

Compiler Construction

Lecture 4: LL parsing

Jeremy Yallop

`jeremy.yallop@cl.cam.ac.uk`

Lent 2026

LL(k)

Recap: recursive descent

LL(k)



Derivations

```
let rec e toks = e' (t toks)
and e' = function
  | ADD :: toks → e' (t toks)
  | toks       → toks (* ∈ *)
```

$$\begin{aligned} E &\rightarrow T E' \\ E' &\rightarrow + T E' \\ E' &\rightarrow \epsilon \\ &\dots \end{aligned}$$

Table

...

Algorithm

Two actions

- matching** (if rhs starts with a terminal, e.g. $E \rightarrow + T E'$)
- predicting** (if rhs starts with a nonterminal, e.g. $E' \rightarrow \mathbf{T} E'$)

Analysis

Q: how do we predict a right-hand side? e.g. given

$$\begin{aligned} A &\rightarrow B \\ A &\rightarrow C \end{aligned}$$

Idea: use the rest of the input (lookahead).

Bottom-up

Plan: precompute all possible rhs for each nonterminal/terminal combination

LL(k)



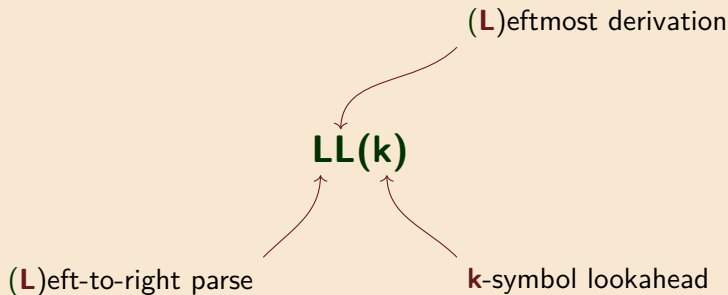
Derivations

Table

Algorithm

Analysis

Bottom-up



Looking at the next k tokens, an LL(k) parser **predicts** the next production. We will consider LL(1).

For LL(1) add an end-of-input marker

LL(k)



Add an end-of-input marker **\$**:

Derivations

$$G_3 = \langle N_3, T_3, P_3, E \rangle$$

$$G'_3 = \langle N'_3, T'_3, P'_3, \mathbf{S} \rangle$$

Table

where

$$N_3 = \{E, E' T, T' F\}$$

$$T_3 = \{+, *, (,), id\}$$

where

$$N'_3 = \{E, E' T, T' F, \mathbf{S}\}$$

$$T'_3 = \{+, *, (,), id, \mathbf{\$}\}$$

Algorithm

$$\mathbf{S} \rightarrow \mathbf{E \$}$$

Analysis

$$P_3 = \begin{array}{lcl} E & \rightarrow & T E' \\ E' & \rightarrow & + T E' \mid \epsilon \\ T & \rightarrow & F T' \\ T' & \rightarrow & * F T' \mid \epsilon \\ F & \rightarrow & (E) \mid id \end{array}$$

$$P'_3 = \begin{array}{lcl} E & \rightarrow & T E' \\ E' & \rightarrow & + T E' \mid \epsilon \\ T & \rightarrow & F T' \\ T' & \rightarrow & * F T' \mid \epsilon \\ F & \rightarrow & (E) \mid id \end{array}$$

Bottom-up

Derivations

A leftmost derivation of $(x+y)$

LL(k)

S

Derivations



Table

Algorithm

Analysis

Bottom-up

S	$\rightarrow$	$E \$$
E	$\rightarrow$	$T E'$
E'	$\rightarrow$	$+ T E'$
E'	$\rightarrow$	ϵ
T	$\rightarrow$	$F T'$
T'	$\rightarrow$	$* F T'$
T'	$\rightarrow$	ϵ
F	$\rightarrow$	(E)
F	$\rightarrow$	id

Idea: Can we turn leftmost derivation s into a stack machine (PDA)?

A leftmost derivation of $(x+y)$

LL(k)

Derivations



Table

Algorithm

Analysis

Bottom-up

$S \Rightarrow_{lm} E \$$

S	$\rightarrow$	$E \$$
E	$\rightarrow$	$T E'$
E'	$\rightarrow$	$+ T E'$
E'	$\rightarrow$	ϵ
T	$\rightarrow$	$F T'$
T'	$\rightarrow$	$* F T'$
T'	$\rightarrow$	ϵ
F	$\rightarrow$	(E)
F	$\rightarrow$	id

Idea: Can we turn leftmost derivation s into a stack machine (PDA)?

A leftmost derivation of $(x+y)$

LL(k)

Derivations



Table

Algorithm

Analysis

Bottom-up

$S \Rightarrow_{lm} E \$$
 $\Rightarrow_{lm} T E' \$$

S	$\rightarrow$	$E \$$
E	$\rightarrow$	$T E'$
E'	$\rightarrow$	$+ T E'$
E'	$\rightarrow$	ϵ
T	$\rightarrow$	$F T'$
T'	$\rightarrow$	$* F T'$
T'	$\rightarrow$	ϵ
F	$\rightarrow$	(E)
F	$\rightarrow$	id

Idea: Can we turn leftmost derivation s into a stack machine (PDA)?

A leftmost derivation of $(x+y)$

LL(k)

Derivations



Table

Algorithm

Analysis

Bottom-up

$S \Rightarrow_{lm} E \$$
 $\Rightarrow_{lm} T E' \$$
 $\Rightarrow_{lm} F T' E' \$$

S	$\rightarrow$	$E \$$
E	$\rightarrow$	$T E'$
E'	$\rightarrow$	$+ T E'$
E'	$\rightarrow$	ϵ
T	$\rightarrow$	$F T'$
T'	$\rightarrow$	$* F T'$
T'	$\rightarrow$	ϵ
F	$\rightarrow$	(E)
F	$\rightarrow$	id

Idea: Can we turn leftmost derivation s into a stack machine (PDA)?

A leftmost derivation of $(x+y)$

LL(k)

Derivations



Table

Algorithm

Analysis

Bottom-up

$S \Rightarrow_{lm} \mathbf{E} \$$
 $\Rightarrow_{lm} \mathbf{T} E' \$$
 $\Rightarrow_{lm} \mathbf{F} T' E' \$$
 $\Rightarrow_{lm} (\mathbf{E}) T' E' \$$

S	$\rightarrow$	$E \$$
E	$\rightarrow$	$T E'$
E'	$\rightarrow$	$+ T E'$
E'	$\rightarrow$	ϵ
T	$\rightarrow$	$F T'$
T'	$\rightarrow$	$* F T'$
T'	$\rightarrow$	ϵ
F	$\rightarrow$	(E)
F	$\rightarrow$	id

Idea: Can we turn leftmost derivation s into a stack machine (PDA)?

A leftmost derivation of $(x+y)$

LL(k)

Derivations



Table

Algorithm

Analysis

Bottom-up

$S \Rightarrow_{lm} E \$$
 $\Rightarrow_{lm} T E' \$$
 $\Rightarrow_{lm} F T' E' \$$
 $\Rightarrow_{lm} (E) T' E' \$$
 $\Rightarrow_{lm} (T E') T' E' \$$

$S \rightarrow E \$$
 $E \rightarrow T E'$
 $E' \rightarrow + T E'$
 $E' \rightarrow \epsilon$
 $T \rightarrow F T'$
 $T' \rightarrow * F T'$
 $T' \rightarrow \epsilon$
 $F \rightarrow (E)$
 $F \rightarrow id$

Idea: Can we turn leftmost derivation s into a stack machine (PDA)?

A leftmost derivation of $(x+y)$

LL(k)

Derivations



Table

Algorithm

Analysis

Bottom-up

$S \Rightarrow_{lm} \mathbf{E} \$$
 $\Rightarrow_{lm} \mathbf{T} E' \$$
 $\Rightarrow_{lm} \mathbf{F} T' E' \$$
 $\Rightarrow_{lm} (\mathbf{E}) T' E' \$$
 $\Rightarrow_{lm} (\mathbf{T} E') T' E' \$$
 $\Rightarrow_{lm} (\mathbf{F} T' E') T' E' \$$

$S \rightarrow E \$$
 $E \rightarrow T E'$
 $E' \rightarrow + T E'$
 $E' \rightarrow \epsilon$
 $T \rightarrow \mathbf{F} T'$
 $T' \rightarrow * F T'$
 $T' \rightarrow \epsilon$
 $F \rightarrow (E)$
 $F \rightarrow id$

Idea: Can we turn leftmost derivation s into a stack machine (PDA)?

A leftmost derivation of $(x+y)$

LL(k)

Derivations



Table

Algorithm

Analysis

Bottom-up

$S \Rightarrow_{lm} E \$$
 $\Rightarrow_{lm} T E' \$$
 $\Rightarrow_{lm} F T' E' \$$
 $\Rightarrow_{lm} (E) T' E' \$$
 $\Rightarrow_{lm} (T E') T' E' \$$
 $\Rightarrow_{lm} (F T' E') T' E' \$$
 $\Rightarrow_{lm} (x T' E') T' E' \$$

$S \rightarrow E \$$
 $E \rightarrow T E'$
 $E' \rightarrow + T E'$
 $E' \rightarrow \epsilon$
 $T \rightarrow F T'$
 $T' \rightarrow * F T'$
 $T' \rightarrow \epsilon$
 $F \rightarrow (E)$
 $F \rightarrow id$

Idea: Can we turn leftmost derivation s into a stack machine (PDA)?

