Computability Theory **Fall 2020**

Problem Set 1 Solution

1. Let $A_n = \{a^k |$ where k is a multiple of $n\}$. Show that for each $n \geqslant 1$, the language A_n is regular.

1. **Proof:** It suffices to construct a DFA M that accept only A_n for each n. The set of states of M is $Q = \{q_0, \ldots, q_{(n-1)}\}$. The state q_0 is the start state and the only accept state. The transition function is $\delta(q_i, a) = q_j$, where $j = i + 1 \bmod n$. For example, $\delta(q_{(n-1)}, a) = q_0$. For each character a that is input, M jumps to the next state. It accepts the string if and only if M stops at q_0. Therefore the length of the string consists of all a's and its length is a multiple of n. ■

2. Read Spiser section 1.4 (page 77 - 82), show that $L = \{0^m 1^n | \ m \neq n\}$ is not regular.

Proof: Note that 0 and 1 are symbols, so 0^m means the string with m 0's. (From Spiser's book) The pumping lemma for regular languages: if A is a regular language, then there is a number p, which is called the pumping length, where if s is any string in A of length at least p, then s may be devided into three pieces, $s = xyz$, satisfying the following conditions:

1. for each $i \geqslant 0$, $xy^iz \in A$,
2. $|y| > 0$, and
3. $|xy| \leqslant p$.

Assume $L = \{0^m 1^n | \ m \neq n\}$ is regular. Let p be the pumping length given by the pumping lemma. Set $m = p$ and $n = p + p!$. The string $s = 0^p 1^{p+p!} \in L$, and $|s| \geqslant p$. By the pumping lemma, s can be devided as xyz with $x = 0^a$, $y = 0^b$, $z = 0^c 1^{p+p!}$, where $b \leqslant 1$ and $a + b + c = p$. However, the string $s' = xy^{(p!/b)+1}z = 0^{a+p!+b+c}1^{p+p!} = 0^{p+p!}1^{p+p!} \notin L$. Contradiction.

■

3. Give a context-free grammar that generates the language

$$B = \{a^i b^j c^k | \ i, j, k \geqslant 0 \text{ and either } i = j \text{ or } j = k\}.$$

3. A CFG L that generate B:
$G = (V, \Sigma, R, S)$. V is $\{S, E_{ab}, E_{bc}, C, A\}$ and Σ is $\{a, b, c\}$. The rules are:
$S \to E_{ab}C | AE_{bc}$
$E_{ab} \to aE_{ab}b | \epsilon$
$E_{bc} \to bE_{bc}c | \epsilon$
$C \to Cc | \epsilon$
$A \to Aa | \epsilon$

Initially substituting $E_{ab}C$ for S generates any string with an equal number of a's and b's followed by any number of c's. Similar for substituting E_{bc} for S.

4. Let C be a context-free language and R be a regular language. Show that the language $R \cup C$ is context free.

Proof: Theorem 3.6 shows that every regular language is context-free. Theorem 3.8 shows that context-free languages are closed under union operation. Thus the result follows.

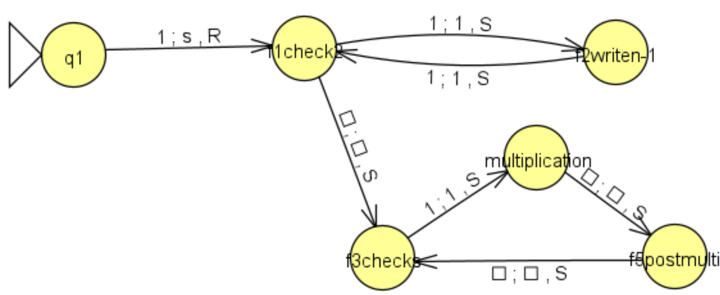
This problem is trivial because there was a typo. I should have wrote $R \cap C$ rather than $R \cup C$. Here is a proof for the original problem:
Let C be a context-free language and R be a regular language. Let P be the PDA that recognizes C, and D be the DFA that recognizes R. If Q is the set of states of P and Q' is the set of states of D, we construct a PDA P' that recognizes $C \cap R$ with the set of states $P \times Q$. P' will do what P does and also keep track of the states of D. It accepts a string ω if and only if it stops at a state $q \in F_P \times F_D$, where F_P is the set of accept states of P and F_D is the set of accept states of D. Since $C \cap R$ is recognized by P', it is context free. ■

