Toda's Theorem, following Lance Fortnow

Advanced Complexity Test

This test is fairly long. Go fast: say what the essential ideas are; but do the computations right, and do not miss any subtle issue!

To give an idea of the relative difficulties, I'm giving the number of lines of my solution to each question.

1 Relativizations

Recall that an oracle machine is a multi-tape machine with a specific query tape, three extra control states Q, YES and NO. Let A be a language. The semantics of the machine with oracle A is as usual, except that when the machine reaches state Q, it then proceeds to state YES if the contents of the query tape is in A, and to NO otherwise. The *relativized* classes $\mathbf{P}^A$, $\mathbf{NP}^A$, $\mathbf{BPP}^A$, etc., are obtained from their classical counterpart by changing the underlying Turing machine model to the corresponding oracle machine, with oracle A.

For classes $\mathcal{C}$, $\mathcal{C}'$ of languages, and assuming the notation $\mathcal{C}^A$ makes sense, the class $\mathcal{C}^{\mathcal{C}'}$ is $\bigcup_{L' \in \mathcal{C}'} \mathcal{C}^{L'}$. I.e., a language L is in $\mathcal{C}^{\mathcal{C}'}$ iff there is a language $L' \in \mathcal{C}'$ such that $L \in \mathcal{C}^{L'}$.

1. *(2 lines, 1.4%)* Assume $\mathcal{C}'$ has a complete language L' (under polynomial time reductions). Show that, for any $\mathcal{C}$ among $\mathbf{P}$, $\mathbf{NP}$, $\mathbf{BPP}$, $\mathbf{RP}$, $\mathcal{C}^{\mathcal{C}'} = \mathcal{C}^{L'}$.

2 The Zachos Lemma

1. *(7 lines, 4.8%)* Show that $\mathbf{NP}^{\mathbf{BPP}} = \mathbf{MA}$.

2. *(13 lines, 9%)* Using this, show that if $\mathbf{NP} \subseteq \mathbf{BPP}$, then $\mathbf{PH} \subseteq \mathbf{BPP}$. This is the *Zachos Lemma*.

3 The Class $\oplus$P (Parity-P)

Let us recall the $\mathbf{BP}\cdot$ operator from the lectures: for any complexity class $\mathcal{C}$, $\mathbf{BP} \cdot \mathcal{C}$ is the class of languages L such that there is a randomized polynomial time Turing machine $\mathcal{A}'$ and a language $D' \in \mathcal{C}$ such that, on input x (of size n):

- If $x \in L$, then $Pr_r[\mathcal{A}'(x,r) \in D'] \geq 2/3$;
- If $x \notin L$, then $Pr_r[\mathcal{A}'(x,r) \in D'] \leq 1/3$.

where probabilities are taken on random strings r of size $q(n)$, for some polynomial q in n.

We define another operator $\oplus\cdot$ ("parity") as follows: $L \in \oplus \cdot \mathcal{C}$ iff there is a language L' in $\mathcal{C}$, and a polynomial $p(n)$, such that:

- $x \in L$ iff the number of strings y of size $p(n)$ such that $(x,y) \in L'$ is *odd*.

I.e., $\oplus \cdot \mathbf{P}$ is the class of language decidable on a non-deterministic Turing machine by letting all the branches of computation have the same length $p(n)$ (this is important), and then accepting iff the number of accepting branches is odd.

1. *(11 lines, 7.6%)* Define the relativization $(\oplus \cdot \mathbf{P})^A$ as $\oplus \cdot (\mathbf{P}^A)$. Show that $(\oplus \cdot \mathbf{P})^{\oplus \cdot \mathbf{P}} = \oplus \cdot \mathbf{P}$. (Hint: count modulo 2.) This is the *Papadimitriou-Zachos Lemma*.

2. *(21 lines, 14.5%)* Let $\Sigma = \mathbb{Z}/2\mathbb{Z}$ in this Section. Recall that a linear hash function $h : \Sigma^m \to \Sigma^{m'}$ is a linear map from $\mathbb{Z}/2\mathbb{Z}^m$ to $\mathbb{Z}/2\mathbb{Z}^{m'}$.

   Let F be a propositional formula, built on propositional variables $x_1, \ldots, x_m$, say. Let X be the set of environments (mappings from the propositional variables $x_1, \ldots, x_m$ to truth-values) ρ that satisfy F (in notation, $\rho \models F$). Let m' be a natural number such that $2^{m'-2} \leq |X| \leq 2^{m'-1}$, where $|X|$ is the cardinality of X. Identify each environment ρ with the obvious vector in Σ^m. Show that:

   $$Pr_{h,b}[\exists! \rho \in \Sigma^m \cdot \rho \models F \text{ and } h(\rho) = b] \geq \frac{1}{8}$$

   where h is drawn at random uniformly among all linear hash functions from Σ^m to $\Sigma^{m'}$, and b is drawn at random uniformly, and independently, in $\Sigma^{m'}$. We write $\exists!$ for "there exists a unique". (Hint: given a fixed ρ, find a lower bound for the probability of the event $C_\rho(h,b)$, defined as holding whenever $h(\rho) = b$ but $h(\rho') \neq b$ for every $\rho' \in X$ such that $\rho' \neq \rho$.)

