CS21
Decidability and Tractability

Lecture 18
February 14, 2024

1

Time Hierarchy Theorem

Theorem: for every proper complexity function $f(n) \geq n$:

$$\mathbf{TIME(f(n)) \subsetneq TIME(f(2n)^3).}$$

- Note: $P \subseteq TIME(2^n) \subsetneq TIME(2^{(2n)^3}) \subseteq EXP$
- Most natural functions (and 2^n in particular) are proper complexity functions. We will ignore this detail in this class.

2

Time Hierarchy Theorem

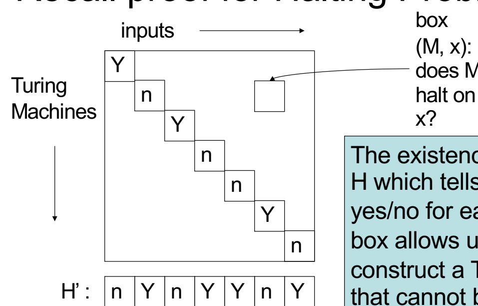
Theorem: for every proper complexity function $f(n) \geq n$:

$$\mathbf{TIME(f(n)) \subsetneq TIME(f(2n)^3).}$$

- Proof idea:
  - use diagonalization to construct a language that is not in TIME(f(n)).
  - constructed language comes with a TM that decides it and runs in time $f(2n)^3$.

3

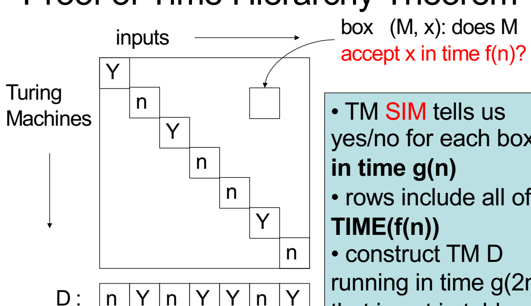
Recall proof for Halting Problem



The existence of H which tells us yes/no for each box allows us to construct a TM H' that cannot be in the table.

4

Proof of Time Hierarchy Theorem



- TM SIM tells us yes/no for each box **in time g(n)**
- rows include all of **TIME(f(n))**
- construct TM D running in time g(2n) that is not in table

5

Proof of Time Hierarchy Theorem

- Proof:
  - SIM is TM deciding language
    { <M, x> : M accepts x in ≤ f(|x|) steps }
  - Claim: SIM runs in time $g(n) = f(n)^3$.
  - define new TM D: on input <M>
    - if SIM accepts <M, <M>>, reject
    - if SIM rejects <M, <M>>, accept
  - D runs in time g(2n)

6

Proof of Time Hierarchy Theorem

- Proof (continued):
  - suppose M in **TIME(f(n))** decides L(D)
    - M(<M>) = SIM(<M, <M>>) ≠ D(<M>)
    - but M(<M>) = D(<M>)
  - contradiction.

7

Proof of Time Hierarchy Theorem

- Claim: there is a TM SIM that decides

  {<M, x> : M accepts x in ≤ f(|x|) steps}

  and runs in time $g(n) = f(n)^3$.
- Proof sketch: SIM has 4 work tapes
  - contents and "virtual head" positions for M's tapes
  - M's transition function and state
  - f(|x|) "+"s used as a clock
  - scratch space

8

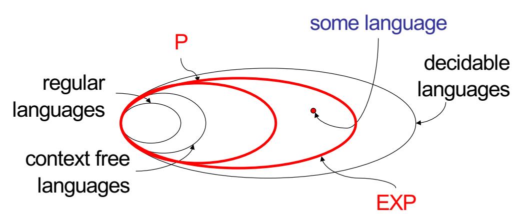
Proof of Time Hierarchy Theorem

- Proof sketch (continued): 4 work tapes
  - contents and "virtual head" positions for M's tapes
  - M's transition function and state
  - f(|x|) "+"s used as a clock
  - scratch space
  - initialize tapes
  - simulate step of M, advance head on tape 3; repeat.
  - can check running time is as claimed.

