**Assignment 3**

**Kewei Qiu**

**# 1003798116**

**Tianyi Long**

**# 1002902889**

**Question 1**

To prove term(x) terminates we first need to find a loop invariant

Define We have , .

WTS ,

Proof: (Simple Induction)

Base Case:

This means that the loop has not been executed and only the line out of loop is

executed

Then by that initialization line, ,

holds

Inductive Step: Let , . Assume holds

We want to show that follows

(By code in the loop)

(By )

(Since by code)

(By code in the loop)

(By )

Hence follows.

By simple induction, we proved that we have

, , which is the loop invariant. ￭

Then we prove termination using loop invariant

Proof: By loop invariant we have

Since and is strictly increasing

We know that is strictly decreasing

Also since , we have exhibited a decreasing sequence of natural numbers linked to

loop iterations.

By principle of well ordering, the set of has a smallest element, which has the index of

the last loop iteration

Hence the loop terminates ￭

**Question 2**

**(a)**

Let

And ,

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |

Define the smallest set such that:

(a)

(b)

Prove

Define , WTS

Proof: (Structural Induction)

Base Case:

The string has 0 , we have , so the implication in the first line of

invariant is true.

The string also has 0 , so the implication in the second line of invariant is vacuously

true.

holds.

Inductive Step: Let , assume holds.

WTS follow

There are two cases to consider

Case :

(By IH )

(one more a)

Case 2:

(By IH )

(add one b)

So follow

The first line of the invariant ensures that all strings with only s are accepted. The

contrapositive of the second line of the invariant ensures that any string that does not drive

the machine to state has at least one , in other words all strings that drive the machine

to state have at least one .

So . ￭

**(b)**

Let

And ,

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |

Define the smallest set such that:

(a)

(b)

Prove

Define , WTS

Proof: (Structural Induction)

Base Case:

The string has 0 , we have , so the implication in the first line of

invariant is true.

The string also has 0 , so the implication in the second line of invariant is vacuously

true.

holds.

Inductive Step: Let , assume holds.

WTS follow

There are two cases to consider

Case :

(By IH )

(one more b)

Case :

(By IH )

(add one a)

So follow

The first line of the invariant ensures that all strings with only s are accepted. The

contrapositive of the second line of the invariant ensures that any string that does not drive

the machine to state has at least one , in other words all strings that drive the machine

to state have at least one .

So . ￭

**(c)**

Let

And ,

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |

Define the smallest set such that:

(a)

(b)

Prove

Define , WTS

Proof: (Structural Induction)

Base Case:

The string has length 0, and 0 is an even number

We have , so the implication in the first line of invariant is true.

The string length 0, and 0 is an even number

So the implication in the second line of invariant is vacuously true.

holds.

Inductive Step: Let , assume holds.

WTS follow

There are two cases to consider

Case :

(By IH )

(one more element)

Case :

(By IH )

(one more element)

So follow

The first line of the invariant ensures that all strings with even length accepted. The

contrapositive of the second line of the invariant ensures that any string that does not drive

the machine to state does not have even length, in other words all strings that drive the

machine to state have odd length.

So . ￭

**(d)**

Let

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

To show accepts :

Denote the states for as , the states for as , their respective transition functions

as and , and the transition function for as .

Inspection of shows that if ,

then .

Thus the following invariant follows by simply taking conjunctions of the invariants of the component machines, for any

The implication on the first line ensures that all strings with end up in state.

The implication on the second line ensures that all strings with end up in state.

The implication on the third line ensures that all strings with end up in state.

The contrapositive of the implications on the forth line ensure that any string that does not drive the machines to one of those 3 states must have.

Hence accepts

**(e)**

Let

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |

To show accepts:

Denote the states for as , the states for as , their respective transition functions as and , and the transition function for as .

Inspection of shows that if ,

Then .

Thus the following invariant follows by simply taking conjunctions of the invariants of the component machines, for any

The implication on the first line ensures that all strings with

end up in state .

The implication on the first line ensures that all strings with

end up in state .

The implication on the first line ensures that all strings with

end up in state .

