CS/ECE 374 B Fall 2019
Homework 3 Problem 1

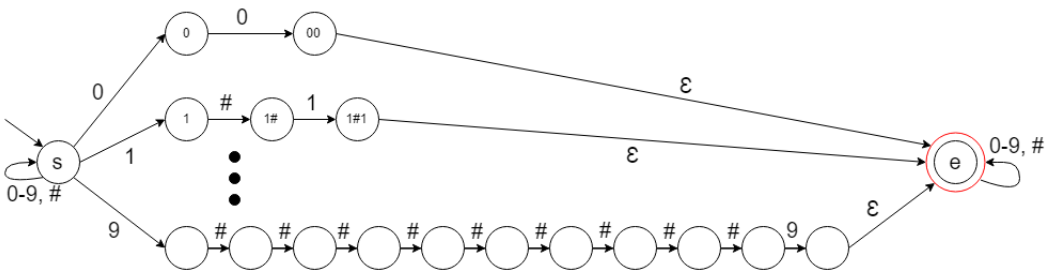
Siyuan Chen (siyuanc2)
Megan Walsh (meganew2)
Jiashuo Tong (jtong8)

Solution:

(a) Regular

Lemma 4.2. Every regular language is accepted by a nondeterministic finite-state automaton.

According to the above lemma, the proposed language is regular because it is accepted by the following NFA:



In this NFA, the start state is s and the accepting state is e. Any combination of characters in the alphabet {0,...,9, #} are accepted in either of these states.

The substring in question is $c\#^c c$. The completed, expanded NFA would consist of a path from s to e for each value of c in 0-9, for a total of 10 parallel paths. Each path would have c+2 states. As there is a finite number of options for c, an automata can be constructed with a finite number of states.

(b) Non-regular

If the language has an infinite fooling set, then the language is not regular.

Consider the infinite set F = $\{10^n \mid n \geq 0\}$, or more simply F = 10^*.
Let x and y be arbitrary distinct strings in F.

The definition of F implies $x = 10^i$ and $y = 10^j$ for some integers i ≠ j.
The suffix z = $\#^{(10)^i} = \#^{1Ei}$ distinguishes x and y, because

$$xz = 10^i\#^{1Ei} \in L$$
$$yz = 10^i\#^{1Ej} \notin L$$

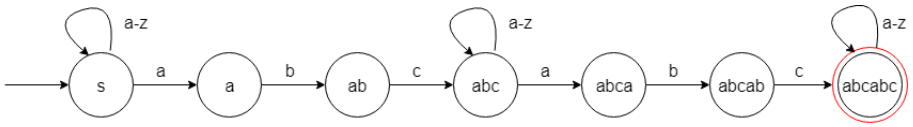
Thus, every pair of distinct strings in F has a distinguishing suffix.
In other words, F is a fooling set for L.
Because F is infinite, L cannot be regular

(c) Regular

Lemma 4.2. Every regular language is accepted by a nondeterministic finite-state automaton.

According to the above lemma, the proposed language is regular.

In designing the NFA that accepts this language, there are $26^3 = 17{,}576$ variations of combinations of 3 characters. Take an arbitrary combination of 3 characters, *abc*. A language containing strings over the alphabet {a,...z} with *abc* repeated in two places would look like this:



This NFA uses *s* as the start state and *abcabc* as the accepting state, allowing any combination of the alphabet before, in between, or after each *abc*.

An NFA like this can be constructed for each variation in the 17,576 possible variations of 3 characters. These smaller NFAs can be combined to form one large NFA, therefore proving the language is regular.

More formally, the language can be represented by the following regular expression:

$$a^*(\sum_{x\in\Sigma^3} x)a^*(\sum_{x\in\Sigma^3} x)a^*$$

Where $a \in \Sigma = \{a, b, ..., z\}$ is arbitrary character and $x \in \Sigma^3$ is any 3 character string. Since the language can be described by a (long) regular expression, the language must be regular.

