CSC236 Assignment #3

Zhongtian Ouyang, Peiqi Wang

December 2, 2016

Problem 1

Proof of Correctness for iterative algorithms.

1. Design an iterative closest pair algorithm for finding the closest pair of points in 2D.
   Precondition: Input is a list of n points in the form (x_i, y_i), where $x_i, y_i \in \mathbb{R}$
   Postcondition: Return a closest pair of points

```
def closestPair(L):
    """ Brute force method for finding the closest pair of points from
     an array L of bivariate points (x_i, y_i)

    @param: Array[Array] L: input array
    @rparam: Array t: the closest pair
    """

    n = len(L)
    u, v = -1, -1
    min = float('inf')

    for i in range(0, n):
        for j in range(i+1, n):
            distance = findDistance(L[i], L[j])
            if distance < min:
                min = distance
                u = i
                v = j
    return (L[u], L[v])

def findDistance(p1, p2):
    return float(math.sqrt( (p1[0]-p2[0])**2 + (p1[1] - p2[1])**2 ))
```

2. Find complexity class

   Solution.
   We notice that there is a nested loop consisting of an outer i loop and inner j loop.

The outer i loop executes $n-1$ times and the inner j loop executes $n-i-1$ times. Therefore the nested loop executes

$$(n-1)+(n-2)+\cdots+1=\frac{n(n-1)}{2}=\frac{1}{2}n^2-\frac{1}{2}n$$

iterations. The complexity of class is therefore $\Theta(n^2)$

□

3. Prove correctness:

   (a) Define Loop Invariant

   Solution.
   The loop invariant for the outer i loop is

   $P(k)$: if outer loop is executed at least k times, then min contains the minimum distance between any point in $L[0..k-1]$ and any point in $L[0..n-1]$. And the indices of the corresponding closest points in L are u and v respectively.

   Here we use induction to prove that $P(i)$ holds for all $i \in \mathbb{N}$

   Basis:
   When $i=0$, min is infinity and $u, v=-1$, Since there is no points at indices -1, $P(0)$ is true trivially.

   Inductive step:
   Let $i>0$. Assume $P(i)$ holds and we prove $P(i+1)$ holds as well.
   Here we use $P(i)$ as the precondition for the inner j loop. Specifically, min contains the minimum distance between any points in $L[0..i-1]$ and any point in $L[0..n-1]$ And the indices of the corresponding closest points in L are u and v respectively. We define the loop invariant for the inner loop as follows,

   Q(k): if the inner loop is executed at least k times, then min is the minimum of **1)** minimum distance between any point in $L[0..i-1]$ and any point in $L[0..n-1]$. **2)** the minimum of $L[i]$ and any points in $L[i+1..i+1+k]$. Also the indices of the corresponding closest points in L are u and v respectively.

   We prove inner state invariant $Q(m)$ holds for all $m \in \mathbb{N}$, where m denotes the iterations of inner loop traversed.

   Basis:
   when $m=0$, before the inner loop first executes, precondition $P(i)$ holds, therefore $Q(0)$ holds.

   Inductive step:
   when $m>0$, let m be an arbitrary number such that $Q(m)$ holds, now we prove

$Q(m+1)$ holds as well. *case 1:* when $distance \geq min$, then min remains to be the minimum given by $P(i)$ and $Q(m)$, therefore $Q(m+1)$ holds.
case 1: when $distance < min$. Notice in line 14 $distance$ denotes the distance beween $L[i]$ and $L[i+1+m+1]$, min is assigned value of $distance$. Now min is the minimum of

i. minimum between any points in $L[0..i-1]$ and any point in $L[0..n-1]$
ii. minimum betwen $L[i]$ and any points in $L[i+1..i+1+m]$
iii. minimum of $L[i]$ and $L[i+1..i+1+m+1]$

u and v correspondingly assigned the indices of closest points. Therefore $Q(m+1)$ holds.

Hence loop invariant $Q(m)$ holds for all $m \in \mathbb{N}$

Therefore, if the inner loop terminates with $m = n-1$, then min contains minimum of

i. the minimum distance between any points in $L[0..i-1]$ and any point in $L[0..n-1]$
ii. the minimum distance between $L[i]$ and any points in $L[0..n-1]$

Therefore min contains minimum of any points in $L[0..i]$ and any points in $L[0..n-1]$. Also by state invariant Q, u, v are corresponding closest points. Therefore $P(i+1)$ holds.