3. *(17 lines, 11.7%)* Deduce a probabilistic polynomial time algorithm that takes as input a propositional formula F, and outputs propositional formulae $F_1, \ldots, F_k$ (for some polynomial k) such that:

   - If F is unsatisfiable, then so are $F_1, \ldots, F_k$;
   - If F is satisfiable, then with high probability, some F_i has a *unique* satisfying assignment (formally, $Pr[F_1$ is uniquely satisfiable or $\ldots$ or F_k is uniquely satisfiable$] \geq 2/3$).

4. *(12 lines, 8.3%)* Conclude that $\mathbf{NP} \subseteq \mathbf{BPP}^{\oplus \cdot \mathbf{P}}$.

4 Relativizing Fagin's Theorem, and SAT

All of our results in the lectures used multi-tape Turing machines. It will be easier to reason with single-tape Turing machines here. Recall that we can encode all multi-tape Turing machines as single-tape Turing machines, encoding all tapes $B_1, \ldots, B_m$ (each with some symbol representing the control state at the position of the head) as the concatenation $B_1\#\ldots\#B_m$ where $\#$ is some fresh separator symbol. Let Q be the alphabet of control states. We shall assume the query tape is encoded as B_1, and that the head is at the beginning of B_1 when the machine enters state Q.

1. *(19 lines,* 13.1*%)* Remember that (Fagin's Theorem) the languages in **NP** are exactly the languages definable by existential second-order formulae. See the Appendix for the precise statement, and for a proof.

   We shall need the following second-order construction. Given a formula F, and first-order variables (occurring or not) $x_1, \ldots, x_k$ in F, a *predicate expression* (not to be confused with a predicate variable P) is an expression of the form $\lambda x_1, \ldots, x_k \cdot F$. The latter is of *arity* k. The notation may be clearer if you write $F(x_1, \ldots, x_k)$ instead of F: $\lambda x_1, \ldots, x_k \cdot F(x_1, \ldots, x_k)$ denotes the predicate of k values that is true exactly when the formula F is true of these values.

   We shall consider a *third-order* predicate symbol $A^?$. A *relativized* second-order formula F_2 is a second-order formula with an additional formula former construct $A^?(E_1, \ldots, E_m)$ where $E_1, \ldots, E_m$ are predicate expressions: i.e., the third-order predicate symbols are applied to (second-order) predicate expressions.

   The semantics is now defined by giving oneself the denotation of $A^?$ as a fixed m-ary relation among relations (denoting $E_1, \ldots, E_m$) of the relevant arities. (Formally, $A^?$ comes with a fixed *signature* $(k_1, \ldots, k_m)$, and we require each E_i to be of arity k_i.)

   This is probably hard to understand. Let us illustrate this in the case that will interest us. This uses notions from the proof of Fagin's Theorem we have already seen, see the Appendix. If A is a language, and the tape alphabet is $a_1, \ldots, a_p$, so that $T_{a_i}(\vec{t}, \vec{x})$ encoded the positions $\vec{x}$ where one could find letter a_i at time t, the predicate expressions $\lambda\vec{x} \cdot T_{a_i}(\vec{t}, \vec{x})$, $1 \leq i \leq p$, altogether encode the contents of the tape at time $\vec{t}$. So one can use $A^?(\lambda\vec{x} \cdot T_{a_1}(\vec{t}, \vec{x}), \ldots, \lambda\vec{x} \cdot T_{a_p}(\vec{t}, \vec{x}))$ as a formula testing whether the contents of the tape is in language $QA\#\Sigma^*$ (i.e., control state, then a word in A, separator, and all the other tapes).

   Show that the languages in $\mathbf{NP}^A$ are exactly those definable by relativized existential second-order formulae.

2. *(29 lines,* 20*%)* From this deduce that FORM-SAT^A is $\mathbf{NP}^A$-complete under polynomial-time reductions, where FORM-SAT^A is defined as follows. This is a variant on FORM-SAT, the problem of deciding whether an input propositional formula is satisfiable. (SAT further restricts the formula to be in clausal form, but FORM-SAT will be easier for our purposes.) FORM-SAT^A is as FORM-SAT, except we now also accept atomic

formulae of the form $A^!(x_1, \ldots, x_q)$ where $x_1, \ldots, x_q$ are propositional variables. We say that $A^!(x_1, \ldots, x_q)$ is true iff the bit-string $x_1, \ldots, x_q$ is an encoding (in a sense that you shall make precise) of a tape contents in the language $QA\#\Sigma^*$.

