Today

- Alternation
- ASPACE vs. TIME
- ATIME vs. SPACE
- Perspective on PSPACE
- Fortnow’s Time/Space lower bound on SAT.

Alternation

- Yesterday: Spoke about MinDNF and NP^NP.
- Possibly a new complexity class?
- Why more powerful? Can alternate between existential choices and universal choices.

Alternation ... Formally

- Turing machine with two special states \( \exists \) and \( \forall \), each with two outgoing transitions.
- \( \exists \) state accepts if one outgoing path accepts.
- \( \forall \) state accepts if both paths accept.
- Computation tree determines resources:
  - Time
  - Space
  - Alternation

Fundamental classes

Notation: ATISP\( [a, t, s] \).

- ATIME\( (t) \)
- ASPACE\( (s) \)
- \( \Sigma_i^P = \text{ATISP}[i, \text{poly}, \text{poly}] \) starting in existential quantifier.
- \( \Pi_i^P = \text{ATISP}[i, \text{poly}, \text{poly}] \) starting in universal quantifier.
- \( \text{PH} = \bigcup_i \Sigma_i^P = \bigcup_i \Pi_i^P \).

Last assertion follows from:

\[
\Sigma_i^P \subseteq \Pi_{i+1}^P, \quad P_i^P \subseteq \Sigma_i + 1^P
\]
Theorem 1: ATIME vs. SPACE

Lemma 1.1: ATIME(s) ⊆ SPACE(s).

Proof: Straightforward simulation, using one extra tape to record stack of ∃'s and ∀'s.

Lemma 1.2: SPACE(s) ⊆ ATIME(s²).

Proof: As in proof of Savitch’s theorem. Let TM A use space s on input x. Make Atime(s²) machine M(c1,c2,t) to check if A goes from configuration c1 to c2 in t steps as follows:

M(c1,c2,t):
GUESS c3 = config at time t/2
FORALL check M(c1,c3,t/2)
check M(c3,c2,t/2).

Theorem: ATIME(poly) = PSPACE.

Theorem 2: ASPACE vs. TIME

Lemma 2.1: ASPACE(s) in TIME(2O(s))

Proof: Make circuit corresponding to ASPACE computation:

• Gates = (C,i): C = config, i = time ∈ [1,2s].
• Wires = (C',i + 1) → (C,i) if C' has arrow pointing to C''. Gates at depth 2s with incoming arrows labelled REJ. Gates labelled ACC/REJ if configuration is accepting/rejecting. Gates label OR/AND depending on their type ∃/∀ etc.
• Gives circuit of size 2s - accepts iff computation accepts.

Theorem 2: ASPACE vs. TIME (contd.)

Lemma 2.2: Time(2s) in ASPACE(O(s))

Proof: Suffices to build machine M that checks if A, on input x, has contents sigma on cell i of configuration after t steps.

M(i,t,sigma): GUESS r1,r2,r3 contents of cells i-1,i,i+1 at time t-1.
Verify (r1,r2,r3,sigma) is consistent
FORALL M(i-1,t-1,r1);
M(i,t-1,r2);
M(i+1,t-1,r3);

Computational philosophy

Comparing candidates for an election: Three options:

• Candidates don’t get to campaign. We make our own decisions based on our own information.
• Candidates get to write a (bounded) position paper/single page ad campaign.
• Candidates are invited to debate.

What is a better system?
Computational philosophy (contd).

Computer scientist’s take: How complex a language can the system prove membership in?

Say thesis is $x \in L$? The masses need to be convinced. How powerful can $L$ be under these scenarios.

Model: Masses/audience as polytime computation.

- Zero input from candidates: $L \in P$.
- Fixed input from candidates: $L \in NP$.
- Full fledged debate between candidates: $L \in PSPACE$.

Debate systems

Use characterization PSPACE = ATIME(poly).

Candidates $E$ (∃) and $U$ ∀:

$E$ candidate claims $x \in L$. $U$ candidate claims $x \notin L$. Every time TM comes to ∃ state, $E$ tells us which way to go. ∀ state $U$ tells us which way to go. Audience watches the debate, and at the end makes its own conclusion on whether $x \in L$ or not, based on TM’s final state.

Complexity of Games

- Typical 2-person game: can evaluate if current position is already won or not; but hard to guess what will happen if we can find optimal strategies.

- For any such game (where win/loss depends only on current configuration and not on history), complexity of deciding who can win is in PSPACE.

- For some games (such as GO/Generalized Geog.), deciding who can win is PSPACE complete. (Again proven using ATIME(poly) = PSPACE.)

A PSPACE complete problem

$TQBF = \{ \phi | \exists x_1, \forall x_2, \ldots, Q_n x_n, \phi(x_1, x_2, \ldots, x_n)

- $x_i$ vector of $n$-variables $x_{i,1}, \ldots, x_{i,n}$.

- $\phi$ - 2CNF formula on $n^2$ variables.

- $Q_i$: alternating quantifiers; $Q_i = \exists$ if $i$ odd, and $Q_i = \forall$ if $i$ even.

Proposition: $TQBF$ is PSPACE complete.

Proof: Uses ATIME(poly) = PSPACE.
Power of Alternation

- Basic notion.
- Captures Time/Space differently.
- Next application shows how powerful it is.

Fortnow’s theorem

For today, will use LIN to mean the class of computations in NEARLY-LINEAR TIME:

$$LIN = \cup_n TIME(n \log n)$$

- Belief: SAT $\not\in L$.
- Belief: SAT $\not\in LIN$.
- Can’t prove any of the above.
- Fortnow’s theorem: Both can not be false!

Proof of Fortnow’s theorem

- For simplicity we’ll prove that if SAT $\in$ Time($n \log n$) and SAT $\in$ L then we reach a contradiction.
- Won’t give full proof: But rather give main steps, leaving steps as exercises.

Main ideas

- Alternation simulates small space computations in little time. (Savitch).
- If NTIME(t) in co-NTIME(t log t), then alternation is not powerful.
- Formal contradiction derived from: ATIME[a,t] $\not\subset$ ATIME[a-1,t/log t].
Fact 1: If $L$ in $\text{NTIME}(t)$, and $x$ of length $n$, then can construct SAT instance $\phi$ of size $t(n) \log t(n)$ such that $x$ in $L$ iff $\phi$ in SAT.

Reference: a 70's paper of Cook.

Proof: Left as exercise.

Fix $a(n) = \sqrt{\log n}$.

Fact 2: $\text{ATIME}[a,t]$ is contained in $\text{NTIME}[t (\log t)^{2a}]$

Proof: Induction on $\#\text{alternations} + \text{Fact 1}$.

Fact 3: If SAT in $L$, then $\text{NTIME}[t (\log t)^{2a}]$ in $\text{SPACE}(\log t + a \log \log t)$.

Proof: Padding

Fact 4: $\text{SPACE}[s]$ in $\text{ATISP}[b,2^{(a/b) \cdot bs}]$ in $\text{ATIME}[b,2^{(a/b)}]

Proof: Exercise 3 of PS 1.
Whither contradiction?

- If we set \( b = a - 1 \) (approximated by \( a \) in our calculations), then ...

- \( \text{ATIME}[a,t] \) is contained in \( \text{ATIME}[b,2^{(\log t + a \log a)}] \), which is a contradiction.