CSCI 742 - Compiler Construction

Lecture 14

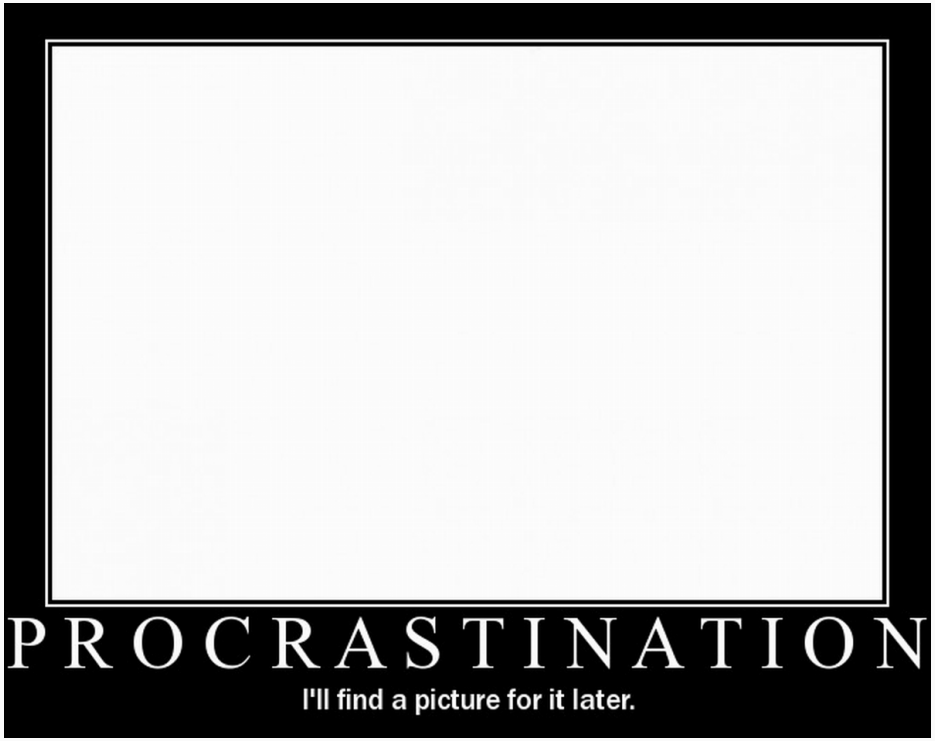
Introduction to Shift/Reduce Parsing

Instructor: Hossein Hojjat

February 16, 2018

LL vs LR

- LL(k) parser predicts which production rule to use by looking at the next k tokens
- LR(k) parser postpone the decision until it has seen input tokens of the entire right-hand side of the production rule
  - more powerful than LL(k) parsers



LR(k) Parsing

- **L**eft-to-right scan of input, **R**ightmost derivation
- Can handle left recursion and common prefixes
- As efficient as any top-down parsing
- Complex to implement
- Generally need automatic tools to construct parser from grammar

Example Grammar for LR Parsing

- Left-factoring is unnecessary for bottom-up parsers
- We can use a more "natural" grammar for additive expressions:

$$
\begin{aligned}
S &\to E \\
E &\to T \mid E + T \\
T &\to \texttt{num} \mid (E)
\end{aligned}
$$

- Consider the input word: $\texttt{num} + (\texttt{num} + \texttt{num})$

Bottom-up Parsing for Example Grammar

- Bottom-up parsing reduces a word to the start symbol by inverting productions
- Productions rules from bottom to top create a rightmost derivation

$\mathrm{num} + (\mathrm{num} + \mathrm{num})$	$T \to \mathrm{num}$
$T + (\mathrm{num} + \mathrm{num})$	$E \to T$
$E + (\mathrm{num} + \mathrm{num})$	$T \to \mathrm{num}$
$E + (T + \mathrm{num})$	$E \to T$
$E + (E + \mathrm{num})$	$T \to \mathrm{num}$
$E + (E + T)$	$E \to E + T$
$E + (E)$	$T \to (E)$
$E + T$	$E \to E + T$
E	$S \to E$
S	

Useful Notation

- **Idea:** Split string into two substrings
- Right substring is still not examined by parer (a string of terminals)
- Left substring has both terminals and non-terminals
- The dividing point is marked by a $|$
- The $|$ is not part of the string
- Initially, all input is unexamined $|a_1 a_2 \cdots a_n$

Shift-Reduce Parsing

- A shift-reduce parser is a form of bottom-up parser whose primary operations are

Shift $ABC \mid \texttt{abc} \Rightarrow ABC\texttt{a} \mid \texttt{bc}$

- Move $\mid$ one place to the right
- Shifts a terminal to the left string

Reduce $A\texttt{abc} \mid \texttt{def} \Rightarrow A\texttt{a}B \mid \texttt{def}$

- Apply an inverse production at the right end of the left string
- In the example we use production rule $B \to bc$

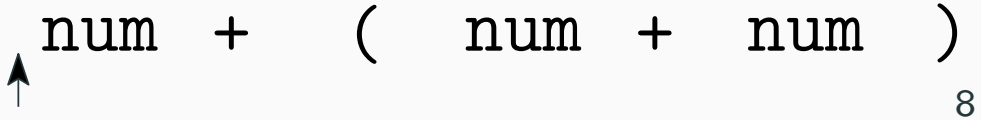
Shift-Reduce Parsing

- read (**shift**) tokens until the right hand side of "correct" production has been seen
- (**reduce**) the right hand side to non-terminal then continue
- done when all input read and reduced to start non-terminal

Shift-Reduce Parsing

| $\texttt{num} + (\texttt{num} + \texttt{num})$ shift

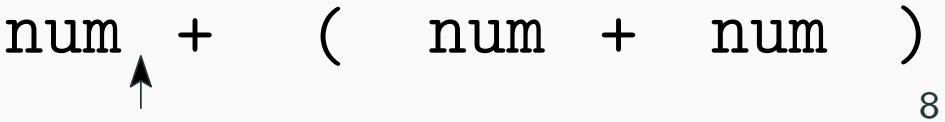
r1	$S \to E$	r4	$E \to T$
r2	$E \to E + T$	r5	$T \to \texttt{num}$
r3	$T \to (E)$		



