

University of Kansas | Drew Davidson

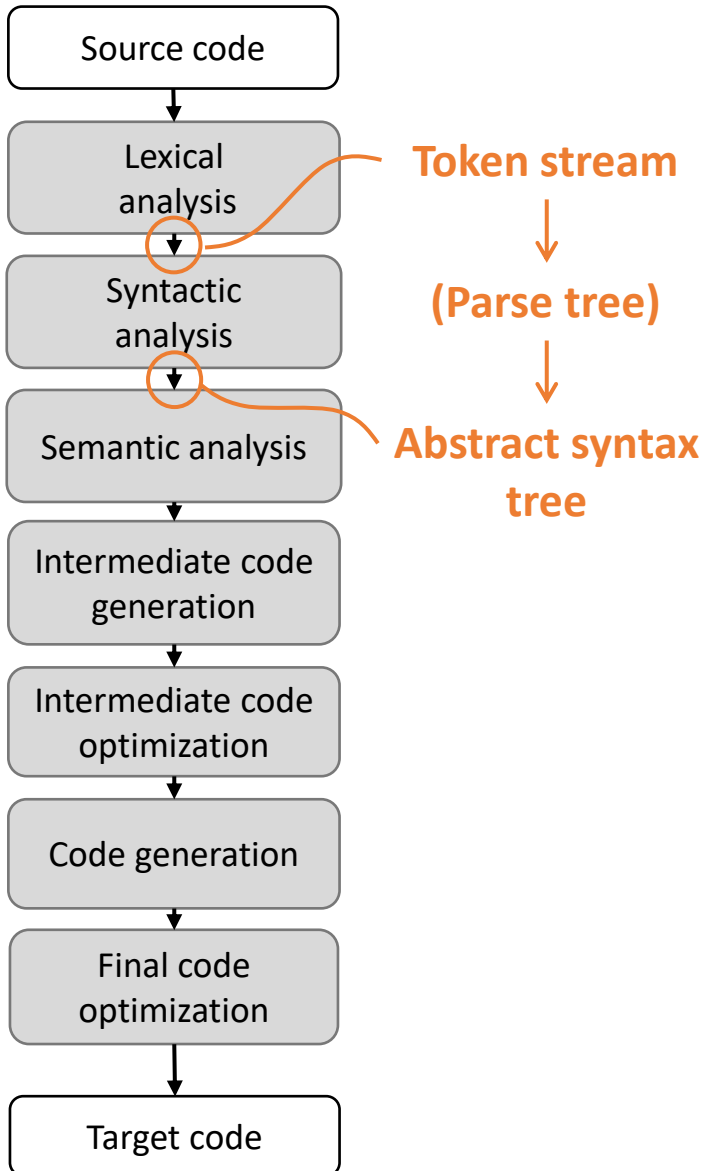
ECS 665

COMPILER *CONSTRUCTION*

(Predictive) Parsing

Compiler Construction

Progress Pics



Syntactic Analysis

- We know ***what*** we want (induce an AST from a given token stream)

Last Time

Review - ASTs

Syntax-Directed Definition

- Showed how to use SDD to induce program's AST
- We can *specify* a correct syntactic analysis

You Should Know

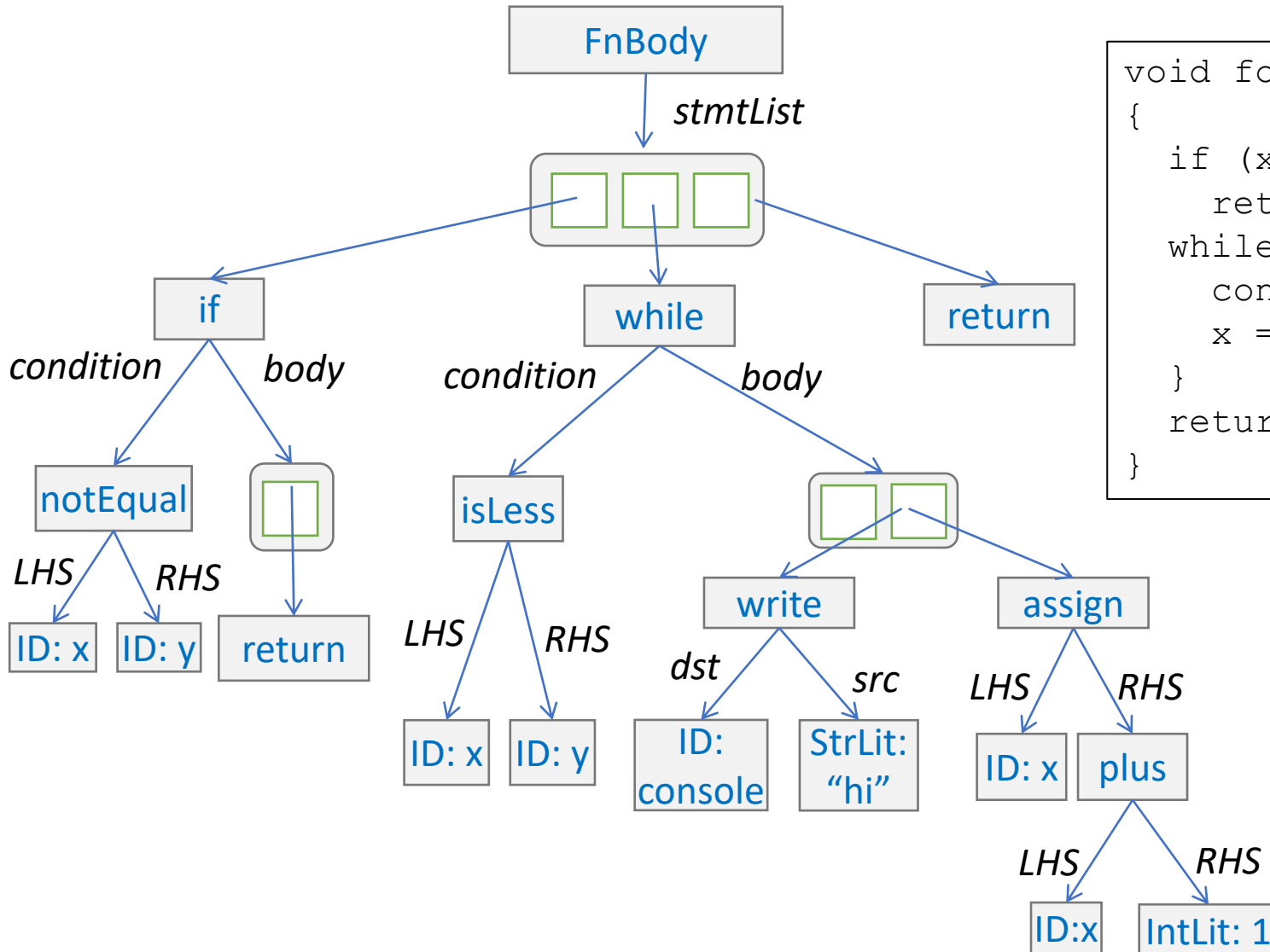
- The concepts of Parse Trees / ASTs
- How to specify syntax-directed definitions (to induce scalar values, data structures, etc. from parse trees)



