Brock Boehler
bvb2@illinois.edu
ECE 374 B Spring 2019
Homework 0

Problem 1

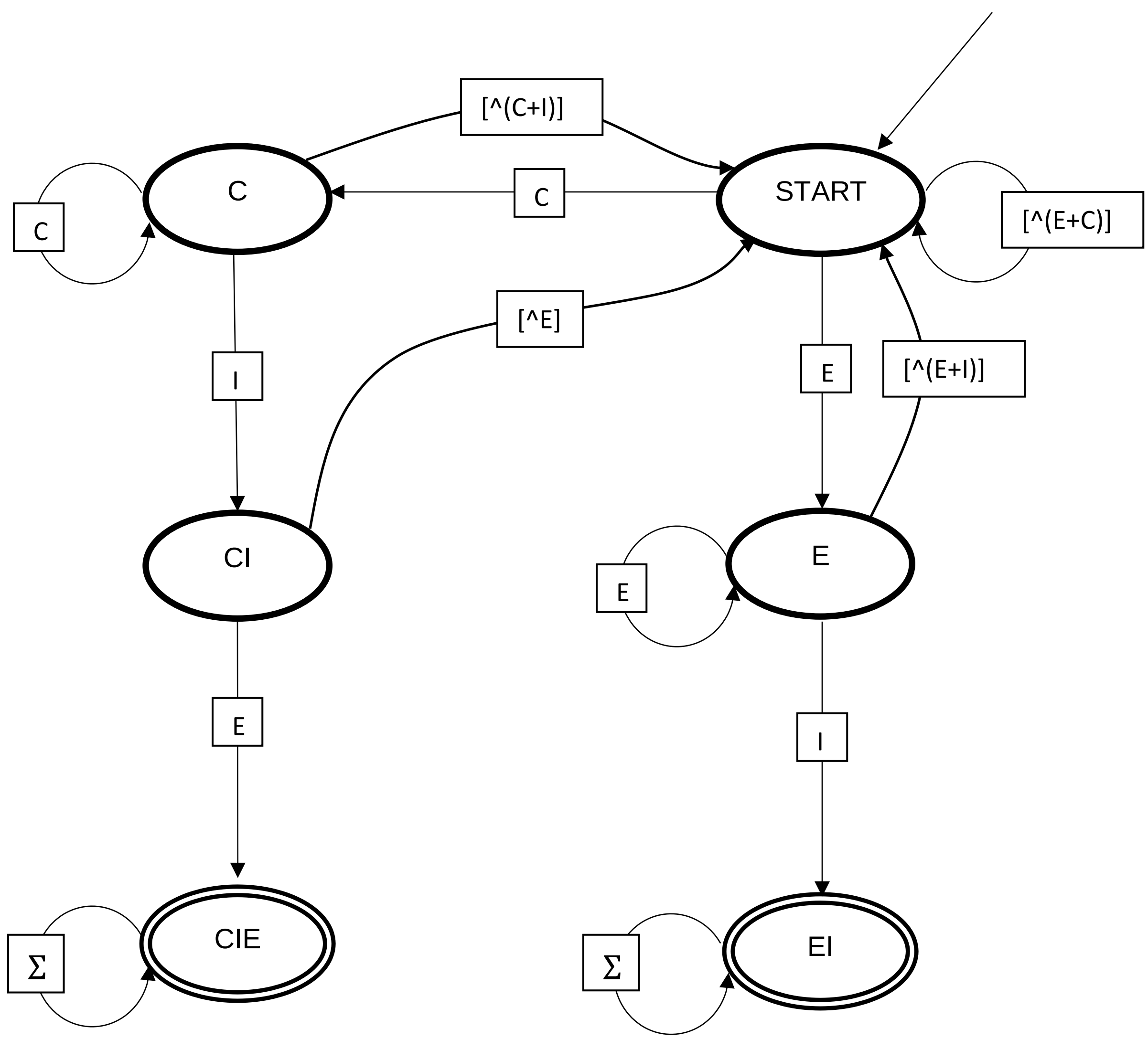
(a) Strings that break "I before E": $\Sigma^*[\hat{}C]('EI')\Sigma^*$