Shift-Reduce Parsing

$\mid \texttt{num} + (\texttt{num} + \texttt{num})$	shift
$\texttt{num} \mid + (\texttt{num} + \texttt{num})$	reduce (r5)

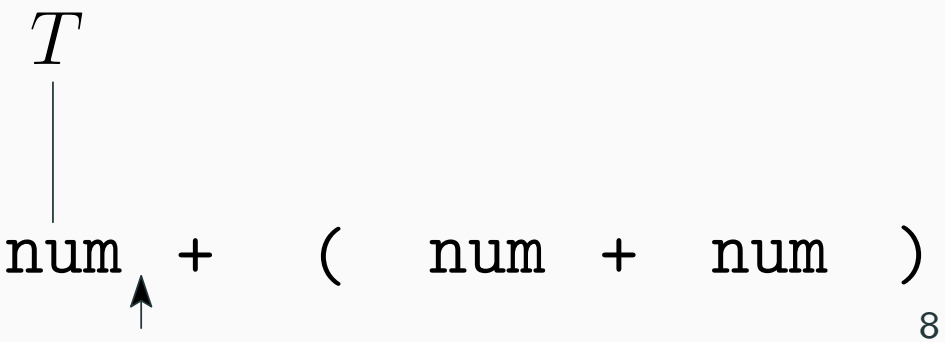
r1	$S \to E$	r4	$E \to T$
r2	$E \to E + T$	r5	$T \to \texttt{num}$
r3	$T \to (E)$		



Shift-Reduce Parsing

$\mid \texttt{num} + (\texttt{num} + \texttt{num})$	shift
$\texttt{num} \mid + (\texttt{num} + \texttt{num})$	reduce (r5)
$T \mid + (\texttt{num} + \texttt{num})$	reduce (r4)

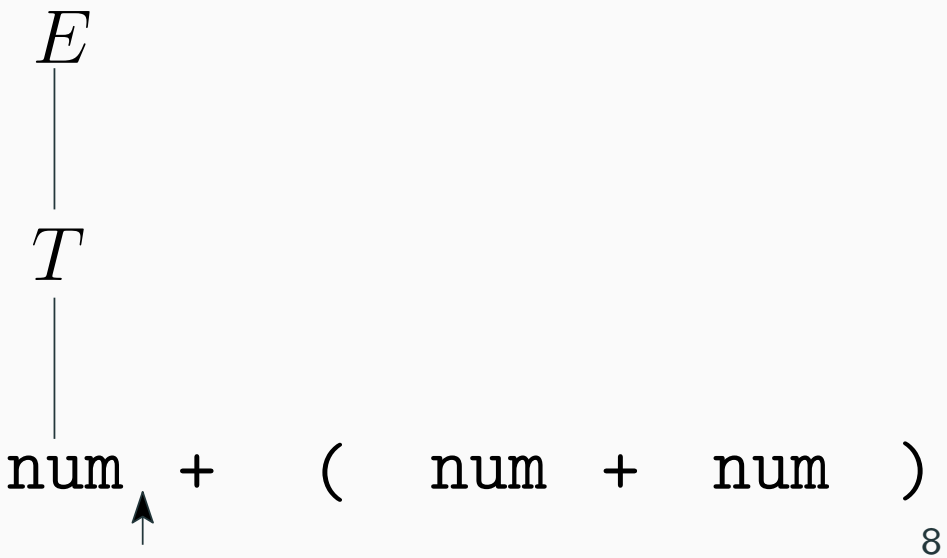
r1	$S \to E$	r4	$E \to T$
r2	$E \to E + T$	r5	$T \to \texttt{num}$
r3	$T \to (E)$		



Shift-Reduce Parsing

$\mid \text{num} + (\text{num} + \text{num})$	shift
$\text{num} \mid + (\text{num} + \text{num})$	reduce (r5)
$T \mid + (\text{num} + \text{num})$	reduce (r4)
$E \mid + (\text{num} + \text{num})$	shift

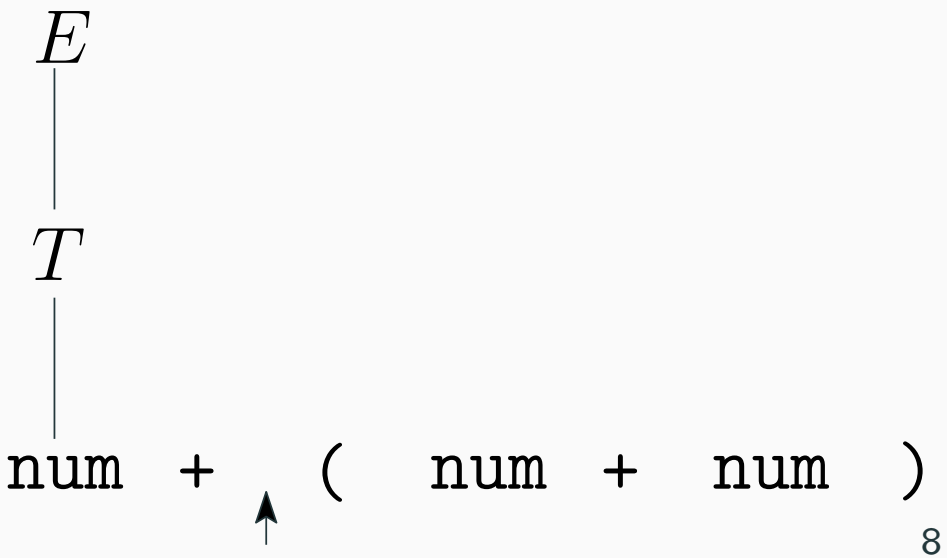
r1	$S \to E$	r4	$E \to T$
r2	$E \to E + T$	r5	$T \to \text{num}$
r3	$T \to (E)$		



Shift-Reduce Parsing

$\mid \text{num} + (\text{num} + \text{num})$	shift
$\text{num} \mid + (\text{num} + \text{num})$	reduce (r5)
$T \mid + (\text{num} + \text{num})$	reduce (r4)
$E \mid + (\text{num} + \text{num})$	shift
$E + \mid (\text{num} + \text{num})$	shift

