TD to UML-B

using

Atlas Transformation Language (ATL)

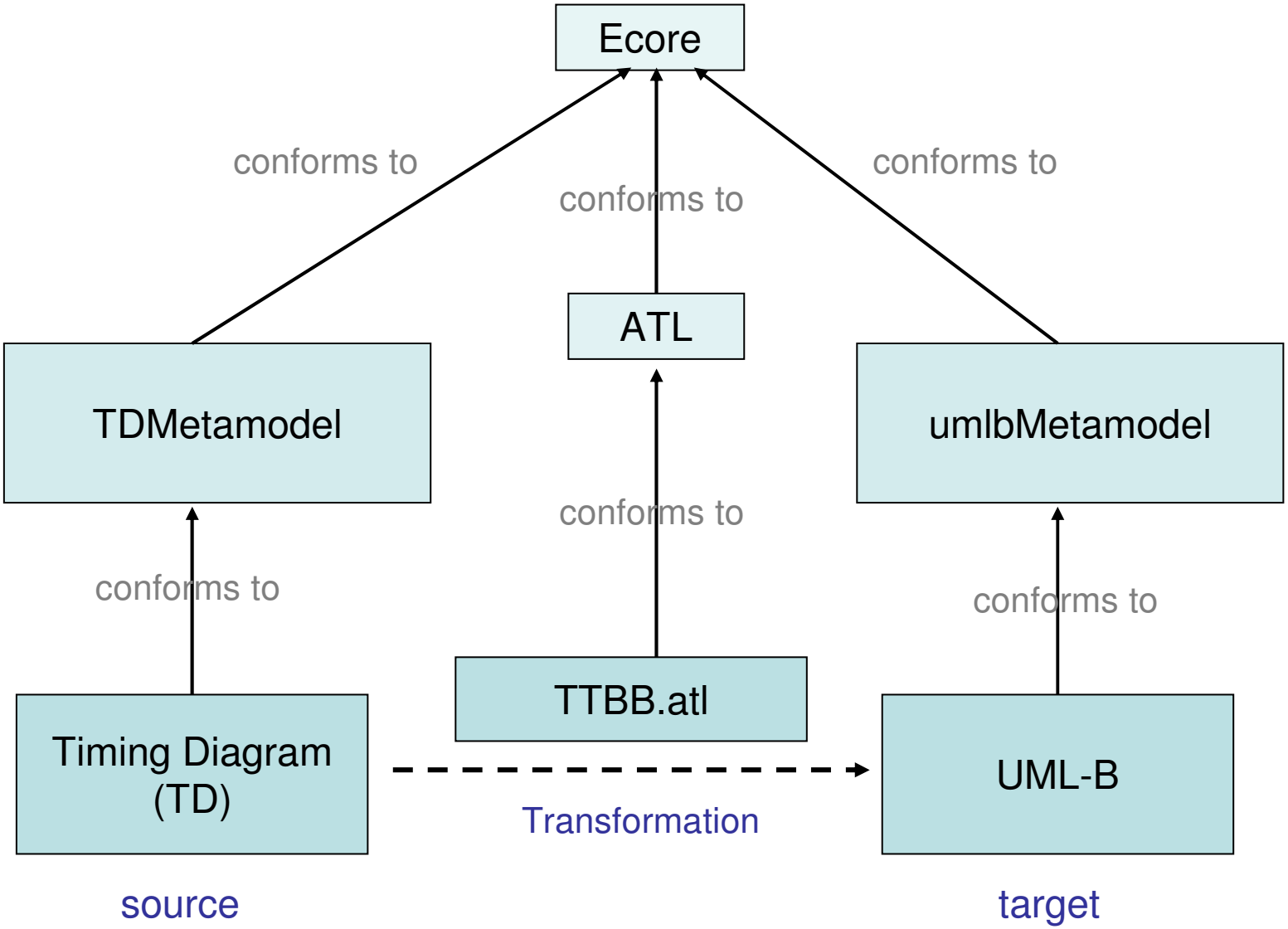
Outlines

- ATL Overview
- Structure of ATL Module
- Timing diagram Metamodel & UML-B Metamodel
- Examples of ATL transformation rules
- Examples of UML-B models are created from ATL
- Examples of Event-B models are created from UML-B
- Point out the manually additional information
- Ongoing works

ATL

- ATL transformation program is composed of rules that define how source model elements are matched and navigated to create and initialize the elements of the target model.

Overview of the TD to UML-B transformation



Structure of ATL Module

- Header
- Imports
- Helpers
- Transformation rules

Structure of ATL Module (cont')

- **Header**
  - defines name of transformation model
  - declares the source and target models

```
- - @atlcompiler atl2006
module TTBB;
create OUT : umlbMetamodel  from IN : TDMetamodel;
```

- **Imports**
  - declare which ATL libraries have to be imported

```
uses strings;
```

Structure of ATL Module (cont')

- **Helpers**
  - Can be viewed as Java methods
  - Make it possible to defined factorized ATL code that can be called from different points of an ATL transformation

```
helper context  TDMetamodel!TDNodeType
        def  :  SimpleCond() : String =
                self.predicates -> iterate(e; ret : String = '' |
                                                ret -> concat(' & ' + e.predicate));
```

- **Transformation rules** express the transformation logic

```
rule Machine {
from t   : TDMetamodel!TDMachine
to     m  : umlbMetamodel!UMLBMachine
            (name <- t.name,
             classes <- t.class),

       e : umlbMetamodel!UMLBEvent
            (name <- 'ticktok'),

       a : umlbMetamodel!UMLBAction
            (name <- 'Action1',
             action <- 'gclock := gclock + 1'),

    gclk : umlbMetamodel!UMLBVariable
            (name <- 'gclock',
             initialValue <- '0'),        - -  initialValue is defined in UMLBvariableElement

     ctx : umlbMetamodel!UMLBContext
            (name <- t.name + '_ctx' )

do {m.events <- m.events.append(e);
      e.actions <- e.actions.append(a);
    m.variables <- m.variables.append(gclk);
    thisModule.umlbproject.constructs <- thisModule.umlbproject.constructs.append(ctx);
    thisModule.umlbmachine <- m;
    }   }
```

