Results that Relativize
Exposition by William Gasarch

1 Oracle Turing Machines

Recall that Turing Machines can computer LOTS of stuff and are often denoted M. We want to define a Turing Machine that can also ask questions to a set A and get back the answers instantly. We will not define this formally (which would be easy) but note a few things in our informal definition

Def 1.1 An *Oracle Turing Machine* is a device $M^{()}$ which can do what a Turing machine does but also has an extra tape (an oracle tape) which is write-only and on which it can write questions. Writing a question will take time that is the length of the question. How can these devises be used? We need not just an input x but also an oracle A which is a set of strings. It makes no sense to say "What is $M^{()}(x)$?". But we can say, for a string x and a set A, what is $M^A(x)$. As with ordinary Turing Machines it may accept, reject, take blah-blah steps, or not halt at all. Note that an Oracle TM is defined ind. of the oracle you are going to use.

Def 1.2 Let $T(n)$ be a computable function (think of it as increasing). Let $A \subseteq \Sigma^*$. X is in $DTIME^A(T(n))$ if there is a OTM $M^{()}$ such that M^A decides X and also, for all x, $M^A(x)$ halts in time $\leq T(|x|)$.

Def 1.3 A is in P^A if there is a polynomial $p(n)$ such that A is in $DTIME^A(p(n))$.

2 Time and Space Classes

Def 2.1 Let $T(n)$ be a computable function (think of it as increasing). X is in $DTIME^A(T(n))$ if there is a MULTITAPE OTM $M^{()}$ such that M^A decides X and also, for all x, $M^A(x)$ halts in time $\leq T(|x|)$. *Convention:* By $DTIME^A(T(n))$ we really mean $DTIME(O(T(n))$. They are actually equivalent by having your OTM just take bigger steps.

Note that this is unfortunately machine dependent. It is possible that if we allow 2-tapes instead of one it would change how much you can do. We won't have to deal with this much since we will usually define classes in terms of multi-tape machines, and we will allow some slack on the time bound, like: $DTIME^A(n^{O(1)})$.

It is known that, for all A, a Multitape $DTIME^A(T(n))$ machine can be simulated by (1) a 1-tape $DTIME^A(T(n)^2)$OTM, and also0 (2) a 2-tape $DTIME^A(T(n)\log T(n))$OTM.

Def 2.2 Let $S(n)$ be a computable function (think of it as increasing). X is in $DSPACE^A(S(n))$ if there is a OTM $M^()$ such that M^A decides X and also, for all x, $M^A(x)$ only uses space $S(|x|)$. *Convention:* By $DSPACE^A(S(n))$ we really mean $DSPACE^A(O(S(n))$. They are actually equivalent by having your OTM just take bigger steps. *Convention:* When dealing with space classes we will have an input tape which is read-only and a separate worktape. When dealing with space-bounded OTMs computing functions we will also have a write-only output tape.

It is known that, for any A, a Multitape $DSPACE^A(S(n))$ machine can be simulated by a 1-tape $DSPACE^A(S(n))$OTM.

Def 2.3 Let $T(n)$ be a computable function (think of it as increasing). X is in $NTIME^A(T(n))$ if there is a Nondet OTM $M^($ such that M^A decides X and also, for all x, $M^A(x)$, on any path, halts in time $\leq T(|x|)$. *Convention:* By $NTIME^A(T(n))$ we really mean $NTIME^A(O(T(n))$. They are actually equivalent by having your OTM just take bigger steps.

Def 2.4 Let $S(n)$ be a computable function (think of it as increasing). X is in $NSPACE^A(S(n))$ if there is a Nondet OTM $M^()$ such that M^A decides X and also, for all x, $M^A(x)$, on any path, only uses space $\leq S(|x|)$. *Convention:* By $NSPACE^A(S(n))$ we really mean $NSPACE^A(O(S(n))$. They are actually equivalent by having your OTM just take bigger steps.

Def 2.5 For all of the definitions below, 1-tape and multitape are equivalent. This is important in the proof of Cooks theorem and later in the proof that a particular lang is EXPSPACE complete and hence not in P.

1. $P^A = DTIME^A(n^{O(1)})$.
2. $NP^A = NTIME^A(n^{O(1)})$.
3. $EXP^A = DTIME^A(2^{n^{O(1)}})$.
4. $NEXP^A = NTIME^A(2^{n^{O(1)}})$.
5. $L^A = DSPACE^A(O(\log n))$.
6. $NL^A = NSPACE^A(O(\log n))$.
7. $PSPACE^A = DSPACE^A(n^{O(1)})$.
8. $EXPSPACE^A = DSPACE^A(2^{n^{O(1)}})$.
9. $NEXPSPACE^A = NSPACE^A(2^{n^{O(1)}})$.

