PMA1050 Discrete Foundations 2004–5
Question Sheet 3

Please complete this by Friday 27 October, when the solutions will be given. The starred questions are for COM6853 students.

1. Recall that a relation R on a set A is said to be:
   - reflexive if xRx for every $x \in A$;
   - symmetric if xRy implies yRx for every $x, y \in A$;
   - transitive if xRy and yRz imply xRz for every $x, y, z \in A$.

   Let A be the power set of $\mathbb{R}$, i.e. the set of all sets of real numbers, and consider the relation $\subseteq$ on A. Is this relation (a) reflexive, (b) symmetric, (c) transitive?

2. Prove directly that
$$\frac{200}{700^2} < \frac{1}{500} - \frac{1}{700} < \frac{200}{500^2}$$

3. Prove that
$$\frac{x(x-1)^2 - 5x + 8}{x+2} = (x-2)^2,$$
   when $x \neq -2$. What happens at $x = -2$?

4. Is the following proof correct ? If not, why not?

   Theorem $1 = 2$.

   Proof
$$\begin{aligned} x = 2 &\Rightarrow x - 1 = 1 \\ &\Rightarrow (x-1)^2 = 1 \\ &\Rightarrow (x-1)^2 = x - 1 \\ &\Rightarrow x^2 - 2x + 1 = x - 1 \\ &\Rightarrow x^2 - 2x = x - 2 \\ &\Rightarrow x(x-2) = x - 2 \\ &\Rightarrow \frac{x(x-2)}{x-2} = \frac{(x-2)}{x-2} \\ &\Rightarrow x = 1 \end{aligned}$$

5. Prove by induction that
$$\sum_{k=1}^{n} k^2 = \frac{n(n+1)(2n+1)}{6}.$$
   * How could you use this result to calculate $\sum_{k=1}^{n} k^3$? Hint: consider
$$\sum_{k=1}^{n} (k+1)^4 - \sum_{k=1}^{n} k^4.$$

6. * Prove by induction that $n^3 < 3^n$ for $n \geqslant 4$.

7. Prove by induction that $11^n - 4^n$ is divisible by 7 for all $n \in \mathbb{N}$.

8. Prove without differentiation that $x + \frac{1}{x} \geqslant 2$ when $x > 0$. What happens at $x = 0$?

9. Prove by exhaustion that $x^2 + y^2 = 11^2$ has no integer solution with $x, y > 0$.

10. Prove by contradiction that $\sqrt{2}$ is irrational — i.e. that $\sqrt{2}$ cannot be expressed in the form p/q for positive integers p, q. You may use without proof the fact that, for an integer a, if a^2 is even, then a is even.

PMA1050 Discrete Foundations 2004–5
Question Sheet 3: Solutions

1. *Recall that a relation R on a set A is said to be:*

   - *reflexive if xRx for every $x \in A$;*
   - *symmetric if xRy implies yRx for every $x, y \in A$;*
   - *transitive if xRy and yRz imply xRz for every $x, y, z \in A$.*

   Let A be the power set of $\mathbb{R}$, i.e. the set of all sets of real numbers, and consider the relation $\subseteq$ on A. Is this relation (a) reflexive, (b) symmetric, (c) transitive?

   The relation $\subseteq$ is reflexive (since $x \subseteq x$ for all x), and transitive (if $x \subseteq y$ and $y \subseteq z$ then $x \subseteq z$). It is not symmetric: for example if x is any non-empty subset of $\mathbb{R}$, then $\emptyset \subseteq x$ but $x \not\subseteq \emptyset$.

2. *Prove directly that*

$$\frac{200}{700^2} < \frac{1}{500} - \frac{1}{700} < \frac{200}{500^2}.$$

   Consider first the upper bound.

$$\frac{1}{500} - \frac{1}{700} = \frac{200}{35 \times 10^4} < \frac{200}{25 \times 10^4}$$

   Consider now the lower bound.

$$\frac{1}{500} - \frac{1}{700} = \frac{200}{35 \times 10^4} > \frac{200}{49 \times 10^4}$$

3. *Prove that*

$$\frac{x(x-1)^2 - 5x + 8}{x+2} = (x-2)^2,$$

   when $x \neq -2$. What happens at $x = -2$?

   By direct expansion,

$$x(x-1)^2 - 5x + 8 = (x+2)(x-2)^2.$$

   Provided $x \neq -2$, we can divide by $(x+2)$. When $x = -2$, this is invalid; in this case, the left hand side of the desired formula reduces to $0/0$, which is meaningless.

4. *Is the following proof correct ? If not, why not?*

   Theorem *1 = 2.*

   Proof

$$\begin{aligned}
x = 2 &\Rightarrow x - 1 = 1 \\
&\Rightarrow (x-1)^2 = 1 \\
&\Rightarrow (x-1)^2 = x - 1 \\
&\Rightarrow x^2 - 2x + 1 = x - 1 \\
&\Rightarrow x^2 - 2x = x - 2 \\
&\Rightarrow x(x-2) = x - 2 \\
&\Rightarrow \frac{x(x-2)}{x-2} = \frac{(x-2)}{x-2} \\
&\Rightarrow x = 1
\end{aligned}$$

   The error is in the penultimate line. It is assumed that $x = 2$, so division by $x - 2$ is division by 0, which is invalid.

5. *Prove by induction that*

$$\sum_{k=1}^{n} k^2 = \frac{n(n+1)(2n+1)}{6}.$$

** How could you use this result to calculate $\sum_{k=1}^{n} k^3$? Hint: consider*

$$\sum_{k=1}^{n}(k+1)^4 - \sum_{k=1}^{n} k^4.$$

Induction base. For $n = 1$ both sides reduce immediately to 1, so the result holds for $n = 1$.

Induction step. We now assume that the theorem is true for a given arbitrary value of n, and show that it is true for $n + 1$. Thus we assume:

Induction hypothesis:

$$\sum_{k=1}^{n} k^2 = \frac{n(n+1)(2n+1)}{6}.$$

Then

$$\begin{aligned}
\sum_{k=1}^{n+1} k^2 &= \frac{n(n+1)(2n+1)}{6} + (n+1)^2, \text{ by the induction hypothesis,} \\
&= \frac{(n+1)(n(2n+1)+6(n+1))}{6} \\
&= \frac{(n+1)(2n^2+7n+6)}{6} \\
&= \frac{(n+1)(n+2)(2n+3)}{6}
\end{aligned}$$

which is the original formula with $(n + 1)$ in place of n. This completes the induction step.

Conclusion: By mathematical induction, the result holds for all $n \geqslant 1$.

*Note that

$$(k+1)^4 = k^4 + 4k^3 + 6k^2 + 4k + 1$$

and thus

$$\sum_{k=1}^{n}(k+1)^4 = \sum_{k=1}^{n} k^4 + 4\sum_{k=1}^{n} k^3 + 6\sum_{k=1}^{n} k^2 + 4\sum_{k=1}^{n} k + n$$

Thus

$$\sum_{k=1}^{n}(k+1)^4 - \sum_{k=1}^{n} k^4 = 4\sum_{k=1}^{n} k^3 + 6\sum_{k=1}^{n} k^2 + 4\sum_{k=1}^{n} k + n$$

or

$$(n+1)^4 - 1 = 4\sum_{k=1}^{n} k^3 + 6\sum_{k=1}^{n} k^2 + 4\sum_{k=1}^{n} k + n$$

Therefore

$$\begin{aligned}
4\sum_{k=1}^{n} k^3 &= (n+1)^4 - 1 - 6\sum_{k=1}^{n} k^2 - 4\sum_{k=1}^{n} k - n \\
&= (n+1)^4 - n(n+1)(2n+1) - 2n(n+1) - (n+1) \\
&= (n+1)\left((n+1)^3 - n(2n+1) - 2n - 1\right) \\
&= (n+1)\left((n+1)^3 - (n+1)(2n+1)\right) \\
&= (n+1)^2\left((n+1)^2 - (2n+1)\right) \\
&= (n+1)^2 n^2,
\end{aligned}$$

so

$$\sum_{k=1}^{n} k^3 = \frac{n^2(n+1)^2}{4} = \left(\frac{n(n+1)}{2}\right)^2.$$

6. * *Prove by induction that* $n^3 < 3^n$ *for* $n \geqslant 4$.

   Induction base. For $n = 4$, we have $n^3 = 64 < 81 = 3^n$.

   Induction step. We now assume that the theorem is true for a given arbitrary value of n, and show that it is true for $n + 1$. Thus we assume:

   Induction hypothesis: $n^3 < 3^n$.

   We have, for $n \geqslant 4$,

$$\begin{aligned}
(n+1)^3 &= n^3 + 3n^2 + 3n + 1 \\
&< n^3 + 3n^2 + 3n^2 + n^2, \text{ since } n > 1, \\
&< n^3 + 8n^2 \\
&\leqslant n^3 + 2n^3, \text{ since } n \geqslant 4, \\
&= 3n^3 \\
&< 3.3^n, \text{ by the induction hypothesis,} \\
&= 3^{n+1}.
\end{aligned}$$

   This completes the induction step.

   Conclusion: By mathematical induction, the result holds for all $n \geqslant 4$.

7. *Prove by induction that* $11^n - 4^n$ *is divisible by* 7 *for all* $n \in \mathbb{N}$.

   Define $S(n) = 11^n - 4^n$; we show that $S(n)$ is divisible by 7 for all $n \geqslant 0$.

   Induction base. For $n = 0$, we have $S(n) = 0$ which is trivially divisible by 7.

   Induction step. We now assume that the theorem is true for a given arbitrary value of n, and show that it is true for $n + 1$. Thus we assume:

   Induction hypothesis: $S(n)$ is divisible by 7.

   Then

$$\begin{aligned}
S(n+1) &= 11^{n+1} - 4^{n+1} \\
&= 11^n \times 11 - 4^n \times 4 \\
&= 11^n(7+4) - 4^n \times 4 \\
&= 4(11^n - 4^n) + 7 \times 11^n \\
&= 4 \times S(n) + 7 \times 11^n.
\end{aligned}$$

   Hence, using the induction hypothesis, $S(n + 1)$ is divisible by 7. This completes the induction step.

   Conclusion: By mathematical induction, the result holds for all $n \geqslant 0$.

8. *Prove without differentiation that* $x + \frac{1}{x} \geqslant 2$ *when* $x > 0$. *What happens at* $x = 0$ *?*

   Since $(x-1)^2 > 0$, it follows that $x^2 - 2x + 1 > 0$. By assumption, $x > 0$, and division by x yields the required result.

9. *Prove by exhaustion that* $x^2 + y^2 = 11^2$ *has no integer solution with* $x, y > 0$. We need only consider the integer values of x and y such that $0 < x < 11, y^2 = 121 - x^2$. There are no integer values of x and y that satisfy these equations.

10. *Prove by contradiction that $\sqrt{2}$ is irrational — i.e. that $\sqrt{2}$ cannot be expressed in the form p/q for positive integers p, q. You may use without proof the fact that, for an integer a, if a^2 is even, then a is even.* Assume that there exist two integers p and q such that $x^2 = 2$ where $x = \frac{p}{q}$. It may be assumed that p and q have no common factors since they may be removed if they were present.

    We have $p^2 = 2q^2$, and thus p^2 is even, and hence p is even. It follows that there exists an integer k such that $p = 2k$, and thus $(2k)^2 = 2q^2$, or $q^2 = 2k^2$. It follows that q^2 is even, and thus q is also even. We conclude that both p and q are even, thus contradicting our earlier assumption that p and q have no common factors.

