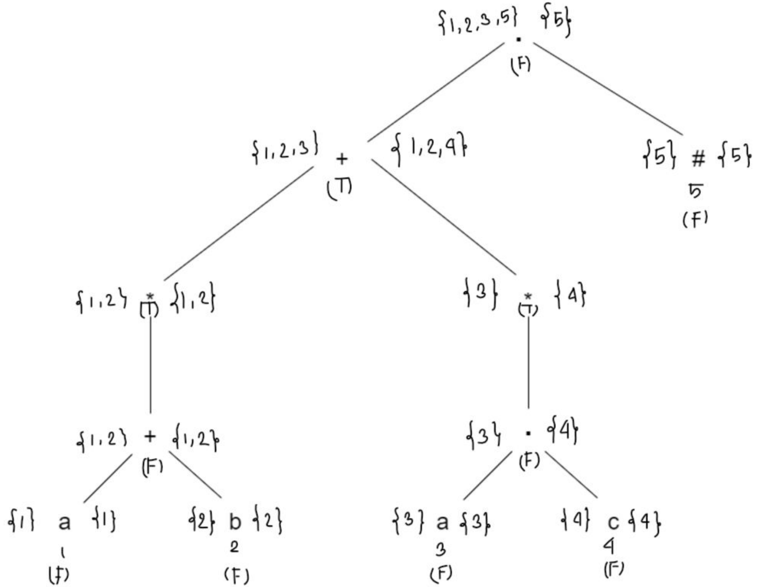
Step 4: Construct Dstates,Construct Dtrans



```
initialize $Dstates$ to contain only the unmarked state $firstpos(n_0)$,
        where $n_0$ is the root of syntax tree $T$ for $(r)\#$;
while ( there is an unmarked state $S$ in $Dstates$ ) {
        mark $S$;
        for ( each input symbol $a$ ) {
                let $U$ be the union of $followpos(p)$ for all $p$
                        in $S$ that correspond to $a$;
                if ( $U$ is not in $Dstates$ )
                        add $U$ as an unmarked state to $Dstates$;
                $Dtran[S, a] = U$;
        }
}
```

State	a	b	c
A	B	C	-
B	C		
C			

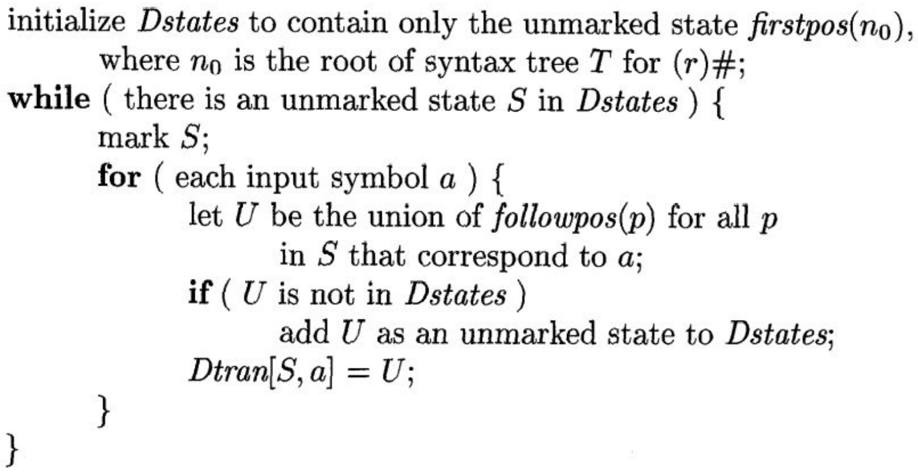
Dtran[B,a] = followpos(1)
=\{1,2,5\}



Node	Followpos	Node	Followpos
1	{1,2,5}	4	{3,5}
2	{1,2,5}	5	{}
3	{4}		

Prepared By: Sukarna Sarker

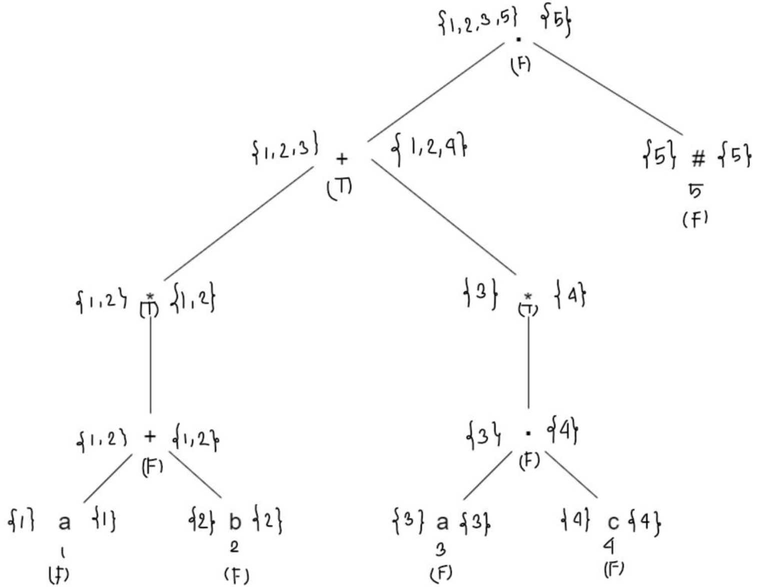
Step 4: Construct Dstates,Construct Dtrans



```
initialize $Dstates$ to contain only the unmarked state $firstpos(n_0)$,
        where $n_0$ is the root of syntax tree $T$ for $(r)\#$;
while ( there is an unmarked state $S$ in $Dstates$ ) {
        mark $S$;
        for ( each input symbol $a$ ) {
                let $U$ be the union of $followpos(p)$ for all $p$
                        in $S$ that correspond to $a$;
                if ( $U$ is not in $Dstates$ )
                        add $U$ as an unmarked state to $Dstates$;
                $Dtran[S, a] = U$;
        }
}
```

