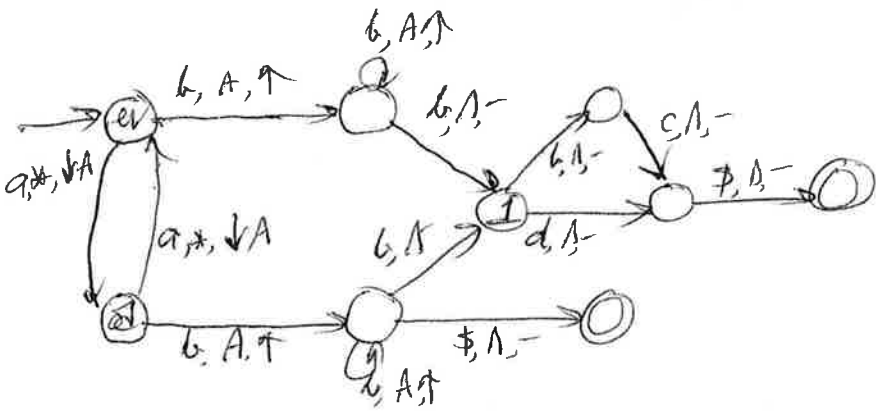
600.271 Automata & Computation Theory
Final Examination
May 12, 2011
In-class, Closed Book
Time: 2 hrs 30 mins.

All the subproblems carry equal weight.

I. Design a deterministic pda for the language:
$\{a^n b^{n+2} c \mid n \geq 1\} \cup \{a^n b^{n+1} d \mid n \geq 1\} \cup \{a^n b^n \mid n \geq 1,\ n \text{ is an odd integer}\}$.

Push a's & keep track of its parity (even or odd). On b's, pop a's.
If pd empty, { no more symbols & odd parity accept.
one additional symbol, followed by d accept
two additional symbols, followed by c accept



II. Design a CFG (i.e. type 2 grammar) for the language:
$\{x \mid x \in \{a,b\}^*,\ |x| \text{ is an even integer},\ x \neq x^R,\ \text{and } \#_a x \text{ is an even integer}\}$.

Generate u & u^R, outside-in. Then generate a mismatch pair. Then generate any pairs. Keep track of the $\#_a$'s.

X : odd $\#_a$'s generated

Y : even $\#_a$'s generated

$S \to aSa \mid bSb \mid aXb \mid bXa$

$X \to aXa \mid bXb \mid aYb \mid bYa$

$Y \to aYa \mid bYb \mid aXb \mid bXa \mid \epsilon$

III. Prove that the following problems are decidable.

1. Given a Post Correspondence Problem (x_1, y_1), $(x_2, y_2), \cdots, (x_n, y_n)$, $x_i, y_i \in \{0,1\}^+$, a dfa M, and a positive integer m, does there exist a string $z \in \{0,1\}^+$ of length no more than m such that $z \in L(M)$, and there exist $i_1, i_2, \cdots, i_k$ satisfying $x_{i_1} x_{i_2} \cdots x_{i_k} = y_{i_1} y_{i_2} \cdots y_{i_k} = z$?

Generate, one after the other, all sequences of length 1, length 2, ..., length m over $\{1,2,\cdots,n\}$. For each string $i_1, \cdots, i_k$ check whether $x_{i_1} x_{i_2} \cdots x_{i_k} = y_{i_1} y_{i_2} \cdots y_{i_k}$, ~~and~~ and $x_{i_1} x_{i_2} \cdots x_{i_k} \in L$ and $|x_{i_1} x_{i_2} \cdots x_{i_k}| \leq m$.
If ~~both~~ all the conditions are satisfied for some sequence, output 'yes' else output 'no' and halt.
This algorithm terminates in a finite number of steps. Also, ~~if~~ each pair contributes a length of at least 1 to z. Hence no sequence with more than m i_j's can result in a string of length $\leq m$. Hence the algorithm is correct.

2. Given a dlba M and positive integers m_1 and m_2, $m_1 < m_2$, do there exist strings x and y such that $m_1 \leq |x|, |y| \leq m_2$, $x \in L(M)$, and $y \notin L(M)$?

Let the dlba have s states & t tape symbols. On any input of length m, if M runs for more than $s m t^m$ steps, then M is in an infinite loop, and hence will not accept the input. Hence we can decide whether M ~~will~~ accepts any chosen input string.