PMA1050 Discrete Foundations 2004–5
Question Sheet 4

Please complete this by Friday 29 October, when the solutions will be given. The starred question is for COM6853 students.

1. Consider the sequence $SEQ = \{1, 3, 9, 27, 81 \ldots\}$.
    (a) Give a simple formula for the nth term $SEQ(n)$ of a sequence which fits these first 5 terms, (i.e. $SEQ(0) = 1$, $SEQ(1) = 3$, ...).
    (b) Give a recursive definition for the sequence SEQ.
2. Is the following a valid recursive definition for a sequence SEQ ? Explain.
    (**B**) $SEQ(0) = 1$;
    (**R**) $SEQ(n+1) = SEQ(n)/(100 - n)$ for $n \geqslant 0$.
3. We define a sequence (a_n) recursively by:
    (**B**) $a_0 = a_1 = 1$;
    (**R**) $a_n = a_{n-1} + 2a_{n-2}$ for $n \geqslant 2$.
    (a) Calculate a_6 recursively.
    (b) Prove that all the terms a_n are odd integers. [Note that since (a_n) is defined recursively, the natural way to prove facts about the a_n is by induction.]
4. Consider the Fibonacci sequence $f_n = f_{n-1} + f_{n-2}$ where $f_1 = f_2 = 1$. Use induction to prove that for all $n \geqslant 1$,
$$f_1 + f_2 + \cdots + f_n = f_{n+2} - 1$$
5. * Consider the infinite sequence defined by $a_n = 3a_{n-1} - 2a_{n-2}$ for $n \geqslant 3$, where $a_1 = 6$ and $a_2 = 11$. Prove that for all $n \geqslant 1$,
$$a_n = \left(5 \times 2^{n-1}\right) + 1.$$
6. Construct the truth table for $[(p \vee q) \wedge r] \;\Rightarrow\; (p \wedge \neg q)$.
7. Construct the truth tables for $\neg(p \vee q) \;\Rightarrow\; r$ and $\neg((p \vee q) \;\Rightarrow\; r)$.

PMA1050 Discrete Foundations 2004–5
Question Sheet 4: Solutions

1. *Consider the sequence* $SEQ = \{1, 3, 9, 27, 81 \ldots\}$.

   (a) Give a simple formula for the nth term $SEQ(n)$ *of a sequence which fits these first 5 terms, (i.e.* $SEQ(0) = 1$, $SEQ(1) = 3$, *...).*

   (b) Give a recursive definition for the sequence SEQ.

   (a) $SEQ(n) = 3^n$. This is, of course, just one formula which fits the given five terms, but it is probably the simplest.

   (b) $SEQ(0) = 1$, $SEQ(n+1) = 3 \times SEQ(n)$ for $n \geqslant 0$.

2. *Is the following a valid recursive definition for a sequence SEQ ? Explain.*

   (B) $SEQ(0) = 1$*;*
   (R) $SEQ(n+1) = SEQ(n)/(100 - n)$ *for* $n \geqslant 0$.

   No. The sequence breaks down at $SEQ(101) = SEQ(100)/0$.

3. *We define a sequence* (a_n) *recursively by:*
   (B) $a_0 = a_1 = 1$*;*
   (R) $a_n = a_{n-1} + 2a_{n-2}$ *for* $n \geqslant 2$.

   (a) Calculate a_6 *recursively.*

   (b) Prove that all the terms a_n *are odd integers. [Note that since* (a_n) *is defined recursively, the natural way to prove facts about the* a_n *is by induction.]*

   (a) $a_6 = a_5 + 2a_4 = a_4 + 2a_3 + 2a_4 = 3(a_3 + 2a_2) + 2a_3 = 5(a_2 + 2a_1) + 6a_2 = 11(a_1 + 2a_0) + 10a_1 = 11 \times 3 + 10 = 43$.

   (b) We have to show that all the terms a_n are odd integers. This is true for $n = 0$ since $a_0 = 1$. We prove it by induction for $n \geqslant 1$.

   Induction base. For $n = 1$, we have $a_1 = 1$, so the result holds for $n = 1$.

   Induction step. We now assume that the result is true for a given arbitrary value of $n \geqslant 1$, and show that it is true for $n + 1$. Thus we assume:

   Induction hypothesis: a_n is an odd integer.

   Then,

   $$a_{n+1} = a_n + 2a_{n-1},$$

   so a_{n+1} is the sum of an even integer (namely $2a_{n-1}$) and a_n which, by the induction hypothesis, is an odd integer. Hence a_{n+1} is an odd integer, and the induction step is proved.

   Conclusion: By mathematical induction, a_n is an odd integer for all $n \geqslant 1$.

4. *Consider the Fibonacci sequence* $f_n = f_{n-1} + f_{n-2}$ *where* $f_1 = f_2 = 1$. *Use induction to prove that for all* $n \geqslant 1$,

   $$f_1 + f_2 + \cdots + f_n = f_{n+2} - 1$$

   Induction base. For $n = 1$, the statement is '$f_1 = f_3 - 1$'. Since $f_1 = 1$ and $f_3 = 2$, this is certainly true.

   Induction step. We now assume that the statement is true for a given arbitrary value of n, and show that it is true for $n + 1$. Thus we assume:

   Induction hypothesis: $f_1 + f_2 + \ldots + f_n = f_{n+2} - 1$

Then

$$\begin{aligned} f_1 + f_2 + \ldots + f_n + f_{n+1} &= f_{n+2} - 1 + f_{n+1}, \qquad \text{by the induction hypothesis,} \\ &= f_{n+3} - 1. \end{aligned}$$

This completes the induction step.

Conclusion: By mathematical induction, the result holds for all $n \geqslant 1$.

5. * *Consider the infinite sequence defined by* $a_n = 3a_{n-1} - 2a_{n-2}$ *for* $n \geqslant 3$, *where* $a_1 = 6$ *and* $a_2 = 11$. *Prove that for all* $n \geqslant 1$,

$$a_n = \left(5 \times 2^{n-1}\right) + 1.$$

We use 'general induction'.

Induction base. The result holds for $n = 1$, since $a_1 = 6 = 5 \times 2^0 + 1$, and for $n = 2$, since $a_2 = 11 = 5 \times 2^1 + 1$.

Induction step. We now assume that the statement is true for all k less than a given arbitrary value of n, and show that it is true for n. Thus we assume:

Induction hypothesis: $a_k = \left(5 \times 2^{k-1}\right) + 1$ for all $k < n$.

Then

$$\begin{aligned} a_n &= 3a_{n-1} - 2a_{n-2} \\ &= 3\left(5 \times 2^{n-2} + 1\right) - 2\left(5 \times 2^{n-3} + 1\right), \qquad \text{by the induction hypothesis,} \\ &= 15 \times 2^{n-2} - 5 \times 2^{n-2} + 1 \\ &= 10 \times 2^{n-2} + 1 \\ &= 5 \times 2^{n-1} + 1 \end{aligned}$$

This completes the induction step.

Conclusion: By mathematical induction, the desired result holds for all $n \geqslant 1$.

6. *Construct the truth table for* $[(p \vee q) \wedge r] \;\Rightarrow\; (p \wedge \neg q)$.

p	q	r	$p \vee q$	$(p \vee q) \wedge r$	$p \wedge \neg q$	$[(p \vee q) \wedge r] \;\Rightarrow\; (p \wedge \neg q)$
T	T	T	T	T	F	F
T	T	F	T	F	F	T
T	F	T	T	T	T	T
T	F	F	T	F	T	T
F	T	T	T	T	F	F
F	T	F	T	F	F	T
F	F	T	F	F	F	T
F	F	F	F	F	F	T

7. *Construct the truth tables for* $\neg(p \vee q) \;\Rightarrow\; r$ *and* $\neg((p \vee q) \;\Rightarrow\; r)$.

p	q	r	$(p \vee q)$	$\neg(p \vee q)$	$\neg(p \vee q) \;\Rightarrow\; r$	$(p \vee q) \;\Rightarrow\; r$	$\neg((p \vee q) \;\Rightarrow\; r)$
T	T	T	T	F	T	T	F
T	T	F	T	F	T	F	T
T	F	T	T	F	T	T	F
T	F	F	T	F	T	F	T
F	T	T	T	F	T	T	F
F	T	F	T	F	T	F	T
F	F	T	F	T	T	T	F
F	F	F	F	T	F	T	F

SCHOOL OF MATHEMATICS AND STATISTICS
PMA1050/PMA6853 Discrete Foundations 2004–05

TEST 1 — SOLUTIONS

Q1. *If* $A = \{1,3,5,6\}$, $B = \{2,3,6\}$ *and* $C = \{1,4,6\}$, *then* $(A \cap B) \cup C$ *equals*
A. $\{1,6\}$ B. $\{1,3,4,6\}$ C. $\{1,2,3,4,5,6\}$ D. $\{4,6\}$.
Answer: **B**.

$$(A \cap B) \cup C = \{3,6\} \cup \{1,4,6\} = \{1,3,4,6\}.$$

Q2. *If* $A = \{1,3,4,5,6\}$, $B = \{2,3,5,6\}$ *and* $C = \{1,4,6\}$, *then* $A \setminus (B \setminus C)$ *equals*
A. $\{1,3,4,5\}$ B. $\emptyset$ C. $\{5\}$ D. $\{1,4,6\}$.
Answer: **D**.

$$\begin{aligned} A \setminus (B \setminus C) &= \{1,3,4,5,6\} \setminus (\{2,3,5,6\} \setminus \{1,4,6\}) \\ &= \{1,3,4,5,6\} \setminus \{2,3,5\} \\ &= \{1,4,6\}. \end{aligned}$$

Q3. *Only one of the following statements is generally true for all sets* A, B, C*: which one?*
A. $A \setminus (B \setminus C) = (A \setminus B) \setminus C$ B. $A \setminus (B \cap C) = (A \setminus B) \setminus C$
C. $A \setminus (B \setminus C) = (A \cup B) \cup C^c$ D. $A \setminus (B \cup C) = (A \setminus B) \setminus C$.
Answer: **D**.

$$A \setminus (B \cup C) = A \cap (B \cup C)^c = A \cap (B^c \cap C^c) = (A \cap B^c) \cap C^c = (A \setminus B) \setminus C.$$

Q4. *All but one of the following sets are empty. Which is the odd one out?*
A. $\{0\}$ B. $\{x \in \mathbb{R} : x^2 = -1\}$
C. $\{1,2,3\} \setminus \{1,2,3,4,5\}$ D. $\{x \in \mathbb{Z} : 4x^2 - 4x + 1 = 0\}$.
Answer: **A**.
The set $\{0\}$ has one element, the fact that it is the number zero is irrelevant.

Q5. *Consider the following rules:*
(i) For each x, *let* y *be such that* $y^2 = \cos x$;
(ii) For each x, *let* y *be such that* $y^3 = \cos x$;
(iii) For each x, *let* y *be such that* $y^3 = x^2$.
The rules which define y *as a* ***function*** *of* x *on the domain* $\mathbb{R}$ *are*
A. *(ii) and (iii)* B. *(i) and (ii)* C. *(ii) only* D. *(iii) only.*
Answer: **A**.
(i) does not define a function for two reasons: (a) $\cos x$ can take negative values, in which case there is no y such that $y^2 = \cos x$; and (b) If $\cos x > 0$ then there are two values of y, one positive and one negative, such that $y^2 = \cos x$.
(ii) and (iii) determine functions since every real number has a unique cube root.