3. *(7 lines, 4.8%)* Show that the result $\mathbf{NP} \subseteq \mathbf{BPP}^{\oplus\cdot\mathbf{P}}$ from Part 3 relativizes (see the precise statement below). To this end, we define $\mathbf{BPP}^{\oplus\cdot\mathbf{P}}$ relativized to oracle A so that both the **BPP** and the $\oplus \cdot \mathbf{P}$ machines have access to the oracle. Since the **BPP** machine can already use a call to a $(\oplus \cdot \mathbf{P})^A$ oracle to obtain answers to queries to A, we define $(\mathbf{BPP}^{\oplus\cdot\mathbf{P}})^A$ as $\mathbf{BPP}^{(\oplus\cdot\mathbf{P})^A} = \mathbf{BPP}^{\oplus\cdot\mathbf{P}^A}$.

   Then show that $\mathbf{NP}^A \subseteq (\mathbf{BPP}^{\oplus\cdot\mathbf{P}})^A$.

4. *(3 lines, 2%)* Show that, if $\mathbf{NP}^A \subseteq \mathbf{BPP}^A$, then $\mathbf{PH}^A \subseteq \mathbf{BPP}^A$. We define the relativized polynomial hierarchy $\mathbf{PH}^A$ as the union of Σ_i^A, $i \geq 1$, where $\Sigma_1^A = \mathbf{NP}^A$, $\Sigma_{i+1}^A = \mathbf{NP}^{\Sigma_i^A}$. (I am not expecting a full, detailed proof, but at least a good roadmap.)

5. *(4 lines, 2.8%)* Deduce the first of Toda's theorems: $\mathbf{PH} \subseteq \mathbf{BPP}^{\oplus\cdot\mathbf{P}}$.

5 Toda's Theorem

The second of Toda's Theorems is $\mathbf{PP}^{\oplus\cdot\mathbf{P}} \subseteq \mathbf{P}^{\oplus\cdot\mathbf{P}}$. This would fill in itself another advanced complexity test. Since $\mathbf{BPP}^A \subseteq \mathbf{PP}^A$ (which is easy), and $\mathbf{P}^{\oplus\cdot\mathbf{P}} \subseteq \mathbf{PSPACE}$ (easy as well) we conclude:

Toda's Theorem: $\mathbf{PH} \subseteq \mathbf{P}^{\oplus\cdot\mathbf{P}} \subseteq \mathbf{PSPACE}$

In particular, the power of counting modulo 2 is very high: somewhere between **PH** and **PSPACE**.

(This part does not contain any question.)

A Fagin's Theorem

We have seen the following version of Fagin's Theorem in the homework assignment:

> The languages in **NP** are exactly those languages L such that there is an existential second-order formula F_2, with no free first-order variable, and only $=$, P and X_a, $a \in \Sigma$, as free predicate variables, such that for every input x of size n, x is in L iff $I_n[(X_a \mapsto D_a)_{a\in\Sigma}, P \mapsto D_<] \models F_2$. Here D_a is the set of positions in x where one finds the letter a (i.e., X_a, $a \in \Sigma$, encode the input x), and $I_n[(X_a \mapsto D_a)_{a\in\Sigma}, P \mapsto D_<]$ is the unique equational structure in standard representation with domain of cardinality n that maps X_a to D_a for each $a \in \Sigma$ and P to the strict ordering $1 < 2 < \ldots < n$.

In one direction, we show that evaluating any existential second-order formula over a model can be done in non-deterministic polynomial time in the size of the model (the formula is fixed).

Conversely, if $\mathcal{M}$ is a non-deterministic polynomial time machine deciding L, we build F_2 as follows. Assume that $\mathcal{M}$ takes time and space bounded by n^k where n is the size of the input. We create the following predicate variables:

- S_q, for each control state q; $S_q(\vec{t})$ should hold exactly when $\mathcal{M}$ is at control state q at time (denoted by) $\vec{t}$;
- T_a, for each tape letter $a \in \Sigma$; $T_a(\vec{t}, \vec{x})$ should hold exactly when the symbol at position (denoted by) $\vec{x}$ at time (denoted by) $\vec{t}$ is a;
- H: $H(\vec{t}, \vec{x})$ should hold exactly when the head is at position $\vec{x}$ at time $\vec{t}$.

