9 Bison

Wednesday, September 27, 2023 10:48 AM

- SHIFT REDUCE PARSER

  Bison : an implementation.

  - input : Grammar Specification.
  - output: source-code for the corresponding shift-reduce parser.
  - designed to work with flex

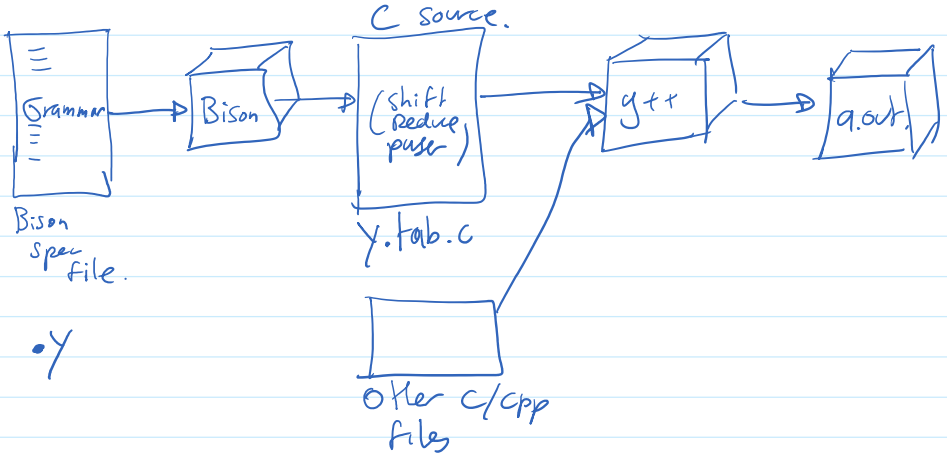
- HISTORY

  '70 yacc S. Johnson. B, C

  '80 GNU R. Corbett Bison.

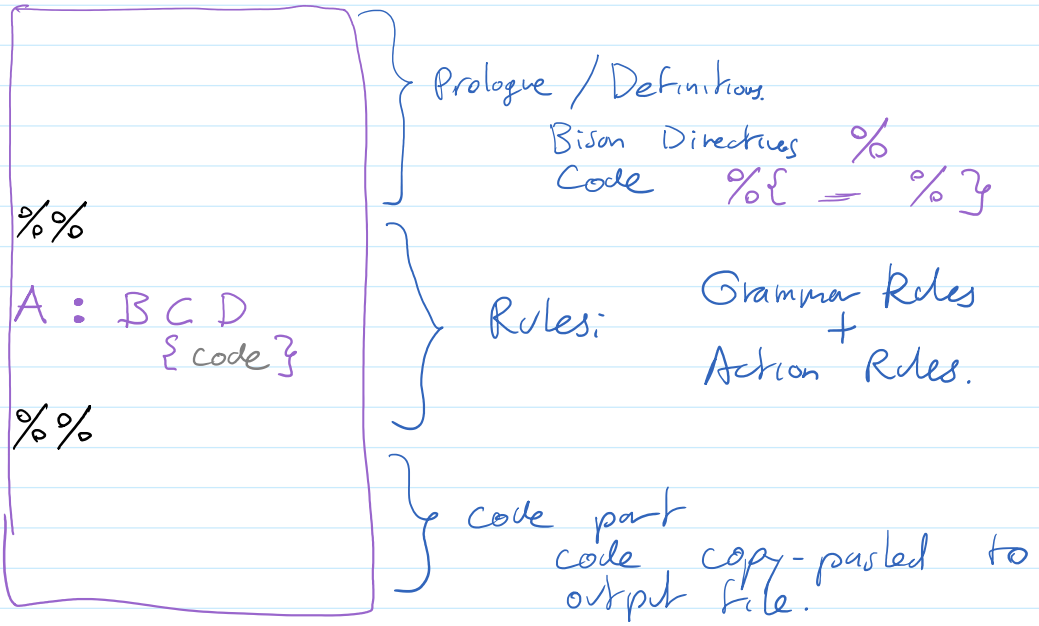
- BISON

  - Bison is a parser generator.



  - .y
  - Terminals symbols are strings.
  - bison expects. yylex() to be available. to feed terminal symbols to the paser.

- THE BISON SPECIFICATION FILE



- Bison allows you to attach a value to each non terminal using special variable $$

- THE CODE GENERATED BY BISON

Functions:

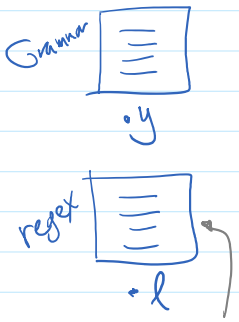
yyparse() - implements the shift-reduce parser.
- calls yylex() to get terminals.

returns 0 if parsing successful
1 if parsing fails

yyerror(char *) - called by yyparse() when an error is encountered.

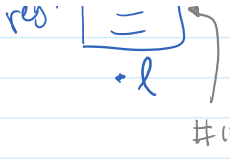
Note: you must write this function.

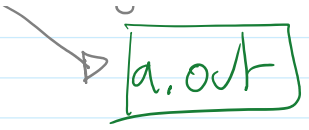
- BISON + FLEX WORKFLOW




```
$ bison -d <bison spec>
$ flex <flex spec>
$ bison <bison spec>
$ g++ -lfl *.c *.cpp
```

→ y.tab.h
→ lex.yy.c
→ y.tab.c
→ a.out





- BISON CONFLICTS
  - Bison will generate warnings when conflicts are found in the grammar.
  - Reduce-Reduce conflict
    Bison resolve conflict in favor of first rule in file.
  - Shift-reduce conflict.
    Bison resolves in favour of shift.

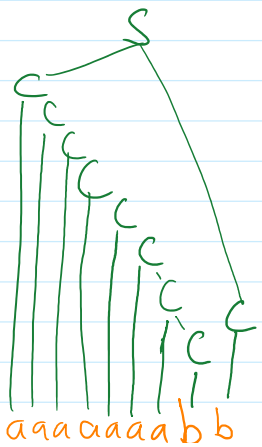
- LINK
  https://www.gnu.org/software/bison/

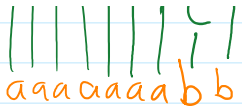
- THE BISON DEMO
  - EXAMPLE #1:

$\Sigma = \{a, b\}$

$S \rightarrow CC$
$C \rightarrow b$
$C \rightarrow aC$

yyparse()
yylex() ← write this manually





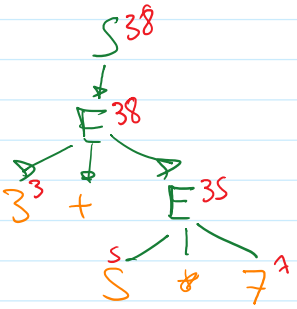
- EXAMPLE #2:

flex + bison "integer calculator"

$S \to \Lambda$
$S \to E$
$E \to int$
$E \to E + E$
$E \to E * E$

- flex: .l
  - recognize integers
  - recognize + *
- bison: .y
  - Parse line
  - Compute "value" of E

3 + 5 * 7



EOF