HKN CS/ECE 374 Midterm 1 Review

Nathan Bleier and Mahir Morshed

For the most part, all about strings!

- String induction (to some extent)
- Regular languages
  - Regular expressions (regexps)
  - Deterministic finite automata (DFAs)
  - Nondeterministic finite automata (NFAs)
- Context-free languages
  - Context-free grammars (CFGs)

Some definitions – regular languages

- One of the following (where A and B are both regular languages):
  - ∅
  - *w* (a string)
  - A^*
  - $A \cup B$
  - AB
  - $A \cap B$
  - $A - B$
  - A^R
  - (Inverse) homomorphisms of A

Some definitions – regular languages (cont.)

- String homomorphism:
  - Given a mapping, *f*, from an alphabet to a language (mapping characters to strings), we create a string homomorphism, *F*, by mapping every character in a string through *f*
  - Example:
    - *f(0) = abc, f(1) = 123, then F(101) = 123abc123*
  - A string homomorphism acting on a regular (context-free) language results in a regular (context-free) language
  - An inverse homomorphism acting on a regular (context-free) language results in a regular (context-free) language

Some definitions - DFA

- Formally defined by Σ (alphabet), Q (states), δ (transitions), s (start), A (acceptors)
  - Σ must necessarily be a finite alphabet
  - Q represented by circles
  - $\delta : Q \times \Sigma \rightarrow Q$ represented by arrows between circles
  - $s \in Q$ represented by source-less arrow entering exactly one state in DFA
  - $A \subseteq Q$ represented by doubly-circled states anywhere in the diagram
- Accepts/recognizes language L iff it accepts all strings in L and rejects all strings not in L

Some definitions – NFA

- Formally defined by Σ (alphabet), Q (states), δ (transitions), s (start), A (acceptors)
  - Σ must necessarily be a finite alphabet
  - Q represented by circles
  - $\delta : Q \times \Sigma \cup \{\varepsilon\} \rightarrow Q$ represented by arrows between circles
  - $s \in Q$ represented by source-less arrow entering exactly one state in NFA
  - $A \subseteq Q$ represented by doubly-circled states anywhere in the diagram
- Accepts/recognizes language L iff it accepts all strings in L and rejects all strings not in L
- DFAs ⇔ NFAs, DFAs ⇔ regexps, regexps ⇔ NFAs *all* possible
  - How best to do these transformations?

A Useful Tool: Myhill-Nerode theorem and Fooling Sets

- We want to prove a language, L, is not regular.
- L is not regular iff there is no DFA which decides L.
- Approach: If we can show that any automaton which decides L has infinitely many states, then we have proven that L is not regular.
- Tool: Fooling set
- Let x, y be strings such that there exists a suffix z such that ($xz \in L$ and $yz \notin L$) OR ($xz \notin L$ and $yz \in L$)
  - For any potential DFA which decides L, xz and yz terminate in different states (one accepting, one rejecting)
  - Thus for any potential DFA which decides L, x and y must terminate in different states
  - Thus there are at least two states in any potential DFA
- Now we must show that there exist infinitely many such prefixes, all of which have distinguishing suffixes for one another

Another Useful Tool: Arden's Theorem

- It is often easier to come up with a DFA for a regular language than an RE.
- We can use the following theorem to help us convert a DFA into an equivalent RE.
- If P and Q are two RE's over the same alphabet, and if L(P) does not contain the empty string, then the recursively defined equation R = Q + RP has the unique solution

$$R = QP^*$$

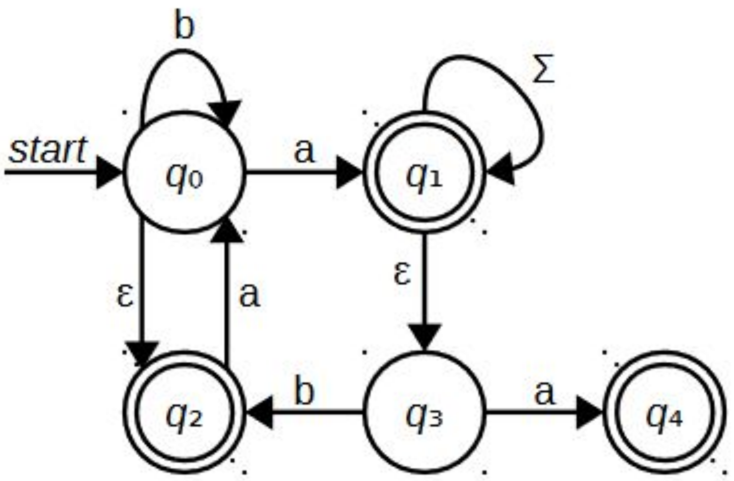
Some definitions – CFG

- <Whereas DFAs/NFAs *decide* certain languages, CFGs *generate* them>
- Formally defined by Σ (terminals), Γ (non-terminals), R (production rules), S (start)
  - Closed under union?
  - Closed under intersection?
  - Closed under concatenation?
  - Closed under set difference?
  - Closed under Kleene star?
- Design techniques...

$$scramble(w) := \begin{cases} w & \text{if } |w| \leq 1 \\ ba \bullet scramble(x) & \text{if } w = abx \text{ for some } a, b \in \Sigma \text{ and } x \in \Sigma^* \end{cases}$$

Prove that scramble(scramble(x)) = x ∀ strings x.

Convert this NFA to a DFA and a regexp:



Construct a CFG for the following language:

$L = \{x \subseteq \{0,1\}^*$

such that

$|x| \geq 2$

and

the symbol at position *i* is the same as that at $i+2\}$

Construct CFGs for the following languages:

$L = \{a^n b^m c^m d^{2n} \mid n \geq 0 , m > 0\}$

$L = \{0^i 1^j 2^k \mid i \neq j \text{ or } j \neq k\}$

Consider a language L ⊆ {0,1}*. What's *definitely* true about this language?

- Is L non-empty?
- Is L infinite?
- Does L contain the empty string?
- Is it the case that $\overline{L}$ is regular if L is the union of two regular languages?
- Is it the case that $\overline{L}$ is context-free if L is the union of two regular languages?
- Is it the case that L is context-free if L is finite?
- Is it the case that L is accepted by a DFA iff L is accepted by a NFA?
- Is it the case that L is accepted by a 42-state DFA iff L is accepted by a 42-state NFA?

The language $\{0^{|w|} \mid w \in L\}$ is regular for every regular language L.

The language $\{0^{|w|} \mid w \in L\}$ is non-regular for every non-regular language L.

For every language $L \subseteq \{0,1\}^*$, if L contains all but a finite number of strings in 0^*, then L is regular.

If L is regular and $L \cap L'$ is not regular, then L' is not regular.

The specific language

$\{w \in \{0,1\}^* \mid w$ represents an integer divisible by 374 in ternary$\}$

is regular.

For all regular languages L, each equivalence class of $\equiv_L$ is a regular language.

Let $L = \{0^i 1^j 0^k \mid 2i = k \text{ or } i = 2k\}$

1. Prove L is not regular
2. Describe a CFG for L