5. Show that every regular language is Turing decidable.

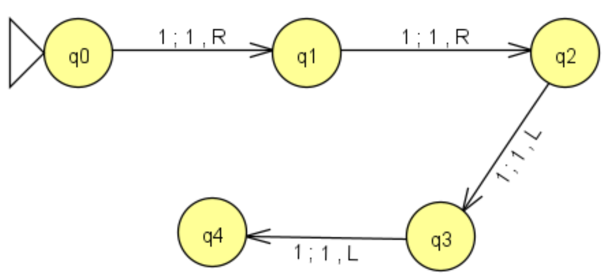
Proof: For any regular language L, there is a DFA D such that $L = L(D)$. Construct a Turing Machine T that simulates D as follows. T's states will be similar to D's. On reaching the end of the input, if T is in a state that corresponds to a final state of D, T halts and accepts; otherwise it halts and rejects. ■

6. Design a Turing machine that computes $f(n) = n!$.

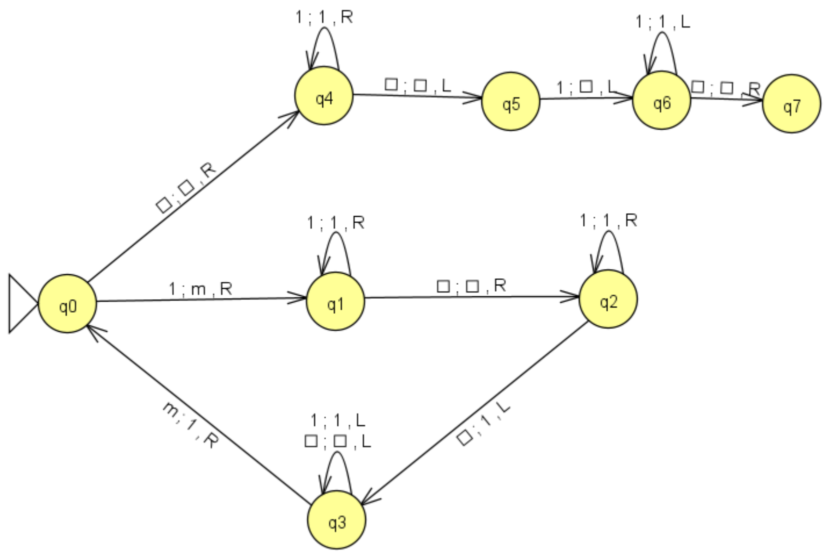
6.



