KU | Fall 2018 | Drew Davidson

EECS 665 COMPILER CONSTRUCTION

16 – SDT in Practice

Lesson Plan

- Last Time:
  - Discussed SDT specification
- This Time:
  - SDT in practice
    - Formalism
    - Implementation

An Opportunity Beyond Class

- Come to the talks!
- We're trying to get a speaker series going
  - Attendance is a sign of strength

Quiz 2 Details

- Study guide is up on the “Announcements” page
- Review Session Sunday at 5:00 PM
  - (Room ????)
  - Bring your KUID! You’ll need it to get in the building
- 6 questions:
  - Choose to answer either a CYK **or** an LL(1) question

Specification of SDT

- Intuition:
  - Attach attribute to parse tree nodes
  - Remove chains of redundant meaning
  - Build the AST from attribute connections
- Formalism:
  - Attribute Grammar
  - Rules to “decorate” treenodes with attributes

Frontend Analysis

Lexical Analysis

- Language abstraction: Regul Expressions
- Recognizer formalism: NFAs/DFAs
- Tool: JLex
- Transducer formalism: Tokenization table
- Output: Token Stream

Syntactic Analysis

Language abstraction: Context-Free Grammars

- Recognizer formalism: CFGs/LR/LL(1)
- Tool: Bison
- Transducer formalism: Attribute Grammar
- Output: AST by way of Parse Tree

We'll "Decorate" The Parse Tree

- Tag Symbols with additional fields or "Attributes"
- Typically refer to the symbol via subscript

```
L ::= L K
```

L_1.x refers to the attribute x of the first L (the one on the LHS)
L_2.x refers to the attribute x of the second L (the one on the RHS)
Subscript may be omitted if it's clear from context (e.g. in the K)

Attribute Types

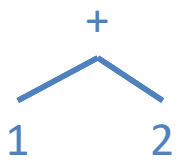
- Synthesized Attributes
  - Come from the children of the node
- Inherited Attributes
  - Come from the parent *and/or siblings* of the node

What do ASTs look like as Code?

AST Classes

- Consider AST node classes
  - We'd like the classes to have a common inheritance tree

AST

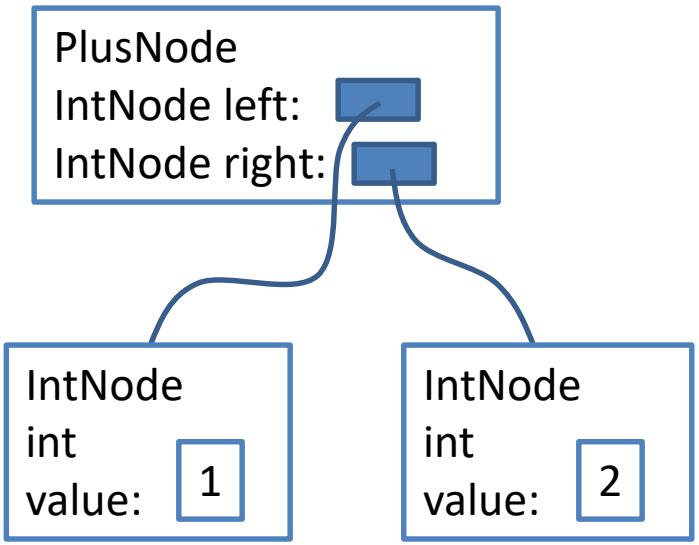


Naïve AST Implementation

```
class PlusNode
{       IntNode left;
        IntNode right;
}


class IntNode
{       int value;
}
```

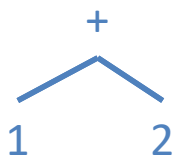
Naïve AST Implementation



AST Classes

- Consider AST node classes
  - We'd like the classes to have a common inheritance tree