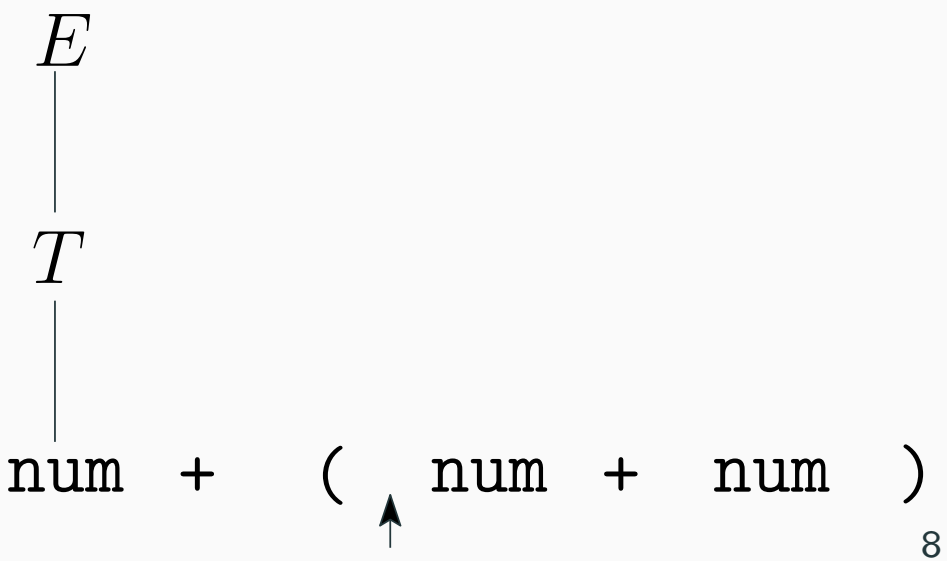
r1	$S \to E$	r4	$E \to T$
r2	$E \to E + T$	r5	$T \to \text{num}$
r3	$T \to (E)$		



Shift-Reduce Parsing

$\mid \text{num} + (\text{num} + \text{num})$	shift
$\text{num} \mid + (\text{num} + \text{num})$	reduce (r5)
$T \mid + (\text{num} + \text{num})$	reduce (r4)
$E \mid + (\text{num} + \text{num})$	shift
$E + \mid (\text{num} + \text{num})$	shift
$E + (\mid \text{num} + \text{num})$	shift

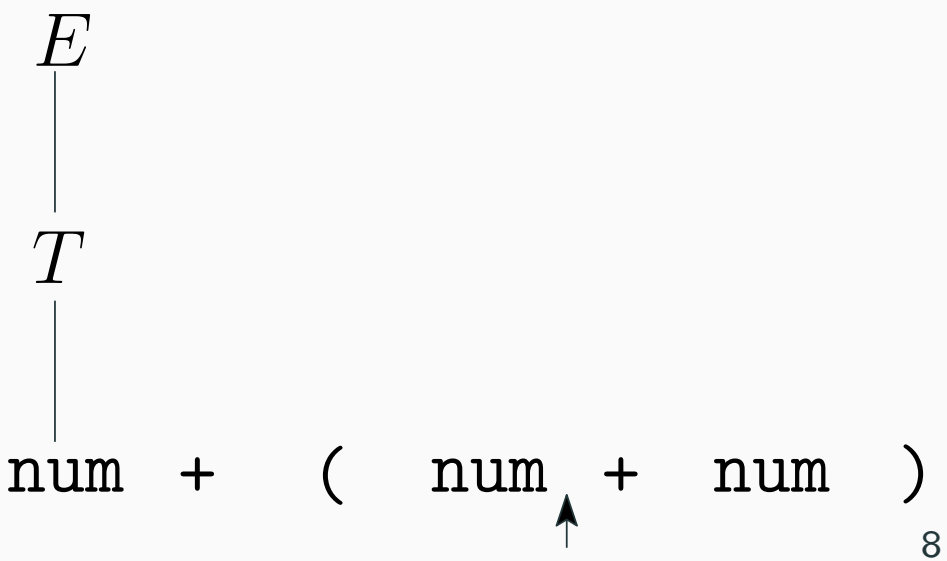
r1	$S \to E$	r4	$E \to T$
r2	$E \to E + T$	r5	$T \to \text{num}$
r3	$T \to (E)$		



Shift-Reduce Parsing

$\mid \text{num} + (\text{num} + \text{num})$	shift
$\text{num} \mid + (\text{num} + \text{num})$	reduce (r5)
$T \mid + (\text{num} + \text{num})$	reduce (r4)
$E \mid + (\text{num} + \text{num})$	shift
$E + \mid (\text{num} + \text{num})$	shift
$E + (\mid \text{num} + \text{num})$	shift
$E + (\text{num} \mid + \text{num})$	reduce (r5)

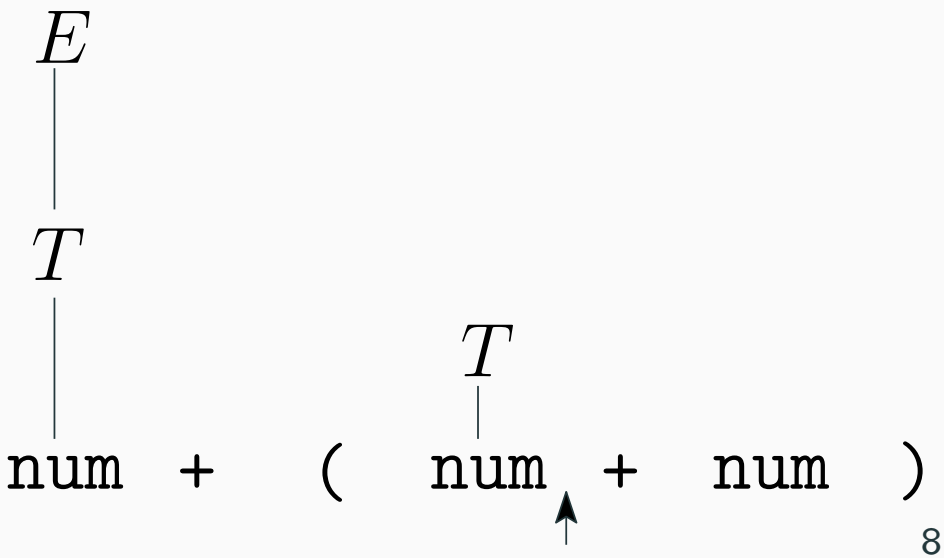
r1	$S \to E$	r4	$E \to T$
r2	$E \to E + T$	r5	$T \to \text{num}$
r3	$T \to (E)$		