F_2 is the existential quantification over all predicate variables S_q, T_a, and H of the first-order formula φ defined as the universal closure (i.e., add $\forall x$ for every free first-order variable in front) of $Z(\vec{0}) \wedge L(\ell) \Rightarrow$ CONSISTENCY $\wedge$ START $\wedge$ COMPUTE $\wedge$ END, where the latter are defined below.

$\vec{0}$ is a k-tuple of first-order variables, and $Z(\vec{0})$ is the first-order formula $\forall \vec{x} \cdot \neg P(\vec{x}, \vec{0})$, stating that $\vec{0}$ encodes time 0. $L(\ell)$ is the first-order formula $\forall x \cdot \neg P(\ell, x)$ stating that variable ℓ should be mapped to n.

- CONSISTENCY states that at each time, the head is always in at most one position, the machine is in at most one control state, and the contents of the tape is unique. This is the conjunction of:
  - $\neg(H(\vec{t}, \vec{x}) \wedge H(\vec{t}, \vec{x}') \wedge \vec{x} \neq \vec{x}')$ (which we prefer to the more natural $H(\vec{t}, \vec{x}) \wedge H(\vec{t}, \vec{x}') \Rightarrow \vec{x} = \vec{x}'$, as the transition to II.6 will be more natural this way), where $\vec{x} \neq \vec{x}'$ is the disjunction of all $x_i \neq x_i'$, $1 \leq i \leq k$;
  - for each pair of distinct control states q, q', $\neg(S_q(\vec{t}) \wedge S_{q'}(\vec{t}))$;
  - for each pair of distinct letters a, b, $\neg(T_a(\vec{t}, \vec{x}) \wedge T_b(\vec{t}, \vec{x}))$.

- START states what the tape contains initially. This is the conjunction of:
  - $S_{q_0}(\vec{0})$ (initially, $\mathcal{M}$ is in the initial state q_0),
  - $H(\vec{0},\vec{0})$ (initially, the head is at position 0),
  - writing $\vec{0}[x]$ for the vector of variables $\vec{0}$, except the last one is replaced by variable x, the conjunction over all letters $a \in \Sigma$ of $X_a(\vec{0}[x]) \Rightarrow T_a(\vec{0}[x])$ (the first n letters on the tape at time 0 are given by the input);
  - $\vec{0}[\ell] <^k \vec{x} \Rightarrow T_{\sqcup}(\vec{x})$, where $\sqcup$ is the blank (all other letters on the tape are blank).
- COMPUTE is more complex, but states the expected transition relation. It is the conjunction of:
  - What part of the tape does not change: the conjunction over $a \in \Sigma$ of $T_a(\vec{t},\vec{x}) \wedge H(\vec{t},\vec{y}) \wedge \vec{x} \neq \vec{y} \wedge \vec{t'} = \vec{t}+1 \Rightarrow T_a(\vec{t'},\vec{x})$; the formula $\vec{t'} = \vec{t}+1$ was defined in II.3.
  - What changes: the conjunction over all control states q and all letters $a \in \Sigma$ of $\mathrm{PRE}(q,a) \Rightarrow \bigvee_{q',b,dir} \mathrm{POST}(q',b,dir)$, where (q',b,dir) ranges over the triples of all possibles next control state (q'), letter to be written (b) and direction (dir, being 0, +1 or −1).
    We let $\mathrm{PRE}(q,a) = S_q(\vec{t}) \wedge H(\vec{t},\vec{x}) \wedge T_a(\vec{t},\vec{x}) \wedge \vec{t'} = \vec{t}+1$.
    And we let:

$$\begin{aligned}
\mathrm{POST}(q',b,dir) &= S'_{q'}(\vec{t'}) \wedge T_b(\vec{t'},\vec{x}) \wedge H(\vec{t'},\vec{x}) \quad \text{if } dir = 0\\
\mathrm{POST}(q',b,dir) &= S'_{q'}(\vec{t'}) \wedge T'_b(\vec{t'},\vec{x}) \wedge (\vec{x'} = \vec{x}+1 \Rightarrow H(\vec{t'},\vec{x'})) \quad \text{if } dir = +1\\
\mathrm{POST}(q',b,dir) &= S'_{q'}(\vec{t'}) \wedge T'_b(\vec{t'},\vec{x}) \wedge (\vec{x} = \vec{x'}+1 \Rightarrow H(\vec{t'},\vec{x'})) \quad \text{if } dir = -1
\end{aligned}$$

- END is the conjunction over all non-final states q of $t_1 = \ell \wedge \ldots \wedge t_k = \ell \Rightarrow \neg S_q(\vec{t})$ (at the last time instant $n^k - 1$, the control state is not non-final). (We may also take the disjunction over all final states q of $t_1 = \ell \wedge \ldots \wedge t_k = \ell \Rightarrow S_q(\vec{t})$.)