■

CS/ECE 374 B Fall 2019
Homework 3 Problem 2

Siyuan Chen (siyuanc2)
Megan Walsh (meganew2)
Jiashuo Tong (jtong8)

Solution: Given a DFA $M(\Sigma, Q_1, \delta_1, q_s, A_1)$ that accepts L, It is desired to construct an NFA N that accepts $f(L)$. The construction is given as follows:
First, for each transition in M: $\delta_1(q_i, a) = q_j$, where $q_i, q_j \in Q_1$ are arbitrary states in M and $a \in \Sigma_1$ is an arbitrary character, construct the following NFA N_{ij} $(Q_{ij}, Q_{ij}, \delta_{ij}, s_{ij}, A_{ij})$:

$$\begin{aligned}
\Sigma_{ij} &= \Sigma_2; \\
Q_{ij} &= \{q_{s,ij}, q_{a,ij}\} \cup \{q_{k,ij} \text{ for } k \text{ in range}(1, |f(a)|+1)\}; \\
s_{ij} &= q_{s,ij}; \\
A_{ij} &= \{q_{a,ij}\};
\end{aligned}$$

δ_{ij} is defined as follows:

$$\begin{aligned}
&\text{If } f(a) \neq \epsilon: \\
\delta_{ij}(q_{s,ij}, f(a)_1) &= \{q_{1,ij}\}; \\
\delta_{ij}(q_{k-1,ij}, f(a)_k) &= \{q_{k,ij}\} \text{ for } k \text{ in range}(2, |f(a)|+1); \\
\delta_{ij}(q_{|f(a)|,ij}, \epsilon) &= \{q_{a,ij}\}; \\
&\text{If } f(a) = \epsilon: \\
\delta_{ij}(q_{s,ij}, a) &= \{q_{a,ij}\}.
\end{aligned}$$

Here $f(a)_i$ denotes the i-th character (1-based) in the string $f(a)$.
NFA N_{ij} accepts the string $f(a)$, which is described in Lemma 1:

Lemma 1: $\delta^*_{ij}(q_{s,ij}, f(a)) \cap A_{ij} \neq \emptyset$ for any $a \in \Sigma_1$ and $q_i, q_j \in Q_1$ such that $\delta_1(q_i, a) = q_j$.
Proof: Assume that for all strings $x \in \Sigma_2$ that $0 < |x| < |f(a)|$, $\delta^*_{ij}(q_{s,ij}, x) = \{q_{|x|,ij}\}$. There are 3 cases to consider:

(a) $x = \epsilon$: (In this case $|x| = 0$, so I.H. does not apply)
From definition of δ_{ij}, $\delta_{ij}(q_{s,ij}, x) = \{q_{a,ij}\}$. Then $\delta^*_{ij}(q_{s,ij}, x) \cap A_{ij} \neq \emptyset$.

(b) $x = b$ where $b \in \Sigma_2$ is an arbitrary character: $(|x| = 1)$
From definition of δ_{ij}, $\delta^*_{ij}(q_{s,ij}, b) = \delta_{ij}(q_{s,ij}, b) = \{q_{|b|,ij}\} = \{q_{1,ij}\}$ and $\delta_{ij}(q_{1,ij}, \epsilon) = \{q_{a,ij}\}$.
Then $\delta^*_{ij}(q_{s,ij}, b) = \{q_{a,ij}\}$ and $\delta^*_{ij}(q_{s,ij}, x) \cap A_{ij} \neq \emptyset$.

(c) $x = y\,b$ where $b \in \Sigma_2$ and $y \in \Sigma_2^*$: $(|x| > 1)$
From I.H., $\delta^*_{ij}(q_{s,ij}, y) = \{q_{|y|,ij}\}$.
Also from definition of δ_{ij}, $\delta_{ij}(q_{|y|,ij}, b) = \{q_{|x|,ij}\}$ and $\delta_{ij}(q_{|x|,ij}, \epsilon) = \{q_{a,ij}\}$.
Then $\delta^*_{ij}(q_{s,ij}, y \cdot b) = \{q_{a,ij}\}$ and $\delta^*_{ij}(q_{s,ij}, x) \cap A_{ij} \neq \emptyset$.

