Harvard University
Computer Science 121

Problem Set 2 Solutions

Due Friday, October 1, 2010 at 1:20 PM.
Submit a single PDF (lastname+ps2 Solutions.pdf) of your solutions to cs121+ps2 Solutions@seas.harvard.edu
Late problem sets may be turned in until Monday, October 4, 2010 at 1:20 PM with a 20% penalty.
See syllabus for collaboration policy.

PROBLEM 1 (2+2+2+2 points)

Translate the following languages over $\Sigma = \{a, b\}$ from English description to regular expressions, or vice versa.
(A) $L = \{w \in \Sigma^* :$ the second letter of w is the same as the last letter, $|w| \geq 2\}$
(B) $L = \{w \in \Sigma^* :$ every a is followed by an odd number of b's$\}$
(C) $(a(a \cup b)^*b) \cup (b(a \cup b)^*a)$
(D) $(b^*abbb^*) \cup (b^*bbab^*) \cup (b^*babb^*) \cup (bbb^*)$

(A) $\Sigma a\Sigma^* a \cup \Sigma b\Sigma^* b \cup \Sigma a \cup \Sigma b$
(B) $b^*(ab(bb)^*)^*$
(C) $L = \{w \in \Sigma^* :$ the first letter of w is different from the last letter, $|w| \geq 2\}$
(D) $L = \{w \in \Sigma^* : w$ has at least two b's and at most one $a\}$

PROBLEM 2 (4+4 points)

(A) Construct a DFA for $L = \{w \in \{0,1\}^* :$ the number represented in binary notation by w is equal to 1 mod 3$\}$. That is, when we interpret w as a number written in binary and divide by 3, we get a remainder of 1. Briefly explain why your DFA works.

Note: for this problem, we ignore leading zeros, so that e.g. $0111 = 7 \in L$.

(B) Convert your DFA for L to a regular expression, using the GNFA construction described in lecture. Show the full GNFA at each step of the construction.

(A) See Figure 1. We use 3 states to keep track of the current residue mod 3; for example, state 0 denotes that the digits read so far represent a multiple of 3. Appending a 0 to a binary number is the same as doubling, which swaps 1 and 2 (since e.g. $2 \cdot 2 = 4 = 1 \mod 3$) and leaves 0 unchanged. Similarly, appending a 1 is like doubling and then adding 1, which sends $0 \to 1, 1 \to 0, 2 \to 2$. We accept in state 1 since this corresponds to 1 mod 3.

(B) See the figures below. We begin by adding new start/accept states with ε-transitions to the old ones. We then remove the original states one at a time, updating transitions as we go along. For example, in the first step, we delete state 2 and add a self-loop to state 0 with an appropriate regular expression. The final regular expression can be seen on the arrow in the last step of the construction.

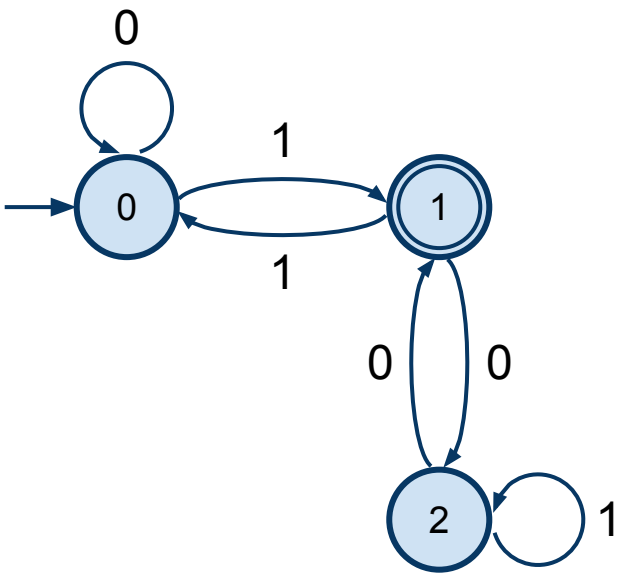
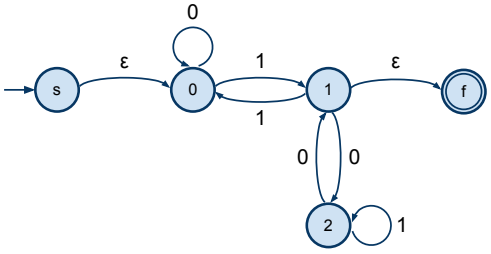
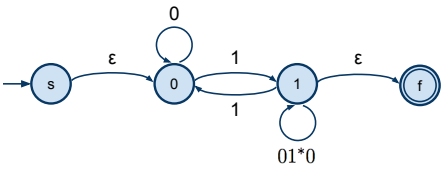
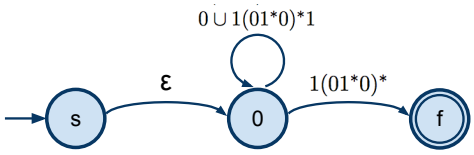


