CSC236 A3

1.

WTS：term(x) terminates.

Proof:

.

Let be x after the ith iteration and be y after the ith iteration.

Define p(i): after the ith iteration of the loop (if it occurs), and and . I will prove that using simple induction on i.

Base case: (by initialization). Since , , hence . By code (line4) we know that if there is a iteration then Since , x is an integer, hence is also an integer. So p(0) follows.

Inductive step:

Let and assume p(i), that is and and . Show that p(i+1) follows. If there is an (i+1)th loop iteration.

Then by code, ,

(# by induction hypothesis we know that . If there is a next iteration, then by code (line4), we know that . Since there is an (i+1)th loop iteration, by the loop condition we know that , and by induction hypothesis we know that hence , and by induction hypothesis we know that , hence , since , hence , hence . Since by induction hypothesis, , is also an integer. So p(i+1) follows.

Hence we’ve proved that after the ith iteration of the loop (if it occurs), and and . Then we will prove termination by this loop iteration.

Try the sequence , since by loop iteration we know that . Hence Hence . Hence each element of the sequence is a natural number. It remains to show that the sequence is strictly decreasing. Suppose that there is an (i+1)th iteration of the loop, then by loop iteration we know that, hence since is monotonic increasing. So, the sequence is strictly decreasing.

Since a strictly decreasing sequence in is finite, and hence has a last (smallest) element. Thus, the loop terminates.

Hence, we’ve proved that term(x) terminates.

2.

(a)

Here is my specification for that accepts :

|  |  |  |
| --- | --- | --- |
|  | a | b |
|  |  |  |
|  |  |  |

Prove that accepts :

Define as the smallest set such that:

(a)

(b)

Define as:

I will prove that by structural induction.

Base case:

contains only and so the implication in the first line of the invariant is true in this case. Also, since does not contain any , the implication in the second line of the invariant is vacuously true. So holds.

Inductive step:

Let , assume . I will show that and follow. There are two cases to consider:

Case : Then

# by

# one more

So follows.

Case : Then

# by

# add one cause the set contain at least one

contains at least one so the implication in the second line of the invariant is true in this case. Also, since does not contain only , the implication in the first line of the invariant is vacuously true. So follows.

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

The contrapositive of the second line of the invariant ensures that any string that does not drive the machine to state does not contain any , in other words all strings that drive the machine to state contain only .

Notice that a string contains only is the same to say that this string , where .

So accepts .

(b)

Here is my specification for that accepts :

|  |  |  |
| --- | --- | --- |
|  | a | b |
|  |  |  |
|  |  |  |

Prove that accepts :

Define as the smallest set such that:

(a)

(b)

Define as:

I will prove that by structural induction.

Base case:

contains only and so the implication in the first line of the invariant is true in this case. Also, since does not contain any , the implication in the second line of the invariant is vacuously true. So holds.

Inductive step:

Let , assume . I will show that and follow. There are two cases to consider:

Case : Then

# by

# one more

So follows.

Case : Then

# by

# add one cause the set contain at least one

contains at least one so the implication in the second line of the invariant is true in this case. Also, since does not contain only , the implication in the first line of the invariant is vacuously true. So follows.

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

The contrapositive of the second line of the invariant ensures that any string that does not drive the machine to state does not contain any , in other words all strings that drive the machine to state contain only .

Notice that a string contains only is the same to say that this string , where .

So accepts .

(c)

Here is my specification for that accepts :

|  |  |  |
| --- | --- | --- |
|  | a | b |
| E | O | O |
| O | E | E |

Prove that accepts :

Define as the smallest set such that:

(a)

(b)

Define as:

I will prove that by structural induction.

Base case:

, an even number, and so the implication in the first line of the invariant is true in this case. Also, since is not odd, the implication in the second line of the invariant is vacuously true. So holds.

Inductive step:

Let , assume . Let . I will show that follows.

# by

# one more element

So follows.

The first line of the invariant ensures that all strings with an even number of elements are 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 an odd number of elements, in other words all strings that drive the machine to state have an even number of elements.

Notice that a string with an even number of elements is the same to say that is even.

So accepts .

(d)

Here is my specification for that accepts :

|  |  |  |
| --- | --- | --- |
|  | a | b |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

Prove that 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 contain only and only end up in state .