Q6. *A function* $f : \{1,2,3\} \to \{1,2,3\}$ *is defined by:* $f(1) = 3$, $f(2) = 2$, $f(3) = 3$.
Then f *is*
A. *1–1 and onto* B. *1–1 but not onto*
C. *onto but not 1–1* D. *neither 1–1 nor onto.*
Answer: **D**.
The function f is not 1–1 since $f(1) = f(3)$, and not onto since there is no x such that $f(x) = 1$.

Q7. *Let $\mathbb{N} = \{0, 1, 2, 3, \ldots\}$ and $\mathcal{P}(\mathbb{N})$ be the power set of $\mathbb{N}$, i.e. the set of all subsets of $\mathbb{N}$. Let $f : \mathcal{P}(\mathbb{N}) \setminus \{\emptyset\} \to \mathbb{N}$ be the function which associates with any subset of $\mathbb{N}$ the smallest element of that subset. Then f is*

A. *injective and surjective* B. *injective but not surjective*
C. *surjective but not injective* D. *neither injective nor surjective.*

Answer: **C**.

Note that 'injective' is just another name for '1–1' and 'surjective' means 'onto'. You need to be familiar with both sets of terminology.

We have deliberately given f the domain $\mathcal{P}(\mathbb{N}) \setminus \{\emptyset\}$ because we can't define $f(\emptyset)$ as the smallest element of $\emptyset$.

The function is surjective, since every $n \in \mathbb{N}$ may be written as $n = f(\{n\})$. It is not injective since, for example, $f(\{1\}) = 1 = f(\{1, 2\})$. Note that we only need **one** example to prove that f is **not** injective; we need to consider **all** n to prove that it **is** surjective.

Q8. *For all but one of the following, induction is an appropriate method of proof. Which is the odd one out?*

A. $\displaystyle\sum_{k=1}^{n} k^3 = \left(\frac{n(n+1)}{2}\right)^2$ *for* $n = 1, 2, 3, \ldots$

B. $x < \sin x$ *for all positive real numbers* x
C. $n! > 2^n$ *for all integers* $n \geqslant 4$
D. *for* $n = 1, 2, 3, \ldots$, *the number of permutations of* $1, 2, \ldots n$ *is* $n!$.

Answer: **B**.

All except **B** involve proving something for $n = 1, 2, 3, \ldots$ (or $4, 5, 6, \ldots$) with some hope of moving from the nth case to the $(n + 1)$th. Result **B** has to be proved for all positive real numbers x, so induction is not appropriate.

Q9. *Consider the compound statements (i) $\neg(P \Rightarrow Q)$, (ii) $P \Rightarrow Q$ (iii) $P \wedge \neg Q$. Then*

A. *(i) and (ii) are equivalent* B. *(ii) and (iii) are equivalent*
C. *(i) and (iii) are equivalent* D. *(i), (ii) and (iii) are equivalent.*

Answer: **C**.

P	Q	$\neg Q$	$P \Rightarrow Q$	$\neg(P \Rightarrow Q)$	$P \wedge \neg Q$
T	T	F	T	F	F
T	F	T	F	T	T
F	T	F	T	F	F
F	F	T	T	F	F

Q10. *Consider the compound statements (i) $(P \vee Q) \vee ((\neg P) \wedge (\neg Q))$ and (ii) $(P \vee Q) \wedge ((\neg P) \wedge (\neg Q))$. Then*

A. *both (i) and (ii) are tautologies*
B. *(i) is a tautology and (ii) is a contradiction*
C. *(i) is a is a contradiction and (ii) is a tautology*
D. *both (i) and (ii) are neither tautologies nor contradictions.*

Answer: **B**.

P	Q	$\neg P$	$\neg Q$	$P \vee Q$	$(\neg P) \wedge (\neg Q)$	(i)	(ii)
T	T	F	F	T	F	T	F
T	F	F	T	T	F	T	F
F	T	T	F	T	F	T	F
F	F	T	T	F	T	T	F

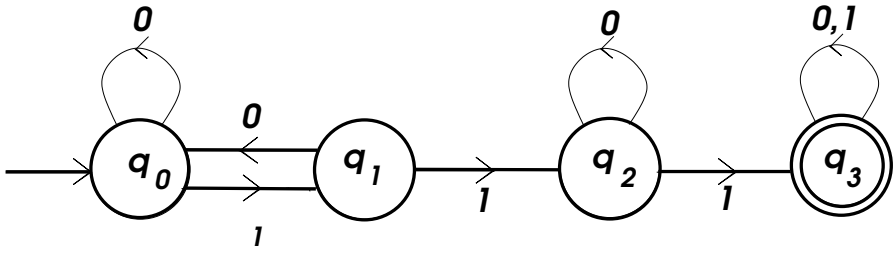
PMA1050/PMA6853 Discrete Foundations 2004–5
Question Sheet 5

Please complete this by Friday 19 November, when the solutions will be given. The starred question is for PMA6853 students.

1. The most common basic logic circuits are the NAND and NOR gates. (These are less complicated electronic circuits than AND or OR gates.) The two-input NAND gate implements the logical connective
$$A \uparrow B = \neg(A \wedge B)$$
and so the one-input NAND gate implements $\neg A$.

   Express the logical connectives $\wedge$, $\vee$ and $\Rightarrow$ in terms of $\uparrow$ and $\neg$, thus demonstrating that these connectives can be implemented in circuits built entirely of NAND gates.

2. A finite automaton M has state set $Q = \{q_0, q_1, q_2, q_3\}$. Its input alphabet is just $\Sigma = \{0, 1\}$, and the rest of its specification is given by the following state diagram. Given a transition table and say what the initial state and final states are.



   What is $L(M)$ for this M?

3. A finite automaton M has state set $Q = \{q_0, q_1, q_2\}$. Its input alphabet is $\Sigma = \{0, 1\}$, with q_0 being the initial state and $F = \{q_0\}$. The transition function δ is given by the following transition table.

δ	0	1
q_0	q_2	q_1
q_1	q_1	q_0
q_2	q_2	q_2

   Give the state diagram for M and describe the language $L(M)$.

4. Sheep go "baa!" or "baaa!" or "baaaa!" and so on. They do not go "baaa" or "ba!" or "!" or ε. Construct a finite automaton (with input alphabet $\Sigma = \{a, b, !\}$) that recognizes "sheep-talk". Give a state diagram **and** a transition table.

5. Design a the finite automaton for C++ floating-point real numbers in scientific notation notation. (It should reject numbers in integer form or in non-scientific notation.) Give a state diagram **or** a transition table.

6. * It would be nice to have a finite automaton which checks whether a mathematical expression has a viable arrangement of brackets. To simplify the problem, suppose the input alphabet is just brackets: $\Sigma = \{\text{'('},\text{')'}\}$. We want a machine M such that the language $L(M)$ accepted by M consists of just those strings of brackets in which the number of opening brackets equals the number of closing brackets and such that in no initial segment of the string does the number of closing brackets exceed the number of opening brackets. Explain why it is impossible to construct such a machine.

Reminder The week 8-12 November is a reading week for this course: no formal lectures and no further homework, but there will be optional help sessions at the usual Wednesday and Thursday times and rooms.

PMA1050 Discrete Foundations 2004–5
Question Sheet 5: Solutions

1. *The most common basic logic circuits are the NAND and NOR gates. (These are less complicated electronic circuits than AND or OR gates.) The two-input NAND gate implements the logical connective*

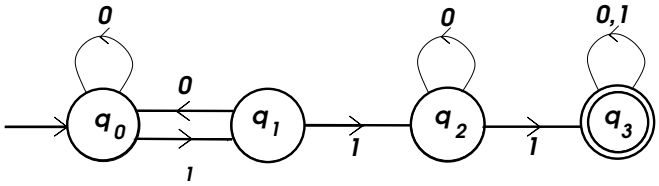
$$A \uparrow B = \neg(A \wedge B)$$

and so the one-input NAND gate implements $\neg A$.

Express the logical connectives $\wedge$, $\vee$ *and* $\Rightarrow$ *in terms of* $\uparrow$ *and* $\neg$, *thus demonstrating that these connectives can be implemented in circuits built entirely of NAND gates.*

$$\begin{aligned} A \wedge B &= \neg(A \uparrow B), \\ A \vee B &= \neg(\neg A \wedge \neg B), \text{ by de Morgan,} \\ &= \neg A \uparrow \neg B, \\ A \Rightarrow B &= \neg A \vee B \\ &= A \uparrow \neg B. \end{aligned}$$

2. *A finite automaton* M *has state set* $Q = \{q_0, q_1, q_2, q_3\}$. *Its input alphabet is just* $\Sigma = \{0, 1\}$, *and the rest of its specification is given by the following state diagram. Given a transition table and say what the initial state and final states are.*



What is $L(M)$ *for this* M*?*

The machine M has initial state q_0; its set of final states is $F = \{q_3\}$ and its transition table is

δ	0	1
q_0	q_0	q_1
q_1	q_0	q_2
q_2	q_2	q_3
q_2	q_3	q_3

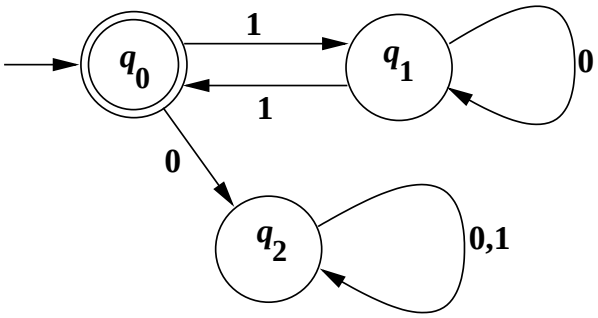
The language $L(M)$ consists of the empty string, together with those strings which contain at least one pair 11 with a further 1 occurring later in the string.

3. *A finite automaton* M *has state set* $Q = \{q_0, q_1, q_2\}$. *Its input alphabet is* $\Sigma = \{0, 1\}$, *with* q_0 *being the initial state and* $F = \{q_0\}$. *The transition function* δ *is given by the following transition table.*

δ	*0*	*1*
q_0	q_2	q_1
q_1	q_1	q_0
q_2	q_2	q_2

Give the state diagram for M *and describe the language* $L(M)$.

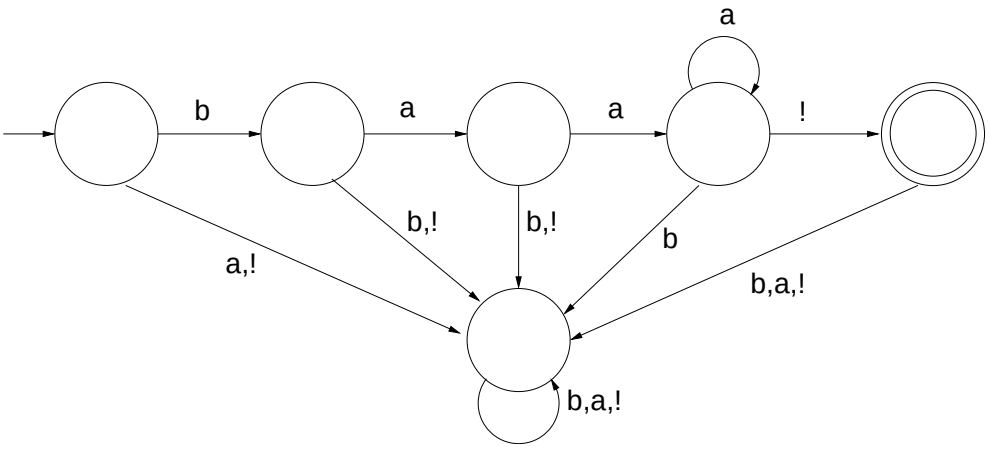
The state diagram is



The language $L(M)$ consists of those strings which begin and end with 1 and such that the string strictly between the initial and final characters is a string of 0s interspersed with pairs 11.

