M13: Practice Lesson

Business Informatics Group
Institute of Software Technology and Interactive Systems
Vienna University of Technology
Favoritenstraße 9-11/188-3, 1040 Vienna, Austria
phone: +43 (1) 58801-18804 (secretary), fax: +43 (1) 58801-18896
office@big.tuwien.ac.at, www.big.tuwien.ac.at

Overview: Timed State Machines

Part 1: Ecore

Part 2: OCL

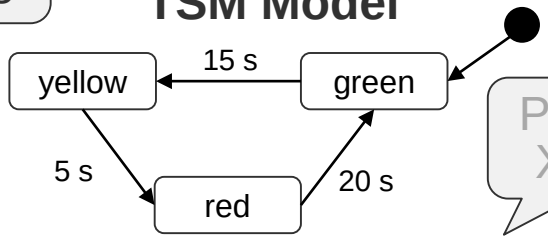
Part 3: Xtext

Part 4: ATL

Part 5: Henshin

Part 6: Xtend

TSM Model



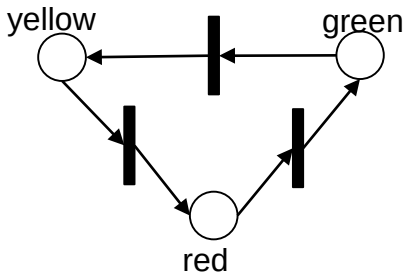

```
state-machine TrafficLightStateMachine
  starts-at green {
    state green ...
    state yellow ...
    state red ...
    transition green-to-yellow 15sec:
        green > yellow
     ...
}
```

Java

```
public class TrafficLightStateMachine {
  static TrafficLightStateMachineStates state =
        States.GREEN;
  ...
  private static void runStateMachine() {
    if (state == States.GREEN) {
      if (watch.time(TimeUnit.SECONDS) > 15) {
        System.out.println("green-to-yellow");
        state = States.YELLOW;
      }
    }
    ...
  }
  enum States { GREEN, YELLOW, RED }
}
```

connectState

Petri Net Model



Overview

- **Part 1:** Define metamodel with Ecore
- **Part 2:** Constrain metamodel with OCL
- **Part 3:** Develop textual concrete syntax with Xtext
- **Part 4:** Develop model-to-model transformation with ATL
- **Part 5:** Develop model-to-model transformation with Henshin
- **Part 6:** Develop code generator with Xtend

Business Informatics Group

Part 1: Define Metamodel with Ecore

Business Informatics Group

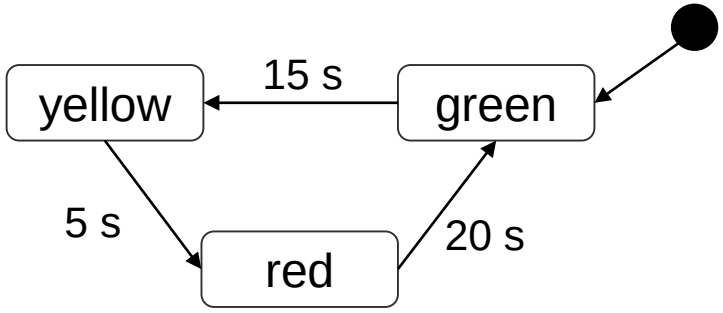
Institute of Software Technology and Interactive Systems
Vienna University of Technology

Favoritenstraße 9-11/188-3, 1040 Vienna, Austria
phone: +43 (1) 58801-18804 (secretary), fax: +43 (1) 58801-18896
office@big.tuwien.ac.at, www.big.tuwien.ac.at

Part 1: Define Metamodel with Ecore

- **Goal:** Develop a metamodel for *Timed State Machines (TSM)*

- **Example TSM Model:**

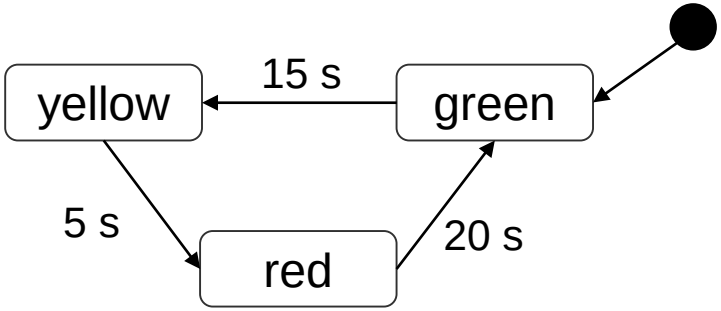


Part 1: Define Metamodel with Ecore

- **Goal:** Develop a metamodel for *Timed State Machines (TSM)*
- **Step 1:** Identify modeling concepts
- **Step 2:** Define modeling concepts in metamodel
- **Step 3:** Test the metamodel by creating example models
- **Step 4:** Generate code

