# Top-Down Parsing

## Lecture 7
## Section 4.4

Rokan Uddin Faruqui

Associate Professor
Dept of Computer Science and Engineering
University of Chittaong, Bangladesh
Email: *rokan@cu.ac.bd*

1. Left Factoring
2. Nullability
3. The FIRST Function
4. The FOLLOW Function
5. The Parse Table Entries
6. The Parse Table
7. Predictive Parsing
8. Assignment

**E —>TE’**
**E’—>+TE’ | epsilon**

**E —> E + T | T**

**S —> AA’**
**A’ —> b | c**

# Outline



# Left Factoring

- A problem occurs when two productions for the same nonterminal begin with the same token.
- We cannot decide which production to use.
- This is not necessarily a problem since we could process the part they have in common, then make a decision based on what follows.

# Left Factoring

- Consider the grammar

$$A \to \alpha\beta \mid \alpha\gamma$$

- We use left factorization to transform it into the form

$$A \to \alpha A'$$
$$A' \to \beta \mid \gamma.$$

- Now we can apply the productions immediately and unambiguously.

# Example

Example (Left Factoring)

- In the earlier example, we had the productions

$$C \rightarrow \textbf{id == num} \mid \textbf{id != num} \mid \textbf{id < num}$$

- To perform left factoring, introduce a nonterminal $C'$:

$$C \rightarrow \textbf{id}\ C'$$
$$C' \rightarrow \textbf{== num} \mid \textbf{!= num} \mid \textbf{< num}$$

# Example

Example (Left Factoring)

- Consider the grammar of `if` statements.

$$S \rightarrow \texttt{if}\ C\ \texttt{then}\ S\ \texttt{else}\ S$$
$$\mid\ \texttt{if}\ C\ \texttt{then}\ S$$

- We rewrite it as

$$S \rightarrow \texttt{if}\ C\ \texttt{then}\ S\ S'$$
$$S' \rightarrow \texttt{else}\ S \mid \varepsilon$$

# Outline



# Table-Driven LL Parsing

- To build the parsing table, we need three concepts:
  - Nullability
  - The FIRST function
  - The FOLLOW function

# Nullability

E' —> +TE'
E' —> e

Definition (Nullable)

A nonterminal $A$ is nullable if

$$A \stackrel{*}{\Rightarrow} \varepsilon.$$

# Nullability

B => e
C => B => e
C =>* e
A => BC => C => e
A =>* e

Example (Nullability)

- In the following grammar, $A$, $B$, and $C$ are nullable.

$$A \rightarrow B\ C$$
$$B \rightarrow A \mid a\ C \mid \varepsilon$$
$$C \rightarrow a\ B \mid B\ A \mid B$$

# Nullability

- In other words, $A$ is nullable if there is a production

$$A \to \varepsilon,$$

or there is a production

$$A \to B_1\ B_2\ \ldots\ B_n,$$

where $B_1$, $B_2$, ..., $B_n$ are nullable.

# Example

Example (Nullability)

- In the grammar

**E —> E + E — ambigous**

**E —> E + T | T**

**T—> T *F| F**

**F —> (E) |id — unambigous but left reciursive**

$$E \to T\ E'$$

$$E' \to +\ T\ E' \mid \varepsilon$$

$$T \to F\ T'$$

$$T' \to *\ F\ T' \mid \varepsilon$$

$$F \to (\ E\ ) \mid \mathbf{id}$$

**E—> TE'|**

- $E'$ and $T'$ are nullable.
- $E$, $T$, and $F$ are not nullable.

# Example

Example (Nullability)

| Nonterminal | Nullable |
|---|---|
| $E$ | No |
| $E'$ | Yes |
| $T$ | No |
| $T'$ | Yes |
| $F$ | No |

# Outline



# The FIRST Function

$$\text{id1} + (\text{id2} * \text{id3})$$

Definition (The FIRST Function)

For any string of grammar symbols $\alpha$, FIRST($\alpha$) is the set of all tokens that may appear as the first symbol in a replacement string of $\alpha$.

# Computing FIRST($X$)

Computing FIRST($X$)

- Let $X$ be a grammar symbol. FIRST($X$) is computed as follows.
  - If $X$ is a token, then FIRST($X$) = $\{X\}$.
  - If $X$ is a nonterminal, then for every production

$$X \to Y_1\ Y_2\ \ldots\ Y_n,$$

    - FIRST($Y_1$) $\subseteq$ FIRST($X$).
    - Furthermore, if $Y_1$, $Y_2$, ..., $Y_k$ are nullable, then

$$\text{FIRST}(Y_{k+1}) \subseteq \text{FIRST}(X).$$

# FIRST

- We are concerned with FIRST($X$) only for the nonterminals of the grammar.
- FIRST($X$) for terminals is trivial.
- According to the definition, to determine FIRST($A$), we must inspect all productions that have $A$ *as their head*.

# Example

FIRST(E) = FIRST(T) = FIRST(F) = {(, id}

FIRST(E') = {+, e } —nullable

FIRST(T') = {*, e} — nullable

Example (FIRST)

- Let the grammar be

A —> BCD

FIRST(A) = FIRST(D) if both B and C are nullable

$$E \rightarrow T\ E'$$

$$E' \rightarrow +\ T\ E' \mid \varepsilon$$

$$T \rightarrow F\ T'$$

$$T' \rightarrow *\ F\ T' \mid \varepsilon$$

$$F \rightarrow (\ E\ ) \mid \mathbf{id}$$

FOLLOW(T) = { + , $, )}

FOLLOW(E') = { $, )}

FOLLOW(T')={ }

FOLLOW(F)={ *,+,$,)}

# Example

Example (FIRST)

- Find FIRST($E$).
  - $E$ occurs as the head in only one production

$$E \to T\ E'$$

  - Therefore,

$$\text{FIRST}(T) \subseteq \text{FIRST}(E).$$

  - Furthermore, $T$ is not nullable.
  - Therefore,

$$\text{FIRST}(E) = \text{FIRST}(T).$$

  - We have yet to determine FIRST($T$).

# Example

Example (FIRST)

- Find FIRST($E$).
  - $E$ occurs as the head in only one production

$$E \rightarrow T\ E'$$

  - Therefore,

$$\text{FIRST}(T) \subseteq \text{FIRST}(E).$$

  - Furthermore, $T$ is not nullable.
  - Therefore,

$$\text{FIRST}(E) = \text{FIRST}(T).$$

  - We have yet to determine FIRST($T$).

# Example

Example (FIRST)

- Find FIRST($E$).
  - $E$ occurs as the head in only one production

$$E \to T\ E'$$

  - Therefore,

$$\text{FIRST}(T) \subseteq \text{FIRST}(E).$$

  - Furthermore, $T$ is not nullable.
  - Therefore,

$$\text{FIRST}(E) = \text{FIRST}(T).$$

  - We have yet to determine FIRST($T$).

# Example

Example (FIRST)

- Find FIRST($E$).
  - $E$ occurs as the head in only one production

$$E \to T\ E'$$

  - Therefore,

$$\text{FIRST}(T) \subseteq \text{FIRST}(E).$$

  - Furthermore, $T$ is not nullable.
  - Therefore,

$$\text{FIRST}(E) = \text{FIRST}(T).$$

  - We have yet to determine FIRST($T$).

# Example

Example (FIRST)

- Find FIRST($E$).
  - $E$ occurs as the head in only one production

$$E \to T\ E'$$

  - Therefore,

$$\text{FIRST}(T) \subseteq \text{FIRST}(E).$$

  - Furthermore, $T$ is not nullable.
  - Therefore,

$$\text{FIRST}(E) = \text{FIRST}(T).$$

  - We have yet to determine FIRST($T$).

