7/3/2023

CS704

Practice Questions

6) Consider the formula $\varphi(x) = \exists x((\forall y\, \neg(y < x) \wedge Q_a(x)))$ over $\{a\}$. Describe the language of words that satisfy φ in English. Also give a regular expression for the language.

Ans Explicitly reading the formula,

there exists a position x, for all positions y such that y is not before x, & position x is labelled with an 'a'.

The same can be interpreted as,

there exists a position x that is labelled with an 'a' & $x \leq y$ for all positions y.

Given that $\Sigma = \{a\}$, the above reading indicates that the language is the set of all words that contain an 'a'.

Regular expression is a^*.

---

7) Consider the formula $\varphi(y) = \exists x(x < y \wedge Q_a(x))$. List the free and bound variables of φ. Also describe the alphabet over which an automaton for φ will be constructed. Assume $\Sigma = \{a\}$.

Ans Variable y is free as it is not quantified.

Variable x is bound by the quantifier $\exists$.

$|\Sigma| = 1$ as $\Sigma = \{a\}$.

The automaton for φ has to assign an interpretation for y over the binary alphabet $\Sigma = \{0,1\}$.

So, the alphabet will be $\Sigma \times \{0,1\}$.

8) Describe the language defined by the sentence below, as a regular expression over the alphabet $\Sigma = \{a, b\}$.

$$\forall x (Q_b(x) \Rightarrow \exists y (y<x \wedge Q_a(y)))$$

Ans Explicitly reading the formula, we get,
for all positions x, x is labelled by a 'b' implies that there exists a position y before x & y is labelled by an 'a'.
So, 'a' occurs before 'b' in all the words that satisfy the given formula.
Regular expression will be $a \cdot (a+b)^*$.

9) Give a regular expression that describes the language defined by the MSO sentence below over the alphabet $\{a, b\}$.

$$\exists X((\exists x\, zero(x) \wedge x \in X) \wedge (\exists x (last(x) \wedge \neg x \in X)) \wedge$$
$$(\forall x \forall y (succ(x,y) \Rightarrow (x \in X \Leftrightarrow \neg y \in X)) \wedge$$
$$\forall x (x \in X \Rightarrow Q_a(x)))$$

Ans Again, explicitly reading the formula, we get,
There exists a set of positions X with the following properties:
First position (of a word) is in X, last position is not in X, ~~alter~~ consecutive positions are not in X, & every position in X is labelled with an 'a'.
The first three parts indicate that all words of even length will be models for X & the first & the last part indicate that positions

in X will be labelled with an 'a' & also begin with an 'a'.

The language is the set of all words of even length that start with an 'a' & alternate between 'a' & 'b' (with 'b' labeling alternate positions).

Regular expression will be

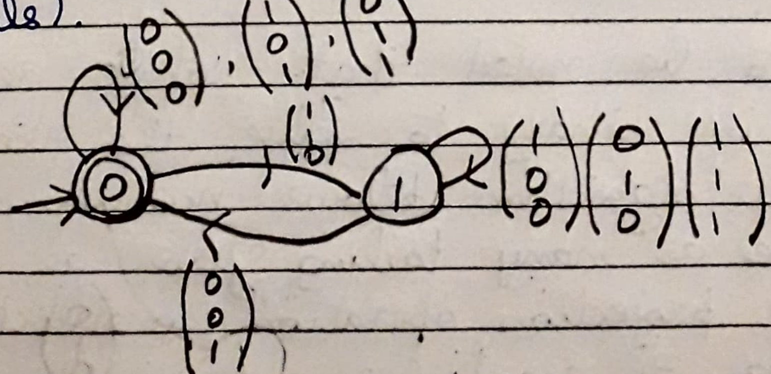
$$ab \cdot (ab)^*.$$

10) Give a finite state automaton that accepts all binary representations of i, j, k such that $i+j=k$

Ans: Re-writing $i+j=k$ as $i+j-k=0$.

The alphabet of this automaton will be $\{0,1\} \times \{0,1\} \times \{0,1\}$, one bit for each of i, j, k in the 3-tuple.