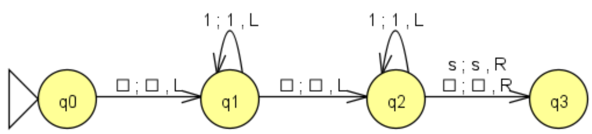
f1check2



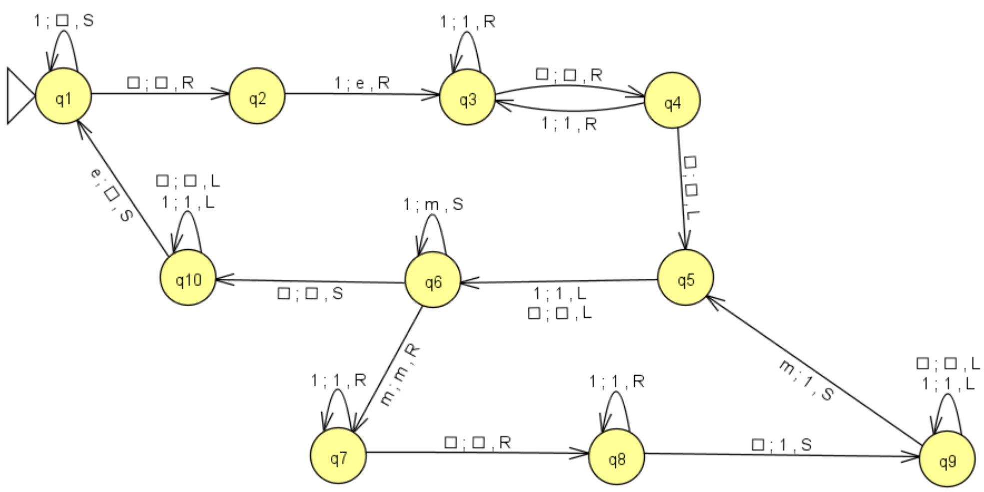
f2write1



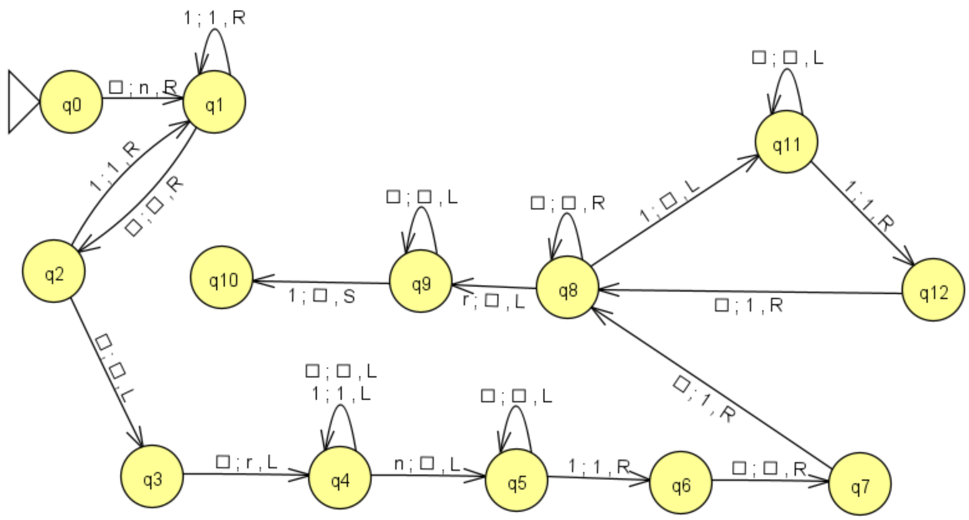
f3checks



multiplication, the same as Davis page 19



f5postmulti



7. Let $g_i(x)$ and $P_i(x)$ be primitive recursive for $i = 0, 1, \dots k$. Show that

$$f(x) = \begin{cases} g_1(x), & \text{if } P_1(x) \\ g_2(x), & \text{if } P_2(x) \\ \dots \\ g_k(x), & \text{otherwise} \end{cases}$$

is primitive recursive.

Proof: This is definition by case. $P_i(x)$ are supposed to be mutually exclusive and exhaustive. I should have made clear about this. Here is the proof:
Let $P_k(x) = \neg(P_1(x) \vee P_2(x) \dots P_k(x))$ Since every P_i is primitive recursive, $P_k(x)$ is also recursive. We have $f(x) = \sum_{i=1}^{k} g_1(x) \times (1 \dot{-} C_{P_i(x)})$, where $C_{P_i(x)}$ is the characteristic function of $P_i(x)$. $f(x)$ is primitive recursive because it is derived from compositions of addition, cutoff subtraction, $g_i(x)$ and $C_i(x)$, which are all primitive recursive functions. ■

8. Show that every primitive recursive function is total.

Proof: By induction on the definition of primitive recursive function.
Base case: $S(x), U_i^n(x^{(n)}), N(x)$ are all total function;
Inductive step: Let $f(x^{(n)}) = g(h_1(x^{(n)}), \dots, h_m(x^{(n)}))$. f is everywhere defined if g and h are total functions.
For primitive recursion, if $f(x^{(n)})$ and $g(x^{(n+2)})$ are total functions, then prove by induction on the first variable of $h(m, x^{(n)})$: $h(0, x^{(n)}) = f(x^{(n)})$, and $h(m+1, x^{(n)}) = g(m, h(m, x^{(n)}), x^{(n)})$. ■

9. Show that for each primitive recursive function there is a monotone primitive recursive function that is everywhere greater.

Proof: For each primitive recursive function $f(x)$, Let
$h(0) = f(0) + 1$
$h(z+1) = g(z, h(z)) = f(z+1) + h(z)$
Since f is primitive recursive, $h(z)$ is also primitive recursive. Also, for any natural number z,

$h(z) = \sum_{i=0}^{z} f(i) + 1 > f(i)$, therefore $h(z)$ is monotone. $\blacksquare$

10. Show that not all recursive functions are primitive recursive.

This is a research question. A full answer requires a proof that some function is not primitive recursive. One famous example is Ackermann function. The appending file provides one proof that Ackermann function is not primitive recursive.

Ackermann function is not primitive recursive*

†

2013-03-11 18:08:37

In this entry, we show that the Ackermann function $A(x,y)$, given by

$$A(0,y)=y+1, \qquad A(x+1,0)=A(x,1), \qquad A(x+1,y+1)=A(x,A(x+1,y))$$

is not primitive recursive. We will utilize the properties of A listed in this entry.

