Politecnico di Milano
Formal Methods for Concurrent and Real-Time Systems

Computer Controller Automatic Transmission
(Mandatory Part and Optional Part)

Alessandra Bonetto
Filippo Sironi
Matteo Villa

2009

Contents

List of Figures

Listings

1 CCAT Class

The *ComputerControlledAutomaticTransmission* class is formalized thanks to the code reported in Listing 1 while Figure 1 shows the *big picture* of our complete designed.

Listing 1: ComputerControlledAutomaticTransmission.trio

```
class ComputerControlledAutomaticTransmission

import:
    HydraulicSystem,
    PlanetaryGearSet,
    TransmissionControlUnit,
    VehicleSpeedSensor,
    EngineSpeedSensor;

signature:

visible:
    torqueConverterState,
    vehicleSpeed,
    engineSpeed;

temporal domain: real;

domains:
    TorqueConverterState: {Attached, Detached};

items:
    TD total torqueConverterState: TorqueConverterState;
    TD total vehicleSpeed: integer;
    TD total engineSpeed: integer;

modules:
    hydraulicSystem: HydraulicSystem;
    planetaryGearSet: PlanetaryGearSet;
    transmissionControlUnit: TransmissionControlUnit;
    vehicleSpeedSensor: VehicleSpeedSensor;
    engineSpeedSensor: EngineSpeedSensor;

connections:
    (direct EngineSpeedSensor.actualSpeed, engineSpeed)
    (direct vehicleSpeedSensor.actualSpeed, vehicleSpeed)
```

```
37      (direct planetaryGearSet.transmissionShaftState,
            torqueConverterState)
38
39      (direct planetaryGearSet.gearShift,
            hydraulicSystem.gearShift)
40      (direct planetaryGearSet.gearDrive,
            hydraulicSystem.gearDrive)
41      (direct planetaryGearSet.gearPark,
            hydraulicSystem.gearPark)
42      (direct planetaryGearSet.gearReverse,
            hydraulicSystem.gearReverse)
43      (direct hydraulicSystem.controlGearShift,
            transmissionControlUnit.controlGearShift)
44      (direct transmissionControlUnit.receiveEngineSpeed,
            engineSpeedSensor.sendSpeed)
45      (direct transmissionControlUnit.receiveVehicleSpeed,
            vehicleSpeedSensor.sendSpeed)
46
47  end
```

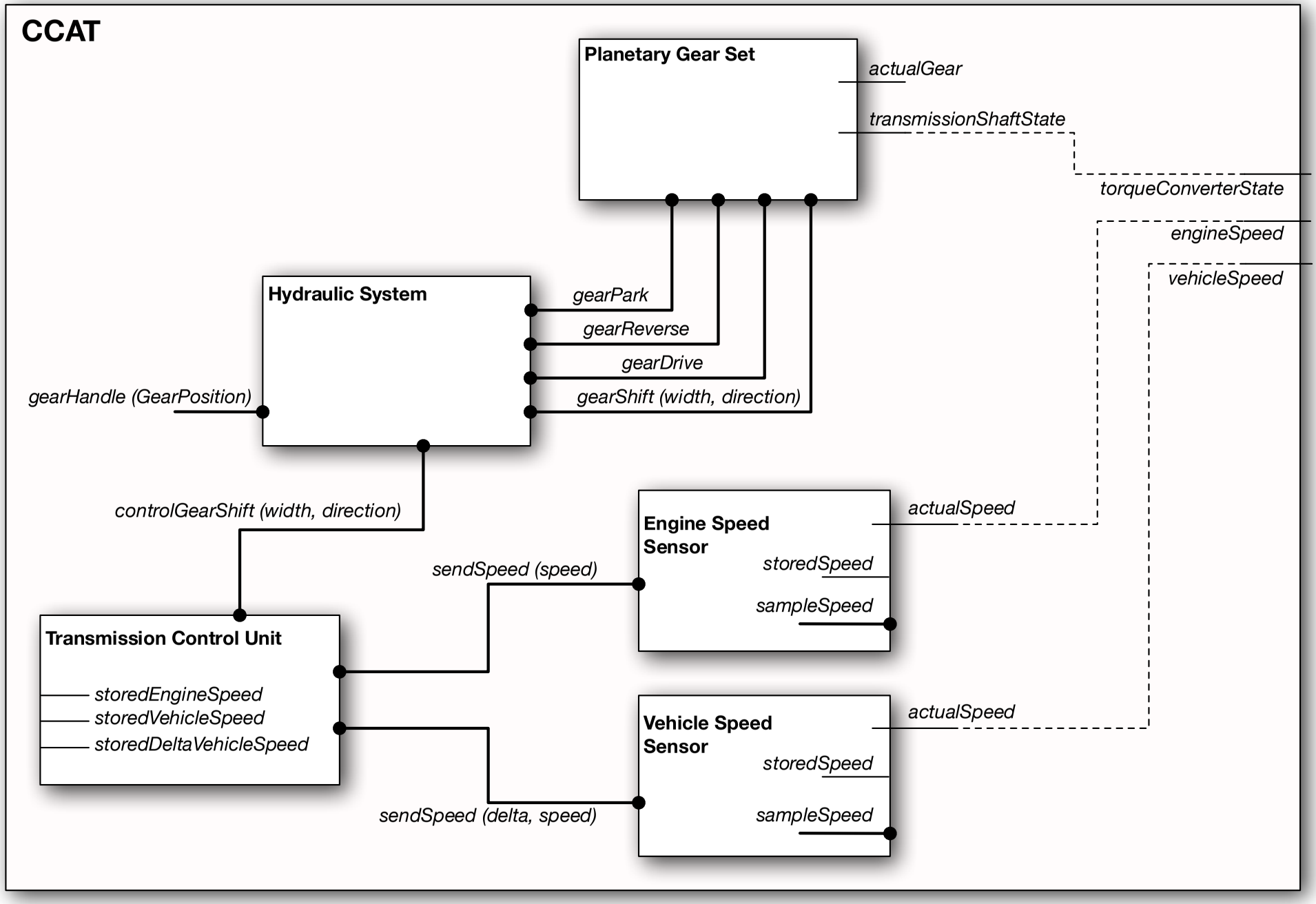

Figure 1: Computer Controller Automatic Transmission

Due to the fact that a TRIO specification is not decidable we have decided to focus our verification process only on the Hydraulic System, described in Section 4, and on the Planetary Gear Set, described in Section 3 and we have used a decidable subset of the TRIO language.

The specification is reported in Listing 2.

