Practical Parsing of Context-Free Languages

5DV037 — Fundamentals of Computer Science

Umeå University
Department of Computing Science

Stephen J. Hegner
hegner@cs.umu.se
http://www.cs.umu.se/~hegner

The Need for Practical Parsing

- PDAs form a central theoretical notion of formal language processing.
- However, they are not directly useful in practice for at least two reasons.
  - Nondeterminism: Real parsers must be deterministic.
  - Structural simplicity: PDAs lack the ability to manage complex data structures and algorithms efficiently.

Contexts: There are at least two distinct contexts in which parsing is essential.

- Designed languages: These include in particular most modern programming languages.
  - The language can and should be designed to be parsed efficiently and unambiguously.
- Evolved languages: These include natural (human) languages and some older programming languages.
  - The language must be parsed as it is given.

- Parsing within these two contexts requires somewhat different tools, and each will be addressed separately.

Parsing of Modern Programming Languages

- Modern programming languages are designed to be parsed efficiently.
- Tools are available to construct parsers automatically from the grammar, provided the latter is given in a special form.
- These tools are available at two levels.
- *Scanner generators* take a regular description of the tokens of the language and produce a *lexical analyzer* or *tokenizer*.

  Examples: Lex, Flex, SimpLex
  - Such tools have already been discussed.
- *Parser generators* (or *compiler compilers*) take as input a CFL in a special form and produce an efficient parser.
  - The terminal symbols of this language are the output strings (words) of the lexical analyzer.

  Examples: Yacc (Yet Another Compiler Compiler), Bison

LR(k) Grammars

- The class of grammars which is known to generate precisely the deterministic CFLs is called the $LR(k)$ grammars.
- The formal definition for such grammars is quite technical and will not be given here.
- Standard parsing for such language:
  - Is left to right (hence the L);
  - Produces rightmost derivations (hence the R);
  - Operates bottom up from the input string;
  - Need look ahead at most k symbols to decide exactly what to do next.

Efficiency: The resulting parser runs in time linear in the size of the input string.

- These parsers are typically *table driven* and difficult to construct by hand.
- Thus, these slides will only illustrate the basic ideas of how determinism is achieved, without illustrating the details of how states are determined.

The Context of the Example

- The context will be the simple grammar with start symbol $\langle Expr \rangle$ and the following productions:

$$\begin{aligned}
\langle Ident \rangle &\rightarrow \texttt{A} \mid \texttt{B} \mid \ldots \mid \texttt{Y} \mid \texttt{Z} \\
\langle Expr \rangle &\rightarrow \langle Expr \rangle + \langle Term \rangle \mid \langle Term \rangle \\
\langle Term \rangle &\rightarrow \langle Term \rangle * \langle Factor \rangle \mid \langle Factor \rangle \\
\langle Factor \rangle &\rightarrow (\langle Expr \rangle) \mid \langle Ident \rangle
\end{aligned}$$

- For compactness, this will be abbreviated to the following:

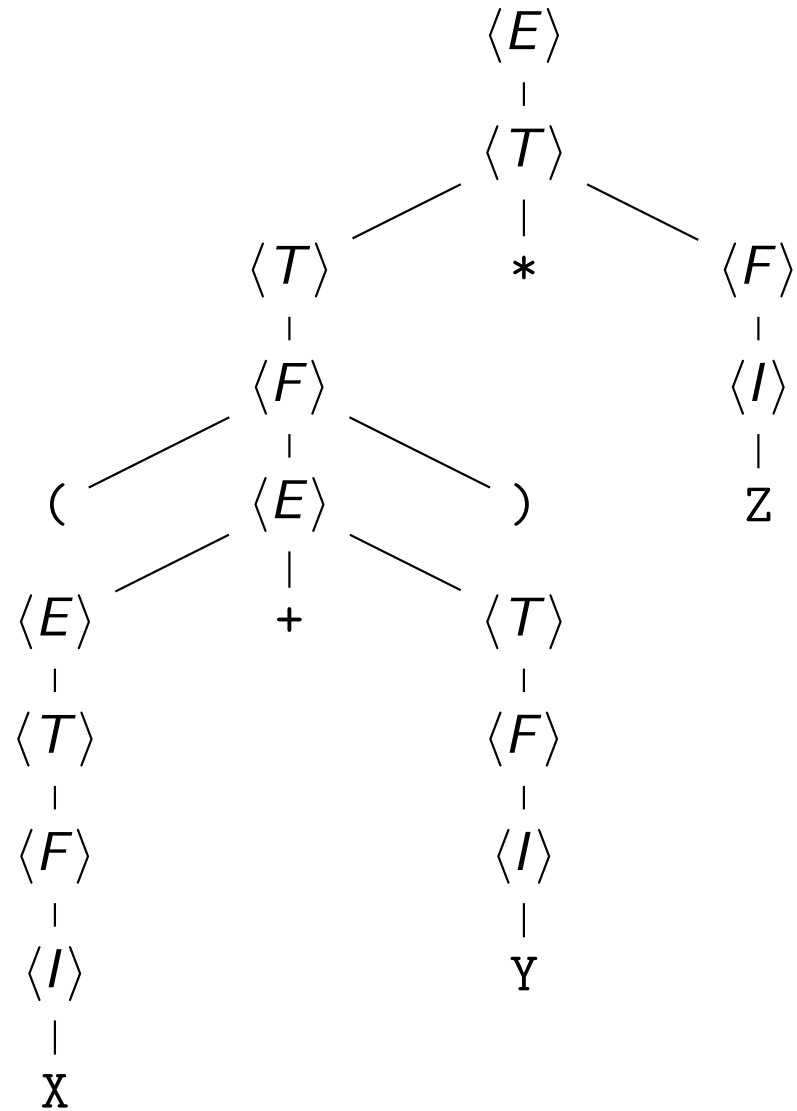
$$\begin{aligned}
\langle I \rangle &\rightarrow \texttt{A} \mid \texttt{B} \mid \ldots \mid \texttt{Y} \mid \texttt{Z} \\
\langle E \rangle &\rightarrow \langle E \rangle + \langle T \rangle \mid \langle T \rangle \\
\langle T \rangle &\rightarrow \langle T \rangle * \langle F \rangle \mid \langle F \rangle \\
\langle F \rangle &\rightarrow (\langle E \rangle) \mid \langle I \rangle
\end{aligned}$$

- The expression to be parsed is `(X+Y)*Z`.
- The dollar sign will be used as an end-of-string marker: `(X+Y)*Z$`.

The Full Parse of the Example Expression

- The parse tree for $(X + Y) * Z$:

$$
\begin{array}{lcl}
\langle I \rangle & \rightarrow & \texttt{A} \mid \texttt{B} \mid \ldots \mid \texttt{Y} \mid \texttt{Z} \\
\langle E \rangle & \rightarrow & \langle E \rangle + \langle T \rangle \mid \langle T \rangle \\
\langle T \rangle & \rightarrow & \langle T \rangle * \langle F \rangle \mid \langle F \rangle \\
\langle F \rangle & \rightarrow & (\langle E \rangle) \mid \langle I \rangle
\end{array}
$$



Shift-Reduce Parsing

- The technique illustrated here is known as *shift-reduce* parsing.
- The input is processed from left to right.
- A list of partial derivation trees is created as the process evolves.
- In a *shift* operation, a new input symbol is processed.
- In a *reduce* operation, a production is applied to the rightmost n partial derivation trees which have already been computed, where n is the number of elements on the right-hand side of the production.
- An internal state in maintained to determine which action to take next.
  - This state is not illustrated explicitly in this example.
- In the example, a lookahead of at most one is required.
  - Thus, the grammar is $LR(1)$.

Example of Shift-Reduce Parsing

$\langle I \rangle \rightarrow$ `A` | `B` | ... | `Y` | `Z`

$\langle E \rangle \rightarrow \langle E \rangle + \langle T \rangle \mid \langle T \rangle$

$\langle T \rangle \rightarrow \langle T \rangle * \langle F \rangle \mid \langle F \rangle$

$\langle F \rangle \rightarrow (\langle E \rangle) \mid \langle I \rangle$

`(X+Y)*Z`

- The input is initialized to the entire string `(X+Y)*Z$`.
- The first step is a shift; the left parenthesis is removed from the input and becomes a one-vertex tree.
- At this point, the system knows that the production $\langle F \rangle \rightarrow (\langle E \rangle)$ must be applied to reduce it, since it is the only production involving a left parenthesis.
- This information is recorded in an internal state (not shown).
- No reduction is possible at this point since the production $\langle F \rangle \rightarrow (\langle E \rangle)$ requires additional terminals.

