BOTTOM-UP PARSING

Bottom-Up Parsing

- Bottom-Up Parser : Constructs a parse tree for an input string beginning at the leaves(the bottom) and working up towards the root(the top)
- We can think of this process as one of “reducing” a string w to the start symbol of a grammar
- Bottom-up parsing is also known as *shift-reduce parsing* because its two main actions are shift and reduce.
  - At each shift action, the current symbol in the input string is pushed to a stack.
  - At each reduction step, the symbols at the top of the stack (this symbol sequence is the right side of a production) will replaced by the non-terminal at the left side of that production.

Shift-Reduce Parsing

- A shift-reduce parser tries to reduce the given input string into the starting symbol.

  a string ➔ the starting symbol

  reduced to

- At each reduction step, a substring of the input matching to the right side of a production rule is replaced by the non-terminal at the left side of that production rule.
- If the substring is chosen correctly, the right most derivation of that string is created in the reverse order.

Rightmost Derivation: $S \underset{rm}{\overset{*}{\Rightarrow}} \omega$

Shift-Reduce Parser finds: $\omega \underset{rm}{\Leftarrow} \ldots \underset{rm}{\Leftarrow} S$

Shift-Reduce Parsing-Example

- Consider the grammar

  S ⟶ aABe

  A ⟶ Abc | b

  B ⟶ d

Input string : abbcde

aAbcde

aAde ⇓ reduction

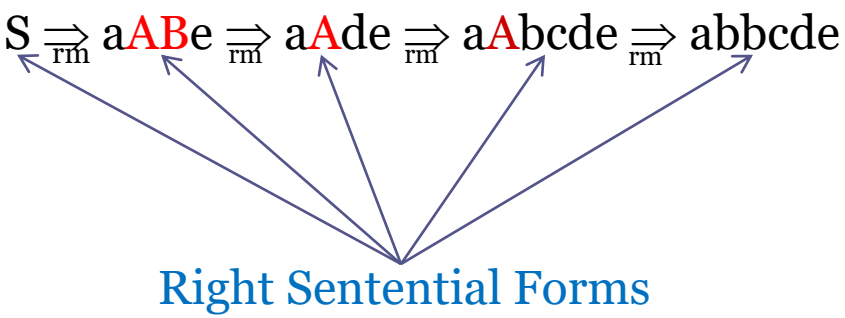
aABe

S

We can scan abbcde looking for a substring that matches the right side of some production.The substrings b and d qualify.Let us choose left most b and replace it by A,the left side of the production A→b;we thus obtain the string aAbcde.Now the substrings Abc,b and d match the right side of some production.Although b is the leftmost substring that matches the right side of the some production,we choose to replace the substring Abc by A,the left side of the production A→Abc.We obtain aAde.Then replacing d by B,and then replacing the entire string by S.Thus,by a sequence of four reductions we are able to reduce abbcde to S

Shift-Reduce Parsing-Example

- These reductions infact trace out the following right-most derivation in reverse



- How do we know which substring to be replaced at each reduction step?

Handle

- Informally, a “handle” of a string is a substring that matches the right side of the production,and whose reduction to nonterminal on the left side of the production represents one step along the reverse of a rightmost derivation
  - But not every substring matches the right side of a production rule is handle.
- Formally , a “handle” of a right sentential form γ (≡ αβω) is a production rule $A \rightarrow \beta$ and a position of γ where the string β may be found and replaced by A to produce the previous right-sentential form in a rightmost derivation of γ.

$$S \overset{*}{\underset{rm}{\Rightarrow}} \alpha A \omega \underset{rm}{\Rightarrow} \alpha\beta\omega$$

  then $A \rightarrow \beta$ in the position following α is a handle of αβω
- The string ω to the right of the handle contains only terminal symbols.

Example

- Consider the example discussed in the beginning,abbcde is a right sentential form whose handle is A→b at position 2.Likewise,aAbcde is a right sentential form whose handle is A→Abc at position 2.
- Sometimes we say “the substring β is a handle of αβω” if the position of β and the production A→β we have in mind are clear.

