2020 年度

東京大学大学院情報理工学系研究科 コンピュータ科学専攻 入学試験問題

専門科目 I

2019 年 8 月 20 日

10:00 – 11:30

注意事項

(1) 試験開始の合図があるまで, この問題冊子を開けないこと.

(2) 3 題すべてに答えよ. 問題ごとに指定された解答用紙を使用すること.

(3) 解答用紙および問題冊子は持ち帰らないこと.

Specialized Subjects I

10:00 – 11:30, August 20, 2019

Entrance Examination (AY 2020)
Department of Computer Science, Graduate School of Information Science and Technology
The University of Tokyo

Notice:

(1) Do not open this problem booklet until the start of the examination is announced.

(2) Answer the following 3 problems. Use the designated answer sheet for each problem.

(3) Do not take the problem booklet or any answer sheet out of the examination room.

下欄に受験番号を記入すること.

Write your examinee's number in the box below.

受験番号	No.

問題 1

Σ を有限アルファベット（すなわち記号の有限集合）とし，ϵ を空列とする．2 つの語 $w_1, w_2 \in \Sigma^*$ のシャッフル $w_1 \otimes w_2 \subseteq \Sigma^*$ を以下によって定義する．

- すべての $w \in \Sigma^*$ について
$$\epsilon \otimes w = w \otimes \epsilon = \{w\}.$$
- すべての $a, b \in \Sigma$ および $w_1, w_2 \in \Sigma^*$ について，
$$(aw_1) \otimes (bw_2) = \{aw \mid w \in w_1 \otimes (bw_2)\} \cup \{bw \mid w \in (aw_1) \otimes w_2\}.$$

さらに，2 つの言語 $L_1, L_2 \subseteq \Sigma^*$ に対し，それらのシャッフル $L_1 \otimes L_2 \subseteq \Sigma^*$ を以下によって定義する．

$$L_1 \otimes L_2 = \bigcup_{w_1 \in L_1, w_2 \in L_2} w_1 \otimes w_2.$$

例えば，

$$\{ab, ba\} \otimes \{\epsilon, c\} = (ab \otimes \epsilon) \cup (ab \otimes c) \cup (ba \otimes \epsilon) \cup (ba \otimes c) = \{ab, cab, acb, abc, ba, cba, bca, bac\}$$

である．

以下の問いに答えよ．

(1) $\{a, bb\} \otimes \{ab, cc\}$ を求めよ．

(2) 決定性有限オートマトン $\mathcal{A}_1 = (Q_1, \Sigma, \delta_1, q_{1,0}, F_1)$ および $\mathcal{A}_2 = (Q_2, \Sigma, \delta_2, q_{2,0}, F_2)$ がそれぞれ言語 L_1 および L_2 を受理するものとする．ただし $Q_i, \delta_i, q_{i,0}, F_i$ は $\mathcal{A}_i$ $(i \in \{1, 2\})$ の状態集合，遷移関数，初期状態，受理状態の集合をそれぞれ表すものとし，遷移関数 $\delta_i \in Q_i \times \Sigma \to Q_i$ $(i \in \{1, 2\})$ は全域関数であると仮定してよい．$L_1 \otimes L_2$ を受理する非決定性オートマトンを与えよ．

(3) 問い (2) の答えが正しいことを証明せよ．

(4) $L_3 = \{a^n b^n \mid n \geq 0\}$，$L_4 = \{c^m d^m \mid m \geq 0\}$ とする．$L_3 \otimes L_4$ は文脈自由言語ではないことを証明せよ．なお，文脈自由言語の反復補題を用いてもよい．

Problem 1

Let Σ be a finite alphabet (i.e., a finite set of letters), and ϵ be the empty sequence. We define the *shuffle* $w_1 \otimes w_2 \subseteq \Sigma^*$ of two words $w_1, w_2 \in \Sigma^*$ as follows.

- For every $w \in \Sigma^*$,
$$\epsilon \otimes w = w \otimes \epsilon = \{w\}.$$
- For every $a, b \in \Sigma$ and $w_1, w_2 \in \Sigma^*$,
$$(aw_1) \otimes (bw_2) = \{aw \mid w \in w_1 \otimes (bw_2)\} \cup \{bw \mid w \in (aw_1) \otimes w_2\}.$$

Furthermore, for two languages $L_1, L_2 \subseteq \Sigma^*$, their shuffle $L_1 \otimes L_2 \subseteq \Sigma^*$ is defined by:

$$L_1 \otimes L_2 = \bigcup_{w_1 \in L_1, w_2 \in L_2} w_1 \otimes w_2.$$

For example, we have:

$$\{ab, ba\} \otimes \{\epsilon, c\} = (ab \otimes \epsilon) \cup (ab \otimes c) \cup (ba \otimes \epsilon) \cup (ba \otimes c) = \{ab, cab, acb, abc, ba, cba, bca, bac\}.$$

Answer the following questions.