`X+Y)*Z$` (

Example of Shift-Reduce Parsing — 2

$\langle I \rangle \rightarrow$ `A` | `B` | ... | `Y` | `Z`

$\langle E \rangle \rightarrow \langle E \rangle + \langle T \rangle \mid \langle T \rangle$

$\langle T \rangle \rightarrow \langle T \rangle * \langle F \rangle \mid \langle F \rangle$

$\langle F \rangle \rightarrow (\langle E \rangle) \mid \langle I \rangle$

`(X+Y)*Z`

- The next step is to process the input symbol `X`.
- This begins with a shift.
- Regardless of what is to follow, this vertex may be reduced with $\langle I \rangle \rightarrow$ `X`.
- and then $\langle F \rangle \rightarrow \langle I \rangle$, and then $\langle T \rangle \rightarrow \langle F \rangle$.
- This is as far as `X` may be reduced without further information.

`X+Y)*Z$` (

Example of Shift-Reduce Parsing — 2

$\langle I \rangle \rightarrow$ `A` | `B` | ... | `Y` | `Z`

$\langle E \rangle \rightarrow \langle E \rangle + \langle T \rangle \mid \langle T \rangle$

$\langle T \rangle \rightarrow \langle T \rangle * \langle F \rangle \mid \langle F \rangle$

$\langle F \rangle \rightarrow (\langle E \rangle) \mid \langle I \rangle$

`(X+Y)*Z`

- The next step is to process the input symbol `X`.
- This begins with a shift.
- Regardless of what is to follow, this vertex may be reduced with $\langle I \rangle \rightarrow$ `X`.
- and then $\langle F \rangle \rightarrow \langle I \rangle$, and then $\langle T \rangle \rightarrow \langle F \rangle$.
- This is as far as `X` may be reduced without further information.

`+Y)*Z$` (`(` (`X`

Example of Shift-Reduce Parsing — 2

$\langle I \rangle \rightarrow$ `A` | `B` | ... | `Y` | `Z`

$\langle E \rangle \rightarrow \langle E \rangle + \langle T \rangle \mid \langle T \rangle$

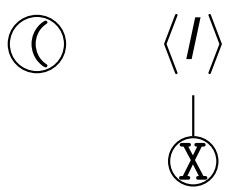
$\langle T \rangle \rightarrow \langle T \rangle * \langle F \rangle \mid \langle F \rangle$

$\langle F \rangle \rightarrow (\langle E \rangle) \mid \langle I \rangle$

`(X+Y)*Z`

- The next step is to process the input symbol `X`.
- This begins with a shift.
- Regardless of what is to follow, this vertex may be reduced with $\langle I \rangle \rightarrow$ `X`.
- and then $\langle F \rangle \rightarrow \langle I \rangle$, and then $\langle T \rangle \rightarrow \langle F \rangle$.
- This is as far as `X` may be reduced without further information.

`+Y)*Z$`



Example of Shift-Reduce Parsing — 2

$\langle I \rangle \rightarrow$ `A` | `B` | ... | `Y` | `Z`

$\langle E \rangle \rightarrow \langle E \rangle + \langle T \rangle \mid \langle T \rangle$

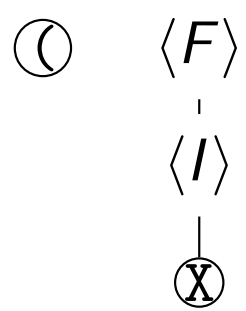
$\langle T \rangle \rightarrow \langle T \rangle * \langle F \rangle \mid \langle F \rangle$

$\langle F \rangle \rightarrow (\langle E \rangle) \mid \langle I \rangle$

`(X+Y)*Z`

- The next step is to process the input symbol `X`.
- This begins with a shift.
- Regardless of what is to follow, this vertex may be reduced with $\langle I \rangle \rightarrow$ `X`.
- and then $\langle F \rangle \rightarrow \langle I \rangle$, and then $\langle T \rangle \rightarrow \langle F \rangle$.
- This is as far as `X` may be reduced without further information.

`+Y)*Z$`



Example of Shift-Reduce Parsing — 2

$\langle I \rangle \rightarrow$ `A` | `B` | ... | `Y` | `Z`

$\langle E \rangle \rightarrow \langle E \rangle + \langle T \rangle \mid \langle T \rangle$

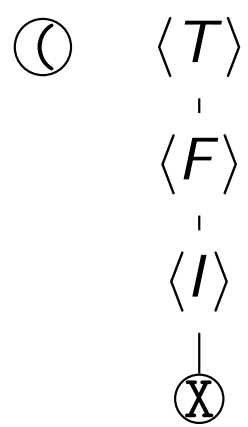
$\langle T \rangle \rightarrow \langle T \rangle * \langle F \rangle \mid \langle F \rangle$

$\langle F \rangle \rightarrow (\langle E \rangle) \mid \langle I \rangle$

`(X+Y)*Z`

- The next step is to process the input symbol `X`.
- This begins with a shift.
- Regardless of what is to follow, this vertex may be reduced with $\langle I \rangle \rightarrow$ `X`.
- and then $\langle F \rangle \rightarrow \langle I \rangle$, and then $\langle T \rangle \rightarrow \langle F \rangle$.
- This is as far as `X` may be reduced without further information.

`+Y)*Z$`



Example of Shift-Reduce Parsing — 3

$\langle I \rangle \rightarrow$ `A` | `B` | ... | `Y` | `Z`

$\langle E \rangle \rightarrow \langle E \rangle + \langle T \rangle \mid \langle T \rangle$

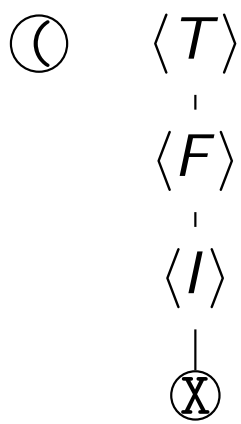
$\langle T \rangle \rightarrow \langle T \rangle * \langle F \rangle \mid \langle F \rangle$

$\langle F \rangle \rightarrow (\langle E \rangle) \mid \langle I \rangle$

`(X+Y)*Z`

- To proceed further requires a *lookahead*.
- Without shifting it to the forest, the next symbol + is identified.
- This enables the system to know that the tree with leaf `X` may be reduced with $\langle E \rangle \rightarrow \langle T \rangle$.
- If the next symbol were instead $*$, this reduction would be incorrect.

`+Y)*Z$`



Example of Shift-Reduce Parsing — 3

$\langle I \rangle \rightarrow$ `A` | `B` | ... | `Y` | `Z`

$\langle E \rangle \rightarrow \langle E \rangle + \langle T \rangle \mid \langle T \rangle$

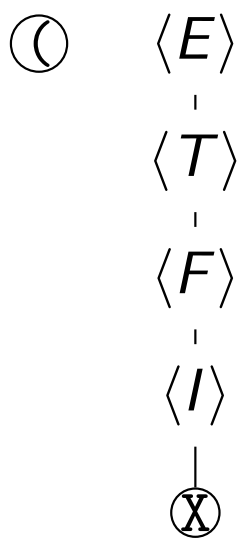
$\langle T \rangle \rightarrow \langle T \rangle * \langle F \rangle \mid \langle F \rangle$

$\langle F \rangle \rightarrow (\langle E \rangle) \mid \langle I \rangle$

`(X+Y)*Z`

- To proceed further requires a *lookahead*.
- Without shifting it to the forest, the next symbol + is identified.
- This enables the system to know that the tree with leaf `X` may be reduced with $\langle E \rangle \rightarrow \langle T \rangle$.
- If the next symbol were instead $*$, this reduction would be incorrect.