Rule's name

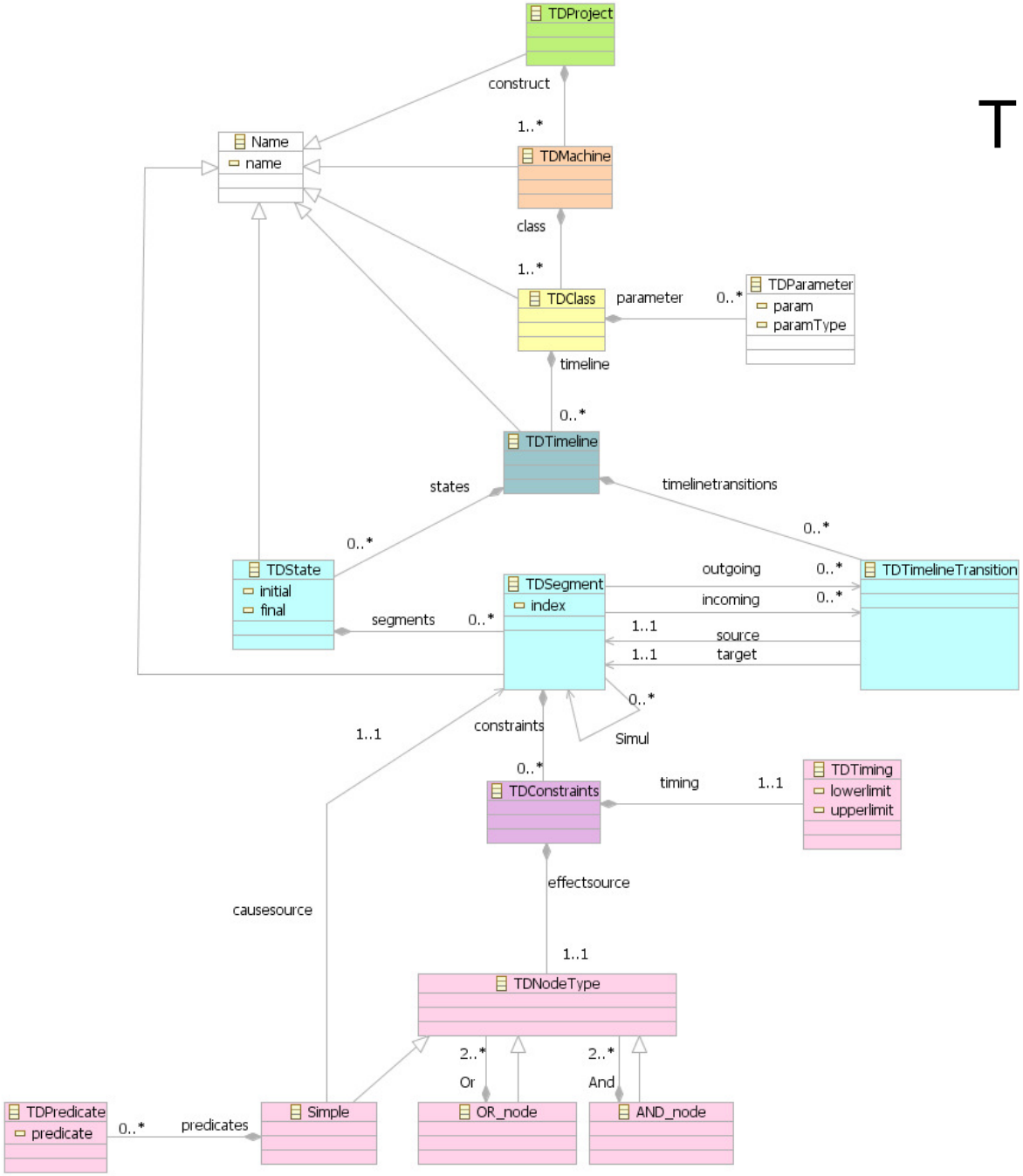
Source model

target models

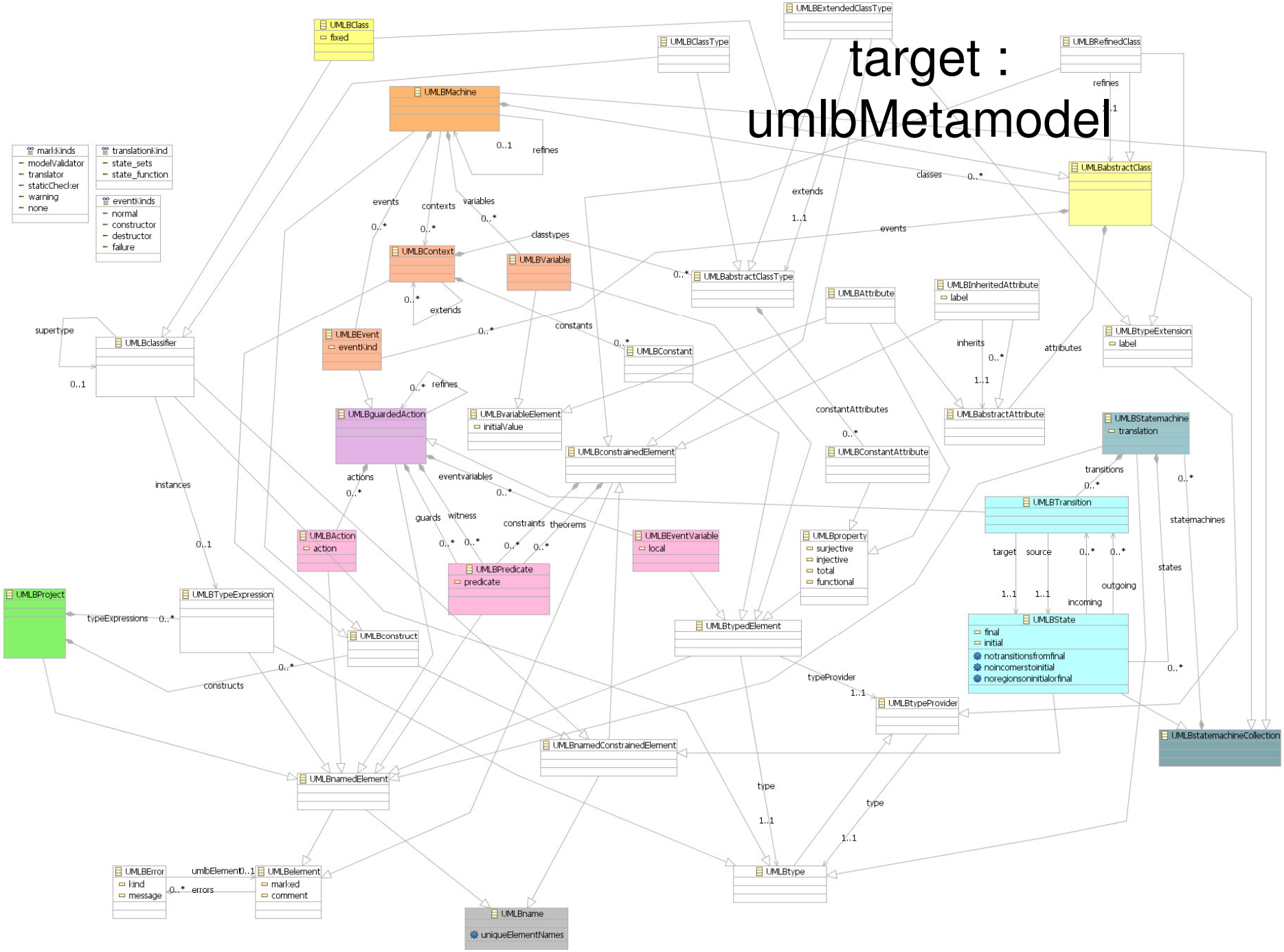
"- -" comment

"do" specify some imperative code that will be executed after the initialization of the target elements generated by the rule

