Lecture 10: Outline

- Case study: a lift system
- The lift system: requirements
- The lift system: models (first developed by J.-R. Abrial)
- The lift system: deconstructing a formalisation

A case study: a lift system (requirements)

1. A lift system consists of a non-zero number of lifts installed in a building
2. The building has a number of floors (more than one)
3. A lift can move between floors in one of two directions: up (unless it is already at the top floor) or down (unless it is already at the ground floor)
4. Each of floors (except ground and top) has two buttons, one to request an up-lift and one to request a down-lift
5. The ground and top floors have one button to request an up-lift and a down-lift respectively
6. When pushed, the buttons remain illuminated until a lift, traveling in the desired direction, visits a floor

The lift system requirements (cont.)

7. If both floor buttons are pushed, only respective one is cancelled after a lift visits a floor
8. Each lift has a set of buttons, one button for each floor, to request a stop at that floor
9. The buttons are illuminated when pressed and cause a lift to visit the corresponding floor (unless the lift is already at that floor)
10. The illumination is cancelled when the corresponding floor is visited
11. Each lift can either moving or stopped at some floor
12. A moving lift stops at a floor if it passes it and either (i) there are internal lift requests to stop at that floor, or (ii) there are floor requests on that floor to travel in the current lift direction

The lift system requirements (cont.)

13. When a lift stops at a floor, the floor is considered visited and all the related requests (both within the lift and on the floor) are cancelled
14. When a lift has no requests to service, it should remain at its final destination and await further requests
15. All requests for lifts from floors must be serviced eventually, with all floors given equal priority
16. All requests for floors within lifts must be serviced eventually, with floors being serviced sequentially in the direction of travel
17. Each lift keeps its current moving direction while there are floors with requests to visit at that direction
18. If there no such requests to continue on its current direction and there servicing requests in the opposite direction, a stopped lift should change its direction to the opposite one

The lift system requirements (cont.)

19. The time is not considered in the first system prototype
20. Once a lift stops at some particular floor, all people inside a lift that requested that floor are considered immediately out
21. Moreover, all people waiting on the floor to go to the lift current direction, are considered immediately in

Lift system: context

CONTEXT
Lift_ctx
SETS
LIFT
DIRECTION
CONSTANTS
up, dn, ground, top, FLOOR,
attracted_up, attracted_dn,
can_continue_up, can_continue_dn
AXIOMS
$partition(DIRECTION, \{up\}, \{dn\})$
$ground \in \mathbb{N}$
$top \in \mathbb{N}$
$ground < top$
$FLOOR = ground \,..\, top$
...

Lift system: notes

- All three ways to introduce your own type are demonstrated here: as an abstract set, as an enumerated set, as a set constant
- The data type *FLOOR* is defined as a set constant
- It is a constructed type (set), defined the number interval between two other constants *ground* and *top*
- The constants *attracted_up*, *attracted_dn*, *can_continue_up*, *can_continue_dn* will be explained later

Lift system: machine

MACHINE
  Lift_mch
SEES
  Lift_ctx
VARIABLES
  moving, floor, dir, in, out
INVARIANT
  $moving \subseteq LIFT$
  $floor \in LIFT \rightarrow FLOOR$
  $dir \in LIFT \rightarrow DIRECTION$
  $in \in FLOOR \leftrightarrow DIRECTION$
  $out \in LIFT \leftrightarrow FLOOR$
...

Lift system: notes

- $floor \in LIFT \rightarrow FLOOR$
  $dir \in LIFT \rightarrow DIRECTION$
  return the current floor and direction for a given lift
- $in \in FLOOR \leftrightarrow DIRECTION$
  contains floor requests, i.e., the requests to get in a lift going in a specific direction

  Example: $5 \mapsto dn \in in$ means that there people on the fifth floor wanting to go down
- $out \in LIFT \leftrightarrow FLOOR$
  contains lift requests to go to a particular floor, i.e., the requests to get out a lift

  Example: $l1 \mapsto 4 \in out$ means that there people in the lift $l1$ wanting to go to 4th floor

Lift system: machine (cont.)

$ground \mapsto dn \notin in$
$top \mapsto up \notin in$
$moving \lhd (out \cap floor) = \varnothing$
$\forall l \cdot l \notin moving \Rightarrow floor(l) \mapsto dir(l) \notin in$

INITIALISATION

$moving, in, out \ := \ \varnothing, \varnothing, \varnothing$
$floor := LIFT \times \{ground\}$
$dir := LIFT \times \{up\}$

...