The implication on the second line ensures that all strings contain only and at least one end up in state .

The implication on the third line ensures that all strings contain at least one and only end up in state .

The contrapositive of the implication on the fourth line ensure that any string that does not drive the machine to state must contain only , or only , or only and only .

Hence accepts .

(e)

Here is my specification for that accepts :

|  |  |  |
| --- | --- | --- |
|  | a | b |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

(Notice that state is unreachable, just keep it for the sake of proof).

Prove that 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 contain only and only , and have an even number of elements end up in state .

The implication on the third line ensures that all strings contain only and at least one , and have an even number of elements end up in state .

The implication on the fifth line ensures that all strings contain at least one and only , and have an even number of elements end up in state .

The contrapositive of the implications on the other lines ensure that any string that does not drive the machines to one of those 5 states must (contain only and have an even number of elements), or (contain only and have an even number of elements), or (contain only and only , and have an even number of elements).

Hence accepts .

3.

Explanation:

**Show that** **:**

Here is my specification for that accepts :

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |

By swapping starting and accepting states and reversing all transitions of , here is my specification for that accepts :

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |

From the above we can see:

1. Both , have three states; , have the same ;

2. The accepting state and starting state of are the same state;

3. The accepting state and starting state of are the same state;

4. In any of the three states of or , (0, 3, 6, 9) keeps the machine in the same state, (1, 4, 7) drives the machine to one of the remaining two states, and (2, 5, 8) drives the machine to the other one of the remaining two states.

Thus, and are actually the same machine.

Since accepts and accepts , we can say that accepts and accepts .

Therefore, .

**Show that :**

Here is my specification for that accepts :

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |

By swapping starting and accepting states and reversing all transitions of , here is my specification for that accepts :

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |

From the above we can see:

1. Both , have three states; , have the same ;

2. The accepting state and starting state of are different states, and they are adjacent;

3. The accepting state and starting state of are different states, and they are adjacent;

4. In any of the three states of or , (0, 3, 6, 9) keeps the machine in the same state, (1, 4, 7) drives the machine to one of the remaining two states, and (2, 5, 8) drives the machine to the other one of the remaining two states.

Thus, and are actually the same machine.

Since accepts and accepts , we can say that accepts and accepts .

Therefore, .

**Show that :**

Here is my specification for that accepts :

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |

By swapping starting and accepting states and reversing all transitions of , here is my specification for that accepts :

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |

From the above we can see:

1. Both , have three states; , have the same ;

2. The accepting state and starting state of are different states, and they are adjacent;

3. The accepting state and starting state of are different states, and they are adjacent;

4. In any of the three states of or , (0, 3, 6, 9) keeps the machine in the same state, (1, 4, 7) drives the machine to one of the remaining two states, and (2, 5, 8) drives the machine to the other one of the remaining two states.

Thus, and are actually the same machine.

Since accepts and accepts , we can say that accepts and accepts .

Therefore, .

4.

(a)

WTS:

Let RE be the set of regular expressions over the alphabet ={0,1}, Define p(r):, I will show that by structural induction on r. Let

Basis: Let r,

For r, then , hence by the definition of RE, we know that , and , hence . And , hence .

For r, then , hence by the definition of RE, we know that , and , hence (since ). And , hence .

For r, then , hence by the definition of RE, we know that , and , hence (since ). And , hence .

For r, then , hence by the definition of RE, we know that , and , hence (since ). And , hence .

So p(r) holds.

Inductive step:

Let t, s, assume p(t), p(s), that is , . Let be those regular expression. I will show that (t+s), (ts), () follows. And I will prove this in 3 cases.

1. To show that (t+s) follows.

Take , since and , hence by the definition of RE, we know that . And we will show that and .

First, show that , let , then , let xx be that regular expression, since xx, xx must be one of the element of , hence must be one of the element of (since by inductive hypothesis and ), hence , hence .

Second, show that , let , hence must be one of the element of . Then , y=, let yy be that regular expression (since by inductive hypothesis and ), hence , hence , that is

Hence and . That is .

So (t+s) follows.

1. To show that (ts) follows.

Take , since and , hence by the definition of RE, we know that. And we will show that and .

First, show that , let , then , let xx be that regular expression, since xx, xx must be a concatenation of one of the element of , Let’s show it as

Hence must be one of the element of must be one of the element of (since by inductive hypothesis and ), hence , hence .