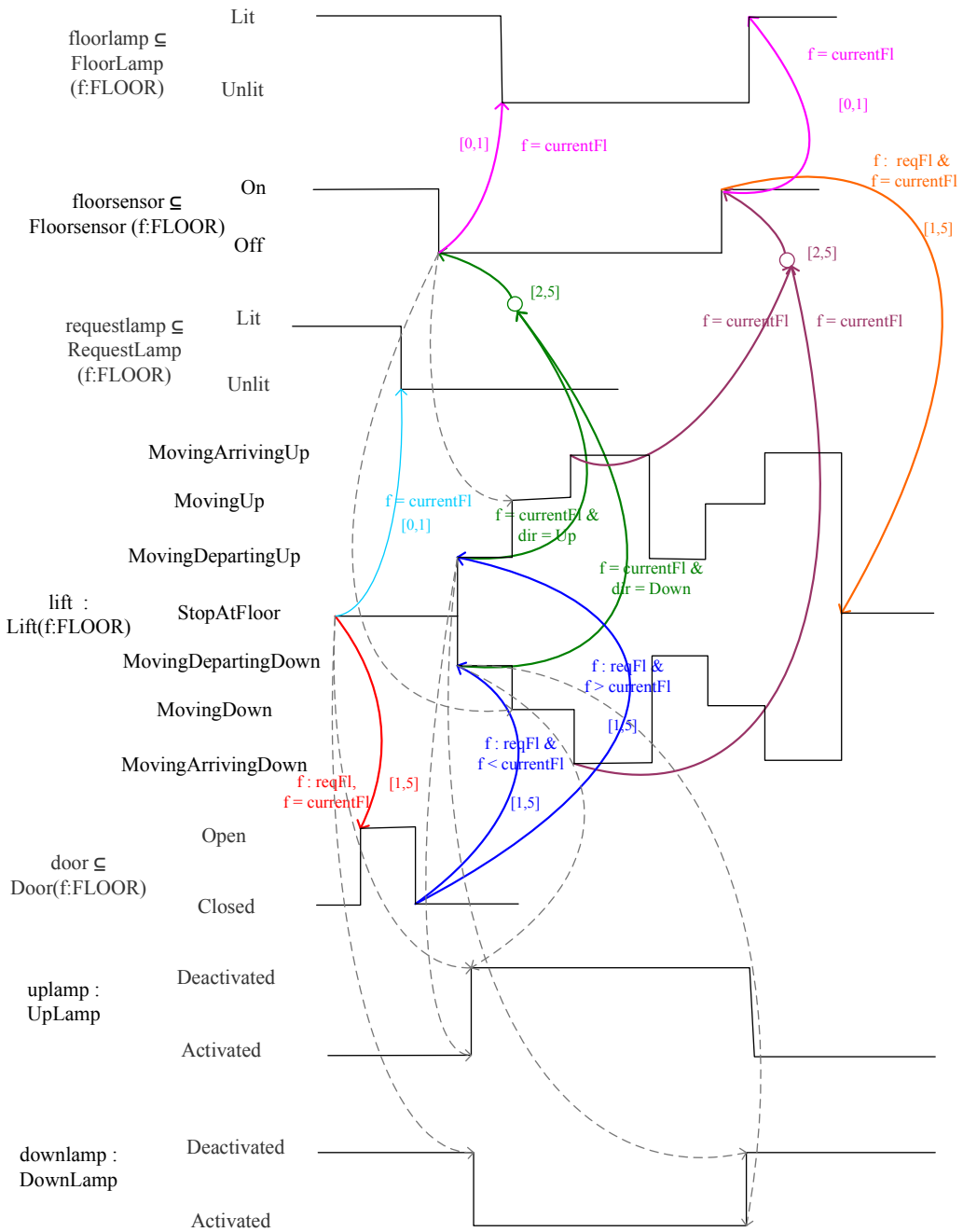
source : TDMetamodel



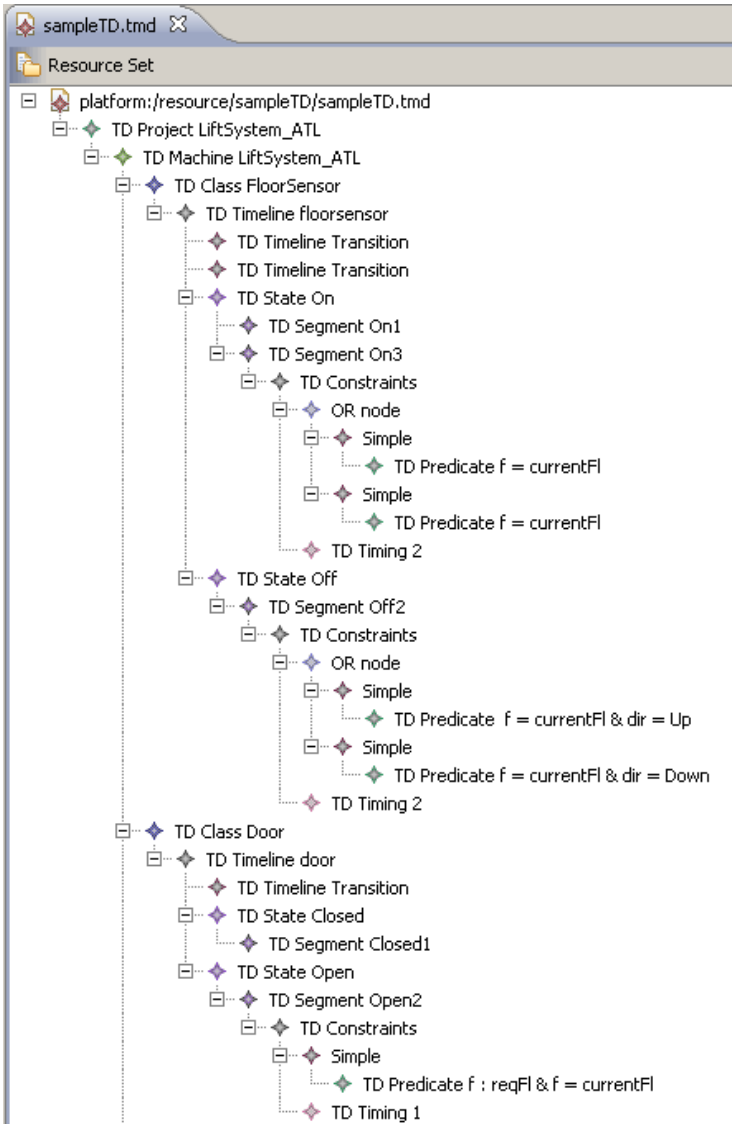
target : umlbMetamodel



TD
Lift Specifications

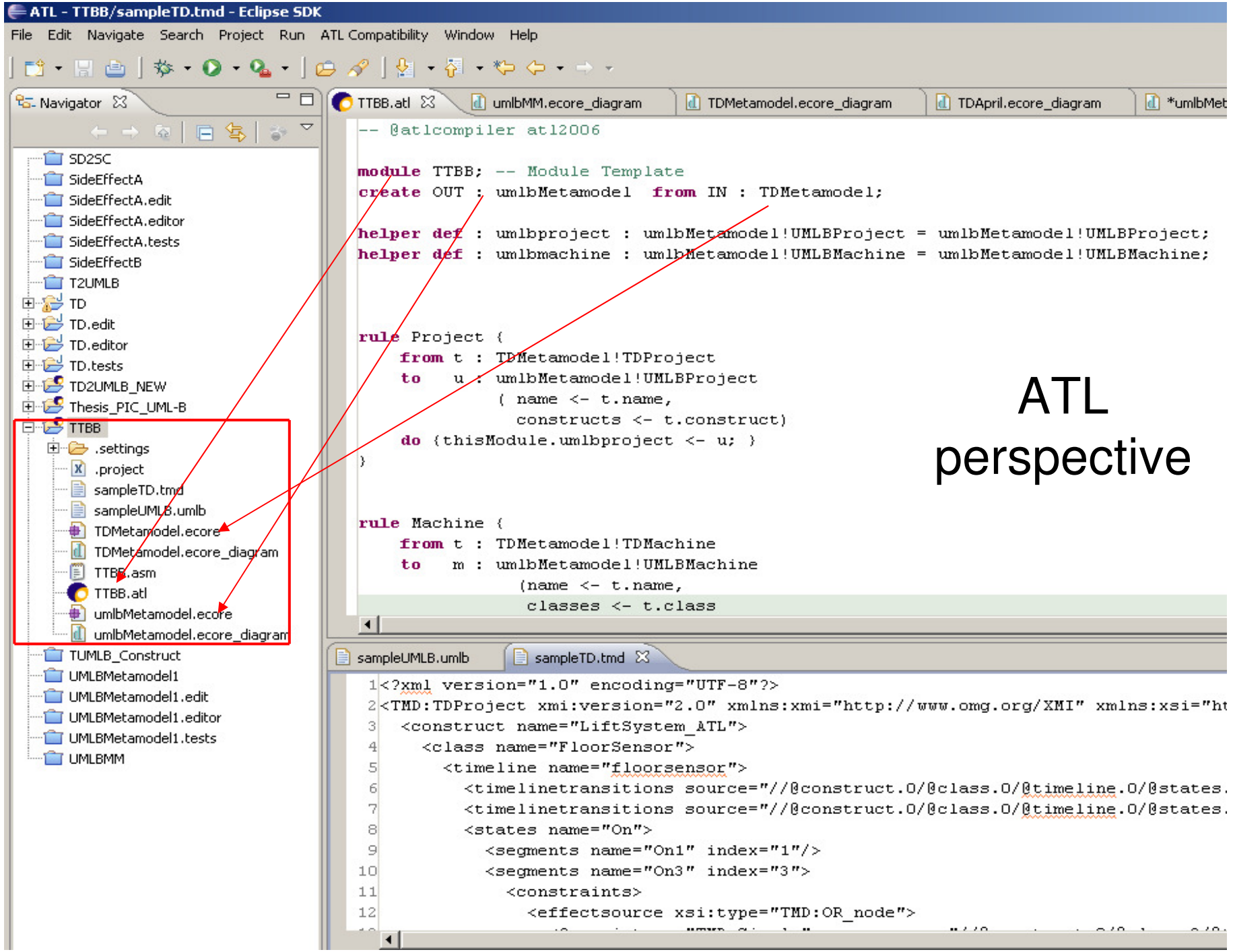


Input : sampleTD.tmd (ecore)



