University of Kansas | Drew Davidson

EECS 665
COMPILER CONSTRUCTION

FOLLOW Sets

Last Time

Review – FIRST Sets

Building LL(1) Parsers

- Transforming grammars:
  - Left factoring
  - Left-recursion elimination
- Building the selector table
  - FIRST Sets

You Should Know

- The intuition behind FIRST and FOLLOW
- The formal definition of FIRST sets

Parsing

Today's Outline

FOLLOW Sets

Building LL(1) Parsers

- LL(1) Game Plan
- Finish up FIRST Sets
- FOLLOW Sets

Parsing

Perspective: Where we're At

LL(1) Game Plan

Parsers are a bit tricky!

- Sadly, you need to know this to build a compiler frontend

The underlying concepts of FIRST and FOLLOW will be useful for LL(1) and other parsers

- (We'll talk about 1 other kind – the LR parsers, which is what BISON generates).

What We're Doing: The Big Picture

LL(1) Game Plan

Example Grammar

S ::= **(** *S* **)**
| **{** *S* **}**
| ***ε***

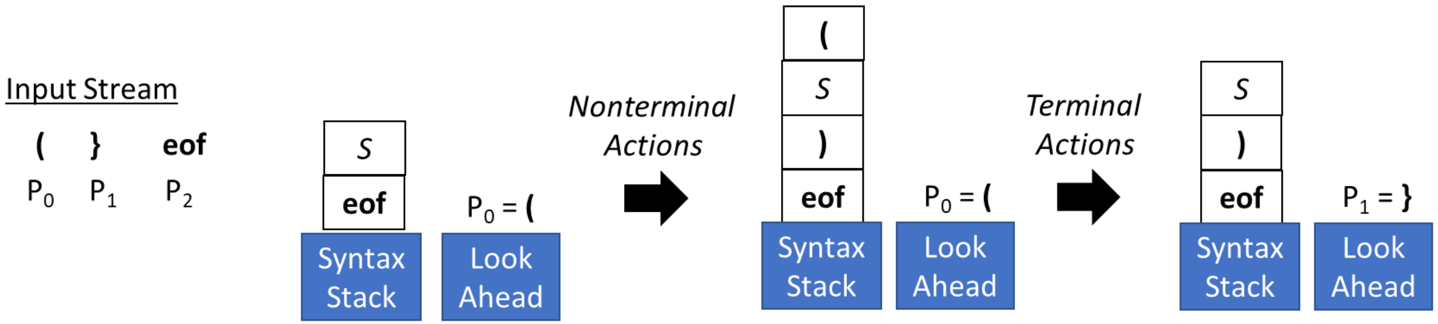
Turning this…

into this…

Selector Table

	(	)	{	}
S	(*S*)	ε	{ *S* }	ε

so we can do this…



Input Stream

(**}** **eof**
P_0 P_1 P_2

S
eof
Syntax Stack

P_0 = **(**
Look Ahead

Nonterminal Actions

(
S
)
eof
Syntax Stack

P_0 = **(**
Look Ahead

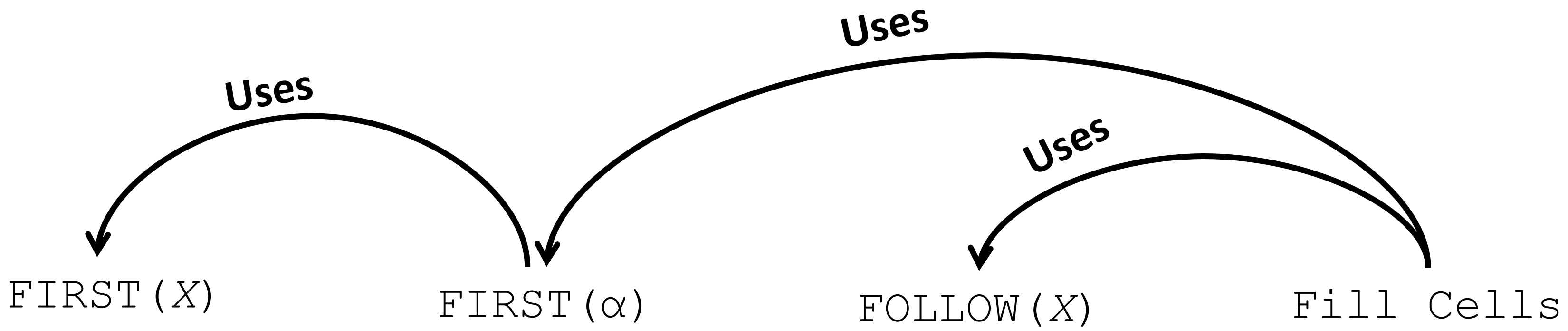
Terminal Actions

S
)
eof
Syntax Stack

P_1 = **}**
Look Ahead

What We're Doing: The Big Picture

Building the LL(1) Selector Table



`FIRST(`X`)`

Step 1.

Build FIRST sets for single symbols of the grammar

`FIRST(`α`)`

Step 2.

Build FIRST sets for RHS symbols strings

`FOLLOW(`X`)`

Step 3.

Build FOLLOW sets for Nonterminals

`Fill Cells`

Step 4.

Place productions

LL(1) Selector Table Algorithm

Building LL(1) Selector Table

```
for each production X ::= α
   if t is in FIRST(α)
      put X ::= α in Table[X][t]
   if ε is in FIRST(α)
      for each t in FOLLOW(X)
         put X::= α in Table[X][t]
```

We rely on FIRST sets and FOLLOW sets for table construction
But these sets will be useful even beyond the LL parsers

LL(1) Parsers Revisited: Big Picture

LL(1) The Big Picture

Grammar

P_1 S ::= X **b** X

P_2 | **b b**

P_3 X ::= **a** X

P_4 | **c**

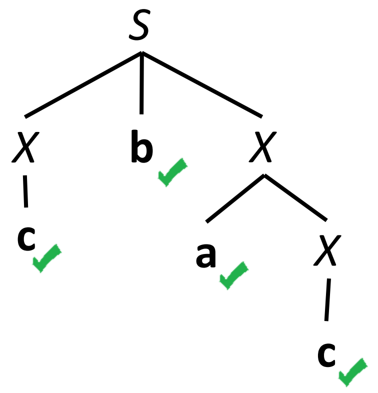
Selector Table

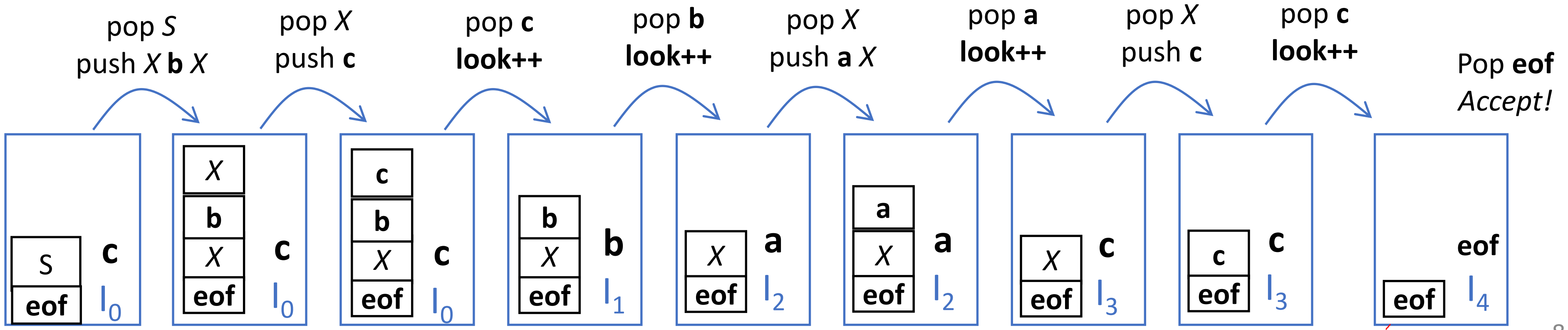
	a	b	c
S	P_1	P_2	P_1
X	P_3		P_4

Token stream

c	b	a	c	eof
l_0	l_1	l_2	l_3	l_4

Predicted Parse Tree





LL(1) Parsers Revisited: Big Picture

LL(1) The Big Picture

Grammar

P_1 *S* ::= *X* **b** *X*
P_2 | **b b**
P_3 X ::= **a** *X*
P_4 | **c**

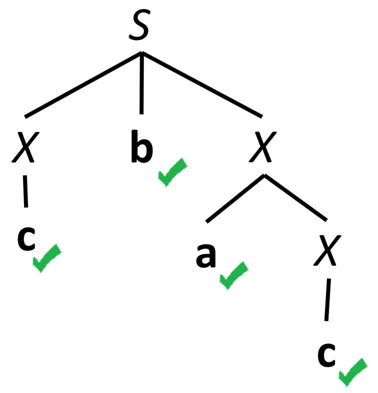
Selector Table

	a	b	c
S	P_1	P_2	P_1
X	P_3		P_4

Token stream

c	**b**	**a**	**c**	**eof**
l_0	l_1	l_2	l_3	l_4

Predicted Parse Tree



LL(1) Parser “Résumé”

