Wednesday, April 11, 2018 8:22 AM

Why are the two definitions the same?

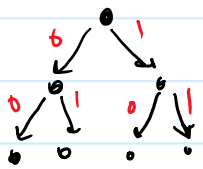
Theorem $L \in NP$ iff it is expressible as:

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

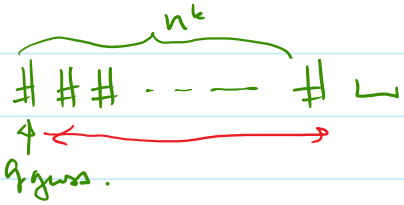
for some poly-time R.

Proof $\Leftarrow$ Show NTM that decides L:

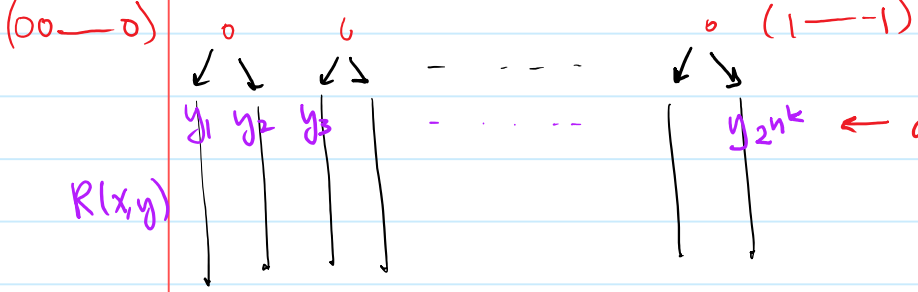
1. Compute $|x|^k$ by marking off $|x|^k$ tape symbols.
2. "Guess" a string y of length $|x|^k$ & write it on the tape.
3. Run R on (x,y) & accept iff R accepts.



→ write this symbol on the tape that holds y.



$\Delta(\#, q_{guess}) = \{(0, q_{guess}, R), (1, q_{guess}, R)\}$



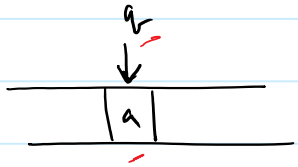
Wednesday, April 11, 2018 8:45 AM

$\Rightarrow$ $\exists$ NTM that decides L in time n^k.

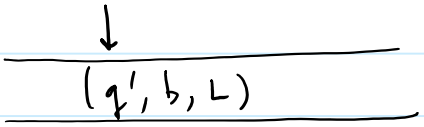
Construct a string y that consists of the non-deterministic choices at each step.

$y = [(q,a,L), (q',b,R), (\bar{q},a,-), \ldots]$

R(x,y): Simulate M on x & verify that each triple is a valid choice given the current state of M & current symbol.



Simulation of M on x



y

R checks that $(q',b,L) \in \delta(q,a)$.

If so accept & continue.

O.w. reject.

If M halts then R halts (acc/rej depending on M).

An accepting computation of M will correspond to some y which causes R to accept (x,y).

If all paths in M reject, there will be no y that can cause R to accept. //

Why NP? There is a huge number of problems that are complete for NP.

Why not EXP? Too strong. Important problems are not complete for EXP.

Thursday, April 12, 2018 7:59 AM

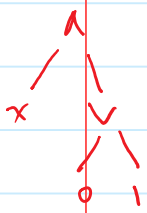
Central question in CS: $P \stackrel{?}{=} NP$

finding a solution vs. verifying a solution.

NP-Completeness

Circuit SAT: Given a Boolean circuit with variables, is there an assignment to the variables that makes the output = 1?

Theorem: Circuit-SAT is NP-complete.



Circuit-SAT $\in$ NP: Guess an assignment for the variables & check if the circuit is satisfied.