Input : sampleTD.tmd (xml)

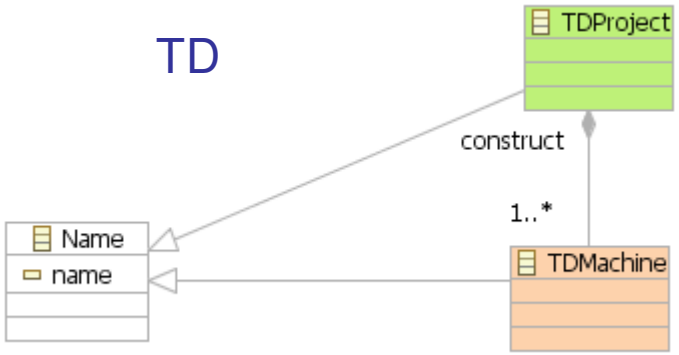
```
<?xml version="1.0" encoding="UTF-8"?>
<TMD:TDProject xmi:version="2.0" xmlns:xmi="http://www.omg.org/XMI
  <construct name="LiftATL">
    <class name="FloorSensor">
      <timeline name="floorsensor">
        <timelinetransitions source="//@construct.0/@class.0/@time
        <states name="On">
          <segments name="On1" index="1"/>
        </states>
        <states name="Off">
          <segments name="Off2" index="2">
            <constraints>
              <effectsource xsi:type="TMD:OR_node">
                <Or xsi:type="TMD:Simple" causesource="//@construc
                  <predicates predicate=" f = currentFl &amp; dir
                </Or>
                <Or xsi:type="TMD:Simple" causesource="//@construc
                  <predicates predicate="f = currentFl &amp; dir =
                </Or>
              </effectsource>
              <timing lowerlimit="2" upperlimit="5"/>
            </constraints>
          </segments>
        </states>
      </timeline>
    </class>
    <class name="Door">
      <timeline name="door">
        <timelinetransitions source="//@construct.0/@class.1/@time
        <states name="Closed">
          <segments name="Closed1" index="1"/>
        </states>
        <states name="Open">
          <segments name="Open2" index="2">
            <constraints>
              <effectsource xsi:type="TMD:Simple" causesource="//@
                <predicates predicate="f : reqFl &amp; f = current
              </effectsource>
              <timing lowerlimit="1" upperlimit="5"/>
            </constraints>
          </segments>
        </states>
      </timeline>
```


ATL - TTBB/sampleTD.tmd - Eclipse SDK
File Edit Navigate Search Project Run ATL Compatibility Window Help
Navigator
SD2SC
SideEffectA
SideEffectA.edit
SideEffectA.editor
SideEffectA.tests
SideEffectB
T2UMLB
TD
TD.edit
TD.editor
TD.tests
TD2UMLB_NEW
Thesis_PIC_UML-B
TTBB
.settings
.project
sampleTD.tmd
sampleUMLB.umlb
TDMetamodel.ecore
TDMetamodel.ecore_diagram
TTBB.asm
TTBB.atl
umlbMetamodel.ecore
umlbMetamodel.ecore_diagram
TUMLB_Construct
UMLBMetamodel1
UMLBMetamodel1.edit
UMLBMetamodel1.editor
UMLBMetamodel1.tests
UMLBMM
TTBB.atl
umlbMM.ecore_diagram
TDMetamodel.ecore_diagram
TDApril.ecore_diagram
-- @atlcompiler atl2006
module TTBB; -- Module Template
create OUT : umlbMetamodel from IN : TDMetamodel;
helper def : umlbproject : umlbMetamodel!UMLBProject = umlbMetamodel!UMLBProject;
helper def : umlbmachine : umlbMetamodel!UMLBMachine = umlbMetamodel!UMLBMachine;
rule Project {
from t : TDMetamodel!TDProject
to u : umlbMetamodel!UMLBProject
(name <- t.name,
constructs <- t.construct)
do {thisModule.umlbproject <- u; }
}
rule Machine {
from t : TDMetamodel!TDMachine
to m : umlbMetamodel!UMLBMachine
(name <- t.name,
classes <- t.class
sampleUMLB.umlb
sampleTD.tmd
<?xml version="1.0" encoding="UTF-8"?>
<TMD:TDProject xmi:version="2.0" xmlns:xmi="http://www.omg.org/XMI" xmlns:xsi="ht
<construct name="LiftSystem_ATL">
<class name="FloorSensor">
<timeline name="floorsensor">
<timelinetransitions source="//@construct.0/@class.0/@timeline.0/@states.
<timelinetransitions source="//@construct.0/@class.0/@timeline.0/@states.
<states name="On">
<segments name="On1" index="1"/>
<segments name="On3" index="3">
<constraints>
<effectsource xsi:type="TMD:OR_node">
ATL
perspective

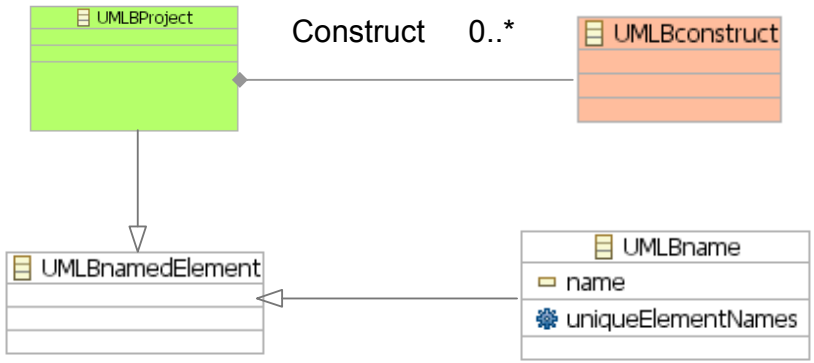
Example ATL rule : Project

```
rule Project {
from t : TDMetamodel!TDProject
to     u :
     umlbMetamodel!UMLBProject
            ( name <- t.name,
              constructs <- t.construct)
do {thisModule.umlbproject <- u; }
}
```