A leftmost derivation of $(x+y)$

LL(k)

Derivations



Table

Algorithm

Analysis

Bottom-up

$$\begin{aligned} S &\Rightarrow_{lm} \mathbf{E} \$ \\ &\Rightarrow_{lm} \mathbf{T} E' \$ \\ &\Rightarrow_{lm} \mathbf{F} T' E' \$ \\ &\Rightarrow_{lm} (\mathbf{E}) T' E' \$ \\ &\Rightarrow_{lm} (\mathbf{T} E') T' E' \$ \\ &\Rightarrow_{lm} (\mathbf{F} T' E') T' E' \$ \\ &\Rightarrow_{lm} (x \mathbf{T}' E') T' E' \$ \\ &\Rightarrow_{lm} (x \mathbf{E}') T' E' \$ \end{aligned}$$

S	$\rightarrow$	$E \$$
E	$\rightarrow$	$T E'$
E'	$\rightarrow$	$+ T E'$
E'	$\rightarrow$	ϵ
T	$\rightarrow$	$F T'$
T'	$\rightarrow$	$* F T'$
T'	$\rightarrow$	ϵ
F	$\rightarrow$	(E)
F	$\rightarrow$	id

Idea: Can we turn leftmost derivation s into a stack machine (PDA)?

A leftmost derivation of $(x+y)$

LL(k)

Derivations



Table

Algorithm

Analysis

Bottom-up

$$\begin{aligned} S &\Rightarrow_{lm} \mathbf{E} \$ \\ &\Rightarrow_{lm} \mathbf{T} E' \$ \\ &\Rightarrow_{lm} \mathbf{F} T' E' \$ \\ &\Rightarrow_{lm} (\mathbf{E}) T' E' \$ \\ &\Rightarrow_{lm} (\mathbf{T} E') T' E' \$ \\ &\Rightarrow_{lm} (\mathbf{F} T' E') T' E' \$ \\ &\Rightarrow_{lm} (x \mathbf{T}' E') T' E' \$ \\ &\Rightarrow_{lm} (x \mathbf{E}') T' E' \$ \end{aligned}$$

$$\Rightarrow_{lm} (x + \mathbf{T} E') T' E' \$$$

S	$\rightarrow$	$E \$$
E	$\rightarrow$	$T E'$
E'	$\rightarrow$	$+ T E'$
E'	$\rightarrow$	ϵ
T	$\rightarrow$	$F T'$
T'	$\rightarrow$	$* F T'$
T'	$\rightarrow$	ϵ
F	$\rightarrow$	(E)
F	$\rightarrow$	id

Idea: Can we turn leftmost derivation s into a stack machine (PDA)?

A leftmost derivation of $(x+y)$

LL(k)

Derivations



Table

Algorithm

Analysis

Bottom-up

$$\begin{aligned} S &\Rightarrow_{lm} E \$ \\ &\Rightarrow_{lm} T E' \$ \\ &\Rightarrow_{lm} F T' E' \$ \\ &\Rightarrow_{lm} (E) T' E' \$ \\ &\Rightarrow_{lm} (T E') T' E' \$ \\ &\Rightarrow_{lm} (F T' E') T' E' \$ \\ &\Rightarrow_{lm} (x T' E') T' E' \$ \\ &\Rightarrow_{lm} (x E') T' E' \$ \end{aligned}$$

$$\begin{aligned} &\Rightarrow_{lm} (x + T E') T' E' \$ \\ &\Rightarrow_{lm} (x + F T' E') T' E' \$ \end{aligned}$$

S	$\rightarrow$	$E \$$
E	$\rightarrow$	$T E'$
E'	$\rightarrow$	$+ T E'$
E'	$\rightarrow$	ϵ
T	$\rightarrow$	$F T'$
T'	$\rightarrow$	$* F T'$
T'	$\rightarrow$	ϵ
F	$\rightarrow$	(E)
F	$\rightarrow$	id

Idea: Can we turn leftmost derivation s into a stack machine (PDA)?

A leftmost derivation of $(x+y)$

LL(k)

Derivations



Table

Algorithm

Analysis

Bottom-up

$$\begin{aligned}
 S &\Rightarrow_{lm} \mathbf{E} \$ \\
 &\Rightarrow_{lm} \mathbf{T} E' \$ \\
 &\Rightarrow_{lm} \mathbf{F} T' E' \$ \\
 &\Rightarrow_{lm} (\mathbf{E}) T' E' \$ \\
 &\Rightarrow_{lm} (\mathbf{T} E') T' E' \$ \\
 &\Rightarrow_{lm} (\mathbf{F} T' E') T' E' \$ \\
 &\Rightarrow_{lm} (x \mathbf{T}' E') T' E' \$ \\
 &\Rightarrow_{lm} (x \mathbf{E}') T' E' \$
 \end{aligned}$$

$$\begin{aligned}
 &\Rightarrow_{lm} (x + \mathbf{T} E') T' E' \$ \\
 &\Rightarrow_{lm} (x + \mathbf{F} T' E') T' E' \$ \\
 &\Rightarrow_{lm} (x + y \mathbf{T}' E') T' E' \$
 \end{aligned}$$

S	$\rightarrow$	$E \$$
E	$\rightarrow$	$T E'$
E'	$\rightarrow$	$+ T E'$
E'	$\rightarrow$	ϵ
T	$\rightarrow$	$F T'$
T'	$\rightarrow$	$* F T'$
T'	$\rightarrow$	ϵ
F	$\rightarrow$	(E)
F	$\rightarrow$	id

Idea: Can we turn leftmost derivation s into a stack machine (PDA)?

A leftmost derivation of $(x+y)$

LL(k)

Derivations



Table

Algorithm

Analysis

Bottom-up

$$\begin{aligned}
 S &\Rightarrow_{lm} \mathbf{E} \$ \\
 &\Rightarrow_{lm} \mathbf{T} E' \$ \\
 &\Rightarrow_{lm} \mathbf{F} T' E' \$ \\
 &\Rightarrow_{lm} (\mathbf{E}) T' E' \$ \\
 &\Rightarrow_{lm} (\mathbf{T} E') T' E' \$ \\
 &\Rightarrow_{lm} (\mathbf{F} T' E') T' E' \$ \\
 &\Rightarrow_{lm} (x \mathbf{T}' E') T' E' \$ \\
 &\Rightarrow_{lm} (x \mathbf{E}') T' E' \$
 \end{aligned}$$

$$\begin{aligned}
 &\Rightarrow_{lm} (x + \mathbf{T} E') T' E' \$ \\
 &\Rightarrow_{lm} (x + \mathbf{F} T' E') T' E' \$ \\
 &\Rightarrow_{lm} (x + y \mathbf{T}' E') T' E' \$ \\
 &\Rightarrow_{lm} (x + y \mathbf{E}') T' E' \$
 \end{aligned}$$

S	$\rightarrow$	$E \$$
E	$\rightarrow$	$T E'$
E'	$\rightarrow$	$+ T E'$
E'	$\rightarrow$	ϵ
T	$\rightarrow$	$F T'$
T'	$\rightarrow$	$* F T'$
T'	$\rightarrow$	ϵ
F	$\rightarrow$	(E)
F	$\rightarrow$	id

Idea: Can we turn leftmost derivation s into a stack machine (PDA)?

A leftmost derivation of $(x+y)$

LL(k)

Derivations



Table

Algorithm

Analysis

Bottom-up

$$\begin{aligned} S &\Rightarrow_{lm} \mathbf{E} \$ \\ &\Rightarrow_{lm} \mathbf{T} E' \$ \\ &\Rightarrow_{lm} \mathbf{F} T' E' \$ \\ &\Rightarrow_{lm} (\mathbf{E}) T' E' \$ \\ &\Rightarrow_{lm} (\mathbf{T} E') T' E' \$ \\ &\Rightarrow_{lm} (\mathbf{F} T' E') T' E' \$ \\ &\Rightarrow_{lm} (x \mathbf{T}' E') T' E' \$ \\ &\Rightarrow_{lm} (x \mathbf{E}') T' E' \$ \end{aligned}$$

$$\begin{aligned} &\Rightarrow_{lm} (x + \mathbf{T} E') T' E' \$ \\ &\Rightarrow_{lm} (x + \mathbf{F} T' E') T' E' \$ \\ &\Rightarrow_{lm} (x + y \mathbf{T}' E') T' E' \$ \\ &\Rightarrow_{lm} (x + y \mathbf{E}') T' E' \$ \\ &\Rightarrow_{lm} (x + y) \mathbf{T}' E' \$ \end{aligned}$$

S	$\rightarrow$	$E \$$
E	$\rightarrow$	$T E'$
E'	$\rightarrow$	$+ T E'$
E'	$\rightarrow$	ϵ
T	$\rightarrow$	$F T'$
T'	$\rightarrow$	$* F T'$
T'	$\rightarrow$	ϵ
F	$\rightarrow$	(E)
F	$\rightarrow$	id

Idea: Can we turn leftmost derivation s into a stack machine (PDA)?

