Compiler Construction

Lecture 4: LL parsing

Jeremy Yallop

jeremy.yallop@cl.cam.ac.uk

Lent 2023

Recap: recursive descent

```
...
and e' = function
   | ADD :: toks → e' (t toks)
   | toks        → toks (* ϵ *)
...
```

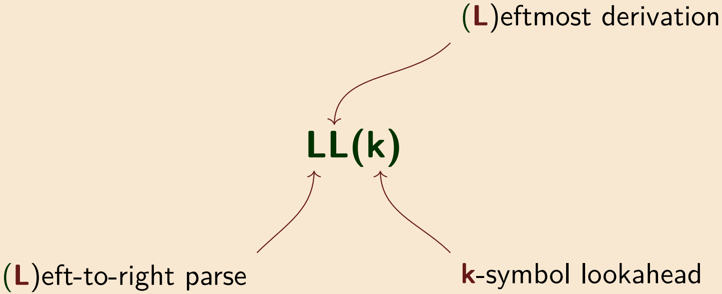
$$\begin{array}{lcl} & & \dots \\ E' & \rightarrow & + \, T \, E' \\ E' & \rightarrow & \epsilon \\ & & \dots \end{array}$$

Two actions **matching** (if rhs starts with terminal)
predicting (if rhs has a nonterminal in front) .

Q: how do we predict a right-hand side? e.g. given $\begin{array}{lcl} A & \rightarrow & B \\ A & \rightarrow & C \end{array}$

Idea: use the rest of the input (look-ahead).

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



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

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

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

where

$$N_3 = \{E, E' \, T, T' \, F\}$$
$$T_3 = \{+, *, (,), id\}$$

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

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

where

$$N'_3 = \{E, E' \, T, T' \, F, \mathbf{S}\}$$
$$T'_3 = \{+, *, (,), id, \$\}$$

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

Derivations

A leftmost derivation of (x+y)

S

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

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

A leftmost derivation of (x+y)

$S \Rightarrow_{lm}$ **E** $\$$

$$
\begin{aligned}
S &\to E\,\$ \\
E &\to T\,E' \\
E' &\to +T\,E' \\
E' &\to \epsilon \\
T &\to F\,T' \\
T' &\to *F\,T' \\
T' &\to \epsilon \\
F &\to (E) \\
F &\to id
\end{aligned}
$$

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

A leftmost derivation of (x+y)

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

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)

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

$$
\begin{aligned}
S &\rightarrow E\ \$ \\
E &\rightarrow T\ E' \\
E' &\rightarrow +T\ E' \\
E' &\rightarrow \epsilon \\
T &\rightarrow F\ T' \\
T' &\rightarrow *F\ T' \\
T' &\rightarrow \epsilon \\
F &\rightarrow (E) \\
F &\rightarrow id
\end{aligned}
$$

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

A leftmost derivation of (x+y)

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

$$\begin{aligned} S &\rightarrow E\ \$ \\ E &\rightarrow T\ E' \\ E' &\rightarrow +T\ E' \\ E' &\rightarrow \epsilon \\ T &\rightarrow F\ T' \\ T' &\rightarrow *F\ T' \\ T' &\rightarrow \epsilon \\ F &\rightarrow (E) \\ F &\rightarrow id \end{aligned}$$

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

A leftmost derivation of (x+y)

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

$$
\begin{aligned}
S &\rightarrow E\ \$ \\
E &\rightarrow T\ E' \\
E' &\rightarrow +T\ E' \\
E' &\rightarrow \epsilon \\
T &\rightarrow F\ T' \\
T' &\rightarrow *F\ T' \\
T' &\rightarrow \epsilon \\
F &\rightarrow (E) \\
F &\rightarrow id
\end{aligned}
$$

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

A leftmost derivation of (x+y)

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

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)

