

Final Exam

This exam consists of five problems of equal weight. Show your work and justify all your answers. Please do not write things down that are irrelevant. You will be graded not only on the correctness of your answer, but also on the clarity you express it.

Problem	Points	Grade
1	20	
2	20	
3	20	
4	20	
5	20	
Total	100	

I have abided by the Academic Honor Code.

Name (please print): _____

Signature: _____

- Let L_1 and L_2 be r.e. sets. Is L_1L_2 , the concatenation of L_1 and L_2 , also an r.e. set? Defend your answer.

Answer: L_1L_2 is an r.e set.

Proof: Since L_1 and L_2 are r.e., there exist Turing machines M_1 and M_2 such that $L_1 = L(M_1)$ and $L_2 = L(M_2)$. Note that

$$L_1L_2 = \{x \mid x = yz, y \in L_1 \text{ and } z \in L_2\}.$$

Construct a new Turing machine M as follows: On input $x = x_1x_2 \cdots x_n$, where $x_i \in \Sigma$, generate all possible partitions of x as

$$(\epsilon, x), \dots, (x[1 : k], x[k + 1 : n]), \dots, (x, \epsilon),$$

where $1 \leq k < n$ and $x[i : j] = x_i \cdots x_j$.

- Set $t \leftarrow 0$.
- Set $t \leftarrow t + 1$. For each of these partitions (y, z) , simulate M_1 on y for t steps and simulate M_2 on z for t steps. If for some partition both simulations halt then M halts. Otherwise, repeat step (2).

It follows from the construction that $L_1L_2 = L(M)$. Thus, L_1L_2 is r.e.

2. Consider the following language, where M is a DTM and x is an input string:

$$L = \{\langle M, x \rangle \mid M \text{ on input } x \text{ writes a } 0 \text{ on the } (|x| + 1)\text{th cell in its output tape}\}.$$

Show that L is not decidable.

Proof: Recall that the Turing machine halting problem

$$H = \{\langle M, x \rangle \mid M \text{ is a Turing machine and } M \text{ on input } x \text{ halts}\}$$

is not decidable. Construct a reduction f from H to L by

$$f(\langle M, x \rangle) = \langle M', x \rangle,$$

where M' is a Turing machine defined as follows: On input x , M' first simulates M on x , and M' does not write anything on its output tape during the simulation. If M on x halts, then M' writes a 0 on the $(|x| + 1)$ th cell of its output tape and halts.

Because on any input $\langle M, x \rangle$, function f outputs the encoding of a program of M' and x , f is a recursive function. It follows from the construction that $\langle M, x \rangle \in H$ iff M on input x halts iff M' on input x writes a 0 on the $(|x| + 1)$ th cell of its output tape iff $\langle M', x \rangle \in L$ iff $f(\langle M, x \rangle) \in L$. Thus, $H \leq_m L$ via f . Since H is not decidable, neither is L .

3. Assume that A is in PSPACE. Is A^* , the Kleene closure of A , also in PSPACE? Defend your answer.

Answer: A^* is in PSPACE.

Proof: Since A is in PSPACE, A is in DSPACE($p(n)$) for some polynomial p . Thus, there is a DTM M that accepts A in $p(n)$ space. Construct a Turing machine M' such that on any input $x = x_1 \cdots x_n$, where $x_i \in \Sigma$, M' evaluates an array T so that $T[i] = 1$ iff the substring $x_i \cdots x_n \in A^*$. Note that $x_i \cdots x_n \in A^*$ iff there is a j with $i \leq j \leq n$ such that $x_i \cdots x_j \in A$ and $x_{j+1} \cdots x_n \in A^*$. The construction of M' is as follows:

- (1) Initially, set $T[i] \leftarrow 0$ for $i = 1, \dots, n$ and set $T[n + 1] \leftarrow 1$.
- (2) For i from n down to 1, check for each j with $i \leq j \leq n$, whether $T[j + 1] = 1$ and M accepts the substring $x_i \cdots x_j$. If such an j exists, set $T[i] \leftarrow 1$.

If $T[1] = 1$, M' accepts x and halts. Otherwise, M' rejects x and halts.

It follows from the construction that M' accepts A^* and on any input of length n , its space complexity is $O((n + 1) + p(n))$. Since $O((n + 1) + p(n))$ is bounded above by a polynomial, A^* is in PSPACE.

4. Let M_1 be an NTM with $NSPACE_{M_1}(x) \leq s(|x|)$, where $s(n) \geq \log n$. It follows from the proof of Savitch's Theorem that there is a DTM M_2 with $DSPACE_{M_2}(x) \leq$

$O(s^2(|x|))$ such that $L(M_1) = L(M_2)$. What is the time complexity of M_2 ? Defend your answer.

Answer: In the proof of Savitch's theorem, M_2 uses a stack to execute the predicate $\text{reachable}(\alpha, \beta, 2^{c \cdot s(n)})$ for some fixed positive integer c , where α is the initial configuration of M_1 on input x of length n , β is a halting configuration of M_1 , and $2^{c \cdot s(n)}$ is the total number of configurations. For convenience, let $c(n) = 2^{c \cdot s(n)}$. The predicate $\text{reachable}(\alpha_1, \alpha_2, 2^i)$ satisfies the following recurrence relation:

For $i > 0$, $\text{reachable}(\alpha_1, \alpha_2, 2^i) = 1$ iff there exists a configuration α_3 such that

$$\text{reachable}(\alpha_1, \alpha_3, 2^{i-1}) = 1 \text{ and } \text{reachable}(\alpha_3, \alpha_2, 2^{i-1}) = 1,$$

and $\text{reachable}(\alpha_1, \alpha_2, 2^0) = 1$ iff $\alpha_1 \vdash \alpha_2$ or $\alpha_1 = \alpha_2$. Let $T(i)$ denote the number of steps for evaluating $\text{reachable}(\alpha_1, \alpha_2, 2^i)$. Let $t(n)$ denote the number of steps for generating all possible configurations. Then we have

$T(0) = O(s(n))$, $T(1) = 2(c(n)T(0) + t(n))$, $T(2) = 2(c(n)T(1) + t(n))$, and in general, $T(i) = 2(c(n)T(i-1) + t(n))$. Thus,

$$\begin{aligned} T(i) &= 2(c(n)[2(c(n)T(i-2) + t(n))] + t(n)) \\ &= 2^2 c^2(n)T(i-2) + 2c(n)t(n) + t(n) \\ &= \dots \\ &= 2^i c^i(n)T(0) + (2^{i-1} c^{i-1}(n) + \dots + 1)t(n) \\ &= 2^i c^i(n)T(0) + (2^i c^i(n) - 1)t(n) \\ &= 2^i c^i(n)[T(0) + t(n)] - t(n). \end{aligned}$$

Thus, evaluating $\text{reachable}(\alpha, \beta, 2^{c \cdot s(n)})$ will take at most

$$\begin{aligned} &2^{c \cdot s(n)} c^{c \cdot s(n)}(n)[T(0) + t(n)] - t(n) \\ &< O(2^{c \cdot s(n)} (2^{c \cdot s(n)})^{c \cdot s(n)} [s(n) + 2^{c \cdot s(n)}]) \\ &= 2^{O(s^2(n))} \end{aligned}$$

steps. Since there are at most $c(n)$ halting configurations, the entire computation of M_2 on input of length n will take at most $O(c(n)2^{O(s^2(n))}) = 2^{O(2^2(n))}$ steps.

5. Instances of k SAT are k -CNF formulas. That is, each clause C in a k -CNF formula consists of exactly k different literals. Moreover, if C contains a literal ℓ , then its complement $\neg\ell$ does not occur in C . We want to determine whether a given k SAT instance is satisfiable. We have shown that SAT is NP-complete. Show that 4SAT is NP-complete.

Proof. We will reduce SAT to 4SAT using the same technique presented in my handout showing that 3SAT is NP-complete. Listed below is the detail.

Let F be any CNF formula. We introduce four new variables y_1, y_2, y_3, y_4 and fifteen new clauses of all 4-literal clause of these four variables except $\{\neg y_1, \neg y_2, \neg y_3, \neg y_4\}$. It is easy to see that the only truth-value assignment to satisfy all these fifteen clauses is $y_1 = y_2 = y_3 = y_4 = 1$.

Let C be a clause of F . Then $C = \{z_1, z_2, \dots, z_k\}$ for some $k \geq 1$, where z_i is a literal. We will replace C by 4-clause(s) as follows.

Case 1: $k = 1$. Replace C by $\{z_1, \neg y_1, \neg y_2, \neg y_3\}$.

Case 2: $k = 2$. Replace C by $\{z_1, z_2, \neg y_1, \neg y_2\}$.

Case 3: $k = 3$. Replace C by $\{z_1, z_2, z_3, \neg y_1\}$.

Case 4: $k = 4$. Do nothing.

Case 5: $k = 5$. Introduce one new variable $u_1 = y_C$, and replace C by the following 4-clauses:

$$\{z_1, z_2, z_3, u_1\}, \{z_4, z_5, \neg u_1, \neg y_1\}, \{\neg z_4, u_1, \neg y_1, \neg y_2\}, \{\neg z_5, u_1, \neg y_1, \neg y_2\}.$$

Note that the last three 4-clauses imply that $z_4 + z_5$ is equivalent to u_1 , and so $z_1 + z_2 + z_3 + u_1$ is equivalent to $z_1 + z_2 + z_3 + z_4 + z_5$.

Case 5: $k > 5$. Inductively apply Case 5 until only clauses of 4 literals left. For example, when $k = 6$, first replace C with the following clauses:

$$\{z_1, z_2, z_3, z_4, u_1\}, \{z_5, z_6, \neg u_1, \neg y_1\}, \{\neg z_5, u_1, \neg y_1, \neg y_2\}, \{\neg z_6, u_1, \neg y_1, \neg y_2\}.$$

Then the last three 4-clauses imply that $z_5 + z_6$ is equivalent to u_1 . Thus, $z_1 + z_2 + z_3 + z_4 + z_5 + z_6$ is equivalent to $z_1 + z_2 + z_3 + z_4 + u_1$. Now replace the first 5-clause $\{z_1, z_2, z_3, z_4, u_1\}$ by the following 4-clauses, where u_2 is a new variable:

$$\{z_1, z_2, z_3, u_2\}, \{z_4, u_1, \neg u_2, \neg y_1\}, \{\neg z_4, u_2, \neg y_1, \neg y_2\}, \{\neg u_1, u_2, \neg y_1, \neg y_2\}.$$

As shown before that the last three 4-clauses imply that u_2 is equivalent to $z_4 + u_1$, and so $z_1 + z_2 + z_3 + z_4 + u_1$ is equivalent to $z_1 + z_2 + z_3 + u_2$.

Let G be the product of all new clauses. Then F is satisfiable if and only if G is satisfiable, and the above transformation from C to a set of 4-clauses can be carried out in time polynomial in k . Thus, $SAT \leq_m^p 4SAT$. This completes the proof.