CS 221 Lecture 6

2/8/2018

TODAY: ALTERNATION, TIME, SPACE

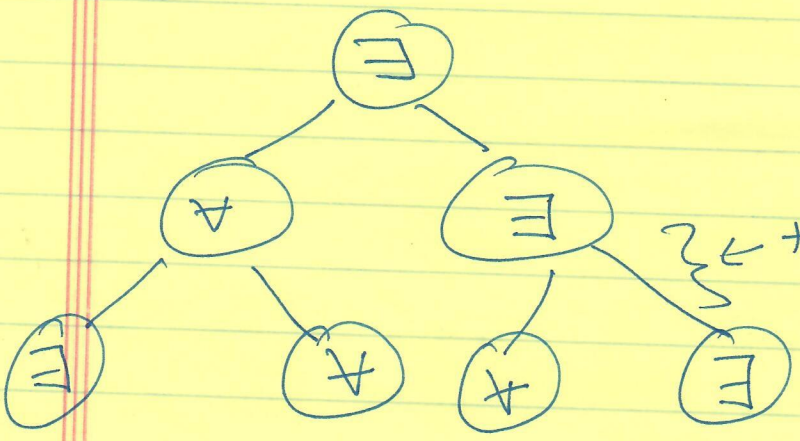
- ATIME vs. SPACE
- ASPACE vs. TIME
- FORTNOW'S THEOREM: $SAT \in L \Rightarrow SAT \notin TIME(n^{1+o(1)})$

———x———

Alternating Algorithm (A.T.M.)

Two special instructions / states: "$\forall x\ f(x)$" or "$\exists x\ f(x)$".

Computation given by tree



← traditional computation.

Time = depth of tree

Space = max space on any path.

ATIME(poly) $\cong$ GO (the game)

ASPACE(poly) $\cong$ CHESS (!)

GO vs. CHESS : - Both have "poly(n)" states space
(described by poly(n) bits)

- Both have poly time verifiability
(is current config winning for white/black?)

(is $state_1 \to state_2$ legitimate move)

- GO: Only poly(n) moves
(# pieces on board increasing with moves)

- CHESS: Could have exp(n) moves without repeating states.

GO (appropriate generalization): ATIME(n) ~~time~~ complete

CHESS ( ) : ASPACE(n) complete.

Classical Relationships

Thm 1: $ATIME(poly) = PSPACE$

Thm 2: $ASPACE(\log n) = P$

} Funny inversion of time & space!

Proof of Thm 1: "$Space(S) \subseteq ATIME(S^2) \subseteq SPACE(S^2)$"

- 2nd containment ... same as QBF $\in$ PSPACE; can simulate depth d tree using $O(d)$ space.
- 1st containment ... Savitch's Theorem ...

$$PSPACE \leq QBF \in ATIME(n).$$

Thm 2 Proof: $TIME(2^{2S(S)}) \subseteq ASPACE(S) \subseteq TIME(2^S)$

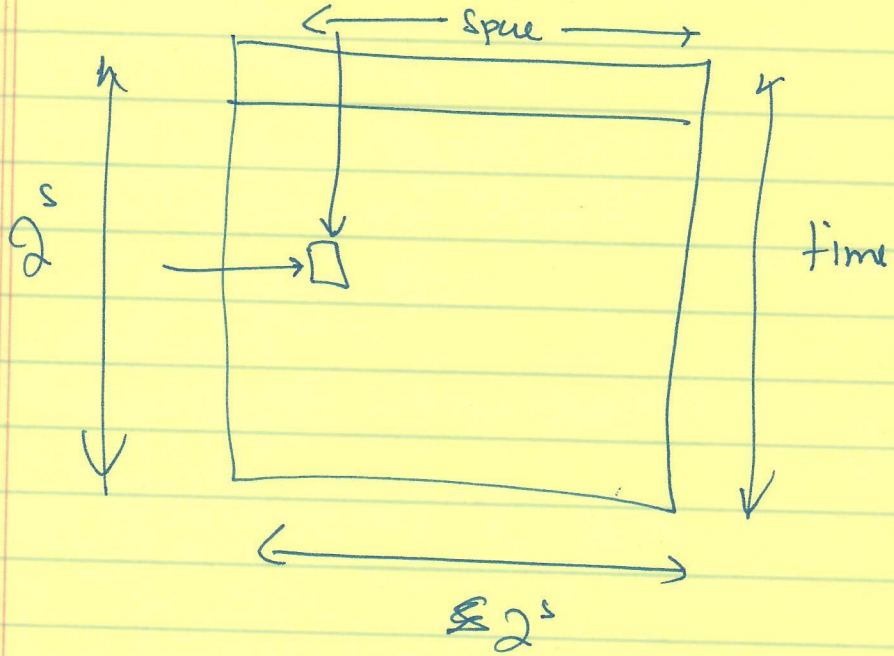
2nd containment: Enumerate all configurations; Build Digraph of configurations. Analyze graph.