# Example

Example (FIRST)

- Find FIRST($T$).
  - $T$ occurs as the head in only one production

$$T \rightarrow F\ T'$$

  - Therefore,

$$\text{FIRST}(F) \subseteq \text{FIRST}(T).$$

  - Furthermore, $F$ is not nullable.
  - Therefore,

$$\text{FIRST}(T) = \text{FIRST}(F).$$

  - We have yet to determine FIRST($F$).

# Example

Example (FIRST)

- Find FIRST($T$).
  - $T$ occurs as the head in only one production

$$T \rightarrow F\ T'$$

  - Therefore,

$$\text{FIRST}(F) \subseteq \text{FIRST}(T).$$

  - Furthermore, $F$ is not nullable.
  - Therefore,

$$\text{FIRST}(T) = \text{FIRST}(F).$$

  - We have yet to determine FIRST($F$).

# Example

Example (FIRST)

- Find FIRST($T$).
  - $T$ occurs as the head in only one production

$$T \rightarrow F\ T'$$

  - Therefore,

$$\text{FIRST}(F) \subseteq \text{FIRST}(T).$$

  - Furthermore, $F$ is not nullable.
  - Therefore,

$$\text{FIRST}(T) = \text{FIRST}(F).$$

  - We have yet to determine FIRST($F$).

# Example

Example (FIRST)

- Find FIRST($T$).
  - $T$ occurs as the head in only one production

$$T \to F\ T'$$

  - Therefore,

$$\text{FIRST}(F) \subseteq \text{FIRST}(T).$$

  - Furthermore, $F$ is not nullable.
  - Therefore,

$$\text{FIRST}(T) = \text{FIRST}(F).$$

  - We have yet to determine FIRST($F$).

# Example

Example (FIRST)

- Find FIRST($T$).
  - $T$ occurs as the head in only one production

$$T \to F\ T'$$

  - Therefore,

$$\text{FIRST}(F) \subseteq \text{FIRST}(T).$$

  - Furthermore, $F$ is not nullable.
  - Therefore,

$$\text{FIRST}(T) = \text{FIRST}(F).$$

  - We have yet to determine FIRST($F$).

# Example

Example (FIRST)

- Find FIRST($F$).
  - FIRST($F$) = {(, **id**}.
- Therefore,
  - FIRST($E$) = {(, **id**}.
  - FIRST($T$) = {(, **id**}.

# Example

Example (FIRST)

- Find FIRST($F$).
  - FIRST($F$) = {(, **id**}.
- Therefore,
  - FIRST($E$) = {(, **id**}.
  - FIRST($T$) = {(, **id**}.

# Example

Example (FIRST)

- Find FIRST($F$).
  - FIRST($F$) = {(, **id**}.
- Therefore,
  - FIRST($E$) = {(, **id**}.
  - FIRST($T$) = {(, **id**}.

# Example

Example (FIRST)

- Find FIRST($E'$).
  - FIRST($E'$) = {+}.
- Find FIRST($T'$).
  - FIRST($T'$) = {*}.

# Example

Example (FIRST)

- Find FIRST($E'$).
  - FIRST($E'$) = {+}.
- Find FIRST($T'$).
  - FIRST($T'$) = {*}.

# Example

Example (FIRST)

- Find FIRST($E'$).
  - FIRST($E'$) = {+}.
- Find FIRST($T'$).
  - FIRST($T'$) = {$*$}.

# Example

Example (FIRST)

- Find FIRST($E'$).
  - FIRST($E'$) = {+}.
- Find FIRST($T'$).
  - FIRST($T'$) = {*}.

# Example

Example (FIRST)

| Nonterminal | Nullable | FIRST |
|---|---|---|
| $E$ | No | $\{($, **id**$\}$ |
| $E'$ | Yes | $\{+\}$ |
| $T$ | No | $\{($, **id**$\}$ |
| $T'$ | Yes | $\{*\}$ |
| $F$ | No | $\{($, **id**$\}$ |

# Example

Example (FOLLOW)

- The grammar

$$R \to R \cup R \mid R\,R \mid R^* \mid (\ R\ ) \mid \mathbf{id}$$

generates all regular expressions over a set of identifiers. Find FIRST($X$) for each nonterminal $X$.

# Outline



# The FOLLOW Function

Definition (The FOLLOW Function)

For any string of grammar symbols $\alpha$, FOLLOW($\alpha$) is the set of all tokens that may follow $\alpha$ in a derivation.

# Computing FOLLOW($A$)

Computing FOLLOW($A$)

- Let $A$ be a nonterminal. FOLLOW($A$) is computed as follows.
  - If $A$ is the start symbol, then $\$ \in \text{FOLLOW}(A)$.
  - If $B \to \alpha A \beta$ is a production, then

$$\text{FIRST}(\beta) \subseteq \text{FOLLOW}(A).$$

  - If $B \to \alpha A$ is a production, or $B \to \alpha A \beta$ is a production and $\beta$ is nullable, then

$$\text{FOLLOW}(B) \subseteq \text{FOLLOW}(A).$$

# FOLLOW

- We are concerned about FOLLOW($A$) only for the nonterminals of the grammar.
- According to the definition, to determine FOLLOW($A$), we must inspect all productions that have $A$ *in their body*.

# Example

FOLLOW(E) = { $, )}

FOLLOW(T) = FIRST(E') U

FOLLOW(F) = {*, +, $, )}

FOLLOW(E')= FOLLOW(E)

FOLLOW(T') = FOLLOW(T)

Example (FOLLOW)

- Let the grammar be

**FOLLOW(T) = FIRST(E') U FOLLOW(E) = {+, $, )}**

$$E \rightarrow T\ E'$$
$$E' \rightarrow +\ T\ E' \mid \varepsilon$$
$$T \rightarrow F\ T'$$
$$T' \rightarrow *\ F\ T' \mid \varepsilon$$
$$F \rightarrow (\ E\ ) \mid \mathbf{id}$$

# Example

Example (FOLLOW)

- Find FOLLOW($E$).
  - $E$ is the start symbol, therefore

$$\$ \in \text{FOLLOW}(E).$$

  - $E$ occurs in the body of only one production.

$$F \rightarrow ( \ E \ )$$

  - Therefore

$$\text{FOLLOW}(E) = \{\$, )\}.$$

# Example

Example (FOLLOW)

- Find FOLLOW($E$).
  - $E$ is the start symbol, therefore

$$\$ \in \text{FOLLOW}(E).$$

  - $E$ occurs in the body of only one production.

$$F \rightarrow ( \ E \ )$$

  - Therefore

$$\text{FOLLOW}(E) = \{\$, )\}.$$

# Example

Example (FOLLOW)

- Find FOLLOW($E$).
  - $E$ is the start symbol, therefore

$$\$ \in \text{FOLLOW}(E).$$

  - $E$ occurs in the body of only one production.

$$F \rightarrow ( \ E \ )$$

  - Therefore

$$\text{FOLLOW}(E) = \{\$, )\}.$$

# Example

Example (FOLLOW)

- Find FOLLOW($E'$).
  - $E'$ occurs in the body of two productions.

$$E \to T\ E'$$
$$E' \to +\ T\ E'$$

  - Therefore,

$$\text{FOLLOW}(E') = \text{FOLLOW}(E) = \{\$, )\}.$$

# Example

Example (FOLLOW)

- Find FOLLOW($E'$).
  - $E'$ occurs in the body of two productions.

$$E \to T\ E'$$
$$E' \to +\ T\ E'$$

  - Therefore,

$$\text{FOLLOW}(E') = \text{FOLLOW}(E) = \{\$, )\}.$$

# Example

Example (FOLLOW)

- Find FOLLOW($T$).
  - $T$ occurs in the body of two productions.

