Modelling an Aircraft Landing System in 
Event-B*

Dominique Méry
Université de Lorraine, LORIA
BP 239, Nancy, France
Dominique.Mery@loria.fr

Neeraj Kumar Singh
McMaster Centre for Software Certification
Hamilton, ON, Canada
singhn10@mcmaster.ca

July 4, 2014

Abstract
The failure of hardware or software in a critical system can lead to loss of lives. The design errors can be main source of the failures that can be introduced during system development process. Formal techniques are an alternative approach to verify the correctness of critical systems, overcoming limitations of the traditional validation techniques such as simulation and testing. The increasing complexity and failure rate brings new challenges in the area of verification and validation of avionic systems. Since the reliability of the software cannot be quantified, the correct by construction approach can implement a reliable system. Refinement plays a major role to build a large system incrementally from an abstract specification to a concrete system. This paper contributes as a stepwise formal development of the landing system of an aircraft. The formal models include the complex behaviour, temporal behaviour and sequence of operations of the landing gear system. The models are formalized in Event-B modelling language, which supports stepwise refinement. This case study is considered as a benchmark for techniques and tools dedicated to the verification of behavioural properties of systems.

Keywords
Abstract model, Event-B, Event-driven approach, Proof-based development, Refinement, Landing Gear System

1 Introduction
In the cutting edge technology of aircraft, the requirements for avionic systems become increasingly complex. The failure of hardware or software in such a complex system

*The current report is the companion paper of the paper [17] accepted for publication in the volume 433 of the serie Communications in Computer Information Science. The Event-B models are available at the link [http://eb2all.loria.fr] Processed on July 4, 2014.
can lead to loss of lives. The increasing complexity and failure rate brings new challenges in the area of verification and validation of avionic systems. The Federal Aviation Administration (FAA) ensures that aircraft meets highest safety standards. The FAA recommends the catastrophic failures of the aircraft and suggests probabilities of failure on the order of per flight hour [12].

Hardware component failures and design errors are two main reasonable factors to major the reliability of the avionics. There are several techniques like redundancy and voting are used to handle the hardware failures. However, the design errors can be introduced at the development phase, which may include errors in the system specification, and errors made during the implementation of the software or hardware [14].

The complexity of software has been tremendously increased. Our experience, intuition and developed methodologies is reliable for building the continuous system, but software exhibits discontinuous behaviour. To verify the correctness of the system, it is highly desirable to reason about millions of sequences of discrete state transitions. Traditional techniques like testing and simulations are infeasible to test the correctness of a system [7]. Since the reliability of the software cannot be quantified, the avionic software must be developed using correct by construction [15] approach that can produce the correct design and implementation of the final system [19].

This paper describes how rigorous analysis employing formal methods can be applied to the software development process. Formal methods is considered as an alternative approach for certification in the DO-178B standard for avionics software development. We propose the refinement based correct by construction approach to develop a critical system. The nature of the refinement that we verify using the RODIN [18] proof tools is a safety refinement. Thus, the behaviour of final resulting system is preserved by an abstract model as well as in the correctly refined models. Proof-based development methods [1] integrate formal proof techniques in the development of software systems. The main idea is to start with a very abstract model of the system under development. Details are gradually added to this first model by building a sequence of more concrete events. The relationship between two successive models in this sequence is refinement [1, 3]. Here we present stepwise development to model and verify such interdisciplinary requirements in Event-B [8, 1] modelling language. The correctness of each step is proved in order to achieve a reliable system.

In this paper, we present the stepwise formalization of the benchmark case study landing system of an aircraft. The current work intends to explore those problems related to modelling the sequence of operations of landing gears and doors associated with hydraulic cylinders under the real-time constraints and to evaluate the refinement process.

The outline of the remaining paper is as follows. Section 2 presents selection of the case study related to the landing system of an aircraft for formalization. In Section 3, we explore the incremental proof-based formal development of the landing system. Finally, in Section 4, we conclude the paper.

2 Basic Overview of Landing Gear System

The landing gear is an essential system that allows an aircraft to land safely, and supports the entire weight of an aircraft during landing and ground operations. The basic engineering and operational behaviors behind a landing gear system are very complex. There are several types of gears, which depend on the aircraft design and its intended use. Most landing gears have wheels to facilitate operation to and form hard surfaces,
Three basic arrangements of landing gear are used: tail wheel type landing gear, tandem landing gear, and tricycle-type landing gear. The most commonly used landing gear arrangement is the tricycle-type landing gear. All these aircraft landing gears are further classified into fixed and retractable categories. Single engine and light weight aircrafts use fixed landing gear while the retractable gear is used in heavy aircrafts.

The landing system controls the maneuvering landing gears and associated doors. Fig. 1 depicts basic components of a landing system. The landing system is made of three different landing sets, which corresponds to front, left and right. The main components of a landing system are doors, landing gears and hydraulic cylinders.

The landing gear system is controlled by the software in nominal operating mode, and an emergency mode is handled analogically. Generally, landing system always use nominal mode. In case of system failure, the pilot can activate the emergency mode. However, the landing system can be activated in emergency mode only when any anomaly is detected in the system.

There are sequential operations of the landing gear system. The sequential operations for extending gears are: open the doors of gear boxes, extend the landing gears, and close the doors. Similarly the sequential operations for retraction gears are: open the door, retract the landing gears, and close the doors. During these sequential operations there are several parameters and conditions, which can be used to assess the health of a landing system.

There are three main components of the landing gear system: 1) mechanical system, 2) digital system, and 3) pilot interface. The mechanical system is composed of three landing sets, where each set contains landing gear box, and a door with latching boxes. The landing gears and doors motions are performed with the help of cylinders. The cylinder position is used to identify the various states of the door or landing gear positions. Hydraulic power is used to control the cylinders with the help of electro-valves. These electro-valves are activated by a digital system. The digital system is composed of two identical computing modules, which execute parallel. The digital system is only the responsible for controlling mechanical parts like gears and doors, and for detecting anomalies. The pilot interface has an Up/Down handle and a set of indicators. The handle is used by pilot for extending or retracting the landing gear.
sequence, and a set of indicators is the different type of lights for giving the actual position of gears and doors, and the system state. A detailed description of the landing gear system is given in [11, 6].

The landing gear system is a critical embedded system, where all the operations are based on the state of a physical device, and required temporal behaviour. The main challenge is to model the system behaviour of the landing gear system, and to prove the safety requirements under the consideration of physical behaviour of hydraulic devices.

3    Formal Development of the Landing System

The development is progressively designing the landing system by integrating observations and elements of the document. The first model is specific as abstract as possible and it captures the different possible big steps of the system by defining the synchronous atomic events. For example, the sequence of door opening, door closing, gear extension and gear retraction etc.

3.1 M1: Moving Up and Down

When the system is moving up (resp. down) till retraction (resp. extension), it will be in a position halt and up (resp. down), namely haltup (resp. haltdown). The first model observes the positions of the global state which considers that the landing system is either moving down from a haltup position, or moving up from a haltdown position. The global state expresses the state of handle at an initialization in a down state (button := DOWN) and the gear system is halted in a haltdown position (phase := haltdown). It means that initially the gear system is extended and locked. Two state variables record these informations namely button and phase. Events model the possible observable actions and modifications over the global system:

- **PressDOWN** is enabled, when the gear system is halted up and retracted; the system is in a new state corresponding to the movingup action. The intention is to extend the gear system.
- **PressUP** is enabled, when the gear system is halted down and extended; the system is in a new state corresponding to the movingdown action. The intention if to retract the gear system.

Moreover, when one of events PressDOWN or PressUP (solid labelled transitions in Fig. 2) is observed, the system should provide a service corresponding to an effective action (dashed labelled transitions in Fig. 2) of the landing system and physically moving gears. The landing system reacts (dashed labelled transitions in Fig. 2) to the orders of the pilot (solid labelled transitions in Fig. 2).

- **movingup** is an action supported by engine which helps to move the landing system into the state haltup and to the retracted state.
- **movingdown** is an action supported by engine which helps to move the landing system into the state haltdown and to the extended state.

Events express that, when the button remains UP (resp. DOWN), the reaction of the system is to reach the state retracted (resp. extended). The current diagram assumes that the system is operating in normal mode. The detection of anomalies is left for the
next refinements. The diagram contains the main goals of the system which is operating in a cyclic mode. The requirements R11bis and R12bis are clearly satisfied, as well as R12 and R11. Other requirements are not considered since they are related to features that are not yet defined.

3.2 M2: Opening and Closing Doors

The model M2 is considering different possible steps in the moving up or in the moving down phases. However, the different steps are possibly victims of counters orders. The pilot decides to press UP and then to press DOWN or reciprocally. These movements affect the classical cycle of the system starting from a locked closed position to another one without interrupt. First observation leads to consider that we identify that doors are alternatively opening and closing. We add a detail on the fact that the doors are opened when they are unlocked and when they are closed, they are locked. A new state is enriching the previous one by a state variable for doors states \( dstate \) and a variable for expressing when doors are locked \( lstate \). Three variables are used to control the possible change of decisions and expressing the sequentialisation of extension scenario or retraction scenario: \( p, l, i \).

The next invariant states that when the doors are opened, the doors are unlocked \( (M_{2,\text{inv}5}) \); when one door is opened, all the doors are opened \( (M_{2,\text{inv}3}) \) and when a door is closed, all the doors are closed \( (M_{2,\text{inv}4}) \).

\[
\begin{align*}
M_{2,\text{inv}1} & : dstate \in \text{DOORS} \rightarrow \text{SDOORS} \\
M_{2,\text{inv}2} & : lstate \in \text{DOORS} \rightarrow \text{LOCKS} \\
M_{2,\text{inv}3} & : \text{dstate}^{-1}[(\text{OPEN})] \neq \emptyset \Rightarrow \text{dstate}^{-1}[(\text{OPEN})] = \text{DOORS} \\
M_{2,\text{inv}4} & : \text{dstate}^{-1}[(\text{CLOSED})] \neq \emptyset \Rightarrow \text{dstate}^{-1}[(\text{CLOSED})] = \text{DOORS} \\
M_{2,\text{inv}5} & : \text{dstate}[\text{DOORS}] = \{\text{OPEN}\} \Rightarrow \text{Istate}[\text{DOORS}] = \{\text{UNLOCKED}\} \\
M_{2,\text{inv}6} & : l = E \land p = R \Rightarrow \text{Istate}[\text{DOORS}] = \{\text{UNLOCKED}\} \\
M_{2,\text{inv}7} & : l = R \land p = E \Rightarrow \text{Istate}[\text{DOORS}] = \{\text{UNLOCKED}\}
\end{align*}
\]

Events are now capturing the observation of opening and closing with possible counter orders by the pilot. We have not yet considered the state of flying or grounding. Initially, doors are closed and the state is haltdown. It means that the landing system is corresponding to a state on ground and should be obviously extended. The three auxiliary variables \( (p, l, i) \) are set to \( R \) to mean that the system is ready to retract whenever the pilot wants. We do not consider the case when a crazy pilot would try to retract when the aircraft is on the ground but we may consider that we observe a safe situation. Further refinements will forbid these kinds of possible behaviours. Events are refining the previous four events and we refine the two events \( \text{PressDown} \) and \( \text{PressUp} \) by events that can interrupt the initial scenario and switch to the other scenario. Fig. 3 describes the state-based automaton for the model M2 and we use the following notations \( UP \) for \( \text{button} = \text{UP} \), \( DN \) for \( \text{button} = \text{DOWN} \), \( C \) for \( \text{dstate}[\text{DOORS}] = \text{OPEN} \),
Figure 3: State-based Automaton for Events in model M2

\{\text{CLOSED}\}, \text{O} \text{ for } \text{lstate}[\text{DOORS}] = \{\text{OPEN}\}, \text{L} \text{ for } \text{lstate}[\text{DOORS}] = \{\text{LOCKED}\}, \text{U} \text{ for } \text{lstate}[\text{DOORS}] = \{\text{UNLOCKED}\}, \text{m'}\text{down} \text{ for } \text{phase} = \text{movingdown}”, \text{m'}\text{up} \text{ for } \text{phase} = \text{movingup}, \text{h'}\text{down} \text{ for } \text{phase} = \text{haltdown}, \text{h'}\text{up} \text{ for } \text{phase} = \text{haltup}. \text{ The dashed and plain arrows present the distinction between two different types of actions. Dashed arrows show that it is an action of the system, and plain arrows show that it is an action of the pilot.}

The diagram Fig. 3 confirms the requirements. The model is validated using ProB and the sequences of retraction and extension are observed according to the requirements.

3.3 M3: Observing the gears

The next observation leads us to consider the full mechanical system. In fact, doors are opened and closed but we have the time to see that gears are either moving out (extension scenario) or moving in (retraction scenario). The next model is refining the previous one by adding gears and observing different states of the gears (\text{gstate} \in \text{GEARS} \rightarrow \text{SGEARS}). SGEARS is defined as enumerated set \text{partition}(\text{SGEARS}, \{\text{RETRACTED}\}, \{\text{EXTENDED}\}, \{\text{RETRACTING}\}, \{\text{EXTENDING}\}) to capture the multiple states of gears. There are obvious invariant properties that express that the doors are opened when the gears are moving. The invariants are listed as follow:

\begin{align*}
M_3,\text{inv1} & : \text{gstate} \in \text{GEARS} \rightarrow \text{SGEARS} \\
M_3,\text{inv2} & \forall \text{door} : (\text{door} \in \text{DOORS} \land \text{dstate}(\text{door}) = \text{CLOSED} \land \text{ran}(\text{gstate}) \neq \{\text{RETRACTED}\}) \\
M_3,\text{inv3} & \forall \text{door} : (\text{door} \in \text{DOORS} \land \text{dstate}(\text{door}) = \text{CLOSED} \land \text{ran}(\text{gstate}) \neq \{\text{EXTENDED}\}) \\
M_3,\text{inv4} & \forall \text{door} : (\text{door} \in \text{DOORS} \land \text{dstate}(\text{door}) = \text{CLOSED} \land \text{ran}(\text{gstate}) \neq \{\text{RETRACTED}\})
\end{align*}
\[ M_{3\_inv4} : \begin{cases} \text{ran}(\text{gstate}) \neq \{\text{RETRACTED}\} \land \text{ran}(\text{gstate}) \neq \{\text{EXTENDED}\} \\ \text{ran}(\text{dstate}) = \{\text{OPEN}\} \\ \text{ran}(\text{dstate}) = \{\text{CLOSED}\} \end{cases} \]

\[ M_{3\_inv5} : \begin{cases} \text{ran}(\text{dstate}) = \{\text{OPEN}\} \Rightarrow \text{ran}(\text{gstate}) \cap \{\text{RETRACTING, EXTENDING}\} = \emptyset \end{cases} \]

\[ M_{3\_inv2} \] and \[ M_{3\_inv3} \] express that when doors are opened, either the gears are extended or the gears are retracted. When the doors are closed, the gears are not in moving state (\[ M_{3\_inv4} \] and \[ M_{3\_inv5} \]). When the gears are moving, the doors are opened. The expression of the simultaneous state of the doors either closed or opened, as well as the gears either extended or retracted, prepare the conditions of the synchronisation over the doors and the gears. Fig. [ ] is now detailed by splitting the two states (\( DN, m'\downarrow \), \( O, U \)) and (\( UP, m'\uparrow \), \( O, U \)) and by considering that the new variable \( \text{gstate} \) is modified at this stage. We are introducing four new events corresponding to the retraction of gears and to the extension of gears.

The retraction phase is decomposed into two events \( \text{retracting\_gears} \) and \( \text{retraction} \) and the gears are transiting from a state \( \text{EXTENDED} \) into the state \( \text{RETRACTING} \) and finally the state \( \text{RETRACTED} \).

The extension phase is decomposed into two events \( \text{extending\_gears} \) and \( \text{extension} \) and the gears are transiting from a state \( \text{RETRACTED} \) into the state \( \text{EXTENDING} \) and finally the state \( \text{EXTENDED} \).

The events \( PU4 \) and \( PD4 \) are both refined into three events which are controlling or switching from the retraction to the extension and vice-versa. The two possible scenarios (extension and retraction) have a meaning and we can address the requirements R21 and R22.

The model M3 is refined into a new model called M30 which is forbidden the use of buttons. The model is clearly satisfying the requirement over the successive actions. ProB is also used to validate the behaviour of system.
Figure 4: State-based Automaton for Events in model M3
3.4 M4: Sensors and Actuators

In this refinement, we address the problem of sensors and actuators. We introduce the management of sensors and actuators by considering the collection of values of sensors and an abstract expression of computing modules for analysing the sensed values. We introduce a list of new variables according to the Fig. 5:

- Variables for expressing the sensors states: handle (for the pilot interface), analogical switch, gear extended (sensor for detecting an extension activity), gear_retracted (sensor for detecting a retraction activity), gear_shock_absorber (sensor for detecting the flight or ground mode), door_closed (sensor for stating when the doors are closed), door_open (sensor for stating when the doors are opened), circuit_pressurized (sensor for the pressure control).

- Variables for getting the evaluation of the sensed states of the system by the computing modules: general_EV, close_EV, retract_EV, extend_EV, open_EV.

- Variables for modelling the computation of the sensed state from the collected sensed values: general_EV_func, close_EV_func, retract_EV_func, extend_EV_func, open_EV_func.

- Variables for collecting output of computing modules: gears_locked_down, gears_man, anomaly.

New variables are used for adding some constraints over guards of previous events:

- HPD1 and HPU1 are two events corresponding to the order by the pilot interface to extend (HPD1) or to retract (HPU1) the gears. For instance, the guard $\forall x :: x \in 1 \ldots 3 \rightarrow handle(x) = UP$ senses that the handle is UP and then it moves to
DOWN (handle :∈ 1 .. 3 → {DOWN}). The sensors are triplicated and we define each sensor value by a function from 1..3 into the sensors values.

- Analogical_switch_closed and Analogical_switch_open are two events for updating the general switch for protecting the system against abnormal behaviour of the digital part.
- Circuit_pressurized manages the sensor of the pressure control.
- Computing_Module_1..2 models in a very abstract way for computing and updating of EV variables using sensors values.
- Failure_Detection detects any failure in the system.

The model introduces sensors and values synthesized from sensors values. We have used a very abstract way to state the values of sensors. The model M4 is not analyse-able with ProB. The previous requirements R11, R11bis, R12, R12bis, R22, R21 are remaining satisfied by the model M4 by refinement. We need to strengthening the guards of events (∀x : x ∈ 1 .. 3 ⇒ handle(x) = button). The reader will notice that the two events HPU1 and HPD1 are the external interfaces for controlling the events to associate the functionality of handle with old variable button. The guard gear_shock_absorber = {a ↦ b | a ∈ 1 .. 3 ∧ b = ground} indicates that now we know that either we are on the ground or not: it means that we assume that sensors are trusted and this assumption is valid. The state of gear_shock_absorber is modified according to the figure 11, page 12 of [6] and it is the reason for updating in two events extension and retraction.

In this refinement, the number of proof obligations is very high (247) but it is possible to add intermediate models for progressive development.

- The two events HPU1 and HPD1 are adding a small amount of new proof obligations.
- The unproved proof obligations appearing in the summary are mainly typing properties and they are discharged either using the SMT solver or a procedure. We consider that they are pseudo-automatic proof obligations.

### 3.5 M5: Managing electro-valves

The model M5 takes into account the management of electro-valves used for moving the gears from a position to another one. Four new variables are modelling pressure states (page 10, subsection 3.2, Electro-valves) and they model the hydraulic features of the system: general_EV_Hout, close_EV_Hout, retract_EV_Hout, extend_EV_Hout, open_EV_Hout, A_Switch_Out. The invariant is stating that either the pressure is on or off by the two possible values: 0 or Hin:

| inv1 : general_EV_Hout ∈ {0, Hin} |
| inv2 : close_EV_Hout ∈ {0, Hin}   |
| inv3 : retract_EV_Hout ∈ {0, Hin} |
| inv4 : extend_EV_Hout ∈ {0, Hin}  |
| inv5 : open_EV_Hout ∈ {0, Hin}    |

The summary of new proof obligations is simply that 19 new proof obligations are generated and automatically discharged. In the previous development, the values were
less precise and we got a problem in the next refinements with some proof obligations to discharge. A new event Update\_Hout is introduced to update the values of sensors for the hydraulic part:

```
EVENT Update\_Hout
BEGIN
  act1 : general\_EV\_Hout : | (general\_EV = TRUE ∧ general\_EV\_Hout' = Hin)
                        ∨ (general\_EV = FALSE ∧ general\_EV\_Hout' = 0)
                        ∨ (A\_Switch\_Out = TRUE ∧ general\_EV\_Hout' = Hin)
                        ∨ (A\_Switch\_Out = FALSE ∧ general\_EV\_Hout' = 0)
  act2 : close\_EV\_Hout : | (close\_EV = TRUE ∧ close\_EV\_Hout' = Hin)
                        ∨ (close\_EV = FALSE ∧ close\_EV\_Hout' = 0)
  act3 : open\_EV\_Hout : | (open\_EV = TRUE ∧ open\_EV\_Hout' = Hin)
                        ∨ (open\_EV = FALSE ∧ open\_EV\_Hout' = 0)
  act4 : extend\_EV\_Hout : | (extend\_EV = TRUE ∧ extend\_EV\_Hout' = Hin)
                        ∨ (extend\_EV = FALSE ∧ extend\_EV\_Hout' = 0)
  act5 : retract\_EV\_Hout : | (retract\_EV = TRUE ∧ retract\_EV\_Hout' = Hin)
                        ∨ (retract\_EV = FALSE ∧ retract\_EV\_Hout' = 0)
END
```

The event Circuit\_pressurized is refined by two events considering that the sensing is OK or not; it assigns the value of $Hout$.

```
EVENT Circuit\_pressurized
REFINES Circuit\_pressurized
WHEN
  grd1 : general\_EV\_Hout = Hin
THEN
  act9 : circuit\_pressurized : ∈ 1 .. 3 → {TRUE}
END

EVENT Circuit\_pressurized\_notOK
REFINES Circuit\_pressurized
WHEN
  grd1 : general\_EV\_Hout = 0
THEN
  act9 : circuit\_pressurized : ∈ 1 .. 3 → {FALSE}
END
```

### 3.6 M6: Integrating Cylinders Behaviours

The next step is to integrate the cylinders behaviour according to the electro-valves circuit and to control the process, which is computing from sensors values the global state of the system. It leads to strengthen guards of events opening and closing doors and gears using cylinders sensors and hydraulic pressure ($opening\_doors\_DOWN, opening\_doors\_UP, closing\_doors\_UP, closing\_doors\_DOWN, unlocking\_UP, unlocking\_DOWN, locking\_DOWN, retracting\_gears, retraction, extending\_gears, extension$). The event CylinderMovingOrStop models the change of the cylinders according to the pressure, when the value of state is cylinder. It leads to a next state which activates the computing modules.
EVENT CylinderMovingOrStop
Cylinder Moving or Stop according to the output of hydraulic circuit

WHEN

WHEN grd1 : state = cylinder
THEN

act1 : SGCylinder : |

SGCylinder′ = {a ↦→ b | a ∈ GEARs × {GCY F, GCY R, GCY L} ∧ b = MOVING}
∧ extend_EV_Hout = Hin
∨ SGCylinder′ = {a ↦→ b | a ∈ GEARs × {GCY F, GCY R, GCY L} ∧ b = STOP}
∧ extend_EV_Hout = 0
∨ SGCylinder′ = {a ↦→ b | a ∈ GEARs × {GCY F, GCY R, GCY L} ∧ b = MOVING}
∧ retract_EV_Hout = Hin
∨ SGCylinder′ = {a ↦→ b | a ∈ GEARs × {GCY F, GCY R, GCY L} ∧ b = STOP}
∧ retract_EV_Hout = 0

act2 : SDCylinder : |

SDCylinder′ = {a ↦→ b | a ∈ DOORS × {DCY F, DCY R, DCY L} ∧ b = MOVING}
∧ open_EV_Hout = Hin
∨ SDCylinder′ = {a ↦→ b | a ∈ DOORS × {DCY F, DCY R, DCY L} ∧ b = STOP}
∧ open_EV_Hout = 0
∨ SDCylinder′ = {a ↦→ b | a ∈ DOORS × {DCY F, DCY R, DCY L} ∧ b = MOVING}
∧ close_EV_Hout = Hin
∨ SDCylinder′ = {a ↦→ b | a ∈ DOORS × {DCY F, DCY R, DCY L} ∧ b = STOP}
∧ close_EV_Hout = 0

act3 : state := computing

END

More than 50 % of the proof obligations are manually discharged. However, it appears that the disjunction of actions allows us to have a unique view of the cylinders behaviours. The proofs to discharge are not complex and are mainly discharged by several clicks on procedures buttons.

3.7 M7: Failure Detection

The model M7 is modelling the detection of different possible failures. Page 16 and page 17 of the case study have given a list of conditions for detecting anomalies: Analogical switch monitoring, Pressure sensor monitoring, Doors motion monitoring, Gears motion monitoring, Expected behavior in case of anomaly. The decision is to refine the event Failure_Detection into six events which are modelling the different cases for failure detection: Failure_Detection_Generic_Monitoring, Failure_Detection_Analogical_Switch, Failure_Detection_Pressure_Sensor, Failure_Detection_Doors, Failure_Detection_Gears, Failure_Detection_Generic_Monitoring. The decision is to postpone the introduction of timing constraints in the last model. However, we have to strengthen the guards of events opening_doors_DOWN, opening_doors_UP, closing_doors_UP, closing_doors_DOWN, unlocking_UP, locking_UP, unlocking_DOWN, locking_DOWN by adding a condition anomaly = FALSE.

3.8 M8: Timing Requirements

The time pattern [9] provides a way to add timing properties. The pattern adds an event tic_tock simulating the progression of time. Timing properties are derived from the document. We agree with possible discussions on the modelling of time but it appears that further works are required to get a better integration of a more real time approach. However, we think that the current model M8 is an abstraction of another automaton with real time features [2].
The pilot uses the handle and the handle is taking some time to change the value of the sensors.

The proof assistant is not efficient on this new refinement. However, now we can cover requirements with timing aspects.

### 3.9 M9: Adding Lights

The last refinement of our development introduces the interface of the pilot: the lights. These lights are modelled by a variable as $pilot\_interface\_light \in colorSet \rightarrow lightState$. Initially, $pilot\_interface\_light$ is set to $\{Green \mapsto Off, Orange \mapsto Off, Red \mapsto Off\}$. The following events are informing the pilot by interpreting the results of the computing modules and they are extracted from the document:

- $pilot\_interface\_Green\_light\_On$: green light is on; when gears locked down is true.
- $pilot\_interface\_Orange\_light\_On$: orange light is on, when gears maneuvering is true.
- $pilot\_interface\_Red\_light\_On$: red light is on, when anomaly is detected (true).
- $pilot\_interface\_Green\_light\_Off$: green light is off, when gears locked down is false.
- $pilot\_interface\_Orange\_light\_Off$: orange light is off, when gears maneuvering is false.
- $pilot\_interface\_Red\_light\_Off$: red light is off, when anomaly is detected (false).
| Model | Requirements            | Total PO | Auto | Man |
|-------|-------------------------|----------|------|-----|
| M1    | R11, R11bis, R12, R12bis | 10       | 10   | 0   |
| M2    | R11, R11bis, R12, R12bis | 33       | 33   | 0   |
| M3    | R11, R11bis, R12, R12bis, R22, R21 | 44 | 44   | 0   |
| M4    | R11, R11bis, R12, R12bis, R22, R21 | 264 | 252  | 12  |
| M5    | R11, R11bis, R12, R12bis, R22, R21 | 19 | 19   | 0   |
| M6    | R11, R11bis, R12, R12bis, R22, R21 | 49 | 20   | 29  |
| M7    | R11, R11bis, R12, R12bis, R22, R21 | 1 | 0    | 1   |
| M8    | R11, R11bis, R12, R12bis, R22, R21 | 56 | 23   | 33  |
| M9    | R11, R11bis, R12, R12bis, R22, R21 | 9 | 3    | 6   |
| Total | R11, R11bis, R12, R12bis, R22, R21 | 529 | 448  | 81  |

Figure 6: Table of requirements satisfied by models and proof statistics

4 Conclusion

Validation and verification are processed by using the ProB tool [16] and Proof Statistics. Validation refers to gaining confidence that the developed formal models are consistent with the requirements, which are expressed in the requirements document [6]. The landing system specification is developed and formally proved by the Event-B tool support prover. The developed formal models are also validated by the ProB tool through animation and model checker tool support of the abstract and successive refined models under some constraints of the tool. These constraints are the selection of parameters for testing the given model, and avoiding the failure of the tool during animation or model checking. However, we use this tool on abstract and all the refined models to check that the developed specification is deadlock free from an initial model to the concrete model. Due to features of ProB, we have used ProB for the models M1, M2 and M3.

The Table-Fig6 is expressing the proof statistics of the development in the RODIN tool. These statistics measure the size of the model, the proof obligations are generated and discharged by the Rodin platform, and those are interactively proved. The complete development of landing system results in 529 (100%) proof obligations, in which 448 (84.68%) are proved completely automatically by the RODIN tool. The remaining 81 (15.31%) proof obligations are proved interactively using RODIN tool. In the models, many proof obligations are generated due to introduction of new functional and temporal behaviors. In order to guarantee the correctness of these functional and temporal behaviors, we have established various invariants in stepwise refinement. Most of the proofs are automatically discharged and the interactively discharged proof obligations are discharged by simple sequence of using automatic procedures of Rodin.

The current version of the development is the nth version. The document describes a concrete system with sensors, mechanical parts and digital part. A first attempt by one of the authors was to propose a sequence of refined models too much close of this description. Then we try to have a global view of the system and to provide a very abstract initial model. In a second round of derivation of models, we got a wrong model, since we did not take into account the counter orders. Finally, the diagram of the Fig. summarizes main steps of the system. From this model, we decide to make elements more concrete and we introduce sensors, computing modules. Timing requirements are added in the pre-last model M8 which is then equipped by lights in
the model M9. Our models are still too abstract and we have to get comments and feedbacks from the domain experts.

References

[1] J.-R. Abrial. *Modeling in Event-B: System and Software Engineering*. 2010.

[2] Rajeev Alur and David L. Dill. A theory of timed automata. *Theor. Comput. Sci.*, 126(2):183–235, 1994.

[3] R.J.R. Back. On correct refinement of programs. *Journal of Computer and System Sciences*, 23(1):49 – 68, 1981.

[4] Dines Bjørner. *Software Engineering: Vol 1 Abstraction and Modelling - Vol 2 Specification of Systems and Languages - Vol 3 Domains, Requirements, and Software Design*. Texts in Theoretical Computer Science. An EATCS Series. Springer, 2006. ISBN: 978-3-540-21149-5.

[5] Dines Bjørner and Martin C. Henson, editors. *Logics of Specification Languages*. EATCS Textbook in Computer Science. Springer, 2007.

[6] Frédéric Boniol and Virginie Wiels. Landing Gear System. http://www.irit.fr, 2013.

[7] R.W. Butler and George B. Finelli. The infeasibility of quantifying the reliability of life-critical real-time software. *Software Engineering, IEEE Transactions on*, 19(1):3–12, 1993.

[8] Dominique Cansell and Dominique Méry. *The event-B Modelling Method: Concepts and Case Studies*, pages 33–140. Springer, 2007. See [5].

[9] Dominique Cansell, Dominique Méry, and Joris Rehm. Time constraint patterns for event b development. In Jacques Julliand and Olga Kouchnarenko, editors, *B 2007: Formal Specification and Development in B*, volume 4355 of *Lecture Notes in Computer Science*, pages 140–154. Springer Berlin Heidelberg, 2006.

[10] ClearSy. Atelier B. http://www.clearsy.com.

[11] Federal Aviation Administration (FAA). *Aircraft Landing Gear System, Chapter 13 in Aviation Maintenance Technician Handbook - Airframe Vol-1*. U.S. Department of Transportation, Washington, D.C., 2012.

[12] Federal Aviation Administration (FAA). System Design and Analysis, Advisory Circular AC 25.1309-1A. http://www.faa.gov, June 1988.

[13] E. Gamma, R. Helm, R. Johnson, R. Vlissides, and P. Gamma. *Design Patterns : Elements of Reusable Object-Oriented Software design Patterns*. Addison-Wesley Professional Computing, 1994.

[14] Sally C. Johnson and Ricky W. Butler. *Formal Methods, Chapter 21 in The Avionics Handbook*. CRC Press, 2001. edited by Cary R. Spitzer.
[15] Gary T. Leavens, Jean-Raymond Abrial, Don Batory, Michael Butler, Alessandro Coglio, Kathi Fisler, Eric Hehner, Cliff Jones, Dale Miller, Simon Peyton-Jones, Murali Sitaraman, Douglas R. Smith, and Aaron Stump. Roadmap for enhanced languages and methods to aid verification. In *Fifth Intl. Conf. Generative Programming and Component Engineering (GPCE 2006)*, pages 221–235. ACM, October 2006.

[16] Michael Leuschel and Michael Butler. *ProB: A Model Checker for B*, pages 855–874. Lecture Notes in Computer Science. Springer, 2003.

[17] Dominique Méry and Neeraj Kumar Singh. Modelling an aircraft landing system in event-b. In *ABZ Case Study*, volume 433 of *Communications in Computer Information Science*. Springer, 2014.

[18] RODIN. Rigorous open development environment for complex systems. http://rodin-b-sharp.sourceforge.net, 2004. 2004-2013.

[19] Neeraj Kumar Singh. *Using Event-B for Critical Device Software Systems*. Springer-Verlag GmbH, 2013.
## A Requirements

| Requirement | Description |
|-------------|-------------|
| R11:        | When the command line is working (normal mode), if the landing gear command button has been pushed DOWN and stays DOWN, then the gears will be locked down and the doors will be seen closed less than 15 seconds after the button has been pushed. |
| R12:        | When the command line is working (normal mode), if the landing gear command button has been pushed UP and stays UP, then the gears will be locked retracted and the doors will be seen closed less than 15 seconds after the button has been pushed. Note that a weaker version of these two requirements could be considered as well. This weaker version does not take into account quantitative time. |
| (R11bis):   | When the command line is working (normal mode), if the landing gear command button has been pushed DOWN and stays DOWN, then eventually the gears will be locked down and the doors will be seen closed. |
| (R12bis):   | When the command line is working (normal mode), if the landing gear command button has been pushed UP and stays UP, then eventually the gears will be locked retracted and the doors will be seen closed. |
| (R21):      | When the command line is working (normal mode), if the landing gear command button remains in the DOWN position, then retraction sequence is not observed. |
| (R22):      | When the command line is working (normal mode), if the landing gear command button remains in the UP position, then outgoing sequence is not observed. |
| (R31):      | When the command line is working (normal mode), the stimulation of the gears outgoing or the retraction electro-valves can only happen when the three doors are locked open. |
| (R32):      | When the command line is working (normal mode), the stimulation of the doors opening or closure electro-valves can only happen when the three gears are locked down or up. |
| (R41):      | When the command line is working (normal mode), opening and closure doors electro-valves are not stimulated simultaneously; outgoing and retraction gears electro-valves are not stimulated simultaneously. |
| (R42):      | When the command line is working (normal mode), opening doors electro-valve and closure doors electro-valve are not stimulated simultaneously outgoing gears electro-valve and retraction gears electro-valve are not stimulated simultaneously. |
| (R51):      | When the command line is working (normal mode), it is not possible to stimulate the maneuvering electro-valve (opening, closure, outgoing or retraction) without stimulating the general electro-valve. |
| (R61):      | If one of the three doors is still seen locked in the closed position more than 0.5 second after stimulating the opening electro-valve, then the boolean output normal mode is set to false. |
| (R62):      | If one of the three doors is still seen locked in the open position more than 0.5 second after stimulating the closure electro-valve, then the boolean output normal mode is set to false. |
If one of the three gears is still seen locked in the down position more than 0.5 second after stimulating the retraction electro-valve, then the boolean output normal mode is set to false.

If one of the three gears is still seen locked in the up position more than 0.5 second after stimulating the outgoing electro-valve, then the boolean output normal mode is set to false.

If one of the three doors is not seen locked in the open position more than 2 seconds after stimulating the opening electro-valve, then the boolean output normal mode is set to false.

If one of the three doors is not seen locked in the closed position more than 2 seconds after stimulating the closure electro-valve, then the boolean output normal mode is set to false.

If one of the three gears is not seen locked in the up position more than 10 seconds after stimulating the retraction electro-valve, then the boolean output normal mode is set to false.

If one of the three gears is not seen locked in the down position more than 10 seconds after stimulating the outgoing electro-valve, then the boolean output normal mode is set to false.

When at least one computing module is working, if the landing gear command button has been DOWN for 15 seconds, and if the gears are not locked down after 15 seconds, then the red light landing gear system failure is on.

When at least one computing module is working, if the landing gear command button has been UP for 15 seconds, and if the gears are not locked retracted after 15 seconds, then the red light landing gear system failure is on.

### B Introduction of the Modeling Framework

We summarize the concepts of the EVENT B modeling language developed by Abrial [1] and indicate the links with the tool called RODIN [18]. The modeling process deals with various languages, as seen by considering the triptych of Bjoerner [4]: $D, S \rightarrow R$. Here, the domain $D$ deals with properties, axioms, sets, constants, functions, relations, and theories. The system model $S$ expresses a model or a refinement-based chain of models of the system. Finally, $R$ expresses requirements for the system to be designed. Considering the EVENT B modeling language, we notice that the language can express safety properties, which are either invariants or theorems in a machine corresponding to the system. Recall that two main structures are available in EVENT B:

- Contexts express static informations about the model.
- Machines express dynamic informations about the model, invariants, safety properties, and events.

A EVENT B model is defined either as a context or as a machine. The triptych of Bjoerner [4, 5] $D, S \rightarrow R$ is translated as follows: $C, M \rightarrow R$, where $C$ is

---

1 The term ‘triptych’ covers the three phases of software development: domain description, requirements prescription and software design.
a context, \( M \) is a machine and \( R \) are the requirements. The relation \( \rightarrow \) is defined to be a logical satisfaction relation with respect to an underlying logico-mathematical theory. The satisfaction relation is supported by the RODIN platform. A machine is organizing events modifying state variables and it uses static informations defined in a context. These basic structure mechanisms are extended by the refinement mechanism which provides a mechanism for relating an abstract model and a concrete model by adding new events or by adding new variables. This mechanism allows us to develop gradually EVENTB models and to validate each decision step using the proof tool. The refinement relationship should be expressed as follows: a model \( M \) is refined by a model \( P \), when \( P \) is simulating \( M \). The final concrete model is close to the behavior of real system that is executing events using real source code. We give details now on the definition of events, refinement and guidelines for developing complex system models.

B.1 Modeling Actions Over States

EVENTB \cite{1} is based on the B notation. It extends the methodological scope of basic concepts to take into account the idea of formal reactive models. Briefly, a formal reactive model is characterized by a (finite) list \( x \) of state variables possibly modified by a (finite) list of events, where an invariant \( I(x) \) states properties that must always be satisfied by the variables \( x \) and maintained by the activation of the events. In the following, we summarize the definitions and principles of formal models and explain how they can be managed by tools \cite{18}.

Generalized substitutions are borrowed from the B notation, which express changes in the value of state variables. An event has three main parts, namely a list of local parameters, a guard and a relation over values denotes pre values of variables and post values of variables. The most common event representation is (ANY \( t \) WHERE \( G(t, x) \) THEN \( x : (R(x, x', t)) \) END). The before-after predicate \( BA(e)(x, x') \), associated with each event, describes the event as a logical predicate for expressing the relationship linking values of the state variables just before \( (x) \) and just after \( (x') \) the execution of event \( e \). The form is semantically equivalent to \( \exists t \cdot (G(t, x) \land R(x, x', t)) \).

| PROOF OBLIGATIONS |
|---------------------|
| \( (\text{INV1}) \) Init\( (x) \Rightarrow I(x) \) |
| \( (\text{INV2}) \) \( I(x) \land BA(e)(x, x') \Rightarrow I(x') \) |
| \( (\text{FIS}) \) \( I(x) \land \text{grd}(e)(x) \Rightarrow \exists y. BA(e)(x, y) \) |

Table-1 EVENTB proof obligations

Proof obligations (INV 1 and INV 2) are produced by the RODIN tool \cite{18} from events to state that an invariant condition \( I(x) \) is preserved. Their general form follows immediately from the definition of the before–after predicate \( BA(e)(x, x') \) of each event \( e \) (see Table-1). Note that it follows from the two guarded forms of the events that this obligation is trivially discharged when the guard of the event is false. Whenever this is the case, the event is said to be disabled. The proof obligation FIS expresses the feasibility of the event \( e \) with respect to the invariant \( I \). By proving feasibility, we achieve that \( BA(e)(x, y) \) provides an after state whenever \( \text{grd}(e)(x) \) holds. This means that the guard indeed represents the enabling condition of the event.

