cpsc 313 assignment 2

Ali Akbari

TOTAL POINTS

45 / 50

QUESTION 1

1 question 1 9 / 10

- **- 0 pts** Correct
- ✓ **- 1 pts** **Minor Errors**
- **- 2 pts** Sloppyness
- **- 3 pts** Sloppiness
- **- 3 pts** Misuse of notation

Base Case

- **- 1 pts** Minor error with Base Case
- **- 2 pts** Errors with Base Case
- **- 3 pts** Major errors
- **- 3 pts** Incorrect Base Case or begging the question
- **- 4 pts** Missing Base Case

Inductive Hypothesis

- **- 1 pts** Errors with Inductive Hypothesis
- **- 2 pts** Inductive Hypothesis Incorrect
- **- 2 pts** Missing Inductive Hypothesis

Inductive Step

- **- 1 pts** Errors with Inductive Step
- **- 2 pts** Errors with Inductive Step
- **- 3 pts** Incorrect Inductive Step
- **- 4 pts** Missing Inductive Step

- **- 10 pts** Missing

1 and so your conclusion is?

QUESTION 2

2 question 2 10 / 10

- ✓ **- 0 pts** **Correct**
- **- 1 pts** Minor Errors
- **- 2 pts** Sloppyness
- **- 3 pts** Sloppiness
- **- 3 pts** Misuse of notation

Base Case

- **- 1 pts** Minor error with Base Case
- **- 2 pts** Errors with Base Case
- **- 3 pts** Major errors
- **- 3 pts** Incorrect Base Case
- **- 4 pts** Missing Base Case

Inductive Hypothesis

- **- 1 pts** Errors with Inductive Hypothesis
- **- 2 pts** Inductive Hypothesis Incorrect
- **- 2 pts** Missing Inductive Hypothesis

Inductive Step

- **- 1 pts** Errors with Inductive Step
- **- 2 pts** Errors with Inductive Step
- **- 3 pts** Incorrect Inductive Step
- **- 4 pts** Missing Inductive Step

- **- 10 pts** Missing

2 all n?

3 how does m relate to n?

QUESTION 3

3 question 3 10 / 10

- ✓ **- 0 pts** **Correct**
- **- 1 pts** Minor Errors
- **- 2 pts** Sloppiness
- **- 3 pts** Sloppiness
- **- 3 pts** Misuse of notation

Base Case

- **- 1 pts** Minor error with Base Case
- **- 2 pts** Errors with Base Case
- **- 3 pts** Major errors
- **- 3 pts** Incorrect Base Case
- **- 4 pts** Missing Base Case

Inductive Hypothesis

- **- 1 pts** Errors with Inductive Hypothesis

- **2 pts** Inductive Hypothesis Incorrect
- **2 pts** Missing Inductive Hypothesis

Inductive Step

- **1 pts** Errors with Inductive Step
- **2 pts** Errors with Inductive Step
- **3 pts** Incorrect Inductive Step
- **4 pts** Missing Inductive Step

- **10 pts** Missing

QUESTION 4

4 question 4 **6 / 10**

- **0 pts** Correct

✓ **- 2 pts Q is not defined or defined incorrectly**

- **2 pts** Transitions are partially correct
- **1 pts** Accept states A not defined or defined incorrectly. The answer is A = L
- **3 pts** Transitions \\delta incorrect or undefined.
- **1 pts** Q is vaguely defined. Its not very clear what Q is.
- **1 pts** There are mistakes in the definition of the transition function.
- **8 pts** Incorrect answer. Please refer to the solution.
- **10 pts** No solution or the solution is not relevant. Please refer to the solution.
- **1 pts** Errors in the solution. Refer to the comments.
- **3 pts** Transition function has major errors.
- **7 pts** Generic solution missing
- **1 pts** Mistakes in definition of Q
- **2 pts** Very Sloppy submission.
- **0 pts** Structure of the proof is confusing.
- **1 pts** Error in the definition of accepting state
- **5 pts** Vague specification of machine
- **1 pts** Sloppy submission

✓ **- 2 pts Transition function is vaguely defined. It not clear which state will the machine transition to from a state q on reading an alphabet a.**

- **2 pts** Transition function not defined clearly.
- **3 pts** Formal description of the DFA missing.

