```
scheme
  LIFT =
    class
      value
        max : Nat • max > 1

      type
        Floor = {| n : Nat • n ∈ { 1 .. max } |},
        Door_Position == open | closed,
        Lift_Position′ == at(fno : Floor) | between(fno1 : Floor, fno2 : Floor),
        Lift_Position =
          {| p : Lift_Position′ • case p of between(f1, f2) → f2 = f1 + 1, _ → true end |}

      type State

      value
        /* observers */
        door_position : Floor × State → Door_Position,
        lift_position : State → Lift_Position

      value
        /* generators */
        open_door : Floor × State → State,
        move_up : State → State

      axiom
        /* observer−generator axioms */
        [ door_position_open_door ]
          ∀ f1, f2 : Floor, σ : State •
            door_position(f2, open_door(f1, σ)) ≡
              if f1 = f2 then open else door_position(f2, σ) end,

        [ lift_position_open_door ]
          ∀ f : Floor, σ : State • lift_position(open_door(f, σ)) ≡ lift_position(σ),

        [ door_position_move_up ]
          ∀ f : Floor, σ : State • door_position(f, move_up(σ)) ≡ door_position(f, σ),

        [ lift_position_move_up ]
          ∀ σ : State •
            lift_position(move_up(σ)) ≡
              case lift_position(σ) of
                at(f) → if f < max then between(f, f + 1) else at(f) end,
                between(f1, f2) → at(f2)
              end

      value
        safe : State → Bool
        safe(σ) ≡ (∀ f : Floor • door_position(f, σ) = open ⇒ lift_position(σ) = at(f)),

        safe_open_door : Floor × State → State
        safe_open_door(f, σ) ≡ if safe(open_door(f, σ)) then open_door(f, σ) else σ end
    end
```