Ex C2 Part 1 "CF"

A. What is the most important property of "context-freeness"?
B. Why the constraint |LHS| = 1 leads to a CFG (cf. unrestricted grammars)?
C. What kind of strings cannot be specified by CFGs?
D. What are the differences between top-down and bottom-up parsing?
E. Why can PDAs capture all of CFLs?
F. Why is the class of DCFLs smaller than that of CFLs?

Review

Languages/Grammars/Automata

- Language: Set of strings ~ problems
- Grammar: Convenient way of specifying languages
  - E.g., $L(G)$ for a CFG G
- Automata: Abstract model of processing languages
  - E.g., $L(G)$, where G is a CFG, needs a PDA
- Chomsky hierarchy: about languages

Ex C2 Part 2 Graphing Tool

- Expression is a series of Terms connected with + or -
  - Exp → Term Exp → Term + Exp Exp → Term - Exp
  - Note: Exp → ε
- Term is a series of Factors connected with * or /
  - Term → Factor Term → Factor * Term Term → Factor / Term
- Factor is Const, Factorial, Log, or Power
  - Factor → Const | Factorial | Log | Power
- Const
  - Const → Digits Const → Digits "." Digits Digits → Digit
  - Digits → Digit Digits Digit → 0 | 1 | .... | 9
- Factorial → "n!"
- Log
  - Log → "logn" | "log" Const "n" | "log(" Exp ")" | "log" Constant "(" Exp ")"
  - Note: "log" Const Const is illegal
  - Note: not including "log(" Constant ")(" Exp ")"
- Power
  - Power → ("n" | "(" Exp ")" | Const) ["^" ("n" | "(" Exp ")" | Const)]

Ex C2 Part 3 Foxtrot Question C

- Possible to represent the same language with a regular grammar?

Foxtrot	→	Basic
Basic	→	**promenadePosition naturalWeave threeStep heelTurn** Basic
Basic	→	**featherFinish** WeaveNext
WeaveNext	→	**weave** WeaveNext
WeaveNext	→	**threeStep heelTurn** [Basic]

Demo

Top-down Parsing

- Example: Recursive-descent parsing
  - Keep the current input and the remaining part of the rule on stack
  - Expand nonterminals
  - Check terminals against the input

- S → A
- A → a A b | a b

Demo

Bottom-up Parsing

- Example: Shift-reduce parsing
  - If the RHS of a rule matches a part of the input, reduce it to the LHS symbol
  - Otherwise, push the leftmost symbol onto the stack and repeat (shift)

`http://www.tcnj.edu/~komagata/ShiftReduce/`

Main CF Property

- Matching growth to the left and right
  - E.g., $\underbrace{(...(}_{n}\ \underbrace{)...)}_{n}$, 0^n1^n, $a_1...a_n a_n...a_1$
- Characterized by *balanced* rules
  - E.g., $A \rightarrow (A)$, $A \rightarrow 0\ A\ 1$, $A \rightarrow a_i\ A\ a_i$
- Characterized by the use of stack
  - E.g., pushing '(' and later popping it and matching with ')'

$0^n1^n2^k3^k$, $0^n2^k3^k1^n$, $0^i2^k3^k0^{(n-i)}1^n$

Group Exercise 1 (CF)

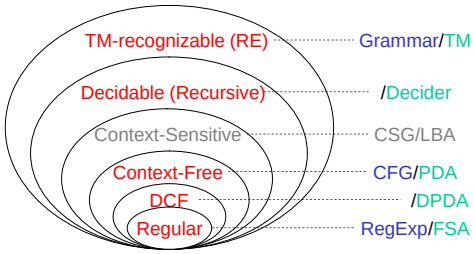
- Which of the following languages is context-free? Why?

1. $0^n1^n2^k3^k$
2. $0^n2^k3^k1^n$
3. $0^i2^k3^k0^{(n-i)}1^n$

Unit C3: Overview

- Review examples of regular languages
  - Review grammars/automata for regular languages
- Understand the effects of nondeterminism for different automata
- Preview Exercise C3 "Regular"

Extended Chomsky Hierarchy



Initial survey problem

Vending Machine

- Imagine a vending machine that accepts nickels, dimes, and quarters and sells Vodka by a paper cup for 35 cents. Draw a diagram of a finite-state machine/automaton that would represent the state of the vending machine. Make additional assumptions if necessary.

Password Screening

- Requirement 1
  - Your password must be non-TM-recognizable.
- Requirement 2
  - Your password must be a palindrome.
- Requirement 3
  - Your password must be at least 6 characters long, contain at least one uppercase letter, *and* at least one symbol *or* a numeral.

Java Comments

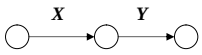
- End-of-line comment: `// ...` (until EOL)
- Traditional comment: `/* ... */`
  - Note: No nesting allowed

Binary Arithmetic

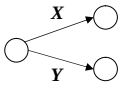
- Do we need a TM?
- Do we need a PDA?

