CS21
Decidability and Tractability

Lecture 19

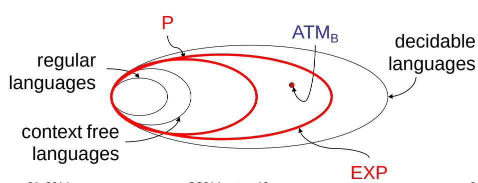
February 21, 2014

Outline

- the class NP
  - an NP-complete problem
  - alternate characterization of NP
  - 3-SAT is NP-complete

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).



Back to 3SAT

- Remember $3SAT \in EXP$
  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

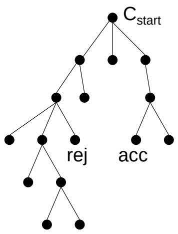
Nondeterministic TMs

- 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:

$$\delta: Q \times \Gamma \rightarrow \wp(Q \times \Gamma \times \{L, R\})$$

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

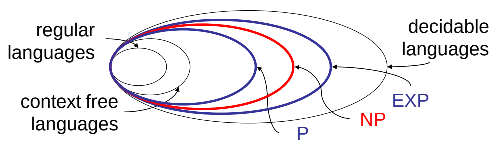
Definition: TIME(t(n)) = {L : there exists a TM M that decides L in time O(t(n))}

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

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

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

NP in relation to P and EXP



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

we do not know if either inclusion is proper

An NP-complete problem

- Version of A_{TM} with a unary time bound, and NTM instead of TM:

  $ANTM_U$ = {<M, x, 1^m> : M is a NTM that accepts x within at most m steps}

Theorem: $ANTM_U$ is NP-complete.

Proof:

- what do we need to show?

An NP-complete problem

- $ANTM_U$ = {<M, x, 1^m> : M is a NTM that accepts x within at most m steps}
- Proof that $ANTM_U$ is NP-complete:
  - Part 1. Need to show $ANTM_U \in$ NP.
    - simulate NTM M on x for m steps; do what M does
    - running time $m^{O(1)}$.
    - n = length of input $\geq$ m
    - running time $\leq m^k \leq n^k$

An NP-complete problem

- $ANTM_U$ = {<M, x, 1^m> : M is a NTM that accepts x within at most m steps}
- Proof that $ANTM_U$ is NP-complete:
  - Part 2. For each language A $\in$ NP, need to give poly-time reduction from A to $ANTM_U$.
  - for a given language A $\in$ NP, we know there is a NTM M_A that decides A in time $g(n) \leq n^k$ for some k.
  - what should reduction f(w) produce?

An NP-complete problem

- $ANTM_U$ = {<M, x, 1^m> : M is a NTM that accepts x within at most m steps}
- Proof that $ANTM_U$ is NP-complete:
  - f(w) = <M_A, w, 1^m> where m = $|w|^k$
  - is f(w) poly-time computable?
    - hardcode M_A and k…
  - YES maps to YES?
    - $w \in A \Rightarrow$ <M_A, w, 1^m> $\in ANTM_U$
  - NO maps to NO?
    - $w \notin A \Rightarrow$ <M_A, w, 1^m> $\notin ANTM_U$

An NP-complete problem

- If you can find a poly-time algorithm for $ANTM_U$ then there is automatically a poly-time algorithm for every problem in NP (i.e., NP = P).
- No one knows if can find a poly-time alg for $ANTM_U$…

Poly-time verifiers

- NP = {L : L decide… e NTM}

“witness” or “certificate”

- Very useful alternate definition of NP:

Theorem: language L is in NP if and only if it is expressible as:

efficiently verifiable

$$L = \{ x \mid \exists y, |y| \le |x|^k, (x, y) \in R \}$$

where R is a language in P.

- poly-time TM M_R deciding R is a “verifier”

Poly-time verifiers

- Example: 3SAT expressible as

3SAT = {φ : φ is a 3-CNF formula for which ∃ assignment A for which (φ, A) ∈ R}

R = {(φ, A) : A is a sat. assign. for φ}

- satisfying assignment A is a “witness” of the satisfiability of φ (it “certifies” satisfiability of φ)
- R is decidable in poly-time

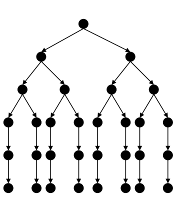
Poly-time verifiers

$$L = \{ x \mid \exists y, |y| \le |x|^k, (x, y) \in R \}$$

Proof: (⇐) give poly-time NTM deciding L

phase 1: “guess” y with $|x|^k$ nondeterministic steps

phase 2: decide if (x, y) ∈ R



Poly-time verifiers

Proof: (⇒) given L ∈ NP, describe L as:

$$L = \{ x \mid \exists y, |y| \le |x|^k, (x, y) \in R \}$$

- L is decided by NTM M running in time n^k
- define the language

  R = { (x, y) : y is an accepting computation history of M on input x}
- check: accepting history has length ≤ $|x|^k$
- check: M accepts x iff ∃y, |y| ≤ $|x|^k$, (x, y) ∈ R

Cook-Levin Theorem

- Gateway to proving lots of natural, important problems NP-complete is:

Theorem (Cook, Levin): 3SAT is NP-complete.