Shift-Reduce Parsing

$\mid \text{num} + (\text{num} + \text{num})$	shift
$\text{num} \mid + (\text{num} + \text{num})$	reduce (r5)
$T \mid + (\text{num} + \text{num})$	reduce (r4)
$E \mid + (\text{num} + \text{num})$	shift
$E + \mid (\text{num} + \text{num})$	shift
$E + (\mid \text{num} + \text{num})$	shift
$E + (\text{num} \mid + \text{num})$	reduce (r5)
$E + (T \mid + \text{num})$	reduce (r4)

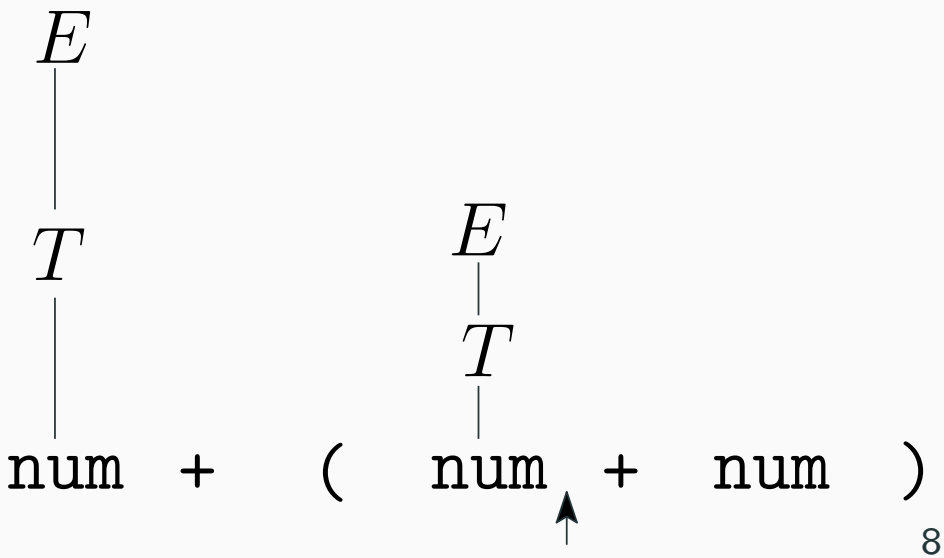
r1	$S \to E$	r4	$E \to T$
r2	$E \to E + T$	r5	$T \to \text{num}$
r3	$T \to (E)$		



Shift-Reduce Parsing

$\mid \text{num} + (\text{num} + \text{num})$	shift
$\text{num} \mid + (\text{num} + \text{num})$	reduce (r5)
$T \mid + (\text{num} + \text{num})$	reduce (r4)
$E \mid + (\text{num} + \text{num})$	shift
$E + \mid (\text{num} + \text{num})$	shift
$E + (\mid \text{num} + \text{num})$	shift
$E + (\text{num} \mid + \text{num})$	reduce (r5)
$E + (T \mid + \text{num})$	reduce (r4)
$E + (E \mid + \text{num})$	shift

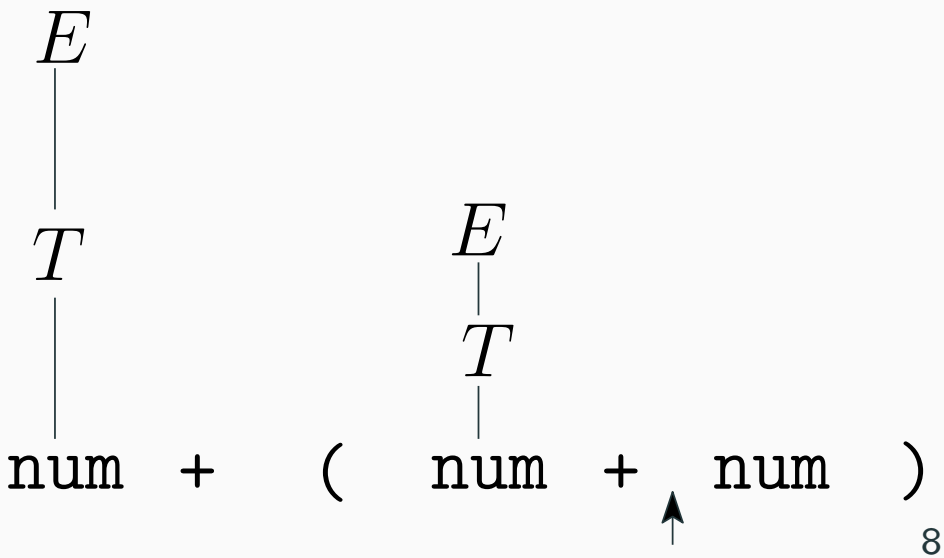
r1	$S \to E$	r4	$E \to T$
r2	$E \to E + T$	r5	$T \to \text{num}$
r3	$T \to (E)$		



Shift-Reduce Parsing

$\mid \text{num} + (\text{num} + \text{num})$	shift
$\text{num} \mid + (\text{num} + \text{num})$	reduce (r5)
$T \mid + (\text{num} + \text{num})$	reduce (r4)
$E \mid + (\text{num} + \text{num})$	shift
$E + \mid (\text{num} + \text{num})$	shift
$E + (\mid \text{num} + \text{num})$	shift
$E + (\text{num} \mid + \text{num})$	reduce (r5)
$E + (T \mid + \text{num})$	reduce (r4)
$E + (E \mid + \text{num})$	shift
$E + (E + \mid \text{num})$	shift