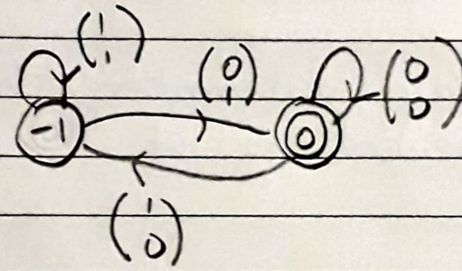
Just giving the final automaton below for reference (without the steps describing the details).



Note: We can also get a three-state automaton the above is not the only correct solution.

11) Construct an automaton that accepts all binary representations of x & y such that $y=2x+1$.

Ans: Rewriting $y=2x+1$ as $y-2x=1$.
Again, depicting only the final automaton without describing the details.
Alphabet is $\Sigma = \{0,1\} \times \{0,1\}$, one bit for each of x & y.



12) While using projection operation for the existential quantifiers during the inductive construction of the automation for a given linear (in)equality in Presburger's algorithm, briefly explain why we have to make a few extra states as final states.

Ans It is to be noted that while representing numbers in binary to solve the linear (in)equality Presburger's algorithm allows numbers in binary to have as many tailing zeros as needed.
During projection operation construction for handling $\exists$ quantifier, we simply 'erase' or 'project' the component corresponding to the variable within the scope of $\exists$.
This operation, when seen in the context

of allowing as many tailing zeros as needed will possibily prevent certain solutions to be accepted.

So, we check on a case-by-case basis (per quantifier-projection operation) & make additional states as final states. Sometimes, this could mean making all states as final.

After performing projection, we have to ~~do~~ check for states that can reach the final states through zeros & make them also as accepting states.

For eg. take a fragment of a particular automation that accepts $x=2$ & $y=4$. Their binary representations are $01\underline{00}$ & 1000.

tailing 0s

We have to ensure that all valid encodings of 2 (010, 0100 ...) are accepted.

13) Easy, so no solution key is provided.

14) Assume that the teacher's regular language is $L = \{w \in \{0,1\}^* \mid \#0s \text{ in } w \text{ is a multiple of } 3\}$. Describe the working of Angluins algorithm for L.

<u>Ans</u> Begin with $S=\{\epsilon\}$ & $E=\{\epsilon\}$.

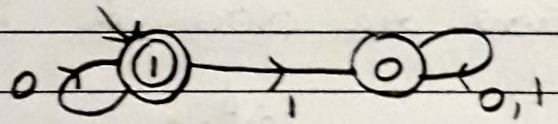
$T(\epsilon,\epsilon) = 1$ as ϵ is in L.

Similarly, $T(0,\epsilon)=0$ & $T(1,\epsilon)=1$. After the first couple of steps, we get

	ε
ε	1
0	0
1	1
00	0
01	0

the above closed & consistent observation table.

Constructing DFA for the table above



An equivalence query to the teacher results in the teacher giving 000 as a counter example.

So, add 00 to S & add 00 to S to keep it closed. We will get the table below

	ε
ε	1
0	0
00	0
000	1
1	1
01	0
001	0
0000	0
0001	1

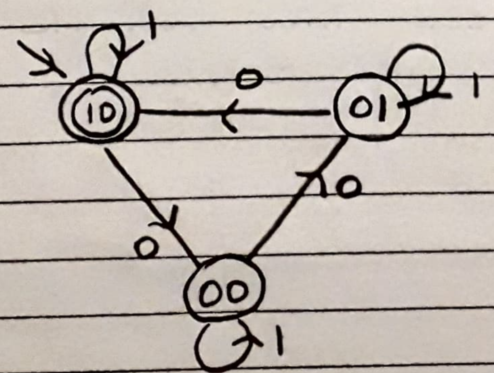
<u>Note</u>: All steps are not listed in this construction

This table is not consistent.
row(00) ≠ row(000).

Now add 0 to E (column) & build a larger observation table.

	ϵ	0
ϵ	1	0
0	0	0
00	0	1
000	1	0
1	1	0
01	0	0
001	0	1
0000	0	0
0001	1	0

The above observation table is closed and consistent. So, construct DFA, it will have three states as below:



An equivalence query to the teacher fetches a 'yes' answer & we are done!

Questions 15. & 16 are not answered here, 15 again follows Angluin's algorithm & 16 wis explained during lectures.