Listing 2: ComputerControlledAutomaticTransmission.trio

```
1  variables
2      controlGearShift: [0..2],
3      gearHandle: [0..3],
4      gearShift: [0..5],
5      actualGear: [0..3],
6      transmissionShaftState: [0..1]
7
8  constants
9      Nothing = 0,
10     TCUShiftOneUp = 1, TCUShiftOneDown = 2,
11     HandleShiftDrive = 1, HandleShiftPark = 2,
           HandleShiftReverse = 3,
12     ShiftOneUp = 1, ShiftOneDown = 2, ShiftDrive = 3,
           ShiftPark = 4, ShiftReverse = 5,
13     First = 0, Second = 1, Park = 2, Reverse = 3,
14     Detached = 0, Attached = 1,
15     FluidDelay = 1, ShiftDelay = 1
16
17 axioms
18     Mechanic:
19         (actualGear = First ->
20             gearShift = Nothing | gearShift = ShiftOneUp |
                   gearShift = ShiftPark | gearShift =
                   ShiftReverse
21         ) &
22         (actualGear = Second ->
23             gearShift = Nothing | gearShift = ShiftOneDown
24         ) &
25         (actualGear = Park ->
26             gearShift = Nothing | gearShift = ShiftDrive |
                   gearShift = ShiftReverse
27         ) &
28         (actualGear = Reverse ->
29             gearShift = Nothing | gearShift = ShiftDrive |
                   gearShift = ShiftPark
30         ) &
```

```
31          (gearShift = ShiftOneUp -> transmissionShaftState
               = Attached) &
32          (gearShift = ShiftDrive | gearShift = ShiftPark |
               gearShift = ShiftReverse ->
33           transmissionShaftState = Detached
34          ) &
35          (transmissionShaftState = Attached -> actualGear =
               First | actualGear = Second | actualGear =
               Reverse) &
36          (actualGear = Park -> transmissionShaftState =
               Detached);
37
38      GearShiftCommand:
39          (controlGearShift = TCUShiftOneUp ->
40              Lasts_ii(gearHandle = Nothing, FluidDelay) &
41              Lasts_ei(controlGearShift = Nothing,
                   FluidDelay) &
42              Futr(gearShift = ShiftOneUp, FluidDelay)
43          ) &
44          (controlGearShift = TCUShiftOneDown ->
45              Lasts_ii(gearHandle = Nothing, FluidDelay) &
46              Lasts_ei(controlGearShift = Nothing,
                   FluidDelay) &
47              Futr(gearShift = ShiftOneDown, FluidDelay)
48          );
49
50      GearHandleCommand:
51          (gearHandle = HandleShiftDrive ->
52              Lasts_ii(controlGearShift = Nothing &
                   transmissionShaftState = Detached,
                   FluidDelay) &
53              Lasts_ei(gearHandle = Nothing, FluidDelay) &
54              Futr(gearShift = ShiftDrive, FluidDelay)
55          ) &
56          (gearHandle = HandleShiftPark ->
57              Lasts_ii(controlGearShift = Nothing &
                   transmissionShaftState = Detached,
                   FluidDelay) &
58              Lasts_ei(gearHandle = Nothing, FluidDelay) &
59              Futr(gearShift = ShiftPark, FluidDelay)
60          ) &
61          (gearHandle = HandleShiftReverse ->
```

```
62              Lasts_ii(controlGearShift = Nothing &
                    transmissionShaftState = Detached,
                    FluidDelay) &
63              Lasts_ei(gearHandle = Nothing, FluidDelay) &
64              Futr(gearShift = ShiftReverse, FluidDelay)
65          );
66
67      GearShiftFirst:
68          (actualGear = First & gearShift = ShiftOneUp ->
69              Lasts_ei(gearShift = Nothing, ShiftDelay) &
                    Futr(actualGear = Second, ShiftDelay)
70          ) &
71          (actualGear = First & gearShift = ShiftPark ->
72              Lasts_ii(transmissionShaftState = Detached,
                    ShiftDelay) &
73              Lasts_ei(gearShift = Nothing, ShiftDelay) &
74              Futr(actualGear = Park, ShiftDelay)
75          ) &
76          (actualGear = First & gearShift = ShiftReverse ->
77              Lasts_ii(transmissionShaftState = Detached,
                    ShiftDelay) &
78              Lasts_ei(gearShift = Nothing, ShiftDelay) &
79              Futr(actualGear = Reverse, ShiftDelay)
80          );
81
82      GearShiftSecond:
83          (actualGear = Second & gearShift = ShiftOneDown ->
84              Lasts_ei(gearShift = Nothing, ShiftDelay) &
                    Futr(actualGear = First, ShiftDelay)
85          );
86
87      GearShiftPark:
88          (actualGear = Park & gearShift = ShiftDrive ->
89              Lasts_ii(transmissionShaftState = Detached,
                    ShiftDelay) &
90              Lasts_ei(gearShift = Nothing, ShiftDelay) &
91              Futr(actualGear = First, ShiftDelay)
92          ) &
93          (actualGear = Park & gearShift = ShiftReverse ->
94              Lasts_ii(transmissionShaftState = Detached,
                    ShiftDelay) &
95              Lasts_ei(gearShift = Nothing, ShiftDelay) &
96              Futr(actualGear = Reverse, ShiftDelay)
```

```
97          );
98
99      GearShiftReverse:
100         (actualGear = Reverse & gearShift = ShiftDrive ->
101             Lasts_ii(transmissionShaftState = Detached,
                    ShiftDelay) &
102             Lasts_ei(gearShift = Nothing, ShiftDelay) &
103             Futr(actualGear = First, ShiftDelay)
104         ) &
105         (actualGear = Reverse & gearShift = ShiftPark ->
106             Lasts_ii(transmissionShaftState = Detached,
                    ShiftDelay) &
107             Lasts_ei(gearShift = Nothing, ShiftDelay) &
108             Futr(actualGear = Park, ShiftDelay)
109         );
```

The same specification reported in Listings 2 has been ported also in Zot and is listed in Listings 3.