Strings that break "except after C": $\Sigma^*('CIE')\Sigma^*$

All strings that break either rules: $\Sigma^*('CIE')\Sigma^* + \Sigma^*[\hat{}C]('EI')\Sigma^*$

(b)

Notation Note: [^(A + B)] specifies the set of all characters included in Σ not including any given characters A and B

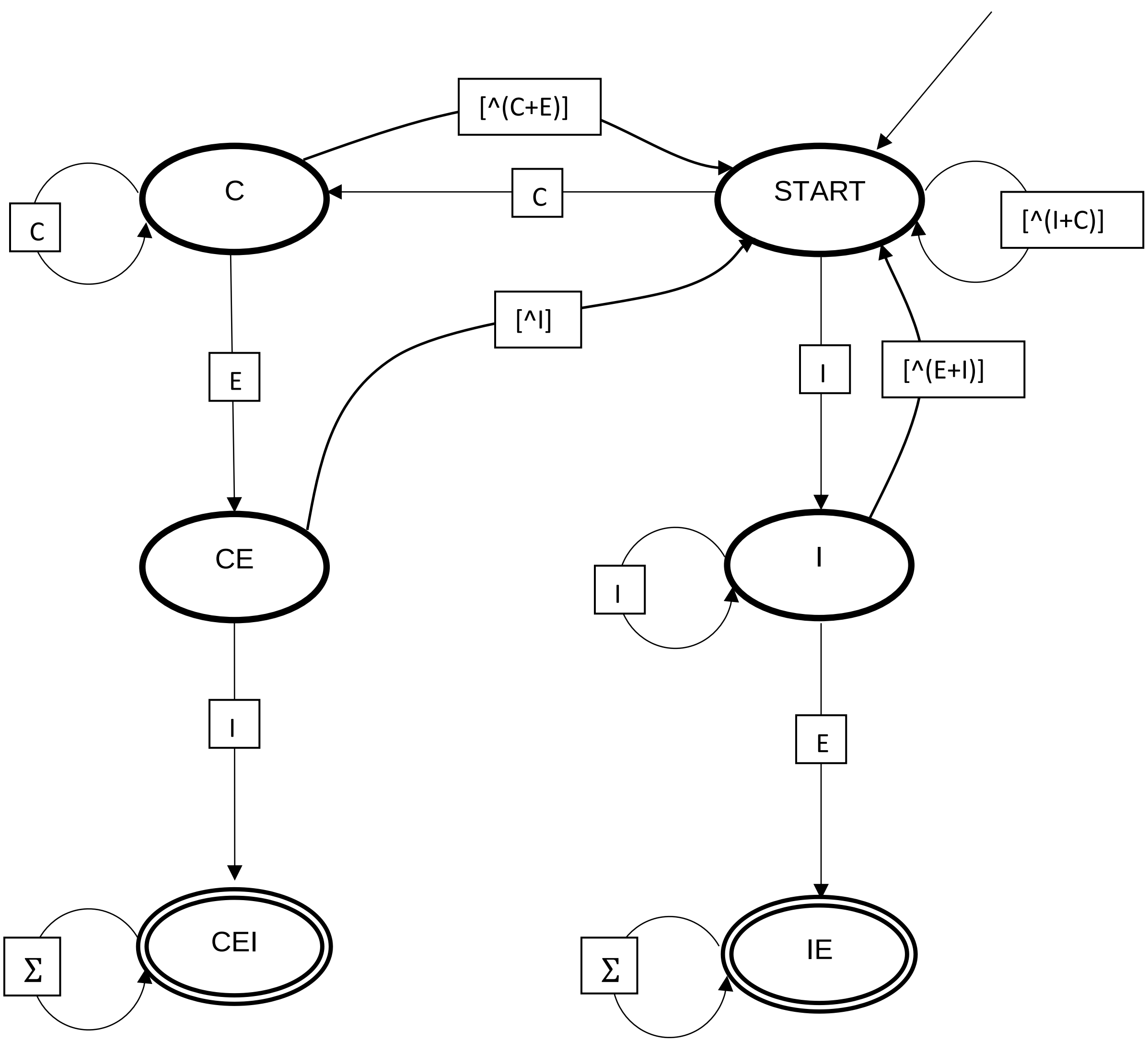


(b) continued:

The states are labeled as follows:

- START – we have not seen anything yet or we not have seen the string to include 'CIE' or 'EI' substrings yet
- C – we have encountered a C
- CI – we have encountered the substring C-I
- CIE – we have encountered the substring C-I-E
- E – we have encountered an E
- EI – we have encountered the substring I-E

(c) Define another DFA that recognizes strings that *do* follow the rule:



Where the states similarly follow:

- START – the DFA has not read anything yet or has not begun to read any substrings resembling CEI or IE
- C – the DFA has read an initial C
- CE – the DFA has read the substring CE
- CEI – the DFA has read the substring CEI and recognized it as following the rule
- I – the DFA has read an initial I
- IE – the DFA has read the substring IE and recognized it as following the rule