$$E \to T\ E'$$
$$E' \to +\ T\ E'$$

  - Therefore, $\text{FOLLOW}(T) \supseteq \text{FIRST}(E') = \{+\}$.
  - However, $E'$ is nullable, therefore it also contains $\text{FOLLOW}(E) = \{\$, )\}$ and $\text{FOLLOW}(E') = \{\$, )\}$.
  - Therefore,

$$\text{FOLLOW}(T) = \{+, \$, )\}.$$

# Example

Example (FOLLOW)

- Find FOLLOW($T$).
  - $T$ occurs in the body of two productions.

$$E \to T\ E'$$
$$E' \to +\ T\ E'$$

  - Therefore, FOLLOW($T$) $\supseteq$ FIRST($E'$) = {+}.
  - However, $E'$ is nullable, therefore it also contains FOLLOW($E$) = {\$, )} and FOLLOW($E'$) = {\$, )}.
  - Therefore,

$$\text{FOLLOW}(T) = \{+, \$, )\}.$$

# Example

Example (FOLLOW)

- Find FOLLOW($T$).
  - $T$ occurs in the body of two productions.

$$E \to T\ E'$$
$$E' \to +\ T\ E'$$

  - Therefore, FOLLOW($T$) $\supseteq$ FIRST($E'$) = {+}.
  - However, $E'$ is nullable, therefore it also contains FOLLOW($E$) = {\$, )} and FOLLOW($E'$) = {\$, )}.
  - Therefore,

$$\text{FOLLOW}(T) = \{+, \$, )\}.$$

# Example

Example (FOLLOW)

- Find FOLLOW($T$).
  - $T$ occurs in the body of two productions.

$$E \to T\ E'$$
$$E' \to +\ T\ E'$$

  - Therefore, $\text{FOLLOW}(T) \supseteq \text{FIRST}(E') = \{+\}$.
  - However, $E'$ is nullable, therefore it also contains $\text{FOLLOW}(E) = \{\$, )\}$ and $\text{FOLLOW}(E') = \{\$, )\}$.
  - Therefore,

$$\text{FOLLOW}(T) = \{+, \$, )\}.$$

# Example

Example (FOLLOW)

- Find FOLLOW($T'$).
  - $T'$ occurs in the body of two productions.

$$T \to F\ T'$$
$$T' \to *\ F\ T'$$

  - Therefore,

$$\text{FOLLOW}(T') = \text{FOLLOW}(T) = \{+, \$, )\}.$$

# Example

Example (FOLLOW)

- Find FOLLOW($T'$).
  - $T'$ occurs in the body of two productions.

$$T \to F\ T'$$
$$T' \to *\ F\ T'$$

  - Therefore,

$$\text{FOLLOW}(T') = \text{FOLLOW}(T) = \{+, \$, )\}.$$

# Example

Example (FOLLOW)

- Find FOLLOW($F$).
  - $F$ occurs in the body of two productions.

$$T \to F\ T'$$
$$T' \to *\ F\ T'.$$

  - Therefore,
$$\text{FOLLOW}(F) \supseteq \text{FIRST}(T') = \{*\}.$$
  - However, $T'$ is nullable, therefore it also contains $\text{FOLLOW}(T) = \{+, \$, )\}$ and $\text{FOLLOW}(T') = \{+, \$, )\}$.
  - Therefore,
$$\text{FOLLOW}(F) = \{*, +, \$, )\}.$$

# Example

Example (FOLLOW)

- Find FOLLOW($F$).
  - $F$ occurs in the body of two productions.

$$T \to F\ T'$$
$$T' \to *\ F\ T'.$$

  - Therefore,

$$\text{FOLLOW}(F) \supseteq \text{FIRST}(T') = \{*\}.$$

  - However, $T'$ is nullable, therefore it also contains $\text{FOLLOW}(T) = \{+, \$, )\}$ and $\text{FOLLOW}(T') = \{+, \$, )\}$.
  - Therefore,

$$\text{FOLLOW}(F) = \{*, +, \$, )\}.$$

# Example

Example (FOLLOW)

- Find FOLLOW($F$).
  - $F$ occurs in the body of two productions.

$$T \to F\ T'$$
$$T' \to *\ F\ T'.$$

  - Therefore,

$$\text{FOLLOW}(F) \supseteq \text{FIRST}(T') = \{*\}.$$

  - However, $T'$ is nullable, therefore it also contains $\text{FOLLOW}(T) = \{+, \$, )\}$ and $\text{FOLLOW}(T') = \{+, \$, )\}$.
  - Therefore,

$$\text{FOLLOW}(F) = \{*, +, \$, )\}.$$

# Example

Example (FOLLOW)

- Find FOLLOW($F$).
  - $F$ occurs in the body of two productions.

$$T \to F\ T'$$
$$T' \to *\ F\ T'.$$

  - Therefore,

$$\text{FOLLOW}(F) \supseteq \text{FIRST}(T') = \{*\}.$$

  - However, $T'$ is nullable, therefore it also contains $\text{FOLLOW}(T) = \{+, \$, )\}$ and $\text{FOLLOW}(T') = \{+, \$, )\}$.
  - Therefore,

$$\text{FOLLOW}(F) = \{*, +, \$, )\}.$$

# Example

Example (FOLLOW)

| Nonterminal | Nullable | FIRST | FOLLOW |
|---|---|---|---|
| $E$ | No | $\{(, \mathbf{id}\}$ | $\{\$, )\}$ |
| $E'$ | Yes | $\{+\}$ | $\{\$, )\}$ |
| $T$ | No | $\{(, \mathbf{id}\}$ | $\{\$, ), +\}$ |
| $T'$ | Yes | $\{*\}$ | $\{\$, ), +\}$ |
| $F$ | No | $\{(, \mathbf{id}\}$ | $\{*, \$, ), +\}$ |

# Example

Example (FOLLOW)

- The grammar

$$S \to ( \ L \ ) \mid \mathbf{id}$$
$$L \to L \ , \ S \mid S$$

generates nested lists of identifiers. Find FOLLOW($X$) for each nonterminal $X$.

# Example

Example (FOLLOW)

- The grammar

$$R \to R \cup R \mid R\,R \mid R^* \mid (\ R\ ) \mid \mathbf{id}$$

generates all regular expressions over a set of identifiers. Find FOLLOW($X$) for each nonterminal $X$.

# Outline



# Nullability and FIRST for Strings

**LL(1) Parsing / Table Driven LL Parsing**

**L = read input left to right**

**L = leftmost derivation**

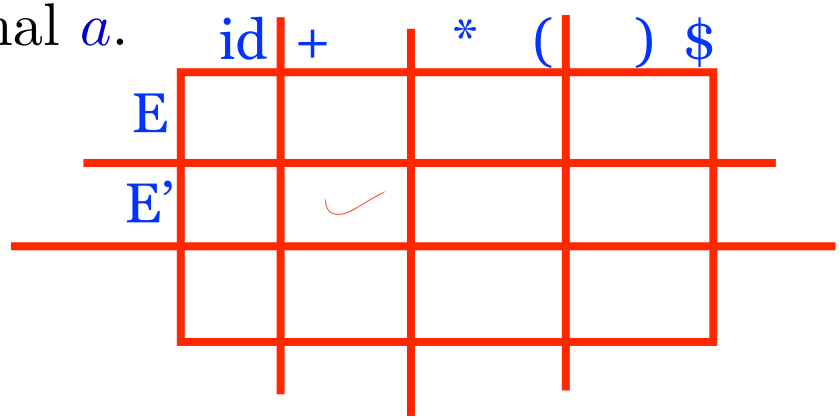
M[E',+]

**5 x 6**

- The parse table has
  - One row for each nonterminal, E, F, T, E', T'
  - One column for each terminal and \$. id, +, *, (, ), \$
- \$ is the end-of-file marker.
- Each cell in the table represents a combination $(A, a)$ of a nonterminal $A$ and a terminal $a$.