Lift system: notes

- The first two invariants (on the last slide) constrain floor requests: it is not possible to request an down-lift from the ground floor or an up-lift from the top floor
- The last two invariants formalise the following modelling requirements:

  Once a lift stops at some particular floor, all the related floor and lift requests are cancelled. Moreover, all people inside a lift that requested that floor are considered immediately out. Moreover, all people waiting on the floor to go to the lift current direction, are considered immediately in

Deconstructing invariants

- *Once a lift stops at some particular floor, all people inside a lift that requested that floor are immediately out (and their lift requests are cancelled)*

  $moving \lhd (out \cap floor) = \varnothing$

- Deconstructing:

  $out \cap floor$
  set of pairs $LIFT \times FLOOR$ such that the lift is on some particular floor and there people requested to move to that floor

  $moving \lhd (out \cap floor)$
  the moving lifts are not considered – the domain subtraction operation keeps only such pairs where lifts are stopped

Deconstructing invariants (cont.)

- $moving \lhd (out \cap floor) = \varnothing$
  there no such pairs left

  It means that are no such requests to get out on a floor once a lift stopped on that floor

  Which can only mean that,
  for any stopped lift, all people inside a lift that requested that floor are considered immediately out (and their lift requests are cancelled)

Deconstructing invariants (cont.)

- *"... Moreover, all people waiting on the floor to go to the lift current direction, are immediately in (and their floor requests are cancelled)"*

  $\forall l \cdot l \notin moving \Rightarrow floor(l) \mapsto dir(l) \notin in$

- Deconstructing:

  For any stopped lift on some floor, there no requests from that floor to go in the direction the lift is going

  $\Rightarrow$

  all people waiting on the floor to go to the lift current direction, are already in (and their floor requests are cancelled)

Lift system: machine (cont.)

$$
\begin{array}{l}
request\_lift\_from\_floor \;=\\
\quad \textbf{ANY}\ f,\ d\\
\quad \textbf{WHEN}\ f \in FLOOR \wedge d \in DIRECTION\\
\qquad\qquad f \mapsto d \neq ground \mapsto dn\\
\qquad\qquad f \mapsto d \neq top \mapsto up\\
\qquad\qquad \forall l.\ l \notin moving \Rightarrow \neg(floor(l) = f \wedge dir(l) = d)\\
\quad \textbf{THEN}\\
\qquad in := in \cup \{f \mapsto d\}\\
\quad \textbf{END}\\
\ldots
\end{array}
$$

Lift system: notes

- The last guard is needed to prove the formulated invariant:

  $\forall l \cdot l \notin moving \Rightarrow floor(l) \mapsto dir(l) \notin in$

- Essentially stating the necessary condition: for any non-moving (stopped) lift on some floor, a lift request from this floor is only allowed if a lift in the desired direction is not already on this floor

Lift system: machine (cont.)

EVENTS

request_floor_from_lift $=$

ANY l, f

WHEN $l \in LIFT \wedge f \in FLOOR$

$l \notin moving \Rightarrow floor(l) \neq f$

THEN

$out := out \cup \{l \mapsto f\}$

END

...

Lift system: notes

- Again, the last guard is needed to prove the formulated invariant:

  $moving \lhd (out \cap floor) = \varnothing$

- Stating the necessary condition: for any non-moving (stopped) lift on some floor, a floor request from this lift is only allowed if the lift is not already on this floor

Lift system: notes

- In the remaining operations, we will often need to check whether there is a reason for a particular lift to go up or down
- In other words, whether there are the floors above or below where the users have requested to get in or out the lift
- We call this property as "*a lift is attracted up (down)*"
- We can formalise "*attracted up*" as follows

$$(dom(in) \cup out[\{l\}]) \cap ((floor(l) + 1) \mathinner{..} top) \neq \varnothing$$

Deconstruction a property

- $dom(in)$
  the floors where the users have requested a lift
- $out[\{l\}]$
  (using the relational image operator) the floors to which the lift is requested to go
- $(floor(l) + 1) \mathinner{..} top$
  the floors above
- All together,
  $(dom(in) \cup out[\{l\}]) \cap ((floor(l) + 1) \mathinner{..} top) \neq \varnothing$

  "There are the floors above where the users have requested to get in or out the lift"

Deconstruction a property (cont.)

- Similarly, "attracted down" can be formalised as follows

  $$(dom(in) \cup out[\{l\}]) \cap (ground \mathrel{..} (floor(l) - 1)) \neq \varnothing$$

  "There are the floors below where the users have requested to get in or out the lift"

- We are going to check these property or their negations as well as use them as a part of more complex definitions quite often. How we could introduce the respective shorthands?

