# Midterm Review

October 2, 2018

# About MidTerm

- Exam Scope: Overview – FuncLang
- Exam Date and Time: Oct. 9th 11:00 - 12:15
- Close Book
- 2016 Fall Midterm on Canvas for reference

# Contents of Review

- Concepts
- Problem Solving and Exercises

# What is a Programming Language

A language that can express all computation:

- **syntax**: validity - is the string a valid program of a programming language (does the string conform to the grammar of the programming language)?
- **semantics**: meaning - how to generate a value from the string?

# Parts of a Programming Language

- Computation: to actually compute, e.g. atomic operations
- Composition: to put together computation, e.g., loops, branches ...
- Abstraction: to achieve scalable programming, e.g., functions, variables

# Classification of Programming Languages

- general-purpose languages: C, Java, Scheme ...
- domain-specific languages: html, dot, sql ...

- High level language: human programs - python, Java, C programs
- Assembly language
- Machine language: compute executes - binary programs

# Programming Paradigms

Way of thinking about computation:

- Imperative programming: steps
- Object-oriented: objects
- Functional Programming: functions
- Logic programming: facts and relations

how to compute

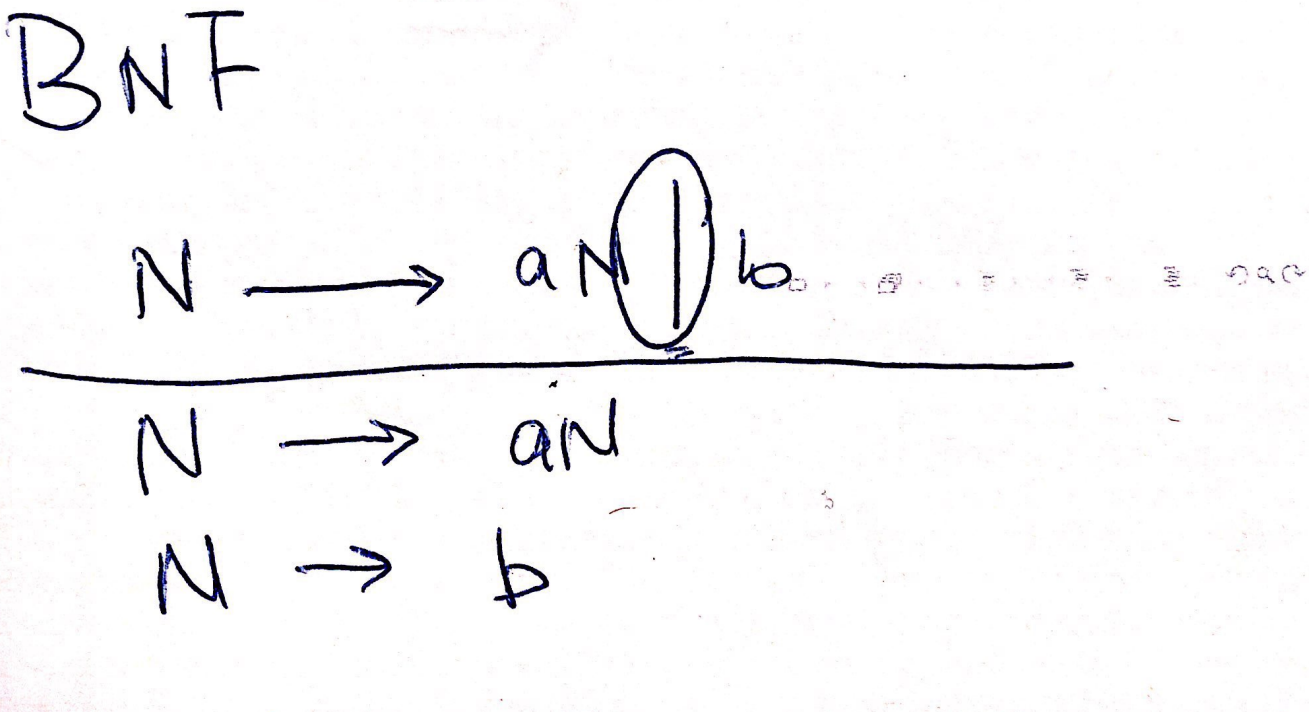
Declarative

what to compute

# Grammar

# Context Free Grammar

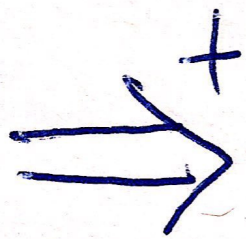
- 4-tuple: start symbol, terminals, non-terminals, production rules
- Start symbol: represent the program
- terminals: tokens
- non-terminals: units that composes the program (invisible in the code, abstract), e.g., Digit, Exp
- production rules: rules to derive from a non-terminal (what a unit consists of?)



# Derivation

Derive a string from the Start symbol by applying a set of production rules

- left-most derivation
- right-most derivation
- what if you cannot find a derivation for the string? Syntax errors in the string



# Parsing

Generate a parse tree from a string

- Start symbol is the root
- Parent is the non-terminal, its children are the terminals or non-terminals on the right hand side of a production rule you select at the current step of derivation
- Terminals are leaves and non-terminals are internal nodes
- If we cannot find a parse tree for a given string, the string does not belong to the language

# Ambiguity

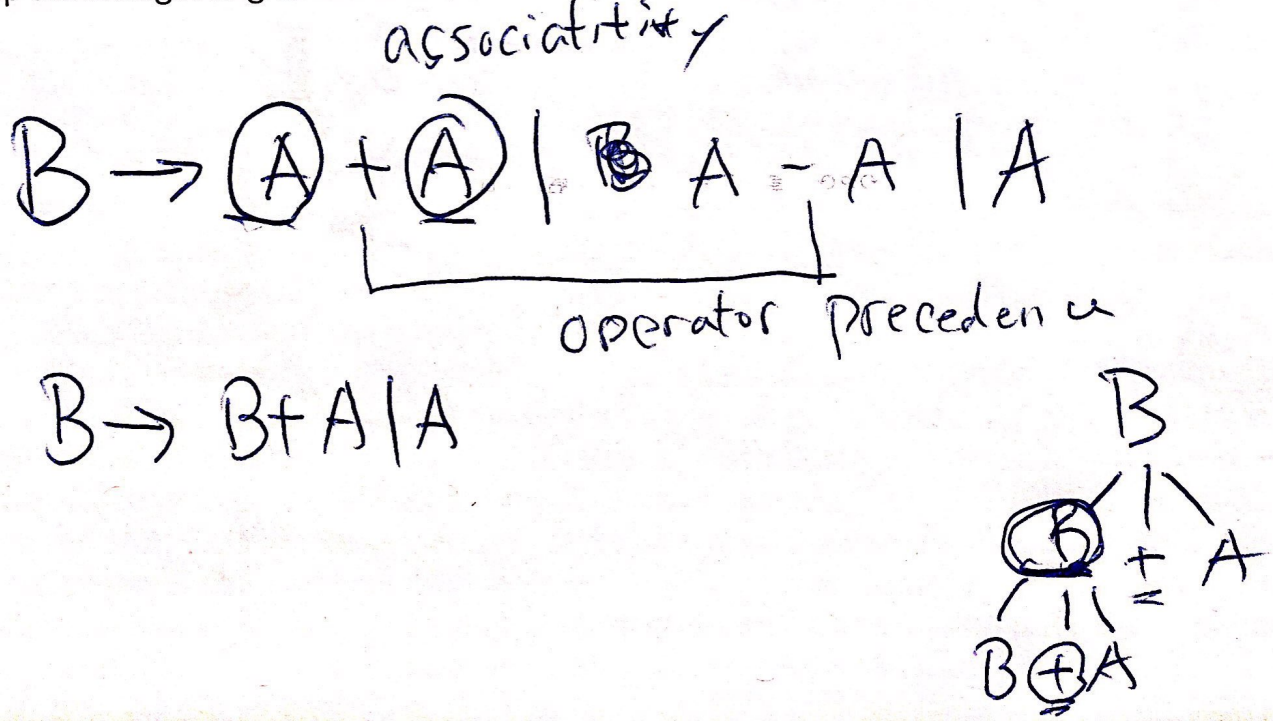
- bad
- can generate different parse trees for the same tree - thus has different meanings for the same string