- The definition of NP^A is equivalent to the following one: $X \in NP^A$ if there exists $B \in P^A$ such that

$$X = \{x \mid (\exists^p y)[(x, y) \in B]\}.$$

- The definition of Relativized space (e.g., L^A) requires some care because of the question of— do you count the oracle tape or not? We will come back to this question later.

3 Easy Relations Between Classes

The following theorem is trivial.

Theorem 3.1 *Let $T(n)$ and $S(n)$ be computable functions (think of as increasing). Let A be any set.*

1. $DTIME^A(T(n)) \subseteq NTIME^A(T(n))$.
2. $DSPACE^A(S(n)) \subseteq NSPACE^A(S(n))$.
3. $DTIME^A(T(n)) \subseteq DSPACE^A(T(n))$.
4. $NTIME^A(T(n)) \subseteq NSPACE^A(T(n))$.

The following theorem is easy but not trivial.

Theorem 3.2 *Let $T(n)$ and $S(n)$ be computable functions (think of as increasing). Let A be any set.*

1. $NTIME^A(T(n)) \subseteq DTIME^A(2^{O(T(n))})$. *(Just simulate ALL possible paths.)*
2. $NTIME^A(T(n)) \subseteq DSPACE^A(O(T(n)))$. *(Just simulate ALL possible paths- keep a counter for which path you are simulating.)*

4 Sophisticated Relations Between Classes

The following theorem has two parts. They are on your HW.

Theorem 4.1 *Let $T(n)$ and $S(n)$ be computable functions (think of as increasing).*

1. $NSPACE^A(S(n)) \subseteq DSPACE^A(O(S(n)^2))$
2. $NSPACE^A(O(S(n)))$ *is closed under complementation.*

Note 4.2 The above theorem requires a modification of the def of $NSPACE^A(S(n))$. We omit discussion of this since but note that it was part of Dr. Gasarch's PhD thesis.

What do we know about NL ? Using the above we get

$$NSPACE^A(\log n) \subseteq DSPACE^A((\log n)^2)) \subseteq DTIME^A(2^{(\log n)^2}.$$

Can we do better? YES! (MAYBE- the theorem below needs some carefully reworking of the def of NL^A.)

Theorem 4.3 $NL^A \subseteq P^A$.

Proof: Let $X \in NL^A$ via OTM M^A. Given x we want to determine if SOME path in $M^A(x)$ goes to an accept state.

We will assume M^A has 1 worktape. The modification for many tapes are easy.

A *configuration* (henceforth config) is a snapshot of the WORKTAPE ONLY. We also include the state (by convention the square we are looking at will be to the states left). Given a config and the input tape, one can determine which configurations you might goto in one step.

Write down all of possible config. There are only $2^{O(\log n)}$ of them which is some poly, say n^c. Consider them to be nodes of a graph. We draw a directed graph from u to v if from u you CAN go to v in one step (note that this depends on both u and x).

$x \in X$ iff there is a path from the start config to an accept config in the graph. This can be determined in time poly in the size of the graph which is poly in n^c so poly. ∎

5 Time and Space Hierarchy Theorems

Important Note: Imagine doing the following: Take a list of OTMs $M_1^{()}, M_2^{()}, \ldots$ and then bound $M_i^{()}$ by $T(n)$. That is, when you run M_i^A if it has not halted by $T(|x|)$ steps then shut it off and declare its answer to be 0. To save on notation we will also call this list

$$M_1^{()}, M_2^{()}, M_3^{()}, \ldots$$

KEY- if a set is in $DTIME^A(T(n)$ then there is an i such that M_i^A decides it in time $T(n)$.

KEY- if M_i^A decides a set then it is in $DTIME^A(T(n))$.

Hence we have a list that represents all of $DTIME^A(T(n))$.

Does more time help? YES but the proof involves some details we will skip.

Theorem 5.1 *(The Time Hierarchy Theorem) For all $T(n)$ there is a set in $DTIME^A(T(n)\log T(n)))$ that is NOT in $DTIME^A(T(n))$.*

Proof:

Let $M_1^A, M_2^A, \ldots,$ represent all of $DTIME^A(T(n))$ as described above.

We construct a set X to NOT be in $DTIME^A(T(n))P$. We will want X and to DISAGREE with M_1^A, to DISAGREE with M_2^A, etc. Lets state this in terms of REQUIREMENTS

$R_i : X$ and M_i^A differ on some string.

We want X to satisfy all of these requirements.

Here is our algorithm for X. It will be a subset of 0^*.

1. Input 0^i.
2. Run $M_i^A(0^i)$. If the results is 1 then output 0. If the results is 0 then output 1.

Note that, for all i, M_i and X DIFFER on 0^i. Hence every R_i is satisfied. Therefore $A \notin DTIME^A(T(n))$.

