Regular Expressions

Lecture Objectives:

1) Explain the concept of BNF (Backus Normal Form or Backus–Naur Form) representation for grammar definition.
2) Define the concepts of a start symbol, a non-terminal symbol, and terminal symbol within a grammar.
3) Explain the concept of a rule in BNF.
4) Compare and contrast terminal symbil coverage, production coverage, and derivation coverage.
5) Explain the practical limitation to using derivation coverage in testing.
6) Define the concept of a ground string.

Design Example

- I want you to design and implement a command line calculator.
  - Allows the user to add, subtract, multiply, divide, and raise to a power.
- Talk with your neighbor.
  - How are you going to build it?

Design Example: Consensus on how to build it

take in 3 args

⇒ value

⇒ operation

} Split these apart?

- one space — String split ⇒ search for spaces

Lets break our design model.

Use a custom language

⇒ Javascript eval

(command

C ⇒ get arguments

from command line

take first argument and

use it

Definitions

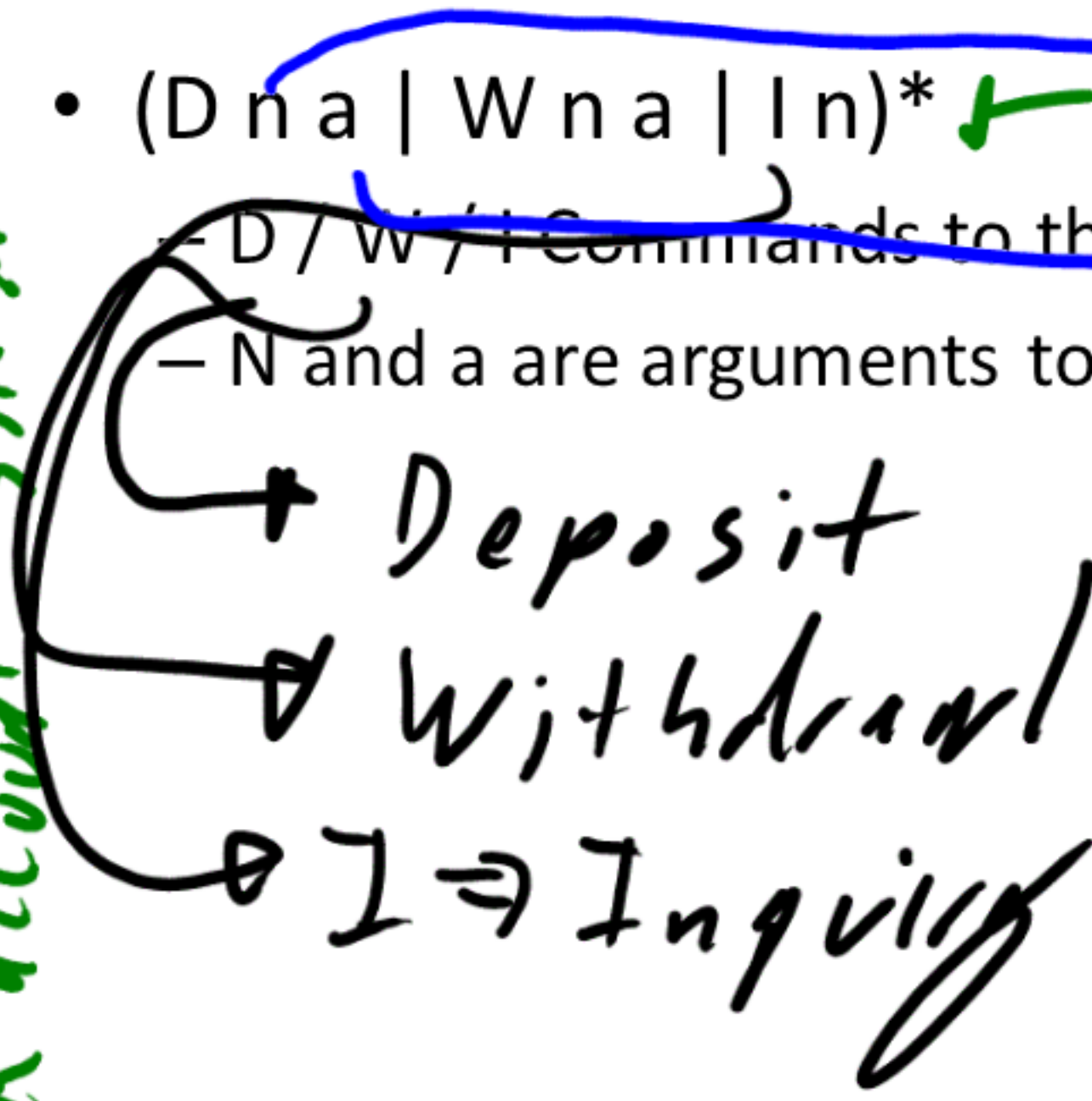
- Grammar
  - A grammar defines the allowable input formats for a program

