COMPSCI 501: Formal Language Theory

Lecture 35: Alternation

Marius Minea
marius@cs.umass.edu

University of Massachusetts Amherst

22 April 2019

Review: Probabilistic Complexity Classes

BPP (bounded error): accepts/rejects with error probability $\epsilon < 1/2$

Amplification lemma: can make error $2^{-p(n)}$ for any polynomial

RP (randomized poly-time): always *rejects* when it should
re-runs make acceptance error arbitrarily small
coRP: always *accepts* when it should

Clearly **RP** $\subseteq$ **BPP** (reject error is zero), likewise **coRP** $\subseteq$ **BPP**

P $\subseteq$ **RP**: no nondeterminism, always right answer

RP $\subseteq$ **NP**: NTM needs no coin, guesses correct path

BPP ? NP (no relation is known). It it believed that **P** $=$ **BPP**.

Polynomial identity testing (last lecture) is in coRP, unknown if in P.

Alternation

Def. **Alternating Turing machine** $=$ nondeterministic TM with two types of states (in addition to q_{acc}, q_{rej}):

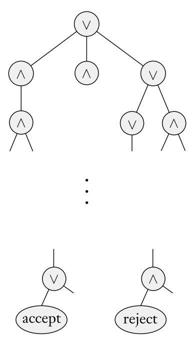
- **universal states** (AND, $\wedge$) accepts if *all* successors do
- **existential states** (OR, $\vee$) accepts if *some* successor does

We've seen this in AND-OR games

Very powerful model of computation

AND / OR need not strictly alternate

Will refine hierarchy
based on number of alternations



Alternating Time and Space Complexity

ATIME$(t(n)) = \{L \mid L$ decided by $O(t(n))$ time alternating TM$\}$

ASPACE$(f(n)) = \{L$ decided by $O(f(n))$ space alternating TM$\}$

AP: alternating polynomial time

APSPACE: alternating polynomial space

AL: alternating logarithmic space

Example: Tautology

TAUT $= \{\phi$ that evaluates to true on all variable assignments$\}$

ϕ tautology $\Leftrightarrow \neg\phi$ *not* satisfiable.

$\Longrightarrow$ *TAUT* $\in$ coNP (in fact, coNP-complete)

TAUT is in AP:

1. **universally** select **all** truth assignments   compare:
   existentially select *one* assignment for *SAT*
2. (next level) evaluate assignment
3. *accept*/*reject* depending on assignment

Will reject if at least one assignment rejects.

By same principle, every coNP problem is in AP

MIN-FORMULA

A Boolean formulas is *minimal* if there is no shorter formula equivalent to it.

Can decide *MIN-FORMULA* in AP:

1. **universally** select all formulas ψ shorter than ϕ
   doable, levels of choices $=$ length of formula - 1
2. **existentially** select a truth assignment
3. evaluate ϕ and ψ
4. *accept* if different, *reject* if same

Accepts if for *all* shorter formulas, *some* assignment differs

Space-Time Connections

(1) ATIME($f(n)$) $\subseteq$ SPACE($f(n)$) $\subseteq$ ATIME($f^2(n)$) for $f(n) \geq n$

(2) ASPACE($f(n)$) $=$ TIME($2^{O(f(n))}$) for $f(n) \geq \log n$

Consequences:

AL $=$ P   (2), $f(n) = \log n$

AP $=$ PSPACE   (1), $f(n) =$ poly(n)

APSPACE $=$ EXPTIME   (2), $f(n) =$ poly(n)

Proof: ATIME($f(n)$) $\subseteq$ SPACE($f(n)$) for $f(n) \geq n$

Simulate alternating $O(f(n))$ time in deterministic $O(f(n))$ space.

Check acceptance in computation, observing AND/OR rules.

Recursive DFS.

Naive: $O(f(n)$ depth, $O(f(n))$ space / config. $\implies$ $O(f^2(n))$

Efficient: keep encoding of choices at each level.
$\implies$ constant space per level (bounded branching)

Improved space: $O(f(n))$

Proof: SPACE($f(n)$) $\subseteq$ ATIME($f^2(n)$) for $f(n) \geq n$

Simulate $O(f(n))$ space in alternating $O(f^2(n))$ time.

Suggests Savitch's theorem. Recall:

YIELD(c_1, c_2, t): go from config. c_1 to c_2 within t steps

1. **existentially** chooses intermediate c_m
2. **universally** evaluate YIELD($c_1, c_m, t/2$) and YIELD($c_m, c_2, t/2$)
3. recurse

Space $O(f(n))$ $\implies$ at most $2^{df(n)}$ configurations for some d
Initial call with $t = df(n)$.

Time used: $O(f(n))$ to generate/write configuration at each level
Recursion depth: $\log 2^{df(n)} = O(f(n))$. $\implies$ total $O(f^2(n))$

Proof: ASPACE($f(n)$) $\subseteq$ TIME($2^{O(f(n))}$) for $f(n) \geq \log n$

Simulate alternating $O(f(n))$ space in deterministic $2^{O(f(n))}$ time

Construct configuration graph – space $df(n)$ per configuration.
$\implies$ $2^{O(f(n))}$ configurations

Repeatedly mark accepting configurations bottom up
(reverse topological order)

Each scan takes $2^{O(f(n))}$ time (bounded node degree)

Each scan markes some new node (else done) $\implies$ $2^{O(f(n))}$ scans

Time complexity: $2^{O(f(n))} \cdot 2^{O(f(n))} = 2^{O(f(n))}$

Proof: ASPACE($f(n)$) $\supseteq$ TIME($2^{O(f(n))}$) for $f(n) \geq \log n$

"$\Leftarrow$" Simulate $2^{O(f(n))}$ time in alternating $O(f(n))$ space

Concept: tableau of configurations, $2^{O(f(n))} \times 2^{O(f(n))}$

Can't store tableau, must only store pointers.
one pointer: size $\log 2^{O(f(n))} = O(f(n))$.

Alternation allows guess & verify without retaining stack!

Top-level: check if lower-left corner can be accepting

1. **existentially** guess contents of (three) cell parents
   check they match transition relation
2. **universally** branch to check the parents

Space needed is just one pointer to next cell $\implies$ $O(f(n))$

Polynomial Time Hierarchy

Defines hierarchy within AP $=$ PSPACE, by bounding number of alternations between $\wedge$ and $\vee$.

Σ_i alternating TM: at most i runs of existential or universal steps, starting with existential steps.

Π_i alternating TM: at most i runs of existential or universal steps, starting with universal steps.

$\Sigma_i\mathbf{P} = \bigcup_k \Sigma_i\text{TIME}(n^k)$ $\Pi_i\mathbf{P} = \bigcup_k \Pi_i\text{TIME}(n^k)$

PH $= \bigcup_i \Sigma_i$P $= \bigcup_i \Pi_i$P

P $= \Sigma_0$P $= \Pi_0$P (no nondeterminism, no alternation)

NP $= \Sigma_1$P, coNP $= \Pi_1$P.