- Goals: to expand the leftmost nonterminal
- Skills: always knows the first leaf of the leftmost nonterminal’s subtree

LL(1) Parsers Revisited: Big Picture

LL(1) The Big Picture

LL(1) Parser "Résumé"

- Goals: to expand the leftmost nonterminal
- Skills: always knows the first leaf of the target nonterminal's subtree

YOU'RE HIRED

In an LL(1) grammar this is a sufficient skillset!

- Can choose correct production when target's first leaf token is given *(FIRST sets)*
- Can choose correct production when there is no leaf token based on next subtree over *(FOLLOW sets)*

FIRST Set Intuition

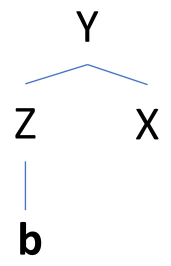
LL(1) The Big Picture

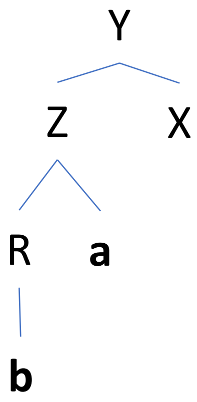
FIRST(X): The set of terminals that begin strings derivable from X, and also, if X can derive ε, then ε is in FIRST(X).

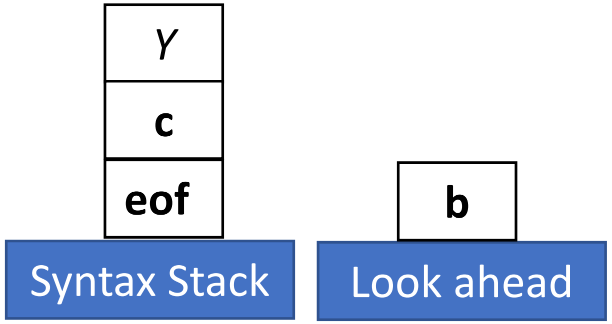
Example Grammar Fragment $\boxed{P_3}$ $Y ::= Z\ X$

Does P3 apply to this lookahead?

- Yes, if b is in FIRST(Z)







FIRST Set Intuition

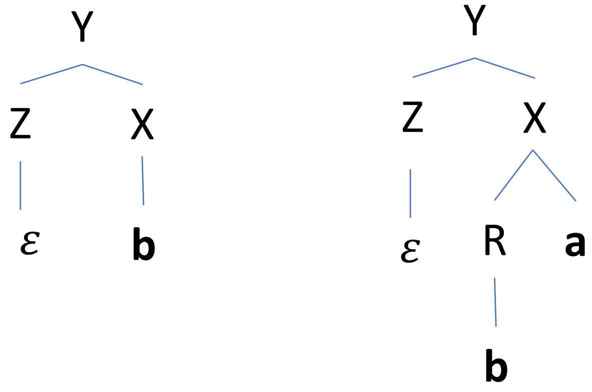
LL(1) The Big Picture

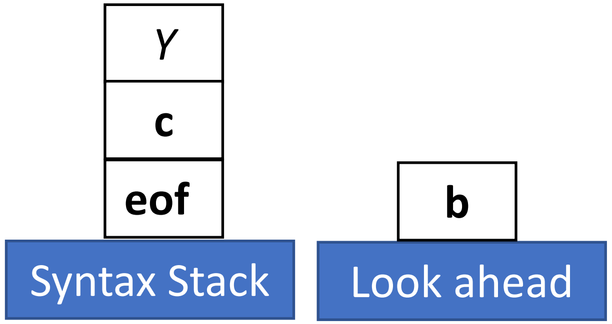
FIRST(X): The set of terminals that begin strings derivable from X, and also, if X can derive ε, then ε is in FIRST(X).

Example Grammar Fragment P$_3$ *Y ::= Z X*

Does P3 apply to this lookahead?

- Yes, if b is in FIRST(Z)
- Yes, if ε is in FIRST(Z) and b is in FIRST(X)




Y

c

eof

Syntax Stack

b

Look ahead

FIRST Set Intuition

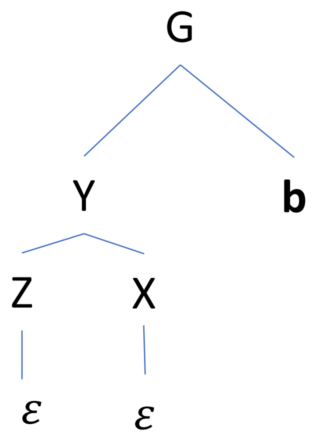
LL(1) The Big Picture

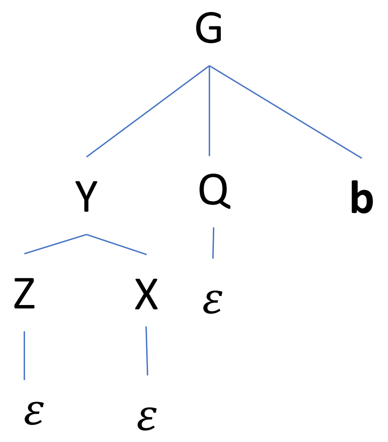
FIRST(X): The set of terminals that begin strings derivable from X, and also, if X can derive ε, then ε is in FIRST(X).

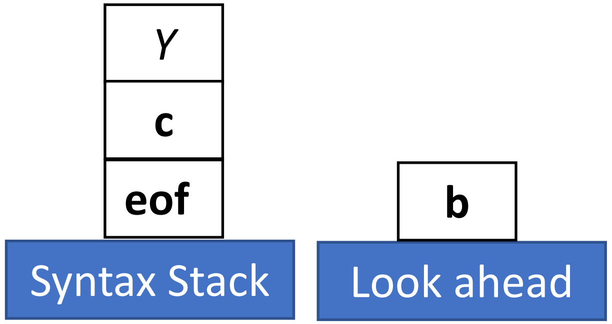
Example Grammar Fragment $\boxed{P_3}$ $Y ::= Z\ X$

Does P3 apply to this lookahead?

- Yes, if b is in FIRST(Z)
- Yes, if ε is in FIRST(Z) and b is in FIRST(X)
- Yes, if ε is in FIRST(Z) and FIRST(X), and b can FOLLOW right after Y







FIRST Set Intuition

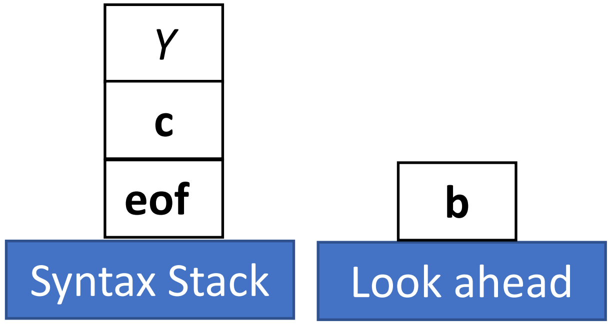
LL(1) The Big Picture

FIRST(X): The set of terminals that begin strings derivable from X, and also, if X can derive ε, then ε is in FIRST(X).