Second, show that , let , hence must be a concatenation of one of the element of . Let show it as , let them be their value. Then , =, let be that regular expression (since by inductive hypothesis and ), hence , hence , that is

Hence and .

So (ts) follows.

1. To show that () follows.

Take , since , hence by the definition of RE, we know that . And we will show that and .

First, show that , let , then , let be that regular expression, since , must be a star of some of the elements of ,let those elements be (since the string are always finite, we only suppose 3 to be the number of the element, and this will cover all the cases) and . And by the definition of the reverse of the string, we know that , and must be one of the element of (since by inductive hypothesis ), hence , hence .

Second, show that , let , hence must be one of the element of , hence y must be a concatenation of some elements of , let them be (since the string are always finite, we only suppose 3 to be the number of the element, and this will cover all the cases).Then y= . And ,===, let be that regular expression (since by inductive hypothesis ), and by the definition of the reverse of the string , and since hence That is y=

Hence and . That is .

So () follows.

Hence we’ve proved that

(b) WTS:

Let RE be the set of regular expressions over the alphabet ={0,1}, Define p(r):, I will show that by structural induction on r. Let

Basis: Let r,

For r, then , hence by the definition of RE, we know that , and , hence . And , hence .

For r, then , hence by the definition of RE, we know that , and , hence (). And , hence .

For r, then , hence by the definition of RE, we know that , and , hence (). And , hence .

For r, then , hence by the definition of RE, we know that , and , hence (). And , hence .

So p(r) holds.

Inductive step:

Let t, s, assume p(t), p(s), that is , . Let be those regular expression. Hence we can write the above as . I will show that (t+s), (ts), () follows. And I will prove this in 3 cases.

1. To show that (t+s) follows.

Take , since and , hence by the definition of RE, we know that . And we will show that and .

First, show that , let then , let y be that value, since , xy must be one of the element of , if xy is a element of L(t), then then (by inductive hypothesis ,if xy is a element of L(s), then then (by inductive hypothesis

Hence .

Second, show that , let (by inductive hypothesis , hence must be one of the element of. If x is an element of that is If x is an element of that is Hence

Hence and . That is .

So (t+s) follows.

1. To show that (ts) follows.

Take , since and and , hence by the definition of RE, we know that. And we will show that and .

First, show that , let , then , let y be that value, since , xy must be a concatenation of one of the element of , Let’s show it as

Hence if |x|<||, then x is some first part of then . (by induction hypothesis

If |, then x is the concatenation of with some first part of, then (by induction hypothesis

Hence .

Second, show that , let by inductive hypothesis ,

If , then , let y be that value, let , then xys, since RE is over hence , hence =, hence .

If , then , hence , hence .

So (ts) follows.

1. To show that () follows.

Take , since , hence by the definition of RE, we know that . And we will show that and .

First, show that , let , , hence (since the string are always finite, we only suppose 3 to be the number of the element, and it will cover all the cases).

if |x|<||, then x is some first part of then . (by induction hypothesis

If ||+||>|, then x is the concatenation of with some first part of, then (by induction hypothesis

If, then x is the concatenation of with some first part of, then (by induction hypothesis

Hence

Second, we will show that .Let ==. Since the string are always finite, we only suppose the first 3 sets to be the constraint of x, and it will cover all the cases, that is = (by inductive hypothesis ,

If , then , let y be that value, then xy , hence , hence .

If , then , hence , hence .

If , then , hence , hence .

So () follows.

Hence we’ve proved that

(c)

WTS: If does not contain the Kleene star, then |L(r)| is finite.

Proof:

Let RE be the set of regular expressions over the alphabet ={0,1}, Define p(r):, I will show that by structural induction on r. Let

Basis: Let r,

For r, obviously does not contain the Kleene star. And L(r)={}, hence |L(r)|=0, hence .

For r, obviously does not contain the Kleene star. And L()={}, hence |L(r)|=1, hence .

For r, obviously does not contain the Kleene star. And L()={}, hence |L(r)|=1, hence .

For r, obviously does not contain the Kleene star. And L()={}, hence |L(r)|=1, hence .

So p(r) holds.

Inductive step:

Let t, s, assume p(t), p(s), that is , . I will show that (t+s), (ts), () follows. And I will prove this in 3 cases.