The key to showing that A is not primitive recursive, is to find a properties shared by all primitive recursive functions, but not by A. One such property is in showing that A in some way "grows" faster than any primitive recursive function. This is formalized by the notion of "majorization", which is explained here.

Proposition 1. *Let $\mathcal{A}$ be the set of all functions majorized by A. Then $\mathcal{PR} \subseteq \mathcal{A}$.*

Proof. We break this up into three parts: show all initial functions are in $\mathcal{A}$, show $\mathcal{A}$ is closed under functional composition, and show $\mathcal{A}$ is closed under primitive recursion. The proof is completed by realizing that $\mathcal{PR}$ is the smallest set satisfying the three conditions.

In the proofs below, $\boldsymbol{x}$ denotes some tuple of non-negative integers $(x_1,\dots,x_n)$ for some n, and $x=\max\{x_1,\dots,x_n\}$. Likewise for $\boldsymbol{y}$ and y.

1. We show that the zero function, the successor function, and the projection functions are in $\mathcal{A}$.
   - $z(n)=0<n+1=A(0,n)$, so $z\in\mathcal{A}$.
   - $s(n)=n+1<n+2=A(1,n)$, so $s\in\mathcal{A}$.
   - $p^k_m(x_1,\dots,x_k)=x_m\le x<x+1=A(0,x)$, so $p^k_m\in\mathcal{A}$.

*⟨*AckermannFunctionIsNotPrimitiveRecursive*⟩ created: ⟨*2013-03-11*⟩ by: ⟨*CWoo*⟩ version: ⟨*42019*⟩ Privacy setting: ⟨*1*⟩ ⟨*Theorem*⟩ ⟨*03D75*⟩

2. Next, suppose $g_1, \ldots, g_m$ are k-ary, and h is m-ary, and that each g_i, and h are in $\mathcal{A}$. This means that $g_i(\boldsymbol{x}) < A(r_i, x)$, and $h(\boldsymbol{y}) < A(s, y)$. Let

$$f = h(g_1, \ldots, g_m), \quad \text{and} \quad g_j(\boldsymbol{x}) = \max\{g_i(\boldsymbol{x}) \mid i = 1, \ldots, m\}.$$

   Then $f(\boldsymbol{x}) < A(s, g_j(\boldsymbol{x})) < A(s, A(r_j, x)) < A(s + r_j + 2, x)$, showing that $f \in \mathcal{A}$.

3. Finally, suppose g is k-ary and h is $(k+2)$-ary, and that $g, h \in \mathcal{A}$. This means that $g(\boldsymbol{x}) < A(r, x)$ and $h(\boldsymbol{y}) < A(s, y)$. We want to show that f, defined by primitive recursion via functions g and h, is in $\mathcal{A}$.

   We first prove the following claim:

$$f(\boldsymbol{x}, n) < A(q, n + x), \quad \text{for some } q \text{ not depending on } x \text{ and } n.$$

   Pick $q = 1 + \max\{r, s\}$, and induct on n. First, $f(\boldsymbol{x}, 0) = g(\boldsymbol{x}) < A(r, x) < A(q, x)$. Next, suppose $f(\boldsymbol{x}, n) < A(q, n+x)$. Then $f(\boldsymbol{x}, n+1) = h(\boldsymbol{x}, n, f(\boldsymbol{x}, n)) < A(s, z)$, where $z = \max\{x, n, f(\boldsymbol{x}, n)\}$. By the induction hypothesis, together with the fact that $\max\{x, n\} \leq n+x < A(q, n+x)$, we see that $z < A(q, n+x)$. Thus, $f(\boldsymbol{x}, n+1) < A(s, z) < A(s, A(q, n+x)) \leq A(q-1, A(q, n+x)) = A(q, n+1+x)$, proving the claim.

   To finish the proof, let $z = \max\{x, y\}$. Then, by the claim, $f(\boldsymbol{x}, y) < A(q, x+y) \leq A(q, 2z) < A(q, 2z+3) = A(q, A(2, z)) = A(q+4, z)$, showing that $f \in \mathcal{A}$.

Since $\mathcal{PR}$ is by definition the smallest set containing the initial functions, and closed under composition and primitive recursion, $\mathcal{PR} \subseteq \mathcal{A}$. □

As a corollary, we have

Corollary 1. *The Ackermann function A is not primitive recursive.*

Proof. Otherwise, $A \in \mathcal{A}$, which is impossible. □