Example Grammar Fragment $\boxed{P_3}$ $Y ::= Z\,X$

Does P3 apply to this lookahead?

- Yes, if b is in FIRST(Z)
- Yes, if ε is in FIRST(Z) and b is in FIRST(X)
- Yes, if ε is in FIRST(Z) and FIRST(X), and b can FOLLOW right after Y



We're interested in a more general question…

FIRST Set Intuition

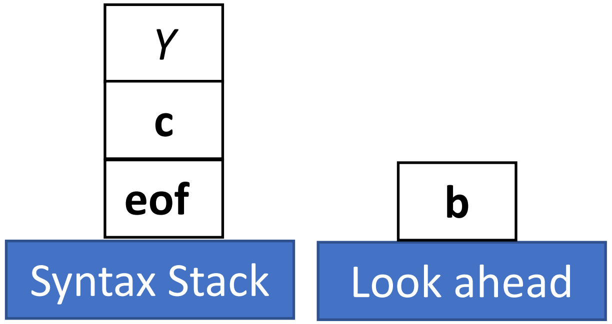
LL(1) The Big Picture

Example Grammar Fragment

P_1	*X* ::= **a** *Y* **c**
P_2	| **c**
P_3	*Y* ::= *Z X*
P_4	*Z* ::= **b**
P_5	| **a**

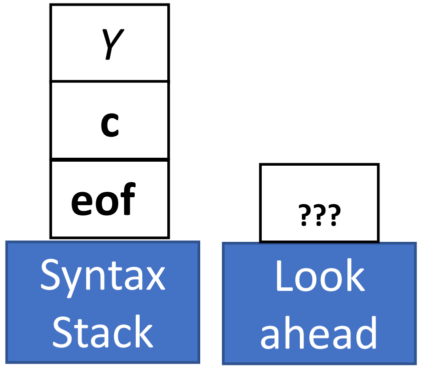
Does P3 apply to this lookahead?

- Yes, if b is in FIRST(Z)
- Yes, if ε is in FIRST(Z) and b is in FIRST(X)
- Yes, if ε is in FIRST(Z) and FIRST(X), and b can FOLLOW right after Y



At what lookahead tokens does P3 apply?

- Those in FIRST(Z)
- If ε is in FIRST(Z), those in FIRST(X)
- If ε is in FIRST(Z) and FIRST(X), those that follow Y



Today's Outline

FOLLOW Sets

Building LL(1) Parsers

- LL(1) Game Plan
- Building a Grammar's FIRST sets
- FOLLOW Sets

Parsing

FIRST Sets: Review what we know

Building a Grammar’s FIRST Sets

Building FIRST for a terminal t

FIRST(t) = { t }

Building FIRST for ε

FIRST(ε) = { ε }

Building FIRST for a symbol string α

Let α be composed of symbols $\alpha_1\ \alpha_2 \ldots \alpha_n$

- C_1: add FIRST(α_1) - ε
- C_2: For all k < n: if $\alpha_1 \ldots \alpha_{k-1}$ is nullable, add FIRST(α_k) - ε
- C_3: If $\alpha_1 \ldots \alpha_n$ is nullable, add ε

Building FIRST for a nonterminal X

For all productions with X on the LHS and $\alpha = \alpha_1\ \alpha_2 \ldots \alpha n$ on the RHS

- C_1: add FIRST(α_1) - ε
- C_2: For all k < n: if $\alpha_1 \ldots \alpha_{k-1}$ is nullable, add FIRST(α_k) - ε
- C_3: If $\alpha_1 \ldots \alpha_n$ is nullable, add ε

Same

Building FIRST for a nonterminal X

For all productions with X on the LHS (i.e. X ::= α)

- Add FIRST(α) to FIRST X

FIRST Sets: Review what we know

Building a Grammar’s FIRST Sets

Building FIRST for a terminal t

FIRST(t) = { t }

Building FIRST for ε

FIRST(ε) = { ε }

Building FIRST for a symbol string α

Let α be composed of symbols $\alpha_1\ \alpha_2 \ldots \alpha_n$

C_1: add FIRST(α_1) - ε

C_2: For all k < n: if $\alpha_1 \ldots \alpha_{k-1}$ is nullable, add FIRST(α_k) - ε

C_3: If $\alpha_1 \ldots \alpha_n$ is nullable, add ε

Mutually recursive (dependency loop)!
This means that there’s one additional step we need…

Building FIRST for a nonterminal X

For all productions with X on the LHS (i.e. X ::= α)

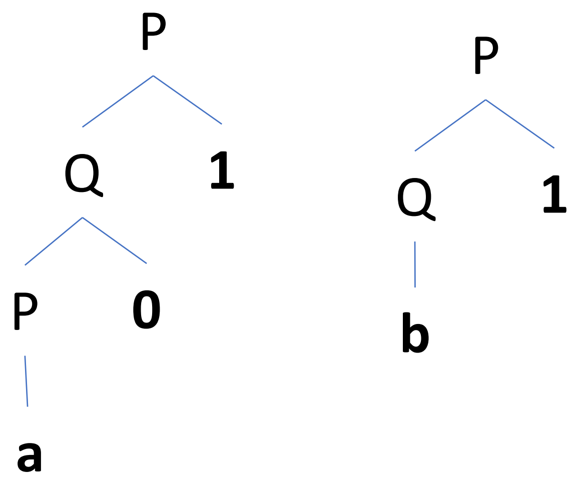
Add FIRST(α) to FIRST X

Building FIRST for all Grammar Symbols

Building Grammar's FIRST Sets

```
For each nonterminal of the grammar
Loop over for all productions (of the form $X ::= \alpha$, wlog)
    Add FIRST($\alpha$) to FIRST($X$)
    (if a set hasn't been computed, use {}, the empty set)
until saturation (no set changes)
```

P ::= Q 1
| a
Q ::= P 0
| b



$\mathrm{FIRST}(P) \subseteq \mathrm{FIRST}(Q) \subseteq \mathrm{FIRST}(P)$

Tricks for Computing FIRST Sets

Building Parser Tables

- Begin by computing the single-symbol FIRST sets for each production's LHS
- Run until saturation
- Can help to work bottom-up
- Compute symbol-string FIRST sets for each production's RHS
- Stay hydrated!

S ::= *X* **b** *X*
| ε
X ::= **a** *X*
| ε

Today’s Outline

FOLLOW Sets

Building LL(1) Parsers

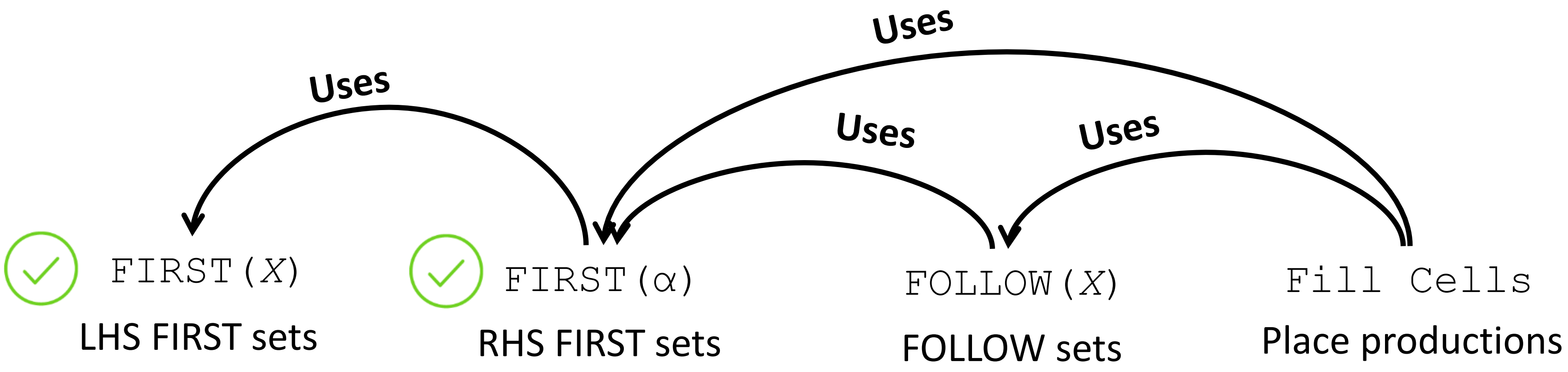
- LL(1) Game Plan
- Building a Grammar’s FIRST sets
- FOLLOW Sets