what can be "said"

- Regular Expression (regex)
  - A regular expression is a sequence of characters that forms a template used to search for strings [Words, phrases, or just about any sequence of characters.] within text. (http://linux.about.com/cs/linux101/g/regularexpressi.htm)

Example Regular Expression (Bank system)

- (D n a | W n a | I n)*
  - D / W / I commands to the program
  - N and a are arguments to the commands



number
Amount

D 101 1000

I 101

W 101 200

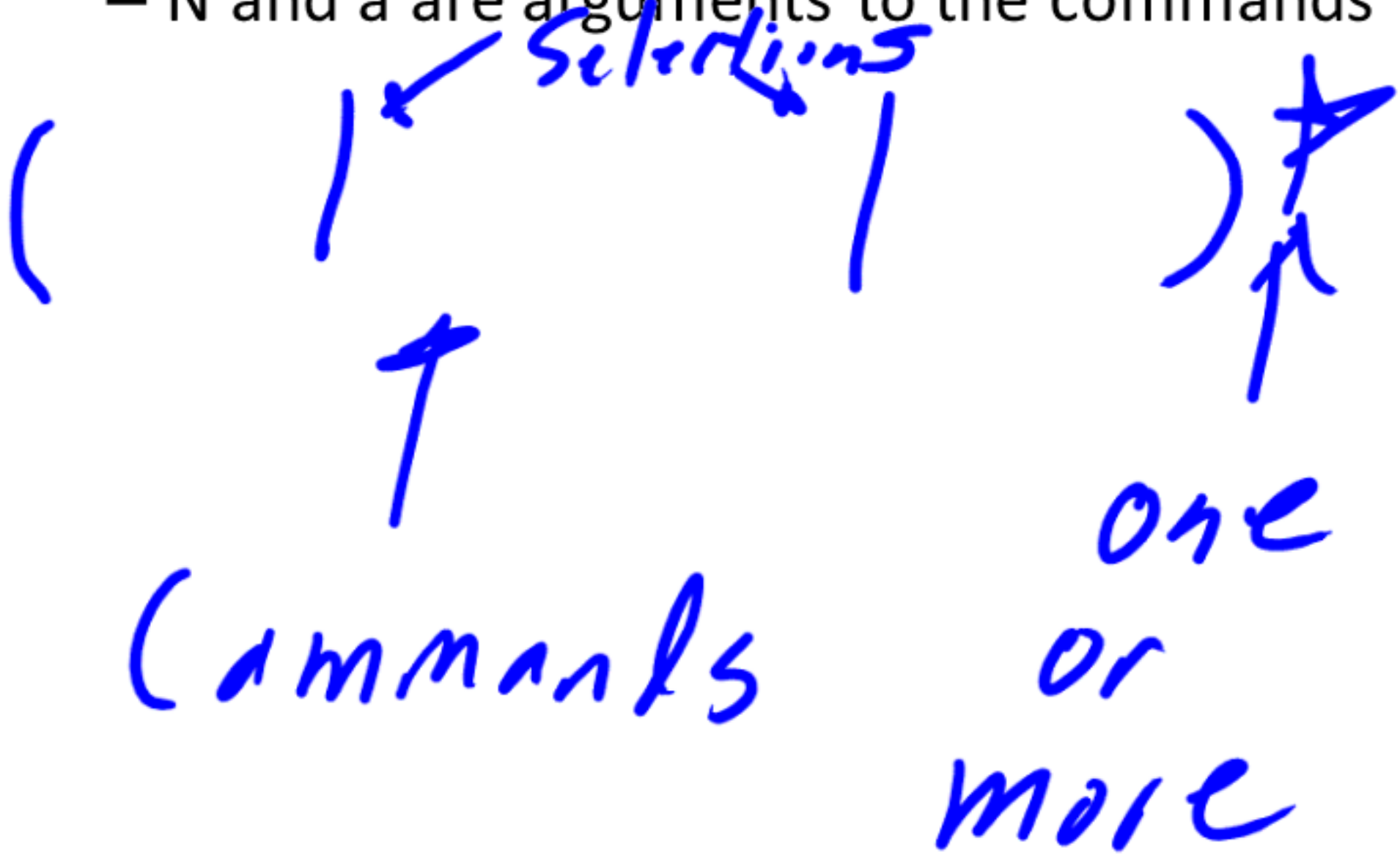
D 123456 7890

I 123456

W 123456 21

Example Regular Expression

- (D n a | W n a | I n)*
  - D / W / I Commands to the program
  - N and a are arguments to the commands



Example Regular Expression

- (D n a | W n a | I n)*
  - D / W / I Commands to the program
  - N and a are arguments to the commands

→ Not terminals

Example Code

```
public static void analyzeLine(String line) {
        if (line.matches("^[D][\\s][0-9]*[\\s][0-9]*")) {
            System.out.println("Deposit " + line);
        }
        if (line.matches("^[W][\\s][0-9]*[\\s][0-9]*")) {
            System.out.println("Withdraw " + line);
        }
        if (line.matches("^[I][\\s][0-9]*")) {
            System.out.println("Inquiry " + line);
        }
}
```

Starts with

D whitespace

any # of #'s 0-9

whitespace

takes a regular expression

Regular Expression

Example Code

```
public static void analyzeLine(String line) {
    if (line.matches("^[D][\\s][0-9]*[\\s][0-9]*")) {
        System.out.println("Deposit " + line);
    }