Listing 3: ComputerControlledAutomaticTransmission.lisp

```
1  (asdf:operate 'asdf:load-op 'bezot)
2  (use-package :trio-utils)
3
4  ; Constants
5  (defvar Nothing 0)
6
7  (defvar TCUShiftOneUp 1)
8  (defvar TCUShiftOneDown 2)
9
10 (defvar HandleDrive 1)
11 (defvar HandlePark 2)
12 (defvar HandleReverse 3)
13
14 (defvar ShiftOneUp 1)
15 (defvar ShiftOneDown 2)
16 (defvar ShiftDrive 3)
17 (defvar ShiftPark 4)
18 (defvar ShiftReverse 5)
19
20 (defvar First 0)
21 (defvar Second 1)
22 (defvar Park 2)
23 (defvar Reverse 3)
24
```

```
25 (defvar Detached 0)
26 (defvar Attached 1)
27
28 (defvar FluidDelay 1)
29 (defvar ShiftDelay 1)
30
31 ; Domains
32 (defvar ControlGearShiftDomain (loop for i from 0 to 2
       collect i))
33 (defvar GearHandleDomain (loop for i from 0 to 3 collect
       i))
34 (defvar GearShiftDomain (loop for i from 0 to 5 collect i))
35 (defvar ActualGearDomain (loop for i from 0 to 3 collect
       i))
36 (defvar TransmissionShaftStateDomain (loop for i from 0 to
       1 collect i))
37
38 ; Variables
39 (define-variable controlGearShift ControlGearShiftDomain)
40 (define-variable gearHandle GearHandleDomain)
41 (define-variable gearShift GearShiftDomain)
42 (define-variable actualGear ActualGearDomain)
43 (define-variable transmissionShaftState
       TransmissionShaftStateDomain)
44
45 ; Axioms
46 (defvar ControlGearShiftMutualExclusion
47     (&&
48         (-E- x '(0 1 2) (-P- controlGearShift x))
49         (-A- x '(0 1 2)
50             (->
51                 (-P- controlGearShift x)
52                 (-A- y '(0 1 2) (-> (!! (= x y)) (!! (-P-
                       controlGearShift y))))
53             )
54         )
55     )
56 )
57
58 (defvar GearHandleMutualExclusion
59     (&&
60         (-E- x '(0 1 2 3) (-P- gearHandle x))
61         (-A- x '(0 1 2 3)
```

```
62              (->
63                  (-P- gearHandle x)
64                  (-A- y '(0 1 2 3) (->   (!! (= x y)) (!!
                        (-P- gearHandle y))))
65              )
66          )
67      )
68  )
69
70  (defvar GearShiftMutualExclusion
71      (&&
72          (-E- x '(0 1 2 3 4 5) (-P- gearShift x))
73          (-A- x '(0 1 2 3 4 5)
74              (->
75                  (-P- gearShift x)
76                  (-A- y '(0 1 2 3 4 5) (->   (!! (= x y))
                        (!! (-P- gearShift y))))
77              )
78          )
79      )
80  )
81
82  (defvar ActualGearMutualExclusion
83      (&&
84          (-E- x '(0 1 2 3) (-P- actualGear x))
85          (-A- x '(0 1 2 3)
86              (->
87                  (-P- actualGear x)
88                  (-A- y '(0 1 2 3) (->   (!! (= x y)) (!!
                        (-P- actualGear y))))
89              )
90          )
91      )
92  )
93
94  (defvar TransmissionShaftStateMutualExclusion
95      (&&
96          (-E- x '(0 1) (-P- transmissionShaftState x))
97          (-A- x '(0 1)
98              (->
99                  (-P- transmissionShaftState x)
100                 (-A- y '(0 1) (->   (!! (= x y)) (!! (-P-
                        transmissionShaftState y))))
```

```
101             )
102         )
103     )
104 )
105
106 (defvar Mechanic
107     (&&
108         (->
109             (actualGear-is First)
110             (||
111                 (gearShift-is Nothing)
112                 (gearShift-is ShiftOneUp)
113                 (gearShift-is ShiftPark)
114                 (gearShift-is ShiftReverse)
115             )
116         )
117         (->
118             (actualGear-is Second)
119             (|| (gearShift-is Nothing) (gearShift-is
                    ShiftOneDown))
120         )
121         (->
122             (actualGear-is Park)
123             (|| (gearShift-is Nothing) (gearShift-is
                    ShiftDrive) (gearShift-is ShiftReverse))
124         )
125         (->
126             (actualGear-is Reverse)
127             (|| (gearShift-is Nothing) (gearShift-is
                    ShiftDrive) (gearShift-is ShiftPark))
128         )
129         (-> (gearShift-is ShiftOneUp)
                (transmissionShaftState-is Attached))
130         (->
131             (||
132                 (gearShift-is ShiftDrive)
133                 (gearShift-is ShiftPark)
134                 (gearShift-is ShiftReverse)
135             )
136             (transmissionShaftState-is Detached)
137         )
138         (->
139             (transmissionShaftState-is Attached)
```

```
140             (|| (actualGear-is First) (actualGear-is
                    Second) (actualGear-is Reverse))
141         )
142         (-> (actualGear-is Park)
                (transmissionShaftState-is Detached))
143         (->
144             (&&
145                 (controlGearShift-is Nothing)
146                 (gearHandle-is Nothing)
147             )
148             (Futr (gearShift-is Nothing) FluidDelay)
149         )
150         (-A- x '(0 1 2 3)
151             (->
152                 (&& (-P- actualGear x) (gearShift-is
                        Nothing))
153                 (Lasts_ii (-P- actualGear x) ShiftDelay)
154             )
155         )
156     )
157 )
158
159 (defvar GearShiftCommand
160     (&&
161         (->
162             (controlGearShift-is TCUShiftOneUp)
163             (Futr (gearShift-is ShiftOneUp) FluidDelay)
164         )
165         (->
166             (controlGearShift-is TCUShiftOneDown)
167             (Futr (gearShift-is ShiftOneDown) FluidDelay)
168         )
169     )
170 )
171
172 (defvar GearHandleCommand
173     (&&
174         (->
175             (gearHandle-is HandleDrive)
176             (&&
177                 (Lasts_ii (&& (controlGearShift-is
                            Nothing) (transmissionShaftState-is
                            Detached)) FluidDelay)
```

```
178                 (Lasts_ei (gearHandle-is Nothing)
                        FluidDelay)
179                 (Futr (gearShift-is ShiftDrive) FluidDelay)
180             )
181         )
182         (->
183             (gearHandle-is HandlePark)