Now that we have two DFA's – one for recognizing words that follows the rule and the other one recognizes words that don't, we can use the product construction to create a new DFA the determines whether a word breaks and follows the rules simultaneously. Let M_1 denote the DFA that recognizes words *not* following the rule, and M_2 denote the DFA of words that follows the rule, and M_3 denote the DFA that is the construction of both.

$$Q_3 \coloneqq Q_1 x Q_2$$

$$s_3 \coloneqq (s_1, s_2) = (START, START)$$

$$q \in Q_1, r \in Q_2, a \in \Sigma$$

$$\delta_3\big((q,r),a\big) := \big(\delta_1(\mathrm{q},a),\ \ \delta_2(\mathrm{r},a)\big)$$

$$A_3 = \{(CIE, CEI), (CIE, IE), (EI, IE), (EI, CEI)\}$$

Notation Notes:

Q_n – collection of states found in M_n

s_n – starting state of M_n

$\delta_n(q, \mathrm{a})$ – state transition function of M_n. See the above DFA diagrams for transition functions of M_1 and M_2

A_n – set of accepting states of A_n

As a simple way of explaining the above DFA, imagine that the two DFA's are running in parallel and whenever both states are in an accepting state, that means we have found a word that both violates and follows the rule.

Problem 2

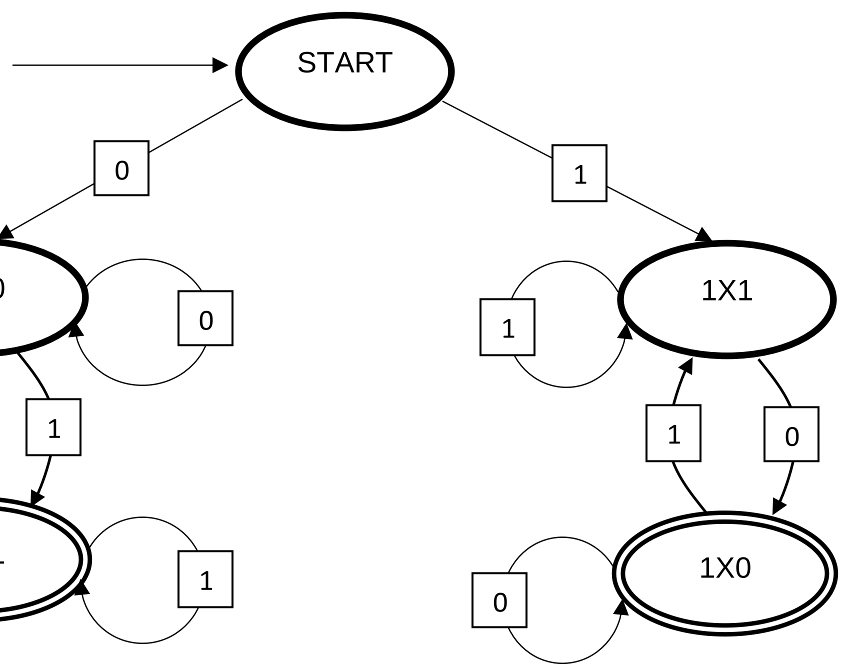
(a) All strings that start and end with a different symbol

