MTH 222 Theory of Computation

First Mid Semester Examination (Exercise set A)

Total marks: 25 | September 3, 2002 | Time: $1+\epsilon$ hours

---

Name: **Roll Number:**

---

1. Which of the following statements are true? (Write True/False in the box provided.) **(5)**

(a) $\mathcal{L}(((ab)^*(ba)^*) \cap ((ba)^*(ab)^*)) = \{\epsilon\}$. False

(b) $abcd \in \mathcal{L}((d^*c^*b^*a^*)^*)$. True

(c) If L_1 and L_2 are regular languages (over some alphabet Σ), then so is their symmetric difference $(L_1 \setminus L_2) \cup (L_2 \setminus L_1)$. True

(d) The language of a DFA (over Σ) in which the only final state is the start state is $\{\epsilon\}$. False

(e) The language $\{\alpha \in \{a,b\}^* \mid \alpha \text{ contains an odd number of } b\text{'s}\}$ is regular. True

---

2. Let $\Sigma := \{a, b, \ldots, z\}$ be the Roman alphabet and L the 'English' language, i.e., the (finite) language of all valid English words over Σ. Does there exist a DFA D with 20 states such that $\mathcal{L}(D) = L$? (**Remark:** If you are stupefied by the incomprehensibleness of this exercise, ask for DDT to kill rodents.) **(5)**

Solution There can not exist a DFA with 20 (or less) states that accepts L. To prove this assertion assume that there exists such a DFA D. The Merriam-Webster dictionary lists the following 20-letter allowed English words:

```
compartmentalization
conventionalizations
counterrevolutionary
departmentalizations
electroencephalogram
incomprehensibleness
indistinguishability
institutionalization
intellectualizations
microminiaturization
```

Since each such word is a member of L, the pumping lemma for regular languages gives us strings $\beta_1, \beta_2, \beta_3 \in \Sigma^*$ with $|\beta_2| \geqslant 1$ such that $\beta_1 \beta_2^k \beta_3 \in L$ for all $k \in \mathbb{Z}_+$. This means that L is infinite, a contradiction. •

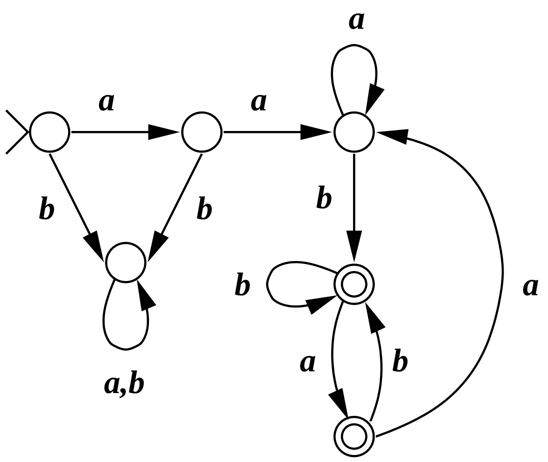
3. Let $\Sigma := \{a, b\}$ and $L := \{\alpha \in \Sigma^* \mid \alpha$ starts with aa but does not end with $aa\}$.

(a) Write a regular expression (over Σ) to represent L. **(2)**

Solution $aa(a \cup b)^* b(a \cup \epsilon)$. •

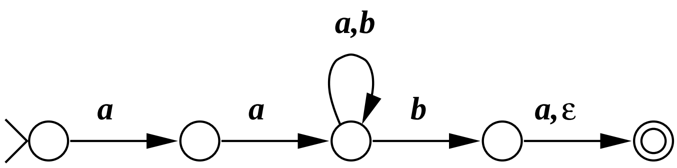
(b) Design a DFA whose language is L. **(4)**

Solution



(c) Design an NFA (or ϵ-NFA) whose language is L. **(3)**

Solution



4. Let Σ be a given alphabet. An extended regular expression (ERE) over Σ is a string α over $\Sigma \uplus \{\emptyset, \epsilon, (,), \cup, ^*, ^+, ?, !, \cdot\}$ defined inductively as follows:

(1) ϕ, ϵ and a are ERE's (for each $a \in \Sigma$).

(2) If α is an ERE, then so is (α).

(3) If α and β are ERE's, then so is $\alpha\beta$.

