Spring, 2013 CIS 511

Introduction to the Theory of Computation
Jean Gallier

Homework 4

March 4, 2013; Due March 21, 2013

"A problems" are for practice only, and should not be turned in.

Problem A1. Given any two context-free languages L_1 and L_2 over the same alphabet Σ, prove that $L_1 \cup L_2$ and L_1L_2 are also context-free.

Problem A2. Let Σ and Δ be some alphabets, and let $h\colon \Sigma^* \to \Delta^*$ be a homomorphism. Given any language $L \subseteq \Sigma^*$, recall that

$$h(L) = \{h(w) \in \Delta^* \mid w \in L\}.$$

Prove that if L is context-free, then $h(L)$ is also context-free.

Problem A3. Given any language $L \subseteq \Sigma^*$, let

$$L^R = \{w^R \mid w \in L\},$$

the *reversal language of* L (where w^R denotes the reversal of the string w). Prove that if L is context-free, then L^R is also context-free.

"B problems" must be turned in.

Problem B1 (80 pts). (i) Prove that the conclusion of the pumping lemma holds for the following language L over $\{a, b\}^*$, and yet, L is **not** regular!

$$L = \{w \mid \exists n \geq 1, \exists x_i \in a^+, \exists y_i \in b^+, 1 \leq i \leq n, n \text{ is not prime}, w = x_1y_1 \cdots x_ny_n\}.$$

(ii) Consider the following version of the pumping lemma. For any regular language L, there is some $m \geq 1$ so that for every $y \in \Sigma^*$, if $|y| = m$, then there exist $u, x, v \in \Sigma^*$ so that

(1) $y = uxv$;

(2) $x \neq \epsilon$;

(3) For all $z \in \Sigma^*$,

$$yz \in L \quad \text{iff} \quad ux^ivz \in L$$

for all $i \geq 0$.

Prove that this pumping lemma holds.

(iii) Prove that the converse of the pumping lemma in (ii) also holds, i.e., if a language L satisfies the pumping lemma in (ii), then it is regular.

(iv) Consider yet another version of the pumping lemma. For any regular language L, there is some $m \geq 1$ so that for every $y \in \Sigma^*$, if $|y| \geq m$, then there exist $u, x, v \in \Sigma^*$ so that

(1) $y = uxv$;

(2) $x \neq \epsilon$;

(3) For all $\alpha, \beta \in \Sigma^*$,

$$\alpha u \beta \in L \quad \text{iff} \quad \alpha u x^i \beta \in L$$

for all $i \geq 0$.

Prove that this pumping lemma holds.

(v) Prove that the converse of the pumping lemma in (iv) also holds, i.e., if a language L satisfies the pumping lemma in (iv), then it is regular.

Problem B2 (80 pts). This problem is based on the method proved correct in Problem B5 of Homework 3.

Given a DFA $D = (Q, \Sigma, \delta, q_0, F)$, for any two states $p, q \in Q$, a fast algorithm for computing the forward closure of the relation $R = \{(p, q)\}$, or detecting a bad pair of states, can be obtained as follows: An equivalence relation on Q is represented by a partition Π. Each equivalence class C in the partition is represented by a tree structure consisting of nodes and (parent) pointers, with the pointers from the sons of a node to the node itself. The root has a null pointer. Each node also maintains a counter keeping track of the number of nodes in the subtree rooted at that node.

Two functions *union* and *find* are defined as follows. Given a state p, $find(p, \Pi)$ finds the root of the tree containing p as a node (not necessarily a leaf). Given two root nodes p, q, $union(p, q, \Pi)$ forms a new partition by merging the two trees with roots p and q as follows: if the counter of p is smaller than that of q, then let the root of p point to q, else let the root of q point to p.

In order to speed up the algorithm, we can modify $find$ as follows: during a call $find(p, \Pi)$, as we follow the path from p to the root r of the tree containing p, we redirect the parent pointer of every node q on the path from p (including p itself) to r.

Say that a pair $\langle p, q \rangle$ is *bad* iff either both $p \in F$ and $q \notin F$, or both $p \notin F$ and $q \in F$. The function *bad* is such that $bad(\langle p, q \rangle) = true$ if $\langle p, q \rangle$ is bad, and $bad(\langle p, q \rangle) = false$ otherwise.

For details of this implementation of partitions, see *Fundamentals of data structures*, by Horowitz and Sahni, Computer Science press, pp. 248-256.