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

$$
\begin{aligned}
S &\rightarrow E\ \$ \\
E &\rightarrow T\ E' \\
E' &\rightarrow +T\ E' \\
E' &\rightarrow \epsilon \\
T &\rightarrow F\ T' \\
T' &\rightarrow *F\ T' \\
T' &\rightarrow \epsilon \\
F &\rightarrow (E) \\
F &\rightarrow id
\end{aligned}
$$

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

A leftmost derivation of (x+y)

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

$$
\begin{aligned}
S &\rightarrow E\ \$ \\
E &\rightarrow T\ E' \\
E' &\rightarrow +T\ E' \\
E' &\rightarrow \epsilon \\
T &\rightarrow F\ T' \\
T' &\rightarrow *F\ T' \\
T' &\rightarrow \epsilon \\
F &\rightarrow (E) \\
F &\rightarrow id
\end{aligned}
$$

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

A leftmost derivation of (x+y)

$S \Rightarrow_{lm} \mathbf{E}\ \$$

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

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

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

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

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

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

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

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

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)

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

$$
\begin{aligned}
S &\rightarrow E\,\$ \\
E &\rightarrow T\ E' \\
E' &\rightarrow +T\ E' \\
E' &\rightarrow \epsilon \\
T &\rightarrow F\ T' \\
T' &\rightarrow *F\ T' \\
T' &\rightarrow \epsilon \\
F &\rightarrow (E) \\
F &\rightarrow id
\end{aligned}
$$

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

A leftmost derivation of (x+y)

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

$$
\begin{aligned}
S &\rightarrow E\,\$ \\
E &\rightarrow T\ E' \\
E' &\rightarrow +T\ E' \\
E' &\rightarrow \epsilon \\
T &\rightarrow F\ T' \\
T' &\rightarrow *F\ T' \\
T' &\rightarrow \epsilon \\
F &\rightarrow (E) \\
F &\rightarrow id
\end{aligned}
$$

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

A leftmost derivation of (x+y)

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

$$
\begin{aligned}
S &\rightarrow E\ \$ \\
E &\rightarrow T\ E' \\
E' &\rightarrow +T\ E' \\
E' &\rightarrow \epsilon \\
T &\rightarrow F\ T' \\
T' &\rightarrow *F\ T' \\
T' &\rightarrow \epsilon \\
F &\rightarrow (E) \\
F &\rightarrow id
\end{aligned}
$$

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

A leftmost derivation of (x+y)

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

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)

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

$$
\begin{aligned}
S &\rightarrow E\,\$ \\
E &\rightarrow T\,E' \\
E' &\rightarrow +T\,E' \\
E' &\rightarrow \epsilon \\
T &\rightarrow F\,T' \\
T' &\rightarrow *F\,T' \\
T' &\rightarrow \epsilon \\
F &\rightarrow (E) \\
F &\rightarrow id
\end{aligned}
$$

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

A leftmost derivation of (x+y)

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

$$
\begin{aligned}
S &\rightarrow E\ \$ \\
E &\rightarrow T\ E' \\
E' &\rightarrow +T\ E' \\
E' &\rightarrow \epsilon \\
T &\rightarrow F\ T' \\
T' &\rightarrow *F\ T' \\
T' &\rightarrow \epsilon \\
F &\rightarrow (E) \\
F &\rightarrow id
\end{aligned}
$$

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

From derivation to stack machine

Plan: if $S \Rightarrow^{+}_{lm} w\alpha\$$ then $\begin{array}{l} w \text{ has been read from the input} \\ \alpha \text{ is on on the stack} \end{array}$.

input	stack	via production

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

From derivation to stack machine

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

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

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

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

input	stack	via production
$(x+y)\$$	S	$S \to E\$$
$(x+y)\$$	$E\$$	$E \to TE'$

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

From derivation to stack machine

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

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

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

From derivation to stack machine

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

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

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

LL(k)
Derivations
Table
Algorithm
Analysis
Bottom-up

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

input	stack	via 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

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

From derivation to stack machine

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

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

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

From derivation to stack machine

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

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

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

From derivation to stack machine

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

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

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

From derivation to stack machine

Plan: if $S \Rightarrow^{+}_{lm} w\alpha\$$ then $\begin{array}{l} w \text{ has been read from the input} \\ \alpha \text{ is on on the stack} \end{array}$.