State	a	b	c
A	B	C	-
B	C	C	
C			

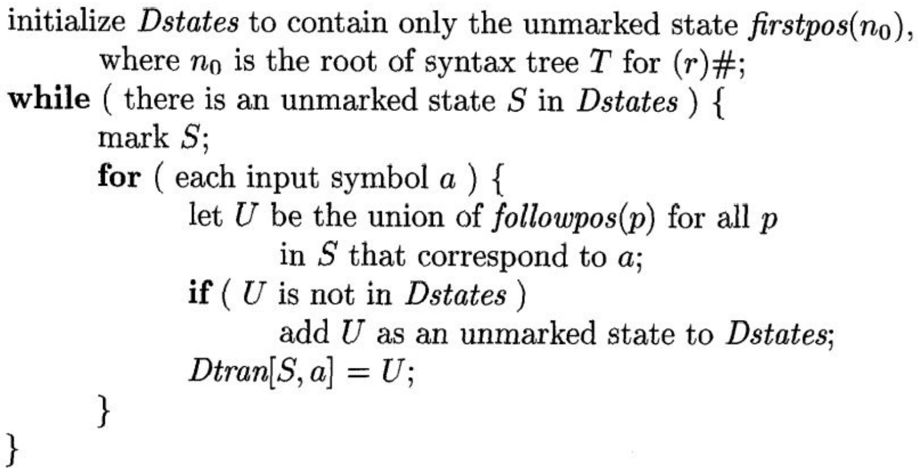
Dtran[B,b] = followpos(2)
={1,2,5}



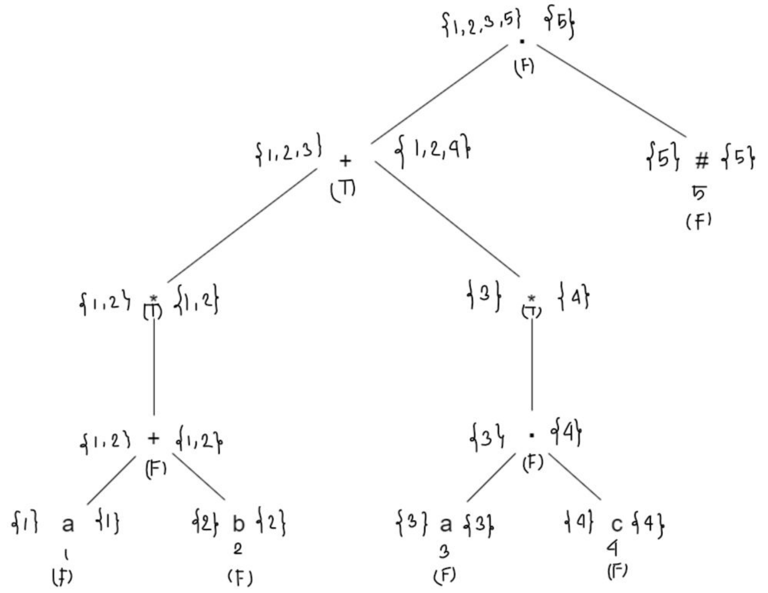
Node	Followpos	Node	Followpos
1	{1,2,5}	4	{3,5}
2	{1,2,5}	5	{}
3	{4}		

Prepared By: Sukarna Sarker

Step 4: Construct Dstates,Construct Dtrans



```
initialize $Dstates$ to contain only the unmarked state $firstpos(n_0)$,
        where $n_0$ is the root of syntax tree $T$ for $(r)\#$;
while ( there is an unmarked state $S$ in $Dstates$ ) {
        mark $S$;
        for ( each input symbol $a$ ) {
                let $U$ be the union of $followpos(p)$ for all $p$
                        in $S$ that correspond to $a$;
                if ( $U$ is not in $Dstates$ )
                        add $U$ as an unmarked state to $Dstates$;
                $Dtran[S, a] = U$;
        }
}
```