4. *Sheep go "baa!" or "baaa!" or "baaaa!" and so on. They do not go "baaa" or "ba!" or "!" or ε. Construct a finite automaton (with input alphabet $\Sigma = \{a, b, !\}$) that recognizes "sheep-talk". Give a state diagram* ***and*** *a transition table.*

   The state diagram is



   The transition table is

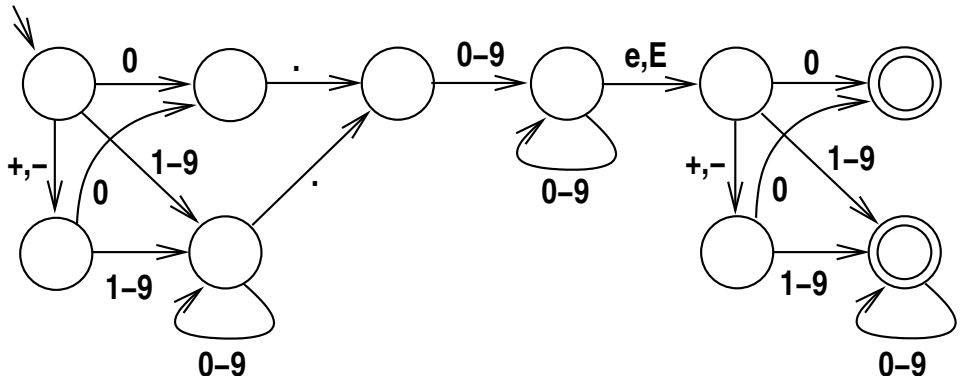
δ	a	b	!
q_0	q_5	q_1	q_5
q_1	q_2	q_5	q_5
q_2	q_3	q_5	q_5
q_3	q_3	q_5	q_4
q_4	q_5	q_5	q_5
q_5	q_5	q_5	q_5

   The state q_0 is the initial state and the set of acceptance states is $F = \{q_4\}$.

5. *Design a the finite automaton for C++ floating-point real numbers in scientific notation notation. (It should reject numbers in integer form or in non-scientific notation.) Give a state diagram* ***or*** *a transition table.*

   The state diagram is

   where it is understood that transitions not shown on the diagram are to a 'rubbish state' which is not final and from which all transitions go to itself.



6. * *It would be nice to have a finite automaton which checks whether a mathematical expression has a viable arrangement of brackets. To simplify the problem, suppose the input alphabet is just brackets:* $\Sigma = \{\text{'('},\text{')'}\}$. *We want a machine* M *such that the language* $L(M)$ *accepted by* M *consists of just those strings of brackets in which the number of opening brackets equals the number of closing brackets and such that in no initial segment of the string does the number of closing brackets exceed the number of opening brackets. Explain why it is impossible to construct such a machine.*

Essentially, the difficulty is that a machine to recognize correct bracketting would have to count the brackets. Since the number of brackets open at some stage could be indefinitely large, and not known at the point at which the machine is built, any finite machine could be swamped by confronting it with a sufficiently large string of opening brackets. The following is a rigorous proof.

Proof. We prove the result by contradiction. Suppose there is a finite automaton M which recognises this language. Suppose M has n states, $Q = \{q_0, q_1, \ldots, q_{n-1}\}$, and set F of final states. Consider the strings $S_k = ((((\ldots($ with k opening brackets and $T_k =))))\ldots)$ with k closing brackets. In particular, consider the $n+1$ different strings

$$S_0, S_1, \ldots, S_n,$$

where $S_0 = \varepsilon$. When input, these result in the machines state changing from q_0 to the states

$$\hat{\delta}(q_0, S_0), \hat{\delta}(q_0, S_1), \hat{\delta}(q_0, S_2), \ldots, \hat{\delta}(q_0, S_n),$$

respectively. Now there are only n states, so at least two of these $n+1$ states must be equal; say

$$\hat{\delta}(q_0, S_i) = \hat{\delta}(q_0, S_j),$$

where $i \neq j$.

The machine accepts S_iT_i, opening i brackets and then closing i brackets, so

$$\hat{\delta}(q_0, S_iT_i) \in F.$$

But

$$\begin{aligned}\hat{\delta}(q_0, S_iT_i) &= \hat{\delta}\left(\hat{\delta}(q_0, S_i), T_i\right) \\ &= \hat{\delta}\left(\hat{\delta}(q_0, S_j), T_i\right) \\ &= \hat{\delta}(q_0, S_jT_i)\end{aligned}$$

So

$$\hat{\delta}(q_0, S_jT_i) \in F.$$

Thus the machine accepts the string S_jT_i consisting of j opening brackets followed by i closing brackets, where $i \neq j$. This contradicts the specification of the machine.

PMA1050/PMA6853 Discrete Foundations 2004–5
Question Sheet 6

Please complete this by Friday 26 November, when the solutions will be given. The starred question is for PMA6853 students.

1. Write regular expressions for each of the following sets defined over the alphabet $\{0, 1\}$:
   (a) The set of all strings with at least one pair of consecutive 0s and at least one pair of consecutive 1s (e.g. 011000 is included; 00101 is not).
   (b) The set of all strings with exactly one pair of consecutive 0s (e.g. 011001 is included; 000101 is not).
   (c) The set of all strings in which every pair of consecutive 0s appears immediately before a pair of consecutive 1s (e.g. ε, 01101 and 1100111 are included; 0001101 and 001011 are not).
   (d) The set of all strings that do not contain the substring 101 (e.g. 011001 is included; 000101 is not).

2. Describe as concisely as possible in English the sets that are denoted by the following regular expressions.
   (a) $\mathbf{1}(\mathbf{1} \cup \mathbf{0})^*\mathbf{0}$
   (b) $(\mathbf{1} \cup \mathbf{01} \cup \mathbf{001})^*(\boldsymbol{\varepsilon} \cup \mathbf{0} \cup \mathbf{00})$
   (c) $(\mathbf{01} \cup \mathbf{10} \cup \mathbf{00} \cup \mathbf{11})^*$

3. Write regular expressions for the following sets defined over the alphabet $\{0, 1\}$.
   (a) The set of all strings of 1s and 0s of length 3 (e.g. 011 is included; 000101 is not).
   (b) The set of all strings with at least two 0s (e.g. 011010 is included; 11101 is not).
   (c) The set of all strings that have at least one 1 and at least one 0 (e.g. 01001 is included; 000, 1111 and ε are not).

4. * It is well known that that an integer is divisible by three if and only if the sum of its digits is divisible by three. Consider the set L of all strings over the alphabet $\Sigma = \{0, 1, \ldots, 9\}$ representing integers divisible by three. For simplicity, we allow integers to have leading zeros (e.g. $0027 \in L$) and we include the empty string in L.
   (a) Design a finite automaton M with $L(M) = L$.
   (b) Produce a regular expression representing L. To do this in a reasonably compact form, you may find it useful to set

$$\begin{aligned} \boldsymbol{A} &= \mathbf{0} \cup \mathbf{3} \cup \mathbf{6} \cup \mathbf{9} \\ \boldsymbol{B} &= \mathbf{1} \cup \mathbf{4} \cup \mathbf{7} \\ \boldsymbol{C} &= \mathbf{2} \cup \mathbf{5} \cup \mathbf{8}. \end{aligned}$$

PMA1050/PMA6853 Discrete Foundations 2004–5
Question Sheet 6: Solutions

1. *Write regular expressions for each of the following sets defined over the alphabet* $\{0,1\}$ *:*

   (a) The set of all strings with at least one pair of consecutive 0s and at least one pair of consecutive 1s (e.g. 011000 is included; 00101 is not).

   (b) The set of all strings with exactly one pair of consecutive 0s (e.g. 011001 is included; 000101 is not).

   (c) The set of all strings in which every pair of consecutive 0s appears immediately before a pair of consecutive 1s (e.g. ε*, 01101 and 1100111 are included; 0001101 and 001011 are not).*

   (d) The set of all strings that do not contain the substring 101 (e.g. 011001 is included; 000101 is not).

   Note that there do not exist unique solutions for this question; several regular expressions may define the same set of strings.

   (a) Strings that have at least one 00 occurring earlier than a 11 are of the form

   (arbitrary string)(00)(arbitrary string)(11)(arbitrary string)

   so the set of all such strings is represented by the regular expression

   $$((\mathbf{1} \cup \mathbf{0})^* \mathbf{00} (\mathbf{1} \cup \mathbf{0})^* \mathbf{11} (\mathbf{1} \cup \mathbf{0})^*).$$

   The set of all strings with at least one 00 occurring later than a 11 is likewise represented by the regular expression

   $$((\mathbf{1} \cup \mathbf{0})^* \mathbf{11} (\mathbf{1} \cup \mathbf{0})^* \mathbf{00} (\mathbf{1} \cup \mathbf{0})^*).$$

   Therefore, the set of all strings with at least one pair of consecutive 0s and at least one pair of consecutive 1s, which is the union of these two sets, is represented by the regular expression

   $$((\mathbf{1} \cup \mathbf{0})^* \mathbf{00} (\mathbf{1} \cup \mathbf{0})^* \mathbf{11} (\mathbf{1} \cup \mathbf{0})^*) \cup ((\mathbf{1} \cup \mathbf{0})^* \mathbf{11} (\mathbf{1} \cup \mathbf{0})^* \mathbf{00} (\mathbf{1} \cup \mathbf{0})^*).$$

   (b) The set of all strings with exactly one pair of consecutive 0s is represented by the regular expression

   $$(\mathbf{01} \cup \mathbf{1})^* \mathbf{00} (\mathbf{10} \cup \mathbf{1})^*.$$

   (c) The set of all strings in which every pair of consecutive 0s appears immediately before a pair of consecutive 1s is represented by the regular expression

   $$(\mathbf{01} \cup \mathbf{1} \cup \mathbf{0011})^* (\mathbf{0} \cup \varepsilon).$$

   (d) The set of all strings that do not contain the substring 101 is represented by the regular expression

   $$(\mathbf{0} \cup \varepsilon)(\mathbf{1} \cup \mathbf{00} \cup \mathbf{000})^* (\mathbf{0} \cup \varepsilon).$$

2. *Describe as concisely as possible in English the sets that are denoted by the following regular expressions.*

   (a) $\boldsymbol{1(1 \cup 0)^* 0}$

   (b) $(\boldsymbol{1 \cup 01 \cup 001})^* (\boldsymbol{\varepsilon \cup 0 \cup 00})$

   (c) $(\boldsymbol{01 \cup 10 \cup 00 \cup 11})^*$

   (a) The expression $\mathbf{1}(\mathbf{1} \cup \mathbf{0})^* \mathbf{0}$ defines the set of all strings that begin with a 1 and end with a 0.