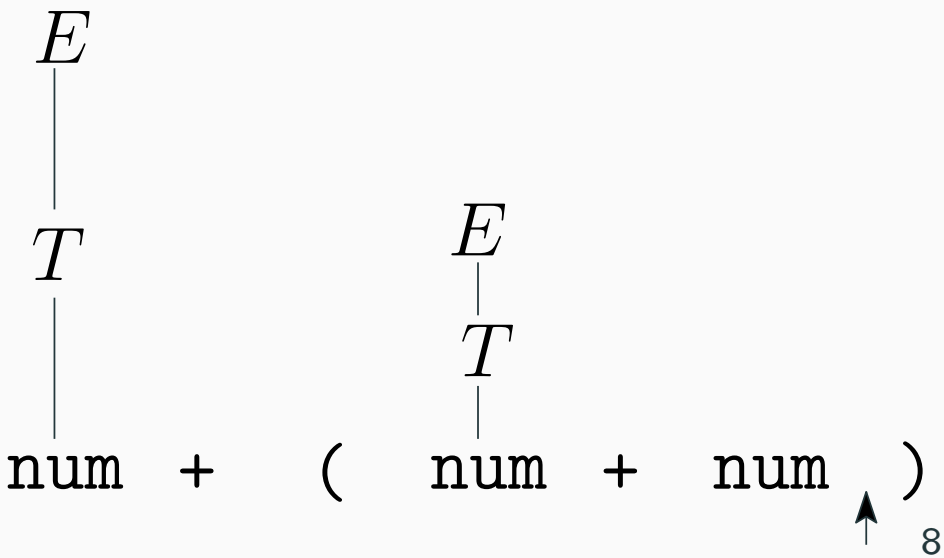
r1	$S \to E$	r4	$E \to T$
r2	$E \to E + T$	r5	$T \to \text{num}$
r3	$T \to (E)$		



Shift-Reduce Parsing

$\mid \text{num} + (\text{num} + \text{num})$	shift
$\text{num} \mid + (\text{num} + \text{num})$	reduce (r5)
$T \mid + (\text{num} + \text{num})$	reduce (r4)
$E \mid + (\text{num} + \text{num})$	shift
$E + \mid (\text{num} + \text{num})$	shift
$E + (\mid \text{num} + \text{num})$	shift
$E + (\text{num} \mid + \text{num})$	reduce (r5)
$E + (T \mid + \text{num})$	reduce (r4)
$E + (E \mid + \text{num})$	shift
$E + (E + \mid \text{num})$	shift
$E + (E + \text{num} \mid)$	reduce (r5)

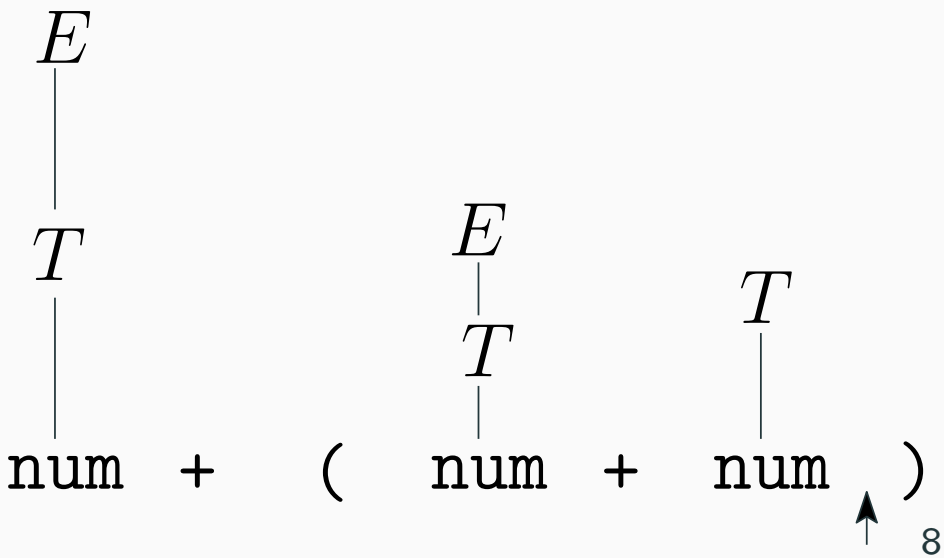
r1	$S \to E$	r4	$E \to T$
r2	$E \to E + T$	r5	$T \to \text{num}$
r3	$T \to (E)$		



Shift-Reduce Parsing

$\mid \text{num} + (\text{num} + \text{num})$	shift
$\text{num} \mid + (\text{num} + \text{num})$	reduce (r5)
$T \mid + (\text{num} + \text{num})$	reduce (r4)
$E \mid + (\text{num} + \text{num})$	shift
$E + \mid (\text{num} + \text{num})$	shift
$E + (\mid \text{num} + \text{num})$	shift
$E + (\text{num} \mid + \text{num})$	reduce (r5)
$E + (T \mid + \text{num})$	reduce (r4)
$E + (E \mid + \text{num})$	shift
$E + (E + \mid \text{num})$	shift
$E + (E + \text{num} \mid)$	reduce (r5)
$E + (E + T \mid)$	reduce (r2)

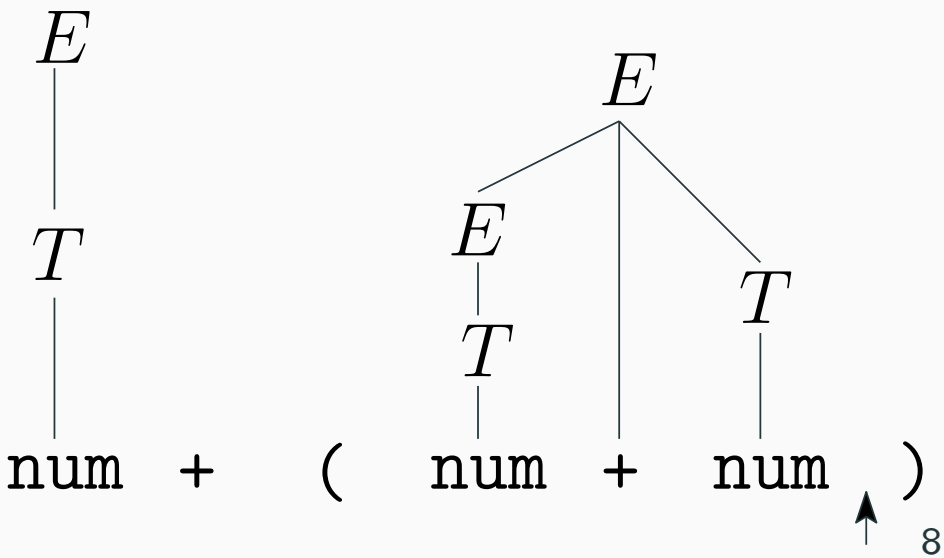
r1	$S \to E$	r4	$E \to T$
r2	$E \to E + T$	r5	$T \to \text{num}$
r3	$T \to (E)$		