`+Y)*Z$`



Example of Shift-Reduce Parsing — 4

$\langle I \rangle \rightarrow$ `A` | `B` | ... | `Y` | `Z`

$\langle E \rangle \rightarrow \langle E \rangle + \langle T \rangle \mid \langle T \rangle$

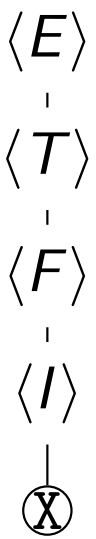
$\langle T \rangle \rightarrow \langle T \rangle * \langle F \rangle \mid \langle F \rangle$

$\langle F \rangle \rightarrow (\langle E \rangle) \mid \langle I \rangle$

`(X+Y)*Z`

- The next step is to shift the +.
- No further reduction is possible without another shift.
- So, shift the next symbol `Y`.
- and reduce it with $\langle I \rangle \rightarrow Y$ and then $\langle F \rangle \rightarrow \langle I \rangle$, and then $\langle T \rangle \rightarrow \langle F \rangle$.

`+Y)*Z$` (



Example of Shift-Reduce Parsing — 4

$\langle I \rangle \rightarrow$ `A` | `B` | ... | `Y` | `Z`

$\langle E \rangle \rightarrow \langle E \rangle + \langle T \rangle \mid \langle T \rangle$

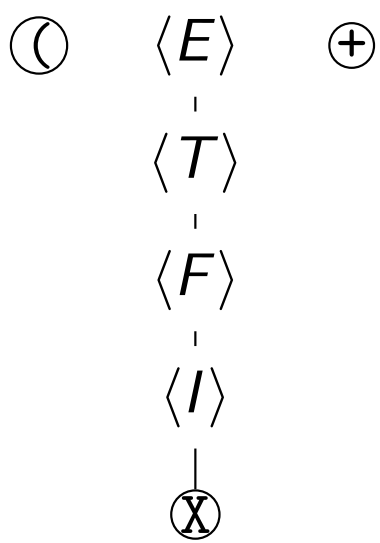
$\langle T \rangle \rightarrow \langle T \rangle * \langle F \rangle \mid \langle F \rangle$

$\langle F \rangle \rightarrow (\langle E \rangle) \mid \langle I \rangle$

`(X+Y)*Z`

- The next step is to shift the +.
- No further reduction is possible without another shift.
- So, shift the next symbol `Y`.
- and reduce it with $\langle I \rangle \rightarrow Y$ and then $\langle F \rangle \rightarrow \langle I \rangle$, and then $\langle T \rangle \rightarrow \langle F \rangle$.

`Y)*Z$`



Example of Shift-Reduce Parsing — 4

$\langle I \rangle \rightarrow$ `A` | `B` | ... | `Y` | `Z`

$\langle E \rangle \rightarrow \langle E \rangle + \langle T \rangle \mid \langle T \rangle$

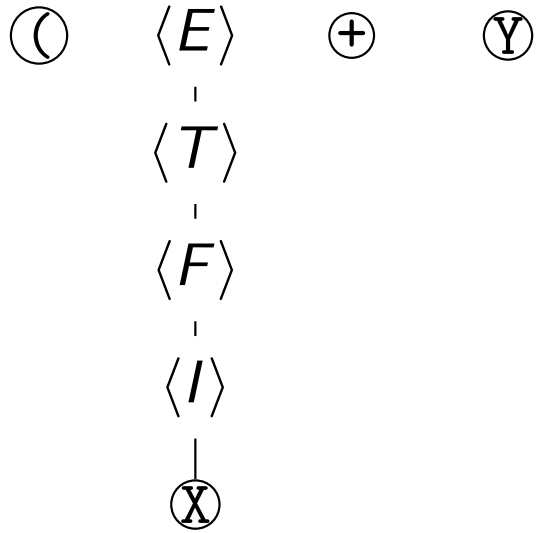
$\langle T \rangle \rightarrow \langle T \rangle * \langle F \rangle \mid \langle F \rangle$

$\langle F \rangle \rightarrow (\langle E \rangle) \mid \langle I \rangle$

`(X+Y)*Z`

- The next step is to shift the +.
- No further reduction is possible without another shift.
- So, shift the next symbol `Y`.
- and reduce it with $\langle I \rangle \rightarrow Y$ and then $\langle F \rangle \rightarrow \langle I \rangle$, and then $\langle T \rangle \rightarrow \langle F \rangle$.

`)*Z$`



Example of Shift-Reduce Parsing — 4

$\langle I \rangle \rightarrow$ `A` | `B` | ... | `Y` | `Z`

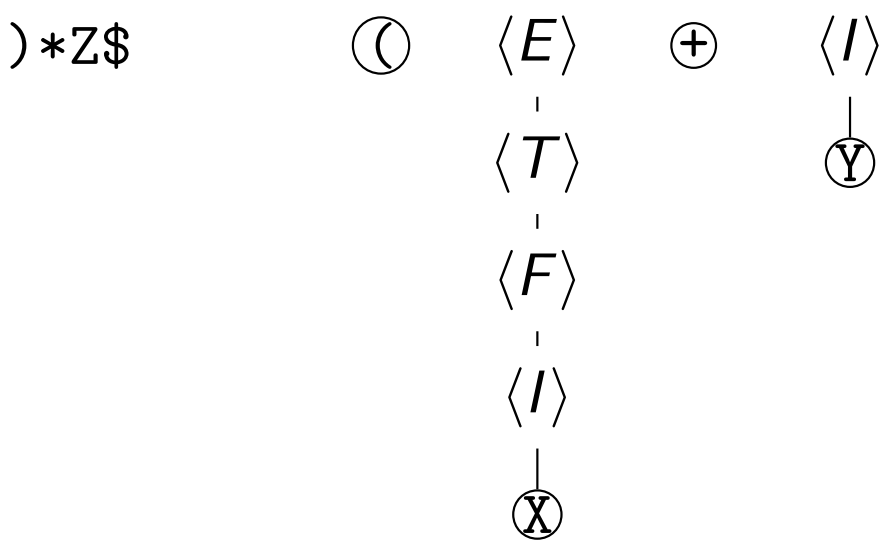
$\langle E \rangle \rightarrow \langle E \rangle + \langle T \rangle \mid \langle T \rangle$

$\langle T \rangle \rightarrow \langle T \rangle * \langle F \rangle \mid \langle F \rangle$

$\langle F \rangle \rightarrow (\langle E \rangle) \mid \langle I \rangle$

`(X+Y)*Z`

- The next step is to shift the +.
- No further reduction is possible without another shift.
- So, shift the next symbol `Y`.
- and reduce it with $\langle I \rangle \rightarrow Y$ and then $\langle F \rangle \rightarrow \langle I \rangle$, and then $\langle T \rangle \rightarrow \langle F \rangle$.



Example of Shift-Reduce Parsing — 4

$\langle I \rangle \rightarrow$ `A` | `B` | ... | `Y` | `Z`

$\langle E \rangle \rightarrow \langle E \rangle + \langle T \rangle \mid \langle T \rangle$

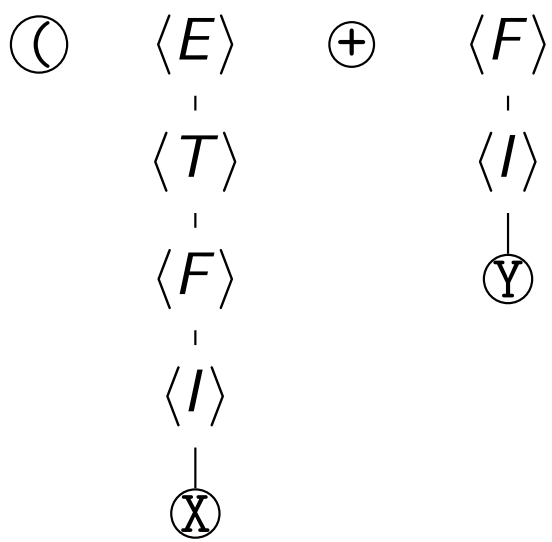
$\langle T \rangle \rightarrow \langle T \rangle * \langle F \rangle \mid \langle F \rangle$

$\langle F \rangle \rightarrow (\langle E \rangle) \mid \langle I \rangle$

`(X+Y)*Z`

- The next step is to shift the +.
- No further reduction is possible without another shift.
- So, shift the next symbol `Y`.
- and reduce it with $\langle I \rangle \rightarrow Y$ and then $\langle F \rangle \rightarrow \langle I \rangle$, and then $\langle T \rangle \rightarrow \langle F \rangle$.

