CS 181 HW1 2021 CS181

CHARLES ZHANG

TOTAL POINTS

28.2 / 32

QUESTION 1

1 Set Operations 3.2 / 4

✓ **- 0.8 pts** c) incorrect

QUESTION 2

2 String Operations 4.5 / 5

✓ **- 0.5 pts** e) Concatenation contains $$\epsilon$$ iff L2 does.

QUESTION 3

3 Language of DFA 2 / 2

✓ **- 0 pts** Correct

QUESTION 4

Four Languages 14 pts

4.1 a 0.5 / 1

✓ **- 0.5 pts** did not mention the finite memory limitation of DFA or the description is not clear

4.2 b 4 / 5

✓ **- 1 pts** Almost correct

"aab" should be accepted.

4.3 c 2 / 2

✓ **- 0 pts** Correct

4.4 d 6 / 6

✓ **- 0 pts** Correct

QUESTION 5

5 Inductive Proof 5 / 6

✓ **- 1 pts** Lack of justification in the induction step, we expected a very detailed explanation for every step of your induction.

QUESTION 6

6 Explain Sipser's System 1 / 1

✓ **+ 1 pts** Correct

+ 0 pts No answer.

Charles Zhang
305413659

CS181 HW#1

1) $X = \{w, x, y, z\}$

$B = \{0, 1\}$

a) $(B \times B) \times B$

$B \times B = \{(0,0), (0,1), (1,0), (1,1)\}$

$(B \times B) \times B = \{((0,0),0), ((0,1),0), ((1,0),0), ((1,1),0), ((0,0),1), ((0,1),1), ((1,0),1), ((1,1),1)\}$

b) $B \times (B \times B)$

$B \times B = \{(0,0), (0,1), (1,0), (1,1)\}$

$B \times (B \times B) = \{(0,(0,0)), (0,(0,1)), (0,(1,0)), (0,(1,1)), (1,(0,0)), (1,(0,1)), (1,(1,0)), (1,(1,1))\}$

c) $B \times B \times B$

$B \times B \times B = \{((0,0),0), ((0,1),0), ((1,0),0), ((1,1),0), ((0,0),1), ((0,1),1), ((1,0),1), ((1,1),1)\}$

d) $|\mathcal{P}(N)| = 2^{|N|}$

$|X \times X| = |X| \cdot |X| = 4 \cdot 4$

$|X \times X| = 16$

$|\mathcal{P}(X \times X)| = 2^{16} = 65536$

e) $\{\varepsilon, a, ac\} \times \{a, c, aa\}$

$= \{(\varepsilon, a), (\varepsilon, c), (\varepsilon, aa), (a, a), (a, c), (a, aa), (ac, a), (ac, c), (ac, aa)\}$

2) $\Sigma' = \{a, b, c\}$

L_2 = language over Σ'

a) $\{\varepsilon, a, ac\} \cdot \{a, c, aa\}$ → ε is concat's identity

$= \{a, c, aa, ac, aaa, aca, acc, acaa\}$

b) $L_2^+ \cdot \{\}$

ϕ

c) $\{\varepsilon\} \times \Sigma'$

$\{\varepsilon\} \times \{a, b, c\}$

$\{(\varepsilon, a), (\varepsilon, b), (\varepsilon, c)\}$

1 Set Operations 3.2 / 4

✓ **- 0.8 pts** c) incorrect

Charles Zhang
305413659

CS181 HW#1

1) $X = \{w, x, y, z\}$

$B = \{0, 1\}$

a) $(B \times B) \times B$

$B \times B = \{(0,0), (0,1), (1,0), (1,1)\}$

$(B \times B) \times B = \{((0,0),0), ((0,1),0), ((1,0),0), ((1,1),0), ((0,0),1), ((0,1),1), ((1,0),1), ((1,1),1)\}$

b) $B \times (B \times B)$

$B \times B = \{(0,0), (0,1), (1,0), (1,1)\}$

$B \times (B \times B) = \{(0,(0,0)), (0,(0,1)), (0,(1,0)), (0,(1,1)), (1,(0,0)), (1,(0,1)), (1,(1,0)), (1,(1,1))\}$

c) $B \times B \times B$

$B \times B \times B = \{((0,0),0), ((0,1),0), ((1,0),0), ((1,1),0), ((0,0),1), ((0,1),1), ((1,0),1), ((1,1),1)\}$

d) $|\mathcal{P}(N)| = 2^{|N|}$

$|X \times X| = |X| \cdot |X| = 4 \cdot 4$

$|X \times X| = 16$

$|\mathcal{P}(X \times X)| = 2^{16} = 65536$

e) $\{\varepsilon, a, ac\} \times \{a, c, aa\}$

$= \{(\varepsilon, a), (\varepsilon, c), (\varepsilon, aa), (a, a), (a, c), (a, aa), (ac, a), (ac, c), (ac, aa)\}$

