CS_116 (2005-2006)
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.

Consider the game played with three piles where one of the piles has exactly one coin; that is, from the configuration $\langle 1, m, n\rangle$ where (without loss of generality) $1 \leq m \leq n$.

(i) Who has the winning strategy when $m=1$, ie, in the game $\langle 1, 1, n\rangle$? Justify your answer. **[2 marks]**

(ii) Who has the winning strategy when $m=n$, ie, in the game $\langle 1, m, m\rangle$? Justify your answer. **[2 marks]**

(iii) Who has the winning strategy when $m=2$ and $n=3$, ie, in the game $\langle 1, 2, 3\rangle$? Justify your answer. **[2 marks]**

(iv) Who has the winning strategy when $m=2$ and $n>3$, ie, in the game $\langle 1, 2, n\rangle$ with $n>3$? Justify your answer. **[2 marks]**

(v) Argue that the second player has the winning strategy when m is even and $n=m+1$. **[3 marks]**

(vi) Argue that the first player has the winning strategy in all cases other than those given in (v) above. **[3 marks]**

(b) 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. **[5 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 $\mathcal{M}$ which expresses this property.

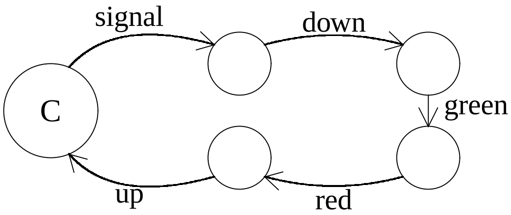
[5 marks]

(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 LTS:



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

[5 marks]

(d) Give the definitions and associated LTS 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

$E \,\|\, F$: 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: $E \,\|\, F$

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

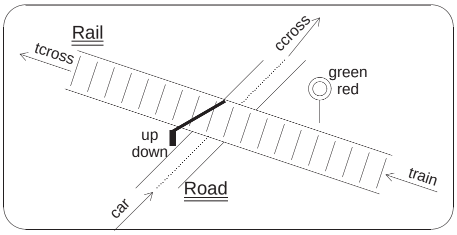
If $F \xrightarrow{a} F'$ and $a \notin \mathcal{L}(E)$ then $E\|F \xrightarrow{a} E\|F'$.

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.

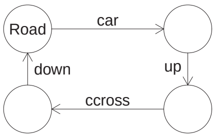


We can view this as three processes working in parallel:

- 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.

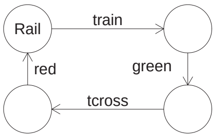
Railway Components

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



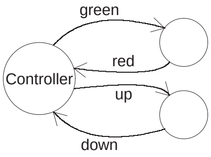
$\mathcal{L}(\text{Road}) = \{\,\text{up, down}\,\}$

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



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

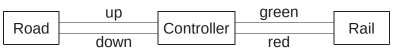
$$\begin{aligned}\texttt{Controller} \overset{\text{def}}{=} \;& \texttt{green.red.Controller} \\ +\;& \texttt{up.down.Controller}\end{aligned}$$

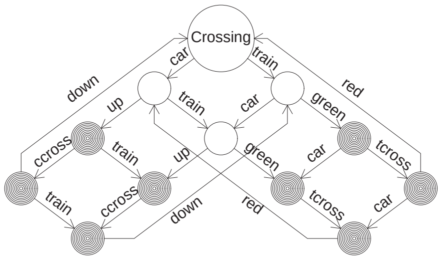


$\mathcal{L}(\text{Controller}) = \{\,\text{green, red, up, down}\,\}$

The Complete Railway System

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





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) The syntax of the Modal Logic $\mathcal{M}$ is given by the following BNF equation:

$$P, Q \quad ::= \quad \mathit{true} \;\mid\; \mathit{false} \;\mid\; P \vee Q \;\mid\; P \wedge Q \;\mid\; \langle a \rangle P \;\mid\; [a]P$$

where a denotes an action.

(i) Define the satisfaction relation $E \models P$, where E is a process and P is a term of $\mathcal{M}$. That is, for each property P of $\mathcal{M}$, say when a process E satisfies P. Your definition should look as follows (the definitions in the first two cases are provided):

$$\begin{array}{lll} E \models \mathit{true} \text{ is always true} & \quad E \models P \vee Q \;\; \mathit{iff} \; \cdots & \quad E \models \langle a \rangle P \;\; \mathit{iff} \; \cdots \\ E \models \mathit{false} \text{ is never true} & \quad E \models P \wedge Q \;\; \mathit{iff} \; \cdots & \quad E \models [a]P \;\; \mathit{iff} \; \cdots \end{array}$$

[6 marks]

(ii) Define the negation $\overline{P}$ of a property P of $\mathcal{M}$. That is, for each property P of $\mathcal{M}$, give a property $\overline{P}$ which is true of a state if and only if P is not true in that state. Your definition should look as follows (the definitions in the first two cases are provided):

$$\begin{array}{rclrclrcl} \overline{\mathit{true}} & = & \mathit{false} & \quad \overline{P \vee Q} & = & \cdots & \quad \overline{\langle a \rangle P} & = & \cdots \\ \overline{\mathit{false}} & = & \mathit{true} & \quad \overline{P \wedge Q} & = & \cdots & \quad \overline{[a]P} & = & \cdots \end{array}$$

[3 marks]

(iii) Define the modal depth $\mathit{md}(P)$ of a property P of $\mathcal{M}$. Your definition should look as follows (the definitions in the first two cases are provided):

$$\begin{array}{rclrclrcl} \mathit{md}(\mathit{true}) & = & 0 & \quad \mathit{md}(P \vee Q) & = & \cdots & \quad \mathit{md}(\langle a \rangle P) & = & \cdots \\ \mathit{md}(\mathit{false}) & = & 0 & \quad \mathit{md}(P \wedge Q) & = & \cdots & \quad \mathit{md}([a]P) & = & \cdots \end{array}$$

[3 marks]

(b) Consider the following two processes:

$$E \overset{\text{def}}{=} a.(b.\mathbf{0} + c.\mathbf{0}) \qquad F \overset{\text{def}}{=} a.b.\mathbf{0} + a.c.\mathbf{0}$$

(i) Draw the LTS for the process term $E{+}F$. Use as few states as possible, and label every state with the process term it represents.

[3 marks]

(ii) Give properties P and Q of $\mathcal{M}$ such that $E \models P$ and $F \models Q$, but $E \not\models Q$ and $F \not\models P$.

[6 marks]

(iii) Give a property R of $\mathcal{M}$ such that $E \models R$ and $F \models R$ but $E + F \not\models R$.

[4 marks]