4 If the letter 'a' is an element of the language, this means that I go to a final state whenever I read an 'a' in the input string. This is incorrect. What you must have written is that if qa is in L, then move to final state.

5 same problem. If concatenating the letter results in a prequel to a string in the language, then move to a regular words. You need to define things correctly. A letter is an element of \Sigma. A string is an element of \Sigma\star. A letter is a one word string that may or may not be an element of the language.

QUESTION 5

5 question 5 **10 / 10**

✓ **- 0 pts Correct**

- **2 pts** When q_1 is the accepting state, it is possible that q_0 does not have any transition to q_1. Hence, the language is empty. This case is missing.
- **3 pts** Missing case: A = empty
- **3 pts** Missing case A = {q_0}
- **3 pts** Missing case A = {q_1} or the possibility that \\epsilon \\in L
- **1 pts** Sloppy submission
- **2 pts** Very sloppy submission

\+ **1 pts** Bonus for pictorial representation of the state machines in LaTex

- **0.5 pts** Errors in solution. Check comments
- **7 pts** Major errors in solution. Only one case is correct.
- **1 pts** Unclear explanation and/or mistakes in solution. Check comments.
- **8 pts** Refer to the solution uploaded. Incorrect answer.
- **10 pts** No solution
- **2 pts** Error in reasoning A = {q_0}
- **3 pts** When q_1 is the only accepting state, it is possible that q_0 has a transition to q_1 on input alphabet a. Hence, (iii) holds. Discussion of this case is missing.

CPSC 313 **Assignment #2**
Ali Akbari **30010402**
F2020

1. Suppose $M = (Q, \Sigma, \delta, q_0, A)$ with extended transition function $\delta^* : Q \times \Sigma^* \to Q$. Prove that for any $q \in Q$ and $x, y \in \Sigma$? that $\delta^* (q, xy) = \delta^* (\delta^* (q, x), y)$. Hint: prove by induction on $|y|$.

Base case:

$Let\, |y| = 0,\ then\ y = \varepsilon$

$Left\ Hand\ side:$

$\delta^* (q, xy) = \delta^* (q, x\varepsilon) = \delta^* (q, x)$

$Right\ Hand\ side:$

$\delta^* (\delta^* (q, x), y) = \delta^* (\delta^* (q, x), \varepsilon) = \delta^* (q, x)$

$LHS = RHS = \delta^* (q, x)$

$So\ this\ holds\ for\ the\ base\ case.$

Induction Step:

$IH:\ Assume\ \delta^* (q, xy) = \delta^* (\delta^* (q, x), y)\ holds\ for\ an\ arbitrary\ y\ where\ |y| = n.$

$\delta^* (q, xyw) = \delta^* (\delta^* (q, x), yw),\ where\ w\ \varepsilon\ \Sigma,\ then\ |yw| = n + 1.$

$\delta^* (q, xyw) = \delta\ (\delta^* (q, xy), w)$

$= \delta\ (\delta^* (\delta^* (q, x), y), w)$ $Further\ expansion\ by\ IH\ (Induction\ Hypothesis)$

$= \delta^* (\delta^* (q, x), yw)$

$\delta^* (\delta^* (q, x), yw) = \delta^* (q, xyw).$

$Thus\ it\ holds\ for\ the\ induction\ step.$

1

1 question 1 **9 / 10**

- **- 0 pts** Correct
- ✓ **- 1 pts Minor Errors**
- **- 2 pts** Sloppyness
- **- 3 pts** Sloppiness
- **- 3 pts** Misuse of notation

Base Case

- **- 1 pts** Minor error with Base Case
- **- 2 pts** Errors with Base Case
- **- 3 pts** Major errors
- **- 3 pts** Incorrect Base Case or begging the question
- **- 4 pts** Missing Base Case

Inductive Hypothesis

- **- 1 pts** Errors with Inductive Hypothesis
- **- 2 pts** Inductive Hypothesis Incorrect
- **- 2 pts** Missing Inductive Hypothesis

Inductive Step

- **- 1 pts** Errors with Inductive Step
- **- 2 pts** Errors with Inductive Step
- **- 3 pts** Incorrect Inductive Step
- **- 4 pts** Missing Inductive Step

- **- 10 pts** Missing

1 and so your conclusion is?