To conclude, by simple induction, we proved that state invariant $P(i)$ holds for all $i \in \mathbb{N}$

□

(b) Prove Partial Correctness

Solution. Prove,

Suppose that the program executes and satisfies precondition. If the program terminates, when it does, the post condition holds.

When the outer for loop terminates, then $i = n-1$, by state invariant proved previously, min is the minimum of any points in $L[0..n-2]$ and $L[0..n-1]$ and u, v are corresponding indices of closest pair of points. We can see that this covers the entirety of points in the list. Therefore the returned pair of points $L[u], L[v]$ are pairs of points with minimum distance.

□

(c) Termination (use either theorem 2.5 in the notes or POW)

Solution.
For the inner loop,let $f_i = n - j_i$ for arbitrary iteration i, therefore f_i is an integer because n and j_i are integers. Since $j_{i+1} = j_i + 1$ as a result of the for loop and

n is a finite integer, then $f_{i+1} < f_i$ for all i. f_i are decreasing series of integers, hence there are only finitely many f_i. The inner loop therefore will terminate.

For the outer loop. Let $g_l = n - i_l$ for arbitrary iteration l, therefore g_l is an integer because n and i_l are integers. Since $i_{l+1} = i_l + 1$ as a result of the for loop and n is a finite integer, then $g_{l+1} < g_l$ for all l. g_l is decreasing series of integers, hence there are finitely many g_i. Also for any iteration, the inner for loop will terminate, therefore the outer loop will terminate

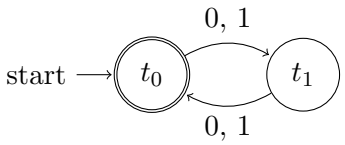
Hence the program terminates. □

Problem 2

DFSAs and their operations

1. Define and draw DFSAs on binary alphabet $\Sigma = \{0, 1\}$ for 2 languages: $L_1(M_1) =$ {all strings with even number of characters in a string}, $L_2(M_2) =$ {all strings that have even number of 1s}

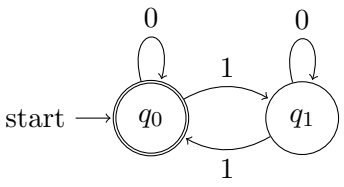
   Solution.
   For M_1



$$Q = \{t_0, t_1\}$$
$$\Sigma = \{0, 1\}$$
$$s = t_0$$
$$F = \{t_0\}$$
$$\delta = Q \times \Sigma \text{ specified below}$$

	0	1
t_0	t_1	t_1
t_1	t_0	t_0

   For M_2



$$
\begin{aligned}
Q &= \{q_0, q_1\} \\
\Sigma &= \{0, 1\} \\
s &= q_0 \\
F &= \{q_0\} \\
\delta &= Q \times \Sigma \text{ specified below}
\end{aligned}
$$

	0	1
q_0	q_0	q_1
q_1	q_1	q_0

□

2. Identify DFSA M_3 for the union of languages $L_1 \cup L_2$ - you can define it formally (don't need to draw).

   Solution.
   We can construct DFSA M_3 for $L_1 \cup L_2$ by,

$$
\begin{aligned}
Q &= \{(t_0, q_0), (t_0, q_1), (t_1, q_0), (t_1, q_1)\} \\
\Sigma &= \{0, 1\} \\
s &= (t_0, q_0) \\
F &= \{(t_0, q_0), (t_0, q_1), (t_1, q_0)\} \\
\delta &= Q \times \Sigma \text{ specified below}
\end{aligned}
$$

	0	1
(t_0, q_0)	(t_1, q_0)	(t_1, q_1)
(t_0, q_1)	(t_1, q_1)	(t_1, q_0)
(t_1, q_0)	(t_0, q_0)	(t_0, q_1)
(t_1, q_1)	(t_0, q_1)	(t_0, q_0)