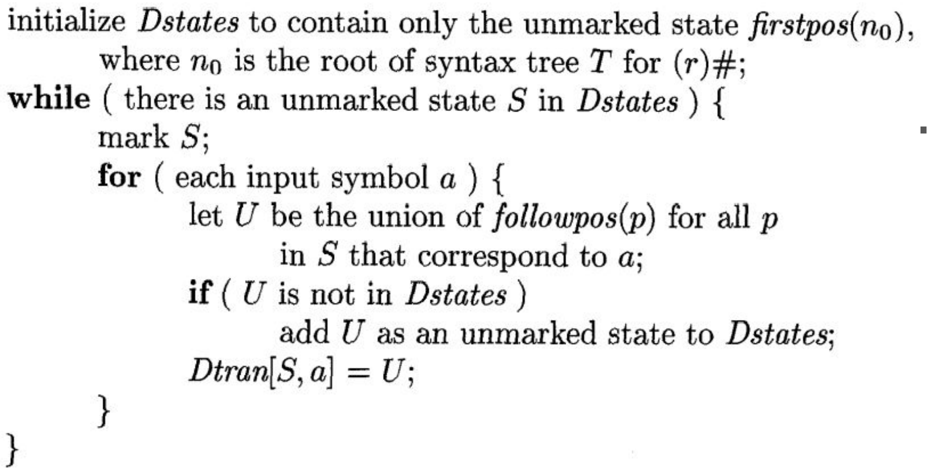
State	a	b	c
A	B	C	-
B	C	C	D
C			
D			

Dtran[B,c] = followpos(4)
=\{3,5\} ->D

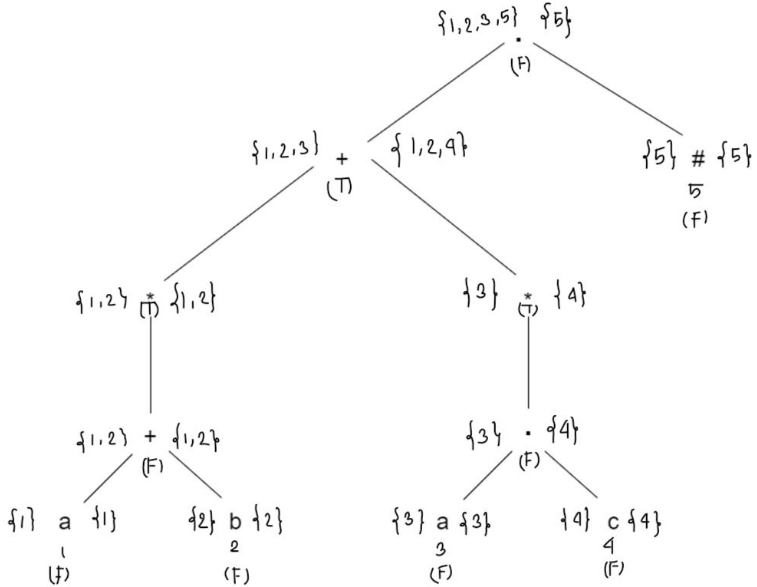
Node	Followpos	Node	Followpos
1	{1,2,5}	4	{3,5}
2	{1,2,5}	5	{}
3	{4}		

Prepared By: Sukarna Sarker

Step 4: Construct Dstates,Construct Dtrans



```
initialize $Dstates$ to contain only the unmarked state $firstpos(n_0)$,
        where $n_0$ is the root of syntax tree $T$ for $(r)\#$;
while ( there is an unmarked state $S$ in $Dstates$ ) {
        mark $S$;
        for ( each input symbol $a$ ) {
                let $U$ be the union of $followpos(p)$ for all $p$
                        in $S$ that correspond to $a$;
                if ( $U$ is not in $Dstates$ )
                        add $U$ as an unmarked state to $Dstates$;
                $Dtran[S, a] = U$;
        }
}
```