Parsing

Selector Table Dependencies

Building the Selector Table

```
for each production X ::= α
    if t is in FIRST(α)
        put X ::= α in Table[X][t]
    if ε is in FIRST(α)
        for each t in FOLLOW(X)
            put X::= α in Table[X][t]
```



✓ FIRST(*X*)

LHS FIRST sets

✓ FIRST(α)

RHS FIRST sets

FOLLOW(*X*)

FOLLOW sets

Fill Cells

Place productions

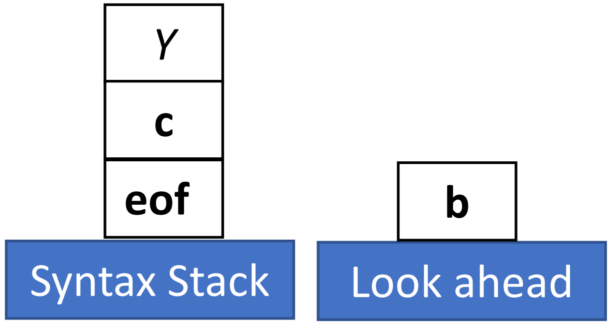
Follow Set Intuition

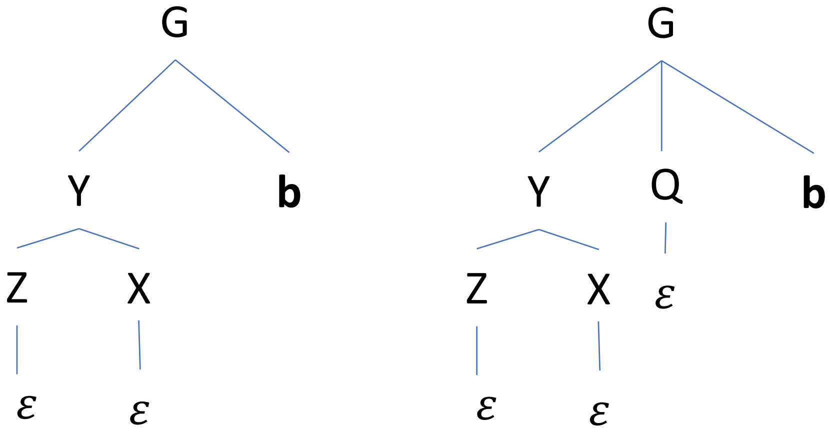
LL(1) The Big Picture

Example Grammar Fragment $\boxed{P_3}$ $Y ::= Z\,X$

Does P3 apply to this lookahead?

- Yes, if b is in FIRST(Z)
- Yes, if ε is in FIRST(Z) and b is in FIRST(X)
- Yes, if ε is in FIRST(Z) and FIRST(X), and b can FOLLOW right after Y

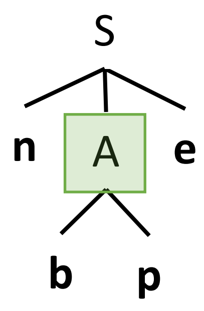




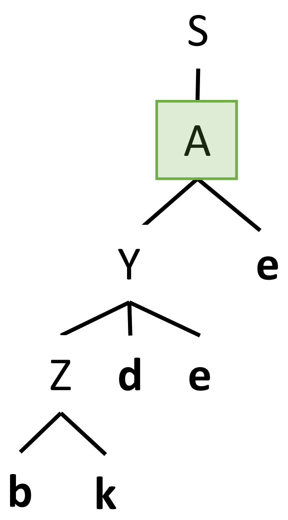
Again, The Parse tree Perspective

Consider the Trees

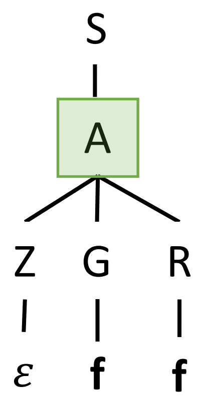
FIRST(X): The set of terminals that begin strings <u>derivable from</u> X, and also, if X can derive ε, then ε is in FIRST(X).



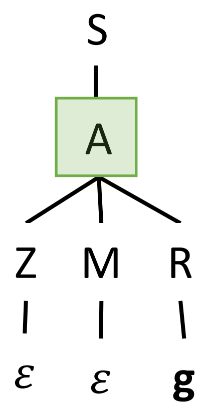
b ∈ FIRST(A)



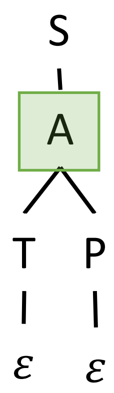
b ∈ FIRST(A)



f ∈ FIRST(A)



g ∈ FIRST(A)



ε ∈ FIRST(A)

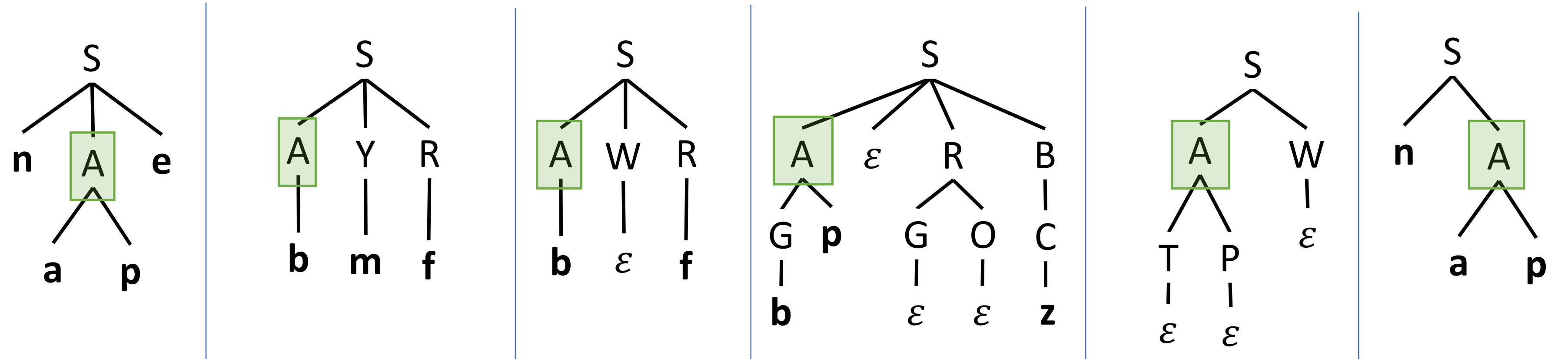
Again, The Parse tree Perspective

Consider the Trees

FIRST(X): The set of terminals that begin strings derivable from X, and also, if X can derive ε, then ε is in FIRST(X).

FOLLOW(X): The set of terminals that begin strings derivable right after X, and **EOF** if there could be *no* terminals after subtree

What does each parse tree say about FOLLOW(A) where S is start?



Yields { **e** }

Yields { **m** }

Yields { **f** }

Yields { **z** }

Yields { **eof** }

Yields { **eof** }

If these were the only parse trees, what is FOLLOW(A)?