1. To show that (t+s) follows.

I will show that (t+s) follows in 4 cases.

Hence by the induction hypothesis, we know that and . (, .)

Since , t+s must also .

And must be a finite number since and .

Hence p(r) holds in this case.

Then t+s must also contain the Kleene star.

Since the assumption is false, p(r) is vacuously true in this case.

Then t+s must also contain the Kleene star.

Since the assumption is false, p(r) is vacuously true in this case.

Then t+s must also contain the Kleene star.

Since the assumption is false, p(r) is vacuously true in this case.

Hence (t+s) follows.

1. To show that (ts) follows.

I will show that (ts) follows in 4 cases.

Hence by the induction hypothesis, we know that and . (, .)

Since , ts must also .

And must be a finite number since and .

Hence p(r) holds in this case.

Then ts must also contain the Kleene star.

Since the assumption is false, p(r) is vacuously true in this case.

Then ts must also contain the Kleene star.

Since the assumption is false, p(r) is vacuously true in this case.

Then ts must also contain the Kleene star.

Since the assumption is false, p(r) is vacuously true in this case.

Hence (ts) follows.

1. To show that () follows.

Since () itself contains the Kleene star.

Since the assumption is false, p(r) is vacuously true in this case.

Hence () follows.

Hence we’ve proved that If does not contain the Kleene star, then |L(r)| is finite.

5.

(a)

WTS: any DFA that accepts has at least nine states, not including dead states.

Proof:

Let ={a,b,c} and , I will prove that any DFA that accepts has at least nine states, not including dead states by contradiction. The proof is the following.

Assume, for the sake of contradiction, the negation of what we are proving, that is there is a DFA that accepts has less than 9 states not including dead states. That it be that value.

Then for this DFA, first ignoring the dead state, we know that if we choose 9 strings over , then there must be at least two strings that will end with the same states since we’ve assumed that this DFA has less than 9 states.

Let , ,,,, and , be these 9 strings. Since each of them can be transferred into a string of by concatenating some string after them, hence by the definition of DFA, none of them are in the dead state. Then at least two strings of them will end with the same states as proved above.

For the strings end with the same states, let s be a string over , then the concatenation of these strings with s must also end in the same state, since s=s and these strings end with the same states.

Hence these concatenations must be both accepted or both rejected.

Since at least two strings of will end with the same states, that means there are two strings from for any string s over , the concatenations of these two with s must always be both accepted or both rejected. (1)

We also know that:

Pair 1 and :

Choose s = aaa.

Thens = aaa, rejected; s = aaaa, accepted.

Pair 2 and :

Choose s = aa.

Then s = aa, rejected; s = aaaa, accepted.

Pair 3 and :

Choose s = a.

Then s = a, rejected; s = aaaa, accepted.

Pair 4 and :

Choose s = .

= is rejected; = aaaa is accepted.

Pair 5 and :

Choose s = bbb.

Then s = bbb, rejected; s = bbbb, accepted.

Pair 6 and :

Choose s = c.

Thens = c, rejected; s = cccc, accepted.

Pair 7 and :

Choose s = b.

Then s = b, rejected; s = bbbb, accepted.

Pair 8 and :

Choose s = ccc.

Then s = ccc, rejected; s = cccc, accepted.

Pair 9 and :

Choose s = aa.

Then s = aaa, rejected; s = aaaa, accepted.

Pair 10 and :

Choose s = a.

Thens = a, rejected; s = aaaa, accepted.

Pair 11 and :

Choose s = .

= a is rejected; = aaaa is accepted.

Pair 12 and :

Choose s = bbb.

Then s = abbb, rejected; s = bbbb accepted.

Pair 13 and :

Choose s = c.

Then s = ac, rejected; s = cccc accepted.

Pair 14 and :

Choose s = b.

Then s = ab, rejected; s = bbbb accepted.

Pair 15 and :

Choose s = ccc.

Then s = accc, rejected; s = cccc accepted.

Pair 16 and :

Choose s = a.

Then s = aaa, rejected; s = aaaa, accepted.

Pair 17 and :

Choose s = .

= aa is rejected; = aaaa is accepted.

Pair 18 and :

Choose s = bbb.

Then s = aabbb, rejected; s = bbbb accepted.

Pair 19 and :

Choose s = c.