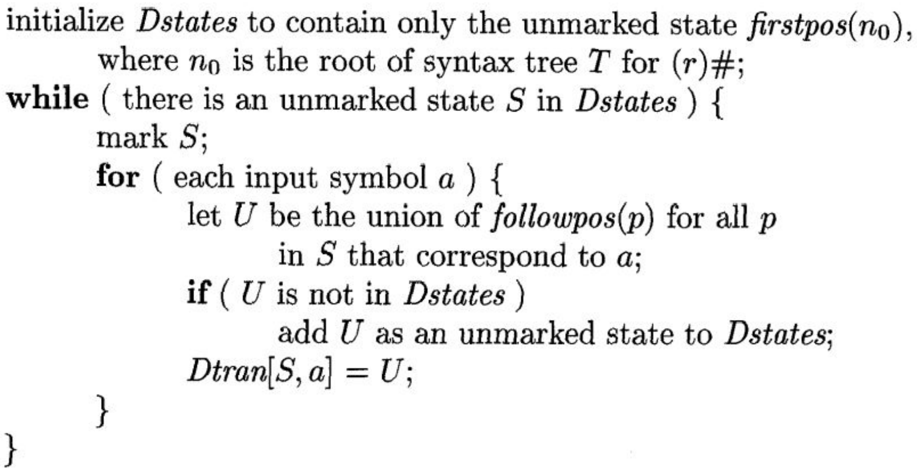
State	a	b	c
A	B	C	-
B	C	C	D
C	C		
D			

Dtran[C,a] = followpos(1)
=\{1,2,5\}

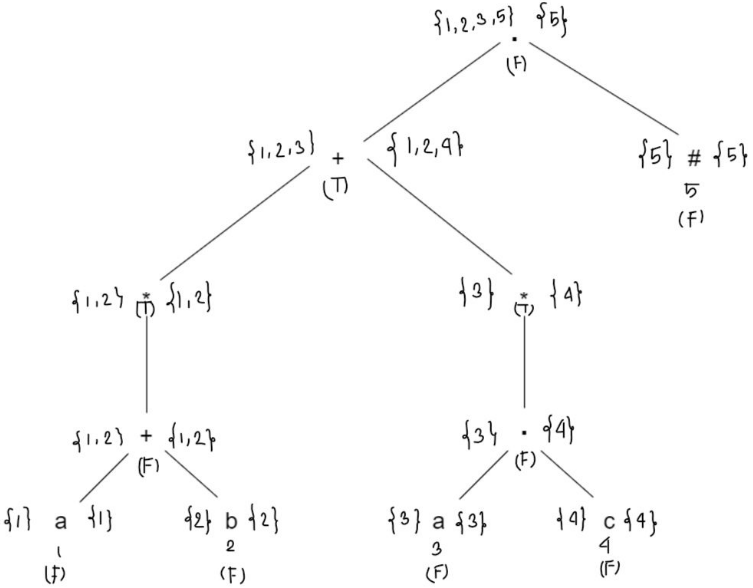
Node	Followpos	Node	Followpos
1	{1,2,5}	4	{3,5}
2	{1,2,5}	5	{}
3	{4}		

Prepared By: Sukarna Sarker

Step 4: Construct Dstates,Construct Dtrans



```
initialize $Dstates$ to contain only the unmarked state $firstpos(n_0)$,
        where $n_0$ is the root of syntax tree $T$ for $(r)\#$;
while ( there is an unmarked state $S$ in $Dstates$ ) {
        mark $S$;
        for ( each input symbol $a$ ) {
                let $U$ be the union of $followpos(p)$ for all $p$
                        in $S$ that correspond to $a$;
                if ( $U$ is not in $Dstates$ )
                        add $U$ as an unmarked state to $Dstates$;
                $Dtran[S, a] = U$;
        }
}
```



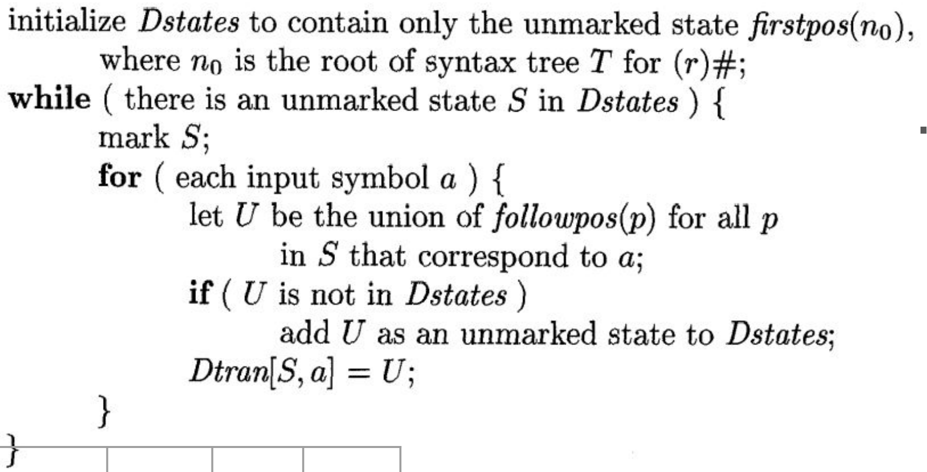
State	a	b	c
A	B	C	-
B	C	C	D
C	C	C	-
D			

Dtran[C,c] = c is not present in node 1,2 or 5

Node	Followpos	Node	Followpos
1	{1,2,5}	4	{3,5}
2	{1,2,5}	5	{}
3	{4}		

Prepared By: Sukarna Sarker

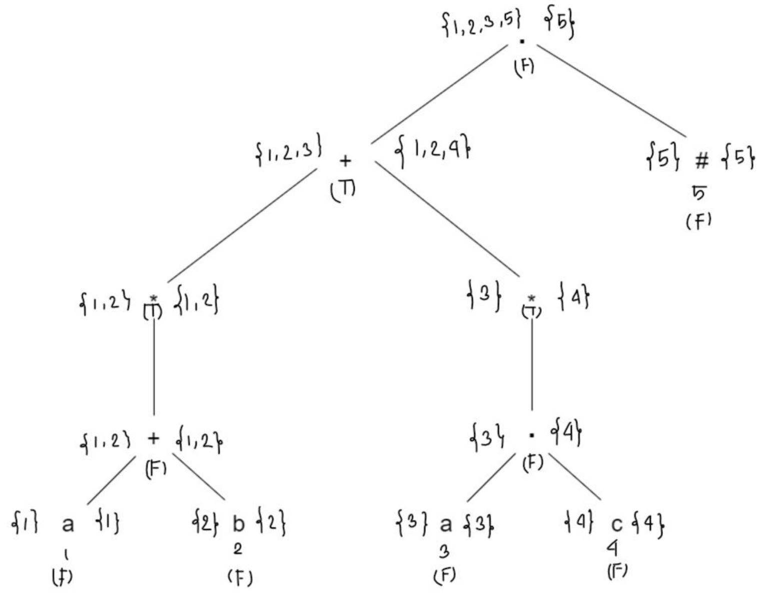
Step 4: Construct Dstates,Construct Dtrans



```
initialize Dstates to contain only the unmarked state firstpos($n_0$),
        where $n_0$ is the root of syntax tree $T$ for $(r)\#$;
while ( there is an unmarked state $S$ in Dstates ) {
        mark $S$;
        for ( each input symbol $a$ ) {
                let $U$ be the union of followpos($p$) for all $p$
                        in $S$ that correspond to $a$;
                if ( $U$ is not in Dstates )
                        add $U$ as an unmarked state to Dstates;
                Dtran[$S, a$] = $U$;
        }
}
```

