Non-determinism.

Note Title 4/13/2013

Can computers replace mathematicians?

$$L = \{ (x, 1^k) \mid \text{Statement } x \text{ has a proof of length} \le k \}$$

A yes answer to this question is among the many profound implications that would result if P = NP.
The class NP is an abbreviation of
Non-deterministic Polynomial time.
The class NP will initially be defined in terms of non-deterministic models of computation, but we shall see that it is equivalent to the witness version that we have seen earlier.

This lecture will cover

- Non-deterministic models of computation
- Non-deterministic time classes
- NP, NP-completeness
- co-NP
- NTIME Hierarchy
- Ladner's theorem.

Recall the definition of non-deterministic Turing Machines:

Q , Σ q_{start}, q_{acc}, q_{rej}

$$\delta : Q \times \Sigma \longrightarrow Q \times \Sigma \times \{L, R, -\}$$

In a non-deterministic TM we have Δ instead of δ

$$\Delta \subseteq \underbrace{(Q \times \Sigma)}_{\text{input}} \times \underbrace{(Q \times \Sigma \times \{L, R, -\})}_{\text{output}}$$

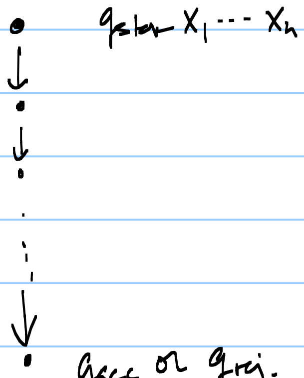
Given the current state and symbol, there may be more than one (or no) choices for what to do next.

In deterministic computation: given a config of the TM, there is a unique next configuration.

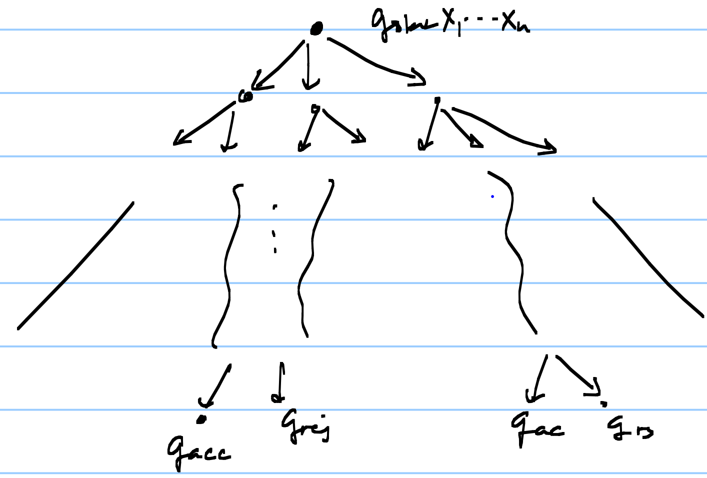
In a Non-deterministic computation: there may be several possibilities for the next configuration.

For a given TM M & input x, we can build a configuration graph (nodes = TM configurations. $(c_1, c_2) \in E$ if c_2 is reachable from c_1 in one step).

Deterministic TM



Non-deterministic TM



In order for an NTM to accept or reject x:

All computation paths must terminate.

Running time: length of the longest root to leaf path.

Space: max # tape cells used on any path from the root to a leaf.

Asymmetric definition $\begin{cases} \text{Accept if } \underline{\text{any}} \text{ leaf is an Accept.} \\ \text{Reject if } \underline{\text{all}} \text{ leaves are Reject.} \end{cases}$

NTIME$(f(n))$: languages decidable by a multi-tape NTM that runs in at most $f(n)$ steps along any computation path where n is the length of the input.

NSPACE$(f(n))$: languages decidable by a multi-tape NTM that touches $\le f(n)$ cells of work tape along any computation path.

Time classes:
$$NP = \bigcup_{k \ge 1} \text{NTIME}(n^k)$$
$$NEXP = \bigcup_{k \ge 1} \text{NTIME}(2^{n^k})$$

Useful alternative view of NP:

Is there a way to prove that $x \in L$ with a poly-sized proof y that can be verified by a poly-time verifier?

$L \in NP \iff \exists$ poly time verifier R and constant k s.t. $x \in L$ iff $\exists y$ $|y| \le |x|^k$ and R accepts (x,y)

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

Examples: 3SAT $= \{ \phi \mid \phi$ is a 3-CNF formula for which $\exists$ assignment A $(\phi, A) \in L(R) \}$

$L(R) = \{ (\phi, A) \mid A$ is a satisfying assignment for $\phi \}$

A is a witness that $\phi \in 3SAT$
R is a poly-time TM.

Other examples: Hamiltonian Path, etc.