184             (&&
185                 (Lasts_ii (&& (controlGearShift-is
                        Nothing) (transmissionShaftState-is
                        Detached)) FluidDelay)
186                 (Lasts_ei (gearHandle-is Nothing)
                        FluidDelay)
187                 (Futr (gearShift-is ShiftPark) FluidDelay)
188             )
189         )
190         (->
191             (gearHandle-is HandleReverse)
192             (&&
193                 (Lasts_ii (&& (controlGearShift-is
                        Nothing) (transmissionShaftState-is
                        Detached)) FluidDelay)
194                 (Lasts_ei (gearHandle-is Nothing)
                        FluidDelay)
195                 (Futr (gearShift-is ShiftReverse)
                        FluidDelay)
196             )
197         )
198     )
199 )
200
201 (defvar GearShiftFirst
202     (&&
203         (->
204             (&& (actualGear-is First) (gearShift-is
                    ShiftOneUp))
205             (&&
206                 (Lasts_ei (gearShift-is Nothing)
                        ShiftDelay)
207                 (Futr (actualGear-is Second) ShiftDelay)
208             )
209         )
210         (->
```

```
211             (&& (actualGear-is First) (gearShift-is
                    ShiftPark))
212             (&&
213                 (Lasts_ii (transmissionShaftState-is
                        Detached) ShiftDelay)
214                 (Lasts_ei (gearShift-is Nothing)
                        ShiftDelay)
215                 (Futr (actualGear-is Park) ShiftDelay)
216             )
217         )
218         (->
219             (&& (actualGear-is First) (gearShift-is
                    ShiftReverse))
220             (&&
221                 (Lasts_ii (transmissionShaftState-is
                        Detached) ShiftDelay)
222                 (Lasts_ei (gearShift-is Nothing)
                        ShiftDelay)
223                 (Futr (actualGear-is Reverse) ShiftDelay)
224             )
225         )
226     )
227 )
228
229 (defvar GearShiftSecond
230     (->
231         (&& (actualGear-is Second) (gearShift-is
                ShiftOneDown))
232         (&&
233             (Lasts_ei (gearShift-is Nothing) ShiftDelay)
234             (Futr (actualGear-is First) ShiftDelay)
235         )
236     )
237 )
238
239 (defvar GearShiftPark
240     (&&
241         (->
242             (&& (actualGear-is Park) (gearShift-is
                    ShiftDrive))
243             (&&
244                 (Lasts_ii (transmissionShaftState-is
                        Detached) ShiftDelay)
```

```
245                 (Lasts_ei (gearShift-is Nothing)
                        ShiftDelay)
246                 (Futr (actualGear-is First) ShiftDelay)
247             )
248         )
249         (->
250             (&& (actualGear-is Park) (gearShift-is
                    ShiftReverse))
251             (&&
252                 (Lasts_ii (transmissionShaftState-is
                        Detached) ShiftDelay)
253                 (Lasts_ei (gearShift-is Nothing)
                        ShiftDelay)
254                 (Futr (actualGear-is Reverse) ShiftDelay)
255             )
256         )
257     )
258 )
259
260 (defvar GearShiftReverse
261     (&&
262         (->
263             (&& (actualGear-is Reverse) (gearShift-is
                    ShiftDrive))
264             (&&
265                 (Lasts_ii (transmissionShaftState-is
                        Detached) ShiftDelay)
266                 (Lasts_ei (gearShift-is Nothing)
                        ShiftDelay)
267                 (Futr (actualGear-is First) ShiftDelay)
268             )
269         )
270         (->
271             (&& (actualGear-is Reverse) (gearShift-is
                    ShiftPark))
272             (&&
273                 (Lasts_ii (transmissionShaftState-is
                        Detached) ShiftDelay)
274                 (Lasts_ei (gearShift-is Nothing)
                        ShiftDelay)
275                 (Futr (actualGear-is Park) ShiftDelay)
276             )
277         )
```

```
278     )
279 )
280
281 (defvar ComputerControlledAutomaticTransmission
282     (Alw
283         (&&
284             ControlGearShiftMutualExclusion
285             GearHandleMutualExclusion
286             GearShiftMutualExclusion
287             ActualGearMutualExclusion
288             TransmissionShaftStateMutualExclusion
289             Mechanic
290             GearShiftCommand
291             GearHandleCommand
292             GearShiftFirst
293             GearShiftSecond
294             GearShiftPark
295             GearShiftReverse
296         )
297     )
298 )
299
300 (defvar PropertyOne
301     (->
302         (&& (actualGear-is First) (controlGearShift-is
                TCUShiftOneUp))
303         (Futr (actualGear-is Second) (+ FluidDelay
                ShiftDelay))
304     )
305 )
306
307 (defvar PropertyTwo
308     (->
309         (&& (actualGear-is First) (gearHandle-is
                HandleReverse))
310         (&&
311             (Futr (actualGear-is Reverse) (+ FluidDelay
                    ShiftDelay))
312             (Lasts_ii (transmissionShaftState-is Detached)
                    (+ FluidDelay ShiftDelay))
313         )
314     )
315 )
```

```
316
317 (defvar PropertyThree
318     (->
319         (&& (actualGear-is First) (gearHandle-is
                HandlePark))
320         (&&
321             (Futr (actualGear-is Park) (+ FluidDelay
                    ShiftDelay))
322             (Until (transmissionShaftState-is Detached)
                    (!! (gearHandle-is Nothing)))
323         )
324     )
325 )
326
327 (bezot:zot 20 ComputerControlledAutomaticTransmission)
328 ;(bezot:zot 20 (&& ComputerControlledAutomaticTransmission
    (!! (Alw PropertyOne))))
329 ;(bezot:zot 20 (&& ComputerControlledAutomaticTransmission
    (!! (Alw PropertyTwo))))
330 ;(bezot:zot 20 (&& ComputerControlledAutomaticTransmission
    (!! (Alw PropertyThree))))
```

