600.271 Automata & Computation Theory
Second Mid-Semester Examination
In-class, Closed Book
April 11, 2012
Time: 1 hr, 10 minutes

I. (10 pts.) Design a context-free grammar for the language $\{xcy \mid x, y \in \{a,b\}^*,\ |x| = |y|,\ x \neq y^R, \text{ and } |x| \text{ is an odd integer}\}$.

S : odd , S_1 : even (before mismatch)
A : odd , A_1 : even (after mismatch)

$X \to a \mid b$

$$\begin{cases} S \to X S_1 X \\ S_1 \to X S X \end{cases}$$

$S \to a A_1 b \mid b A_1 a$ (mismatch)
$S_1 \to a A b \mid b A a$ (mismatch)

$$\begin{cases} A \to X A_1 X \\ A_1 \to X A X \mid c \end{cases}$$

II. (10 pts.) Specify a decision procedure for the following problem:

- Given n dfas, $M_1, M_2, \cdots, M_n$, over the alphabet Σ, do there exist 2 dfas M_k and M_ℓ, $k \neq \ell$, such that $L(M_k) \cup L(M_\ell) = \Sigma^*$?

Given any M_i and M_j, we can test $L(M_i) \cup L(M_j) = \Sigma^*$ as follows:

Method 1:

Design a dfa for the language $\Sigma^* - (L(M_i) \cup L(M_j))$,
construct an nfa for $L(M_i) \cup L(M_j)$
convert it to dfa
swap final and nonfinal states.
check whether there exists a path from the initial state to a final state.

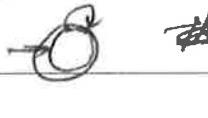
Method 2:

First argue that if $L(M_i) \cup L(M_j) \neq \Sigma^*$ then $\exists$ an input string of length $\leq s_1 s_2$ (where s_1 and s_2 are the number of states of M_i and M_j, resp.)
Feed every string of length $\leq s_1 s_2$ into M_i & M_j. If some string is rejected by both M_i & M_j then $L(M_i) \cup L(M_j) \neq \Sigma^*$.

Overall procedure

For every pair (i, j) check whether $L(M_i) \cup L(M_j) = \Sigma^*$.
If for some pair the check succeeds respond 'yes'
else respond 'no' and halt.

Method 0.

Design a dfa for the lang $L(M_i) \cup L(M_j)$.
construct an nfa for $L(M_i) \cup L(M_j)$
Convert it to dfa.
Minimize the number of states.
If the minimized dfa is 

$L(M_i) \cup L(M_j) = \Sigma^*$.

III. (10 pts.) Design a Turing machine for computing the following function:

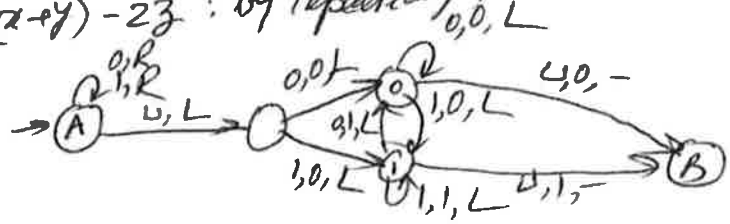
$$f(x,y,z) = \begin{cases} 2x+y-2z & \text{if } 2x+y-2z \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

$x \to 2x$ by shifting left & appending '0' on the right

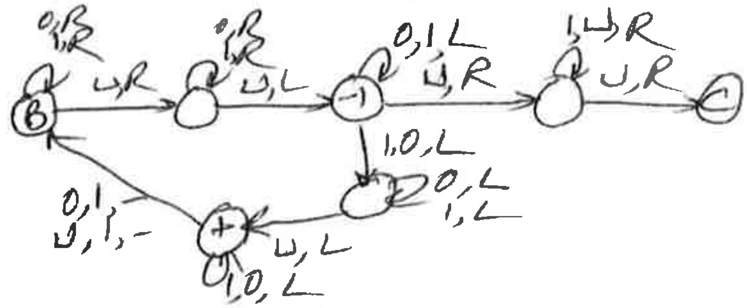
$2x+y$: by repeatedly subtracting 1 from y & adding 1 to $2x$.

$z \to 2z$: by appending 0 on the right.

$(2x+y)-2z$: by repeatedly subtracting 1 from $2x+y$ & $2z$.

