**COT 4210 Fall 2014 Sample Problems Key**

1. Draw a DFA to recognize the set of strings over **{a,b}\***that contain the same number of occurrences of the substring **ab** as of the substring **ba**.

a

b

b

a

ab

a

a

**λ**

b

a



b

ba

b

b

 **2**. Present the transition diagram or table for a DFA that accepts the regular set denoted by the expression **(0+1)\* (010 + 11) (0 + 1)\***

|  |  |  |
| --- | --- | --- |
|  | **0** | **1** |
|  **<λ>** | <0> | <1> |
| **<0>** | <0> | <01> |
| **<01>** | <010+11> | <010+11> |
| **<1>** | <0> | <010+11> |
| **<010+11>** | <010+11> | <010+11> |

**3.** Consider the following assertion:

Let **R** be a regular language, then any set **S**, such that **S ∪ R = S**, is also regular.

State whether you believe this statement to be True or False by circling your answer.

 **~~TRUE~~** **FALSE**

If you believe that this assertion is True, present a convincing argument (not formal proof) to back up your conjecture. If you believe that it is False, present a counterexample using known regular and non-regular languages, **R** and **S**, respectively.

Let **R** = φ and **S = anbn**; **S ∪ R = anbn = S**, but S is not regular.

 **4.** Assume that **L1** and **L1∩L2** are both regular languages. Is **L2** necessarily a regular language? If so, prove this, otherwise show that **L2** could either be regular or non-regular.

**L2** could either be regular or non-regular

**L2** regular: **L1** = φ; **L2** = φ; **L1∩L2** = φ

**L2** non-regular: **L1** = φ; **L2** = **anbn**; **L1∩L2** = φ

**5.** Let **L** be defined as the language accepted by the finite state automaton **A**:

**A**

**B**

**C**

**D**

**E**

1

λ

0

1

0,1

λ

0

1

**A:**

 **a.)** Fill in the following table, showing the -closures for each of **A**’s states.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **State** | **A** | **B** | **C** | **D** | **E** |
| **-closure** | **{ A }** | **{ B , C }** | **{ C }** | **{ D, E }** | **{ E }** |

 **b.)** Convert **A** to an equivalent deterministic finite state automaton. Use states like **AC** to denote the subset of states **{A,C}**. Be careful -- -closures are important.

**A**

**BC**

**BCDE**

1

1

1

0

0

**A:**

**Λ**

0,1

0

**6.** Let **L** be defined as the language accepted by the finite state automaton **A**:

**A**

**B**

**C**

**A:**

0,1

0

0

1

0,1

0

 Change **A** to a GNFA (although I don’t see the need for all those φ transitions). Now, using the technique of replacing transition letters by regular expressions and then ripping (collapsing) states, develop the regular expression associated with **A** that represents the set (language) **L**.

**S**

**A**

**B**

**C**

**F**

**A:**

λ

0+1

0

0

1

0+1

λ

0

**S**

**B**

**C**

**F**

**A:**

0+1

0

1

0+1+0(0+1)

λ

00

0

**S**

**C**

**F**

**A:**

000\*1+0+1+0(0+1)

λ

0+1+00\*1

**S**

**F**

**A:**

(0+1+00\*1)(000\*1+0+1+0(0+1))\*

**Note:** (0+1+00\*1)(000\*1+0+1+0(0+1))\* = (0+1+0+1)(0+1)\* = (0+1)(0+1)\* + (0+1)(0+1)\* = (0+1)+

 **7.** Let **L** be defined as the language accepted by the finite state automaton **A**:

**A**

**B**

**C**

**A:**

0,1

0

0

1

0,1

0

 Using the technique of regular equations, develop the regular expression associated with **A** that represents the set (language) **L**.

**A = λ + C0**

**B = A0 + B0**

**C = A(0+1) + B1 + C(0+1)**

**B = 0 + C00 + B0 = (0 + C00) 0\***

**C = (0+1) + C0(0+1) + (0 + C00)0\*1 + C(0+1))**

 **= (0+1) + 0+1 + C(00 +01 + 0 +1 + 00+1)**

 **= (0+1) + 0+1 + C(00 + 0 +1 + 0+1)**

 **= (0+1+0+1) (00 + 0 +1 + 0+1)\***

 **= (0+1+0+1) (0 +1)\* = (0+1)+**

 **8.** Given a finite state automaton denoted by the transition table shown below, and assuming that **5** and **6** are final states, fill in the equivalent states matrix I have provided. Use this to create an equivalent, minimal state automaton. State **1** is the start state.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **a** | **b** | **c** |  |  |  |  |  |  |  |
| **>1** | **4** | **6** | **2** |  | **2** | **4,5X2,61,2** |  |  |  |  |
| **2** | **5** | **2** | **1** |  | **3** | **2,45,6** | **4,5X2,51,4** |  |  |  |
| **3** | **4** | **5** | **4** |  | **4** | **4,5X4,62,3** | **1,3** | **4,5X3,4** |  |  |
| **4** | **5** | **4** | **3** |  | **5** | **X** | **X** | **X** | **X** |  |
| **5** | **4** | **6** | **1** |  | **6** | **X** | **X** | **X** | **X** | **2,41,3** |
| **6** | **2** | **6** | **3** |  |  | **1** | **2** | **3** | **4** | **5** |