$$R = \mathbf{1}\boldsymbol{\Sigma}^{*}\mathbf{0} + \mathbf{0}\boldsymbol{\Sigma}^{*}\mathbf{1}$$

The following DFA decides whether or not a string has different start and end character for a binary alphabet

Explanation of States:

- START – we have not read anything yet
- 0X0 – the string starts and ends with 0
- 0X1 – the string starts with 0 and ends with 1
- 1X1 – the string starts with 1 and ends with 1
- 1X0 – the string starts with 1 and ends with 0

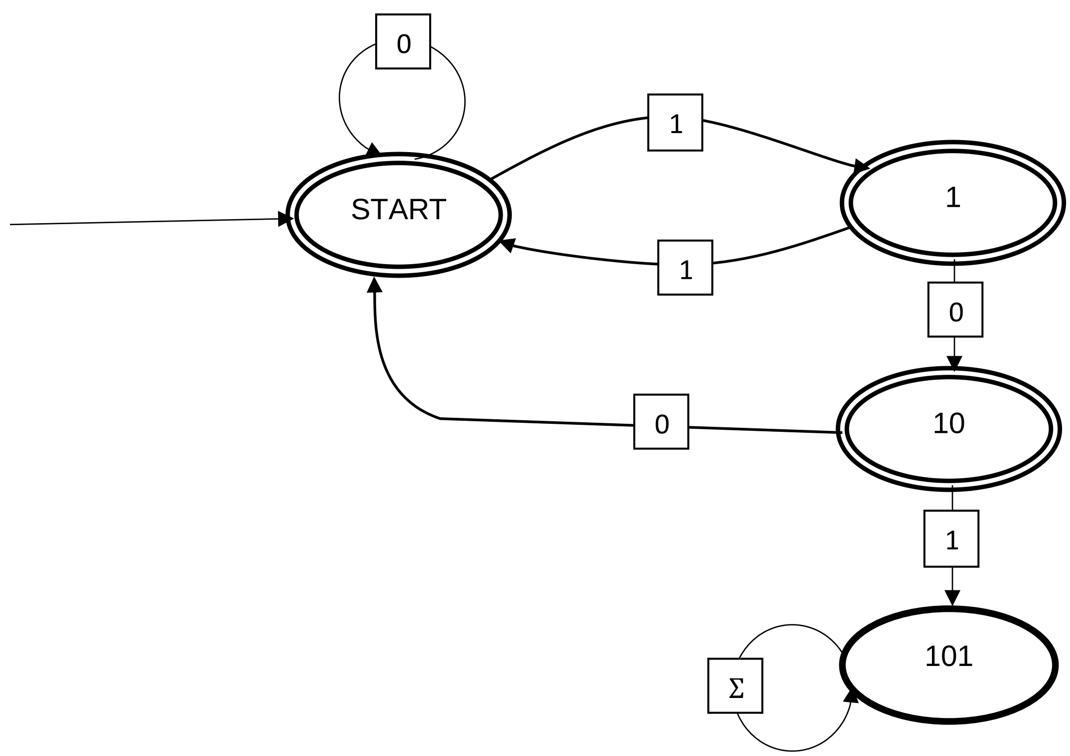


(b) All strings that do not contain the substring 101

$$\boldsymbol{R} = \mathbf{0}^*(1 + 00 + 000)^*\mathbf{0}^*$$

Explanation of States:

