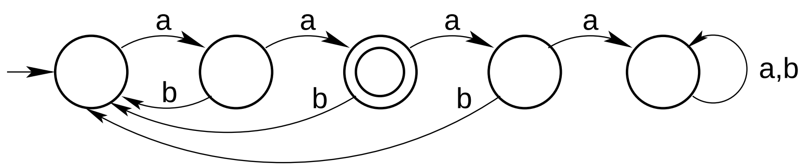
CpSc 421 Final Exam **December 6, 2008**

Do problem 0 and any **five** of problems 1-10.
If you attempt more than five of problems 1-10, please indicate which ones you want graded – otherwise, I'll make an arbitrary choice.

Graded on a scale of 100 points.
You can attempt from 68 to 110 points depending on which problems you choose. If you score over 100, you get to keep the extra credit.

1. (**12 points**)

   (a) (**6 points**) Write a regular expression that generates the language that is recognized by the DFA below:



   (b) (**6 points**) Draw the state transition diagram for a DFA or NFA that recognizes the language generated by the regular expression $(ab)^*(aba)^*$.

2. (**12 points**) Let $G$ be the CFG with start variable $S$, terminals a and b, and rules:

$$\begin{array}{rcl} S & \rightarrow & T \mid U \\ T & \rightarrow & \epsilon \mid \mathsf{a}T\mathsf{bb} \\ U & \rightarrow & \epsilon \mid \mathsf{ba}U\mathsf{a} \end{array}$$

   (a) Which of the strings below are generated by $G$?

   aabbbb  baa  bbaaaa  aaaabb  bababaaaa  baabaa

   (b) Write a grammar that generates the language:

$$\{a^m b^n \mid (m = 2n + 3) \vee (3m = 5n)\}$$

3. (**12 points**) Let Equiv be a Java method that takes as its input the source code for two Java methods and determines whether or not they return the same values for all possible values of their input parameters. For example, consider the methods:

```
int mult1(int a, int b) {
    return(a + b);
}
```

```
int mult2(int a, int b) {
    int p = 0;
    for(int i = 0; i < a; i++)
        p = p+b;
    return(p);
}
```

```
int mult3(int a, int b) {
    return(a & b);
}
```

Equiv(⟨mult1⟩, ⟨mult2⟩) should return true, but Equiv(⟨mult1⟩, ⟨mult3⟩) should return false. Here, ⟨mult1⟩ denotes the source code for method mult1 and likewise for the other methods.

Show that it is impossible to write an implementation of Equiv that works for all possible Java methods. In particular, show that if a method such as Equiv could be written, then you could use it to solve the halting problem. Assume that all of these methods can use unlimitted amounts of memory.

4. (**12 points**)

   (a) (**2 points**) Give a one or two sentence definition of what it means for a language to be in NP.

   (b) (**2 points**) Give a one or two sentence definition of what it means for a language to be NP hard.

   (c) (**2 points**) Give a one or two sentence definition of what it means for a language to be NP complete.

   (d) (**6 points**) State which of the languages below are in NP, which are NP hard, and which are NP complete:

      i. Strings consisting of an equal number of a's and b's.

      ii. Strings that describe a Turing machine that halt when run with the empty string as its input.

      iii. Strings that describe a satisfiable Boolean formula.

      iv. Strings that are the binary encoding of prime number.

      v. Strings that describe a graph, $G = (V, E)$, and an integer $k$, such that there is a set $U \subseteq V$ with $|U| = k$ such that for every edge $(v_1, v_2) \in E$ either $v_1 \in U$ or $v_2 \in U$ or both.

      vi. The empty language.

5. (**20 points**) A context-free grammar is *right linear* iff every rule is of the form $A \rightarrow xB$ or $A \rightarrow x$ where $A$ and $B$ are variables and $x$ is a string of terminals. Prove that if $G$ is a right-linear context-free grammar, then $L(G)$ is regular.

6. (**20 points**) Let $A = \{\langle M \rangle \mid L(M) = (L(M))^{\mathcal{R}}\}$ where $\langle M \rangle$ is a string describing Turing machine $M$, and $(L(M))^{\mathcal{R}}$ is the language consisting of all strings whose reversals are in $L(M)$.

   Determine whether or not $A$ is Turing decidable, and give a short proof.

7. (**20 points**) Let $spaceBound(M, w, n)$ be true iff Turing machine $M$ accesses at most $n$ tape squares when run with input $w$. Let

$$\begin{aligned} B_1 &= \{M\#w \mid spaceBound(M, w, 2^{|w|})\} \\ B_2 &= \{M \mid \forall w.\ spaceBound(M, w, 2^{|w|})\} \end{aligned}$$

   One of these langauages is decidable and one is not. Determine which is which and give a short proof for each answer.