A leftmost derivation of $(x+y)$

LL(k)

Derivations



Table

Algorithm

Analysis

Bottom-up

$S \Rightarrow_{lm} E \$$
 $\Rightarrow_{lm} T E' \$$
 $\Rightarrow_{lm} F T' E' \$$
 $\Rightarrow_{lm} (E) T' E' \$$
 $\Rightarrow_{lm} (T E') T' E' \$$
 $\Rightarrow_{lm} (F T' E') T' E' \$$
 $\Rightarrow_{lm} (x T' E') T' E' \$$
 $\Rightarrow_{lm} (x E') T' E' \$$

$\Rightarrow_{lm} (x + T E') T' E' \$$
 $\Rightarrow_{lm} (x + F T' E') T' E' \$$
 $\Rightarrow_{lm} (x + y T' E') T' E' \$$
 $\Rightarrow_{lm} (x + y E') T' E' \$$
 $\Rightarrow_{lm} (x + y) T' E' \$$
 $\Rightarrow_{lm} (x + y) E' \$$

S	$\rightarrow$	$E \$$
E	$\rightarrow$	$T E'$
E'	$\rightarrow$	$+ T E'$
E'	$\rightarrow$	ϵ
T	$\rightarrow$	$F T'$
T'	$\rightarrow$	$* F T'$
T'	$\rightarrow$	ϵ
F	$\rightarrow$	(E)
F	$\rightarrow$	id

Idea: Can we turn leftmost derivation s into a stack machine (PDA)?

A leftmost derivation of $(x+y)$

LL(k)

Derivations



Table

Algorithm

Analysis

Bottom-up

$$\begin{aligned}
 S &\Rightarrow_{lm} E \$ \\
 &\Rightarrow_{lm} T E' \$ \\
 &\Rightarrow_{lm} F T' E' \$ \\
 &\Rightarrow_{lm} (E) T' E' \$ \\
 &\Rightarrow_{lm} (T E') T' E' \$ \\
 &\Rightarrow_{lm} (F T' E') T' E' \$ \\
 &\Rightarrow_{lm} (x T' E') T' E' \$ \\
 &\Rightarrow_{lm} (x E') T' E' \$ \\
 &\Rightarrow_{lm} (x + T E') T' E' \$ \\
 &\Rightarrow_{lm} (x + F T' E') T' E' \$ \\
 &\Rightarrow_{lm} (x + y T' E') T' E' \$ \\
 &\Rightarrow_{lm} (x + y E') T' E' \$ \\
 &\Rightarrow_{lm} (x + y) T' E' \$ \\
 &\Rightarrow_{lm} (x + y) \$
 \end{aligned}$$

S	$\rightarrow$	$E \$$
E	$\rightarrow$	$T E'$
E'	$\rightarrow$	$+ T E'$
E'	$\rightarrow$	ϵ
T	$\rightarrow$	$F T'$
T'	$\rightarrow$	$* F T'$
T'	$\rightarrow$	ϵ
F	$\rightarrow$	(E)
F	$\rightarrow$	id

Idea: Can we turn leftmost derivation s into a stack machine (PDA)?

From derivation to stack machine

LL(k)

Plan: if $S \Rightarrow_{lm}^+ w\alpha\$$ then w has been read from the input
 α is on the stack

Derivations



Table

Algorithm

Analysis

Bottom-up

input	stack	use production
-------	-------	----------------

How do we automate selection of the production to use at each step?

From derivation to stack machine

LL(k)

Plan: if $S \Rightarrow_{lm}^+ w\alpha\$$ then w has been read from the input
 α is on the stack

Derivations



Table

input	stack	use production
$(x + y)\$$	S	$S \rightarrow E\$$

Algorithm

Analysis

Bottom-up

How do we automate selection of the production to use at each step?

From derivation to stack machine

LL(k)

Plan: if $S \Rightarrow_{lm}^+ w\alpha\$$ then w has been read from the input
 α is on the stack .

Derivations



Table

input	stack	use production
$(x + y)\$$	S	$S \rightarrow E\$$
$(x + y)\$$	$E\$$	$E \rightarrow TE'$

Algorithm

Analysis

Bottom-up

How do we automate selection of the production to use at each step?

From derivation to stack machine

LL(k)

Plan: if $S \Rightarrow_{lm}^+ w\alpha\$$ then w has been read from the input
 α is on the stack .

Derivations



Table

input	stack	use production
$(x + y)\$$	S	$S \rightarrow E\$$
$(x + y)\$$	$E\$$	$E \rightarrow TE'$
$(x + y)\$$	$TE'\$$	$T \rightarrow FT'$

Algorithm

Analysis

Bottom-up

How do we automate selection of the production to use at each step?

From derivation to stack machine

LL(k)

Plan: if $S \Rightarrow_{lm}^+ w\alpha\$$ then w has been read from the input
 α is on the stack .

Derivations



Table

input	stack	use production
$(x + y)\$$	S	$S \rightarrow E\$$
$(x + y)\$$	$E\$$	$E \rightarrow TE'$
$(x + y)\$$	$TE'\$$	$T \rightarrow FT'$
$(x + y)\$$	$FT'E'\$$	$F \rightarrow (E)$

Algorithm

Analysis

Bottom-up

How do we automate selection of the production to use at each step?

From derivation to stack machine

LL(k)

Plan: if $S \Rightarrow_{lm}^+ w\alpha\$$ then w has been read from the input
 α is on the stack .

Derivations



Table

input	stack	use production
$(x + y)\$$	S	$S \rightarrow E\$$
$(x + y)\$$	$E\$$	$E \rightarrow TE'$
$(x + y)\$$	$TE'\$$	$T \rightarrow FT'$
$(x + y)\$$	$FT'E'\$$	$F \rightarrow (E)$
$(x + y)\$$	$(E)T'E'\$$	match

Algorithm

Analysis

Bottom-up

How do we automate selection of the production to use at each step?

From derivation to stack machine

LL(k)

Plan: if $S \Rightarrow_{lm}^+ w\alpha\$$ then w has been read from the input
 α is on the stack .

Derivations



Table

Algorithm

Analysis

Bottom-up

input	stack	use production
$(x + y)\$$	S	$S \rightarrow E\$$
$(x + y)\$$	$E\$$	$E \rightarrow TE'$
$(x + y)\$$	$TE'\$$	$T \rightarrow FT'$
$(x + y)\$$	$FT'E'\$$	$F \rightarrow (E)$
$(x + y)\$$	$(E)T'E'\$$	match
$x + y)\$$	$E)T'E'\$$	$E \rightarrow TE'$

How do we automate selection of the production to use at each step?

From derivation to stack machine

LL(k)

Plan: if $S \Rightarrow_{lm}^+ w\alpha\$$ then w has been read from the input
 α is on the stack .

Derivations



Table

input	stack	use production
$(x + y)\$$	S	$S \rightarrow E\$$
$(x + y)\$$	$E\$$	$E \rightarrow TE'$
$(x + y)\$$	$TE'\$$	$T \rightarrow FT'$
$(x + y)\$$	$FT'E'\$$	$F \rightarrow (E)$
$(x + y)\$$	$(E)T'E'\$$	match
$x + y)\$$	$E)T'E'\$$	$E \rightarrow TE'$
$x + y)\$$	$TE')T'E'\$$	$T \rightarrow FT'$

Algorithm

Analysis

Bottom-up

How do we automate selection of the production to use at each step?

From derivation to stack machine

LL(k)

Plan: if $S \Rightarrow_{lm}^+ w\alpha\$$ then w has been read from the input
 α is on the stack .

Derivations



Table

input	stack	use production
$(x + y)\$$	S	$S \rightarrow E\$$
$(x + y)\$$	$E\$$	$E \rightarrow TE'$
$(x + y)\$$	$TE'\$$	$T \rightarrow FT'$
$(x + y)\$$	$FT'E'\$$	$F \rightarrow (E)$
$(x + y)\$$	$(E)T'E'\$$	match
$x + y)\$$	$E)T'E'\$$	$E \rightarrow TE'$
$x + y)\$$	$TE')T'E'\$$	$T \rightarrow FT'$
$x + y)\$$	$FT'E')T'E'\$$	$F \rightarrow id$

Algorithm

Analysis

Bottom-up

How do we automate selection of the production to use at each step?

From derivation to stack machine

LL(k)

Plan: if $S \Rightarrow_{lm}^+ w\alpha\$$ then w has been read from the input
 α is on the stack .

Derivations



Table

Algorithm

Analysis

Bottom-up

input	stack	use production
$(x + y)\$$	S	$S \rightarrow E\$$
$(x + y)\$$	$E\$$	$E \rightarrow TE'$
$(x + y)\$$	$TE'\$$	$T \rightarrow FT'$
$(x + y)\$$	$FT'E'\$$	$F \rightarrow (E)$
$(x + y)\$$	$(E)T'E'\$$	match
$x + y)\$$	$E)T'E'\$$	$E \rightarrow TE'$
$x + y)\$$	$TE')T'E'\$$	$T \rightarrow FT'$
$x + y)\$$	$FT'E')T'E'\$$	$F \rightarrow id$
$x + y)\$$	$idT'E')T'E'\$$	match

How do we automate selection of the production to use at each step?

