

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

Computer Controller Automatic Transmission
(Mandatory Part)

Alessandra Bonetto
Filippo Sironi
Matteo Villa

2009

Contents

1	CCAT Class	5
2	Vehicle/EngineSpeedSensor Classes	8
3	PlanetaryGearSet Class	11
4	HydraulicSystem Class	16
5	TransmissionControlUnit Class	18
6	Annotations	23

List of Figures

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

Listings

1	ComputerControlledAutomaticTransmission.trio	5
2	VehicleSpeedSensor.trio	8
3	EngineSpeedSensor.trio	9
4	PlanetaryGearSet.trio	11
5	HydraulicSystem.trio	16
6	TransmissionControlUnit.trio	18

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
```

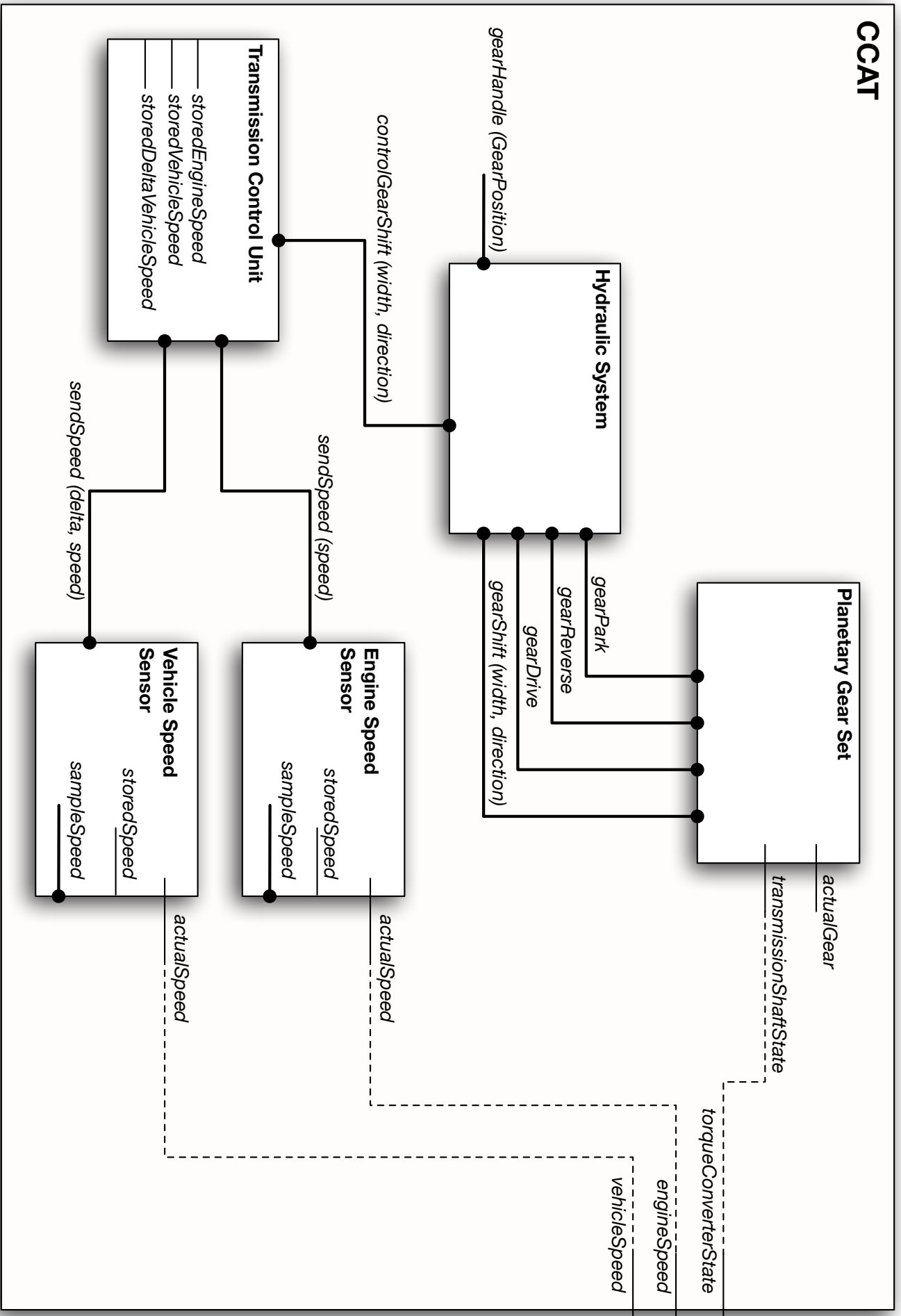


Figure 1: Computer Controller Automatic Transmission

2 Vehicle/EngineSpeedSensor Classes

The *VehicleSpeedSensor* is formalized thanks to the code reported in Listing 2 while the *EngineSpeedSensor* is formalized thanks to the code reported in Listing 3.

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 2: 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 3: EngineSpeedSensor.trio