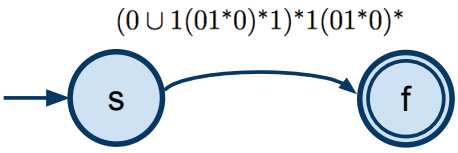
Figure 1: DFA for Problem 3.A.

GNFA construction for Problem 3.B:









PROBLEM 3 (12 points)

Let Σ and Δ be alphabets. Consider a function $\varphi : \Sigma \to \Delta^*$. Extend φ to a function from $\Sigma^* \to \Delta^*$ such that:

$$\begin{aligned} \varphi(\varepsilon) &= \varepsilon \\ \varphi(w\sigma) &= \varphi(w)\varphi(\sigma), \text{ for any } w \in \Sigma^*, \sigma \in \Sigma \end{aligned}$$

For example, if $\Sigma = \Delta = \{a, b\}$, $\varphi(a) = ab$, and $\varphi(b) = aab$, then $\varphi(aab) = \varphi(aa)\varphi(b) = \varphi(a)\varphi(a)\varphi(b) = ababaab$. Notice how φ operates on each input symbol individually. Any function $\varphi : \Sigma^* \to \Delta^*$ defined in this way from a function $\varphi : \Sigma \to \Delta^*$ is called a **homomorphism**.

Prove that the set of regular languages is closed under homomorphism. Specifically, prove by induction on the length of R that $\varphi[L(R)]$ is regular for any regular expression R and any homomorphism φ. (In other words, prove that, if L is regular, then $\varphi[L]$ is also regular, for any homomorphism φ.)

We begin with several lemmata about homomorphisms which will simplify the rest of the proof.

Lemma 1. For any languages $L_1, L_2 \subseteq \Sigma^*$, $\varphi[(L_1 \cup L_2)] = \varphi[L_1] \cup \varphi[L_2]$.

Proof (of lemma).

$$\begin{aligned} \varphi[(L_1 \cup L_2)] &= \{\varphi(w) : w \in (L_1 \cup L_2)\} \text{ by definition of } \varphi[L] \\ &= \{\varphi(w) : w \in L_1 \text{ or } w \in L_2\} \text{ by the definition of language union} \\ &= \{\varphi(w) : w \in L_1\} \cup \{\varphi(w) : w \in L_2\} \text{ by the definition of set union} \\ &= \varphi[L_1] \cup \varphi[L_2] \text{ by the definition of } \varphi[L] \text{ applied twice.} \end{aligned}$$

Lemma 2. For any strings $x, y \in \Sigma^*$, $\varphi(xy) = \varphi(x)\varphi(y)$.

