The University of Melbourne
Semester 2 Assessment 2009

Department of Computer Science and Software Engineering

433-295 Discrete Structures

Reading Time: 15 minutes

Exam Duration: 2 hours

This paper has 4 pages, including this front page.

Identical Examination Papers: None

Common Content: None

Authorised Materials:
This is a closed book exam. Electronic devices, including calculators and laptop computers are **not** permitted.

Calculators:
No calculators are permitted.

Instructions to Invigilators:
Students should be provided with a script book. They may retain the exam paper.

Instructions to Students:
This examination counts for 60% of the total assessment in the subject (40% being allocated to assignments during semester). There are 5 questions of equal weight—all should be attempted. Make sure that your answers are *readable.* Any unreadable parts will be considered wrong. For each sub-question, the weight is indicated. Be careful to allocate your time according to the value of each question. The marks add to a total of 60.

Question 1

a. Consider the set $S = \{\varphi_1, \varphi_2, \varphi_3, \varphi_4, \varphi_5, \varphi_6\}$ of propositional formulas, where the six elements are

$$\begin{array}{ll} \varphi_1: & (P \land Q) \Rightarrow R \\ \varphi_2: & (P \lor Q) \Rightarrow R \\ \varphi_3: & (P \Rightarrow R) \land (Q \Rightarrow R) \\ \varphi_4: & (P \Rightarrow R) \lor (Q \Rightarrow R) \\ \varphi_5: & P \Rightarrow (Q \Rightarrow R) \\ \varphi_6: & Q \Rightarrow (P \Rightarrow R) \end{array}$$

Partition S into equivalence classes: Formulas that are logically equivalent should be grouped together. [6 marks]

b. A propositional formula is in *negation normal form* (NNF) iff it uses only $\land$ (conjunction, and), $\lor$ (disjunction, or), and $\neg$ (negation, not), and moreover, negation is only applied to variables. For example, (1) below is *not* in NNF.

$$(P \land \neg Q) \lor \neg(Q \lor R) \tag{1}$$

For any given propositional formula φ, we can always find an equivalent formula in NNF. For example, (2) below is equivalent to (1), and (2) is in NNF.

$$(P \land \neg Q) \lor (\neg Q \land \neg R) \tag{2}$$

Note that a formula does not have to be in conjunctive normal form, nor in disjunctive normal form, to be in negation normal form. Also note that NNF is not a *canonical* form. For example, (3) below is *also* in NNF, and is also equivalent to (1).

$$(P \lor \neg R) \land \neg Q \tag{3}$$

Consider the type `BoolExp`, defined as follows:

```
data BoolExp
  = Var String            -- Propositional variable
  | Not BoolExp           -- Negation
  | And BoolExp BoolExp   -- Conjunction
  | Or BoolExp BoolExp    -- Disjunction
```

Write a Haskell function `nnf :: BoolExp -> BoolExp` which takes an arbitrary formula of type `BoolExp` and turns it into an equivalent formula in negation normal form. [6 marks]

Question 2

Consider the following closed first-order predicate logic formulas:

$$\begin{array}{rcl} F_1 & = & \forall x\ (R(x,x)) \\ F_2 & = & \forall x\ \forall y\ (R(x,y) \Rightarrow R(y,x)) \\ F_3 & = & \forall x\ \forall y\ \forall z\ (R(x,y) \wedge R(y,z) \Rightarrow R(x,z)) \\ F_4 & = & \forall x\ \forall y\ (R(x,y) \Rightarrow R(x,x)) \end{array}$$

a. Show that $(F_2 \wedge F_3) \Rightarrow F_1$ is not valid. [2 marks]

b. Suppose somebody rejects the result of the previous question, claiming to have a proof of the contrary. They say that every symmetric transitive binary relation R (over some set A) must necessarily be reflexive. The argument goes:

> "Let a be any element of A. Let b be any element of A for which $R(a,b)$ holds. Since R is symmetric, $R(b,a)$ also holds. Since R is transitive, and we have both $R(a,b)$ and $R(b,a)$, we must have $R(a,a)$. That is, R is reflexive."

Explain what is wrong with that argument. [2 marks]

c. Using resolution, show that $(F_2 \wedge F_3) \Rightarrow F_4$ is valid. [8 marks]

Question 3

Recall that a function $f : X \to Y$ is *injective* iff $f(x) = f(y) \Rightarrow x = y$, and that f is *surjective* iff $f[X] = Y$, that is, the range of f is *all* of the co-domain Y. Also recall that f is *bijective* iff it is both injective and surjective.

a. Give an example of a function which is injective but not surjective. [1 mark]

b. Give an example of a function which is surjective but not injective. [1 mark]

c. Let $f : X \to Y$ be a function for which an injective left-inverse exists, that is, there is some injective $g : Y \to X$ such that $g \circ f$ is the identity function on X (for all $x \in X$, $g(f(x)) = x$). Show that f is bijective. [4 marks]

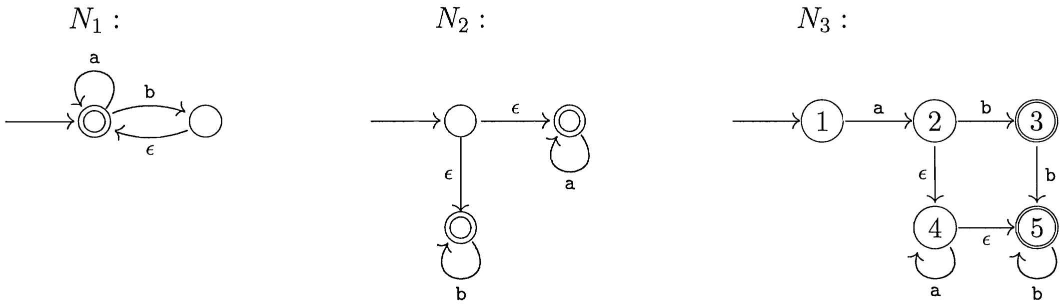
d. Write a Haskell function

```
isInjectiveFct :: (Eq a, Eq b) => [(a,b)] -> Bool
```

which will determine whether a (finite) binary relation, represented as a list of pairs, is an injective function. You may use any Prelude function, including `fst` and `snd` (which return the first and second components from a pair), and also the library function `nub`, which removes duplicates from a list. [6 marks]

Question 4

Below are three non-deterministic finite-state automata recognising languages over the alphabet $\{\mathtt{a}, \mathtt{b}\}$.



a. Give a regular expression for $L(N_1)$, the language recognised by N_1. [1 mark]

b. Give a regular expression for $L(N_2)$, the language recognised by N_2. [1 mark]

c. Give a regular expression for $L(N_2)^c$, that is, the complement of $L(N_2)$. [3 marks]

d. Using the subset construction method, transform N_3 to an equivalent DFA. Label the DFA's states so that it is clear how you obtained the DFA from the NFA. [4 marks]

e. Give the simplest possible regular expression for $L(N_3)$, the language recognised by N_3. [3 marks]

Question 5

a. Show, using mathematical induction, that every integer greater than 17 can be written as a sum of 4s and 7s. That is, for every $n > 17$, there exist non-negative integers i and j such that $n = 4i + 7j$. [4 marks]

b. Let G be the following *ambiguous* context-free grammar:

$$S \rightarrow \epsilon \mid S\, a\, a\, a\, a \mid S\, a\, a\, a\, a\, a\, a\, a$$

Describe a string that demonstrates the ambiguity of G, that is, a string which has two different parse trees. [2 marks]

c. Find an unambiguous context-free grammar equivalent to G. You may use the result in part (a) even if you didn't answer that part. [6 marks]