(1) Compute $\{a, bb\} \otimes \{ab, cc\}$.

(2) Suppose that deterministic finite automata $\mathcal{A}_1 = (Q_1, \Sigma, \delta_1, q_{1,0}, F_1)$ and $\mathcal{A}_2 = (Q_2, \Sigma, \delta_2, q_{2,0}, F_2)$ accept languages L_1 and L_2, respectively. Here, $Q_i, \delta_i, q_{i,0}$, and F_i are respectively the set of states, the transition function, the initial state, and the set of final states of $\mathcal{A}_i$ ($i \in \{1, 2\}$). You may assume that the transition functions $\delta_i \in Q_i \times \Sigma \to Q_i$ ($i \in \{1, 2\}$) are total functions. Give a non-deterministic automaton that accepts $L_1 \otimes L_2$.

(3) Prove the correctness of your answer for question (2) above.

(4) Let $L_3 = \{a^n b^n \mid n \geq 0\}$ and $L_4 = \{c^m d^m \mid m \geq 0\}$. Prove that $L_3 \otimes L_4$ is not a context-free language. Here, you may use the pumping lemma for context-free languages.

問題 2

下記のCコードは，食事する哲学者の問題における各哲学者の振舞いをモデル化したものである．

```
void philosopher(int i) {
  do {
    pickup(i);
    eat();
    putdown(i);
    think();
  } while (1);
}
```

マルチプロセッサシステム上で5つのスレッドが並行に動作しており，各スレッドはphilosopher関数を実行するものとする．引数iはスレッド番号であり，i= 0, 1, ..., 4とする．philosopher関数は，eat関数とthink関数を交互に繰り返し実行するが，隣り合った哲学者（各i= 0, 1, ..., 4についてi番目と(i+1)%5番目のスレッド）が同時にeat関数を実行しないように，eat関数の実行の前後ではpickup関数とputdown関数を用いてスレッドの同期をとる．以下では，バイナリセマフォを用いてpickup関数とputdown関数を実現する問題について考える．ここで，Cコード中ではバイナリセマフォXに対するP操作とV操作はそれぞれwait(X)とsignal(X)で表現するものとし，各バイナリセマフォのカウンタ値は1に初期化されているものとする．また，eat関数とthink関数は，それぞれの関数外に影響を与える副作用を発生させないものとする．

以下の問いに答えよ．

(1) 各i= 0, 1, ..., 4について，R[i]をバイナリセマフォとする．以下のpickup関数とputdown関数を用いるとデッドロックが起きる可能性がある．どのような場合にデッドロックが起こるか示せ．

```
void pickup(int i) {
  wait(R[i]);
  wait(R[(i+1)%5]);
}
```

```
void putdown(int i) {
  signal(R[i]);
  signal(R[(i+1)%5]);
}
```

(2) 各i= 0, 1, ..., 4について，S[i]をバイナリセマフォとし，state[i]をi番目のスレッドの状態を表す共有変数とする．また，mutexを全スレッド間の排他制御を実現するためのバイナリセマフォとする．デッドロックを起こさずに少なくとも1つのスレッドがeat関数とthink関数を繰り返し実行できるようにするために，pickup関数とputdown関数を以下のように書き換える．void型のtest関数は，特定の条件が満たされたときのみ，state[i]をeatingに設定してからS[i]に対してsignal関数を実行する．test関数をCコードで作成せよ．ここで各スレッドが飢餓状態に陥ることは考慮しなくてよい．また，state[i]の初期値はthinkingとする．

```
enum {thinking, eating, waiting} state[5];

void pickup(int i) {
  wait(mutex);
  state[i] = waiting;
  test(i);
  signal(mutex);
  wait(S[i]);
}
```

```
void putdown(int i) {
  wait(mutex);
  state[i] = thinking;
  test((i+4)%5);
  test((i+1)%5);
  signal(mutex);
}
```

(3) 問い(2)のCコードにおいて，実行可能なスレッドはいずれ必ずスケジュールされるものとした場合に，スレッドが飢餓状態に陥る可能性があるかどうかを答えよ．可能性があるならば，どのように飢餓状態が起こるかを述べるとともに，飢餓状態を回避するためのコードの修正方法を簡単に説明せよ．可能性がないならば，その理由を説明せよ．

Problem 2

The following C code models the behavior of each philosopher in the dining philosophers problem.

```
void philosopher(int i) {
  do {
    pickup(i);
    eat();
    putdown(i);
    think();
  } while (1);
}
```