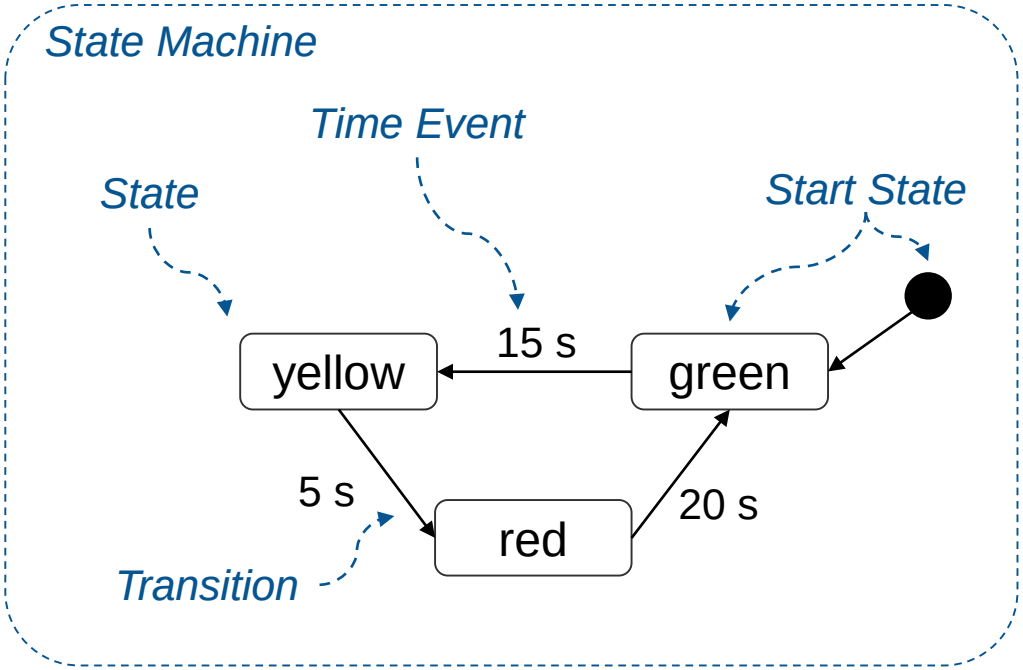
Part 1: Define Metamodel with Ecore

- **Step 1:** Identify modeling concepts of TSM



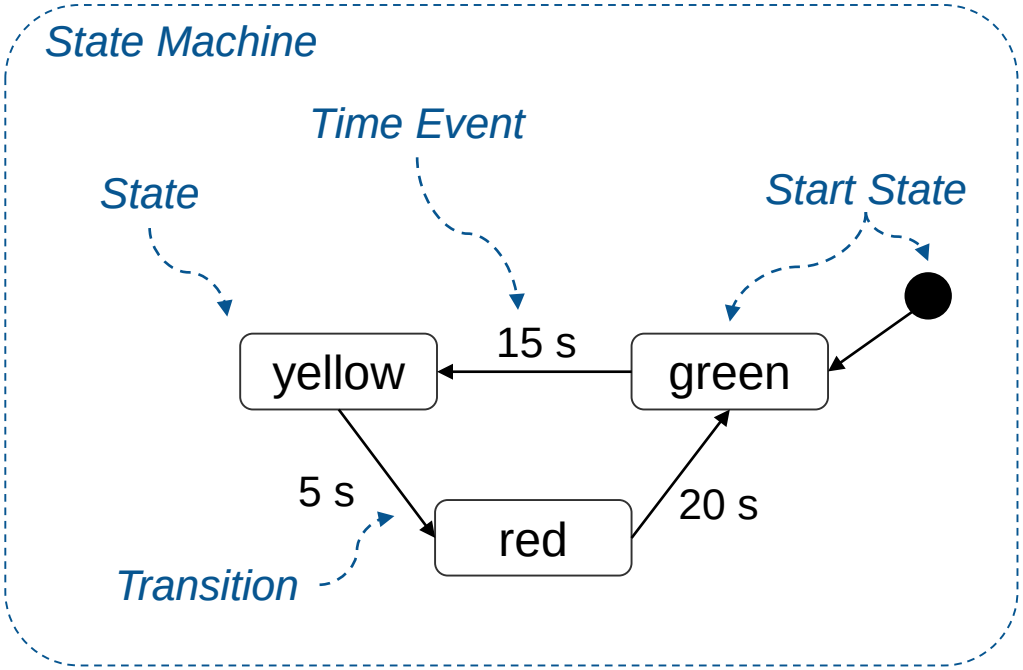
Part 1: Define Metamodel with Ecore

