BU CS 332 – Theory of Computation

Lecture 5:

- More on pumping
- Regular expressions
- Regular expressions = regular languages

Reading:
Sipser Ch 1.3

Mark Bun
February 5, 2020
More on Pumping
Pumping Lemma (Formal)

Let $L$ be a regular language.

Then there exists a “pumping length” $p$ such that

For every $w \in L$ where $|w| \geq p$, $w$ can be split into three parts $w = xyz$ where:

1. $|y| > 0$
2. $|xy| \leq p$
3. $xy^iz \in L$ for all $i \geq 0$
General Strategy for proving $L$ is not regular

Proof by contradiction: assume $L$ is regular. Then there is a pumping length $p$.

1. Find $w \in L$ with $|w| \geq p$

2. Show that $w$ cannot be pumped

3. Conclude $L$ must not have been regular
Claim: $L = \{0^i 1^j \mid i > j \geq 0\}$ is not regular

Proof: Assume $L$ is regular with pumping length $p$

1. Find $w \in L$ with $|w| \geq p$

2. Show that $w$ cannot be pumped
   Formally, if $w = xyz$ with $|xy| \leq p$, then...
Pumping a language can be lots of work... 
Let’s try to reuse that work!

How might we show that 
\[ BALANCED = \{w \mid w \text{ has an equal } \# \text{ of 0s and 1s} \}\] 
is not regular?

\[ \{0^n1^n \mid n \geq 0\} = BALANCED \cap \{w \mid \text{ all 0s in } w \text{ appear before all 1s} \} \]
Using Closure Properties

If $A$ is not regular, we can show a related language $B$ is not regular

By contradiction: If $B$ is regular, then $B \cap C (= A)$ is regular.

But $A$ is not regular so neither is $B$!
Example

Prove $B = \{0^i 1^j | i \neq j\}$ is not regular using nonregular language $A = \{0^n 1^n | n \geq 0\}$ and regular language $C = \{w \mid \text{all } 0\text{s in } w \text{ appear before all } 1\text{s}\}$
Regular Expressions
Regular Expressions

• A different way of describing regular languages
• A regular expression expresses a (possibly complex) language by combining simple languages using the regular operations

“Simple” languages: $\emptyset, \{\varepsilon\}, \{a\}$ for some $a \in \Sigma$

Regular operations:

- **Union:** $A \cup B$
- **Concatenation:** $A \circ B = \{ab | a \in A, b \in B\}$
- **Star:** $A^* = \{a_1 a_2 \ldots a_n | n \geq 0 \text{ and } a_i \in A\}$
Regular Expressions – Syntax

A regular expression $R$ is defined recursively using the following rules:

1. $\varepsilon$, $\emptyset$, and $a$ are regular expressions for every $a \in \Sigma$

2. If $R_1$ and $R_2$ are regular expressions, then so are $(R_1 \cup R_2)$, $(R_1 \circ R_2)$, and $(R_1^*)$

Examples: (over $\Sigma = \{a, b, c\}$)

$(a \circ b)$  
$(((a \circ (b^*)) \circ c) \cup (((a^*) \circ b))*)$  
$(\emptyset^*)$
Regular Expressions – Semantics

$L(R)$ = the language a regular expression describes

1. $L(\emptyset) = \emptyset$
2. $L(\varepsilon) = \{\varepsilon\}$
3. $L(a) = \{a\}$ for every $a \in \Sigma$
4. $L((R_1 \cup R_2)) = L(R_1) \cup L(R_2)$
5. $L((R_1 \circ R_2)) = L(R_1) \circ L(R_2)$
6. $L((R_1^*)) = (L(R_1))^*$

Example: $L((((a^*) \circ (b^*)))) =$
Simplifying Notation

• Omit $\circ$ symbol: $(ab) = (a \circ b)$

• Omit many parentheses, since union and concatenation are associative:

\[
(a \cup b \cup c) = (a \cup (b \cup c)) = (((a \cup b) \cup c)
\]

• Order of operations: Evaluate star, then concatenation, then union

\[
ab^* \cup c = (a(b^*)) \cup c
\]
Examples

Let $\Sigma = \{0, 1\}$

1. $\{w | w \text{ contains exactly one } 1\}$

2. $\{w | w \text{ has length at least } 3 \text{ and its third symbol is } 0\}$

3. $\{w | \text{every odd position of } w \text{ is } 1\}$
Syntactic Sugar

• For alphabet \( \Sigma \), the regex \( \Sigma \) represents \( L(\Sigma) = \Sigma \)

• For regex \( R \), the regex \( R^+ = RR^* \)

Not captured by regular expressions: Backreferences
Equivalence of Regular Expressions, NFAs, and DFAs
Regular Expressions Describe Regular Languages

**Theorem:** A language $A$ is regular if and only if it is described by a regular expression

**Theorem 1:** Every regular expression has an equivalent NFA

**Theorem 2:** Every NFA has an equivalent regular expression
Regular expression -> NFA

**Theorem 1:** Every regex has an equivalent NFA

**Proof:** Induction on size of a regex

**Base cases:**

\[ R = \emptyset \]

\[ R = \varepsilon \]

\[ R = a \]
Regular expression -> NFA

Theorem 1: Every regex has an equivalent NFA
Proof: Induction on size of a regex

Inductive step:

\[ R = (R_1 \cup R_2) \]

\[ R = (R_1 R_2) \]

\[ R = (R_1^*) \]
Example

Convert \((1(0 \cup 1))^*\) to an NFA
NFA -> Regular expression

Theorem 2: Every NFA has an equivalent regex

Proof idea: Simplify NFA by “ripping out” states one at a time and replacing with regexes
Generalized NFAs

• Every transition is labeled by a regex
• One start state with only outgoing transitions
• Only one accept state with only incoming transitions
• Start state and accept state are distinct
Generalized NFA Example

\[ R(q_s, q) = \]
\[ R(q_a, q) = \]
\[ R(q, q_s) = \]
NFA -> Regular expression

\[ k \] states

\[ k + 2 \] states

\[ k + 1 \] states

2 states
NFA -> GNFA

- Add a new start state with no incoming arrows.
- Make a unique accept state with no outgoing arrows.
GNFA -> Regular expression

Idea: While the machine has more than 2 states, rip one out and relabel the arrows with regexes to account for the missing state
GNFA -> Regular expression

**Idea:** While the machine has more than 2 states, rip one out and relabel the arrows with regexes to account for the missing state.
GNFA -> Regular expression

Idea: While the machine has more than 2 states, rip one out and relabel the arrows with regexes to account for the missing state
GNFA -> Regular expression

Idea: While the machine has more than 2 states, rip one out and relabel the arrows with regexes to account for the missing state
Example