Cpt S 516 Homework # 0

Electronic Hand-in (in PDF and generated from LATEX, to me) before the deadline
Absolutely no late homework!

This set of homework is to make sure you have all the needed backgraound for this course. A typical student will solve most of the problems in 4 hours. Sisper's book is a good reference (if you need background materials).

1. (standard, 20pt) A language is a set of words over a given alphabet Σ. We say that word $w = a_1 \cdots a_n$, for some n, is a *shuffle* of word w_1 and word w_2 if one can find k (for some $k \geq 0$) positions $1 \leq i_0 < \cdots < i_k \leq n$ in w such that the sequence of symbols on these positions is exactly w_1 and, once the symbols on these positions are removed, the w becomes w_2. Given two languages L_1 and L_2, we use $L_1 \| L_2$ to denote the language of all words w such that, for some $w_1 \in L_1$ and $w_2 \in L_2$, w is a shuffle of w_1 and w_2. Prove that (1). If L_1 and L_2 are regular languages, then so is $L_1 \| L_2$. (2). If L_1 is a regular language and L_2 is a context-free language, then $L_1 \| L_2$ is a context-free language.

2. (standard, 20pt) Let Σ be an alphabet. For a word $w = a_1 \cdots a_n$ and a (total) function h from Σ to Σ, we use $h(w)$ to denote $h(a_1) \cdots h(a_n)$. For a language L over the alphabet, we use $h(L)$ to denote $\{h(w) : w \in L\}$. Prove that (1). If L is a regular language, then so is $h(L)$. (2). There is a non-regular language L such that $h(L)$ is a regular language for some h.

3. (standard, 20pt) Let Σ be an alphabet. Given two words w_1 and w_2, a *match* between w_1 and w_2 is a word w satisfying $w_1 = w_1' w$ and $w_2 = w w_2'$ for some w_1' and w_2'. Given two languages L_1 and L_2, we use $L_1 \heartsuit L_2$ to denote the set of matches between w_1 and w_2 for all $w_1 \in L_1, w_2 \in L_2$. Prove that if L_1 and L_2 are regular languages, then so is $L_1 \heartsuit L_2$.

4. (standard, 20pt) A B-bounded PDA (pushdown automaton) is a PDA M such that it crashes whenever its stack height reaches B. Show that the language $\{0^n 1^n : n \geq 1\}$ can not be accepted by a B-bounded PDA for any B.

5. (hard, 20pt) Notice that a PDA could have a move that does not read from the input (only performs stack operation and state transition). Therefore, it

is possible for the PDA to run forever on some given input. Prove that there is an algorithm that answers whether, for a given PDA, there is an input word on which the PDA has an infinite execution.

6. (hard, 20pt) Let Σ be a finite set of colors, and G be a directed graph on which each arc is labeled with a color in Σ. An ω-*walk* is an infinite path

$$v_0 c_0 v_1 c_1 v_2 \cdots$$

on G such that, for each i, G has an arc from v_i to v_{i+1} with color c_i. In particular, we say the walk *starts from* v_0. The walk is *fair* if there are infinitely many prefixes of the walk on which the number of red arcs, the number of green arcs and the number of blue arcs are all the same. Prove that there is an algorithm to decide whether the graph G has a fair walk starting from a given node v_0.

7. (research, 20pt) In finite automata, there is no concept about time; or we can think in this way: each move takes one unit time. But in reality (e.g., in designing an embedded system), timing is an explicit requirement which can be expressed through timers. Timers can be used to control moves of the automata. Investigate your ways of adding timers to the definition of finite automata (hence your extended finite automata are able to describe a simple class of real-time systems).

8. (Putting theory to Practice, 20pt — you must do this one!) As I talked in class, machines/programs are to regulate event sequences. In theory, we study various automata (such as finite automata, etc) to accept a regular language. However, in practice, trying to regulate a desired sequence pattern for events is not easy at all. here is a problem that you want to solve.

There are two kids A and B and there are two apples c, d on the table. Each kid wants to pick up an apple from the table, holds it for a while, and puts it back to the table. Therefore, there are the following kinds of events:

A idles;
B idles;
A picks up c;
A picks up d;
B picks up c;
B picks up d;

A puts back c;
A puts back d;
B puts back c;
B puts back d.

Notice that, as usual, we assume the interleaving model: at any moment there is at most one event is happending (e.g., A and B can not pick up apple at the same time – this is not possible in computer science).

Intially, A and B holds no apple and the two apples c and d are on the table. I want to regulate the event sequences so that at any time, any kid can not hold two apples. Obviously, you need additional control on how the kids are going to behave. Can you design such a control for the kids?