```

1 class EngineSpeedSensor (const sampleInterval , const
2     sampleDelay)
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 ,  
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 4.

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`).

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 4: 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         GearDriveShift:
43             (actualGear = Reverse and gearDrive implies (Lasts
                (actualGear = Reverse, driveGearShiftDelay) and
                Futr (actualGear = First,
                    driveGearShiftDelay))) and
44             (actualGear = Park and gearDrive implies (Lasts
                (actualGear = Park, driveGearShiftDelay) and
                Futr (actualGear = First,
                    driveGearShiftDelay))) and
45             (actualGear = First or actualGear = Second or
                actualGear = Third implies not gearDrive) and
46             (gearDrive iff transmissionShaftState = Detached);
47
48         GearShiftsFirst:
49             (actualGear = First implies Alw (not gearDrive and
                not ex gearShiftWidth (gearShiftDirection = Down
                    and gearShift (gearShiftWidth,
                        gearShiftDirection)))) and
50             (actualGear = First and gearShiftWidth = 1 and
                gearShiftDirection = Up and gearShift
                (gearShiftWidth, gearShiftDirection) implies
                Lasts (actualGear = First,
                    singleGearShiftDelay) and Futr (actualGear =
                    Second, singleGearShiftDelay)) and
51             (actualGear = First and gearShiftWidth = 2 and
                gearShiftDirection = Up and gearShift

```

```

52         (gearShiftWidth , gearShiftDirection) implies
53         Lasts (actualGear = First , dualGearShiftDelay)
54         and Futr (actualGear = Third ,
        dualGearShiftDelay);

52 GearShiftsSecond :
53     (actualGear = Second implies Alw (not gearDrive and
54         not gearPark and not gearReverse and not ex
        gearShiftDirection (gearShiftWidth = 2 and
        gearShift (gearShiftWidth ,
55         gearShiftDirection)))) and
        (actualGear = Second and gearShiftWidth = 1 and
        gearShiftDirection = Up and gearShift
        (gearShiftWidth , gearShiftDirection) implies
        Lasts (actualGear = Second ,
56         singleGearShiftDelay) and Futr (actualGear =
        Third , singleGearShiftDelay)) and
        (actualGear = Second and gearShiftWidth = 1 and
        gearShiftDirection = Down and gearShift
        (gearShiftWidth , gearShiftDirection) implies
        Lasts (actualGear = Second ,
57         singleGearShiftDelay) and Futr (actualGear =
58         First , singleGearShiftDelay));

58 GearShiftsThird :
59     (actualGear = Third implies Alw (not gearDrive and
        not gearPark and not gearReverse and not ex
        gearShiftWidth (gearShiftDirection = Up and
        gearShift (gearShiftWidth ,
60         gearShiftDirection)))) and
        (actualGear = Third and gearShiftWidth = 1 and
        gearShiftDirection = Down and gearShift
        (gearShiftWidth , gearShiftDirection) implies
        Lasts (actualGear = Third ,
        singleGearShiftDelay) and Futr (actualGear =
        Second , singleGearShiftDelay)) and
61     (actualGear = Third and gearShiftWidth = 2 and
        gearShiftDirection = Down and gearShift
        (gearShiftWidth , gearShiftDirection) implies
        Lasts (actualGear = Third , dualGearShiftDelay)
        and Futr (actualGear = First ,
        dualGearShiftDelay));
62

```