`)*Z$`



Example of Shift-Reduce Parsing — 4

$\langle I \rangle \rightarrow$ `A` | `B` | ... | `Y` | `Z`

$\langle E \rangle \rightarrow \langle E \rangle + \langle T \rangle \mid \langle T \rangle$

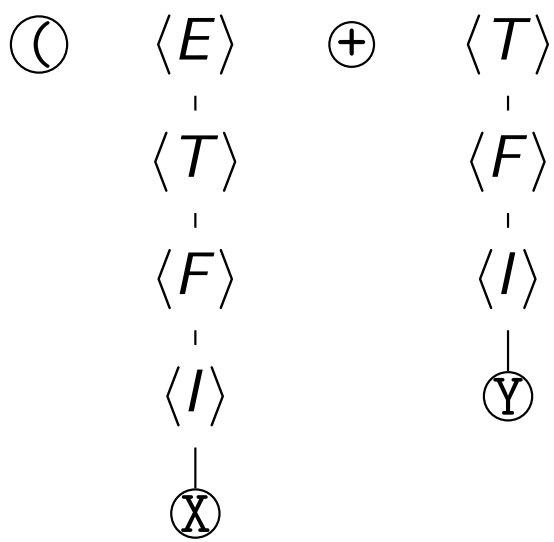
$\langle T \rangle \rightarrow \langle T \rangle * \langle F \rangle \mid \langle F \rangle$

$\langle F \rangle \rightarrow (\langle E \rangle) \mid \langle I \rangle$

`(X+Y)*Z`

- The next step is to shift the +.
- No further reduction is possible without another shift.
- So, shift the next symbol `Y`.
- and reduce it with $\langle I \rangle \rightarrow Y$ and then $\langle F \rangle \rightarrow \langle I \rangle$, and then $\langle T \rangle \rightarrow \langle F \rangle$.

`)*Z$`



Example of Shift-Reduce Parsing — 5

$\langle I \rangle \rightarrow$ A | B | ... | Y | Z

$\langle E \rangle \rightarrow \langle E \rangle + \langle T \rangle \mid \langle T \rangle$

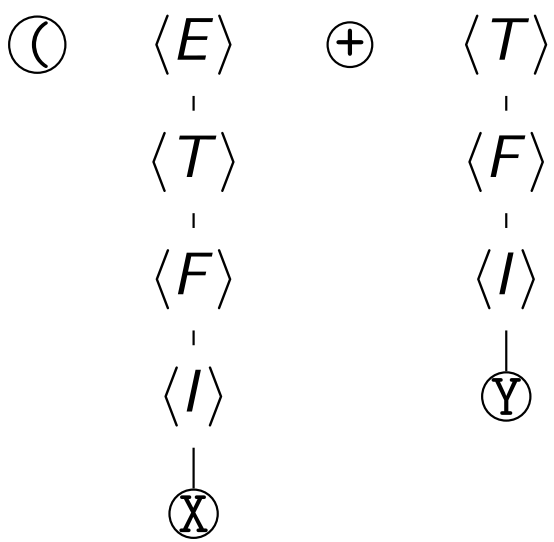
$\langle T \rangle \rightarrow \langle T \rangle * \langle F \rangle \mid \langle F \rangle$

$\langle F \rangle \rightarrow (\langle E \rangle) \mid \langle I \rangle$

(X+Y)*Z

- The next step involves a reduction of the rightmost three trees.
- using the production $\langle E \rangle \rightarrow \langle E \rangle + \langle T \rangle$.
- This is followed by a shift of the next input symbol).

)*Z$



Example of Shift-Reduce Parsing — 5

$\langle I \rangle \rightarrow$ A | B | ... | Y | Z

$\langle E \rangle \rightarrow \langle E \rangle + \langle T \rangle \mid \langle T \rangle$

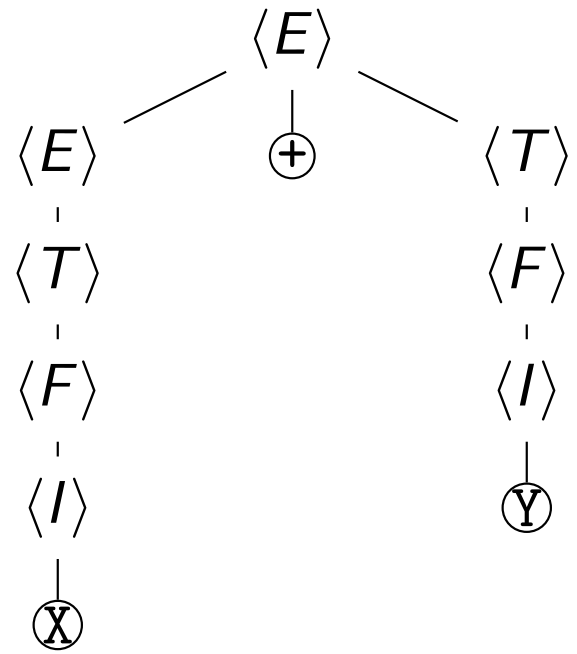
$\langle T \rangle \rightarrow \langle T \rangle * \langle F \rangle \mid \langle F \rangle$

$\langle F \rangle \rightarrow (\langle E \rangle) \mid \langle I \rangle$

(X+Y)*Z

- The next step involves a reduction of the rightmost three trees.
- using the production $\langle E \rangle \rightarrow \langle E \rangle + \langle T \rangle$.
- This is followed by a shift of the next input symbol).

)*Z$ (



Example of Shift-Reduce Parsing — 5

$\langle I \rangle \rightarrow$ `A` | `B` | ... | `Y` | `Z`

$\langle E \rangle \rightarrow \langle E \rangle + \langle T \rangle \mid \langle T \rangle$

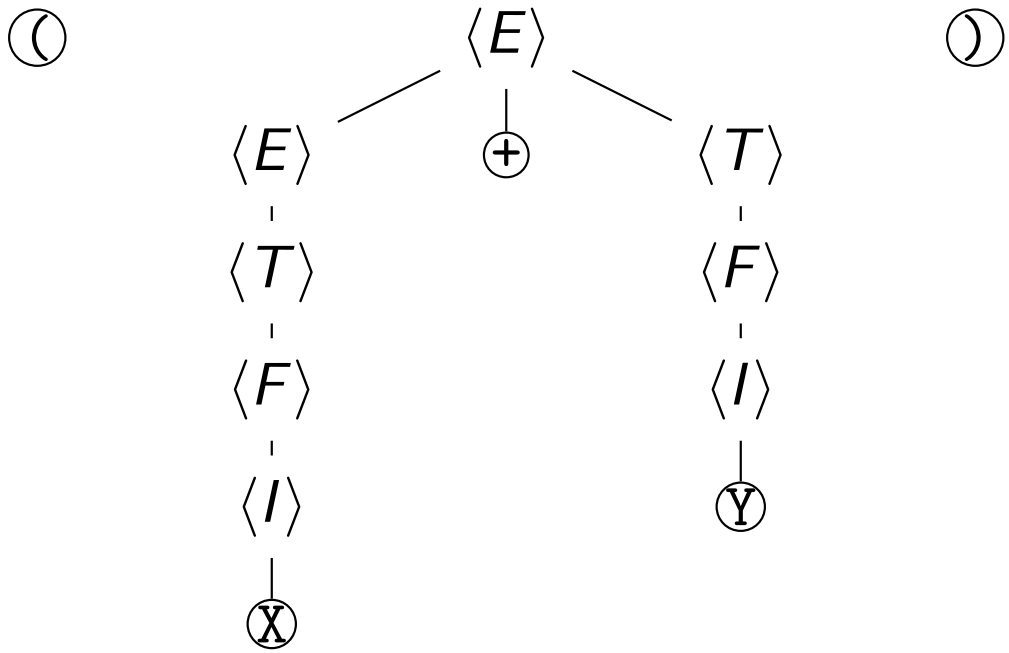
$\langle T \rangle \rightarrow \langle T \rangle * \langle F \rangle \mid \langle F \rangle$

$\langle F \rangle \rightarrow (\langle E \rangle) \mid \langle I \rangle$

`(X+Y)*Z`

- The next step involves a reduction of the rightmost three trees.
- using the production $\langle E \rangle \rightarrow \langle E \rangle + \langle T \rangle$.
- This is followed by a shift of the next input symbol `)`.

