Informatics 1A Computation and Logic
Compulsory Exercise

Answer all the questions below and submit your answer (on paper) to the Informatics Teaching Organisation by **noon** on **Thursday 22nd of November**. This exercise does not contribute to your final mark for the course but the questions it contains are taken from earlier exams, so it gives you important practice. Use this opportunity to prepare for the real exam. If this were a real exam then you would have two hours in which to answer the questions.

1. Using a truth table show whether the expression

$$((a \rightarrow b)\ and\ not(b)) \rightarrow not(a)$$

   is or is not a tautology.

2. You are given the following proof rules:

Rule name	Sequent	Supporting proofs
immediate	$\mathcal{F} \vdash A$	$A \in \mathcal{F}$
and_intro	$\mathcal{F} \vdash A\ and\ B$	$\mathcal{F} \vdash A,\ \mathcal{F} \vdash B$
or_intro_left	$\mathcal{F} \vdash A\ or\ B$	$\mathcal{F} \vdash A$
or_intro_right	$\mathcal{F} \vdash A\ or\ B$	$\mathcal{F} \vdash B$
or_elim	$\mathcal{F} \vdash C$	$A\ or\ B \in \mathcal{F},\ [A\|\mathcal{F}] \vdash C,\ [B\|\mathcal{F}] \vdash C$
imp_elim	$\mathcal{F} \vdash B$	$A \rightarrow B \in \mathcal{F},\ \mathcal{F} \vdash A$
imp_intro	$\mathcal{F} \vdash A \rightarrow B$	$[A\|\mathcal{F}] \vdash B$

   where $\mathcal{F} \vdash A$ means that expression A can be proved from set of axioms $\mathcal{F}$; $A \in \mathcal{F}$ means that A is an element of set $\mathcal{F}$; $[A|\mathcal{F}]$ is the set constructed by adding A to set $\mathcal{F}$; $A \rightarrow B$ means that A implies B; $A\ and\ B$ means that A and B both are true; and $A\ or\ B$ means that at least one of A or B is true.

   (a) Using the proof rules above, prove the following:

$$[b \rightarrow c] \quad \vdash \quad (a\ or\ b) \rightarrow (a\ or\ c)$$

   Show precisely how the proof rules are applied.

(b) Using only the proof rules above it is not possible to perform the following proof:

$$[(a \; and \; b)] \quad \vdash \quad a$$

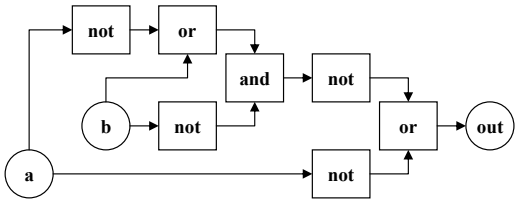
Briefly explain why the proof rules fail to support this proof.

(c) Briefly explain, in general terms, the concepts of soundness and completeness of a proof system.

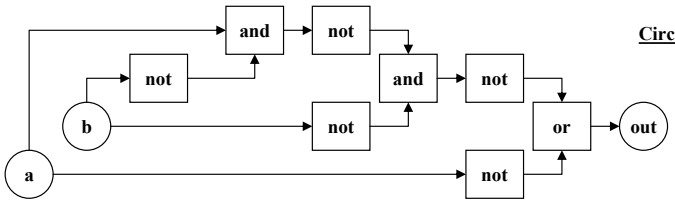
3. The diagram below gives two different logic circuits (Circuit 1 and Circuit 2) constructed from the following components:

- An "and" connective (drawn in the diagram as a box labelled "and") sends as output the signal "true" only if both its inputs are signalling "true", otherwise it signals "false".
- An "or" connective (drawn in the diagram as a box labelled "or") sends as output the signal "false" only if both its inputs are signalling "false", otherwise it signals "true".
- A "not" connective (drawn in the diagram as a box labelled "not") sends as output the signal "true" if its input signals "false", otherwise it signals "false" if its input signals "true".

Signals are propagated through the circuit in the direction indicated by the arrows connecting components. Each circuit has two inputs (the circles labelled "a" and "b" in the diagram) and each has a single output (the circle labelled "out" in the diagram).



Circuit 1



Circuit 2

Suppose that you have been given these circuits by an engineer who has two concerns: he is unsure whether or not the two circuits give the same responses to inputs, and he has a suspicion that at least one of the circuits shows little change in output regardless of input.

(a) Describe each of the two circuits as an expression in propositional logic, where the signal produced as the output of the circuit should correspond to the truth value of the expression.

(b) For each of the two propositional expressions construct a truth table that calculates the truth value of the expression for all combinations of truth values for its two inputs.

(c) Write a brief analysis of the circuits to answer the engineer's concerns (given above) using your truth tables as evidence.

