129. EquivPractice

1) A = {w | w contains an occurrence of abb}

a. *ab* and *ba* are ≡A inequivalent because a specific word *z* can be chosen such that *abz* ∈ A and *baz* ∉ A. *z = b*. abb ∈ A and *bab* ∉ A.

b. *λ* and *abb* are ≡A inequivalent because a specific word *z* can be chosen such that λz ∉ A and *abbz* ∈ A. *z* = *λ*, λλ∉ A and *abbλ* ∈ A.

c. *λ* and *ba* are ≡A inequivalent because a specific word *z* can be chosen such that λz ∉ A and *baz* ∈ A. *z* = *bb*, λbb∉ A and *babb* ∈ A.

d. *abb* and *babba* are ≡A equivalent because no specific word *z* can be chosen such that abbz ∈ A and *babba* ∉ A or vice versa. This is true because of the nature of the language. Nothing can be concatenated to either string to remove the occurrence of *abb* from the string. Because both strings are in the language to begin with, they cannot be taken out. This is not true for all languages but is true for this one.

2) B = {w | |w| is even}

a. *aab* and *ab* and ≡B inequivalent because a specific word *z* can be chosen such that *aabz* ∈ B and *abz* ∉ B. A *z* can also be chosen such that *abz* ∈ B and *aabz* ∉ B. For the first statement we let *z* = b. *aabb* ∈ B and *abb* ∉ B. For the second let *z* = λ. *aabλ* ∉ B and *abλ* ∈ B, this is true because concatenating the empty string is the same as not concatenating anything at all.

b. *λ* and *a* are ≡B inequivalent because a specific word *z* can be chosen such that *λz* ∉ B and *az* ∈ B. Let *z* = *a*. *λa* ∉ B because *λa* is equal to a and *aa* ∈ B.

3) C = {aibj | I < j}

a. *ab* and *ba* are ≡c inequivalent because a specific word *z* can be chosen such that *abz* ∈ C and *baz* ∉ C. Let *z* = *b*. *abb* ∈ C and *bab* ∉ C.

b. *λ* and *abb* are ≡c inequivalent because a specific word *z* can be chosen such that *λz* ∉ C and *abbz* ∈ C. Let *z* = *λ. λλ* ∉ C and *abbλ* ∈ C. This is true because in the string λλ i = j = 0, and in *abb* i = 1 < 2 = j.

c. *bba* and *ba* are ≡c equivalent because there is no such word *z* that can be concatenated such that *bbaz* ∈ C and *ba* ∉ C. This is due to the nature of the language. In c there can never be an occurrence of *a* after and occurrence of *b*. In other words, if there is an occurrence of *b* the only symbol that can be read is another *b****.*** No word *z* can be added to either string to change the fact that an *a* appears after a *b.*

145. NonRegPractice

a) A = {anb2n| n ≥ 0}

* Consider the infinite collection of strings {an | n ≥ 0}. Each of these strings is inequivalent to the other with respect to A. Since: for i ≠ j the strings ai and aj are separated by the string *z* = b2i A is nonregular

b) B = {anbmcn|n,m ≥ 0}

* Consider the infinite collection of strings {akbp | k,p ≥ 0}. Each of these strings is inequivalent to the other with respect to B. Since: for i ≠ j the strings aibj and ajbi are separated by the string z = ci or z = cj B is nonregular

e) E = {w ϵ {a,b}\* | ∃x ϵ {a,b}\*, xx=w}

* Consider the infinite collection of strings aa,bb,aaaa,bbbb ,abab,baba,abbabb,abaaba…. To show that they are inequivalent, with i % 2 = 0 and j % 2 =0, consider any aibjai and ajbiaj we can use the word z = bj. Because E has infinitely many equivalence classes it is nonregular.

f) R = {w ϵ {a,b}\* | ∃x ϵ {a,b}\*, xxR=w}

* Consider the infinite collection of strings b,ab,aab,…,aib,… Each string is inequivalent from the others. For any aib and ajb we can use the word z = bai to separate them. Since R has infinitely many equivalence classes it is nonregular.