(helps to augment algorithm with time counter; reject if counter $> 2^S$.)

Converts graph to DAG.

1st Containment: $\text{Time}(2^s) \subseteq \text{ASPACE}(O(s))$:

Key idea: look at state evolution



CONTENT (i,j,b) = true if $(i,j)^{th}$ bit $= b$

Key idea: content (i,j,b) = function of few previous contents.

e.g. content $(\cancel{\quad}, i-1, j+t, b_t)$ for $t \in [\pm c]$

~~to~~ Goal = Verify content $(2^s, 0, 1) = 1$.

Verifying content (i,j,b)

$\exists$ player $\xrightarrow{b_t \text{ for } t \in [\pm c]}$

$\xleftarrow{t_0 \in [\pm c]}$ $\forall$

Now. verify content $(i-1, j+t_0, b_{t_0})$:

FORTNOW'S THEOREM : $SAT \in L \Rightarrow SAT \notin TIME(n^{1+o(1)})$

Intuition: Assume $SAT \in L$ AND $SAT \in TIME(n^{1+o(1)})$

$SAT \in TIME(n^{1+o(1)})$

$\Rightarrow$ Non-determinism not powerful

$\Rightarrow$ Alternation not powerful - for time complexity

$SAT \in L$

$\Rightarrow Time(t) \cong SPACE(\log t)$

But Alternation is powerful for small ~~time~~ space!

$\Rightarrow$ Contradiction!

Formal Proof:

Put together contradict Time Hierarchy

① "Alternation is Powerful":

$$SAT \in L \Rightarrow Time(T(n)) \subseteq ATIME\left[a, \tilde{T}(n)^{1/a}\right] \quad \forall a$$

↑ # alternations

[alternation reduces time complexity]

② $\forall a \exists k_a$ s.t. $\forall \epsilon$

$\downarrow SAT \in Time(n^{1+\epsilon})$

$\Rightarrow$ ~~$ATIME(T(n)) \in$~~

~~$ATIME[a, T(n)] \subseteq TIME[T(n) \log^{k} T(n)]$~~

$$ATIME[a, T(n)] \subseteq TIME\left[T(n)^{1+k_a\epsilon}\right]$$

Proof of (1): SAT $\in L \Rightarrow$

$\Rightarrow$ $NTIME(T(n)) \subseteq SPACE(c \log T(n))$

[Not immediate. Uses strong NP-completeness [Look]]

$\Rightarrow$ $TIME(T(n)) \subseteq SPACE(c \log T(n))$.

$\subseteq TISP(c \log T(n), T(n)^c)$

"A la Savitch" $\subseteq ATIME(a, T(n)^{c/a})$

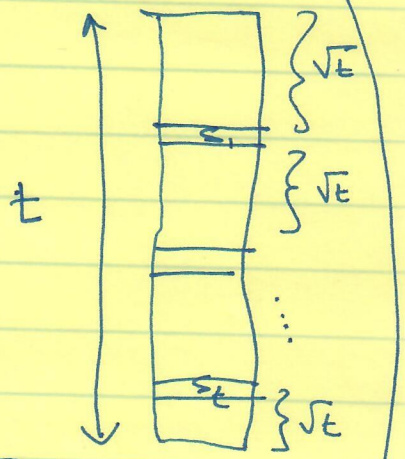
"a la Savitch"

Example $a=2$

$\exists$ $S_1, S_2 \cdots \sqrt{t}$ $\forall$

i

"$S_i \Rightarrow S_{i+1}$?"



shows $SPACE(s(n)) \subseteq ATIME[2, 2^{s(n)/2}]$

Proof of (2): $SAT \in TIME(n^{1+\epsilon})$

$\Rightarrow NTIME(T(n)) \subseteq TIME(T(n)^{1+\epsilon})$

$\Rightarrow \exists.\forall.\ TIME(T(n)) \subseteq \exists.TIME(T(n)^{1+\epsilon})$

$\subseteq TIME(T(n)^{(1+\epsilon)^2})$

$\Rightarrow \vdots$

$\Rightarrow ATIME(a, T(n)) \subseteq TIME(T(n)^{(1+\epsilon)^a}) \approx\approx TIME(T(n)^{1+\epsilon a})$

$\boxtimes$

[Concludes "Proof" of Fortnow's Theorem]

Caveats with Proof:

- Played fast & loose with NTIME & SAT & in general $ATIME(a,\ldots)$ & a-quantified SAT.
- Problems are equivalent with $\log T(n)$ loss (& in which this makes no difference), but verifying equivalence at all stages makes correct proof a bit more dairy.