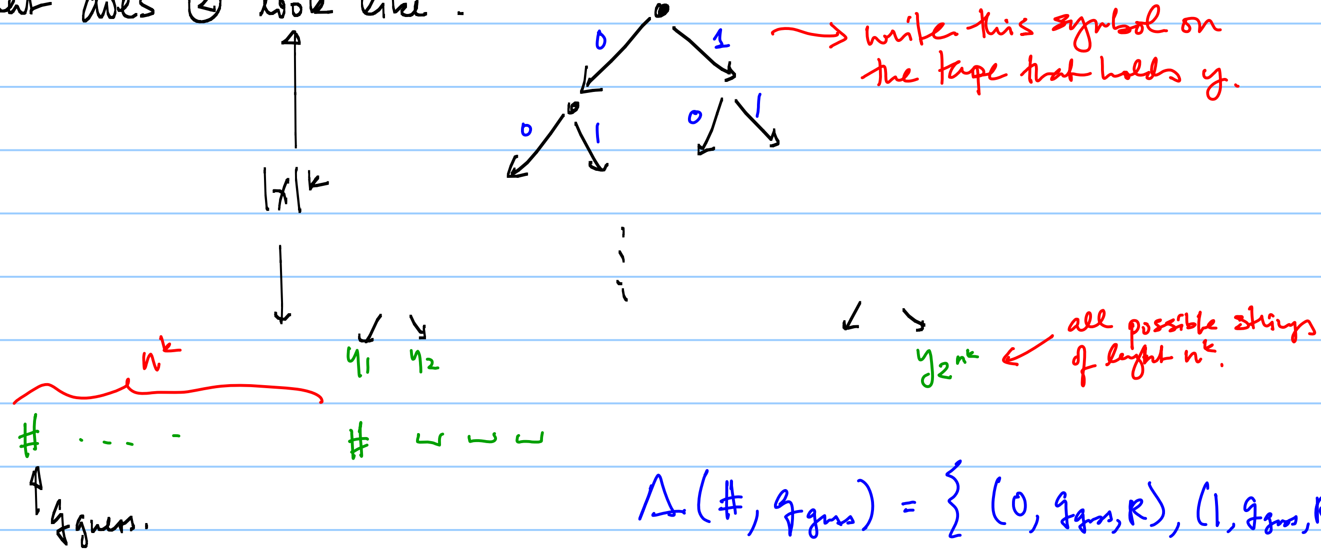
Why are these two definitions the same?

Theorem: $L \in NP$ iff it is expressable as:
$L \in \{ x \mid \exists y \; |y| \le |x|^k \;\; (x,y) \in R \}$
for some poly-time TM R.

Proof: $\Leftarrow$ Show NTM that decides L.

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

What does ② look like?



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

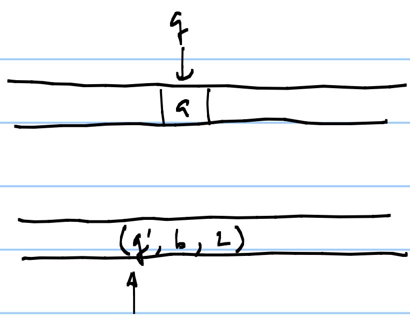
$\Rightarrow \exists$ NTM M that decides L in time n^k

Construct a string y consisting of the non-deterministic choices @ each step.

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

$|x|^k$ triples

$R(x, y)$: simulates M on x & verifies that each triple is a valid choice given current state of M & symbol on the tape



Simulation of M on x

y

R checks if $[(q, a) \times (q', b, L)] \in \Delta_M$
If so execute step & continue
If not, reject.

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

An accepting path of 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 are a huge number of problems that are complete for NP.

Why not EXP? too strong
important problems are not complete for EXP.

Central question in computer science: $P \overset{?}{=} NP$

finding a solution vs. verifying a solution.

NP-completeness:

Circuit SAT: Given a boolean ckt with variables, is there an assignment to the variables that makes it output 1?

Theorem: Circuit-SAT is NP-complete.

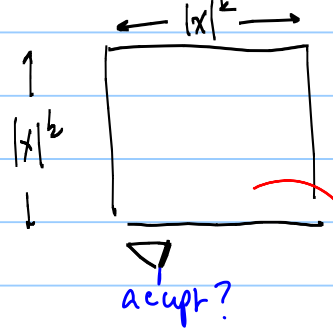
Circuit SAT $\in NP$: guess assignment for the variables and check if circuit is satisfied.

$\forall \; L \in NP \quad L \propto$ Circuit SAT

$\exists \; k$ + Poly-time R:

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

Recall the tableau from last lecture corresponding to the computation of R:



First row of tableau:

$$\underbrace{q_s/x_1 \mid \cdots \mid x_n}_{0/1 \text{ inputs.}} \mid \underbrace{y_1 \cdots\cdots y_m}_{\text{variable inputs to circuit}} \mid \cdots \qquad m = |x|^k$$

as discussed before, this can be made into a circuit.

Reduction: given x output C_x.

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

Witness version for NEXP: (or "proof system" version)

Theorem

$L \in$ NEXP if and only if $\exists k$ & poly time R s.t.

$$\{x \mid \exists y \quad |y| \le 2^{|x|^k} \quad R \text{ accepts } (x,y)\}$$

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

Pf of theorem similar to the proof for NP.

SUCCINT Ckt SAT

Succinctly encoded Boolean circuit with n non-variable inputs & m variable inputs

Is there a setting of the variables that causes the circuit to evaluate to 1?