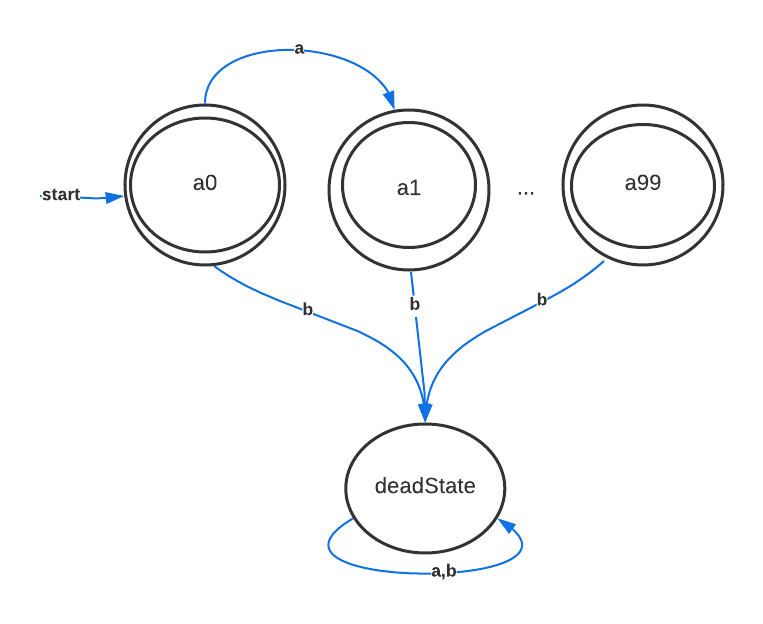
g) The set of strings of a*s* and b*s* whose length is a perfect square.

* Consider the collection of infinite strings, λ,aaaa,bbbb,aaaaaaaaa,bbbbbbbbb…. ,. Each of these strings is inequivalent from the other. Let x = and y=, n < m. Let z = a2n+1. Or let x = and y=, n < m. and let z = b2n+1. With the chosen z xz is in the language and yz is not. We can prove this by doing the calculation yz = =, m2+2n+1 cannot be a perfect square because if n < m then 2n+1 < 2m+1 and to be equal to the next perfect square((m+1)2 =m2+2m+1) in the sequence m would have to be equal to n. therefor this language is nonregular (infinitely many equivalence classes).

147. BoundedExponents

a) A = {aibj | i ≥ j and j ≤ 100}

* We can prove that A is regular by thinking about the expression in different parts. We can rewrite aibj as **aja\*bj**. A DFA can be built to accept aj, (A DFA with 100 states, all accepting, and all have a transition with the symbol *a*), a DFA to accept a\* is very simple to construct, (two states, the start state accepting with a transition to itself with the symbol *a,* and a transition to another state with the symbol *b*. The second state is a dead state, so its transitions loop back to itself). A DFA to accept bj is the same as the DFA to accept aj but all the transition uses the symbol *b*. (See below for pictures of the DFAs) Using concatenation we can combine these DFAs to create a machine that accepts A. Because we can create a DFA through concatenation we can conclude that A is regular.



aj:

Diagram

Description automatically generated

a\*:

Diagram

Description automatically generatedbj:

b) B ={aibj | i ≥ j and j ≥ 100}

* Take the infinite set of strings AB = {an | n ≥ 100}. For each string in this set ai is inequivalent to aj with respect to B. Let z = bi and i < j, this means that aibi is an element of B but aibj is not an element of B. Because any two elements from AB are inequivalent to each other with respect to B, there are an infinite number of equivalence states so B cannot be regular. B is nonregular.

156. AcceptEveryEvenDFA

On input M.

CONSTRUCT a DFA ME such that ME accepts all even length strings.

CONSTRUCT a DFA P such that L(P) = L(M) ∩ L(ME);

CALL Algorithm DFA Equivalence on P and ME;

RETURN that answer.

159. DifferInfiniteDFA

On input M and N

CONSTRUCT a DFA MC to be the complement of M;

CONSTRUCT a DFA NC to be the complement of N;

CONSTRUCT a DFA P to be (M ∩ NC) ∪ (MC ∩ N);

CALL Algorithm DFA Infinite on P

Return that answer