2) $\Sigma' = \{a, b, c\}$

L_2 = language over Σ'

a) $\{\varepsilon, a, ac\} \cdot \{a, c, aa\}$ → ε is concat's identity

$= \{a, c, aa, ac, aaa, aca, acc, acaa\}$

b) $L_2^+ \cdot \{\}$

ϕ

c) $\{\varepsilon\} \times \Sigma'$

$\{\varepsilon\} \times \{a, b, c\}$

$\{(\varepsilon, a), (\varepsilon, b), (\varepsilon, c)\}$

d) $\{\} \times L_2^+$

$\boxed{\emptyset}$

e) $\{\varepsilon\} \cdot L_2^+ \rightarrow \varepsilon$ is what's identity

L_2^+, no it doesn't contain the empty string, as the Kleene operation on a language excludes the empty string

3) $\Sigma' = \{0,1\}$

The language L_3 accepts any odd-length strings beginning with 1 and any even-length strings beginning in 0, with the exception of the empty string, which is rejected.

4) a) $\Sigma' = \{0,1,=,+\}$

$L_{4a} = \{w \in \Sigma'^+ \mid w \text{ is of the form } x+y=z, \text{ where } x,y,z \in \{0,1\}^+\}$

↳ correct binary sum

This is a non-FSL language, because, although it would be possible to implement a DFA to check for formatting of strings, FSLs do not have the capability to check if the binary sum is actually correct.

b) $\Sigma' = \{a,b\}$

$L_{4b} = \{w \in \Sigma'^+ \mid \text{all runs of } a\text{'s in } w \text{ are of even length, and all runs of } b\text{'s in } w \text{ are of odd length}\}$

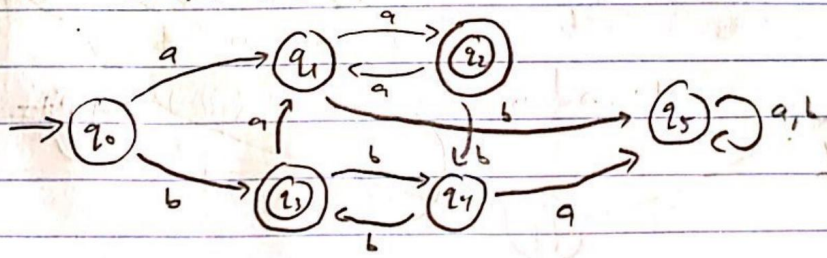
Q → states

Σ → alphabet

δ → transition function

q_0 → start state

F → accept states



2 String Operations 4.5 / 5

✓ **- 0.5 pts** **e) Concatenation contains $$\epsilon$$ iff L2 does.**

d) $\{\} \times L_2^+$

$\boxed{\emptyset}$

e) $\{\varepsilon\} \cdot L_2^+ \rightarrow \varepsilon$ is what's identity

L_2^+, no it doesn't contain the empty string, as the Kleene operation on a language excludes the empty string

3) $\Sigma' = \{0,1\}$

The language L_3 accepts any odd-length strings beginning with 1 and any even-length strings beginning in 0, with the exception of the empty string, which is rejected.

4) a) $\Sigma' = \{0,1,=,+\}$

$L_{4a} = \{w \in \Sigma'^+ \mid w \text{ is of the form } x+y=z, \text{ where } x,y,z \in \{0,1\}^+\}$

↳ correct binary sum

This is a non-FSL language, because, although it would be possible to implement a DFA to check for formatting of strings, FSLs do not have the capability to check if the binary sum is actually correct.

b) $\Sigma' = \{a,b\}$

$L_{4b} = \{w \in \Sigma'^+ \mid \text{all runs of } a\text{'s in } w \text{ are of even length, and all runs of } b\text{'s in } w \text{ are of odd length}\}$