Now generate ~~the~~ all strings of length $m_1, m_1+1, \ldots, m_2$ one after the other. ~~For~~ We test whether the string generated is accepted by M. If at least one of the generated strings is accepted & another generated string is rejected, we output 'yes' & halt; otherwise we output 'no' & halt.

IV. Prove that the following set S is recursively enumerable by providing an appropriate enumeration algorithm:

$S = \{([M_1], [M_2]) \mid M_1 \text{ and } M_2 \text{ are TMs, and } M_1 \text{ halts on blank tape}\}$.

We know that we can enumerate the codings of all TMs: $[M_1], [M_2], \ldots$ (unfortunate that the M_1 & M_2 here are not the same as the ones in the definition of S).

We dovetail the simulations of $M_1, M_2, \ldots$ on blank tape. When we discover that some M_k has halted, we cannot simply output $([M_k], [M_1]), ([M_k], [M_2]), \ldots$. We dovetail this outputting.

Let T be the current step number. It starts at 1.

Let S_1 be the set of TMs that have already halted. Initially it is set to ϕ.

Let Q be the set of processes (TMs being simulated on BT) being simulated. Initially $Q = \phi$.

* At step T, we output for every $M_k \in S_1$, $([M_k], [M_T])$. We place the simulation of M_T on BT in Q. We simulate for an additional step of every process in Q. If some TM M_k halts, we output $([M_k], [M_1]), ([M_k], [M_2]), \ldots, ([M_k], [M_T])$ & then place $[M_k]$ in S_1.

We increment T & repeat *.

V. Prove the undecidability of the following problems.

1. Given $[M]$ and a positive integer n, does TM M accept a string of length n and reject a string of length n? (Hint: Reduce the BTHP to this problem.)

We prove BTHP $\leq_m$ this prob

Typical inst: $[M]$ $([M'], n)$

Goal: Given $[M]$, transform it $([M'], n)$ s.t. TM M halts on BT iff TM M' accepts some string of length n & rejects some string of length n.

We choose $n=1$. TM M' rejects the input 0 (which is of length 1. On input 1 (which is of length 1), M' simulates M on BT. If M halts on BT, then M' accepts the input 1.

Hence M halts on BT $\Rightarrow$ M' accepts 1 & rejects 0.

M doesn't halt on BT $\Rightarrow$ M' rejects 1 & 0.

Hence the construction is correct.

The transformation from $[M]$ to $([M'], n)$ is computable.

Hence the given problem is undecidable.

2. Given deterministic pdas M_1 and M_2, and a dfa M_3, is $L(M_1) \cap L(M_2) \cap L(M_3) \neq \phi$? (Hint: Reduce the PCP to this problem.)

PCP $\leq_m$ this problem

Typical inst. $E = (x_1, y_1), \ldots, (x_n, y_n)$ $[M_1], [M_2], [M_3]$

goal: Given E transform it to $([M_1], [M_2], [M_3])$ s.t. E has a solution iff $L(M_1) \cap L(M_2) \cap L(M_3) \neq \phi$.

M_1 accepts the lang $\{x_{i_1} x_{i_2} \cdots x_{i_k} c_{i_k} \cdots c_2 c_1 \mid k \geq 1, i_1, \ldots, i_k \in \{1, 2, \ldots, n\}\}$

M_2 accepts the lang $\{y_{i_1} y_{i_2} \cdots y_{i_k} c_{i_k} \cdots c_2 c_1 \mid k \geq 1, i_1, \ldots, i_k \in \{1, 2, \ldots, n\}\}$

M_3 accepts the lang $\{0, 1, c_1, c_2, \ldots, c_n\}^*$ (E is over the alphabet $\{0, 1\}$)

If E has a solution, i.e.

$x_{i_1} \cdots x_{i_k} = y_{i_1} \cdots y_{i_k}$ then

$L(M_1) \cap L(M_2) \cap L(M_3)$ contains $x_{i_1} \cdots x_{i_k} c_{i_k} \cdots c_2 c_1$.

If E has no solution then $L(M_1) \cap L(M_2) \cap L(M_3) = \phi$.

Hence the transform is correct.

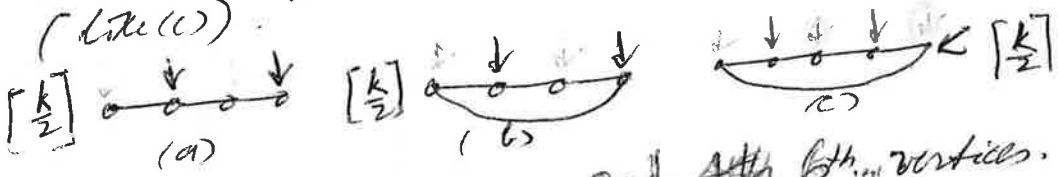
The transformation is computable.

