McGill University COMP531 Winter 2010 **Instructor: Phuong Nguyen**

Assignment 1 solution

Question 1 (Exercise 2.6 (b) in the text) [10pt] HIGH-LEVEL IDEA. We describe how the universal nondeterministic Turing machine U works on input (α, x). The idea is to nondeterministically guess a computation of the machine M_α on input x and accept if and only if the guess is indeed a valid accepting computation. In order to run in time $\mathcal{O}(T(|x|))$ (where $T(|x|)$ is the running time of M_α on x), we need to design the guesses so that it can be verified efficiently. For example, we cannot guess the entire configuration of M_α for every step, because this would require time $T(n)^2$ to verify.

Our guesses will be a sequence of $T(n)$ "description tuples" for the moves of M_α. Suppose M_α has k tapes in total. A description tuple of a move of M_α is a tuple that contains

- the current state,
- the next state,
- k symbols that M_α reads on its k tapes,
- k symbols that M_α will write on the tapes,
- and k directions L or R or S (for left, right, or stay) for the movement of the tape heads.

Although U does not know the value of $T(|x|)$, we can guarantee that it always runs in time $\mathcal{O}(T(|x|)$ by requiring that its guesses are complete computations of M_α (so the last state is always either q_{accept} or q_{reject}).

To verify that our guesses correctly describe an accepting computation of M_α on x, we need to verify that

(1) each description tuple obeys the transition functions of M_α,

(2) the symbols that are read on the tapes of M_α are guessed correctly,

(3) the last description tuple reach the accepting state q_{accept}.

For (1) we simply go though the sequence of guesses once and verify that each tuple satisfies one of the transition functions of M_α. (For this purpose the transition functions of M_α is stored in a separate work tape of U.) We check (2) for each tape of M_α, one by one, using a separate work tape of M_α, by going through the sequence of guesses and following the actions specified by the guesses to reconstruct the content of the work tape of M_α. The consistency of the guess for each symbol that M_α reads can be easily verified during reconstruction. We will need k sweeps through the guesses. Finally (3) is done by simply looking at the last desciption tuple.

SOME DETAILS. The universal nondeterministic Turing machine U has three work tapes. On input (α, x) it decodes α and stores M_α's transition function on the first work tape. The seconde work tape will be used to store M_α's current state during simulation. Then it guesses a sequence of tuples of the form

$$(q, q', s_1, \ldots, s_k, s'_1, \ldots, s'_k, D_1, \ldots, D_k)$$

where $D_i \in \{L, R, S\}$, for $1 \leq i \leq k$, so that in the last tuple q' is either q_{accept} or q_{reject}..

Next it goes through this sequence and checks that for each tuple either

$$\delta_0(q, s_1, \ldots, s_k) = (s'_1, \ldots, s'_k, D_1, \ldots, D_k)$$

or

$$\delta_1(q, s_1, \ldots, s_k) = (s'_1, \ldots, s'_k, D_1, \ldots, D_k)$$

where δ_0, δ_1 are the transition functions of M_α, and that in the last tuple $q' = q_{accept}$.

Finally, for each i, $1 \leq i \leq k$ the machine U uses the last work tape to reconstruct the work tape content of the i-th work tape of M_α as follows. For example, consider $i = 1$. U goes through the sequence of guesses, checks that the current symbol that it reads on the third tape is s_1, then it prints the symbol s'_1 on this tape, and moves the tape head according to D_1. If at any point the symbol it reads is not the same as s_1 it rejects. After reconstructing the content of the first tape, it erases all symbols on its last tape and goes on to reconstruct M_α's second work tape, etc.

If all the above are verified, then U accepts. The running time of U is clearly bounded by $cT(|x|)$ for some constant c that depends on the machine M_α.

Question 2 [10pt] This problem can be solved in the same way that we have used to show (in class) that $\mathbf{P} = \mathbf{NP}$ implies $\mathbf{EXP} = \mathbf{NEXP}$.

Suppose $DTIME(n) = NTIME(n)$. It suffices to show that $NTIME(n^k) = DTIME(n^k)$ for any $k \in \mathbb{N}$. Let L be any language in $NTIME(n^k)$, i.e., there is an NTM M for L that works in time $\mathcal{O}(n^k)$. Let

$$L' = \{x01^{n^k - (n+1)} \ : \ x \in L \wedge n = |x|\}$$

Then M can be modified to obtain an NTM M' that accepts L' in linear time. By the assumption that $NTIME(n) = DTIME(n)$, there is a deterministic TM M'' that accepts L' in linear time. So an $\mathcal{O}(n^k)$ deterministic algorithm for L is as follows. On input x of length n, pad x to obtain to obtain $y = x01^{n^k - (n+1)}$, then run M'' on y and accepts iff M'' accepts.