From derivation to stack machine

LL(k)

Plan: if $S \Rightarrow_{lm}^+ w\alpha\$$ then w has been read from the input
 α is on the stack

Derivations



Table

Algorithm

Analysis

Bottom-up

input	stack	use production
$(x + y)\$$	S	$S \rightarrow E\$$
$(x + y)\$$	$E\$$	$E \rightarrow TE'$
$(x + y)\$$	$TE'\$$	$T \rightarrow FT'$
$(x + y)\$$	$FT'E'\$$	$F \rightarrow (E)$
$(x + y)\$$	$(E)T'E'\$$	match
$x + y)\$$	$E)T'E'\$$	$E \rightarrow TE'$
$x + y)\$$	$TE')T'E'\$$	$T \rightarrow FT'$
$x + y)\$$	$FT'E')T'E'\$$	$F \rightarrow id$
$x + y)\$$	$idT'E')T'E'\$$	match
$+y)\$$	$T'E')T'E'\$$	$T' \rightarrow \epsilon$

How do we automate selection of the production to use at each step?

From derivation to stack machine

LL(k)

Plan: if $S \Rightarrow_{lm}^+ w\alpha\$$ then w has been read from the input
 α is on the stack

Derivations



Table

Algorithm

Analysis

Bottom-up

input	stack	use production
$(x + y)\$$	S	$S \rightarrow E\$$
$(x + y)\$$	$E\$$	$E \rightarrow TE'$
$(x + y)\$$	$TE'\$$	$T \rightarrow FT'$
$(x + y)\$$	$FT'E'\$$	$F \rightarrow (E)$
$(x + y)\$$	$(E)T'E'\$$	match
$x + y)\$$	$E)T'E'\$$	$E \rightarrow TE'$
$x + y)\$$	$TE')T'E'\$$	$T \rightarrow FT'$
$x + y)\$$	$FT'E')T'E'\$$	$F \rightarrow id$
$x + y)\$$	$idT'E')T'E'\$$	match
$+y)\$$	$T'E')T'E'\$$	$T' \rightarrow \epsilon$
$+y)\$$	$E')T'E'\$$	$E' \rightarrow +TE'$

How do we automate selection of the production to use at each step?

From derivation to stack machine

LL(k)

Plan: if $S \Rightarrow_{lm}^+ w\alpha\$$ then

w has been read from the input
 α is on the stack

Derivations



Table

Algorithm

Analysis

Bottom-up

input	stack	use production	input	stack	use production
$(x + y)\$$	S	$S \rightarrow E\$$	$+y)\$$	$+TE')T'E'\$$	match
$(x + y)\$$	$E\$$	$E \rightarrow TE'$			
$(x + y)\$$	$TE'\$$	$T \rightarrow FT'$			
$(x + y)\$$	$FT'E'\$$	$F \rightarrow (E)$			
$(x + y)\$$	$(E)T'E'\$$	match			
$x + y)\$$	$E)T'E'\$$	$E \rightarrow TE'$			
$x + y)\$$	$TE')T'E'\$$	$T \rightarrow FT'$			
$x + y)\$$	$FT'E')T'E'\$$	$F \rightarrow id$			
$x + y)\$$	$idT'E')T'E'\$$	match			
$+y)\$$	$T'E')T'E'\$$	$T' \rightarrow \epsilon$			
$+y)\$$	$E')T'E'\$$	$E' \rightarrow +TE'$			

How do we automate selection of the production to use at each step?

From derivation to stack machine

LL(k)

Plan: if $S \Rightarrow_{lm}^+ w\alpha\$$ then

w has been read from the input
 α is on the stack

Derivations



Table

Algorithm

Analysis

Bottom-up

input	stack	use production	input	stack	use production
$(x + y)\$$	S	$S \rightarrow E\$$	$+y)\$$	$+TE')T'E'\$$	match
$(x + y)\$$	$E\$$	$E \rightarrow TE'$	$y)\$$	$TE')T'E'\$$	$T \rightarrow FT'$
$(x + y)\$$	$TE'\$$	$T \rightarrow FT'$			
$(x + y)\$$	$FT'E'\$$	$F \rightarrow (E)$			
$(x + y)\$$	$(E)T'E'\$$	match			
$x + y)\$$	$E)T'E'\$$	$E \rightarrow TE'$			
$x + y)\$$	$TE')T'E'\$$	$T \rightarrow FT'$			
$x + y)\$$	$FT'E')T'E'\$$	$F \rightarrow id$			
$x + y)\$$	$idT'E')T'E'\$$	match			
$+y)\$$	$T'E')T'E'\$$	$T' \rightarrow \epsilon$			
$+y)\$$	$E')T'E'\$$	$E' \rightarrow +TE'$			

How do we automate selection of the production to use at each step?

From derivation to stack machine

LL(k)

Plan: if $S \Rightarrow_{lm}^+ w\alpha\$$ then

w has been read from the input
 α is on the stack

Derivations



Table

Algorithm

Analysis

Bottom-up

input	stack	use production	input	stack	use production
$(x + y)\$$	S	$S \rightarrow E\$$	$+y)\$$	$+TE')T'E'\$$	match
$(x + y)\$$	$E\$$	$E \rightarrow TE'$	$y)\$$	$TE')T'E'\$$	$T \rightarrow FT'$
$(x + y)\$$	$TE'\$$	$T \rightarrow FT'$	$y)\$$	$FT'E')T'E'\$$	$F \rightarrow id$
$(x + y)\$$	$FT'E'\$$	$F \rightarrow (E)$			
$(x + y)\$$	$(E)T'E'\$$	match			
$x + y)\$$	$E)T'E'\$$	$E \rightarrow TE'$			
$x + y)\$$	$TE')T'E'\$$	$T \rightarrow FT'$			
$x + y)\$$	$FT'E')T'E'\$$	$F \rightarrow id$			
$x + y)\$$	$idT'E')T'E'\$$	match			
$+y)\$$	$T'E')T'E'\$$	$T' \rightarrow \epsilon$			
$+y)\$$	$E')T'E'\$$	$E' \rightarrow +TE'$			

How do we automate selection of the production to use at each step?

From derivation to stack machine

LL(k)

Plan: if $S \Rightarrow_{lm}^+ w\alpha\$$ then

w has been read from the input
 α is on the stack

Derivations



Table

Algorithm

Analysis

Bottom-up

input	stack	use production	input	stack	use production
$(x + y)\$$	S	$S \rightarrow E\$$	$+y)\$$	$+TE')T'E'\$$	match
$(x + y)\$$	$E\$$	$E \rightarrow TE'$	$y)\$$	$TE')T'E'\$$	$T \rightarrow FT'$
$(x + y)\$$	$TE'\$$	$T \rightarrow FT'$	$y)\$$	$FT'E')T'E'\$$	$F \rightarrow id$
$(x + y)\$$	$FT'E'\$$	$F \rightarrow (E)$	$y)\$$	$idT'E')T'E'\$$	match
$(x + y)\$$	$(E)T'E'\$$	match			
$x + y)\$$	$E)T'E'\$$	$E \rightarrow TE'$			
$x + y)\$$	$TE')T'E'\$$	$T \rightarrow FT'$			
$x + y)\$$	$FT'E')T'E'\$$	$F \rightarrow id$			
$x + y)\$$	$idT'E')T'E'\$$	match			
$+y)\$$	$T'E')T'E'\$$	$T' \rightarrow \epsilon$			
$+y)\$$	$E')T'E'\$$	$E' \rightarrow +TE'$			

How do we automate selection of the production to use at each step?

From derivation to stack machine

LL(k)

Plan: if $S \Rightarrow_{lm}^+ w\alpha\$$ then

w has been read from the input
 α is on the stack

Derivations



Table

Algorithm

Analysis

Bottom-up

input	stack	use production	input	stack	use production
$(x + y)\$$	S	$S \rightarrow E\$$	$+y)\$$	$+TE')T'E'\$$	match
$(x + y)\$$	$E\$$	$E \rightarrow TE'$	$y)\$$	$TE')T'E'\$$	$T \rightarrow FT'$
$(x + y)\$$	$TE'\$$	$T \rightarrow FT'$	$y)\$$	$FT'E')T'E'\$$	$F \rightarrow id$
$(x + y)\$$	$FT'E'\$$	$F \rightarrow (E)$	$y)\$$	$idT'E')T'E'\$$	match
$(x + y)\$$	$(E)T'E'\$$	match	$)\$$	$T'E')T'E'\$$	$T' \rightarrow \epsilon$
$x + y)\$$	$E)T'E'\$$	$E \rightarrow TE'$			
$x + y)\$$	$TE')T'E'\$$	$T \rightarrow FT'$			
$x + y)\$$	$FT'E')T'E'\$$	$F \rightarrow id$			
$x + y)\$$	$idT'E')T'E'\$$	match			
$+y)\$$	$T'E')T'E'\$$	$T' \rightarrow \epsilon$			
$+y)\$$	$E')T'E'\$$	$E' \rightarrow +TE'$			

How do we automate selection of the production to use at each step?

From derivation to stack machine

LL(k)

Plan: if $S \Rightarrow_{lm}^+ w\alpha\$$ then

w has been read from the input
 α is on the stack

Derivations



Table

Algorithm

Analysis

Bottom-up

input	stack	use production	input	stack	use production
$(x + y)\$$	S	$S \rightarrow E\$$	$+y)\$$	$+TE')T'E'\$$	match
$(x + y)\$$	$E\$$	$E \rightarrow TE'$	$y)\$$	$TE')T'E'\$$	$T \rightarrow FT'$
$(x + y)\$$	$TE'\$$	$T \rightarrow FT'$	$y)\$$	$FT'E')T'E'\$$	$F \rightarrow id$
$(x + y)\$$	$FT'E'\$$	$F \rightarrow (E)$	$y)\$$	$idT'E')T'E'\$$	match
$(x + y)\$$	$(E)T'E'\$$	match	$)\$$	$T'E')T'E'\$$	$T' \rightarrow \epsilon$
$x + y)\$$	$E)T'E'\$$	$E \rightarrow TE'$	$)\$$	$E')T'E'\$$	$E' \rightarrow \epsilon$
$x + y)\$$	$TE')T'E'\$$	$T \rightarrow FT'$			
$x + y)\$$	$FT'E')T'E'\$$	$F \rightarrow id$			
$x + y)\$$	$idT'E')T'E'\$$	match			
$+y)\$$	$T'E')T'E'\$$	$T' \rightarrow \epsilon$			
$+y)\$$	$E')T'E'\$$	$E' \rightarrow +TE'$			

How do we automate selection of the production to use at each step?