input	stack	via 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

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

input	stack	via 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

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

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

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

From derivation to stack machine

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

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

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

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

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

input	stack	via 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
$+y)\$$	$T'E')T'E'\$$	$T' \rightarrow \epsilon$
$+y)\$$	$E')T'E'\$$	$E' \rightarrow +TE'$

input	stack	via production
$+y)\$$	$+TE')T'E'\$$	match
$y)\$$	$TE')T'E'\$$	$T \rightarrow FT'$

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

From derivation to stack machine

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

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

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

From derivation to stack machine

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

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

input	stack	via production
$+y)\$$	$+TE')T'E'\$$	match
$y)\$$	$TE')T'E'\$$	$T \to FT'$
$y)\$$	$FT'E')T'E'\$$	$F \to id$
$y)\$$	$idT'E')T'E'\$$	match

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

LL(k)
Derivations
Table
Algorithm
Analysis
Bottom-up

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

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

input	stack	via production
$+y)\$$	$+TE')T'E'\$$	match
$y)\$$	$TE')T'E'\$$	$T \to FT'$
$y)\$$	$FT'E')T'E'\$$	$F \to id$
$y)\$$	$idT'E')T'E'\$$	match
$)\$$	$T'E')T'E'\$$	$T' \to \epsilon$

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

From derivation to stack machine

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

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

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

From derivation to stack machine

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

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

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

From derivation to stack machine

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

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

input	stack	via production
$+y)\$$	$+TE')T'E'\$$	match
$y)\$$	$TE')T'E'\$$	$T \to FT'$
$y)\$$	$FT'E')T'E'\$$	$F \to id$
$y)\$$	$idT'E')T'E'\$$	match
$)\$$	$T'E')T'E'\$$	$T' \to \epsilon$
$)\$$	$E')T'E'\$$	$E' \to \epsilon$
$)\$$	$)T'E'\$$	match
$\$$	$T'E'\$$	$T' \to \epsilon$

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

From derivation to stack machine

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

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

input	stack	via production
$+y)\$$	$+TE')T'E'\$$	match
$y)\$$	$TE')T'E'\$$	$T \to FT'$
$y)\$$	$FT'E')T'E'\$$	$F \to id$
$y)\$$	$idT'E')T'E'\$$	match
$)\$$	$T'E')T'E'\$$	$T' \to \epsilon$
$)\$$	$E')T'E'\$$	$E' \to \epsilon$
$)\$$	$)T'E'\$$	match
$\$$	$T'E'\$$	$T' \to \epsilon$
$\$$	$E'\$$	$E' \to \epsilon$

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

From derivation to stack machine

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

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

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

Building the table

FIRST

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 nonterminal in a grammar (details later):

$$\begin{array}{lcl} S & \rightarrow & E\,\$ \\ E & \rightarrow & T\,E' \\ E' & \rightarrow & +\,T\,E' \mid \epsilon \\ T & \rightarrow & F\,T' \\ T' & \rightarrow & *\,F\,T' \mid \epsilon \\ F & \rightarrow & (E) \mid id \end{array} \qquad \begin{array}{lcl} \text{FIRST}(S) & = & \{\,(\,,\ id\,\} \\ \text{FIRST}(E) & = & \{\,(\,,\ id\,\} \\ \text{FIRST}(E') & = & \{\,+\,,\ \epsilon\,\} \\ \text{FIRST}(T) & = & \{\,(\,,\ id\,\} \\ \text{FIRST}(T') & = & \{\,*\,,\ \epsilon\,\} \\ \text{FIRST}(F) & = & \{\,(\,,\ id\,\} \end{array}$$