**M[E,id]**

**M[E', +]**




# Nullability and FIRST for Strings

E—>TE'|E'
A—> A1A2A3.....A9
FIRST(A)=

- We need to extend the definitions of nullable and FIRST to *strings* of grammar symbols.
  - $\varepsilon$ is nullable.
  - Let $\alpha = X_1 X_2 \ldots X_n$ be a string of grammar symbols. Then $\alpha$ is nullable if each $X_i$ is nullable.

# Nullability and FIRST for Strings

- By definition, $\text{FIRST}(\varepsilon) = \varnothing$.
- For the string $\alpha = X_1 X_2 \ldots X_n$,
  - Let $k$ be the largest integer for which $X_1$, $X_2$, ..., and $X_k$ are nullable.
  - Then
$$\text{FIRST}(\alpha) = \text{FIRST}(X_1) \cup \ldots \cup \text{FIRST}(X_{k+1}).$$
- We will not need FOLLOW for strings.

# Parse Table Construction : Algorithm

$$E \to T\,E'$$
$$E' \to +\,T\,E' \mid \varepsilon$$
$$T \to F\,T'$$
$$T' \to *\,F\,T' \mid \varepsilon$$
$$F \to (\,E\,) \mid id$$

**INPUT** : Grammar $G$.

**OUTPUT** : Parsing table $M$

**METHOD** :

**FIRST(TE')=FIRST(T)={(,id}**

**M[E,(] = E → T E′**

**M[E, id] = E → T E′**

- Each table entry is a production $A \to \alpha$.
- Rules for entering $A \to \alpha$ in the table:
  - For every rule $A \to \alpha$ and for every $a \in \text{FIRST}(\alpha)$, write $A \to \alpha$ in cell $(A, a)$.
  - For every rule $A \to \alpha$, where $\alpha$ is nullable, and for every $a \in \text{FOLLOW}(A)$, write $A \to \alpha$ in cell $(A, a)$.
- Write "error" in all cells that do not contain a production.

# Parse Table Entries

- The interpretation of $A \to \alpha$ in cell $(A, a)$ is that if $A$ is on top of the stack and we encounter $a$ in the input, then we should replace $A$ by $\alpha$ on the stack.
- Push the symbols of $\alpha$ onto the stack in *reverse order*, from right to left, so that the first (leftmost) symbol is on the top of the stack.

# Example

Input: G
Output: M

1. E -->TE'

A —> alpha, FIRST(alpha) = FIRST(TE')= FIRST(T) = {(, id}

M[E, (] = E—>TE'

M[E, id] = E—>TE'

3. E' —> e

FOLLOW(E') = {$, ) }

M[E', $] = E' —> e

M[E', )] = E' —> e

Example (Parse Table)

- Let the grammar be

4. T —>FT'

FIRST(FT') =FIRST(F) = {(,id}

M[T, (] = T —>FT'

M[T, id] = T—>FT'

2. E' —> +TE'

FIRST(+TE') = {+}

M[E', +] = E' —> +TE'

$$E \rightarrow T\ E'$$

$$E' \rightarrow +\ T\ E' \mid \varepsilon$$

$$T \rightarrow F\ T'$$

$$T' \rightarrow *\ F\ T' \mid \varepsilon$$

$$F \rightarrow (\ E\ ) \mid \mathbf{id}$$

F—> (E)

F —> id

# Example

Example (Parse Table)

- Recall

| Nonterminal | Nullable | FIRST | FOLLOW |
|---|---|---|---|
| $E$ | No | $\{(, \mathbf{id}\}$ | $\{\$, )\}$ |
| $E'$ | Yes | $\{+\}$ | $\{\$, )\}$ |
| $T$ | No | $\{(, \mathbf{id}\}$ | $\{\$, ), +\}$ |
| $T'$ | Yes | $\{*\}$ | $\{\$, ), +\}$ |
| $F$ | No | $\{(, \mathbf{id}\}$ | $\{*, \$, ), +\}$ |

# Outline



# Example

Example (Parse Table)

- Consider the production $E \to T\ E'$.
  - $\text{FIRST}(T\ E') = \text{FIRST}(T) = \{(, \textbf{id}\}$.
  - Therefore, enter $E \to T\ E'$ in cells $(E, ($ and $(E, \textbf{id})$.
  - $T\ E'$ is not nullable.

# Example

Example (Parse Table)

- Consider the production $E' \to \varepsilon$.
- $\text{FIRST}(\varepsilon) = \varnothing$.
- $\varepsilon$ is nullable and $\text{FOLLOW}(E') = \{\$, )\}$.
- Therefore, enter $E' \to \varepsilon$ in cells $(E', \$)$ and $(E', )$.

# Example

E—> E+ E

Example (Parse Table)

- Handle the other productions similarly.
- In which cells do we enter $E' \rightarrow + \; T \; E'$?
- In which cells do we enter $T \rightarrow F \; T'$?
- In which cells do we enter $T' \rightarrow * \; F \; T'$?
- In which cells do we enter $T' \rightarrow \varepsilon$?
- In which cells do we enter $F \rightarrow ( \; E \; )$?
- In which cells do we enter $F \rightarrow \mathbf{id}$?

# Example

E' —> +TE'
A —> alpha
FIRST(alpha)
T —> FT
FIRST(+TE') = {+}
M[E', +] = E' —>TE'
FIRST(FT) = FIRST(F) = { id, ( }
T' —> *FT'
FISRT(*FT') = { * }
M[T, id] = T—>FT'
M[T, (] = T —> FT'

**M[E, (] = E → T E′**
**M[E, id] = E → T E′**

Example (Parse Table)

| | + | * | ( | ) | id | $ |
|---|---|---|---|---|---|---|
| $E$ | | | $E \to T\ E'$ | | $E \to T\ E'$ | |
| $E'$ | $E' \to +\ T\ E'$ | | | $E' \to \varepsilon$ | | $E' \to \varepsilon$ |
| $T$ | | | $T \to F\ T'$ | | $T \to F\ T'$ | |
| $T'$ | $T' \to \varepsilon$ | $T' \to *\ F\ T'$ | | $T' \to \varepsilon$ | | $T' \to \varepsilon$ |
| $F$ | | | $F \to (\ E\ )$ | | $F \to \mathbf{id}$ | |

**E′ → ε**

Is it LL(1)?

T' —>e
FOLLOW(T') = { +, ),

**FOLLOW(E') = { $, ) }**
**M[E', $] = E′ → ε**
**M[E', )] = E′ → ε**

# Outline



# Predictive Parsing

Definition (LL(1) Grammar)

An LL(1) grammar is a grammar whose predictive parse table does not contain any multiple entries.

- A multiple entry would indicate that we could not decide which production to apply.