Proof (of lemma). Prove by induction on $|y|$. For $|y| = 0$, we have $xy = xe = x$ so that $\varphi(xy) = \varphi(x)$ trivially. Now, assume that for all y with $|y| \leq k$, $\varphi(xy) = \varphi(x)\varphi(y)$. Consider any string y of length $k + 1$. Then $y = u\sigma$ for some $u \in \Sigma^*$ and some $\sigma \in \Sigma$, and so $\varphi(xy) = \varphi(xu\sigma) = \varphi(xu)\varphi(\sigma)$ by the definition of a homomorphism. By our induction hypothesis, this is equal to $\varphi(x)\varphi(u)\varphi(\sigma)$ (since $|u| = k$) and reversing the definition of a homomorphism tells us that this in turn is equal to $\varphi(x)\varphi(y)$ as was to be shown.

Lemma 3. For any number of strings $x_1, x_2, \ldots, x_n \in \Sigma^*$, $\varphi(x_1 x_2 \ldots x_n) = \varphi(x_1)\varphi(x_2)\ldots\varphi(x_n)$.

Proof (of lemma). Induct on n. The base case is for $n = 0$, in which case there are no x_i at all, and $\varphi(\varepsilon) = \varphi(\varepsilon)$ trivially. If we now assume that this result holds for some number of strings n, we show that it holds for $n + 1$ strings as well. Consider $\varphi(x_1 \ldots x_n x_{n+1})$. By lemma 2, we can break this down into $\varphi(x_1 \ldots x_n)\varphi(x_{n+1})$ (treating $x_1 \ldots x_n$ as the"x" and x_{n+1} as the"y"). We then apply the inductive hypothesis on the n remaining x_i to obtain $\varphi(x_1)\ldots\varphi(x_n)\varphi(x_{n+1})$ which is the desired result.

Lemma 4. For any languages $L_1, L_2 \subseteq \Sigma^*$, $\varphi[L_1 \circ L_2] = \varphi[L_1] \circ \varphi[L_2]$.

Proof (of lemma).

$$\begin{aligned}
\varphi[L_1 \circ L_2] &= \{\varphi(w) : w \in L_1 \circ L_2\} \text{ by definition of } \varphi[L] \\
&= \{\varphi(w) : w = xy, w \in L_1 \text{ and } y \in L_2\} \text{ by definition of language concatenation} \\
&= \{\varphi(xy) : x \in L_1 \text{ and } y \in L_2\} \text{ substituting} \\
&= \{\varphi(x)\varphi(y) : x \in L_1 \text{ and } y \in L_2\} \text{ by Lemma 2} \\
&= \{\varphi(x) : x \in L_1\} \circ \{\varphi(y) : y \in L_2\} \text{ by definition of language concatenation} \\
&= \varphi[L_1] \circ \varphi[L_2] \text{ by definition of } \varphi[L] \text{ applied twice.}
\end{aligned}$$

Lemma 5. For any language $L \subseteq \Sigma^*$, $\varphi[L^*] = \varphi[L]^*$.

Proof (of lemma).

$$\begin{aligned}
\varphi[L^*] &= \{\varphi(w) : w \in L^*\} \text{ by definition of } \varphi[L] \\
&= \{\varphi(w) : w = w_1 w_2 \dots w_n, w_i \in L\} \text{ by definion of Kleene star} \\
&= \{\varphi(w_1 w_2 \dots w_n) : w_i \in L\} \text{ by substitution} \\
&= \{\varphi(w_1)\varphi(w_2) \dots \varphi(w_n) : w_i \in L\} \text{ by lemma 3} \\
&= \{\varphi(x) : x \in L\}^* \text{ by definition of Kleene star} \\
&= \varphi[L]^* \text{ by definition of } \varphi[L].
\end{aligned}$$

Now it's time for the main proof! Using these lemmata, we will now show that, given a regular language $L(R)$ (for some regular expression R), $\varphi[L(R)]$ is regular, that is, we can find some regular expression R' such that $\varphi[L(R)] = L(R')$. We will prove this by inducting on the length of R.

Base Case: $|R| = 1$. By definition of a regular expression, either $R = \boldsymbol{\emptyset}$, $R = \boldsymbol{\varepsilon}$, or $R = \boldsymbol{\sigma}$, $\sigma \in \Sigma$.