The intention of specifying a guard of an event is that the event may always occur when a given guard is true. There is, however, some interaction between guards and nondeterministic assignments, namely \( x : |BA(e)(x, x') \). The predicate \( BA(e)(x, x') \)
of an action $x : BA(e)(x, x')$ is not satisfiable or a set $(S)$ is empty in an action predicate $(v : \in S)$. Both cases show violations of the event feasibility proof obligation. We say that an assignment is feasible if there is an after-state satisfying the corresponding before-after predicate. For each event, its feasibility must be proved. Note, that for deterministic assignments the proof of feasibility is trivial. Also note, that feasibility of the initialization of a machine yields the existence of an initial state of the machine. It is not necessary to require an extra initialization.

B.2 Model Refinement

The refinement of a formal model allows us to enrich the model via a step-by-step approach and is the foundation of our correct-by-construction approach [15]. Refinement provides a way to strengthen invariants and to add details to a model. It is also used to transform an abstract model to a more concrete version by modifying the state description. This is done by extending the list of state variables (possibly suppressing some of them), by refining each abstract event to a set of possible concrete versions, and by adding new events. The abstract $(x)$ and concrete $(y)$ state variables are linked by means of a gluing invariant $J(x, y)$. A number of proof obligations ensure that (1) each abstract event is correctly refined by its corresponding concrete version, (2) each new event refines $skip$, (3) no new event takes control for ever, and (4) relative deadlock freedom is preserved. Details of the formulation of these proofs follows.

We suppose that an abstract model $AM$ with variables $x$ and invariant $I(x)$ is refined by a concrete model $CM$ with variables $y$ and gluing invariant $J(x, y)$. Event $e$ is in abstract model $AM$ and event $f$ is in concrete model $CM$. Event $f$ refines event $e$. $BA(e)(x, x')$ and $BA(f)(y, y')$ are predicates of events $e$ and $f$ respectively, we have to prove the following statement, corresponding to proof obligation (1):

$$I(x) \land J(x, y) \land BA(f)(y, y') \Rightarrow \exists x' \cdot (BA(e)(x, x') \land J(x', y'))$$

The new events introduced in a refinement step can be viewed as hidden events not visible to the environment of a system and are thus outside the control of the environment. In EVENT B, requiring a new event to refine $skip$ means that the effect of the new event is not observable in the abstract model. Any number of executions of an internal action may occur in between each execution of a visible action. Now, proof obligation (2) states that $BA(f)(y, y')$ must refine $skip (x' = x)$, generating the following simple statement to prove (2):

$$I(x) \land J(x, y) \land BA(f)(y, y') \Rightarrow J(x, y')$$

In refining a model, an existing event can be refined by strengthening the guard and/or the before–after predicate (effectively reducing the degree of nondeterminism), or a new event can be added to refine the skip event. The feasibility condition is crucial to avoiding possible states that have no successor, such as division by zero. Furthermore, this refinement guarantees that the set of traces of the refined model contains (up to stuttering) the traces of the resulting model. The refinement of an event $e$ by an event $f$ means that the event $f$ simulates the event $e$.

The EVENT B modeling language is supported by the RODIN platform [18] and has been introduced in publications [1], where the many case studies and discussions about the language itself and the foundations of the EVENT B approach. The language
of generalized substitutions is very rich, enabling the expression of any relation between states in a set-theoretical context. The expressive power of the language leads to a requirement for help in writing relational specifications, which is why we should provide guidelines for assisting the development of Event B models.

B.3 Time-Based Pattern in Event-B

The purpose of a design pattern [13] is to capture structures and to make decisions within a design that are common to similar modeling and analysis tasks. They can be re-applied when undertaking similar tasks in order to reduce the duplication of effort. The design pattern approach is the possibility to reuse solutions from earlier developments in the current project. This will lead to a correct refinement in the chain of models, without arising proof obligations. Since the correctness (i.e proof obligations are proved) of the pattern has been proved during its development, nothing is to be proved again when using this pattern.

The landing gear system is characterized by their functions, which can be expressed by analyzing the real-time patterns. Sequence of operations related to doors and gears, are performed under the real-time constraints. D. Cansell et. all [9] have introduced the time constraint pattern. In this case study, we use the same time pattern to solve the timing requirements of the landing system. This time pattern is fully based on timed automaton. The timed automaton is a finite state machine that is useful to model the components of real-time systems. In a model, the timed automata interacts with each other and defines a timed transition system. Besides ordinary action transitions that can represent input, output and internal actions. A timed transition system has time progress transitions. Such time progress transitions result in synchronous progress of all clock variables in the model. Here we apply the time pattern to model the sequential operations of doors and gears of the landing system in continuous progressive time constraint. In the model every events are controlled under time constraints, which means action of any event activates only when time constraint satisfies on specific time. The time progress is also an event, so there is no modification of the underlying B language. It is only a modeling technique instead of a specialized formal system.

The timed variable is in \( \mathbb{N} \) (natural numbers) but the time constraint can be written in terms involving unknown constants or expressions between different times. Finally, the timed event observations can be constrained by other events which determine future activations.

B.4 Tools Environments for Event B

The Event B modeling language is supported by the Atelier B [10] environment and by the RODIN platform [18]. Both environments provide facilities for editing machines, refinements, contexts and projects, for generating proof obligations corresponding to a given property, for proving proof obligations in an automatic or/and interactive process and for animating models. The internal prover is shared by the two environments and there are hints generated by the prover interface for helping the interactive proofs. However, the refinement process of machines should be progressive when adding new elements to a given current model and the goal is to distribute the complexity of proofs through the proof-based refinement. These tools are based on logical and semantical concepts of Event B models (machines, contexts,refinement) and our methodology for modeling medical protocol or guidelines can be built from them.
MACHINE M1
SEES C0
VARIABLES
  button
  phase
INVARIANTS
  inv1 : button ∈ POSITIONS
  inv2 : phase ∈ PHASES
  inv3 : phase = movingup ⇒ button = UP
  inv4 : phase = movingdown ⇒ button = DOWN
  inv5 : button = UP ⇒ phase ∉ {movingdown, haltdown}
  inv6 : button = DOWN ⇒ phase ∉ {movingup, haltup}
EVENTS
Initialisation
begin
  act1 : button := DOWN
  act2 : phase := haltdown
end
Event PressDOWN ≜
when
  grd1 : button = UP
then
  act1 : phase := movingdown
  act2 : button := DOWN
end
Event PressUP ≜
when
  grd1 : button = DOWN
then
  act1 : phase := movingup
  act2 : button := UP
end
Event movingup ≜
when
  grd1 : phase = movingup
then
  act1 : phase := haltup
end
An Event-B Specification of M2
Creation Date: 27Jan2014 @ 10:44:59 AM

MACHINE M2
REFINES M1
SEES C0

VARIABLES

dstate
lstate
phase
button
p
l
i

INvariants

inv1 : dstate ∈ DOORS → SDOORS
inv2 : dstate⁻¹[OPEN] \neq \emptyset \Rightarrow dstate⁻¹[OPEN] = DOORS
when one door is open, each door is open.
inv3 : dstate⁻¹[CLOSED] \neq \emptyset \Rightarrow dstate⁻¹[CLOSED] = DOORS
when a door is closed, each door is closed.
inv6 : lstate ∈ DOORS \rightarrow SLOCKS
inv7 : dstate[DOORS] = \{OPEN\} \Rightarrow lstate[DOORS] = \{UNLOCKED\}
inv12 : p ∈ P
inv13 : l ∈ P
inv14 : i ∈ P
inv15 : l = E \land p = R \Rightarrow lstate[DOORS] = \{UNLOCKED\}
inv16 : l = R \land p = E \Rightarrow lstate[DOORS] = \{UNLOCKED\}

EVENTS
Initialisation
extended
begin
act1 : button := DOWN
act2 : phase := haltdown
act3 : dstate : \[(dstate' \in DOORS \rightarrow SDOORS \land dstate' = \{a \mapsto b|a \in DOORS \land b = CLOSED\})\]

missing elements of the invariant
act4 : lstate := \{a \mapsto b|a \in DOORS \land b = LOCKED\}
act5 : p := R
act6 : l := R
act7 : i := R

end

Event opening_doors_DOWN \(=\)
when

grd1 : dstate[DOORS] = \{CLOSED\}
grd5 : lstate[DOORS] = \{UNLOCKED\}
grd7 : phase = movingdown
grd8 : p = R
grd9 : l = R
then

act1 : dstate := \{a \mapsto b|a \in DOORS \land b = OPEN\}
act2 : p := E

end

Event opening_doors_UP \(=\)
when

grd1 : dstate[DOORS] = \{CLOSED\}
grd4 : lstate[DOORS] = \{UNLOCKED\}
grd5 : phase = movingup
grd6 : p = E
grd7 : l = E
then

act1 : dstate := \{a \mapsto b|a \in DOORS \land b = OPEN\}
act2 : p := R

end

Event closing_doors_UP \(=\)
any

\(f\)
where

grd1 : dstate[DOORS] = \{OPEN\}
grd3 : \(f \in DOORS \rightarrow SDOORS\)
grd4 : \(\forall e \in DOORS \Rightarrow f(e) = CLOSED\)
grd5 : phase = movingup
grd6 : p = R
then

act1 : dstate := f

end

Event closing_doors_DOWN \(=\)
any

\(f\)
where

\[
\begin{align*}
grd1 &: \text{dstate}[DOORS] = \{\text{OPEN}\} \\
grd3 &: f \in DOORS \rightarrow SDOORS \\
grd4 &: \forall e \in DOORS \Rightarrow f(e) = \text{CLOSED} \\
grd5 &: \text{phase} = \text{movingdown} \\
grd6 &: p = E
\end{align*}
\]

then

\[
\begin{align*}
\text{act1} &: \text{dstate} := f
\end{align*}
\]

end

Event \textit{unlocking\_UP} \cong

when

\[
\begin{align*}
grd3 &: \text{lstate}[DOORS] = \{\text{LOCKED}\} \\
grd4 &: \text{phase} = \text{movingup} \\
grd5 &: l = E \\
grd6 &: p = E \\
grd7 &: i = E
\end{align*}
\]

then

\[
\begin{align*}
\text{act1} &: \text{lstate} := \{a \mapsto b|a \in DOORS \land b = \text{UNLOCKED}\}
\end{align*}
\]

end

Event \textit{locking\_UP} \cong

refines \textit{movingup}

when

\[
\begin{align*}
grd3 &: \text{dstate}[DOORS] = \{\text{CLOSED}\} \\
grd4 &: \text{phase} = \text{movingup} \\
grd5 &: \text{lstate}[DOORS] = \{\text{UNLOCKED}\} \\
grd6 &: p = R \\
grd7 &: i = E
\end{align*}
\]

then

\[
\begin{align*}
\text{act1} &: \text{lstate} := \{a \mapsto b|a \in DOORS \land b = \text{LOCKED}\} \\
\text{act3} &: \text{phase} := \text{haltup} \\
\text{act4} &: l := R \\
& \quad \text{added by D Mery}
\end{align*}
\]

end

Event \textit{unlocking\_DOWN} \cong

when

\[
\begin{align*}
grd3 &: \text{lstate}[DOORS] = \{\text{LOCKED}\} \\
grd4 &: \text{phase} = \text{movingdown} \\
grd5 &: l = R \\
grd6 &: p = R \\
grd7 &: i = R
\end{align*}
\]

then

\[
\begin{align*}
\text{act1} &: \text{lstate} := \{a \mapsto b|a \in DOORS \land b = \text{UNLOCKED}\}
\end{align*}
\]

end

Event \textit{locking\_DOWN} \cong

refines \textit{movingdown}
when

\[ \text{grd1 : } dstate[\text{DOORS}] = \{\text{CLOSED}\} \]
\[ \text{grd2 : } \text{phase} = \text{movingdown} \]
\[ \text{grd3 : } lstate[\text{DOORS}] = \{\text{UNLOCKED}\} \]
\[ \text{grd4 : } p = E \]
\[ \text{grd5 : } l = R \]

then

\[ \text{act1 : } lstate := \{a \mapsto b | a \in \text{DOORS} \land b = \text{LOCKED}\} \]
\[ \text{act3 : } \text{phase} := \text{haltdown} \]
\[ \text{act4 : } l := E \]

end

Event \( PD1 \equiv \) reflects \( \text{PressDOWN} \)

when

\[ \text{grd1 : } \text{button} = \text{UP} \]
\[ \text{grd2 : } \text{phase} = \text{haltup} \]

then

\[ \text{act1 : } \text{phase} := \text{movingdown} \]
\[ \text{act2 : } \text{button} := \text{DOWN} \]
\[ \text{act3 : } l := R \]
\[ \text{act4 : } p := R \]
\[ \text{act5 : } i := R \]

end

Event \( PU1 \equiv \) reflects \( \text{PressUP} \)

when

\[ \text{grd1 : } \text{button} = \text{DOWN} \]
\[ \text{grd2 : } \text{phase} = \text{haltdown} \]

then

\[ \text{act1 : } \text{phase} := \text{movingup} \]
\[ \text{act2 : } \text{button} := \text{UP} \]
\[ \text{act3 : } l := E \]
\[ \text{act4 : } p := E \]
\[ \text{act5 : } i := E \]

end

Event \( PU2 \equiv \) reflects \( \text{PressUP} \)

when

\[ \text{grd1 : } l = R \]
\[ \text{grd2 : } p = R \]
\[ \text{grd3 : } \text{phase} = \text{movingdown} \]
\[ \text{grd4 : } \text{button} = \text{DOWN} \]
\[ \text{grd5 : } i = R \]
\[ \text{grd6 : } lstate[\text{DOORS}] = \{\text{LOCKED}\} \]

then

\[ \text{act1 : } \text{phase} := \text{movingup} \]
act4 : button := UP
act5 : l := E
act6 : p := E
act7 : i := R
end

Event CompletePU2 \equiv
refines movingup
when
grd1 : phase = movingup
grd2 : button = UP
grd3 : l = E
grd4 : p = E
grd5 : i = R
then
act1 : phase := haltup
end

Event PU3 \equiv
refines PressUP
when
grd1 : dstate[DOORS] = \{CLOSED\}
grd2 : lstate[DOORS] = \{UNLOCKED\}
grd3 : phase = movingdown
grd4 : p = R
grd5 : l = R
grd6 : button = DOWN
then
act1 : phase := movingup
act2 : p := R
act3 : l := E
act4 : button := UP
end

Event PU4 \equiv
refines PressUP
when
grd1 : dstate[DOORS] = \{OPEN\}
grd2 : phase = movingdown
grd3 : p = E
then
grd4 : button = DOWN
then
act1 : phase := movingup
act2 : p := R
act3 : button := UP
act4 : i := E
act5 : l := E
end

Event PU5 \equiv
refines $\mathtt{PressUP}$
\begin{itemize}
\item when
\begin{itemize}
\item $\text{grd1} : \text{dstate}[DOORS] = \{\text{CLOSED}\}$
\item $\text{grd2} : \text{phase} = \text{movingdown}$
\item $\text{grd3} : p = E$
\item $\text{grd4} : \text{button} = \text{DOWN}$
\item $\text{grd5} : \text{lstate}[DOORS] = \{\text{UNLOCKED}\}$
\end{itemize}
\item then
\begin{itemize}
\item $\text{act1} : \text{phase} := \text{movingup}$
\item $\text{act3} : \text{button} := \text{UP}$
\item $\text{act4} : i := E$
\item $\text{act5} : l := E$
\end{itemize}
\end{itemize}
\begin{itemize}
\item Event $PD2 \triangleq$
\item refines $\mathtt{PressDOWN}$
\begin{itemize}
\item when
\begin{itemize}
\item $\text{grd1} : l = E$
\item $\text{grd2} : p = E$
\item $\text{grd3} : \text{phase} = \text{movingup}$
\item $\text{grd4} : i = E$
\item $\text{grd5} : \text{lstate}[DOORS] = \{\text{LOCKED}\}$
\end{itemize}
\item then
\begin{itemize}
\item $\text{act1} : \text{phase} := \text{movingdown}$
\item $\text{act2} : \text{button} := \text{DOWN}$
\item $\text{act3} : l := R$
\item $\text{act4} : p := R$
\item $\text{act5} : i := E$
\end{itemize}
\end{itemize}
\end{itemize}
\begin{itemize}
\item Event $\text{CompletePD2} \triangleq$
\item refines $\mathtt{movingdown}$
\begin{itemize}
\item when
\begin{itemize}
\item $\text{grd1} : \text{phase} = \text{movingdown}$
\item $\text{grd2} : \text{button} = \text{DOWN}$
\item $\text{grd3} : l = R$
\item $\text{grd4} : p = R$
\item $\text{grd5} : i = E$
\end{itemize}
\item then
\begin{itemize}
\item $\text{act1} : \text{phase} := \text{haltdown}$
\end{itemize}
\end{itemize}
\end{itemize}
\begin{itemize}
\item Event $PD3 \triangleq$
\item refines $\mathtt{PressDOWN}$
\begin{itemize}
\item when
\begin{itemize}
\item $\text{grd1} : \text{dstate}[DOORS] = \{\text{CLOSED}\}$
\item $\text{grd2} : \text{lstate}[DOORS] = \{\text{UNLOCKED}\}$
\item $\text{grd3} : \text{phase} = \text{movingup}$
\item $\text{grd4} : p = E$
\end{itemize}
\end{itemize}
\end{itemize}
grd5 : \( l = E \)
grd6 : \( \text{button} = UP \)

then

act1 : \( \text{phase} := \text{moving down} \)
act2 : \( p := E \)
act3 : \( l := R \)
act4 : \( \text{button} := \text{DOWN} \)

end

Event \( PD4 \) \( \equiv \)
refines \( \text{PressDOWN} \)
when

grd1 : \( dstate[\text{DOORS}] = \{ \text{OPEN} \} \)
grd2 : \( \text{phase} = \text{moving up} \)
grd3 : \( p = R \)
grd4 : \( \text{button} = UP \)
then

act1 : \( \text{phase} := \text{moving down} \)
act2 : \( p := E \)
act3 : \( \text{button} := \text{DOWN} \)
act4 : \( i := R \)
act5 : \( l := R \)

end

Event \( PD5 \) \( \equiv \)
refines \( \text{PressDOWN} \)
when

grd1 : \( dstate[\text{DOORS}] = \{ \text{CLOSED} \} \)
grd2 : \( \text{phase} = \text{moving up} \)
grd3 : \( p = R \)
grd4 : \( \text{button} = UP \)
grd5 : \( lstate[\text{DOORS}] = \{ \text{UNLOCKED} \} \)
then

act1 : \( \text{phase} := \text{moving down} \)
act2 : \( \text{button} := \text{DOWN} \)
act3 : \( i := R \)
act4 : \( l := R \)

end

END

E M3
SEES C0

VARIABLES

dstate
lstate
phase
button
p
l
i
gstate

INVARIANTS

\[ M_{3\_inv1} : gstate \in GEARS \rightarrow SGEARS \]
\[ M_{3\_inv3} : \forall \text{door} \cdot \text{door} \in DOORS \land dstate(\text{door}) = CLOSED \land \text{ran}(gstate) \neq \{\text{RETRACTED}\} \Rightarrow \text{ran}(gstate) = \{\text{EXTENDED}\} \]
   gears can not be out or moving in this case.
\[ M_{3\_inv6} : \forall \text{door} \cdot \text{door} \in DOORS \land dstate(\text{door}) = CLOSED \land \text{ran}(gstate) \neq \{\text{EXTENDED}\} \Rightarrow \text{ran}(gstate) = \{\text{RETRACTED}\} \]
\[ M_{3\_inv7} : \text{ran}(gstate) \neq \{\text{RETRACTED}\} \land \text{ran}(gstate) \neq \{\text{EXTENDED}\} \Rightarrow \text{ran}(dstate) = \{\text{OPEN}\} \]
\[ M_{3\_inv11} : \text{ran}(dstate) = \{\text{CLOSED}\} \Rightarrow \text{ran}(gstate) \cap \{\text{RETRACTING, EXTENDING}\} = \emptyset \]

EVENTS

Initialisation
extended

begin

act1 : button := DOWN
act2 : phase := halldown
act3 : dstate : \{(dstate' \in DOORS \rightarrow SDOORS \land dstate' = \{a \mapsto b | a \in DOORS \land b = CLOSED\})
   missing elements of the invariant
act4 : lstate := \{a \mapsto b | a \in DOORS \land b = LOCKED\}
act5 : p := R
act6 : l := R
act7 : i := R
act8 : gstate : \{(gstate' \in GEARS \rightarrow SGEARS \land gstate' = \{a \mapsto b | a \in GEARS \land b = EXTENDED\})

end

Event opening_doors_DOWN \equiv
extends opening_doors_DOWN

when

grd1 : dstate[DOORS] = \{CLOSED\}
grd5 : lstate[DOORS] = \{UNLOCKED\}
grd7 : phase = movingdown
grd8 : p = R
grd9 : l = R
then
  \( \text{act1} : \text{dstate} := \{a \mapsto b | a \in \text{DOORS} \land b = \text{OPEN} \} \)
  \( \text{act2} : p := E \)
end

Event \( \text{opening\_doors\_UP} \equiv \) extends \( \text{opening\_doors\_UP} \)

when
  \( \text{grd1} : \text{dstate}[\text{DOORS}] = \{\text{CLOSED} \} \)
  \( \text{grd4} : \text{lstate}[\text{DOORS}] = \{\text{UNLOCKED} \} \)
  \( \text{grd5} : \text{phase} = \text{movingup} \)
  \( \text{grd6} : p = E \)
  \( \text{grd7} : l = E \)
then
  \( \text{act1} : \text{dstate} := \{a \mapsto b | a \in \text{DOORS} \land b = \text{OPEN} \} \)
  \( \text{act2} : p := R \)
end

Event \( \text{closing\_doors\_UP} \equiv \) refines \( \text{closing\_doors\_UP} \)

any
  \( f \)
where
  \( \text{grd1} : \text{dstate}[\text{DOORS}] = \{\text{OPEN} \} \)
  \( \text{grd3} : f \in \text{DOORS} \rightarrow \text{SDOORS} \)
  \( \text{grd4} : \forall e : e \in \text{DOORS} \Rightarrow f(e) = \text{CLOSED} \)
  \( \text{grd5} : \text{phase} = \text{movingup} \)
  \( \text{grd6} : p = R \)
  \( \text{grd7} : \text{gstate}[\text{GEARS}] = \{\text{RETRACTED} \} \)
then
  \( \text{act1} : \text{dstate} := f \)
end

Event \( \text{closing\_doors\_DOWN} \equiv \) refines \( \text{closing\_doors\_DOWN} \)

any
  \( f \)
where
  \( \text{grd1} : \text{dstate}[\text{DOORS}] = \{\text{OPEN} \} \)
  \( \text{grd3} : f \in \text{DOORS} \rightarrow \text{SDOORS} \)
  \( \text{grd4} : \forall e : e \in \text{DOORS} \Rightarrow f(e) = \text{CLOSED} \)
  \( \text{grd5} : \text{phase} = \text{movingdown} \)
  \( \text{grd6} : p = E \)
  \( \text{grd7} : \text{gstate}[\text{GEARS}] = \{\text{EXTENDED} \} \)
then
  \( \text{act1} : \text{dstate} := f \)
end

Event \( \text{unlocking\_UP} \equiv \)
extends unlocking_UP
when
\[ \text{grd3 : } lstate[DOORS] = \{\text{LOCKED}\} \]
\[ \text{grd4 : } \text{phase} = \text{movingup} \]
\[ \text{grd5 : } l = E \]
\[ \text{grd6 : } p = E \]
\[ \text{grd7 : } i = E \]
then
\[ \text{act1 : } lstate := \{a \mapsto b|a \in DOORS \land b = \text{UNLOCKED}\} \]
end

Event locking_UP  \equiv
extends locking_UP
when
\[ \text{grd3 : } dstate[DOORS] = \{\text{CLOSED}\} \]
\[ \text{grd4 : } \text{phase} = \text{movingup} \]
\[ \text{grd5 : } lstate[DOORS] = \{\text{UNLOCKED}\} \]
\[ \text{grd6 : } p = R \]
\[ \text{grd7 : } i = E \]
then
\[ \text{act1 : } lstate := \{a \mapsto b|a \in DOORS \land b = \text{LOCKED}\} \]
\[ \text{act3 : } \text{phase} := \text{haltup} \]
\[ \text{act4 : } l := R \]
added by D Mery
end

Event unlocking_DOWN  \equiv
extends unlocking_DOWN
when
\[ \text{grd3 : } lstate[DOORS] = \{\text{LOCKED}\} \]
\[ \text{grd4 : } \text{phase} = \text{movingdown} \]
\[ \text{grd5 : } l = R \]
\[ \text{grd6 : } p = R \]
\[ \text{grd7 : } i = R \]
then
\[ \text{act1 : } lstate := \{a \mapsto b|a \in DOORS \land b = \text{UNLOCKED}\} \]
end

Event locking_DOWN  \equiv
extends locking_DOWN
when
\[ \text{grd1 : } \text{dstate}[DOORS] = \{\text{CLOSED}\} \]
\[ \text{grd2 : } \text{phase} = \text{movingdown} \]
\[ \text{grd3 : } lstate[DOORS] = \{\text{UNLOCKED}\} \]
\[ \text{grd4 : } p = E \]
\[ \text{grd5 : } l = R \]
then
\[ \text{act1 : } lstate := \{a \mapsto b|a \in DOORS \land b = \text{LOCKED}\} \]
act3 : \text{phase} := \text{haltdown} \\
act4 : l := E \\
\textbf{end}

\textbf{Event} \ PD1 \ \textbf{extends} \ PD1 \\
\textbf{when} \\
\quad \text{grd1 : button} = \text{UP} \\
\quad \text{grd2 : phase} = \text{haltup} \\
\textbf{then} \\
\quad \text{act1 : phase} := \text{movingdown} \\
\quad \text{act2 : button} := \text{DOWN} \\
\quad \text{act3 : l} := R \\
\quad \text{act4 : p} := R \\
\quad \text{act5 : i} := R \\
\textbf{end}

\textbf{Event} \ PU1 \ \textbf{extends} \ PU1 \\
\textbf{when} \\
\quad \text{grd1 : button} = \text{DOWN} \\
\quad \text{grd2 : phase} = \text{haltdown} \\
\textbf{then} \\
\quad \text{act1 : phase} := \text{movingup} \\
\quad \text{act2 : button} := \text{UP} \\
\quad \text{act3 : l} := E \\
\quad \text{act4 : p} := E \\
\quad \text{act5 : i} := E \\
\textbf{end}

\textbf{Event} \ PU2 \ \textbf{extends} \ PU2 \\
\textbf{when} \\
\quad \text{grd1 : l} = R \\
\quad \text{grd2 : p} = R \\
\quad \text{grd3 : phase} = \text{movingdown} \\
\quad \text{grd4 : button} = \text{DOWN} \\
\quad \text{grd5 : i} = R \\
\quad \text{grd6 : lstate}[\text{DOORS}] = \{\text{LOCKED}\} \\
\textbf{then} \\
\quad \text{act1 : phase} := \text{movingup} \\
\quad \text{act4 : button} := \text{UP} \\
\quad \text{act5 : l} := E \\
\quad \text{act6 : p} := E \\
\quad \text{act7 : i} := R \\
\textbf{end}

\textbf{Event} \ CompletePU2 \ \textbf{extends} \ CompletePU2 \\
\textbf{when}
\text{grd1} : \text{phase} = \text{movingup} \\
\text{grd2} : \text{button} = UP \\
\text{grd3} : l = E \\
\text{grd4} : p = E \\
\text{grd5} : i = R \\
\text{then} \\
\text{act1} : \text{phase} := \text{haltup} \\
\text{end} \\
\text{Event} \ PU3 \equiv \\
\text{extends} \ PU3 \\
\text{when} \\
\text{grd1} : \text{dstate}[\text{DOORS}] = \{\text{CLOSED}\} \\
\text{grd2} : \text{lstate}[\text{DOORS}] = \{\text{UNLOCKED}\} \\
\text{grd3} : \text{phase} = \text{movingdown} \\
\text{grd4} : p = R \\
\text{grd5} : l = R \\
\text{grd6} : \text{button} = DOWN \\
\text{then} \\
\text{act1} : \text{phase} := \text{movingup} \\
\text{act2} : p := R \\
\text{act3} : l := E \\
\text{act4} : \text{button} := UP \\
\text{end} \\
\text{Event} \ PU4 \equiv \\
\text{extends} \ PU4 \\
\text{when} \\
\text{grd1} : \text{dstate}[\text{DOORS}] = \{\text{OPEN}\} \\
\text{grd2} : \text{phase} = \text{movingdown} \\
\text{grd3} : p = E \\
\text{grd4} : \text{button} = DOWN \\
\text{then} \\
\text{act1} : \text{phase} := \text{movingup} \\
\text{act2} : p := R \\
\text{act3} : \text{button} := UP \\
\text{act4} : i := E \\
\text{act5} : l := E \\
\text{end} \\
\text{Event} \ PU5 \equiv \\
\text{extends} \ PU5 \\
\text{when} \\
\text{grd1} : \text{dstate}[\text{DOORS}] = \{\text{CLOSED}\} \\
\text{grd2} : \text{phase} = \text{movingdown} \\
\text{grd3} : p = E \\
\text{grd4} : \text{button} = DOWN \\
\text{grd5} : \text{lstate}[\text{DOORS}] = \{\text{UNLOCKED}\} \\
\text{then} \
act1 : phase := movingup
act3 : button := UP
act4 : i := E
act5 : l := E

end

Event PD2 ≡
extends PD2
when
grd1 : l = E
grd2 : p = E
grd3 : phase = movingup
grd4 : i = E
grd5 : lstate[DOORS] = \{LOCKED\}
then
act1 : phase := movingdown
act2 : button := DOWN
act3 : l := R
act4 : p := R
act5 : i := E
end

Event CompletePD2 ≡
extends CompletePD2
when
grd1 : phase = movingdown
grd2 : button = DOWN
grd3 : l = R
grd4 : p = R
grd5 : i = E
then
act1 : phase := haltdown
end

Event PD3 ≡
extends PD3
when
grd1 : dstate[DOORS] = \{CLOSED\}
grd2 : lstate[DOORS] = \{UNLOCKED\}
grd3 : phase = movingup
grd4 : p = E
grd5 : l = E
grd6 : button = UP
then
act1 : phase := movingdown
act2 : p := E
act3 : l := R
act4 : button := DOWN
end
Event $PD4 \equiv$
extends $PD4$
when
  grd1 : $dstate[DOORS] = \{OPEN\}$
  grd2 : $phase = movingup$
  grd3 : $p = R$
  grd4 : $button = UP$
then
  act1 : $phase := movingdown$
  act2 : $p := E$
  act3 : $button := DOWN$
  act4 : $i := R$
  act5 : $l := R$
end
Event $PD5 \equiv$
extends $PD5$
when
  grd1 : $dstate[DOORS] = \{CLOSED\}$
  grd2 : $phase = movingup$
  grd3 : $p = R$
  grd4 : $button = UP$
  grd5 : $lstate[DOORS] = \{UNLOCKED\}$
then
  act1 : $phase := movingdown$
  act2 : $button := DOWN$
  act3 : $i := R$
  act4 : $l := R$
end
Event retracting_gears $\equiv$
when
  grd1 : $dstate[DOORS] = \{OPEN\}$
  grd2 : $gstate[GEARS] = \{EXTENDED\}$
  grd3 : $p = R$
then
  act1 : $gstate := \{a \mapsto b | a \in GEARs \land b = RETRACTING\}$
end
Event retraction $\equiv$
when
  grd1 : $dstate[DOORS] = \{OPEN\}$
  grd2 : $gstate[GEARS] = \{RETRACTING\}$
then
  act1 : $gstate := \{a \mapsto b | a \in GEARs \land b = RETRACTED\}$
end
Event extending_gears $\equiv$
when
\begin{align*}
grd1 & : dstate[DOORS] = \{\text{OPEN}\} \\
grd2 & : gstate[GEARS] = \{\text{RETRACTED}\} \\
grd3 & : p = E \\
then\hspace{2cm}
act1 & : gstate := \{a \mapsto b | a \in \text{GEARS} \land b = \text{EXTENDING}\} \\
end
\end{align*}

Event \textit{extension} \triangleq

\begin{align*}
grd1 & : dstate[DOORS] = \{\text{OPEN}\} \\
grd2 & : gstate[GEARS] = \{\text{EXTENDING}\} \\
then\hspace{2cm}
act1 & : gstate := \{a \mapsto b | a \in \text{GEARS} \land b = \text{EXTENDED}\} \\
end
\end{align*}

\textbf{END}

\textbf{F} \textbf{M4}

\textbf{MACHINE} \hspace{1cm} \textbf{M4}

\begin{align*}
\text{Reading Sensor} \\
\text{Computing Module}
\end{align*}

\textbf{REFINES} \hspace{1cm} \textbf{M3}

\textbf{SEES} \hspace{1cm} \textbf{C1}

\textbf{VARIABLES}

\begin{align*}
dstate \\
lstate \\
phase \\
button \\
p \\
l \\
i \\
gstate \\
handle \\
analogical\_switch \\
gear\_extended \\
gear\_retracted \\
gear\_shock\_absorber \\
door\_closed \\
door\_open \\
circuit\_pressurized \\
general\_EV \\
close\_EV
\end{align*}
retract_EV
extend_EV
open_EV
gears_locked_down
gears_man
anomaly
general_EV_func
close_EV_func
retract_EV_func
extend_EV_func
open_EV_func
gears_locked_down_func
gears_man_func
anomaly_func
A_Switch_Out

INVARIANTS

inv3 : handle ∈ 1..3 → POSITIONS
inv4 : analogical_switch ∈ 1..3 → A_Switch
inv5 : gear_extended ∈ 1..3 → (GEARS → BOOL)
inv6 : gear_retracted ∈ 1..3 → (GEARS → BOOL)
inv7 : gear_shock_absorber ∈ 1..3 → GEAR_ABSORBER
inv8 : door_closed ∈ 1..3 → (DOORS → BOOL)
inv9 : door_open ∈ 1..3 → (DOORS → BOOL)
inv10 : circuit_pressurized ∈ 1..3 → BOOL
inv13 : general_EV ∈ BOOL
inv14 : close_EV ∈ BOOL
inv15 : retract_EV ∈ BOOL
inv16 : extend_EV ∈ BOOL
inv18 : open_EV ∈ BOOL
inv19 : gears_locked_down ∈ BOOL
inv20 : gears_man ∈ BOOL
inv21 : anomaly ∈ BOOL
inv22 : general_EV_func ∈ (1..3 → POSITIONS) × (1..3 → A_Switch) × (1..3 → (GEARS → BOOL)) × (1..3 → (GEARS → BOOL)) × (1..3 → GEAR_ABSORBER) × (1..3 → (DOORS → BOOL)) × (1..3 → (DOORS → BOOL)) × (1..3 → BOOL) → BOOL
inv23 : close_EV_func ∈ (1..3 → POSITIONS) × (1..3 → A_Switch) × (1..3 → (GEARS → BOOL)) × (1..3 → (GEARS → BOOL)) × (1..3 → GEAR_ABSORBER) × (1..3 → (DOORS → BOOL)) × (1..3 → (DOORS → BOOL)) × (1..3 → BOOL) → BOOL
inv24 : retract_EV_func ∈ (1..3 → POSITIONS) × (1..3 → A_Switch) × (1..3 → (GEARS → BOOL)) × (1..3 → (GEARS → BOOL)) × (1..3 → GEAR_ABSORBER) × (1..3 → (DOORS → BOOL)) × (1..3 → (DOORS → BOOL)) × (1..3 → BOOL) → BOOL
\text{inv25} : \text{extend\_EV\_func} \in (1 \ldots 3 \to \text{POSITIONS}) \times (1 \ldots 3 \to \text{A\_Switch}) \times (1 \ldots 3 \to (\text{GEARS} \to \text{BOOL})) \times (1 \ldots 3 \to (\text{GEARS} \to \text{BOOL})) \times (1 \ldots 3 \to \text{GEAR\_ABSORBER}) \times (1 \ldots 3 \to (\text{DOORS} \to \text{BOOL})) \times (1 \ldots 3 \to (\text{DOORS} \to \text{BOOL})) \times (1 \ldots 3 \to \text{BOOL}) \to \text{BOOL} \\
\text{inv26} : \text{open\_EV\_func} \in (1 \ldots 3 \to \text{POSITIONS}) \times (1 \ldots 3 \to \text{A\_Switch}) \times (1 \ldots 3 \to (\text{GEARS} \to \text{BOOL})) \times (1 \ldots 3 \to (\text{GEARS} \to \text{BOOL})) \times (1 \ldots 3 \to \text{GEAR\_ABSORBER}) \times (1 \ldots 3 \to (\text{DOORS} \to \text{BOOL})) \times (1 \ldots 3 \to (\text{DOORS} \to \text{BOOL})) \times (1 \ldots 3 \to \text{BOOL}) \to \text{BOOL} \\
\text{inv27} : \text{gears\_locked\_down\_func} \in (1 \ldots 3 \to \text{POSITIONS}) \times (1 \ldots 3 \to \text{A\_Switch}) \times (1 \ldots 3 \to (\text{GEARS} \to \text{BOOL})) \times (1 \ldots 3 \to (\text{GEARS} \to \text{BOOL})) \times (1 \ldots 3 \to \text{GEAR\_ABSORBER}) \times (1 \ldots 3 \to (\text{DOORS} \to \text{BOOL})) \times (1 \ldots 3 \to (\text{DOORS} \to \text{BOOL})) \times (1 \ldots 3 \to \text{BOOL}) \to \text{BOOL} \\
\text{inv28} : \text{gears\_man\_func} \in (1 \ldots 3 \to \text{POSITIONS}) \times (1 \ldots 3 \to \text{A\_Switch}) \times (1 \ldots 3 \to (\text{GEARS} \to \text{BOOL})) \times (1 \ldots 3 \to (\text{GEARS} \to \text{BOOL})) \times (1 \ldots 3 \to \text{GEAR\_ABSORBER}) \times (1 \ldots 3 \to (\text{DOORS} \to \text{BOOL})) \times (1 \ldots 3 \to (\text{DOORS} \to \text{BOOL})) \times (1 \ldots 3 \to \text{BOOL}) \to \text{BOOL} \\
\text{inv29} : \text{anomaly\_func} \in (1 \ldots 3 \to \text{POSITIONS}) \times (1 \ldots 3 \to \text{A\_Switch}) \times (1 \ldots 3 \to (\text{GEARS} \to \text{BOOL})) \times (1 \ldots 3 \to (\text{GEARS} \to \text{BOOL})) \times (1 \ldots 3 \to \text{GEAR\_ABSORBER}) \times (1 \ldots 3 \to (\text{DOORS} \to \text{BOOL})) \times (1 \ldots 3 \to (\text{DOORS} \to \text{BOOL})) \times (1 \ldots 3 \to \text{BOOL}) \to \text{BOOL} \\
\text{inv30} : \text{A\_Switch\_Out} \in \text{BOOL} \\
\text{M1\_inv1} : \text{button} \in \text{POSITIONS} \\
\text{M1\_inv2} : \text{phase} \in \text{PHASES} \\
\text{M1\_inv3} : \text{phase} = \text{movingup} \Rightarrow \text{button} = \text{UP} \\
\text{M1\_inv4} : \text{phase} = \text{movingdown} \Rightarrow \text{button} = \text{DOWN} \\
\text{M1\_inv5} : \text{button} = \text{UP} \Rightarrow \text{phase} \notin \{\text{movingdown}, \text{haltdown}\} \\
\text{M1\_inv6} : \text{button} = \text{DOWN} \Rightarrow \text{phase} \notin \{\text{movingup}, \text{haltup}\} \\
\text{M2\_inv1} : \text{dstate} \in \text{DOORS} \rightarrow \text{SDOORS} \\
\text{M2\_inv2} : \text{dstate}^{-1}[\{\text{OPEN}\}] \neq \emptyset \Rightarrow \text{dstate}^{-1}[\{\text{OPEN}\}] = \text{DOORS} \\
\text{when one door is open, each door is open.} \\
\text{M2\_inv3} : \text{dstate}^{-1}[\{\text{CLOSED}\}] \neq \emptyset \Rightarrow \text{dstate}^{-1}[\{\text{CLOSED}\}] = \text{DOORS} \\
\text{when a door is closed, each door is closed} \\
\text{M2\_inv6} : \text{lstate} \in \text{DOORS} \rightarrow \text{SLOCKS} \\
\text{M2\_inv7} : \text{dstate}[\text{DOORS}] = \{\text{OPEN}\} \Rightarrow \text{lstate}[\text{DOORS}] = \{\text{UNLOCKED}\} \\
\text{M2\_inv12} : p \in P \\
\text{M2\_inv13} : l \in P \\
\text{M2\_inv14} : i \in P \\
\text{M2\_inv15} : l = E \land p = R \Rightarrow \text{lstate}[\text{DOORS}] = \{\text{UNLOCKED}\} \\
\text{M2\_inv16} : l = R \land p = E \Rightarrow \text{lstate}[\text{DOORS}] = \{\text{UNLOCKED}\} \\
\text{M3\_inv1} : \text{gstate} \in \text{GEARS} \rightarrow \text{SGEARS} \\
\text{M3\_inv3} : \forall \text{door} \in \text{DOORS} \wedge \text{dstate}(\text{door}) = \text{CLOSED} \wedge \text{ran}(\text{gstate}) \neq \{\text{RETRACTED}\} \Rightarrow \text{ran}(\text{gstate}) = \{\text{EXTENDED}\} \\
\text{gears can not be out or moving in this case.} \\
\text{M3\_inv6} : \forall \text{door} \in \text{DOORS} \wedge \text{dstate}(\text{door}) = \text{CLOSED} \wedge \text{ran}(\text{gstate}) \neq \{\text{EXTENDED}\} \Rightarrow \text{ran}(\text{gstate}) = \{\text{RETRACTED}\}
EVENTS
Initialisation