    if (line.matches("^[W][\\s][0-9]*[\\s][0-9]*")) {
        System.out.println("Withdraw " + line);
    }
    if (line.matches("^[I][\\s][0-9]*")) {
        System.out.println("Inquiry " + line);
    }
}
```

'0'-'9' => terminals

'D' 'W' 'I' => terminal

Context-free grammar (CFG)

- formal grammar in which every production rule is of the form
- $V \rightarrow w$

Single nonterminal symbol

string of terminals

Terminal symbols are the elementary symbols of the language defined by a formal grammar. *Nonterminal symbols* (or *syntactic variables*) are replaced by groups of terminal symbols according to the production rules.

Recognizers and Generators

- Recognizer ⇒ tests things
  - A program which decides whether or not a given string (or test case) is in the grammar
- Generator ⇒ makes things
  - A tool which derives a set of terminals from a grammar which represents valid entries

Coverage Related to Symbols

- Terminal Symbol Coverage
  - TR contains each terminal symbol in the grammar G
- Production Coverage
  - TR contains each production p in the grammar G
- Derivation Coverage
  - TR contains every possible string that can be generated from the grammar G

Our Example Again

- (D n a | W n a | I n)*
  - 14 terminal symbols

Getting back to our calculator

expr = term | expr "+" term | expr "-" term

term = factor | term "*" factor | term "/" factor

factor = atom | atom "^" factor

atom = number | "(" expr ")“

number = digit $^{1\text{-}n}$

digit =‘0’ ‘1’ ‘2’ ‘3’ ‘4’ ‘5’ ‘6’ ‘7’ ‘8’ ‘9’

Lets develop some tests