- **Step 1:** Identify modeling concepts of TSM



Part 1: Define Metamodel with Ecore

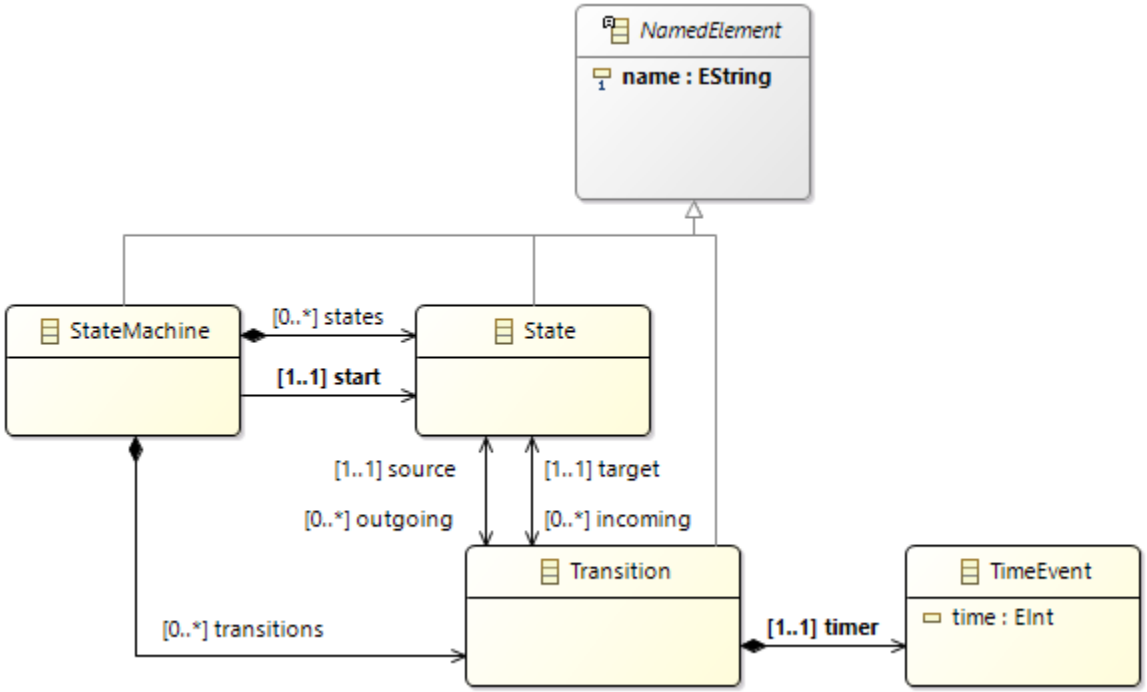
- **Step 2:** Define TSM modeling concepts in metamodel