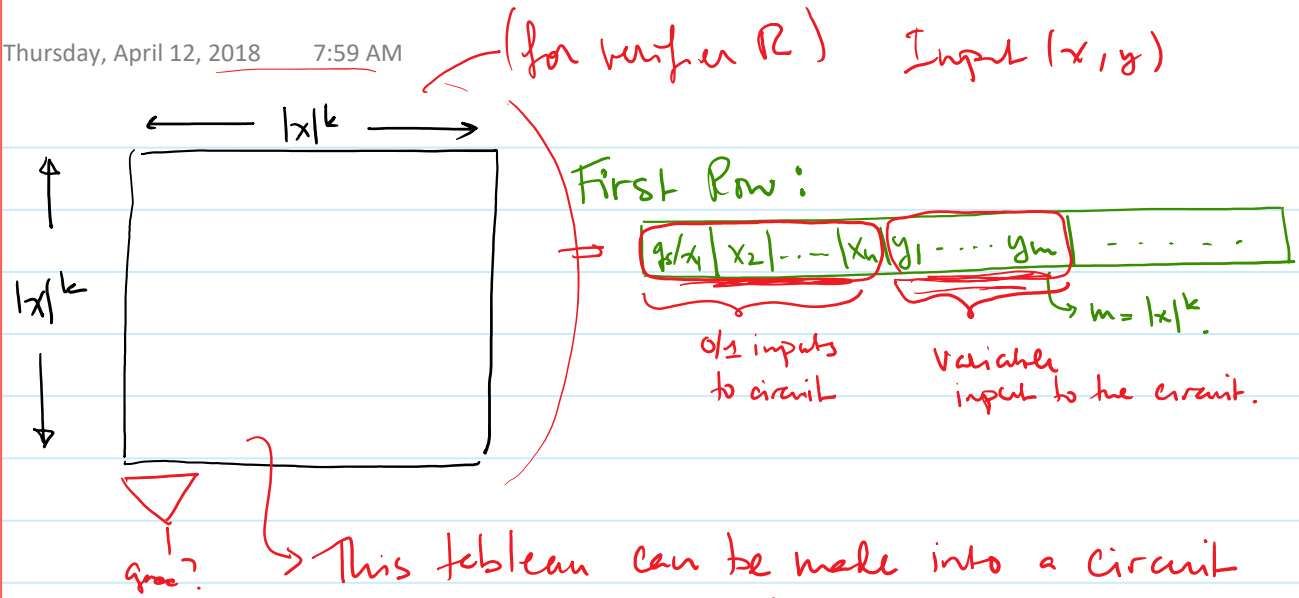
$\forall$ $L \in NP$ $L \alpha$ Circuit-SAT

↳ $\exists$ k & poly time R

$L = \{x \mid \exists y \ |y| \leq |x|^k, R(x,y) \text{ accepts}\}$

Make a tableau (as in previous proofs) that corresponds to the computation of R on input (x,y)

Thursday, April 12, 2018 7:59 AM



Reduction: Given x, output C_x

only difference w/ P-completeness proof are the input variables $y_1 \cdots y_m$.

$x \in L \iff$ If $\exists$ values that can be assigned to $y_1 \cdots y_m$ that makes $C_x = 1$. //

Witness version of NEXP:

Theorem: $L \in NEXP$ if & only if $\exists$ k & polytime R s.t.

$L = \{x \mid \exists y \ |y| \leq 2^{|x|^k}$ R accepts $(x,y)\}$.

R is poly time in $|x| + |y|$ because $|y|$ is already exp long in $|x|$.

Proof similar to proof for NP.

Friday, April 13, 2018 8:00 AM

SUCCINCT Ckt SAT

Succinctly encode Boolean circuit with n non-variable inputs and m variable inputs.

Is there a setting of the circuit that evaluates to 1?

6 node/gate types: $\wedge$, $\vee$ (in-degree = 2)

$\neg$ (in degree = 1)

0 1 var (in-degree = 0).

Theorem: SUCCINCT Ckt SAT is NEXP-complete.