Shift-Reduce Parsing

$\mid \text{num} + (\text{num} + \text{num})$	shift
$\text{num} \mid + (\text{num} + \text{num})$	reduce (r5)
$T \mid + (\text{num} + \text{num})$	reduce (r4)
$E \mid + (\text{num} + \text{num})$	shift
$E + \mid (\text{num} + \text{num})$	shift
$E + (\mid \text{num} + \text{num})$	shift
$E + (\text{num} \mid + \text{num})$	reduce (r5)
$E + (T \mid + \text{num})$	reduce (r4)
$E + (E \mid + \text{num})$	shift
$E + (E + \mid \text{num})$	shift
$E + (E + \text{num} \mid)$	reduce (r5)
$E + (E + T \mid)$	reduce (r2)
$E + (E \mid)$	shift

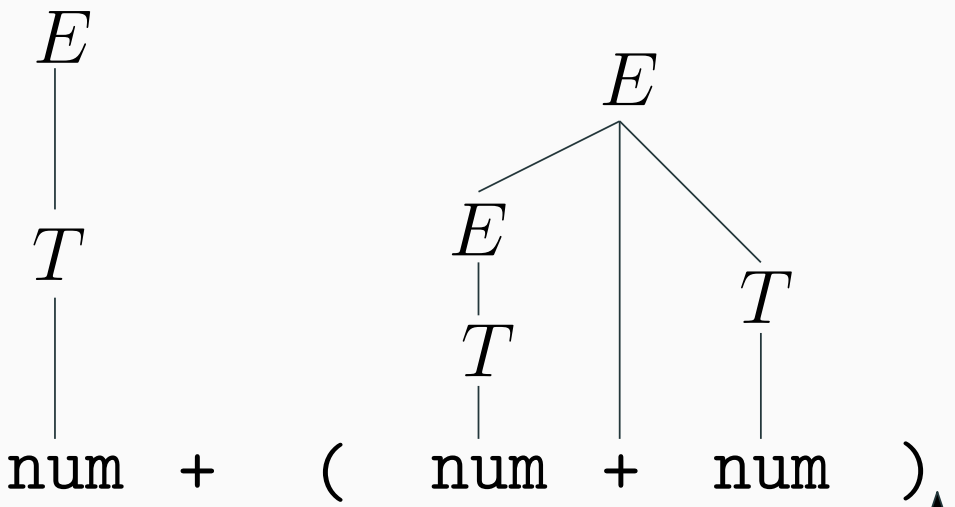
r1	$S \to E$	r4	$E \to T$
r2	$E \to E + T$	r5	$T \to \text{num}$
r3	$T \to (E)$		



Shift-Reduce Parsing

$\mid \text{num} + (\text{num} + \text{num})$	shift
$\text{num} \mid + (\text{num} + \text{num})$	reduce (r5)
$T \mid + (\text{num} + \text{num})$	reduce (r4)
$E \mid + (\text{num} + \text{num})$	shift
$E + \mid (\text{num} + \text{num})$	shift
$E + (\mid \text{num} + \text{num})$	shift
$E + (\text{num} \mid + \text{num})$	reduce (r5)
$E + (T \mid + \text{num})$	reduce (r4)
$E + (E \mid + \text{num})$	shift
$E + (E + \mid \text{num})$	shift
$E + (E + \text{num} \mid)$	reduce (r5)
$E + (E + T \mid)$	reduce (r2)
$E + (E \mid)$	shift
$E + (E) \mid$	reduce (r3)

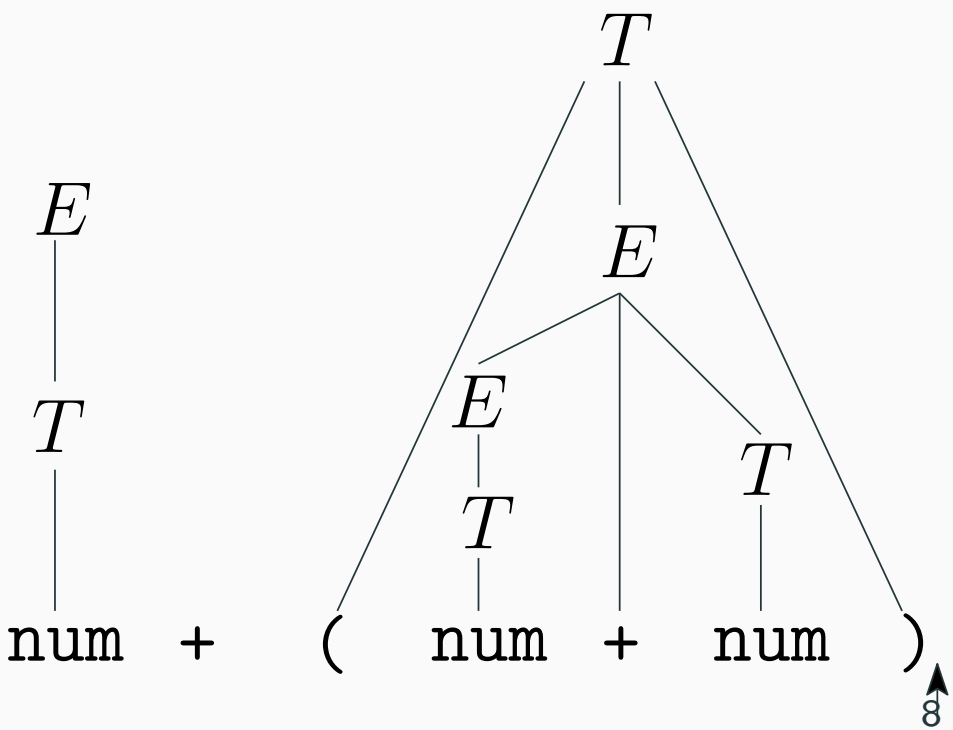
r1	$S \to E$	r4	$E \to T$
r2	$E \to E + T$	r5	$T \to \text{num}$
r3	$T \to (E)$		