   (b) The expression $(\mathbf{1} \cup \mathbf{01} \cup \mathbf{001})^* (\boldsymbol{\varepsilon} \cup \mathbf{0} \cup \mathbf{00})$ defines the set of all strings that do not contain the substring 000.

(c) The expression $(\mathbf{01} \cup \mathbf{10} \cup \mathbf{00} \cup \mathbf{11})^*$ defines the set of all strings with an even (including zero) number of characters.

3. *Write regular expressions for the following sets defined over the alphabet* $\{0, 1\}$.

   (a) The set of all strings of 1s and 0s of length 3 (e.g. 011 is included; 000101 is not).

   (b) The set of all strings with at least two 0s (e.g. 011010 is included; 11101 is not).

   (c) The set of all strings that have at least one 1 and at least one 0 (e.g. 01001 is included; 000, 1111 and ε *are not).*

   (a) The set of all strings of 1s and 0s of length 3 is represented by the regular expression

   $$(\mathbf{0} \cup \mathbf{1})(\mathbf{0} \cup \mathbf{1})(\mathbf{0} \cup \mathbf{1}).$$

   (b) Strings with at least two 0s are of the form

   (start)(1st expression containing 0)(middle)(2nd expression containing 0)(end)

   Thus the set of all such strings is represented by the regular expression

   $$(\mathbf{0} \cup \mathbf{1})^*\mathbf{0}(\mathbf{0} \cup \mathbf{1})^*\mathbf{0}(\mathbf{0} \cup \mathbf{1})^*.$$

   (c) The set of all strings that have at least one 1 and at least one 0 is represented by the regular expression

   $$(\mathbf{0} \cup \mathbf{1})^*\mathbf{0}(\mathbf{0} \cup \mathbf{1})^*\mathbf{1}(\mathbf{0} \cup \mathbf{1})^* \cup (\mathbf{0} \cup \mathbf{1})^*\mathbf{1}(\mathbf{0} \cup \mathbf{1})^*\mathbf{0}(\mathbf{0} \cup \mathbf{1})^*.$$

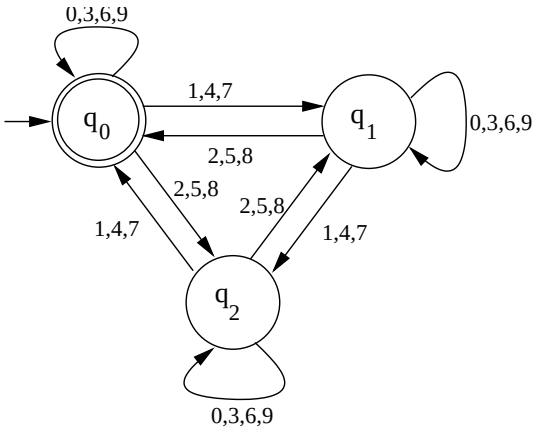
4. * *It is well known that that an integer is divisible by three if and only if the sum of its digits is divisible by three. Consider the set* L *of all strings over the alphabet* $\Sigma = \{0, 1, \ldots, 9\}$ *representing integers divisible by three. For simplicity, we allow integers to have leading zeros (e.g.* $0027 \in L$*) and we include the empty string in* L.

   (a) Design a finite automaton M *with* $L(M) = L$.

   (b) Produce a regular expression representing L. *To do this in a reasonably compact form, you may find it useful to set*

   $$\begin{aligned} \boldsymbol{A} &= \mathbf{0} \cup \mathbf{3} \cup \mathbf{6} \cup \mathbf{9} \\ \boldsymbol{B} &= \mathbf{1} \cup \mathbf{4} \cup \mathbf{7} \\ \boldsymbol{C} &= \mathbf{2} \cup \mathbf{5} \cup \mathbf{8}. \end{aligned}$$

   (a) A suitable FA is



(b) Clearly we have to consider three groups of digits:

$$\begin{aligned} \boldsymbol{A} &= \mathbf{0} \cup \mathbf{3} \cup \mathbf{6} \cup \mathbf{9} \\ \boldsymbol{B} &= \mathbf{1} \cup \mathbf{4} \cup \mathbf{7} \\ \boldsymbol{C} &= \mathbf{2} \cup \mathbf{5} \cup \mathbf{8}. \end{aligned}$$

We consider regular expressions corresponding to the various paths through the state diagram

(i) $\boldsymbol{A}$ represents the path from q_0 just looping straight back to q_0.

(ii) $\boldsymbol{BA^*C}$ represents paths going from q_0 to q_1, possibly looping at q_1, then returning to q_0.

(iii) $(\boldsymbol{A} \cup \boldsymbol{BA^*C})^*$ represents all paths from q_0 to itself not visiting q_2.

(iv) Likewise $(\boldsymbol{A} \cup \boldsymbol{BA^*C})^*$ also represents all paths from q_1 to itself not visiting q_0.

(v) Therefore $\boldsymbol{B}(\boldsymbol{A} \cup \boldsymbol{BA^*C})^*\boldsymbol{C}$ represents all paths from q_0 to itself, via q_1, with no intermediate visits to q_0, and not using the transition from q_2 to q_0.

(vi) Similarly, $\boldsymbol{C}(\boldsymbol{A} \cup \boldsymbol{CA^*B})^*\boldsymbol{B}$ represents all paths from q_0 to itself via q_2, with no intermediate visits to q_0, and not using the transition from q_1 to q_0.

(vii) $\boldsymbol{B}(\boldsymbol{A} \cup \boldsymbol{BA^*C})^*\boldsymbol{BA^*B}$ represents paths going once round the triangle, clockwise, with no intermediate visits to q_0.

(viii) $\boldsymbol{C}(\boldsymbol{A} \cup \boldsymbol{CA^*B})^*\boldsymbol{CA^*C}$ represents paths going once round, anticlockwise, with no intermediate visits to q_0.

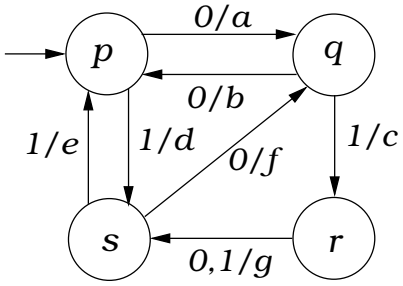
The complete set of paths from q_0 to itself are all concatenations of paths (i), (v), (vi), (vii) and (viii). The required regular expression is

$$(\boldsymbol{A} \cup \boldsymbol{B}(\boldsymbol{A} \cup \boldsymbol{BA^*C})^*\boldsymbol{C} \cup \boldsymbol{C}(\boldsymbol{A} \cup \boldsymbol{CA^*B})^*\boldsymbol{B} \cup \boldsymbol{B}(\boldsymbol{A} \cup \boldsymbol{BA^*C})^*\boldsymbol{BA^*B} \cup \boldsymbol{C}(\boldsymbol{A} \cup \boldsymbol{CA^*B})^*\boldsymbol{CA^*C})^*.$$

PMA1050/PMA6853 Discrete Foundations 2004–5
Question Sheet 7

Please complete this by Friday 3 December, when the solutions will be given. The starred questions are for COM6853 students.

1. The Mealy machine shown below has input alphabet $\Sigma = \{0, 1\}$ and output alphabet $\Gamma = \{a, b, c, d, e, f, g\}$. Note that the label '$0, 1/g$' indicates a transition which is made, with output g, on input either 0 or 1. What is the output generated by the input 0010101?



2. The one's complement of an input bit string is a string that has 1 wherever there was a 0, and a 0 wherever there was a 1; for example, the one's complement of 001 is 110. Construct a Mealy machine that computes the one's complement.

3. Construct a Mealy machine that outputs 1 whenever the substring 101 is encountered, and 0 at all other times; i.e. it outputs 1 only at the point where the current input is 1 and the previous two inputs have been 10.

4. Design a Mealy machine to implement a two unit delay. The input alphabet should be $\Sigma = \{a, b\}$ and the output alphabet $\Gamma = \{a, b, x\}$. The first two letters of the output string should be xx and the nth letter should be the $(n-2)$nd letter of the input string, for $n \geqslant 2$. Thus input *abbaba* produces output *xxabba*, etc..

5. * Suppose we have two finite automata $M_1 = (Q_1, \Sigma_1, \delta_1, q_{01}, F_1)$ and $M_2 = (Q_2, \Sigma_2, \delta_2, q_{02}, F_2)$, accepting languages $L_1 = L(M_1)$, $L_2 = L(M_2)$, respectively. Let $\Sigma = \Sigma_1 \times \Sigma_2$ be the alphabet of pairs (a, b) of characters with $a \in \Sigma_1$ and $b \in \Sigma_2$. (This is the usual Cartesian product of two sets.) Let L be the language over Σ defined by

$$L = \{(a_1, b_1)(a_2, b_2) \dots (a_n, b_n) : a_1 a_2 \dots a_n \in L_1 \text{ and } b_1 b_2 \dots b_n \in L_2\}.$$

   (This is not the same as saying $L = L_1 \times L_2$; why not?)

   By developing a notion of the 'Cartesian product' of two automata, or otherwise, show that there is a finite automaton M such that $L(M) = L$.

   Hence, or otherwise, show that if $M_1 = (Q_1, \Sigma, \delta_1, q_{01}, F_1)$ and $M_2 = (Q_2, \Sigma, \delta_2, q_{02}, F_2)$ are two finite automata with the same input alphabet, then there is a finite automaton M such that $L(M) = L(M_1) \cap L(M_2)$.

6. * **One** of the following 'laws' is generally true for regular expressions $\boldsymbol{R}, \boldsymbol{S}, \boldsymbol{T}$; prove the one that is true and provide a counterexample for the ones which are false:
   (i) $(\boldsymbol{R})^* \cup \boldsymbol{S} = (\boldsymbol{R} \cup \boldsymbol{S})^*$;
   (ii) $(\boldsymbol{R} \cup \boldsymbol{S})\boldsymbol{T} = \boldsymbol{RT} \cup \boldsymbol{ST}$;
   (iii) $\boldsymbol{R} \cup (\boldsymbol{ST}) = (\boldsymbol{R} \cup \boldsymbol{S})(\boldsymbol{R} \cup \boldsymbol{T})$;
   (iv) $\boldsymbol{R}^* = (\boldsymbol{RR})^*$.

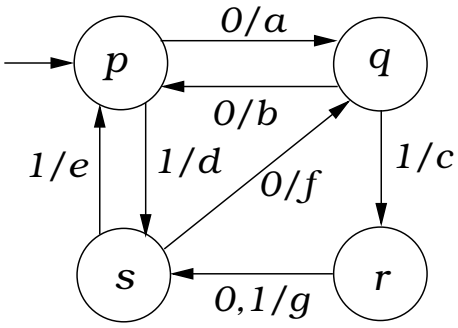
7. * Consider a Mealy machine fed with an infinite input stream. (I use the word 'stream' rather than 'string' to indicate that it is an unending succession of characters: my terminology.) We say that such a stream is *eventually periodic* if it is of the form

$$a_1 a_2 \dots a_k b_1 b_2 \dots b_n b_1 b_2 \dots b_n b_1 b_2 \dots b_n b_1 b_2 \dots b_n \dots$$

   for some positive integers k, n. Show that if the input stream is eventually periodic, then so is the output stream. (Warning: the output stream will not necessarily have the same parameters k, n.)