AST



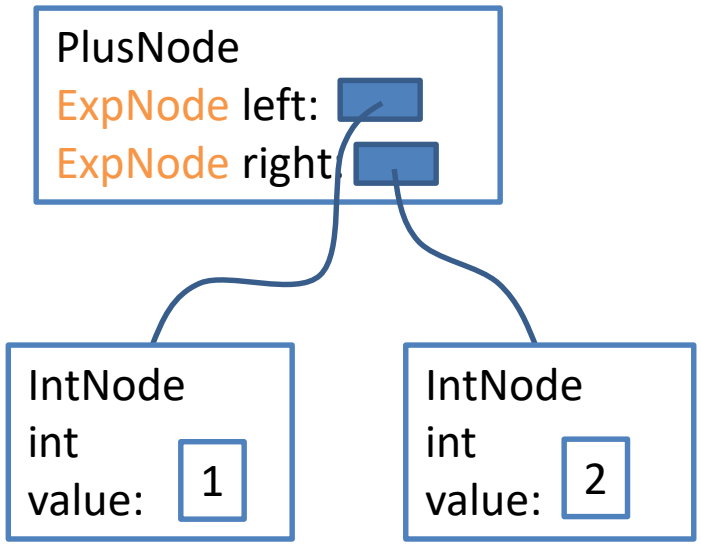
Naïve AST Implementation

```
class PlusNode
{       IntNode left;
        IntNode right;
}

class IntNode
{       int value;
}
```