9

So far…

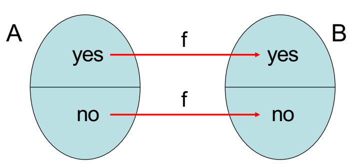
- We have defined the complexity classes P (polynomial time), EXP (exponential time)



10

Poly-time reductions

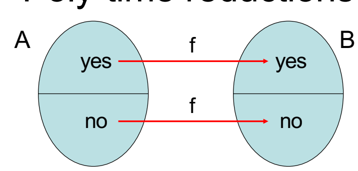
- Type of reduction we will use:
  - "many-one" poly-time reduction (commonly)
  - "mapping" poly-time reduction (book)



reduction from language A to language B

11

Poly-time reductions



- function f should be poly-time computable

<u>Definition</u>: f : Σ*→ Σ* is poly-time computable if for some $g(n) = n^{O(1)}$ there exists a g(n)-time TM M_f such that on every w ∈ Σ*, M_f halts with f(w) on its tape.

12

Poly-time reductions

Definition: A $\leq_P$ B ("A reduces to B") if there is a poly-time computable function f such that for all w

$$w \in A \Leftrightarrow f(w) \in B$$

- as before, condition equivalent to:
  – YES maps to YES *and* NO maps to NO
- as before, meaning is:
  – B is at least as "hard" (or expressive) as A

Poly-time reductions

Theorem: if A $\leq_P$ B and B $\in$ P then A $\in$ P.

Proof:
– a poly-time algorithm for deciding A:
– on input w, compute f(w) in poly-time.
– run poly-time algorithm to decide if f(w) $\in$ B
– if it says "yes", output "yes"
– if it says "no", output "no"

Example

- 2SAT = {CNF formulas with 2 literals per clause for which there exists a satisfying truth assignment}
- L = {directed graph G, and list of pairs of vertices $(u_1, v_1), (u_2, v_2), \ldots, (u_k, v_k)$, such that there is no i for which [u_i is reachable from v_i in G and v_i is reachable from u_i in G]}
- We gave a poly-time reduction from 2SAT to L.
- determined that 2SAT $\in$ P from fact that L $\in$ P

Hardness and completeness

- Reasonable that can efficiently transform one problem into another.
- Surprising:
  – can often find a special language L so that **every** language in a given complexity class reduces to L!
  – powerful tool

Hardness and completeness

- Recall:
  – a language L is a set of strings
  – a complexity class C is a set of languages

Definition: a language L is C-hard if for every language A $\in$ C, A poly-time reduces to L; i.e., A $\leq_P$ L.

meaning: L is at least as "hard" as anything in C

Hardness and completeness

- Recall:
  – a language L is a set of strings
  – a complexity class C is a set of languages

Definition: a language L is C-complete if L is C-hard and L $\in$ C

meaning: L is a "hardest" problem in C

An EXP-complete problem

- Version of A_{TM} with a time bound:

  ATM_B = {<M, x, m> : M is a TM that accepts x within at most m steps}

Theorem: ATM_B is EXP-complete.

Proof:

- what do we need to show?

19

An EXP-complete problem

- ATM_B = {<M, x, m> : M is a TM that accepts x within at most m steps}
- Proof that ATM_B is EXP-complete:
  - Part 1. Need to show $ATM_B \in$ EXP.
    - simulate M on x for m steps; accept if simulation accepts; reject if simulation doesn't accept.
    - running time $m^{O(1)}$.
    - n = length of input ≥ $\log_2 m$
    - running time $\leq m^k = 2^{(\log m)k} \leq 2^{(kn)}$

20

An EXP-complete problem

- ATM_B = {<M, x, m> : M is a TM that accepts x within at most m steps}
- Proof that ATM_B is EXP-complete:
  - Part 2. For each language $A \in$ EXP, need to give poly-time reduction from A to ATM_B.
  - for a given language $A \in$ EXP, we know there is a TM M_A that decides A in time $g(n) \leq 2^{n^k}$ for some k.
  - what should reduction f(w) produce?