**Don’t forget to construct and write down your new, equivalent automaton!!**

**FIX**

b

**1,3**

**2,4**

**5,6**

**A:**

c

b

a

a, c

b

c

a

 **9.** Use the Pumping Lemma to show that the following languages are not regular:

 **a.**) **L = { an bm ct | n > m** or **n > t,** and **n, m, t** ≥ **0 }**

Let **N** be given by the P.L.

Choose **w = aN+1bNcN** Clearly **w ∈ L**

Let **w = xyz**, **|xy| ≤ N**, **|y|>0** be given by the P.L.

By P.L., **xyiz ∈ L**, for all **i ≥ 0**

Let **i=0**, then P.L. requires that **xz ∈ L**

However, since **|xy| ≤ N**, then **y** is a non-empty string consisting of only **a**’s.

Thus, we have that **aN+1-|y| bN cN** **∈ L**.

But, this is not so, since **N+1-|y| ≤ N** and hence the number of **a**’s is not greater than either of the number of **b**’s or **c**’s. This proves that L cannot be regular.

 **b.) L = { an bm | n ≤ m,** and **n, m** ≥ **0 }**

Let **N** be given by the P.L.

Choose **w = aNbN** Clearly **w ∈ L**

Let **w = xyz**, **|xy| ≤ N**, **|y|>0** be given by the P.L.

By P.L., **xyiz ∈ L**, for all **i ≥ 0**

Let **i=2**, then P.L. requires that **xy2z ∈ L**

However, since **|xy| ≤ N**, then **y** is a non-empty string consisting of only **a**’s.

Thus, we have that **aN+|y| bN ∈ L**.

But, this is not so, since **N+|y| > N** and hence the number of **a**’s is greater than the number of **b**’s. This proves that L cannot be regular.

**c.** Let **NonPrime** = { **aq** | **q is not a prime** }

 This language is not easily shown non-regular using the Pumping Lemma. However, we already saw in class how to prove that **Prime** = { **ap** | **p is a prime** } is non-regular using the Pumping Lemma. Explain how you could use this latter fact to show **NonPrime** is non-regular.

As regular languages are closed under complement, a demonstration that **NonPrime** is regular could be used to show **Prime** is also regular, but that is already known to be false, so **NonPrime** must itself be non-regular.

 **10.** DFAs accept the class of regular languages. Regular expressions denote the class of regular sets. The equivalence of these is seen by a proof that every regular set is a regular language and vice versa. The first part of this, that every regular set is a regular language, can be done by first showing that the basis regular sets (**Ø , {  } , { a | a ∈ ∑ }**) are each accepted by a DFA.

 **i.)** Demonstrate a DFA for each of the basis regular sets.

**Ø**

**a**

**∑**

**λ**

**{  }**

**∑**

**∑**

**λ**

**{ a }**

**∑**

**∑**

**λ**

Let **L1** be generated by the DFA **A1 = ( Q1 , ∑ , δ1 , q1 , F1 )** and **L2** be generated by the DFA
**A2 = ( Q2 , ∑ , δ2 , q2 , F2 )**.

 **ii.)** Present a construction from **A1 , A2** that produces a DFA **A3** for **L1** ∪ **L2**.
Hint: Cross product.

**A3 = ( Q1× Q2 , ∑ , δ3 , < q1,q2 >, F3 )**.

**δ3 (<q,r>, a) = <δ1 (q, a) , δ2 (r, a) >** where q **∈ Q1** ; **r ∈ Q2**; and **a ∈ Σ**

**F3 = F1×Q2** ∪ **Q1×F2**

Clearly **δ3**\*( **< q1,q2 >, w) = < δ1**\*( **q1 , w), δ2**\*( **q2 , w) >**

And thus **w ∈ L3**, just in case **w ∈ L1** or **w ∈ L2**

 **iii.)** What remains to be done to show that every regular set is a regular language? Don’t do the proof, just state what two steps still needs to be done.

Present a construction from **A1** and **A2** that produces a DFA for **L1 L2**. This is generally not done directly but by first showing NFAs are equivalent to DFAs and then producing a NFA for **L1 L2**.

Present a construction from **A1** that produces a DFA for **L1**\*. This is generally not done directly but by first showing NFAs are equivalent to DFAs and then producing a NFA for **L1**\*.