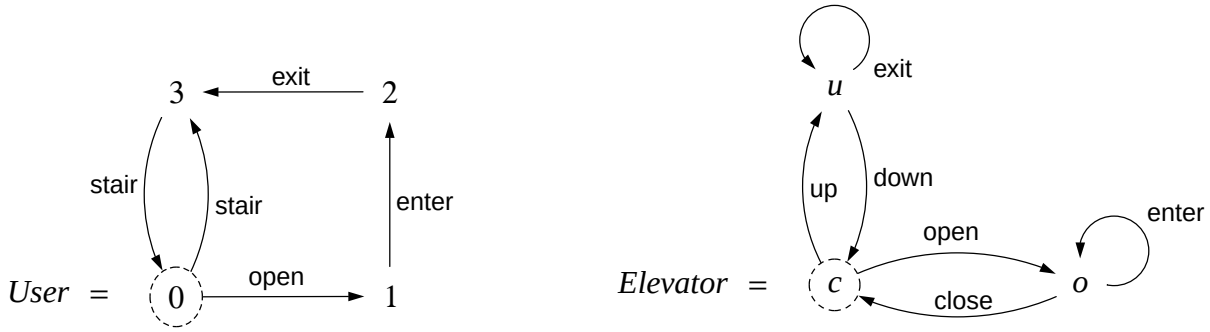
McMaster University
Department of Computing and Software
Dr. W. Kahl

SFWR ENG 3BB4 — Software Design III — Concurrent System Design

27 February 2007

Exercise 3.1: Labelled Transition Systems — Elevator (extended from Midterm 2, 2004)

Let the following two labelled transition systems be given:

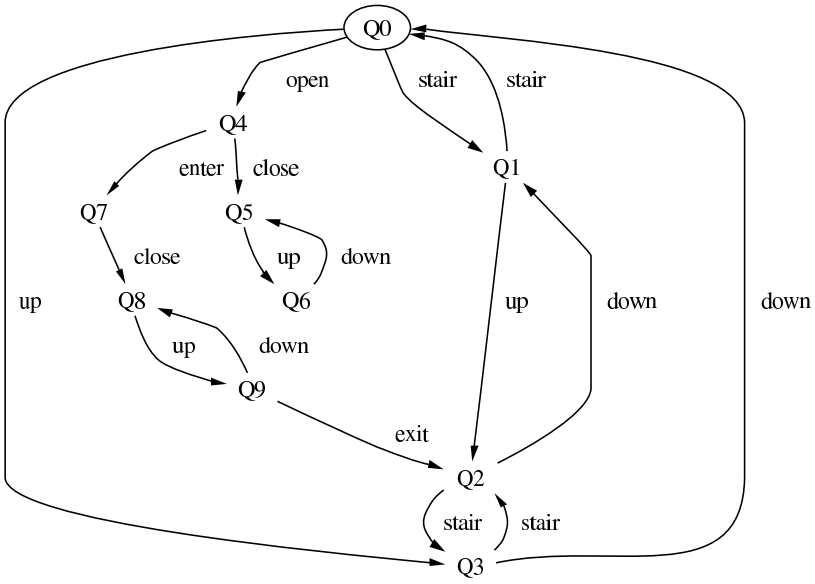


(a) Draw the reachable part of the composed LTS *Elevator* ‖ *User*.

Solution Hints

```
Elevator = (open -> Open1 | up -> Up),
Open1 = (enter -> Open1 | close -> Elevator),
Up = (exit -> Up | down -> Elevator).

User = (open -> enter -> In | stair -> Out2),
In = (exit -> Out2),
Out2 = (stair -> User).
```



(b) In the system *Elevator* || *User*, what "goes wrong" relative to common sense?
Clearly define the property you find **violated** as a property of traces. Is this a **safety property** or a **liveness property**? **Justify!**

Solution Hints

Common sense description:

– ***Expected:*** If the *User* does not enter quick enough, they keep standing outside the closed *Elevator*, and although the *Elevator* can still go up and down, the *User* ***cannot*** ever do anything anymore.

Violated property as trace predicate:

- Each trace contains infinitely many events in the label set of *User*.
- ***Equivalently***: For each trace t and each position $m : \mathbb{N}$ there is a position $n : \mathbb{N}$ with $n \geq m$ such that $t.n \in \mathbf{L}_{User}$.

This is a **liveness property**.

(c) For each of the following questions, if the answer is yes, supply a trace; if no, explain. Does the statement reflect a **safety property** or a **liveness property**? Reformulate the property involved as a statement about traces.