The proof uses the same ideas as the proof that SUCCINCT-CVAL is EXP-complete.

The tableau is a record of R's computation on input $X_1 \ldots X_n$ with variables $y_1 \ldots y_m$

$m = 2^{n^k}$ and the size of the ckt is poly(n, m).

Friday, April 13, 2018 8:00 AM

Complement Classes

If C is a complexity class $co\text{-}C$ is the class containing complements of the languages in C.

$$L \in C \iff co\text{-}L \in co\text{-}C$$

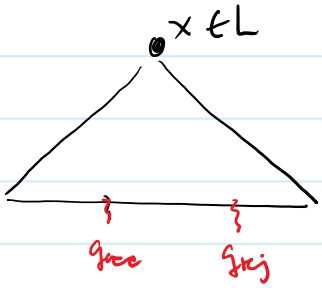
Note that $co\text{-}L$ is not quite $(\Sigma^* - L)$ because invalid encodings are always excluded.

L = Graphs that have a Hamiltonian cycle
$co\text{-}L$ = Graphs with no Hamiltonian cycle.
(both languages exclude strings that are not valid encodings of graphs.)

Some classes are closed under complement:

$$co\text{-}P = P.$$

What about $co\text{-}NP$



Can't just exchange q_{acc} + q_{rej}.
(Computations w/ q_{acc} + q_{rej} at the leaves would be accepting in both cases.)

An NTM w/ q_{acc} + q_{rej} swapped would still accept x.

$P = NP$ implies $NP = co\text{-}NP$
Believed that $NP \neq co\text{-}NP$

NP - languages w/ short proofs.
$co\text{-}NP$ - unlikely to have short proofs.
↳ all of them are unsatisfiable

co-NP - unlikely to have short proofs.
↳ all ϕ that are unsatisfiable.

Believed that
$NP \neq co\text{-}NP$

Friday, April 13, 2018 8:00 AM

Non-Deterministic Time Hierarchy Theorem

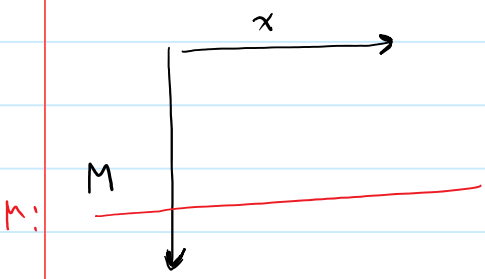
If f & g are proper functions and $f(n+1)$ is $o(g(n))$ then $NTIME(f(n)) \subsetneq NTIME(g(n))$

Will only show $NTIME(t(n)) \subsetneq NTIME(t(n)^{1.1})$

Will assume the existence of an NTM called NSIM On input $\langle M, x \rangle$ (M is nondeterministic) NSIM simulates M on x. If M runs in time $t(n)$ then NSIM runs in time $t(n)^{1.1}$

Can't just naively diagonalize. It's not clear how to flip the output of an NTM, (because of the asymmetry in the acceptance criteria)

Will use a technique called "lazy diagonalization"



Note that when we build the table it's not essential to flip the diagonal. We just need to find a language that differs from every row somewhere.

Each row of the table corresponds to a string. 3 possibilities.

- String not a valid encoding
- String encodes an NTM that does not halt in time $t(n)$.
- String encodes an NTM that halts in time $t(n)$.

Friday, April 13, 2018 8:30 AM

Here is what D does on input x

If $x \notin 1^*$ reject.

If $x = 1^n$, compute i s.t. $f(i) < n \leq f(i+1)$.

can be done in time $O(n)$

If i is not a valid encoding of a TM $\Rightarrow$ reject.

O.w let M_i be the TM encoded by i.

If $n < f(i+1)$ use NSIM to simulate M_i on 1^{n+1} for $t(n+1)$ steps.

If M_i does not finish $\rightarrow$ accept.

