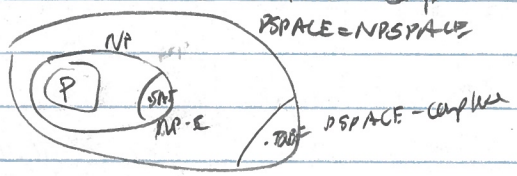
Michael Forbes
miforbes@illinois.edu

2019-01-09-31.4 → 2019-02-CS.1
2019-02-05.2 ← CS 579

CS 579 Computational Complexity · Lecture 7 1/4 22

last time: space complexity
PSPACE, NPSPACE
TQBF ∈ PSPACE

today: PSPACE vs NPSPACE
TQBF is PSPACE complete



ex: NPSPACE [nondeterministic poly space]

Q: convert beer to wine? one letter at a time, in English

beer → beet → beat → seat → sent → went → want → wand → wind
↘ bent ↗
[word ladder]

$LADDER_{DFA} = \{ \langle M, s, t \rangle : M$ DFA $L(M)$ contains ladder $s \to t$ [defn not crucial]

$s = x_0 \to x_1 \to x_2 \to \cdots \to x_m = t$
- $x_i \in L(M)$
- $\Delta(x_i, x_{i+1}) = 1$ [hamming dist]

Prop. $\in$ NPSPACE

Pf: "on $\langle M, s, t \rangle$:
1) check if $s, t \in L(M)$, $|s| = |t|$, reject if not
2) $s_0 := s$, $n := |s| = |t|$
3) do
a) guess $s_{i+1} \in \Sigma^n$
b) if $\Delta(s_i, s_{i+1}) \neq 1$, $s_{i+1} \notin L(M)$ reject
c) if $s_{i+1} = t$, accept

"

Correctness: clear

complexity: space: only store M, s, t, s_i, s_{i+1} $O(n)$ space
time: [halts] need to check $\leq |\Sigma|^n$ steps $O(n)$ space

[PSPACE subset] [might have long ladders, so not NP]

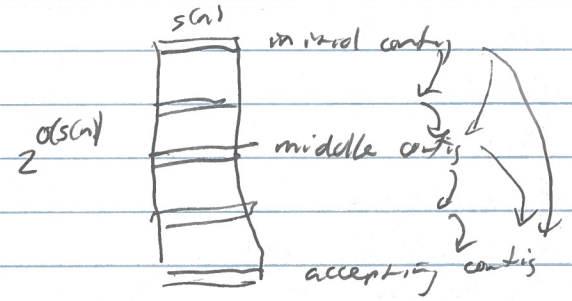
thm [Savitch]: $s(n) : \mathbb{N} \to \mathbb{N}$
- $s(n) \geq n$ (mostly)
- space constructible: $1^n \to 1^{s(n)}$ can be computed in $O(s(n))$ space

then $NSPACE(s) \subseteq SPACE(s^2)$

Rmk:
- not needed [but simplifies proof slightly, see book for full details]
- open if it is tight.

Michael Forbes
MForbes@illinois.edu
2019-02-05.2 ⇆ 2019-02-05.1
2019-02-05.3
CS 579

idea:



[Q: get start → end?]
[long path, many choices per step]
[reverse]

Pf: A decided by NTM N in space s(n)
want: TM M accepting A in space $O((s(n))^2)$
def: C_1, C_2 configurations of N [of as in cook lemma]
↳ space-s(n)
$C_1 \xrightarrow{\leq t} C_2$ if N starting on C_1 can reach C_2 in $\leq t$ steps

wlog: N has unique reachable space-s(n) accepting config
[for convenience] = #q_acc ␣ ⋯ ␣ #
s(n)-3

Pf: replace N w/ N' ↩ uses the tape
N' = "simulate N"
uses space conservability
- if reaches accept state,
  - overwrite first s(n) cells w/ ␣
  - move head all the way to left
  - accept
- otherwise reject

space $O(s(n))$ [need to know s(n)]

lem: N accepts x iff $C_{init_x} \to C_{acc}$ in $\leq 2^{O(s(n))}$ steps
[from SPACE(s) ⊆ TIME(2^{s})]

recursive procedure to decide $C_{start} \xrightarrow{\leq t} C_{end}$

$O(s(n))$ $t > 1$: 1) for all C_{mid} [recursive knowing s(n)] [wlog t is power of 2, as $\leq t$ steps]
reuse space → a) recursively test $C_{start} \xrightarrow{t/2} C_{mid}$
→ b) " $C_{mid} \xrightarrow{t/2} C_{end}$
c) if both possible, accept
2) reject