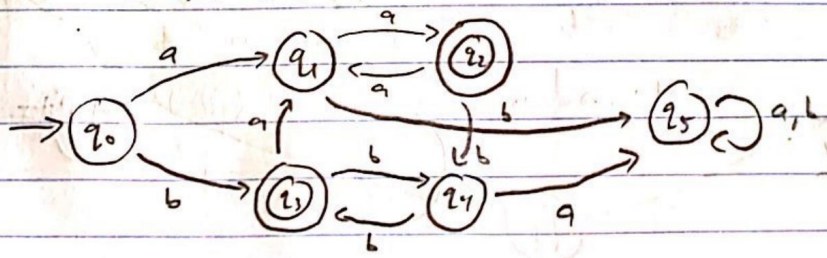
Q → states

Σ → alphabet

δ → transition function

q_0 → start state

F → accept states



3 Language of DFA **2 / 2**

✓ **- 0 pts** **Correct**

d) $\{\} \times L_2^+$

$\boxed{\emptyset}$

e) $\{\varepsilon\} \cdot L_2^+ \rightarrow \varepsilon$ is what's identity

L_2^+, no it doesn't contain the empty string, as the Kleene operation on a language excludes the empty string

3) $\Sigma' = \{0,1\}$

The language L_3 accepts any odd-length strings beginning with 1 and any even-length strings beginning in 0, with the exception of the empty string, which is rejected.

4) a) $\Sigma' = \{0,1,=,+\}$

$L_{4a} = \{w \in \Sigma'^+ \mid w \text{ is of the form } x+y=z, \text{ where } x,y,z \in \{0,1\}^+\}$

↳ correct binary sum

This is a non-FSL language, because, although it would be possible to implement a DFA to check for formatting of strings, FSLs do not have the capability to check if the binary sum is actually correct.

b) $\Sigma' = \{a,b\}$

$L_{4b} = \{w \in \Sigma'^+ \mid \text{all runs of } a\text{'s in } w \text{ are of even length, and all runs of } b\text{'s in } w \text{ are of odd length}\}$

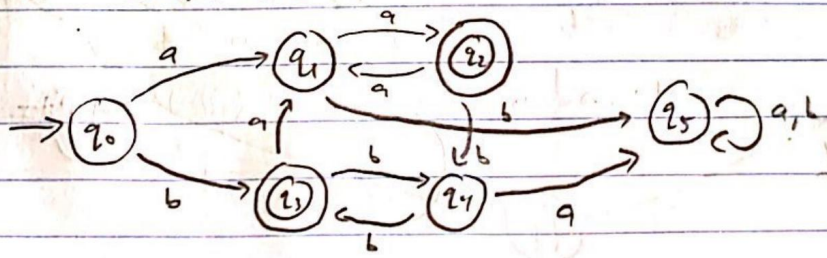
Q → states

Σ → alphabet

δ → transition function

q_0 → start state

F → accept states



4.1 a **0.5 / 1**

✓ **- 0.5 pts** did not mention the finite memory limitation of DFA or the description is not clear

d) $\{\} \times L_2^+$

$\boxed{\emptyset}$

e) $\{\varepsilon\} \cdot L_2^+ \rightarrow \varepsilon$ is what's identity

L_2^+, no it doesn't contain the empty string, as the Kleene operation on a language excludes the empty string

3) $\Sigma' = \{0,1\}$

The language L_3 accepts any odd-length strings beginning with 1 and any even-length strings beginning in 0, with the exception of the empty string, which is rejected.

4) a) $\Sigma' = \{0,1,=,+\}$

$L_{4a} = \{w \in \Sigma'^+ \mid w \text{ is of the form } x+y=z, \text{ where } x,y,z \in \{0,1\}^+\}$

↳ correct binary sum

This is a non-FSL language, because, although it would be possible to implement a DFA to check for formatting of strings, FSLs do not have the capability to check if the binary sum is actually correct.

b) $\Sigma' = \{a,b\}$

$L_{4b} = \{w \in \Sigma'^+ \mid \text{all runs of } a\text{'s in } w \text{ are of even length, and all runs of } b\text{'s in } w \text{ are of odd length}\}$