2. Prove by induction on n that if L is a language and R is a regular expression such that $L = L(R)$ then there exists a regular expression R_n such that $L(R_n) = L^n$. Be sure to use the fact that if R_1 and R_2 are regular expressions then $L(R_1 R_2) = L(R_1) \cdot L(R_2)$.

Base case:

$Let\ n = 1,\ then$

$L(R_n) = L^n \Rightarrow L(R) = L^1 = L(R) = L$

$When\ n = 1,\ there\ is\ a\ regular\ expression\ R,\ such\ that\ L(R_n) = L^n.$

$So\ this\ holds\ for\ the\ base\ case.$

Inductive Hypothesis:

$IH: suppose\ L\ is\ language\ and\ R_n\ is\ regular\ expression\ such\ that\ L = L(R)\ then\ there\ exists\ a\ regular\ expression\ R_n\ such\ that\ L(R_n) = L^n\ for\ n \geq 1.$

Induction Step:

$Prove\ this\ holds\ true\ for\ L^{n+1}.$

$L^{n+1} = L^n \cdot L^1$

$L^{n+1} = L^n \cdot L^1 = L(R_n) \cdot L(R)$ by IH (Induction Hypothesis)

$L^{n+1} = L^n \cdot L^1 = L(R_n) \cdot L(R) = L(R_n R)$ by given fact

$The\ given\ fact\ shows\ that\ the\ product\ of\ any\ two\ regular\ expressions\ is\ also\ a\ regular\ expression.$

$Let\ the\ product\ of\ the\ expressions\ equal\ R_m = R_n R.$

$Then\ there\ exist\ a\ regular\ expression\ R_m\ that\ hold\ for\ L^{n+1}.\ Thus\ it\ holds\ for\ the\ induction\ step.$

$This\ proves\ by\ induction\ for\ n \geq 1\ that\ if\ L\ is\ a\ language\ and\ R\ is\ a\ regular\ expression\ such\ that$

$L = L(R)\ then\ there\ is\ a\ regular\ expression\ R_n\ so\ that\ L(R_n) = L^n.$

2 question 2 **10 / 10**

- ✓ **- 0 pts** Correct
- **- 1 pts** Minor Errors
- **- 2 pts** Sloppyness
- **- 3 pts** Sloppiness
- **- 3 pts** Misuse of notation

Base Case

- **- 1 pts** Minor error with Base Case
- **- 2 pts** Errors with Base Case
- **- 3 pts** Major errors
- **- 3 pts** Incorrect Base Case
- **- 4 pts** Missing Base Case

Inductive Hypothesis

- **- 1 pts** Errors with Inductive Hypothesis
- **- 2 pts** Inductive Hypothesis Incorrect
- **- 2 pts** Missing Inductive Hypothesis

Inductive Step

- **- 1 pts** Errors with Inductive Step
- **- 2 pts** Errors with Inductive Step
- **- 3 pts** Incorrect Inductive Step
- **- 4 pts** Missing Inductive Step

- **- 10 pts** Missing

2 all n?

3 how does m relate to n?

3. Every finite-sized language L has a regular expression that accepts it. Informally, the expression for $\{w, x, y\}$ *is* $w + x + y$. Prove this is true for all finite-sized languages using induction on the number of words in L. You may need to use these definitions in your proof: (i) if w is a word then w is also a regular expression with $L(w) = \{w\}$, (ii) if R_1 and R_2 are regular expressions then $R_1 + R_2$ is a regular expression, and (iii) $L(R_1 + R_2) = L(R_1) \cup L(R_2)$.

Base case:

Let L be a finite language, and let n be the number of words in L.

Suppose L has no elements, so $n = 0$

$L = \varnothing$, *and a regular expression for L would be* $\varnothing$, *by the definition of a regular language.*

A language with no elements, i.e empty set { } has a regular expression of $\varnothing$.

So this holds for the base case.

Induction Step:

IH : *Suppose if L is a finite language that has n strings and R is regular expression such that R accepts L so* $L(R)$, *and,* $|L(R)| = n$.