Make these extend ExpNode

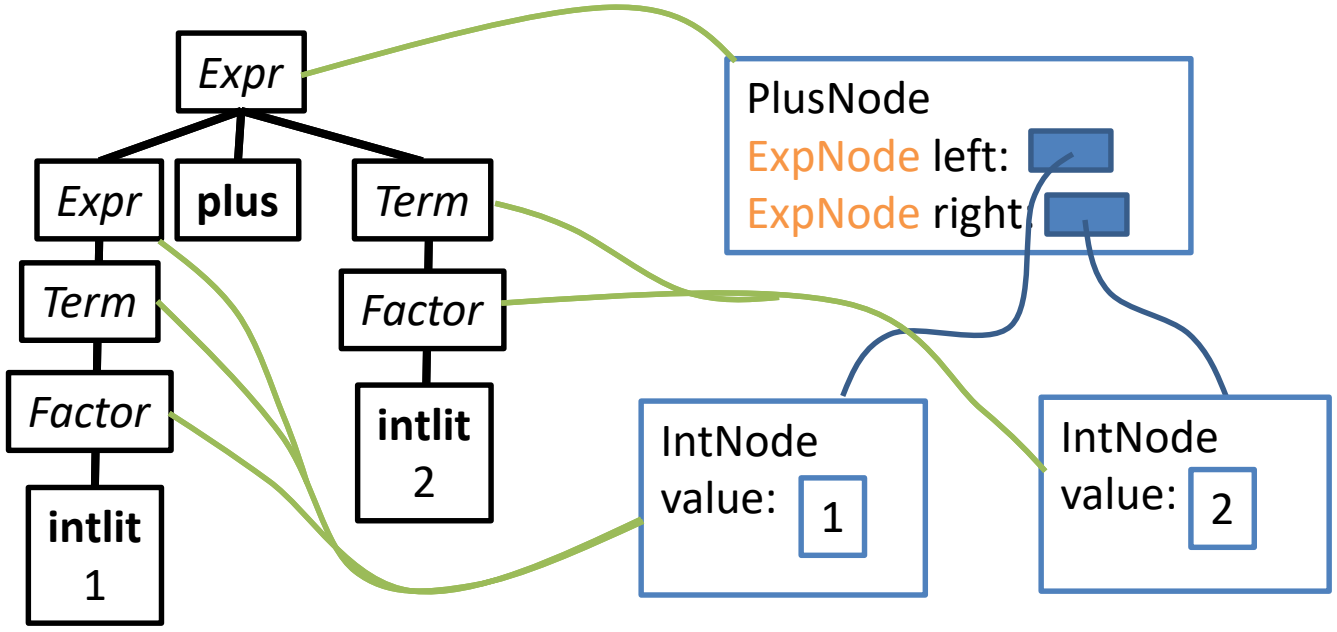
Better AST Implementation



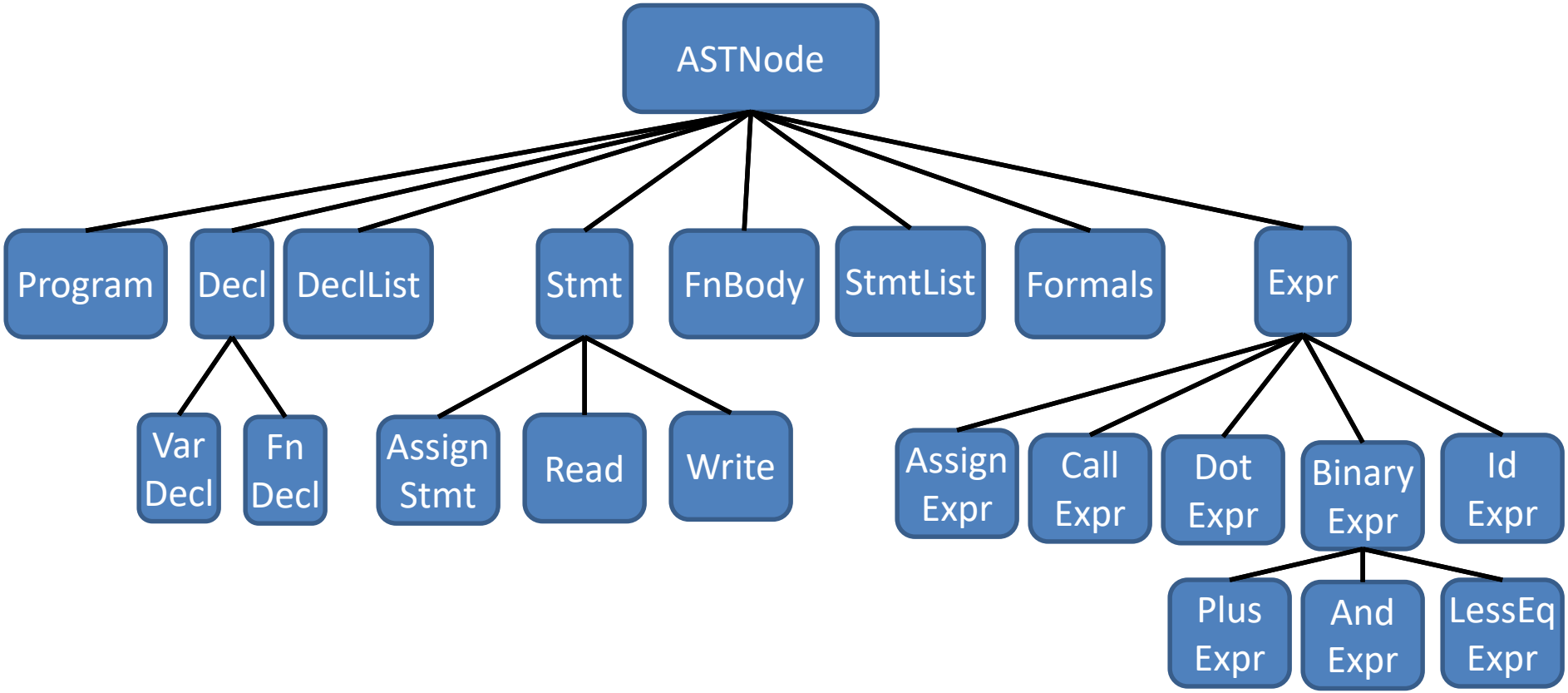
ASTs for Expressions

CFG		Translation Rules
Expr	-> Expr + Term	*Expr1*.trans = new PlusNode(*Expr2*.trans, *Term*.trans)
	| Term	*Expr*.terms = Term.trans
Term	-> Term * Factor	*Term1*.trans = new TimesNode(Term2.trans, *Factor*.trans)
	| Factor	*Term*.trans = *Factor*.trans
Factor	-> intlit	*Factor*.trans = new IntNode(**intlit**.value)
	| (Expr)	*Factor*.trans = *Expr*.trans