begin

act1 : button := DOWN
act2 : phase := haltdown
act3 : dstate : [(dstate' ∈ DOORS → SDOORS ∧ dstate' = {a ↦ b|a ∈ DOORS ∧ b = CLOSED})]

missing elements of the invariant
act4 : lstate := {a ↦ b|a ∈ DOORS ∧ b = LOCKED}
act5 : p := R
act6 : l := R
act7 : i := R
act8 : gstate : [(gstate' ∈ GEARS → SGEARS ∧ gstate' = {a ↦ b|a ∈ GEARS ∧ b = EXTENDED})]
act14 : handle ∈ 1..3 → {DOWN}
act15 : analogical_switch ∈ 1..3 → {open}
act16 : gear_extended ∈ 1..3 → {GEARS → {TRUE}}
act17 : gear_retracted ∈ 1..3 → {GEARS → {FALSE}}
act18 : gear_shock_absorber ∈ 1..3 → {ground}
act19 : door_closed ∈ 1..3 → {DOORS → {TRUE}}
act20 : door_open ∈ 1..3 → {DOORS → {FALSE}}
act21 : circuit_pressurized ∈ 1..3 → {FALSE}
act22 : general_EV := FALSE
act23 : close_EV := TRUE
act24 : retract_EV := FALSE
act25 : extend_EV := TRUE
act27 : open_EV := FALSE
act28 : gears_locked_down := TRUE
act29 : gears_man := FALSE
act30 : anomaly := FALSE
act31 : general_EV_func ∈ (1..3 → POSITIONS) × (1..3 → A_Switch) × (1..3 → GEAR_ABSORBER) × (1..3 → DOORS) × (1..3 → DOORS)
act32 : close_EV_func ∈ (1..3 → POSITIONS) × (1..3 → A_Switch) × (1..3 → GEAR_ABSORBER) × (1..3 → DOORS) × (1..3 → DOORS)
act33 : retract_EV_func ∈ (1..3 → POSITIONS) × (1..3 → A_Switch) × (1..3 → GEAR_ABSORBER) × (1..3 → DOORS) × (1..3 → DOORS)
Event opening_doors_DOWN $\equiv$
\begin{align*}
\text{act34} &: \text{extend_EV\_func} \in (1..3 \to \text{POSITIONS}) \times (1..3 \to A_{\text{Switch}}) \times (1..3 \to (\text{GEARS} \to \text{BOOL})) \times (1..3 \to (\text{GEARS} \to \text{BOOL})) \times (1..3 \to (\text{GEARS} \to \text{BOOL})) \times (1..3 \to (\text{DOORS} \to \text{BOOL})) \times (1..3 \to \text{DOORS} \to \text{BOOL}) \\
\text{act35} &: \text{open_EV\_func} \in (1..3 \to \text{POSITIONS}) \times (1..3 \to A_{\text{Switch}}) \times (1..3 \to (\text{GEARS} \to \text{BOOL})) \times (1..3 \to (\text{GEARS} \to \text{BOOL})) \times (1..3 \to (\text{GEARS} \to \text{BOOL})) \times (1..3 \to (\text{DOORS} \to \text{BOOL})) \times (1..3 \to (\text{DOORS} \to \text{BOOL})) \times (1..3 \to \text{DOORS} \to \text{BOOL}) \\
\text{act36} &: \text{gears\_locked\_down\_func} \in (1..3 \to \text{POSITIONS}) \times (1..3 \to A_{\text{Switch}}) \times (1..3 \to (\text{GEARS} \to \text{BOOL})) \times (1..3 \to (\text{GEARS} \to \text{BOOL})) \times (1..3 \to (\text{GEARS} \to \text{BOOL})) \times (1..3 \to (\text{DOORS} \to \text{BOOL})) \times (1..3 \to (\text{DOORS} \to \text{BOOL})) \times (1..3 \to \text{DOORS} \to \text{BOOL}) \\
\text{act37} &: \text{gears\_man\_func} \in (1..3 \to \text{POSITIONS}) \times (1..3 \to A_{\text{Switch}}) \times (1..3 \to (\text{GEARS} \to \text{BOOL})) \times (1..3 \to (\text{GEARS} \to \text{BOOL})) \times (1..3 \to (\text{GEARS} \to \text{BOOL})) \times (1..3 \to (\text{DOORS} \to \text{BOOL})) \times (1..3 \to (\text{DOORS} \to \text{BOOL})) \times (1..3 \to \text{DOORS} \to \text{BOOL}) \\
\text{act38} &: \text{anomaly\_func} \in (1..3 \to \text{POSITIONS}) \times (1..3 \to A_{\text{Switch}}) \times (1..3 \to (\text{GEARS} \to \text{BOOL})) \times (1..3 \to (\text{GEARS} \to \text{BOOL})) \times (1..3 \to (\text{GEARS} \to \text{BOOL})) \times (1..3 \to (\text{DOORS} \to \text{BOOL})) \times (1..3 \to (\text{DOORS} \to \text{BOOL})) \times (1..3 \to \text{DOORS} \to \text{BOOL}) \\
\text{act39} &: A_{\text{Switch\_Out}} := \text{FALSE}
\end{align*}

\textbf{Event} opening_doors\_DOWN $\equiv$
\begin{align*}
\text{when} & \quad \text{when} \\
\text{grd1} &: \text{dstate}[\text{DOORS}] = \{\text{CLOSED}\} \\
\text{grd5} &: \text{lstate}[\text{DOORS}] = \{\text{UNLOCKED}\} \\
\text{grd7} &: \text{phase} = \text{movingdown} \\
\text{grd8} &: p = R \\
\text{grd9} &: l = R \\
\text{grd10} &: \text{door\_open} = \{a \mapsto b | a \in 1..3 \land b \in \text{DOORS} \to \{\text{FALSE}\}\} \\
\text{grd11} &: \text{door\_closed} = \{a \mapsto b | a \in 1..3 \land b \in \text{DOORS} \to \{\text{FALSE}\}\} \\
\text{grd12} &: \forall x : x \in 1..3 \Rightarrow \text{handle}(x) = \text{button} \\
\text{then} & \quad \text{then} \\
\text{act1} &: \text{dstate} := \{a \mapsto b | a \in \text{DOORS} \land b = \text{OPEN}\} \\
\text{act2} &: p := E \\
\text{act3} &: \text{door\_open} \in 1..3 \to \text{DOORS} \to \{\text{TRUE}\}\)
\end{align*}

\textbf{Event} opening_doors\_UP $\equiv$
\begin{align*}
\text{when} & \quad \text{when} \\
\text{grd1} &: \text{dstate}[\text{DOORS}] = \{\text{CLOSED}\} \\
\text{grd4} &: \text{lstate}[\text{DOORS}] = \{\text{UNLOCKED}\} \\
\text{grd5} &: \text{phase} = \text{movingup} \\
\text{grd6} &: p = E \\
\text{grd7} &: l = E \\
\text{grd8} &: \text{door\_open} = \{a \mapsto b | a \in 1..3 \land b \in \text{DOORS} \to \{\text{FALSE}\}\} \\
\text{grd9} &: \text{door\_closed} = \{a \mapsto b | a \in 1..3 \land b \in \text{DOORS} \to \{\text{FALSE}\}\}
\end{align*}
∀x·x ∈ 1..3 ⇒ handle(x) = button

act1 : dstate := {a ↦ b|a ∈ DOORS ∧ b = OPEN}
at2 : p := R
act3 : door_open := 1..3 → (DOORS → {TRUE})

Event closing_doors_UP ≡
refines closing_doors_UP

any

f

where

grd1 : dstate[DOORS] = {OPEN}
grd3 : f ∈ DOORS → SDOORS
grd4 : ∀e ∈ DOORS ⇒ f(e) = CLOSED
grd5 : phase = movingup
grd6 : p = R
grd7 : gstate[GEARS] = {RETRACTED}
grd8 : ∀x·x ∈ 1..3 ⇒ handle(x) = button

then

act1 : dstate := f

end

Event closing_doors_DOWN ≡
refines closing_doors_DOWN

any

f

where

grd1 : dstate[DOORS] = {OPEN}
grd3 : f ∈ DOORS → SDOORS
grd4 : ∀e ∈ DOORS ⇒ f(e) = CLOSED
grd5 : phase = movingdown
grd6 : p = E
grd7 : gstate[GEARS] = {EXTENDED}
grd8 : ∀x·x ∈ 1..3 ⇒ handle(x) = button

then

act1 : dstate := f

end

Event unlocking_UP ≡
refines unlocking_UP

when

grd3 : lstate[DOORS] = {LOCKED}
grd4 : phase = movingup
grd5 : l = E
grd6 : p = E
grd7 : i = E
grd8 : door_open = {a ↦ b|a ∈ 1..3 ∧ b ∈ DOORS → {FALSE}}
grd9 : door_closed = {a ↦ b|a ∈ 1..3 ∧ b ∈ DOORS → {TRUE}}
∀x·x ∈ 1..3 ⇒ handle(x) = button

act1: lstate := \{a → b | a ∈ DOORS ∧ b = UNLOCKED\}
act2: door_closed ∈ 1..3 → (DOORS → {FALSE})

end

Event locking_UP ≡
refines locking_UP
when

grd3: dstate[DOORS] = {CLOSED}
grd4: phase = movingup
grd5: lstate[DOORS] = {UNLOCKED}
grd6: p = R
grd7: l = E
grd9: door_open = \{a → b | a ∈ 1..3 ∧ b ∈ DOORS → {FALSE}\}
grd10: door_closed = \{a → b | a ∈ 1..3 ∧ b ∈ DOORS → {FALSE}\}
grd11: ∀x·x ∈ 1..3 ⇒ handle(x) = button

then

act1: lstate := \{a → b | a ∈ DOORS ∧ b = LOCKED\}
act3: phase := haltup
act4: l := R
act44: door_closed ∈ 1..3 → (DOORS → {TRUE})

end

Event unlocking_DOWN ≡
refines unlocking_DOWN
when

grd3: lstate[DOORS] = {LOCKED}
grd4: phase = movingdown
grd5: l = R
grd6: p = R
grd7: l = R
grd8: door_open = \{a → b | a ∈ 1..3 ∧ b ∈ DOORS → {FALSE}\}
grd9: door_closed = \{a → b | a ∈ 1..3 ∧ b ∈ DOORS → {TRUE}\}
grd10: ∀x·x ∈ 1..3 ⇒ handle(x) = button

then

act1: lstate := \{a → b | a ∈ DOORS ∧ b = UNLOCKED\}
act2: door_closed ∈ 1..3 → (DOORS → {FALSE})

end

Event locking_DOWN ≡
refines locking_DOWN
when

grd1: dstate[DOORS] = {CLOSED}
grd2: phase = movingdown
grd3: lstate[DOORS] = {UNLOCKED}
grd4: p = E
grd5: l = R
\begin{verbatim}

grd7 : door_open = \{a \mapsto b | a \in 1..3 \land b \in DOORS \rightarrow \{FALSE\}\}
grd8 : door_closed = \{a \mapsto b | a \in 1..3 \land b \in DOORS \rightarrow \{FALSE\}\}
grd9 : \forall x \cdot x \in 1..3 \Rightarrow handle(x) = button

act1 : lstate := \{a \mapsto b | a \in DOORS \land b = LOCKED\}
act3 : phase := haltdown
act4 : l := E
act5 : door_closed :\in 1..3 \rightarrow (DOORS \rightarrow \{TRUE\})

\end{verbatim}
then

act1 : phase := movingup
act4 : button := UP
act5 : l := E
act6 : p := E
act7 : i := R

end

Event CompletePU2 ≡
refines CompletePU2

when

grd1 : phase = movingup
grd2 : button = UP
grd3 : l = E
grd4 : p = E
grd5 : i = R

then

act1 : phase := haltup

end

Event PU3 ≡
refines PU3

when

grd1 : dstate[DOORS] = \{CLOSED\}
grd2 : lstate[DOORS] = \{UNLOCKED\}
grd3 : phase = movingdown
grd4 : p = R
grd5 : l = R
grd6 : button = DOWN
grd7 : door_open = \{a \mapsto b | a \in 1..3 \land b \in DOORS \rightarrow \{FALSE\}\}
grd8 : door_closed = \{a \mapsto b | a \in 1..3 \land b \in DOORS \rightarrow \{FALSE\}\}
grd9 : \forall x \cdot x \in 1..3 \Rightarrow handle(x) = UP

then

act1 : phase := movingup
act2 : p := R
act3 : l := E
act4 : button := UP

end

Event PU4 ≡
refines PU4

when

grd1 : dstate[DOORS] = \{OPEN\}
grd2 : phase = movingdown
grd3 : p = E
grd4 : button = DOWN
grd7 : \forall x \cdot x \in 1..3 \Rightarrow handle(x) = UP

then

act1 : phase := movingup
act2 : p := R
act3 : button := UP
act4 : i := E
act5 : l := E

end

Event $PU5 \sqsubseteq$
refines $PU5$

when

grd1 : dstate[DOORS] = \{CLOSED\}
grd2 : phase = movingdown
grd3 : p = E
grd4 : button = DOWN
grd5 : lstate[DOORS] = \{UNLOCKED\}
grd6 : door_open = \{a \mapsto b | a \in 1..3 \land b \in DOORS \rightarrow \{FALSE\}\}
grd7 : door_closed = \{a \mapsto b | a \in 1..3 \land b \in DOORS \rightarrow \{FALSE\}\}
grd8 : \forall x \cdot x \in 1..3 \Rightarrow handle(x) = UP

then

act1 : phase := movingup
act3 : button := UP
act4 : i := E
act5 : l := E

end

Event $PD2 \sqsubseteq$
refines $PD2$

when

grd1 : l = E
grd2 : p = E
grd3 : phase = movingup
grd4 : i = E
grd5 : lstate[DOORS] = \{LOCKED\}
grd6 : \forall x \cdot x \in 1..3 \Rightarrow handle(x) = DOWN

then

act1 : phase := movingdown
act2 : button := DOWN
act3 : l := R
act4 : p := R
act5 : i := E

end

Event $CompletePD2 \sqsubseteq$
refines $CompletePD2$

when

grd1 : phase = movingdown
grd2 : button = DOWN
grd3 : l = R
grd4 : p = R
grd5 : i = E

then

act1 : phase := haltdown
Event $PD_3 \cong$
refs $PD_3$
when
  grd1 : $dstate[DOORS] = \{CLOSED\}$
  grd2 : $lstate[DOORS] = \{UNLOCKED\}$
  grd3 : $phase = \text{movingup}$
  grd4 : $p = E$
  grd5 : $l = E$
  grd6 : $button = UP$
  grd7 : $door\_open = \{a \mapsto b | a \in 1..3 \land b \in DOORS \rightarrow \{FALSE\}\}$
  grd8 : $door\_closed = \{a \mapsto b | a \in 1..3 \land b \in DOORS \rightarrow \{FALSE\}\}$
then
  act1 : $phase := \text{movingdown}$
  act2 : $p := E$
  act3 : $l := R$
  act4 : $button := DOWN$
end
Event $PD_4 \cong$
refs $PD_4$
when
  grd1 : $dstate[DOORS] = \{OPEN\}$
  grd2 : $phase = \text{movingup}$
  grd3 : $p = R$
  grd4 : $button = UP$
  grd6 : $\forall x.x \in 1..3 \Rightarrow handle(x) = DOWN$
then
  act1 : $phase := \text{movingdown}$
  act2 : $p := E$
  act3 : $button := DOWN$
  act4 : $i := R$
  act5 : $l := R$
end
Event $PD_5 \cong$
refs $PD_5$
when
  grd1 : $dstate[DOORS] = \{CLOSED\}$
  grd2 : $phase = \text{movingup}$
  grd3 : $p = R$
  grd4 : $button = UP$
  grd5 : $lstate[DOORS] = \{UNLOCKED\}$
  grd6 : $door\_open = \{a \mapsto b | a \in 1..3 \land b \in DOORS \rightarrow \{FALSE\}\}$
  grd7 : $door\_closed = \{a \mapsto b | a \in 1..3 \land b \in DOORS \rightarrow \{FALSE\}\}$
then
  act1 : $phase := \text{movingdown}$
act2 : button := DOWN
act3 : i := R
act4 : l := R

end

Event retracting_gears ≡
refines retracting_gears
when

grd1 : dstate[DOORS] = {OPEN}
grd2 : gstate[GEARS] = {EXTENDED}
grd3 : p = R
grd6 : gear_extended = \{a \mapsto b | a \in 1..3 \land b \in GEARs \rightarrow \{TRUE\}\}
grd7 : gear_retracted = \{a \mapsto b | a \in 1..3 \land b \in GEARs \rightarrow \{FALSE\}\}
grd8 : gear_shock_absorber = \{a \mapsto b | a \in 1..3 \land b = \text{ground}\}
grd9 : \forall x \cdot x \in 1..3 \Rightarrow handle(x) = \text{button}

then

act1 : gstate := \{a \mapsto b | a \in GEARs \land b = \text{RETRACTING}\}
act2 : gear_extended := \in 1..3 \rightarrow (GEARS \rightarrow \{FALSE\})
act3 : gear_shock_absorber := \{a \mapsto b | a \in 1..3 \land b = \text{flight}\}

end

Event retraction ≡
refines retraction
when

grd1 : dstate[DOORS] = {OPEN}
grd2 : gstate[GEARS] = {RETRACTING}
grd4 : gear_extended = \{a \mapsto b | a \in 1..3 \land b \in GEARs \rightarrow \{FALSE\}\}
grd5 : gear_retracted = \{a \mapsto b | a \in 1..3 \land b \in GEARs \rightarrow \{FALSE\}\}
grd6 : gear_shock_absorber = \{a \mapsto b | a \in 1..3 \land b = \text{flight}\}
grd7 : \forall x \cdot x \in 1..3 \Rightarrow handle(x) = \text{button}

then

act1 : gstate := \{a \mapsto b | a \in GEARs \land b = \text{RETRACTED}\}
act2 : gear_retracted := \in 1..3 \rightarrow (GEARS \rightarrow \{TRUE\})

end

Event extending_gears ≡
refines extending_gears
when

grd1 : dstate[DOORS] = {OPEN}
grd2 : gstate[GEARS] = {RETRACTED}
grd3 : p = E
grd5 : gear_retracted = \{a \mapsto b | a \in 1..3 \land b \in GEARs \rightarrow \{TRUE\}\}
grd6 : gear_extended = \{a \mapsto b | a \in 1..3 \land b \in GEARs \rightarrow \{FALSE\}\}
grd7 : gear_shock_absorber = \{a \mapsto b | a \in 1..3 \land b = \text{flight}\}
grd8 : \forall x \cdot x \in 1..3 \Rightarrow handle(x) = \text{button}

then

act1 : gstate := \{a \mapsto b | a \in GEARs \land b = \text{EXTENDING}\}
act2 : gear_retracted := \in 1..3 \rightarrow (GEARS \rightarrow \{FALSE\})

end
Event \( \text{extension} \triangleq \text{refines} \text{extension} \)
when

\[\begin{align*}
g_{\text{dstate}}[\text{DOORS}] &= \{\text{OPEN}\} \\
g_{\text{gstate}}[\text{GEARS}] &= \{\text{EXTENDING}\} \\
g_{\text{gear_retracted}} &= \{a \mapsto b | a \in 1..3 \land b \in \text{GEARS} \Rightarrow \{\text{FALSE}\}\} \\
g_{\text{gear_extended}} &= \{a \mapsto b | a \in 1..3 \land b \in \text{GEARS} \Rightarrow \{\text{FALSE}\}\} \\
g_{\text{gear_shock_absorber}} &= \{a \mapsto b | a \in 1..3 \land b = \text{flight}\} \\
g_{\forall}x : x \in 1..3 \Rightarrow \text{handle}(x) = \text{button} \\
\end{align*}\]

then

\[\begin{align*}
\text{act1} : & \text{gstate} := \{a \mapsto b | a \in \text{GEARS} \land b = \text{EXTENDED}\} \\
\text{act2} : & \text{gear_extended} : \in 1..3 \Rightarrow (\text{GEARS} \Rightarrow \{\text{TRUE}\}) \\
\text{act3} : & \text{gear_shock_absorber} := \{a \mapsto b | a \in 1..3 \land b = \text{ground}\} \\
\end{align*}\]

Event \( \text{HPDI} \triangleq \)
when

\[\begin{align*}
g_{\forall}x : x \in 1..3 \Rightarrow \text{handle}(x) = \text{UP} \\
\end{align*}\]

then

\[\text{act2} : \text{handle} : \in 1..3 \Rightarrow \{\text{DOWN}\} \]

Event \( \text{HPU1} \triangleq \)
when

\[\begin{align*}
g_{\forall}x : x \in 1..3 \Rightarrow \text{handle}(x) = \text{DOWN} \\
\end{align*}\]

then

\[\text{act2} : \text{handle} : \in 1..3 \Rightarrow \{\text{UP}\} \]

Event \( \text{Analogical switch closed} \triangleq \)
any

\[\begin{align*}
in & \text{ in port} \\
\text{where} \\
\text{g}_{\text{in}} & = \text{general}_{\text{EV}} \\
g_{\forall}x : x \in 1..3 \Rightarrow (\text{handle}(x) = \text{UP} \lor \text{handle}(x) = \text{DOWN}) \\
\text{then} \\
\text{act3} & : \text{analogical switch} : \in 1..3 \Rightarrow \{\text{closed}\} \\
\text{act4} & : \text{A Switch Out} := \text{TRUE} \\
\end{align*}\]

Event \( \text{Analogical switch open} \triangleq \)
any

\[\begin{align*}
in & \text{ in port} \\
\text{where} \\
\text{g}_{\text{in}} & = \text{general}_{\text{EV}} \\
g_{\forall}x : x \in 1..3 \Rightarrow (\text{handle}(x) = \text{UP} \lor \text{handle}(x) = \text{DOWN}) \\
\text{then} \\
\text{act3} & : \text{analogical switch} : \in 1..3 \Rightarrow \{\text{open}\} \\
\end{align*}\]
Event Circuit

begin

act9 : circuit_pressurized \in 1..3 \rightarrow BOOL

end

Event Computing Module 1

begin

act1 : general_EV := general_EV_func(handle \mapsto analogical_switch \mapsto
gear_extended \mapsto gear_retracted \mapsto gear_shock_absorber \mapsto door_open \mapsto
door_closed \mapsto circuit_pressurized)

act2 : close_EV := close_EV_func(handle \mapsto analogical_switch \mapsto
gear_extended \mapsto gear_retracted \mapsto gear_shock_absorber \mapsto door_open \mapsto
door_closed \mapsto circuit_pressurized)

act3 : retract_EV := retract_EV_func(handle \mapsto analogical_switch \mapsto
gear_extended \mapsto gear_retracted \mapsto gear_shock_absorber \mapsto door_open \mapsto
door_closed \mapsto circuit_pressurized)

act4 : extend_EV := extend_EV_func(handle \mapsto analogical_switch \mapsto
gear_extended \mapsto gear_retracted \mapsto gear_shock_absorber \mapsto door_open \mapsto
door_closed \mapsto circuit_pressurized)

act5 : open_EV := open_EV_func(handle \mapsto analogical_switch \mapsto
gear_extended \mapsto gear_retracted \mapsto gear_shock_absorber \mapsto door_open \mapsto
door_closed \mapsto circuit_pressurized)

act6 : gears_locked_down := gears_locked_down_func(handle \mapsto analogical_switch \mapsto
gear_extended \mapsto gear_retracted \mapsto gear_shock_absorber \mapsto door_open \mapsto
door_closed \mapsto circuit_pressurized)

act7 : gears_man := gears_man_func(handle \mapsto analogical_switch \mapsto
gear_extended \mapsto gear_retracted \mapsto gear_shock_absorber \mapsto door_open \mapsto
door_closed \mapsto circuit_pressurized)

act8 : anomaly := anomaly_func(handle \mapsto analogical_switch \mapsto
gear_extended \mapsto gear_retracted \mapsto gear_shock_absorber \mapsto door_open \mapsto
door_closed \mapsto circuit_pressurized)

end

Event Failure Detection

begin

act1 : anomaly := TRUE

end

END

G M5

An Event-B Specification of M5
Creation Date: 27Jan2014 @ 10:44:59 AM

MACHINE M5
Hydraulic circuit output for Electro-valves.
REFINES M4
SEES C1

VARIABLES

dstate
lstate
phase
button
p
l
i
gstate
handle
analogical_switch
gear_extended
gear_retracted
gear_shock_absorber
door_closed
door_open
circuit_pressurized
general_EV
close_EV
retract_EV
extend_EV
open_EV
gears_locked_down
gears_man
anomaly
general_EV_func
close_EV_func
retract_EV_func
extend_EV_func
open_EV_func
gears_locked_down_func
gears_man_func
anomaly_func
general_EV_Hout
close_EV_Hout
retract_EV_Hout
extend_EV_Hout
open_EV_Hout
A_Switch_Out

INVARIANTS

inv1 : general_EV_Hout ∈ {0, Hin}
inv2 : close_EV_Hout ∈ {0, Hin}
inv3 : retract_EV_Hout ∈ \{0, Hin\}
inv4 : extend_EV_Hout ∈ \{0, Hin\}
inv5 : open_EV_Hout ∈ \{0, Hin\}

EVENTS
Initialisation extended

begin

act1 : button := DOWN
act2 : phase := haltdown
act3 : dstate : |(dstate′ ∈ DOORS → SDOORS ∧ dstate′ = {a → b|a ∈ DOORS ∧ b = CLOSED})

missing elements of the invariant
act4 : lstate := {a → b|a ∈ DOORS ∧ b = LOCKED}
act5 : p := R
act6 : l := R
act7 : i := R
act8 : gstate : |(gstate′ ∈ GEARS → SGEARS ∧ gstate′ = {a → b|a ∈ GEARS ∧ b = EXTENDED})
act14 : handle := ∈ 1..3 → {DOWN}
act15 : analogical_switch := ∈ 1..3 → {open}
act16 : gear_extended := ∈ 1..3 → (GEARS → {TRUE})
act17 : gear_retracted := ∈ 1..3 → (GEARS → {FALSE})
act18 : gear_shock_absorber := ∈ 1..3 → {ground}
act19 : door_closed := ∈ 1..3 → (DOORS → {TRUE})
act20 : door_open := ∈ 1..3 → (DOORS → {FALSE})
act21 : circuit_pressurized := ∈ 1..3 → {FALSE}
act22 : general_EV := FALSE
act23 : close_EV := TRUE
act24 : retract_EV := FALSE
act25 : extend_EV := TRUE
act27 : open_EV := FALSE
act28 : gears_locked_down := TRUE
act29 : gears_man := FALSE
act30 : anomaly := FALSE
act31 : general_EV_func := ∈ (1..3 → POSITIONS) × (1..3 → A_Switch) × (1..3 → (GEARS → BOOL)) × (1..3 → (DOORS → BOOL)) × (1..3 → BOOL)
act32 : close_EV_func := ∈ (1..3 → POSITIONS) × (1..3 → A_Switch) × (1..3 → (GEARS → BOOL)) × (1..3 → (DOORS → BOOL)) × (1..3 → BOOL)
act33 : retract_EV_func := ∈ (1..3 → POSITIONS) × (1..3 → A_Switch) × (1..3 → (GEARS → BOOL)) × (1..3 → (DOORS → BOOL)) × (1..3 → BOOL)

52
\[
\text{act34 : extend\_EV\_func} : \in (1..3 \rightarrow \text{POSITIONS}) \times (1..3 \rightarrow A_{\text{Switch}}) \times (1..3 \rightarrow (\text{GEARS} \rightarrow \text{BOOL})) \times (1..3 \rightarrow \text{GEAR\_ABSORBER}) \times (1..3 \rightarrow (\text{DOORS} \rightarrow \text{BOOL})) \times (1..3 \rightarrow (\text{DOORS} \rightarrow \text{BOOL})) \times (1..3 \rightarrow \text{Hout}) \rightarrow \text{BOOL} \\
\text{act35 : open\_EV\_func} : \in (1..3 \rightarrow \text{POSITIONS}) \times (1..3 \rightarrow A_{\text{Switch}}) \times (1..3 \rightarrow (\text{GEARS} \rightarrow \text{BOOL})) \times (1..3 \rightarrow (\text{GEARS} \rightarrow \text{BOOL})) \times (1..3 \rightarrow \text{GEAR\_ABSORBER}) \times (1..3 \rightarrow (\text{DOORS} \rightarrow \text{BOOL})) \times (1..3 \rightarrow (\text{DOORS} \rightarrow \text{BOOL})) \times (1..3 \rightarrow \text{Hout}) \rightarrow \text{BOOL} \\
\text{act36 : gears\_locked\_down\_func} : \in (1..3 \rightarrow \text{POSITIONS}) \times (1..3 \rightarrow A_{\text{Switch}}) \times (1..3 \rightarrow (\text{GEARS} \rightarrow \text{BOOL})) \times (1..3 \rightarrow (\text{GEARS} \rightarrow \text{BOOL})) \times (1..3 \rightarrow \text{GEAR\_ABSORBER}) \times (1..3 \rightarrow (\text{DOORS} \rightarrow \text{BOOL})) \times (1..3 \rightarrow (\text{DOORS} \rightarrow \text{BOOL})) \times (1..3 \rightarrow \text{Hout}) \rightarrow \text{BOOL} \\
\text{act37 : gears\_man\_func} : \in (1..3 \rightarrow \text{POSITIONS}) \times (1..3 \rightarrow A_{\text{Switch}}) \times (1..3 \rightarrow (\text{GEARS} \rightarrow \text{BOOL})) \times (1..3 \rightarrow (\text{GEARS} \rightarrow \text{BOOL})) \times (1..3 \rightarrow \text{GEAR\_ABSORBER}) \times (1..3 \rightarrow (\text{DOORS} \rightarrow \text{BOOL})) \times (1..3 \rightarrow (\text{DOORS} \rightarrow \text{BOOL})) \times (1..3 \rightarrow \text{Hout}) \rightarrow \text{BOOL} \\
\text{act38 : anomaly\_func} : \in (1..3 \rightarrow \text{POSITIONS}) \times (1..3 \rightarrow A_{\text{Switch}}) \times (1..3 \rightarrow (\text{GEARS} \rightarrow \text{BOOL})) \times (1..3 \rightarrow (\text{GEARS} \rightarrow \text{BOOL})) \times (1..3 \rightarrow \text{GEAR\_ABSORBER}) \times (1..3 \rightarrow (\text{DOORS} \rightarrow \text{BOOL})) \times (1..3 \rightarrow (\text{DOORS} \rightarrow \text{BOOL})) \times (1..3 \rightarrow \text{Hout}) \rightarrow \text{BOOL} \\
\text{act39 : A\_Switch\_Out} := \text{FALSE} \\
\text{act40 : close\_EV\_Hout} := 0 \\
\text{act41 : retract\_EV\_Hout} := 0 \\
\text{act42 : extend\_EV\_Hout} := 0 \\
\text{act43 : open\_EV\_Hout} := 0 \\
\text{act44 : general\_EV\_Hout} := 0 \\
\end{array}
\]

\textbf{Event} opening\_doors\_DOWN \equiv \\
\textbf{extends} opening\_doors\_DOWN \\
\textbf{when} \\
\text{grd1} : dstate[\text{DOORS}] = \{\text{CLOSED}\} \\
\text{grd5} : lstate[\text{DOORS}] = \{\text{UNLOCKED}\} \\
\text{grd7} : phase = \text{movingdown} \\
\text{grd8} : p = R \\
\text{grd9} : l = R \\
\text{grd10} : door\_open = \{a \mapsto b | a \in 1..3 \land b \in \text{DOORS} \rightarrow \{\text{FALSE}\}\} \\
\text{grd11} : door\_closed = \{a \mapsto b | a \in 1..3 \land b \in \text{DOORS} \rightarrow \{\text{FALSE}\}\} \\
\text{grd12} : \forall x : x \in 1..3 \Rightarrow handle(x) = \text{button} \\
\textbf{then} \\
\text{act1} : dstate := \{a \mapsto b | a \in \text{DOORS} \land b = \text{OPEN}\} \\
\text{act2} : p := E \\
\text{act3} : door\_open : \in 1..3 \rightarrow (\text{DOORS} \rightarrow \{\text{TRUE}\}) \\
\textbf{end} \\
\textbf{Event} opening\_doors\_UP \equiv \\
\textbf{extends} opening\_doors\_UP \\
\textbf{when} \\
\text{grd1} : dstate[\text{DOORS}] = \{\text{CLOSED}\} \\
\text{grd4} : lstate[\text{DOORS}] = \{\text{UNLOCKED}\}
grd5 : phase = movingup
grd6 : l = E
grd7 : l = E
grd8 : door_open = \{a \mapsto \{b \in 1..3 \land b \in DOORS \Rightarrow \{False\}\}\}
grd9 : door_closed = \{a \mapsto \{b \in 1..3 \land b \in DOORS \Rightarrow \{False\}\}\}
grd10 : \forall x \cdot x \in 1..3 \Rightarrow handle(x) = button

then

act1 : dstate := \{a \mapsto \{b \in DOORS \land b = OPEN\}\}
act2 : p := R
act3 : door_open \in 1..3 \Rightarrow \{DOORS \Rightarrow \{TRUE\}\}\)

end

Event closing_doors_UP ≡
extends closing_doors_UP
any

f
where

grd1 : dstate[DOORS] = \{OPEN\}
grd3 : f \in DOORS \rightarrow SDOORS
grd4 : \forall e \cdot e \in DOORS \Rightarrow f(e) = CLOSED
grd5 : phase = movingup
grd6 : p = R
grd7 : gstate[GEARS] = \{RETRACTED\}
grd8 : \forall x \cdot x \in 1..3 \Rightarrow handle(x) = button

then

act1 : dstate := f

end

Event closing_doors_DOWN ≡
extends closing_doors_DOWN
any

f
where

grd1 : dstate[DOORS] = \{OPEN\}
grd3 : f \in DOORS \rightarrow SDOORS
grd4 : \forall e \cdot e \in DOORS \Rightarrow f(e) = CLOSED
grd5 : phase = movingdown
grd6 : p = E
grd7 : gstate[GEARS] = \{EXTENDED\}
grd8 : \forall x \cdot x \in 1..3 \Rightarrow handle(x) = button

then

act1 : dstate := f

end

Event unlocking_UP ≡
extends unlocking_UP
when

grd3 : lstate[DOORS] = \{LOCKED\}
grd4 : phase = movingup

54
\text{grd5} : l = E \\
\text{grd6} : p = E \\
\text{grd7} : i = E \\
\text{grd8} : \text{door\_open} = \{a \mapsto b \mid a \in 1..3 \land b \in \text{DOORS} \rightarrow \{\text{FALSE}\}\} \\
\text{grd9} : \text{door\_closed} = \{a \mapsto b \mid a \in 1..3 \land b \in \text{DOORS} \rightarrow \{\text{TRUE}\}\} \\
\text{grd10} : \forall x \cdot x \in 1..3 \Rightarrow \text{handle}(x) = \text{button} \\
\text{then} \\
\text{act1} : \text{lstate} := \{a \mapsto b \mid a \in \text{DOORS} \land b = \text{UNLOCKED}\} \\
\text{act2} : \text{door\_closed} : \in 1..3 \rightarrow (\text{DOORS} \rightarrow \{\text{FALSE}\}) \\
\text{end} \\
\text{Event locking\_UP} \triangleq \text{extends locking\_UP} \\
\text{when} \\
\text{grd3} : \text{dstate}[\text{DOORS}] = \{\text{CLOSED}\} \\
\text{grd4} : \text{phase} = \text{movingup} \\
\text{grd5} : \text{lstate}[\text{DOORS}] = \{\text{UNLOCKED}\} \\
\text{grd6} : p = R \\
\text{grd7} : i = E \\
\text{grd9} : \text{door\_open} = \{a \mapsto b \mid a \in 1..3 \land b \in \text{DOORS} \rightarrow \{\text{FALSE}\}\} \\
\text{grd10} : \text{door\_closed} = \{a \mapsto b \mid a \in 1..3 \land b \in \text{DOORS} \rightarrow \{\text{FALSE}\}\} \\
\text{grd11} : \forall x \cdot x \in 1..3 \Rightarrow \text{handle}(x) = \text{button} \\
\text{then} \\
\text{act1} : \text{lstate} := \{a \mapsto b \mid a \in \text{DOORS} \land b = \text{LOCKED}\} \\
\text{act3} : \text{phase} := \text{haltup} \\
\text{act4} : l := R \\
\text{added by D Mery} \\
\text{act44} : \text{door\_closed} : \in 1..3 \rightarrow (\text{DOORS} \rightarrow \{\text{TRUE}\}) \\
\text{end} \\
\text{Event unlocking\_DOWN} \triangleq \text{extends unlocking\_DOWN} \\
\text{when} \\
\text{grd3} : \text{lstate}[\text{DOORS}] = \{\text{LOCKED}\} \\
\text{grd4} : \text{phase} = \text{movingdown} \\
\text{grd5} : l = R \\
\text{grd6} : p = R \\
\text{grd7} : i = R \\
\text{grd8} : \text{door\_open} = \{a \mapsto b \mid a \in 1..3 \land b \in \text{DOORS} \rightarrow \{\text{FALSE}\}\} \\
\text{grd9} : \text{door\_closed} = \{a \mapsto b \mid a \in 1..3 \land b \in \text{DOORS} \rightarrow \{\text{TRUE}\}\} \\
\text{grd10} : \forall x \cdot x \in 1..3 \Rightarrow \text{handle}(x) = \text{button} \\
\text{then} \\
\text{act1} : \text{lstate} := \{a \mapsto b \mid a \in \text{DOORS} \land b = \text{UNLOCKED}\} \\
\text{act2} : \text{door\_closed} : \in 1..3 \rightarrow (\text{DOORS} \rightarrow \{\text{FALSE}\}) \\
\text{end} \\
\text{Event locking\_DOWN} \triangleq \text{extends locking\_DOWN} \\
\text{when} \\
\text{grd1} : \text{dstate}[\text{DOORS}] = \{\text{CLOSED}\}
grd2 : phase = movingdown
grd3 : lstate[DOORS] = \{UNLOCKED\}
grd4 : p = E
grd5 : l = R
grd7 : door_open = \{a \mapsto b | a \in 1..3 \land b \in DOORS \rightarrow \{FALSE\}\}
grd8 : door_closed = \{a \mapsto b | a \in 1..3 \land b \in DOORS \rightarrow \{FALSE\}\}
grd9 : \forall x \cdot x \in 1..3 \Rightarrow handle(x) = button

act1 : lstate := \{a \mapsto b | a \in DOORS \land b = LOCKED\}
act3 : phase := haltdown
act4 : l := E
act5 : door_closed := 1..3 \rightarrow (DOORS \rightarrow \{TRUE\})

end

Event PD1 \equiv
extends PD1
when

grd1 : button = UP
grd2 : phase = halo\text{\footnotesize{\textasciicircum}}
then

act1 : phase := movingdown
act2 : button := DOWN
act3 : l := R
act4 : p := R
act5 : i := R

end

Event PU1 \equiv
extends PU1
when

grd1 : button = DOWN
grd2 : phase = halo\text{\footnotesize{\textasciicircum}}
then

act1 : phase := movingup
act2 : button := UP
act3 : l := E
act4 : p := E
act5 : i := E

end

Event PU2 \equiv
extends PU2
when

grd1 : l = R
grd2 : p = R
grd3 : phase = movingdown
grd4 : button = DOWN
grd5 : i = R
\text{grd6} : \text{lstate}[DOORS] = \{\text{LOCKED}\}
\text{grd7} : \text{door\_open} = \{a \mapsto b | a \in 1..3 \land b \in DOORS \Rightarrow \{\text{FALSE}\}\}
\text{grd8} : \text{door\_closed} = \{a \mapsto b | a \in 1..3 \land b \in DOORS \Rightarrow \{\text{TRUE}\}\}
\text{grd9} : \forall x \cdot x \in 1..3 \Rightarrow \text{handle}(x) = UP