TD

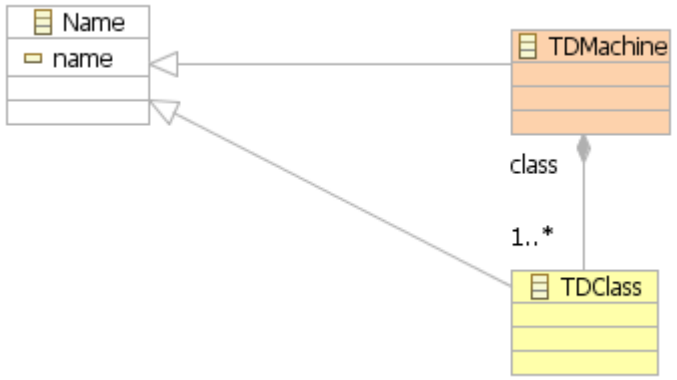


UMLB

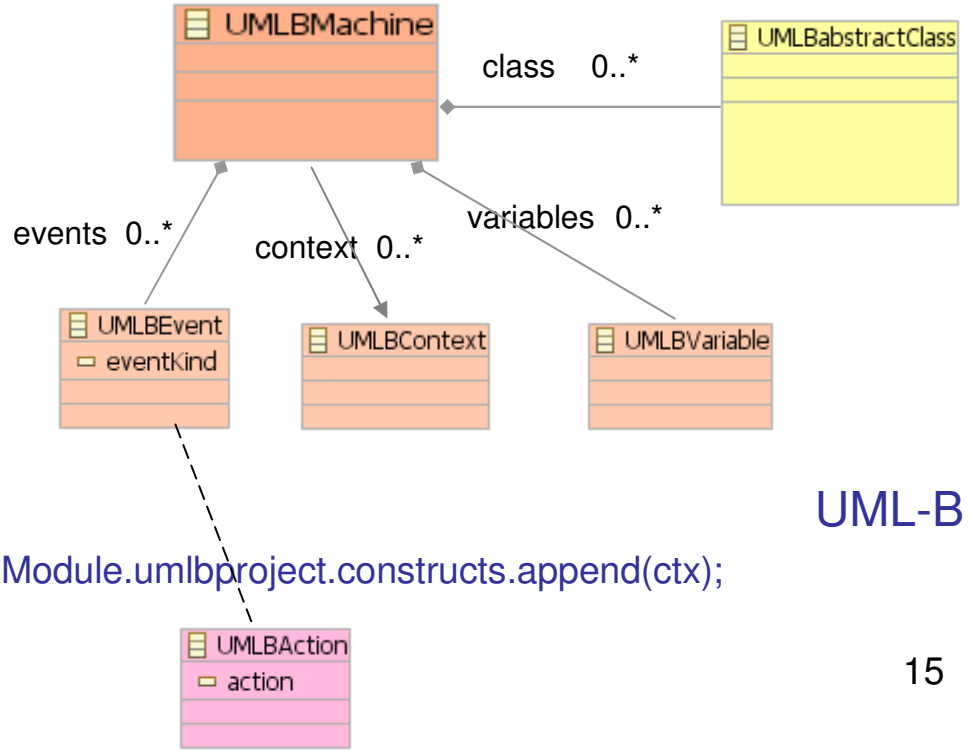


Example ATL rule : Machine

```
rule Machine {
from t   : TDMetamodel!TDMachine
to     m  : umlbMetamodel!UMLBMachine
            (name <- t.name,
             classes <- t.class),

       e : umlbMetamodel!UMLBEvent
            (name <- 'ticktok'),

       a : umlbMetamodel!UMLBAction
            (name <- 'Action1',
              action <- 'gclock := gclock + 1'),

    gclk : umlbMetamodel!UMLBVariable
            (name <- 'gclock',
             initialValue <- '0'),

     ctx : umlbMetamodel!UMLBContext
            (name <- t.name + '_ctx' )

 do {m.events <- m.events.append(e);
       e.actions <- e.actions.append(a);
     m.variables <- m.variables.append(gclk);
     thisModule.umlbproject.constructs <- thisModule.umlbproject.constructs.append(ctx);
     thisModule.umlbmachine <- m;
     }   }
```



TD

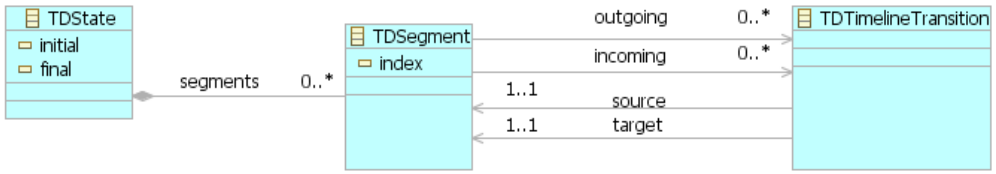


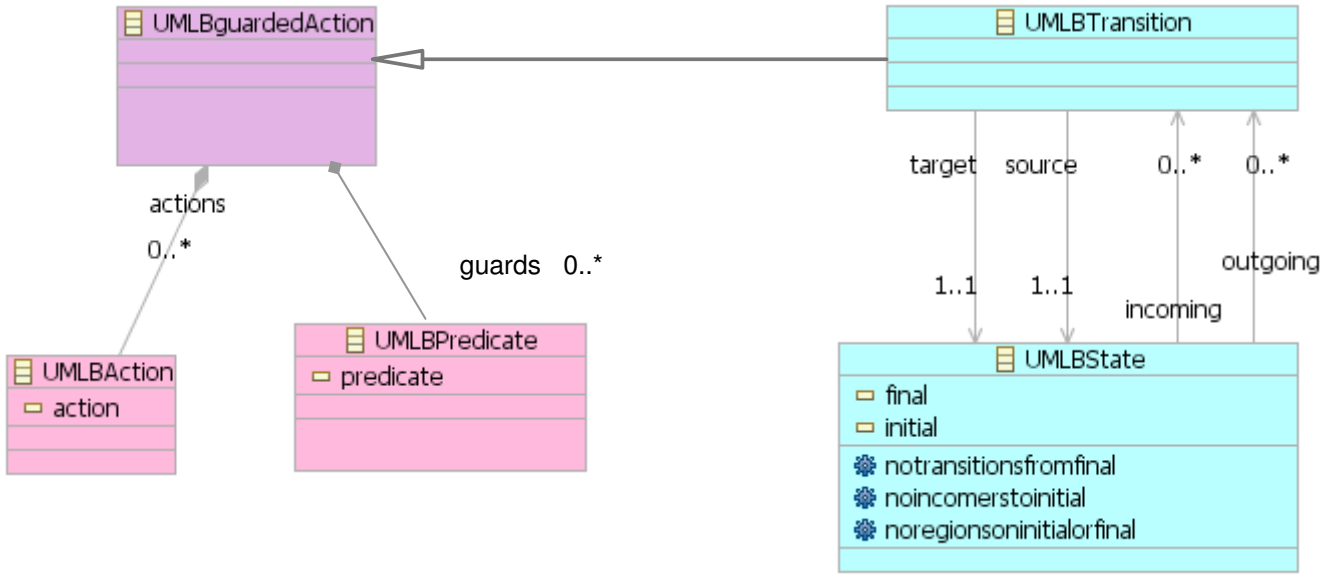
UML-B

Example ATL rule : Transition

```
rule Transition {
from t : TDMetamodel!TDTimelineTransition
to   u : umlbMetamodel!UMLBTransition
        (  name <- t.target.getTransitionName(),
           target <- t.target.eContainer(),
           source <- t.source.eContainer(),
           guards <- t.target.constraints,
           actions <- t.target.Simul
         ) ,
. . . ..
}
```

TD





UML-B

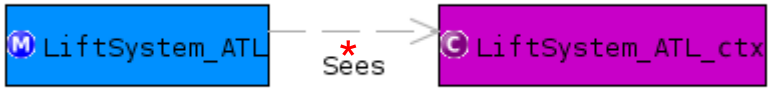
helper : getTransitionName()

```
helper context  TDMetamodel!TDSegment
        def :  getTransitionName() : String =
        let simuls : Set(TDMetamodel!TDSegment) =
        TDMetamodel!TDSegment.allInstances() ->
                                                    select(c|c.Simul -> includes(self))
        in
          if simuls -> isEmpty() then
            self.eContainer().eContainer().name + self.eContainer().name
       else
            simuls.last().getTransitionName()
       endif;
```

