CSE547 Class 25

Jeremy Buhler

April 26, 2017

1 The Polynomial Time Hierarchy

- Alternating TMs can mix $\exists$ and $\forall$ states arbitrarily in their computations.
- But we don't always need this much generality.
- For example, the ATM we built to decide MINFORM performs a sequence of $\forall$ moves (enumerations), followed by a sequence of $\exists$ moves (choices), followed by a computation in P.
- Similarly, the ATM for YIELDN alternates sequences of $\exists$ moves, followed by sequences of $\forall$ moves, and finally (at the bottom of the recursion) does a deterministic computation.
- Let's think about machines that follow this less general schema for using alternation.
- Let C be a computation of an ATM. An $\exists$-*run* is a maximal contiguous sequence of moves in C, all of which are either deterministic or $\exists$, with no intervening $\forall$ moves.
- Similarly, we can define a $\forall$-run as a sequence free of $\exists$ moves.
- A general ATM computation alternates runs of one and the other kind of move.
- **Defn**: a language L is in class Σ_i if it is accepted by a polynomial-time alternating TM A such that
  - every computation of A contains at most i runs, and
  - the first run is of A is always a run of $\exists$ moves.
- **Defn**: a language L is in class Π_i if it is accepted by a polynomial-time alternating TM A such that
  - every computation of A contains at most i runs, and
  - the first run is of A is always a run of $\forall$ moves.

How about some intuition?

- We can think of Σ_i and Π_i as describing polynomial-time TMs wrapped in i levels of alternating quantifiers.

- For example, a machine M deciding a language in Σ_5 accepts a string w iff

$$\exists x_1.\forall x_2.\exists x_3.\forall x_4.\exists x_5.P(w, x_1 \ldots x_5)$$

  where $x_1 \ldots x_5$ are "choice strings" and P is some polynomial-time decidable predicate on the input and these strings.

- , For a machine deciding a language in Π_5, the same description applies, except that the quantifiers are all flipped so that we start with $\forall$.

- Note that Σ_1 describes exactly the languages decided by $\exists$-TMs (i.e. the class NP).

- Similarly, Π_1 describes exactly the languages decided by $\forall$-TMs (i.e. the class coNP).

- The MINFORM problem defined previously is in Π_2.

- Note that Π_i is the same as co-Σ_i.

- Also, by definition, $P = \Sigma_0 = \Pi_0$.

The Σ_i and Π_i classes constitute the *polynomial hierarchy*.

- We know that every Σ_i and Π_i is in AP and therefore PSPACE.

- However, the union

$$\text{PH} = \bigcup_{i>0} \Sigma_i = \bigcup_{i>0} \Pi_i$$

  is not the same as AP, because AP contains alternating TMs that do not consistently implement a fixed, input-independent number of runs.

- To illustrate this point, consider the PSPACE-complete problem TQBF.

- The "obvious" solution to TQBF with an ATM needs to do a number of alternations that depends on the number of quantifiers in the input formula.

- However, consider the restriction of TQBF to formulas with at most k alternating quantifiers, the first of which is $\exists$.

- This restriction, $\text{TBQF}_k^{\exists}$, is in Σ_k.

- Similarly, $\text{TBQF}_k^{\forall}$ is in Π_k.

- In fact, by a similar argument to the one we did to show the PSPACE-completeness of TQBF, these two problems are complete for their respective classes.

The polynomial hierarchy generalizes the P=NP question.

- We suspect, but do not have a proof, that Σ_{i+1} contains strictly more languages than Σ_i.

- In other words, allowing more levels of quantifier confers more power to decide things in polynomial time.

- This generalizes our belief that nondeterminism or ∀-enumeration let us solve some problems faster than deterministic computation (that is, that P is a *proper* subset of NP and co-NP).
- But maybe our belief is wrong?
- It can be shown that, if
  - $\Sigma_i = \Sigma_{i+1}$ (equivalently, $\Pi_i = \Pi_{i+1}$), or
  - $\Sigma_i = \Pi_i$

  for any i, then for all $j > i$, $\Sigma_j = \Pi_j = \Sigma_i$.
- This is called *hierarchy collapse to level* i.
- In particular, if $P = NP$, then $P = PH$.
- Equivalently, if P is a proper subset of Σ_i or Π_i for any $i > 0$, then $P \neq NP$.
- Various results separating computational complexity classes (e.g. in quantum computing and Boolean circuit models) state that "if X is possible, then PH collapses to Σ_2 (or Σ_3)".