The contrapositive of the implications on the other 5 lines ensure that any string that does not drive the machines to one of those 3 states must have

Hence accepts

**Question 3**

**(a) construct machines**

(1) Let

And:

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

(2) Let

And:

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

(3) Let

And:

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**(b) explain equality**

(1) Let

And:

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

When creating a machine accepting , we find that it is the same as itself:

① Both of two machines have the same 3 states

② Both of two machines the same start and accept state

③ Giving to any state will let the machine hold its current state

④ Giving to any state will let the machine jump to the next state and this creates a loop of states with no duplicate and no omit

⑤ Giving to any state will let the machine jump to the previous state and this creates a loop of states with no duplicate and no omit

By this we may find out that are aexctly the same

Hence we can conclude that

(2) Let

And:

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

When creating a machine accepting , we find that it is the same as itself:

① Both of two machines have the same 3 states

② Both of two machines have one start and one accepting state and they are different

③ Giving to any state will let the machine hold its current state

④ Giving to any state will let the machine jump to the next state and this creates a loop of states with no duplicate and no omit

⑤ Giving to any state will let the machine jump to the previous state and this creates a loop of states with no duplicate and no omit

By this we may find out that are aexctly the same

Hence we can conclude that

(3) Let

And:

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

When creating a machine accepting , we find that it is the same as itself:

① Both of two machines have the same 3 states

② Both of two machines have one start and one accepting state and they are different

③ Giving to any state will let the machine hold its current state

④ Giving to any state will let the machine jump to the next state and this creates a loop of states with no duplicate and no omit

⑤ Giving to any state will let the machine jump to the previous state and this creates a loop of states with no duplicate and no omit

By this we may find out that are aexctly the same

Hence we can conclude that

**Question 4**

**(a)**

Let , , then .

Define

Want to show:

Proof: (Structural Induction)

Base Case: Case 1: , Then

Case 2: , Then

Case 3: , ,

Then

So hold.

Inductive Step: Let

i.e.

Want to show:

There are 3 cases to consider

Case 1: for some be regular expression over .

And

Then

(by IH)

Case 2: for some and be regular expression over .

And

Then

(Since IH)

Case 3: for some and be regular expression over .

And

Then

(Since IH)

So all three cases hold.

By structural induction, combine base cases and inductive step, we proved

￭

**(b)**

Let .

Define

Want to show:

Proof: (Structural Induction)

Base Case: Case 1: ,

Then

Case 2: ,

Then

Case 3: , ,

Then

So hold.

Inductive Step: Assume

i.e.

Want to show:

There are 3 cases to consider

Case 1: for some be regular expression over .

And

Then

(by IH)

Case 2: for some and be regular expression over .

And

Then

(Since IH)

Case 3: for some and be regular expression over .

And

Then

(Since IH)

So all three cases hold.

By structural induction, combine base cases and inductive step, we proved

￭

(c)

Define

Want to show:

Proof: (Structural Induction)

Base Case: Case 1: , r does not have Kleene Star

Then is finite

Case 2: , r does not have Kleene Star

Then is finite

Case 3: , , r does not have Kleene Star

Then is finite

So hold.

Inductive Step: Let , assume hold

There are 3 cases to consider

Case 1: , has a Kleene Star then is vacuously true.

Case 2: , does not have Kleene Star, neither.

, is finite (By IH) so is .

Case 3: , does not have Kleene Star, neither.

, is finite (By IH).

Then is finite so is .

So all three cases hold.

By structural induction, combine base cases and inductive step, we proved that

￭

**Question 5**

**(a)**

Proof: (Contradiction)

Suppose, for the propose of contradiction, that we can find a correct DfA that has 8 states

By Pigeonhole Principle, if we choose 9 strings over , then at least two of those strings

must end at the same state

We pick nine prefixes of length 2 of , as nine strings, i.e. aa, ab, ac, ba, bb, bc, ca, cb, cc

For one of the pairs of strings in previous line (we don’t know which pair),

the supposed 8-state DFA is forced into the same state for both strings (because of

Pigeonhole), and and must be both accepted or both rejected, for any string

s.t.