FA Components

- Sequence



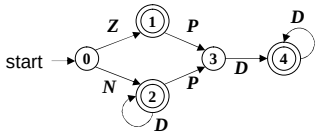
- Alternative



- Repetition (zero or more times)

Finite-State Automata (FSA)

- A machine that consists of the three components shown on the previous slide
- Example

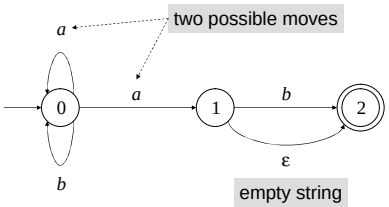


: accepting states

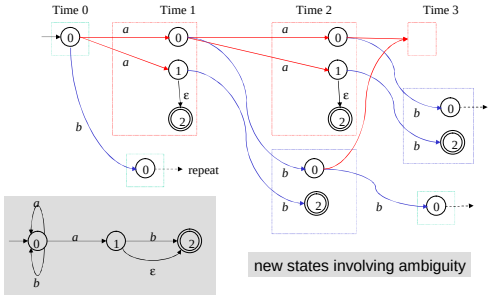
Deterministic FA (DFA)

- **M** = $(Q, \Sigma, \delta, q_0, F)$
  - Q: set of states
  - Σ: set of input symbols
  - δ: transition function
    $Q \times \Sigma \rightarrow Q$
  - q_0: initial state $\in Q$
  - F: set of final states $\subseteq Q$

Nondeterministic Example



Tracing Example 1



Observation

- Due to nondeterminism and ε-transitions, the result of a move can be a combination of possible next states.
- For a set of states Q, such a combination is a member of the power set of Q, i.e., $P(Q)$.
  - E.g., $P(\{1, 2\}) = \{\emptyset, \{1\}, \{2\}, \{1, 2\}\}$
- Naturally, any DFA is an NFA.

Nondeterministic FA (NFA)

- $\mathbf{M} = (Q, \Sigma, \delta, q_0, F)$
  - Q: set of states
  - Σ: set of input symbols
  - δ: transition function
    $Q \times (\Sigma \cup \{\varepsilon\}) \rightarrow P(Q)$ cf. transition relation
  - q_0: initial state $\in Q$
  - F: set of final states $\subseteq Q$

Nondeterminism

- Possibility of multiple transitions from a state on a single input
- Instead of a single state, considering a set of states
  - S as the set of all states
  - $P(S)$ as the power set of S
  - The next state $\in P(S)$
- Power set ~ nondeterminism
  exponential growth

Effects of Nondeterminism

- TMs: No effects on power (i.e., the set of languages that can be recognized)
- PDAs: Affects the power (i.e., PDAs ≠ DPDAs)
- FSAs: No effects on the power (i.e., the set of languages that can be recognized)
- Efficiency/speed: exponential slow down

Regular Grammar

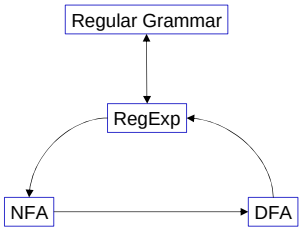
- $\mathbf{G} = (N, T, R, S)$
  - N: finite set of nonterminals [upper case]
  - T: finite set of terminals [lower case]
  - R: finite set of rules $A \rightarrow \alpha B$ or $A \rightarrow B \alpha$ where
    - α is a string made up of the elements of T
    - $B \in N$
  - S: start symbol $\in N$
- Regular languages = $\{\mathbf{L}(\mathbf{G}) \mid \mathbf{G}$ is a regular grammar$\}$

Regular Expression (RegExp)

- Practical alternative to regular grammar
- Expression components (with sets X, Y)
  - Sequence: $\boldsymbol{X\,Y}$ Notation: $X^n = \underbrace{X \ldots X}_{n}$
  - Alternative: $\boldsymbol{X} \mid \boldsymbol{Y}$
  - Repetition (zero or more times): $\boldsymbol{X}^*$ Kleene closure

  A single regular expression represents a *set*. Convention: Singleton {**a**} abbreviated as **a**
- Equivalence of RegExp and regular grammars

Equivalence



Main *Regular* Property

- Language: Uncoordinated repetition
  - E.g., 0^m1^n, "a very very .... long noun phrase"
- Grammar: Characterized by recursion *at edge*
  - E.g., $A \rightarrow a\ A$, $A \rightarrow A\ a$
  - Equivalently, Kleene closure, X^*
- Automaton: Characterized by finite states
  - E.g., looping represented by a cycle

More Examples

- FP numbers
- URL
- Spanish spelling (cf. English spelling)
- Noun phrase in English (except for CF components)
- Musical code sequence (?)
- Your daily tasks

Unit Summary

- Main property of regular languages: *uncoordinated* repetition, cf. CF
- RegExp is a practical way of specifying a broad range of simple languages.
- FA is a simple and fast model for processing regular languages.

Summary Question

- How far have you been thinking Exercise C5 (Comprehensive Exercise)? Explain. If not yet, what is your current thought?
- Questions/Comments/Suggestions