- Case $\boldsymbol{\emptyset}$: $\varphi[L(R)] = \varphi[L(\boldsymbol{\emptyset})] = \varphi[\emptyset] = \emptyset$
  But the empty set is regular: it is described by the regular expression $\boldsymbol{\emptyset}$.
- Case $\boldsymbol{\varepsilon}$: $\varphi[L(R)] = \varphi[L(\boldsymbol{\varepsilon})] = \varphi[\{\varepsilon\}] = \{\varepsilon\}$
  This set contains only one string, namely ε, so it is regular.
- Case $\boldsymbol{\sigma}$: $\varphi[L(R)] = \varphi[L(\boldsymbol{\sigma})] = \varphi[\{\sigma\}] = \{\varphi(\sigma)\}$.
  The set containing only the string $\varphi(\sigma)$ is a finite (one-element) set, and therefore is regular.

Inductive Hypothesis: Given any regular language $L(R)$, $\varphi[L(R)]$ is regular for all $1 \leq |R| \leq n$.

Inductive Step: Take R, any regular expression of length $n+1$. There are three possible cases: either $R = (R_1 \cup R_2)$, $R = (R_1 \circ R_2)$, or $R = (R_1^*)$, for some regular expressions R_1 and R_2.

No matter which case we are dealing with, R_1 and R_2 have length less than or equal to n. So we can apply the inductive hypothesis to both to conclude that $\varphi[L(R_1)]$ and $\varphi[L(R_2)]$ are regular. This means that we can write

$$\begin{aligned}
\varphi[L(R_1)] &= L(R_1') \\
\varphi[L(R_2)] &= L(R_2')
\end{aligned}$$

where R'_1 and R'_2 are regular expressions.

Case 1: $R = (R_1 \cup R_2)$

$$\begin{aligned}\varphi[L(R)] &= \varphi[L((R_1 \cup R_2))] \\ &= \varphi[L(R_1) \cup L(R_2)] \text{ by the definition of RE union} \\ &= \varphi[L(R_1)] \cup \varphi[L(R_2)] \text{ by lemma 1}\end{aligned}$$

Applying the substitutions obtained from the inductive hypothesis, we get

$$\begin{aligned}&= L(R'_1) \cup L(R'_2) \\ &= L((R'_1 \cup R'_2)) \text{ by the definition of RE union.}\end{aligned}$$

Since R'_1 and R'_2 are regular expressions, $(R'_1 \cup R'_2)$ is also a regular expression. So $\varphi[L(R)]$ is represented by a regular expression and is therefore regular.

Case 2: $R = (R_1 \circ R_2)$

$$\begin{aligned}\varphi[L(R)] &= \varphi[L((R_1 \circ R_2))] \\ &= \varphi[L(R_1) \circ L(R_2)] \text{ by the definition of RE concatenation} \\ &= \varphi[L(R_1)] \circ \varphi[L(R_2)] \text{ by lemma 4}\end{aligned}$$

Applying the substitutions obtained from the inductive hypothesis, we get

$$\begin{aligned}&= L(R'_1) \circ L(R'_2) \\ &= L((R'_1 \circ R'_2)) \text{ by the definition of RE concatenation.}\end{aligned}$$

Since R'_1 and R'_2 are regular expressions, $(R'_1 R'_2)$ is also a regular expression. So $\varphi[L(R)]$ is represented by a regular expression and is therefore regular.

Case 3: $R = (R_1^*)$

$$\begin{aligned}\varphi[L(R)] &= \varphi[L((R_1^*))] \\ &= \varphi[L(R_1)^*] \text{ by the definition of Kleene star} \\ &= \varphi[L(R_1)]^* \text{ by lemma 5}\end{aligned}$$

Applying the substitutions obtained from the inductive hypothesis, we get

$$\begin{aligned}&= L(R'_1)^* \\ &= L((R'^*_1)), \text{ by the definition of Kleene star.}\end{aligned}$$

Since R'_1 is a regular expression, (R'^*_1) is a regular expression. So $\varphi[L(R)]$ is represented by a regular expression and is therefore regular.