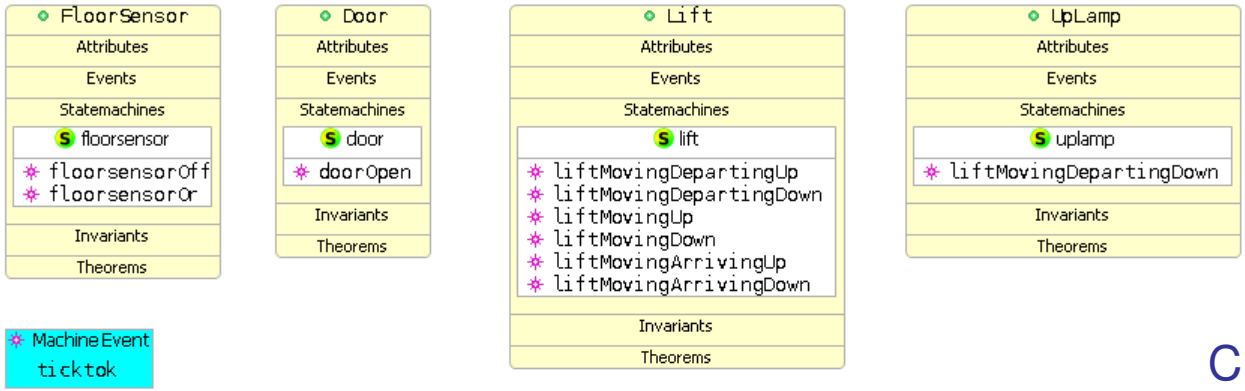
Output : sampleUMLB.umlb

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<umlbMetamodel:UMLBProject xmi:version="2.0" xmlns:xmi="http://www.omg.org/XMI" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xmlns:umlbMetamodel="http:///umlbMetamodel.ecore" name="LiftSystemATL">
  <constructs xsi:type="umlbMetamodel:UMLBMachine" name="LiftATL">
    <variables name="gclock" initialValue="0"/>
    <variables name="floorsensorOnTime" initialValue="0"/>
    <variables name="floorsensorOffTime" initialValue="0"/>
    <variables name="doorClosedTime" initialValue="0"/>
    <variables name="doorOpenTime" initialValue="0"/>
    <variables name="liftStopAtFloorTime" initialValue="0"/>
    <variables name="liftMovingNearUpTime" initialValue="0"/>
    <variables name="liftMovingNearDownTime" initialValue="0"/>
    <variables name="liftMovingUpTime" initialValue="0"/>
    <variables name="liftMovingDownTime" initialValue="0"/>
    <variables name="uplampActivatedTime" initialValue="0"/>
    <variables name="uplampDeactivatedTime" initialValue="0"/>
    <variables name="downlampActivatedTime" initialValue="0"/>
    <variables name="downlampDeactivatedTime" initialValue="0"/>
    <classes xsi:type="umlbMetamodel:UMLBClass" name="FloorSensor">
      <statemachines name="floorsensor">
        <states name="On" outgoing="//@constructs.0/@classes.0/@statemachines.0/@transitions.0"/>
        <states name="Off" incoming="//@constructs.0/@classes.0/@statemachines.0/@transitions.0"/>
        <transitions name="floorsensorOff" target="//@constructs.0/@classes.0/@statemachines.0/@states.1" source="//@constructs.0/@classes.0/@statemachines.0/@states.0">
          <actions name="gClockAction" action="floorsensorOffTime := gclock"/>
          <guards name="TimingCnstrntGuard" predicate="((gclock - liftMovingNearUptime >= 2)  &amp;amp; (gclock - liftMovingNearUptime &amp;lt;= 5) &amp;amp;  f = currentFl &amp;amp; dir = Up) or
            ((gclock - liftMovingNearDowntime >= 2)  &amp;amp; (gclock - liftMovingNearDowntime &amp;lt;= 5) &amp;amp; f = currentFl &amp;amp; dir = Down)"/>
        </transitions>
      </statemachines>
    </classes>
    <classes xsi:type="umlbMetamodel:UMLBClass" name="Door">
      <statemachines name="door">
        <states name="Closed" outgoing="//@constructs.0/@classes.1/@statemachines.0/@transitions.0"/>
        <states name="Open" incoming="//@constructs.0/@classes.1/@statemachines.0/@transitions.0"/>
        <transitions name="doorOpen" target="//@constructs.0/@classes.1/@statemachines.0/@states.1" source="//@constructs.0/@classes.1/@statemachines.0/@states.0">
          <actions name="gClockAction" action="doorOpenTime := gclock"/>
          <guards name="TimingCnstrntGuard" predicate="(gclock - liftStopAtFloortime >= 1)  &amp;amp; (gclock - liftStopAtFloortime &amp;lt;= 5) &amp;amp; f : reqFl &amp;amp; f = currentFl"/>
        </transitions>
      </statemachines>
    </classes>