Then we can proceed to define the NFA N $(\Sigma_2, Q_2, \delta_2, s_2, A_2)$:

$$\Sigma_2 = \Sigma_2;$$
$$Q_2 = \{q_s', q_a'\} \cup Q_1 \cup Q_{ij} \text{ for all } N_{ij} \text{ constructed above;}$$
$$s_2 = q_s';$$
$$A_2 = q_a';$$

δ_2 is defined as follows:

$$\delta_2(q_s', \epsilon) = \{q_s\};$$
$$\delta_2(q_a, \epsilon) = \{q_a'\} \text{ for all } q_a \in A_1;$$

For each NFA N_{ij}, from Lemma 1, δ_2 inherits all rules from δ_{ij}, such that:

$$q_{a,ij} \in \delta_2^*(q_{s,ij}, f(a));$$

Additionally, for each NFA N_ij:

$$\delta_2(q_i, \epsilon) = \{q_{s,ij}\};$$
$$\delta_2(q_{a,ij}, \epsilon) = \{q_j\}.$$

In other words:

$$q_j \in \delta_2^*(q_i, f(a)) \text{ for each transition } \delta_1(q_i, a) = q_j.$$

Claim 1: For all $w \in L(M)$, $f(w) \in L(N)$.
In other words, for all $w \in \Sigma_1$ such that $\delta_1^*(q_s, w) \in A_1$, prove that $\delta_2^*(q_s', f(w)) \cap A_2 \neq \emptyset$.
Assume that for all strings $x \in \Sigma_1$ that $|x| < |w|$, $\delta_1^*(q_s, x) = p$ for some $p \in Q_1$, then $p \in \delta_2^*(q_s', f(x))$.
There are 3 cases to consider:

(a) $w = \epsilon$:
If $w = \epsilon$ and $w \in L(M)$, $q_s \in A_1$.
From definition of δ_2, $\delta_2(q_s', \epsilon) = \{q_s\}$. Also since $q_s \in A_1$, $\delta_2(q_s, \epsilon) = \{q_a'\}$.
From definition of f, $f(\epsilon) = \epsilon$. Then $\delta_2^*(q_s', f(w)) = \delta_2^*(q_s', \epsilon) = \cup_{r \in \delta_2(q_s', \epsilon)} \delta_2(r, \epsilon) = \{q_a'\}$ and $\{q_a'\} \cap A_2 \neq \emptyset$.

(b) $w = a$ where $a \in \Sigma_1$ is an arbitrary character:
If $w = a$ and $w \in L(M)$, $\exists p \in A_1$ s.t. $\delta_1(q_s, a) = p$.
From definition of δ_2, $\delta_2(q_s', \epsilon) = \{q_s\}$, $\delta_2^*(q_s, f(a)) = \{p\}$, and $\delta_2(p, \epsilon) = \{q_a'\}$.
From the above three equations and definition of δ^* for NFA, $\delta_2^*(q_s', f(a)) = \{q_a'\}$, $\delta_2(p, \epsilon) = \{q_a'\}$, and $\{q_a'\} \cap A_2 \neq \emptyset$.

(c) $w = ax$ where $a \in \Sigma_1$ and $x \in \Sigma_1^*$:
If $w = xa$ and $w \in L(M)$, $\exists p \in Q_1$ and $q \in A_1$ s.t. $\delta_1^*(q_s, a) = p$ and $\delta_1(p, x) = q$.
From definition of δ_2, $p \in \delta_2^*(q_s', f(a))$. From I.H., $q \in \delta_2^*(p, f(x))$.
Therefore, $q \in \cup_{r \in \delta_2^*(q_s', f(a))} \delta_2^*(r, f(x)) = \delta_2^*(q_s', f(a) \cdot f(x))$. Because $f(a) \cdot f(x) = f(ax) = f(w)$, $q \in \delta_2^*(q_s', f(w))$.
Since $q \in A_1$, $\delta_2(q, \epsilon) = \{q_a'\}$.
Therefore, $q_a' \in \delta_2^*(q_s', f(w))$ and $\delta_2^*(q_s', f(w)) \cap A_2 \neq \emptyset$.