Metamodel: `at.ac.tuwien.big.tsm/model/tsm.ecore`

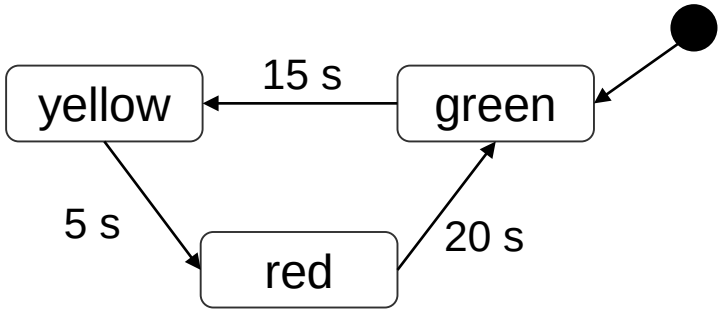
Part 1: Define Metamodel with Ecore

- **Step 2:** Define TSM modeling concepts in metamodel



Part 1: Define Metamodel with Ecore

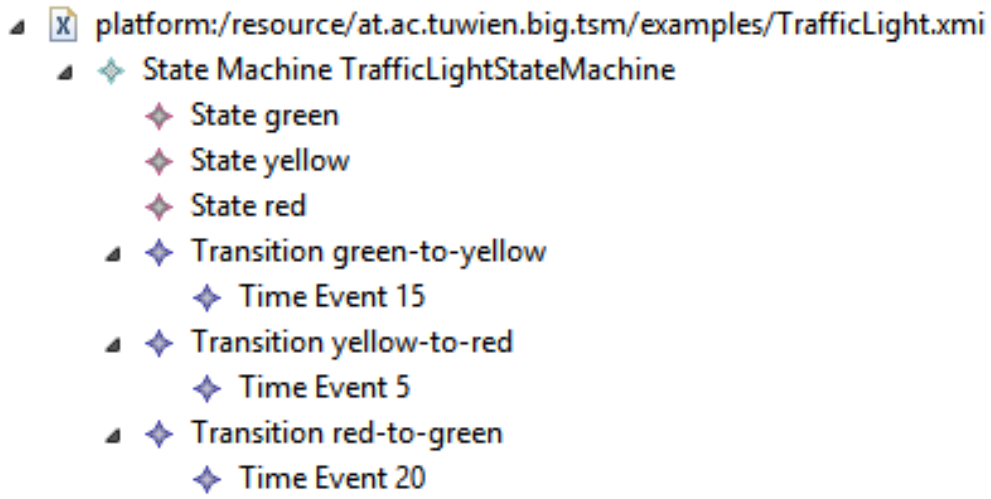
- **Step 3:** Test the metamodel by creating example models



Example model: `at.ac.tuwien.big.tsm/examples/TrafficLight.xmi`

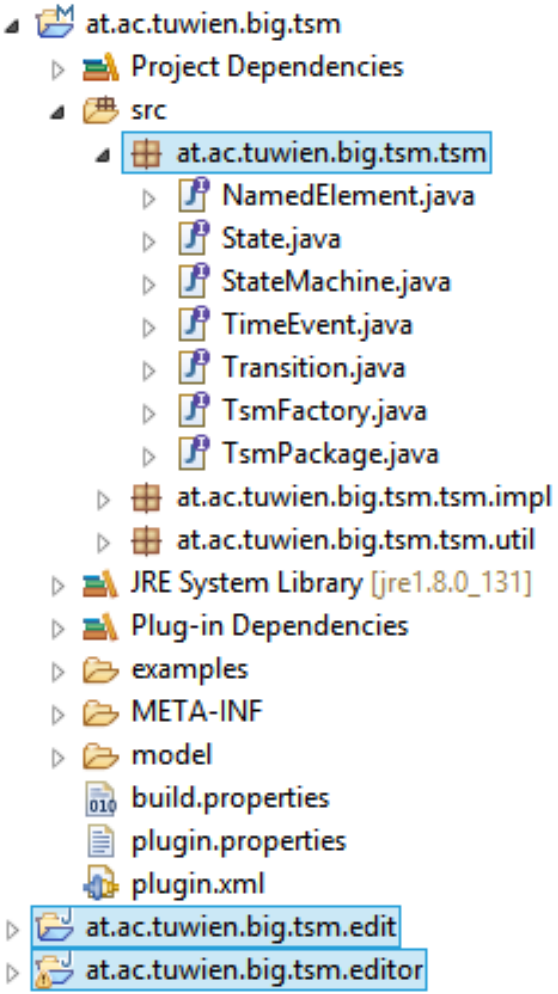
Part 1: Define Metamodel with Ecore

- **Step 3:** Test the metamodel by creating example models



Part 1: Define Metamodel with Ecore

- **Step 4:** Generate code



Generator model: `at.ac.tuwien.big.tsm/model/tsm.genmodel`

Business Informatics Group

Part 2: Constrain Metamodel with OCL

Business Informatics Group

Institute of Software Technology and Interactive Systems
Vienna University of Technology