`*Z$`



Example of Shift-Reduce Parsing — 6

$\langle I \rangle \rightarrow$ A | B | ... | Y | Z

$\langle E \rangle \rightarrow \langle E \rangle + \langle T \rangle \mid \langle T \rangle$

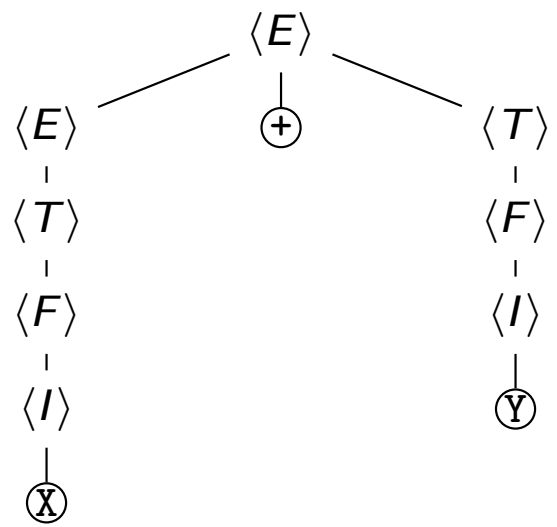
$\langle T \rangle \rightarrow \langle T \rangle * \langle F \rangle \mid \langle F \rangle$

$\langle F \rangle \rightarrow (\langle E \rangle) \mid \langle I \rangle$

(X+Y)*Z

- Now the rightmost three trees may be reduced with $\langle F \rangle \rightarrow (\langle E \rangle)$.
- yielding a single tree.
- A further reduction with $\langle T \rangle \rightarrow \langle F \rangle$ is always necessary at this point.

*Z$ (



)

Example of Shift-Reduce Parsing — 6

$\langle I \rangle \rightarrow$ `A` | `B` | ... | `Y` | `Z`

$\langle E \rangle \rightarrow \langle E \rangle + \langle T \rangle \mid \langle T \rangle$

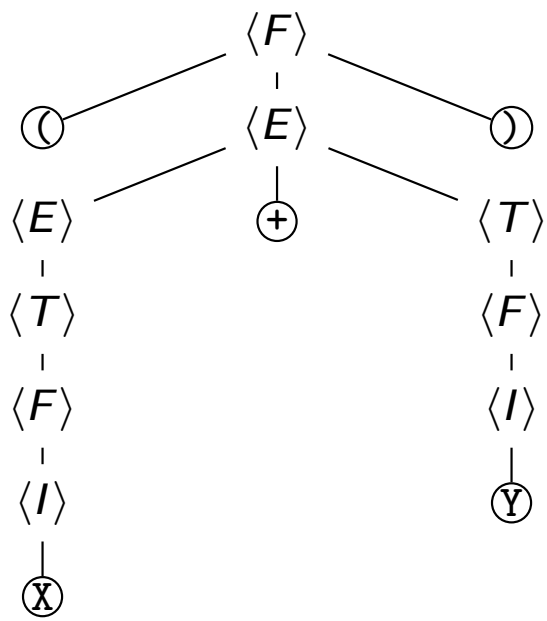
$\langle T \rangle \rightarrow \langle T \rangle * \langle F \rangle \mid \langle F \rangle$

$\langle F \rangle \rightarrow (\langle E \rangle) \mid \langle I \rangle$

`(X+Y)*Z`

- Now the rightmost three trees may be reduced with $\langle F \rangle \rightarrow (\langle E \rangle)$.
- yielding a single tree.
- A further reduction with $\langle T \rangle \rightarrow \langle F \rangle$ is always necessary at this point.

*Z$



Example of Shift-Reduce Parsing — 6

$\langle I \rangle \rightarrow$ `A` | `B` | ... | `Y` | `Z`

$\langle E \rangle \rightarrow \langle E \rangle + \langle T \rangle \mid \langle T \rangle$

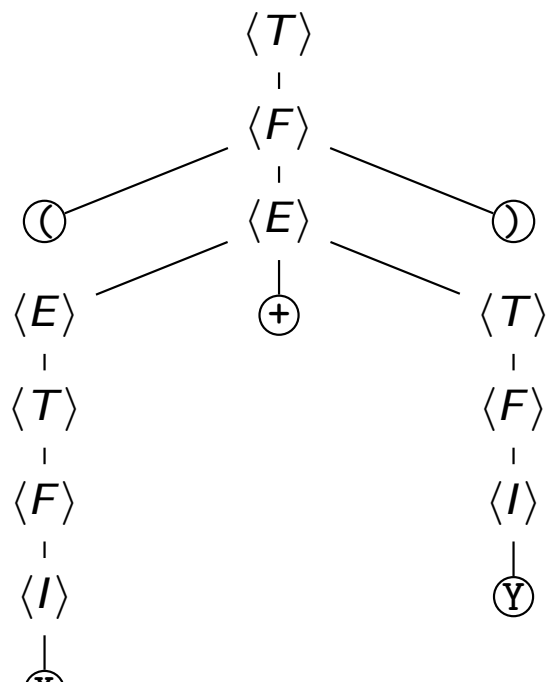
$\langle T \rangle \rightarrow \langle T \rangle * \langle F \rangle \mid \langle F \rangle$

$\langle F \rangle \rightarrow (\langle E \rangle) \mid \langle I \rangle$

`(X+Y)*Z`

- Now the rightmost three trees may be reduced with $\langle F \rangle \rightarrow (\langle E \rangle)$.
- yielding a single tree.
- A further reduction with $\langle T \rangle \rightarrow \langle F \rangle$ is always necessary at this point.

`*Z$`



Example of Shift-Reduce Parsing — 7

nn

$\langle I \rangle \rightarrow$ `A` | `B` | ... | `Y` | `Z`

$\langle T \rangle \rightarrow \langle T \rangle * \langle F \rangle \mid \langle F \rangle$

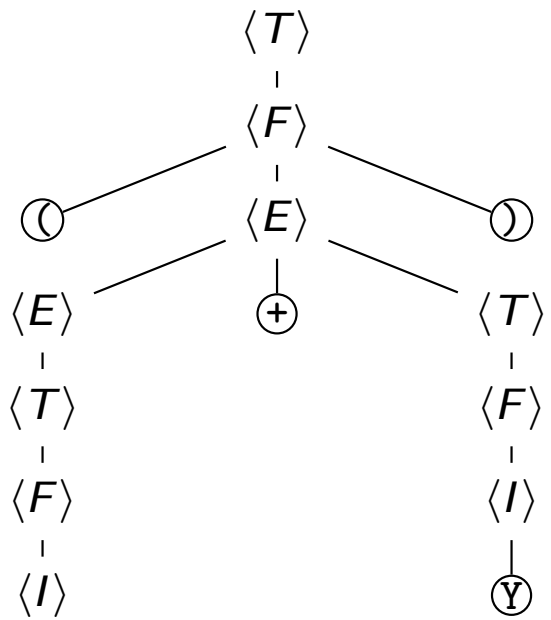
`(X+Y)*Z`

$\langle E \rangle \rightarrow \langle E \rangle + \langle T \rangle \mid \langle T \rangle$

$\langle F \rangle \rightarrow (\langle E \rangle) \mid \langle I \rangle$

- A lookahead is required to determine whether or not to reduce this tree with $\langle E \rangle \rightarrow \langle T \rangle$.
- If the next character were + or end of string, the answer would be yes.
- However, it is *, so the answer is no.
- Thus, a shift is performed, and another.