There are five threads running concurrently on a multiprocessor system, each of which executes the `philosopher` function. The argument `i` is the index of each thread, where `i`$= 0, 1, ..., 4$. In the `philosopher` function, the `eat` and `think` functions are executed in turn repeatedly, while the `pickup` and `putdown` functions are used for synchronization between threads, respectively, before and after the execution of the `eat` function so that two philosophers sitting side by side (namely, the `i`-th and the `(i+1)%5`-th threads for each `i`$= 0, 1, ..., 4$) cannot simultaneously execute the `eat` function. Now consider the problem of implementing the `pickup` and `putdown` functions, using binary semaphores. Here, the P and V operations on a binary semaphore `X` are respectively expressed as `wait(X)` and `signal(X)` in C code, and the counter value of each binary semaphore is initialized to 1. Also, assume that the `eat` and `think` functions never cause a side effect that may influence the outside of each function.

Answer the following questions.

(1) For each `i`$= 0, 1, ..., 4$, let `R[i]` be a binary semaphore. A deadlock may occur when the following `pickup` and `putdown` functions are used. Describe how such a deadlock may occur.

```
void pickup(int i) {
  wait(R[i]);
  wait(R[(i+1)%5]);
}
```

```
void putdown(int i) {
  signal(R[i]);
  signal(R[(i+1)%5]);
}
```

(2) For each `i`$= 0, 1, ..., 4$, let `S[i]` be a binary semaphore, and `state[i]` be a shared variable that represents the state of the `i`-th thread. Also, let `mutex` be a binary semaphore that is used to achieve a mutual exclusion on all the threads. In order for at least one thread to be able to execute the `eat` and `think` functions repeatedly without a deadlock, the `pickup` and `putdown` functions are redefined as follows. The `void`-type `test` function sets `state[i]` to `eating` and calls the `signal` function for `S[i]`, if a certain condition is satisfied. Describe the `test` function using C code. Note that you need not consider starvation of each thread. Also, assume that the initial value of `state[i]` is `thinking`.

```
enum {thinking, eating, waiting} state[5];
```

```
void pickup(int i) {
  wait(mutex);
  state[i] = waiting;
  test(i);
  signal(mutex);
  wait(S[i]);
}
```

```
void putdown(int i) {
  wait(mutex);
  state[i] = thinking;
  test((i+4)%5);
  test((i+1)%5);
  signal(mutex);
}
```

(3) Regarding the C code in question (2), answer whether or not a thread may suffer from starvation, assuming that any enabled thread is eventually scheduled. If your answer is "yes", describe how the starvation occurs and briefly explain how to modify the code to avoid the starvation. If your answer is "no", then explain the reason.

問題 3

この問題では，文字列 s の長さを $\ell(s)$ と書く．文字列 s の i 番目の文字を $s[i]$ と書き，最初の文字は $s[0]$ である．また s の最初の i 文字を除いて得られる文字列を $s+i$ と書く．これらの表記では $0 \leq i < \ell(s)$ を仮定する．例えば $s = \mathrm{PROBLEM}$ のとき $s[0] = \mathrm{P}$ であり，$s+3 = \mathrm{BLEM}$ である．また，文字の種類は N 個（N は 2 以上の整数定数）であり，各文字 c に対して相異なる N 以下の正整数 $numval(c)$ が対応するものとする．なお s と i から $s+i$ を求める演算，ならびに c から $numval(c)$ を求める演算は $O(1)$ の計算時間でできるものとする．また整数の加算・乗算・剰余の計算は $O(1)$ の時間でできるものとし，整数演算でオーバーフローはおきないものとする．

文字列 p と s が与えられたとき，s の中で p とマッチする最初の位置 i，すなわち

$$\forall j \in \{0, 1, \ldots, \ell(p)-1\}. \quad s[i+j] = p[j]$$

を満たす最小の非負整数 i を見つける問題 FIND を考える．ただしそのような i がないときには $i = -1$ とするものとする．以下，$\ell(s) > \ell(p) > 0$ とする．

関数 $eq(r, p)$ は，$0 < \ell(p) \leq \ell(r)$ を満たす文字列 r, p に対し，r の最初の $\ell(p)$ 文字が p と一致するときに 1, それ以外では 0 を返す関数で，時間計算量は $O(\ell(p))$ とする．問題 FIND を解く以下のアルゴリズムを「アルゴリズム S」と呼ぶことにする．

```
for (i = 0; i <= ℓ(s) − ℓ(p); i++)
    if (eq(s + i, p) == 1)
        return i;
return −1;
```

以下の問いに答えよ．

(1) アルゴリズム S の最悪時時間計算量のオーダーを $\ell(s)$ と $\ell(p)$ を用いて示せ．

以下では，文字列 s に対して最初の m 文字のハッシュ値 $h(s, m)$ を

$$h(s, m) = \left(\sum_{i=0}^{m-1} numval(s[i]) \cdot d^{m-i-1} \right) \bmod q$$