**Syntax-Directed
Definition**

AST For Input Code Snippet

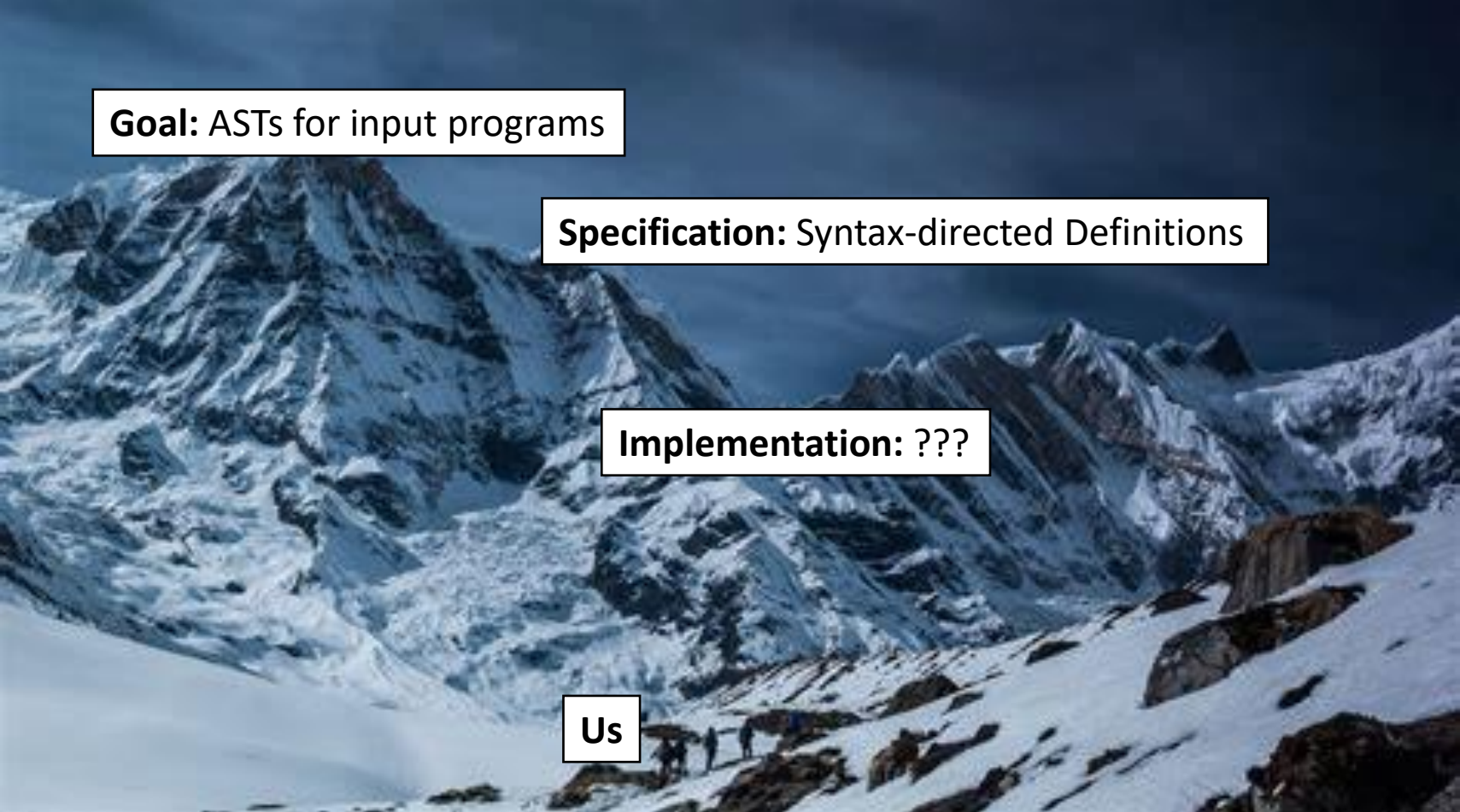
Abstract Syntax Trees – ASTs in Code



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

The Goal and The Path

Abstract Syntax Trees – ASTs in Code



Goal: ASTs for input programs

Specification: Syntax-directed Definitions

Implementation: ???

Us

COMPILER

LAND

Code
Generation

Execution

Runtime
Environment

Intermediate
Representation

Optimization

SDD

Semantics

Parsing

Lexical
Analysis

Syntactic
Definition



Today's Outline

(Predictive) Parsing

Parsing

- Complexity

The LL(k) Languages

- Intro
- LL(1) parsing
- LL(1) transformations



Parsing

The Price of Expressive Power

(Predictive) Parsing – Parsing Complexity

Parsing is costly

- Jumping from Regular Languages to CF-Languages incurs penalties
 - Lose lots of language properties
 - Complexity for some operations goes up



Power always comes at a price

Parsing: Bad News/Good News

(Predictive) Parsing – Parsing Complexity

Bad News

- Complexity - CFL recognition:
 - CYK: $O(n^3)$
 - Best known bound: $O(n^{2.3728639})$
- Parsing is at least as hard

Good News

- We don't need to recognize all CFLs
 - We can assume ambiguity-free syntax
 - Exchange expressiveness for faster parsers



Scaling Back Expressive Power

(Predictive) Parsing – Parsing Complexity

Compilers aren't concerned with arbitrary languages

- Programming languages *should* be unambiguous
- Complex operations can be built from simple syntax

*Paying for features
we don't need!*



“Don't buy the truck if
you ain't haulin' freight”
-Drew Davidson

Restricting the Language Class

(Predictive) Parsing – Parsing Complexity

- Allows us to...
 - Build linear-time parsers
 - Detect ambiguity
- Let's consider one such class of restrictions