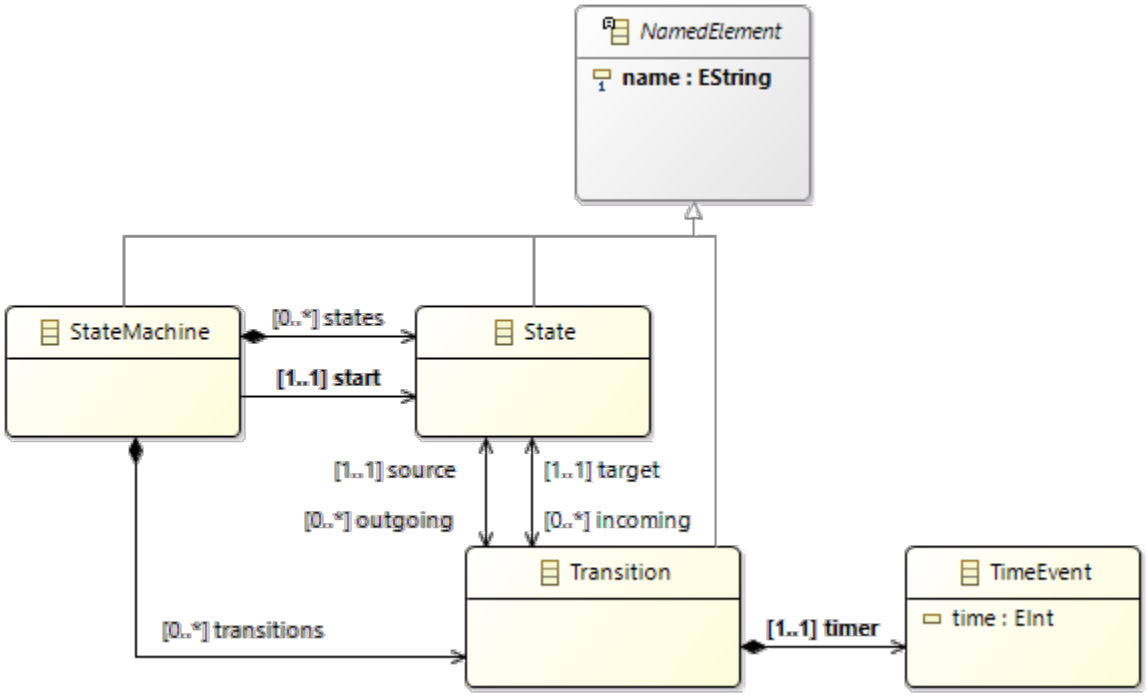
Favoritenstraße 9-11/188-3, 1040 Vienna, Austria
phone: +43 (1) 58801-18804 (secretary), fax: +43 (1) 58801-18896
office@big.tuwien.ac.at, www.big.tuwien.ac.at

Part 2: Constrain Metamodel with OCL

- **Goal:** Further constrain TSM metamodel to precisely specify how valid TSM models have to look like
- **Step 1:** Identify constraints that cannot be expressed in the metamodel
- **Step 2:** Add OCL constraints to the metamodel
- **Step 3:** Test the metamodel by creating example models
  1. Models that fulfill the OCL constraints
  2. Models that violate (single) OCL constraints