From derivation to stack machine

LL(k)

Plan: if $S \Rightarrow_{lm}^+ w\alpha\$$ then

w has been read from the input
 α is on the stack

Derivations



Table

Algorithm

Analysis

Bottom-up

input	stack	use production	input	stack	use production
$(x + y)\$$	S	$S \rightarrow E\$$	$+y)\$$	$+TE')T'E'\$$	match
$(x + y)\$$	$E\$$	$E \rightarrow TE'$	$y)\$$	$TE')T'E'\$$	$T \rightarrow FT'$
$(x + y)\$$	$TE'\$$	$T \rightarrow FT'$	$y)\$$	$FT'E')T'E'\$$	$F \rightarrow id$
$(x + y)\$$	$FT'E'\$$	$F \rightarrow (E)$	$y)\$$	$idT'E')T'E'\$$	match
$(x + y)\$$	$(E)T'E'\$$	match	$)\$$	$T'E')T'E'\$$	$T' \rightarrow \epsilon$
$x + y)\$$	$E)T'E'\$$	$E \rightarrow TE'$	$)\$$	$E')T'E'\$$	$E' \rightarrow \epsilon$
$x + y)\$$	$TE')T'E'\$$	$T \rightarrow FT'$	$)\$$	$)T'E'\$$	match
$x + y)\$$	$FT'E')T'E'\$$	$F \rightarrow id$			
$x + y)\$$	$idT'E')T'E'\$$	match			
$+y)\$$	$T'E')T'E'\$$	$T' \rightarrow \epsilon$			
$+y)\$$	$E')T'E'\$$	$E' \rightarrow +TE'$			

How do we automate selection of the production to use at each step?

From derivation to stack machine

LL(k)

Plan: if $S \Rightarrow_{lm}^+ w\alpha\$$ then

w has been read from the input
 α is on the stack

Derivations



Table

Algorithm

Analysis

Bottom-up

input	stack	use production	input	stack	use production
$(x + y)\$$	S	$S \rightarrow E\$$	$+y)\$$	$+TE')T'E'\$$	match
$(x + y)\$$	$E\$$	$E \rightarrow TE'$	$y)\$$	$TE')T'E'\$$	$T \rightarrow FT'$
$(x + y)\$$	$TE'\$$	$T \rightarrow FT'$	$y)\$$	$FT'E')T'E'\$$	$F \rightarrow id$
$(x + y)\$$	$FT'E'\$$	$F \rightarrow (E)$	$y)\$$	$idT'E')T'E'\$$	match
$(x + y)\$$	$(E)T'E'\$$	match	$)\$$	$T'E')T'E'\$$	$T' \rightarrow \epsilon$
$x + y)\$$	$E)T'E'\$$	$E \rightarrow TE'$	$)\$$	$E')T'E'\$$	$E' \rightarrow \epsilon$
$x + y)\$$	$TE')T'E'\$$	$T \rightarrow FT'$	$)\$$	$)T'E'\$$	match
$x + y)\$$	$FT'E')T'E'\$$	$F \rightarrow id$	$\$$	$T'E'\$$	$T' \rightarrow \epsilon$
$x + y)\$$	$idT'E')T'E'\$$	match			
$+y)\$$	$T'E')T'E'\$$	$T' \rightarrow \epsilon$			
$+y)\$$	$E')T'E'\$$	$E' \rightarrow +TE'$			

How do we automate selection of the production to use at each step?

From derivation to stack machine

LL(k)

Plan: if $S \Rightarrow_{lm}^+ w\alpha\$$ then

w has been read from the input
 α is on the stack

Derivations



Table

Algorithm

Analysis

Bottom-up

input	stack	use production	input	stack	use production
$(x + y)\$$	S	$S \rightarrow E\$$	$+y)\$$	$+TE')T'E'\$$	match
$(x + y)\$$	$E\$$	$E \rightarrow TE'$	$y)\$$	$TE')T'E'\$$	$T \rightarrow FT'$
$(x + y)\$$	$TE'\$$	$T \rightarrow FT'$	$y)\$$	$FT'E')T'E'\$$	$F \rightarrow id$
$(x + y)\$$	$FT'E'\$$	$F \rightarrow (E)$	$y)\$$	$idT'E')T'E'\$$	match
$(x + y)\$$	$(E)T'E'\$$	match	$)\$$	$T'E')T'E'\$$	$T' \rightarrow \epsilon$
$x + y)\$$	$E)T'E'\$$	$E \rightarrow TE'$	$)\$$	$E')T'E'\$$	$E' \rightarrow \epsilon$
$x + y)\$$	$TE')T'E'\$$	$T \rightarrow FT'$	$)\$$	$)T'E'\$$	match
$x + y)\$$	$FT'E')T'E'\$$	$F \rightarrow id$	$\$$	$T'E'\$$	$T' \rightarrow \epsilon$
$x + y)\$$	$idT'E')T'E'\$$	match	$\$$	$E'\$$	$E' \rightarrow \epsilon$
$+y)\$$	$T'E')T'E'\$$	$T' \rightarrow \epsilon$			
$+y)\$$	$E')T'E'\$$	$E' \rightarrow +TE'$			

How do we automate selection of the production to use at each step?

From derivation to stack machine

LL(k)

Plan: if $S \Rightarrow_{lm}^+ w\alpha\$$ then

w has been read from the input
 α is on the stack

Derivations



Table

Algorithm

Analysis

Bottom-up

input	stack	use production	input	stack	use production
$(x + y)\$$	S	$S \rightarrow E\$$	$+y)\$$	$+TE')T'E'\$$	match
$(x + y)\$$	$E\$$	$E \rightarrow TE'$	$y)\$$	$TE')T'E'\$$	$T \rightarrow FT'$
$(x + y)\$$	$TE'\$$	$T \rightarrow FT'$	$y)\$$	$FT'E')T'E'\$$	$F \rightarrow id$
$(x + y)\$$	$FT'E'\$$	$F \rightarrow (E)$	$y)\$$	$idT'E')T'E'\$$	match
$(x + y)\$$	$(E)T'E'\$$	match	$)\$$	$T'E')T'E'\$$	$T' \rightarrow \epsilon$
$x + y)\$$	$E)T'E'\$$	$E \rightarrow TE'$	$)\$$	$E')T'E'\$$	$E' \rightarrow \epsilon$
$x + y)\$$	$TE')T'E'\$$	$T \rightarrow FT'$	$)\$$	$)T'E'\$$	match
$x + y)\$$	$FT'E')T'E'\$$	$F \rightarrow id$	$\$$	$T'E'\$$	$T' \rightarrow \epsilon$
$x + y)\$$	$idT'E')T'E'\$$	match	$\$$	$E'\$$	$E' \rightarrow \epsilon$
$+y)\$$	$T'E')T'E'\$$	$T' \rightarrow \epsilon$	$\$$	$\$$	accept!
$+y)\$$	$E')T'E'\$$	$E' \rightarrow +TE'$			

How do we automate selection of the production to use at each step?

The LL(1) parsing table

LL(k)

Derivations

Table

● ○ ○ ○

Algorithm

Analysis

Bottom-up

The **FIRST set** for a sequence of symbols α represents the terminals that may occur at the start of derivations of α (and ϵ , if $\alpha \Rightarrow^* \epsilon$)

$$\text{FIRST}(\alpha) = \{a \in T \mid \exists \beta \in (N \cup T)^*, \alpha \Rightarrow^* a\beta\} \cup \{\epsilon \mid \alpha \Rightarrow^* \epsilon\}$$

We can compute FIRST for each rhs and nonterminal (details later):

S	$\rightarrow$	$E \$$	$\text{FIRST}(S)$	$=$	$\{ (, id \}$
E	$\rightarrow$	$T E'$	$\text{FIRST}(E)$	$=$	$\{ (, id \}$
E'	$\rightarrow$	$+ T E' \mid \epsilon$	$\text{FIRST}(E')$	$=$	$\{ +, \epsilon \}$
T	$\rightarrow$	$F T'$	$\text{FIRST}(T)$	$=$	$\{ (, id \}$
T'	$\rightarrow$	$* F T' \mid \epsilon$	$\text{FIRST}(T')$	$=$	$\{ *, \epsilon \}$
F	$\rightarrow$	$(E) \mid id$	$\text{FIRST}(F)$	$=$	$\{ (, id \}$

LL(k)

The **FOLLOW set** for a nonterminal A represents the terminals that may follow A in a derivation from the start symbol

$$\text{FOLLOW}(A) = \{a \mid \exists \alpha\beta, S \Rightarrow^+ \alpha A a \beta\}$$

We can compute FOLLOW for each nonterminal in a grammar (details later):

$S \rightarrow E \$$	
$E \rightarrow T E'$	$\text{FOLLOW}(E) = \{), \$\}$
$E' \rightarrow + T E' \mid \epsilon$	$\text{FOLLOW}(E') = \{), \$\}$
$T \rightarrow F T'$	$\text{FOLLOW}(T) = \{+,), \$\}$
$T' \rightarrow * F T' \mid \epsilon$	$\text{FOLLOW}(T') = \{+,), \$\}$
$F \rightarrow (E) \mid id$	$\text{FOLLOW}(F) = \{+, *,), \$\}$

Q: is $) \in \text{FOLLOW}(E)$? Yes: $S \Rightarrow E\$ \Rightarrow TE'\$ \Rightarrow FT'E'\$ \Rightarrow (E)T'E'\$$

Derivations

Table



Algorithm

Analysis

Bottom-up

The LL(1) Parsing table M

LL(k)

The parsing table maps $\langle \text{nonterminal}, \text{terminal} \rangle$ pairs to right-hand sides.