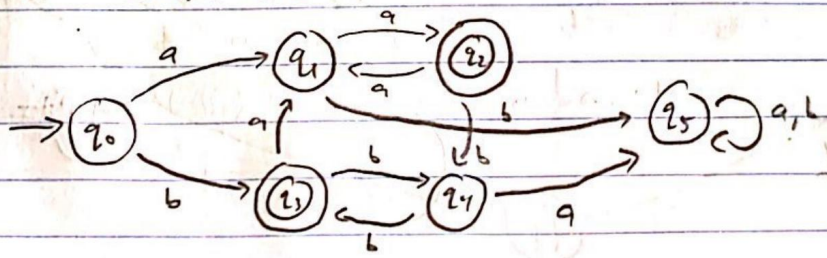
Q → states

Σ → alphabet

δ → transition function

q_0 → start state

F → accept states



$Q = \{q_0, q_1, q_2, q_3, q_4, q_5, q_6\}$

$\Sigma' = \{a, b\}$

$\delta =$

	a	b
q_0	q_1	q_3
q_1	q_2	q_5
q_2	q_1	q_4
q_3	q_1	q_4
q_4	q_5	q_3
q_5	q_5	q_5

$z_0 = q_0$

$F = \{q_2, q_3\}$

Since the language is defined by runs of a and b, it could be trivial or ambiguous to include the ε in the language, as a run is defined as having 1+ characters.

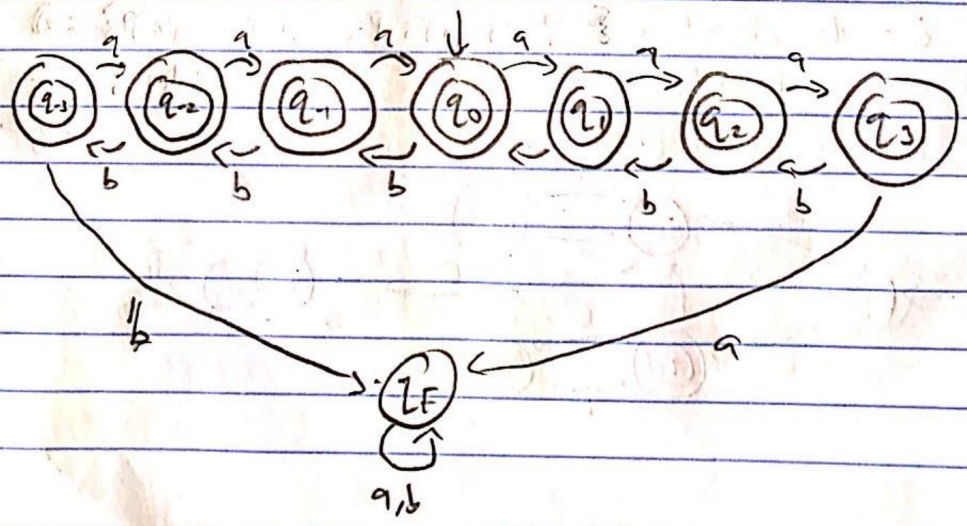
c) $\Sigma' = \{a, b\}$

$L_{4c} = \{w \in \Sigma'^{+} \mid |\#(a,w) - \#(b,w)| < 4\}$

This is not possible to implement as a DFA, as there could be an indeterminate amount of 1 letter before a single occurrance of the other, resulting in a need for an indeterminate number of states to keep count of that letter.

d) $\Sigma' = \{a, b\}$

$L_{4d} = \{w \in \Sigma'^{+} \mid |\#(a,y) - \#(b,y)| < 4, \text{ for all prefixes, } y \text{ of } w\}$



4.2 b 4 / 5

✓ **- 1 pts** **Almost correct**

"aab" should be accepted.

$Q = \{q_0, q_1, q_2, q_3, q_4, q_5, q_6\}$

$\Sigma' = \{a, b\}$

$\delta =$

	a	b
q_0	q_1	q_3
q_1	q_2	q_5
q_2	q_1	q_4
q_3	q_1	q_4
q_4	q_5	q_3
q_5	q_5	q_5

$z_0 = q_0$

$F = \{q_2, q_3\}$

Since the language is defined by runs of a and b, it could be trivial or ambiguous to include the ε in the language, as a run is defined as having 1+ characters.