This completes the induction. Thus, given any regular language L, we can find a regular expression for $\varphi[L]$, thereby showing it regular. This proves the important result that the regular languages are closed under homomorphisms.

PROBLEM 4 (12 points)

Let SUBSTRINGS$(L, A) = \{w \in L$: every substring of w is in A$\}$. Prove that if L and A are regular, then so is SUBSTRINGS(L, A). Hint: can you make use of any closure properties we already know?

Solution:

We know A is regular, and we know regular languages are closed under compliment, so that $\overline{A}$ is also regular and therefore has a DFA recognizing it. Let D be a DFA accepting $\overline{A}$.

From D, we construct an NFA N as follows:

1. Use the same states and transitions, i.e. if $\delta_D(q, \sigma) = q'$ then $\delta_D(q, \sigma) = \{q'\}$.
2. Introduce a new start state with an ε-arrow to the old start state.
3. Introduce a new final state with ε-arrows from old accepting states.
4. Add a self-loop on all symbols σ to the two new states.
5. Make all old states non-start and non-accepting.

Given the 5-tuple description of D, one can use these rules to construct the 5-tuple description of N. One can also draw a transition diagram for N to get an intuition for how it works.

Then, we claim that N accepts exactly the set of strings that have some substring in $\overline{A}$. Note that we have two directions of inclusion to prove.

First, in order for N to accept a string w , the accepting computation will read some symbols in the new start state, transition on ε to the old start state, follow the (deterministic) computation of D to an old accepting state, follow an ε-arrow to the new accepting state, and then stay there until finally accepting. In other words, w has some symbols, then follows an accepting computation of D, and then has some more symbols. Therefore, w has a substring accepted by D, namely, the portion of the string read while simulating D.

For the other direction, if w has a substring accepted by D, then N can stay in the new start state for all the symbols before that substring, then follow the computation path of D on the substring, ending on an old accepting state, then enter the new accepting state, read the rest of the symbols, and then accept, so that N accepts w.

Thus, $L(N)$ is the set of strings with some substring in $\overline{A}$, so that $\overline{L(N)}$ is the set of strings such that all substrings are in A.

Again, by closure under complement, we know that $\overline{L(N)}$ is regular. Finally, because regular languages are also closed under intersection, we know that $\overline{L(N)} \cap L$ is regular. But this is exactly the language SUBSTRINGS(L, A), so we are done.

PROBLEM 5 (4+4+4+4 points)

(A) Define the $\lesssim$ relation on nonempty sets by $S \lesssim T$ if there is an onto map from T to S. [1] Show that $S \lesssim T$ if and only if there is a one-to-one map from S to T, so this relation is in fact the same as the one that was defined in Problem Set 0.
(B) Show that for every nonempty set S, $S \lesssim P(S)$.
(C) Show that for every set S, $P(S) \not\lesssim S$. (Hint: The diagonalization technique does not require enumerating the set in question.)
(D) Conclude that there are infinitely many different equivalence classes under the relation $\sim$ defined in Problem Set 0, i.e. infinitely many different "infinite cardinalities".

(A) We need to show both directions. First $\Rightarrow$:

Given $S \lesssim T$, let g be an onto map from T to S. We now construct a one-to-one map $f : S \to T$ as follows. $\forall s \in S$, let $A_s = \{t \in T : g(t) = s\}$ (this definition of A_s is called the pre-image of s under g). Now, using the axiom of choice, we define $t_s \in A_s$ as an arbitrary element in A_s. Because g is onto, A_s is non-empty for all s, so t_s is defined for all $s \in S$. Then we simply define $f(s) = t_s$. Because g is a function, all the A_s are disjoint (otherwise g would take the same t to multiple s), so f is one-to-one.