Example: 1 + 2

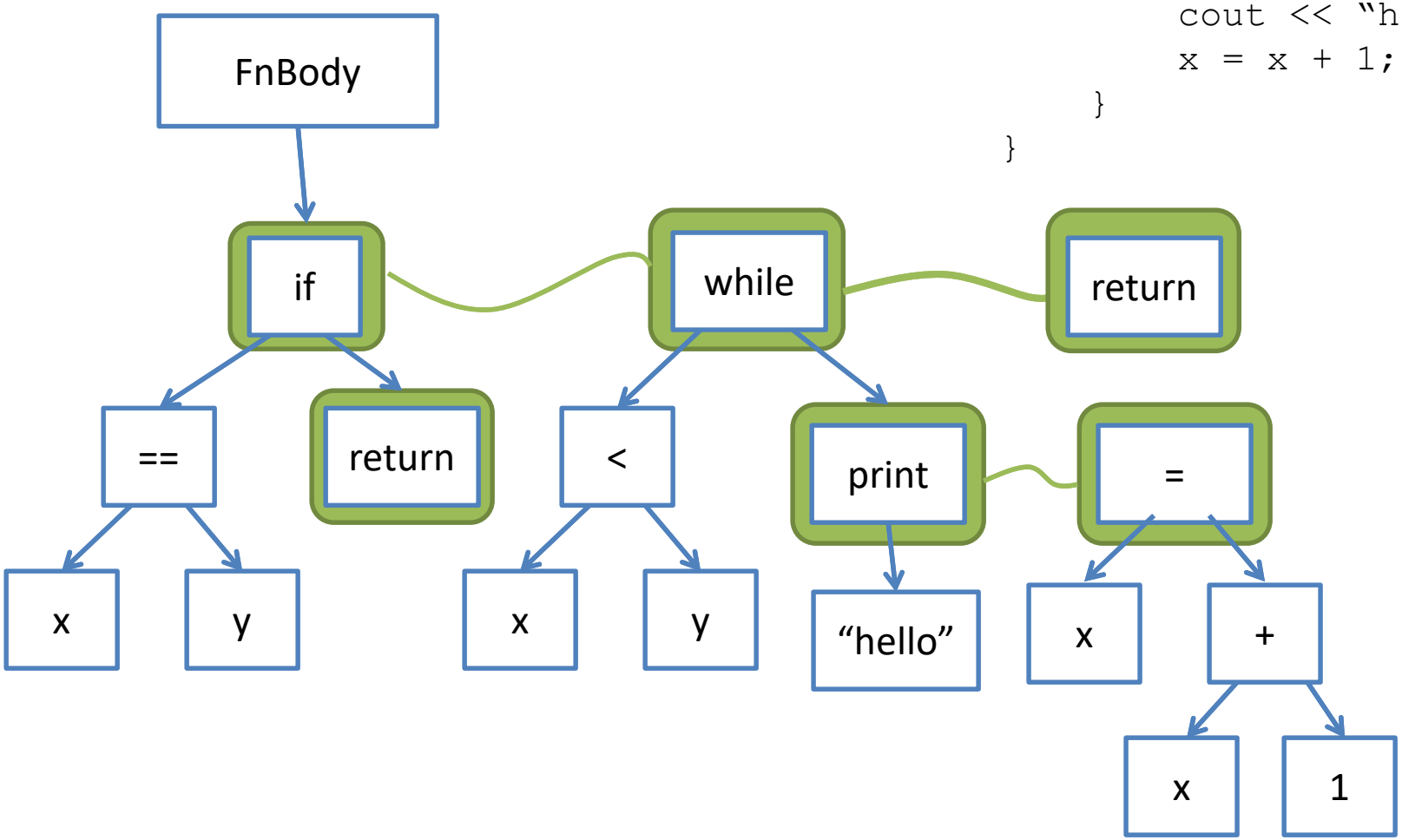


AST Node Hierarchy Snippet



AST for an Input Snippet

```
void foo(int x, int y){
    if (x == y){
        return;
    }
    while ( x < y){
        cout << "hello";
        x = x + 1;
    }
}
```



Next Time

- Well, the quiz…
- But after that! SDT for top-down grammars

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

Add FIRST(Y_1) - {ε}

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

If ε is in all RHS symbols, add ε

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

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

Add FIRST(β) – {ε}

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

Table[X][t]

```
for each production X ⟶ α
 for each terminal t in FIRST(α)
   put α in Table[X][t]
 if ε is in FIRST(α){
   for each terminal t in FOLLOW(X){
     put α in Table[X][t]
```

CFG $S \rightarrow (\ S\ ) \mid \{\ S\ \} \mid \varepsilon$

FIRST (*S*) =

FIRST ((*S*)) =

FIRST ({ *S* }) =

FIRST (ε) =

FOLLOW (S) =

	(	)	{	}	eof
S					

FIRST(α) for $\alpha = Y_1\ Y_2 \ldots Y_k$
Add FIRST(Y_1) - {ε}
If ε is in FIRST($Y_{1 \text{ to } i-1}$): add FIRST(Y_i) – {ε}
If ε is in all RHS symbols, add ε

FOLLOW(A) for $X \longrightarrow \alpha\ A\ \beta$
If A is the start, add **eof**
Add FIRST(β) – {ε}
Add FOLLOW(*X*) if ε in FIRST(β) or β empty