Question 3 (Exercise 2.5 in the text) [10pt] **a**) To show that PRIME is in *co*-**NP** we need to show that there is a certificate for the fact that a given number n is a composite number. Such a certificate can be taken to be a pair (y, z) of two numbers such that $1 < y, z < n$ and $y \cdot z = n$. The fact that $y \cdot z = n$ is easily verified in time polynomial in the length of n.

b) HIGH-LEVEL IDEA. The certificate for $n \in PRIME$ will contain r that is guaranteed by Lucas' test. Then the fact that $r^{n-1} \equiv 1 \mod n$ can be done by repeated squaring. In order to be able to verify the other condition, i.e., for all prime divisor q of $n-1$, $r^{\frac{n-1}{q}} \not\equiv 1 \mod n$, we will simply supply a list of all prime divisors of $q_1, q_2, \ldots, q_k$ of $n-1$ (probably with repetitions) such that $q_1 \cdot q_2 \cdot \ldots \cdot q_k = n-1$, and verify that for each q_i, $r^{\frac{n-1}{q_i}} \not\equiv 1 \mod n$. Now the fact that q_i are primes also needs to be certified, and we will construct the certificates for the q_i in the same way. This suggests that the certificate for n can be viewed as consisting of at most $\log(n)$ parts:

$$w_1\#w_2\#\ldots\#w_\ell$$

(where $\ell \leq \log(n)$) so that

- w_1 contains the list $(r, q_1, q_2, \ldots, q_k)$ for n as above;

- for $j \geq 1$: for each prime p in w_j, w_{j+1} contains the list $(r', q'_1, q'_2, \ldots, q'_{k'})$ that verifies that p is a prime according to Lucas' test.

There are at most $\log(n)$ such parts because the value of maximum prime in w_{j+1} is less than half of the same value for w_j.

We have to argue that the length of w_j does not grows too fast. This can be seen as follows. Consider the length of w_1. Because $q_1 \cdot q_2 \cdot \ldots \cdot q_k = n - 1$ we have

$$\sum_i \log(q_i) = \log(n-1)$$

so

$$\sum_i \lceil \log(q_i) \rceil \leq \lceil \log(n) \rceil + k$$

Also, $r < n$ and $k \leq \log(n)$. Thus the total length of w_1 is at most $\mathcal{O}(\log(n))$. Similarly we can show that w_2 is of length at most

$$\sum_i \mathcal{O}(\log(q_i)) = \mathcal{O}(\log(n))$$

and generally, w_j has length $\mathcal{O}(\log(n))$. As a result, the total length of the certificate is $\mathcal{O}((\log(n))^2)$.

SOME DETAILS. The NTM M for PRIME works as follows. On input n it guesses a nondeterministic string

$$w_1 \# w_2 \# \ldots \# w_\ell$$

and verifies that this satisfies the conditions above. That is, it verifies that:

- w_1 is a list of the form $(r, q_1, q_2, \ldots, q_k)$ where
  - $r^{\frac{n-1}{q_i}} \not\equiv 1 \mod n$
  - $q_1 \cdot q_2 \cdot \ldots \cdot q_k = n - 1$
  - for each i, $r^{\frac{n-1}{q_i}} \not\equiv 1 \mod n$.
- for $1 \leq j < \ell$, for each prime $p > 3$ appearing in w_j, w_{j+1} contains a list $(r', q'_1, q'_2, \ldots, q'_{k'})$ such that
  - $(r')^{\frac{p-1}{q'_i}} \not\equiv 1 \mod p$
  - $q'_1 \cdot q'_2 \cdot \ldots \cdot q'_{k'} = p - 1$
- $\ell \leq \log(n)$

The running time of M is clearly polynomial in the length of n.