(4) If α and β are ERE's, then so is $\alpha \cup \beta$.

(5) If α is an ERE, then so is α^*.

(6) If α is an ERE, then so is α^+.

(7) If α is an ERE, then so is $\alpha?$.

(8) If α is an ERE, then so is $!\alpha$.

(9) $\cdot$ is an ERE.

(10) Nothing is an ERE unless it follows from (1)–(9) above.

Rules (1)–(5) bear the same meanings as for RE's. The informal meanings for (6)–(9) are as follows:

α^+ means one or more occurrence(s) of α.

$\alpha?$ means 0 or 1 occurrence of α.

$!\alpha$ means $\beta \in \Sigma^*$ belongs to the language of $!\alpha$, if and only if $\beta \notin \mathcal{L}(\alpha)$.

$\cdot$ means the single occurrence of any element of Σ.

(a) Formally extend the definition of $\mathcal{L}$ for ERE's, i.e., for ERE's α and β and $a \in \Sigma$ complete the following definitions: **(4)**

$\mathcal{L}(\phi) := \phi$. $\mathcal{L}(\epsilon) := \{\epsilon\}$. $\mathcal{L}(a) := \{a\}$.

$\mathcal{L}((\alpha)) := \mathcal{L}(\alpha)$.

$\mathcal{L}(\alpha\beta) := \mathcal{L}(\alpha)\mathcal{L}(\beta)$.

$\mathcal{L}(\alpha \cup \beta) := \mathcal{L}(\alpha) \cup \mathcal{L}(\beta)$.

$\mathcal{L}(\alpha^*) := (\mathcal{L}(\alpha))^*$.

$\boxed{\mathcal{L}(\alpha^+) := (\mathcal{L}(\alpha))(\mathcal{L}(\alpha))^*.}$

$\boxed{\mathcal{L}(\alpha?) := \mathcal{L}(\alpha) \cup \{\epsilon\}.}$

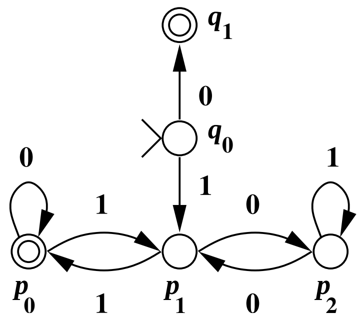
$\boxed{\mathcal{L}(!\alpha) := \Sigma^* \setminus \mathcal{L}(\alpha).}$

$\boxed{\mathcal{L}(\cdot) := \Sigma.}$

(b) Let $\Sigma := \{a, b\}$. Find a regular expression (over Σ) whose language is the same as the language of the ERE $aa(\cdot^*)(!a)(a?)$. **(2)**

Solution $aa(a \cup b)^*(\epsilon \cup b \cup (a \cup b)(a \cup b)(a \cup b)^*)(a \cup \epsilon)$. This simplifies to $aa(a \cup b)^*$. Note that by our definition $\mathcal{L}(!a)$ is $\Sigma^* \setminus \{a\}$ and not $\Sigma \setminus \{a\}$. •

5. [Bonus problem] Give an informal description of the language accepted by the following NFA. **(10)**



Solution The above NFA – call it N – accepts valid binary representations of all non-negative integer multiples of 3. In order to see how let us name the states as above. The automaton is at the start state q_0, if and only if it has not consumed any input. But the empty string is not a valid binary representation of any non-negative integer. So q_0 is not a final state. If N reads 0 at the very beginning, it accepts the input string, if and only if no further symbols appear. Thus the integer 0 (a multiple of 3) is accepted in state q_1, whereas 0 at the beginning followed by any non-empty string causes the automaton to go to the 'stuck' position.