2 Vehicle/EngineSpeedSensor Classes

The *VehicleSpeedSensor* is formalized thanks to the code reported in Listing 4 while the *EngineSpeedSensor* is formalized thanks to the code reported in Listing 5.

During the formalization of sensors we decided to simplify the design assuming that every time a `sampleSpeed` event occurs the state variable `actualSpeed` - which is `time dependent` and `total` - is automatically updated with the actual measured speed. This means we don't provide any axioms formalizing this behavior.

Moreover, we specified the starting point of the constant frequency sample chain saying that sometimes in the past there was a `sampleSpeed` occurrence. Further more, we guarantee that `sampleSpeed` events will occur at constant frequency. In addition, if the sensor has memory we imposed that the `storedValue` is equal to 0. These can be consider just like the "initial conditions" of the system.

At the end, we guaranteed a sensor performs the needed action if and only if a sample event occur.

We didn't write any axioms specifying the fact that a `sendSpeed` event is mutually exclusive with itself due to the `total time dependent` parameter it accepts.

Listing 4: VehicleSpeedSensor.trio

```
1  class VehicleSpeedSensor (const sampleInterval, const
       sampleDelay)
2
3  signature:
4
5  visible:
6      actualSpeed,
7      sendSpeed;
8
9  temporal domain: real;
10
11 items:
12     TI sampleInterval: real;
13     TI sampleDelay: real;
14     TD total storedSpeed: integer;
15     TD total actualSpeed: integer;
16     event sendSpeed (integer, integer);
17     event sampleSpeed;
18
```

```
19 axioms:
20 vars:
21     deltaSpeed: integer;
22     speed: integer;
23 formulae:
24     SpeedValues:
25         actualSpeed >= 0 and storedSpeed >= 0;
26
27     BeginSample:
28         SomP (storedSpeed = 0 & sampleSpeed);
29
30     SamplingDefinition:
31         sampleSpeed implies Futr (sampleSpeed,
               sampleInterval) and not Lasts (sampleSpeed,
               sampleInterval);
32
33     SamplingAction:
34         sampleSpeed implies Futr (deltaSpeed = actualSpeed
               - storedSpeed and speed = actualSpeed and
               sendSpeed (deltaSpeed, speed) and Lasts
               (storedSpeed = actualSpeed, sampleInterval),
               sampleDelay);
35
36     SendSpeed:
37         deltaSpeed = actualSpeed - storedSpeed and
               actualSpeed = speed and sendSpeed (deltaSpeed,
               speed) implies Past (sampleSpeed, sampleDelay);
38
39 end
```

Listing 5: EngineSpeedSensor.trio

```
1  class EngineSpeedSensor (const sampleInterval, const
       sampleDelay)
2
3  signature:
4
5  visible: actualSpeed, sendSpeed;
6
7  temporal domain: real;
8
9  items:
10     TI sampleInterval: real;
11     TI sampleDelay: real;
```

```
12      TD total actualSpeed: integer;
13      event sendSpeed (integer);
14      event sampleSpeed;
15
16  axioms:
17  vars:
18      speed: integer;
19  formulae:
20      SpeedValues:
21          actualSpeed >= 0;
22
23      BeginSample:
24          SomP (sampleSpeed);
25
26      SamplingDefinition:
27          sampleSpeed implies Futr (sampleSpeed,
                sampleInterval) and not Lasts (sampleSpeed,
                sampleInterval);
28
29      SampleSpeedActions:
30          sampleSpeed implies Futr (actualSpeed = speed and
                sendSpeed (speed), sampleDelay);
31
32      SendSpeed:
33          actualSpeed = speed and sendSpeed (speed) implies
                Past (sampleSpeed, sampleDelay);
34
35  end
```

3 PlanetaryGearSet Class

The *PlanetaryGearSet* class is formalized thanks to the code reported in Listing 6.

The Planetary Gear Set guarantees that every time a gear shift event occurs the `actualGear` will be maintained until the shift is finished.

Inside this component are defined all axioms limiting gear shifts to effective ones only (e.g. it is impossible to shift down a gear if `actualGear` is `First`). The Planetary Gear Set permits to shift up to two gear at the same time (as the specification asks), however, the Transmission Control Unit doesn't use this possibility because in a real Planetary Gear Set this is not possible.

Moreover, through the formalization of the Planetary Gear Set we impose that we can't receive a gear shift event if we are in the middle of a gear shift. Different gear shifting times are defined for different gears and different steps.

The gears `Drive`, `Park`, and `Reverse` can be selected if and only if the transmission shaft is decoupled from the engine.

The state of the Planetary Gear Set changes if and only if an event occurs.