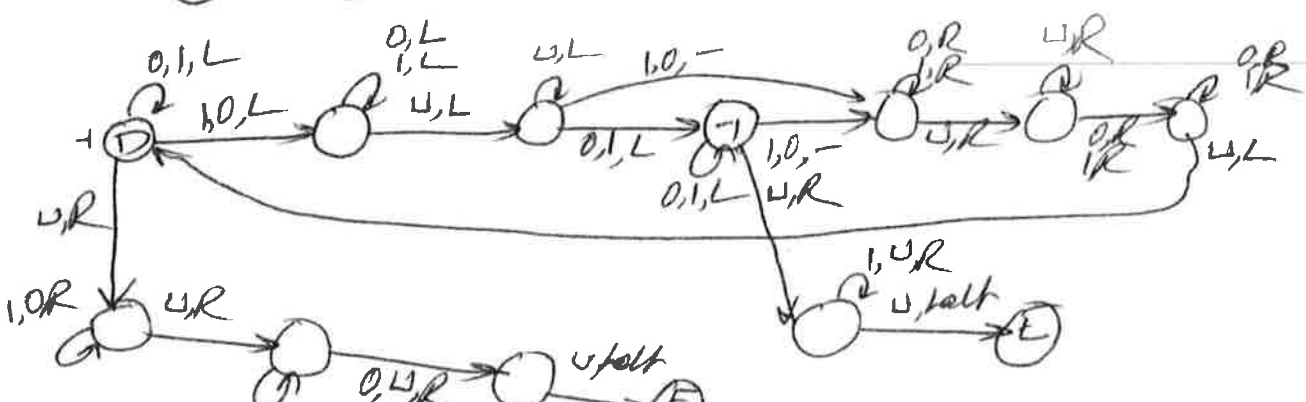


A to B : $x \to 2x$



B to C : $2x+y$

C to D : $z \to 2z$



D to E : $2x+y-2z$

D to F : 0

IV. (10 pts.) Solve one of the following problems.

(a) • Assume that testing whether a given TM halts on exactly one input is undecidable. Prove that testing whether a given TM halts on exactly two inputs is undecidable.

(b) • Recall that a given R-PCP (x_0, y_0), (x_1, y_1), $\cdots$, (x_n, y_n) has a solution if there is a sequence of indices $i_1, i_2, \cdots, i_m$, $m \geq 1$, such that $x_0 x_{i_1} x_{i_2} \cdots x_{i_m} = y_0 y_{i_1} y_{i_2} \cdots y_{i_m}$. Prove that the R-PCP problem is undecidable assuming that the PCP problem is undecidable.

(a) We show 1-input problem $\leq_m$ 2-input problem

Typical inst [M] [M']

Transform M to M' s.t. M halts on exactly one input iff M' halts on exactly 2 inputs.

M': On any input i, if $i \neq 0$, M' simulates M on i and i-1 by ~~dovetailing~~ dovetailing. If M halts on at least one of these, M' halts.

If i = 0, M' simulates M on 0.

<u>correctness</u>

If M halts on 1 input, say on i. Then M' halts on i and i+1

If M doesn't halt on one input:

If M halts on no input: M' halts on no input, i.e. M' doesn't halt on 2 inputs.

If M halts on 2 or more inputs $i_1, \ldots, i_m$ then M' halts on $i_1, i_1+1, i_2, i_2+1, \ldots, i_m, i_m+1$. Even when m = 2 and $i_2 = i_1+1$, then M' halts on 3 inputs: i_1, i_1+1, i_1+2.

Also the transformation from [M] to [M'] is computable.

(b) We show PCP $\leq$ R-PCP

typical inst E $(x,y), (x,y)$ E' $(u_0, v_0), (u_1, v_1), \ldots, (u_m, v_m)$

Transform E to E' st E has a solution iff E' has a solution.

E': $(u_0, v_0) = (c, cc)$

For each $(a_1 \cdots a_n, b_1 b_2 \cdots b_p)$ in E introduce

$(ca_1ca_2 \cdots ca_n, b_1cb_2c \cdots b_pc)$ and

$(ca_1ca_2 \cdots ca_nc, b_1cb_2c \cdots b_pc)$.

Hence E' has 1 + 2n pairs.

(The second pair is used as the last pair in the solution).

Argue the correctness.