State	a	b	c
A	B	C	-
B	C	C	D
C	C	C	-
D	E		
E			

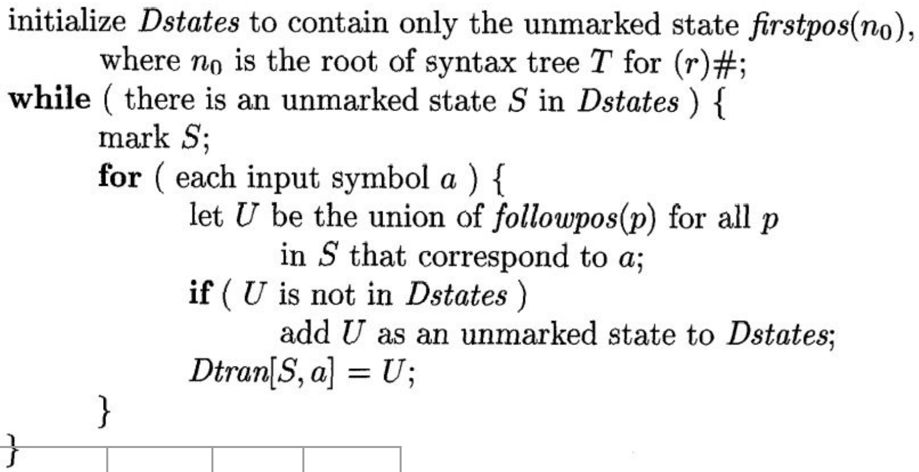
Dtran[D,a] = followpos(3)
={4} ->E



Node	Followpos	Node	Followpos
1	{1,2,5}	4	{3,5}
2	{1,2,5}	5	{}
3	{4}		

Prepared By: Sukarna Sarker

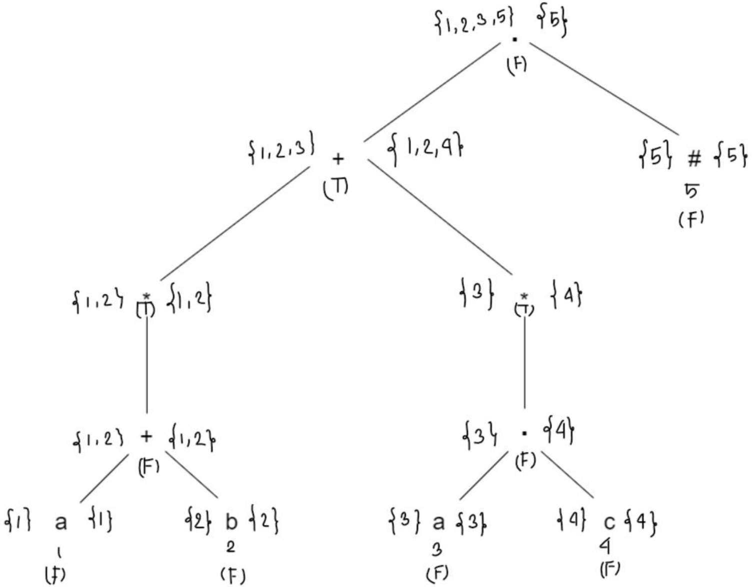
Step 4: Construct Dstates,Construct Dtrans



```
initialize Dstates to contain only the unmarked state firstpos($n_0$),
        where $n_0$ is the root of syntax tree $T$ for $(r)\#$;
while ( there is an unmarked state $S$ in Dstates ) {
        mark $S$;
        for ( each input symbol $a$ ) {
                let $U$ be the union of followpos($p$) for all $p$
                        in $S$ that correspond to $a$;
                if ( $U$ is not in Dstates )
                        add $U$ as an unmarked state to Dstates;
                Dtran$[S, a] = U$;
        }
}
```

State	a	b	c
A	B	C	-
B	C	C	D
C	C	C	-
D	E	-	
E			

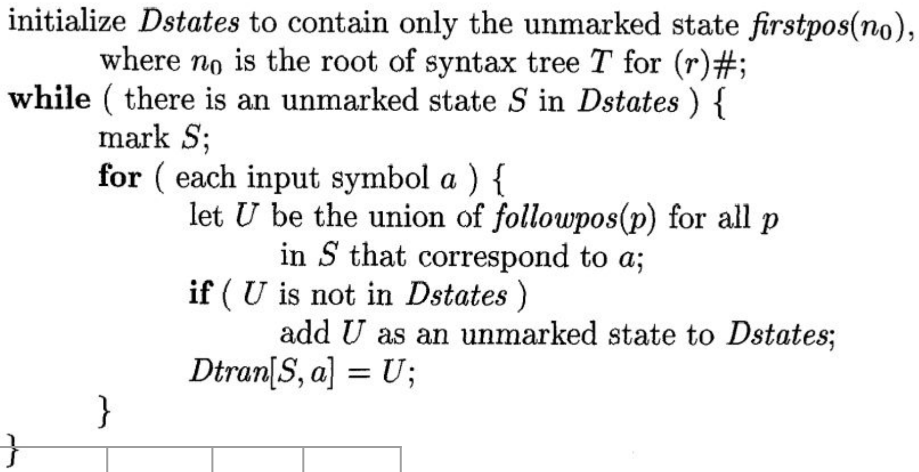
Dtran[D,b] = b is not present in node 3 or 5