*Z$



Example of Shift-Reduce Parsing — 7

nn

$\langle I \rangle \rightarrow$ A | B | ... | Y | Z

$\langle E \rangle \rightarrow \langle E \rangle + \langle T \rangle \mid \langle T \rangle$

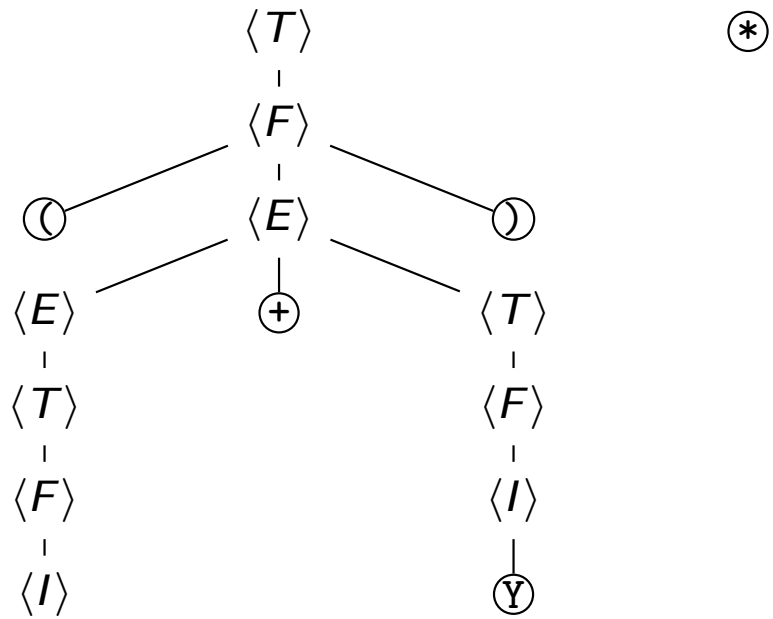
$\langle T \rangle \rightarrow \langle T \rangle * \langle F \rangle \mid \langle F \rangle$

$\langle F \rangle \rightarrow (\langle E \rangle) \mid \langle I \rangle$

(X+Y)*Z

- A lookahead is required to determine whether or not to reduce this tree with $\langle E \rangle \rightarrow \langle T \rangle$.
- If the next character were + or end of string, the answer would be yes.
- However, it is *, so the answer is no.
- Thus, a shift is performed, and another.

Z$



Example of Shift-Reduce Parsing — 7

nn

$\langle I \rangle \rightarrow$ `A` | `B` | ... | `Y` | `Z`

$\langle T \rangle \rightarrow \langle T \rangle * \langle F \rangle \mid \langle F \rangle$

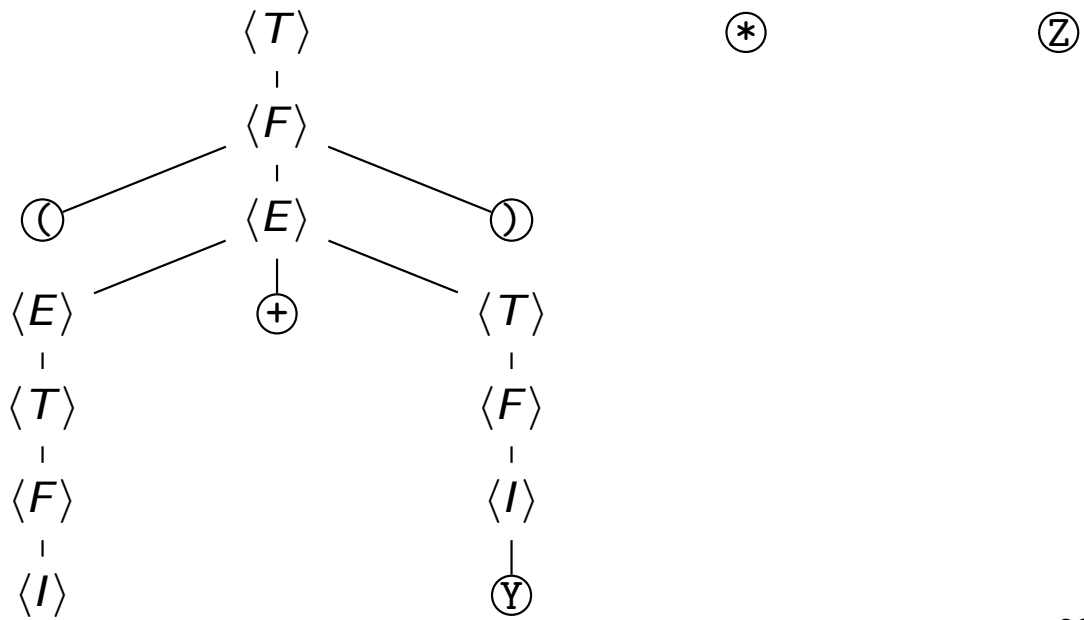
`(X+Y)*Z`

$\langle E \rangle \rightarrow \langle E \rangle + \langle T \rangle \mid \langle T \rangle$

$\langle F \rangle \rightarrow (\langle E \rangle) \mid \langle I \rangle$

- A lookahead is required to determine whether or not to reduce this tree with $\langle E \rangle \rightarrow \langle T \rangle$.
- If the next character were + or end of string, the answer would be yes.
- However, it is $*$, so the answer is no.
- Thus, a shift is performed, and another.

$

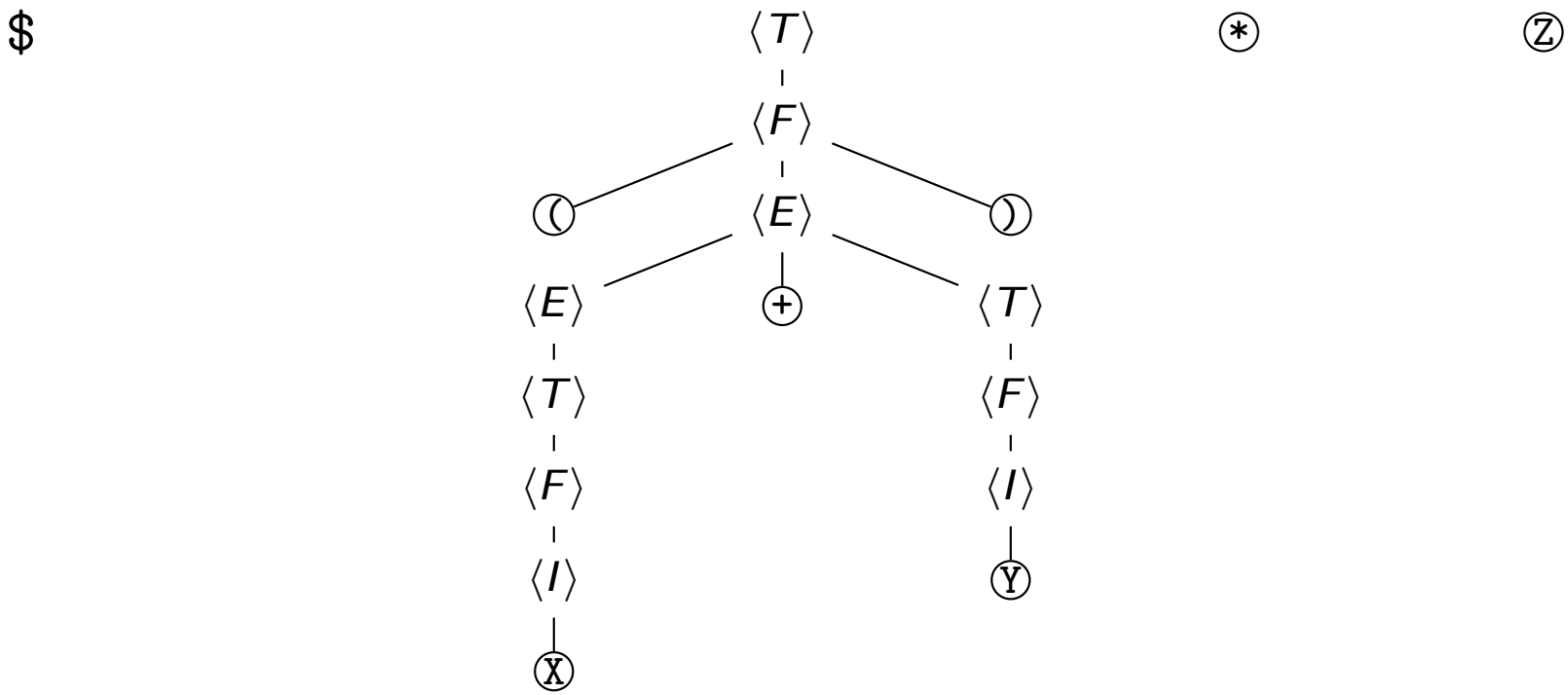


Example of Shift-Reduce Parsing — 8

$\langle I \rangle \rightarrow$ A | B | ... | Y | Z $\langle T \rangle \rightarrow \langle T \rangle * \langle F \rangle \mid \langle F \rangle$ (X+Y)*Z

$\langle E \rangle \rightarrow \langle E \rangle + \langle T \rangle \mid \langle T \rangle$ $\langle F \rangle \rightarrow (\langle E \rangle) \mid \langle I \rangle$

- Now Z is reduced using $\langle I \rangle \rightarrow$ Z, and then $\langle F \rangle \rightarrow \langle I \rangle$.
- Next, the remaining trees are reduced using $\langle T \rangle \rightarrow \langle T \rangle * \langle F \rangle$.
- Finally, reduce with $\langle E \rangle \rightarrow \langle T \rangle$.