We will show, for each possible pair, that this is NOT true

Pair 1: aa and ab

Choose = aa

Then = aaaa is accepted, = abaa is rejected

Pair 2: aa and ac

Choose = aa

Then = aaaa is accepted, = acaa is rejected

Pair 3: aa and ba

Choose = aa

Then = aaaa is accepted, = baaa is rejected

Pair 4: aa and bb

Choose = aa

Then = aaaa is accepted, = bbaa is rejected

Pair 5: aa and bc

Choose = aa

Then = aaaa is accepted, = bcaa is rejected

Pair 6: aa and ca

Choose = aa

Then = aaaa is accepted, = caaa is rejected

Pair 7: aa and cb

Choose = aa

Then = aaaa is accepted, = cbaa is rejected

Pair 8: aa and cc

Choose = aa

Then = aaaa is accepted, = ccaa is rejected

Pair 9: ab and ac

Choose = ba

Then = abba is accepted, = acba is rejected

Pair 10: ab and ba

Choose = ba

Then = abba is accepted, = baba is rejected

Pair 11: ab and bb

Choose = ba

Then = abba is accepted, = bbba is rejected

Pair 12: ab and bc

Choose = ba

Then = abba is accepted, = bcba is rejected

Pair 13: ab and ca

Choose = ba

Then = abba is accepted, = caba is rejected

Pair 14: ab and cb

Choose = ba

Then = abba is accepted, = cbba is rejected

Pair 15: ab and cc

Choose = ba

Then = abba is accepted, = ccba is rejected

Pair 16: ac and ba

Choose = ca

Then = acca is accepted, = baca is rejected

Pair 17: ac and bb

Choose = ca

Then = acca is accepted, = bbca is rejected

Pair 18: ac and bc

Choose = ca

Then = acca is accepted, = bcca is rejected

Pair 19: ac and ca

Choose = ca

Then = acca is accepted, = caca is rejected

Pair 20: ac and cb

Choose = ca

Then = acca is accepted, = cbca is rejected

Pair 21: ac and cc

Choose = ca

Then = acca is accepted, = ccca is rejected

Pair 22: ba and bb

Choose = ab

Then = baab is accepted, = bbab is rejected

Pair 23: ba and bc

Choose = ab

Then = baab is accepted, = bcab is rejected

Pair 24: ba and ca

Choose = ab

Then = baab is accepted, = caab is rejected

Pair 25: ba and cb

Choose = ab

Then = baab is accepted, = cbab is rejected

Pair 26: ba and cc

Choose = ab

Then = baab is accepted, = ccab is rejected

Pair 27: bb and bc

Choose = bb

Then = bbbb is accepted, = bcbb is rejected

Pair 28: bb and ca

Choose = bb

Then = bbbb is accepted, = cabb is rejected

Pair 29: bb and cb

Choose = bb

Then = bbbb is accepted, = cbbb is rejected

Pair 30: bb and cc

Choose = bb

Then = bbbb is accepted, = ccbb is rejected

Pair 31: bc and ca

Choose = cb

Then = bccb is accepted, = cacb is rejected

Pair 32: bc and cb

Choose = cb

Then = bccb is accepted, = cbcb is rejected

Pair 33: bc and cc

Choose = cb

Then = bccb is accepted, = cccb is rejected

Pair 34: ca and cb

Choose = ac

Then = caac is accepted, = cbac is rejected

Pair 35: ca and cc

Choose = ac

Then = caac is accepted, = ccac is rejected

Pair 36: cb and cc

Choose = bc

Then = cbbc is accepted, = ccbc is rejected

By now we have showed that none two of those nine strings we chosen must end at the

same state, which is a contradiction to our assumption

Hence we can conclude that any DFA accepts has at least nine states ￭

**(b)**

By (a) we may find out that the critical of finding list number of states is finding the largest number of prefixes with the same length and at the same time every pair of them may end at different states

To make the number of prefixes as large as possible, we need to let the length of the prefix be as large as possible

So for a string s.t. = n, is the largest possible length of the prefix which can ensure that every pair of them may end at different states

Since = 3, for a string s.t. = n, there are different prefixes

Hence by conclusion of (a), any DFA accepts has at least states

Since the length of a string , n, which belongs to , is infinite

may also be infinite because it depends only on n

This gives us a result that the DFA may end up with infinite number of states when is infinite long, which is impossible because the number of states of a DFA is finite

Hence we cannot say anything about a DFA that accepts simply by generalizing the result of part (a)