Disallowing Heavyweight Languages

Today's Outline

Preview Lecture 8 – Predictive Parsing

Parsing

- Complexity

The LL(k) languages

- Intro
- LL(1) parsing
- LL(1) transformations

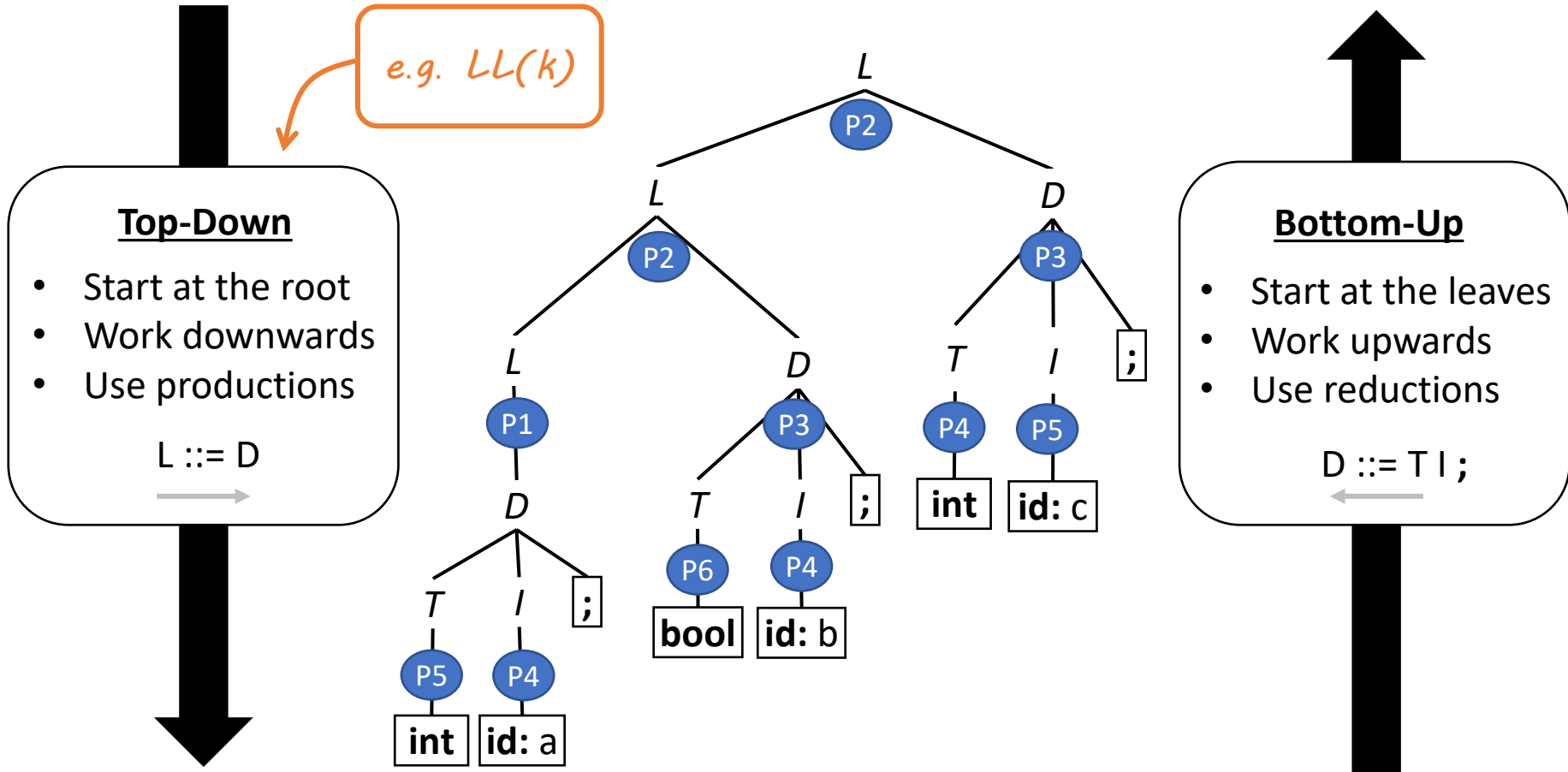


***Toyota Yaris: A sensible compact
LL(1) languages are the Yaris of the
Parsing world***

Parser Type

LL(k) Parsers - Definitions

Most parsers classified by how they build the parse tree



Input String:

```
int a; bool b; int c;
```

Predictive Parsers

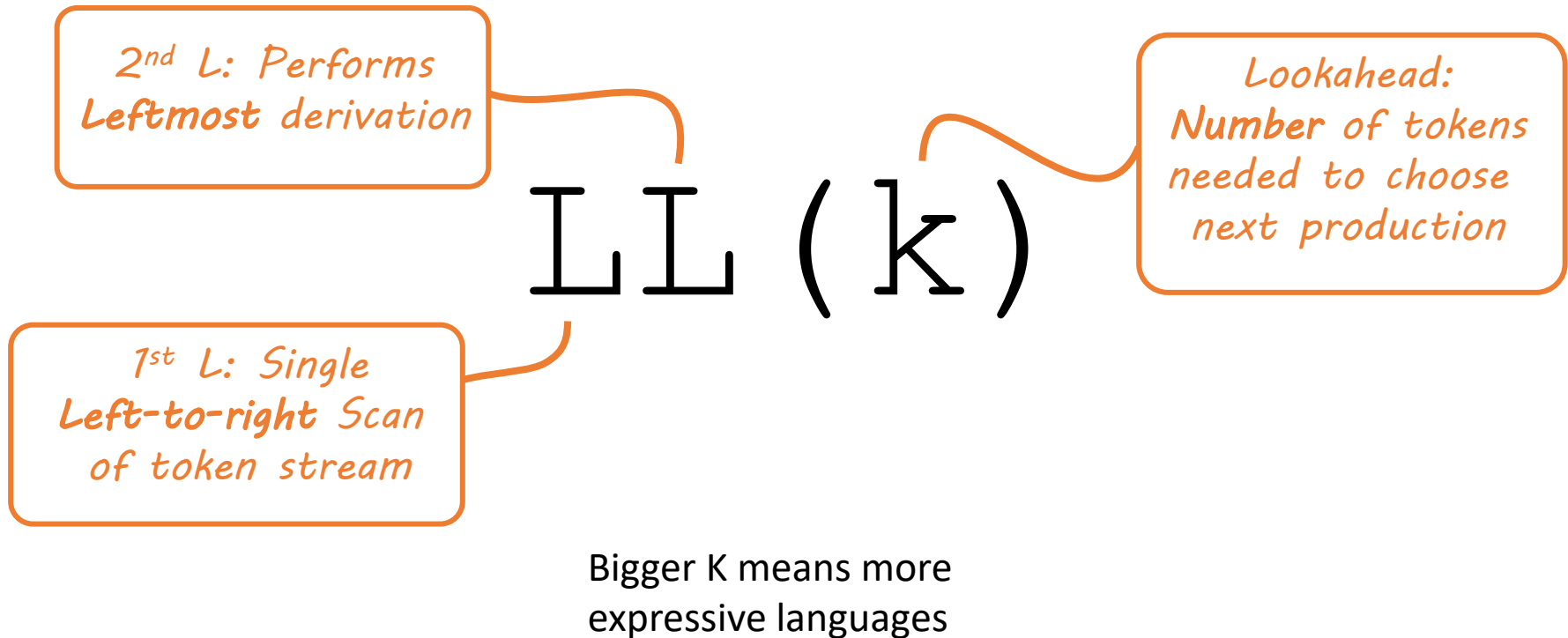
LL(k) Parsers - Definitions

- “Predict” correct production for the given nonterminal before seeing all RHS symbols
- Based on input prefix
“Guess and check” the suffix



LL Grammars: Anatomy of Notation

LL Grammars - Intro



LL Grammar Examples

LL Grammars - Intro

LL(0) Grammars

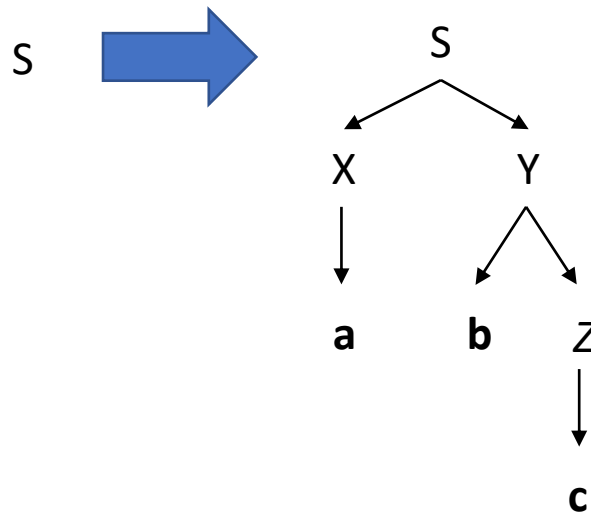
Takes leftmost derivation
step knowing **zero**
undervived input tokens

$S ::= X Y$

$X ::= a$

$Y ::= b Z$

$Z ::= c$



LL Grammar Examples

LL Grammars - Intro

LL(0) Grammars

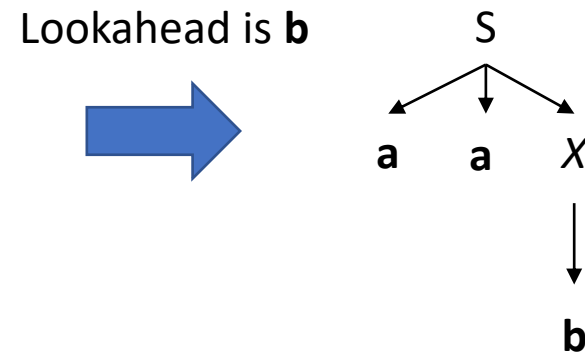
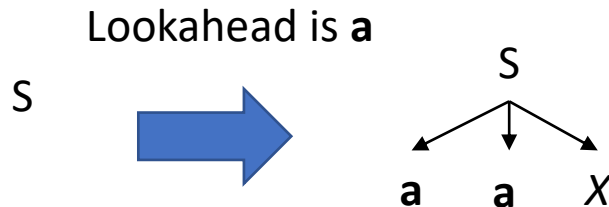
Takes leftmost derivation
step knowing **zero**
underived input tokens

$S ::= X Y$
 $X ::= a$
 $Y ::= b Z$
 $Z ::= c$

LL(1) Grammars

Takes leftmost derivation
step knowing **one**
underived input token

$S ::= a a X$
 | $b X$
 $X ::= a$
 | b



LL Grammar Examples

LL Grammars - Intro

LL(0) Grammars

Takes leftmost derivation
step knowing **zero**
underived input tokens

$$\begin{aligned} S &::= X Y \\ X &::= a \\ Y &::= b Z \\ Z &::= c \end{aligned}$$

LL(1) Grammars

Takes leftmost derivation
step knowing **one**
underived input token

$$\begin{aligned} S &::= a a X \\ &\quad | b X \\ X &::= a \\ &\quad | b \end{aligned}$$

LL(2) Grammars

Takes leftmost derivation
step knowing **two**
underived input tokens

$$\begin{aligned} S &::= a a \\ &\quad | a b \end{aligned}$$

(Lookahead **a** is insufficient)

Lookahead is **a a**



LL Grammar Examples

LL Grammars - Intro

LL(1) Grammars

Efficient
table-driven
parsers

Takes leftmost derivation
step knowing **one**
underived input token

Row:
leftmost leaf
nonterminal

Column
Lookahead
token

Cell:
Production

LL Grammar Examples

LL Grammars - Intro

With some grammars you can
just eyeball the selector table

$$\begin{array}{l} S ::= (S) \\ \quad | \{ S \} \\ \quad | \epsilon \end{array}$$

Row:
leftmost leaf
nonterminal

	(	{	)	}
S	(S)	{ S }	ϵ	ϵ

Column
Lookahead
token

Cell:
Production

Today's Outline

(Predictive) Parsing

Parsing

- Complexity

A New Type of Language - LL

- Intro
- LL(1) parsing
- LL(1) transformations



Parsing

LL(1) Algorithm Psuedocode

Predictive Parsing - LL(1) Parsing

```
stack.push(eof)
stack.push(Start non-term)
lookahead = scanner.first_token()
Repeat
    if stack.top is a terminal
        match stack.top with lookahead
        pop y from the stack
        lookahead = scanner.next_token()
    if stack.top is a nonterminal
        X = stack.pop()
        get P = table[X,lookahead]
        push P's RHS symbols Right-to-Left
Until one of the following:
    stack is empty (accept)
    stack.top is a terminal that doesn't match t (reject)
    stack.top is a non-term and table entry is empty (reject)
```

Initialization

Case 1
Terminal on stack:
check prediction

Case 2
Non-terminal on stack:
Derive new prediction

Exit

Example Grammar

$S ::= (S)$
 $\quad | \{ S \}$
 $\quad | \epsilon$

LL(1) Example

(Predictive) Parsing - LL(1) Parsing

Initialization

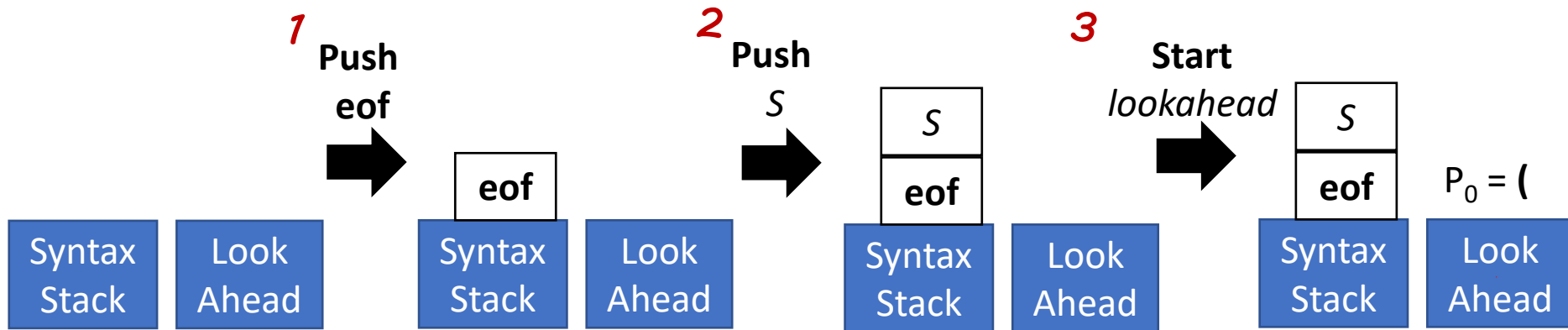
- 1 `stack.push(eof)`
- 2 `stack.push(Start non-term)`
- 3 `lookahead = scanner.first_token()`