Derivations

Initialize M :

for each $A \in N$, $a \in T$, $M[A, a] = \{\}$

Populate M :

for each $A \in N$

for each production $A \rightarrow \alpha$

if $a \in \text{FIRST}(\alpha)$ and $a \neq \epsilon$

then $M[A, a] = M[A, a] \cup \{\alpha\}$

else if $\epsilon \in \text{FIRST}(\alpha)$

then for each $b \in \text{FOLLOW}(A)$

$M[A, b] = M[A, b] \cup \{\alpha\}$

	id	$+$	$\dots$
E	TE'		$\dots$
E'		$+TE'$	$\dots$
$\dots$	$\dots$	$\dots$	$\dots$

Table



Algorithm

Analysis

Bottom-up

Table M for grammar G'_3

LL(k)

FOLLOW sets for G'_3 :

S	E	E'	T	T'	F
	) \$	) \$	+) \$	+) \$	+ *) \$

Derivations

FIRST sets for G'_3 :

$S \rightarrow E\$$	$E \rightarrow TE'$	$E' \rightarrow +TE'$	$E' \rightarrow \epsilon$	$T \rightarrow FT'$	$T' \rightarrow *FT'$	$F \rightarrow (E)$	$F \rightarrow id$
(id	(id	+	ϵ	(id	*	(	id

Table



Algorithm

Table M for G'_3 :

	id	+	*	(	)	\$
S	$E\$$			$E\$$		
E	TE'			TE'		
E'		$+TE'$			ϵ	ϵ
T	FT'			FT'		
T'		ϵ	$*FT'$		ϵ	ϵ
F	id			(E)		

Analysis

Bottom-up

The algorithm

The LL(1) Parsing Algorithm

LL(k)

Derivations

Table

Algorithm



Analysis

Bottom-up

```
a := NextToken()
X := TopOfStack()
while (X ≠ $)
    if X = a (* match *)
        then Pop(); a := NextToken()
    else if M[X, a] = {α} (* predict *)
        then Pop(); Push(α)
X := TopOfStack()
```

Using M to parse (x+y)

LL(k)

Derivations

Table

Algorithm



Analysis

Bottom-up

input

stack

action

Using M to parse (x+y)

LL(k)

Derivations

input	stack	action
$(x + y)\$$	S	$M[S, (] = \{E\}$

Table

Algorithm



Analysis

Bottom-up

Using M to parse (x+y)

LL(k)

Derivations

input	stack	action
$(x + y)\$$	S	$M[S, (] = \{E\$ \}$
$(x + y)\$$	$E\$$	$M[E, (] = \{TE' \}$

Table

Algorithm



Analysis

Bottom-up

Using M to parse (x+y)

LL(k)

Derivations

Table

Algorithm



Analysis

Bottom-up

input	stack	action
$(x + y)\$$	S	$M[S, (] = \{E\$ \}$
$(x + y)\$$	$E\$$	$M[E, (] = \{TE' \}$
$(x + y)\$$	$TE'\$$	$M[T, (] = \{FT' \}$

Using M to parse (x+y)

LL(k)

Derivations

Table

input	stack	action
$(x + y)\$$	S	$M[S, (] = \{E\}$
$(x + y)\$$	$E\$$	$M[E, (] = \{TE'\}$
$(x + y)\$$	$TE'\$$	$M[T, (] = \{FT'\}$
$(x + y)\$$	$FT'E'\$$	$M[F, (] = \{(E)\}$

Algorithm



Analysis

Bottom-up

Using M to parse (x+y)

LL(k)

Derivations

Table

Algorithm



Analysis

Bottom-up

input	stack	action
$(x + y)\$$	S	$M[S, (] = \{E\}$
$(x + y)\$$	$E\$$	$M[E, (] = \{TE'\}$
$(x + y)\$$	$TE'\$$	$M[T, (] = \{FT'\}$
$(x + y)\$$	$FT'E'\$$	$M[F, (] = \{(E)\}$
$(x + y)\$$	$(E)T'E'\$$	match

Using M to parse (x+y)

LL(k)

Derivations

Table

Algorithm



Analysis

Bottom-up

input	stack	action
$(x + y)\$$	S	$M[S, (] = \{E\}$
$(x + y)\$$	$E\$$	$M[E, (] = \{TE'\}$
$(x + y)\$$	$TE'\$$	$M[T, (] = \{FT'\}$
$(x + y)\$$	$FT'E'\$$	$M[F, (] = \{(E)\}$
$(x + y)\$$	$(E)T'E'\$$	match
$x + y)\$$	$E)T'E'\$$	$M[E, id] = \{TE'\}$

Using M to parse (x+y)

LL(k)

Derivations

Table

Algorithm



Analysis

Bottom-up

input	stack	action
$(x + y)\$$	S	$M[S, (] = \{E\}$
$(x + y)\$$	$E\$$	$M[E, (] = \{TE'\}$
$(x + y)\$$	$TE'\$$	$M[T, (] = \{FT'\}$
$(x + y)\$$	$FT'E'\$$	$M[F, (] = \{(E)\}$
$(x + y)\$$	$(E)T'E'\$$	match
$x + y)\$$	$E)T'E'\$$	$M[E, id] = \{TE'\}$
$x + y)\$$	$TE')T'E'\$$	$M[T, id] = \{FT'\}$

Using M to parse (x+y)

LL(k)

Derivations

Table

Algorithm



Analysis

Bottom-up

input	stack	action
$(x + y)\$$	S	$M[S, (] = \{E\}$
$(x + y)\$$	$E\$$	$M[E, (] = \{TE'\}$
$(x + y)\$$	$TE'\$$	$M[T, (] = \{FT'\}$
$(x + y)\$$	$FT'E'\$$	$M[F, (] = \{(E)\}$
$(x + y)\$$	$(E)T'E'\$$	match
$x + y)\$$	$E)T'E'\$$	$M[E, id] = \{TE'\}$
$x + y)\$$	$TE')T'E'\$$	$M[T, id] = \{FT'\}$
$x + y)\$$	$FT'E')T'E'\$$	$M[F, id] = \{id\}$

Using M to parse (x+y)

LL(k)

Derivations

Table

Algorithm



Analysis

Bottom-up

input	stack	action
$(x + y)\$$	S	$M[S, (] = \{E\}$
$(x + y)\$$	$E\$$	$M[E, (] = \{TE'\}$
$(x + y)\$$	$TE'\$$	$M[T, (] = \{FT'\}$
$(x + y)\$$	$FT'E'\$$	$M[F, (] = \{(E)\}$
$(x + y)\$$	$(E)T'E'\$$	match
$x + y)\$$	$E)T'E'\$$	$M[E, id] = \{TE'\}$
$x + y)\$$	$TE')T'E'\$$	$M[T, id] = \{FT'\}$
$x + y)\$$	$FT'E')T'E'\$$	$M[F, id] = \{id\}$
$x + y)\$$	$idT'E')T'E'\$$	match

Using M to parse (x+y)

LL(k)

Derivations

Table

Algorithm



Analysis

Bottom-up

input	stack	action
(x + y)\$	S	$M[S, (] = \{E\}$
(x + y)\$	E\$	$M[E, (] = \{TE'\}$
(x + y)\$	TE'\$	$M[T, (] = \{FT'\}$
(x + y)\$	FT'E'\$	$M[F, (] = \{(E)\}$
(x + y)\$	(E)T'E'\$	match
x + y)\$	E)T'E'\$	$M[E, id] = \{TE'\}$
x + y)\$	TE')T'E'\$	$M[T, id] = \{FT'\}$
x + y)\$	FT'E')T'E'\$	$M[F, id] = \{id\}$
x + y)\$	idT'E')T'E'\$	match
+y)\$	T'E')T'E'\$	$M[T', +] = \{\epsilon\}$

LL(k)

Derivations

Table

Algorithm



Analysis

Bottom-up

input	stack	action
(x + y)\$	S	$M[S, (] = \{E\}$
(x + y)\$	E\$	$M[E, (] = \{TE'\}$
(x + y)\$	TE'\$	$M[T, (] = \{FT'\}$
(x + y)\$	FT'E'\$	$M[F, (] = \{(E)\}$
(x + y)\$	(E)T'E'\$	match
x + y)\$	E)T'E'\$	$M[E, id] = \{TE'\}$
x + y)\$	TE')T'E'\$	$M[T, id] = \{FT'\}$
x + y)\$	FT'E')T'E'\$	$M[F, id] = \{id\}$
x + y)\$	idT'E')T'E'\$	match
+y)\$	T'E')T'E'\$	$M[T', +] = \{\epsilon\}$
+y)\$	E')T'E'\$	$M[E', +] = \{+TE'\}$