Question 4 (Exercises 2.10 and 2.29 in the text) [10pt] **a**) Suppose L_1 and L_2 are defined by:

$$x \in L_1 \Leftrightarrow \exists y, |y| \leq t_1(|x|) R_1(x, y)$$
$$x \in L_2 \Leftrightarrow \exists y, |y| \leq t_2(|x|) R_2(x, y)$$

for some polynomial t_1, t_2 and polytime relations R_1, R_2. Then $L_1 \cap L_2$ can be defined by

$$x \in L_1 \cap L_2 \Leftrightarrow \exists y, |y| \leq t_1(|x|) + t_2(|x|) + 1,$$
$$y = y_1 \# y_2 \wedge |y_1| \leq t_1(|x|) \wedge |y_2| \leq t_2(|x|) \wedge (R_1(x, y_1) \wedge R_2(x, y_2))$$

Similarly, $L_1 \cup L_2$ can be define by

$$x \in L_1 \cup L_2 \Leftrightarrow \exists y, |y| \leq t_1(|x|) + t_2(|x|) + 1,$$
$$y = y_1 \# y_2 \wedge |y_1| \leq t_1(|x|) \wedge |y_2| \leq t_2(|x|) \wedge (R_1(x, y_1) \vee R_2(x, y_2))$$

These show that both $L_1 \cap L_2$ and $L_1 \cup L_2$ belong to **NP**.

b) Suppose L_1 and L_2 are defined by:

$$x \in L_1 \Leftrightarrow \exists y, |y| \leq t_1(|x|) R_1(x, y)$$
$$x \in L_2 \Leftrightarrow \exists y, |y| \leq t_2(|x|) R_2(x, y)$$

and also

$$x \notin L_1 \Leftrightarrow \exists y, |y| \leq t_1(|x|) R_1'(x, y)$$
$$x \notin L_2 \Leftrightarrow \exists y, |y| \leq t_2(|x|) R_2'(x, y)$$

for some polynomial t_1, t_2 and polytime relations R_1, R_2, R_1', R_2' (here we use without loss of generality the same bound for two definitions of the L_i).

Then $L_1 \oplus L_2$ can be defined as follows:

$$\begin{aligned}
x \in L_1 \oplus L_2 \Leftrightarrow & \quad (x \in L_1 \wedge x \notin L_2) \vee (x \notin L_1 \wedge x \in L_2) \\
\Leftrightarrow & \quad (\exists y, |y| \leq t_1(|x|) R_1(x, y) \wedge \exists y, |y| \leq t_2(|x|) R_2'(x, y)) \vee \\
& \quad (\exists y, |y| \leq t_1(|x|) R_1'(x, y) \wedge \exists y, |y| \leq t_2(|x|) R_2(x, y)) \\
\Leftrightarrow & \quad \exists y, |y| \leq t_1(|x|) + t_2(|x|) + 1, y = y_1 \# y_2 \wedge |y_1| \leq t_1(|x|) \wedge |y_2| \leq t_2(|x|) \wedge \\
& \quad (R_1(x, y) \wedge R_2'(x, y)) \vee (R_1'(x, y) \wedge R_2(x, y))
\end{aligned}$$

This shows that $L_1 \oplus L_2$ is in **NP**. Similar arguments show that $L_1 \oplus L_2$ is in *co*-**NP**. Therefore $L_1 \oplus L_2 \in \mathbf{NP} \cup co\text{-}\mathbf{NP}$ as desired.

Question 5 [10pt] **a**) Suppose that L^* is in **P**, then there is a Turing machine M that accepts L^* in time n^k for some constant k. We construct a polytime Turing machine M' for L as follows. The machine M' on input x of length n will append 01^{n^2} to the end of x and then simulates M on $x01^{n^2}$. M' accepts x if and only if M accepts $x01^{n^2}$.

The running time of M' is $\mathcal{O}(n^2) + n^k$ which is a polynomial in n.

b) We know by the Space Hierarchy Theorem that there is a language L in $DSPACE(n^2)$ but L is not in $DSPACE(n)$. Since L is in $DSPACE(n^2)$, there is a Turing machine M that accepts L in space $\mathcal{O}(n^2)$. Construct M^* for L^* as follows. On input y, M^* reject if y is not of the form $x01^{n^2}$ where $n = |x|$. Now for y of this form, M^* simulates M on input x and accepts if and only if M accepts. Since M accepts L, M^* accepts L^*. The space used by M^* is the maximal of (i) the space used to verify that y is of the right form and (ii) the space used by M. Here the task in (i) can be done in space $|x|$, and the space in (ii) is $\mathcal{O}(|x|^2)$, which is $\mathcal{O}(|y|)$. So M^* works in linear space, i.e., $L^* \in DSPACE(n)$.