- Two solutions:
  - install the Theory extension (extra time for learning needed);
  - define the respective higher-order functions *attracted_up* and *attracted_dn* in the model context and use them in the machine when needed.

Lift system: context (cont.)

2 extra axioms are added to the context for each definition: one for giving the type for such a higher-order function, the other one for defining its returned values

$$
\begin{array}{l}
attracted\_up \in LIFT \times (LIFT \rightarrow FLOOR) \times (FLOOR \leftrightarrow DIRECTION) \\
\qquad \times (LIFT \leftrightarrow FLOOR) \rightarrow BOOL \\
\forall l, floor, in, out \cdot \; attracted\_up(l \mapsto floor \mapsto in \mapsto out) = \\
\qquad bool(\, (dom(in) \cup out[\{l\}]) \cap ((floor(l) + 1) \,..\, top) \neq \varnothing \,) \\
\\
attracted\_dn \in LIFT \times (LIFT \rightarrow FLOOR) \times (FLOOR \leftrightarrow DIRECTION) \\
\qquad \times (LIFT \leftrightarrow FLOOR) \rightarrow BOOL \\
\forall l, floor, in, out \cdot \; attracted\_dn(l \mapsto floor \mapsto in \mapsto out) = \\
\qquad bool(\, (dom(in) \cup out[\{l\}]) \cap (ground \,..\, (floor(l) - 1)) \neq \varnothing \,)
\end{array}
$$

Lift system: notes

- The functions take as parameters the given lift as well as the functions containing info about lift floor, direction, and the requests to get in or out, and evaluate all them together them as *TRUE* or *FALSE*
- In addition to the typing axiom, the other axiom gives the exact definition of such evaluation
- The operator *bool* directly returns *TRUE* or *FALSE* for the given logical expression (predicate)
- It allows to replace two axioms:
  $\forall x, y, \dots\ cond \;\Rightarrow\; f(x \mapsto y \mapsto \dots) = TRUE$
  and
  $\forall x, y, \dots\ not\ cond \;\Rightarrow\; f(x \mapsto y \mapsto \dots) = FALSE$
- Now we can directly use the introduced functions in machine event guards

Lift system: machine (cont.)

$$
\begin{array}{l}
depart\_up \;= \\
\quad \textbf{ANY } l \\
\quad \textbf{WHEN} \\
\qquad l \notin moving \\
\qquad dir(l) = up \\
\qquad attracted\_up(l \mapsto floor \mapsto in \mapsto out) = TRUE \\
\quad \textbf{THEN} \\
\qquad moving \;:=\; moving \cup \{l\} \\
\qquad floor(l) \;:=\; floor(l) + 1 \\
\quad \textbf{END}
\end{array}
$$

...

Lift system: machine (cont.)

$$
\begin{array}{l}
depart\_dn \;=\\
\quad \textbf{ANY } l\\
\quad \textbf{WHEN}\\
\qquad l \notin moving\\
\qquad dir(l) = dn\\
\qquad attracted\_dn(l \mapsto floor \mapsto in \mapsto out) = TRUE\\
\quad \textbf{THEN}\\
\qquad moving \;:=\; moving \cup \{l\}\\
\qquad floor(l) \;:=\; floor(l) - 1\\
\quad \textbf{END}\\
\ldots
\end{array}
$$

Lift system: machine (cont.)

$change\_up\_to\_dn \;=$

ANY l

WHEN

$l \notin moving$

$dir(l) = up$

$attracted\_up(l \mapsto floor \mapsto in \mapsto out) = FALSE$

$attracted\_dn(l \mapsto floor \mapsto in \mapsto out) = TRUE$

THEN

$in \;:=\; in \setminus \{floor(l) \mapsto dn\}$

$dir(l) \;:=\; dn$

END

...

Lift system: machine (cont.)

$change\_dn\_to\_up \ =$
  ANY l
  WHEN
    $l \notin moving$
    $dir(l) = dn$
    $attracted\_dn(l \mapsto floor \mapsto in \mapsto out) = FALSE$
    $attracted\_up(l \mapsto floor \mapsto in \mapsto out) = TRUE$
  THEN
    $in \ := \ in \setminus \{floor(l) \mapsto up\}$
    $dir(l) \ := \ up$
  END

...