- Recall: 3SAT = {φ : φ is a CNF formula with 3 literals per clause for which there exists a satisfying truth assignment}

Cook-Levin Theorem

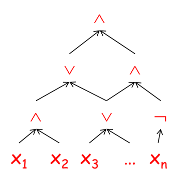
- Proof outline
  - show CIRCUIT-SAT is NP-complete

    CIRCUIT-SAT = {C : C is a Boolean circuit for which there exists a satisfying truth assignment}

  - show 3SAT is NP-complete (reduce from CIRCUIT SAT)

Boolean Circuits

- Boolean circuit C
  - directed acyclic graph
  - nodes: AND (∧); OR (∨); NOT (¬); variables x_i



- C computes function $f:\{0,1\}^n \rightarrow \{0,1\}$ in natural way
  - identify C with function f it computes
- size = # nodes

Boolean Circuits

- every function $f:\{0,1\}^n \rightarrow \{0,1\}$ computable by a circuit of size at most $O(n2^n)$
  - AND of n literals for each x such that f(x) = 1
  - OR of up to 2^n such terms

CIRCUIT-SAT is NP-complete

Theorem: CIRCUIT-SAT is NP-complete

CIRCUIT-SAT = {C : C is a Boolean circuit for which there exists a satisfying truth assignment}

Proof:

- Part 1: need to show CIRCUIT-SAT ∈ NP.
  - can express CIRCUIT-SAT as:

CIRCUIT-SAT = {C : C is a Boolean circuit for which ∃ x such that (C, x) ∈ R}

R = {(C, x) : C is a Boolean circuit and C(x) = 1}

CIRCUIT-SAT is NP-complete

CIRCUIT-SAT = {C : C is a Boolean circuit for which there exists a satisfying truth assignment}

Proof:

- Part 2: for each language A ∈ NP, need to give poly-time reduction from A to CIRCUIT-SAT
- for a given language A ∈ NP, we know

  $A = \{x \mid \exists\, y, |y| \le |x|^k, (x, y) \in R\}$

  and there is a (deterministic) TM M_R that decides R in time $g(n) \le n^c$ for some c.

CIRCUIT-SAT is NP-complete

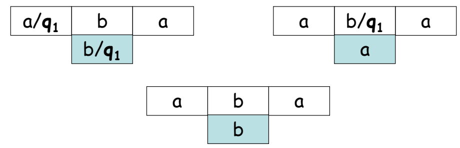
- Tableau (configurations written in an array) for machine M_R on input w = (x, y):

w_1/q_s	w_2	…	w_n	…	_
w_1	w_2/q_1	…	w_n	…	_
w_1/q_1	a	…	w_n	…	_
⋮					⋮
_/q_a	_	…	_	…	_

- height = time taken = $|w|^c$
- width = space used $\le |w|^c$

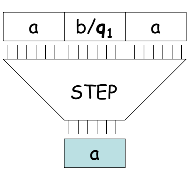
CIRCUIT-SAT is NP-complete

- Important observation: contents of cell in tableau determined by 3 others above it:



CIRCUIT-SAT is NP-complete

- Can build Boolean circuit STEP
  - input (binary encoding of) 3 cells
  - output (binary encoding of) 1 cell



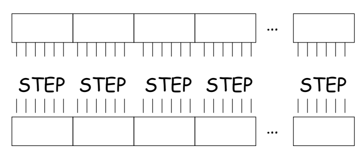
- each output bit is some function of inputs
- can build circuit for each
- size is independent of size of tableau

CIRCUIT-SAT is NP-complete

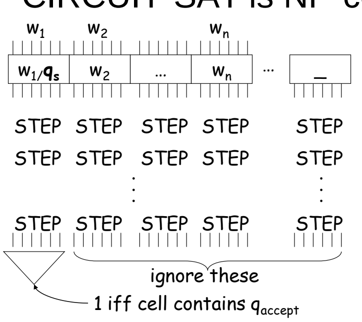
Tableau for M_R on input w = (x, y)

w_1/q_s	w_2	…	w_n	…	_
w_1	w_2/q_1	…	w_n	…	_
		⋮			⋮

- $|w|^c$ copies of STEP compute row i from i-1



CIRCUIT-SAT is NP-complete



This circuit $C_{M_R, w}$ has inputs $w_1w_2…w_n$ and C(w) = 1 iff M_R accepts input w.

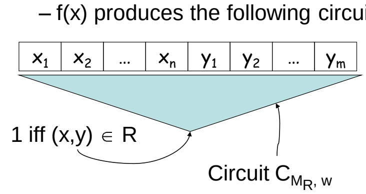
Size = $O(|w|^{2c})$

CIRCUIT-SAT is NP-complete

- recall: we are reducing language A:

  $A = \{ x \mid \exists y, |y| \le |x|^k, (x, y) \in R \}$

  to CIRCUIT-SAT.
- f(x) produces the following circuit:



- hardwire input x
- leave y as variables

CIRCUIT-SAT is NP-complete

– is f(x) poly-time computable?
  - hardcode M_R, k and c
  - circuit has size $O(|w|^{2c})$; $|w| = |(x,y)| \le n + n^k$
  - each component easy to describe efficiently from description of M_R

– YES maps to YES?
  - $x \in A \Rightarrow \exists\, y$, M_R accepts $(x, y) \Rightarrow f(x) \in$ CIRCUIT-SAT

– NO maps to NO?
  - $x \notin A \Rightarrow \forall\, y$, M_R rejects $(x, y) \Rightarrow f(x) \notin$ CIRCUIT-SAT