$t = 1$: 1) accept if - $C_{start} = C_{end}$
- $C_{start} \to C_{end}$ in 1 transition of N

Michael Forbes
mforbes@illinois.edu

correctness: clear
complexity: recursion depth: $D(t) \le 1 + D(t/2)$
$= O(\lg t)$
space $S(t) \le O(s(n)) + S(t/2)$ ↳ reuse space
↑ recursion stack, C_{mid}
$= O((\lg t) \cdot s(n))$
↳ $t \le 2^{O(s(n))}$
$= O((s(n))^2)$

Cor: NPSPACE = PSPACE
$= \bigcup_k NSPACE(n^k) \subseteq \bigcup_k SPACE(n^{2k})$

Q: P vs PSPACE?

def: B PSPACE complete if
- $B \in$ PSPACE
- $\forall A \in$ PSPACE $A \le_p B$ ← poly time reduc

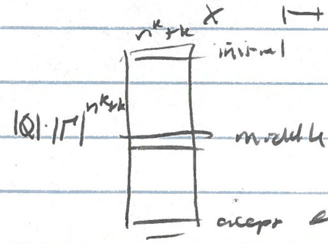
Cor: B PSPACE complete. $B \in P$ iff P = PSPACE [so B true ? and suffices after ≤]

Thm TQBF = { ⟨φ⟩ : φ is true quantified boolean formula
$= Q_1 x_1 Q_2 x_2 \ldots Q_n x_n \; \psi(x_1, \ldots x_n)$
↑ {∃, ∀}

is PSPACE complete

Pf. a) ∈ PSPACE [last time]
b) PSPACE-hard

A ∈ PSPACE decided by $n^k + k$ space TM M
want A $\le_p$ TQBF
$x \mapsto \varphi_x$ = "M accepts x" [like Cook Levin]



idea $\varphi_{C_{start}, C_{end}, t}$ = "M starting at C_1 yields C_2 in ≤ t steps"
$t > 1$: $= \exists C_{mid} \; \varphi_{C_{start}, C_{mid}, t/2} \wedge \varphi_{C_{mid}, C_{end}, t/2}$
$O(s n^k)$ boolean variables [can recursively expand]

$t = 1$: $= C_{start} = C_{mid} \vee$ "$C_{start} \vdash C_{mid}$"
↖ same as Cook Levin

Michael Forbes
mforbes@illinois.edu
2019-02-05.4 ⇐ → 2019-02-05.3
2019-02-07-1

Clm: A acc x iff $\varphi_{c_{init_x}, c_{acc}, 2^{O(s n^k)}} \in TQBF$

[correctness is clear] ↳ - only $\exists$-quantifiers [$\subseteq NP$]

- exponentially big [so not poly time reduction]

idea: use $\forall$ to simulate "and"

$$\varphi_{c_{start}, c_{end}, t} = \exists c_{mid}\ \forall (c_1, c_2) \in \{ (c_{start}, c_{mid}), (c_{mid}, c_{end}) \}\ \varphi_{c_1, c_2, t/2}$$

↓

$$\forall (c_1, c_2) \left(\begin{array}{l} (c_1, c_2) = (c_{start}, c_{mid}) \vee \\ (c_1, c_2) = (c_{mid}, c_{end}) \end{array} \right) \longrightarrow \varphi_{c_1, c_2, t/2}$$

$$a \to b \equiv \neg a \vee b$$

correctness: clear

complexity: recursion depth: $D(t) \leq 1 + D(t/2) = O(\lg t)$

output size: $S(t) \leq O(s n^k) + S(t/2)$
$\leq O(\lg t \cdot n^k)$
↳ $t \leq 2^{O(n^k)}$
$= O(n^{2k})$

time $\in$ poly(n)

today: PSPACE = NPSPACE
TQBF is PSPACE-complete

next time: - more PSPACE completeness
- log space