Bottom up Parser

Bottom up Parser

- LR methods (Left-to-right, Rightmost derivation)
  - SLR
  - Canonical LR (CALR)
  - Look Ahead LR (LALR)
- Other special cases:
  - Shift-reduce parsing
  - Operator-precedence parsing

Bottom up parser

- Bottom up parsers build a derivation by working from the input back toward the start symbol
  - Builds parse tree from leaves to root
  - Builds reverse rightmost derivation

$A \to \underline{\alpha B \beta}$

handle

$$\cdots \underbrace{\boxed{\alpha B \beta}}_{A} \cdots$$

Handle

- Since Bottom up parsers match the RHS of production with LHS, a concept called ‘handle’ is defined
- A *handle* is a substring of grammar symbols in a *right-sentential form* that matches a right-hand side of a production
- A handle’s reduction to the non-terminal on the LHS represents one step along the reverse of a rightmost derivation
- This sub-string is a handle

Handle - Example

- Expression Grammar Handles
  - id
  - E * E
  - (E)

Shift Reduce Parser

- Simplest of the Bottom up Parsers
- *Shift* input symbols until a handle is found.
- *Reduce* the substring to the non-terminal on the LHS of the corresponding production.

Shift Reduce Parser

- A shift-reduce parser has 4 actions:
  - *Shift* the next input symbol is shifted onto the stack
  - *Reduce* the handle that is at top of stack
    - pop handle
    - push appropriate LHS symbol
  - *Accept* and stop parsing & report success
  - *Error* recovery routine is called

Consider a grammar

- 1. S → E
- 2. E → E **+** E
- 3. E → E ***** E
- 4. E → **id**

Stack	Action	Input
$	shift	id + id * id $
$ id	Reduce by rule 4	+id*id $
$ E	shift	
$ E+	Shift	id * id $
$ E + id	Reduce by rule 4	* id $
$ E + E	Reduce by Rule 2	* id $
$ E	Shift	* id $
$ E *	Shift	id $

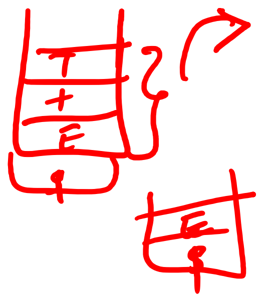
Parsing action

E → E+T | T
T → T*F | F
F → (E) | id

Stack	Action	Input
$ E *	Shift	id $
$ E * id	Reduce by rule 4	$
$ E * E	Reduce by rule 3	$
$ E	Accept	$

$	id +id * id $
$id	+id * id$
$ F	
$ T	
$E	+id *id$
$ E+	
$ E+ id	*id$

$ E+ F	
$ E+T =	
$ E+T	*id$
$ E^{+T} *	id$
$ E^{+T} * id	$
$ E^{+T} * F	Error



Acceptance

- When the stack has the start symbol and the input is exhausted, the shift reduce parser goes to an accepting state

Conflicts

- Shift-reduce and reduce-reduce conflicts are caused by
  - The limitations of the parsing method (even when the grammar is unambiguous)
  - Ambiguity of the grammar

$A \to \alpha \underline{B \beta .}$

$C \to \alpha \underline{B \beta \gamma}$