\text{end}

\text{Event} \; \text{CompletePU2} \cong
\text{extends} \; \text{CompletePU2}

\text{when}
\begin{align*}
\text{grd1} : \text{phase} &= \text{movingup} \\
\text{grd2} : \text{button} &= UP \\
\text{grd3} : \text{l} &= E \\
\text{grd4} : \text{p} &= E \\
\text{grd5} : \text{i} &= R
\end{align*}

\text{end}

\text{Event} \; \text{PU3} \cong
\text{extends} \; \text{PU3}

\text{when}
\begin{align*}
\text{grd1} : \text{dstate}[DOORS] &= \{\text{CLOSED}\} \\
\text{grd2} : \text{lstate}[DOORS] &= \{\text{UNLOCKED}\} \\
\text{grd3} : \text{phase} &= \text{movingdown} \\
\text{grd4} : \text{p} &= R \\
\text{grd5} : \text{l} &= R \\
\text{grd6} : \text{button} &= DOWN \\
\text{grd7} : \text{door\_open} = \{a \mapsto b | a \in 1..3 \land b \in DOORS \Rightarrow \{\text{FALSE}\}\} \\
\text{grd8} : \text{door\_closed} = \{a \mapsto b | a \in 1..3 \land b \in DOORS \Rightarrow \{\text{FALSE}\}\} \\
\text{grd9} : \forall x \cdot x \in 1..3 \Rightarrow \text{handle}(x) = UP
\end{align*}

\text{then}
\begin{align*}
\text{act1} : \text{phase} &= \text{movingup} \\
\text{act2} : \text{p} &= R \\
\text{act3} : \text{l} &= E \\
\text{act4} : \text{button} &= UP
\end{align*}

\text{end}

\text{Event} \; \text{PU4} \cong
\text{extends} \; \text{PU4}

\text{when}
\begin{align*}
\text{grd1} : \text{dstate}[DOORS] &= \{\text{OPEN}\} \\
\text{grd2} : \text{phase} &= \text{movingdown} \\
\text{grd3} : \text{p} &= E \\
\text{grd4} : \text{button} &= DOWN
\end{align*}
grd7 : $\forall x \cdot x \in 1..3 \Rightarrow handle(x) = UP$

act1 : phase := movingup
act2 : $p := R$
act3 : $button := UP$
act4 : $i := E$
act5 : $l := E$

end

Event $PU5 \sqsubseteq$
extends $PU5$
when

grd1 : dstate[DOORS] = \{CLOSED\}
grd2 : phase = movingdown
grd3 : $p = E$
grd4 : $button = DOWN$
grd5 : lstate[DOORS] = \{UNLOCKED\}
grd6 : door_open = \{a \mapsto b | a \in 1..3 \land b \in DOORS \rightarrow \{FALSE\}\}$
grd7 : door_closed = \{a \mapsto b | a \in 1..3 \land b \in DOORS \rightarrow \{FALSE\}\}$
grd8 : $\forall x \cdot x \in 1..3 \Rightarrow handle(x) = UP$

then

act1 : phase := movingup
act3 : $button := UP$
act4 : $i := E$
act5 : $l := E$

end

Event $PD2 \sqsubseteq$
extends $PD2$
when

grd1 : $l = E$
grd2 : $p = E$
grd3 : phase = movingup
grd4 : $i := E$
grd5 : lstate[DOORS] = \{LOCKED\}
grd6 : $\forall x \cdot x \in 1..3 \Rightarrow handle(x) = DOWN$

then

act1 : phase := movingdown
act2 : $button := DOWN$
act3 : $l := R$
act4 : $p := R$
act5 : $i := E$

end

Event $CompletePD2 \sqsubseteq$
extends $CompletePD2$
when

grd1 : phase = movingdown
grd2 : $button = DOWN$
grd3 : $l = R$
grd4 : \( p = R \)

grd5 : \( i = E \)

\textbf{then}

\textbf{act1} : \textit{phase} := \textit{haltdown}

\textbf{end}

\textbf{Event} \( PD3 \equiv \)

\textbf{extends} \( PD3 \)

\textbf{when}

\begin{align*}
grd1 : & \text{dstate}[DOORS] = \{\text{CLOSED}\} \\
grd2 : & \text{lstate}[DOORS] = \{\text{UNLOCKED}\} \\
grd3 : & \text{phase} = \text{movingup} \\
grd4 : & p = E \\
grd5 : & l = E \\
grd6 : & \text{button} = \text{UP} \\
grd7 : & \text{door}\_\text{open} = \{a \mapsto b | a \in 1..3 \land b \in \text{DOORS} \rightarrow \{\text{FALSE}\}\} \\
grd8 : & \text{door}\_\text{closed} = \{a \mapsto b | a \in 1..3 \land b \in \text{DOORS} \rightarrow \{\text{FALSE}\}\} \\
grd9 : & \forall x \cdot x \in 1..3 \Rightarrow \text{handle}(x) = \text{DOWN}
\end{align*}

\textbf{then}

\begin{align*}
\text{act1} : & \text{phase} := \text{movingdown} \\
\text{act2} : & p := E \\
\text{act3} : & l := R \\
\text{act4} : & \text{button} := \text{DOWN}
\end{align*}

\textbf{end}

\textbf{Event} \( PD4 \equiv \)

\textbf{extends} \( PD4 \)

\textbf{when}

\begin{align*}
grd1 : & \text{dstate}[DOORS] = \{\text{OPEN}\} \\
grd2 : & \text{phase} = \text{movingup} \\
grd3 : & p = R \\
grd4 : & \text{button} = \text{UP} \\
grd5 : & \forall x \cdot x \in 1..3 \Rightarrow \text{handle}(x) = \text{DOWN}
\end{align*}

\textbf{then}

\begin{align*}
\text{act1} : & \text{phase} := \text{movingdown} \\
\text{act2} : & p := E \\
\text{act3} : & \text{button} := \text{DOWN} \\
\text{act4} : & i := R \\
\text{act5} : & l := R
\end{align*}

\textbf{end}

\textbf{Event} \( PD5 \equiv \)

\textbf{extends} \( PD5 \)

\textbf{when}

\begin{align*}
grd1 : & \text{dstate}[DOORS] = \{\text{CLOSED}\} \\
grd2 : & \text{phase} = \text{movingup} \\
grd3 : & p = R \\
grd4 : & \text{button} = \text{UP} \\
grd5 : & \text{lstate}[DOORS] = \{\text{UNLOCKED}\} \\
grd6 : & \text{door}\_\text{open} = \{a \mapsto b | a \in 1..3 \land b \in \text{DOORS} \rightarrow \{\text{FALSE}\}\}
\end{align*}
grd7 : \(\text{door\_closed} = \{a \mapsto b | a \in 1..3 \land b \in DOORS \rightarrow \{\text{FALSE}\}\}\)
grd8 : \(\forall x \cdot x \in 1..3 \Rightarrow \text{handle}(x) = \text{DOWN}\)

then

act1 : \(\text{phase} := \text{moving\_down}\)
act2 : \(\text{button} := \text{DOWN}\)
act3 : \(i := R\)
act4 : \(l := R\)

end

Event retracting\_gears \(\triangleq\) extends retracting\_gears

when

grd1 : \(\text{dstate}[\text{DOORS}] = \{\text{OPEN}\}\)
grd2 : \(\text{gstate}[\text{GEARS}] = \{\text{EXTENDED}\}\)
grd3 : \(p = E\)
grd6 : \(\text{gear\_extended} = \{a \mapsto b | a \in 1..3 \land b \in \text{GEARS} \rightarrow \{\text{TRUE}\}\}\)
grd7 : \(\text{gear\_retracted} = \{a \mapsto b | a \in 1..3 \land b \in \text{GEARS} \rightarrow \{\text{FALSE}\}\}\)
grd8 : \(\text{gear\_shock\_absorber} = \{a \mapsto b | a \in 1..3 \land b = \text{ground}\}\)
grd9 : \(\forall x \cdot x \in 1..3 \Rightarrow \text{handle}(x) = \text{button}\)

then

act1 : \(\text{gstate} := \{a \mapsto b | a \in \text{GEARS} \land b = \text{RETRACTING}\}\)
act2 : \(\text{gear\_retracted} \in 1..3 \Rightarrow (\text{GEARS} \rightarrow \{\text{FALSE}\})\)
act3 : \(\text{gear\_shock\_absorber} := \{a \mapsto b | a \in 1..3 \land b = \text{flight}\}\)

end

Event retraction \(\triangleq\) extends retraction

when

grd1 : \(\text{dstate}[\text{DOORS}] = \{\text{OPEN}\}\)
grd2 : \(\text{gstate}[\text{GEARS}] = \{\text{RETRACTING}\}\)
grd4 : \(\text{gear\_extended} = \{a \mapsto b | a \in 1..3 \land b \in \text{GEARS} \rightarrow \{\text{FALSE}\}\}\)
grd5 : \(\text{gear\_retracted} = \{a \mapsto b | a \in 1..3 \land b \in \text{GEARS} \rightarrow \{\text{FALSE}\}\}\)
grd6 : \(\text{gear\_shock\_absorber} = \{a \mapsto b | a \in 1..3 \land b = \text{flight}\}\)
grd7 : \(\forall x \cdot x \in 1..3 \Rightarrow \text{handle}(x) = \text{button}\)

then

act1 : \(\text{gstate} := \{a \mapsto b | a \in \text{GEARS} \land b = \text{RETRACTED}\}\)
act2 : \(\text{gear\_retracted} \in 1..3 \Rightarrow (\text{GEARS} \rightarrow \{\text{TRUE}\})\)

end

Event extending\_gears \(\triangleq\) extends extending\_gears

when

grd1 : \(\text{dstate}[\text{DOORS}] = \{\text{OPEN}\}\)
grd2 : \(\text{gstate}[\text{GEARS}] = \{\text{RETRACTED}\}\)
grd3 : \(p = E\)
grd5 : \(\text{gear\_retracted} = \{a \mapsto b | a \in 1..3 \land b \in \text{GEARS} \rightarrow \{\text{TRUE}\}\}\)
grd6 : \(\text{gear\_extended} = \{a \mapsto b | a \in 1..3 \land b \in \text{GEARS} \rightarrow \{\text{FALSE}\}\}\)
grd7 : \(\text{gear\_shock\_absorber} = \{a \mapsto b | a \in 1..3 \land b = \text{flight}\}\)
grd8 : \(\forall x \cdot x \in 1..3 \Rightarrow \text{handle}(x) = \text{button}\)
then
\[ \text{act1 : } gstate := \{ a \mapsto b | a \in \text{GEARS} \land b = \text{EXTENDING} \} \]
\[ \text{act2 : } \text{gear_retracted} : \in 1..3 \rightarrow (\text{GEARS} \rightarrow \{\text{FALSE}\}) \]
end

Event extension \(\triangleq\)
extends extension
when
\[ \text{grd1 : } dstate[\text{DOORS}] = \{\text{OPEN}\} \]
\[ \text{grd2 : } gstate[\text{GEARS}] = \{\text{EXTENDING}\} \]
\[ \text{grd4 : } \text{gear_retracted} = \{ a \mapsto b | a \in 1..3 \land b \in \text{GEARS} \rightarrow \{\text{FALSE}\} \} \]
\[ \text{grd5 : } \text{gear_extended} = \{ a \mapsto b | a \in 1..3 \land b \in \text{GEARS} \rightarrow \{\text{FALSE}\} \} \]
\[ \text{grd6 : } \text{gear_shock_absorber} = \{ a \mapsto b | a \in 1..3 \land b = \text{flight}\} \]
\[ \text{grd7 : } \forall x \cdot x \in 1..3 \Rightarrow \text{handle}(x) = \text{button} \]
then
\[ \text{act1 : } gstate := \{ a \mapsto b | a \in \text{GEARS} \land b = \text{EXTENDED} \} \]
\[ \text{act2 : } \text{gear_extended} : \in 1..3 \rightarrow (\text{GEARS} \rightarrow \{\text{TRUE}\}) \]
\[ \text{act3 : } \text{gear_shock_absorber} := \{ a \mapsto b | a \in 1..3 \land b = \text{ground}\} \]
end

Event HPD1 \(\triangleq\)
extends HPD1
when
\[ \text{grd3 : } \forall x \cdot x \in 1..3 \Rightarrow \text{handle}(x) = \text{UP} \]
then
\[ \text{act2 : } \text{handle} : \in 1..3 \rightarrow \{\text{DOWN}\} \]
end

Event HPU1 \(\triangleq\)
extends HPU1
when
\[ \text{grd3 : } \forall x \cdot x \in 1..3 \Rightarrow \text{handle}(x) = \text{DOWN} \]
then
\[ \text{act2 : } \text{handle} : \in 1..3 \rightarrow \{\text{UP}\} \]
end

Event Analogical_switch_closed \(\triangleq\)
extends Analogical_switch_closed
any
in in port
where
\[ \text{grd1 : } in = \text{general}_E V \]
\[ \text{grd2 : } \forall x \cdot x \in 1..3 \Rightarrow (\text{handle}(x) = \text{UP} \lor \text{handle}(x) = \text{DOWN}) \]
then
\[ \text{act3 : } \text{analogical_switch} : \in 1..3 \rightarrow \{\text{closed}\} \]
\[ \text{act4 : } A_{\text{Switch}_\text{Out}} := \text{TRUE} \]
end

Event Analogical_switch_open \(\triangleq\)
extends Analogical_switch_open
any
in in port
where
grd1 : in = general_EV
grd2 : ∀x : x ∈ I..3 ⇒ (handle(x) = UP ∨ handle(x) = DOWN)
then
act3 : analogical_switch :∈ 1..3 → {open}
act4 : A_Switch_Out := FALSE
end
Event Circuit_pressurized_OK \overset{\cdot}{=}
refines Circuit_pressurized
when
grd1 : general_EV.Hout = Hin
then
act9 : circuit_pressurized :∈ 1..3 → {TRUE}
end
Event Circuit_pressurized_notOK \overset{\cdot}{=}
refines Circuit_pressurized
when
grd1 : general_EV.Hout = 0
then
act9 : circuit_pressurized :∈ 1..3 → {FALSE}
end
Event Computing_Module_1.2 \overset{\cdot}{=}
extends Computing_Module_1.2
begin
act1 : general_EV := general_EV.func(handle → analogical_switch →
gear_extended → gear_retracted → gear_shock_absorber → door_open →
door_closed → circuit_pressurized)
act2 : close_EV := close_EV.func(handle → analogical_switch →
gear_extended → gear_retracted → gear_shock_absorber → door_open →
door_closed → circuit_pressurized)
act3 : retract_EV := retract_EV.func(handle → analogical_switch →
gear_extended → gear_retracted → gear_shock_absorber → door_open →
door_closed → circuit_pressurized)
act4 : extend_EV := extend_EV.func(handle → analogical_switch →
gear_extended → gear_retracted → gear_shock_absorber → door_open →
door_closed → circuit_pressurized)
act5 : open_EV := open_EV.func(handle → analogical_switch →
gear_extended → gear_retracted → gear_shock_absorber → door_open →
door_closed → circuit_pressurized)
act6 : gears_locked_down := gears_locked_down.func(handle →
analogical_switch → gear_extended → gear_retracted →
gear_shock_absorber → door_open → door_closed → circuit_pressurized)
act7 : gears_man := gears_man.func(handle → analogical_switch →
gear_extended → gear_retracted → gear_shock_absorber → door_open →
door_closed → circuit_pressurized)
act8 : anomaly := anomaly_func(handle ↦ analogical_switch ↦
gear_extended ↦ gear_retracted ↦ gear_shock_absorber ↦ door_open ↦
door_closed ↦ circuit_pressurized)
end

Event Update_Hout ≜ Assign the value of Hout
begin
  act1 : general_EV_Hout : |((general_EV = TRUE ∧ general_EV_Hout' = Hin) ∨ (general_EV = FALSE ∧ general_EV_Hout' = 0))
     ∧ (A_Switch_Out = TRUE ∧ general_EV_Hout' = Hin) ∨
     (A_Switch_Out = FALSE ∧ general_EV_Hout' = 0))
     pass the current value of hydraulic input port (Hin) to hydraulic output port (Hout)
  act2 : close_EV_Hout : |((close_EV = TRUE ∧ close_EV_Hout' = Hin) ∨ (close_EV = FALSE ∧ close_EV_Hout' = 0))
  act3 : open_EV_Hout : |((open_EV = TRUE ∧ open_EV_Hout' = Hin) ∨ (open_EV = FALSE ∧ open_EV_Hout' = 0))
  act4 : extend_EV_Hout : |((extend_EV = TRUE ∧ extend_EV_Hout' = Hin) ∨ (extend_EV = FALSE ∧ extend_EV_Hout' = 0))
  act5 : retract_EV_Hout : |((retract_EV = TRUE ∧ retract_EV_Hout' = Hin) ∨ (retract_EV = FALSE ∧ retract_EV_Hout' = 0))
end

Event Failure_Detection ≜ extends Failure_Detection
begin
  act1 : anomaly := TRUE
end

END

H M6

An Event-B Specification of M6
Creation Date: 27Jan2014 @ 10:44:59 AM

MACHINE M6
Integration of Cylinder behavior according to the Electro-valves circuit
Strengthening guards of opening and closing doors and gears using cylinders
sensors, and hydraulic pressure.

REFINES M5
SEES C1

VARIABLES
  dstate
  lstate
  phase
  button
  p
  l

63
Invariants

\[ \text{inv1} : SDCylinder \in DOORS \times \{DCY F, DCY R, DCY L\} \rightarrow S_{CYLINDER} \]
\[ \text{inv2} : SGCylinder \in GEARs \times \{GCY F, GCY R, GCY L\} \rightarrow S_{CYLINDER} \]
\[ \text{inv17} : \text{state} \in SPHASES \]
\[ \text{inv4} : SDCylinder = \{a \mapsto b | a \in DOORS \times CYLINDER \land b = \text{STOP} \} \land \\
\text{dstate}^{-1}[\{\text{CLOSED}\}] \neq \emptyset \Rightarrow \text{dstate}^{-1}[\{\text{CLOSED}\}] = DOORS \]
\textbf{EVENTS}

\textbf{Initialisation}

\textit{extended}

\textbf{begin}

act1 : button := DOWN
act2 : phase := haltdown
act3 : dstate : ([dstate' ∈ DOORS → SDOORS ∧ dstate' = {a → b|a ∈ DOORS ∧ b = CLOSED})

missing elements of the invariant
act4 : lstate := {a → b|a ∈ DOORS ∧ b = LOCKED}
act5 : p := R
act6 : l := R
act7 : i := R

act8 : gstate : ([gstate' ∈ GEARS → SGEARS ∧ gstate' = {a → b|a ∈ GEARS ∧ b = EXTENDED})
act14 : handle : ∈ 1..3 → \{DOWN\}
act15 : analogical_switch : ∈ 1..3 → \{open\}
act16 : gear_extended : ∈ 1..3 → \{GEARS → \{TRUE\}\}
act17 : gear_retracted : ∈ 1..3 → \{GEARS → \{FALSE\}\}
act18 : gear_shock_absorber : ∈ 1..3 → \{ground\}
act19 : door_closed : ∈ 1..3 → \{DOORS → \{TRUE\}\}
act20 : door_open : ∈ 1..3 → \{DOORS → \{FALSE\}\}
act21 : circuit_pressureized : ∈ 1..3 → \{FALSE\}
act22 : general_EV := FALSE
act23 : close_EV := TRUE
act24 : retract_EV := FALSE
act25 : extend_EV := TRUE
act27 : open_EV := FALSE
act28 : gears_locked_down := TRUE
act29 : gears_man := FALSE
act30 : anomaly := FALSE

65
act31 : general_EV_func ≜ (1..3 → POSITIONS) × (1..3 → A_Switch) × (1..3 → (GEARS → BOOL)) × (1..3 → (GEARS → BOOL)) × (1..3 → GEAR_ABSORBER) × (1..3 → (DOORS → BOOL)) × (1..3 → (DOORS → BOOL)) × (1..3 → BOOL) → BOOL
act32 : close_EV_func ≜ (1..3 → POSITIONS) × (1..3 → A_Switch) × (1..3 → (GEARS → BOOL)) × (1..3 → (GEARS → BOOL)) × (1..3 → GEAR_ABSORBER) × (1..3 → (DOORS → BOOL)) × (1..3 → (DOORS → BOOL)) × (1..3 → BOOL) → BOOL
act33 : retract_EV_func ≜ (1..3 → POSITIONS) × (1..3 → A_Switch) × (1..3 → (GEARS → BOOL)) × (1..3 → (GEARS → BOOL)) × (1..3 → GEAR_ABSORBER) × (1..3 → (DOORS → BOOL)) × (1..3 → (DOORS → BOOL)) × (1..3 → BOOL) → BOOL
act34 : extend_EV_func ≜ (1..3 → POSITIONS) × (1..3 → A_Switch) × (1..3 → (GEARS → BOOL)) × (1..3 → (GEARS → BOOL)) × (1..3 → GEAR_ABSORBER) × (1..3 → (DOORS → BOOL)) × (1..3 → (DOORS → BOOL)) × (1..3 → BOOL) → BOOL
act35 : open_EV_func ≜ (1..3 → POSITIONS) × (1..3 → A_Switch) × (1..3 → (GEARS → BOOL)) × (1..3 → (GEARS → BOOL)) × (1..3 → GEAR_ABSORBER) × (1..3 → (DOORS → BOOL)) × (1..3 → (DOORS → BOOL)) × (1..3 → BOOL) → BOOL
act36 : gears_locked_down_func ≜ (1..3 → POSITIONS) × (1..3 → A_Switch) × (1..3 → (GEARS → BOOL)) × (1..3 → (GEARS → BOOL)) × (1..3 → GEAR_ABSORBER) × (1..3 → (DOORS → BOOL)) × (1..3 → (DOORS → BOOL)) × (1..3 → BOOL) → BOOL
act37 : gears_man_func ≜ (1..3 → POSITIONS) × (1..3 → A_Switch) × (1..3 → (GEARS → BOOL)) × (1..3 → (GEARS → BOOL)) × (1..3 → GEAR_ABSORBER) × (1..3 → (DOORS → BOOL)) × (1..3 → (DOORS → BOOL)) × (1..3 → BOOL) → BOOL
act38 : anomaly_func ≜ (1..3 → POSITIONS) × (1..3 → A_Switch) × (1..3 → (GEARS → BOOL)) × (1..3 → (GEARS → BOOL)) × (1..3 → GEAR_ABSORBER) × (1..3 → (DOORS → BOOL)) × (1..3 → (DOORS → BOOL)) × (1..3 → BOOL) → BOOL
act39 : A_Switch_Out := FALSE
act40 : close_EV_Hout := 0
act41 : retract_EV_Hout := 0
act42 : extend_EV_Hout := 0
act43 : open_EV_Hout := 0
act44 : general_EV_Hout := 0
act45 : SDCylinder ∈ DOORS × {DCY_F, DCY_R, DCY_L} → {STOP}
act46 : SGCylinder ∈ GEAR × {GCY_F, GCY_R, GCY_L} → {STOP}
act46 : state := computing

end
Event opening_doors_DOWN ≜
extends opening_doors_DOWN
when
grd1 : dstate[DOORS] = {CLOSED}
grd5 : lstate[DOORS] = {UNLOCKED}
grd7 : phase = movingdown
grd8 : p = R
grd9 : l = R

66
grd10 : $\text{door\_open} = \{a \mapsto b | a \in 1..3 \land b \in \text{DOORS} \rightarrow \{\text{FALSE}\}\}$
grd11 : $\text{door\_closed} = \{a \mapsto b | a \in 1..3 \land b \in \text{DOORS} \rightarrow \{\text{FALSE}\}\}$
grd12 : $\forall x \in 1..3 \Rightarrow \text{handle}(x) = \text{button}$
grd3 : $\text{SDCylinder} = \{a \mapsto b | a \in \text{DOORS} \times \text{CYLINDER} \land b = \text{MOVING}\}$

then

act1 : $\text{dstate} := \{a \mapsto b | a \in \text{DOORS} \land b = \text{OPEN}\}$
act2 : $p := E$
act3 : $\text{door\_open} \in 1..3 \rightarrow (\text{DOORS} \rightarrow \{\text{TRUE}\})$

end

Event $\text{opening\_doors\_UP}$
extends $\text{opening\_doors\_UP}$

when

act1 : $\text{dstate}[\text{DOORS}] = \{\text{CLOSED}\}$
act4 : $\text{lstate}[\text{DOORS}] = \{\text{UNLOCKED}\}$
act5 : $\text{phase} = \text{movingup}$
act6 : $p = E$
act7 : $l = E$
act8 : $\text{door\_open} = \{a \mapsto b | a \in 1..3 \land b \in \text{DOORS} \rightarrow \{\text{FALSE}\}\}$
act9 : $\text{door\_closed} = \{a \mapsto b | a \in 1..3 \land b \in \text{DOORS} \rightarrow \{\text{FALSE}\}\}$
act10 : $\forall x \in 1..3 \Rightarrow \text{handle}(x) = \text{button}$
grd3 : $\text{SDCylinder} = \{a \mapsto b | a \in \text{DOORS} \times \text{CYLINDER} \land b = \text{MOVING}\}$

then

act1 : $\text{dstate} := \{a \mapsto b | a \in \text{DOORS} \land b = \text{OPEN}\}$
act2 : $p := R$
act3 : $\text{door\_open} \in 1..3 \rightarrow (\text{DOORS} \rightarrow \{\text{TRUE}\})$

end

Event $\text{closing\_doors\_UP}$
extends $\text{closing\_doors\_UP}$
an

where

act1 : $\text{dstate}[\text{DOORS}] = \{\text{OPEN}\}$
grd3 : $f \in \text{DOORS} \Rightarrow \text{SDOORS}$
grd4 : $\forall e \in \text{DOORS} \Rightarrow f(e) = \text{CLOSED}$
grd5 : $\text{phase} = \text{movingup}$
grd6 : $p = R$
grd7 : $\text{gstate}[\text{GEARS}] = \{\text{RETRACTED}\}$

then

act1 : $\text{dstate} := f$

end

Event $\text{closing\_doors\_DOWN}$
extends $\text{closing\_doors\_DOWN}$
an

where
grd1 : dstate[DOORS] = \{OPEN\}
grd3 : f ∈ DOORS \rightarrow SDOORS
grd4 : ∀e ∈ DOORS ⇒ f(e) = CLOSED
grd5 : phase = movingdown
grd6 : p = E
grd7 : gstate[GEARS] = \{EXTENDED\}
grd8 : ∀x \cdot x ∈ 1..3 ⇒ handle(x) = button

then

act1 : dstate := f

end

Event \textit{unlocking}\_\textit{UP} \equiv
extends \textit{unlocking}\_\textit{UP}
when

grd3 : lstate[DOORS] = \{LOCKED\}
grd4 : phase = movingup
grd5 : l = E
grd6 : p = E
grd7 : i = E
grd8 : door\_open = \{a \mapsto b|a \in 1..3 \land b \in DOORS \rightarrow \{FALSE\}\}
grd9 : door\_closed = \{a \mapsto b|a \in 1..3 \land b \in DOORS \rightarrow \{TRUE\}\}
grd10 : ∀x \cdot x ∈ 1..3 ⇒ handle(x) = button

then

act1 : lstate := \{a \mapsto b|a \in DOORS \land b = UNLOCKED\}
act2 : door\_closed := \in 1..3 \rightarrow (DOORS \rightarrow \{FALSE\})

end

Event \textit{locking}\_\textit{UP} \equiv
extends \textit{locking}\_\textit{UP}
when

grd3 : dstate[DOORS] = \{CLOSED\}
grd4 : phase = movingup
grd5 : lstate[DOORS] = \{UNLOCKED\}
grd6 : p = R
grd7 : l = E
grd9 : door\_open = \{a \mapsto b|a \in 1..3 \land b \in DOORS \rightarrow \{FALSE\}\}
grd10 : door\_closed = \{a \mapsto b|a \in 1..3 \land b \in DOORS \rightarrow \{FALSE\}\}
grd11 : ∀x \cdot x ∈ 1..3 ⇒ handle(x) = button
grd8 : SDCylinder = \{a \mapsto b|a ∈ DOORS \times CYLINDER \land b = STOP\}

then

act1 : lstate := \{a \mapsto b|a ∈ DOORS \land b = LOCKED\}
act3 : phase := haltup
act4 : l := R

added by D Mery

act44 : door\_closed := \in 1..3 \rightarrow (DOORS \rightarrow \{TRUE\})

end

Event \textit{unlocking}\_\textit{DOWN} \equiv
extends \textit{unlocking}\_\textit{DOWN}
when

68
\begin{align*}
\text{grd3} & : \text{lstate}[\text{DOORS}] = \{\text{LOCKED}\} \\
\text{grd4} & : \text{phase} = \text{movingdown} \\
\text{grd5} & : \text{l} = \text{R} \\
\text{grd6} & : \text{p} = \text{R} \\
\text{grd7} & : \text{i} = \text{R} \\
\text{grd8} & : \text{door} \text{\_open} = \{a \mapsto b|a \in 1..3 \land b \in \text{DOORS} \rightarrow \{\text{FALSE}\}\} \\
\text{grd9} & : \text{door} \text{\_closed} = \{a \mapsto b|a \in 1..3 \land b \in \text{DOORS} \rightarrow \{\text{TRUE}\}\} \\
\text{grd10} & : \forall x \cdot x \in 1..3 \Rightarrow \text{handle}(x) = \text{button} \\
\text{then} \\
\text{act1} & : \text{lstate} := \{a \mapsto b|a \in \text{DOORS} \land b = \text{UNLOCKED}\} \\
\text{act2} & : \text{door} \text{\_closed} :\in 1..3 \rightarrow (\text{DOORS} \rightarrow \{\text{FALSE}\}) \\
\end{align*}

Event locking\textunderscore DOWN \equiv 
extends locking\_DOWN 
when 

\begin{align*}
\text{grd1} & : \text{dstate}[\text{DOORS}] = \{\text{CLOSED}\} \\
\text{grd2} & : \text{phase} = \text{movingdown} \\
\text{grd3} & : \text{lstate}[\text{DOORS}] = \{\text{UNLOCKED}\} \\
\text{grd4} & : \text{p} = \text{E} \\
\text{grd5} & : \text{l} = \text{R} \\
\text{grd7} & : \text{door} \text{\_open} = \{a \mapsto b|a \in 1..3 \land b \in \text{DOORS} \rightarrow \{\text{FALSE}\}\} \\
\text{grd8} & : \text{door} \text{\_closed} = \{a \mapsto b|a \in 1..3 \land b \in \text{DOORS} \rightarrow \{\text{FALSE}\}\} \\
\text{grd9} & : \forall x \cdot x \in 1..3 \Rightarrow \text{handle}(x) = \text{button} \\
\text{grd6} & : \text{SDCylinder} = \{a \mapsto b|a \in \text{DOORS} \times \text{CYLINDER} \land b = \text{STOP}\} \\
\text{then} \\
\text{act1} & : \text{lstate} := \{a \mapsto b|a \in \text{DOORS} \land b = \text{LOCKED}\} \\
\text{act3} & : \text{phase} := \text{haltdown} \\
\text{act4} & : \text{l} := \text{E} \\
\text{act5} & : \text{door} \text{\_closed} :\in 1..3 \rightarrow (\text{DOORS} \rightarrow \{\text{TRUE}\}) \\
\end{align*}

Event PD1 \equiv 
extends PD1 
when 

\begin{align*}
\text{grd1} & : \text{button} = \text{UP} \\
\text{grd2} & : \text{phase} = \text{haltup} \\
\text{grd3} & : \forall x \cdot x \in 1..3 \Rightarrow \text{handle}(x) = \text{DOWN} \\
\text{then} \\
\text{act1} & : \text{phase} := \text{movingdown} \\
\text{act2} & : \text{button} := \text{DOWN} \\
\text{act3} & : \text{l} := \text{R} \\
\text{act4} & : \text{p} := \text{R} \\
\text{act5} & : \text{i} := \text{R} \\
\end{align*}

Event PU1 \equiv 
extends PU1 
when 

\begin{align*}
\text{grd1} & : \text{button} = \text{DOWN} \\
\end{align*}
$\text{grd2} : \text{phase} = \text{haltdown}$
$\text{grd3} : \forall x \cdot x \in 1..3 \Rightarrow \text{handle}(x) = \text{UP}$

then

\begin{align*}
\text{act1} & : \text{phase} := \text{movingup} \\
\text{act2} & : \text{button} := \text{UP} \\
\text{act3} & : l := E \\
\text{act4} & : p := E \\
\text{act5} & : i := E
\end{align*}

end

\begin{align*}
\text{Event } PU2 \cong \\
\text{extends } PU2
\end{align*}
when

\begin{align*}
\text{grd1} & : l = R \\
\text{grd2} & : p = R \\
\text{grd3} & : \text{phase} = \text{movingdown} \\
\text{grd4} & : \text{button} = \text{DOWN} \\
\text{grd5} & : i = R \\
\text{grd6} & : \text{lstate}[\text{DOORS}] = \{\text{LOCKED}\} \\
\text{grd7} & : \text{door\_open} = \{a \mapsto b | a \in 1..3 \land b \in \text{DOORS} \Rightarrow \{\text{FALSE}\}\} \\
\text{grd8} & : \text{door\_closed} = \{a \mapsto b | a \in 1..3 \land b \in \text{DOORS} \Rightarrow \{\text{TRUE}\}\} \\
\text{grd9} & : \forall x \cdot x \in 1..3 \Rightarrow \text{handle}(x) = \text{UP}
\end{align*}

then

\begin{align*}
\text{act1} & : \text{phase} := \text{movingup} \\
\text{act4} & : \text{button} := \text{UP} \\
\text{act5} & : l := E \\
\text{act6} & : p := E \\
\text{act7} & : i := R
\end{align*}

end

\begin{align*}
\text{Event } \text{CompletePU2} \cong \\
\text{extends } \text{CompletePU2}
\end{align*}
when

\begin{align*}
\text{grd1} & : \text{phase} = \text{movingup} \\
\text{grd2} & : \text{button} = \text{UP} \\
\text{grd3} & : l = E \\
\text{grd4} & : p = E \\
\text{grd5} & : i = R
\end{align*}

then

\begin{align*}
\text{act1} & : \text{phase} := \text{haltup}
\end{align*}

end

\begin{align*}
\text{Event } PU3 \cong \\
\text{extends } PU3
\end{align*}
when

\begin{align*}
\text{grd1} & : \text{dstate}[\text{DOORS}] = \{\text{CLOSED}\} \\
\text{grd2} & : \text{lstate}[\text{DOORS}] = \{\text{UNLOCKED}\} \\
\text{grd3} & : \text{phase} = \text{movingdown} \\
\text{grd4} & : p = R \\
\text{grd5} & : l = R
\end{align*}
\[ \text{grd6} : \text{button} = \text{DOWN} \]
\[ \text{grd7} : \text{door\_open} = \{ a \mapsto b | a \in 1..3 \land b \in \text{DOORS} \rightarrow \{\text{FALSE}\}\} \]
\[ \text{grd8} : \text{door\_closed} = \{ a \mapsto b | a \in 1..3 \land b \in \text{DOORS} \rightarrow \{\text{FALSE}\}\} \]
\[ \text{grd9} : \forall x \cdot x \in 1..3 \Rightarrow \text{handle}(x) = \text{UP} \]

\[ \begin{align*}
\text{act1} & : \text{phase} := \text{movingup} \\
\text{act2} & : p := R \\
\text{act3} & : l := E \\
\text{act4} & : \text{button} := \text{UP} \\
\end{align*} \]

\textbf{Event } \textit{PU4} \equiv \textbf{extends } \textit{PU4} \textbf{ when } \begin{align*}
\text{grd1} & : \text{dstate}[\text{DOORS}] = \{\text{OPEN}\} \\
\text{grd2} & : \text{phase} = \text{movingdown} \\
\text{grd3} & : p = E \\
\text{grd4} & : \text{button} = \text{DOWN} \\
\text{grd7} & : \forall x \cdot x \in 1..3 \Rightarrow \text{handle}(x) = \text{UP} \\
\end{align*} \]

\[ \begin{align*}
\text{act1} & : \text{phase} := \text{movingup} \\
\text{act2} & : p := R \\
\text{act3} & : \text{button} := \text{UP} \\
\text{act4} & : l := E \\
\text{act5} & : i := E \\
\end{align*} \]

\textbf{Event } \textit{PU5} \equiv \textbf{extends } \textit{PU5} \textbf{ when } \begin{align*}
\text{grd1} & : \text{dstate}[\text{DOORS}] = \{\text{CLOSED}\} \\
\text{grd2} & : \text{phase} = \text{movingdown} \\
\text{grd3} & : p = E \\
\text{grd4} & : \text{button} = \text{DOWN} \\
\text{grd5} & : \text{lstate}[\text{DOORS}] = \{\text{UNLOCKED}\} \\
\text{grd6} & : \text{door\_open} = \{ a \mapsto b | a \in 1..3 \land b \in \text{DOORS} \rightarrow \{\text{FALSE}\}\} \\
\text{grd7} & : \text{door\_closed} = \{ a \mapsto b | a \in 1..3 \land b \in \text{DOORS} \rightarrow \{\text{FALSE}\}\} \\
\text{grd8} & : \forall x \cdot x \in 1..3 \Rightarrow \text{handle}(x) = \text{UP} \\
\end{align*} \]

\[ \begin{align*}
\text{act1} & : \text{phase} := \text{movingup} \\
\text{act3} & : \text{button} := \text{UP} \\
\text{act4} & : i := E \\
\text{act5} & : l := E \\
\end{align*} \]

\textbf{Event } \textit{PD2} \equiv \textbf{extends } \textit{PD2} \textbf{ when } \begin{align*}
\text{grd1} & : l = E \\
\text{grd2} & : p = E \\
\end{align*} \]
grd3 : phase = movingup
grd4 : i = E
grd5 : lstate[DOORS] = \{LOCKED\}
grd6 : \forall x \cdot x \in 1..3 \Rightarrow handle(x) = DOWN

then

act1 : phase := movingdown
act2 : button := DOWN
act3 : l := R
act4 : p := R
act5 : i := E

does

Event \textit{CompletePD2} \equiv
extends \textit{CompletePD2}
when

grd1 : phase = movingdown
grd2 : button = DOWN
grd3 : l = R
grd4 : p = R
grd5 : i = E

then

act1 : phase := haltdown

does

Event \textit{PD3} \equiv
extends \textit{PD3}
when

grd1 : dstate[DOORS] = \{CLOSED\}
grd2 : lstate[DOORS] = \{UNLOCKED\}
grd3 : phase = movingup
grd4 : p = E
grd5 : l = E
grd6 : button = UP
grd7 : door_open = \{a \mapsto b | a \in 1..3 \land b \in \text{DOORS} \Rightarrow \{FALSE\}\}
grd8 : door_closed = \{a \mapsto b | a \in 1..3 \land b \in \text{DOORS} \Rightarrow \{FALSE\}\}
grd9 : \forall x \cdot x \in 1..3 \Rightarrow handle(x) = DOWN

then

act1 : phase := movingdown
act2 : p := E
act3 : l := R
act4 : button := DOWN

does

Event \textit{PD4} \equiv
extends \textit{PD4}
when

grd1 : dstate[DOORS] = \{OPEN\}
grd2 : phase = movingup
grd3 : p = R
grd4 : button = UP