Node	Followpos	Node	Followpos
1	{1,2,5}	4	{3,5}
2	{1,2,5}	5	{}
3	{4}		

Prepared By: Sukarna Sarker

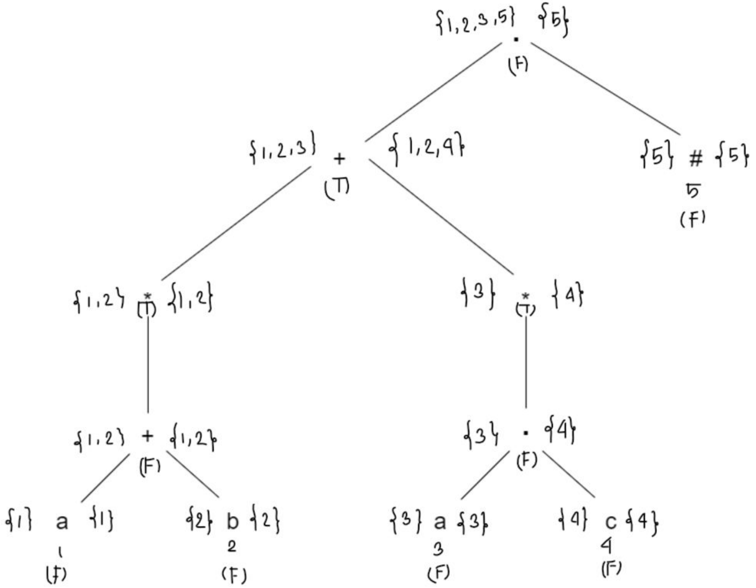
Step 4: Construct Dstates,Construct Dtrans



```
initialize $Dstates$ to contain only the unmarked state $firstpos(n_0)$,
        where $n_0$ is the root of syntax tree $T$ for $(r)\#$;
while ( there is an unmarked state $S$ in $Dstates$ ) {
        mark $S$;
        for ( each input symbol $a$ ) {
                let $U$ be the union of $followpos(p)$ for all $p$
                        in $S$ that correspond to $a$;
                if ( $U$ is not in $Dstates$ )
                        add $U$ as an unmarked state to $Dstates$;
                $Dtran[S, a] = U$;
        }
}
```

State	a	b	c
A	B	C	-
B	C	C	D
C	C	C	-
D	E	-	-
E			

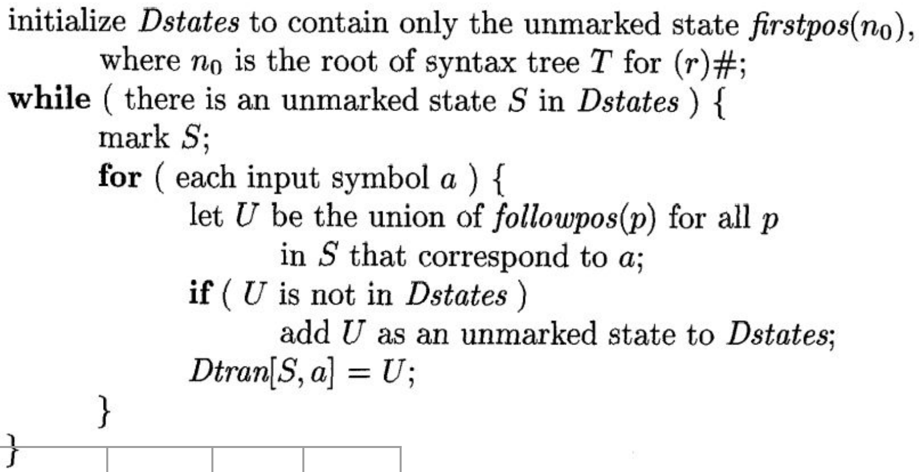
Dtran[D,c] = c is not present
In node 3 or 5