Example of Shift-Reduce Parsing — 8

$\langle I \rangle \rightarrow$ `A` | `B` | ... | `Y` | `Z`

$\langle E \rangle \rightarrow \langle E \rangle + \langle T \rangle \mid \langle T \rangle$

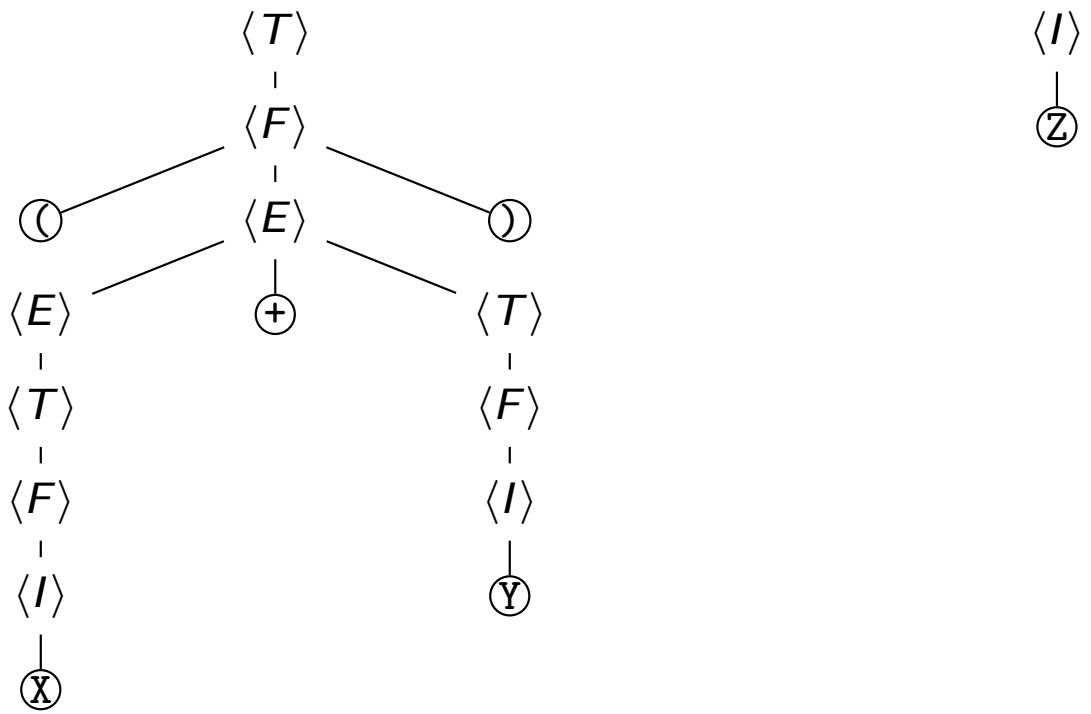
$\langle T \rangle \rightarrow \langle T \rangle * \langle F \rangle \mid \langle F \rangle$

$\langle F \rangle \rightarrow (\langle E \rangle) \mid \langle I \rangle$

`(X+Y)*Z`

- Now `Z` is reduced using $\langle I \rangle \rightarrow$ `Z`, and then $\langle F \rangle \rightarrow \langle I \rangle$.
- Next, the remaining trees are reduced using $\langle T \rangle \rightarrow \langle T \rangle * \langle F \rangle$.
- Finally, reduce with $\langle E \rangle \rightarrow \langle T \rangle$.

$



Example of Shift-Reduce Parsing — 8

$\langle I \rangle \rightarrow$ A | B | ... | Y | Z

$\langle E \rangle \rightarrow \langle E \rangle + \langle T \rangle \mid \langle T \rangle$

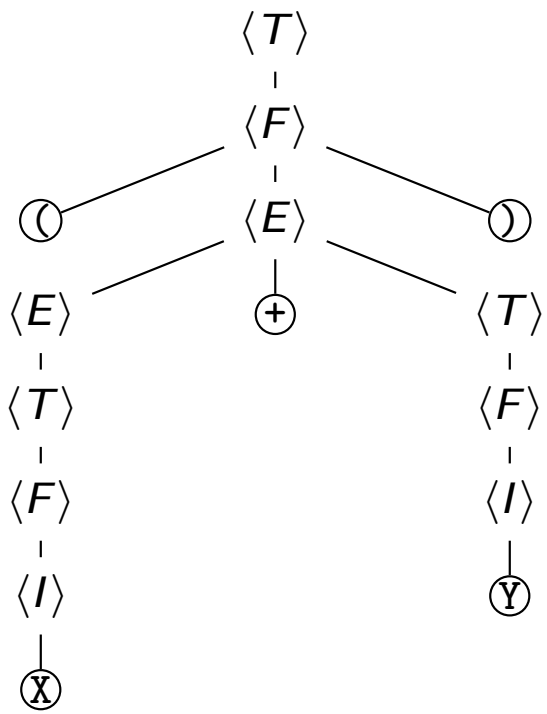
$\langle T \rangle \rightarrow \langle T \rangle * \langle F \rangle \mid \langle F \rangle$

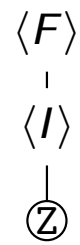
$\langle F \rangle \rightarrow (\langle E \rangle) \mid \langle I \rangle$

(X+Y)*Z

- Now Z is reduced using $\langle I \rangle \rightarrow$ Z, and then $\langle F \rangle \rightarrow \langle I \rangle$.
- Next, the remaining trees are reduced using $\langle T \rangle \rightarrow \langle T \rangle * \langle F \rangle$.
- Finally, reduce with $\langle E \rangle \rightarrow \langle T \rangle$.

$





Example of Shift-Reduce Parsing — 8

$\langle I \rangle \rightarrow$ A | B | ... | Y | Z

$\langle E \rangle \rightarrow \langle E \rangle + \langle T \rangle \mid \langle T \rangle$

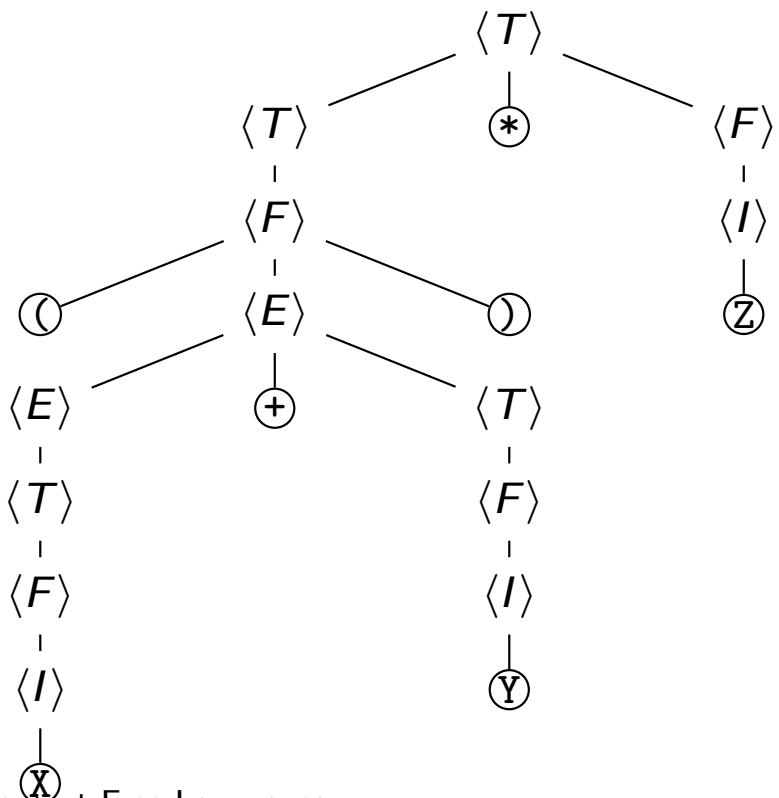
$\langle T \rangle \rightarrow \langle T \rangle * \langle F \rangle \mid \langle F \rangle$

$\langle F \rangle \rightarrow (\langle E \rangle) \mid \langle I \rangle$

(X+Y)*Z

- Now Z is reduced using $\langle I \rangle \rightarrow$ Z, and then $\langle F \rangle \rightarrow \langle I \rangle$.
- Next, the remaining trees are reduced using $\langle T \rangle \rightarrow \langle T \rangle * \langle F \rangle$.
- Finally, reduce with $\langle E \rangle \rightarrow \langle T \rangle$.