Now prove that $L(R)$ *is true for* $n + 1$ *strings.*

w is an arbitrary word over Σ *that does not already exist in* $L(R)$, *and* $|L(w)| = 1$.

By definition w is also a $\boxed{\textit{regular language.}}$

$L(w) = \{w\}$, *w is a regular expression* — *by definition* (i)

$L(R) \cup L(w) = L(R + w)$ *is a regular language* — *by definition* (iii)

Since R and w are regular expressions, so is $R + w$ — *by definition* (ii)

$|L(R)| = n$ *and* $|L(w)| = 1$ *so,* — *by* IH(*inductive hypothesis*)

$|L(R) + w| = n + 1$

Then there exist a regular expression $R + w$ *that accepts the finite language L for string size* $n + 1$:

Thus it holds for the induction step.

3 question 3 **10 / 10**

- ✓ **- 0 pts** Correct
- **- 1 pts** Minor Errors
- **- 2 pts** Sloppiness
- **- 3 pts** Sloppiness
- **- 3 pts** Misuse of notation

Base Case

- **- 1 pts** Minor error with Base Case
- **- 2 pts** Errors with Base Case
- **- 3 pts** Major errors
- **- 3 pts** Incorrect Base Case
- **- 4 pts** Missing Base Case

Inductive Hypothesis

- **- 1 pts** Errors with Inductive Hypothesis
- **- 2 pts** Inductive Hypothesis Incorrect
- **- 2 pts** Missing Inductive Hypothesis

Inductive Step

- **- 1 pts** Errors with Inductive Step
- **- 2 pts** Errors with Inductive Step
- **- 3 pts** Incorrect Inductive Step
- **- 4 pts** Missing Inductive Step

- **- 10 pts** Missing

4. Give a generic construction for a finite state machine that accepts a nonempty finite-size language $L = \{w1, \dots wn\}$. Define all parts of the machine. (You do not need to prove it accepts the langauge). Hint: do this for a simple language like $\{\varepsilon, a, aa, ba, ab\}$ and then generalize what you did so it can work for any finite language. Hint: if you make $Q \subseteq \Sigma^*$ you can have an easy to describe construction.

A generic construction for a finite state machine that accepts a non-empty language is defined in the following:

$L = \{w1, \dots wn\}$, *since L is finite, L is also a regular language.*

Let $M = (Q, \Sigma, \delta, q_0, A)$ *be a DFA that accepts a non − empty finite language L.*

$Q = Q \subseteq \Sigma^*$ *which is any number of states needed to define a DFA for finite Language L.*

Σ = *Sigma is the same* Σ *that comes with the language L.*

Let q be a state in Q and let a $\varepsilon\ \Sigma$.

All transitions that lead to cycle or loops with a final state is not a DFA that works for a finite Language.

$\delta(q, a)$ = *in a state q read a letter a and go to a new state, if the letter is an element of the language the new state must be a final state.*

$\delta(q, a)$ = *in a state q read a letter a and go to a new state, if the letter is not an element of the language or, if the letter is not a prequel or an extension to a string from the language set then the new state is not a final state and must be a trap state, i.e sink state* q_e.

$\delta(q, a)$ = *in a state q read a letter a and go to a new state, if the letter is a prequel or an extension to a string from the language set then the new state is not a final state and must be a regular state.*

Assume any non − defined transition leads to a non − accepting state, i.e trap state.

q_0 *is the starting state.*

A = *The accepted final states.*

If the language contains epsilon then q_0 *the starting state, must be a final state.*

A state is a final state if a sequence of letters, through the transitions starting from the starting state to the current state forms a string which is an element of the langauge set, then the current state should be a final state.

The sink state is not a final state.