Shift-Reduce Parsing

$\mid \text{num} + (\text{num} + \text{num})$	shift
$\text{num} \mid + (\text{num} + \text{num})$	reduce (r5)
$T \mid + (\text{num} + \text{num})$	reduce (r4)
$E \mid + (\text{num} + \text{num})$	shift
$E + \mid (\text{num} + \text{num})$	shift
$E + (\mid \text{num} + \text{num})$	shift
$E + (\text{num} \mid + \text{num})$	reduce (r5)
$E + (T \mid + \text{num})$	reduce (r4)
$E + (E \mid + \text{num})$	shift
$E + (E + \mid \text{num})$	shift
$E + (E + \text{num} \mid)$	reduce (r5)
$E + (E + T \mid)$	reduce (r2)
$E + (E \mid)$	shift
$E + (E) \mid$	reduce (r3)
$E + T \mid$	reduce (r2)

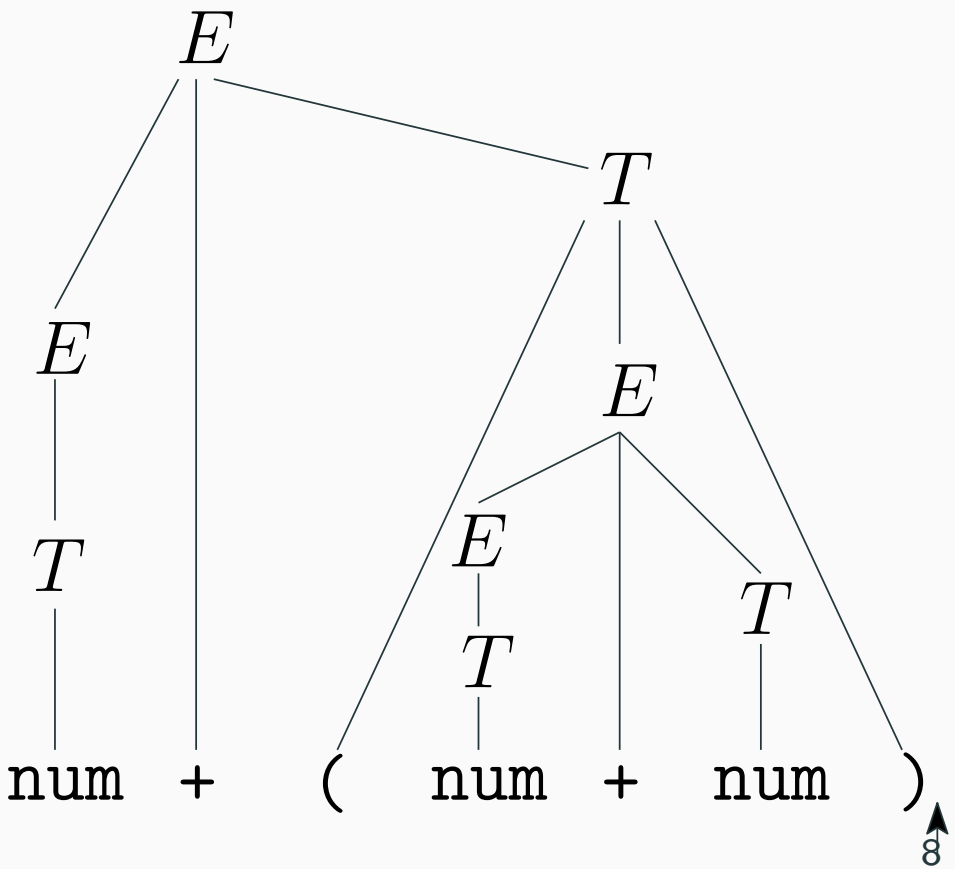
r1	$S \to E$	r4	$E \to T$
r2	$E \to E + T$	r5	$T \to \text{num}$
r3	$T \to (E)$		



Shift-Reduce Parsing

$\mid \text{num} + (\text{num} + \text{num})$	shift
$\text{num} \mid + (\text{num} + \text{num})$	reduce (r5)
$T \mid + (\text{num} + \text{num})$	reduce (r4)
$E \mid + (\text{num} + \text{num})$	shift
$E + \mid (\text{num} + \text{num})$	shift
$E + (\mid \text{num} + \text{num})$	shift
$E + (\text{num} \mid + \text{num})$	reduce (r5)
$E + (T \mid + \text{num})$	reduce (r4)
$E + (E \mid + \text{num})$	shift
$E + (E + \mid \text{num})$	shift
$E + (E + \text{num} \mid)$	reduce (r5)
$E + (E + T \mid)$	reduce (r2)
$E + (E \mid)$	shift
$E + (E) \mid$	reduce (r3)
$E + T \mid$	reduce (r2)
$E \mid$	reduce (r1)