{ **e**, **m**, **f**, **z**, **eof** }

The Importance of FOLLOW

Building Parser Tables

S ::= X **b**
X ::= **a**
X ::= ε

	a	**b**
S	*S ::= X* **b**	*S ::= X* **b**
X		

FIRST(X **b**) = { **a**, **b** }

The Importance of FOLLOW

Building Parser Tables

S ::= X **b**
X ::= **a**
X ::= ε

	a	**b**
S	*S ::= X* **b**	*S::= X* **b**
X	*X* **::= a**	

FIRST(X **b**) = { **a**, **b** }

FIRST(**a**) = { **a** }

The Importance of FOLLOW

Building Parser Tables

S ::= X **b**
X ::= **a**
X ::= ε

	a	**b**
S	*X* **b**	*X* **b**
X	**a**	ε

FIRST(X **b**) = { **a**, **b** }

FIRST(**a**) = { **a** }

We need to know that
b *follows* X
to place this

The Importance of FOLLOW

Building Parser Tables

S ::= X **b**
X ::= **a**
X ::= ε

	a	**b**
S	*X* **b**	*X* **b**
X	**a**	ε

FIRST(X **b**) = { **a**, **b** }

FIRST(**a**) = { **a** }

We need to know that
b *follows* X
to place this

S ::= X
X ::= **a** *X*
X ::= ε

	a	**EOF**
S	*X*	*X*
X	**a** *X*	ε

FOLLOW Sets, Formally

Building Parser Tables

Those terminals *derivable* *immediately after X*

$$\mathrm{FOLLOW}(X) = \left\{ t \;\middle|\; \left(t \in \Sigma \wedge S \overset{+}{\Rightarrow} \alpha X \mathbf{t} \beta \right) \vee \left(t = \mathbf{eof} \wedge S \overset{+}{\Rightarrow} \alpha X \right) \right\}$$

also eof *when X ends a derivation*

Example: Building Follow Sets

Building Parser Tables

FOLLOW(*X*) for each nonterminal X

C_1: If *X* is the start nonterminal, add **eof**

For all *Z* ::= α *X* β (where α and/or β may be empty)

  C_2: Add FIRST(β) – {ε}

  C_3: If ε is in FIRST(β) add FOLLOW(*Z*)

  C_4: If β is empty add FOLLOW(*Z*)

Repeat for each nonterminal until saturation

Grammar

❶ S ::= **a**
❷ S ::= **b** R
❸ Q ::= ε
❹ R ::= Q **c**
❺ R ::= Q S
❻ R ::= Q Q

FIRST(S) = { a, b }

FIRST(Q) = { ε }

FIRST(R) = { **c**, **a**, **b**, ε}

FIRST(Q **c**) = { c }

FIRST(Q S) = { a, b }

FIRST(Q Q) = { ε }

➡ FOLLOW(S) = { **eof** }

FOLLOW(Q)

FOLLOW(R)

FOLLOW(*X*) for nonterminal *X*

C_1: If *X* is the start nonterminal, add **eof**

For all *Z* ::= α *X* β (where α and/or β may be empty)

- C_2: Add FIRST(β) – {ε}
- C_3: If ε is in FIRST(β) add FOLLOW(*Z*)
- C_4: If β is empty add FOLLOW(*Z*)

Repeat for each nonterminal until saturation

Building Follow(S) *(S in for X)*

C_1: S is the start nonterminal, so add **eof**

Rules of the form $\underset{R}{Z}$ ::= $\underset{Q}{\alpha}$ $\underset{S}{X}$ $\underset{\text{empty}}{\beta}$

R ::= Q S

C_2: β is empty, so add nothing

C_3: β is empty, so N/A

C_4: β is empty, so add FOLLOW(R), which is currently nothing

Grammar

❶ $S ::= \mathbf{a}$

❷ $S ::= \mathbf{b}\ R$

❸ $Q ::= \varepsilon$

❹ $R ::= Q\ \mathbf{c}$

❺ $R ::= Q\ S$

❻ $R ::= Q\ Q$

FOLLOW(*X*) for nonterminal *X*

C_1: If *X* is the start nonterminal, add **eof**

For all $Z ::= \alpha\ X\ \beta$ (where α and/or β may be empty)

- C_2: Add FIRST(β) – {ε}
- C_3: If ε is in FIRST(β) add FOLLOW(*Z*)
- C_4: If β is empty add FOLLOW(*Z*)

Repeat for each nonterminal until saturation

FIRST(S) = { a, b }

FIRST(Q) = { ε }

FIRST(R) = { **c**, **a**, **b**, ε}

FIRST(Q **c**) = { c }

FIRST(Q S) = { a, b }

FIRST(Q Q) = { ε }

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

➡ FOLLOW(Q)

FOLLOW(R)

Building Follow(Q) *(Q in for X)*

C_1: N/A (*Q* not the start nonterminal)

Rules of the form $\underset{R}{Z} ::= \underset{\text{empty}}{\alpha}\ \underset{Q}{X}\ \underset{c}{\beta}$

$\boxed{R ::= \boxed{Q}\ \mathbf{c}}$ adds { **c** }

$R ::= \boxed{Q}\ S$

$R ::= \boxed{Q}\ Q$

$R ::= Q\ \boxed{Q}$

C_2: β is **c**, add FIRST(**c**) - ε = { **c** }

C_3: β is **c**, $\varepsilon \notin$ FIRST(**c**), so N/A

C_4: β is not empty, so N/A

Grammar

❶ *S* ::= **a**
❷ *S* ::= **b** *R*
❸ *Q* ::= *ε*
❹ *R* ::= *Q* **c**
❺ *R* ::= *Q S*
❻ *R* ::= *Q Q*

FOLLOW(*X*) for nonterminal *X*

C_1: If *X* is the start nonterminal, add **eof**

For all *Z* ::= α *X* β (where α and/or β may be empty)

- C_2: Add FIRST(β) – {ε}
- C_3: If ε is in FIRST(β) add FOLLOW(*Z*)
- C_4: If β is empty add FOLLOW(*Z*)

Repeat for each nonterminal until saturation

FIRST(S) = { a, b }
FIRST(Q) = { *ε* }
FIRST(R) = { **c**, **a**, **b**, *ε*}
FIRST(Q **c**) = { c }
FIRST(Q S) = { a, b }
FIRST(Q Q) = { *ε* }
FOLLOW(S) = { **eof** }
➡ FOLLOW(Q)
FOLLOW(R)

Building Follow(Q) (Q in for X)

C_1: N/A (*Q* not the start nonterminal)

Rules of the form $\underset{R}{Z} ::= \underset{empty}{\alpha}\ \underset{Q}{X}\ \underset{S}{\beta}$

$R ::= \boxed{Q}\,\mathbf{c}$ adds { **c** }

$\boxed{R ::= \boxed{Q}\,S}$ adds {**a**,**b**}

$R ::= \boxed{Q}\,Q$

$R ::= Q\,\boxed{Q}$

C_2: β is *S*, FIRST(*S*) - *ε* = { **a**, **b** }

C_3: β is *S*, *ε* ∉ FIRST(*S*), so N/A

C_4: β is not empty, so N/A

Grammar

❶ S ::= **a**
❷ S ::= **b** R
❸ Q ::= ε
❹ R ::= Q ***c***
❺ R ::= Q S
❻ R ::= Q Q

FOLLOW(X) for nonterminal X

C_1: If X is the start nonterminal, add **eof**

For all Z ::= α X β (where α and/or β may be empty)

C_2: Add FIRST(β) – {ε}
C_3: If ε is in FIRST(β) add FOLLOW(Z)
C_4: If β is empty add FOLLOW(Z)

Repeat for each nonterminal until saturation

FIRST(S) = { a, b }
FIRST(Q) = { ε }
FIRST(R) = { **c**, **a**, **b**, ε}
FIRST(Q **c**) = { c }
FIRST(Q S) = { a, b }
FIRST(Q Q) = { ε }
FOLLOW(S) = { **eof** }
➡ FOLLOW(Q)
FOLLOW(R)

Building Follow(Q) *(Q in for X)*

C_1: N/A (Q not the start nonterminal)

Rules of the form $\underset{R}{Z} ::= \underset{\text{empty}}{\alpha}\ \underset{Q}{X}\ \underset{Q}{\beta}$