Now assume that N reads a 1 at the beginning. This means that N is handling a positive integer. In this case the automaton subsequently remains in the states p_0, p_1 and p_2, where being in p_i indicates that the string read so far represents a positive integer of the form $3k + i$ (for some $k \in \mathbb{Z}_+$). When N is in state p_i and reads $a \in \{0, 1\}$, it moves to the state p_j, where $j = (2(3k + i) + a) \operatorname{rem} 3 = (2i + a) \operatorname{rem} 3$, where $c \operatorname{rem} 3$ denotes the remainder of division of c by 3. The indicated transitions can be easily verified to satisfy this formula. •

MTH 222 Theory of Computation

First Mid Semester Examination (Exercise set B)

Total marks: 25 | September 3, 2002 | Time: $1 + \epsilon$ hours

---

Name: **Roll Number:**

---

1. Which of the following statements are true? (Write True/False in the box provided.) **(5)**

(a) The language of a DFA (over Σ) in which every state except the start state is final is Σ^+. False

(b) $\mathcal{L}((ab^*ba^*) \cap (ba^*ab^*)) = \{\epsilon\}$. False

(c) $abcd \in \mathcal{L}((b^*a^*)^*(d^*c^*)^*)$. True

(d) The language $\{\alpha \in \{a,b\}^* \mid \alpha \text{ contains an even number of } a\text{'s}\}$ is regular. True

(e) If L_1 and L_2 are regular languages (over some alphabet Σ), then so is their exclusive or $(L_1 \cup L_2) \setminus (L_1 \cap L_2)$. True

---

2. Let L be a finite language over the binary alphabet $\{0,1\}$. Assume that $|L| = m$. Let D be a DFA with n states such that $\mathcal{L}(D) = L$. Show that $n \geqslant \log_2(m+1)$. **(5)**

Solution Let l be the length of the longest string in L. Since L is given to be finite, l is finite too. We also have $m \leqslant 1 + 2 + 2^2 + \cdots + 2^l = 2^{l+1} - 1$, i.e., $l + 1 \geqslant \log_2(m+1)$. If $\alpha \in L$ is of length l and if $l \geqslant n$, then by the pumping lemma we have strings $\beta_1, \beta_2, \beta_3 \in \Sigma^*$ such that $|\beta_2| \geqslant 1$ and $\beta_1\beta_2^k\beta_3 \in L$ for all $k \in \mathbb{Z}_+$ implying that L is infinite, a contradiction. Thus $l < n$, i.e., $n \geqslant l + 1 \geqslant \log_2(m+1)$. •

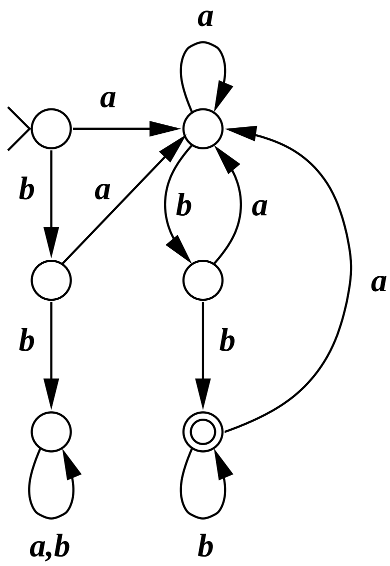
3. Let $\Sigma := \{a, b\}$ and $L := \{\alpha \in \Sigma^* \mid \alpha$ ends with bb but does not start with $bb\}$.

(a) Write a regular expression (over Σ) to represent L. **(2)**

Solution $(b \cup \epsilon)a(a \cup b)^*bb$. •

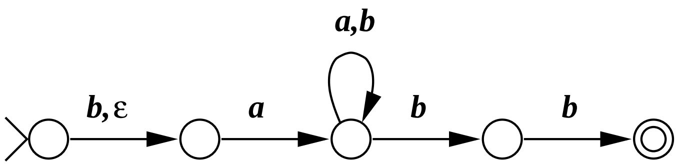
(b) Design a DFA whose language is L. **(4)**

Solution



(c) Design an NFA (or ϵ-NFA) whose language is L. **(3)**

Solution



4. Let Σ be a given alphabet. An *extended regular expression* (ERE) over Σ is a string α over $\Sigma \uplus \{\emptyset, \epsilon, (,), \cup, ^*, ^+, ?, !, \cdot\}$ defined inductively as follows:

(1) ϕ, ϵ and a are ERE's (for each $a \in \Sigma$).

(2) If α is an ERE, then so is (α).

(3) If α and β are ERE's, then so is $\alpha\beta$.

(4) If α and β are ERE's, then so is $\alpha \cup \beta$.

(5) If α is an ERE, then so is α^*.

(6) $\cdot$ is an ERE.