Node	Followpos	Node	Followpos
1	{1,2,5}	4	{3,5}
2	{1,2,5}	5	{}
3	{4}		

Prepared By: Sukarna Sarker

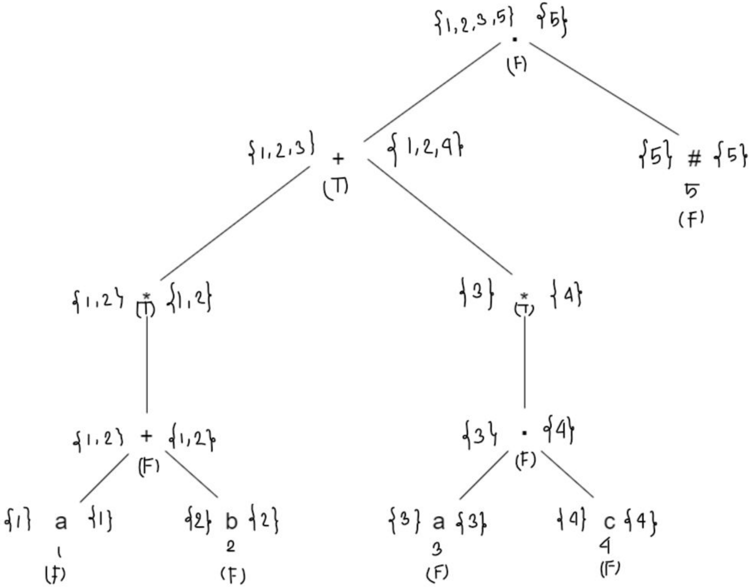
Step 4: Construct Dstates,Construct Dtrans



```
initialize $Dstates$ to contain only the unmarked state $firstpos(n_0)$,
        where $n_0$ is the root of syntax tree $T$ for $(r)\#$;
while ( there is an unmarked state $S$ in $Dstates$ ) {
        mark $S$;
        for ( each input symbol $a$ ) {
                let $U$ be the union of $followpos(p)$ for all $p$
                        in $S$ that correspond to $a$;
                if ( $U$ is not in $Dstates$ )
                        add $U$ as an unmarked state to $Dstates$;
                $Dtran[S, a] = U$;
        }
}
```

State	a	b	c
A	B	C	-
B	C	C	D
C	C	C	-
D	E	-	-
E	-		

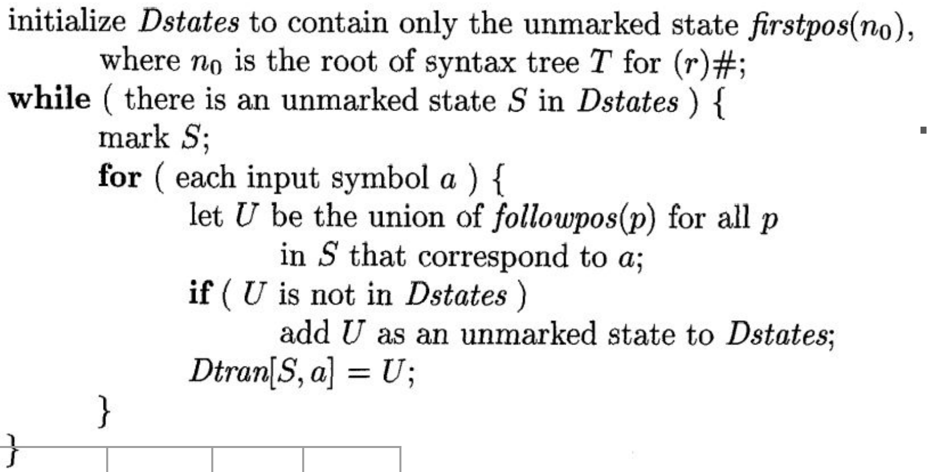
Dtran[E,a] = a is not present In node 4



Node	Followpos	Node	Followpos
1	{1,2,5}	4	{3,5}
2	{1,2,5}	5	{}
3	{4}		

Prepared By: Sukarna Sarker

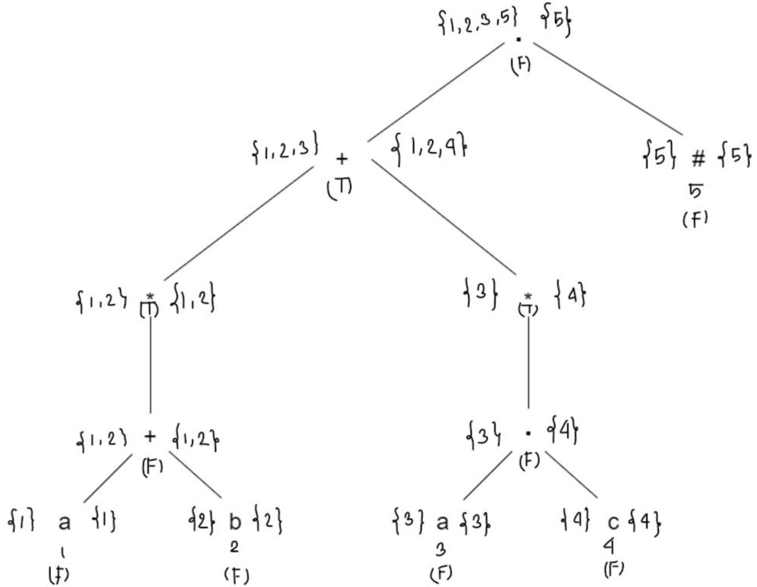
Step 4: Construct Dstates,Construct Dtrans



```
initialize Dstates to contain only the unmarked state firstpos(n_0),
        where n_0 is the root of syntax tree T for (r)#;
while ( there is an unmarked state S in Dstates ) {
        mark S;
        for ( each input symbol a ) {
                let U be the union of followpos(p) for all p
                        in S that correspond to a;
                if ( U is not in Dstates )
                        add U as an unmarked state to Dstates;
                Dtran[S, a] = U;
        }
}
```

