

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

Computer Controller Automatic Transmission
(Mandatory Part and Optional Part not complete yet)

Alessandra Bonetto
Filippo Sironi
Matteo Villa

2009

Contents

1	CCAT Class	5
2	Vehicle/EngineSpeedSensor Classes	12
3	PlanetaryGearSet Class	15
4	HydraulicSystem Class	21
5	TransmissionControlUnit Class	24
6	Annotations	29
7	Properties	30

List of Figures

1	Computer Controller Automatic Transmission	7
---	--	---

Listings

1	ComputerControlledAutomaticTransmission.trio	5
2	ComputerControlledAutomaticTransmission.t2p	8
3	VehicleSpeedSensor.trio	12
4	EngineSpeedSensor.trio	13
5	PlanetaryGearSet.trio	15
6	HydraulicSystem.trio	21
7	TransmissionControlUnit.trio	24
8	Property 1	30
9	Property 2	30
10	Property 3	30

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

```

1  class ComputerControlledAutomaticTransmission
2
3  import :
4      HydraulicSystem ,
5      PlanetaryGearSet ,
6      TransmissionControlUnit ,
7      VehicleSpeedSensor ,
8      EngineSpeedSensor ;
9
10 signature :
11
12 visible :
13     torqueConverterState ,
14     vehicleSpeed ,
15     engineSpeed ;
16
17 temporal domain : real ;
18
19 domains :
20     TorqueConverterState : { Attached , Detached } ;
21
22 items :
23     TD total torqueConverterState : TorqueConverterState ;
24     TD total vehicleSpeed : integer ;
25     TD total engineSpeed : integer ;
26
27 modules :
28     hydraulicSystem : HydraulicSystem ;
29     planetaryGearSet : PlanetaryGearSet ;
30     transmissionControlUnit : TransmissionControlUnit ;
31     vehicleSpeedSensor : VehicleSpeedSensor ;
32     engineSpeedSensor : EngineSpeedSensor ;
33
34 connections :
35     (direct EngineSpeedSensor.actualSpeed , engineSpeed)
36     (direct vehicleSpeedSensor.actualSpeed , vehicleSpeed)