□

3. Identify DFSA M_4 for the intersection of languages $L_1 \cap L_2$ - you can define it formally (don't need to draw).

Solution.
We can construct DFSA M_4 for $L_1 \cap L_2$ by,

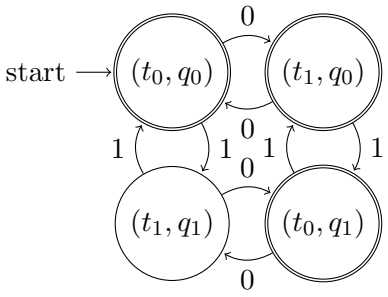
$$
\begin{aligned}
Q &= \{(t_0, q_0), (t_0, q_1), (t_1, q_0), (t_1, q_1)\} \\
\Sigma &= \{0, 1\} \\
s &= (t_0, q_0) \\
F &= \{(t_0, q_0)\} \\
\delta &= Q \times \Sigma \text{ specified below}
\end{aligned}
$$

	0	1
(t_0, q_0)	(t_1, q_0)	(t_1, q_1)
(t_0, q_1)	(t_1, q_1)	(t_1, q_0)
(t_1, q_0)	(t_0, q_0)	(t_0, q_1)
(t_1, q_1)	(t_0, q_1)	(t_0, q_0)

□

4. Find and prove a state invariant for M_3.

Solution.
First we visualize DFSA for M_3,



Here we define predicate

$$
P(x) : \delta^*((t_0, q_0), x) = \begin{cases} (t_0, q_0), & x \text{ has even number of characters and even number of 1's} \\ (t_1, q_0), & x \text{ has odd number of characters and even number of 1's} \\ (t_0, q_1), & x \text{ has even number of characters and odd number of 1's} \\ \text{`}(t_1, q_1)\text{`}, & x \text{ has odd number of characters and odd number of 1's} \end{cases}
$$

Now we use structural induction to prove that $P(x)$ holds for all strings $x \in \Sigma^*$

Proof.
Basis:
let $x = \epsilon$, x in this case has zero of 0s and zero 1s. So x is a string with even number of characters and even number of 1s and also $\delta^*((t_0, q_0), x) = (t_0, q_0)$. Therefore $P(x)$ holds
Inductive step:
Let $x = ya$, where $y \in \Sigma^*$ and $a \in \Sigma$. Now assume $P(y)$ holds; this is the inductive hypothesis. We prove that $P(x)$ also holds.
Case 1: when $a = 0$

$$
\begin{aligned}
&\delta^*((t_0, q_0), x) \\
&\quad = \delta^*((t_0, q_0), y0) \\
&\quad = \delta(\delta^*((t_0, q_0), y), 0) \\
&\quad = \begin{cases} \delta((t_0, q_0), 0), & y \text{ has even number of characters and even number of 1's} \\ \delta((t_1, q_0), 0), & y \text{ has odd number of characters and even number of 1's} \\ \delta((t_0, q_1), 0), & y \text{ has even number of characters and odd number of 1's} \\ \delta((t_1, q_1), 0), & y \text{ has odd number of characters and odd number of 1's} \end{cases} \quad \text{by I.H. P(y) holds}
\end{aligned}
$$

because x has one more character and one more zero, we have

$$
= \begin{cases} (t_1, q_0), & x \text{ has odd number of characters and even number of 1's} \\ (t_0, q_0), & x \text{ has even number of characters and even number of 1's} \\ (t_1, q_1), & x \text{ has odd number of characters and odd number of 1's} \\ (t_0, q_1), & x \text{ has even number of characters and odd number of 1's} \end{cases} \quad \text{by transition function}
$$

Therefore, $P(x)$ holds in this case.
Case 2:
Similar to Case 1 we arrive at $P(x)$ holds in this case.
Therefore, $P(x)$ holds for all strings $x \in \Sigma^*$ by structural induction
We proved that $P(x)$ is a state invariant for M_3

□

□

Problem 3

Equivalence of languages and regular expressions
Language L over alphabet $\Sigma = \{a, b\}$ consists of all strings that start with a and have odd lengths or start with b and have even lengths: $\{s | s$ starts with a and has odd length, or starts with b and has even length$\}$.