72
∀ x · x ∈ 1..3 ⇒ handle(x) = DOWN

then

act1 : phase := movingdown
act2 : p := E
act3 : button := DOWN
act4 : i := R
act5 : l := R

end

Event PD5
extends PD5
when

drd1 : dstate[DOORS] = {CLOSED}
drd2 : phase = movingup
drd3 : p = R
drd4 : button = UP
drd5 : lstate[DOORS] = {UNLOCKED}
drd6 : door_open = {a ↦ b | a ∈ 1..3 ∧ b ∈ DOORS ⇒ {FALSE}}
drd7 : door_closed = {a ↦ b | a ∈ 1..3 ∧ b ∈ DOORS ⇒ {FALSE}}
drd8 : ∀ x ∈ 1..3 ⇒ handle(x) = DOWN

then

act1 : phase := movingdown
act2 : button := DOWN
act3 : i := R
act4 : l := R

end

Event retracting_gears
extends retracting_gears
when

drd1 : dstate[DOORS] = {OPEN}
drd2 : gstate[GEARS] = {EXTENDED}
drd3 : p = R
drd6 : gear_extended = {a ↦ b | a ∈ 1..3 ∧ b ∈ GEARS ⇒ {TRUE}}
drd7 : gear_retracted = {a ↦ b | a ∈ 1..3 ∧ b ∈ GEARS ⇒ {FALSE}}
drd8 : gear_shock_absorber = {a ↦ b | a ∈ 1..3 ∧ b = flight}
drd9 : ∀ x ∈ 1..3 ⇒ handle(x) = button
drd5 : SGCylinder = {a ↦ b | a ∈ GEARS × CYLINDER ∧ b = MOVING}

then

act1 : gstate := {a ↦ b | a ∈ GEARS ∧ b = RETRACTING}
act2 : gear_extended := {a ∈ 1..3 ⇒ GEARS ⇒ {FALSE}}
act3 : gear_shock_absorber := {a ↦ b | a ∈ 1..3 ∧ b = flight}

end

Event retraction
extends retraction
when

drd1 : dstate[DOORS] = {OPEN}
drd2 : gstate[GEARS] = {RETRACTING}
drd4 : gear_extended = {a ↦ b | a ∈ 1..3 ∧ b ∈ GEARS ⇒ {FALSE}}
\[ \text{grd5 : } \text{gear\_retracted} = \{ a \mapsto b | a \in 1..3 \land b \in \text{GEARS} \rightarrow \{ \text{FALSE} \} \} \]

\[ \text{grd6 : } \text{gear\_shock\_absorber} = \{ a \mapsto b | a \in 1..3 \land b = \text{flight} \} \]

\[ \text{grd7 : } \forall x \cdot x \in 1..3 \Rightarrow \text{handle}(x) = \text{button} \]

\[ \text{grd3 : } \text{SGCylinder} = \{ a \mapsto b | a \in \text{GEARS} \times \text{CYLINDER} \land b = \text{STOP} \} \]

\textbf{then}

\[ \text{act1 : } \text{gstate} := \{ a \mapsto b | a \in \text{GEARS} \land b = \text{RETRACTED} \} \]

\[ \text{act2 : } \text{gear\_retracted} \in 1..3 \rightarrow (\text{GEARS} \rightarrow \{ \text{TRUE} \}) \]

\textbf{when}

\[ \text{Event extending\_gears} \triangleq \]

\textbf{extends extending\_gears}

\[ \text{grd1 : } \text{dstate}[\text{DOORS}] = \{ \text{OPEN} \} \]

\[ \text{grd2 : } \text{gstate}[\text{GEARS}] = \{ \text{RETRACTED} \} \]

\[ \text{grd3 : } p = E \]

\[ \text{grd5 : } \text{gear\_retracted} = \{ a \mapsto b | a \in 1..3 \land b \in \text{GEARS} \rightarrow \{ \text{TRUE} \} \} \]

\[ \text{grd6 : } \text{gear\_extended} = \{ a \mapsto b | a \in 1..3 \land b \in \text{GEARS} \rightarrow \{ \text{FALSE} \} \} \]

\[ \text{grd7 : } \text{gear\_shock\_absorber} = \{ a \mapsto b | a \in 1..3 \land b = \text{flight} \} \]

\[ \text{grd8 : } \forall x \cdot x \in 1..3 \Rightarrow \text{handle}(x) = \text{button} \]

\[ \text{grd4 : } \text{SGCylinder} = \{ a \mapsto b | a \in \text{GEARS} \times \text{CYLINDER} \land b = \text{MOVING} \} \]

\textbf{then}

\[ \text{act1 : } \text{gstate} := \{ a \mapsto b | a \in \text{GEARS} \land b = \text{EXTENDING} \} \]

\[ \text{act2 : } \text{gear\_retracted} \in 1..3 \rightarrow (\text{GEARS} \rightarrow \{ \text{FALSE} \}) \]

\textbf{end}

\[ \text{Event extension} \triangleq \]

\textbf{extends extension}

\[ \text{grd1 : } \text{dstate}[\text{DOORS}] = \{ \text{OPEN} \} \]

\[ \text{grd2 : } \text{gstate}[\text{GEARS}] = \{ \text{EXTENDING} \} \]

\[ \text{grd4 : } \text{gear\_retracted} = \{ a \mapsto b | a \in 1..3 \land b \in \text{GEARS} \rightarrow \{ \text{FALSE} \} \} \]

\[ \text{grd5 : } \text{gear\_extended} = \{ a \mapsto b | a \in 1..3 \land b \in \text{GEARS} \rightarrow \{ \text{FALSE} \} \} \]

\[ \text{grd6 : } \text{gear\_shock\_absorber} = \{ a \mapsto b | a \in 1..3 \land b = \text{flight} \} \]

\[ \text{grd7 : } \forall x \cdot x \in 1..3 \Rightarrow \text{handle}(x) = \text{button} \]

\[ \text{grd3 : } \text{SGCylinder} = \{ a \mapsto b | a \in \text{GEARS} \times \text{CYLINDER} \land b = \text{STOP} \} \]

\textbf{then}

\[ \text{act1 : } \text{gstate} := \{ a \mapsto b | a \in \text{GEARS} \land b = \text{EXTENDED} \} \]

\[ \text{act2 : } \text{gear\_extended} \in 1..3 \rightarrow (\text{GEARS} \rightarrow \{ \text{TRUE} \}) \]

\[ \text{act3 : } \text{gear\_shock\_absorber} := \{ a \mapsto b | a \in 1..3 \land b = \text{ground} \} \]

\textbf{end}

\[ \text{Event HPD1} \triangleq \]

\textbf{extends HPD1}

\[ \text{grd3 : } \forall x \cdot x \in 1..3 \Rightarrow \text{handle}(x) = \text{UP} \]

\textbf{then}

\[ \text{act2 : } \text{handle} \in 1..3 \rightarrow \{ \text{DOWN} \} \]

\textbf{end}

\[ \text{Event HPUI} \triangleq \]
extends HPUI
when
grd3 : ∀x· x ∈ 1..3 ⇒ handle(x) = DOWN
then
act2 : handle ∈ 1..3 → {UP}
end

Event Analogical_switch_closed ≡
extends Analogical_switch_closed
any
in in port
where
grd1 : in = general_EV
grd2 : ∀x· x ∈ 1..3 ⇒ (handle(x) = UP ∨ handle(x) = DOWN)
then
act3 : analogical_switch ∈ 1..3 → {closed}
act4 : A_Switch_Out := TRUE
end

Event Analogical_switch_open ≡
extends Analogical_switch_open
any
in in port
where
grd1 : in = general_EV
grd2 : ∀x· x ∈ 1..3 ⇒ (handle(x) = UP ∨ handle(x) = DOWN)
then
act3 : analogical_switch ∈ 1..3 → {open}
act4 : A_Switch_Out := FALSE
end

Event Circuit_pressurized_OK ≡
extends Circuit_pressurized_OK
when
grd1 : general_EV_Hout = Hin
then
act9 : circuit_pressurized ∈ 1..3 → {TRUE}
end

Event Circuit_pressurized_notOK ≡
extends Circuit_pressurized_notOK
when
grd1 : general_EV_Hout = 0
then
act9 : circuit_pressurized ∈ 1..3 → {FALSE}
end

Event Computing_Module_1_2 ≡
extends Computing_Module_1_2
when

\( \text{grd1} : \text{state} = \text{computing} \)

then

\( \text{act1} : \text{general_EV} := \text{general_EV_func}(\text{handle} \mapsto \text{analogical_switch} \mapsto \text{gear} \text{\_extended} \mapsto \text{gear} \text{\_retracted} \mapsto \text{gear} \text{\_shock} \text{\_absorber} \mapsto \text{door} \text{\_open} \mapsto \text{door} \text{\_closed} \mapsto \text{circuit} \text{\_pressurized}) \)

\( \text{act2} : \text{close_EV} := \text{close_EV_func}(\text{handle} \mapsto \text{analogical_switch} \mapsto \text{gear} \text{\_extended} \mapsto \text{gear} \text{\_retracted} \mapsto \text{gear} \text{\_shock} \text{\_absorber} \mapsto \text{door} \text{\_open} \mapsto \text{door} \text{\_closed} \mapsto \text{circuit} \text{\_pressurized}) \)

\( \text{act3} : \text{retract_EV} := \text{retract_EV_func}(\text{handle} \mapsto \text{analogical_switch} \mapsto \text{gear} \text{\_extended} \mapsto \text{gear} \text{\_retracted} \mapsto \text{gear} \text{\_shock} \text{\_absorber} \mapsto \text{door} \text{\_open} \mapsto \text{door} \text{\_closed} \mapsto \text{circuit} \text{\_pressurized}) \)

\( \text{act4} : \text{extend_EV} := \text{extend_EV_func}(\text{handle} \mapsto \text{analogical_switch} \mapsto \text{gear} \text{\_extended} \mapsto \text{gear} \text{\_retracted} \mapsto \text{gear} \text{\_shock} \text{\_absorber} \mapsto \text{door} \text{\_open} \mapsto \text{door} \text{\_closed} \mapsto \text{circuit} \text{\_pressurized}) \)

\( \text{act5} : \text{open_EV} := \text{open_EV_func}(\text{handle} \mapsto \text{analogical_switch} \mapsto \text{gear} \text{\_extended} \mapsto \text{gear} \text{\_retracted} \mapsto \text{gear} \text{\_shock} \text{\_absorber} \mapsto \text{door} \text{\_open} \mapsto \text{door} \text{\_closed} \mapsto \text{circuit} \text{\_pressurized}) \)

\( \text{act6} : \text{gears} \text{\_locked} \text{\_down} := \text{gears} \text{\_locked} \text{\_down_func}(\text{handle} \mapsto \text{analogical_switch} \mapsto \text{gear} \text{\_extended} \mapsto \text{gear} \text{\_retracted} \mapsto \text{gear} \text{\_shock} \text{\_absorber} \mapsto \text{door} \text{\_open} \mapsto \text{door} \text{\_closed} \mapsto \text{circuit} \text{\_pressurized}) \)

\( \text{act7} : \text{gears} \text{\_man} := \text{gears} \text{\_man_func}(\text{handle} \mapsto \text{analogical_switch} \mapsto \text{gear} \text{\_extended} \mapsto \text{gear} \text{\_retracted} \mapsto \text{gear} \text{\_shock} \text{\_absorber} \mapsto \text{door} \text{\_open} \mapsto \text{door} \text{\_closed} \mapsto \text{circuit} \text{\_pressurized}) \)

\( \text{act8} : \text{anomaly} := \text{anomaly_func}(\text{handle} \mapsto \text{analogical_switch} \mapsto \text{gear} \text{\_extended} \mapsto \text{gear} \text{\_retracted} \mapsto \text{gear} \text{\_shock} \text{\_absorber} \mapsto \text{door} \text{\_open} \mapsto \text{door} \text{\_closed} \mapsto \text{circuit} \text{\_pressurized}) \)

\( \text{act9} : \text{state} := \text{electroV alve} \)

end

Event \( \text{Update} \text{\_Hout} \equiv \)

Assign the value of Hout

extends \( \text{Update} \text{\_Hout} \)

when

\( \text{grd1} : \text{state} = \text{electroV alve} \)

then

\( \text{act1} : \text{general} \text{\_EV} \text{\_Hout} := |(\text{general} \text{\_EV} = \text{TRUE} \land \text{general} \text{\_EV} \text{\_Hout}' = \text{Hin}) \lor (\text{general} \text{\_EV} = \text{FALSE} \land \text{general} \text{\_EV} \text{\_Hout}' = 0)\)

\( \lor (A_{\text{Switch}\text{\_Out}} = \text{TRUE} \land \text{general} \text{\_EV} \text{\_Hout}' = \text{Hin}) \lor (A_{\text{Switch}\text{\_Out}} = \text{FALSE} \land \text{general} \text{\_EV} \text{\_Hout}' = 0)) \)

pass the current value of hydraulic input port (Hin) to hydraulic output port

\( \text{Hout} \)

\( \text{act2} : \text{close} \text{\_EV} \text{\_Hout} := |(\text{close} \text{\_EV} = \text{TRUE} \land \text{close} \text{\_EV} \text{\_Hout}' = \text{Hin}) \lor (\text{close} \text{\_EV} = \text{FALSE} \land \text{close} \text{\_EV} \text{\_Hout}' = 0)\) 

\( \text{act3} : \text{open} \text{\_EV} \text{\_Hout} := |(\text{open} \text{\_EV} = \text{TRUE} \land \text{open} \text{\_EV} \text{\_Hout}' = \text{Hin}) \lor (\text{open} \text{\_EV} = \text{FALSE} \land \text{open} \text{\_EV} \text{\_Hout}' = 0)\) 

\( \text{act4} : \text{extend} \text{\_EV} \text{\_Hout} := |(\text{extend} \text{\_EV} = \text{TRUE} \land \text{extend} \text{\_EV} \text{\_Hout}' = \text{Hin}) \lor (\text{extend} \text{\_EV} = \text{FALSE} \land \text{extend} \text{\_EV} \text{\_Hout}' = 0)\) 

\( \text{act5} : \text{retract} \text{\_EV} \text{\_Hout} := |(\text{retract} \text{\_EV} = \text{TRUE} \land \text{retract} \text{\_EV} \text{\_Hout}' = \text{Hin}) \lor (\text{retract} \text{\_EV} = \text{FALSE} \land \text{retract} \text{\_EV} \text{\_Hout}' = 0)\) 

\( \text{act6} : \text{state} := \text{cylinder} \)

76
.Event CylinderMovingOrStop ≜ Cylinder Moving or Stop according to the output of hydraulic circuit

when

grd1 : state = cylinder

then

act1 : SGCylinder : ||{(SGCylinder' = {a ↦ b | a ∈ GEARS × {GCY F, GCY R, GCY L} ∧ b = MOVING} ∧ extend_EV_Hout = Hin)} ∨ (SGCylinder' = {a ↦ b | a ∈ GEARS × {GCY F, GCY R, GCY L} ∧ b = STOP} ∧ extend_EV_Hout = 0)} ∨ (SGCylinder' = {a ↦ b | a ∈ GEARS × {GCY F, GCY R, GCY L} ∧ b = MOVING}) ∧ retract_EV_Hout = Hin) ∨ (SGCylinder' = {a ↦ b | a ∈ GEARS × {GCY F, GCY R, GCY L} ∧ b = STOP}) ∧ retract_EV_Hout = 0)

act2 : SDCylinder : ||{(SDCylinder' = {a ↦ b | a ∈ DOORS × {DCY F, DCY R, DCY L} ∧ b = MOVING} ∧ open_EV_Hout = Hin)} ∨ (SDCylinder' = {a ↦ b | a ∈ DOORS × {DCY F, DCY R, DCY L} ∧ b = STOP}) ∧ open_EV_Hout = 0)} ∨ (SDCylinder' = {a ↦ b | a ∈ DOORS × {DCY F, DCY R, DCY L} ∧ b = MOVING}) ∧ close_EV_Hout = Hin) ∨ (SDCylinder' = {a ↦ b | a ∈ DOORS × {DCY F, DCY R, DCY L} ∧ b = STOP}) ∧ close_EV_Hout = 0)

act3 : state := computing

end

Event Failure_Detection ≜ extends Failure_Detection

begin

act1 : anomaly := TRUE

end

END

I M7

An Event-B Specification of M7
Creation Date: 27Jan2014 @ 10:44:59 AM

MACHINE M7
Failure Modelling
Generic Monitoring failure
Failure detection is added for doors and gears motion monitoring (Page 17)
Analogical Switch Monitoring failure (Page 16)
Pressure Sensor Monitoring failure (Page 16)
But timing requirements can be added only in last.

REFINES M6
SEES C1

VARIABLES
dstate
lstate
phase
button
p
l
i
gstate
handle
analogical_switch
gear_extended
gear_retracted
gear_shock_absorber
door_closed
door_open
circuit_pressurized
general_EV
close_EV
retract_EV
extend_EV
open_EV
gears_locked_down
gears_man
anomaly
general_EV_func
close_EV_func
retract_EV_func
extend_EV_func
open_EV_func
gears_locked_down_func
gears_man_func
anomaly_func
general_EV_Hout
close_EV_Hout
retract_EV_Hout
extend_EV_Hout
open_EV_Hout
A_Switch_Out
SDCylinder  State of Door Cylinder
SGCylinder  State of Gear Cylinder
state
EVENTS
Initialisation
extended
begin

act1 : button := DOWN
act2 : phase := haltdown
act3 : dstate : [(dstate' ∈ DOORS → SDOORS ∧ dstate' = {a → b|a ∈ DOORS ∧ b = CLOSED})]

missing elements of the invariant
act4 : lstate := {a → b|a ∈ DOORS ∧ b = LOCKED}
act5 : p := R
act6 : l := R
act7 : i := R
act8 : gstate : [(gstate' ∈ GEARs → SGEARs ∧ gstate' = {a → b|a ∈ GEARs ∧ b = EXTENDED})]
act14 : handle : ∈ 1..3 → {DOWN}
act15 : analogical_switch : ∈ 1..3 → {open}
act16 : gear_extended : ∈ 1..3 → {GEARS → {TRUE}}
act17 : gear_retracted : ∈ 1..3 → {GEARS → {FALSE}}
act18 : gear_shock_absorber : ∈ 1..3 → {ground}
act19 : door_closed : ∈ 1..3 → {DOORS → {TRUE}}
act20 : door_open : ∈ 1..3 → {DOORS → {FALSE}}
act21 : circuit_pressurized : ∈ 1..3 → {FALSE}
act22 : general_EV := FALSE
act23 : close_EV := TRUE
act24 : retract_EV := FALSE
act25 : extend_EV := TRUE
act27 : open_EV := FALSE
act28 : gears_locked_down := TRUE
act29 : gears_man := FALSE
act30 : anomaly := FALSE
act31 : general_EV_func ∈ (1..3 → POSITIONS) × (1..3 → A_Switch) × (1..3 → (GEARS → BOOL)) × (1..3 → (GEARS → BOOL)) × (1..3 → GEAR_ABSORBER) × (1..3 → (DOORS → BOOL)) × (1..3 → (DOORS → BOOL)) × (1..3 → BOOL) → BOOL
act32 : close_EV_func ∈ (1..3 → POSITIONS) × (1..3 → A_Switch) × (1..3 → (GEARS → BOOL)) × (1..3 → (GEARS → BOOL)) × (1..3 → GEAR_ABSORBER) × (1..3 → (DOORS → BOOL)) × (1..3 → (DOORS → BOOL)) × (1..3 → BOOL) → BOOL
act33 : retract_EV_func ∈ (1..3 → POSITIONS) × (1..3 → A_Switch) × (1..3 → (GEARS → BOOL)) × (1..3 → (GEARS → BOOL)) × (1..3 → GEAR_ABSORBER) × (1..3 → (DOORS → BOOL)) × (1..3 → (DOORS → BOOL)) × (1..3 → BOOL) → BOOL
act34 : extend_EV_func ∈ (1..3 → POSITIONS) × (1..3 → A_Switch) × (1..3 → (GEARS → BOOL)) × (1..3 → (GEARS → BOOL)) × (1..3 → GEAR_ABSORBER) × (1..3 → (DOORS → BOOL)) × (1..3 → (DOORS → BOOL)) × (1..3 → BOOL) → BOOL
act35 : open_EV_func ∈ (1..3 → POSITIONS) × (1..3 → A_Switch) × (1..3 → (GEARS → BOOL)) × (1..3 → (GEARS → BOOL)) × (1..3 → GEAR_ABSORBER) × (1..3 → (DOORS → BOOL)) × (1..3 → (DOORS → BOOL)) × (1..3 → BOOL) → BOOL
\textbf{Event} \textit{opening\_doors\_DOWN} \equiv \textit{opening}\_\textit{doors} \_\textit{DOWN} \\
\textbf{extends} \textit{opening}\_\textit{doors} \_\textit{DOWN} \\
\textbf{when} \\
\text{grd1} : \text{dstate}[\text{DOORS}] = \{\text{CLOSED}\} \\
\text{grd5} : \text{lstate}[\text{DOORS}] = \{\text{UNLOCKED}\} \\
\text{grd7} : \text{phase} = \text{moving\_down} \\
\text{DOORS} \\
\text{grd8} : p = R \\
\text{grd9} : l = R \\
\text{grd10} : \text{door\_open} = \{a \rightarrow b | a \in 1..3 \land b \in \text{DOORS} \rightarrow \{\text{FALSE}\}\} \\
\text{grd11} : \text{door\_closed} = \{a \rightarrow b | a \in 1..3 \land b \in \text{DOORS} \rightarrow \{\text{FALSE}\}\} \\
\text{grd12} : \forall x: x \in 1..3 \Rightarrow \text{handle}(x) = \text{button} \\
\text{grd3} : \text{SDCylinder} = \{a \rightarrow b | a \in \text{DOORS} \times \text{CYLINDER} \land b = \text{MOVING}\} \\
\text{grd13} : \text{anomaly} = \text{FALSE} \\
\textbf{then} \\
\text{act1} : \text{dstate} := \{a \rightarrow b | a \in \text{DOORS} \land b = \text{OPEN}\} \\
\text{act2} : p := \text{E} \\
\text{act3} : \text{door\_open} := \{1..3 \rightarrow (\text{DOORS} \rightarrow \{\text{TRUE}\}\} \\
\textbf{end} \\
\textbf{Event} \textit{opening\_doors\_UP} \equiv \textit{opening}\_\textit{doors} \_\textit{UP} \\
\textbf{extends} \textit{opening}\_\textit{doors} \_\textit{UP} \\
\textbf{when} \\
\text{grd1} : \text{dstate}[\text{DOORS}] = \{\text{CLOSED}\} \\
\text{grd4} : \text{lstate}[\text{DOORS}] = \{\text{UNLOCKED}\} \\
\text{grd5} : \text{phase} = \text{moving\_up} \\
\text{DOORS} \\
\text{grd6} : p = \text{E} \\
\text{grd7} : l = \text{E}
$\text{grd8:} \text{door\_open} = \{a \mapsto b | a \in 1..3 \land b \in \text{DOORS} \Rightarrow \{\text{FALSE}\}\}$

$\text{grd9:} \text{door\_closed} = \{a \mapsto b | a \in 1..3 \land b \in \text{DOORS} \Rightarrow \{\text{FALSE}\}\}$

$\text{grd10:} \forall x \cdot x \in 1..3 \Rightarrow \text{handle}(x) = \text{button}$

$\text{grd3:} \text{SDCylinder} = \{a \mapsto b | a \in \text{DOORS} \times \text{CYLINDER} \land b = \text{MOVING}\}$

then

$\text{act1:} \quad \text{dstate} := \{a \mapsto \text{OPEN} \land b \in \text{DOORS} \Rightarrow \{\text{FALSE}\}\}$

$\text{act2:} \quad \text{p} := R$

$\text{act3:} \quad \text{door\_open} : \in 1..3 \Rightarrow \{\text{DOORS} \Rightarrow \{\text{TRUE}\}\}$

end

Event $\text{closing\_doors\_UP} \equiv$

extends $\text{closing\_doors\_UP}$

any

$f$

where

$\text{grd1:} \quad \text{dstate}[\text{DOORS}] = \{\text{OPEN}\}$

$\text{grd3:} \quad f \in \text{DOORS} \Rightarrow \text{SDOORS}$

$\text{grd4:} \quad \forall e \cdot e \in \text{DOORS} \Rightarrow f(e) = \text{CLOSED}$

$\text{grd5:} \quad \text{phase} = \text{movingup}$

$\text{grd6:} \quad \text{p} = R$

$\text{grd7:} \quad \text{gstate}[\text{GEARS}] = \{\text{RETRACTED}\}$

$\text{grd8:} \quad \forall x \cdot x \in 1..3 \Rightarrow \text{handle}(x) = \text{button}$

$\text{grd9:} \quad \text{anomaly} = \text{FALSE}$

then

$\text{act1:} \quad \text{dstate} := f$

end

Event $\text{closing\_doors\_DOWN} \equiv$

extends $\text{closing\_doors\_DOWN}$

any

$f$

where

$\text{grd1:} \quad \text{dstate}[\text{DOORS}] = \{\text{OPEN}\}$

$\text{grd3:} \quad f \in \text{DOORS} \Rightarrow \text{SDOORS}$

$\text{grd4:} \quad \forall e \cdot e \in \text{DOORS} \Rightarrow f(e) = \text{CLOSED}$

$\text{grd5:} \quad \text{phase} = \text{movingdown}$

$\text{grd6:} \quad \text{p} = E$

$\text{grd7:} \quad \text{gstate}[\text{GEARS}] = \{\text{EXTENDED}\}$

$\text{grd8:} \quad \forall x \cdot x \in 1..3 \Rightarrow \text{handle}(x) = \text{button}$

$\text{grd9:} \quad \text{anomaly} = \text{FALSE}$

then

$\text{act1:} \quad \text{dstate} := f$

end

Event $\text{unlocking\_UP} \equiv$

extends $\text{unlocking\_UP}$

when

$\text{grd3:} \quad \text{lstate}[\text{DOORS}] = \{\text{LOCKED}\}$
grd4 : phase = movingup
grd5 : l = E
grd6 : p = E
grd7 : i = E
grd8 : door_open = \{a \mapsto b | a \in 1..3 \land b \in DOORS \rightarrow \{FALSE\}\}
grd9 : door_closed = \{a \mapsto b | a \in 1..3 \land b \in DOORS \rightarrow \{TRUE\}\}
grd10 : \forall x \cdot x \in 1..3 \Rightarrow handle(x) = button
grd11 : anomaly = FALSE

then

act1 : lstate := \{a \mapsto b | a \in DOORS \land b = UNLOCKED\}
act2 : door_closed := \in 1..3 \rightarrow (DOORS \rightarrow \{FALSE\})

end

Event locking_UP \equiv
extends locking_UP

when

grd3 : lstate[DOORS] = \{CLOSED\}
grd4 : phase = movingup
grd5 : lstate[DOORS] = \{UNLOCKED\}
grd6 : p = R
grd7 : l = E
grd9 : door_open = \{a \mapsto b | a \in 1..3 \land b \in DOORS \rightarrow \{FALSE\}\}
grd10 : door_closed = \{a \mapsto b | a \in 1..3 \land b \in DOORS \rightarrow \{FALSE\}\}
grd11 : \forall x \cdot x \in 1..3 \Rightarrow handle(x) = button
grd8 : SDCylinder = \{a \mapsto b | a \in DOORS \times CYLINDER \land b = STOP\}
grd12 : anomaly = FALSE

then

act1 : lstate := \{a \mapsto b | a \in DOORS \land b = LOCKED\}
act3 : phase := haltup
act4 : l := R
added by D Mery
act44 : door_closed := \in 1..3 \rightarrow (DOORS \rightarrow \{TRUE\})

end

Event unlocking_DOWN \equiv
extends unlocking_DOWN

when

grd3 : lstate[DOORS] = \{LOCKED\}
grd4 : phase = movingdown
grd5 : l = R
grd6 : p = R
grd7 : i = R
grd8 : door_open = \{a \mapsto b | a \in 1..3 \land b \in DOORS \rightarrow \{FALSE\}\}
grd9 : door_closed = \{a \mapsto b | a \in 1..3 \land b \in DOORS \rightarrow \{TRUE\}\}
grd10 : \forall x \cdot x \in 1..3 \Rightarrow handle(x) = button
grd11 : anomaly = FALSE

then

act1 : lstate := \{a \mapsto b | a \in DOORS \land b = UNLOCKED\}
act2 : door_closed := \in 1..3 \rightarrow (DOORS \rightarrow \{FALSE\})

82
Event locking\_DOWN \triangleq
extends locking\_DOWN
when

grd1 : dstate[DOORS] = \{CLOSED\}
grd2 : phase = movingdown
grd3 : lstate[DOORS] = \{UNLOCKED\}
grd4 : p = E
grd5 : l = R
grd7 : door\_open = \{a \mapsto b | a \in 1..3 \land b \in DOORS \rightarrow \{FALSE\}\}
grd8 : door\_closed = \{a \mapsto b | a \in 1..3 \land b \in DOORS \rightarrow \{FALSE\}\}
grd9 : \forall x \cdot x \in 1..3 \Rightarrow handle(x) = button
grd10 : SDCylinder = \{a \mapsto b | a \in DOORS \times CYLINDER \land b = STOP\}

then

act1 : lstate := \{a \mapsto b | a \in DOORS \land b = LOCKED\}
act3 : phase := haltdown
act4 : l := E
act5 : door\_closed := \{1..3 \rightarrow (DOORS \rightarrow \{TRUE\})\}

end

Event PD1 \triangleq
extends PD1
when

grd1 : button = UP
grd2 : phase = haltup
grd3 : \forall x \cdot x \in 1..3 \Rightarrow handle(x) = DOWN

then

act1 : phase := movingdown
act2 : button := DOWN
act3 : l := R
act4 : p := R
act5 : i := R

end

Event PU1 \triangleq
extends PU1
when

grd1 : button = DOWN
grd2 : phase = haltdown
grd3 : \forall x \cdot x \in 1..3 \Rightarrow handle(x) = UP

then

act1 : phase := movingup
act2 : button := UP
act3 : l := E
act4 : p := E
act5 : i := E

end

Event PU2 \triangleq
extends \textit{PU2}

\textbf{when}

\begin{align*}
\text{grd1} : & \ l = R \\
\text{grd2} : & \ p = R \\
\text{grd3} : & \ \text{phase} = \text{movingdown} \\
\text{grd4} : & \ \text{button} = \text{DOWN} \\
\text{grd5} : & \ i = R \\
\text{grd6} : & \ lstate[\text{DOORS}] = \{\text{LOCKED}\} \\
\text{grd7} : & \ \text{door}\_\text{open} = \{a \mapsto b \mid a \in 1..3 \land b \in \text{DOORS} \rightarrow \{\text{FALSE}\}\} \\
\text{grd8} : & \ \text{door}\_\text{closed} = \{a \mapsto b \mid a \in 1..3 \land b \in \text{DOORS} \rightarrow \{\text{TRUE}\}\} \\
\text{grd9} : & \ \forall x \cdot x \in 1..3 \Rightarrow \text{handle}(x) = \text{UP}
\end{align*}

\textbf{then}

\begin{align*}
\text{act1} : & \ \text{phase} := \text{movingup} \\
\text{act4} : & \ \text{button} := \text{UP} \\
\text{act5} : & \ l := E \\
\text{act6} : & \ p := E \\
\text{act7} : & \ i := R
\end{align*}

\textbf{end}

\textbf{Event \textit{CompletePU2} }\hat{=}\textit{ extends CompletePU2}

\textbf{when}

\begin{align*}
\text{grd1} : & \ \text{phase} = \text{movingup} \\
\text{grd2} : & \ \text{button} = \text{UP} \\
\text{grd3} : & \ l = E \\
\text{grd4} : & \ p = E \\
\text{grd5} : & \ i = R
\end{align*}

\textbf{then}

\begin{align*}
\text{act1} : & \ \text{phase} := \text{haltup}
\end{align*}

\textbf{end}

\textbf{Event \textit{PU3} }\hat{=}\textit{ extends \textit{PU3}}

\textbf{when}

\begin{align*}
\text{grd1} : & \ \text{dstate[DOORS]} = \{\text{CLOSED}\} \\
\text{grd2} : & \ \text{lstate[DOORS]} = \{\text{UNLOCKED}\} \\
\text{grd3} : & \ \text{phase} = \text{movingdown} \\
\text{grd4} : & \ p = R \\
\text{grd5} : & \ l = R \\
\text{grd6} : & \ \text{button} = \text{DOWN} \\
\text{grd7} : & \ \text{door}\_\text{open} = \{a \mapsto b \mid a \in 1..3 \land b \in \text{DOORS} \rightarrow \{\text{FALSE}\}\} \\
\text{grd8} : & \ \text{door}\_\text{closed} = \{a \mapsto b \mid a \in 1..3 \land b \in \text{DOORS} \rightarrow \{\text{FALSE}\}\} \\
\text{grd9} : & \ \forall x \cdot x \in 1..3 \Rightarrow \text{handle}(x) = \text{UP}
\end{align*}

\textbf{then}

\begin{align*}
\text{act1} : & \ \text{phase} := \text{movingup} \\
\text{act2} : & \ p := R \\
\text{act3} : & \ l := E \\
\text{act4} : & \ \text{button} := \text{UP}
\end{align*}

\textbf{end}
Event \( PU4 \cong \) extends \( PU4 \)

when

\begin{align*}
& \text{grd1} : dstate[DOORS] = \{OPEN\} \\
& \text{grd2} : phase = \text{movingdown} \\
& \text{grd3} : p = E \\
& \text{grd4} : button = \text{DOWN} \\
& \text{grd7} : \forall x \cdot x \in 1..3 \Rightarrow handle(x) = UP
\end{align*}

then

\begin{align*}
& \text{act1} : phase := \text{movingup} \\
& \text{act2} : p := R \\
& \text{act3} : button := UP \\
& \text{act4} : i := E \\
& \text{act5} : l := E
\end{align*}

end

Event \( PU5 \cong \) extends \( PU5 \)

when

\begin{align*}
& \text{grd1} : dstate[DOORS] = \{CLOSED\} \\
& \text{grd2} : phase = \text{movingdown} \\
& \text{grd3} : p = E \\
& \text{grd4} : button = \text{DOWN} \\
& \text{grd5} : lstate[DOORS] = \{UNLOCKED\} \\
& \text{grd6} : \text{door\_open} = \{a \mapsto b | a \in 1..3 \land b \in DOORS \rightarrow \{FALSE\}\} \\
& \text{grd7} : \text{door\_closed} = \{a \mapsto b | a \in 1..3 \land b \in DOORS \rightarrow \{FALSE\}\} \\
& \text{grd8} : \forall x \cdot x \in 1..3 \Rightarrow handle(x) = UP
\end{align*}

then

\begin{align*}
& \text{act1} : phase := \text{movingup} \\
& \text{act3} : button := UP \\
& \text{act4} : i := E \\
& \text{act5} : l := E
\end{align*}

event

Event \( PD2 \cong \) extends \( PD2 \)

when

\begin{align*}
& \text{grd1} : l = E \\
& \text{grd2} : p = E \\
& \text{grd3} : phase = \text{movingup} \\
& \text{grd4} : i = E \\
& \text{grd5} : lstate[DOORS] = \{LOCKED\} \\
& \text{grd6} : \forall x \cdot x \in 1..3 \Rightarrow handle(x) = \text{DOWN}
\end{align*}

then

\begin{align*}
& \text{act1} : phase := \text{movingdown} \\
& \text{act2} : button := \text{DOWN} \\
& \text{act3} : l := R \\
& \text{act4} : p := R \\
& \text{act5} : i := E
\end{align*}
Event \( \text{CompletePD2} \) \( \triangleq \)
extends \( \text{CompletePD2} \)

\begin{verbatim}
  when
  \text{grd1} : \text{phase} = \text{movingdown}
  \text{grd2} : \text{button} = \text{DOWN}
  \text{grd3} : \text{l} = \text{R}
  \text{grd4} : \text{p} = \text{R}
  \text{grd5} : \text{i} = \text{E}

  then
  \text{act1} : \text{phase} := \text{haltdown}
\end{verbatim}

Event \( \text{PD3} \) \( \triangleq \)
extends \( \text{PD3} \)

\begin{verbatim}
  when
  \text{grd1} : \text{dstate}[\text{DOORS}] = \{ \text{CLOSED} \}
  \text{grd2} : \text{lstate}[\text{DOORS}] = \{ \text{UNLOCKED} \}
  \text{grd3} : \text{phase} = \text{movingup}
  \text{grd4} : \text{p} = \text{E}
  \text{grd5} : \text{l} = \text{E}
  \text{grd6} : \text{button} = \text{UP}
  \text{grd7} : \text{door_open} = \{ a \mapsto b \mid a \in 1..3 \land b \in \text{DOORS} \rightarrow \{ \text{FALSE} \} \}
  \text{grd8} : \text{door_closed} = \{ a \mapsto b \mid a \in 1..3 \land b \in \text{DOORS} \rightarrow \{ \text{FALSE} \} \}
  \text{grd9} : \forall x \cdot x \in 1..3 \Rightarrow \text{handle}(x) = \text{DOWN}

  then
  \text{act1} : \text{phase} := \text{movingdown}
  \text{act2} : \text{p} := \text{E}
  \text{act3} : \text{l} := \text{R}
  \text{act4} : \text{button} := \text{DOWN}
\end{verbatim}

Event \( \text{PD4} \) \( \triangleq \)
extends \( \text{PD4} \)

\begin{verbatim}
  when
  \text{grd1} : \text{dstate}[\text{DOORS}] = \{ \text{OPEN} \}
  \text{grd2} : \text{phase} = \text{movingup}
  \text{grd3} : \text{p} = \text{R}
  \text{grd4} : \text{button} = \text{UP}
  \text{grd6} : \forall x \cdot x \in 1..3 \Rightarrow \text{handle}(x) = \text{DOWN}

  then
  \text{act1} : \text{phase} := \text{movingdown}
  \text{act2} : \text{p} := \text{E}
  \text{act3} : \text{button} := \text{DOWN}
  \text{act4} : \text{i} := \text{R}
  \text{act5} : \text{l} := \text{R}
\end{verbatim}