4 question 4 **6 / 10**

- **- 0 pts** Correct
- ✓ **- 2 pts Q is not defined or defined incorrectly**
- **- 2 pts** Transitions are partially correct
- **- 1 pts** Accept states A not defined or defined incorrectly. The answer is A = L
- **- 3 pts** Transitions \\delta incorrect or undefined.
- **- 1 pts** Q is vaguely defined. Its not very clear what Q is.
- **- 1 pts** There are mistakes in the definition of the transition function.
- **- 8 pts** Incorrect answer. Please refer to the solution.
- **- 10 pts** No solution or the solution is not relevant. Please refer to the solution.
- **- 1 pts** Errors in the solution. Refer to the comments.
- **- 3 pts** Transition function has major errors.
- **- 7 pts** Generic solution missing
- **- 1 pts** Mistakes in definition of Q
- **- 2 pts** Very Sloppy submission.
- **- 0 pts** Structure of the proof is confusing.
- **- 1 pts** Error in the definition of accepting state
- **- 5 pts** Vague specification of machine
- **- 1 pts** Sloppy submission
- ✓ **- 2 pts Transition function is vaguely defined. It not clear which state will the machine transition to from a state q on reading an alphabet a.**
- **- 2 pts** Transition function not defined clearly.
- **- 3 pts** Formal description of the DFA missing.

4 If the letter 'a' is an element of the language, this means that I go to a final state whenever I read an 'a' in the input string. This is incorrect. What you must have written is that if qa is in L, then move to final state.

5 same problem. If concatenating the letter results in a prequel to a string in the language, then move to a regular words. You need to define things correctly. A letter is an element of \Sigma. A string is an element of \Sigma\star. A letter is a one word string that may or may not be an element of the language.

5. Suppose L is a regular language, and $M = (Q, \Sigma, \delta, q_0, A)$ is a deterministic finite state machine such that $L(M) = L$. Prove that if $|Q| = 2$ then at least one of the following hold: $(i)\ L = \varnothing$ $(ii)\ \varepsilon \in L$, or $(iii)\ \exists a \in \Sigma$ such that $a \in L$. Note that different M with $|Q| = 2$ may have a different property that holds. Hint: do a case analysis for configurations of A.

Since the total number of states is 2, *and we start with* q_0, *then the set of states is* $Q = \{q_0, q_1\}$.
Then with a case analysis on the set of final states A we get that :

Case 1:
If there is no final state i.e the set of final state $A = \varnothing$ *then no language is accepted, so* $(i)\ L = \varnothing,\ L(M) = \varnothing$.
So property (i) *holds for this case.*

Case 2:
If there is one final state and it is q_0, *so* $q_0 \in A$, *then epsilon must be part of the language,* $(ii)\ \varepsilon \in L,\ L(M) = \varepsilon$.
So property (ii) *holds for this case.*

Case 3:
If there is more than one final states and q_1 *is one of them, so* $q_1 \in A$ *then there is word a* $(iii)\ \exists a \in \Sigma$,
that transitions from q_0 *to* q_1. $\delta(q_0, a) = q_1$ *so this entails that* $a \in L$.
So property (iii) *holds for this case.*

Case 4:
If there is one final state and it is q_1, *so* $q_1 \in A$ *and* q_0 *is not a final state,* $q_0 \notin A$.
Then there must be no transition from q_0 *to* q_1, *otherwise case* 3 *would hold, this means that no words/ the language is accepted. since no language transitions to the final state the language of the machine is* $L(M) = \varnothing$.
So property (i) *holds for this case.*

5 question 5 **10 / 10**

✓ **- 0 pts** Correct

- 2 pts When q_1 is the accepting state, it is possible that q_0 does not have any transition to q_1. Hence, the language is empty. This case is missing.

- 3 pts Missing case: A = empty

- 3 pts Missing case A = {q_0}

- 3 pts Missing case A = {q_1} or the possibility that \\epsilon \\in L

- 1 pts Sloppy submission

- 2 pts Very sloppy submission

+ 1 pts Bonus for pictorial representation of the state machines in LaTex

- 0.5 pts Errors in solution. Check comments

- 7 pts Major errors in solution. Only one case is correct.

- 1 pts Unclear explanation and/or mistakes in solution. Check comments.

- 8 pts Refer to the solution uploaded. Incorrect answer.

- 10 pts No solution

- 2 pts Error in reasoning A = {q_0}

- 3 pts When q_1 is the only accepting state, it is possible that q_0 has a transition to q_1 on input alphabet a. Hence, (iii) holds. Discussion of this case is missing.