(5)

closed under union

The basic idea of closure under union is ~~that~~ you have two languages that belong to NP

$L_1 \in NP$ and $L_2 \in NP$

and three NDA TMs: M_1, M_2 and M_3

M_3 will accept "input w" as long as M_1 accepts w (otherwise reject), M_2 accepts w (otherwise reject) or they both do.

* this is essentially the same as when we're proving P is closed under union, but whether to run M_1 or M_2 is decided nondeterministically.

(5) cont.

closed under concatenation:

like before we have $L_1 \in NP$ and $N_2 \in NP$ and M_1, M_2 and M_3, but this time ~~we~~ for input w, it gets split up and M_3 checks if M_1 accepts w_1 and if M_2 accepts w_2. Otherwise, reject.

// Sidenote this is done polynomially because for length, n in w, stage 2 will only take $n+1$ time

"running M_1 on w_1 takes $O(n^{k_1})$ time, and running M_2 on w_2 takes $O(n^{k_2})$ time, so stage 2 runs in time $O(n^{k_1}) + O(n^{k_2}) = O(n^{\max(k_1,k_2)})$ which is polynomial in n"

$k^1 + k^2 \rightarrow$ constants

*Again, essentially the same for when we prove P is closed under concatenation except ~~that~~ w gets split up nondeterministically and M_1 and M_2 each get ran once.

6

NP closed under kleene star

We have an NTM, M ~~that~~ first checks whether our input w is apart of ε. If it is accept.

If it doesn't accept, w will have to be split up into a range from 1-|w|. "$1 \le m \le |w|$, where m is how many ways it gets split up" (people.cs.aau.dk)

M goes through each nondeterministic piece of w and checks if it's in M. If it is, accept. Otherwise, reject that piece

*we see ~~that~~ this is done in a polynomial amount of time, because M only runs |w| amount of times

(6) cont

P closed under Kleene star

We have a TM decider M ~~that~~ checks our input w and if $w \in L$ accept.

We initialize a table (i, j) where $T =$ true if $w_{i,j} \in A^*$ The length goes up from 1 to n and the decider will go through all strings of w from the range of $1 \leq i < j \leq n$

"M = On input $w = w_1 w_2 \ldots w_n$

1. If w is the empty string, accept
2. Initialize $T[i,j] = 0$ for $1 \leq i \leq j \leq n$
3. For $i = 1$ to n,
4. Set $T[i,i] = 1$ if w_i is in A
5. For $l = 2$ to n,
6. For $i = 1$ to $n - l + 1$,
7. Let $j = i + l - 1$,
8. If $w_i \ldots w_j$ is in A, set $T[i,j] = 1$
9. For $k = i$ to $j - 1$,
10. If $T[i,k] = 1$ and $T[k,j] = 1$, set $T[i,j] = 1$
11. Accept if $T[1,n] = 1$, otherwise reject"

cs.princeton.edu

Time spent: i spent about three days on this one

Students I worked with: no one, but i did consult my friend on how to approach **question 2**

Sources:

Q2:

Textbook, https://mathworld.wolfram.com/PolynomialTime.html

I wasn't sure how to do this question so I asked my friend how they started it and they **said "all** we have to do here is explain 7.14 theorem in the book." so that's what I did

Q3:

Textbook, slides,
http://cs.jhu.edu/~cs363/fall2013/assign9_sln.pdf

https://people.cs.umass.edu/~barring/cs601sum03/hw/4sol.html

Q4:

Q5:

http://www.public.asu.edu/~ccolbou/src/555hw4s16sol.pdf ebiba/Do...

https://web.njit.edu/~marvin/cs341/hw/hwsoln11.pdf

http://ais.informatik.uni-freiburg.de/teaching/ss15/bridging/exercise/solutions/exercise09.pdf

Q6:

P:

http://www.public.asu.edu/~ccolbou/src/555hw4s16sol.pdf

https://www.cs.umd.edu/~gasarch/COURSES/452/F14/poly.pdf

https://www.cs.princeton.edu/courses/archive/fall03/cs487/hw7sol.pdf

NP:

http://people.cs.aau.dk/~srba/courses/tutorials-CC-10/t13-sol.pdf

file:///Users/jazminebiba/Downloads/section10_sols.pdf

section10_sols.pdf

③

So we have two graphs : F and G. We have a pair of vertices (x, y) ~~that~~ should appear as edges in both F and G.

We have two loops that do this by first looking up $f(x)$, $f(y)$ and then seeing if they're both identical edges. We're checking the n^2 pair of edges, so that is the final and polynomial time.

"we accept" iff each of the pairs satisfies the condition ~~that~~ (x, y) and $(f(x), f(y))$ are either both edges of non-edges.

(2)

(2, 3, ...)

theorem 7.14 so two stages are run in "m" times so n^2

M = "On input ⟨G, s, t⟩, where G is a directed with nodes s and t:

1. ~~[illegible]~~ Place a mark on node s (constant)

"runs at most m times" 2. Repeat the following until no additional nodes are marked:

3. Scan all the edges of G. If an edge (a, b) is found going from a marked node a to an unmarked node b, mark node b.

4. If t is ~~[illegible]~~ marked accept. Otherwise, reject. (constant)

"We see that stages 1 and 4 are executed only once. Stage 3 runs at most m times because each time except the last it marks an additional node in G."

So total is: 1 + 1 + m . polynomial size G

M is a polynomial time algorithm for PATH.

"An algorithm is said to be soluable in polynomial time if the number of steps required to complete the algorithm for a given input is $O(n^k)$. * where n is the complexity of the input"

mathworld.wolfram