Listing 6: PlanetaryGearSet.trio

```
1  class PlanetaryGearSet (const singleGearShiftDelay, const
       dualGearShiftDelay, const driveGearShiftDelay, const
       parkGearShiftDelay, const reverseGearShiftDelay)
2
3  signature:
4
5  visible:
6      actualGear,
7      transmissionShaftState,
8      gearShift,
9      gearDrive,
10     gearPark,
11     gearReverse,
12
13 temporal domain: real;
14
15 domains:
16     Gear: {First, Second, Third, Park, Reverse};
17     TransmissionShaftState: {Attached, Detached};
18     ShiftWidth: 1..2;
19     ShiftDirection: {Up, Down};
20
21 items:
22     TI singleGearShiftDelay: real;
```

```
23      TI dualGearShiftDelay: real;
24      TI driveGearShiftDelay: real;
25      TI parkGearShiftDelay: real;
26      TI reverseGearShiftDelay: real;
27      TD total actualGear: Gear;
28      TD total transmissionShaftState:
            TransmissionShaftState;
29      event gearShift (ShiftWidth, ShiftDirection);
30      event gearDrive;
31      event gearPark;
32      event gearReverse;
33
34  axioms:
35  vars:
36      gearShiftWidth: ShiftWidth;
37      gearShiftWidth2: ShiftWidth;
38      gearShiftDirection: ShiftDirection;
39      gearShiftDirection2: ShiftDirection;
40      gear: Gear;
41  formulae:
42      Mechanics:
43          all gearShiftWidth (gearShiftDirection = Up ->
                transmissionShaftState=Attached);
44
45      GearShiftDrive:
46          gearDrive implies transmissionShaftState =
                Detached;
47
48      GearShiftFirst:
49          (actualGear = First implies not gearDrive and not
                ex gearShiftWidth (gearShiftDirection = Down
                and gearShift (gearShiftWidth,
                gearShiftDirection))) and
50          (actualGear = First implies SomF (gearPark or
                gearReverse or ex gearShiftWidth,
                gearShiftDirection (gearShift (gearShiftWidth,
                gearShiftDirection)))) and
51          (actualGear = First and gearShiftWidth = 1 and
                gearShiftDirection = Up and gearShift
                (gearShiftWidth, gearShiftDirection) implies
                Lasts (actualGear = First,
                singleGearShiftDelay) and Futr (actualGear =
                Second, singleGearShiftDelay)) and
```

```
52          (actualGear = First and gearShiftWidth = 2 and
                gearShiftDirection = Up and gearShift
                (gearShiftWidth, gearShiftDirection) implies
                Lasts (actualGear = First, dualGearShiftDelay)
                and Futr (actualGear = Third,
                dualGearShiftDelay) and
53          (actualGear = First and gearPark implies Lasts
                (actualGear = First, parkGearShiftDelay) and
                Futr (actualGear = Park, parkGearShiftDelay))
                and
54          (actualGear = First and gearReverse implies Lasts
                (actualGear = First, reverseGearShiftDelay) and
                Futr (actualGear = Reverse,
                reverseGearShiftDelay));
55
56      GearShiftSecond:
57          (actualGear = Second implies not gearDrive and not
                gearPark and not gearReverse and not ex
                gearShiftDirection (gearShiftWidth = 2 and
                gearShift (gearShiftWidth,
                gearShiftDirection))) and
58          (actualGear = Second implies SomF (ex
                gearShiftDirection (gearShiftWidth = 1 and
                gearShift (gearShiftWidth,
                gearShiftDirection)))) and
59          (actualGear = Second and gearShiftWidth = 1 and
                gearShiftDirection = Up and gearShift
                (gearShiftWidth, gearShiftDirection) implies
                Lasts (actualGear = Second,
                singleGearShiftDelay) and Futr (actualGear =
                Third, singleGearShiftDelay)) and
60          (actualGear = Second and gearShiftWidth = 1 and
                gearShiftDirection = Down and gearShift
                (gearShiftWidth, gearShiftDirection) implies
                Lasts (actualGear = Second,
                singleGearShiftDelay) and Futr (actualGear =
                First, singleGearShiftDelay));
61
62      GearShiftThird:
63          (actualGear = Third implies not gearDrive and not
                gearPark and not gearReverse and not ex
                gearShiftWidth (gearShiftDirection = Up and
                gearShift (gearShiftWidth,
```

```
            gearShiftDirection))) and
64          (actualGear = Third implies SomF (ex
               gearShiftWidth (gearShiftDirection = Down and
               gearShift (gearShiftWidth,
               gearShiftDirection)))) and
65          (actualGear = Third and gearShiftWidth = 1 and
               gearShiftDirection = Down and gearShift
               (gearShiftWidth, gearShiftDirection) implies
               Lasts (actualGear = Third,
               singleGearShiftDelay) and Futr (actualGear =
               Second, singleGearShiftDelay)) and
66          (actualGear = Third and gearShiftWidth = 2 and
               gearShiftDirection = Down and gearShift
               (gearShiftWidth, gearShiftDirection) implies
               Lasts (actualGear = Third, dualGearShiftDelay)
               and Futr (actualGear = First,
               dualGearShiftDelay));
67
68      GearShiftReverse:
69          (actualGear = Reverse implies not gearReverse and
               all gearShiftWidth, gearShiftDirection (not
               gearShift (gearShiftWidth,
               gearShiftDirection))) and
70          (actualGear = Reverse implies SomF (gearDrive or
               gearPark)) and
71          (actualGear = Reverse and gearDrive implies Lasts
               (actualGear = Reverse, driveGearShiftDelay) and
               Futr (actualGear = First, driveGearShiftDelay))
               and
72          (actualGear = Reverse and gearPark implies Lasts
               (actualGear = Reverse, parkGearShiftDelay) and
               Futr (actualGear = Park, parkGearShiftDelay))
               and
73          (gearReverse implies transmissionShaftState =
               Detached);
74
75      GearShiftPark:
76          (actualGear = Park implies not gearPark and all
               gearShiftWidth, gearShiftDirection (not
               gearShift (gearShiftWidth,
               gearShiftDirection))) and
77          (actualGear = Park implies SomF (gearDrive or
               gearReverse)) and
```

```
78          (actualGear = Park and gearDrive implies Lasts
               (actualGear = Park, reverseGearShiftDelay) and
               Futr (actualGear = First, driveGearShiftDelay))
               and
79          (actualGear = Park and gearReverse implies Lasts
               (actualGear = Park, reverseGearShiftDelay) and
               Futr (actualGear = Reverse,
               reverseGearShiftDelay)) and
80          (actualGear = Park implies transmissionShaftState
               = Detached) and
81          (gearPark implies transmissionShaftState =
               Detached);
82
83      GearShiftTimings:
84          all gearShiftDirection ((actualGear = First or
               actualGear = Second or actualGear = Third) and
               gearShiftWidth = 1 and gearShift
               (gearShiftWidth, gearShiftDirection) implies
               not Lasts (gearDrive or gearPark or gearReverse
               or ex gearShiftWidth2, gearShiftDirection2
               (gearShift (gearShiftWidth2,
               gearShiftDirection2)), singleGearShiftDelay))
               and
85          all gearShiftDirection ((actualGear = First or
               actualGear = Third) and gearShiftWidth = 2 and
               gearShift (gearShiftWidth, gearShiftDirection)
               implies not Lasts (gearDrive or gearPark or
               gearReverse or ex gearShiftWidth2,
               gearShiftDirection2 (gearShift
               (gearShiftWidth2, gearShiftDirection2)),
               dualGearShiftDelay)) and
86          ((actualGear = Reverse and gearDrive) implies not
               Lasts (gearDrive or gearPark or gearReverse or
               ex gearShiftWidth2, gearShiftDirection2
               (gearShift (gearShiftWidth2,
               gearShiftDirection2)), driveGearShiftDelay)) and
87          ((actualGear = Reverse and gearPark) implies not
               Lasts (gearDrive or gearPark or gearReverse or
               ex gearShiftWidth2, gearShiftDirection2
               (gearShift (gearShiftWidth2,
               gearShiftDirection2)), parkGearShiftDelay)) and
88          ((actualGear = Park and gearDrive) implies not
               Lasts (gearDriver or gearPark or gearReverse or
```

```
            ex gearShiftWidth2, gearShiftDirection2
            (gearShift (gearShiftWidth2,
            gearShiftDirection2)), driveGearShiftDelay)) and
89          ((actualGear = Park and gearReverse) implies not
            Lasts (gearDrive or gearPark or gearReverse or
            ex gearShiftWidth2, gearShiftDirection2
            (gearShift (gearShiftWidth2,
            gearShiftDirection2)), reverseGearShiftDelay));
90
91      Nothing:
92          all gear (actualGear = gear and not (all
            gearShiftWidth, gearShiftDirection (gearShift
            (gearShiftWidth, gearShiftDirection)) or
            gearDrive or gearPark or gearReverse) implies
            UpToNow (actualGear = gear) and NowOn
            (actualGear = gear));
93
94  end
```

4 HydraulicSystem Class

The *HydraulicSystem* class is formalized thanks to the code reported in Listing 7.

The first assumption we made before modelling the Hydraulic System was that every valve and electrovalve configuration imposes the same fluid propagation delay; this means that for every command that the Hydraulic System propagates the delay will always be the same. This behavior is formalized with the `time independent constant fluidPropagationDelay`.

The *manual valve*, which permit the driver to manually select the gear mode, is modelled thanks to the `gearHandle` event and the `GearHandle` axiom. During the time in which the Hydraulic System propagate a command there can be no `gearHandle` event which somehow means the fluid propagation is faster then the driver reaction time (which is a realistic assumption).

Moreover, thanks to the `MutualExclusion` axiom, it's impossible to generate two `gearHandle` event at the same time which means that the gear handle can't be for example in Park and Drive mode at the same instant.