Now the other direction $\Leftarrow$:

Given a one-to-one function $f : S \to T$, we show that $S \lesssim T$ by constructing an onto function $g : T \to S$. First, we choose an arbitrary element $k \in S$ (ok since S is non-empty). Next, observe that because f is one-to-one, it is invertible, and f^{-1} is defined for all t in the image of f (denoted by $f[S]$). We define $g(t) = f^{-1}(t)$ for $t \in f[S]$, and $g(t) = k$ for $t \notin f[S]$. It is clear from the definition that the range of g will include all of S, and it is defined for all $t \in T$, so it is an onto map.

(B) Let k be some fixed element of S. Define $f : P(S) \to S$ as follows. If $x \in P(S)$ is a singleton set $\{y\}$ (where $y \in S$) then define $f(x) = y$. For all other $x \in P(S)$, define $f(x) = k$. Since for each $y \in S$, $f(\{y\}) = y$, f is onto. (Note: given part A, you could also show this by constructing a one-to-one map $g : S \to P(S)$ as $g(y) = \{y\}$.)

(C) Assume by contradiction that $f : S \to P(S)$ is an onto function. Define $X = \{a \in S : a \notin f(a)\}$. Since f is onto and $X \in P(S)$, there must be some b in S such that $f(b) = X$. Now there are two cases: either $b \in X$ or $b \notin X$.

If $b \in X$, then given the definition of X, $b \notin f(b)$. But $f(b) = X$, which means that if $b \in X$ then $b \notin X$ – a contradiction. On the other hand, if $b \notin X$, then the definition of X means that $b \in f(b)$. But $f(b) = X$, which contradicts the fact that $b \notin X$.

Since both $b \in X$ and $b \notin X$ lead to contradictions, we must conclude that our original assumption that f is onto was false.

(D) Taken together, parts B and C show that for any non-empty set S, $S \lesssim P(S)$, but $S \not\sim P(S)$, so $P(S)$ has strictly larger cardinality. Now let S be any infinite set, and take the sequence $\{S, P(S), P(P(S)), ...\}$. This is an infinite sequence, where each element has a strictly larger car-

[1] Students of logic may notice that the domain of this relation would be the set of all sets, which cannot be defined without leading to paradoxes. The way around this is to talk about a relation on the 'class' of all sets, but the distinction between the notions of a 'class' and a 'set' is beyond the scope of CS121.

dinality than the one before it. Therefore, we can conclude that there are infinitely many different "infinite cardinalities".

PROBLEM 6 (Challenge! 2+1 points)

(A) Let $L/A = \{x : wx \in A \text{ for some } w \in L\}$. Show that if A is regular and L is *any* language, then L/A is regular.
(B) Suppose $L = \{a^n : n \text{ is greater than 2, is even, and cannot be expressed as the sum of two primes}\}$, and $A = \Sigma^*$. Why doesn't this contradict your proof?

(A) Since A is regular, there is a DFA $D = (Q, \Sigma, \delta, q_0, F)$ that recognizes it. Construct an NFA $N = (Q \cup \{s\}, \Sigma, \delta', s, F)$, where:

- $\delta'(q, \sigma) = \{\delta(q, \sigma)\}$ for all $q \in Q$, $\Sigma \in \sigma$
- $\delta'(q, \epsilon) = \{\}$
- $\delta'(s, \epsilon) = \{q \in Q : \delta^*(q_0, w) = q \text{ for some } w \in L\}$
- $\delta'(s, \sigma) = \{\}$ for all $\sigma \in \Sigma$.

Then, for any input string, there will be an accepting path from s (through some epsilon-edge to a $q \in Q$) to some $q_f \in F$ if and only if the input string can be appended to a string in L (driving D from q_0 to q), and ultimately leave D in q_f – i.e., the concatenation of some string in L with the input string yields a string in A.

(B) If Goldbach's conjecture (every even integer greater than 2 can be expressed as the sum of two primes) is true, then $L/A = \{\}$. If it is false, then $L/A = \Sigma^*$. *Either of these is regular*, whether or not we know which one it is!