Then s = aac, rejected; s = cccc accepted.

Pair 20 and :

Choose s = b.

Thens = aab, rejected; s = bbbb accepted.

Pair 21 and :

Choose s = ccc.

Then s = aaccc, rejected; s = cccc accepted.

Pair 22 and :

Choose s = .

= aaa is rejected; = aaaa is accepted.

Pair 23 and :

Choose s = bbb.

Then s = aaabbb, rejected; s = bbbb accepted.

Pair 24 and :

Choose s = c.

Then s = aaac, rejected; s = cccc accepted.

Pair 25 and :

Choose s = b.

Then s = aaab, rejected; s = bbbb accepted.

Pair 26 and :

Choose s = ccc.

Then s = aaaccc, rejected; s = cccc accepted.

Pair 27 and :

Choose s = .

= aaaa is accepted; = b is rejected.

Pair 28 and :

Choose s = .

= aaaa is accepted; = ccc is rejected.

Pair 29 and:

Choose s = .

= aaaa is accepted; = bbb is rejected.

Pair 30 and :

Choose s = .

= aaaa is accepted; = c is rejected.

Pair 31 and:

Choose s = c.

Then s = bc, rejected; s = cccc accepted.

Pair 32 and :

Choose s = b.

Then s = bb, rejected; s = bbbb accepted.

Pair 33 and :

Choose s = ccc.

Then s = bccc, rejected; s = cccc accepted.

Pair 34 and :

Choose s = b.

Then s = cccb, rejected;s = bbbb accepted.

Pair 35 and :

Choose s = ccc.

Then s = cccccc, rejected; s = cccc accepted.

Pair 36 and :

Choose s = ccc.

Then s = bbbccc, rejected; s = cccc accepted.

Hence for any two strings of , there is a string s over , make the concatenations of these two with s be one accepted and one rejected.

---><--- contradiction! The conclusion above is exactly the negation of (1) which we’ve assumed before. Since assuming that there is a DFA that accepts has less than 9 states leads to a contradiction, the assumption is false.

Hence we’ve proved that any DFA that accepts has at least nine states, not including dead states.

(b)

The DFA that accepts does not exist, the proof is the following:

Let , let

I will first prove that for , any DFA that accepts has at least states, not including the dead states. The proof is the following.

I will prove that for any 2 different prefixes of length , there is a string s over , the concatenation of these two prefixes with s will one be rejected one be accepted. We will prove this in 2 cases.

1.n is an even number. Then . Let x, y be any 2 different prefixes of length , let s=, then xs=x and |xs|=n, which is obviously accepted, while ys=y is rejected since , hence ys is not reversible. Hence the concatenation of these two prefixes with s are one rejected one accepted.

2. n is an odd number. Then . Let x, y be any 2 different prefixes of length , let s=, then xs = x and |xs| = | x, which is obviously accepted, while ys = y is rejected since , hence ys is not reversible. Hence the concatenation of these two prefixes with s are one rejected one accepted.

Hence we’ve proved that for any 2 different prefixes of length , there is a string s over , the concatenation of these two prefixes with s will one be rejected one be accepted. Hence the strings of these prefixes (since , hence the number of prefixes of length is ) are all end in different states as proved in (a), otherwise two of them end in the same states, for any string s over , the concatenation of those 2 with s will end in the same states which is a contradiction of what we proved before that any 2 different prefixes of length , there is a string s over , the concatenation of these two prefixes with s will one be rejected one be accepted.

Since the strings of these prefixes are all end in different states, and none of them are dead state since all of these prefixes can be leaded into a string of by concatenate with a string s over as showed in the prove above, hence there must be at least states, not including the dead states.

Hence, we’ve proved that for , any DFA that accepts has at least states, not including the dead states.

Generalize:

Let .

For the sake of contradiction, assume that there exist a DFA that accepts .

Then from what we’ve proven above, this DFA has at least states, not including the dead states.

I will show that this assumption for the general case is false.

We define by removing from . From the above proof, we didn’t prove any state be accepted or rejected depending on the length of .

Thus, removing does not affect the least number of states the DFA has.

So, for , does not have an upper bound, which means does not have an upper bound.

As . ----><---- contradiction, since a DFA should have a finite number of states (by the definition of DFA).

Therefore, I’ve shown that there does not exist a DFA that accepts .