Event \( \text{PD5} \) \( \triangleq \)
extends \( \text{PD5} \)
when

\begin{align*}
grd1 & : dstate[DOORS] = \{ \text{CLOSED} \} \\
grd2 & : phase = \text{movingup} \\
grd3 & : p = R \\
grd4 & : button = \text{UP} \\
grd5 & : lstate[DOORS] = \{ \text{UNLOCKED} \} \\
grd6 & : door\_open = \{ a \mapsto a \in 1..3 \land b \in DOORS \to \{ \text{FALSE} \} \} \\
grd7 & : door\_closed = \{ a \mapsto a \in 1..3 \land b \in DOORS \to \{ \text{FALSE} \} \} \\
grd8 & : \forall x \cdot x \in 1..3 \Rightarrow handle(x) = DOWN
\end{align*}

then

\begin{align*}
act1 & : phase := \text{movingdown} \\
act2 & : button := \text{DOWN} \\
act3 & : i := R \\
act4 & : l := R
\end{align*}

end

Event retracting\_gears \equiv

extends retracting\_gears

when

\begin{align*}
grd1 & : dstate[DOORS] = \{ \text{OPEN} \} \\
grd2 & : gstate[GEARS] = \{ \text{EXTENDED} \} \\
grd3 & : p = R \\
grd6 & : gear\_extended = \{ a \mapsto b | a \in 1..3 \land b \in GEARS \to \{ \text{TRUE} \} \} \\
grd7 & : gear\_retracted = \{ a \mapsto b | a \in 1..3 \land b \in GEARS \to \{ \text{FALSE} \} \} \\
grd8 & : gear\_shock\_absorber = \{ a \mapsto b | a \in 1..3 \land b = \text{ground} \} \\
grd9 & : \forall x \cdot x \in 1..3 \Rightarrow handle(x) = button \\
grd5 & : SGCylinder = \{ a \mapsto b | a \in GEARS \times CYLINDER \land b = \text{MOVING} \}
\end{align*}

then

\begin{align*}
act1 & : gstate := \{ a \mapsto b | a \in GEARS \land b = \text{RETRACTING} \} \\
act2 & : gear\_extended := 1..3 \to (GEARS \to \{ \text{FALSE} \}) \\
act3 & : gear\_shock\_absorber := \{ a \mapsto b | a \in 1..3 \land b = \text{flight} \}
\end{align*}

end

Event retraction \equiv

extends retraction

when

\begin{align*}
grd1 & : dstate[DOORS] = \{ \text{OPEN} \} \\
grd2 & : gstate[GEARS] = \{ \text{RETRACTING} \} \\
grd4 & : gear\_extended = \{ a \mapsto b | a \in 1..3 \land b \in GEARS \to \{ \text{FALSE} \} \} \\
grd5 & : gear\_retracted = \{ a \mapsto b | a \in 1..3 \land b \in GEARS \to \{ \text{FALSE} \} \} \\
grd6 & : gear\_shock\_absorber = \{ a \mapsto b | a \in 1..3 \land b = \text{flight} \} \\
grd7 & : \forall x \cdot x \in 1..3 \Rightarrow handle(x) = button \\
grd3 & : SGCylinder = \{ a \mapsto b | a \in GEARS \times CYLINDER \land b = \text{STOP} \}
\end{align*}

then

\begin{align*}
act1 & : gstate := \{ a \mapsto b | a \in GEARS \land b = \text{RETRACTED} \} \\
act2 & : gear\_retracted := 1..3 \to (GEARS \to \{ \text{TRUE} \})
\end{align*}

end

Event extending\_gears \equiv

extends extending\_gears

87
when

grd1 : dstate[DOORS] = \{OPEN\}
grd2 : gstate[GEARS] = \{RETRACTED\}
grd3 : p = E

grd5 : gear_retracted = \{a \mapsto b | a \in 1..3 \land b \in GEARS \rightarrow \{TRUE\}\}
grd6 : gear_extended = \{a \mapsto b | a \in 1..3 \land b \in GEARS \rightarrow \{FALSE\}\}
grd7 : gear_shock_absorber = \{a \mapsto b | a \in 1..3 \land b = flight\}
grd8 : \forall x \cdot x \in 1..3 \Rightarrow handle(x) = button

grd4 : SGCylinder = \{a \mapsto b | a \in GEARS \times CYLINDER \land b = MOVING\}

then

act1 : gstate := \{a \mapsto b | a \in GEARS \land b = EXTENDING\}
act2 : gear_retracted := 1..3 \rightarrow (GEARS \rightarrow \{FALSE\})

end

Event extension ≡
extends extension
when

grd1 : dstate[DOORS] = \{OPEN\}
grd2 : gstate[GEARS] = \{EXTENDING\}
grd4 : gear_retracted = \{a \mapsto b | a \in 1..3 \land b \in GEARS \rightarrow \{FALSE\}\}
grd5 : gear_extended = \{a \mapsto b | a \in 1..3 \land b \in GEARS \rightarrow \{FALSE\}\}
grd6 : gear_shock_absorber = \{a \mapsto b | a \in 1..3 \land b = flight\}
grd7 : \forall x \cdot x \in 1..3 \Rightarrow handle(x) = button

grd3 : SGCylinder = \{a \mapsto b | a \in GEARS \times CYLINDER \land b = STOP\}

then

act1 : gstate := \{a \mapsto b | a \in GEARS \land b = EXTENDED\}
act2 : gear_extended := 1..3 \rightarrow (GEARS \rightarrow \{TRUE\})
act3 : gear_shock_absorber := \{a \mapsto b | a \in 1..3 \land b = ground\}

end

Event HPD1 ≡
extends HPD1
when

grd3 : \forall x \cdot x \in 1..3 \Rightarrow handle(x) = UP

then

act2 : handle := 1..3 \rightarrow \{DOWN\}

end

Event HPUI ≡
extends HPUI
when

grd3 : \forall x \cdot x \in 1..3 \Rightarrow handle(x) = DOWN

then

act2 : handle := 1..3 \rightarrow \{UP\}

end

Event Analogical_switch_closed ≡
extends Analogical_switch_closed
any
in in port

where

\[
\begin{align*}
grd1 & : \text{in} = \text{general}_E V \\
grd2 & : \forall x \cdot x \in 1 \ldots 3 \Rightarrow (\text{handle}(x) = \text{UP} \lor \text{handle}(x) = \text{DOWN})
\end{align*}
\]

then

\[
\begin{align*}
\text{act3} & : \text{analogical}\_\text{switch} : \in 1 \ldots 3 \rightarrow \{\text{closed}\} \\
\text{act4} & : \text{A}\_\text{Switch}\_\text{Out} := \text{TRUE}
\end{align*}
\]

\textbf{Event} \textit{Analogical\_switch\_open} \equiv\
\textbf{extends} \textit{Analogical\_switch\_open}

\textbf{any}

\[
\begin{align*}
in & \text{ in port}
\end{align*}
\]

where

\[
\begin{align*}
grd1 & : \text{in} = \text{general}_E V \\
grd2 & : \forall x \cdot x \in 1 \ldots 3 \Rightarrow (\text{handle}(x) = \text{UP} \lor \text{handle}(x) = \text{DOWN})
\end{align*}
\]

then

\[
\begin{align*}
\text{act3} & : \text{analogical}\_\text{switch} : \in 1 \ldots 3 \rightarrow \{\text{open}\} \\
\text{act4} & : \text{A}\_\text{Switch}\_\text{Out} := \text{FALSE}
\end{align*}
\]

\textbf{Event} \textit{Circuit\_pressurized\_OK} \equiv\
\textbf{extends} \textit{Circuit\_pressurized\_OK}

\textbf{when}

\[
\begin{align*}
\text{act9} & : \text{circuit}\_\text{pressurized} : \in 1 \ldots 3 \rightarrow \{\text{TRUE}\}
\end{align*}
\]

\textbf{Event} \textit{Circuit\_pressurized\_notOK} \equiv\
\textbf{extends} \textit{Circuit\_pressurized\_notOK}

\textbf{when}

\[
\begin{align*}
\text{act9} & : \text{circuit}\_\text{pressurized} : \in 1 \ldots 3 \rightarrow \{\text{FALSE}\}
\end{align*}
\]

\textbf{Event} \textit{Computing\_Module\_1\,2} \equiv\
\textbf{extends} \textit{Computing\_Module\_1\,2}

\textbf{when}

\[
\begin{align*}
\text{act1} & : \text{general}_E V := \text{general}_E V\_\text{func}(\text{handle} \mapsto \text{analogical}_\text{switch} \mapsto \text{gear}_\text{extended} \mapsto \text{gear}_\text{retracted} \mapsto \text{gear}_\text{shock}\_\text{absorber} \mapsto \text{door}_\text{open} \mapsto \text{door}_\text{closed} \mapsto \text{circuit}_\text{pressurized}) \\
\text{act2} & : \text{close}_E V := \text{close}_E V\_\text{func}(\text{handle} \mapsto \text{analogical}_\text{switch} \mapsto \text{gear}_\text{extended} \mapsto \text{gear}_\text{retracted} \mapsto \text{gear}_\text{shock}\_\text{absorber} \mapsto \text{door}_\text{open} \mapsto \text{door}_\text{closed} \mapsto \text{circuit}_\text{pressurized})
\end{align*}
\]
act3 : retract_EV := retract_EV_func(handle ↔ analogical_switch ↔
gear_extended ↔ gear_retracted ↔ gear_shock_absorber ↔ door_open ↔
door_closed ↔ circuit_ pressurized)
act4 : extend_EV := extend_EV_func(handle ↔ analogical_switch ↔
gear_extended ↔ gear_retracted ↔ gear_shock_absorber ↔ door_open ↔
door_closed ↔ circuit_ pressurized)
act5 : open_EV := open_EV_func(handle ↔ analogical_switch ↔
gear_extended ↔ gear_retracted ↔ gear_shock_absorber ↔ door_open ↔
door_closed ↔ circuit_ pressurized)
act6 : gears_locked_down := gears_locked_down_func(handle ↔
analogical_switch ↔ gear_extended ↔ gear_retracted ↔
gear_shock_absorber ↔ door_open ↔ door_closed ↔ circuit_ pressurized)
act7 : gears_man := gears_man_func(handle ↔ analogical_switch ↔
gear_extended ↔ gear_retracted ↔ gear_shock_absorber ↔ door_open ↔
door_closed ↔ circuit_ pressurized)
act8 : anomaly := anomaly_func(handle ↔ analogical_switch ↔
gear_extended ↔ gear_retracted ↔ gear_shock_absorber ↔ door_open ↔
door_closed ↔ circuit_ pressurized)
act9 : state := electroValue
end

Event Update_Hout ≡

Assign the value of Hout

extends Update_Hout
when
then

grd1 : state = electroValue

act1 : general_EV_Hout : ( (general_EV = TRUE ∧ general_EV_Hout' =
Hin ) ∨ (general_EV = FALSE ∧ general_EV_Hout' = 0 ) ) ∧ ( A_Switch_Out = TRUE ∧ general_EV_Hout' =
Hin ) ∨ ( A_Switch_Out = FALSE ∧ general_EV_Hout' = 0 ) )

pass the current value of hydraulic input port (Hin) to hydraulic output port (Hout)

act2 : close_EV_Hout : ( (close_EV = TRUE ∧ close_EV_Hout' =
Hin ) ∨ (close_EV = FALSE ∧ close_EV_Hout' = 0 ) )
act3 : open_EV_Hout : ( (open_EV = TRUE ∧ open_EV_Hout' =
Hin ) ∨ (open_EV = FALSE ∧ open_EV_Hout' = 0 ) )
act4 : extend_EV_Hout : ( (extend_EV = TRUE ∧
extend_EV_Hout' = Hin ) ∨ (extend_EV = FALSE ∧ extend_EV_Hout' = 0 ) )
act5 : retract_EV_Hout : ( (retract_EV = TRUE ∧ retrieve_EV_Hout' =
Hin ) ∨ (retract_EV = FALSE ∧ retract_EV_Hout' = 0 ) )
act6 : state := cylinder

end

Event CylinderMovingOrStop ≡

Cylinder Moving or Stop according to the output of hydraulic circuit

extends CylinderMovingOrStop
when
then

grd1 : state = cylinder

90
act1 : SGCylinder : \[\{ (SGCylinder') = \{ a \mapsto b | a \in GEARS \times \{ GCY_F, GCY_R, GCY_L \} \land b = MOVING \} \land \text{extend}_{EV_Hout} = Hin \} \lor \]
\[ (SGCylinder') = \{ a \mapsto b | a \in GEARS \times \{ GCY_F, GCY_R, GCY_L \} \land b = STOP \} \land \text{extend}_{EV_Hout} = 0 \} \lor \]
\[ (SGCylinder') = \{ a \mapsto b | a \in GEARS \times \{ GCY_F, GCY_R, GCY_L \} \land b = MOVING \} \land \text{retract}_{EV_Hout} = Hin \} \lor \]
\[ (SGCylinder') = \{ a \mapsto b | a \in GEARS \times \{ GCY_F, GCY_R, GCY_L \} \land b = STOP \} \land \text{retract}_{EV_Hout} = 0 \}) \]

act2 : SDCylinder : \[\{ (SDCylinder') = \{ a \mapsto b | a \in DOORS \times \{ DCY_F, DCY_R, DCY_L \} \land b = MOVING \} \land \text{open}_{EV_Hout} = Hin \} \lor \]
\[ (SDCylinder') = \{ a \mapsto b | a \in DOORS \times \{ DCY_F, DCY_R, DCY_L \} \land b = STOP \} \land \text{open}_{EV_Hout} = 0 \} \lor \]
\[ (SDCylinder') = \{ a \mapsto b | a \in DOORS \times \{ DCY_F, DCY_R, DCY_L \} \land b = MOVING \} \land \text{close}_{EV_Hout} = Hin \} \lor \]
\[ (SDCylinder') = \{ a \mapsto b | a \in DOORS \times \{ DCY_F, DCY_R, DCY_L \} \land b = STOP \} \land \text{close}_{EV_Hout} = 0 \}) \]

act3 : state := \text{computing}

Event Failure Detection Generic Monitoring \( \hat{=} \)
extends Failure Detection

when
Generic Monitoring using all sensors

\[
grd1 : (\forall x, y, z : x \in 1..3 \land y \in 1..3 \land z \in 1..3 \land x \neq y \land y \neq z \land x \neq z \Rightarrow \\
\text{handle}(x) \neq \text{handle}(y) \land \text{handle}(y) \neq \text{handle}(z) \land \text{handle}(x) \neq \text{handle}(z))
\]

\[
\lor
\]

\[
\forall x, y, z : x \in 1..3 \land y \in 1..3 \land z \in 1..3 \land x \neq y \land y \neq z \land x \neq z \Rightarrow \\
\text{analogical_switch}(x) \neq \text{analogical_switch}(y) \land \text{analogical_switch}(y) \neq \text{analogical_switch}(z) \land \text{analogical_switch}(z) \neq \text{analogical_switch}(x)
\]

\[
\lor
\]

\[
\forall x, y, z : x \in 1..3 \land y \in 1..3 \land z \in 1..3 \land x \neq y \land y \neq z \land x \neq z \Rightarrow \\
\text{gear_retracted}(x) \neq \text{gear_retracted}(y) \land \text{gear_retracted}(y) \neq \text{gear_retracted}(z) \land \text{gear_retracted}(z) \neq \text{gear_retracted}(x)
\]

\[
\lor
\]

\[
\forall x, y, z : x \in 1..3 \land y \in 1..3 \land z \in 1..3 \land x \neq y \land y \neq z \land x \neq z \Rightarrow \\
\text{gear_shock_absorber}(x) \neq \text{gear_shock_absorber}(y) \land \text{gear_shock_absorber}(y) \neq \text{gear_shock_absorber}(z) \land \text{gear_shock_absorber}(z) \neq \text{gear_shock_absorber}(x)
\]

\[
\lor
\]

\[
\forall x, y, z : x \in 1..3 \land y \in 1..3 \land z \in 1..3 \land x \neq y \land y \neq z \land x \neq z \Rightarrow \\
\text{door_open}(x) \neq \text{door_open}(y) \land \text{door_open}(y) \neq \text{door_open}(z) \land \text{door_open}(z) \neq \text{door_open}(x)
\]

\[
\lor
\]

\[
\forall x, y, z : x \in 1..3 \land y \in 1..3 \land z \in 1..3 \land x \neq y \land y \neq z \land x \neq z \Rightarrow \\
\text{door_closed}(x) \neq \text{door_closed}(y) \land \text{door_closed}(y) \neq \text{door_closed}(z) \land \text{door_closed}(z) \neq \text{door_closed}(x)
\]

\[
\lor
\]

\[
\forall x, y, z : x \in 1..3 \land y \in 1..3 \land z \in 1..3 \land x \neq y \land y \neq z \land x \neq z \Rightarrow \\
\text{circuit_pressurized}(x) \neq \text{circuit_pressurized}(y) \land \text{circuit_pressurized}(y) \neq \text{circuit_pressurized}(z) \land \text{circuit_pressurized}(z) \neq \text{circuit_pressurized}(x)
\]

Event Failure_Detection_Analogical_Switch \(\subseteq\) Failure_Detection
when

\[
grd1 : \text{analogical_switch} = \{a \mapsto b | a \in 1..3 \land b = \text{open}\}
\]

\[
\lor
\]

\[
\text{analogical_switch} = \{a \mapsto b | a \in 1..3 \land b = \text{closed}\}
\]

Gears motion monitoring without considering time
then

\[
\text{act1} : \text{anomaly} := \text{TRUE}
\]

end

Event Failure_Detection_Pressure_Sensor \(\subseteq\)

\[
\text{act1} : \text{anomaly} := \text{TRUE}
\]

end
extends Failure_Detection
when
  grd1 : circuit_pressurized ≠ \{a → b|a ∈ 1..3 ∧ b = TRUE\}
  \lor 
  circuit_pressurized ≠ \{a → b|a ∈ 1..3 ∧ b = FALSE\}
  Circuit pressurized motion monitoring without considering time
then
  act1 : anomaly := TRUE
end

Event Failure_Detection_Doors \equiv
extends Failure_Detection
when
  grd1 : door_closed ≠ \{a → b|a ∈ 1..3 ∧ b ∈ DOORS → \{FALSE\}\}
  \lor 
  door_open ≠ \{a → b|a ∈ 1..3 ∧ b ∈ DOORS → \{TRUE\}\}
  \lor 
  door_open ≠ \{a → b|a ∈ 1..3 ∧ b ∈ DOORS → \{FALSE\}\}
  \lor 
  door_closed ≠ \{a → b|a ∈ 1..3 ∧ b ∈ DOORS → \{TRUE\}\}
  Doors motion monitoring without considering time
then
  act1 : anomaly := TRUE
end

Event Failure_Detection_Gears \equiv
extends Failure_Detection
when
  grd1 : gear_retracted ≠ \{a → b|a ∈ 1..3 ∧ b ∈ GEARs → \{FALSE\}\}
  \lor 
  gear_retracted ≠ \{a → b|a ∈ 1..3 ∧ b ∈ GEARs → \{TRUE\}\}
  \lor 
  gear_retracted ≠ \{a → b|a ∈ 1..3 ∧ b ∈ GEARs → \{FALSE\}\}
  \lor 
  gear_extended ≠ \{a → b|a ∈ 1..3 ∧ b ∈ GEARs → \{TRUE\}\}
  Gears motion monitoring without considering time
then
  act1 : anomaly := TRUE
end
END

J M8

An Event-B Specification of M8
Creation Date: 27Jan2014 @ 10:44:59 AM

MACHINE M8
Timing Requirements.
REFINES M7
SEES C1
VARIABLES

dstate
lstate
phase
button
p
l
i
gstate
handle
analogical_switch
gear_extended
gear_retracted
gear_shock_absorber
door_closed
door_open
circuit_pressurized
general_EV
close_EV
retract_EV
extend_EV
open_EV
gears_locked_down
gears_man
anomaly
general_EV_func
close_EV_func
retract_EV_func
extend_EV_func
open_EV_func
gears_locked_down_func
gears_man_func
anomaly_func
general_EV_Hout
close_EV_Hout
retract_EV_Hout
extend_EV_Hout
open_EV_Hout
SDCylinder  State of Door Cylinder
SGCylinder  State of Gear Cylinder
A_Switch_Out  State of Gear Cylinder
state

time current time

at a future event activation set.

index To take a function to index different sets for event activation set

handleUp_interval To keep an update time duration after handle up

handleDown_interval To keep an update time duration after handle down

INVARIANTS

inv1 : time ∈ N
  current updating time

inv2 : at ⊆ N × N
  a set of times for activating event

inv3 : ran(at) ≠ ∅ ⇒ time ≤ min(ran(at))
  if activation is a non empty set then the current time will
  be less than or equal to the minimum of activation set.

inv4 : index ∈ N
  an index for event activation set to store multiple identical values

inv5 : handleUp_interval ∈ N
  time interval after handle up

inv6 : handleDown_interval ∈ N
  time interval after handle down

EVENTS

Initialisation

extended

begin

act1 : button := DOWN

act2 : phase := haltdown

act3 : dstate : [(dstate' ∈ DOORS → SDOORS ∧ dstate' = \{a → b|a ∈ DOORS ∧ b = CLOSED\})
  missing elements of the invariant

act4 : lstate := \{a → b|a ∈ DOORS ∧ b = LOCKED\}

act5 : p := R

act6 : l := R

act7 : i := R

act8 : gstate : [(gstate' ∈ GEARs → SGEARS ∧ gstate' = \{a → b|a ∈ GEARs ∧ b = EXTENDED\})
  missing elements of the invariant

act14 : handle : ∈ 1..3 → {DOWN}

act15 : analogical_switch : ∈ 1..3 → {open}

act16 : gear_extended : ∈ 1..3 → (GEARS → {TRUE})

act17 : gear_retracted : ∈ 1..3 → (GEARS → {FALSE})

act18 : gear_shock_absorber : ∈ 1..3 → {ground}

act19 : door_closed : ∈ 1..3 → {DOORS → {TRUE}}

act20 : door_open : ∈ 1..3 → {DOORS → {FALSE}}

act21 : circuit_pressurized : ∈ 1..3 → {FALSE}

act22 : general_EV := FALSE

act23 : close_EV := TRUE

act24 : retract_EV := FALSE

act25 : extend_EV := TRUE

act27 : open_EV := FALSE
act28 : gears_locked_down := TRUE
act29 : gears_man := FALSE
act30 : anomaly := FALSE
act31 : general_EV_func ∈ (1..3 → POSITIONS) × (1..3 → A_Switch) × (1..3 → (GEARS → BOOL)) × (1..3 → (GEARS → BOOL)) × (1..3 → (DOORS → BOOL)) × (1..3 → (DOORS → BOOL)) × (1..3 → BOOЛ)
act32 : close_EV_func ∈ (1..3 → POSITIONS) × (1..3 → A_Switch) × (1..3 → (GEARS → BOOL)) × (1..3 → (GEARS → BOOL)) × (1..3 → (GEARS → BOOL)) × (1..3 → (DOORS → BOOL)) × (1..3 → (DOORS → BOOL)) × (1..3 → BOOЛ)
act33 : retract_EV_func ∈ (1..3 → POSITIONS) × (1..3 → A_Switch) × (1..3 → (GEARS → BOOL)) × (1..3 → (GEARS → BOOL)) × (1..3 → (GEARS → BOOL)) × (1..3 → (DOORS → BOOL)) × (1..3 → (DOORS → BOOL)) × (1..3 → BOOЛ)
act34 : extend_EV_func ∈ (1..3 → POSITIONS) × (1..3 → A_Switch) × (1..3 → (GEARS → BOOL)) × (1..3 → (GEARS → BOOL)) × (1..3 → (GEARS → BOOL)) × (1..3 → (DOORS → BOOL)) × (1..3 → (DOORS → BOOL)) × (1..3 → BOOЛ)
act35 : open_EV_func ∈ (1..3 → POSITIONS) × (1..3 → A_Switch) × (1..3 → (GEARS → BOOL)) × (1..3 → (GEARS → BOOL)) × (1..3 → (GEARS → BOOL)) × (1..3 → (DOORS → BOOL)) × (1..3 → (DOORS → BOOL)) × (1..3 → BOOЛ)
act36 : gears_locked_down_func ∈ (1..3 → POSITIONS) × (1..3 → A_Switch) × (1..3 → (GEARS → BOOL)) × (1..3 → (GEARS → BOOL)) × (1..3 → (GEARS → BOOL)) × (1..3 → (DOORS → BOOL)) × (1..3 → (DOORS → BOOL)) × (1..3 → BOOЛ)
act37 : gears_man_func ∈ (1..3 → POSITIONS) × (1..3 → A_Switch) × (1..3 → (GEARS → BOOL)) × (1..3 → (GEARS → BOOL)) × (1..3 → (GEARS → BOOL)) × (1..3 → (DOORS → BOOL)) × (1..3 → (DOORS → BOOL)) × (1..3 → BOOЛ)
act38 : anomaly_func ∈ (1..3 → POSITIONS) × (1..3 → A_Switch) × (1..3 → (GEARS → BOOL)) × (1..3 → (GEARS → BOOL)) × (1..3 → (GEARS → BOOL)) × (1..3 → (DOORS → BOOL)) × (1..3 → (DOORS → BOOL)) × (1..3 → BOOЛ)
act39 : A_Switch_Out := FALSE
act40 : close_EV_Hout := 0
act41 : retract_EV_Hout := 0
act42 : extend_EV_Hout := 0
act43 : open_EV_Hout := 0
act44 : general_EV_Hout := 0
act45 : SDCylinder ∈ DOORS × {DCY_F, DCY_R, DCY_L} → {STOP}
act46 : GCGylinder ∈ GEARS × {GCY_F, GCY_R, GCY_L} → {STOP}
act47 : at := Ø
act48 : time := 0
act49 : index := 0
act50 : handleUp_interval := 0
act51 : handleDown_interval := 0

end
 Event \( opening\_doors\_DOWN \) \( \triangleq \)
\( \text{extends}\) opening\_doors\_DOWN
\(\text{when}\)

\begin{align*}
\text{grd1} & : dstate[DOORS] = \{CLOSED\} \\
\text{grd5} & : lstate[DOORS] = \{UNLOCKED\} \\
\text{grd7} & : \text{phase} = \text{movingdown} \\
\text{grd8} & : p = R \\
\text{grd9} & : l = R \\
\text{grd10} & : \text{door\_open} = \{a \mapsto b | a \in 1..3 \land b \in DOORS \rightarrow \{FALSE\}\} \\
\text{grd11} & : \text{door\_closed} = \{a \mapsto b | a \in 1..3 \land b \in DOORS \rightarrow \{FALSE\}\} \\
\text{grd12} & : \forall \, x \in 1..3 \Rightarrow handle(x) = \text{button} \\
\text{grd13} & : SDCylinder = \{a \mapsto b | a \in DOORS \times CYLINDER \land b = \text{MOVING}\} \\
\text{then}\)

\begin{align*}
\text{act1} & : dstate := \{a \mapsto b | a \in DOORS \land b = \text{OPEN}\} \\
\text{act2} & : p := E \\
\text{act3} & : door\_open :\in 1..3 \rightarrow (DOORS \rightarrow \{\text{TRUE}\}) \\
\text{act4} & : \text{at} := \text{at} \cup \{(index + 1) \mapsto (time + 100)\} \\
& \text{minimal interval for door open to gear extension} \\
\text{act5} & : index := index + 1
\end{align*}

\(\text{end}\)

Event \( opening\_doors\_UP \) \( \triangleq \)
\( \text{extends}\) opening\_doors\_UP
\(\text{when}\)

\begin{align*}
\text{grd1} & : dstate[DOORS] = \{CLOSED\} \\
\text{grd4} & : lstate[DOORS] = \{UNLOCKED\} \\
\text{grd5} & : \text{phase} = \text{movingup} \\
\text{grd6} & : p = E \\
\text{grd7} & : l = E \\
\text{grd8} & : \text{door\_open} = \{a \mapsto b | a \in 1..3 \land b \in DOORS \rightarrow \{FALSE\}\} \\
\text{grd9} & : \text{door\_closed} = \{a \mapsto b | a \in 1..3 \land b \in DOORS \rightarrow \{FALSE\}\} \\
\text{grd10} & : \forall \, x \in 1..3 \Rightarrow handle(x) = \text{button} \\
\text{grd13} & : SDCylinder = \{a \mapsto b | a \in DOORS \times CYLINDER \land b = \text{MOVING}\} \\
\text{then}\)

\begin{align*}
\text{act1} & : dstate := \{a \mapsto b | a \in DOORS \land b = \text{OPEN}\} \\
\text{act2} & : p := R \\
\text{act3} & : door\_open :\in 1..3 \rightarrow (DOORS \rightarrow \{\text{TRUE}\}) \\
\text{act4} & : \text{at} := \text{at} \cup \{(index + 1) \mapsto (time + 100)\} \\
& \text{minimal interval for door open to gear retraction} \\
\text{act5} & : index := index + 1
\end{align*}

\(\text{end}\)

Event \( closing\_doors\_UP \) \( \triangleq \)
\( \text{extends}\) closing\_doors\_UP
\(\text{any}\)

\begin{align*}
\text{f} \\
\text{where}\)

\begin{align*}
\text{grd1} & : dstate[DOORS] = \{OPEN\}
\end{align*}
grd3 : f ∈ DOORS → SDOORS  
grd4 : ∀e ∈ DOORS ⇒ f(e) = CLOSED  
grd5 : phase = movingup  
grd6 : p = R  
grd7 : gstate[GEARS] = {RETRACTED}  
grd8 : ∀x · x ∈ 1..3 ⇒ handle(x) = button  
grd9 : anomaly = FALSE  

then  
act1 : dstate := f  
end  

Event closingdoors DOWN  
extends closingdoors DOWN  
any  

f  
where  
grd1 : dstate[DOORS] = {OPEN}  
grd3 : f ∈ DOORS → SDOORS  
grd4 : ∀e ∈ DOORS ⇒ f(e) = CLOSED  
grd5 : phase = movingdown  
grd6 : p = E  
grd7 : gstate[GEARS] = {EXTENDED}  
grd8 : ∀x · x ∈ 1..3 ⇒ handle(x) = button  
grd9 : anomaly = FALSE  

then  
act1 : dstate := f  
end  

Event unlocking UP  
extends unlocking UP  
when  
grd3 : lstate[DOORS] = {LOCKED}  
grd4 : phase = movingup  
grd5 : l = E  
grd6 : p = E  
grd7 : i = E  
grd8 : door_open = {a ∈ 1..3 ∧ b ∈ DOORS ⇒ {FALSE}}  
grd9 : door_closed = {a ∈ 1..3 ∧ b ∈ DOORS ⇒ {TRUE}}  
grd10 : ∀x · x ∈ 1..3 ⇒ handle(x) = button  
grd11 : anomaly = FALSE  

then  
act1 : lstate := {a ∈ DOORS ∧ b = UNLOCKED}  
act2 : door_closed :∈ 1..3 → (DOORS → {FALSE})  
end  

Event locking UP  
extends locking UP  
when  
grd3 : dstate[DOORS] = {CLOSED}  

98
grd4 : phase = movingup
grd5 : lstate[DOORS] = \{UNLOCKED\}
grd6 : p = R
grd7 : l = E
grd9 : door_{open} = \{a \mapsto b | a \in 1 \ldots 3 \land b \in DOORS \rightarrow \{FALSE\}\}
grd10 : door_{closed} = \{a \mapsto b | a \in 1 \ldots 3 \land b \in DOORS \rightarrow \{FALSE\}\}
grd11 : \forall x \cdot x \in 1 \ldots 3 \Rightarrow handle(x) = \text{button}
grd8 : SDCylinder = \{a \mapsto b | a \in DOORS \times CYLINDER \land b = \text{STOP}\}
grd12 : \text{anomaly} = \text{FALSE}

then

act1 : lstate := \{a \mapsto b | a \in DOORS \land b = \text{LOCKED}\}
act3 : phase := \text{haltup}
act4 : l := R

added by D Mery

act44 : door_{closed} : \in 1 \ldots 3 \rightarrow (DOORS \rightarrow \{\text{TRUE}\})
act5 : at := at \cup \{(\text{index} + 1) \mapsto (\text{time} + 100)\}

minimal interval for door closed to gear extension/retraction
act6 : index := index + 1

end

Event unlocking\_DOWN \underset{\Delta}{=} 
extends unlocking\_DOWN

when

grd3 : lstate[DOORS] = \{LOCKED\}
grd4 : phase = movingdown
grd5 : l = R
grd6 : p = R
grd7 : i = R
grd8 : door_{open} = \{a \mapsto b | a \in 1 \ldots 3 \land b \in DOORS \rightarrow \{FALSE\}\}
grd9 : door_{closed} = \{a \mapsto b | a \in 1 \ldots 3 \land b \in DOORS \rightarrow \{TRUE\}\}
grd10 : \forall x \cdot x \in 1 \ldots 3 \Rightarrow handle(x) = \text{button}
grd11 : \text{anomaly} = \text{FALSE}

then

act1 : lstate := \{a \mapsto b | a \in DOORS \land b = \text{UNLOCKED}\}
act2 : door_{closed} : \in 1 \ldots 3 \rightarrow (DOORS \rightarrow \{\text{FALSE}\})

end

Event locking\_DOWN \underset{\Delta}{=} 
extends locking\_DOWN

when

grd1 : dstate[DOORS] = \{CLOSED\}
grd2 : phase = movingdown
grd3 : lstate[DOORS] = \{UNLOCKED\}
grd4 : p = E
grd5 : l = R
grd7 : door_{open} = \{a \mapsto b | a \in 1 \ldots 3 \land b \in DOORS \rightarrow \{FALSE\}\}
grd8 : door_{closed} = \{a \mapsto b | a \in 1 \ldots 3 \land b \in DOORS \rightarrow \{FALSE\}\}
grd9 : \forall x \cdot x \in 1 \ldots 3 \Rightarrow handle(x) = \text{button}
grd6 : SDCylinder = \{a \mapsto b | a \in DOORS \times CYLINDER \land b = \text{STOP}\}
grd10 : \text{anomaly} = \text{FALSE}
then

\[ \text{act1} : \text{lstate} := \{a \mapsto b | a \in \text{DOORS} \land b = \text{LOCKED}\} \]
\[ \text{act3} : \text{phase} := \text{haltdown} \]
\[ \text{act4} : l := E \]
\[ \text{act5} : \text{door\_closed} : \in 1..3 \rightarrow (\text{DOORS} \rightarrow \{\text{TRUE}\}) \]
\[ \text{act6} : \text{at} := \text{at} \cup \{((\text{index} + 1) \mapsto (\text{time} + 1))\} \]

minimal interval for door closed to extension/retraction
\[ \text{act7} : \text{index} := \text{index} + 1 \]
end

Event $PD1 \triangleq$
extends $PD1$

when

\[ \text{grd1} : \text{button} = UP \]
\[ \text{grd2} : \text{phase} = \text{haltup} \]
\[ \text{grd3} : \forall x \cdot x \in 1..3 \Rightarrow \text{handle}(x) = \text{DOWN} \]
then

\[ \text{act1} : \text{phase} := \text{movingdown} \]
\[ \text{act2} : \text{button} := \text{DOWN} \]
\[ \text{act3} : l := R \]
\[ \text{act4} : p := R \]
\[ \text{act5} : i := R \]
end

Event $PU1 \triangleq$
extends $PU1$

when

\[ \text{grd1} : \text{button} = \text{DOWN} \]
\[ \text{grd2} : \text{phase} = \text{haltdown} \]
\[ \text{grd3} : \forall x \cdot x \in 1..3 \Rightarrow \text{handle}(x) = \text{UP} \]
then

\[ \text{act1} : \text{phase} := \text{movingup} \]
\[ \text{act2} : \text{button} := \text{UP} \]
\[ \text{act3} : l := E \]
\[ \text{act4} : p := E \]
\[ \text{act5} : i := E \]
end

Event $PU2 \triangleq$
extends $PU2$

when

\[ \text{grd1} : l = R \]
\[ \text{grd2} : p = R \]
\[ \text{grd3} : \text{phase} = \text{movingdown} \]
\[ \text{grd4} : \text{button} = \text{DOWN} \]
\[ \text{grd5} : i = R \]
\[ \text{grd6} : \text{lstate}[\text{DOORS}] = \{\text{LOCKED}\} \]
\[ \text{grd7} : \text{door\_open} = \{a \mapsto b | a \in 1..3 \land b \in \text{DOORS} \rightarrow \{\text{FALSE}\}\} \]
\[ \text{grd8} : \text{door\_closed} = \{a \mapsto b | a \in 1..3 \land b \in \text{DOORS} \rightarrow \{\text{TRUE}\}\} \]
\[ \forall x \cdot x \in 1..3 \Rightarrow \text{handle}(x) = \text{UP} \]
then

act1 : phase := movingup
act4 : button := UP
act5 : l := E
act6 : p := E
act7 : i := R

end

Event CompletePU2 \equiv
extends CompletePU2

when

grd1 : phase = movingup
grd2 : button = UP
grd3 : l = E
grd4 : p = E
grd5 : i = R

then

act1 : phase := haltup

end

Event PU3 \equiv
extends PU3

when

grd1 : dstate[DOORS] = \{CLOSED\}
grd2 : lstate[DOORS] = \{UNLOCKED\}
grd3 : phase = movingdown
grd4 : p = R
grd5 : l = R
grd6 : button = DOWN
grd7 : door_open = \{a \mapsto b | a \in 1..3 \land b \in DOORS \rightarrow \{FALSE\}\}
grd8 : door_closed = \{a \mapsto b | a \in 1..3 \land b \in DOORS \rightarrow \{FALSE\}\}
grd9 : \forall x \cdot x \in 1..3 \Rightarrow handle(x) = UP

then

act1 : phase := movingup
act2 : p := R
act3 : l := E
act4 : button := UP

end

Event PU4 \equiv
extends PU4

when

grd1 : dstate[DOORS] = \{OPEN\}
grd2 : phase = movingdown
grd3 : p = E
grd4 : button = DOWN
grd7 : \forall x \cdot x \in 1..3 \Rightarrow handle(x) = UP

then

act1 : phase := movingup
act2 : p := R
act3 : button := UP
act4 : i := E
act5 : l := E