# Predictive Parsing

E —> E + T | T
T —> T * F | F
F —> (E) | id

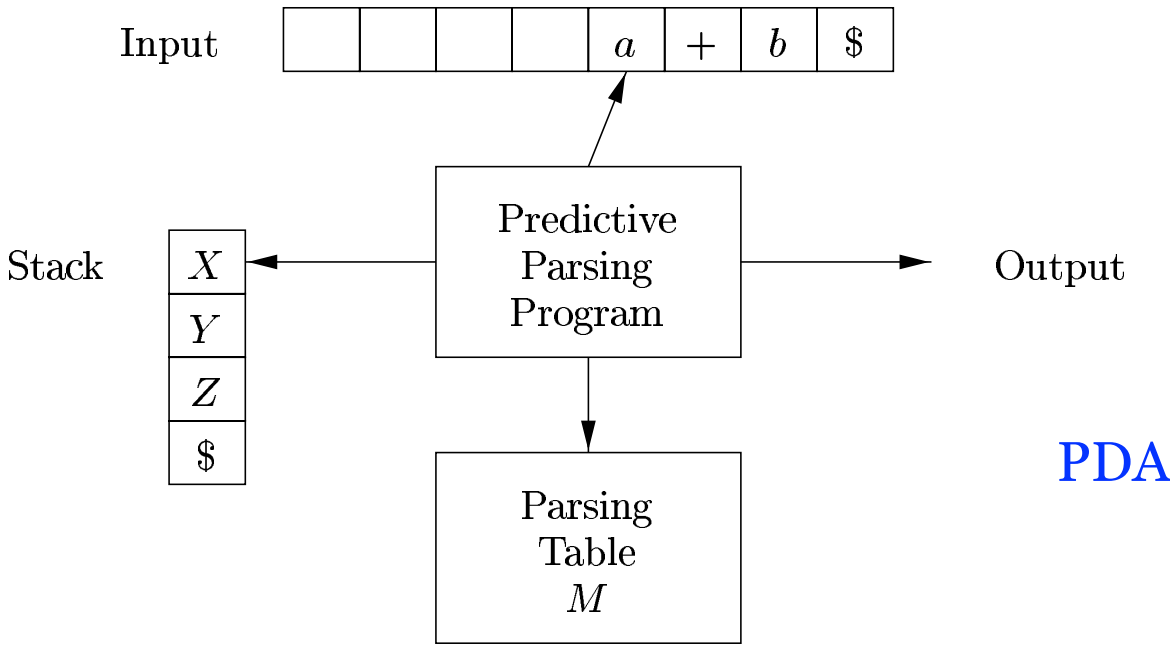
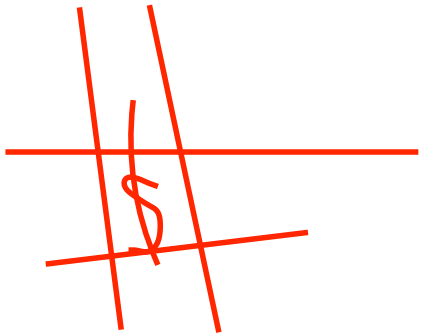


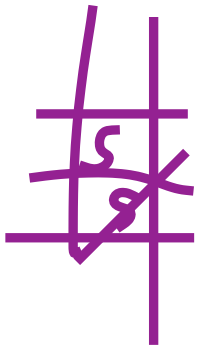

Figure: Model of a table-driven predictive parsing

# Predictive Parsing Algorithm

- The predictive parsing algorithm uses
  - The parse table,
  - An input buffer containing a sequence of tokens,
  - A stack of grammar symbols.

# Predictive Parsing Algorithm



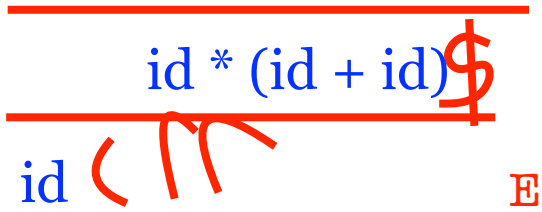


The Parsing Algorithm

- Initialize the input buffer to the input followed by \$.
- Initialize the stack to \$ and $S$, with $S$ on the top.

**E\$** **id+id*id\$**

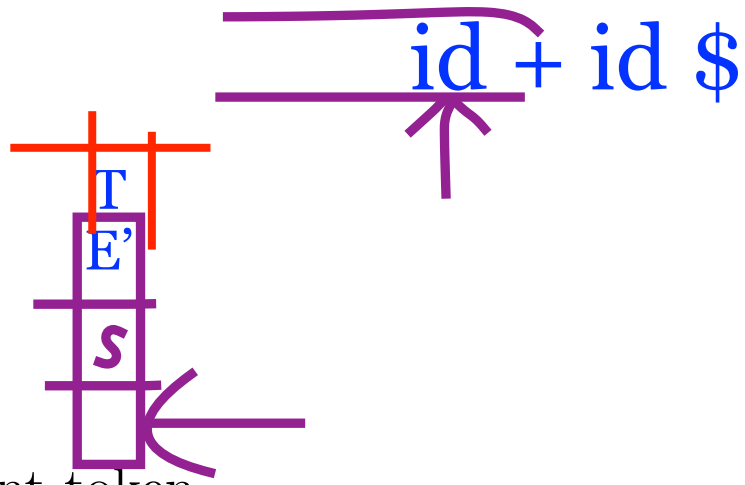
# Predictive Parsing Algorithm




The Parsing Algorithm

- Consider the top stack symbol $X$.
- There are three possibilities.
  - $X$ is a terminal.
  - $X$ is a nonterminal.
  - $X$ is \$.

# Predictive Parsing Algorithm




The Parsing Algorithm

- If $X$ is a terminal, then
  - If $X$ matches the current token,
    - Pop $X$ from the stack.
    - Advance to the next token.
  - Otherwise, it is an error.

# Predictive Parsing Algorithm

The Parsing Algorithm

- If $X$ is a nonterminal, then
  - Use $X$ together with the current token to get the entry from the parse table.
  - If the entry is a production,
    - Pop $X$ from the stack.
    - Push the symbols on the right-hand side of the production, from right to left, onto the stack.
  - Otherwise, it is an error.

# Predictive Parsing Algorithm

The Parsing Algorithm

- If $X$ is \$, then
  - If the current token is also \$,
    - Accept the input.
  - Otherwise, it is an error.

# Example

| Stack | Input |
|---|---|
| E$ | ( id + id ) * id $ |
| TE'$ | ( id + id ) * id $ |
| FT'E'$ | ( id + id ) * id $ |
| (id)T'E'$ | id + id ) * id $ |
| id)T'E'$ | |

M[E,(]

Example (Predictive Parsing)

- Parse the string **(id + id)*id**.

| Stack | Input |
|---|---|
| $E | (id + id) * id$ |
| $E'T | (id+id) * id$ |
| $E'T'F | (id+id) * id$ |
| $E'T')E( | (id+id) * id$ |
| $E'T')E | id+id)*id$ |

(
E
)
T'
E'
$

X = (
Input = (
M[T, (] = E —> TE'

# Example

M[E,(] = E—>TE'

(id + id) * id$

input: (
stack: E

$E'T

$E

Example (Predictive Parsing)

[E, (] = E —> TE'

| Stack | Input |
|---|---|
| $ $E$ | ( id + id ) * id $ |
| $ $E'\ T$ | ( id + id ) * id $ |
| $ $E'\ T'\ F$ | ( id + id ) * id $ |
| $ $E'\ T'\ )\ E\ ($ | ( id + id ) * id $ |
| $ $E'\ T'\ )\ E$ | id + id ) * id $ |
| $ $E'\ T'\ )\ E'\ T$ | id + id ) * id $ |
| $ $E'\ T'\ )\ E'\ T'\ F$ | id + id ) * id $ |
| $ $E'\ T'\ )\ E'\ T'$ id | id + id ) * id $ |
| $ $E'\ T'\ )\ E'\ T'$ | + id ) * id $ |
| $ $E'\ T'\ )\ E'$ | + id ) * id $ |
| $ $E'\ T'\ )\ E'\ T$ + | + id ) * id $ |
| $ $E'\ T'\ )\ E'\ T$ | id ) * id $ |

# Example

Example (Predictive Parsing)

| Stack | Input |
|---|---|
| \$ $E$ | ( **id** + **id** ) $*$ **id** \$ |
| \$ $E'$ $T$ | ( **id** + **id** ) $*$ **id** \$ |
| \$ $E'$ $T'$ $F$ | ( id + id ) $*$ id \$ |
| \$ $E'$ $T'$ ) $E$ ( | ( id + id ) $*$ id \$ |
| \$ $E'$ $T'$ ) $E$ | id + id ) $*$ id \$ |
| \$ $E'$ $T'$ ) $E'$ $T$ | id + id ) $*$ id \$ |
| \$ $E'$ $T'$ ) $E'$ $T'$ $F$ | id + id ) $*$ id \$ |
| \$ $E'$ $T'$ ) $E'$ $T'$ id | id + id ) $*$ id \$ |
| \$ $E'$ $T'$ ) $E'$ $T'$ | + id ) $*$ id \$ |
| \$ $E'$ $T'$ ) $E'$ | + id ) $*$ id \$ |
| \$ $E'$ $T'$ ) $E'$ $T$ + | + id ) $*$ id \$ |
| \$ $E'$ $T'$ ) $E'$ $T$ | id ) $*$ id \$ |