State	a	b	c
A	B	C	-
B	C	C	D
C	C	C	-
D	E	-	-
E	-	-	

Dtran[E,b] = b is not present
In node 4



Node	Followpos	Node	Followpos
1	{1,2,5}	4	{3,5}
2	{1,2,5}	5	{}
3	{4}		

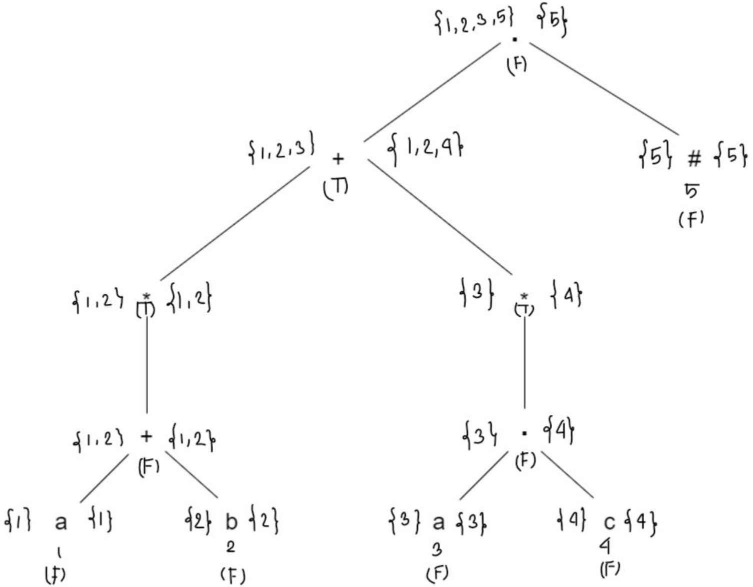
Prepared By: Sukarna Sarker

Step 4: Construct Dstates,Construct Dtrans

```
initialize $Dstates$ to contain only the unmarked state $firstpos(n_0)$,
        where $n_0$ is the root of syntax tree $T$ for $(r)\#$;
while ( there is an unmarked state $S$ in $Dstates$ ) {
        mark $S$;
        for ( each input symbol $a$ ) {
                let $U$ be the union of $followpos(p)$ for all $p$
                        in $S$ that correspond to $a$;
                if ( $U$ is not in $Dstates$ )
                        add $U$ as an unmarked state to $Dstates$;
                $Dtran[S, a] = U$;
        }
}
```

State	a	b	c
A	B	C	-
B	C	C	D
C	C	C	-
D	E	-	-
E	-	-	D

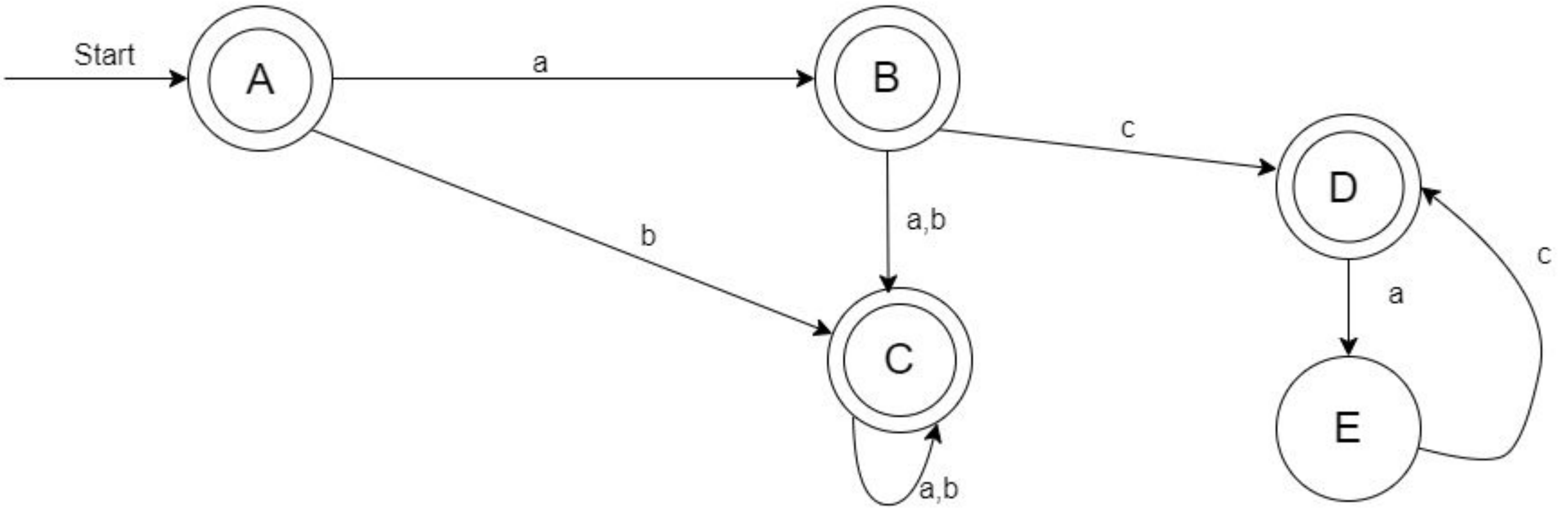
Dtran[E,c] = followpos(4)
= {3,5}



Node	Followpos	Node	Followpos
1	{1,2,5}	4	{3,5}
2	{1,2,5}	5	{}
3	{4}		

Prepared By: Sukarna Sarker

Step 5: Draw the DFA for the transition trable.



Prepared By: Sukarna Sarker