. . . . . .
```

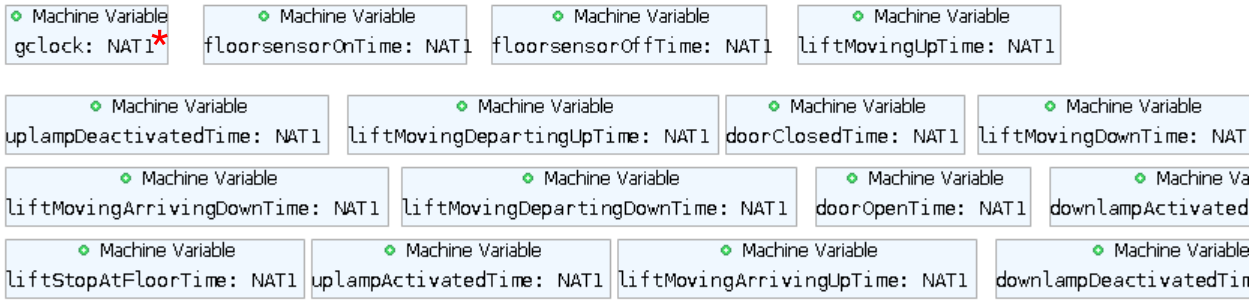
Parts of UML-B are created from ATL

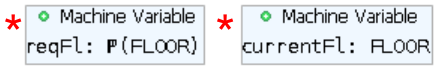


Package diagram,
Context diagram*



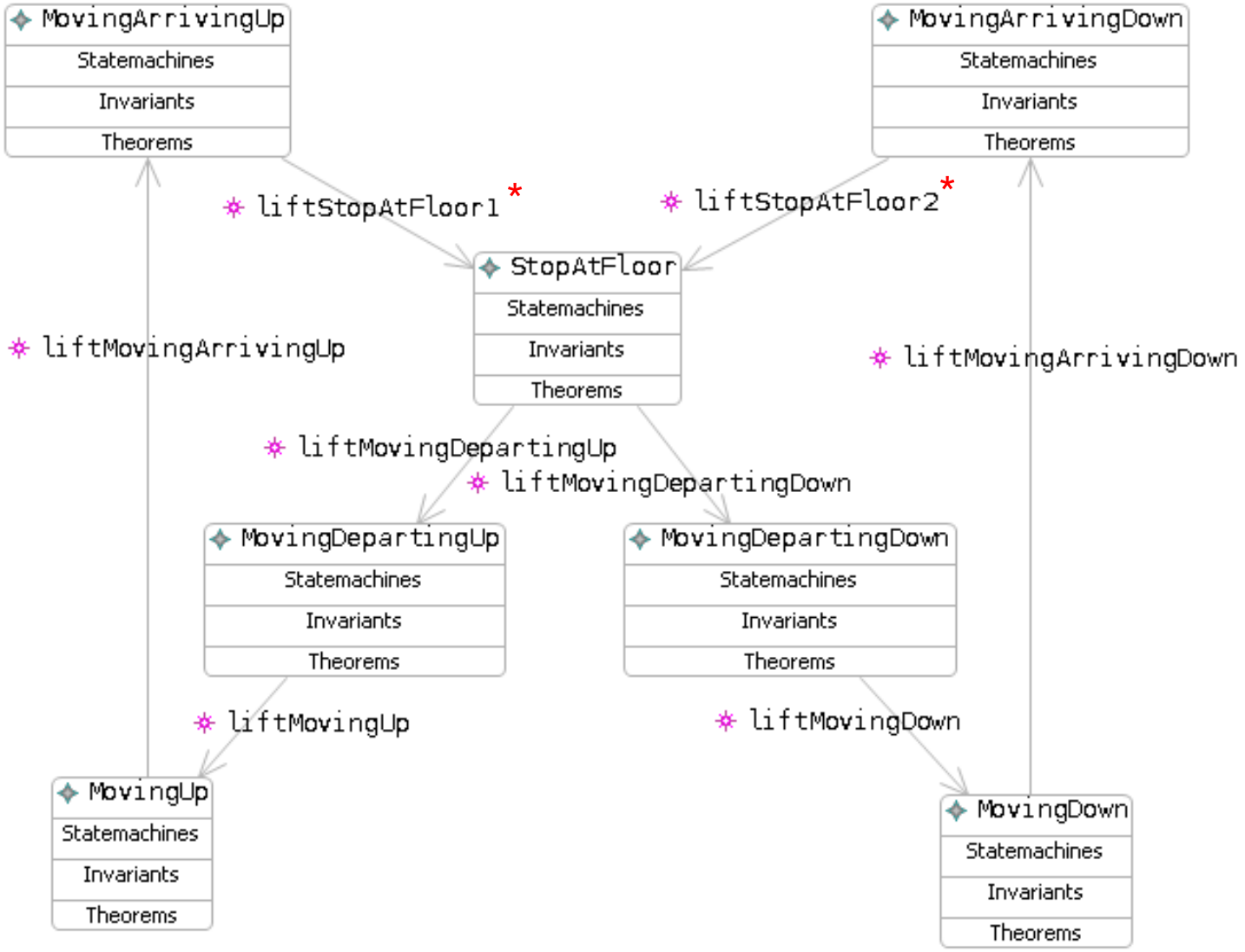
Class diagram



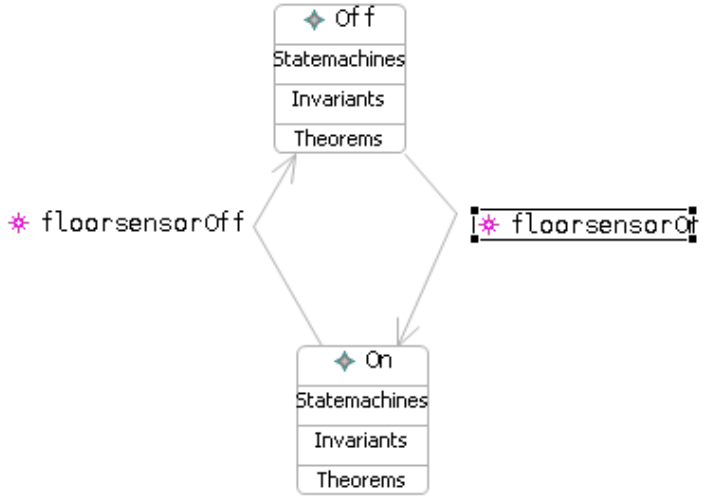


Note : * Manually create

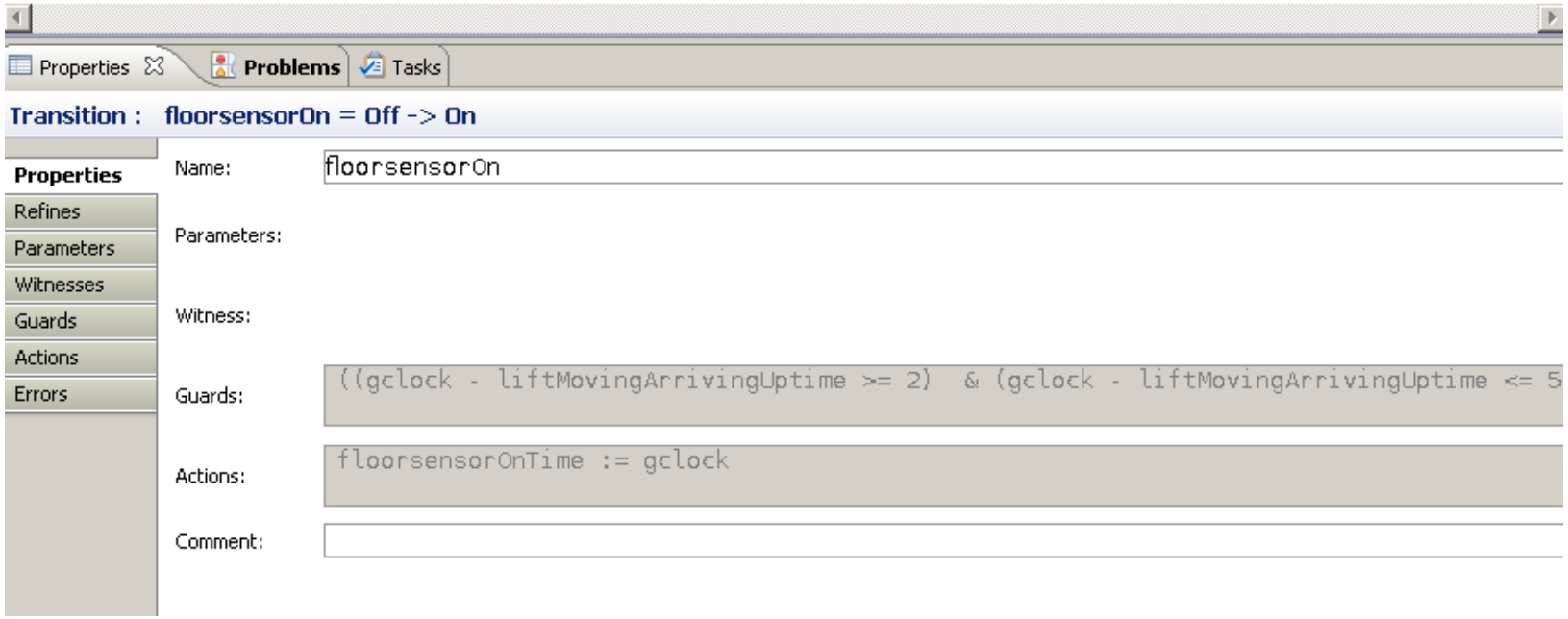
Parts of UML-B are created from ATL (cont')



Examples of UML-B are created from ATL (cont')



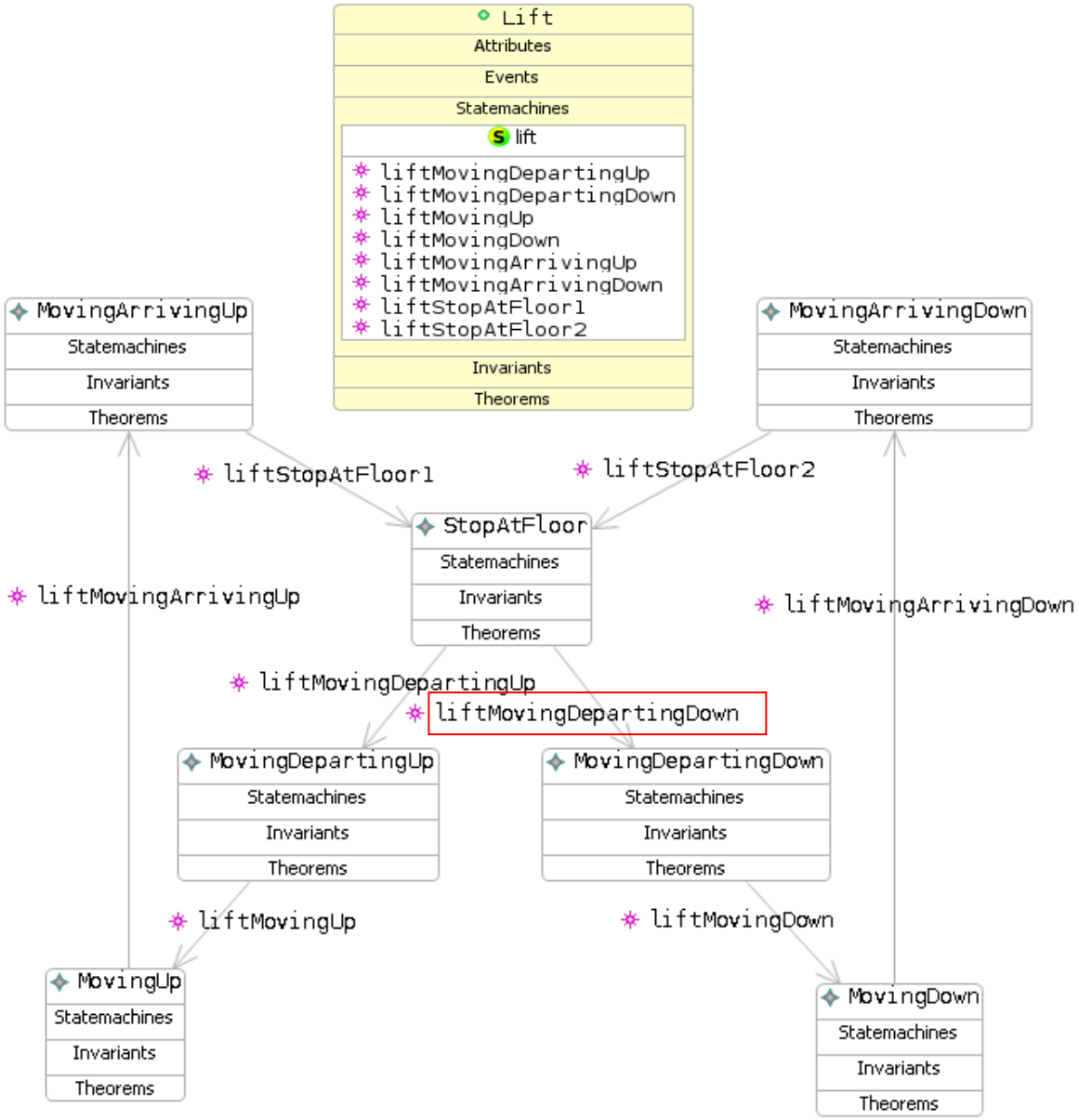
State diagram

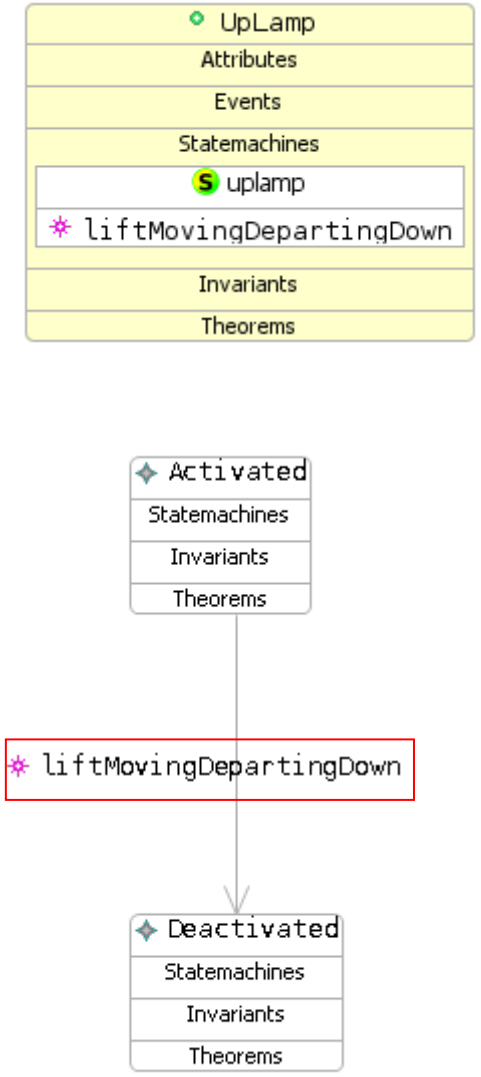


Examples of Event-B are created from UML-B