$



Example of Shift-Reduce Parsing — 8

$\langle I \rangle \rightarrow$ A | B | ... | Y | Z

$\langle E \rangle \rightarrow \langle E \rangle + \langle T \rangle \mid \langle T \rangle$

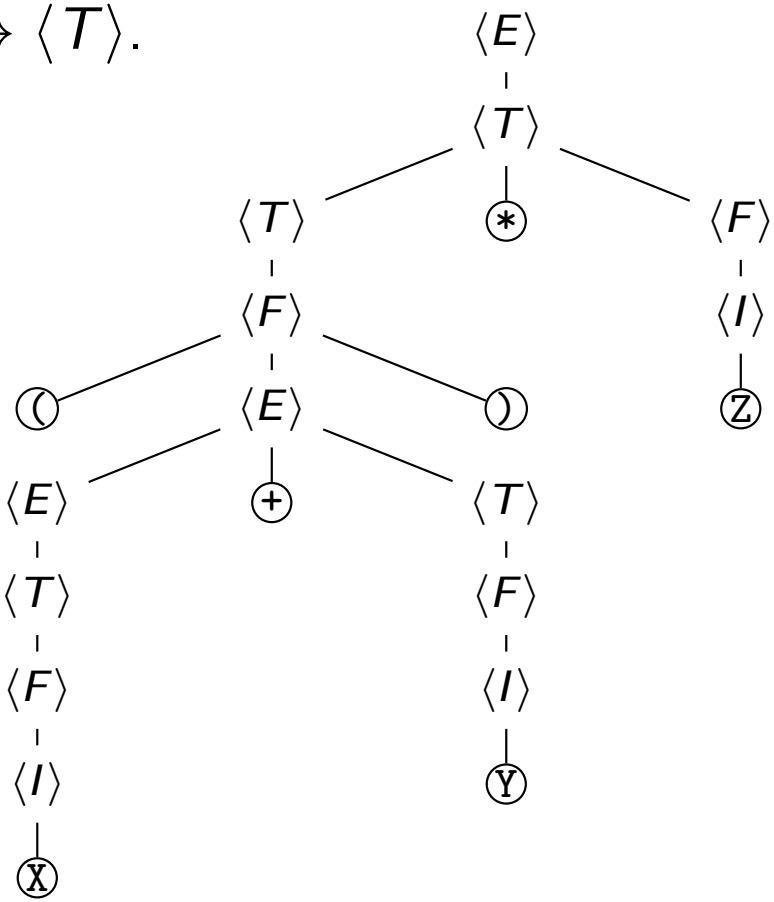
$\langle T \rangle \rightarrow \langle T \rangle * \langle F \rangle \mid \langle F \rangle$

$\langle F \rangle \rightarrow (\langle E \rangle) \mid \langle I \rangle$

(X+Y)*Z

- Now Z is reduced using $\langle I \rangle \rightarrow$ Z, and then $\langle F \rangle \rightarrow \langle I \rangle$.
- Next, the remaining trees are reduced using $\langle T \rangle \rightarrow \langle T \rangle * \langle F \rangle$.
- Finally, reduce with $\langle E \rangle \rightarrow \langle T \rangle$.

$



Important Points about Shift-Reduce Parsing

- The system must determine which step to take next.
- This information is encoded in a *state table*.
- It is difficult to construct such a table by hand,
- but there exist compiler-compilers which do it automatically, provided the grammar is in the proper format.
- Common tools for building such parsers:
  - Yacc
  - Bison
- Generally, these tools are limited to unambiguous grammars.
- However, extensions for ambiguous grammars have been developed:
  - *GLR*, or *Generalized LR* parsing.
  - These tools may generate slower parsers for unambiguous grammars, and so should be used only if needed.

Computer Language vs. Natural Language

- Natural Language (e.g., English, Swedish) poses a different set of problems than do programming languages.
- Computer languages are designed to be easy to parse.
  - At least, most modern ones are.
- Natural languages have evolved without any consideration for automated parsing, and are inherently ambiguous.

  Example: I saw the girl on the hill with a telescope.

  Example: Time flies like an arrow.
- If a program contains a syntax error, it is flagged and must be fixed before the program can be run.
- In natural language, an interface which behaves in that manner would be rejected by almost all users.
  - A natural language program must be as flexible as possible and involve error correction.

Computer Language vs. Natural Language — 2

- Understanding of NL depends upon a detailed knowledge of a "commonsense" context, often more than grammar.
  Understandable but ungrammatical: Her not here.
  Grammatical but not understandable:
  Colorless green ideas sleep furiously.
  Ungrammatical only with deeper analysis:
  Sue hurt himself. The dog barks chocolate.
- It is debatable what constitutes a legal sentence.
  Example: It's me.
  Example: Its me.
  Example: ...to boldly go where no one has gone before.
  Example: That be a good idea.
- Simple errors often affect neither understandability nor ambiguity.
  Example: The police was there.
  Example: One of my friends were there.

Questions to Be Considered

- In this short presentation, it is not possible to address all of these issues surrounding natural language (NL).
- Only two questions will be addressed briefly.

Question: Can a typical NL be modelled with a CFG?

Question: To the extent that the answer to the first question is "yes", how can it be parsed?

CFGs and Natural Language

Question: Can a typical NL be modelled with a CFG?

Answer: Actually, this question received a lot of attention from linguists during 1960's – 1980s.

- Much more attention than it deserves.
- See Chapter 16: "Footloose and Context Free" in *The Great Eskimo Vocabulary Hoax and other Irreverent Essays on the Study of Language* by Geoffrey K. Pullum, U. Chicago Press, 1991.

Key Points: (not from the above paper)

- It does not really matter whether NLs are context free or not.
- As with programming languages, if a CFG is to be used in parsing, the solution is to *overgenerate* and then filter out unwanted strings by other means.
- In formalisms for NL which use CFGs, that CFG is called a *context-free backbone* for the formalism.
- Some formalisms use a CFG backbone, while others handle parsing in other ways.

Managing Ambiguity in Parsing

- There are two flavors of NL systems.

Wide coverage: Handle "general" sentences in the NL, regardless of topic.

Narrow coverage: Handle sentences within a particular domain.

Example: A natural-language interface to a database or a system.

- Wide-coverage parsers must always handle ambiguity.
- Narrow-coverage parsers may need to deal with it, depending upon the domain and breadth of coverage within that domain.
- In general, the need to handle ambiguity is important.

Managing Ambiguity in Parsing — 2

- There are two flavors of such parsers which are in use:
  - Chart parsers: This is the traditional approach, developed during the 1970's
    - Worst-case complexity n^3.
  - Extensions to LR parsers: *Super LR* or *SLR* parsing has recently been developed in an attempt to extend LR parsing to ambiguous grammars, with a particular eye towards natural language.
    - They also have worst-case complexity n^3, but may fall back to n on unambiguous fragments of the grammar.
- Both are effective.

Caveats regarding natural language processing:

- Parsing is only the tip of the iceberg.
- NL understanding is far more complex.
- Modern parsers often employ statistical techniques, and learn the grammar from studying a large corpus of examples.