浙江大学 2012-2013 学年 秋冬 学期

《计算理论》课程期末考试试卷答案

课程号：21120520 开课学院：计算机学院

考试试卷：☑ A卷 ☐ B卷

考试形式：☑ 闭卷 ☐ 开卷，允许带 ________入场

考试日期: 2013 年 1 月 18 日，考试时间：120 分钟

诚信考试，沉着应考，杜绝违纪

考生姓名________________ 学号________________ 所属院系________________

题序	1	2	3	4	5	6	总分
得分							
评卷人							

Zhejiang University
Theory of Computation, Fall-Winter 2012
Final Exam(Solution)

1. **(24%) Determine whether the following statements are true or false. If it is true fill a ○ otherwise a × in the bracket before the statement.**

   (a) (○) Language $\{xyz \mid x, y, z \in \{a, b\}^* \text{ and } x = z^R \text{ and } |x| \geq 1\}$ is regular.

   (b) (○) Let A and B be two regular languages, then $A \oplus B$ is also regular, where $A \oplus B = (A - B) \cup (B - A)$.

   (c) (×) Just as Turing Machine's encoding, every DFA M can also be encoded as strings "M", then the language {"M"| DFA M rejects "M"} is regular.

   (d) (×) Language $\{a^m b^n c^k d^l | m, n, k, l \in \mathbb{N}, m \geq l \text{ and } n \leq k\}$ is not context free.

   (e) (×) For languages L_1, L_2 and L_3, if $L_1 \subseteq L_2 \subseteq L_3$ and both L_1 and L_3 are context free, then L_2 is also context free.

   (f) (×) k-tapes Turing Machines can decide more languages than 1-tape Turing Machines.

   (g) (○) Language {"M" | Turing machine M halts on at least 2013 inputs} is recursively enumerable, but not recursive.

   (h) (○) The set of all primitive recursive functions is a proper subset of the set of all recursive functions.

   (i) (○) There exists a language L such that L is recursively enumerable, and $\overline{L}$ is recursive.

   (j) (×) Language {"M" | Turing machine M does not halt on "M"} is recursively enumerable.

   (k) (○) The recursively enumerable languages are closed under intersection, but not closed under complement.

   (l) (○) There are countably many Turing machines, and uncountably many languages, so most languages are not recursively enumerable.

2. **(20%)** Decide whether the following languages are regular or not and provide a formal proof for your answer.

   (a) $L_1 = \{a^m b^n c^k \mid m, n, k \in \mathbb{N} \text{ and } m \neq n + k\ \}$.

   (b) $L_2 = \{a^m b^n c^k \mid m, n, k \in \mathbb{N} \text{ and } (m \not\equiv (n + k)) \bmod 2\ \}$.

   Solution:

   (a) L_1 **is not regular.** $\cdots\cdots$ **5pt**
   Assume L_1 is regular, then $\overline{L_1}$ is also regular, therefore so is $\overline{L_1} \cap a^*b^*$. Let n be the constant whose existence the pumping theorem guarantees.
   – Choose string $w = a^n b^n \in \overline{L_1} \cap a^*b^*$, where $n \in \mathbb{N}$ and $n \geq 1$. So the pumping theorem must hold.
   – Let $w = xyz$ such that $|xy| \leq n$ and $y \neq e$, then $y = a^i$ where $i > 0$. But then $xz = a^{n-i}b^n \notin \overline{L_1} \cap a^*b^*$.
   The theorem fails, therefore $\overline{L_1} \cap a^*b^*$ is not regular, hence L_1 is not regular. $\cdots\cdots$ **5pt**

   (b) L_2 **is regular.** $\cdots\cdots$ **5pt**
   Since L_2 can be represented by the following regular expression:

   $$(aa)^*((bb)^*b(cc)^* \cup (bb)^*(cc)^*c) \cup (aa)^*a((bb)^*b(cc)^*c \cup (bb)^*(cc)^*).$$

   $\cdots\cdots$ **5pt**

3. **(20%)** Let $\Sigma = \{a, b, c\}$. Let $L_3 = \{w | w \in \{a, b, c\}^*, \#_b(w) = \#_c(w)\}$. Where $\#_z(w)$ is the number of appearances of the character z in w. For example, the string $x = baccabcbcb \in L_3$, since $\#_b(x) = \#_c(x) = 4$. Similarly, the string $x = abcaba \notin L_3$, since $\#_b(x) = 2$ and $\#_c(x) = 1$.

   (a) Construct a context-free grammar that generates the language L_3.

   (b) Construct a pushdown automata that accepts L_3.