Using M to parse (x+y)

LL(k)

Derivations

Table

Algorithm



Analysis

Bottom-up

input	stack	action
(x + y)\$	S	$M[S, (] = \{E\}$
(x + y)\$	E\$	$M[E, (] = \{TE'\}$
(x + y)\$	TE'\$	$M[T, (] = \{FT'\}$
(x + y)\$	FT'E'\$	$M[F, (] = \{(E)\}$
(x + y)\$	(E)T'E'\$	match
x + y)\$	E)T'E'\$	$M[E, id] = \{TE'\}$
x + y)\$	TE')T'E'\$	$M[T, id] = \{FT'\}$
x + y)\$	FT'E')T'E'\$	$M[F, id] = \{id\}$
x + y)\$	idT'E')T'E'\$	match
+y)\$	T'E')T'E'\$	$M[T', +] = \{\epsilon\}$
+y)\$	E')T'E'\$	$M[E', +] = \{+TE'\}$

input	stack	action
+y)\$	+TE')T'E'\$	match

Using M to parse (x+y)

LL(k)

Derivations

Table

Algorithm



Analysis

Bottom-up

input	stack	action
(x + y)\$	S	$M[S, () = \{E\}]$
(x + y)\$	E\$	$M[E, () = \{TE'\}]$
(x + y)\$	TE'\$	$M[T, () = \{FT'\}]$
(x + y)\$	FT'E'\$	$M[F, () = \{(E)\}]$
(x + y)\$	(E)T'E'\$	match
x + y)\$	E)T'E'\$	$M[E, id] = \{TE'\}$
x + y)\$	TE')T'E'\$	$M[T, id] = \{FT'\}$
x + y)\$	FT'E')T'E'\$	$M[F, id] = \{id\}$
x + y)\$	idT'E')T'E'\$	match
+y)\$	T'E')T'E'\$	$M[T', +] = \{\epsilon\}$
+y)\$	E')T'E'\$	$M[E', +] = \{+TE'\}$

input	stack	action
+y)\$	+TE')T'E'\$	match
y)\$	TE')T'E'\$	$M[T, id] = \{FT'\}$

Using M to parse (x+y)

LL(k)

Derivations

Table

Algorithm



Analysis

Bottom-up

input	stack	action
(x + y)\$	S	$M[S, (] = \{E\}$
(x + y)\$	E\$	$M[E, (] = \{TE'\}$
(x + y)\$	TE'\$	$M[T, (] = \{FT'\}$
(x + y)\$	FT'E'\$	$M[F, (] = \{(E)\}$
(x + y)\$	(E)T'E'\$	match
x + y)\$	E)T'E'\$	$M[E, id] = \{TE'\}$
x + y)\$	TE')T'E'\$	$M[T, id] = \{FT'\}$
x + y)\$	FT'E')T'E'\$	$M[F, id] = \{id\}$
x + y)\$	idT'E')T'E'\$	match
+y)\$	T'E')T'E'\$	$M[T', +] = \{\epsilon\}$
+y)\$	E')T'E'\$	$M[E', +] = \{+TE'\}$

input	stack	action
+y)\$	+TE')T'E'\$	match
y)\$	TE')T'E'\$	$M[T, id] = \{FT'\}$
y)\$	FT'E')T'E'\$	$M[F, id] = \{id\}$

Using M to parse (x+y)

LL(k)

Derivations

Table

Algorithm



Analysis

Bottom-up

input	stack	action
(x + y)\$	S	$M[S, (] = \{E\}$
(x + y)\$	E\$	$M[E, (] = \{TE'\}$
(x + y)\$	TE'\$	$M[T, (] = \{FT'\}$
(x + y)\$	FT'E'\$	$M[F, (] = \{(E)\}$
(x + y)\$	(E)T'E'\$	match
x + y)\$	E)T'E'\$	$M[E, id] = \{TE'\}$
x + y)\$	TE')T'E'\$	$M[T, id] = \{FT'\}$
x + y)\$	FT'E')T'E'\$	$M[F, id] = \{id\}$
x + y)\$	idT'E')T'E'\$	match
+y)\$	T'E')T'E'\$	$M[T', +] = \{\epsilon\}$
+y)\$	E')T'E'\$	$M[E', +] = \{+TE'\}$

input	stack	action
+y)\$	+TE')T'E'\$	match
y)\$	TE')T'E'\$	$M[T, id] = \{FT'\}$
y)\$	FT'E')T'E'\$	$M[F, id] = \{id\}$
y)\$	idT'E')T'E'\$	match

Using M to parse (x+y)

LL(k)

Derivations

Table

Algorithm



Analysis

Bottom-up

input	stack	action
(x + y)\$	S	$M[S, (] = \{E\}$
(x + y)\$	E\$	$M[E, (] = \{TE'\}$
(x + y)\$	TE'\$	$M[T, (] = \{FT'\}$
(x + y)\$	FT'E'\$	$M[F, (] = \{(E)\}$
(x + y)\$	(E)T'E'\$	match
x + y)\$	E)T'E'\$	$M[E, id] = \{TE'\}$
x + y)\$	TE')T'E'\$	$M[T, id] = \{FT'\}$
x + y)\$	FT'E')T'E'\$	$M[F, id] = \{id\}$
x + y)\$	idT'E')T'E'\$	match
+y)\$	T'E')T'E'\$	$M[T', +] = \{\epsilon\}$
+y)\$	E')T'E'\$	$M[E', +] = \{+TE'\}$

input	stack	action
+y)\$	+TE')T'E'\$	match
y)\$	TE')T'E'\$	$M[T, id] = \{FT'\}$
y)\$	FT'E')T'E'\$	$M[F, id] = \{id\}$
y)\$	idT'E')T'E'\$	match
)\$	T'E')T'E'\$	$M[T',)] = \{\epsilon\}$

Using M to parse (x+y)

LL(k)

Derivations

Table

Algorithm



Analysis

Bottom-up

input	stack	action
(x + y)\$	S	$M[S, (] = \{E\}$
(x + y)\$	E\$	$M[E, (] = \{TE'\}$
(x + y)\$	TE'\$	$M[T, (] = \{FT'\}$
(x + y)\$	FT'E'\$	$M[F, (] = \{(E)\}$
(x + y)\$	(E)T'E'\$	match
x + y)\$	E)T'E'\$	$M[E, id] = \{TE'\}$
x + y)\$	TE')T'E'\$	$M[T, id] = \{FT'\}$
x + y)\$	FT'E')T'E'\$	$M[F, id] = \{id\}$
x + y)\$	idT'E')T'E'\$	match
+y)\$	T'E')T'E'\$	$M[T', +] = \{\epsilon\}$
+y)\$	E')T'E'\$	$M[E', +] = \{+TE'\}$

input	stack	action
+y)\$	+TE')T'E'\$	match
y)\$	TE')T'E'\$	$M[T, id] = \{FT'\}$
y)\$	FT'E')T'E'\$	$M[F, id] = \{id\}$
y)\$	idT'E')T'E'\$	match
)\$	T'E')T'E'\$	$M[T',)] = \{\epsilon\}$
)\$	E')T'E'\$	$M[E',)] = \{\epsilon\}$

Using M to parse (x+y)

LL(k)

Derivations

Table

Algorithm



Analysis

Bottom-up

input	stack	action
(x + y)\$	S	$M[S, (] = \{E\}$
(x + y)\$	E\$	$M[E, (] = \{TE'\}$
(x + y)\$	TE'\$	$M[T, (] = \{FT'\}$
(x + y)\$	FT'E'\$	$M[F, (] = \{(E)\}$
(x + y)\$	(E)T'E'\$	match
x + y)\$	E)T'E'\$	$M[E, id] = \{TE'\}$
x + y)\$	TE')T'E'\$	$M[T, id] = \{FT'\}$
x + y)\$	FT'E')T'E'\$	$M[F, id] = \{id\}$
x + y)\$	idT'E')T'E'\$	match
+y)\$	T'E')T'E'\$	$M[T', +] = \{\epsilon\}$
+y)\$	E')T'E'\$	$M[E', +] = \{+TE'\}$

input	stack	action
+y)\$	+TE')T'E'\$	match
y)\$	TE')T'E'\$	$M[T, id] = \{FT'\}$
y)\$	FT'E')T'E'\$	$M[F, id] = \{id\}$
y)\$	idT'E')T'E'\$	match
)\$	T'E')T'E'\$	$M[T',)] = \{\epsilon\}$
)\$	E')T'E'\$	$M[E',)] = \{\epsilon\}$
)\$	)T'E'\$	match

Using M to parse (x+y)

LL(k)

