8 Shift-Reduce Parsing

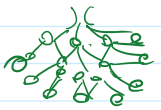
Wednesday, September 20, 2023 11:40 AM

Grammars (Context Free) solve the "specification" problem.

what about the "recognition" problem?

- "The Parsing Problem"
  decide whether $w \in L$ — L is a language specified by a grammar.

Algorithm to - construct derivations.
- construct parse tree



TYPES OF PARSERS:

- Top Down Parsers
  - construct parse tree from root (start symbol) to the leafs. (w)
    - "Recursive Descent Parser"
- Bottom-up Pasers.
  - construct pase tree from leaves (w) to root (start symbol)
    - "Shift-Reduce Parser"

• THE SHIFT-REDUCE PARSER

- D. Knuth.

- Reads input from Left to Right
- Produces Rightmost Derivation

Called: LR-Parser LR(k) $k = 0, 1$
k is the number of lookahead symbols that a parser needs to read to know what to do.

- LR(1)
- Not a General Parser.
  Limited to "LR-Grammars"

- Intuition About the shift-reduce parser.

Iteratively:

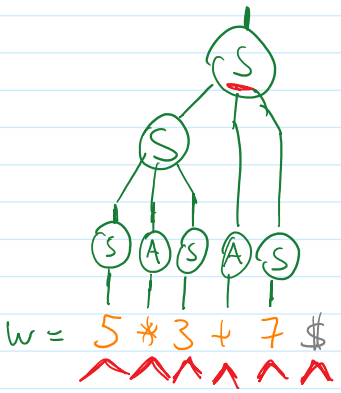
- shift: read next symbol from the input
- reduce: build an intermidiate node on the parse tree i.e. take nodes that correspond to the body of a rule, and connect them to a parent node that correspond to the head.

E.G.

E.G.

$S \rightarrow SAS \mid 5 \mid 3 \mid 7$

$A \rightarrow + \mid *$



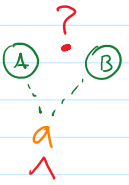
- CONFLICTS IN SHIFT-REDUCE PARSER
  - Reduce-Reduce Conflict

$S \rightarrow A \mid B$

$A \rightarrow a$

$B \rightarrow a$

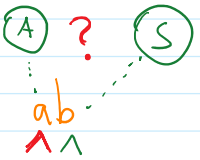
S
A
a



  - Shift-Reduce Conflict

$S \rightarrow ab \mid Ab$

$A \rightarrow a$



- SHIFT-REDUCE TRACE

```
PROCEDURE shift-reduce ()
  stack.push(0)     // 0 is the start state
  input := w$       // w is the input string
  x := first symbol of input

  WHILE ~stop DO
    s := top of stack
    IF action[s,x] = shift t
      stack.push(t)
      x := next input symbol

    ELSIF action[s,x] = reduce t  // A → β
      pop len(β) symbols from stack
      t := stack.top()
      stack.push( goto[t,A] )
      output( A → β )

    ELSIF action[s,x] = accept
      stop := true
```

E.G.

1) $S \to CC$
2) $C \to aC$
3) $C \to b$

abb

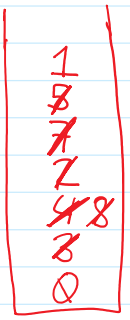
ACTION

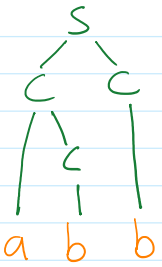
	a	b	$
0	S3	S4	
1			acc
2	S6	S7	
3	S3	S4	
4	R3	R3	
5			R1
6	S6	S7	
7			R3
8	R2	R2	
9			R2

GOTO

	S	C
0	1	2
1		
2		5
3		8
4		
5		
6		9
7		
8		
9		

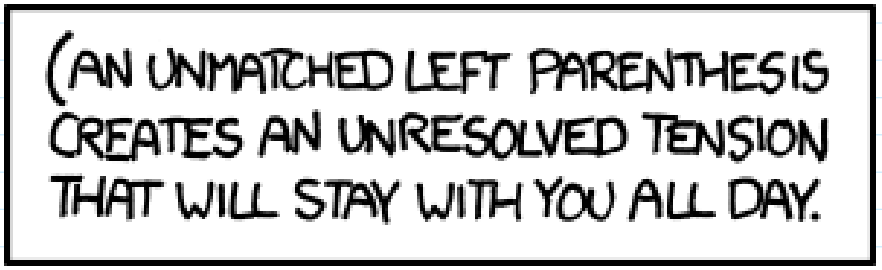
Word abb$





- $C \to b$
- $C \to aC$
- $C \to b$
- $S \to CC$

- XKCD:



- EOF -