Event PU5 \models
extends PU5
when

grd1 : dstate[DOORS] = {CLOSED}
grd2 : phase = movingdown
grd3 : p = E
grd4 : button = DOWN
grd5 : lstate[DOORS] = \{UNLOCKED\}
grd6 : door_open = \{ a \mapsto b | a \in 1..3 \land b \in DOORS \rightarrow FALSE \}\}
grd7 : door_closed = \{ a \mapsto b | a \in 1..3 \land b \in DOORS \rightarrow FALSE \}\}
grd8 : \forall x \cdot x \in 1..3 \Rightarrow handle(x) = UP

then

act1 : phase := movingup
act3 : button := UP
act4 : i := E
act5 : l := E

end

Event PD2 \models
extends PD2
when

grd1 : l = E
grd2 : p = E
grd3 : phase = movingup
grd4 : i = E
grd5 : lstate[DOORS] = \{LOCKED\}
grd6 : \forall x \cdot x \in 1..3 \Rightarrow handle(x) = DOWN

then

act1 : phase := movingdown
act2 : button := DOWN
act3 : l := R
act4 : p := R
act5 : i := E

end

Event CompletePD2 \models
extends CompletePD2
when

grd1 : phase = movingdown
grd2 : button = DOWN
grd3 : l = R
grd4 : p = R
grd5 : i = E

then

act1 : phase := haltdown
Event \( PD3 \) \( \subseteq \) extends \( PD3 \) when

\[
\begin{align*}
grd1 & : dstate[DOORS] = \{CLOSED\} \\
grd2 & : lstate[DOORS] = \{UNLOCKED\} \\
grd3 & : phase = movingup \\
grd4 & : p = E \\
grd5 & : l = E \\
grd6 & : button = UP \\
grd7 & : door_{open} = \{a \mapsto b|a \in 1..3 \land b \in DOORS \rightarrow \{FALSE\}\} \\
grd8 & : door_{closed} = \{a \mapsto b|a \in 1..3 \land b \in DOORS \rightarrow \{FALSE\}\} \\
grd9 & : \forall x \cdot x \in 1..3 \Rightarrow handle(x) = DOWN 
\end{align*}
\]

then

\[
\begin{align*}
act1 & : phase := movingdown \\
act2 & : p := E \\
act3 & : l := R \\
act4 & : button := DOWN
\end{align*}
\]

end

Event \( PD4 \) \( \subseteq \) extends \( PD4 \) when

\[
\begin{align*}
grd1 & : dstate[DOORS] = \{OPEN\} \\
grd2 & : phase = movingup \\
grd3 & : p = R \\
grd4 & : button = UP \\
grd5 & : \forall x \cdot x \in 1..3 \Rightarrow handle(x) = DOWN
\end{align*}
\]

then

\[
\begin{align*}
act1 & : phase := movingdown \\
act2 & : p := E \\
act3 & : button := DOWN \\
act4 & : i := R \\
act5 & : l := R
\end{align*}
\]

end

Event \( PD5 \) \( \subseteq \) extends \( PD5 \) when

\[
\begin{align*}
grd1 & : dstate[DOORS] = \{CLOSED\} \\
grd2 & : phase = movingup \\
grd3 & : p = R \\
grd4 & : button = UP \\
grd5 & : lstate[DOORS] = \{UNLOCKED\} \\
grd6 & : door_{open} = \{a \mapsto b|a \in 1..3 \land b \in DOORS \rightarrow \{FALSE\}\} \\
grd7 & : door_{closed} = \{a \mapsto b|a \in 1..3 \land b \in DOORS \rightarrow \{FALSE\}\} \\
grd8 & : \forall x \cdot x \in 1..3 \Rightarrow handle(x) = DOWN
\end{align*}
\]

then

\[
act1 : phase := movingdown
\]
\[act2 : button := DOWN\]
\[act3 : i := R\]
\[act4 : l := R\]

\[\textbf{Event retracting_gears} \equiv\]
\[\textbf{extends}\ retracting_gears\ any\]

\[\textit{ind}\]
\[\textbf{where}\]
\[\text{grd1} : dstate[DOORS] = \{OPEN\}\]
\[\text{grd2} : gstate[GEARS] = \{EXTENDED\}\]
\[\text{grd3} : p = R\]
\[\text{grd6} : gear_{\text{extended}} = \{a \mapsto b | a \in 1..3 \land b \in GEARS \rightarrow \{TRUE\}\}\]
\[\text{grd7} : gear_{\text{retracted}} = \{a \mapsto b | a \in 1..3 \land b \in GEARS \rightarrow \{FALSE\}\}\]
\[\text{grd8} : gear_{\text{shock Absorber}} = \{a \mapsto b | a \in 1..3 \land b = \text{ground}\}\]
\[\text{grd9} : \forall x \cdot x \in 1..3 \Rightarrow handle(x) = button\]
\[\text{grd10} : at \neq \emptyset\]
\[\text{grd11} : time \in \text{ran}(at)\]
\[\text{grd12} : \text{ind} \in \text{dom}(at) \land \text{ind} \mapsto \text{time} \in at\]

\[\text{then}\]
\[\text{act1} : gstate := \{a \mapsto b | a \in GEARS \land b = \text{RETRACTING}\}\]
\[\text{act2} : gear_{\text{extended}} :\in 1..3 \rightarrow \{GEARS \rightarrow \{FALSE\}\}\]
\[\text{act3} : gear_{\text{shock Absorber}} := \{a \mapsto b | a \in 1..3 \land b = \text{flight}\}\]
\[\text{act4} : at := at \setminus \{\text{ind} \mapsto \text{time}\}\]

\[\textbf{end}\]
\[\textbf{Event retracting_gears} \equiv\]
\[\textbf{extends}\ retracting_gears\ any\]

\[\textit{ind}\]
\[\textbf{where}\]
\[\text{grd1} : dstate[DOORS] = \{OPEN\}\]
\[\text{grd2} : gstate[GEARS] = \{RETRACTING\}\]
\[\text{grd4} : gear_{\text{extended}} = \{a \mapsto b | a \in 1..3 \land b \in GEARS \rightarrow \{FALSE\}\}\]
\[\text{grd5} : gear_{\text{retracted}} = \{a \mapsto b | a \in 1..3 \land b \in GEARS \rightarrow \{FALSE\}\}\]
\[\text{grd6} : gear_{\text{shock Absorber}} = \{a \mapsto b | a \in 1..3 \land b = \text{flight}\}\]
\[\text{grd7} : \forall x \cdot x \in 1..3 \Rightarrow handle(x) = button\]
\[\text{grd3} : SGCylinder = \{a \mapsto b | a \in GEARS \times CYLINDER \land b = \text{STOP}\}\]

\[\text{then}\]
\[\text{act1} : gstate := \{a \mapsto b | a \in GEARS \land b = \text{RETRACTED}\}\]
\[\text{act2} : gear_{\text{retracted}} :\in 1..3 \rightarrow \{GEARS \rightarrow \{TRUE\}\}\]

\[\textbf{end}\]
\[\textbf{Event extending_gears} \equiv\]
\[\textbf{extends}\ extending_gears\ any\]

\[\textit{ind}\]
\[\textbf{where}\]
\[\text{grd1} : dstate[DOORS] = \{OPEN\}\]
\text{grd2} : gstate[GEARS] = \{\text{RETRACTED}\}
\text{grd3} : p = E
\text{grd5} : gear\_retracted = \{a \mapsto b | a \in 1..3 \land b \in GEARS \rightarrow \{\text{TRUE}\}\}
\text{grd6} : gear\_extended = \{a \mapsto b | a \in 1..3 \land b \in GEARS \rightarrow \{\text{FALSE}\}\}
\text{grd7} : gear\_shock\_absorber = \{a \mapsto b | a \in 1..3 \land b = \text{flight}\}
\text{grd8} : \forall x \cdot x \in 1..3 \Rightarrow \text{handle}(x) = \text{button}
\text{grd4} : SGCylinder = \{a \mapsto b | a \in \text{GEARS} \times \text{CYLINDER} \land b = \text{MOVING}\}
\text{grd9} : at \neq \emptyset
\text{grd10} : time \in \text{ran}(at)
\text{grd11} : \text{ind} \in \text{dom}(at) \land \text{ind} \mapsto \text{time} \in at
\text{then}
\text{act1} : gstate := \{a \mapsto b | a \in \text{GEARS} \land b = \text{EXTENDING}\}
\text{act2} : gear\_retracted := \{1..3 \rightarrow (\text{GEARS} \rightarrow \{\text{FALSE}\})\}
\text{act3} : at := at \cup \{\text{ind} \mapsto \text{time}\}
\text{end}
\text{Event} \text{ extension} \cong \text{extends} \text{ extension}
\text{when}
\text{grd1} : dstate[DOORS] = \{\text{OPEN}\}
\text{grd2} : gstate[GEARS] = \{\text{EXTENDING}\}
\text{grd4} : gear\_retracted = \{a \mapsto b | a \in 1..3 \land b \in \text{GEARS} \rightarrow \{\text{FALSE}\}\}
\text{grd5} : gear\_extended = \{a \mapsto b | a \in 1..3 \land b \in \text{GEARS} \rightarrow \{\text{FALSE}\}\}
\text{grd6} : gear\_shock\_absorber = \{a \mapsto b | a \in 1..3 \land b = \text{flight}\}
\text{grd7} : \forall x \cdot x \in 1..3 \Rightarrow \text{handle}(x) = \text{button}
\text{grd3} : SGCylinder = \{a \mapsto b | a \in \text{GEARS} \times \text{CYLINDER} \land b = \text{STOP}\}
\text{then}
\text{act1} : gstate := \{a \mapsto b | a \in \text{GEARS} \land b = \text{EXTENDED}\}
\text{act2} : gear\_extended := \{1..3 \rightarrow (\text{GEARS} \rightarrow \{\text{TRUE}\})\}
\text{act3} : gear\_shock\_absorber := \{a \mapsto b | a \in 1..3 \land b = \text{ground}\}
\text{end}
\text{Event} \text{ HPD1} \cong \text{extends} \text{ HPD1}
\text{when}
\text{grd3} : \forall x \cdot x \in 1..3 \Rightarrow \text{handle}(x) = \text{UP}
\text{then}
\text{act2} : \text{handle} := \{1..3 \rightarrow \{\text{DOWN}\}\}
\text{act3} : at := at \cup \{(\text{index} + 1) \mapsto (\text{time} + 160)\}
\text{analagical switch is seen open 160ms after handle position has changed}
\text{act4} : \text{handleDown\_interval} := \text{time} + 40000
\text{add a new time interval (current time + handle not changed interval)}
\text{in the event activation set}
\text{act5} : \text{handleUp\_interval} := 0
\text{update the handle up interval as 0}
\text{act6} : \text{index} := \text{index} + 1
\text{update the current index value}
\text{end}
\text{Event} \text{ HPUI1} \cong
extends $HPUI$

when

$grd3 : \forall x \cdot x \in 1..3 \Rightarrow \text{handle}(x) = DOWN$

then

$act2 : \text{handle} : \in 1..3 \rightarrow \{UP\} $

$act3 : \text{at} := \text{at} \cup \{(index + 1) \mapsto (\text{time} + 160)\}$

analogue switch is seen open 160ms after handle position has changed

$act4 : \text{handleUp_interval} := \text{time} + 40000$

add a new time interval (current time + handle not changed interval)

in the event activation set

$act5 : \text{handleDown_interval} := 0$

update the handle down interval as 0

$act6 : \text{index} := \text{index} + 1$

update the current index value

end

Event $\text{Analogical\_switch\_closed} \triangleq$

extends $\text{Analogical\_switch\_closed}$

any

$in$ in port

$ind$

where

$grd1 : \text{in} = \text{general\_EV}$

$grd2 : \forall x \cdot x \in 1..3 \Rightarrow (\text{handle}(x) = UP \lor \text{handle}(x) = DOWN)$

$grd3 : \text{at} \neq \emptyset$

$grd4 : \text{time} \in \text{ran}(\text{at})$

$grd5 : \text{ind} \in \text{dom}(\text{at}) \land \text{ind} \mapsto \text{time} \in \text{at}$

then

$act3 : \text{analogue\_switch} : \in 1..3 \rightarrow \{\text{closed}\} $

$act4 : \text{A\_Switch\_Out} := \text{TRUE}$

$act5 : \text{at} := (\text{at} \cup \{(index + 1) \mapsto (\text{time} + 1200)\}) \setminus \{\text{ind} \mapsto \text{time}\}$

from closed to open 1.2 sec.

$act6 : \text{index} := \text{index} + 1$

end

Event $\text{Analogical\_switch\_open} \triangleq$

extends $\text{Analogical\_switch\_open}$

any

$in$ in port

$ind$

where

$grd1 : \text{in} = \text{general\_EV}$

$grd2 : \forall x \cdot x \in 1..3 \Rightarrow (\text{handle}(x) = UP \lor \text{handle}(x) = DOWN)$

$grd3 : \text{at} \neq \emptyset$

$grd4 : \text{time} \in \text{ran}(\text{at})$

$grd5 : \text{ind} \in \text{dom}(\text{at}) \land \text{ind} \mapsto \text{time} \in \text{at}$

then

$act3 : \text{analogue\_switch} : \in 1..3 \rightarrow \{\text{open}\} $

$act4 : \text{A\_Switch\_Out} := \text{FALSE}$
act5 : \( at := (at \cup \{(index + 1) \mapsto (time + 800)\}) \setminus \{ind \mapsto time\} \)
from open to closed .8 sec.

act6 : \( index := index + 1 \)

Event Circuit\_pressurized\_OK \( \triangleq \)
extends Circuit\_pressurized\_OK
when

grd1 : \( general\_EV\_Hout = Hin \)

end

act9 : \( circuit\_pressurized \in 1..3 \mapsto \{TRUE\} \)

Event Circuit\_pressurized\_notOK \( \triangleq \)
extends Circuit\_pressurized\_notOK
when

grd1 : \( general\_EV\_Hout = 0 \)

end

Event Computing\_Module\_1,2 \( \triangleq \)
extends Computing\_Module\_1,2
when

grd1 : state = computing
then

act1 : \( general\_EV := general\_EV\_func(handle \mapsto \text{analogical\_switch} \mapsto \text{gear\_extended} \mapsto \text{gear\_retracted} \mapsto \text{gear\_shock\_absorber} \mapsto \text{door\_open} \mapsto \text{door\_closed} \mapsto \text{circuit\_pressurized}) \)
act2 : \( close\_EV := \text{close\_EV\_func(handle \mapsto \text{analogical\_switch} \mapsto \text{gear\_extended} \mapsto \text{gear\_retracted} \mapsto \text{gear\_shock\_absorber} \mapsto \text{door\_open} \mapsto \text{door\_closed} \mapsto \text{circuit\_pressurized}) \)
act3 : \( retract\_EV := \text{retract\_EV\_func(handle \mapsto \text{analogical\_switch} \mapsto \text{gear\_extended} \mapsto \text{gear\_retracted} \mapsto \text{gear\_shock\_absorber} \mapsto \text{door\_open} \mapsto \text{door\_closed} \mapsto \text{circuit\_pressurized}) \)
act4 : \( extend\_EV := \text{extend\_EV\_func(handle \mapsto \text{analogical\_switch} \mapsto \text{gear\_extended} \mapsto \text{gear\_retracted} \mapsto \text{gear\_shock\_absorber} \mapsto \text{door\_open} \mapsto \text{door\_closed} \mapsto \text{circuit\_pressurized}) \)
act5 : \( open\_EV := \text{open\_EV\_func(handle \mapsto \text{analogical\_switch} \mapsto \text{gear\_extended} \mapsto \text{gear\_retracted} \mapsto \text{gear\_shock\_absorber} \mapsto \text{door\_open} \mapsto \text{door\_closed} \mapsto \text{circuit\_pressurized}) \)
act6 : \( gears\_locked\_down := \text{gears\_locked\_down\_func(handle \mapsto \text{analogical\_switch} \mapsto \text{gear\_extended} \mapsto \text{gear\_retracted} \mapsto \text{gear\_shock\_absorber} \mapsto \text{door\_open} \mapsto \text{door\_closed} \mapsto \text{circuit\_pressurized}) \)
act7 : \( gears\_man := \text{gears\_man\_func(handle \mapsto \text{analogical\_switch} \mapsto \text{gear\_extended} \mapsto \text{gear\_retracted} \mapsto \text{gear\_shock\_absorber} \mapsto \text{door\_open} \mapsto \text{door\_closed} \mapsto \text{circuit\_pressurized}) \)
act8 : \( anomaly := \text{anomaly\_func(handle \mapsto \text{analogical\_switch} \mapsto \text{gear\_extended} \mapsto \text{gear\_retracted} \mapsto \text{gear\_shock\_absorber} \mapsto \text{door\_open} \mapsto \text{door\_closed} \mapsto \text{circuit\_pressurized}) \)
act9 : \( state := \text{electroValve} \)
Event \( \text{Update\_Hout} \) \( \triangleq \) Assign the value of Hout

extends \( \text{Update\_Hout} \)

when

\begin{align*}
\text{grd1} & : \text{state} = \text{electroValve} \\
\text{then}
\end{align*}

\begin{align*}
\text{act1} & : \text{general\_EV\_Hout} : \left( (\text{general\_EV} = \text{TRUE} \land \text{general\_EV\_Hout}' = \text{Hin}) \lor (\text{general\_EV} = \text{FALSE} \land \text{general\_EV\_Hout}' = 0) \lor (\text{A\_Switch\_Out} = \text{TRUE} \land \text{general\_EV\_Hout}' = \text{Hin}) \lor (\text{A\_Switch\_Out} = \text{FALSE} \land \text{general\_EV\_Hout}' = 0) \right) \\
& \quad \text{pass the current value of hydraulic input port (Hin) to hydraulic output port (Hout)}
\end{align*}

\begin{align*}
\text{act2} & : \text{close\_EV\_Hout} : \left( (\text{close\_EV} = \text{TRUE} \land \text{close\_EV\_Hout}' = \text{Hin}) \lor (\text{close\_EV} = \text{FALSE} \land \text{close\_EV\_Hout}' = 0) \right) \\
\text{act3} & : \text{open\_EV\_Hout} : \left( (\text{open\_EV} = \text{TRUE} \land \text{open\_EV\_Hout}' = \text{Hin}) \lor (\text{open\_EV} = \text{FALSE} \land \text{open\_EV\_Hout}' = 0) \right) \\
\text{act4} & : \text{extend\_EV\_Hout} : \left( (\text{extend\_EV} = \text{TRUE} \land \text{extend\_EV\_Hout}' = \text{Hin}) \lor (\text{extend\_EV} = \text{FALSE} \land \text{extend\_EV\_Hout}' = 0) \right) \\
\text{act5} & : \text{retract\_EV\_Hout} : \left( (\text{retract\_EV} = \text{TRUE} \land \text{retract\_EV\_Hout}' = \text{Hin}) \lor (\text{retract\_EV} = \text{FALSE} \land \text{retract\_EV\_Hout}' = 0) \right) \\
\text{act6} & : \text{state} := \text{cylinder} \\
\text{act7} & : \text{at} := \text{at} \cup \\
& \quad \{ (\text{index} + 1) \mapsto (\text{time} + 2000) \} \cup \\
& \quad \{ (\text{index} + 2) \mapsto (\text{time} + 10000) \} \cup \\
& \quad \{ (\text{index} + 3) \mapsto (\text{time} + 500) \} \cup \\
& \quad \{ (\text{index} + 4) \mapsto (\text{time} + 2000) \} \cup \\
& \quad \{ (\text{index} + 5) \mapsto (\text{time} + 500) \} \cup \\
& \quad \{ (\text{index} + 6) \mapsto (\text{time} + 2000) \} \cup \\
& \quad \{ (\text{index} + 7) \mapsto (\text{time} + 500) \} \cup \\
& \quad \{ (\text{index} + 8) \mapsto (\text{time} + 10000) \} \cup \\
& \quad \{ (\text{index} + 9) \mapsto (\text{time} + 500) \} \cup \\
& \quad \{ (\text{index} + 10) \mapsto (\text{time} + 10000) \} \\
\end{align*}

\begin{align*}
\text{general EV} 2 \text{ (time is given in comments in sec. while in model these are in ms.)}
\text{general EV} 10 \\
\text{opening EV} 0.5 \\
\text{opening EV} 2 \\
\text{closure EV} 0.5 \\
\text{closure EV} 2 \\
\text{retraction EV} 0.5 \\
\text{retraction EV} 10 \\
\text{extension} 0.5 \\
\text{extension 10}
\end{align*}

\text{act8} : \text{index} := \text{index} + 10

Event \( \text{CylinderMovingOrStop} \) \( \triangleq \) Cylinder Moving or Stop according to the output of hydraulic circuit

extends \( \text{CylinderMovingOrStop} \)
when

grd1 : state = cylinder
then

act1 : SGCylinder : 
[functionality]
(SGCylinder′ = \{a \mapsto b|a \in GEARS \times \{GCY F, GCY R, GCY L\} \land b = MOVING\} \land extend_EV_Hout = Hin) \lor
(SGCylinder′ = \{a \mapsto b|a \in GEARS \times \{GCY F, GCY R, GCY L\} \land b = STOP\} \land extend_EV_Hout = 0) \lor
(SGCylinder′ = \{a \mapsto b|a \in GEARS \times \{GCY F, GCY R, GCY L\} \land b = MOVING\} \land retract_EV_Hout = Hin) \lor
(SGCylinder′ = \{a \mapsto b|a \in GEARS \times \{GCY F, GCY R, GCY L\} \land b = STOP\} \land retract_EV_Hout = 0)

act2 : SDCylinder : 
[functionality]
(SDCylinder′ = \{a \mapsto b|a \in DOORS \times \{DCY F, DCY R, DCY L\} \land b = MOVING\} \land open_EV_Hout = Hin) \lor
(SDCylinder′ = \{a \mapsto b|a \in DOORS \times \{DCY F, DCY R, DCY L\} \land b = STOP\} \land open_EV_Hout = 0) \lor
(SDCylinder′ = \{a \mapsto b|a \in DOORS \times \{DCY F, DCY R, DCY L\} \land b = MOVING\} \land close_EV_Hout = Hin) \lor
(SDCylinder′ = \{a \mapsto b|a \in DOORS \times \{DCY F, DCY R, DCY L\} \land b = STOP\} \land close_EV_Hout = 0)

act3 : state := computing

end

Event Failure_Detection_Generic_Monitoring \deq
extends Failure_Detection_Generic_Monitoring
when


\[ \text{generic Monitoring using all sensors} \]

\[ \text{then} \]

\[ \text{act1} : \text{anomaly := TRUE} \]

\[ \text{end} \]

\[ \text{Event Failure Detection Analogical Switch} \]

\[ \text{extends} \]

\[ \text{Failure Detection Analogical Switch} \]

\[ \text{any} \]

\[ \text{ind} \]

\[ \text{where} \]

\[ \text{grd1} : \text{analogical_switch} = \{ a \mapsto b | a \in 1..3 \land b = \text{open} \} \]

\[ \lor \]

\[ \text{analogical_switch} = \{ a \mapsto b | a \in 1..3 \land b = \text{closed} \} \]

\[ \text{Gears motion monitoring without considering time} \]

\[ \text{grd2} : \text{at} \neq \emptyset \]

\[ \text{grd3} : \text{time} \in \text{ran(at)} \]
grd4 : ind ∈ dom(at) ∧ ind ↦→ time ∈ at

then

act1 : anomaly := TRUE
act2 : at := at \ \{ind ↦→ time\}

end

Event check\_handle\_delay ≡

This event is used to set 280ms in the set ”at”
for event activation to detect anomaly
and detect that handle is not change from last 40 sec.

when

grd1 : time = handleUp\_interval
\lor
\quad time = handleDown\_interval

current time is either equal to handle up interval or equal to the handle down interval

then

act1 : at := at \cup \{(index + 1) ↦→ (time + 280)\}

To add a new interval to the event activation set

act3 : index := index + 1

update the current index value

end

Event Failure\_Detection\_Pressure\_Sensor ≡

extends Failure\_Detection\_Pressure\_Sensor

any

where

ind

grd1 : circuit\_pressurized \neq \{a ↦→ b | a ∈ 1..3 ∧ b = TRUE\}
\lor
\quad circuit\_pressurized \neq \{a ↦→ b | a ∈ 1..3 ∧ b = FALSE\}

Circuit pressurized motion monitoring without considering time

grd2 : at \neq ∅
grd3 : time ∈ ran(at)
grd4 : ind ∈ dom(at) ∧ ind ↦→ time ∈ at

then

act1 : anomaly := TRUE
act2 : at := at \backslash \{ind ↦→ time\}

end

Event Failure\_Detection\_Doors ≡

extends Failure\_Detection\_Doors

any

where

ind
Doors motion monitoring without considering time

\[ \text{grd1: door\_closed} \neq \{ a \mapsto b \mid a \in 1..3 \land b \in DOORS \rightarrow \{ \text{FALSE} \} \} \]
\[ \lor \]
\[ \text{door\_open} \neq \{ a \mapsto b \mid a \in 1..3 \land b \in DOORS \rightarrow \{ \text{TRUE} \} \} \]
\[ \lor \]
\[ \text{door\_open} \neq \{ a \mapsto b \mid a \in 1..3 \land b \in DOORS \rightarrow \{ \text{FALSE} \} \} \]
\[ \lor \]
\[ \text{door\_closed} \neq \{ a \mapsto b \mid a \in 1..3 \land b \in DOORS \rightarrow \{ \text{TRUE} \} \} \]

Gears motion monitoring without considering time

\[ \text{grd2: at} \neq \emptyset \]
\[ \text{grd3: time} \in \text{ran}(at) \]
\[ \text{grd4: ind} \in \text{dom}(at) \land \text{ind} \mapsto \text{time} \in at \]

then

\[ \text{act1: anomaly} := \text{TRUE} \]
\[ \text{act2: at} := at \setminus \{ \text{ind} \mapsto \text{time} \} \]
end

Event Failure\_Detection\_Gears \equiv
extends Failure\_Detection\_Gears
any
\[ \text{ind} \]
where

\[ \text{grd1: gear\_retracted} \neq \{ a \mapsto b \mid a \in 1..3 \land b \in GEARS \rightarrow \{ \text{FALSE} \} \} \]
\[ \lor \]
\[ \text{gear\_retracted} \neq \{ a \mapsto b \mid a \in 1..3 \land b \in GEARS \rightarrow \{ \text{TRUE} \} \} \]
\[ \lor \]
\[ \text{gear\_extended} \neq \{ a \mapsto b \mid a \in 1..3 \land b \in GEARS \rightarrow \{ \text{FALSE} \} \} \]
\[ \lor \]
\[ \text{gear\_extended} \neq \{ a \mapsto b \mid a \in 1..3 \land b \in GEARS \rightarrow \{ \text{TRUE} \} \} \]

Gears motion monitoring without considering time

\[ \text{grd2: at} \neq \emptyset \]
\[ \text{grd3: time} \in \text{ran}(at) \]
\[ \text{grd4: ind} \in \text{dom}(at) \land \text{ind} \mapsto \text{time} \in at \]

then

\[ \text{act1: anomaly} := \text{TRUE} \]
\[ \text{act2: at} := at \setminus \{ \text{ind} \mapsto \text{time} \} \]
end

Event tic\_tock \equiv

\[ \text{tm} \]

\[ \text{grd1: tm} \in \mathbb{N} \]
\[ \text{grd2: tm} > \text{time} \]

\[ \text{to take a new value of time in the future} \]
\[ \text{grd3: ran}(at) \neq \emptyset \Rightarrow \text{tm} \leq \text{min}(\text{ran}(at)) \]

then

\[ \text{act1: time} := \text{tm} \]

\[ \text{assign a new value of time to the current time} \]
MACHINE M9
     Pilot interface light implementation
REFINES M8
SEES C1, C2
VARIABLES
     dstate
     lstate
     phase
     button
     p
     l
     i
     gstate
     handle
     analogical_switch
     gearextended
     gearretracted
     gearshock_absorber
     door_closed
     door_open
     circuit_pressurized
     general_EV
     close_EV
     retract_EV
     extend_EV
     open_EV
     gears_locked_down
     gears_man
     anomaly
     general_EV_func
     close_EV_func
     retract_EV_func
     extend_EV_func
     open_EV_func
     gears_locked_down_func
     gears_man_func
anomaly_func

gen general_EV_Hout
close_EV_Hout
retract_EV_Hout
extend_EV_Hout
open_EV_Hout

SDCylinder State of Door Cylinder
SGCylinder State of Gear Cylinder
A_Switch_Out State of Gear Cylinder
state

time current time
at a future event activation set.
index To take a function to index different sets for event activation set
handleUp_interval To keep an update time duration after handle up
handleDown_interval To keep an update time duration after handle down

pilot_interface_light current pilot interface light

INVIANTS

inv1 : pilot_interface_light ∈ colorSet → lightState
a function to map from colorset to light state

EVENTS
Initialisation
extended

begin

act1 : button := DOWN
act2 : phase := haltdown
act3 : dstate : [(dstate' ∈ DOORS → SDOORS ∧ dstate' = {a ↦ b|a ∈ DOORS ∧ b = CLOSED})]
               missing elements of the invariant
act4 : lstate := {a ↦ b|a ∈ DOORS ∧ b = LOCKED}
act5 : p := R
act6 : l := R
act7 : i := R
act8 : gstate : [(gstate' ∈ GEARs → SGEARS ∧ gstate' = {a ↦ b|a ∈ GEARs ∧ b = EXTENDED})]
act14 : handle ∈ 1..3 → {DOWN}
act15 : analogical_switch ∈ 1..3 → {open}
act16 : gear_extended ∈ 1..3 → (GEARS → {TRUE})
act17 : gear_retracted ∈ 1..3 → (GEARS → {FALSE})
act18 : gear_shock_absorber ∈ 1..3 → {ground}
act19 : door_closed ∈ 1..3 → (DOORS → {TRUE})
act20 : door_open ∈ 1..3 → (DOORS → {FALSE})
act21 : circuit_pressurized ∈ 1..3 → {FALSE}
act22 : general_EV := FALSE
act23 : close_EV := TRUE
act24 : retract_EV := FALSE
act25 : extend_EV := TRUE
\begin{verbatim}
act27 : open_EV := FALSE
act28 : gears_locked_down := TRUE
act29 : gears_man := FALSE
act30 : anomaly := FALSE
act31 : general_EV_func := (1 .. 3 \rightarrow \text{POSITIONS}) \times (1 .. 3 \rightarrow \text{A_Switch}) \times (1 .. 3 \rightarrow (\text{GEARS} \rightarrow \text{BOOL})) \times (1 .. 3 \rightarrow (\text{GEARS} \rightarrow \text{BOOL})) \times (1 .. 3 \rightarrow (\text{GEAR_ABSORBER}) \times (1 .. 3 \rightarrow (\text{DOORS} \rightarrow \text{BOOL})) \times (1 .. 3 \rightarrow (\text{DOORS} \rightarrow \text{BOOL})) \times (1 .. 3 \rightarrow \text{BOOL}) \rightarrow \text{BOOL}
act32 : close_EV_func := (1 .. 3 \rightarrow \text{POSITIONS}) \times (1 .. 3 \rightarrow \text{A_Switch}) \times (1 .. 3 \rightarrow (\text{GEARS} \rightarrow \text{BOOL})) \times (1 .. 3 \rightarrow (\text{GEARS} \rightarrow \text{BOOL})) \times (1 .. 3 \rightarrow (\text{GEAR_ABSORBER}) \times (1 .. 3 \rightarrow (\text{DOORS} \rightarrow \text{BOOL})) \times (1 .. 3 \rightarrow (\text{DOORS} \rightarrow \text{BOOL})) \times (1 .. 3 \rightarrow \text{BOOL}) \rightarrow \text{BOOL}
act33 : retract_EV_func := (1 .. 3 \rightarrow \text{POSITIONS}) \times (1 .. 3 \rightarrow \text{A_Switch}) \times (1 .. 3 \rightarrow (\text{GEARS} \rightarrow \text{BOOL})) \times (1 .. 3 \rightarrow (\text{GEARS} \rightarrow \text{BOOL})) \times (1 .. 3 \rightarrow (\text{GEAR_ABSORBER}) \times (1 .. 3 \rightarrow (\text{DOORS} \rightarrow \text{BOOL})) \times (1 .. 3 \rightarrow (\text{DOORS} \rightarrow \text{BOOL})) \times (1 .. 3 \rightarrow \text{BOOL}) \rightarrow \text{BOOL}
act34 : extend_EV_func := (1 .. 3 \rightarrow \text{POSITIONS}) \times (1 .. 3 \rightarrow \text{A_Switch}) \times (1 .. 3 \rightarrow (\text{GEARS} \rightarrow \text{BOOL})) \times (1 .. 3 \rightarrow (\text{GEARS} \rightarrow \text{BOOL})) \times (1 .. 3 \rightarrow (\text{GEAR_ABSORBER}) \times (1 .. 3 \rightarrow (\text{DOORS} \rightarrow \text{BOOL})) \times (1 .. 3 \rightarrow (\text{DOORS} \rightarrow \text{BOOL})) \times (1 .. 3 \rightarrow \text{BOOL}) \rightarrow \text{BOOL}
act35 : open_EV_func := (1 .. 3 \rightarrow \text{POSITIONS}) \times (1 .. 3 \rightarrow \text{A_Switch}) \times (1 .. 3 \rightarrow (\text{GEARS} \rightarrow \text{BOOL})) \times (1 .. 3 \rightarrow (\text{GEARS} \rightarrow \text{BOOL})) \times (1 .. 3 \rightarrow (\text{GEAR_ABSORBER}) \times (1 .. 3 \rightarrow (\text{DOORS} \rightarrow \text{BOOL})) \times (1 .. 3 \rightarrow (\text{DOORS} \rightarrow \text{BOOL})) \times (1 .. 3 \rightarrow \text{BOOL}) \rightarrow \text{BOOL}
act36 : gears_locked_down_func := (1 .. 3 \rightarrow \text{POSITIONS}) \times (1 .. 3 \rightarrow \text{A_Switch}) \times (1 .. 3 \rightarrow (\text{GEARS} \rightarrow \text{BOOL})) \times (1 .. 3 \rightarrow (\text{GEARS} \rightarrow \text{BOOL})) \times (1 .. 3 \rightarrow (\text{GEAR_ABSORBER}) \times (1 .. 3 \rightarrow (\text{DOORS} \rightarrow \text{BOOL})) \times (1 .. 3 \rightarrow (\text{DOORS} \rightarrow \text{BOOL})) \times (1 .. 3 \rightarrow \text{BOOL}) \rightarrow \text{BOOL}
act37 : gears_man_func := (1 .. 3 \rightarrow \text{POSITIONS}) \times (1 .. 3 \rightarrow \text{A_Switch}) \times (1 .. 3 \rightarrow (\text{GEARS} \rightarrow \text{BOOL})) \times (1 .. 3 \rightarrow (\text{GEARS} \rightarrow \text{BOOL})) \times (1 .. 3 \rightarrow (\text{GEAR_ABSORBER}) \times (1 .. 3 \rightarrow (\text{DOORS} \rightarrow \text{BOOL})) \times (1 .. 3 \rightarrow (\text{DOORS} \rightarrow \text{BOOL})) \times (1 .. 3 \rightarrow \text{BOOL}) \rightarrow \text{BOOL}
act38 : anomaly_func := (1 .. 3 \rightarrow \text{POSITIONS}) \times (1 .. 3 \rightarrow \text{A_Switch}) \times (1 .. 3 \rightarrow (\text{GEARS} \rightarrow \text{BOOL})) \times (1 .. 3 \rightarrow (\text{GEARS} \rightarrow \text{BOOL})) \times (1 .. 3 \rightarrow (\text{GEAR_ABSORBER}) \times (1 .. 3 \rightarrow (\text{DOORS} \rightarrow \text{BOOL})) \times (1 .. 3 \rightarrow (\text{DOORS} \rightarrow \text{BOOL})) \times (1 .. 3 \rightarrow \text{BOOL}) \rightarrow \text{BOOL}
act39 : A_Switch_Out := FALSE
act40 : close_EV_Hout := 0
act41 : open_EV_Hout := 0
act42 : extend_EV_Hout := 0
act43 : general_EV_Hout := 0
act45 : SDCylinder := \{\text{DOORS} \times \{\text{DCYF}, \text{DCYR}, \text{DCYL}\} \rightarrow \{\text{STOP}\}
act46 : SGCylinder := \{\text{GEARS} \times \{\text{GCYF}, \text{GCYR}, \text{GCYL}\} \rightarrow \{\text{STOP}\}
act26 : state := computing
act47 : at := \emptyset
act48 : time := 0
act49 : index := 0
act50 : handleUp_interval := 0
act51 : handleDown_interval := 0
\end{verbatim}
act52 : pilot_interface_light ::= {Green $\mapsto$ Off, Orange $\mapsto$ Off, Red $\mapsto$ Off}

end

Event opening_doors_DOWN $\cong$
events opening_doors_DOWN when

grd1 : dstate[DOORS] = {CLOSED}
grd5 : lstate[DOORS] = {UNLOCKED}
grd7 : phase = movingdown

grd8 : p = R

grd9 : l = R

grd10 : door_open = \{a $\mapsto$ b\mid a \in 1..3 \land b \in DOORS \rightarrow \{FALSE\}\}
grd11 : door_closed = \{a $\mapsto$ b\mid a \in 1..3 \land b \in DOORS \rightarrow \{FALSE\}\}
grd12 : \forall x \cdot x \in 1..3 \Rightarrow handle(x) = button

grd3 : SDCylinder = \{a $\mapsto$ b\mid a \in DOORS \times CYLINDER \land b = MOVING\}
grd13 : anomaly = FALSE

then

act1 : dstate ::= \{a $\mapsto$ b\mid a \in DOORS \land b = OPEN\}
act2 : p := E
act3 : door_open ::= 1..3 \rightarrow (DOORS \rightarrow \{TRUE\})
act4 : at ::= at \cup \{(index + 1) \mapsto (time + 100)\}
minimal interval for door open to gear extension

end

Event opening_doors_UP $\cong$
events opening_doors_UP when

grd1 : dstate[DOORS] = {CLOSED}
grd4 : lstate[DOORS] = {UNLOCKED}
grd5 : phase = movingup

grd6 : p = E

grd7 : l = E

grd8 : door_open = \{a $\mapsto$ b\mid a \in 1..3 \land b \in DOORS \rightarrow \{FALSE\}\}
grd9 : door_closed = \{a $\mapsto$ b\mid a \in 1..3 \land b \in DOORS \rightarrow \{FALSE\}\}
grd10 : \forall x \cdot x \in 1..3 \Rightarrow handle(x) = button

grd3 : SDCylinder = \{a $\mapsto$ b\mid a \in DOORS \times CYLINDER \land b = MOVING\}
grd11 : anomaly = FALSE

then

act1 : dstate ::= \{a $\mapsto$ b\mid a \in DOORS \land b = OPEN\}
act2 : p := R
act3 : door_open ::= 1..3 \rightarrow (DOORS \rightarrow \{TRUE\})
act4 : at ::= at \cup \{(index + 1) \mapsto (time + 100)\}
minimal interval for door open to gear retraction

end

Event closing_doors_UP $\cong$
events closing_doors_UP any
\[ f \]

where

\[ \text{grd1} : \text{dstate}[\text{DOORS}] = \{\text{OPEN}\} \]
\[ \text{grd3} : f \in \text{DOORS} \rightarrow \text{SDOORS} \]
\[ \text{grd4} : \forall e \cdot e \in \text{DOORS} \Rightarrow f(e) = \text{CLOSED} \]
\[ \text{grd5} : \text{phase} = \text{movingup} \]
\[ \text{grd6} : p = E \]
\[ \text{grd7} : \text{gstate}[\text{GEARS}] = \{\text{RETRACTED}\} \]
\[ \text{grd8} : \forall x \cdot x \in 1..3 \Rightarrow \text{handle}(x) = \text{button} \]
\[ \text{grd9} : \text{anomaly} = \text{FALSE} \]

then

\[ \text{act1} : \text{dstate} := f \]

end

Event \text{closing\_doors\_DOWN} \equiv
extends \text{closing\_doors\_DOWN}  

any

\[ f \]

where

\[ \text{grd1} : \text{dstate}[\text{DOORS}] = \{\text{OPEN}\} \]
\[ \text{grd3} : f \in \text{DOORS} \rightarrow \text{SDOORS} \]
\[ \text{grd4} : \forall e \cdot e \in \text{DOORS} \Rightarrow f(e) = \text{CLOSED} \]
\[ \text{grd5} : \text{phase} = \text{movingdown} \]
\[ \text{grd6} : p = E \]
\[ \text{grd7} : \text{gstate}[\text{GEARS}] = \{\text{EXTENDED}\} \]
\[ \text{grd8} : \forall x \cdot x \in 1..3 \Rightarrow \text{handle}(x) = \text{button} \]
\[ \text{grd9} : \text{anomaly} = \text{FALSE} \]

then

\[ \text{act1} : \text{dstate} := f \]

end

Event \text{unlocking\_UP} \equiv
extends \text{unlocking\_UP}  

when

\[ \text{grd3} : \text{lstate}[\text{DOORS}] = \{\text{LOCKED}\} \]
\[ \text{grd4} : \text{phase} = \text{movingup} \]
\[ \text{grd5} : l = E \]
\[ \text{grd6} : p = E \]
\[ \text{grd7} : i = E \]
\[ \text{grd8} : \text{door\_open} = \{a \mapsto b \mid a \in 1..3 \land b \in \text{DOORS} \rightarrow \{\text{FALSE}\}\} \]
\[ \text{grd9} : \text{door\_closed} = \{a \mapsto b \mid a \in 1..3 \land b \in \text{DOORS} \rightarrow \{\text{TRUE}\}\} \]
\[ \text{grd10} : \forall x \cdot x \in 1..3 \Rightarrow \text{handle}(x) = \text{button} \]
\[ \text{grd11} : \text{anomaly} = \text{FALSE} \]

then

\[ \text{act1} : \text{lstate} := \{a \mapsto b \mid a \in \text{DOORS} \land b = \text{UNLOCKED}\} \]
\[ \text{act2} : \text{door\_closed} \in 1..3 \rightarrow (\text{DOORS} \rightarrow \{\text{FALSE}\}) \]

end

Event \text{locking\_UP} \equiv

117
extends locking\_UP
\begin{itemize}
  \item grd3 : dstate[DOORS] = \{CLOSED\}
  \item grd4 : phase = movingup
  \item grd5 : lstate[DOORS] = \{UNLOCKED\}
  \item grd6 : p = R
  \item grd7 : l = E
  \item grd9 : door\_open = \{a \rightarrow b|a \in 1..3 \land b \in DOORS \rightarrow \{FALSE\}\}
  \item grd10 : door\_closed = \{a \rightarrow b|a \in 1..3 \land b \in DOORS \rightarrow \{FALSE\}\}
  \item grd11 : \forall x \in 1..3 \Rightarrow handle(x) = button
  \item grd8 : SDCylinder = \{a \rightarrow b|a \in DOORS \times CYLINDER \land b = STOP\}
  \item grd12 : anomaly = FALSE
\end{itemize}
\textbf{then}
\begin{itemize}
  \item act1 : lstate := \{a \rightarrow b|a \in DOORS \land b = LOCKED\}
  \item act3 : phase := haltup
  \item act4 : l := R
  \hspace{1cm} added by D Mery
  \item act44 : door\_closed :\in 1..3 \rightarrow (DOORS \rightarrow \{TRUE\})
  \item act5 : at := at \cup \{(index + 1) \rightarrow (time + 100)\}
  \hspace{1cm} minimal interval for door closed to gear extension/retraction
  \item act6 : index := index + 1
\end{itemize}
\textbf{end}
\textbf{Event unlocking\_DOWN} $\triangleq$
\begin{itemize}
  \item extends unlocking\_DOWN
  \item grd3 : lstate[DOORS] = \{LOCKED\}
  \item grd4 : phase = movingdown
  \item grd5 : l = R
  \item grd6 : p = R
  \item grd7 : i = R
  \item grd8 : door\_open = \{a \rightarrow b|a \in 1..3 \land b \in DOORS \rightarrow \{FALSE\}\}
  \item grd9 : door\_closed = \{a \rightarrow b|a \in 1..3 \land b \in DOORS \rightarrow \{TRUE\}\}
  \item grd10 : \forall x \in 1..3 \Rightarrow handle(x) = button
  \item grd11 : anomaly = FALSE
\end{itemize}
\textbf{then}
\begin{itemize}
  \item act1 : lstate := \{a \rightarrow b|a \in DOORS \land b = UNLOCKED\}
  \item act2 : door\_closed :\in 1..3 \rightarrow (DOORS \rightarrow \{FALSE\})
\end{itemize}
\textbf{end}
\textbf{Event locking\_DOWN} $\triangleq$
\begin{itemize}
  \item extends locking\_DOWN
  \item grd1 : dstate[DOORS] = \{CLOSED\}
  \item grd2 : phase = movingdown
  \item grd3 : lstate[DOORS] = \{UNLOCKED\}
  \item grd4 : p = E
  \item grd5 : l = R
  \item grd7 : door\_open = \{a \rightarrow b|a \in 1..3 \land b \in DOORS \rightarrow \{FALSE\}\}
  \item grd8 : door\_closed = \{a \rightarrow b|a \in 1..3 \land b \in DOORS \rightarrow \{FALSE\}\}
\end{itemize}
\[
\forall x \cdot x \in 1..3 \Rightarrow \text{handle}(x) = \text{button}
\]
\[
\text{SDCylinder} = \{a \mapsto b \mid a \in \text{DOORS} \times \text{CYLINDER} \land b = \text{STOP}\}
\]
\[
\text{anomaly} = \text{FALSE}
\]

then

\[
\text{act1} : \text{lstate} := \{a \mapsto b \mid a \in \text{DOORS} \land b = \text{LOCKED}\}
\]
\[
\text{act3} : \text{phase} := \text{haltdown}
\]
\[
\text{act4} : l := E
\]
\[
\text{act5} : \text{door\_closed} : \in 1..3 \Rightarrow \text{DOORS} \Rightarrow \{\text{TRUE}\}
\]
\[
\text{act6} : a\_t := a\_t \cup \{(\text{index} + 1) \mapsto (\text{time} + 100)\}
\]
minimal interval for door closed to extension/retraction
\[
\text{act7} : \text{index} := \text{index} + 1
\]

end

Event \( \text{PD1} \) \( \cong \) extends \( \text{PD1} \)

when

\[
\text{grd1} : \text{button} = \text{UP}
\]
\[
\text{grd2} : \text{phase} = \text{haltup}
\]
\[
\text{grd3} : \forall x \cdot x \in 1..3 \Rightarrow \text{handle}(x) = \text{DOWN}
\]

then

\[
\text{act1} : \text{phase} := \text{movingdown}
\]
\[
\text{act2} : \text{button} := \text{DOWN}
\]
\[
\text{act3} : l := E
\]
\[
\text{act4} : p := R
\]
\[
\text{act5} : i := R
\]

end

Event \( \text{PU1} \) \( \cong \) extends \( \text{PU1} \)

when

\[
\text{grd1} : \text{button} = \text{DOWN}
\]
\[
\text{grd2} : \text{phase} = \text{haltdown}
\]
\[
\text{grd3} : \forall x \cdot x \in 1..3 \Rightarrow \text{handle}(x) = \text{UP}
\]

then

\[
\text{act1} : \text{phase} := \text{movingup}
\]
\[
\text{act2} : \text{button} := \text{UP}
\]
\[
\text{act3} : l := E
\]
\[
\text{act4} : p := E
\]
\[
\text{act5} : i := E
\]

end

Event \( \text{PU2} \) \( \cong \) extends \( \text{PU2} \)

when

\[
\text{grd1} : l = R
\]
\[
\text{grd2} : p = R
\]
\[
\text{grd3} : \text{phase} = \text{movingdown}
\]
\[
\text{grd4} : \text{button} = \text{DOWN}
\]
\[
\text{grd5} : i = R
\]
\[
\text{grd6} : \text{lstate}[\text{DOORS}] = \{\text{LOCKED}\}
\]