Listing 7: HydraulicSystem.trio

```
1  class HydraulicSystem (const fluidPropagationDelay)
2
3  signature:
4
5  visible:
6      gearHandle,
7      gearShift,
8      gearDrive,
9      gearPark,
10     gearReverse,
11     controlGearShift;
12
13 temporal domain: real;
14
15 domains:
16     GearPosition: {Drive, Park, Reverse};
17     ShiftWidth: 1..2;
18     ShiftDirection: {Up, Down};
19
20 items:
21     TI fluidPropagationDelay: real;
22     event gearHandle (GearPosition);
23     event gearShift (ShiftWidth, ShiftDirection);
24     event gearDrive;
```

```
25      event gearPark;
26      event gearReverse;
27      event controlGearShift (ShiftWidth, ShiftDirection);
28
29  axioms:
30  vars:
31      gear: GearPosition;
32      gear2: GearPosition;
33      gearShiftWidth: ShiftWidth;
34      gearShiftWidth2: ShiftWidth;
35      gearShiftDirection: ShiftDirection;
36      gearShiftDirection2: ShiftDirection;
37  formulae:
38      GearHandleCommand:
39          (gear = Drive and gearHandle (gear) implies not
               Lasts (all gear2 (gear2 <> gear implies
               gearHandle (gear2) or ex gearShiftWidth,
               gearShiftDirection (controlGearShift
               (gearShiftWidth, gearShiftDirection))),
               fluidPropagationDelay) and Futr (gearDrive,
               fluidPropagationDelay)) and
40          (gear = Park and gearHandle (gear) implies not
               Lasts (all gear2 (gear2 <> gear implies
               gearHandle (gear2) or ex gearShiftWidth,
               gearShiftDirection (controlGearShift
               (gearShiftWidth, gearShiftDirection))),
               fluidPropagationDelay) and Futr (gearPark,
               fluidPropagationDelay)) and
41          (gear = Reverse and gearHandle (gear) implies not
               Lasts (all gear2 (gear2 <> gear implies
               gearHandle (gear2) or ex gearShiftWidth,
               gearShiftDirection (controlGearShift
               (gearShiftWidth, gearShiftDirection))),
               fluidPropagationDelay) and Futr (gearReverse,
               fluidPropagationDelay));
42
43      PropagateGearShiftCommand:
44          all gearShiftWidth, gearShiftDirection
               (controlGearShift (gearShiftWidth,
               gearShiftDirection) implies not Lasts
               (gearDrive or gearPark or gearReverse or ex
               gearShiftWidth2, gearShiftDirection2
               (controlGearShift (gearShiftWidth2,
```

```
            gearShiftDirection2)), fluidPropagationDelay)
            and Futr (gearShift (gearShiftWidth,
            gearShiftDirection), fluidPropagationDelay));
45
46      MutualExclusions:
47          all gear (gearHandle (gear) implies all gear2
              (gear <> gear2 implies not gearHandle (gear2)));
48
49  end
```

5 TransmissionControlUnit Class

The *TransmissionControlUnit* class is formalized thanks to the code reported in Listing 8.

Our first formalization of the Transmission Control Unit didn't take in account the possibility to have asynchronous sensors; the latest version of the Transmission Control Unit permits to manage asynchronous sensors thanks to internal memory modelled with three `time dependent total` values.

When handle the necessity to scale gears till the `First` with the assumption that the human reaction is way slower than sampling frequency and mechanical reactions, so, when the vehicle stops, the axiom which handle the gear scale manage to be "active" the necessary amount of times to scale all the gears.

The Transmission Control Unit guarantees that it doesn't raise more than one gear shift event per instant and it receives at most one event per instant from each sensor (this is described also in Section 2 and so guaranteed in VehicleSpeedSensor and EngineSpeedSensor class).