Handle Pruning

- A rightmost derivation in reverse can be obtained by “handle pruning”.That is,we start with a string of terminals w that we wish to parse.If ω is a sentence of grammar at hand,then ω = γ,where γ_n is the nth right-sentential form of some as yet unknown rightmost derivation

$$S = \gamma_0 \underset{rm}{\Rightarrow} \gamma_1 \underset{rm}{\Rightarrow} \gamma_2 \underset{rm}{\Rightarrow} \cdots \underset{rm}{\Rightarrow} \gamma_{n-1} \underset{rm}{\Rightarrow} \gamma_n = \omega$$

Input string

Handle Pruning

$$S = \gamma_0 \underset{rm}{\Rightarrow} \gamma_1 \underset{rm}{\Rightarrow} \gamma_2 \underset{rm}{\Rightarrow} \cdots \underset{rm}{\Rightarrow} \gamma_{n-1} \underset{rm}{\Rightarrow} \gamma_n = \omega$$

- Start from γ_n, find a handle $A_n \rightarrow \beta_n$ in γ_n, and replace β_n in by A_n to get γ_{n-1}.
- Then find a handle $A_{n-1} \rightarrow \beta_{n-1}$ in γ_{n-1}, and replace β_{n-1} in by A_{n-1} to get γ_{n-2}.
- Repeat this, until we reach S.

A Shift-Reduce Parser

$E \rightarrow E+T \mid T$
$T \rightarrow T*F \mid F$
$F \rightarrow (E) \mid id$

Right-Most Derivation of `id+id*id`

$E \Rightarrow E+T \Rightarrow E+T*F \Rightarrow E+T*id \Rightarrow E+F*id$
$\Rightarrow E+id*id \Rightarrow T+id*id \Rightarrow F+id*id \Rightarrow id+id*id$

Right-Most Sentential form	HANDLE	Reducing Production
id+id*id	id	F→id
F+id*id	F	T→F
T+id*id	T	E→T
E+id*id	id	F→id
E+F*id	F	T→F
E+T*id	Id	F→id
E+T*F	T*F	T→T*F
E+T	E+T	E→E+T
E		

A Stack Implementation of a Shift-Reduce Parser

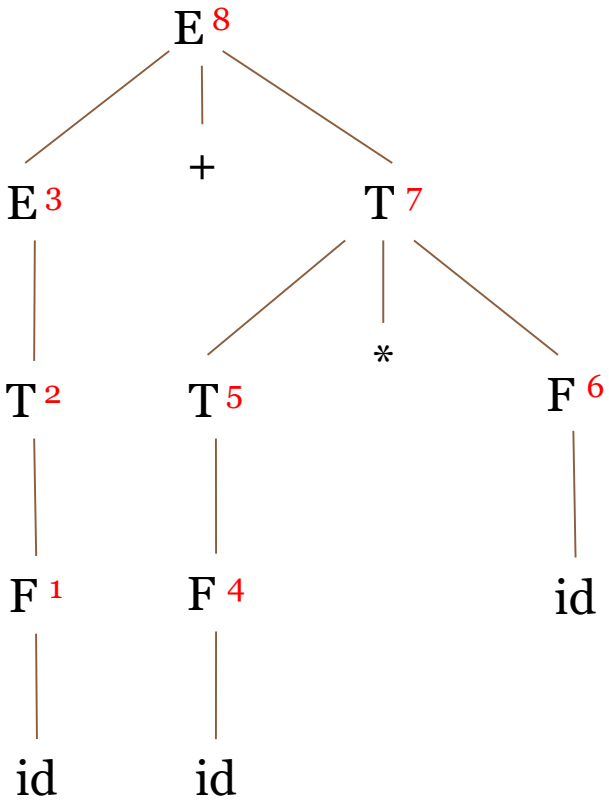
- There are four possible actions of a shift-parser action:
  1. Shift : The next input symbol is shifted onto the top of the stack.
  2. Reduce: Replace the handle on the top of the stack by the non-terminal.
  3. Accept: Successful completion of parsing.
  4. Error: Parser discovers a syntax error, and calls an error recovery routine.

- Initial stack just contains only the end-marker $.
- The end of the input string is marked by the end-marker $.

A Stack Implementation of A Shift-Reduce Parser

Stack	Input	Action
$	id+id*id$shift	
$id	+id*id$	Reduce by F→id
$F	+id*id$	Reduce by T→F
$T	+id*id$	Reduce by E→T
$E	+id*id$	Shift
$E+	Id*id$	Shift
$E+id	*id$	Reduce by F→id
$E+F	*id$	Reduce by T→F
$E+T	*id$	Shift
$E+T*	id$	Shift
$E+T*id	$	Reduce by F→id
$E+T*F	$	Reduce by T→T*F
$E+T	$	Reduce by E →E+T
$E	$	Accept

Parse Tree



THANK YOU