(7) If α is an ERE, then so is $\alpha?$.

(8) If α is an ERE, then so is α^+.

(9) If α is an ERE, then so is $!\alpha$.

(10) Nothing is an ERE unless it follows from (1)–(9) above.

Rules (1)–(5) bear the same meanings as for RE's. The informal meanings for (6)–(9) are as follows:

$\cdot$ means the single occurrence of any element of Σ.

$\alpha?$ means 0 or 1 occurrence of α.

α^+ means one or more occurrence(s) of α.

$!\alpha$ means $\beta \in \Sigma^*$ belongs to the language of $!\alpha$, if and only if $\beta \notin \mathcal{L}(\alpha)$.

(a) Formally extend the definition of $\mathcal{L}$ for ERE's, i.e., for ERE's α and β and $a \in \Sigma$ complete the following definitions: **(4)**

$\mathcal{L}(\phi) := \phi$. $\mathcal{L}(\epsilon) := \{\epsilon\}$. $\mathcal{L}(a) := \{a\}$.

$\mathcal{L}((\alpha)) := \mathcal{L}(\alpha)$.

$\mathcal{L}(\alpha\beta) := \mathcal{L}(\alpha)\mathcal{L}(\beta)$.

$\mathcal{L}(\alpha \cup \beta) := \mathcal{L}(\alpha) \cup \mathcal{L}(\beta)$.

$\mathcal{L}(\alpha^*) := (\mathcal{L}(\alpha))^*$.

$\boxed{\mathcal{L}(\cdot) := \Sigma\,.}$

$\boxed{\mathcal{L}(\alpha?) := \mathcal{L}(\alpha) \cup \{\epsilon\}\,.}$

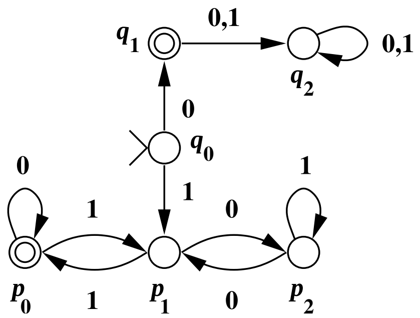
$\boxed{\mathcal{L}(\alpha^+) := (\mathcal{L}(\alpha))(\mathcal{L}(\alpha))^*.}$

$\boxed{\mathcal{L}(!\alpha) := \Sigma^* \setminus \mathcal{L}(\alpha).}$

(b) Let $\Sigma := \{a, b\}$. Find a regular expression (over Σ) whose language is the same as the language of the ERE $(b?)(!b)(\cdot^*)bb$. **(2)**

Solution $(b \cup \epsilon)(\epsilon \cup a \cup (a \cup b)(a \cup b)(a \cup b)^*)(a \cup b)^*bb$. This simplifies to $(a \cup b)^*bb$. Note that by our definition $\mathcal{L}(!b)$ is $\Sigma^* \setminus \{b\}$ and not $\Sigma \setminus \{b\}$. •

5. [Bonus problem] Give an informal description of the language accepted by the following DFA. **(10)**



Solution The above DFA – call it D – accepts valid binary representations of all non-negative integer multiples of 3. In order to see how let us name the states as above. The automaton is at the start state q_0, if and only if it has not consumed any input. But the empty string is not a valid binary representation of any non-negative integer. So q_0 is not a final state. If D reads 0 at the very beginning, it accepts the input string, if and only if no further symbols appear. Thus the integer 0 (a multiple of 3) is accepted in state q_1, whereas 0 at the beginning followed by any non-empty string causes the automaton to go to (and remain in) the non-final state q_2.

Now assume that D reads a 1 at the beginning. This means that D is handling a positive integer. In this case the automaton subsequently remains in the states p_0, p_1 and p_2, where being in p_i indicates that the string read so far represents a positive integer of the form $3k + i$ (for some $k \in \mathbb{Z}_+$). When D is in state p_i and reads $a \in \{0, 1\}$, it moves to the state p_j, where $j = (2(3k + i) + a) \operatorname{rem} 3 = (2i + a) \operatorname{rem} 3$, where $c \operatorname{rem} 3$ denotes the remainder of division of c by 3. The indicated transitions can be easily verified to satisfy this formula. •