CS 332 – Spring 2021, Assignment 3

Colaborators: None

Answer 1

(a) A language where strings can have any number of ”a”, ”b”, or ”c” characters, yet it must be made up of just a single type of characters (ie: if it’s ”a” it can’t have ”b” or ”c”, and vice versa, but it can have whatever amount of that character).

(b) A Language where strings star and end with a ”1”, and have a number of 0’s in the middle that is divisible by 3.

(c) A Language where strings start with ”a” and end with ”b”, and have any amount of repetition of the string ”ba” in the middle.

(d) An empty language.

(e) A language with only the empty string, or an empty language.

Answer 3

(a) $NFA.union(N_1, N_2)$ has $s_1 + s_2 + 1$ states; $NFA.concatenate(N_1, N_2)$ has $s_1 + s_2$ states; $NFA.star(N_1)$ has $s_1 + 1$ states.

(b) 1 state.

(c)

Base Case $k = 1$:

If R has size $k = 1$, then R either contains a single symbol a or is made up of ϵ or $\emptyset$, which means that $S(k) = 1$, which is $1 \leq 2$, so our theory holds.

Assuming that $S(k) \leq 2k$ then for $k + 1$ we have:

The size of R is increased by 1, this means that something was added to R, if it was a single symbol (a, ϵ, $\emptyset$), then $S(k + 1) = 2k + 1$ which is less than $2(k + 1) = 2k + 2$.

If the symbol added was $*$, then we have that $RegexToNFA(R) = R = R_1^* = NFA.star(R_1)$, here we know that the size of R_1 is the size of R, minus one (for the * symbol), so we still have that $S(k + 1) = S(k) \leq 2k$, thus our theory holds.

If the symbol added was a $\circ$ or a $\cup$, then we know that $RegexToNFA(R) = R = R_1 \cup R_2 = NFA.union(R_1, R_2)$ or $RegexToNFA(R) = R = R_1 \circ R_2 = NFA.concatenate(R_1, R_2)$, in both cases we know that the size of R_1 plus the size of R_2, must be equal to the size of R minus 1 (for the symbol in question $\circ$ or $\cup$), thus the size of $RegexToNFA(R_1)$ plus the size of $RegexToNFA(R_2)$ must be equal to $S(k)$ which is smaller than $2k$, and if we add the other symbol we'll get at most $2k + 1$, thus proving our theory.

(d)

Base case $i = 1$:

Assuming that N has a single symbol labeling each transition, that means that when we rip out the absolute first state, and re-write the transitions, we are, at most, adding a single symbol to any specific symbol, in addition we might have to add an operator ($\cup$, $\circ$, $*$), which means that for $l(i) \leq 4^2 - 3 = 13$. Since we know we have added a single symbol and even if we have to add all the 3 operators, we will have a transition with +4 length, and since we know that at start the transitions only had a single symbol, then we have a max length $l(1) = 5 \leq 13$.

Assuming $l(i) \leq 4^{i+1} - 3$ then for $i + 1$:

We know that the last biggest expression would be $4^{i+1} - 3$, so even if we join two expressions of that size, we will add at most $2 \times (4^{i+1} - 3) + 1$, which is less than $4^{i+2} - 3$, thus this is still true.

(e) We know that the final return value of $NFAtoRegex(N)$ will have an expression of size $4^{k-2+1} - 3$, where k is the number of states in the GNFA made from the NFA N. Thus we know that the biggest expression will have size $4^{k-1} - 3$, however, we know that to make an GNFA from an NFA, we need to add two states, so we know that $s = k - 2 \equiv s + 1 = k - 1$, so we have that the final expression, will have size $4^{s+1} - 3$, which is smaller than 4^{s+1}. Thus we have that the size of the regular expression from $NFAtoRegex(N)$ will be at most 4^{s+1}.

Answer 4

(a) We have 6 different pairs (00 01, 00 10, 00 11, 01 10, 10 11), for each of these, we can find a string z such that exactly one of xz, yz is in REP_2, simply if we pick $z = x$ or $z = y$, for each of the pairs listed above we can pick z to be 00, 00, 00, 01, 01, 10, and we see that xz is in REP_2 and yz isn't (I'm assuming that the first string of the pair listed is x and the second is y).

(b) As we saw in class, if a set of strings S is pairwise distinguishable in a language L, then the language must have at least $\|S\|$ states. As we saw above, S is pairwise distinguishable in REP_2, since for every pair x, y, there is a string z so that if xz is in REP_2 yz isn't. Thus, knowing that S is pairwise distinguishable in REP_2, we have that by the pigeonhole principle (and as seen in class), if REP_2 has less that $\|S\|$ states, then there are $x, y \in S$ so that the DFA M representing REP_2, ends up on the same state with x and y.

Thus, and again, as we saw in class, the DFA representing REP_2 must have at least $\|S\|$ states.

(c) Since we have established before that if any set of strings S is pairwise distinguishable in a language L, then the language must have at least $\|S\|$ states, then we know that REP_k will have $\|S_k\|$ states, now we just need to establish how bit S_k is.

Since we know that our alphabet has two symbols 0 and 1, then we know that there are 2^k combinations of the symbols with length k, thus, S_k will have 2^k size. This is because, the biggest S that is pairwise distinguishable in REP_k will include every possible combination of the symbols with length k, thus REP_k needs to accommodate this S_k, so it needs 2^k states.

(d) We know that, by the subset method, any NFA using the alphabet $\{0, 1\}$, can be transformed into a DFA with 2^k states (as proven in class and on the textbook).

This means, by the corollary, that any DFA with 2^k states has an equivalent NFA with k states, this means that, since we have already established that the DFA for REP_k has 2^k states, then there must exist an NFA for REP_k with k states.

Answer 5

(a) If we assume a set $S = \{0^n \| n \geq 0\}$, for any pair x and y in S there is a way of finding an string z so that either xz or yz are in L_1 but the other one isn't. However, as we know, the minimum number of states an NFA has is the size of the largest pairwise set we can find, however this pairwise set S is infinite, so there is no way to upper bound the number of states in the NFA built by L_1.

(b) Again if we assume a set $S = \{0^n 1^n \| n \geq 0\}$, for any pair of $x = 0^n 1^n$ and $y = 0^m 1^m$ I can just pick 0^n and xz won't be in L_2 while yz will, thus we have a pairwise set. However, and again, S is infinite, thus we cannot upper bound the number of states in L_2.

(c) If we assume $S = \{0^i 1^j \| i, j \geq 0\}$, then for any pair $x = 0^a 1^b$ and $y = 0^c 1^d$ we know that either $a \neq c$ or $b \neq d$, so we can always pick $z = xx$ or $z = yy$ and one string would be in L_3 while the other won't, making this a pairwise set. Again the set is infinite, so we cannot upper bound the number of states in L_3, thus making it non-regular.

(d) If we assume $S = \{0^i 1^j \| i, j \geq 0\}$, then for any pair $x = 0^a 1^b$ and $y = 0^c 1^d$, there is some string $z = /1/w$ where w is the binary representation of $x + 1$, so that xz is in L_4, and yz isn't. Thus making this a pairwise set, and again, since it's an infinite set, we cannot upper bound the states of L_4, thus making it non-regular.