Selector Table

	(	)	{	}
S	(S)	ϵ	{ S }	ϵ

Input Stream

({ }) **eof**
 P_0 P_1 P_2 P_3 P_4



Example Grammar

$S ::= (S)$
 $\quad | \{ S \}$
 $\quad | \epsilon$

LL(1) Example

(Predictive) Parsing - LL(1) Parsing

Initialization

- 1 `stack.push(eof)`
- 2 `stack.push(Start non-term)`
- 3 `lookahead = scanner.first_token()` “Predicted”

Selector Table

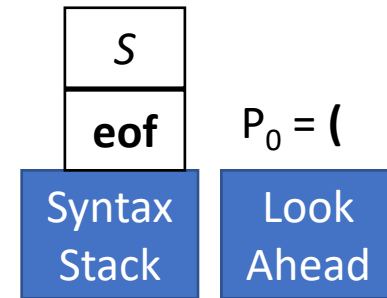
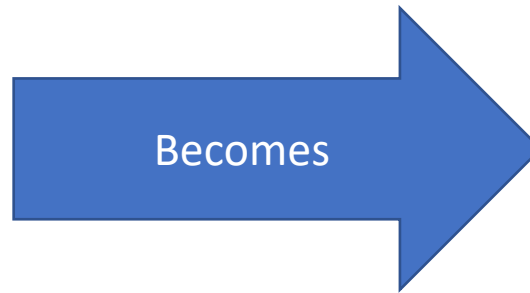
	(	)	{	}
S	(S)	ϵ	{ S }	ϵ

Input Stream

(	{	}	)	eof
P ₀	P ₁	P ₂	P ₃	P ₄

Derivation Sequence

S



Example Grammar

$S ::= (S)$
 $\quad | \{ S \}$
 $\quad | \epsilon$

LL(1) Example

(Predictive) Parsing - LL(1) Parsing

Nonterminal on stack:

- 1 $X = \text{pop}(\text{stack})$
- 2 $\text{get } P = \text{table}[X, \text{lookahead}]$
- 3 push P 's symbols *Right-to-Left*

Selector Table

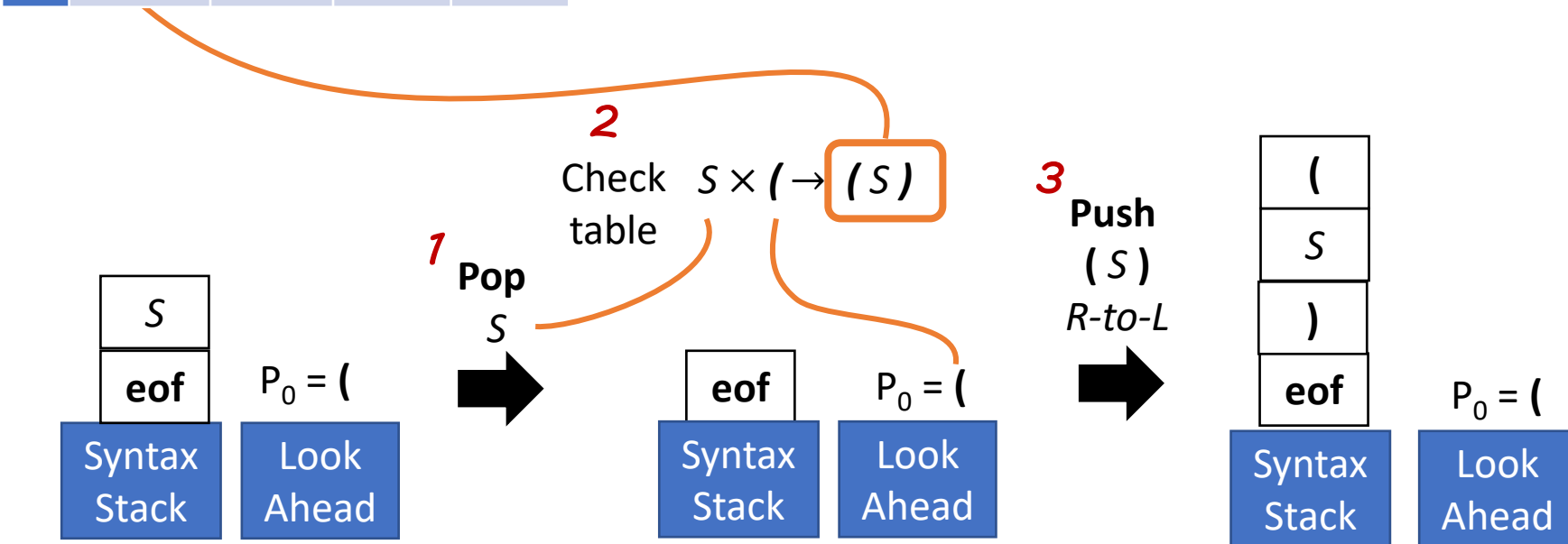
	(	)	{	}
S	(S)	ϵ	{ S }	ϵ

Input Stream

(	{	}	)	eof
P_0	P_1	P_2	P_3	P_4

"Predicted" Derivation Sequence

S



Example Grammar

$S ::= (S)$
 $\quad | \{ S \}$
 $\quad | \epsilon$

LL(1) Example

(Predictive) Parsing - LL(1) Parsing
Nonterminal on stack:

- 1 $X = \text{pop}(\text{stack})$
- 2 $\text{get } P = \text{table}[X, \text{lookahead}]$
- 3 push P 's symbols *Right-to-Left*

Selector Table

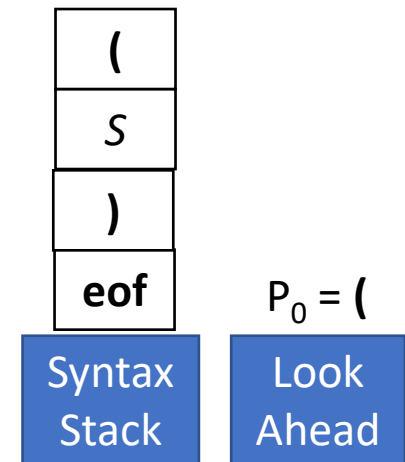
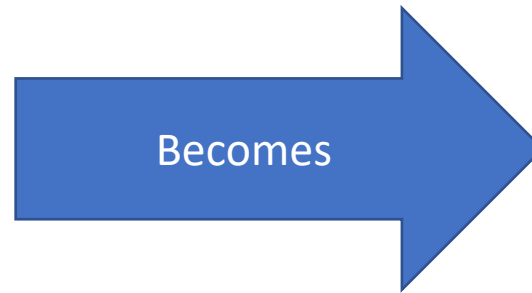
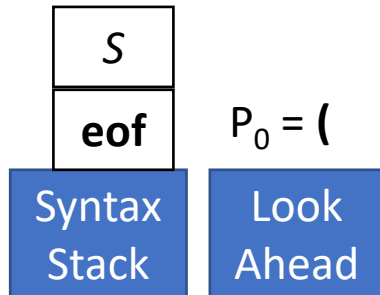
	(	)	{	}
S	(S)	ϵ	{ S }	ϵ

Input Stream

({ }) eof
 P_0 P_1 P_2 P_3 P_4

“Predicted” Derivation Sequence

$S \Rightarrow (S)$



Example Grammar

$$S ::= (S)$$
$$| \{ S \}$$
$$| \epsilon$$

LL(1) Example

(Predictive) Parsing - LL(1) Parsing

Terminal on stack:

- 1** match stack.top with lookahead
- 2** pop stack
- 3** lookahead = scanner.next_token() “Predicted”

Selector Table

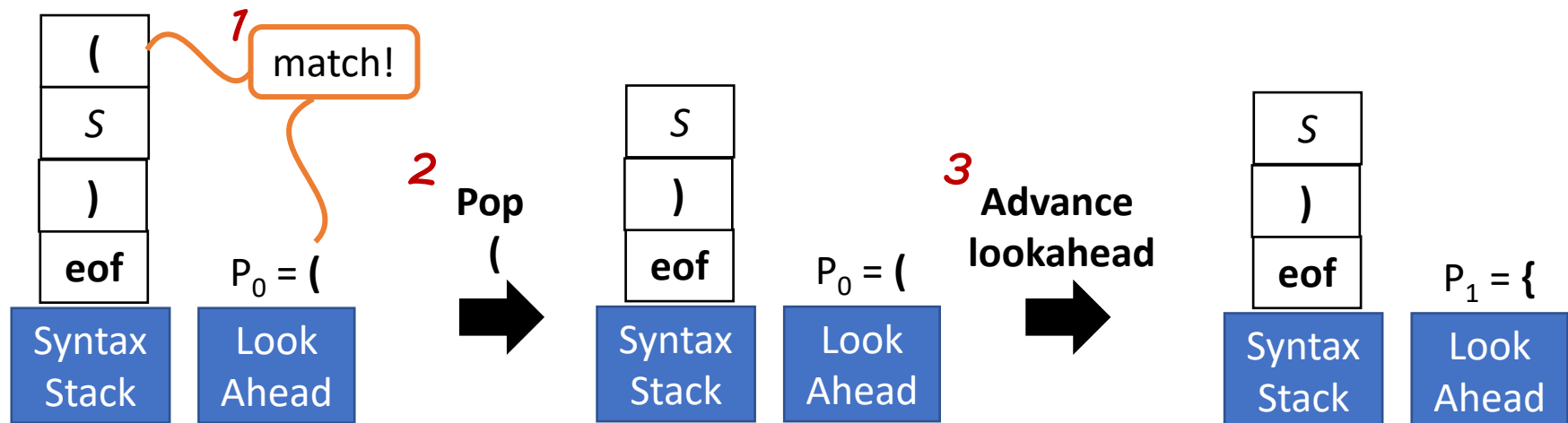
	(	)	{	}
S	(S)	ϵ	{ S }	ϵ

Input Stream

(	{	}	)	eof
P ₀	P ₁	P ₂	P ₃	P ₄

Derivation Sequence

$S \Rightarrow (S)$



Example Grammar

$S ::= (S)$
 $\quad | \{ S \}$
 $\quad | \epsilon$

LL(1) Example

(Predictive) Parsing - LL(1) Parsing

Terminal on stack:

- 1** match `stack.top` with lookahead
- 2** pop stack
- 3** `lookahead = scanner.next_token()` “Predicted”

Selector Table

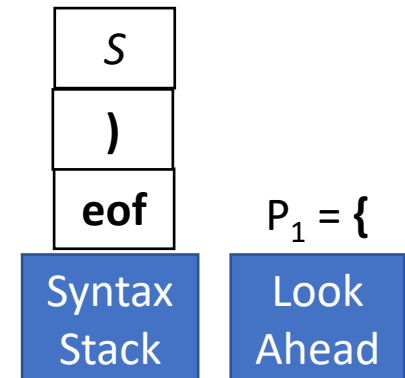
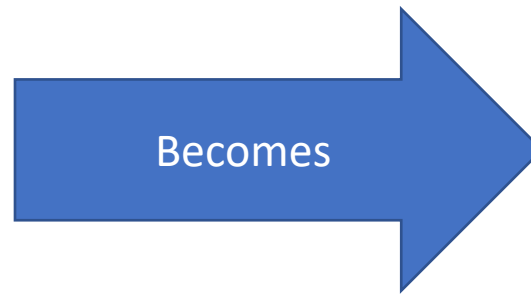
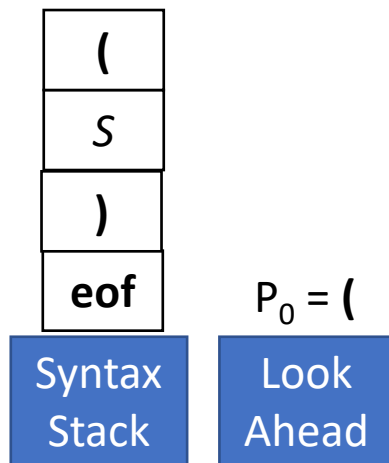
	(	)	{	}
S	(S)	ϵ	{ S }	ϵ

Input Stream

({ }) eof
 P_0 P_1 P_2 P_3 P_4
✓

Derivation Sequence

$S \Rightarrow (S)$



Example Grammar

$S ::= (S)$
 $\quad | \{ S \}$
 $\quad | \epsilon$

LL(1) Example

(Predictive) Parsing - LL(1) Parsing

Nonterminal on stack:

- 1 $X = \text{pop}(\text{stack})$
- 2 $\text{get } P = \text{table}[X, \text{lookahead}]$
- 3 $\text{push } P\text{'s RHS symbols Right-to-Left}$

Selector Table

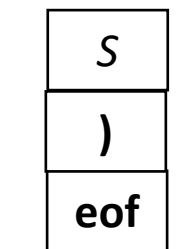
	(	)	{	}
S	(S)	ϵ	{ S }	ϵ

Input Stream

({ }) eof
 $P_0 \quad P_1 \quad P_2 \quad P_3 \quad P_4$

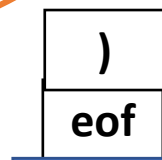
"Predicted" Derivation Sequence

$S \Rightarrow (S)$

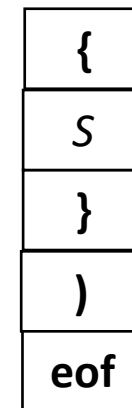


2 Check table $S \times \{ \rightarrow \{ S \}$

1 Pop S



3 Push { S }
R-to-L



Syntax
Stack

Look
Ahead

Syntax
Stack

Look
Ahead

Syntax
Stack

Look
Ahead

Example Grammar

$S ::= (S)$
 $\quad | \{ S \}$
 $\quad | \epsilon$

LL(1) Example

(Predictive) Parsing - LL(1) Parsing

Nonterminal on stack:

1 $X = \text{pop}(\text{stack})$

2 get $P = \text{table}[X, \text{lookahead}]$

3 push P 's RHS symbols *Right-to-Left*

Selector Table

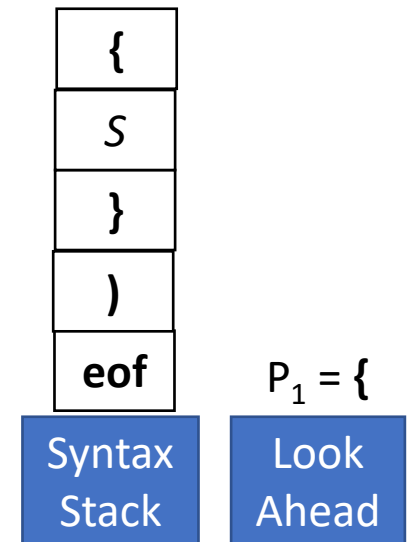
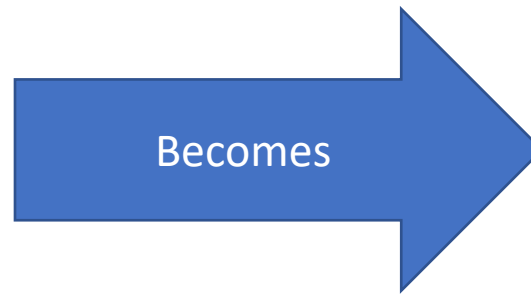
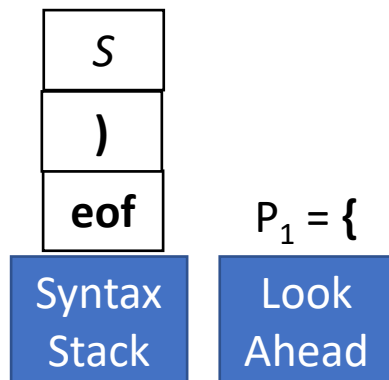
	(	)	{	}
S	(S)	ϵ	{ S }	ϵ

Input Stream

({ }) eof
 P_0 P_1 P_2 P_3 P_4
✓

“Predicted” Derivation Sequence

$S \Rightarrow (S)$
 $\Rightarrow (\{ S \})$



Example Grammar

$S ::= (S)$
 $\quad | \{ S \}$
 $\quad | \epsilon$

LL(1) Example

(Predictive) Parsing - LL(1) Parsing

Terminal on stack:

- 1 match stack.top with lookahead
- 2 pop stack
- 3 lookahead = scanner.next_token()

"Predicted"

Selector Table

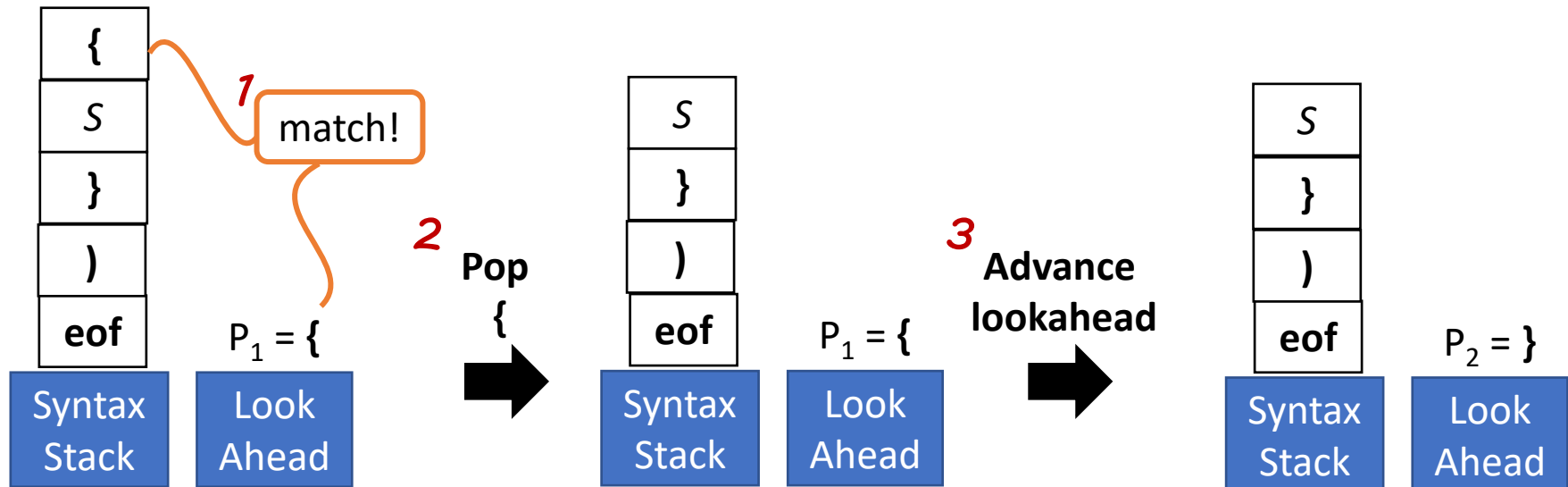
	(	)	{	}
S	(S)	ϵ	{ S }	ϵ

Input Stream

({ }) eof
P₀ P₁ P₂ P₃ P₄
✓

Derivation Sequence

$S \Rightarrow (S)$
 $\Rightarrow (\{ S \})$



Example Grammar

$S ::= (S)$
 $\quad | \{ S \}$
 $\quad | \epsilon$

LL(1) Example

(Predictive) Parsing - LL(1) Parsin)