Otherwise output whatever M_i does Time $t(n+1)^{1.1}$

If M_i runs in time $t(n)$ then $1^n \in L(D) \Leftrightarrow 1^{n+1} \in L(M_i)$

If $n = f(i+1)$ Deterministically (brute force)

Simulate M_i on input $1^{f(i)+1}$ for $t(f(i)+1)$ steps

D accepts on input $1^{f(i+1)}$ iff M_i rejects $1^{f(i)+1}$

Time: $2^{t(f(i)+1)}$ steps of M_i

with simulation overhead $[2^{t(f(i)+1)}]^{1.1} \leq f(i+1) = n$

linear time!

Friday, April 13, 2018 8:00 AM

$D(L) \in NTIME(t(n)^{1.1})$

$D(L) \notin NTIME(t(n))$

We will construct a language L computable by NTM D that runs in time $t(n)^{1.1}$ such that for every NTM M that runs in time $t(n)$, $\exists$ z s.t.

$$D(z) \neq M(z).$$

↳ z's don't have to be consecutive.

In fact will only use z's in 1^*

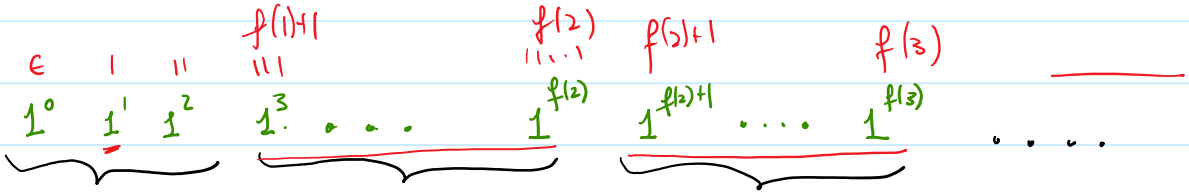
M_i = i^{th} string in lexicographic order interpreted as a TM.

t - linear $t(n) = n$

Define: $f(1) = 2$.

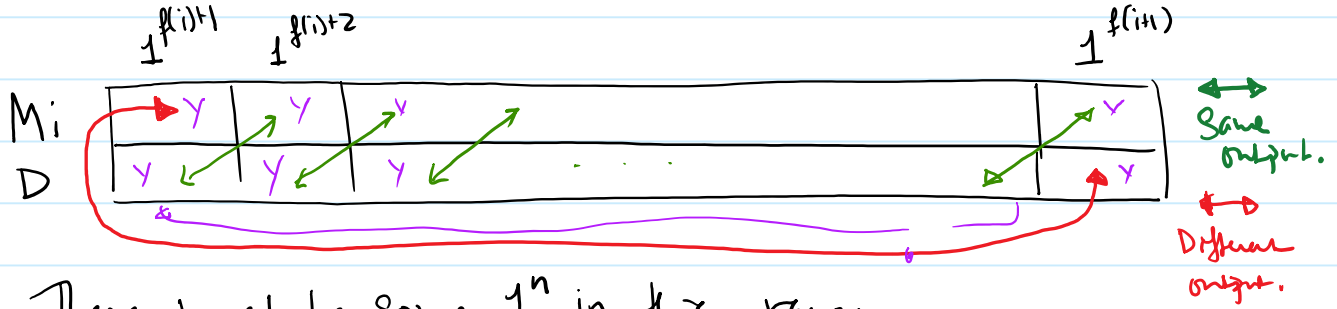
$$f(i+1) = \left[2^{t(f(i)+1)}\right]^{1.1}$$

$$f(i+1) = 2^{f(i)\cdot 1.1}$$



If M_i is an NTM that runs in time $t(n)$ then $L(D)$ and $L(M_i)$ will differ somewhere in the range

$$1^{f(i)+1} \quad 1^{f(i)+2} \quad \ldots \quad 1^{f(i+1)}$$



There must be some 1^n in this range on which they disagree.