How do we get $A \in DTIME^A(T(n)) \log T(n))$? It is KNOWN that any multitape $DTIME^A(T(n))$ OTM can be SIMULATED by a 2-tape OTM in time $T(n) \log T(n)$. So we use this to run $M_i^A(0^i)$ in the algorithm. The entire algorithm can then be one in time $T(n) \log T(n)$. ∎

The following theorem is proved similarly:

Theorem 5.2 *(The Space Hierarchy Theorem) Let S_1 and S_2 be computable functions (think of them as increasing). Assume $\lim_{n\to\infty} \frac{S_1(n)}{S_2(n)} = \infty$. Then $DSPACE^A(S_2(n))s \subseteq DSPACE^A(S_1(n))$.*

6 So What Of It?

All of the results in the last section *relativize*. That is, if blah blah P, NP, EXP is true then, for all A, blah blah P^A, NP^A, EXP^A is true.

Intuitively what the proofs had in common is that we ran machines and observed them but never went inside of the machine. Hence we could just as well ran an oracle machine.

We won't define *relative* formally. However, the next theorem can be interrupted as saying that techniques that relativize will not suffice to resolve P vs NP.

Theorem 6.1

1. *There exists A such that $P^A = NP^A$.*
2. *There exists B such that $P^B \neq NP^B$.*

Proof:

Let

$$M_1^{()}, M_2^{()}, M_3^{()}, \ldots$$

be all poly time OTMs.

1) We construct A such that $P^A = NP^A$ by coding into it results. Formally (and we'll see later this makes sense)

$$A = \{(i, x, 0^n) \mid \text{ some path of } M_i^A(x) \text{ accepts in} \leq n \text{ steps } \}.$$

This looks circular. This looks like you have to know A to compute A. But note that if we simulate all paths of $M_i^A(x)$ for n steps then we only write down queries of size $\leq n$. Also note that the string $(i, x, 0^n)$ is longer than n. Hence if we just decide each string in order by length, we will have enough of A to determine what $M_i^A(x)$ does in n steps.

More formally:

Let $w_1, w_2, w_3, \cdots$ be all strings in $\{0,1\}^*$ in lex order. So the shorter strings come first. We have a fast map that maps strings to triples in $\{0,1\}^* \times \{0,1\}^* \times 0^*$. Make sure that if w maps to $(i, x, 0^n)$ then $n < w$. This may not be possible if w is very short, so we'll just say that we do this for strings of length ≥ 100.

ALGORITHMS FOR A

1. *Input(w_i). If $|w_i| \leq 99$ then reject.*
2. *For all strings of length $< |w_i|$ determine if $w_i \in A$ or not. Let the set of strings that are determined to be in A by A'.*
3. *Determine x, i, n such that w_i maps to $(i, x, 0^n)$.*
4. *Run $M_i^{A'}(x)$ on all paths for n steps. Note that this will make queries of length $\leq n$ hence this computation is identical to that of $M_i^A(x)$ even though we don't know all of A yet.*
5. *If there exists a computation path such that $M_i^{A'}(x)$ accepts within n steps then accept (so $w_i \in A$) else reject (so $w_i \notin A$).*

We claim that $P^A = NP^A$. Let $X \in NP^A$ via M_i^A. Assume that M_i^A runs in time $\leq p(n)$ where p is a polynomial. The following algorithm shows $X \in P^A$:

To determine $x \in X$ make the query "$(i, x, 0^{p(|x|)}) \in A$?" If it says YES then ACCEPT. If it says NO then REJECT.

2) We construct B such that $P^B \neq NP^B$ by diagonalization.

For ANY B let

$$L^B = \{0^n \mid (\exists y)[|y| = n \text{ AND } y \in B]\}.$$

Note that, for ANY B, $L^B \in NP^B$. We construct a B such that $L^B \notin P^B$.
Consider the following requirements:
R_i: $L(M_i^B) \neq L^B$.
ALGORITHM FOR B

1. *Input(0^n)*

2. *Run the algorithm for all $m < n$ to determine which requirements have been satisfied. Let R_i be the least requirement that is not satisfied. Let B' be the set of strings that have been put into B so far.*

3. *If there are no strings in B' of length n AND $p_i(n) < 2^n$ then do the following: Run $M_i^{B'}(0^n)$. If it ACCEPTS then R_i is satisfied sine there are NO strings of length n in B' and we now know to NEVER put in such a string, so $0^n \notin L^B$. If it REJECTS then, since it only ran for $p_i(n)$ steps and $p_i(n) < 2^n$, there exists some string $y \in \{0,1\}^n$ that has NOT been queries. But this string in. KEY: this will NOT interfere with the computation since y was never queries. KEY: We now have that $0^n \in L^B$ but $M_i(0^n)$ REJECTS. Hence R_i is satisfied.*

Since all of the R_i are satisfied, $L^B \in NP^B - P^B$ so $P^B \neq NP^B$.

∎