21

An EXP-complete problem

- ATM_B = {<M, x, m> : M is a TM that accepts x within at most m steps}
- Proof that ATM_B is EXP-complete:
  - f(w) = <M_A, w, m> where $m = 2^{|w|^k}$
  - is f(w) poly-time computable?
    - hardcode M_A and k…
  - YES maps to YES?
    - $w \in A \Rightarrow$ <M_A, w, m> $\in ATM_B$
  - NO maps to NO?
    - $w \notin A \Rightarrow$ <M_A, w, m> $\notin ATM_B$

22

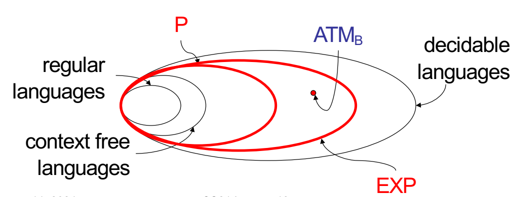
An EXP-complete problem

- A C-complete problem is a surrogate for the entire class C.
- For example: if you can find a poly-time algorithm for ATM_B then there is automatically a poly-time algorithm for every problem in EXP (i.e., EXP = P).
- Can you find a poly-time alg for ATM_B?

23

An EXP-complete problem

- Can you find a poly-time alg for ATM_B?
- NO! we showed that P $\subsetneq$ EXP.
- ATM_B is not tractable (intractable).



24

Back to 3SAT

- Remember 3SAT ∈ EXP
  3SAT = {formulas in CNF with 3 literals per clause for which there exists a satisfying truth assignment}
- It seems hard. Can we show it is intractable?
  - formally, can we show 3SAT is EXP-complete?

Back to 3SAT

- can we show 3SAT is EXP-complete?
- Don't know how to. Believed unlikely.
- One reason: there is an important positive feature of 3SAT that doesn't seem to hold for problems in EXP (e.g. ATM_B):

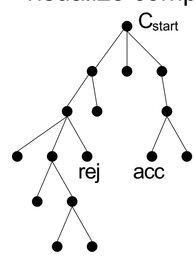
3SAT is decidable in polynomial time by a nondeterministic TM

Nondeterministic TMs

- Recall: nondeterministic TM
- informally, TM with several possible next configurations at each step
- formally, A NTM is a 7-tuple
  $(Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$ where:
  - everything is the same as a TM except the transition function:
    $\delta: Q \times \Gamma \rightarrow \mathcal{P}(Q \times \Gamma \times \{L, R\})$

Nondeterministic TMs

visualize computation of a NTM M as a tree



- nodes are configurations
- leaves are accept/reject configurations
- M accepts if and only if there exists an accept leaf
- M is a decider, so no paths go on forever
- running time is max. path length

The class NP

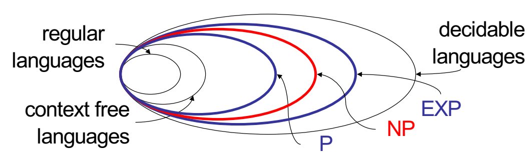
Definition: TIME(t(n)) = {L : there exists a TM M that decides L in time O(t(n))}

$$P = \cup_{k \geq 1} TIME(n^k)$$

Definition: NTIME(t(n)) = {L : there exists a NTM M that decides L in time O(t(n))}

$$NP = \cup_{k \geq 1} NTIME(n^k)$$

NP in relation to P and EXP



- P ⊆ NP (poly-time TM ***is*** a poly-time NTM)
- NP ⊆ EXP
  - configuration tree of n^k-time NTM has ≤ b^{n^k} nodes
  - can traverse entire tree in $O(b^{n^k})$ time

we do not know if either inclusion is proper