Session 5: LTC

ingmar.dasseville@cs.kuleuven.be
laurent.janssens@cs.kuleuven.be

November 8, 2016

1 Garage door controller

If you have not completed this exercise last week, do it now. Otherwise move on to the next exercises.

In this exercise, we are specifying a simple controller for a garage door. A garage is operated by a controller and two remotes both with an “open” and a “close” button. The controller has several state variables, the most important are:

- “position” is a dynamic function taking values in 1,..,5 to represent the position of the port: 1 is closed and 5 is opened.
- “opening_door” represents that the door is opening.
- “closing_door” represents that the door is closing.

As long as “opening_door” is true, the port opens up, one position per time step, until the door is fully open, at which time “opening_door” is switched off. The inverse happens when “closing_door” is true. The two remotes can be used simultaneously and when they issue contradictory orders, opening has priority to closing the garage, to avoid damage to cars. The initial state is that the door is closed and that both “opening_door” and ”closing_door” are false.

- Write down this model in the linear time calculus.
- Write the axioms for the guided simulation:
  - $t = 0$: door is closed.
  - $t = 1$: driver A uses one remote to open door.
  - $t = 10$: driver A has left the garage and uses the same remote to close the door
  - $t = 12$: driver B wants to enter the garage and uses the second remote to open door.

  In what state is the controller at t = 13?

- A desired implicit invariant is that “opening_door” and “closing_door” are not true simultaneously. Do you have an intuition as to how you would prove this invariant? There is a weak and a strong way to check it.

2 Nim

Nim is a game for two players. In the initial situation, a set of matches are on the table in the form of a set of heaps of increasing size, respectively 1,3,5,7,... . When modeling the game, you may assume that there are only four heaps. In turns, the players take a number of matches away from a heap: at least one and at most as much as there are in the heap. The player that takes the last match loses the game. Use the following vocabulary:

- Type int Time
- Type Player : Players in the game
- Type Matches : a finite integer type, representing a number of matches
- Type Heap : Heaps in the game
- Predicate Nb(Heap,Matches,Time) : number of matches on the heap.
- Predicate Turn(Player,Time) : the player is on turn
- Predicate Winner(Player,Time) : the player wins the game
- Predicate GameOver(Time) : the game is game over, i.e. no move is possible
- Predicate Takes(Player,Heap,Matches,Time) : the action

Once you have your theory, you can run a number of verification tasks. Do this by creating additional theories representing the questions and checking whether they are satisfiable when merged with your theory.

- Either player has the possibility to win the game
- In every state, it is the turn of exactly one player
- Every game comes to an end. (You might need to check this for only three heaps, for computational reasons)
- The game is not over until all the heaps are empty
- There is a game that last for exactly 16 moves
- There is a first move such that if the first player plays it on time 0, then the game is over on time 1. This proposition is obviously false. Then try to verify whether the following formula is entailed your theory: $\exists h1\ \exists nr1 (Takes(P1, h1, nr1, 0) \Rightarrow GameOver(1))$

- Suppose the first player wants the game to end as quickly as possible. How many moves does he need?
  Hint: use two specifications: one to show that the first player can always make sure that the game ends after n moves, and one to show that this is not the case for n-1.

3 Vending Machine

Starting from the VendingMachine.idp skeleton provided on Toledo, model a beverage vending machine using LTC. The aim is to model the vending machine from a user's standpoint, i.e. the actions that are available are actually actions performed by a user.

The users can pay one coin at a time (the coins are represented per 10 cents, e.g. 1 is a 10 cent coin), press a button to receive a certain beverage, ask for a refund or restock the machine. The machine keeps track of the current amount paid, as well as the stock of the different beverages. Each beverage has a certain price.

The actions work as follows:

pay(c) denotes a user inserting a coin of value c. The value of the coin should be added to current amount paid. However the machine will not accept anymore coins when the current amount paid is already larger than the price of the most expensive beverage.

press_button(b) means a users pressed the button for beverage b. Users can press a button at any time, but will only receive a beverage if they have paid at least the beverage's price. IF the user receives the beverage, its price is deducted from the currently paid amount.

ask_refund happens when the user asks to have his money returned. The currently paid amount is returned to the user. Obviously this is only possible if there is some of amount of currency in the machine.

restock allows the machine to be restocked. It is only meaningful to restock the machine when the stock of at least one of the beverages is below its maximum.

Only one action can happen at any one time.

The machine has an obvious short coming, i.e. when you select a beverage and paid more than the price, it does not automatically return the remaining money. To get the rest of the money back a user has ask for a refund.

Tasks

1. Model this scenario by means of Linear Time Calculus.
2. Use IDP to simulate the workings of the vending machine.

3. Check that it is impossible to have an amount over 24 in the machine. Use the isinvariant(T, T', S) procedure for this. This returns true if T' can be proven to be an invariant of T in structure S. T' has to be a theory with a single sentence of the form $\forall t[Time] : \phi[t]$.

4. This only shows this is an invariant for the the given structure. Complete the single state and bistate invariants and the provided procedure to check whether it is actually an invariant of the theory.

5. Check that there can not be two restocks following each other. Why is this not really an invariant (consult the definition of an invariant)?