CS_M23

FORMAL METHODS FOR SYSTEM RELIABILITY

(*Attempt 2 questions out of 3*)

Question 1

(a) Prove that Superman doesn't exist.

To do this you may make the following assumptions:

P_1: If Superman were able and willing to prevent evil, he would do so.

P_2: Superman does not prevent evil.

P_3: If Superman were unable to prevent evil, he would be impotent; and if he were unwilling to prevent evil, he would be malevolent.

P_4: If Superman exists, he is neither impotent nor malevolent.

Carry out your proof as follows. First introduce the following abbreviations:

$a =$ "Superman is able to prevent evil."
$w =$ "Superman is willing to prevent evil."
$i =$ "Superman is impotent."
$m =$ "Superman is malevolent."
$p =$ "Superman prevents evil."
$e =$ "Superman exists."

(i) The first assumption translates into the following logical statement:

$$P_1 \;=\; (a \wedge w) \rightarrow p.$$

Translate the remaining assumptions P_2, P_3 and P_4.

[3 marks]

(ii) Use assumptions P_1 and P_2 to prove that $\neg a \vee \neg w$.

[3 marks]

(iii) Use assumption P_3, and the fact proved in (ii), to prove that $i \vee m$.

[2 marks]

(iv) Use assumption P_4, and the fact proved in (iii), to draw your conclusion.

[2 marks]

(Please turn over.)

(b) Define the following sets:

S = the set of all students;

M = the set of all modules

and the following relations:

$R = \{\, m \in M \mid \text{module } m \text{ is a required module} \,\}$;

$T = \{\, (s, m) \in S \times M \mid \text{student } s \text{ takes module } m \,\}$;

$G = \{\, (s, g) \in S \times \{1, 2.1, 2.2, 3\} \mid \text{student } s \text{ graduates with a grade } g \,\}$.

Translate the following into logical statements:

(i) Every student takes every required module.

[2 marks]

(ii) No student who graduated with a 3rd (that is, with a grade of 3) took every module.

[2 marks]

(iii) At most two students graduate with a 1st (that is, a grade of 1).

[3 marks]

(c) In this question, R represents a binary relation on a set A, and $I = \{(x, x) \mid x \in A\}$ represents the identity relation in A.

Match up each of the properties in the left column with the appropriate property in the right column, and write the property out as a logical statement.

reflexive	$R \circ R \subseteq R$
irreflexive	$R \cap R^{-1} = \emptyset$
symmetric	$I \subseteq R$
asymmetric	$R \cap R^{-1} \subseteq I$
antisymmetric	$R^{-1} = R$
transitive	$R \cap I = \emptyset$

(Note: *irreflexive* means that no element is related to itself; and *asymmetric* means that no element is related to any element which is related to it.)

[8 marks]

Question 2

In the game of NIM, an arbitrary number of piles of coins are formed, each with an arbitrary number of coins in them, and two players alternate in removing one or more coins from any one pile. The player who takes the last coin is declared to be the winner.

(a) Explain carefully how you can determine who has the winning strategy in NIM, and which moves are the winning moves. Use examples, and make it clear why the procedure works.

[8 marks]

(b) Without referring to the above general theory of NIM, argue that in 2-pile NIM:

- the 2nd player has the winning strategy if the piles contain an *equal* number of coins;
- the 1st player has the winning strategy if the piles contain an *unequal* number of coins.

[4 marks]

(c) Who has the winning strategy in NIM when you start with n piles each containing the same number of coins? Explain your answer.

[3 marks]

Consider the following process definition.

$$X \stackrel{\text{def}}{=} a.X + a.Y + a.b.Z$$
$$Y \stackrel{\text{def}}{=} a.Z + b.X$$
$$Z \stackrel{\text{def}}{=} a.Y + a.Z + b.X$$

(d) Draw a transition system which includes the states X, Y and Z as defined above. Your transition system should have as few states and transitions as possible. Label each state with the appropriate process expression.

[4 marks]

(e) For which n is it the case that $X \sim_n Y$? Explain.

[2 marks]

(f) For which n is it the case that $X \sim_n Z$? Explain.

[2 marks]

(g) For which n is it the case that $Y \sim_n Z$? Explain.

[2 marks]

Question 3

In this question we re-examine the Railway Level Crossing System (see the following page).

(a) Consider the *safety* property that a car may not cross at the same time as a train.

(i) Explain informally why this property is satisfied at every state of the System.

(ii) Give an expression of the modal logic $\mathcal{M}$ which expresses this property.

[5 marks]

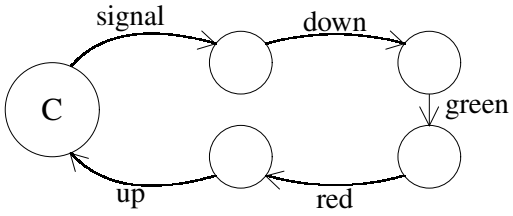
(b) Consider the *liveness* property that if a car arrives, eventually the barrier may go up.