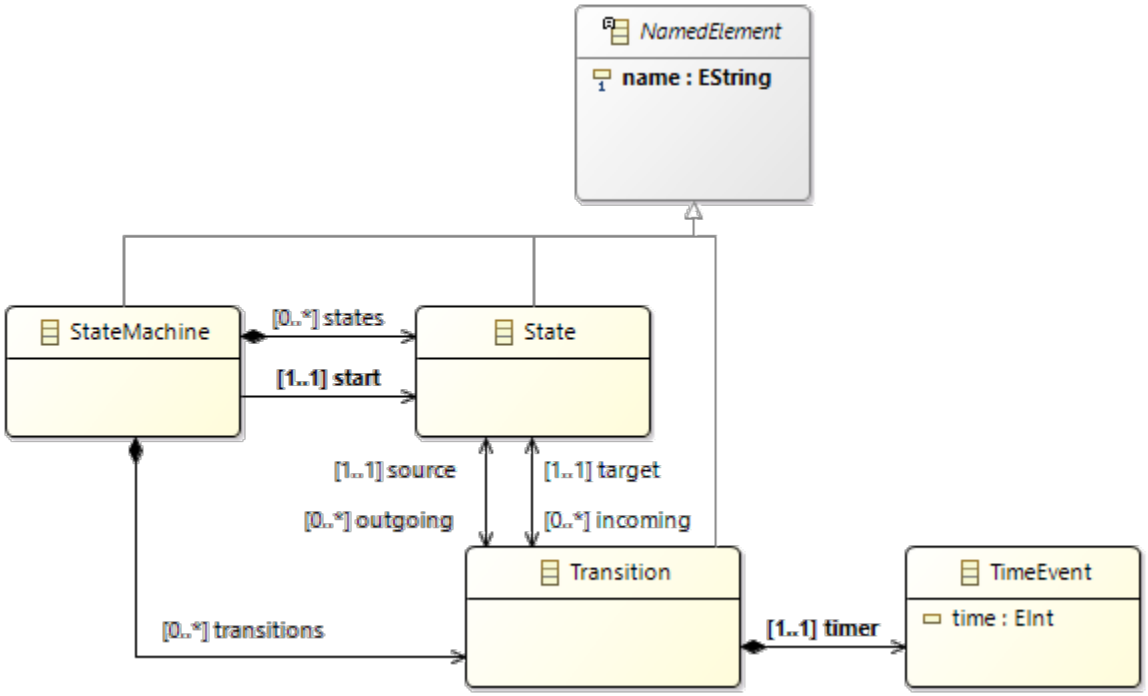
Part 2: Constrain Metamodel with OCL

- **Step 1:** Identify constraints that cannot be expressed in the metamodel



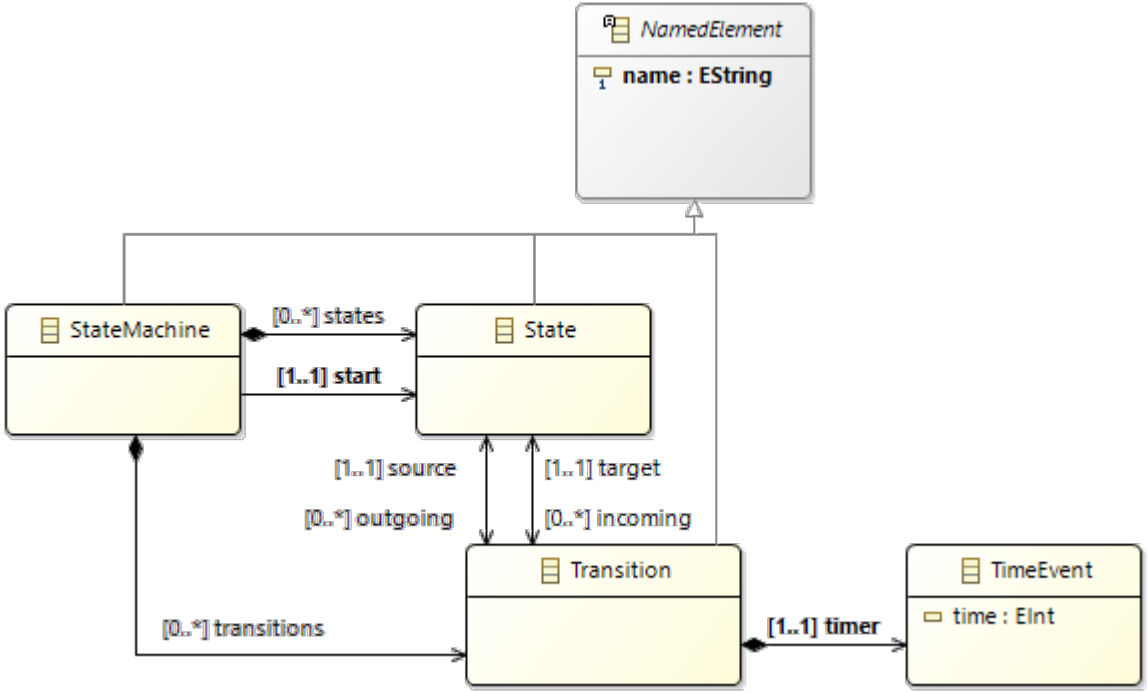
Part 2: Constrain Metamodel with OCL

- **Step 1:** Identify constraints that cannot be expressed in the metamodel
  1. Time of time event has to be greater than 0
  2. A state can have only one outgoing edge
     (could be constrained in metamodel)
  3. The names of states have to start with lower case letter