1. What is a regular expression R corresponding to language L?

$$R = a((a+b)(a+b))^* + b(a+b)((a+b)(a+b))^*$$

2. Prove that your regular expression R is indeed equivalent to L

Proof.
Here we denote $L(R)$ as the language described by R over alpabet Σ. We show containment in both directions to prove equivalence, i.e.,

$$L(R) = L \iff L(R) \subseteq L \wedge L \subseteq L(R)$$

To prove $L(R) \subseteq L$, let $x \in L(R)$, then by definition of regular language,

$$x \in L(a((a+b)(a+b))^*) \cup L(b(a+b)((a+b)(a+b))^*)$$

Case 1: When $x \in L(a((a+b)(a+b))^*)$
Then $x \in L(a) \circ (L((a+b)(a+b))^*$. Let $y \in L(a) = \{a\}$, $w \in L((a+b)(a+b) = \{aa, ab, ba, bb\}$. Then $y = a$ and w^* is just concatenation of strings in $\{aa, ab, ba, bb\}$, where the length of any string in the set is 2. Here $\exists k \in \mathbb{N}, |w^*| = 2k$ as follows. Therefore yw^* is of length $2k+1$, hence odd. Note that $x = yw^*$, then x is a string that starts with a and has odd length. We see that $x \in L$ as expected.
Case 2: When $x \in L(b(a+b)((a+b)(a+b))^*)$
Then $x \in L(a) \circ L(a+b) \circ L((a+b)(a+b))$. Let $y \in L(a) = \{b\}$, $u \in L(a+b) = \{a, b\}$, and $w \in L((a+b)(a+b)) = \{aa, ab, ba, bb\}$. Then $y = b$, u is either a or b so of length 1, and w^* has length $2k$ for some k as explained previously. Then $|yuw| = 1 + 1 + 2k = 2(1+k)$, hence even. Note $x = yuw$, therefore x is a string that starts with b and has even length. We see that $x \in L$ as expected.
Therefore, $x \in L(a((a+b)(a+b))^* + b(a+b)((a+b)(a+b))^*) \Rightarrow x \in L$, we proved that $L(R) \subseteq L$

To prove $L \subseteq L(R)$, let $x \in L$. Then

$x \in \{s : s$ starts with a and has odd length$\} \cup \{s | s$ starts with b and has even length$\}$

Case 1: x is a string that starts with a and has odd length $2k+1$ for some $k \in \mathbb{N}$. Then the substring excluding prefix a has even length $2k$. Let

$$x = yw_1w_2 \dots w_k$$

where $y = a$ and $w_i \in \{aa, ab, ba, bb\}$ for $1 \leq i \leq k$. This construction is valid because $|y| = 1$ and $|w_1w_2 \dots w_k| = 2k$ and therefore $|x| = 1 + 2k$ is as wanted. Also we see that $y \in L(a)$ and $w_i \in L((a+b)(a+b))$. Because x contains arbitrary k number of w_i, then $w_1w_2 \dots w_k \in L((a+b)(a+b))^* = L(((a+b)(a+b))^*)$ Then

$x \in L(a) \circ L(((a+b)(a+b))^*) = L(a((a+b)(a+b))^*)$. Therefore, $x \in L(a((a+b)(a+b))^*) \cup L(b(a+b)((a+b)(a+b))^*)$ as wanted.
Case 2: x starts with b and has even length, $2k+2$ for some $k \in \mathbb{N}$ (i.e. x cannot be empty). Then the substring excluding prefix b has odd length $2k+1$. let

$$x = yuw_1w_2 \dots w_k$$

where $y = a$, $u \in \{a, b\}$, and $w_i \in \{aa, ab, ba, bb\}$ for $1 \leq i \leq k$. This construction is valid because $|y| = 1$, $|u| = 1$ and $|w_1w_2 \dots w_k| = 2k$ and therefore $|x| = 2 + 2k$ as wanted. Also note that $y \in L(a)$, $u \in L(a+b)$, $w_i \in L((a+b)(a+b))$. Because x contains arbitrary number k of w_i, then $w_1w_2 \dots w_k \in L((a+b)(a+b))^* = L(((a+b)(a+b))^*)$. Then by $x = yuw_1w_2 \dots w_k$, $x \in L(a) \circ L(a+b) \circ L(((a+b)(a+b))^*) = L(a(a+b)((a+b)(a+b))^*)$. Therefore $x \in L(a((a+b)(a+b))^*) \cup L(b(a+b)((a+b)(a+b))^*)$ as wanted.

Since $x \in L \Rightarrow x \in L(R)$, we proved that $L \subseteq L(R)$.

To conclude, because $L(R) \subseteq L \wedge L \subseteq L(R)$, then $L = L(R)$.

□