```

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

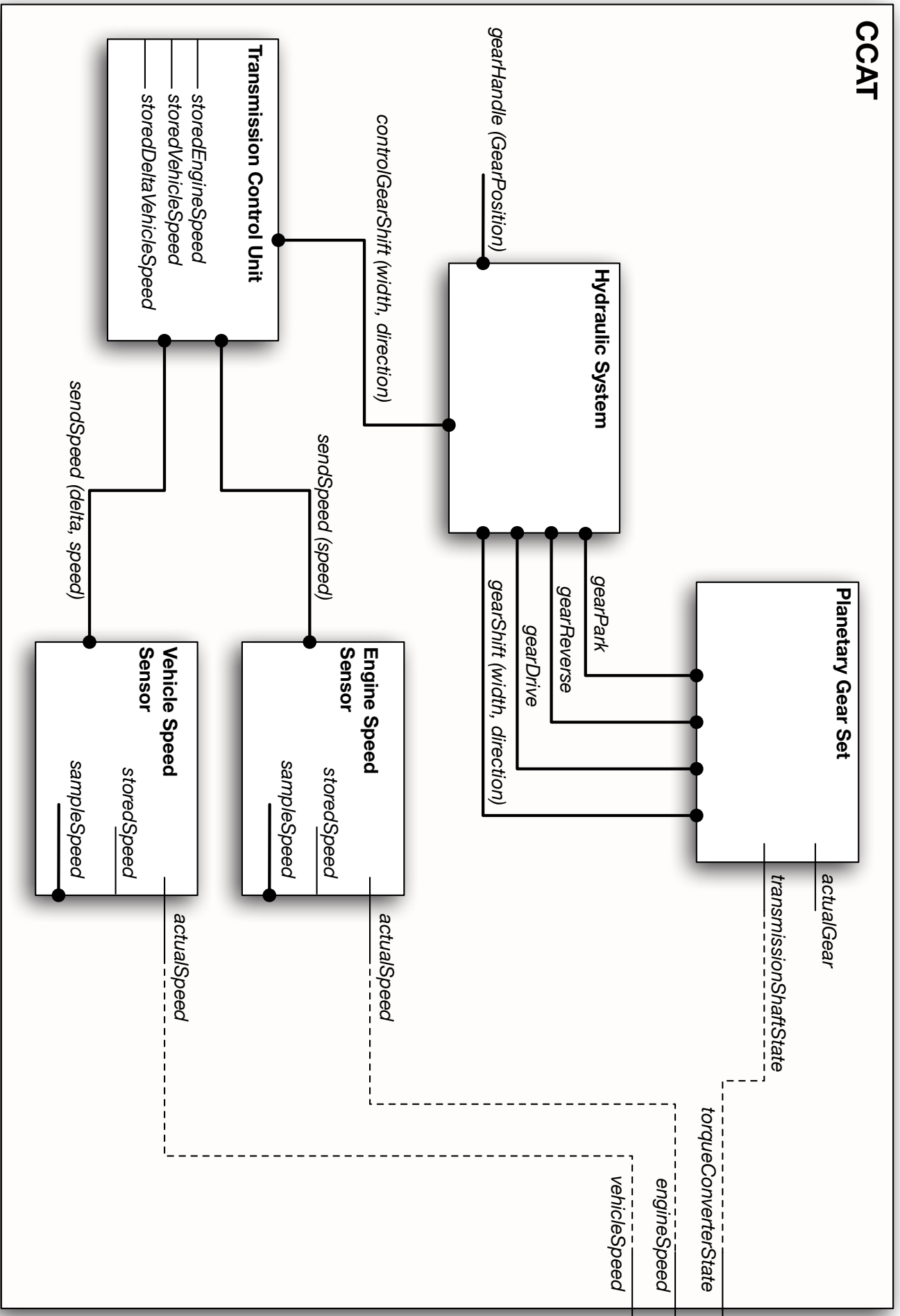


Figure 1: Computer Controller Automatic Transmission

Due to limitations in the verification tools 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.

The specification is reported in Listing 2.

Listing 2: ComputerControlledAutomaticTransmission.t2p

```

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     TCUOneUp = 1, TCUOneDown=2,
11     HandleDrive=1, HandlePark=2, HandleReverse=3,
12     ShiftOneUp=1, ShiftOneDown=2, ShiftDrive=3,
13     ShiftPark=4, ShiftReverse=5,
14     First=0, Second=1, Park=2, Reverse=3,
15     Detached=0, Attached=1,
16     FluidPropagationDelay=1, SingleGearShiftDelay=2,
17     DriveGearShiftDelay=3, ParkGearShiftDelay=3,
18     ReverseGearShiftDelay=3
19
20 axioms
21     Mechanics :
22         ( gearShift  $\Diamond$  Nothing  $\rightarrow$  ( gearShift=ShiftOneUp  $\rightarrow$ 
23             transmissionShaftState=Attached ) &
24             ( gearShift=ShiftDrive  $\rightarrow$ 
25                 transmissionShaftState=Detached ) &
26             ( gearShift=ShiftPark  $\rightarrow$ 
27                 transmissionShaftState=Detached ) &
28             ( gearShift=ShiftReverse  $\rightarrow$ 
29                 transmissionShaftState=Detached )) &
30         ( transmissionShaftState=Attached  $\rightarrow$ 
31             actualGear=First | actualGear=Second |
32             actualGear=Reverse );
33
34     PropagateGearShiftCommand :
35         controlGearShift  $\Diamond$  Nothing  $\rightarrow$ 
36         ( controlGearShift=TCUOneUp |
37             controlGearShift=TCUOneDown  $\rightarrow$ 

```



```

    Lasts_ii(gearHandle=Nothing ,
    FluidPropagationDelay) &
    Lasts_ei(controlGearShift=Nothing ,
    FluidPropagationDelay)) &
25 (controlGearShift=TCUOneUp ->
    Futr(gearShift=ShiftOneUp ,
    FluidPropagationDelay)) &
26 (controlGearShift=TCUOneDown ->
    Futr(gearShift=ShiftOneDown ,
    FluidPropagationDelay));

27
28 GearHandleCommand:
29   gearHandle<>Nothing ->
30   (gearHandle=HandleDrive ->
    Lasts_ii(controlGearShift=Nothing &
    transmissionShaftState=Detached ,
    FluidPropagationDelay) &
    Lasts_ei(gearHandle=Nothing ,
    FluidPropagationDelay) &
    Futr(gearShift=ShiftDrive ,
    FluidPropagationDelay)) &
31 (gearHandle=HandlePark ->
    Lasts_ii(controlGearShift=Nothing &
    transmissionShaftState=Detached ,
    FluidPropagationDelay) &
    Lasts_ei(gearHandle=Nothing ,
    FluidPropagationDelay) &
    Futr(gearShift=ShiftPark ,
    FluidPropagationDelay)) &
32 (gearHandle=HandleReverse ->
    Lasts_ii(controlGearShift=Nothing &
    transmissionShaftState=Detached ,
    FluidPropagationDelay) &
    Lasts_ei(gearHandle=Nothing ,
    FluidPropagationDelay) &
    Futr(gearShift=ShiftReverse ,
    FluidPropagationDelay));

33
34 GearShiftFirst:
35 (actualGear=First -> gearShift=Nothing |
    gearShift=ShiftOneUp | gearShift=ShiftPark |
    gearShift=ShiftReverse) &