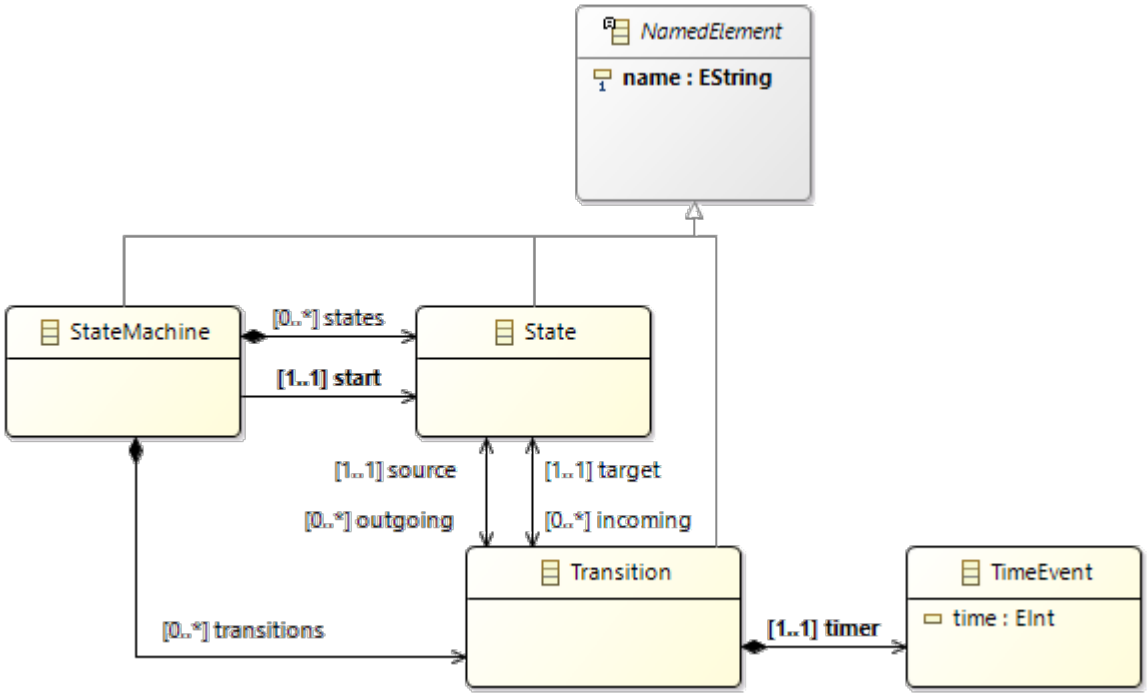
Part 2: Constrain Metamodel with OCL

- **Step 2:** Add OCL constraints to metamodel
  1. Time of time event has to be greater than 0



Part 2: Constrain Metamodel with OCL

- **Step 2:** Add OCL constraints to metamodel
  1. Time of time event has to be greater than 0

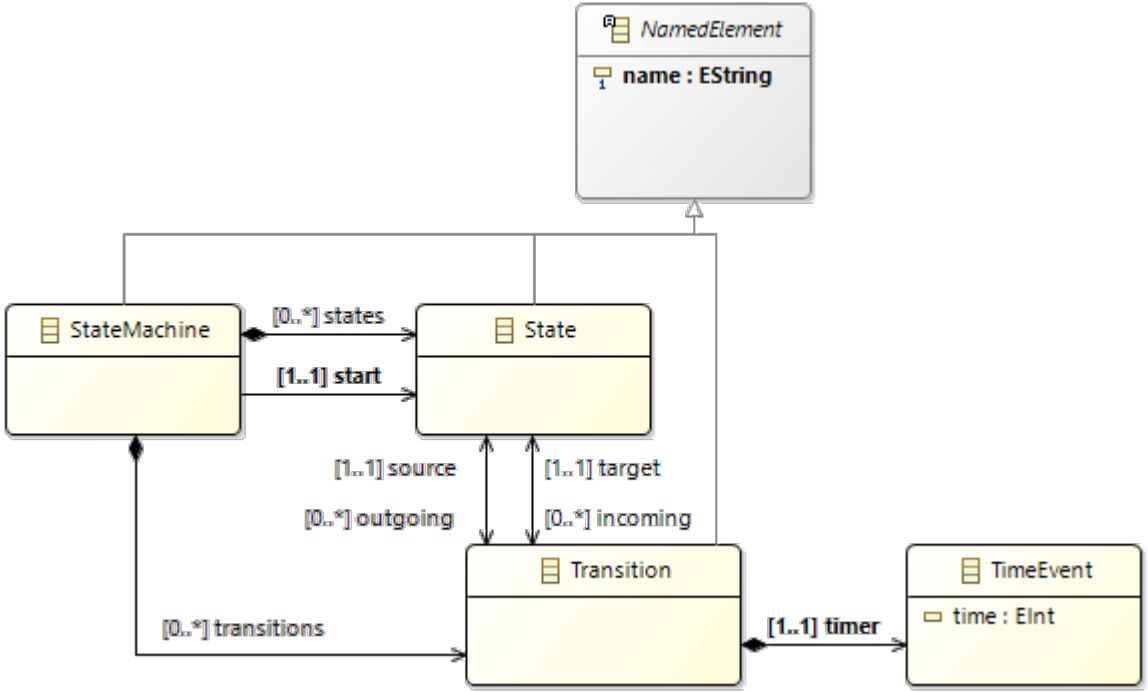



```
context TimeEvent
invariant self.time > 0
```