In all cases we conclude that $\delta_2^*(q_s', f(w)) \cap A_2 \neq \emptyset$ for all $w \in \Sigma_1$ such that $\delta_1^*(q_s, w) \in A_1$, or in common language, NFA N accepts $f(L)$.

Claim 2: For all $f(w) \in L(N)$, $w \in L(M)$.
Assume that for all strings $x \in \Sigma_1$ that $|x| < |w|$, $p \in \delta_2^*(q_s', f(x))$ for some $p \in Q_1$, then $\delta_1^*(q_s, x) = p$.
There are 3 cases to consider:

(a) $w = \epsilon$:
From definition of $f(w)$, $f(w) = f(\epsilon) = \epsilon$. Since $f(w) \in L(N)$, from definition of δ_2, $\exists p \in \delta_2^*(q_s', \epsilon)$ such that $p \in A_1$. Since DFA M does not permit epsilon-transition, p must be q_s and because $p \in A_1$, $q_s \in A_1$ which means DFA M accepts empty strings, or $w \in L(M)$.

(b) $w = a$ where $a \in \Sigma_1$ is an arbitrary character:
Since $f(a) \in L(N)$, $\exists p \in A_1$ such that $p \in \delta_2^*(q_s', f(a)) = \cup_{r in \delta_2(q_s', \epsilon)} \delta_2^*(r, f(a))$ from definition of δ_2.
As $\delta_2(q_s', \epsilon) = \{q_s\}$, there is only one possible r which is q_s. From definition of δ_2, for each $q_i, q_j \in Q_1$, $q_j \in \delta_2^*(q_i, f(a))$ indicates that $\delta_1(q_i, a) = q_j$. Here $p \in \delta_2^*(q_s, f(a))$, which means $\delta_1(q_s, a) = p \in A_1$ or or $w \in L(M)$.

(c) $w = ax$ where $a \in \Sigma_1$ and $x \in \Sigma_1^*$:
Since $f(ax) \in L(N)$, $\exists p \in A_1$ such that $p \in \delta_2^*(q_s', f(ax))$ from definition of δ_2. As $\delta_2(q_s', \epsilon) = \{q_s\}$, the above statement can be rewritten as $p \in \delta_2^*(q_s, f(ax))$.
As $f(ax) = f(a) \cdot f(x)$, $p \in \cup_{r in \delta_2^*(q_s, f(a))} \delta_2^*(r, f(x))$.
From definition of δ_2, $\exists q \in Q_1$ such that $q \in \delta_2^*(q_s, f(a))$ and $p \in \delta_2^*(q, f(x))$.
From I.H., $\delta_1^*(q, x) = p$. From definition of δ_2, $\delta_1^*(q_s, a) = q$.
Therefore $\delta_1^*(q_s, ax) = \delta_1^*(\delta_1^*(q_s, a), x) = p \in A_1$, which indicates $w \in L(M)$.

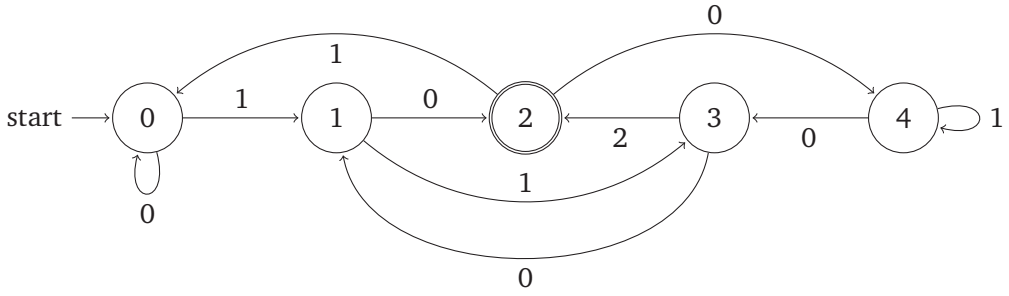
In all cases we conclude that $w \in L(M)$ for all $f(w) \in L(N)$. Combining with proof for Claim 1 we conclude that $f(w) \in L(N)$ if and only if $w \in L(M)$, or NFA N accepts only $f(L)$. ■