Then, the algorithm is as follows:

```
function $unif[p, q, \Pi, dd]$: $flag$;
  begin
    $trans := left(dd)$; $ff := right(dd)$; $pq := (p, q)$; $st := (pq)$; $flag := 1$;
    $k := Length(first(trans))$;
    while $st \neq () \wedge flag \neq 0$ do
      $uv := top(st)$; $uu := left(uv)$; $vv := right(uv)$;
      $pop(st)$;
      if $bad(ff, uv) = 1$ then $flag := 0$
      else
        $u := find(uu, \Pi)$; $v := find(vv, \Pi)$;
        if $u \neq v$ then
          $union(u, v, \Pi)$;
          for $i = 1$ to $k$ do
            $u1 := delta(trans, uu, k - i + 1)$; $v1 := delta(trans, vv, k - i + 1)$;
            $uv := (u1, v1)$; $push(st, uv)$
          endfor
        endif
      endif
    endwhile
  end
```

The initial partition Π is the identity relation on Q, i.e., it consists of blocks $\{q\}$ for all state $q \in Q$. The algorithm uses a stack st. We are assuming that the DFA dd is specified by a list of two sublists, the first list, denoted $left(dd)$ in the pseudo-code above, being a representation of the transition function, and the second one, denoted $right(dd)$, the set of final states. The transition function itself is a list of lists, where the i-th list represents the i-th row of the transition table for dd. The function $delta$ is such that $delta(trans, i, j)$ returns the j-th state in the i-th row of the transition table of dd. For example, we have a DFA

$$dd = (((2, 3), (2, 4), (2, 3), (2, 5), (2, 3), (7, 6), (7, 8), (7, 9), (7, 6)), (5, 9))$$

consisting of 9 states labeled $1, \ldots, 9$, and two final states 5 and 9. Also, the alphabet has two letters, since every row in the transition table consists of two entries. For example, the two transitions from state 3 are given by the pair $(2, 3)$, which indicates that $\delta(3, a) = 2$ and $\delta(3, b) = 3$.

Implement the above algorithm, and test it at least for the above DFA dd and the pairs of states $(1, 6)$ and $(1, 7)$. Pay particular attention to the input and output format. In particular, ouput the current partition at every round through the **while** loop. Explain your data structures.

Please, consult the instructions posted on the web page for CIS511, Homework section, for instructions on the format for the input and output for this computer program.

Extra Credit (up to **120 pts).** Implement your program in such a way that it displays the simultaneous parallel forward moves in the DFA and the updating of the trees representing the blocks of the partition. There are programming languages, such as `Mathematica`, that have primitives to manipulate and output trees.

Problem B3 (50 pts). Prove that the language

$$L = \{a^{4n+3} \mid 4n + 3 \text{ is prime}\}$$

is not regular.

Hint. First, you will have to prove that there are infinitely many primes of the form $4n + 3$. The list of such primes begins with

$$3,\ 7,\ 11,\ 19,\ 23,\ 31,\ 43\cdots$$

Say we already have $n + 1$ of these primes, denoted by

$$3,\ p_1,\ p_2, \cdots, p_n,$$

where $p_i > 3$. Consider the number

$$m = 4p_1p_2 \cdots p_n + 3.$$

If $m = q_1 \cdots q_k$ is a prime factorization of m, prove that $q_j > 3$ for $j = 1, \ldots k$ and that no q_j is equal to any of the p_i's. Prove that one of the q_j's must be of the form $4n + 3$, which shows that there is a prime of the form $4n + 3$ greater than any of the previous primes of the same form.

Problem B4 (60 pts). Let $D = (Q, \Sigma, \delta, q_0, F)$ be a *trim* DFA. Consider the following procedure:

(1) Form an NFA, N^R, by reversing all the transitions of D, i.e., there is a transition from p to q on input $a \in \Sigma$ in N iff $\delta(q, a) = p$ in D.

(2) Apply the subset construction to the NFA, N^R, obtained in (1), taking the start state to be the set F. The final states of the DFA obtained by applying the subset construction to N^R are all the subsets containing q_0. Then, trim the resulting DFA, to obtain the DFA D^R.

Observe that $L(D^R) = L(D)^R$.

Now, apply the above procedure to D, getting D^R, and apply this procedure again, to get D^{RR}. Prove that D^{RR} is a minimal DFA for $L = L(D)$.

Hint. First prove that if δ_R is the transition function of D^R, then for every $w \in \Sigma^*$ and for every state, $T \subseteq Q$, of D^R,

$$\delta_R^*(T, w) = \{q \in Q \mid \delta^*(q, w^R) \in T\}.$$

Problem B5 (60 pts). Give context-free grammars for the following languages:

(a) $L_5 = \{wcw^R \mid w \in \{a, b\}^*\}$ (w^R denotes the reversal of w)

(b) $L_6 = \{a^m b^n \mid 1 \leq m \leq n \leq 2m\}$

For any fixed integer $K \geq 2$,

$L_7 = \{a^m b^n \mid 1 \leq m \leq n \leq Km\}$

(c) $L_8 = \{a^n b^n \mid n \geq 1\} \cup \{a^n b^{2n} \mid n \geq 1\}$

(d) $L_9 = \{a^m b^n a^m b^p \mid m, n, p \geq 1\} \cup \{a^m b^{4n} a^p b^{4n} \mid m, n, p \geq 1\}$

(e) $L_{10} = \{xcy \mid |x| = 2|y|, \ x, y \in \{a, b\}^*\}$

In each case, give a justification of the fact that your grammar generates the desired language.

Problem B6 (50 pts). Give context-free grammars for the languages

$$\begin{aligned} L_1 &= \{xcy \mid x \neq y, \ x, y \in \{a, b\}^*\} \\ L_2 &= \{xcy \mid x \neq y^R, \ x, y \in \{a, b\}^*\}. \end{aligned}$$

TOTAL: 380 + 120 points.