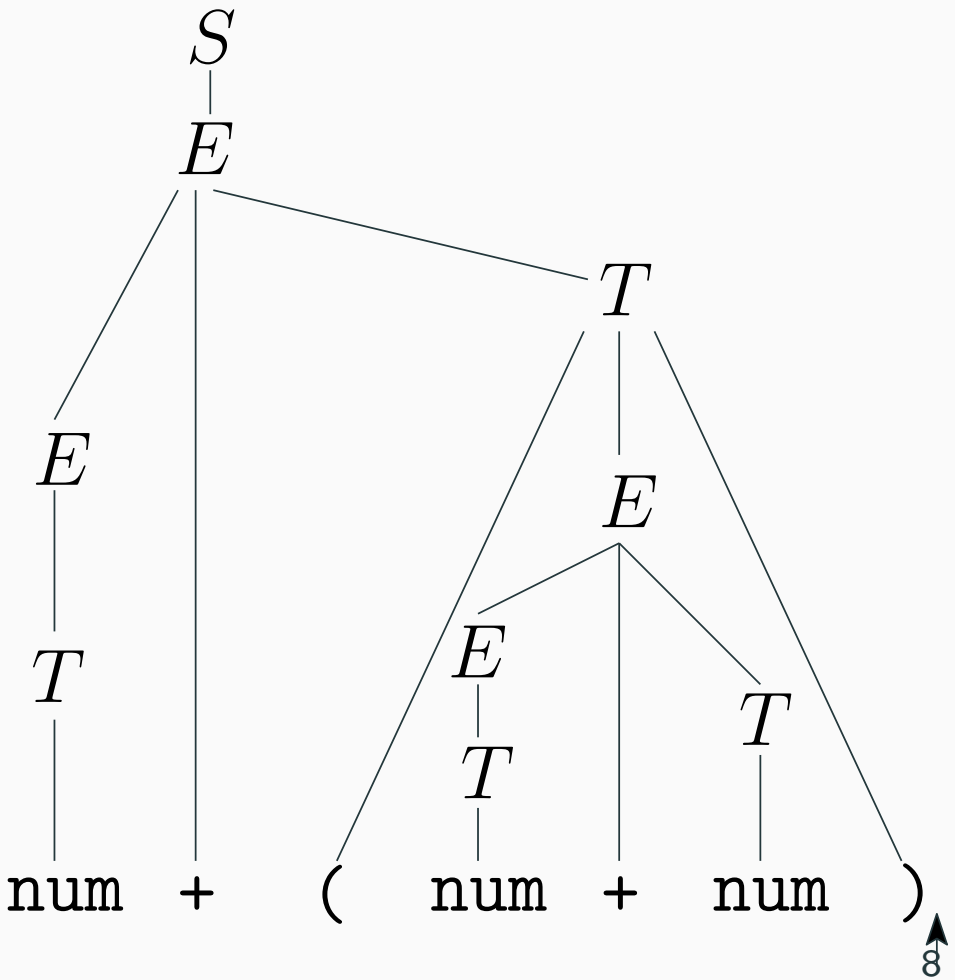
r1	$S \to E$	r4	$E \to T$
r2	$E \to E + T$	r5	$T \to \text{num}$
r3	$T \to (E)$		



Shift-Reduce Parsing

$\mid \text{num} + (\text{num} + \text{num})$	shift
$\text{num} \mid + (\text{num} + \text{num})$	reduce (r5)
$T \mid + (\text{num} + \text{num})$	reduce (r4)
$E \mid + (\text{num} + \text{num})$	shift
$E + \mid (\text{num} + \text{num})$	shift
$E + (\mid \text{num} + \text{num})$	shift
$E + (\text{num} \mid + \text{num})$	reduce (r5)
$E + (T \mid + \text{num})$	reduce (r4)
$E + (E \mid + \text{num})$	shift
$E + (E + \mid \text{num})$	shift
$E + (E + \text{num} \mid)$	reduce (r5)
$E + (E + T \mid)$	reduce (r2)
$E + (E \mid)$	shift
$E + (E) \mid$	reduce (r3)
$E + T \mid$	reduce (r2)
$E \mid$	reduce (r1)
$S \mid$	

r1	$S \to E$	r4	$E \to T$
r2	$E \to E + T$	r5	$T \to \text{num}$
r3	$T \to (E)$		



Stack

- Left substring can be implemented by a stack
- Top of the stack is the |
- **Shift** pushes a terminal on the stack
- **Reduce** pops 0 or more symbols off of the stack (production RHS) and pushes a non-terminal on the stack (production LHS)

Shift-reduce Parsing with stack

derivation	stack	input stream	action
$\mathrm{num} + (\mathrm{num} + \mathrm{num}) \leftarrow$		num+(num+num)	shift
$\mathrm{num} + (\mathrm{num} + \mathrm{num}) \leftarrow$	num	+(num + num)	reduce $T \to \mathrm{num}$
$T + (\mathrm{num} + \mathrm{num}) \leftarrow$	T	+(num + num)	reduce $E \to T$
$E + (\mathrm{num} + \mathrm{num}) \leftarrow$	E	+(num + num)	shift
$E + (\mathrm{num} + \mathrm{num}) \leftarrow$	$E+$	(num + num)	shift
$E + (\mathrm{num} + \mathrm{num}) \leftarrow$	$E + ($	num + num)	shift
$E + (\mathrm{num} + \mathrm{num}) \leftarrow$	$E + (\mathrm{num}$	+num)	reduce $T \to \mathrm{num}$
$E + (T + \mathrm{num}) \leftarrow$	$E + (T$	+num)	reduce $E \to T$
$E + (E + \mathrm{num}) \leftarrow$	$E + (E$	+num)	shift
$E + (E + \mathrm{num}) \leftarrow$	$E + (E+$	num)	shift
$E + (E + \mathrm{num}) \leftarrow$	$E + (E + \mathrm{num}$	)	reduce $E \to T$
$E + (E + T) \leftarrow$	$E + (E + T$	)	reduce $E \to E + T$
$E + (E) \leftarrow$	$E + (E$	)	shift
$E + (E) \leftarrow$	$E + (E)$		reduce $T \to (E)$
$E + T \leftarrow$	$E + T$		reduce $E \to E + T$
E	E		reduce $S \to E$

Shift-Reduce Conflicts

- shift-reduce conflict:
  when it is possible to both shift or reduce

Example:

$$\begin{aligned} S \rightarrow & \texttt{if}\ E\ \texttt{then}\ S \\ \mid & \ \texttt{if}\ E\ \texttt{then}\ S\ \texttt{else}\ S \end{aligned}$$

- Consider step $\texttt{if}\ E\ \texttt{then}\ S \mid$
- If `else` follows then we can shift or reduce

Reduce-Reduce Conflicts

- reduce-reduce conflict:
  when it is possible to reduce by two different productions

Example:

r1	$S \to E$
r2	$E \to E + T$
r3	$T \to (E)$
r4	$E \to T$
r5	$T \to \texttt{num}$

- Consider step $E + T \mid$
- We can reduce by $E \to T$ giving $E + E$
- A fatal mistake! No way to reduce to the start symbol anymore

Problem

- How do we know which action to take: shift or reduce, which production?
- Sometimes can reduce but should not
  - For example $X \to \epsilon$ can always be reduced
- Sometimes can reduce in more than one way

Action Selection Problem

- Given stack σ and look-ahead symbol b, should parser:
- **shift** b onto the stack (making it σb)
- **reduce** some production $X \to \gamma$ assuming stack has the form $\alpha\gamma$ (making it αX)

LR Parsing Engine

- Use a set of parser states
- Use a stack of states
- Use a parsing table to:
  - Determine what action to apply (shift/reduce)
  - Determine the next state
- The parser actions can be precisely determined from the table