Terminal on stack:

- 1 match `stack.top` with `lookahead`
- 2 pop stack
- 3 `lookahead = scanner.next_token()`

Selector Table

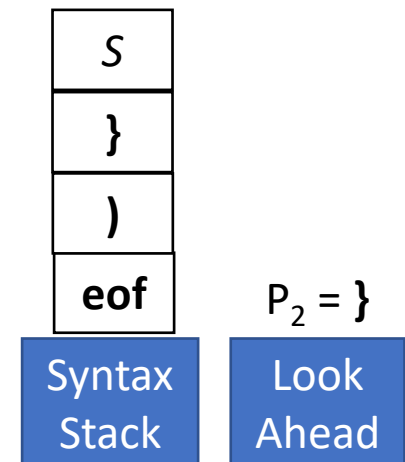
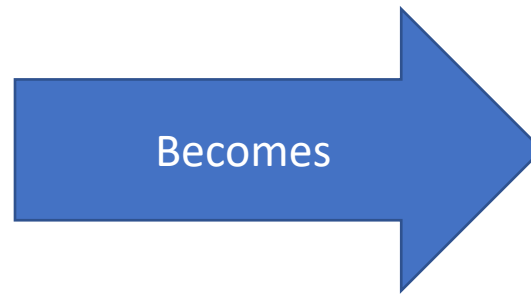
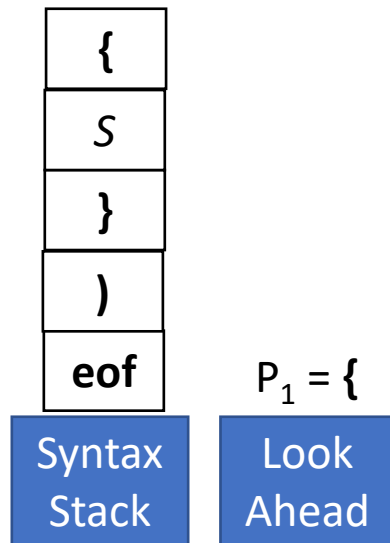
	(	)	{	}
S	(S)	ϵ	{ S }	ϵ

Input Stream

(	{	}	)	eof
P ₀	P ₁	P ₂	P ₃	P ₄
✓	✓			

“Predicted” Derivation Sequence

$S \Rightarrow (S)$
 $\Rightarrow (\{ S \})$



Example Grammar

$$S ::= (S)$$
$$| \{ S \}$$
$$| \epsilon$$

LL(1) Example

(Predictive) Parsing - LL(1) Parsin

Nonterminal on stack:

1 $X = \text{pop}(\text{stack})$

2 $\text{get } P = \text{table}[X, \text{lookahead}]$

3 push P 's RHS symbols *Right-to-Left* "Predicted"

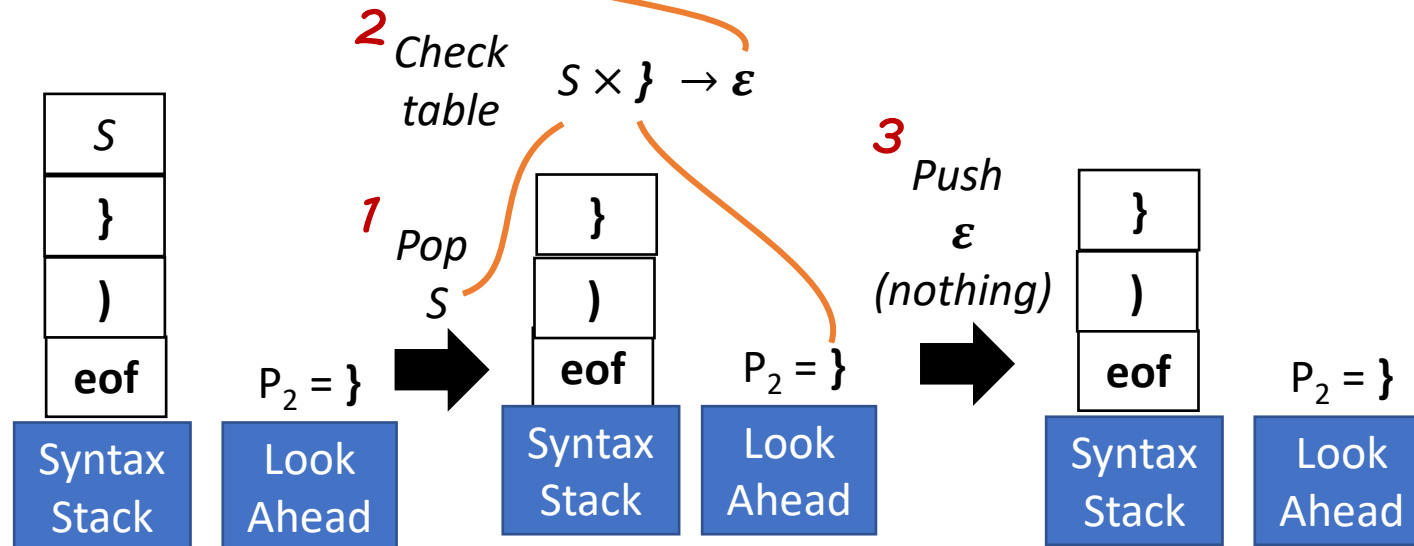
Selector Table

	(	)	{	}
S	(S)	ϵ	{ S }	ϵ

Input Stream

(	{	}	)	eof
P_0	P_1	P_2	P_3	P_4
✓	✓			

Derivation Sequence

$$S \Rightarrow (S)$$
$$\Rightarrow (\{ S \})$$


Example Grammar

$$S ::= (S)$$
$$| \{ S \}$$
$$| \epsilon$$

LL(1) Example

(Predictive) Parsing - LL(1) Parsing

Nonterminal on stack:

1 $X = \text{pop}(\text{stack})$

2 $\text{get } P = \text{table}[X, \text{lookahead}]$

3 push P 's RHS symbols *Right-to-Left* "Predicted"

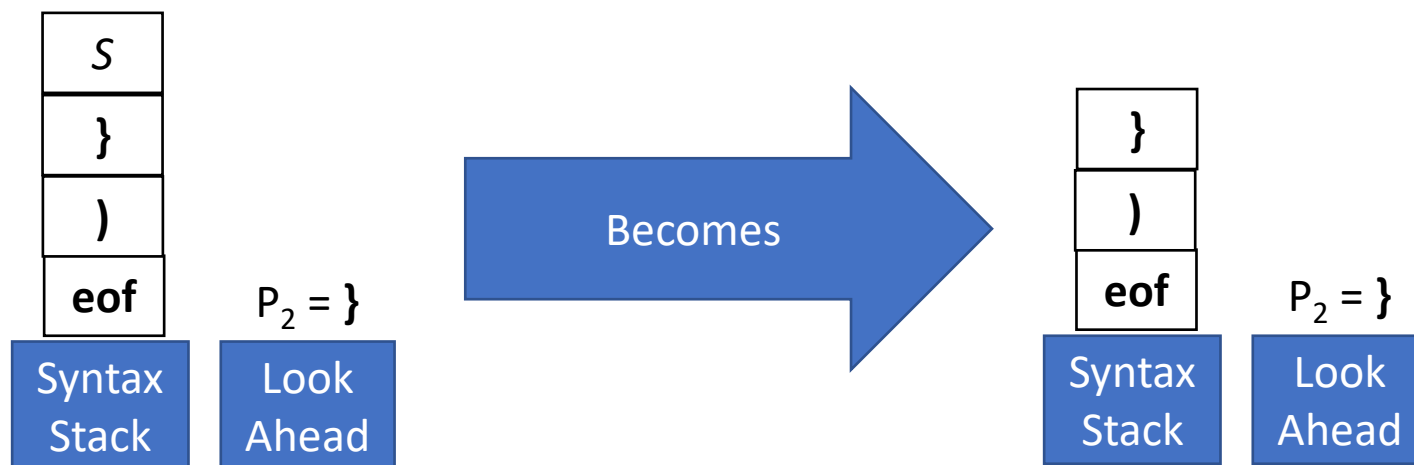
Selector Table

	(	)	{	}
S	(S)	ϵ	{ S }	ϵ

Input Stream

(	{	}	)	eof
P_0	P_1	P_2	P_3	P_4
✓	✓			

Derivation Sequence

$$S \Rightarrow (S)$$
$$\Rightarrow (\{ S \})$$
$$\Rightarrow (\{ \})$$


Example Grammar

$S ::= (S)$
 $\quad | \{ S \}$
 $\quad | \epsilon$

LL(1) Example

(Predictive) Parsing - LL(1) Parsing

Terminal on stack:

- 1 match stack.top with lookahead
- 2 pop stack
- 3 lookahead = scanner.next_token()