Table[X][t]

```
for each production X ⟶ α
 for each terminal t in FIRST(α)
   put α in Table[X][t]
 if ε is in FIRST(α){
   for each terminal t in FOLLOW(X){
     put α in Table[X][t]
```

CFG $S \rightarrow (\ S\ ) \mid \{\ S\ \} \mid \varepsilon$

FIRST (*S*) = { **{** , **(** , **ε** }
FIRST (**(** *S* **)**) =
FIRST (**{** *S* **}**) =
FIRST (ε) =
FOLLOW (S) =

	(	**)**	**{**	**}**	**eof**
S					

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

Add FIRST(Y_1) - {ε}

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

If ε is in all RHS symbols, add ε

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

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

Add FIRST(β) – {ε}

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

Table[X][t]

```
for each production X ⟶ α
 for each terminal t in FIRST(α)
   put α in Table[X][t]
 if ε is in FIRST(α){
   for each terminal t in FOLLOW(X){
     put α in Table[X][t]
```

CFG $S \rightarrow (\ S\ ) \mid \{\ S\ \} \mid \varepsilon$

FIRST (*S*) = { **{** , **(** , **ε** }

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

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

FIRST (ε) =

FOLLOW (S) =

	(	**)**	**{**	**}**	**eof**
S					

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

Add FIRST(Y_1) - {ε}

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

If ε is in all RHS symbols, add ε

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

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

Add FIRST(β) – {ε}

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

Table[X][t]

```
for each production X ⟶ α
 for each terminal t in FIRST(α)
   put α in Table[X][t]
 if ε is in FIRST(α){
   for each terminal t in FOLLOW(X){
     put α in Table[X][t]
```

CFG $S \rightarrow (\ S\ ) \mid \{\ S\ \} \mid \varepsilon$

FIRST (*S*) = { **{** , **(** , **ε** }

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

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

FIRST (ε) =

FOLLOW (S) =

	(	**)**	**{**	**}**	**eof**
S					

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

Add FIRST(Y_1) - {ε}

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

If ε is in all RHS symbols, add ε

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

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

Add FIRST(β) – {ε}

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

Table[X][t]

```
for each production X ⟶ α
 for each terminal t in FIRST(α)
   put α in Table[X][t]
 if ε is in FIRST(α){
   for each terminal t in FOLLOW(X){
     put α in Table[X][t]
```

CFG $S \rightarrow (\ S\ ) \mid \{\ S\ \} \mid \varepsilon$

FIRST (*S*) = { **{** , **(** , **ε** }

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

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

FIRST (ε) = { ε }

FOLLOW (S) =

	(	**)**	**{**	**}**	**eof**
S					

FIRST(α) for $\alpha = Y_1\ Y_2 \ldots Y_k$
Add FIRST(Y_1) - {ε}
If ε is in FIRST($Y_{1\ to\ i-1}$): add FIRST(Y_i) – {ε}
If ε is in all RHS symbols, add ε

FOLLOW(A) for $X \longrightarrow \alpha\ A\ \beta$
If A is the start, add **eof**
Add FIRST(β) – {ε}
Add FOLLOW(*X*) if ε in FIRST(β) or β empty

Table[X][t]

```
for each production X ⟶ α
 for each terminal t in FIRST(α)
   put α in Table[X][t]
 if ε is in FIRST(α){
   for each terminal t in FOLLOW(X){
     put α in Table[X][t]
```

CFG $S \rightarrow (\ S\ ) \mid \{\ S\ \} \mid \varepsilon$

FIRST (*S*) = { **{** , **(** , **ε** }
FIRST (**(** *S* **)**) = { **(** }
FIRST (**{** *S* **}**) = { **{** }
FIRST (ε) = { ε }
FOLLOW (S) = { **eof,), }** }

	(	**)**	**{**	**}**	**eof**
S					

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

Add FIRST(Y_1) - {ε}

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

If ε is in all RHS symbols, add ε

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

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

Add FIRST(β) – {ε}

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

Table[X][t]

```
for each production X ⟶ α
 for each terminal t in FIRST(α)
   put α in Table[X][t]
 if ε is in FIRST(α){
   for each terminal t in FOLLOW(X){
     put α in Table[X][t]
```

CFG $S \rightarrow (\ S\ ) \mid \{\ S\ \} \mid \varepsilon$

FIRST (*S*) = { **{** , **(** , **ε** }

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

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

FIRST (ε) = { ε }

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

	(	**)**	**{**	**}**	**eof**
S	(S)	ε	{ S }	ε	ε