PRIFYSGOL CYMRU; UNIVERSITY OF WALES

DEGREE EXAMINATIONS MAY/JUNE 2002

SWANSEA

Computer Science

CS 116 Modelling Computing Systems

Attempt 2 questions out of 3

Time allowed: 2 hours

Students are permitted to use the dictionaries provided by the University

Students are NOT permitted to use calculators

CS_116
MODELLING COMPUTING SYSTEMS
(*Attempt 2 questions out of 3*)

Question 1

(a) 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. Whoever takes the last coin is declared to be the winner.

Assuming both players play optimally, determine which player will win the game of NIM when played with four piles containing the following numbers of coins:

(i) 1, 3, 4, 5
(ii) 2, 3, 4, 5
(iii) 3, 3, 4, 5
(iv) 4, 3, 4, 5

In each case, justify your answer referring to the general theory of NIM (balanced versus unbalanced positions). In the cases in which the first player will win, give *all* possible winning first moves.

[8 marks]

(b) Suppose we change the rules slightly so that the first player, instead of removing some coins from a pile, has the additional option of *creating a new pile* of any size (with at least one coin in it); the first player may do this at most once during a play of the game. Under which circumstances can the first player force a win with the help of this extra move? (Consider, in particular, the two situations in which the game starts from an unbalanced, respectively a balanced, position.) Justify your answer.

[5 marks]

(c) In **POOR MAN'S NIM**, there is only one pile, and the two players alternately remove either 1, 2 or 3 coins from the pile. Again, the player to take the last coin wins.

(i) For each number n from 1 to 10, explain who has the winning strategy in POOR MAN'S NIM starting from a pile of n coins. In the cases in which the first player has the winning strategy, state how many coins (1, 2 or 3) the first player should take.

[6 marks]

(ii) Generalize the above by explaining who has the winning strategy in POOR MAN'S NIM starting from a pile of n coins for an arbitrary n.

[3 marks]

(iii) Generalise the above further by explaining who has the winning strategy in POOR MAN'S NIM starting from a pile of n coins for an arbitrary n, but where players may alternately remove between 1 and k coins (above, we had k = 3).

[3 marks]

Question 2

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 M which expresses this property.

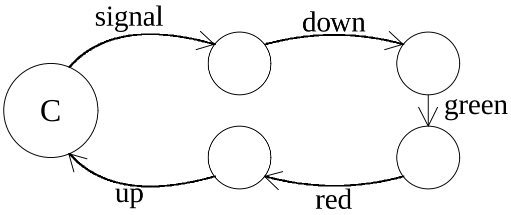
[5 marks]

(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

Synchronisation Merge

Synchronisation Sorts

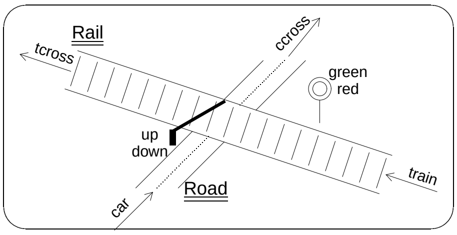
- Each process E has a ***synchronisation sort*** $\mathcal{L}(E)$.
- Every state of a process has the same sort.
- Concurrent processes must synchronise on actions which are common to their respective sorts.

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

Example: Railway Level Crossing

Consider the following railway level crossing.

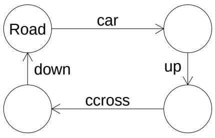


We can view this as three processes working in parallel:

- 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.

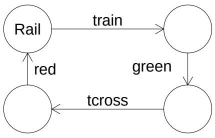
Railway Components

$$\texttt{Road} \overset{\text{def}}{=} \texttt{car.up.ccross.down.Road}$$



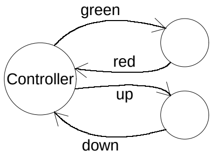
$\mathcal{L}(\text{Road}) = \{\text{up}, \text{down}\}$

$$\texttt{Rail} \overset{\text{def}}{=} \texttt{train.green.tcross.red.Rail}$$



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

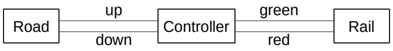
$$\texttt{Controller} \overset{\text{def}}{=} \texttt{green.red.Controller} + \texttt{up.down.Controller}$$

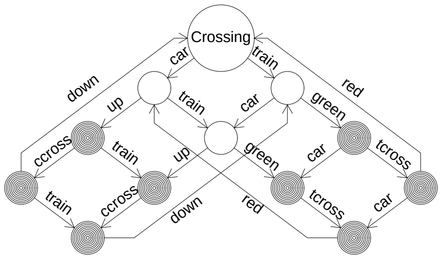


$\mathcal{L}(\text{Controller}) = \{\text{green}, \text{red}, \text{up}, \text{down}\}$

The Complete Railway System

$$\texttt{Crossing} \overset{\text{def}}{=} \texttt{Road} \parallel \texttt{Controller} \parallel \texttt{Rail}$$





Desirable Properties:

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

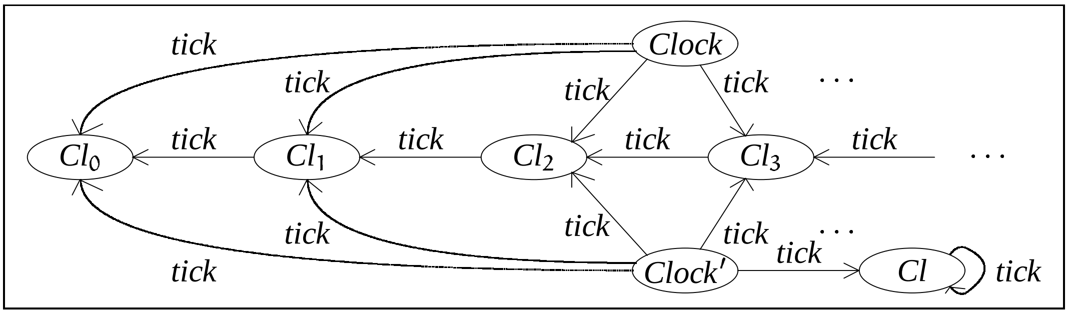
- A car may not cross at the same time as a train.

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

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

Question 3

(a) Recall the following clock processes.



(i) Explain why $Clock \sim_n Clock'$ for every $n \in \mathbb{N}$. That is, explain how the second player (Bob) can win any bisimulation game of any length $n \in \mathbb{N}$ which starts with the tokens on *Clock* and *Clock*′.

[4 marks]

(ii) Explain why $Clock \not\sim Clock'$.

[4 marks]

(iii) Explain why no formula of the modal logic M can distinguish between *Clock* and *Clock*′.

[4 marks]

(iv) Prove, by induction on n, that for all $n \in \mathbb{N}$ $Cl_n \sim_n Cl_{n+1}$.

[4 marks]

(b) Consider the following two statements about a computer:

(i) "The computer consists of three parts: a CPU, a memory unit, and a bus for communication with the environment."

(ii) "The emergency button can be pushed; this will halt the computer, which will then not do anything further."

One of these statements can be expressed in the modal logic M; express it.

[3 marks]

The other statement cannot be formalized in M; explain why not.

[3 marks]

(c) Give an example of a pair of processes E and F and a pair of formulas P and Q of the modal logic M such that

$$E \models P \quad \text{and} \quad F \models Q \quad \text{but} \quad E + F \not\models P \wedge Q.$$

[3 marks]