# Approaches for eliminating ambiguity

Modify grammar rules:

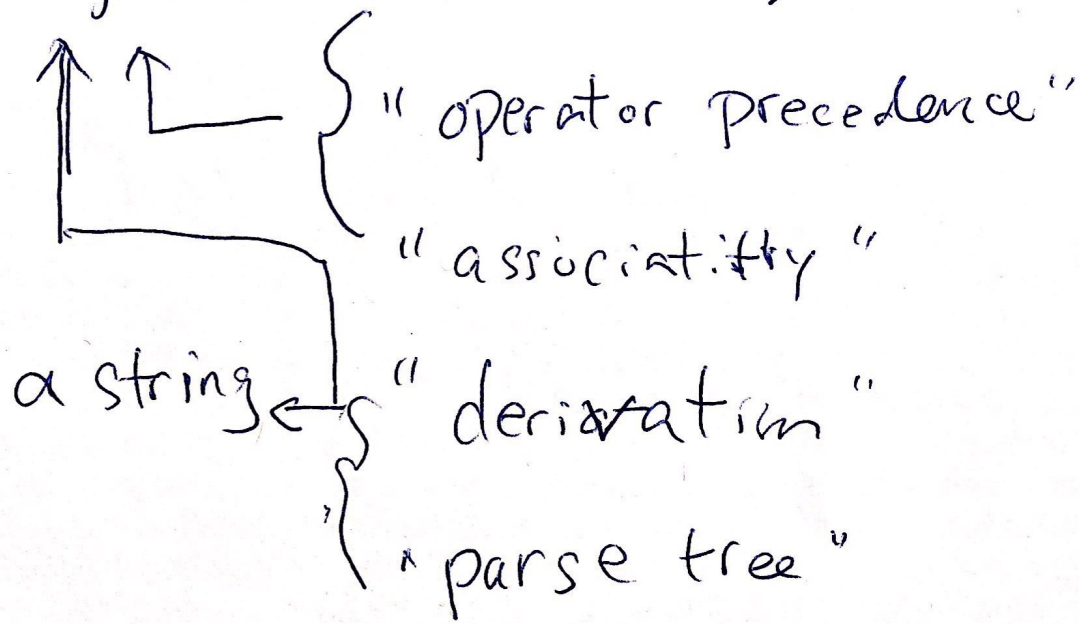
- Delimiters (e.g., parenthesis) – need to add terminals
- Precedence for operators (intuitively, the operators that have higher priorities should be located in the lower part of the parse tree, and thus further from the start symbol in the grammar rule)
- Associativity (the current operand should compute with the left or right operand, grammar: allow expansion for the left side, thus the left hand operand should repeat the non-terminal on the right hand side)

You can also keep the ambiguous grammar rules, but add additional rules



Types of problems

① Ask about a grammar "Is it ambigious"




② Create a grammar

ArithLang

# Grammar

- Prefix
- Contains only numbers and arithmetic operators

# Interpreter

- Reader: from a program to a AST, editing .g and automatically generate .java files
- Evaluator: traverse AST to generate values, 1) extending value types if needed 2) extending AST classes to store new nodes 3) update visitor class to compute values
- Printer

# VarLang and DefineLang

# Variables

- Definition and use of a variable
- Scoping
- Free/bound variables
- Environment: pass the right value to the expression; global variable will be defined in the initial environment and be visible entire interpreter life time

FuncLang

# Functions

- Lambda Expression: lambda expression is a function, it has values, and can be passed as parameters, returns from a function and stored in the environment
- Call
- Function with a Name: Combine lambda expression with let and define expressions
- List and pair
- Built-in functions for list: car, cdr, cons, null?
- Recursive calls
- High order functions
- Control Structure: if then else
- Currying

Questions?