Lift system: notes

- Sometimes we need to check whether a lift can skip a floor while moving further to its direction or it needs to stop
- These notions (*can_continue_up* and *can_continue_down*) rely on the introduced notions *attracted_up* and *attracted_down*
- We can formalise *can_continue_up* as follows

$$l \mapsto floor(l) \notin out \;\wedge\; floor(l) \mapsto dir(l) \notin in \;\wedge\; attracted\_up(\ldots)$$

- Similarly, *can_continue_down* is defined as follows

$$l \mapsto floor(l) \notin out \;\wedge\; floor(l) \mapsto dir(l) \notin in \;\wedge\; attracted\_dn(\ldots)$$

Deconstruction a property

- $l \mapsto floor(l) \notin out$
  there no people that want to get out on this floor
- $floor(l) \mapsto dir(l) \notin in$
  there no people that want to get in on this floor to move in the lift direction
- $attracted\_up(...)$
  there are the floors above where people requested to get in or out

Lift system: context (cont.)

Introducing the corresponding higher-order definitions in the context

$$can\_continue\_up \in LIFT \times (LIFT \to FLOOR) \times (LIFT \to DIRECTION) \\ \times (FLOOR \leftrightarrow DIRECTION) \times (LIFT \leftrightarrow FLOOR) \to BOOL$$

$$\forall l, floor, dir, in, out \cdot\ can\_continue\_up(l \mapsto floor \mapsto dir \mapsto in \mapsto out) = \\ bool(\ l \mapsto floor(l) \notin out\ \wedge\ floor(l) \mapsto dir(l) \notin in \\ \wedge\ attracted\_up(l \mapsto floor \mapsto in \mapsto out) = TRUE\ )$$

$$can\_continue\_dn \in LIFT \times (LIFT \to DIRECTION) \times (LIFT \to FLOOR) \\ \times (FLOOR \leftrightarrow DIRECTION) \times (LIFT \leftrightarrow FLOOR) \to BOOL$$

$$\forall l, floor, dir, in, out \cdot\ can\_continue\_dn(l \mapsto floor \mapsto dir \mapsto in \mapsto out) = \\ bool(\ l \mapsto floor(l) \notin out\ \wedge\ floor(l) \mapsto dir(l) \notin in \\ \wedge\ attracted\_dn(l \mapsto floor \mapsto in \mapsto out) = TRUE\ )$$

END

Lift system: machine (cont.)

$$
\begin{array}{l}
\quad continue\_up \;=\\
\quad\quad \textbf{ANY } l\\
\quad\quad \textbf{WHEN}\\
\quad\quad\quad l \in moving\\
\quad\quad\quad dir(l) = up\\
\quad\quad\quad can\_continue\_up(l \mapsto floor \mapsto dir \mapsto in \mapsto out) = TRUE\\
\quad\quad \textbf{THEN}\\
\quad\quad\quad floor(l) \;:=\; floor(l) + 1\\
\quad\quad \textbf{END}\\
\ldots
\end{array}
$$

Lift system: machine (cont.)

$continue\_dn \ =$

ANY l

WHEN

$l \in moving$

$dir(l) = dn$

$can\_continue\_dn(l \mapsto floor \mapsto dir \mapsto in \mapsto out) = TRUE$

THEN

$floor(l) \ := \ floor(l) - 1$

END

...

Lift system: machine (cont.)

$stop\_up$ =
  ANY l
  WHEN
    $l \in moving$
    $dir(l) = up$
    $can\_continue\_up(l \mapsto floor \mapsto dir \mapsto in \mapsto out) = FALSE$
  THEN
    $moving := moving \setminus \{l\}$
    $out := out \setminus \{l \mapsto floor(l)\}$
    $in := in \setminus \{floor(l) \mapsto dir(l)\}$
  END

...

Lift system: machine (cont.)

$stop\_dn \ =$
ANY l
WHEN
$l \in moving$
$dir(l) = dn$
$can\_continue\_dn(l \mapsto floor \mapsto dir \mapsto in \mapsto out) = FALSE$
THEN
$moving \ := \ moving \setminus \{l\}$
$out \ := \ out \setminus \{l \mapsto floor(l)\}$
$in \ := \ in \setminus \{floor(l) \mapsto dir(l)\}$
END

...

Lift system: notes

- An example of the system execution:
  $depart\_up \rightarrow continue\_up \rightarrow stop\_up \rightarrow depart\_up \rightarrow continue\_up \rightarrow continue\_up \rightarrow stop\_up \rightarrow change\_up\_to\_dn \rightarrow depart\_dn \rightarrow continue\_dn \rightarrow \ldots$
- In between this sequence, any number of the events *request_lift_from_floor* and *request_floor_from_lift* can occur
- Overall, the developed lift models demonstrate how we can handle/distribute the system complexity, by defining necessary higher-order notions in the model context and thus simplifying modelling and verifying system dynamics (machine)