Hence the given problem is undecidable.

VI. Design a P algorithm for one of the following problems. Estimate its speed.

1. Given an undirected graph G of degree ≤ 2, compute the minimum size of its vertex cover. (Hint: Understand the structure of graphs when the degree is ≤ 2). $\leq n$
2. Given a directed graph G, vertices u and v, and a positive integer k, does G contain a path (need not be simple) from u to v of length at least k and at most $2k$?
3. Given a SAT expression E of length n in which at most $2\log_2 n$ variables can appear as both uncomplemented and complemented, is E satisfiable?

1. When degree ≤ 2, G is the union of some simple paths (like (a)), some even length cycles (like (b)), and some odd length cycles (like (c)).



k is the number of vertices

In each case, pick ~~every~~ 2nd, 4th, 6th... vertices. Argue that no smaller VC can exist. This can be implemented to run in $O(n+m)$ steps. (Size $\sum_i \lceil \frac{k_i}{2} \rceil$ k_i is the number of vertices in the kth component)

2. For $i=0,1,\ldots,2k$ we compute $S_i = \{x \mid$ there exists a path of length i from u to $x\}$.

Let $S_0 = \{u\}$.

Having computed S_i, ~~do~~ to compute S_{i+1}:

for every $x \in S_i$ & every edge $\{x,y\}$, place y in S_{i+1}

We then check whether ~~v is in S_k~~, $v \in S_k$, $v \in S_{k+1}$, ... $v \in S_{2k}$. If one of the tests succeeds, we output 'yes' else we output 'no'.

Speed: Computation of each set can be done in $O(m)$ steps. Hence speed $= O(km)$. If $k = O(n^c)$ then the algo is in P. If not, the algo. can be easily modified.

3. For each variable ~~which~~ that occurs only as complemented or uncomplemented we set it ~~to~~ so that the value of the occurrence is 'T'. For the remaining $2\log_2 n$ variables there are $2^{2\log_2 n} = n^2$ possible choices. For each choice we evaluate the expression. If one of the evaluations is 'T' then output 'yes' else output 'no' & halt

Speed $= O(n^2 \cdot n) = O(n^3)$.

VII. Show that the following problem is NP-complete.

Given a connected undirected graph G (with n vertices) and a positive integer $k \geq \frac{n}{2}$, does G contain a vertex cover of size k? (Hint: The standard vertex cover problem is NP-complete.)

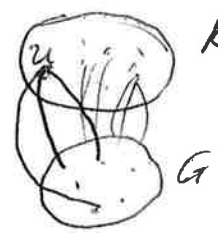
(1) The problem is in NP since it is a special case of VC. Even directly, guess $v_1, \ldots, v_k$ & verify that they cover all edges. If so, print 'yes' else print 'no' & halt. The crudest implementation, in which we check that each edge of G has an end point in $\{v_1, \ldots, v_k\}$, runs in $O(kn)$ steps. Hence the run time is polynomial. ~~If this~~ Correctness is as in ~~the~~ VC.

(2) We show that $VC \leq_p$ this prob

Typical inst: G, k $\quad G', k'$ $\quad k' \geq \frac{n'}{2}$ (n' is the number of vertices of G')

Goal: Transform G, k to G', k' st. G has a ~~a~~ k vertex cover if G' has a k' vertex cover (assume $k \leq n-2$)

G' :



K_n completely connected graph of n vertices

$k' = n+k$.

G' has $2n$ vertices: n vertices forming G & n vertices forming a complete graph. For every new vertex u, there is an edge to every vertex in G.

G has a k vertex cover $\Rightarrow$ G' has an $n+k$ vertex cover (the new n vertices & the k vertices in G)

Any vertex cover in G' must include all the new vertices (If at least 2 are missed, then an edge is missed. If one is missed, then all the vertices ~~are~~ in G must be chosen, otherwise some edge between the missed new edge & a vertex in G will be missed. Then VC of G' is of size $2n-1 > k'$). Thus if G' has a VC of size $n+\alpha$, then G has a VC of size α. Hence the transformation is correct.

The transformation is polynomial time computable.