Selector Table

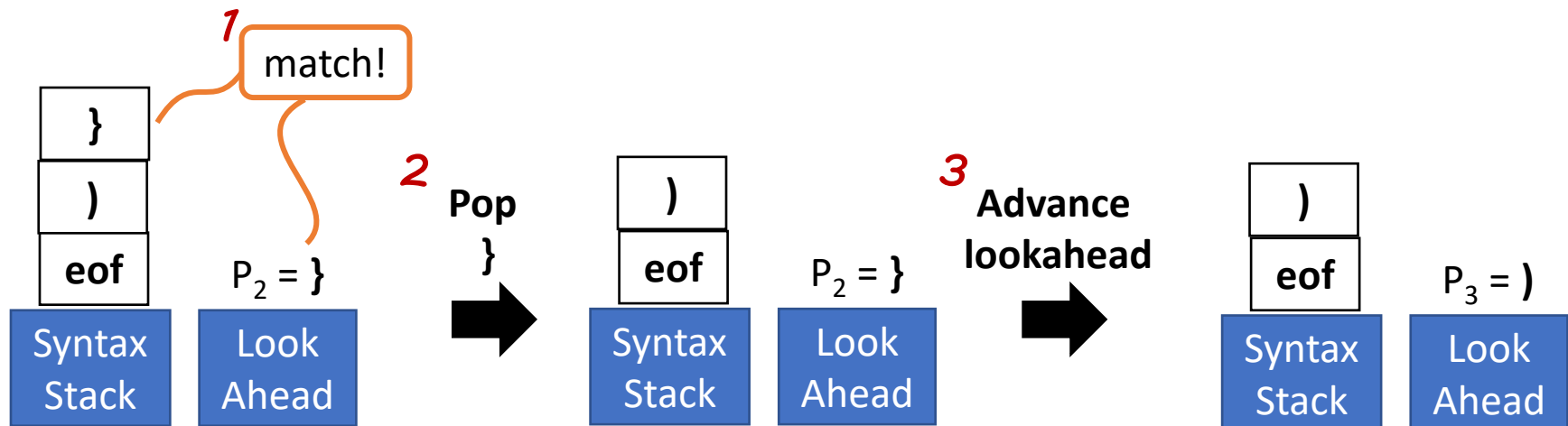
	(	)	{	}
S	(S)	ϵ	{ S }	ϵ

Input Stream

(	{	}	)	eof
P ₀	P ₁	P ₂	P ₃	P ₄
✓	✓	✓		

“Predicted” Derivation Sequence

$S \Rightarrow (S)$
 $\Rightarrow (\{ S \})$
 $\Rightarrow (\{ \})$



Example Grammar

$S ::= (S)$
 $\quad | \{ S \}$
 $\quad | \epsilon$

LL(1) Example

(Predictive) Parsing - LL(1) Parsing

Terminal on stack:

- 1 match `stack.top` with lookahead
- 2 pop stack
- 3 lookahead = `scanner.next_token()`

Selector Table

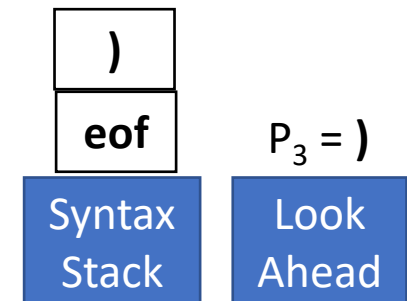
	(	)	{	}
S	(S)	ϵ	{ S }	ϵ

Input Stream

(	{	}	)	eof
P ₀	P ₁	P ₂	P ₃	P ₄
✓	✓	✓		

“Predicted” Derivation Sequence

$S \Rightarrow (S)$
 $\Rightarrow (\{ S \})$
 $\Rightarrow (\{ \})$



Example Grammar

$S ::= (S)$
 $\quad | \{ S \}$
 $\quad | \epsilon$

LL(1) Example

(Predictive) Parsing - LL(1) Parsing

Terminal on stack:

- 1 match `stack.top` with `lookahead`
- 2 pop `stack`
- 3 `lookahead = scanner.next_token()`

Selector Table

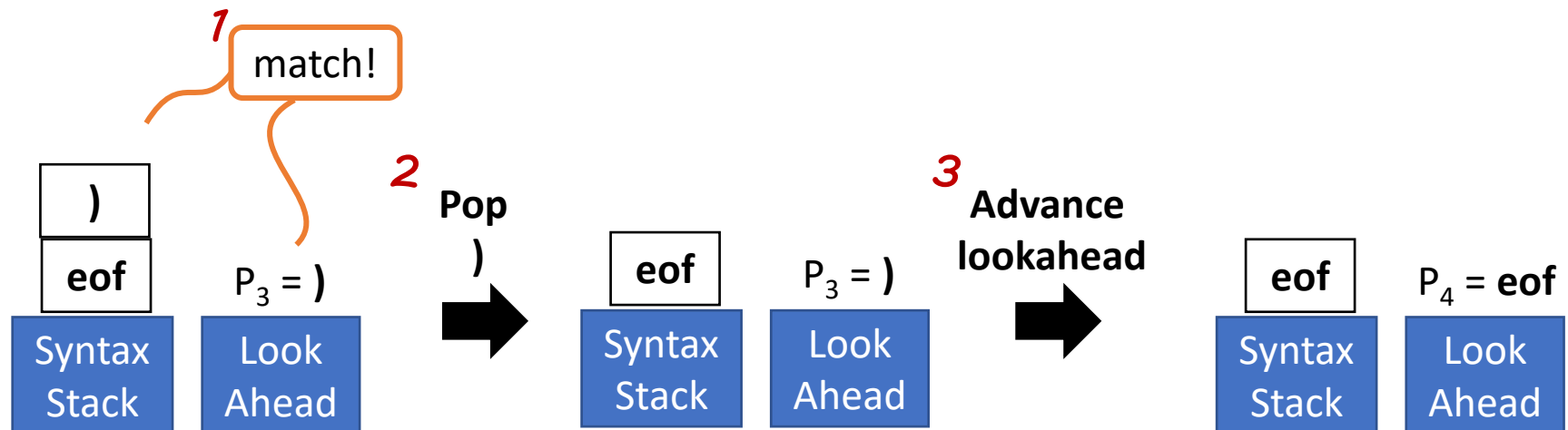
	(	)	{	}
S	(S)	ϵ	{ S }	ϵ

Input Stream

(	{	}	)	eof
P ₀	P ₁	P ₂	P ₃	P ₄
✓	✓	✓		

“Predicted” Derivation Sequence

$S \Rightarrow (S)$
 $\Rightarrow (\{ S \})$
 $\Rightarrow (\{ \})$



Example Grammar

$S ::= (S)$
 $\quad | \{ S \}$
 $\quad | \epsilon$

LL(1) Example

(Predictive) Parsing - LL(1) Parsing

Terminal on stack:

- 1 match `stack.top` with lookahead
- 2 pop stack
- 3 lookahead = `scanner.next_token()`

Selector Table

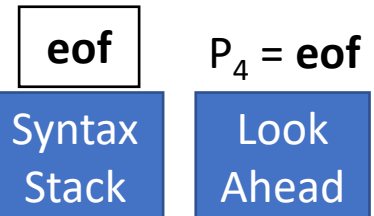
	(	)	{	}
S	(S)	ϵ	{ S }	ϵ

Input Stream

(	{	}	)	eof
P ₀	P ₁	P ₂	P ₃	P ₄
✓	✓	✓	✓	

“Predicted” Derivation Sequence

$S \Rightarrow (S)$
 $\Rightarrow (\{ S \})$
 $\Rightarrow (\{ \})$



Example Grammar

$S ::= (S)$
 $\quad | \{ S \}$
 $\quad | \epsilon$

LL(1) Example

(Predictive) Parsing - LL(1) Parsing

Terminal on stack:

- 1 match `stack.top` with lookahead
- 2 pop stack
- 3 lookahead = `scanner.next_token()`