R ::= $\boxed{Q}$**c** adds { **c** }

R ::= $\boxed{Q}$S adds {**a**,**b**}

R ::= $\boxed{Q}$Q adds { }

R ::= Q$\boxed{Q}$ adds { }

C_2: β is Q, FIRST(Q) - ε = { }

C_3: β is Q, Z is R, $\varepsilon \in$ FIRST(Q), add FOLLOW(R) = { }

C_4: β is not empty, so N/A

Grammar

❶ *S* ::= **a**
❷ *S* ::= **b** *R*
❸ *Q* ::= *ε*
❹ *R* ::= *Q* **c**
❺ *R* ::= *Q S*
❻ *R* ::= *Q Q*

FOLLOW(*X*) for nonterminal *X*

C_1: If *X* is the start nonterminal, add **eof**

For all *Z* ::= α *X* β (where α and/or β may be empty)

- C_2: Add FIRST(β) – {ε}
- C_3: If ε is in FIRST(β) add FOLLOW(*Z*)
- C_4: If β is empty add FOLLOW(*Z*)

Repeat for each nonterminal until saturation

FIRST(S) = { a, b }
FIRST(Q) = { *ε* }
FIRST(R) = { **c**, **a**, **b**, *ε*}
FIRST(Q **c**) = { c }
FIRST(Q S) = { a, b }
FIRST(Q Q) = { *ε* }
FOLLOW(S) = { **eof** }
➡ FOLLOW(Q)
FOLLOW(R)

Building Follow(Q) *(Q in for X)*

C_1: N/A (*Q* not the start nonterminal)

Rules of the form Z ::= α *X* β

R Q Q empty

R ::= $\boxed{Q}$ **c** adds { **c** }

R ::= $\boxed{Q}$ S adds {**a**,**b**}

R ::= $\boxed{Q}$ *Q* adds { }

$\boxed{\text{R ::= } Q \boxed{Q}}$ adds { }

C_2: β is empty, so add { }

C_3: β is empty, so N/A

C_4: β is not empty, Z is R, add FOLLOW(R) = { }

Grammar

❶ *S* ::= **a**
❷ *S* ::= **b** *R*
❸ *Q* ::= *ε*
❹ *R* ::= *Q* ***c***
❺ *R* ::= *Q S*
❻ *R* ::= *Q Q*

FOLLOW(*X*) for nonterminal *X*

C_1: If *X* is the start nonterminal, add **eof**

For all *Z* ::= α *X* β (where α and/or β may be empty)

- C_2: Add FIRST(β) – {ε}
- C_3: If ε is in FIRST(β) add FOLLOW(*Z*)
- C_4: If β is empty add FOLLOW(*Z*)

Repeat for each nonterminal until saturation

FIRST(S) = { a, b }
FIRST(Q) = { *ε* }
FIRST(R) = { **c**, **a**, **b**, *ε*}
FIRST(Q **c**) = { C }
FIRST(Q S) = { a, b }
FIRST(Q Q) = { *ε* }
FOLLOW(S) = { **eof** }
➡ FOLLOW(Q) = { **c**, **a**, **b** }
FOLLOW(R)

Building Follow(Q) (Q in for X)

C_1: N/A (*Q* not the start nonterminal)

Rules of the form Z ::= α *X* β

R ::= $\boxed{Q}$**c** adds { **c** }

R ::= $\boxed{Q}$S adds {**a**,**b**}

R ::= $\boxed{Q}$*Q* adds { }

R ::= *Q*$\boxed{Q}$ adds { }

Grammar

❶ *S* ::= **a**
❷ *S* ::= **b** *R*
❸ *Q* ::= *ε*
❹ *R* ::= *Q* ***c***
❺ *R* ::= *Q S*
❻ *R* ::= *Q Q*

FOLLOW(*X*) for nonterminal *X*

C_1: If *X* is the start nonterminal, add **eof**

For all *Z* ::= α *X* β (where α and/or β may be empty)

- C_2: Add FIRST(β) – {ε}
- C_3: If ε is in FIRST(β) add FOLLOW(*Z*)
- C_4: If β is empty add FOLLOW(*Z*)

Repeat for each nonterminal until saturation

FIRST(S) = { a, b }
FIRST(Q) = { *ε* }
FIRST(R) = { **c**, **a**, **b**, *ε*}
FIRST(Q **c**) = { c }
FIRST(Q S) = { a, b }
FIRST(Q Q) = { *ε* }
FOLLOW(S) = { **eof** }
FOLLOW(Q) = { **c**, **a**, **b** }
➡ FOLLOW(R)

Building Follow(R) *(R in for X)*

C_1: N/A (*R* not the start nonterminal)

Rules of the form $\underset{S}{Z} ::= \underset{b}{\alpha}\ \underset{R}{X}\ \underset{\text{empty}}{\beta}$

S ::= **b** $\boxed{R}$ adds { **eof** }

C_2: β is empty, add { }

C_3: β is empty, N/A

C_4: Z is S, add FOLLOW(S) = { **eof** }

Grammar

❶ *S* ::= **a**

❷ *S* ::= **b** *R*

❸ *Q* ::= ε

❹ *R* ::= *Q* ***c***

❺ *R* ::= *Q S*

❻ *R* ::= *Q Q*

FOLLOW(*X*) for nonterminal *X*

C_1: If *X* is the start nonterminal, add **eof**

For all *Z* ::= α *X* β (where α and/or β may be empty)

- C_2: Add FIRST(β) – {ε}
- C_3: If ε is in FIRST(β) add FOLLOW(*Z*)
- C_4: If β is empty add FOLLOW(*Z*)

Repeat for each nonterminal until saturation

FIRST(S) = { a, b }

FIRST(Q) = { ε }

FIRST(R) = { **c**, **a**, **b**, ε}

FIRST(Q **c**) = { c }

FIRST(Q S) = { a, b }

FIRST(Q Q) = { ε }

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

FOLLOW(Q) = { **c**, **a**, **b** }

➡ FOLLOW(R) = { **eof** }

S ::= **b** $\boxed{R}$ adds { **eof** }

Grammar

❶ *S* ::= **a**
❷ *S* ::= **b** *R*
❸ *Q* ::= ε
❹ *R* ::= *Q* **c**
❺ *R* ::= *Q S*
❻ *R* ::= *Q Q*

FOLLOW(*X*) for nonterminal *X*

C_1: If *X* is the start nonterminal, add **eof**

For all *Z* ::= α *X* β (where α and/or β may be empty)

- C_2: Add FIRST(β) – {ε}
- C_3: If ε is in FIRST(β) add FOLLOW(*Z*)
- C_4: If β is empty add FOLLOW(*Z*)

Repeat for each nonterminal until saturation

FIRST(S) = { a, b }
FIRST(Q) = { ε }
FIRST(R) = { **c**, **a**, **b**, ε}
FIRST(Q **c**) = { c }
FIRST(Q S) = { a, b }
FIRST(Q Q) = { ε }
FOLLOW(S) = { **eof** }
FOLLOW(Q) = { **c**, **a**, **b** }
➡ FOLLOW(R) = { **eof** }

All done?