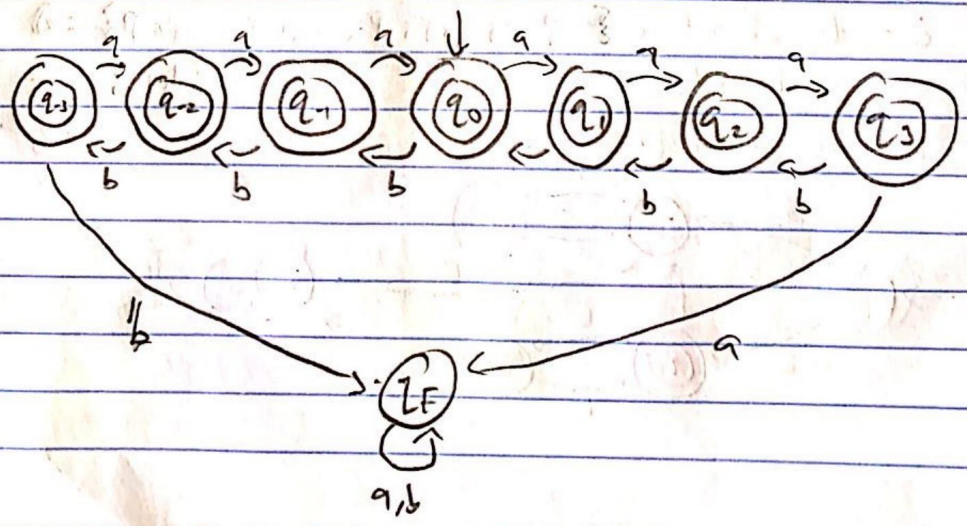
c) $\Sigma' = \{a, b\}$

$L_{4c} = \{w \in \Sigma'^{+} \mid |\#(a,w) - \#(b,w)| < 4\}$

This is not possible to implement as a DFA, as there could be an indeterminate amount of 1 letter before a single occurrance of the other, resulting in a need for an indeterminate number of states to keep count of that letter.

d) $\Sigma' = \{a, b\}$

$L_{4d} = \{w \in \Sigma'^{+} \mid |\#(a,y) - \#(b,y)| < 4, \text{ for all prefixes, } y \text{ of } w\}$



4.3 C **2 / 2**

✓ **- 0 pts** Correct

$Q = \{q_0, q_1, q_2, q_3, q_4, q_5, q_6\}$

$\Sigma' = \{a, b\}$

$\delta =$

	a	b
q_0	q_1	q_3
q_1	q_2	q_5
q_2	q_1	q_4
q_3	q_1	q_4
q_4	q_5	q_3
q_5	q_5	q_5

$z_0 = q_0$

$F = \{q_2, q_3\}$

Since the language is defined by runs of a and b, it could be trivial or ambiguous to include the ε in the language, as a run is defined as having 1+ characters.

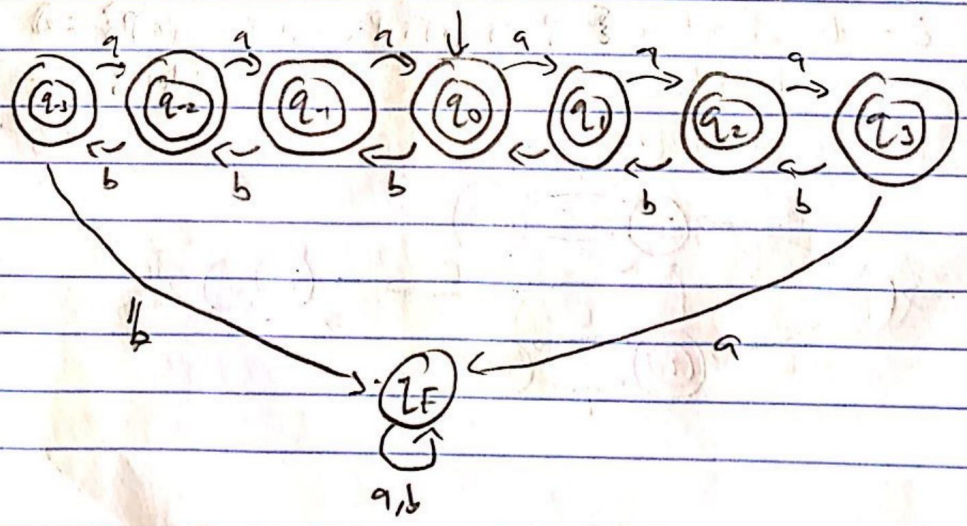
c) $\Sigma' = \{a, b\}$

$L_{4c} = \{w \in \Sigma'^{+} \mid |\#(a,w) - \#(b,w)| < 4\}$

This is not possible to implement as a DFA, as there could be an indeterminate amount of 1 letter before a single occurrance of the other, resulting in a need for an indeterminate number of states to keep count of that letter.