Derivations

Table

Algorithm



Analysis

Bottom-up

input	stack	action
(x + y)\$	S	$M[S, (] = \{E\}$
(x + y)\$	E\$	$M[E, (] = \{TE'\}$
(x + y)\$	TE'\$	$M[T, (] = \{FT'\}$
(x + y)\$	FT'E'\$	$M[F, (] = \{(E)\}$
(x + y)\$	(E)T'E'\$	match
x + y)\$	E)T'E'\$	$M[E, id] = \{TE'\}$
x + y)\$	TE')T'E'\$	$M[T, id] = \{FT'\}$
x + y)\$	FT'E')T'E'\$	$M[F, id] = \{id\}$
x + y)\$	idT'E')T'E'\$	match
+y)\$	T'E')T'E'\$	$M[T', +] = \{\epsilon\}$
+y)\$	E')T'E'\$	$M[E', +] = \{+TE'\}$

input	stack	action
+y)\$	+TE')T'E'\$	match
y)\$	TE')T'E'\$	$M[T, id] = \{FT'\}$
y)\$	FT'E')T'E'\$	$M[F, id] = \{id\}$
y)\$	idT'E')T'E'\$	match
)\$	T'E')T'E'\$	$M[T',)] = \{\epsilon\}$
)\$	E')T'E'\$	$M[E',)] = \{\epsilon\}$
)\$	)T'E'\$	match
\$	T'E'\$	$M[T', \$] = \{\epsilon\}$

Using M to parse (x+y)

LL(k)

Derivations

Table

Algorithm



Analysis

Bottom-up

input	stack	action
(x + y)\$	S	$M[S, (] = \{E\}$
(x + y)\$	E\$	$M[E, (] = \{TE'\}$
(x + y)\$	TE'\$	$M[T, (] = \{FT'\}$
(x + y)\$	FT'E'\$	$M[F, (] = \{(E)\}$
(x + y)\$	(E)T'E'\$	match
x + y)\$	E)T'E'\$	$M[E, id] = \{TE'\}$
x + y)\$	TE')T'E'\$	$M[T, id] = \{FT'\}$
x + y)\$	FT'E')T'E'\$	$M[F, id] = \{id\}$
x + y)\$	idT'E')T'E'\$	match
+y)\$	T'E')T'E'\$	$M[T', +] = \{\epsilon\}$
+y)\$	E')T'E'\$	$M[E', +] = \{+TE'\}$

input	stack	action
+y)\$	+TE')T'E'\$	match
y)\$	TE')T'E'\$	$M[T, id] = \{FT'\}$
y)\$	FT'E')T'E'\$	$M[F, id] = \{id\}$
y)\$	idT'E')T'E'\$	match
)\$	T'E')T'E'\$	$M[T',)] = \{\epsilon\}$
)\$	E')T'E'\$	$M[E',)] = \{\epsilon\}$
)\$	)T'E'\$	match
\$	T'E'\$	$M[T', \$] = \{\epsilon\}$
\$	E'\$	$M[E', \$] = \{\epsilon\}$

Using M to parse (x+y)

LL(k)

Derivations

Table

Algorithm



Analysis

Bottom-up

input	stack	action
(x + y)\$	S	$M[S, (] = \{E\}$
(x + y)\$	E\$	$M[E, (] = \{TE'\}$
(x + y)\$	TE'\$	$M[T, (] = \{FT'\}$
(x + y)\$	FT'E'\$	$M[F, (] = \{(E)\}$
(x + y)\$	(E)T'E'\$	match
x + y)\$	E)T'E'\$	$M[E, id] = \{TE'\}$
x + y)\$	TE')T'E'\$	$M[T, id] = \{FT'\}$
x + y)\$	FT'E')T'E'\$	$M[F, id] = \{id\}$
x + y)\$	idT'E')T'E'\$	match
+y)\$	T'E')T'E'\$	$M[T', +] = \{\epsilon\}$
+y)\$	E')T'E'\$	$M[E', +] = \{+TE'\}$

input	stack	action
+y)\$	+TE')T'E'\$	match
y)\$	TE')T'E'\$	$M[T, id] = \{FT'\}$
y)\$	FT'E')T'E'\$	$M[F, id] = \{id\}$
y)\$	idT'E')T'E'\$	match
)\$	T'E')T'E'\$	$M[T',)] = \{\epsilon\}$
)\$	E')T'E'\$	$M[E',)] = \{\epsilon\}$
)\$	)T'E'\$	match
\$	T'E'\$	$M[T', \$] = \{\epsilon\}$
\$	E'\$	$M[E', \$] = \{\epsilon\}$
\$	\$	accept!

Analysis

LL(k)

Semantically:

$$\text{NULLABLE}(\alpha) = \text{true} \quad \text{iff} \quad \alpha \Rightarrow^* \epsilon$$

Table

Inductively:

$$\text{NULLABLE}(\epsilon) = \text{true}$$

$$\text{NULLABLE}(c) = \text{false} \quad (c \in T)$$

$$\text{NULLABLE}(A) = \bigvee_{A \rightarrow \alpha} \text{NULLABLE}(\alpha) \quad (A \in N)$$

$$\text{NULLABLE}(X\beta) = \text{NULLABLE}(X) \wedge \text{NULLABLE}(\beta) \quad (X \in T \cup N)$$

Analysis



Bottom-up

Computing NULLABLE: example

LL(k)

Derivations

$$\begin{aligned}\text{NULLABLE}(\epsilon) &= \text{true} \\ \text{NULLABLE}(c) &= \text{false} && (c \in T) \\ \text{NULLABLE}(A) &= \bigvee_{A \rightarrow \alpha} \text{NULLABLE}(\alpha) && (A \in N) \\ \text{NULLABLE}(X\beta) &= \text{NULLABLE}(X) \wedge \text{NULLABLE}(\beta) && (X \in T \cup N)\end{aligned}$$

Table

Algorithm

$E \rightarrow aF$
 $E \rightarrow \epsilon$
 $F \rightarrow E$

Analysis



Bottom-up

$$\text{NULLABLE}(a) = \text{false}$$

$$\text{NULLABLE}(\epsilon) = \text{true}$$

$$\begin{aligned}\text{NULLABLE}(aF) &= \text{NULLABLE}(a) \wedge \text{NULLABLE}(F) \\ &= \text{false} \wedge \text{NULLABLE}(F) \\ &= \text{false}\end{aligned}$$

$$\begin{aligned}\text{NULLABLE}(E) &= \text{NULLABLE}(aF) \vee \text{NULLABLE}(\epsilon) \\ &= \text{false} \vee \text{true} \\ &= \text{true}\end{aligned}$$

$$\begin{aligned}\text{NULLABLE}(F) &= \text{NULLABLE}(E) \\ &= \text{true}\end{aligned}$$

LL(k)

Derivations

Table

Algorithm

Analysis



Bottom-up

Initialize FIRST sets:

for each $a \in T$, $\text{FIRST}(a) := \{a\}$

for each $A \in N$, $\text{FIRST}(A) := \{\}$

Populate FIRST sets:

while FIRST changes

if $A \rightarrow X_1 X_2 \dots X_k$ is a production then

if $\text{NULLABLE}(X_1 X_2 \dots X_k)$

then $\text{FIRST}(A) := \text{FIRST}(A) \cup \{\epsilon\}$

for each j in $1 \dots k$

$\text{FIRST}(A) := \text{FIRST}(A) \cup (\text{FIRST}(X_j) - \{\epsilon\})$

if not $\text{NULLABLE}(X_j)$ then break

LL(k)

Derivations

Table

Algorithm

Analysis



Bottom-up

Initialize FOLLOW sets:

for each $A \in N$, $\text{FOLLOW}(A) := \{\}$

$\text{FOLLOW}(S) := \{\$ \}$ (S is the original start symbol)

Populate FOLLOW sets:

while FOLLOW changes

if $A \rightarrow \alpha B \beta$ is a production ($B \in N$, $\beta \neq \epsilon$)

then $\text{FOLLOW}(B) := \text{FOLLOW}(B) \cup (\text{FIRST}(\beta) - \{\epsilon\})$

if $A \rightarrow \alpha B \beta$ is a production and $\epsilon \in \text{FIRST}(\beta)$

then $\text{FOLLOW}(B) := \text{FOLLOW}(B) \cup \text{FOLLOW}(A)$

if $A \rightarrow \alpha B$ is a production ($B \in N$)

then $\text{FOLLOW}(B) := \text{FOLLOW}(B) \cup \text{FOLLOW}(A)$

Bottom-up parsing

Many grammars cannot be parsed using LL(1)

LL(k)

Derivations

$$\begin{aligned} S &\rightarrow d \mid XYS \\ Y &\rightarrow c \mid \epsilon \\ X &\rightarrow Y \mid a \end{aligned}$$

FIRST: $\begin{array}{|c|c|} \hline XYS & Y \\ \hline acd\epsilon & c\epsilon \\ \hline \end{array}$

FOLLOW: $\begin{array}{|c|c|} \hline X & Y \\ \hline acd & acd \\ \hline \end{array}$

Table

	a	c	d
S	XYS	XYS	XYS d
X	Y a	Y	Y
Y	ϵ	ϵ c	ϵ

Table M:

Algorithm

Analysis

There are multiple entries for $M[S, d]$. The grammar is **ambiguous**, and **not LL(1)**.

Bottom-up



Bottom-up (LR) parsing is more powerful

LL(k)

Recall: we had to rewrite G_2 to eliminate **left recursion**.

$$G_2 = \langle N_2, T_1, P_2, E \rangle$$

where

$$\begin{aligned} P_2 = \quad & E \rightarrow E + T \mid T && \text{(expressions)} \\ & T \rightarrow T * F \mid F && \text{(terms)} \\ & F \rightarrow (E) \mid id && \text{(factors)} \end{aligned}$$

Bottom-up parsing can process a wider class of grammars.

With bottom-up parsing there is no need to eliminate left recursion.

Bottom-up



Next time: bottom-up parsing foundations