```

```

36      (actualGear=First & gearShift=ShiftOneUp ->
        Lasts_ei(gearShift=Nothing,
          SingleGearShiftDelay) & Futr(actualGear=Second,
          SingleGearShiftDelay)) &
37      (actualGear=First & gearShift=ShiftPark ->
        Lasts_ii(transmissionShaftState=Detached,
          ParkGearShiftDelay) &
        Lasts_ei(gearShift=Nothing, ParkGearShiftDelay)
        & Futr(actualGear=Park, ParkGearShiftDelay)) &
38      (actualGear=First & gearShift=ShiftReverse ->
        Lasts_ii(transmissionShaftState=Detached,
          ReverseGearShiftDelay) &
        Lasts_ei(gearShift=Nothing,
          ReverseGearShiftDelay) &
        Futr(actualGear=Reverse,
          ReverseGearShiftDelay));
39
40      GearShiftSecond:
41      (actualGear=Second -> gearShift=Nothing |
        gearShift=ShiftOneDown) &
42      (actualGear=Second & gearShift=ShiftOneDown ->
        Lasts_ei(gearShift=Nothing,
          SingleGearShiftDelay) & Futr(actualGear=First,
          SingleGearShiftDelay));
43
44      GearShiftReverse:
45      gearShift <> Nothing ->
46      (actualGear=Reverse -> gearShift=Nothing |
        gearShift=ShiftDrive | gearShift=ShiftPark) &
47      (actualGear=Reverse & gearShift=ShiftDrive ->
        Lasts_ii(transmissionShaftState=Detached,
          DriveGearShiftDelay) &
        Lasts_ei(gearShift=Nothing,
          DriveGearShiftDelay) & Futr(actualGear=First,
          DriveGearShiftDelay)) &
48      (actualGear=Reverse & gearShift=ShiftPark ->
        Lasts_ii(transmissionShaftState=Detached,
          ParkGearShiftDelay) &
        Lasts_ei(gearShift=Nothing, ParkGearShiftDelay)
        & Futr(actualGear=Park, ParkGearShiftDelay));
49
50      GearShiftPark:

```

```

51      (actualGear=Park ->
52          transmissionShaftState=Detached) &
53      (actualGear=Park -> gearShift=Nothing |
54          gearShift=ShiftDrive | gearShift=ShiftReverse) &
55      (actualGear=Park & gearShift=ShiftDrive ->
56          Lasts_ii(transmissionShaftState=Detached,
57              DriveGearShiftDelay) &
58          Lasts_ei(gearShift=Nothing,
59              DriveGearShiftDelay) & Futr(actualGear=First,
60              DriveGearShiftDelay)) &
61      (actualGear=Park & gearShift=ShiftReverse ->
62          Lasts_ii(transmissionShaftState=Detached,
63              ReverseGearShiftDelay) &
64          Lasts_ei(gearShift=Nothing,
65              ReverseGearShiftDelay) &
66          Futr(actualGear=Reverse,
67              ReverseGearShiftDelay));
68
69  Property:
70      actualGear=First & controlGearShift=TCUOneUp & not
71          Futr(actualGear=Second,
72              FluidPropagationDelay+SingleGearShiftDelay);
73      actualGear=First & gearHandle=HandleReverse & (not
74          Futr(actualGear=Reverse,
75              FluidPropagationDelay+ReverseGearShiftDelay) |
76          not Lasts(transmissionShaftState=Detached,
77              FluidPropagationDelay+ReverseGearShiftDelay));
78      actualGear=First & gearHandle=HandlePark & (not
79          Futr(actualGear=Park,
80              FluidPropagationDelay+ParkGearShiftDelay) | not
81          Since(transmissionShaftState=Detached,
82              controlGearShift <math>\Diamond</math> Nothing |
83              gearHandle <math>\Diamond</math> Nothing));

```

2 Vehicle/EngineSpeedSensor Classes

The *VehicleSpeedSensor* is formalized thanks to the code reported in Listing 3 while the *EngineSpeedSensor* is formalized thanks to the code reported in Listing 4.

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 considered 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 occurs.