# Example

M[F,(] = F –> (E)

Example (Predictive Parsing)

| Stack | Input |
|---|---|
| \$ $E$ | ( **id** + **id** ) ∗ **id** \$ |
| \$ $E'$ $T$ | ( **id** + **id** ) ∗ **id** \$ |
| \$ $E'$ $T'$ $F$ | ( **id** + **id** ) ∗ **id** \$ |
| \$ $E'$ $T'$ ) $E$ ( | ( id + id ) ∗ id \$ |
| \$ $E'$ $T'$ ) $E$ | id + id ) ∗ id \$ |
| \$ $E'$ $T'$ ) $E'$ $T$ | id + id ) ∗ id \$ |
| \$ $E'$ $T'$ ) $E'$ $T'$ $F$ | id + id ) ∗ id \$ |
| \$ $E'$ $T'$ ) $E'$ $T'$ id | id + id ) ∗ id \$ |
| \$ $E'$ $T'$ ) $E'$ $T'$ | + id ) ∗ id \$ |
| \$ $E'$ $T'$ ) $E'$ | + id ) ∗ id \$ |
| \$ $E'$ $T'$ ) $E'$ $T$ + | + id ) ∗ id \$ |
| \$ $E'$ $T'$ ) $E'$ $T$ | id ) ∗ id \$ |

# Example

F -> (E)

Example (Predictive Parsing)

| Stack | Input |
|---|---|
| \$ $E$ | ( id + id ) * id \$ |
| \$ $E'$ $T$ | ( id + id ) * id \$ |
| \$ $E'$ $T'$ $F$ | ( id + id ) * id \$ |
| \$ $E'$ $T'$ ) $E$ ( | ( id + id ) * id \$ |
| \$ $E'$ $T'$ ) $E$ | id + id ) * id \$ |
| \$ $E'$ $T'$ ) $E'$ $T$ | id + id ) * id \$ |
| \$ $E'$ $T'$ ) $E'$ $T'$ $F$ | id + id ) * id \$ |
| \$ $E'$ $T'$ ) $E'$ $T'$ id | id + id ) * id \$ |
| \$ $E'$ $T'$ ) $E'$ $T'$ | + id ) * id \$ |
| \$ $E'$ $T'$ ) $E'$ | + id ) * id \$ |
| \$ $E'$ $T'$ ) $E'$ $T$ + | + id ) * id \$ |
| \$ $E'$ $T'$ ) $E'$ $T$ | id ) * id \$ |

# Example

Example (Predictive Parsing)

| Stack | Input |
|---|---|
| \$ $E$ | ( id + id ) * id \$ |
| \$ $E'$ $T$ | ( id + id ) * id \$ |
| \$ $E'$ $T'$ $F$ | ( id + id ) * id \$ |
| \$ $E'$ $T'$ ) $E$ ( | ( id + id ) * id \$ |
| \$ $E'$ $T'$ ) $E$ | id + id ) * id \$ |
| \$ $E'$ $T'$ ) $E'$ $T$ | id + id ) * id \$ |
| \$ $E'$ $T'$ ) $E'$ $T'$ $F$ | id + id ) * id \$ |
| \$ $E'$ $T'$ ) $E'$ $T'$ id | id + id ) * id \$ |
| \$ $E'$ $T'$ ) $E'$ $T'$ | + id ) * id \$ |
| \$ $E'$ $T'$ ) $E'$ | + id ) * id \$ |
| \$ $E'$ $T'$ ) $E'$ $T$ + | + id ) * id \$ |
| \$ $E'$ $T'$ ) $E'$ $T$ | id ) * id \$ |

# Example

Example (Predictive Parsing)

| Stack | Input |
|---|---|
| \$ $E$ | ( **id** + **id** ) ∗ **id** \$ |
| \$ $E'$ $T$ | ( **id** + **id** ) ∗ **id** \$ |
| \$ $E'$ $T'$ $F$ | ( **id** + **id** ) ∗ **id** \$ |
| \$ $E'$ $T'$ ) $E$ ( | ( **id** + **id** ) ∗ **id** \$ |
| \$ $E'$ $T'$ ) $E$ | **id** + **id** ) ∗ **id** \$ |
| \$ $E'$ $T'$ ) $E'$ $T$ | **id** + **id** ) ∗ **id** \$ |
| \$ $E'$ $T'$ ) $E'$ $T'$ $F$ | id + id ) ∗ id \$ |
| \$ $E'$ $T'$ ) $E'$ $T'$ id | id + id ) ∗ id \$ |
| \$ $E'$ $T'$ ) $E'$ $T'$ | + id ) ∗ id \$ |
| \$ $E'$ $T'$ ) $E'$ | + id ) ∗ id \$ |
| \$ $E'$ $T'$ ) $E'$ $T$ + | + id ) ∗ id \$ |
| \$ $E'$ $T'$ ) $E'$ $T$ | id ) ∗ id \$ |

# Example

Example (Predictive Parsing)

| Stack | Input |
|---|---|
| \$ $E$ | ( **id** + **id** ) ∗ **id** \$ |
| \$ $E'$ $T$ | ( **id** + **id** ) ∗ **id** \$ |
| \$ $E'$ $T'$ $F$ | ( **id** + **id** ) ∗ **id** \$ |
| \$ $E'$ $T'$ ) $E$ ( | ( **id** + **id** ) ∗ **id** \$ |
| \$ $E'$ $T'$ ) $E$ | **id** + **id** ) ∗ **id** \$ |
| \$ $E'$ $T'$ ) $E'$ $T$ | **id** + **id** ) ∗ **id** \$ |
| \$ $E'$ $T'$ ) $E'$ $T'$ $F$ | **id** + **id** ) ∗ **id** \$ |
| \$ $E'$ $T'$ ) $E'$ $T'$ id | id + id ) ∗ id \$ |
| \$ $E'$ $T'$ ) $E'$ $T'$ | + id ) ∗ id \$ |
| \$ $E'$ $T'$ ) $E'$ | + id ) ∗ id \$ |
| \$ $E'$ $T'$ ) $E'$ $T$ + | + id ) ∗ id \$ |
| \$ $E'$ $T'$ ) $E'$ $T$ | id ) ∗ id \$ |

# Example

Example (Predictive Parsing)

| Stack | Input |
|---|---|
| \$ $E$ | ( id + id ) * id \$ |
| \$ $E'$ $T$ | ( id + id ) * id \$ |
| \$ $E'$ $T'$ $F$ | ( id + id ) * id \$ |
| \$ $E'$ $T'$ ) $E$ ( | ( id + id ) * id \$ |
| \$ $E'$ $T'$ ) $E$ | id + id ) * id \$ |
| \$ $E'$ $T'$ ) $E'$ $T$ | id + id ) * id \$ |
| \$ $E'$ $T'$ ) $E'$ $T'$ $F$ | id + id ) * id \$ |
| \$ $E'$ $T'$ ) $E'$ $T'$ id | id + id ) * id \$ |
| \$ $E'$ $T'$ ) $E'$ $T'$ | + id ) * id \$ |
| \$ $E'$ $T'$ ) $E'$ | + id ) * id \$ |
| \$ $E'$ $T'$ ) $E'$ $T$ + | + id ) * id \$ |
| \$ $E'$ $T'$ ) $E'$ $T$ | id ) * id \$ |