PMA1050/PMA6853 Discrete Foundations 2004–5
Question Sheet 7: Solutions

1. *The Mealy machine shown below has input alphabet* $\Sigma = \{0, 1\}$ *and output alphabet* $\Gamma = \{a, b, c, d, e, f, g\}$. *Note that the label '*$0, 1/g$*' indicates a transition which is made, with output* g*, on input either 0 or 1. What is the output generated by the input* 0010101*?*



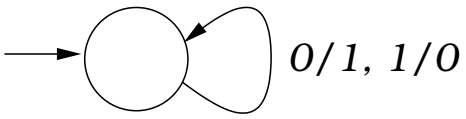
On input 0010101, the machine moves as follows (outputs shown over the arrows).

$$p \xrightarrow{a} q \xrightarrow{b} p \xrightarrow{d} s \xrightarrow{f} q \xrightarrow{c} r \xrightarrow{g} s \xrightarrow{e} p.$$

Thus the output string is $abdfcge$.

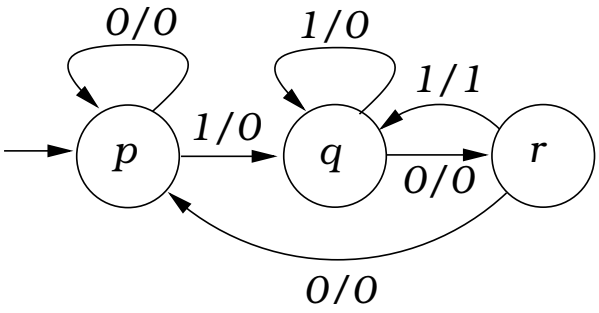
2. *The one's complement of an input bit string is a string that has 1 wherever there was a 0, and a 0 wherever there was a 1; for example, the one's complement of 001 is 110. Construct a Mealy machine that computes the one's complement.*

This can be implemented by the following single-state Mealy machine, (with $\Sigma = \Gamma = \{0, 1\}$).



3. *Construct a Mealy machine that outputs 1 whenever the substring 101 is encountered, and 0 at all other times; i.e. it outputs 1 only at the point where the current input is 1 and the previous two inputs have been 10.*

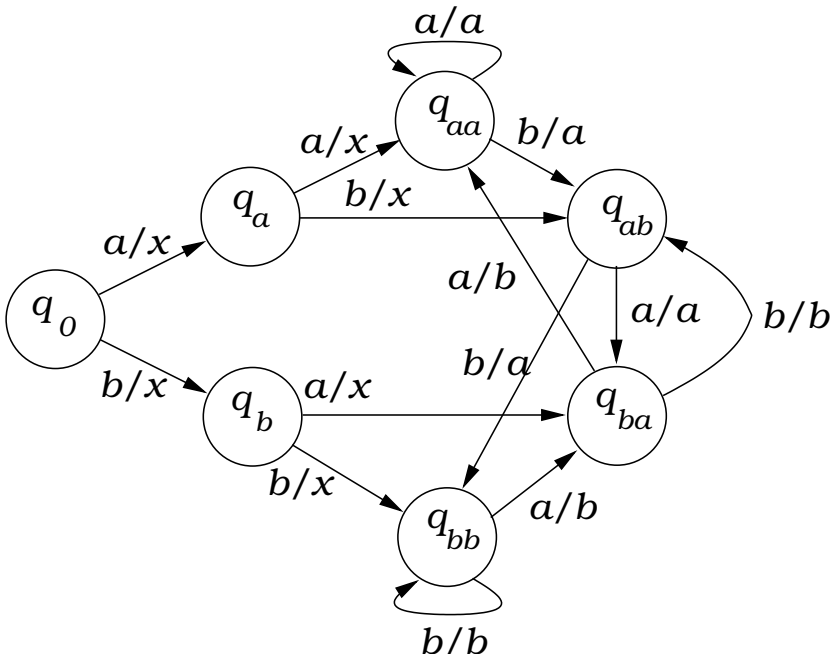
Again, $\Sigma = \Gamma = \{0, 1\}$. The state p is the starting state and corresponds to the either no characters



having been received or the last pair of characters received having been 00; q corresponds to the last character received having been a 1 and r corresponds to the last pair of characters received having been 10. The '1' arrow out of r is therefore labelled as giving output 1, indicating that a 101 string has been discovered.

4. *Design a Mealy machine to implement a two unit delay. The input alphabet should be* $\Sigma = \{a, b\}$ *and the output alphabet* $\Gamma = \{a, b, x\}$*. The first two letters of the output string should be* xx *and the* n*th letter should be the* $(n-2)$*nd letter of the input string, for* $n \geqslant 2$*. Thus input abbaba produces output xxabba, etc..*

   The following Mealy machine implements a two unit delay. As specified, the input alphabet is $\Sigma = \{a, b\}$ and the output alphabet $\Gamma = \{a, b, x\}$. The states remember up to the last two inputs.



   The starting state q_0 corresponds to no input received so far. States q_a, q_b correspond to only a single character, a or b having been received. States q_{aa}, q_{ab}, q_{ba}, q_{bb} correspond to at least two characters having been received and the last two characters having been aa, ab, ba, bb, respectively. Given these interpretations of the states, the transition and output functions are obvious.

5. * *Suppose we have two finite automata* $M_1 = (Q_1, \Sigma_1, \delta_1, q_{01}, F_1)$ *and* $M_2 = (Q_2, \Sigma_2, \delta_2, q_{02}, F_2)$*, accepting languages* $L_1 = L(M_1)$*,* $L_2 = L(M_2)$*, respectively. Let* $\Sigma = \Sigma_1 \times \Sigma_2$ *be the alphabet of pairs* (a, b) *of characters with* $a \in \Sigma_1$ *and* $b \in \Sigma_2$*. (This is the usual Cartesian product of two sets.) Let* L *be the language over* Σ *defined by*

$$L = \{(a_1, b_1)(a_2, b_2) \dots (a_n, b_n) : a_1 a_2 \dots a_n \in L_1 \text{ and } b_1 b_2 \dots b_n \in L_2\}.$$

   (This is not the same as saying $L = L_1 \times L_2$*; why not?)*

   By developing a notion of the 'Cartesian product' of two automata, or otherwise, show that there is a finite automaton M *such that* $L(M) = L$*.*

   Hence, or otherwise, show that if $M_1 = (Q_1, \Sigma, \delta_1, q_{01}, F_1)$ *and* $M_2 = (Q_2, \Sigma, \delta_2, q_{02}, F_2)$ *are two finite automata with the same input alphabet, then there is a finite automaton* M *such that* $L(M) = L(M_1) \cap L(M_2)$*.*

   We define the Cartesian product $M = (Q, \Sigma, \delta, q_0, F)$ of M_1 and M_2 by:

$$Q \quad = \quad Q_1 \times Q_2,$$

$$\begin{aligned} \Sigma &= \Sigma_1 \times \Sigma_2, \qquad \text{given,} \\ q_0 &= (q_{01}, q_{02}), \\ F &= F_1 \times F_2, \end{aligned}$$

with the transition function $\delta : Q \times \Sigma \to Q$ defined by

$$\delta((q_1, q_2), (a_1, a_2)) = (\delta_1(q_1, a_1), \delta_2(q_2, a_2)).$$

It is easy to see that M acting on an input $(a_1, b_1)(a_2, b_2) \dots (a_n, b_n)$ simply mimics the action of the two machines M_1 and M_2 working side-by-side, on inputs $a_1 a_2 \dots a_n$ and $b_1 b_2 \dots b_n$. The definition of F means that the string is accepted if and only if $a_1 a_2 \dots a_n$ is accepted by M_1 and $b_1 b_2 \dots b_n$ by M_2.

If $M_1 = (Q_1, \Sigma, \delta_1, q_{01}, F_1)$ and $M_2 = (Q_2, \Sigma, \delta_2, q_{02}, F_2)$ are two finite automata with the same input alphabet, we let M be as above, except that we restrict the input alphabet to $\Sigma' = \{(a, a) : a \in \Sigma\}$. Then M accepts the language

$$L = \{(a_1, a_1)(a_2, a_2) \dots (a_n, a_n) : a_1 a_2 \dots a_n \in L_1 \cap L_2\}.$$

Clearly Σ' can be identified with Σ and the definition of M adjusted accordingly, to give a machine over the alphabet Σ accepting $L_1 \cap L_2$.

6. * ***One*** *of the following 'laws' is generally true for regular expressions* $\boldsymbol{R}, \boldsymbol{S}, \boldsymbol{T}$*; prove the one that is true and provide a counterexample for the ones which are false:*
(i) $(\boldsymbol{R})^* \cup \boldsymbol{S} = (\boldsymbol{R} \cup \boldsymbol{S})^*$*;*
(ii) $(\boldsymbol{R} \cup \boldsymbol{S})\boldsymbol{T} = \boldsymbol{RT} \cup \boldsymbol{ST}$*;*
(iii) $\boldsymbol{R} \cup (\boldsymbol{ST}) = (\boldsymbol{R} \cup \boldsymbol{S})(\boldsymbol{R} \cup \boldsymbol{T})$*;*
(iv) $\boldsymbol{R}^* = (\boldsymbol{RR})^*$*.*

For all our counterexamples, we let the alphabet be $\Sigma = \{0, 1\}$ and let $\boldsymbol{R} = \boldsymbol{0}$, $\boldsymbol{S} = \boldsymbol{T} = \boldsymbol{1}$.

(i) $(\boldsymbol{R})^* \cup \boldsymbol{S} = (\boldsymbol{R} \cup \boldsymbol{S})^*$ is **false**: in our example, the left hand side corresponds to either a string of 0s or a 1, whereas the right hand side represents any string.

(ii) $(\boldsymbol{R} \cup \boldsymbol{S})\boldsymbol{T} = \boldsymbol{RT} \cup \boldsymbol{ST}$ is **true**: the left hand side corresponds to a string from R or from S followed by a string from T. This is the same as saying either a string from R followed by a string from T or a string from S followed by a string from T, which is what the right-hand side represents.

(iii) $\boldsymbol{R} \cup (\boldsymbol{ST}) = (\boldsymbol{R} \cup \boldsymbol{S})(\boldsymbol{R} \cup \boldsymbol{T})$ is **false**: in our example, the left hand side corresponds to either 0 or 11, whereas the right hand side represents the strings 00, 01, 10 and 11.

(iv) $\boldsymbol{R}^* = (\boldsymbol{RR})^*$ is **false**: in our example, the left hand side corresponds to any string of 0s, whereas the right hand side represents just the strings of 0s of even length.

7. * *Consider a Mealy machine fed with an infinite input stream. (I use the word 'stream' rather than 'string' to indicate that it is an unending succession of characters: my terminology.) We say that such a stream is* eventually periodic *if it is of the form*

$$a_1 a_2 \dots a_k b_1 b_2 \dots b_n b_1 b_2 \dots b_n b_1 b_2 \dots b_n b_1 b_2 \dots b_n \dots$$

for some positive integers k, n*. Show that if the input stream is eventually periodic, then so is the output stream. (Warning: the output stream will not necessarily have the same parameters* k, n*.)*

A Mealy machine is fed with the input stream

$$a_1 a_2 \dots a_k b_1 b_2 \dots b_n b_1 b_2 \dots b_n b_1 b_2 \dots b_n b_1 b_2 \dots b_n \dots$$

Let q_r be the state after the $(k + rn)$th input character. Since the machine has only finitely many states, there must exist $r \neq s$ such that $q_r = q_s$, say with $r < s$. Thus, after the $(k + sn)$th input,

the machine is in the same state as after the $(k+rn)$th input, and it subsequently receives the same input. Therefore it must give the same output. Thus the output is of the form:

$$x_1x_2\ldots x_{k+rn}y_1y_2\ldots y_{(s-r)n}y_1y_2\ldots y_{(s-r)n}y_1y_2\ldots y_{(s-r)n}\cdots,$$

which is eventually periodic.