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 3: *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 ,
32             sampleInterval) and not Lasts (sampleSpeed ,
33             sampleInterval);
34
35     SamplingAction:
36         sampleSpeed implies Futr (deltaSpeed = actualSpeed
37             - storedSpeed and speed = actualSpeed and
38             sendSpeed (deltaSpeed , speed) and Lasts
39             (storedSpeed = actualSpeed , sampleInterval),
40             sampleDelay);
41
42     SendSpeed:
43         deltaSpeed = actualSpeed - storedSpeed and
44         actualSpeed = speed and sendSpeed (deltaSpeed ,
45         speed) implies Past (sampleSpeed , sampleDelay);
46
47 end

```

Listing 4: EngineSpeedSensor.trio

```

1 class EngineSpeedSensor (const sampleInterval , const
2     sampleDelay)
3
4 signature:
5
6 visible: actualSpeed , sendSpeed;
7
8 temporal domain: real;
9
10 items:
11     TI sampleInterval: real;
12     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 ,
28             sampleInterval) and not Lasts (sampleSpeed ,
29             sampleInterval);
30
31     SampleSpeedActions:
32         sampleSpeed implies Futr (actualSpeed = speed and
33             sendSpeed (speed), sampleDelay);
34
35     SendSpeed:
36         actualSpeed = speed and sendSpeed (speed) implies
37             Past (sampleSpeed , sampleDelay);
38
39 end
```

3 PlanetaryGearSet Class

The *PlanetaryGearSet* class is formalized thanks to the code reported in Listing 5.

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 5: 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  GearShiftSecond:
56
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  GearShiftThird:
62
63      (actualGear = Third implies not gearDrive and not
        gearPark and not gearReverse and not ex
        gearShiftWidth (gearShiftDirection = Up and
        gearShift (gearShiftWidth,

```

```

64         gearShiftDirection))) and
        (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 =
66          Second , singleGearShiftDelay)) and
        (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

```

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

4 HydraulicSystem Class

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

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 6: 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 <math>\Diamond</math> 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 <math>\Diamond</math> 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 <math>\Diamond</math> 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 ,

```

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

5 TransmissionControlUnit Class

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

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 7: TransmissionControlUnit.trio

```

1  class TransmissionControlUnit
2
3  signature:
4
5  visible:
6      controlGearShift ,
7      receiveEngineSpeed ,
8      receiveVehicleSpeed ;
9
10 temporal domain: real;
11
12 domains:
13     ShiftWidth: 1..2;
14     ShiftDirection: {Up, Down};
15
16 items:
17     TD total storedEngineSpeed: integer;
18     TD total storedDeltaVehicleSpeed: integer;
19     TD total storedVehicleSpeed: integer;
20     event controlGearShift (ShiftWidth, ShiftDirection);
21     event receiveEngineSpeed (integer);
22     event receiveVehicleSpeed (integer, integer);
23
24 axioms:
25 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 ,

```

```

45      gearShiftDirection1)) and
      (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 ,
146      (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 ,

```

```

51         vehicleSpeed) and engineSpeed >= 1500 and
           engineSpeed < 3000 implies all gearShiftWidth1 ,
           gearShiftDirection1 (not controlGearShift
           (gearShiftWidth1 , gearShiftDirection1))) and
52     (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
53     (all engineSpeed (not receiveEngineSpeed
           (engineSpeed)) and receiveVehicleSpeed
           (deltaVehicleSpeed , vehicleSpeed) and
           storedEngineSpeed >= 1500 and storedEngineSpeed
           < 3000 implies all gearShiftWidth1 ,
           gearShiftDirection1 (not controlGearShift
           (gearShiftWidth1 , gearShiftDirection1))) and
54     (all engineSpeed (not receiveEngineSpeed
           (engineSpeed)) and all deltaVehicleSpeed ,
           vehicleSpeed (not receiveVehicleSpeed
           (deltaVehicleSpeed , vehicleSpeed)) implies all
           gearShiftWidth1 , gearShiftDirection1 (not
           controlGearShift (gearShiftWidth1 ,
           gearShiftDirection1))));
55
56 ReceivingEventAction :
           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 its 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

Listing 8: Property 1

```
actualGear=First & controlGearShift=TCUOneUp & not  
  Futr(actualGear=Second,  
    FluidPropagationDelay+SingleGearShiftDelay);
```

Listing 9: Property 2

```
actualGear=First & gearHandle=HandleReverse & (not  
  Futr(actualGear=Reverse,  
    FluidPropagationDelay+ReverseGearShiftDelay) | not  
  Lasts(transmissionShaftState=Detached,  
    FluidPropagationDelay+ReverseGearShiftDelay));
```

Listing 10: Property 3

```
actualGear=First & gearHandle=HandlePark & (not  
  Futr(actualGear=Park,  
    FluidPropagationDelay+ParkGearShiftDelay) | not  
  Since(transmissionShaftState=Detached,  
    controlGearShift  $\Diamond$  Nothing | gearHandle  $\Diamond$  Nothing));
```