– Is it possible that after an enter event happend in the system *Elevator* || *User*, no exit will happen anymore?

Solution Hints

Yes: open enter close (up down)$^\infty$

The statement reflects violation of a **liveness property**.

In each trace, after each enter event there is eventually an exit event.

Equivalently: For each trace t and each position $m : \mathbb{N}$ with $t.m =$ enter there is a position $n : \mathbb{N}$ with $n > m$ such that $t.n =$ exit.

– Is it possible that after an enter event happend in the system *Elevator* $\|$ *User* an up or a down event happens before the next close event?

Solution Hints

No: enter is shared, and the user can only engage in exit, another shared event, after enter, so only the elevator can act, but the only possible action of the elevator after enter is already close.

Trace property: For a given trace, is there an enter event followed (not necessarily directly) by an up or down event before the next close event?

This is a safety property violation.

The corresponding safety property is that for every trace, and for every enter event in that trace, there are no up or down events in that trace before the next (if any) close event.

Intuitively: If the elevator started moving before the door closes, somebody might get squished — a safety concern.

Exercise 3.2: Modelling Car Keys (Adapted from Midterm 1, 2002)

You are to model certain aspects of locking a car that, for the sake of simplicity, is assumed to have only **one** door and **one** key, and **no** other parts (e.g. windows or roofs) that might be open or opened.

It is possible to lock and unlock the **door** no matter whether it is open or closed, but the door can only be opened while it is unlocked.

The **key** can be inserted into the ignition and removed from it, and while the key is *not* in the ignition, it can lock and unlock the door.

The **driver** can use the door to get into and out of the car. If the driver is outside the car and the door is closed, then the driver can use the key to lock and unlock the door; otherwise it is only possible to use a button to lock and unlock the door. If the driver is inside the car and the door is closed, then the driver can insert the key into the ignition and remove it.

(a) Draw an LTS for the behaviour of the **door**, starting in a closed and locked state.

Solution Hints

```
DOOR = (unlock -> CU),
CU = (lock -> DOOR | open -> OU),
OU = (lock -> OL | close -> CU),
OL = (unlock -> OU | close -> DOOR).
```

Refined for two locking agents:

```
DOOR = (key.unlock -> CU | butten.unlock -> CU),
CU = (key.lock -> DOOR | button.lock -> DOOR | open -> OU),
OU = (key.lock -> OL | button.lock -> OL | close -> CU),
OL = (key.unlock -> OU | button.unlock -> OU | close -> DOOR).
```

(b) Draw an LTS for the behaviour of the **key**, starting outside the ignition.

Solution Hints

```
KEY = ({key.lock, key.unlock} -> KEY | insert -> remove -> KEY).
```

(c) Draw an LTS for the behaviour of the **driver**, starting outside the closed car.

Solution Hints

```
DRIVER = ({key.lock, key.unlock} -> DRIVER
        |open -> X
        ),
X = (close -> DRIVER
   |{button.lock, button.unlock} -> X
   |close -> INSIDE
```

```
  ),
INSIDE = (open -> X
      |{button.lock, button.unlock} -> INSIDE
      |{insert, remove} -> INSIDE
      ).
```

(d) Use parallel composition to assemble the three components together into a single LTS process *CARKEY*.

(e) We call a state *desperate* if the driver is outside the vehicle, the door is closed and locked, and the key inside the ignition. Is it possible that the process *CARKEY* reaches a desperate state? If yes, exhibit a trace; if no, explain.

Solution Hints

key.unlock
open
close
insert
open
button.lock
close

(f) Are desparate states deadlock states? Explain!

Solution Hints

Yes: The driver outside the closed and locked door can only do key.unlock, but the key is inside the ignition and can only do remove, and the door has only shared actions: No actions are possible from such a state.

(g) Is the absence of desparate states a safety condition or a liveness condition? Explain!

Solution Hints

If “desperate state” as such is considered “something bad”, then this is a safety property.

But the state is “desperate” because no actions (leading to eventual driving) are possible, and this latter property is the most basic liveness property.

(h) *Infinitely many uses* of the car are documented by traces in which both unlocking by key and insertion of the key into the ignition have infinitely many occurrences.

Are infinitely many uses of the car still possible if the locking button breaks? Explain in terms of traces! What kind of property is this?

Solution Hints

This is a liveness property.

There are traces documenting infinitely many uses of the car with only finitely many occorrences of button.lock and of button.unlock.

This is the case since

– it is possible to get from key.unlock to insert without using the button, and

– it is possible to get from insert to key.unlock without using the button, cycling as follows: key.unlock
open
close
insert
remove
open
close
key.lock
key.unlock