4. Convert the expression $((p \; and \; q) \rightarrow r) \; and \; (p \rightarrow q)$ into an equivalent expression containing only '*or*', '*and*' and '*not*' logical operators (with *not* applying only to propositions, not to larger terms) by applying the following rules of equivalence between expressions:

$(P \rightarrow Q)$	is equivalent to	$not(P) \; or \; Q$
$not(P \; and \; Q)$	is equivalent to	$not(P) \; or \; not(Q)$

Show precisely how the equivalence rules are applied.

5. You are given the following set of sets of propositions in clausal form (where the elements of each set are assumed to be disjoint, so $[not(a), b]$ is equivalent to $not(a) \; or \; b$).

$$[\,[a],\; [not(a), b]\,]$$

Apply resolution to these sets of propositions to determine whether they allow the conclusion that proposition b is true. Show the steps of inference in detail.

6. The genetic information contained in DNA is often expressed as a sequence of the characters `a`, `c`, `g` and `t`. Only certain sequences of these characters are permitted; valid sequences being restricted (for the purposes of this question) to those satisfying the following conditions:

- A *gene* sequence consists of a *catbox* sequence, followed by a *base* sequence, followed by a *tatabox* sequence.

- A *catbox* sequence consists of a *pyrimidine* followed by the sequence `[c,a,a,t]`.
- A *base* sequence can EITHER be empty (consisting of no characters) OR can contain a *purine* followed by a further *base* sequence OR can contain a *pyrimidine* followed by a further *base* sequence.
- A *purine* is EITHER the character `a` OR the character `g`.
- A *pyramidine* is EITHER the character `c` OR the character `t`.
- A *tatabox* sequence consists of the sequence `[t,a,t,a]` followed by a *purine*, followed by the character `a`.

(a) Draw a Finite State Machine (FSM) that will accept sequences that are valid according to the definition above and will reject those that are invalid. Indicate which parts of your FSM diagram correspond to the catbox, base and tatabox components. Your machine can be non-deterministic.

(b) Suppose that we want to construct an alternative FSM acceptor. In this acceptor a *gene* sequence consists of a *catbox* sequence, followed by a *base* sequence, followed by a *tatabox* sequence, followed by a second *base* sequence. The acceptor must also ensure that the number of occurrences of the character "`a`" is the same in the first and second *base* sequences. Describe briefly how you would extend your original FSM to deal with this new problem or, if it is not possible to construct the FSM, explain why.

7. You are given the following description of the events that take place when someone, who we shall call "Dave", answers a telephone:

   Dave waits for the phone to ring. If it rings he picks up the phone and asks the caller who he/she wants to contact. If the call is to a wrong number then Dave informs the caller of this and hangs up the phone. If the caller wants to contact someone who lives with Dave then he either brings that person to the phone and they talk to the caller or, if that person is not at home, Dave tells the caller to ring back later and hangs up the phone. If the call is for Dave himself then he talks to the caller. After some time talking to the caller, Dave (or the person he brings to the phone) will say goodbye and hang up the phone. After the phone has been hung up Dave waits for it to ring again.

   (a) Identify all the individual states in the telephone answering scenario above.

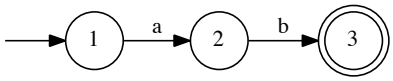
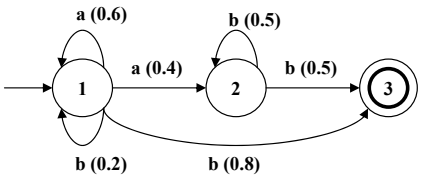


Figure 1: M_1

(b) Define a finite state machine that models the behaviour of this telephone answering system.

(c) Briefly explain whether or not your finite state machine is deterministic.

8. The diagram below describes a Probabilistic Finite State Machine that accepts strings over the alphabet `{a, b}`. Each transition between states has a probability of occurrence (for example the transition between state 1 and 2 accepting character "`a`" has probability 0.4.



Explain how you would calculate the probability that the string `aabb` would be accepted by this FSM.

9. Given the machines M_1 (Figure 1), M_2 (Figure 2) and M_3 (Figure 3), compose new machines:

(a) $M_1 M_2^*$.

(b) $M_1 | (M_3 M_2)$.

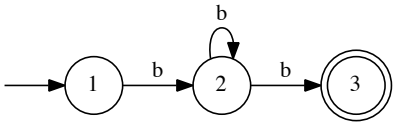


Figure 2: M_2

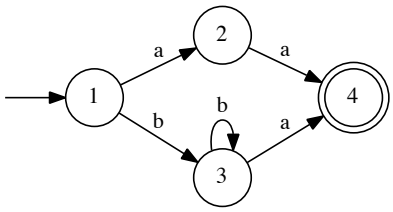


Figure 3: M_3