CMSC452 Elementary Theory of Computation, Fall 2001

Sketch of Solutions: Homework 8

Note: For minor errors, 4 or 5 partial credits are given to each graded unit worthy of 8 or 10 points, respectively.

Problem 5.2.2 [10 points]

Suppose there were a universal finite automaton U. The U would have some fixed number of states k, and could never simulate any finite automaton with more than k states.

Formally, suppose that given M with k states have n equivalence classes where $n>k$. Then there are n strings w_1, w_2,…, w_n which must drive M to distinct states q_1, q_2, … q_n. Thus, "M""w_1", "M""w_2",…, "M""w_n" must drive U to n distinct states, a contradiction.

Problem 5.3.2 [10+10=20 points]

For problems 5.3.2 and 5.3.3, by theorem 4.5.1, if a non-deterministic Turing machine semi-decides (or decides) a language L, there is a standard Turing machine which semi-decides (or decides) L. Thus, to show the existence of a non-deterministic Turing machine suffices.

Suppose that given Turing machines M_1 and M_2 semi-decide $L(M_1)$ and $L(M_2)$, respectively.

1. For union, define non-deterministic Turing machine M non-deterministically to simulate M_1 and M_2 on given input w. It accepts when the chosen machine accepts. Then, M accepts $L(M')=L(M_1)\cup L(M_2)$. [10 points]
2. For intersection, define a 2-tape Turing machine M' which does the following on input w. First, it copies w onto the second tape. Then it runs M_1 on its first tape. If M_1 halts, M' runs M_2 on the second tape, halting if M_2 halts. Then M' accepts $L(M')=L(M_1)\cap L(M_2)$. [10 points]

Problem 5.3.3 [8×5=40 points]

1. For union, given Turing machines M_1 and M_2 which decide $L(M_1)$ and $L(M_2)$ respectively, define 2-tape Turing machine M to copy the given input w into the 2^{nd} tape and simulate both M_1 and M_2 on the 2 tapes respectively. It rejects when both machines rejects, accepts when either machine accepts. Then M decides the language $L(M)=L(M_1)\cup L(M_2)$.. [8 points]
2. For complementation, given Turing machine M decides the language L(M), let M' be the Turing machine which is the same as M except that it has the states y and n interchanged. The M' decides the language $L(M)^c=\sum^*-L(M)$. [8 points]
3. For concatenation, given Turing machine M decides the language L(M), let M" be non-deterministic 2-tape Turing machine which non-deterministically splits given input w into two strings x and y, and copies y to its second tape and leaves x on the first tape. M" then runs M_1 on its first tape; if M_1 accepts, M" runs M_2 on its second tape. Then M" decides $L(M")=L(M_1)L(M_2)$. [8 points]

4. For Kleene star, given Turing machine M decides the language L(M), let M* be the non-deterministic 2-tape Turing machine, which non-deterministically splits given input w into strings w_1, w_2,…,w_k. Then for each w_i, i=1,2,…,k, M* copy it to its second tape and runs M on w_i, If M accepts all the w_i,s then M* accepts w. Then M* decides L(M*)=L(M)*. [8 points]
5. For intersection, since $L_1 \cap L_2 = (L_1^c \cup L_2^c)^c$, the proven closure properties of union and complementation suffice.[8 points]

Problem 5.4.2 (a)(c) [10+10=20 points]

(a) It is undecidable. Suppose it could be solved by some machine G. Given a Turing machine M with the halting state h and a string w, we could run G with input (M,w,h). If there are multiple halting states, we could run G repeatedly for each of them. This would make halting problem decidable. It contradicts to theorem 5.3.1. [10 points]

(c) It is solvable. We can examine δ to see whether there is a rule of the form $\delta(p,r)=(q,s)$. If there exists such a rule, then there is such a configuration. For example, $(p,\blacktriangleright \sqcup \underline{r})$). If it does not exist, then no such a configuration exists. Thus, it is solvable. [10 points]

Problem 5.5.1 [10 points]

<Claim:> Given a pushdown automaton M with one state, M accepts all strings if and only if it accepts all strings of length one.

<Proof of claim:> Since M has only one state, the final state is just the start state. If it accepts all strings of length one, $(s,\sigma,e) \vdash_M^* (s,e,e)$ for each $\sigma \in \Sigma$. Then, by induction, we can easily show that $(s,w,e) \vdash_M^* (s,e,e)$ for any given string $w \in \Sigma^*$. On the other hand, it is trivial that if M does not accept some string of length 1, then it cannot accept all the strings.

With the proven claim, the problem of determining whether a given pushdown automaton M with only 1 state accepts all strings of length one is decidable, by converting M to a context-free grammar G, converting G to an equivalent grammar G' in Chomsky Normal Form, and then running the dynamic programming algorithm on G' with each string of length one.