Today

- Alternation as a resource.
- ATIME vs. DSPACE
- ASPACE vs. DTIME
- The Polynomial Hierarchy

Story so far

- Would love to prove NP $\neq$ P.
- But Diagonalization can't do this? (Relativization).
- Circuit complexity may be able to, but haven't succeeded so far.
- Hence ... moving away from question.

Some Goals

- Try to understand the structure outside NP.
- E.g., co-NP ... (universal machines).
- Look at the power of NP, when relativized with NP. (machines with universal and existential state, but only one alternation.)
- In general, machines with both quantifiers can solve TQBF (and hence PSPACE) in polytime.
- Bounded number of alternations gives what?

DNF Minimization

Defn: $\textsc{MinDNF}$ is the language consisting of pairs (ϕ, k), such that ϕ is a DNF formula such that no DNF formula with fewer than k literals is equivalent to ϕ.

Prop: $\textsc{MinDNF}$ is in $\mathsf{NP}^{\mathsf{NP}}$.

Proof: Below is an NP oracle machine M that accesses a SAT oracle:

- Guess a formula ψ with fewer than k literals.
- Ask SAT oracle if there exists an assignment x such that $\psi(x) \neq \phi(x)$.
- Accept if oracle says NO.

Note: we get the power to negate the oracles' response (or do any other polynomial time computation on it).

Alternation

- Suppose we built this into a Turing machine.
- Machine has two special states: $\exists$ and $\forall$, both with two arcs leading out.
  - $\exists$ state accepts if one of the two paths leading out accepts.
  - $\forall$ state accepts if both paths accept.
- Alternation = Resource: write down computation tree: Count max. $\#$ times we alternate enter an $\exists$ node and then a $\forall$ node.
- This is a (valuable) resource!

Alternations

- Henceforth will consider machines both existential and universal states. with three resources restrictions: space, time, alternations.
- Will see ...
  - Leads to intriguing recharacterizations of known classes.
  - Introduces a new hierarchy of classes; and a new assumption (the mother of all assumptions in complexity theory).
  - Leads to a strong lower bound (e.g., SAT does not have linear size, log-depth uniform circuits).

Alternating complexity classes

- Classes:
  - ATIME$[t]$ = Languages accepted by ATMs running in time $t(n)$.
  - ASPACE$[s]$ = Languages accepted by ATMs using space $s(n)$.
  - (only of technical interest) ATISP$[a, t, s]$ = ... $a(n)$ alternations, $t(n)$ time, and $s(n)$ space.
- PH: Σ_i^P = languages accepted by polytime bounded ATMs starting in existential state and making at most $i - 1$ alternations.

Basic theorems about alternations

Thm 1: ATIME(f) $\subseteq$ SPACE(f) $\subseteq$ATIME(f^2).

Thm 1: ASPACE(f) = TIME($2^{O(f)}$).

Theorem 1: ATIME vs. SPACE

Lemma 1.1: ATIME(s) $\subseteq$ SPACE(s).

Proof: Straightforward simulation, using one extra tape to record stack of $\exists$'s and $\forall$'s.

Lemma 1.2: SPACE(s) $\subseteq$ ATIME(s^2).

Proof: As in proof of Savitch's theorem. Let TM A use space s on input x. Make Atime(s^2) 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($2^{O(s)}$)

Proof: Make circuit corresponding to ASPACE computation:

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

Theorem 2: ASPACE vs. TIME (contd.)

Lemma 2.2: Time(2^s) 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 ($\exists$) and U $\forall$:

E candidate claims $x \in L$. U candidate claims $x \notin L$. Every time TM comes to $\exists$ state, E tells us which way to go. $\forall$ 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 $= \{\phi | \exists \mathbf{x}_1, \forall \mathbf{x}_2, \ldots, Q_n \mathbf{x}_n, \phi(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n)$

- $\mathbf{x}_i$ vector of n-variables $x_{i,1}, \ldots, x_{i,n}$.
- ϕ - 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.