Listing 8: TransmissionControlUnit.trio

```
class TransmissionControlUnit

signature:

visible:
    controlGearShift,
    receiveEngineSpeed,
    receiveVehicleSpeed;

temporal domain: real;

domains:
    ShiftWidth: 1..2;
    ShiftDirection: {Up, Down};

items:
    TD total storedEngineSpeed: integer;
    TD total storedDeltaVehicleSpeed: integer;
    TD total storedVehicleSpeed: integer;
    event controlGearShift (ShiftWidth, ShiftDirection);
    event receiveEngineSpeed (integer);
    event receiveVehicleSpeed (integer, integer);

axioms:
vars:
```

```
26      engineSpeed: integer;
27      engineSpeed1: integer;
28      engineSpeed2: integer;
29      deltaVehicleSpeed: integer;
30      deltaVehicleSpeed1: integer;
31      deltaVehicleSpeed2: integer;
32      vehicleSpeed: integer;
33      vehicleSpeed1: integer;
34      vehicleSpeed2: integer;
35      gearShiftWidth1: ShiftWidth;
36      gearShiftWidth2: ShiftWidth;
37      gearShiftDirection1: ShiftDirection;
38      gearShiftDirection2: ShiftDirection;
39  formulae:
40      GearShifts:
41          (receiveEngineSpeed (engineSpeed) and
              receiveVehicleSpeed (deltaVehicleSpeed,
              vehicleSpeed) and engineSpeed >= 3000 and
              vehicleSpeed > 0 implies gearShiftWidth1 = 1
              and gearShiftDirection1 = Up and
              controlGearShift (gearShiftWidth1,
              gearShiftDirection1)) and
42          (receiveEngineSpeed (engineSpeed) and all
              deltaVehicleSpeed, vehicleSpeed (not
              receiveVehicleSpeed (deltaVehicleSpeed,
              vehicleSpeed)) and engineSpeed >= 3000 and
              storedVehicleSpeed > 0 implies gearShiftWidth1
              = 1 and gearShiftDirection1 = Up and
              controlGearShift (gearShiftWidth1,
              gearShiftDirection1)) and
43          (all engineSpeed (not receiveEngineSpeed
              (engineSpeed)) and receiveVehicleSpeed
              (deltaVehicleSpeed, vehicleSpeed) and
              storedEngineSpeed >= 3000 and vehicleSpeed > 0
              implies gearShiftWidth1 = 1 and
              gearShiftDirection1 = Up and controlGearShift
              (gearShiftWidth1, gearShiftDirection1)) and
44          (receiveEngineSpeed (engineSpeed) and
              receiveVehicleSpeed (deltaVehicleSpeed,
              vehicleSpeed) and engineSpeed <= 1500 and
              deltaVehicleSpeed <= 0 implies gearShiftWidth1
              = 1 and gearShiftDirection1 = Down and
              controlGearShift (gearShiftWidth1,
```

```
            gearShiftDirection1)) and
45          (receiveEngineSpeed (engineSpeed) and all
            deltaVehicleSpeed, vehicleSpeed (not
            receiveVehicleSpeed (deltaVehicleSpeed,
            vehicleSpeed)) and engineSpeed <= 1500 and
            storedDeltaVehicleSpeed <= 0 implies
            gearShiftWidth1 = 1 and gearShiftDirection1 =
            Down and controlGearShift (gearShiftWidth1,
            gearShiftDirection1)) and
46          (all engineSpeed (not receiveEngineSpeed
            (engineSpeed)) and receiveVehicleSpeed
            (deltaVehicleSpeed, vehicleSpeed) and
            storedEngineSpeed <= 1500 and
            (deltaVehicleSpeed <= 0 or vehicleSpeed = 0)
            implies gearShiftWidth1 = 1 and
            gearShiftDirection1 = Down and controlGearShift
            (gearShiftWidth1, gearShiftDirection1)) and
47          (receiveEngineSpeed (engineSpeed) and
            receiveVehicleSpeed (deltaVehicleSpeed,
            vehicleSpeed) and engineSpeed <= 1500 and
            deltaVehicleSpeed > 0 implies all
            gearShiftWidth1, gearShiftDirection1 (not
            controlGearShift (gearShiftWidth1,
            gearShiftDirection1))) and
48          (receiveEngineSpeed (engineSpeed) and all
            deltaVehicleSpeed, vehicleSpeed (not
            receiveVehicleSpeed (deltaVehicleSpeed,
            vehicleSpeed)) and engineSpeed <= 1500 and
            storedDeltaVehicleSpeed >= 0 and
            storedVehicleSpeed > 0 implies all
            gearShiftWidth1, gearShiftDirection1 (not
            controlGearShift (gearShiftWidth1,
            gearShiftDirection1))) and
49          (all engineSpeed (not receiveEngineSpeed
            (engineSpeed)) and receiveVehicleSpeed
            (deltaVehicleSpeed, vehicleSpeed) and
            storedEngineSpeed <= 1500 and deltaVehicleSpeed
            >= 0 and vehicleSpeed > 0 implies all
            gearShiftWidth1, gearShiftDirection1 (not
            controlGearShift (gearShiftWidth1,
            gearShiftDirection1))) and
50          (receiveEngineSpeed (engineSpeed) and
            receiveVehicleSpeed (deltaVehicleSpeed,
```

```
            vehicleSpeed) and engineSpeed >= 1500 and
            engineSpeed < 3000 implies all gearShiftWidth1,
            gearShiftDirection1 (not controlGearShift
            (gearShiftWidth1, gearShiftDirection1))) and
51      (receiveEngineSpeed (engineSpeed) and all
            deltaVehicleSpeed, vehicleSpeed (not
            receiveVehicleSpeed (deltaVehicleSpeed,
            vehicleSpeed)) and engineSpeed >= 1500 and
            engineSpeed < 3000 implies all gearShiftWidth1,
            gearShiftDirection1 (not controlGearShift
            (gearShiftWidth1, gearShiftDirection1))) and
52      (all engineSpeed (not receiveEngineSpeed
            (engineSpeed)) and receiveVehicleSpeed
            (deltaVehicleSpeed, vehicleSpeed) and
            storedEngineSpeed >= 1500 and storedEngineSpeed
            < 3000 implies all gearShiftWidth1,
            gearShiftDirection1 (not controlGearShift
            (gearShiftWidth1, gearShiftDirection1))) and
53      (all engineSpeed (not receiveEngineSpeed
            (engineSpeed)) and all deltaVehicleSpeed,
            vehicleSpeed (not receiveVehicleSpeed
            (deltaVehicleSpeed, vehicleSpeed)) implies all
            gearShiftWidth1, gearShiftDirection1 (not
            controlGearShift (gearShiftWidth1,
            gearShiftDirection1)));
54
55  ReceivingEventAction:
56      all deltaVehicleSpeed1, vehicleSpeed1
            (receiveVehicleSpeed (deltaVehicleSpeed1,
            vehicleSpeed1) implies Until
            (storedDeltaVehicleSpeed = deltaVehicleSpeed1
            and storedVehicleSpeed = vehicleSpeed1, ex
            deltaVehicleSpeed2, vehicleSpeed2
            (receiveVehicleSpeed (deltaVehicleSpeed2,
            vehicleSpeed2)))) and
57      all engineSpeed1 (receiveEngineSpeed
            (engineSpeed1) implies Until (storedEngineSpeed
            = engineSpeed1, ex engineSpeed2
            (receiveEngineSpeed (engineSpeed2))));
58
59  MutualExclusions:
60      all gearShiftWidth1, gearShiftDirection1
            (controlGearShift (gearShiftWidth1,
```

```
            gearShiftDirection1) implies all
            gearShiftWidth2, gearShiftDirection2
            (gearShiftWidth1 <> gearShiftWidth2 and
            gearShiftDirection1 <> gearShiftDirection2
            implies not controlGearShift (gearShiftWidth2,
            gearShiftDirection2))) and
61          all engineSpeed1 (receiveEngineSpeed
            (engineSpeed1) implies all engineSpeed2
            (engineSpeed2 <> engineSpeed1 implies not
            receiveEngineSpeed (engineSpeed2))) and
62          all deltaVehicleSpeed1, vehicleSpeed1
            (receiveVehicleSpeed (deltaVehicleSpeed1,
            vehicleSpeed1) implies all deltaVehicleSpeed2,
            vehicleSpeed2 (deltaVehicleSpeed2 <>
            deltaVehicleSpeed1 and vehicleSpeed2 <>
            vehicleSpeed1 implies not receiveVehicleSpeed
            (deltaVehicleSpeed2, vehicleSpeed2)));
63
64  end
```

6 Annotations

During the last phase of our modelling we decided not to formalize the *Torque Converter* and this decision depends on the way the Torque Converter works.

The Torque Converter is a mechanical component that works coupling and decoupling the *Transmission Shaft* and the *Engine Shaft*. It solves is duty without the necessity to receive commands from any component of the system and this is the cause we have decided to remove it from our model.

Anyway, the state of the Torque Converter is really important for the system since it gives information that permits to insert or not to insert some gears and other details that aren't taken into account in this project.

7 Properties

In this Section 7 are reported the three properties we have proved in their decidable TRIO form; these formulae are available in their *Zot* form directly in Listings 3.

In order to verify these three properties we needed to translate them from their "present/future" form to a "present/past" form, because of some inherent limitations of *Spin*. The latter property has been verified only with Zot that doesn't need any translation.

Listing 9: Property 1

```
actualGear = First & controlGearShift = TCUShiftOneUp ->
    Futr(actualGear = Second, FluidDelay + ShiftDelay);
```

Listing 10: Property 2

```
actualGear = First & gearHandle = HandleShiftReverse ->
    Futr(actualGear = Reverse, FluidDelay + ShiftDelay) &
    Lasts_ii(transmissionShaftState = Detached, FluidDelay
    + ShiftDelay));
```

Listing 11: Property 3

```
actualGear = First & gearHandle = HandleShiftPark ->
    Futr(actualGear = Park, FluidDelay + ShiftDelay) &
    Until(transmissionShaftState = Detached, gearHandle <>
    Nothing));
```

Log files are available at this page `http://code.google.com/p/ccat/source/browse/#svn/trunk/log`.