CS/ECE 374 B Fall 2019
Homework 3 Problem 3

Siyuan Chen (siyuanc2)
Megan Walsh (meganew2)
Jiashuo Tong (jtong8)

Solution:

(a) The DFA that accpets binary strings that have remainder of 2 when divided by 5 is



$S \to 0S \mid 1A$	All binary strings that have remainder of $0 + \epsilon$
$A \to 0B \mid 1C$	All binary strings that have remainder of 1
$B \to 1S \mid 0D \mid \epsilon$	All binary strings that have remainder of 2
$C \to 1B \mid 0A$	All binary strings that have remainder of 3
$D \to 0C \mid 1D$	All binary strings that have remainder of 4

The DFA is constructed using transfer function $\delta(p, 0) = ((2p) \bmod 5)$ and $\delta(p, 1) = ((2p + 1) \bmod 5)$ and accepts all strings that have a remainder of 2 when divided by 5. The CFG, constructed by closely following the transformation defined in the DFA, accepts the same strings.

(b) Strings over the over the alphabet $\{0, 1\}$ that have two blocks of 0's of equal length.

$S \to ZIMIZ \mid ZIM \mid MIZ \mid M$	All strings with two blocks of 0's of equal length
$M \to E \mid 0M0$	Extends E, forming two equal blocks of 0's at both ends
$E \to 0I0 \mid 0IZI0$	Strings that have exactly one 0 on both end
$I \to 1 \mid 1I$	Arbitrary number of 1's
$Z \to \epsilon \mid 0Z \mid 1Z$	Arbitrary string of 0 and 1's

Production E generates strings that have exactly one 0 on its both ends, separated by at least one 1. It is extended by production M that forms equal length 0-blocks on its both ends. Finally, production S concatenates arbitrary strings around M and ensures the 0-blocks aren't accidentally extended by ensuring that when there is extra strings, there is at least one 1 between the 0-block and arbitrary strings.

(c) There is ambiguous description in the problem statement. Assuming that "decimal numbers" are "whole numbers" or "integers", the following CFG, heavily inspired by an example in Jeff's notes on page 9 of Lecture 5, is generated to represent arithmetic expressions with

minimum parentheses:

$S \to T \mid T + S$	Arithmetic expressions extensible by addition
$T \to F \mid F * T$	Summable expressions without parens
$F \to X \mid X^X \mid X^{(Y)} \mid (Y)^X \mid (Y)^{(Y)} \mid (T + S)$	Multipliable expressions without parens
$X \to 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9 \mid XX$	Integers exponentiable without parens
$Y \to F * T \mid T + S \mid X^X \mid X^{(Y)} \mid (Y)^X \mid (Y)^{(Y)}$	Factors exponentiable with parens

The CFG processes expressions in an incremental manner. $S \to T \mid T + S$ accepts summed expressions of variable number of terms. Each of the terms may contain arbitray number of factors multiplied together, which is accepted by $T \to F \mid F * T$. F accepts factors that can be safely multiplied with other factors without additional parentheses. These factors can be either integers, summed expressions with parentheses, or exponents formed between integers (X) or summed summed expressions with parentheses (Y).
If it is desired that all numbers in the expression are single digit numbers, i.e., $1 + 1$ is accepted but $11 + 12$ is not, then the production $X \to XX$ can be removed.

■