で定める．ここで d と q は正整数の定数で，$0 < m \leq \ell(s)$ とする．

(2) $i < \ell(s) - m$ とし，$h' = h(s+i, m)$ と $d_m = d^{m-1}$ は計算してあるものとする．このとき，$h(s+i+1, m)$ の値を $O(1)$ で求めるアルゴリズムまたは式を示せ．

(3) $h(p, \ell(p)) = h(s+i, \ell(p))$ となる最小の非負整数 i を $O(\ell(s)+\ell(p))$ 時間で求める（ただし等号を満たす i がなければ -1 を答える）アルゴリズム H_0 を答えよ．

また，アルゴリズム H_0 が求める値が問題 FIND の解でないのはどのような場合か答えよ．

(4) 以下の条件をすべて満たすアルゴリズム H を示せ．(a) 問題 FIND の解を常に与える．(b) ハッシュ値 $h(s, m)$ と関数 $eq(r, p)$ を用いて解を探す．(c) 各 s, p に対して $h(p, \ell(p)) = h(s+i, \ell(p))$ を満たす i の個数が s, p に依らず $O(1)$ と仮定したとき，$O(\ell(s)+\ell(p))$ 時間で実行できる．

さらに，アルゴリズム H の時間計算量が $O(\ell(s)+\ell(p))$ より大きくなるのはどのような場合か答えよ．また，アルゴリズム H の最悪時の時間計算量のオーダーを $\ell(s)$ と $\ell(p)$ を用いて表せ．

Problem 3

In this problem, the length of a string s is written $\ell(s)$, and the ith character of s is written $s[i]$, where the first character is $s[0]$. The string obtained by removing the first i characters from s is written $s + i$. We assume $0 \leq i < \ell(s)$ in $s[i]$ and $s + i$. For example, if s = PROBLEM, then $s[0]$ = P and $s + 3$ = BLEM. The set of characters consists of N characters, where N is an integer constant no less than 2, and for each character c a distinct positive integer $numval(c) \leq N$ is defined. Suppose that the computation of $s+i$ for given s and i, and that of $numval(c)$ for given c, take $O(1)$ time. Also suppose that each of integer addition, multiplication and remainder takes $O(1)$ time, and that overflow will never occur in integer operations.

We consider the following problem FIND: For given strings p and s, find the first position i at which s matches p. In other words, i is the least non-negative integer that satisfies

$$\forall j \in \{0, 1, \ldots, \ell(p) - 1\}. \quad s[i + j] = p[j].$$

In case there is no such i, we define $i = -1$. In the following, we assume $\ell(s) > \ell(p) > 0$.

For strings r and p with $0 < \ell(p) \leq \ell(r)$, let function $eq(r, p)$ return 1 if the first $\ell(p)$ characters of r equal p, and return 0 otherwise. Suppose that the time complexity of $eq(r, p)$ is $O(\ell(p))$. The following algorithm S solves the problem FIND:

```
for (i = 0; i <= ℓ(s) − ℓ(p); i ++)
    if (eq(s + i, p) == 1)
        return i;
return −1;
```

Answer the following questions.

(1) Express the order of the worst-case time complexity of algorithm S in terms of $\ell(s)$ and $\ell(p)$.

In the following, the hash value $h(s, m)$ of the first m characters of string s is defined by

$$h(s, m) = \left(\sum_{i=0}^{m-1} numval(s[i]) \cdot d^{m-i-1} \right) \bmod q,$$

where d and q are positive integer constants, and $0 < m \leq \ell(s)$ is assumed.

(2) Assume that $i < \ell(s) - m$ holds, and that $h' = h(s + i, m)$ and $d_m = d^{m-1}$ have been precomputed. Show an algorithm or an expression to compute $h(s + i + 1, m)$ in $O(1)$ time.

(3) Give an algorithm H_0 that finds the least non-negative integer i that satisfies $h(p, \ell(p)) = h(s + i, \ell(p))$ (but answers -1 if no such i exists) in time $O(\ell(s) + \ell(p))$.

Also, answer in what condition the algorithm H_0 outputs a value which **is not** the solution of problem FIND.

(4) Give an algorithm H that satisfies all of the following conditions: (a) it always answers the solution of problem FIND, (b) it searches for the answer by using hash $h(s, m)$ and function $eq(r, p)$, and (c) if we assume that the number of integers i that satisfy $h(p, \ell(p)) = h(s + i, \ell(p))$ for given s and p is $O(1)$ independently of s and p, then the algorithm H runs in time $O(\ell(s) + \ell(p))$.

In addition, show in what condition the time complexity of the algorithm H is larger than $O(\ell(s) + \ell(p))$. Also, answer the order of the worst-case time complexity of the algorithm H in terms of $\ell(s)$ and $\ell(p)$.