(i) Explain informally why this property is satisfied at every state of the System.

(ii) How does this property differ from the property that if a car arrives, eventually the barrier *will* go up.

[5 marks]

Normally, a barrier remains up until a train arrives; this signals the controller, which then lowers the barrier, then turns the signal green, then turns the signal red again, and finally raises the barrier once again. The new controller C is thus represented by the following automaton:



(c) Give a definition for C. (This includes defining its sort $\mathcal{L}(\mathrm{C})$.)

[5 marks]

(d) Give the definitions and associated automata for the new Road and Rail systems Ro and Ra, respectively, which correspond to the new Controller C. (Keep in mind that the new Road system starts in a state where the barrier is up; and the new Rail system must signal the controller when a train arrives, using the new event "signal" common to their sorts.)

[5 marks]

(e) Now consider the liveness properties again.

(i) Is it now the case that, if the barrier is down when a car arrives, then the barrier will eventually go up?

(ii) Is it now the case that, if the signal is red when a train arrives, then the signal will eventually turn green?

[5 marks]

The Railway Level Crossing System

Concurrent Processes

$E \parallel F$: Executing processes E and F concurrently.

Synchronisation Sorts

- Each process E has a ***synchronisation sort*** $\mathcal{L}(E)$.
- Every state of a process has the same sort.
- Processes running concurrently must synchronise on actions which are common to their respective sorts.
- $\mathcal{L}(E \| E) = \mathcal{L}(E) \cup \mathcal{L}(F)$

Synchronisation Merge: $E \parallel F$

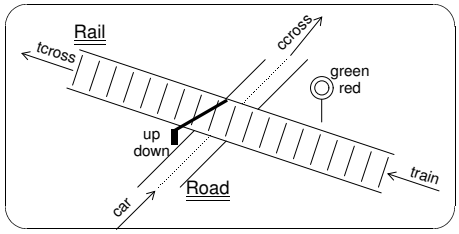
If $E \xrightarrow{a} E'$ and $a \notin \mathcal{L}(F)$ then $E\|F \xrightarrow{a} E'\|F$.

If $F \xrightarrow{a} F'$ and $a \notin \mathcal{L}(E)$ then $E\|F \xrightarrow{a} E\|F'$.

If $E \xrightarrow{a} E'$ and $F \xrightarrow{a} F'$ and $a \in \mathcal{L}(E) \cup \mathcal{L}(F)$ then $E\|F \xrightarrow{a} E'\|F'$.

Example: Railway Level Crossing

Consider the following railway level crossing.

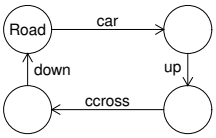


We can view this as three processes working in parallel:

- A `Rail` process, which represents the arrival of trains, assuring that they only cross if the signal is green.
- A `Road` process, which represents the arrival of cars, assuring that they only cross if the barrier is up.
- A `Controller` process, which regulates the signal and barrier, assuring that the barrier is never up at the same time that the signal is green.

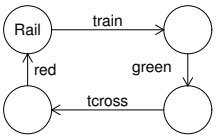
Railway Components

`Road` $\stackrel{\text{def}}{=}$ `car.up.ccross.down.Road`



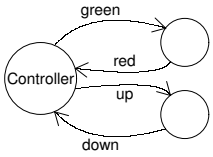
$\mathcal{L}(\text{Road}) = \{ \text{up}, \text{down} \}$

`Rail` $\stackrel{\text{def}}{=}$ `train.green.tcross.red.Rail`



$\mathcal{L}(\text{Rail}) = \{ \text{green}, \text{red} \}$

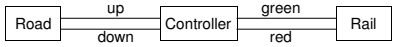
`Controller` $\stackrel{\text{def}}{=}$ `green.red.Controller` $+$ `up.down.Controller`

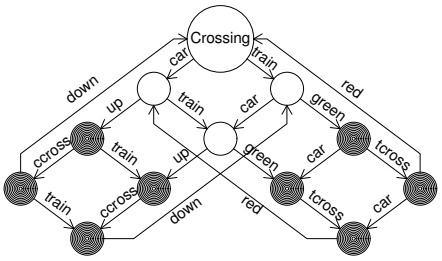


$\mathcal{L}(\text{Controller}) = \{ \text{green}, \text{red}, \text{up}, \text{down} \}$

The Complete Railway System

`Crossing` $\stackrel{\text{def}}{=}$ `Road` $\parallel$ `Controller` $\parallel$ `Rail`





Desirable Properties:

Safety Properties: (***No crashes***)

- A car may not cross at the same time as a train.

Liveness Properties: (***Eventual service***)

- If a car arrives, eventually the barrier may go up.
- If a train arrives, eventually the signal may turn green.