- START – we have read nothing or what we have read so far does not contain a 101
- 1 – We have read an initial 1
- 10 – We have read an initial 1 followed by a 0
- 101 – We have read a consecutive 1-0-1 and have located our substring

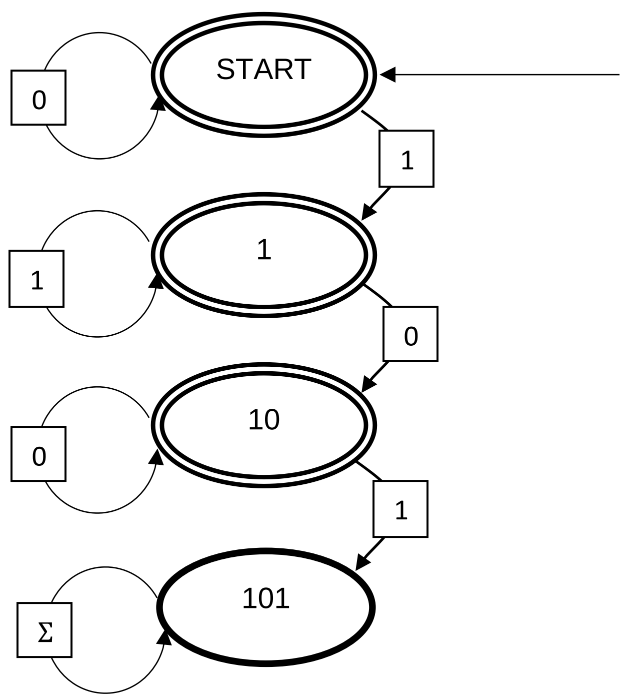


(c) All strings that do not contain the *subsequence* 101

$$\boldsymbol{R} = \mathbf{0}^{*}\mathbf{1}^{*}\mathbf{0}^{*}$$

Explanation of States:

- START – We have not read anything, or we have not read a 1 yet
- 1 – we have read a single 1
- 10 – we have read a 1 and a 0
- 101 – we have read a 1, 0 and another 1



Problem 3

L(M_1) = Language of all strings containing the substring ‘10’

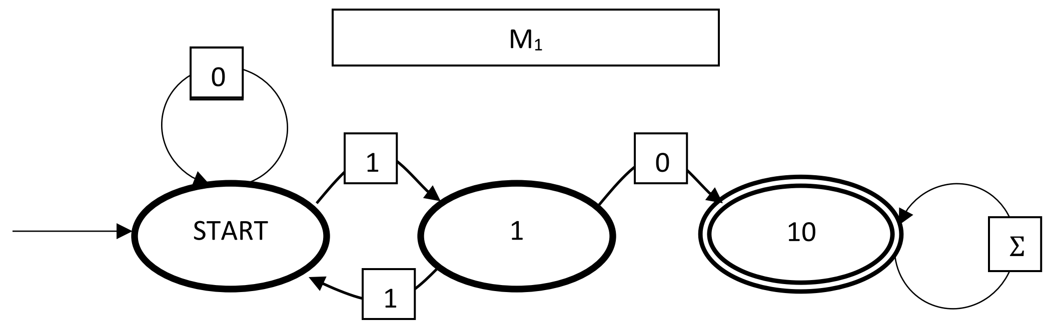
$$\boldsymbol{\Sigma}^*(10)\boldsymbol{\Sigma}^*$$