```
floorsensorOff   ≙
WHICH IS
  ordinary
ANY
  self
WHERE
  self.type   :   self ∈ FloorSensor
  source state   :   floorsensor(self) = On
  floorsensorOff.TimingCnstrntGuard   :   ((gclock − liftMovingDepartingUptime ≥ 2)  ∧ (
THEN
  target state   :   floorsensor(self) ≔ Off
  floorsensorOff.gClockAction   :   floorsensorOffTime ≔ gclock
END

floorsensorOn   ≙
WHICH IS
  ordinary
ANY
  self
WHERE
  self.type   :   self ∈ FloorSensor
  source state   :   floorsensor(self) = Off
  floorsensorOn.TimingCnstrntGuard   :   ((gclock − liftMovingArrivingUptime ≥ 2)  ∧ (gc
THEN
  target state   :   floorsensor(self) ≔ On
  floorsensorOn.gClockAction   :   floorsensorOnTime ≔ gclock
END
```

Ongoing works : Simultaneity

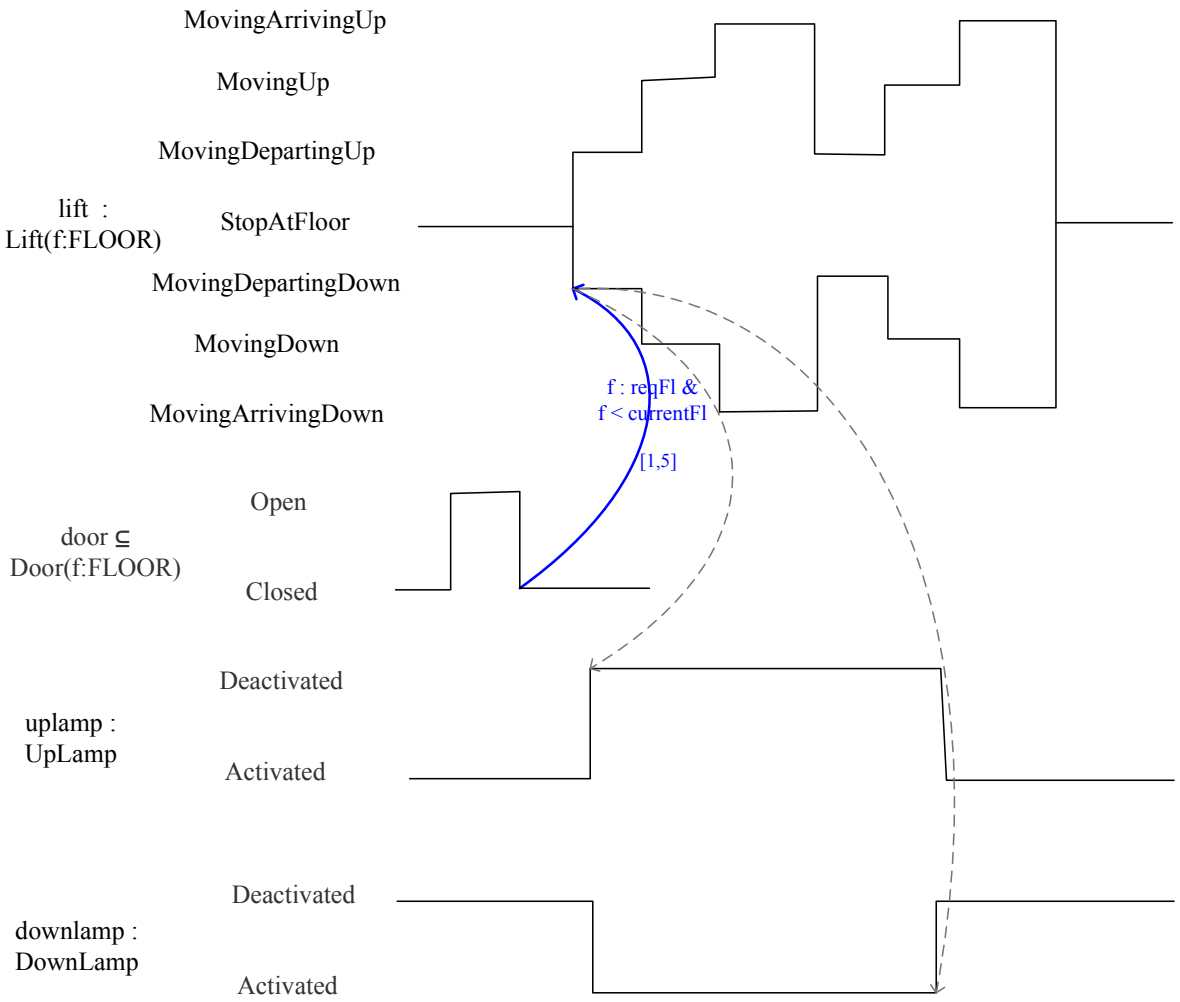




Ongoing works : Simultaneity (cont')

```
liftMovingDepartingDown   ≙
STATUS
  ordinary
ANY
  UpLampSelf      //   contextual instance of class UpLamp
  LiftSelf      //   contextual instance of class Lift
WHERE
  UpLampSelf.type   :   UpLampSelf ∈ UpLamp
  LiftSelf.type   :   LiftSelf ∈ Lift
  uplamp_isin_Activated   :   uplamp(UpLampSelf) = Activated
  lift_isin_StopAtFloor   :   lift(LiftSelf) = StopAtFloor
  liftMovingDepartingDown.TimingCnstrntGuard   :   (gclock − doorClosedtime ≥ 1)  ∧ (gclock − door(
THEN
  lift_enterState_MovingDepartingDown   :   lift(LiftSelf) ≔ MovingDepartingDown
  liftMovingDepartingDown.gClockAction   :   uplampDeactivatedTime ≔ gclock
  uplamp_enterState_Deactivated   :   uplamp(UpLampSelf) ≔ Deactivated
END
```

Ongoing works : Simultaneity (cont')



A very special THANKS to

Dr. Colin Snook