Lecture 6:

- Intro to Grammars

Announcements:

- HW-1 out

Formal Grammars

A set of (declarative) rules that define a language's syntax (grammar)

- a "language" here broadly means a set of allowable strings $\{s_1, s_2, \ldots\}$
- for this class, the set of allowable programs in a programming language

In PL implementation, grammars can be used within

- Lexers (lexical analysis) — e.g., check numbers, strings, comments
- Parsers (syntax analysis) — check if syntax is correct

Different "classes" of grammars

- "regular" grammars specify regular languages (think regular expressions)
- "context free" grammars specify context-free languages (most PLs)
- ... and so on
- we'll just cover the basics of regular and context-free grammars

Aside: Grammars are closely tied to computation ...

- If view language of a grammar as output of a program (e.g., binary numbers)
- A grammar exactly "computes" the output
- Here, computing is performed by applying grammar rules to *derive* outputs
- We'll see examples of derivations later

Grammar Rules

Grammar rules define productions (aka rewritings)

$S \rightarrow \texttt{a}$

Here we say S produces (or yields) a

- S is a **non-terminal** symbol (LHS of a rule) ... sometimes as <s>
- a is a **terminal** symbol ... a part of a string
- terminal and non-terminal symbols are disjoint
- set of terminals is the **alphabet** of the language
- often a distinguished **start** symbol

Rules can be applied to create a **derivation** of a string

- from start, repeatedly apply rules until only terminals remain
- we'll see examples soon

Concatenation

$S \to \texttt{ab}$

This says S yields the string consisting of $\texttt{a}$ followed by $\texttt{b}$

There can be many ways to define the same language

$T \to UV$

$U \to \texttt{a}$

$V \to \texttt{b}$

Here T yields the same exact language as S (i.e., $\{\texttt{ab}\}$)

Alternation

$S \to \texttt{a} \mid \texttt{b}$

This says S yields the string $\texttt{a}$ <u>or</u> $\texttt{b}$

The | symbol is special (meta) syntax for separate S-rules:

$S \to \texttt{a}$

$S \to \texttt{b}$

This says S yields the string $\texttt{a}$ or S yields the string $\texttt{b}$

The empty string

$S \rightarrow \texttt{a} \mid \epsilon$

Where ϵ denotes the special "empty" terminal

This example says S yields either the string **a** or **""** (empty string)

Check In: How can this rule be rewritten to not contain alternation?

Kleene Star (Closure)

$S \rightarrow \texttt{a}^*$

This says S yields the strings with zero or more **a**'s

Namely, the strings: **""**, **a**, **aa**, **aaa**, . . ., and in general $\texttt{a}^n$ for $n \geq 0$

Combining Kleene star and concatenation:

$S \rightarrow \texttt{a}^*\texttt{b}^*$

This says S yields strings with zero or more **a**'s followed by zero or more **b**'s

This is the strings: **""**, **a**, **b**, **aa**, **ab**, **bb**, and in general $\texttt{a}^n\texttt{b}^m$ for $n, m \geq 0$

Check In: What is the language defined by the following rule?

$S \rightarrow \texttt{a}^* \mid \texttt{b}^*$

Summary – Things to Know

1. Basic grammar rules and terminology (terminals, non-terminals, etc.)

2. Concatenation, Alternation, Empty String, and Kleene Star