To write this formally,

$$\hat{\delta}\left(q_0, a_1a_2\ldots a_k\left(b_1b_2\ldots b_n\right)^r\right) = q_r, \qquad \widehat{\omega}\left(q_0, a_1a_2\ldots a_k\left(b_1b_2\ldots b_n\right)^r\right) = x_1x_2\ldots x_{k+rn}$$

and

$$\hat{\delta}\left(q_r, \left(b_1b_2\ldots b_n\right)^{s-r}\right) = q_s = q_r, \qquad \widehat{\omega}\left(q_r, \left(b_1b_2\ldots b_n\right)^{s-r}\right) = y_1y_2\ldots y_{(s-r)n}.$$

PMA1050/PMA6853 Discrete Foundations 2004–5
Question Sheet 8

Please complete this by Friday 10 December, when the solutions will be given. The starred question is for PMA6853 students.

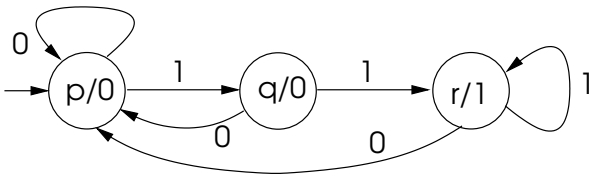
1. Find the unreachable states of the Mealy machine with states p, q, r, s, t, initial state p and the following transition/output table.

	input	p	q	r	s	t
δ	0	q	p	t	q	t
δ	1	s	q	r	p	q
ω	0	x	x	x	x	x
ω	1	y	x	y	y	x

2. Minimize the Mealy machine with states p, q, r, s, t, initial state p and the following transition/output table (which is not the same as in the previous question).

	input	p	q	r	s	t
δ	0	q	r	t	q	t
δ	1	s	q	r	p	q
ω	0	x	x	x	x	x
ω	1	y	x	y	y	x

3. The Moore machine shown below has input and output alphabets $\Sigma = \Gamma = \{0, 1\}$ What is the output generated by the input 01011101? How is the output related to the input generally?



4. The one's complement of an input bit string is a string that has 1 wherever there was a 0, and a 0 wherever there was a 1; for example, the one's complement of 001 is 110. Construct a Moore machine that computes the one's complement. (Compare this with the equivalent Mealy machine constructed last week.)

5. Construct a Moore machine that outputs 1 whenever the substring 101 is encountered, and 0 at all other times; i.e. it outputs 1 only at the point where the current input is 1 and the previous two inputs have been 10. (Compare this with the equivalent Mealy machine constructed last week.)

6. Minimize the Mealy machine with states $p, q, r, s, t, u, v, w,$, initial state p and the following transition/output table.

	input	p	q	r	s	t	u	v	w
δ	0	r	v	t	w	s	t	s	t
δ	1	r	r	r	s	u	u	p	p
δ	2	s	q	w	p	p	w	u	w
ω	0	2	2	1	1	2	1	0	2
ω	1	1	1	1	1	1	1	2	1
ω	2	2	2	0	0	2	0	0	2

7. *

   (a) Show that for every Moore machine there is an equivalent Mealy machine (i.e. one with the same translation function) with the same number of states, or fewer.

   (b) Show that for every Mealy machine M there is an equivalent Moore machine M'. Obtain an upper bound on the number of states in M', in terms of the number of states in M, the size of the input alphabet and the size of the output alphabet.

PMA1050/PMA6853 Discrete Foundations 2004–5
Question Sheet 8: Solutions

1. *Find the unreachable states of the Mealy machine with states p, q, r, s, t, initial state p and the following transition/output table.*

	input	p	q	r	s	t
δ	0	q	p	t	q	t
δ	1	s	q	r	p	q
ω	0	x	x	x	x	x
ω	1	y	x	y	y	x

The initial state is p, and $\delta(p, 0) = q$, $\delta(p, 1) = s$, so p, q and s are reachable. However, the transitions from q and s just go to p and q, so r and t are unreachable.

2. *Minimize the Mealy machine with states p, q, r, s, t, initial state p and the following transition/output table (which is not the same as in the previous question).*

	input	p	q	r	s	t
δ	0	q	r	t	q	t
δ	1	s	q	r	p	q
ω	0	x	x	x	x	x
ω	1	y	x	y	y	x

All the states are reachable.

The states fall into the following 1-equivalence classes: $\{p, r, s\}$, $\{q, t\}$.

The 2-equivalence classes are $\{p, r, s\}$, $\{q\}$, $\{t\}$.

The 3-equivalence classes are $\{p, s\}$, $\{r\}$, $\{q\}$, $\{t\}$.

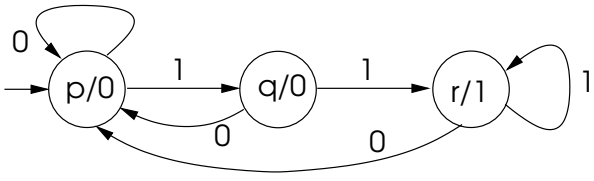
The 4-equivalence classes are $\{p, s\}$, $\{r\}$, $\{q\}$, $\{t\}$.

The situation has stabilized, so we have a minimal machine with states v, corresponding to $\{p, s\}$, r. q and t. Its transition/output table is

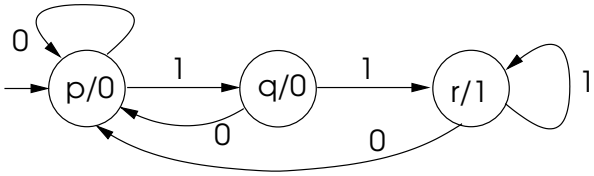
	input	v	q	r	t
δ	0	q	r	t	t
δ	1	v	q	r	q
ω	0	x	x	x	x
ω	1	y	x	y	x

The start state is v.

3. *The Moore machine shown below has input and output alphabets $\Sigma = \Gamma = \{0, 1\}$ What is the output generated by the input* 01011101*? How is the output related to the input generally?*

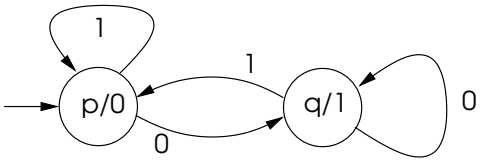


The output generated by the input 01011101 is 00001100. Generally, the machine outputs 1 when the previous two inputs have been 11 and 0 at all other times.



4. *The one's complement of an input bit string is a string that has 1 wherever there was a 0, and a 0 wherever there was a 1; for example, the one's complement of 001 is 110. Construct a Moore machine that computes the one's complement. (Compare this with the equivalent Mealy machine constructed last week.)*

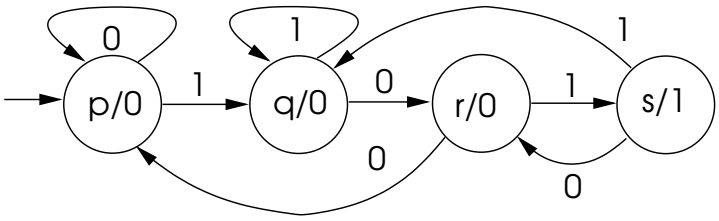
   The one's complement of an input bit string — a string that has 1 wherever there was a 0, and a 0 wherever there was a 1 — can be implemented by the following Moore machine, (with $\Sigma = \Gamma = \{0, 1\}$). Notice that this machine has two states — the minimum possible since we need



   one state outputting 1s and another outputting 0s. The corresponding Mealy machine only had one state.

5. *Construct a Moore machine that outputs 1 whenever the substring 101 is encountered, and 0 at all other times; i.e. it outputs 1 only at the point where the current input is 1 and the previous two inputs have been 10. (Compare this with the equivalent Mealy machine constructed last week.)*

   Our Moore machine that outputs 1 whenever the substring 101 is encountered, and 0 at all other times needs four states — one more than the corresponding Mealy machine. Again, $\Sigma = \Gamma = \{0, 1\}$.



   The state p is the starting state and corresponds to the either no characters having been received or the last pair of characters received having been 00; q corresponds to the last character received having been a 1 and r corresponds to the last pair of characters received having been 10. We then need a further state s for the machine to go to when 101 has been received and from which it outputs 1.

6. *Minimize the Mealy machine with states $p, q, r, s, t, u, v, w,$, initial state p and the following transition/output table.*

	input	p	q	r	s	t	u	v	w
δ	0	r	v	t	w	s	t	s	t
δ	1	r	r	r	s	u	u	p	p
δ	2	s	q	w	p	p	w	u	w
ω	0	2	2	1	1	2	1	0	2
ω	1	1	1	1	1	1	1	2	1
ω	2	2	2	0	0	2	0	0	2

We first observe that q and v are unreachable, and these are the only unreachable states. After deleting these, the machine's table looks like this.

	input	p	r	s	t	u	w
δ	0	r	t	w	s	t	t
δ	1	r	r	s	u	u	p
δ	2	s	w	p	p	w	w
ω	0	2	1	1	2	1	2
ω	1	1	1	1	1	1	1
ω	2	2	0	0	2	0	2

The remaining states fall into the following 1-equivalence classes: $\{p, t, w\}$, $\{r, s, u\}$.

The 2-equivalence classes are $\{p\}$, $\{t\}$, $\{w\}$, $\{r, s, u\}$.

The 3-equivalence classes are $\{p\}$, $\{t\}$, $\{w\}$, $\{r, u\}$ $\{s\}$.

The 4-equivalence classes are $\{p\}$, $\{t\}$, $\{w\}$, $\{r, u\}$ $\{s\}$.

The situation has stabilized, so we have a minimal machine with states p (the start state), t, w, s and x, corresponding to $\{r, u\}$. Its transition/output table is

	input	p	x	s	t	w
δ	0	x	t	w	s	t
δ	1	x	x	s	x	p
δ	2	s	w	p	p	w
ω	0	2	1	1	2	2
ω	1	1	1	1	1	1
ω	2	2	0	0	2	2

7. *

(a) Show that for every Moore machine there is an equivalent Mealy machine (i.e. one with the same translation function) with the same number of states, or fewer.

(b) Show that for every Mealy machine M there is an equivalent Moore machine M'. Obtain an upper bound on the number of states in M', in terms of the number of states in M, the size of the input alphabet and the size of the output alphabet.

(a) Given a Moore machine $M = (Q, \Sigma, \Gamma, q_0, \delta, \omega)$, we form an equivalent Mealy machine $M' = (Q, \Sigma, \Gamma, q_0, \delta, \omega')$ by defining

$$\omega'(q, a) = \omega(\delta(q, a)).$$

Expressed informally, what we have done is to produce a Mealy machine with the same set of states, the same initial state and the same transition function, but the output function is defined so that the output of M' for a given transition is the output of M at the end state of that transition.