Selector Table

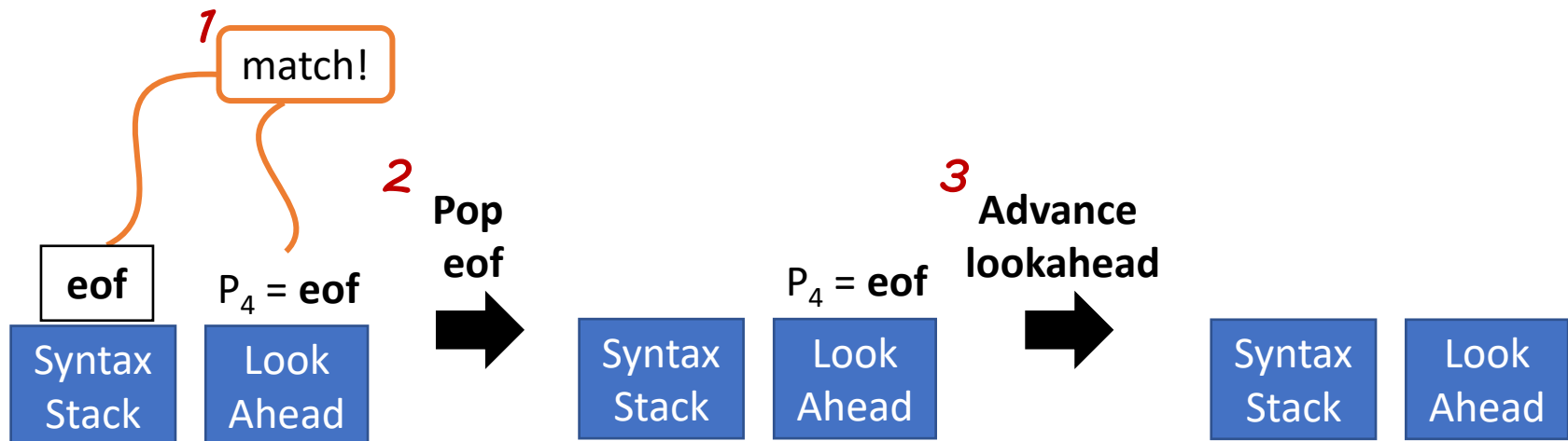
	(	)	{	}
S	(S)	ϵ	{ S }	ϵ

Input Stream

(	{	}	)	eof
P ₀	P ₁	P ₂	P ₃	P ₄
✓	✓	✓	✓	✓

“Predicted” Derivation Sequence

$S \Rightarrow (S)$
 $\Rightarrow (\{ S \})$
 $\Rightarrow (\{ \})$



Example Grammar

$S ::= (S)$
 $\quad | \{ S \}$
 $\quad | \epsilon$

LL(1) Example

(Predictive) Parsing - LL(1) Parsing

Terminal on stack:

- 1 match `stack.top` with lookahead
- 2 pop stack
- 3 lookahead = `scanner.next_token()`

Selector Table

	(	)	{	}
S	(S)	ϵ	{ S }	ϵ

Input Stream

(	{	}	)	eof
P ₀	P ₁	P ₂	P ₃	P ₄
✓	✓	✓	✓	✓

“Predicted” Derivation Sequence

$S \Rightarrow (S)$
 $\Rightarrow (\{ S \})$
 $\Rightarrow (\{ \})$

Syntax
Stack

Look
Ahead

Example Grammar

$$\begin{aligned} S &::= (S) \\ &| \{ S \} \\ &| \varepsilon \end{aligned}$$

LL(1) Example

(Predictive) Parsing - LL(1) Parsing

Exit

Stack and Lookahead both empty:
Accept!

Selector Table

	(	)	{	}
S	(S)	ε	{ S }	ε

Input Stream

(	{	}	)	eof
P ₀	P ₁	P ₂	P ₃	P ₄
$\Rightarrow \{ (\}$	$\Rightarrow \{ (\}$	$\Rightarrow \{ (\}$	$\Rightarrow \{ (\}$	$\Rightarrow \{ (\}$

"Predicted" Derivation Sequence

$$\begin{aligned} S &\Rightarrow (S) \\ &\Rightarrow (\{ S \}) \\ &\Rightarrow (\{ \}) \end{aligned}$$


Syntax
Stack

Look
Ahead

Running LL(1)

(Predictive) Parsing - LL(1) Parsing

Linear parse!

- Once we verify a position we never look at it again

Pretty fast given general parsing complexity

- Especially once you start combining steps

Let's do another quick example

- This time we'll show rejecting bad input

Example Grammar

$$S ::= (S)$$
$$| \{ S \}$$
$$| \epsilon$$

LL(1) Reject Example

(Predictive) Parsing - LL(1) Parsing

Selector Table

	(	)	{	}
S	(S)	ϵ	{ S }	ϵ

Input Stream

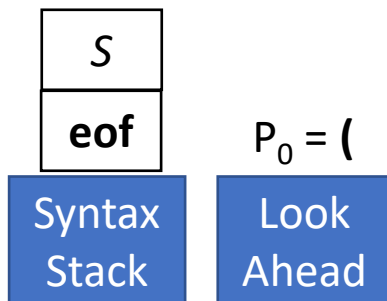
(	}	eof
P ₀	P ₁	P ₂

"Predicted" Derivation Sequence

$$S \Rightarrow (S)$$

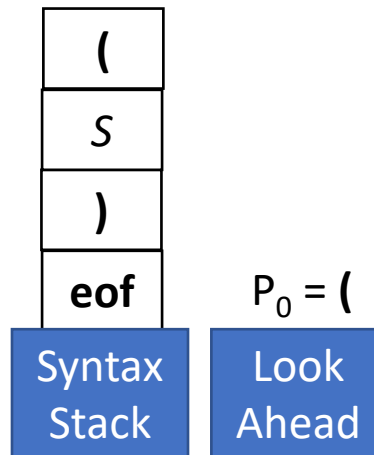
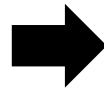
Initialization

1,2,3



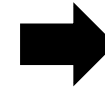
Nonterminal
Actions

1,2,3



Terminal
Actions

1,2,3



Example Grammar

$$S ::= (S)$$
$$| \{ S \}$$
$$| \epsilon$$

LL(1) Reject Example

(Predictive) Parsing - LL(1) Parsing

Selector Table

	(	)	{	}
S	(S)	ϵ	{ S }	ϵ

Input Stream

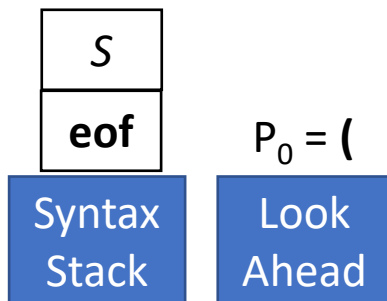
(	}	eof
P ₀	P ₁	P ₂

"Predicted" Derivation Sequence

$$S \Rightarrow (S)$$

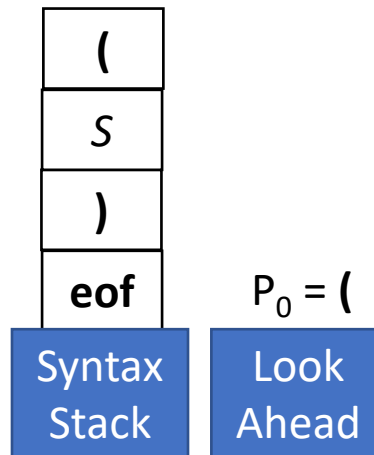
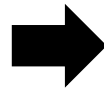
Initialization

1,2,3



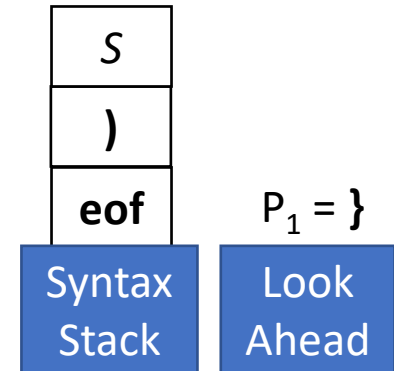
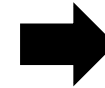
Nonterminal
Actions

1,2,3



Terminal
Actions

1,2,3



Example Grammar

$S ::= (S)$
 $\quad | \{ S \}$
 $\quad | \epsilon$

LL(1) Reject Example

(Predictive) Parsing - LL(1) Parsing

Selector Table

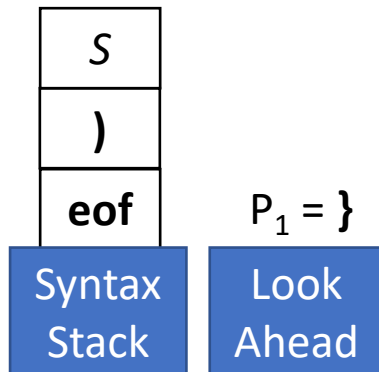
	(	)	{	}
S	(S)	ϵ	{ S }	ϵ

Input Stream

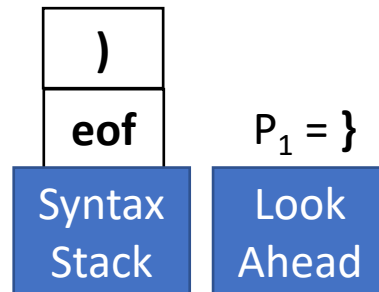
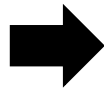
(} eof
 P_0 P_1 P_2

"Predicted" Derivation Sequence

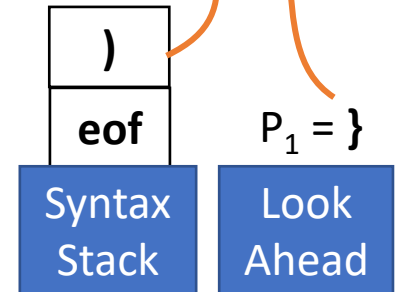
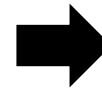
$S \Rightarrow (S)$



Nonterminal
Actions
1,2,3



Terminal
Actions
1 - Fails!



This Grammar is Great!

(Predictive) Parsing - LL(1) Parsing

With 1 token we know exactly what production it starts



...as long as the grammar is LL(1)

Consider a Concerning Grammar...

(Predictive) Parsing - LL(1) Parsing

Our prior
LL(2) Grammar

$S ::= a a$
 $\quad | a b$



Equivalent
LL(1) Grammar

$S ::= a X$
 $X ::= a \mid b$

*The Language
may be LL(1)
Even when the
Grammar is not LL(1)*