# Example

Example (Predictive Parsing)

| Stack | Input |
|---|---|
| \$ $E$ | ( id + id ) * id \$ |
| \$ $E'$ $T$ | ( id + id ) * id \$ |
| \$ $E'$ $T'$ $F$ | ( id + id ) * id \$ |
| \$ $E'$ $T'$ ) $E$ ( | ( id + id ) * id \$ |
| \$ $E'$ $T'$ ) $E$ | id + id ) * id \$ |
| \$ $E'$ $T'$ ) $E'$ $T$ | id + id ) * id \$ |
| \$ $E'$ $T'$ ) $E'$ $T'$ $F$ | id + id ) * id \$ |
| \$ $E'$ $T'$ ) $E'$ $T'$ id | id + id ) * id \$ |
| \$ $E'$ $T'$ ) $E'$ $T'$ | + id ) * id \$ |
| \$ $E'$ $T'$ ) $E'$ | + id ) * id \$ |
| \$ $E'$ $T'$ ) $E'$ $T$ + | + id ) * id \$ |
| \$ $E'$ $T'$ ) $E'$ $T$ | id ) * id \$ |

# Example

Example (Predictive Parsing)

| Stack | Input |
|---|---|
| \$ $E$ | ( id + id ) * id \$ |
| \$ $E'$ $T$ | ( id + id ) * id \$ |
| \$ $E'$ $T'$ $F$ | ( id + id ) * id \$ |
| \$ $E'$ $T'$ ) $E$ ( | ( id + id ) * id \$ |
| \$ $E'$ $T'$ ) $E$ | id + id ) * id \$ |
| \$ $E'$ $T'$ ) $E'$ $T$ | id + id ) * id \$ |
| \$ $E'$ $T'$ ) $E'$ $T'$ $F$ | id + id ) * id \$ |
| \$ $E'$ $T'$ ) $E'$ $T'$ id | id + id ) * id \$ |
| \$ $E'$ $T'$ ) $E'$ $T'$ | + id ) * id \$ |
| \$ $E'$ $T'$ ) $E'$ | + id ) * id \$ |
| \$ $E'$ $T'$ ) $E'$ $T$ + | + id ) * id \$ |
| \$ $E'$ $T'$ ) $E'$ $T$ | id ) * id \$ |

# Example

Example (Predictive Parsing)

| Stack | Input |
|---|---|
| \$ $E$ | ( id + id ) * id \$ |
| \$ $E'$ $T$ | ( id + id ) * id \$ |
| \$ $E'$ $T'$ $F$ | ( id + id ) * id \$ |
| \$ $E'$ $T'$ ) $E$ ( | ( id + id ) * id \$ |
| \$ $E'$ $T'$ ) $E$ | id + id ) * id \$ |
| \$ $E'$ $T'$ ) $E'$ $T$ | id + id ) * id \$ |
| \$ $E'$ $T'$ ) $E'$ $T'$ $F$ | id + id ) * id \$ |
| \$ $E'$ $T'$ ) $E'$ $T'$ id | id + id ) * id \$ |
| \$ $E'$ $T'$ ) $E'$ $T'$ | + id ) * id \$ |
| \$ $E'$ $T'$ ) $E'$ | + id ) * id \$ |
| \$ $E'$ $T'$ ) $E'$ $T$ + | + id ) * id \$ |
| \$ $E'$ $T'$ ) $E'$ $T$ | id ) * id \$ |

# Example

Example (Predictive Parsing)

| Stack | Input |
|---|---|
| \$ $E$ | ( id + id ) * id \$ |
| \$ $E'$ $T$ | ( id + id ) * id \$ |
| \$ $E'$ $T'$ $F$ | ( id + id ) * id \$ |
| \$ $E'$ $T'$ ) $E$ ( | ( id + id ) * id \$ |
| \$ $E'$ $T'$ ) $E$ | id + id ) * id \$ |
| \$ $E'$ $T'$ ) $E'$ $T$ | id + id ) * id \$ |
| \$ $E'$ $T'$ ) $E'$ $T'$ $F$ | id + id ) * id \$ |
| \$ $E'$ $T'$ ) $E'$ $T'$ id | id + id ) * id \$ |
| \$ $E'$ $T'$ ) $E'$ $T'$ | + id ) * id \$ |
| \$ $E'$ $T'$ ) $E'$ | + id ) * id \$ |
| \$ $E'$ $T'$ ) $E'$ $T$ + | + id ) * id \$ |
| \$ $E'$ $T'$ ) $E'$ $T$ | id ) * id \$ |

# Example

Example (Predictive Parsing)

| Stack | Input |
|---|---|
| \$ $E'$ $T'$ ) $E'$ $T'$ $F$ | **id** ) * **id** \$ |
| \$ $E'$ $T'$ ) $E'$ $T'$ id | id ) * id \$ |
| \$ $E'$ $T'$ ) $E'$ $T'$ | ) * id \$ |
| \$ $E'$ $T'$ ) $E'$ | ) * id \$ |
| \$ $E'$ $T'$ ) | ) * id \$ |
| \$ $E'$ $T'$ | * id \$ |
| \$ $E'$ $T'$ $F$ * | * id \$ |
| \$ $E'$ $T'$ $F$ | id \$ |
| \$ $E'$ $T'$ id | id \$ |
| \$ $E'$ $T'$ | \$ |
| \$ $E'$ | \$ |
| \$ | \$ |

# Example

Example (Predictive Parsing)

| Stack | Input |
|---|---|
| \$ $E'$ $T'$ ) $E'$ $T'$ $F$ | **id** ) * **id** \$ |
| \$ $E'$ $T'$ ) $E'$ $T'$ **id** | **id** ) * **id** \$ |
| \$ $E'$ $T'$ ) $E'$ $T'$ | ) * id \$ |
| \$ $E'$ $T'$ ) $E'$ | ) * id \$ |
| \$ $E'$ $T'$ ) | ) * id \$ |
| \$ $E'$ $T'$ | * id \$ |
| \$ $E'$ $T'$ $F$ * | * id \$ |
| \$ $E'$ $T'$ $F$ | id \$ |
| \$ $E'$ $T'$ **id** | id \$ |
| \$ $E'$ $T'$ | \$ |
| \$ $E'$ | \$ |
| \$ | \$ |

# Example

Example (Predictive Parsing)

| Stack | Input |
|---|---|
| \$ $E'$ $T'$ ) $E'$ $T'$ $F$ | **id** ) * **id** \$ |
| \$ $E'$ $T'$ ) $E'$ $T'$ **id** | **id** ) * **id** \$ |
| \$ $E'$ $T'$ ) $E'$ $T'$ | ) * **id** \$ |
| \$ $E'$ $T'$ ) $E'$ | ) * id \$ |
| \$ $E'$ $T'$ ) | ) * id \$ |
| \$ $E'$ $T'$ | * id \$ |
| \$ $E'$ $T'$ $F$ * | * id \$ |
| \$ $E'$ $T'$ $F$ | id \$ |
| \$ $E'$ $T'$ **id** | id \$ |
| \$ $E'$ $T'$ | \$ |
| \$ $E'$ | \$ |
| \$ | \$ |

# Example

Example (Predictive Parsing)

| Stack | Input |
|---|---|
| \$ $E'$ $T'$ ) $E'$ $T'$ $F$ | **id** ) * **id** \$ |
| \$ $E'$ $T'$ ) $E'$ $T'$ **id** | **id** ) * **id** \$ |
| \$ $E'$ $T'$ ) $E'$ $T'$ | ) * **id** \$ |
| \$ $E'$ $T'$ ) $E'$ | ) * **id** \$ |
| \$ $E'$ $T'$ ) | ) * id \$ |
| \$ $E'$ $T'$ | * id \$ |
| \$ $E'$ $T'$ $F$ * | * id \$ |
| \$ $E'$ $T'$ $F$ | id \$ |
| \$ $E'$ $T'$ **id** | id \$ |
| \$ $E'$ $T'$ | \$ |
| \$ $E'$ | \$ |
| \$ | \$ |