d) $\Sigma' = \{a, b\}$

$L_{4d} = \{w \in \Sigma'^{+} \mid |\#(a,y) - \#(b,y)| < 4, \text{ for all prefixes, } y \text{ of } w\}$



$Q = \{q_{-3}, q_{-2}, q_{-1}, q_0, q_1, q_2, q_3, q_F\}$

$\Sigma' = \{a, b\}$

$\delta =$

	a	b
q_{-3}	q_{-2}	q_F
q_{-2}	q_{-1}	q_{-3}
q_{-1}	q_0	q_{-2}
q_0	q_1	q_{-1}
q_1	q_2	q_0
q_2	q_3	q_1
q_3	q_F	q_2
q_F	q_F	q_F

$q_0 = q_0$

$F = \{q_{-3}, q_{-2}, q_{-1}, q_0, q_1, q_2, q_3\}$

5) Base case: Let $w = \varepsilon$

$|w| = 0$

$\varepsilon = xy$

$x = \varepsilon, y = \varepsilon$

$w^R = y^R x^R$

$\varepsilon^R = \varepsilon^R \varepsilon^R$

$\varepsilon = \varepsilon\varepsilon$

$\varepsilon = \varepsilon$ ✓ → Base case solved

Inductive Step: Assume: $w = xy$ and $(xy)^R = y^R x^R$ for all $|w| \leq n$

Prove: the same holds for all $|w| = n+1$

$w' = xy'$, where $|y'| = |y| + 1$ → prove $w'^R = y'^R x^R$

$w'^R = (xy')^R$

$w'^R = (xyc)^R$, where $c \in \Sigma'$

$w'^R = c(xy)^R$, by definition of reverse

$w'^R = c(y^R x^R)$, by inductive hypothesis

$w'^R = (cy^R)x^R$

$w'^R = y'^R x^R$ ✓ → Inductive step solved

4.4 d **6 / 6**

✓ **- 0 pts** Correct

$Q = \{q_{-3}, q_{-2}, q_{-1}, q_0, q_1, q_2, q_3, q_F\}$

$\Sigma' = \{a, b\}$

$\delta =$

	a	b
q_{-3}	q_{-2}	q_F
q_{-2}	q_{-1}	q_{-3}
q_{-1}	q_0	q_{-2}
q_0	q_1	q_{-1}
q_1	q_2	q_0
q_2	q_3	q_1
q_3	q_F	q_2
q_F	q_F	q_F

$q_0 = q_0$

$F = \{q_{-3}, q_{-2}, q_{-1}, q_0, q_1, q_2, q_3\}$

5) Base case: Let $w = \varepsilon$

$|w| = 0$

$\varepsilon = xy$

$x = \varepsilon, y = \varepsilon$

$w^R = y^R x^R$

$\varepsilon^R = \varepsilon^R \varepsilon^R$

$\varepsilon = \varepsilon\varepsilon$

$\varepsilon = \varepsilon$ ✓ → Base case solved

Inductive Step: Assume: $w = xy$ and $(xy)^R = y^R x^R$ for all $|w| \leq n$

Prove: the same holds for all $|w| = n+1$

$w' = xy'$, where $|y'| = |y| + 1$ → prove $w'^R = y'^R x^R$

$w'^R = (xy')^R$

$w'^R = (xyc)^R$, where $c \in \Sigma'$

$w'^R = c(xy)^R$, by definition of reverse

$w'^R = c(y^R x^R)$, by inductive hypothesis

$w'^R = (cy^R)x^R$

$w'^R = y'^R x^R$ ✓ → Inductive step solved

5 Inductive Proof 5 / 6

✓ **- 1 pts** **Lack of justification in the induction step, we expected a very detailed explanation for every step of your induction.**

6) All sections are numbered from 0-10, with subsections numbered S.x, where S is the section and x is the subsection number (counting from 1). All figures/examples/theorems/etc. are numbered as S.y, where S is the section they're in, and y is the figure/example/theorem/etc. number (using the same counter and starting from 1). Finally, exercises are numbered S.z, where S is the section number and z is the exercise number (counting from 1). Some problems have subproblems, denoted by a), b), c), etc.

6 Explain Sipser's System 1 / 1

✓ **+ 1 pts** **Correct**

+ 0 pts No answer.