   Solution: (a) The CFG for L_3 is $G = (V, \Sigma, S, R)$, where $V = \{S, A, a, b, c\}$, $\Sigma = \{a, b, c\}$, and $\cdots\cdots$ **3pt**

   $$R = \{S \to bSc|cSb|SS|AS|e, A \to aA|e\}.$$

   $\cdots\cdots$ **7pt**

   (b) The PDA $M = (K, \Sigma, \Gamma, \Delta, s, F)$ is defined below:

   $K = \{\underline{p, q}\}$

   $\Sigma = \{a, b, c\}$

   $\Gamma = \{\underline{S, A, a, b, c}\}$

   $s = \underline{p}$

   $F = \underline{\{q\}}$

(q, σ, β)	(p, γ)
(p, e, e)	(q, S)
(q, e, S)	(q, aSb)
(q, e, S)	(q, bSa)
(q, e, S)	(q, SS)
(q, e, S)	(q, AS)
(q, e, S)	(q, e)
(q, e, A)	(q, Aa)
(q, e, A)	(q, e)
(q, e, a)	(q, a)
(q, e, b)	(q, b)
(q, e, c)	(q, c)

$\cdots\cdots$ 10**pt**

4. **(20%)** The function $\varphi : \mathbb{N} \to \mathbb{N}$ given by

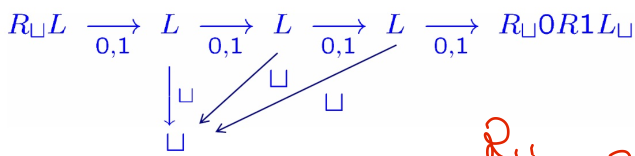
$$\varphi(x) = \begin{cases} x, & \text{if } x < 8 \\ 4x+1, & \text{if } x \geq 8 \end{cases}$$

(a) Try to construct a Turing Machine to compute the function $\varphi(x)$. When describing the Turing machines, you can use the elementary Turing machines described in textbook. Always assume that the Turing machines start computation from the configuration $\triangleright\underline{\sqcup}x$ where x is represented by binary string, i.e. $x \in \{0,1\}^*$.

(b) Show that the function $\varphi(x)$ is primitive recursive.

Solution:

(a) We can design the following Turing Machine to compute $\varphi(x)$:



$R_\sqcup$

R_c

$\cdots\cdots$ 10**pt**

(b) Since $\varphi(x)$ can be expressed by

$$\varphi(x) = (x < 8) \cdot x + (1 \sim (x < 8)) \cdot (4x+1)$$

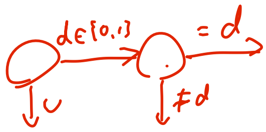
where $x < 8$ is a primitive recursive predicate and x and $4x+1$ are primitive recursive, therefore so is $\varphi(x)$.

$\cdots\cdots$ 10**pt**

ww^R

A

aSa
bSb



5. **(16%)** Let

$$L_4 = \{\text{“}M_1\text{”}\text{“}M_2\text{”} \mid M_1 \text{ and } M_2 \text{ are TMs, both } M_1 \text{ and } M_2 \text{ halt on the string } ab\}.$$

(a) Show that L_4 is recursively enumerable. An informal description suffices.

(b) Show that L_4 is not recursive.

Solution:

(a) On input “M_1”“M_2”, we can use Universal Turing machine to simulate both M_1 and M_2 on the string ab. If both M_1 and M_2 halt then we halt and “M_1”“M_2” $\in L_4$, otherwise continuous to simulation and the Universal Turing machine U always check the halting computation of M_1 and M_2 on string ab. Hence, L_4 is recursively enumerable. $\cdots\cdots$ 10**pt**

(b) We show that if there were an algorithm for L_4, then there would be an algorithm for solving the unsolvable halting problem $H = \{\text{“}M\text{”} \mid M \text{ halts on } e\}$.

Assume L_4 is decidable. Then, there exists a TM T that decides L_4. We can construct T_H that decides H using T:

T_H(“M”):

1. Construct TM M_1: Input y, if $y \neq ab$ reject; otherwise, simulate M on e, and if M halts on e, accept.
2. Construct TM M_2: Input y, if $y \neq ab$ and $y \neq e$ reject; otherwise, simulate M on e, and if M halts on e, accept.
3. Simulate T on "M_1" "M_2".
4. If T accepts "M_1" "M_2", accept.
5. If T rejects "M_1" "M_2", reject.

Then $L(M_1) = \{ab\}$ if M halts on e; $L(M_1) = \emptyset$ otherwise. $L(M_2) = \{e, ab\}$ if M halts on e; $L(M_2) = \emptyset$ otherwise.

This correctly decides H. If"M" $\in H$, then M halts on e, then $L(M_1) = \{ab\}$, and $L(M_2) = \{e, ab\}$, hence both M_1 and M_2 halt on ab, so T accepts "M_1""M_2" and then T_H accepts "M" in step 4. If "M" $\notin H$, then M does not halt on e, then neither M_1 nor M_2 halts on ab. So, T rejects "M_1""M_2" and then T_H rejects in step 5.

But the halting language H is known to be undecidable, this is a contradiction. Thus our assumption that there was a machine T deciding L_4 must have been incorrect. There is no machine deciding L_4. L_4 is not recursive.

$\cdots\cdots$ 6**pt**

Enjoy your Spring Festival!