```

63 GearShiftsReverse:
64     (actualGear = Reverse implies Alw (not gearReverse
        and all gearShiftWidth, gearShiftDirection (not
        gearShift (gearShiftWidth,
        gearShiftDirection)))) and
65     (actualGear = Reverse implies SomF (gearDrive or
        gearPark)) and (actualGear = First and
        gearReverse implies Lasts (actualGear = First,
        reverseGearShiftDelay) and Futr (actualGear =
        Reverse, reverseGearShiftDelay)) and
66     (actualGear = Park and gearReverse implies Lasts
        (actualGear = Park, reverseGearShiftDelay) and
        Futr (actualGear = Reverse,
        reverseGearShiftDelay)) and
67     (gearReverse iff transmissionShaftState =
        Detached);
68
69 GearShiftsPark:
70     (actualGear = Park implies Alw (not gearPark and
        all gearShiftWidth, gearShiftDirection (not
        gearShift (gearShiftWidth,
        gearShiftDirection)))) and
71     (actualGear = Park implies SomF (gearDrive or
        gearReverse)) and
72     (actualGear = First and gearPark implies Lasts
        (actualGear = First, parkGearShiftDelay) and
        Futr (actualGear = Park, parkGearShiftDelay))
        and
73     (actualGear = Reverse and gearPark implies Lasts
        (actualGear = Reverse, parkGearShiftDelay) and
        Futr (actualGear = Park, parkGearShiftDelay) Futr
        (actualGear = Park, parkGearShiftDelay)) and
74     (gearPark iff transmissionShaftState = Detached);
75
76 GearShiftsTimings:
77     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))

```

```

78         and
79         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
80         ((actualGear = Reverse and gearDrive) implies not
            Lasts (gearDrive or gearPark or gearReverse or
            ex gearShiftWidth2, gearShiftDirection2
            (gearShift (gearShiftWidth2,
            gearShiftDirection2)), driveGearShiftDelay)) and
81         ((actualGear = Reverse and gearPark) implies not
            Lasts (gearDrive or gearPark or gearReverse or
            ex gearShiftWidth2, gearShiftDirection2
            (gearShift (gearShiftWidth2,
            gearShiftDirection2)), parkGearShiftDelay)) and
82         ((actualGear = Park and gearDrive) implies not
            Lasts (gearDrive or gearPark or gearReverse or
            ex gearShiftWidth2, gearShiftDirection2
            (gearShift (gearShiftWidth2,
            gearShiftDirection2)), driveGearShiftDelay)) and
            ((actualGear = Park and gearReverse) implies not
            Lasts (gearDrive or gearPark or gearReverse or
            ex gearShiftWidth2, gearShiftDirection2
            (gearShift (gearShiftWidth2,
            gearShiftDirection2)), reverseGearShiftDelay));
83
84     Nothing:
85     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));
86
87 end

```

4 HydraulicSystem Class

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

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 5: 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         gearShiftDirection: ShiftDirection;
35     formulae:
36         GearHandle:
37             (gear = Drive and gearHandle (gear) implies Futr
38              (gearDrive, fluidPropagationDelay) and
39              (gear = Park and gearHandle (gear) implies Futr
40              (gearPark, fluidPropagationDelay) and
41              (gear = Reverse and gearHandle (gear) implies Futr
42              (gearReverse, fluidPropagationDelay));
43
44         GearHandleEvent:
45             (gearDrive implies not Lasts (gearReverse or
46              gearPark, fluidPropagationDelay)) and
47             (gearReverse implies not Lasts (gearDrive or
48              gearPark, fluidPropagationDelay)) and
49             (gearPark implies not Lasts (gearReverse or
50              gearDrive, fluidPropagationDelay));
51
52         PropagateGearShiftCommand:
53             all gearShiftWidth, gearShiftDirection
54             (controlGearShift (gearShiftWidth,
55              gearShiftDirection) implies Futr (gearShift
56              (gearShiftWidth, gearShiftDirection),
57              fluidPropagationDelay));
58
59         MutualExclusions:
60             all gear (gearHandle (gear) implies all gear2 (gear
61              <> gear2 implies not gearHandle (gear2)));
62     end

```

5 TransmissionControlUnit Class

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

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 6: 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 ,
           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 ,
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

```

```

51      engineSpeed < 3000 implies all gearShiftWidth1 ,
      gearShiftDirection1 (not controlGearShift
      (gearShiftWidth1 , gearShiftDirection1))) and
      (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 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.