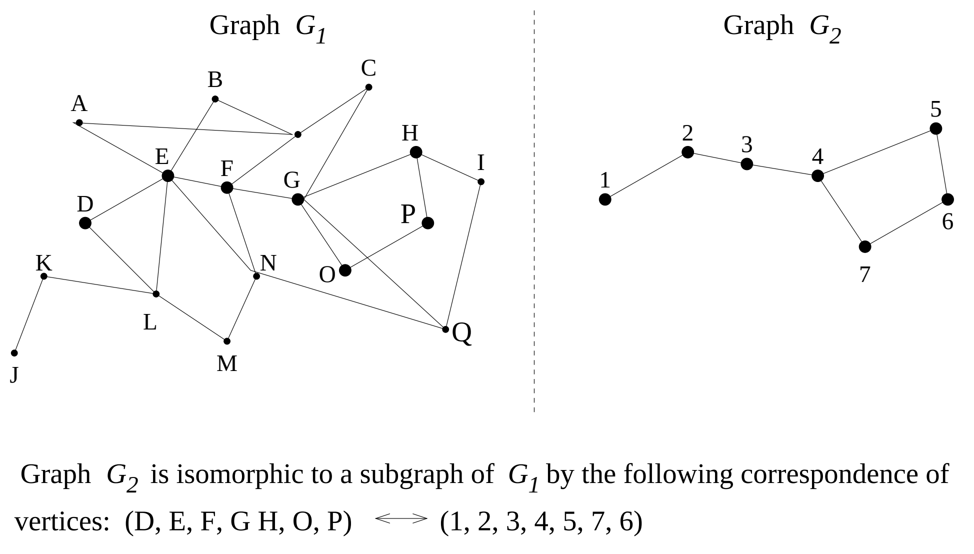

Graph $G_1$
Graph $G_2$
A
B
C
D
E
F
G
H
I
J
K
L
M
N
O
P
Q
1
2
3
4
5
6
7
Graph $G_2$ is isomorphic to a subgraph of $G_1$ by the following correspondence of vertices: (D, E, F, G H, O, P) $\longleftrightarrow$ (1, 2, 3, 4, 5, 7, 6)

Figure 1: A subgraph isomorphism example.

8. (**20 points**) Let $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ be graphs with nodes $V_i$ and edges $E_i$ (with $i \in \{1, 2\}$). The SUBGRAPH ISOMORPHISM problem is to determine whether or not $G_2$ is isomorphic to a subgraph of $G_1$. Show that SUBGRAPH ISOMORPHISM is NP-complete.

   Hints:

   - Consider a reduction from CLIQUE.
   - "Graph $G$ is a *subgraph* of graph $H$ if the nodes of $G$ are a subset of the nodes of $H$, and the edges of $G$ are the edges of $H$ on the corresponding nodes." (Sipser, page 11).
   - Graphs $G$ and $H$ are *isomorphic* if the nodes of $G$ can be reordered so that it is identical to $H$. (see Sipser, exercise 7.11).
   - See figure 1.

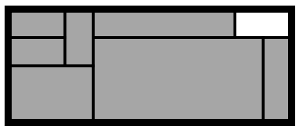
9. (**25 points**) Let $D$ be a DFA, $G$ be a CFG, and $M$ be a TM. Define

$$\begin{aligned} A(M,D) &= \{y \mid \exists x \in L(M).\ xy \in L(D)\} \\ B &= \{\langle M, D, y\rangle \mid y \in A(M,D)\} \\ C(M,G) &= \{y \mid \exists x \in L(M).\ xy \in L(G)\} \end{aligned}$$

   (a) (**9 points**) Prove that $A(M, D)$ is regular.

   (b) (**8 points**) Prove that $B$ is Turing recognizable but not Turing decidable.

   (c) (**8 points**) Prove that $C(M, G)$ is not necessarily context free.



Rectangles (1, 2), (1, 2), (1, 2), (1,3), (1,5), (2,3), and (3,6)
can be packed into rectangle (10,4).

Figure 2: A subgraph isomorphism example.

10. (**25 points**) Let $Q, R_1, R_2, \ldots R_k$ be rectangles. Each rectangle is specified by two integers, one for its width and the other for its height. The RECTANGLE PACKING problem is to determine if it is possible to place rectangles $R_1 \ldots R_k$ in rectangle $Q$ such that none of the $R_i$ rectangles overlap. Each rectangle may be placed at an arbitrary location and with an arbitrary orientation in $Q$ as long as it is contained completely in $Q$. See figure 2 for an example.

    Show that RECTANGLE PACKING is strongly NP-complete.

    For **20 points**, you can show that RECTANGLE PACKING is NP-complete (without showing the *strongly* part required for a 25 point solution).