119
$\text{grd7} : \text{door\_open} = \{ a \mapsto b | a \in 1..3 \land b \in \text{DOORS} \rightarrow \{\text{FALSE}\} \}$

$\text{grd8} : \text{door\_closed} = \{ a \mapsto b | a \in 1..3 \land b \in \text{DOORS} \rightarrow \{\text{TRUE}\} \}$

$\text{grd9} : \forall x \cdot x \in 1..3 \Rightarrow \text{handle}(x) = UP$

then

$act1 : \text{phase} := \text{movingup}$

$act4 : \text{button} := \text{UP}$

$act5 : l := E$

$act6 : p := E$

$act7 : i := R$

end

Event $\text{CompletePU2} \sqsupset$
extends $\text{CompletePU2}$

when

$\text{grd1} : \text{phase} = \text{movingup}$

$\text{grd2} : \text{button} = \text{UP}$

$\text{grd3} : l = E$

$\text{grd4} : p = E$

$\text{grd5} : i = R$

then

$act1 : \text{phase} := \text{haltup}$

end

Event $\text{PU3} \sqsupset$
extends $\text{PU3}$

when

$\text{grd1} : \text{dstate}[\text{DOORS}] = \{\text{CLOSED}\}$

$\text{grd2} : \text{lstate}[\text{DOORS}] = \{\text{UNLOCKED}\}$

$\text{grd3} : \text{phase} = \text{movingdown}$

$\text{grd4} : p = R$

$\text{grd5} : l = R$

$\text{grd6} : \text{button} = \text{DOWN}$

$\text{grd7} : \text{door\_open} = \{ a \mapsto b | a \in 1..3 \land b \in \text{DOORS} \rightarrow \{\text{FALSE}\} \}$

$\text{grd8} : \text{door\_closed} = \{ a \mapsto b | a \in 1..3 \land b \in \text{DOORS} \rightarrow \{\text{FALSE}\} \}$

$\text{grd9} : \forall x \cdot x \in 1..3 \Rightarrow \text{handle}(x) = UP$

then

$act1 : \text{phase} := \text{movingup}$

$act2 : p := R$

$act3 : l := E$

$act4 : \text{button} := \text{UP}$

end

Event $\text{PU4} \sqsupset$
extends $\text{PU4}$

when

$\text{grd1} : \text{dstate}[\text{DOORS}] = \{\text{OPEN}\}$

$\text{grd2} : \text{phase} = \text{movingdown}$

$\text{grd3} : p = E$

$\text{grd4} : \text{button} = \text{DOWN}$

$\text{grd7} : \forall x \cdot x \in 1..3 \Rightarrow \text{handle}(x) = UP$
then

act1 : phase := movingup
act2 : p := R
act3 : button := UP
act4 : i := E
act5 : l := E

end

Event $P_{U5}$ defines $P_{U5}$ extends $P_{U5}$ when

grd1 : $dstate[DOORS] = \{CLOSED\}$
grd2 : phase = movingdown
grd3 : $p = E$
grd4 : button = DOWN
grd5 : $lstate[DOORS] = \{UNLOCKED\}$
grd6 : $door_{open} = \{a \mapsto b|a \in 1..3 \land b \in DOORS \Rightarrow \{FALSE\}\}$
grd7 : $door_{closed} = \{a \mapsto b|a \in 1..3 \land b \in DOORS \Rightarrow \{FALSE\}\}$
grd8 : $\forall x \cdot x \in 1..3 \Rightarrow handle(x) = UP$

then

act1 : phase := movingup
act3 : button := UP
act4 : i := E
act5 : l := E

end

Event $P_{D2}$ defines $P_{D2}$ extends $P_{D2}$ when

grd1 : $l = E$
grd2 : $p = E$
grd3 : phase = movingup
grd4 : $i = E$
grd5 : $lstate[DOORS] = \{LOCKED\}$
grd6 : $\forall x \cdot x \in 1..3 \Rightarrow handle(x) = DOWN$

then

act1 : phase := movingdown
act2 : button := DOWN
act3 : $l := R$
act4 : $p := R$
act5 : $i := E$

end

Event $CompletePD2$ defines $CompletePD2$ extends $CompletePD2$ when

grd1 : phase = movingdown
grd2 : button = DOWN
grd3 : $l = R$
grd4 : $p = R$
grd5 : $i = E$
then
  act1 : phase := haltdown
end

Event $PD3 \equiv$
  extends $PD3$
  when
  grd1 : dstate[DOORS] = {CLOSED}
  grd2 : lstate[DOORS] = {UNLOCKED}
  grd3 : phase = movingup
  grd4 : $p = E$
  grd5 : $l = E$
  grd6 : button = UP
  grd7 : door_open = \{a \mapsto b | a \in 1..3 \land b \in DOORS \rightarrow \{FALSE\}\}
  grd8 : door_closed = \{a \mapsto b | a \in 1..3 \land b \in DOORS \rightarrow \{FALSE\}\}
  grd9 : \forall x \cdot x \in 1..3 \Rightarrow handle(x) = DOWN
then
  act1 : phase := movingdown
  act2 : $p := E$
  act3 : $l := R$
  act4 : button := DOWN
end

Event $PD4 \equiv$
  extends $PD4$
  when
  grd1 : dstate[DOORS] = {OPEN}
  grd2 : phase = movingup
  grd3 : $p = R$
  grd4 : button = UP
  grd6 : \forall x \cdot x \in 1..3 \Rightarrow handle(x) = DOWN
then
  act1 : phase := movingdown
  act2 : $p := E$
  act3 : button := DOWN
  act4 : $i := R$
  act5 : $l := R$
end

Event $PD5 \equiv$
  extends $PD5$
  when
  grd1 : dstate[DOORS] = {CLOSED}
  grd2 : phase = movingup
  grd3 : $p = R$
  grd4 : button = UP
  grd5 : lstate[DOORS] = {UNLOCKED}
  grd6 : door_open = \{a \mapsto b | a \in 1..3 \land b \in DOORS \rightarrow \{FALSE\}\}
  grd7 : door_closed = \{a \mapsto b | a \in 1..3 \land b \in DOORS \rightarrow \{FALSE\}\}
  grd8 : \forall x \cdot x \in 1..3 \Rightarrow handle(x) = DOWN

122
then
  act1 : \text{phase} := \text{movingdown}
  act2 : \text{button} := \text{DOWN}
  act3 : i := R
  act4 : l := R
end

\textbf{Event} retracting\_gears \equiv
\text{extends} retracting\_gears
\textbf{any}

\textbf{ind where}

\begin{align*}
\text{grd1} &: \text{dstate[DOORS]} = \{\text{OPEN}\} \\
\text{grd2} &: \text{gstate[GEARS]} = \{\text{EXTENDED}\} \\
\text{grd3} &: p = R \\
\text{grd4} &: \text{gear\_extended} = \{a \mapsto b | a \in 1..3 \land b \in \text{GEARS} \rightarrow \{\text{TRUE}\}\} \\
\text{grd5} &: \text{gear\_retracted} = \{a \mapsto b | a \in 1..3 \land b \in \text{GEARS} \rightarrow \{\text{FALSE}\}\} \\
\text{grd6} &: \text{gear\_shock\_absorber} = \{a \mapsto b | a \in 1..3 \land b = \text{ground}\} \\
\text{grd7} &: \forall x : x \in 1..3 \Rightarrow \text{handle}(x) = \text{button} \\
\text{grd8} &: \text{SGCylinder} = \{a \mapsto b | a \in \text{GEARS} \times \text{CYLINDER} \land b = \text{MOVING}\} \\
\text{grd9} &: \text{at} \neq \emptyset \\
\text{grd10} &: \text{time} \in \text{ran}(\text{at}) \\
\text{grd11} &: \text{ind} \in \text{dom}(\text{at}) \land \text{ind} \mapsto \text{time} \in \text{at}
\end{align*}

then
  act1 : \text{gstate} := \{a \mapsto b | a \in \text{GEARS} \land b = \text{RETRACTING}\} \\
  act2 : \text{gear\_extended} := 1..3 \rightarrow (\text{GEARS} \rightarrow \{\text{FALSE}\}) \\
  act3 : \text{gear\_shock\_absorber} := \{a \mapsto b | a \in 1..3 \land b = \text{flight}\} \\
  act4 : \text{at} := \text{at} \setminus \{\text{ind} \mapsto \text{time}\}
end

\textbf{Event} retraction \equiv
\text{extends} retraction
\textbf{when}

\begin{align*}
\text{grd1} &: \text{dstate[DOORS]} = \{\text{OPEN}\} \\
\text{grd2} &: \text{gstate[GEARS]} = \{\text{RETRACTING}\} \\
\text{grd3} &: \text{gear\_extended} = \{a \mapsto b | a \in 1..3 \land b \in \text{GEARS} \rightarrow \{\text{FALSE}\}\} \\
\text{grd4} &: \text{gear\_retracted} = \{a \mapsto b | a \in 1..3 \land b \in \text{GEARS} \rightarrow \{\text{FALSE}\}\} \\
\text{grd5} &: \text{gear\_shock\_absorber} = \{a \mapsto b | a \in 1..3 \land b = \text{flight}\} \\
\text{grd6} &: \forall x : x \in 1..3 \Rightarrow \text{handle}(x) = \text{button} \\
\text{grd7} &: \text{SGCylinder} = \{a \mapsto b | a \in \text{GEARS} \times \text{CYLINDER} \land b = \text{STOP}\}
\end{align*}

then
  act1 : \text{gstate} := \{a \mapsto b | a \in \text{GEARS} \land b = \text{RETRACTED}\} \\
  act2 : \text{gear\_retracted} := 1..3 \rightarrow (\text{GEARS} \rightarrow \{\text{TRUE}\})
end

\textbf{Event} extending\_gears \equiv
\text{extends} extending\_gears
\textbf{any}

\textbf{ind where}
grd1 : dstate[DOORS] = \{OPEN\}
grd2 : gstate[GEARS] = \{RETRACTED\}
grd3 : p = E
grd5 : gear_retracted = \{a \mapsto b | a \in 1..3 \land b \in GEARS \rightarrow \{TRUE\}\}
grd6 : gear_extended = \{a \mapsto b | a \in 1..3 \land b \in GEARS \rightarrow \{FALSE\}\}
grd7 : gear_shock_absorber = \{a \mapsto b | a \in 1..3 \land b = flight\}
grd8 : \forall x \cdot x \in 1..3 \Rightarrow handle(x) = button
grd4 : SGCylinder = \{a \mapsto b | a \in GEARS \times CYLINDER \land b = MOVING\}
grd9 : at \neq \emptyset
grd10 : time \in \text{ran}(at)
grd11 : ind \in \text{dom}(at) \land ind \mapsto time \in at

then

act1 : gstate := \{a \mapsto b | a \in GEARS \land b = EXTENDING\}
act2 : gear_retracted := 1..3 \rightarrow (GEARS \rightarrow \{FALSE\})
act3 : at := at \setminus \{ind \mapsto time\}

end

Event extension \cong extends extension
when

grd1 : dstate[DOORS] = \{OPEN\}
grd2 : gstate[GEARS] = \{EXTENDING\}
grd4 : gear_retracted = \{a \mapsto b | a \in 1..3 \land b \in GEARS \rightarrow \{FALSE\}\}
grd5 : gear_extended = \{a \mapsto b | a \in 1..3 \land b \in GEARS \rightarrow \{FALSE\}\}
grd6 : gear_shock_absorber = \{a \mapsto b | a \in 1..3 \land b = flight\}
grd7 : \forall x \cdot x \in 1..3 \Rightarrow handle(x) = button
grd3 : SGCylinder = \{a \mapsto b | a \in GEARS \times CYLINDER \land b = STOP\}

then

act1 : gstate := \{a \mapsto b | a \in GEARS \land b = EXTENDED\}
act2 : gear_retracted := 1..3 \rightarrow (GEARS \rightarrow \{TRUE\})
act3 : gear_shock_absorber := \{a \mapsto b | a \in 1..3 \land b = ground\}

end

Event HPDI \cong extends HPDI
when

then

grd3 : \forall x \cdot x \in 1..3 \Rightarrow handle(x) = UP

act2 : handle := 1..3 \rightarrow \{DOWN\}
act3 : at := at \cup \{(index + 1) \mapsto (time + 160)\}

analoga switch is seen open 160ms after handle position has changed
act4 : handleDown_interval := time + 40000
in the event activation set

add a new time interval (current time + handle not changed interval)
act5 : handleUp_interval := 0
update the handle up interval as 0
act6 : index := index + 1
update the current index value

end

124
Event \( HPU1 \doteq\)
\[\text{extends } HPU1\]
\[\text{when}\]
\[\text{grd3 : } \forall x \cdot x \in 1..3 \Rightarrow \text{handle}(x) = DOW N\]
\[\text{then}\]
\[\text{act2} : \text{handle} \in 1..3 \rightarrow \{UP\}\]
\[\text{act3} : \text{at} := \text{at} \cup \{(index + 1) \mapsto (time + 160)\}\]
\[\text{analogical switch is seen open 160ms after handle position has changed}\]
\[\text{act4} : \text{handleUp}_\text{interval} := \text{time} + 40000\]
\[\text{add a new time interval (current time + handle not changed interval)}\]
\[\text{in the event activation set}\]
\[\text{act5} : \text{handleDown}_\text{interval} := 0\]
\[\text{update the handle down interval as 0}\]
\[\text{act6} : \text{index} := \text{index} + 1\]
\[\text{update the current index value}\]
\[\text{end}\]

Event \( \text{Analogical switch closed} \doteq\)
\[\text{extends } \text{Analogical switch closed}\]
\[\text{any}\]
\[\text{in in port}\]
\[\text{ind}\]
\[\text{where}\]
\[\text{grd1} : \text{in} = \text{general}_\text{EV}\]
\[\text{grd2} : \forall x \cdot x \in 1..3 \Rightarrow (\text{handle}(x) = UP \lor \text{handle}(x) = DOW N)\]
\[\text{grd3} : \text{at} \neq \emptyset\]
\[\text{grd4} : \text{time} \in \text{ran}(\text{at})\]
\[\text{grd5} : \text{ind} \in \text{dom}(\text{at}) \land \text{ind} \mapsto \text{time} \in \text{at}\]
\[\text{then}\]
\[\text{act3} : \text{analogical switch} \in 1..3 \rightarrow \{\text{closed}\}\]
\[\text{act4} : A\_\text{Switch}\_\text{Out} := \text{TRUE}\]
\[\text{act5} : \text{at} := (\text{at} \cup \{(index + 1) \mapsto (time + 1200)\}) \setminus \{\text{ind} \mapsto \text{time}\}\]
\[\text{from closed to open 1.2 sec.}\]
\[\text{act6} : \text{index} := \text{index} + 1\]
\[\text{end}\]

Event \( \text{Analogical switch open} \doteq\)
\[\text{extends } \text{Analogical switch open}\]
\[\text{any}\]
\[\text{in in port}\]
\[\text{ind}\]
\[\text{where}\]
\[\text{grd1} : \text{in} = \text{general}_\text{EV}\]
\[\text{grd2} : \forall x \cdot x \in 1..3 \Rightarrow (\text{handle}(x) = UP \lor \text{handle}(x) = DOW N)\]
\[\text{grd3} : \text{at} \neq \emptyset\]
\[\text{grd4} : \text{time} \in \text{ran}(\text{at})\]
\[\text{grd5} : \text{ind} \in \text{dom}(\text{at}) \land \text{ind} \mapsto \text{time} \in \text{at}\]
\[\text{then}\]
\[\text{act3} : \text{analogical switch} \in 1..3 \rightarrow \{\text{open}\}\]
\[\text{act4} : A\_\text{Switch}\_\text{Out} := \text{FALSE}\]
act5 : at := (at \cup \{(index + 1) \mapsto \text{time} + 800\}) \setminus \{\text{ind} \mapsto \text{time}\}
from open to closed .8 sec.
act6 : index := index + 1

Event Circuit\_pressurized\_OK \triangleq 
extends Circuit\_pressurized\_OK
when

grd1 : general\_EV\_Hout = Hin

end

Event Circuit\_pressurized\_notOK \triangleq 
extends Circuit\_pressurized\_notOK
when

grd1 : general\_EV\_Hout = 0

end

Event Computing\_Module\_1\_2 \triangleq 
extends Computing\_Module\_1\_2
when

grd1 : state = computing
then

act1 : general\_EV := general\_EV\_func(handle \mapsto \text{analogical\_switch} \mapsto \text{gear\_extended} \mapsto \text{gear\_retracted} \mapsto \text{gear\_shock\_absorber} \mapsto \text{door\_open} \mapsto \text{door\_closed} \mapsto \text{circuit\_pressurized})
act2 : close\_EV := close\_EV\_func(handle \mapsto \text{analogical\_switch} \mapsto \text{gear\_extended} \mapsto \text{gear\_retracted} \mapsto \text{gear\_shock\_absorber} \mapsto \text{door\_open} \mapsto \text{door\_closed} \mapsto \text{circuit\_pressurized})
act3 : retract\_EV := retract\_EV\_func(handle \mapsto \text{analogical\_switch} \mapsto \text{gear\_extended} \mapsto \text{gear\_retracted} \mapsto \text{gear\_shock\_absorber} \mapsto \text{door\_open} \mapsto \text{door\_closed} \mapsto \text{circuit\_pressurized})
act4 : extend\_EV := extend\_EV\_func(handle \mapsto \text{analogical\_switch} \mapsto \text{gear\_extended} \mapsto \text{gear\_retracted} \mapsto \text{gear\_shock\_absorber} \mapsto \text{door\_open} \mapsto \text{door\_closed} \mapsto \text{circuit\_pressurized})
act5 : open\_EV := open\_EV\_func(handle \mapsto \text{analogical\_switch} \mapsto \text{gear\_extended} \mapsto \text{gear\_retracted} \mapsto \text{gear\_shock\_absorber} \mapsto \text{door\_open} \mapsto \text{door\_closed} \mapsto \text{circuit\_pressurized})
act6 : gears\_locked\_down := gears\_locked\_down\_func(handle \mapsto \text{analogical\_switch} \mapsto \text{gear\_extended} \mapsto \text{gear\_retracted} \mapsto \text{gear\_shock\_absorber} \mapsto \text{door\_open} \mapsto \text{door\_closed} \mapsto \text{circuit\_pressurized})
act7 : gears\_man := gears\_man\_func(handle \mapsto \text{analogical\_switch} \mapsto \text{gear\_extended} \mapsto \text{gear\_retracted} \mapsto \text{gear\_shock\_absorber} \mapsto \text{door\_open} \mapsto \text{door\_closed} \mapsto \text{circuit\_pressurized})
act8 : anomaly := anomaly\_func(handle \mapsto \text{analogical\_switch} \mapsto \text{gear\_extended} \mapsto \text{gear\_retracted} \mapsto \text{gear\_shock\_absorber} \mapsto \text{door\_open} \mapsto \text{door\_closed} \mapsto \text{circuit\_pressurized})
act9 : state := electroValve
Event \( Update\_Hout \) \( \triangleq \) Assign the value of \( Hout \)

extends \( Update\_Hout \)

when

\[ \text{grd1} : \text{state} = \text{electroValve} \]

then

\[ \text{act1} : \text{general}\_EV\_Hout : ||(\text{general}\_EV = \text{TRUE} \land \text{general}\_EV\_Hout' = Hin) \lor (\text{general}\_EV = \text{FALSE} \land \text{general}\_EV\_Hout' = 0) \lor (A\_Switch\_Out = \text{TRUE} \land \text{general}\_EV\_Hout' = Hin)) \lor (A\_Switch\_Out = \text{FALSE} \land \text{general}\_EV\_Hout' = 0)) \]

pass the current value of hydraulic input port (\( Hin \)) to hydraulic output port

\[ \text{act2} : \text{close}\_EV\_Hout : ||(\text{close}\_EV = \text{TRUE} \land \text{close}\_EV\_Hout' = Hin) \lor (\text{close}\_EV = \text{FALSE} \land \text{close}\_EV\_Hout' = 0)) \]

\[ \text{act3} : \text{open}\_EV\_Hout : ||(\text{open}\_EV = \text{TRUE} \land \text{open}\_EV\_Hout' = Hin) \lor (\text{open}\_EV = \text{FALSE} \land \text{open}\_EV\_Hout' = 0)) \]

\[ \text{act4} : \text{extend}\_EV\_Hout : ||(\text{extend}\_EV = \text{TRUE} \land \text{extend}\_EV\_Hout' = Hin) \lor (\text{extend}\_EV = \text{FALSE} \land \text{extend}\_EV\_Hout' = 0)) \]

\[ \text{act5} : \text{retract}\_EV\_Hout : ||(\text{retract}\_EV = \text{TRUE} \land \text{retract}\_EV\_Hout' = Hin) \lor (\text{retract}\_EV = \text{FALSE} \land \text{retract}\_EV\_Hout' = 0)) \]

\[ \text{act6} : \text{state} := \text{cylinder} \]

\[ \text{act7} : \text{at} := \text{at} \cup \{(\text{index} + 1) \mapsto (\text{time} + 2000)\} \cup \\ \{(\text{index} + 2) \mapsto (\text{time} + 10000)\} \cup \{(\text{index} + 3) \mapsto (\text{time} + 500)\} \cup \{(\text{index} + 4) \mapsto (\text{time} + 2000)\} \cup \{(\text{index} + 5) \mapsto (\text{time} + 500)\} \cup \{(\text{index} + 6) \mapsto (\text{time} + 2000)\} \cup \{(\text{index} + 7) \mapsto (\text{time} + 500)\} \cup \{(\text{index} + 8) \mapsto (\text{time} + 10000)\} \cup \{(\text{index} + 9) \mapsto (\text{time} + 500)\} \cup \{(\text{index} + 10) \mapsto (\text{time} + 10000)\} \]

general EV 2 (time is given in comments in sec. while in model these are in ms.)

general EV 10
opening EV 0.5
opening EV 2
closure EV 0.5
closure EV 2
retraction EV 0.5
retraction EV 10
extension 0.5
extension 10

\[ \text{act8} : \text{index} := \text{index} + 10 \]

Event \( CylinderMovingOrStop \) \( \triangleq \) Cylinder Moving or Stop according to the output of hydraulic circuit

extends \( CylinderMovingOrStop \)

127
when \( \text{grd1} : \text{state} = \text{cylinder} \) then

\[
\text{act1} : SGCylinder : \quad \left( \left( SGCylinder' = \{ a \mapsto b \mid a \in \text{GEARS} \times \{ \text{GCY F}, \text{GCY R}, \text{GCY L} \} \land b = \text{MOVING} \} \land \text{extend}_{E V \_Hout} = \text{Hin} \right) \lor \right.
\]

\[
\quad \left. \left( SGCylinder' = \{ a \mapsto b \mid a \in \text{GEARS} \times \{ \text{GCY F}, \text{GCY R}, \text{GCY L} \} \land b = \text{STOP} \} \land \text{extend}_{E V \_Hout} = 0 \right) \lor \right.
\]

\[
\quad \left. \left( SGCylinder' = \{ a \mapsto b \mid a \in \text{GEARS} \times \{ \text{GCY F}, \text{GCY R}, \text{GCY L} \} \land b = \text{MOVING} \} \land \text{retract}_{E V \_Hout} = \text{Hin} \right) \lor \right.
\]

\[
\quad \left. \left( SGCylinder' = \{ a \mapsto b \mid a \in \text{GEARS} \times \{ \text{GCY F}, \text{GCY R}, \text{GCY L} \} \land b = \text{STOP} \} \land \text{retract}_{E V \_Hout} = 0 \right) \right) \]

\[
\text{act2} : SDCylinder : \quad \left( \left( SDCylinder' = \{ a \mapsto b \mid a \in \text{DOORS} \times \{ \text{DCY F}, \text{DCY R}, \text{DCY L} \} \land b = \text{MOVING} \} \land \text{open}_{E V \_Hout} = \text{Hin} \right) \lor \right.
\]

\[
\quad \left. \left( SDCylinder' = \{ a \mapsto b \mid a \in \text{DOORS} \times \{ \text{DCY F}, \text{DCY R}, \text{DCY L} \} \land b = \text{STOP} \} \land \text{open}_{E V \_Hout} = 0 \right) \lor \right.
\]

\[
\quad \left. \left( SDCylinder' = \{ a \mapsto b \mid a \in \text{DOORS} \times \{ \text{DCY F}, \text{DCY R}, \text{DCY L} \} \land b = \text{MOVING} \} \land \text{close}_{E V \_Hout} = \text{Hin} \right) \lor \right.
\]

\[
\quad \left. \left( SDCylinder' = \{ a \mapsto b \mid a \in \text{DOORS} \times \{ \text{DCY F}, \text{DCY R}, \text{DCY L} \} \land b = \text{STOP} \} \land \text{close}_{E V \_Hout} = 0 \right) \right) \]

\[
\text{act3} : \quad \text{state} := \text{computing} \]

\[
\text{Event} \quad \text{Failure Detection Generic Monitoring} \quad \hat{=} \quad \text{extends} \quad \text{Failure Detection Generic Monitoring} \]

\[
\text{when} \]
\[
\text{grd1} : (\forall x, y, z \cdot x \in 1..3 \land y \in 1..3 \land z \in 1..3 \land x \neq y \land y \neq z \land x \neq z \Rightarrow \\
(\text{handle}(x) \neq \text{handle}(y) \land \text{handle}(y) \neq \text{handle}(z) \land \text{handle}(x) \neq \text{handle}(z))) \\
\lor \\
(\forall x, y, z \cdot x \in 1..3 \land y \in 1..3 \land z \in 1..3 \land x \neq y \land y \neq z \land x \neq z \Rightarrow \\
(\text{analogueal_switch}(x) \neq \text{analogueal_switch}(y) \land \text{analogueal_switch}(y) \neq \\
\text{analogueal_switch}(z) \land \text{analogueal_switch}(z) \neq \text{analogueal_switch}(x))) \\
\lor \\
(\forall x, y, z \cdot x \in 1..3 \land y \in 1..3 \land z \in 1..3 \land x \neq y \land y \neq z \land x \neq z \Rightarrow \\
(\text{gear_retracted}(x) \neq \text{gear_retracted}(y) \land \text{gear_retracted}(y) \neq \\
\text{gear_retracted}(z) \land \text{gear_retracted}(z) \neq \text{gear_retracted}(x))) \\
\lor \\
(\forall x, y, z \cdot x \in 1..3 \land y \in 1..3 \land z \in 1..3 \land x \neq y \land y \neq z \land x \neq z \Rightarrow \\
(\text{gear_shock_absorber}(x) \neq \text{gear_shock_absorber}(y) \land \\
\text{gear_shock_absorber}(y) \neq \text{gear_shock_absorber}(z) \land \text{gear_shock_absorber}(x) \neq \\
\text{gear_shock_absorber}(z))) \\
\lor \\
(\forall x, y, z \cdot x \in 1..3 \land y \in 1..3 \land z \in 1..3 \land x \neq y \land y \neq z \land x \neq z \Rightarrow \\
(\text{door_open}(x) \neq \text{door_open}(y) \land \text{door_open}(y) \neq \text{door_open}(z) \land \\
\text{door_open}(z) \neq \text{door_open}(x))) \\
\lor \\
(\forall x, y, z \cdot x \in 1..3 \land y \in 1..3 \land z \in 1..3 \land x \neq y \land y \neq z \land x \neq z \Rightarrow \\
(\text{door_closed}(x) \neq \text{door_closed}(y) \land \text{door_closed}(y) \neq \text{door_closed}(z) \land \\
\text{door_closed}(z) \neq \text{door_closed}(x))) \\
\lor \\
(\forall x, y, z \cdot x \in 1..3 \land y \in 1..3 \land z \in 1..3 \land x \neq y \land y \neq z \land x \neq z \Rightarrow \\
(\text{circuit_pressurized}(x) \neq \text{circuit_pressurized}(y) \land \\
\text{circuit_pressurized}(y) \neq \text{circuit_pressurized}(z) \land \text{circuit_pressurized}(z) \neq \\
\text{circuit_pressurized}(x)))
\]

Generic Monitoring using all sensors

\[\text{act1} : \text{anomaly} := \text{TRUE}\]

end

Event Failure_Detection_Analogical_Switch \preceq

extends Failure_Detection_Analogical_Switch

any

ind

where

\[\text{grd1} : \text{analogueal_switch} = \{a \mapsto b | a \in 1..3 \land b = \text{open}\} \]

\[\text{analogueal_switch} = \{a \mapsto b | a \in 1..3 \land b = \text{closed}\}\]

Gears motion monitoring without considering time

\[\text{grd2} : \text{at} \neq \emptyset\]

\[\text{grd3} : \text{time} \in \text{ran(at)}\]

129
Event `check_handle_delay` $\checkmark$

This event is used to set 280ms in the set “at”
for event activation to detect anomaly
and detect that handle is not change from last 40 sec.

extends `check_handle_delay`

when

grd1 : $time = handleUp\_interval$
\lor
\quad  
\quad  $time = handleDown\_interval$
\quad current time is either equal to handle up interval or equal to the handle down interval

then

act1 : $at := at \cup \{(index + 1) \mapsto (time + 280)\}$
\quad To add a new interval to the event activation set
act3 : $index := index + 1$
\quad update the current index value

end

Event `Failure\_Detection\_Pressure\_Sensor` $\checkmark$

extends `Failure\_Detection\_Pressure\_Sensor`

any

where

grd1 : $circuit\_pressurized \neq \{a \mapsto b | a \in 1..3 \land b = TRUE\}$
\lor
\quad  $circuit\_pressurized \neq \{a \mapsto b | a \in 1..3 \land b = FALSE\}$
\quad Circuit pressurized motion monitoring without considering time

grd2 : $at \neq \emptyset$
grd3 : $time \in ran(at)$
grd4 : $ind \in dom(at) \land ind \mapsto time \in at$

then

act1 : $anomaly := TRUE$
act2 : $at := at \setminus \{ind \mapsto time\}$

end

Event `Failure\_Detection\_Doors` $\checkmark$

extends `Failure\_Detection\_Doors`

any

where

$ind$
Doors motion monitoring without considering time

\[ \text{door\_closed} \neq \{ a \mapsto b \mid a \in 1..3 \land b \in \text{DOORS} \rightarrow \{ \text{FALSE} \} \} \]
\[ \lor \]
\[ \text{door\_open} \neq \{ a \mapsto b \mid a \in 1..3 \land b \in \text{DOORS} \rightarrow \{ \text{TRUE} \} \} \]
\[ \lor \]
\[ \text{door\_open} \neq \{ a \mapsto b \mid a \in 1..3 \land b \in \text{DOORS} \rightarrow \{ \text{FALSE} \} \} \]
\[ \lor \]
\[ \text{door\_closed} \neq \{ a \mapsto b \mid a \in 1..3 \land b \in \text{DOORS} \rightarrow \{ \text{TRUE} \} \} \]

Gears motion monitoring without considering time

\[ \text{gear\_retracted} \neq \{ a \mapsto b \mid a \in 1..3 \land b \in \text{GEARS} \rightarrow \{ \text{FALSE} \} \} \]
\[ \lor \]
\[ \text{gear\_retracted} \neq \{ a \mapsto b \mid a \in 1..3 \land b \in \text{GEARS} \rightarrow \{ \text{TRUE} \} \} \]
\[ \lor \]
\[ \text{gear\_extended} \neq \{ a \mapsto b \mid a \in 1..3 \land b \in \text{GEARS} \rightarrow \{ \text{FALSE} \} \} \]
\[ \lor \]
\[ \text{gear\_extended} \neq \{ a \mapsto b \mid a \in 1..3 \land b \in \text{GEARS} \rightarrow \{ \text{TRUE} \} \} \]

Event Failure\_Detection\_Gears \eqdef \text{Failure\_Detection\_Gears} \text{any}

\[ \text{ind} \]

where

\[ \text{grd1} : \text{gear\_retracted} \neq \{ a \mapsto b \mid a \in 1..3 \land b \in \text{GEARS} \rightarrow \{ \text{FALSE} \} \} \]
\[ \lor \]
\[ \text{gear\_retracted} \neq \{ a \mapsto b \mid a \in 1..3 \land b \in \text{GEARS} \rightarrow \{ \text{TRUE} \} \} \]
\[ \lor \]
\[ \text{gear\_extended} \neq \{ a \mapsto b \mid a \in 1..3 \land b \in \text{GEARS} \rightarrow \{ \text{FALSE} \} \} \]
\[ \lor \]
\[ \text{gear\_extended} \neq \{ a \mapsto b \mid a \in 1..3 \land b \in \text{GEARS} \rightarrow \{ \text{TRUE} \} \} \]

Event tic\_tock \eqdef \text{tic\_tock} \text{any}

\[ \text{tm} \]

where

\[ \text{grd1} : \text{tm} \in \mathbb{N} \]
\[ \text{grd2} : \text{tm} > \text{time} \]

\[ \text{to take a new value of time in the future} \]
\[ \text{grd3} : \text{ran}(\text{at}) \neq \emptyset \Rightarrow \text{tm} \leq \min(\text{ran}(\text{at})) \]

\[ \text{act1} : \text{time} := \text{tm} \]

\[ \text{assign a new value of time to the current time} \]
Event $\text{pilot\_interface\_Green\_light\_On} \overset{\text{end}}{\equiv}$

- green light is on when gears locked down is true

  - when
    
    - $\text{grd1} : g\hat{a}r\hat{e}r\hat{s}\_\text{locked\_down} = \text{TRUE}$
    
    - gears locked down must be true
  
  - then
    
    - $\text{act1} : \text{pilot\_interface\_light}(\text{Green}) := \text{On}$
    
    - To set on of Green light of pilot interface light

Event $\text{pilot\_interface\_Orange\_light\_On} \overset{\text{end}}{\equiv}$

- orange light is on when gears maneuvering is true

  - when
    
    - $\text{grd1} : g\hat{a}r\hat{e}r\hat{s}\_\text{man} = \text{TRUE}$
    
    - gears maneuvering must be true
  
  - then
    
    - $\text{act1} : \text{pilot\_interface\_light}(\text{Orange}) := \text{On}$
    
    - To set on of Orange light of pilot interface light

Event $\text{pilot\_interface\_Red\_light\_On} \overset{\text{end}}{\equiv}$

- red light is on when anomaly is detected (true)

  - when
    
    - $\text{grd1} : \text{anomaly} = \text{TRUE}$
    
    - anomaly must be true
    
    - $\text{grd2} : \text{pilot\_interface\_light}(\text{Red}) = \text{Off}$
    
  - then
    
    - $\text{act1} : \text{pilot\_interface\_light}(\text{Red}) := \text{On}$
    
    - To set on of Red light of pilot interface light

Event $\text{pilot\_interface\_Green\_light\_Off} \overset{\text{end}}{\equiv}$

- green light is off when gears locked down is false

  - when
    
    - $\text{grd1} : g\hat{a}r\hat{e}r\hat{s}\_\text{locked\_down} = \text{FALSE}$
    
    - gears locked down must be false
  
  - then
    
    - $\text{act1} : \text{pilot\_interface\_light}(\text{Green}) := \text{Off}$
    
    - To set off of Green light of pilot interface light

Event $\text{pilot\_interface\_Orange\_light\_Off} \overset{\text{end}}{\equiv}$

- orange light is off when gears maneuvering is false
when
  \[\text{grd1} : \text{gears\_man} = FALSE\]
  \hspace{1cm} \text{gears maneuvering must be false}
then
  \[\text{act1} : \text{pilot\_interface\_light(Orange)} := \text{Off}\]
  \hspace{1cm} \text{To set off of Orange light of pilot interface light}
end

Event \[\text{pilot\_interface\_Red\_light\_Off} \equiv\]
\hspace{1cm} \text{red light is off when anomaly is detected (false)}
when
  \[\text{grd1} : \text{anomaly} = FALSE\]
  \hspace{1cm} \text{anomaly must be false}
then
  \[\text{grd2} : \text{pilot\_interface\_light(Red)} = \text{On}\]
  \[\text{act1} : \text{pilot\_interface\_light(Red)} := \text{Off}\]
  \hspace{1cm} \text{To set off of Red light of pilot interface light}
end
END