FOLLOW

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

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

The LL(1) Parsing table M

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

Initialize M:

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

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 $b \in \text{FOLLOW}(A)$

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

Table M for grammar G_3'

FOLLOW sets for G_3':

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

FIRST sets for G_3':

$E\$$	TE'	$+TE'$	ϵ	FT'	$*FT'$	(E)	id
$(id$	$(id$	$+$	ϵ	$(id$	$*$	$($	id

Table M for G_3':

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

The algorithm

The LL(1) Parsing Algorithm

$a :=$ NextToken()
$X :=$ TopOfStack()
while ($X \neq \$$)
    if $X = a$ (* match *)
        then pop; $a :=$ NextToken()
    else if $M[X, a] = \{\alpha\}$ (* predict *)
        then pop; push α
    $X :=$ TopOfStack()

Using M to parse (x+y)

input	stack	action

Using M to parse (x+y)

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

Using M to parse (x+y)

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

Using M to parse (x+y)

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)

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

Using M to parse (x+y)

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)

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)

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)

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)

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)

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
$+y)\$$	$T'E')T'E'\$$	$M[T',+] = \{\epsilon\}$

Using M to parse (x+y)

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)

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)

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)

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)

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)

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)

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)

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)

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)

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)

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

Computing NULLABLE

Semantically:

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

Inductively:

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

Computing NULLABLE: example

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

$$
\begin{aligned}
E &\to aF \\
E &\to \epsilon \\
F &\to E
\end{aligned}
$$

$$
\begin{aligned}
NULLABLE(a) &= \text{false} \\
\text{NULLABLE}(eps) &= \text{true} \\
\text{NULLABLE}(aE) &= \text{NULLABLE}(a) \wedge \text{NULLABLE}(F) \\
&= \text{false} \wedge \text{NULLABLE}(F) \\
&= \text{false} \\
\text{NULLABLE}(E) &= \text{NULLABLE}(aF) \vee \text{NULLABLE}(eps) \\
&= \text{false} \vee \text{true} \\
&= \text{true} \\
\text{NULLABLE}(F) &= \text{NULLABLE}(E) \\
&= \text{true}
\end{aligned}
$$

Computing FIRST

Initialize FIRST sets:

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

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

Populate FIRST sets:

while FIRST changes

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

if NULLABLE($X_1 X_2 \ldots X_k$)

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

for each j in 1 ...k

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

if not NULLABLE(X_j) then break

Computing FOLLOW

Initialize FOLLOW sets:

For all $A \in N$, FOLLOW(A) := $\{\}$

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

Populate FOLLOW sets:

while FOLLOW changes

if $A \rightarrow \alpha B \beta$ is a production $(B \in N, \beta \neq \epsilon)$
then FOLLOW(B) := FOLLOW(B) $\cup$ (FIRST(β) - $\{\epsilon\}$)

if $A \rightarrow \alpha B \beta$ is a production and $\epsilon \in$ FIRST(β)
then FOLLOW(B) := FOLLOW(B) $\cup$ FOLLOW(A)

if $A \rightarrow \alpha B$ is a production $(B \in N)$
then FOLLOW(B) := FOLLOW(B) $\cup$ FOLLOW(A)

Bottom-up parsing

Many grammars cannot be parsed using LL(1)

$$
\begin{array}{lcl}
S & \rightarrow & d \mid X\,Y\,S \\
Y & \rightarrow & c \mid \epsilon \\
X & \rightarrow & Y \mid a
\end{array}
$$

FIRST:

$X\,Y\,S$	Y
$a\;c\;d\;\epsilon$	$c\;\epsilon$

FOLLOW:

X	Y
$a\;c\;d$	$a\;c\;d$

Table M:

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

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

Bottom-up (LR) parsing is more powerful

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

where

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

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

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

Next time: bottom-up parsing foundations