Grammar

❶ *S* ::= **a**
❷ *S* ::= **b** *R*
❸ *Q* ::=
❹ *R*

FOLLOW(*X*) for nonterm

C$_1$: If *X* is the start n

For all *Z* ::= α *X* β (w

C$_2$: Add FIRST(β)

If n FIRS

a

er

Recall computing FOLLOW(Q)

We used a set that later changed!

in for

nonterminal)

m Z ::= α *X* β

R *Q* *Q* *empty*

FIR

FIRST(Q S) = {

FIRST(Q Q) = { ε

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

FOLLOW(Q)

FOLLOW(R)

c }

R ::= Q S adds {**a**,**b**}

R ::= *Q* *Q* adds { }

R ::= *Q* *Q* adds { }

C$_2$: β is empty, so add { }

C$_3$: β is empty, so N/A

C$_4$: β is not empty, Z is R, add FOLLOW(R) = { }

Grammar

❶ S ::= **a**
❷ S ::= **b** R
❸ Q ::= ε
❹ R ::= Q ***c***
❺ R ::= Q S
❻ R ::= Q Q

FOLLOW(X) for nonterminal X

C_1: If X is the start nonterminal, add **eof**

For all Z ::= α X β (where α and/or β may be empty)

C_2: Add FIRST(β) – {ε}
C_3: If ε is in FIRST(β) add FOLLOW(Z)
C_4: If β is empty add FOLLOW(Z)

Repeat for each nonterminal until saturation

FIRST(S) = { a, b }
FIRST(Q) = { ε }
FIRST(R) = { **c**, **a**, **b**, ε}
FIRST(Q **c**) = { c }
FIRST(Q S) = { a, b }
FIRST(Q Q) = { ε }
FOLLOW(S) = { **eof** }
FOLLOW(Q) = { **c**, **a**, **b** }
➡ FOLLOW(R) = { **eof** }

PSA
Run FOLLOW and FIRST computations until saturation

Round 2

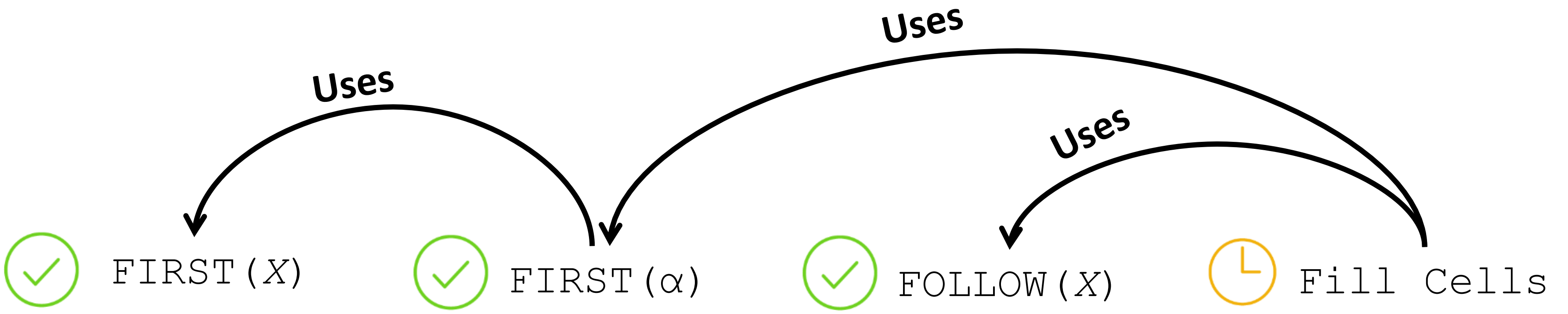
FOLLOW(S) = { **eof }**
FOLLOW(Q) = { **c, a, b, eof** }
FOLLOW(R) = { **eof** }

Round 3

FOLLOW(S) = { **eof** }
FOLLOW(Q) = { **c, a, b, eof** }
FOLLOW(R) = { **eof** }

Review: Set Dependencies

Building LL(!) Selector Table



✓ `FIRST(X)`

Step 1.

Build FIRST sets for single symbols of the grammar

✓ `FIRST(α)`

Step 2.

Build FIRST sets for RHS symbols strings

✓ `FOLLOW(X)`

Step 3.

Build FOLLOW sets for Nonterminals

◷ `Fill Cells`

Step 4.

Place productions

LL(1) Selector Table Algorithm

Building LL(1) Selector Table

```
for each production $X$ ::= $\alpha$
   for each terminal t in FIRST($\alpha$)
      put X ::= $\alpha$ in Table[$X$][t]
   if $\varepsilon$ is in FIRST($\alpha$)
      for each t in FOLLOW(X)
         put X::= $\alpha$ in Table[$X$][t]
```

LL(1) Selector Table Algorithm

Building LL(1) Selector Table

Time permitting: Examples

Table[X][t]

```
for each production $X$ ::= $\alpha$
 for each terminal t in FIRST($\alpha$)
   put $X$ ::= $\alpha$ in Table[$X$][t]
 if ε is in FIRST($\alpha$)
   for each terminal t in FOLLOW(X)
     put $X$ ::= $\alpha$ in Table[$X$][t]
```

CFG

S ::= B **c** | D B

B ::= **a b** | **c** S

D ::= **d** | ε

FIRST (S) = { **a, c, d** }

FIRST (B) = { **a, c** }

FIRST (D) = { **d**, ε }

FIRST (B **c**) = { **a, c** }

FIRST (D B) = { **d, a, c** }

FIRST (**a b**) = { **a** }

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

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

FOLLOW (B) = { **c**, **eof** }

FOLLOW (D) = { **a, c** }

	a	**b**	**c**	**d**	**eof**
S					
B	**a b**				
D					

For each production X ::= α

B ::= **a b**

B **a b**

Look at terminals in FIRST(α) = { **a** }:

Put B ::= **a b** @ Table[*B*][**a**]

ε is not in FIRST(α) = { **a** }:

Done with this production

Table[X][t]

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

CFG

S	::=	B **c** | D B
B	::=	**a b** | **c** S
D	::=	**d** | ε

FIRST (S) = { **a, c, d** }
FIRST (B) = { **a, c** }
FIRST (D) = { **d**, ε }
FIRST (B **c**) = { **a, c** }
FIRST (D B) = { **d, a, c** }
FIRST (**a b**) = { **a** }
FIRST (**c** *S*) = { **c** }

FOLLOW (S) = { **eof, c** }
FOLLOW (B) = { **c**, **eof** }
FOLLOW (D) = { **a, c** }

	a	**b**	**c**	**d**	**eof**
S					
B	**a b**				
D	ε		ε		

For each production X ::= α

$D ::= \varepsilon$ D ε

Look at terminals in FIRST(α) = { ε }

There are none

Because ε is in FIRST(α)

Look at everything in Follow(X) = { **a**, **c** }

Put D ::= ε @ Table[*D*][**a**]

Put D ::= ε @ Table[*D*][**c**]

Table[X][t]

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

CFG

S ::= B **c** | D B

B ::= **a b** | **c** S

D ::= **d** | ε

FIRST (S) = { **a, c, d** }
FIRST (B) = { **a, c** }
FIRST (D) = { **d**, ε }
FIRST (B **c**) = { **a, c** }
FIRST (D B) = { **d, a, c** }
FIRST (**a b**) = { **a** }
FIRST (**c** *S*) = { **c** }

FOLLOW (S) = { **eof, c** }
FOLLOW (B) = { **c**, **eof** }
FOLLOW (D) = { **a, c** }

	a	**b**	**c**	**d**	**eof**
S	*D B*		*D B*		
B	**a b**				
D	ε		ε		

For each production X ::= α

S ::= D B   *S*   *D B*

Look at terminals in FIRST(α) = { **d, a, c** }

Put S ::= D B @ Table[S][**d**]

Put S ::= D B @ Table[S][**a**]

Put S ::= D B @ Table[S][**c**]

ε is not in FIRST(α) = { **d**, **a, c** }:

Done with this production

Table[X][t]

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

CFG

S ::= B **c** | D B

B ::= **a b** | **c** S

D ::= **d** | ε

FIRST (S) = { **a, c, d** }

FIRST (B) = { **a, c** }

FIRST (D) = { **d**, ε }

FIRST (B **c**) = { **a, c** }

FIRST (D B) = { **d, a, c** }

FIRST (**a b**) = { **a** }

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

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

FOLLOW (B) = { **c**, **eof** }

FOLLOW (D) = { **a, c** }

	a	**b**	**c**	**d**	**eof**
S	*D B* *B* **c**		*D B* *B* **c**		
B	**a b**				
D	*ε*		*ε*		

For each production X ::= α

S ::= B **c** *S* *B* **c**

Look at terminals in FIRST(α) = { **a, c** }

Put S ::= B C @ Table[S][**a**]

Put S ::= B C @ Table[S][**c**]

ε is not in FIRST(α) = { **a** }:

Done with this production

Table[X][t]

```
for each production X ::= α
 for each           in FIRST(α)
   put              e[X][t]
 if ε is
   for e            t in FOLLOW(X)
     put            able[X][t]
```

Collision! Grammar is not LL(1)

CFG

S ::= B **c** | D B

B ::= **a b** | **c** S

D ::= **d** | ε

FIRST (S) = { **a, c, d** }

FIRST (B) = { **a, c** }

FIRST (D) = { **d**, ε }

FIRST (B **c**) = { **a, c** }

FIRST (D B) = { **d, a, c** }

FIRST (**a b**) = { **a** }

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

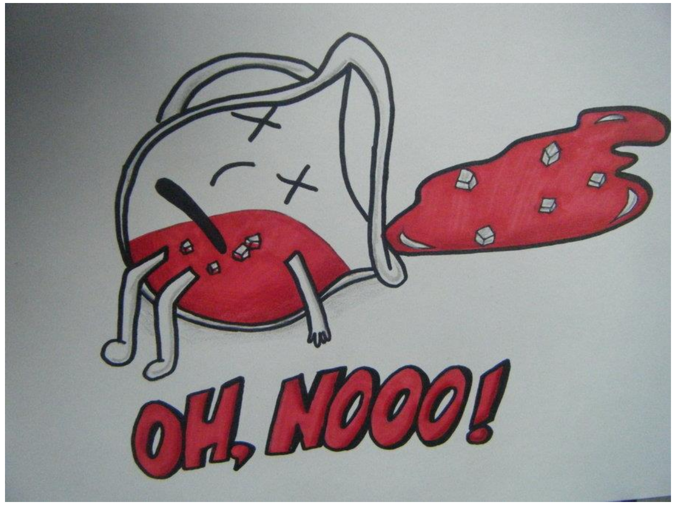
FOLLOW (S) = { **eof, c** }

FOLLOW (B) = { **c, eof** }

FOLLOW (D) = { **a, c** }

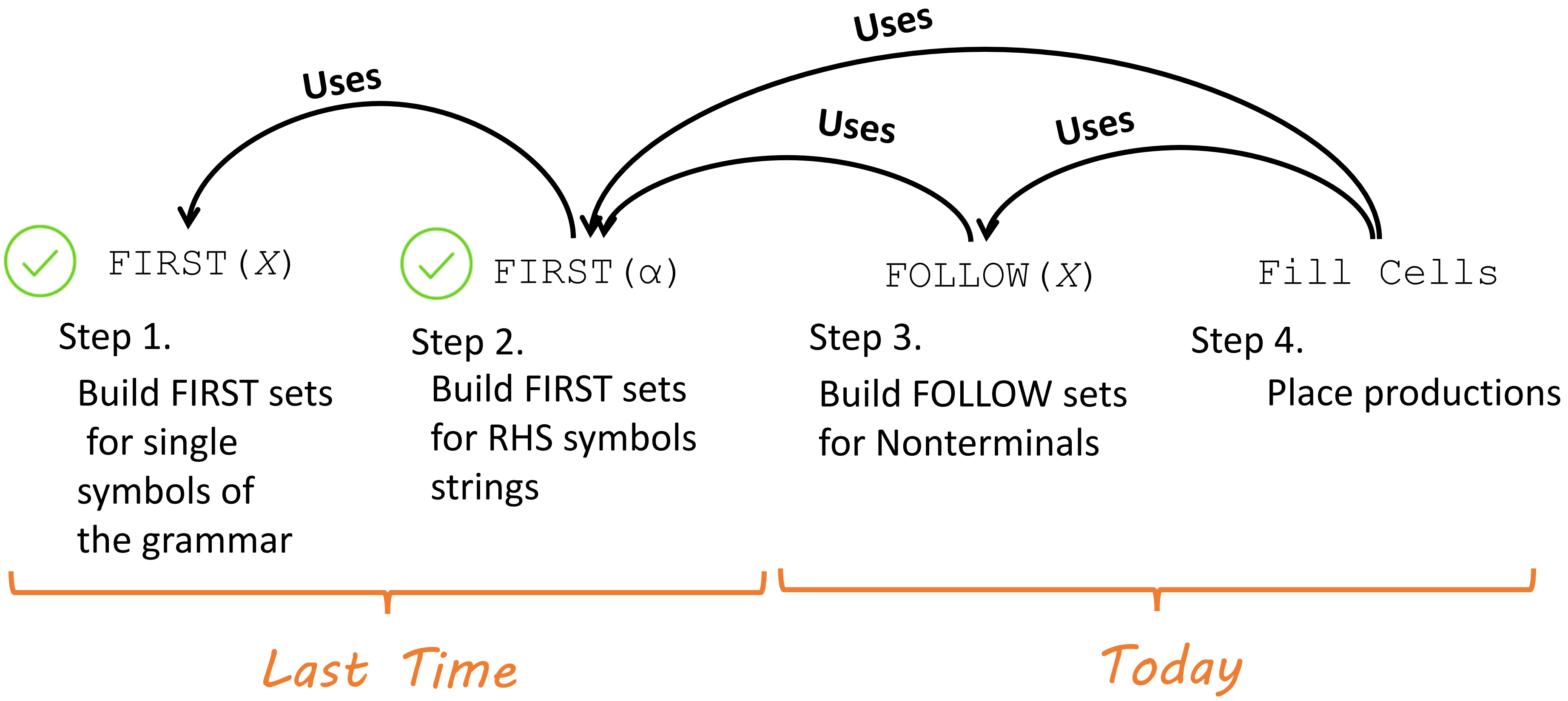
	a	**b**	**c**	**d**	**eof**
S	*D B* *B* **c**		*D B* *B* **c**		
B	**a b**				
D	*ε*		*ε*		

Another Collision! Grammar is still not LL(1)



Review: Selector Table Dependencies

Review Lecture 9 – FIRST Sets



✓ `FIRST(X)`

Step 1.

Build FIRST sets for single symbols of the grammar

✓ `FIRST(α)`

Step 2.

Build FIRST sets for RHS symbols strings

`FOLLOW(X)`

Step 3.

Build FOLLOW sets for Nonterminals

`Fill Cells`

Step 4.

Place productions

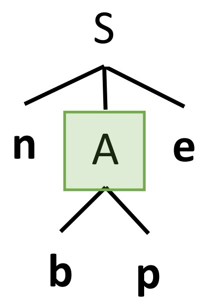
Last Time

Today

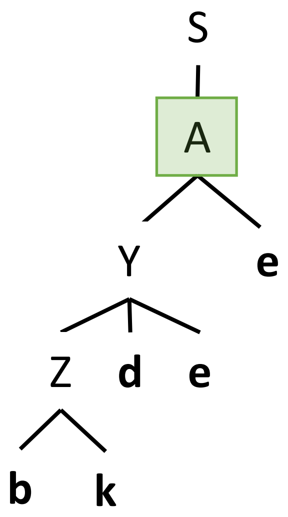
A Parse Tree Perspective

Building LL(1) Selector Table: FIRST sets, single symbol

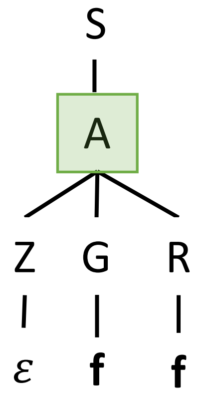
FIRST(X): The set of terminals that begin strings derivable from X, and also, if X can derive ε, then ε is in FIRST(X).



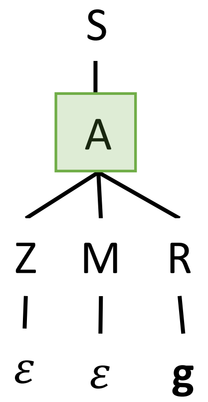
b ∈ FIRST(A)



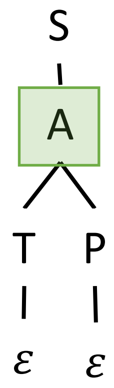
b ∈ FIRST(A)



f ∈ FIRST(A)



g ∈ FIRST(A)



ε ∈ FIRST(A)