Theorem SUCCINCT Ckt SAT is NEXP-complete.

Pf uses same ideas as the proof that SUCCINT-CVAL is EXP-complete.

the tableau is a record of the verifier's (R's) computation on input $x = x_1 \cdots x_n$ with variables $y_1, \ldots, y_m$. In this case $m = 2^{n^k}$ and the size of the circuit is poly(n, m).

Complement Classes

If C is a complexity class then co-C is the class containing complements of languages in C.

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

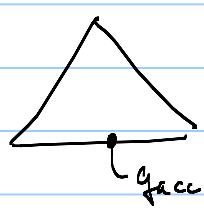
Note that co-L is not quite $(\Sigma^* - L)$ because invalid encodings are always excluded from the language.

L = graphs w/ a Hamiltonian cycle
co-L = graphs w/ no Hamiltonian cycle
(both languages exclude strings which are not valid encodings of graphs).

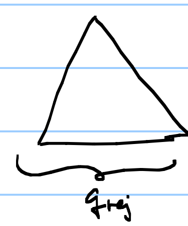
Some classes are closed under complement:
co-P = P.

What about co-NP? We can't just exchange q_{acc} & q_{rej}.

NP: $x \in L$



$x \notin L$



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

P = NP implies NP = co-NP

Believed that NP ≠ co-NP.

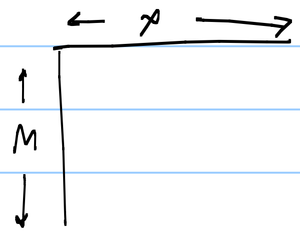
Non-deterministic Time Hierarchy Theorem: If f & g are time constructible (proper) and $f(n+1)$ is $o(g(n))$ then $NTIME(f(n)) \subsetneq NTIME(g(n))$

We will only show: $NTIME(t(n)) \subsetneq NTIME(t(n)^{1.1})$

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

We 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).

We will use a technique called "lazy diagonalization". Note that when we build this tableau:



it's not essential to flip the diagonal. We just need to find a language which differs from every row somewhere.

Each row in the table corresponds to a string. Some strings will not encode a TM at all. Other strings will encode a TM that does not halt in time $t(n)$. Some will encode $t(n)$-time NTM's.

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

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

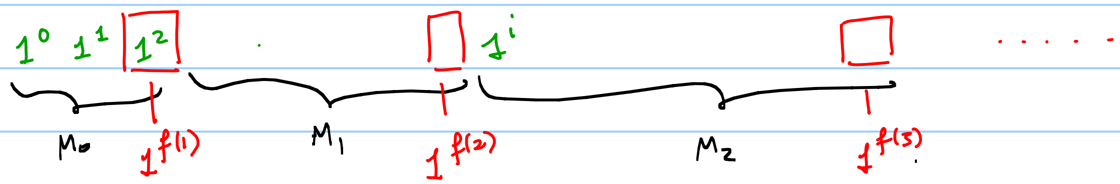
(z's don't have to be consecutive).

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

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

Define $f(1) = 2$

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



If M_i is an NTM which runs in time $t(n)$, then the language L (i.e. $D(L)$) will differ from $L(M_i)$ somewhere in the range $1^{f(i)+1}, 1^{f(i)+2}, \ldots, 1^{f(i+1)}$.

Here's what D does on input x:

If $x \notin 1^*$ then reject.

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

If $\langle i \rangle$ is not a valid encoding of a TM ⇒ reject.

O.W let M_i be the NTM encoded by string $\langle i \rangle$. — $\langle i \rangle$ = binary encoding of int i.

If $f(i) < n < f(i+1)$, use NSIM to simulate M_i on 1^{n+1} for $t(n+1)$ steps. If M_i does not finish ⇒ accept. Otherwise output whatever M_i does. Takes time $t(n+1)^{1.1}$.

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

If $n = f(i+1)$

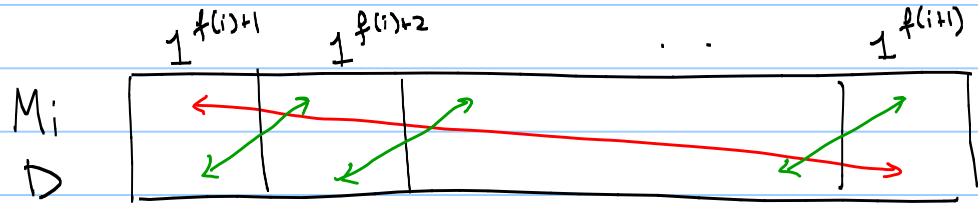
Deterministically (brute force) simulate M_i on input $1^{f(i)+1}$

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

$\to$ assuming branching factor ≤ 2.

This takes $2^{t(f(i)+1)}$ steps of M_i. With simulation overhead $\left[2^{t(f(i)+1)}\right]^{1.1} \le 2^{t(f(i))^{1.1}} \le f(i+1) = n$

linear time!



$\leftrightarrow$ means same output

$\leftrightarrow$ means different output.

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

//