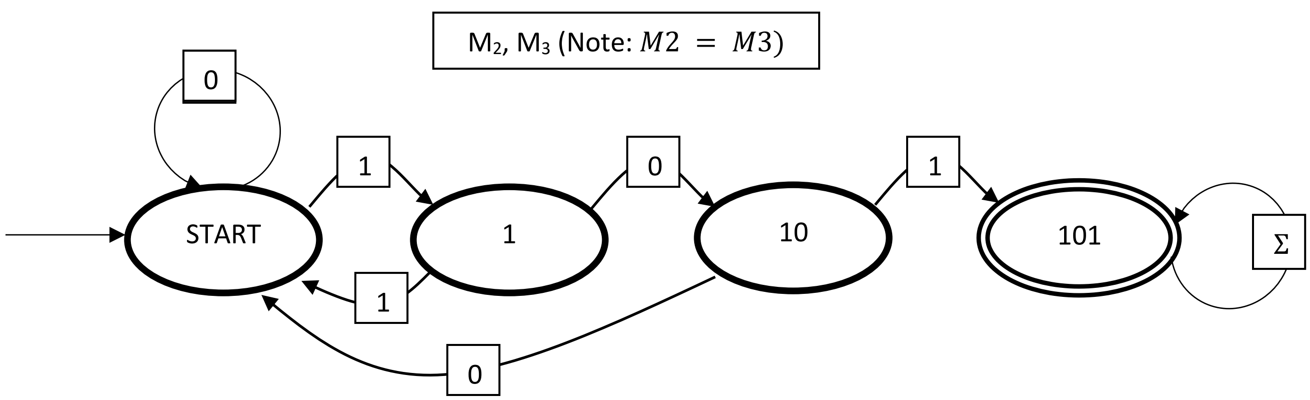
L(M_2) = Language of all strings containing the substring ‘101’

$$\boldsymbol{\Sigma}^*(101)\boldsymbol{\Sigma}^*$$

L(M_3) = $L(M_1) \cap \mathrm{L}(M_2)$ = Language of all strings containing the substring ‘101’

$$\boldsymbol{\Sigma}^*(101)\boldsymbol{\Sigma}^*$$





M_1 Explanation of States:

- START – we have not read anything, or we have not begun reading a '10' substring
- 1 – we have read an initial 1
- 10 – we have read a 1 followed immediately by a 0, finding our substring

M_2 and M_3 Explanation of States:

- START – we have not read anything, or we have not began reading our '101' substring
- 1 – we have read an initial 1
- 10 – we have read a 1, followed by a 0
- 101 – we have read a 1, followed by a 0, followed by a 1, thus finding our substring '101'

Justification:

(a) M_1 has 3 states and M_2 has 4 states as shown by above. This satisfies the condition of $|Q_1| > 2 and |Q_2| > 2$

(b) Both M_1 and M_2 use the minimal states, the languages can no be expressed by a DFA of less states

(c) $L(M_1)$ and $L(M_2)$ describe different languages. See the initial description for what each language / DFA describes, the recognize different things.

(d) $L(M_1)$ recognizes strings that contain '10'. $L(M_2)$ recognizes strings that contains '101'. Because '10' is a substring of '101', any language that contains the substring '101' then also contains the substring '10'. Because of this, $L(M_2)$ is a subset of $L(M_1)$, which means that:

$$L(\mathbf{\Sigma}^*(101)\mathbf{\Sigma}^*) \subset L(\mathbf{\Sigma}^*(10)\mathbf{\Sigma}^*$$

$$\rightarrow L(M_2) \subset L(M_1)$$

$$\rightarrow L(M_2) = L(M_1) \cap (M_2)$$

$$\rightarrow L(M_3) = L(M_2) = L(M_1) \cap L(M_2)$$

(e) $L(M_3)$ is the concatenation of Σ^* and '101', and because $|\Sigma^*| = \infty$, this means that $|L(M_3)| = |\Sigma^* 101 \Sigma^*| = \infty$

(f) $|Q_3| < |Q_1| x |Q_2|$
$4 < 4 * 3$
$4 < 12$