# Example

Example (Predictive Parsing)

| Stack | Input |
|---|---|
| \$ $E'$ $T'$ ) $E'$ $T'$ $F$ | **id** ) * **id** \$ |
| \$ $E'$ $T'$ ) $E'$ $T'$ **id** | **id** ) * **id** \$ |
| \$ $E'$ $T'$ ) $E'$ $T'$ | ) * **id** \$ |
| \$ $E'$ $T'$ ) $E'$ | ) * **id** \$ |
| \$ $E'$ $T'$ ) | ) * **id** \$ |
| \$ $E'$ $T'$ | * id \$ |
| \$ $E'$ $T'$ $F$ * | * id \$ |
| \$ $E'$ $T'$ $F$ | id \$ |
| \$ $E'$ $T'$ **id** | id \$ |
| \$ $E'$ $T'$ | \$ |
| \$ $E'$ | \$ |
| \$ | \$ |

# Example

Example (Predictive Parsing)

| Stack | Input |
|---|---|
| \$ $E'$ $T'$ ) $E'$ $T'$ $F$ | **id** ) * **id** \$ |
| \$ $E'$ $T'$ ) $E'$ $T'$ **id** | **id** ) * **id** \$ |
| \$ $E'$ $T'$ ) $E'$ $T'$ | ) * **id** \$ |
| \$ $E'$ $T'$ ) $E'$ | ) * **id** \$ |
| \$ $E'$ $T'$ ) | ) * **id** \$ |
| \$ $E'$ $T'$ | * **id** \$ |
| \$ $E'$ $T'$ $F$ * | * id \$ |
| \$ $E'$ $T'$ $F$ | id \$ |
| \$ $E'$ $T'$ **id** | id \$ |
| \$ $E'$ $T'$ | \$ |
| \$ $E'$ | \$ |
| \$ | \$ |

# Example

Example (Predictive Parsing)

| Stack | Input |
|---|---|
| \$ $E'$ $T'$ ) $E'$ $T'$ $F$ | **id** ) * **id** \$ |
| \$ $E'$ $T'$ ) $E'$ $T'$ **id** | **id** ) * **id** \$ |
| \$ $E'$ $T'$ ) $E'$ $T'$ | ) * **id** \$ |
| \$ $E'$ $T'$ ) $E'$ | ) * **id** \$ |
| \$ $E'$ $T'$ ) | ) * **id** \$ |
| \$ $E'$ $T'$ | * **id** \$ |
| \$ $E'$ $T'$ $F$ * | * **id** \$ |
| \$ $E'$ $T'$ $F$ | **id** \$ |
| \$ $E'$ $T'$ **id** | **id** \$ |
| \$ $E'$ $T'$ | \$ |
| \$ $E'$ | \$ |
| \$ | \$ |

# Example

Example (Predictive Parsing)

| Stack | Input |
|---|---|
| \$ $E'$ $T'$ ) $E'$ $T'$ $F$ | **id** ) * **id** \$ |
| \$ $E'$ $T'$ ) $E'$ $T'$ **id** | **id** ) * **id** \$ |
| \$ $E'$ $T'$ ) $E'$ $T'$ | ) * **id** \$ |
| \$ $E'$ $T'$ ) $E'$ | ) * **id** \$ |
| \$ $E'$ $T'$ ) | ) * **id** \$ |
| \$ $E'$ $T'$ | * **id** \$ |
| \$ $E'$ $T'$ $F$ * | * **id** \$ |
| \$ $E'$ $T'$ $F$ | **id** \$ |
| \$ $E'$ $T'$ **id** | **id** \$ |
| \$ $E'$ $T'$ | \$ |
| \$ $E'$ | \$ |
| \$ | \$ |

# Example

Example (Predictive Parsing)

| Stack | Input |
|---|---|
| \$ $E'$ $T'$ ) $E'$ $T'$ $F$ | **id** ) * **id** \$ |
| \$ $E'$ $T'$ ) $E'$ $T'$ **id** | **id** ) * **id** \$ |
| \$ $E'$ $T'$ ) $E'$ $T'$ | ) * **id** \$ |
| \$ $E'$ $T'$ ) $E'$ | ) * **id** \$ |
| \$ $E'$ $T'$ ) | ) * **id** \$ |
| \$ $E'$ $T'$ | * **id** \$ |
| \$ $E'$ $T'$ $F$ * | * **id** \$ |
| \$ $E'$ $T'$ $F$ | **id** \$ |
| \$ $E'$ $T'$ **id** | **id** \$ |
| \$ $E'$ $T'$ | \$ |
| \$ $E'$ | \$ |
| \$ | \$ |

# Example

Example (Predictive Parsing)

| Stack | Input |
|---|---|
| \$ $E'$ $T'$ ) $E'$ $T'$ $F$ | **id** ) * **id** \$ |
| \$ $E'$ $T'$ ) $E'$ $T'$ **id** | **id** ) * **id** \$ |
| \$ $E'$ $T'$ ) $E'$ $T'$ | ) * **id** \$ |
| \$ $E'$ $T'$ ) $E'$ | ) * **id** \$ |
| \$ $E'$ $T'$ ) | ) * **id** \$ |
| \$ $E'$ $T'$ | * **id** \$ |
| \$ $E'$ $T'$ $F$ * | * **id** \$ |
| \$ $E'$ $T'$ $F$ | **id** \$ |
| \$ $E'$ $T'$ **id** | **id** \$ |
| \$ $E'$ $T'$ | \$ |
| \$ $E'$ | \$ |
| \$ | \$ |

# Example

Example (Predictive Parsing)

| Stack | Input |
|---|---|
| \$ $E'$ $T'$ ) $E'$ $T'$ $F$ | **id** ) * **id** \$ |
| \$ $E'$ $T'$ ) $E'$ $T'$ **id** | **id** ) * **id** \$ |
| \$ $E'$ $T'$ ) $E'$ $T'$ | ) * **id** \$ |
| \$ $E'$ $T'$ ) $E'$ | ) * **id** \$ |
| \$ $E'$ $T'$ ) | ) * **id** \$ |
| \$ $E'$ $T'$ | * **id** \$ |
| \$ $E'$ $T'$ $F$ * | * **id** \$ |
| \$ $E'$ $T'$ $F$ | **id** \$ |
| \$ $E'$ $T'$ **id** | **id** \$ |
| \$ $E'$ $T'$ | \$ |
| \$ $E'$ | \$ |
| \$ | \$ |

# Example

$E

)id + id) * id

Example (Predictive Parsing)

$E

M[E, )] =

M[E, )]

| Stack | Input |
|---|---|
| $ $E'\ T'\ )\ E'\ T'\ F$ | id ) * id $ |
| $ $E'\ T'\ )\ E'\ T'$ id | id ) * id $ |
| $ $E'\ T'\ )\ E'\ T'$ | ) * id $ |
| $ $E'\ T'\ )\ E'$ | ) * id $ |
| $ $E'\ T'\ )$ | ) * id $ |
| $ $E'\ T'$ | * id $ |
| $ $E'\ T'\ F$ * | * id $ |
| $ $E'\ T'\ F$ | id $ |
| $ $E'\ T'$ id | id $ |
| $ $E'\ T'$ | $ |
| $ $E'$ | $ |
| $ | $ |

# Outline



# Assignment

Homework

- For the grammar

$$S \rightarrow ( \ L \ ) \mid \mathbf{id}$$
$$L \rightarrow L \ , \ S \mid S$$

construct a parse table for this grammar.

- Parse the expression **(id , id)**.