(b) Given a Mealy machine $M = (Q, \Sigma, \Gamma, q_0, \delta, \omega)$, we form an equivalent Moore machine $M' = (Q', \Sigma, \Gamma, q_0', \delta', \omega')$ by defining $Q' = Q \times \Gamma$, with $\omega'((q, \gamma)) = \gamma$ and

$$\delta'((q, \gamma), a) = (\delta(q, a), \omega(q, a)).$$

We set $q_0' = (q_0, \gamma)$ for some $\gamma \in \Gamma$ (it doesn't matter which γ).

What we have done is to split each state of M into several states, one for each possible output. The transition function for M' is like the transition function for M, but it chooses which of the several states to move to by the output that M would have made during that transition.

Since $|Q'| = |Q|.|\Gamma|$, this gives us an upper bound on the number of states required by an equivalent Moore machine.

This is not the only possible solution: here is another. Define an equivalent Moore machine $M'' = (Q'', \Sigma, \Gamma, q_0'', \delta'', \omega'')$ by $Q'' = (Q \times \Sigma) \cup \{q_0''\}$, with $\omega''((q,a)) = \omega((q,a))$ and

$$\delta''((q,a),b) = (\delta(q,a),b), \qquad \delta''(q_0'',a) = (q_0,a).$$

Like the previous machine, this has a group of states corresponding to each state q of the original machine. The action of this machine mimics that of M, but with a one step delay. On input a, if the machine M moves from state q to state p, outputting $\omega(q,a)$, then M'' moves *to* state (q,a). In state (q,a), the output of M'' is $\omega''((q,a)) = \omega((q,a))$, the output that M makes in moving from state q to state p in reaction to input a.

Since $|Q''| = 1 + |Q|.|\Sigma|$, this gives us another upper bound on the number of states required by an equivalent Moore machine. We can combine the two, to give the upper bound

$$\min\{|Q|.|\Gamma|,\ 1 + |Q|.|\Sigma|\}.$$

PMA1050/PMA6853 Discrete Foundations 2004–5
Question Sheet 9

Please complete this by Thursday 16 December, when the solutions will be given. The starred question is for PMA6853 students.

1. For each of the following production sets, classify the grammar as type 0, type 1 (context-sensitive), Type 2 (context-free) or Type 3 (regular), and in the last case, say if it is right-linear or left-linear. Then describe the language generated; in the case of regular languages, do this by giving the corresponding regular expression. (In all cases the non-terminal variables of the grammar are those capital letters appearing in the production set and the terminals are the lower-case letters; the starting symbol is S.)

   (a) $S \to aSb$, $S \to Ab$, $A \to Ab$, $A \to a$.

   (b) $S \to aSb$, $S \to C$, $aC \to Ca$, $C \to c$.

   (c) $S \to aS$, $S \to baA$, $A \to bA$, $A \to cA$, $A \to a$.

   (d) $S \to AB$, $A \to aAbC$, $bCb \to bA$, $A \to cA$, $A \to a$. (Trick question!)

   (e) $S \to AC$, $A \to aAb$, $bC \to bcC$, $C \to aC$, $A \to \varepsilon$, $C \to \varepsilon$.

2. Find two grammars, one right-linear and one left-linear, whose languages correspond to the regular expression

$$\mathbf{01}^*\mathbf{21}(\mathbf{11} \cup \mathbf{20})^*(\mathbf{1} \cup \boldsymbol{\varepsilon})$$

3. Find a context free grammar G over the alphabet $\Sigma = \{a,b,c,d\}$ such that

$$L(G) = \{a^n b^m c^m d^n : n, m \geqslant 0\}.$$

4. * Devise an algorithm for minimizing a context-free grammar: i.e. replacing it by an equivalent grammar with as few non-terminal variables as possible. (This is an open-ended question: do not expect to get a perfect algorithm.)

Apply your algorithm to the grammar with the following productions. (as usual, the non-terminal variables of the grammar are those capital letters appearing in the production set and the terminals are the lower-case letters; the starting symbol is S.)

$S \to aAb, \quad S \to aBb,$

$A \to aAb, \quad A \to a, \quad A \to Dc, \quad A \to Fc,$

$B \to cBb, \quad B \to cGa, \quad B \to b,$

$C \to aEbb, \quad C \to c,$

$D \to b, \quad D \to Fb,$

$E \to aC, \quad E \to aE, \quad E \to a,$

$F \to Db, \quad F \to b,$

$G \to Ga.$

PMA1050/PMA6853 Discrete Foundations 2004–5
Question Sheet 9: Solutions

1. *For each of the following production sets, classify the grammar as type 0, type 1 (context-sensitive), Type 2 (context-free) or Type 3 (regular), and in the last case, say if it is right-linear or left-linear. Then describe the language generated; in the case of regular languages, do this by giving the corresponding regular expression. (In all cases the non-terminal variables of the grammar are those capital letters appearing in the production set and the terminals are the lower-case letters; the starting symbol is S.)*

   (a) $S \to aSb$, $S \to Ab$, $A \to Ab$, $A \to a$.

   (b) $S \to aSb$, $S \to C$, $aC \to Ca$, $C \to c$.

   (c) $S \to aS$, $S \to baA$, $A \to bA$, $A \to cA$, $A \to a$.

   (d) $S \to AB$, $A \to aAbC$, $bCb \to bA$, $A \to cA$, $A \to a$. *(Trick question!)*

   (e) $S \to AC$, $A \to aAb$, $bC \to bcC$, $C \to aC$, $A \to \varepsilon$, $C \to \varepsilon$.

   (a) This grammar is context-free, but not regular (because of the first production).

   $$L(G) = \{a^n b^m : m \geqslant n \geqslant 1\}.$$

   (b) This grammar is a type 0 grammar which is not even context-sensitive (because of the third production).

   $$L(G) = \{a^i c a^j b^{i+j} : i, j \geqslant 0,\ i + j \geqslant 1\}.$$

   (c) This grammar is right-linear, and therefore regular: $L(G)$ is given by the regular expression

   $$\boldsymbol{a}^*\boldsymbol{aba}(\boldsymbol{b} \cup \boldsymbol{c})^* a$$

   (d) This grammar is a type 0 grammar which is not even context-sensitive (because of the third production).

   Any derivation must start with $S \to AB$, but thereafter, the B cannot be changed, so we can never reach a string of non-terminals (not even the empty string). Therefore

   $$L(G) = \emptyset.$$

   (e) This grammar is context-sensitive but not context-free (because of the third production).

   $$L(G) = \{a^i b^i c^j a^k : i \geqslant 1, j = 0 \text{ or } 1, k \geqslant 0\} \cup \{a^k : k \geqslant 0\}.$$

2. *Find two grammars, one right-linear and one left-linear, whose languages correspond to the regular expression*

   $$\mathbf{01}^*\mathbf{21}(\mathbf{11} \cup \mathbf{20})^*(\mathbf{1} \cup \boldsymbol{\varepsilon})$$

   A suitable right-linear grammar has productions

   $$S \to 0A, \quad A \to 1A, \quad A \to 21B, \quad B \to 11B, \quad B \to 02B, \quad B \to 1, B \to \varepsilon.$$

   Note how this is constructed by simply working through the regular expression from left to right.

   To construct a left-linear grammar, we simply work through the regular expression from right to left. The grammar has productions

   $$S \to A, \quad S \to A1, \quad A \to A11, \quad A \to A02, \quad A \to B21, \quad B \to B1, \quad B \to 0.$$

   For both grammars, $\Sigma = \{0, 1, 2\}$, $N = \{S, A, B\}$ and the starting variable is S.

3. *Find a context free grammar G over the alphabet $\Sigma = \{a, b, c, d\}$ such that*

$$L(G) = \{a^n b^m c^m d^n : n, m \geqslant 0\}.$$

A suitable grammar G has $N = \{S, A\}$, starting symbol S and productions

$$S \to aSd, \qquad S \to A, \qquad A \to bAc, \qquad A \to \varepsilon.$$

4. * *Devise an algorithm for minimizing a context-free grammar: i.e. replacing it by an equivalent grammar with as few non-terminal variables as possible. (This is an open-ended question: do not expect to get a perfect algorithm.)*

Apply your algorithm to the grammar with the following productions. (as usual, the non-terminal variables of the grammar are those capital letters appearing in the production set and the terminals are the lower-case letters; the starting symbol is S.)

$S \to aAb, \qquad S \to aBb,$

$A \to aAb, \qquad A \to a, \qquad A \to Dc, \qquad A \to Fc,$

$B \to cBb, \qquad B \to cGa, \qquad B \to b,$

$C \to aEbb, \qquad C \to c,$

$D \to b, \qquad D \to Fb,$

$E \to aC, \qquad E \to aE, \qquad E \to a,$

$F \to Db, \quad F \to b,$

$G \to Ga.$

The first step is to remove 'unreachable non-terminals', that is, non-terminals which do not appear in any sequence of $\Rightarrow$s starting with S. In our case, C and E are unreachable.

Next, we remove all 'ineffective non-terminals' (my terminology) meaning non-terminals from which no sequence of $\Rightarrow$s reaches a string of terminals. These can play no part in generating strings in $L(G)$. In our case, only G is ineffective.

Having removed the unreachable and the ineffective non-terminals and all productions involving them, we are left with the grammar with non-terminals S, A, B, D, F and the productions

$S \to aAb, \qquad S \to aBb,$

$A \to aAb, \qquad A \to a, \qquad A \to Dc, \qquad A \to Fc,$

$B \to cBb, \qquad B \to b,$

$D \to b, \qquad D \to Fb,$

$F \to Db, \quad F \to b.$

We now merge 'equivalent' non-terminals, meaning non-terminals which generate the same strings. We first say that two non-terminals are k-equivalent if they generate the same strings in derivations of length less than or equal to k. As with the minimization of Mealy machines, the set of non-terminals splits into k-equivalence classes, for each k. Two non-terminals are $k+1$-equivalent if and only if a single $\Rightarrow$ takes them to strings which are k-equivalent, in the obvious sense. It follows that if the k-equivalence classes and $(k+1)$-equivalence classes coincide, then these are the equivalence classes, which can then be merged into single non-terminals.

Looking for 1-equivalence means looking for productions of the form $X \to w$ where $X \in N$ and $w \in \Sigma*$. If two non terminals have exactly the same sets of such productions, then they are 1-equivalent.

In our case, the 1-equivalence classes are $\{A\}$, $\{B, D, F\}$, and $\{S\}$.

Then D and F are 2-equivalent, because when we look at the productions starting with D and with F, namely $D \to Fb$ and $F \to Db$, their right-hand sides are 1-equivalent. (We need not look

at productions of the form $D \to w$ and $F \to w$ with $w \in \Sigma^*$ again: they have already been dealt with in finding that D and F are 1-equivalent.) However, B is not 2-equivalent to D and F, since there is a production $B \to cBb$ and cBb is not 1-equivalent to Fb and Db.

Thus the 2-equivalence classes are $\{A\}$, $\{B\}$, $\{D, F\}$, and $\{S\}$.

A similar argument show that D and F are 3-equivalent, so the 3-equivalence classes are the same as the 2-equivalence classes. The process has finished. We must merge D and F into a new non-terminal; call it J.

The final grammar has non-terminals S, A, B, J (starting symbol S) and the productions

$$S \to aAb, \quad S \to aBb, \quad A \to aAb, \quad A \to a, \quad B \to cBb, \quad B \to b, \quad A \to Jc, \quad J \to b, \quad J \to Jb.$$