Part 2: Constrain Metamodel with OCL

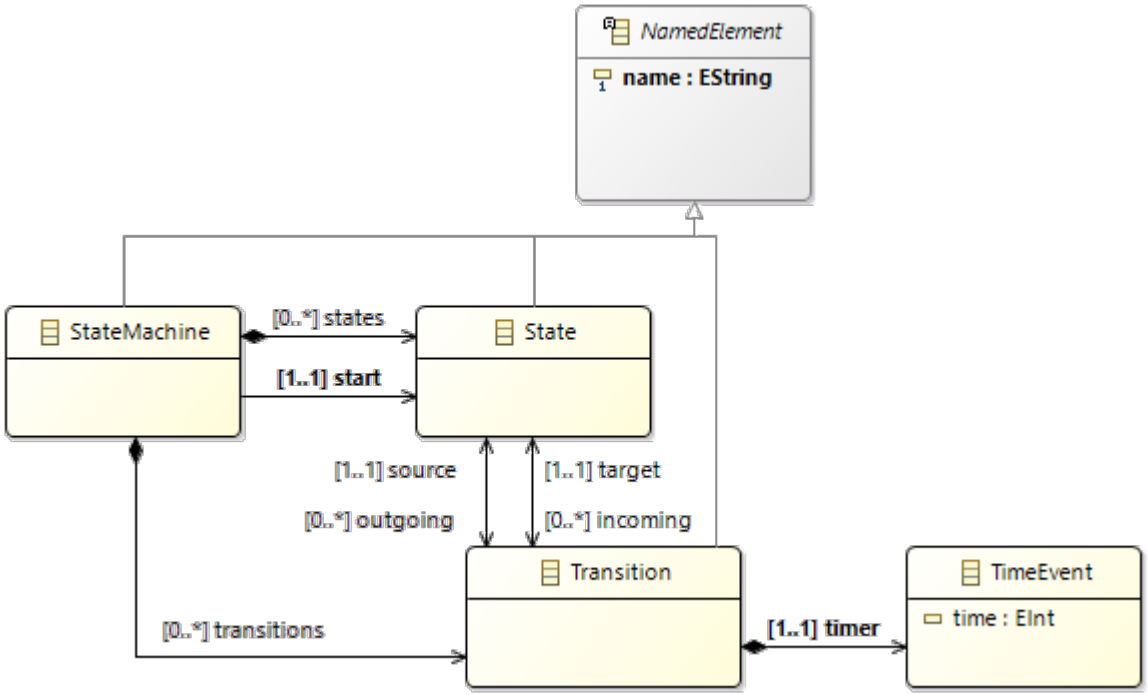
- **Step 2:** Add OCL constraints to metamodel

  2. A state can have only one outgoing edge



Part 2: Constrain Metamodel with OCL

- **Step 2:** Add OCL constraints to metamodel
  2. A state can have only one outgoing edge

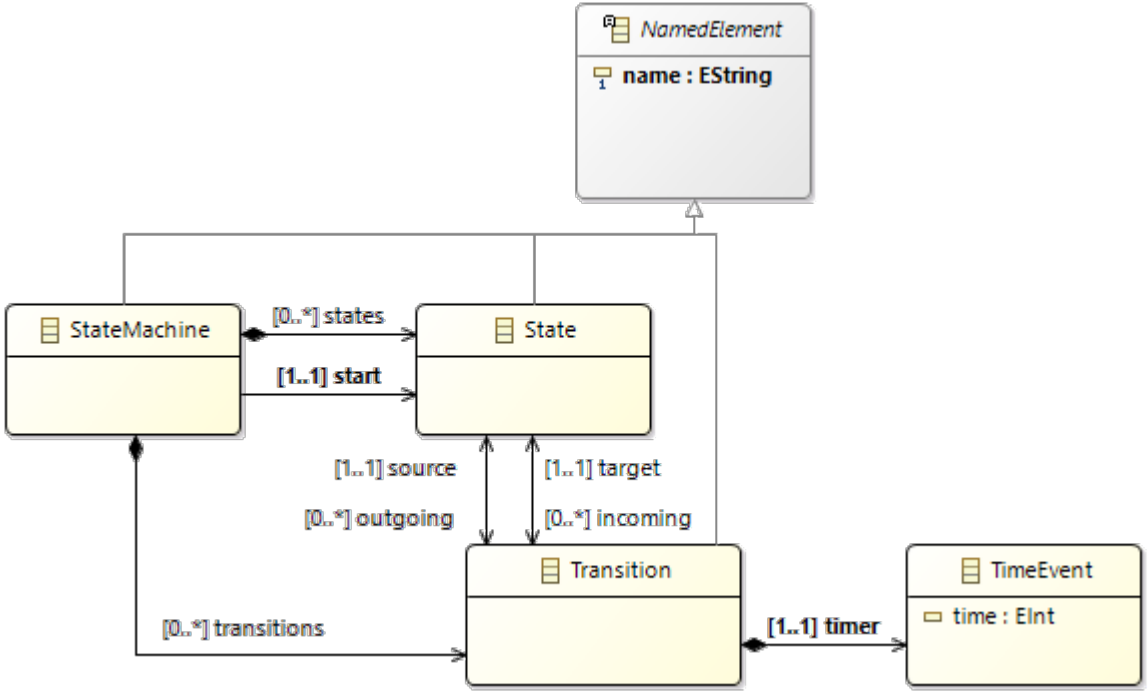



```
context State
invariant self.outgoing -> size() = 1
```

Part 2: Constrain Metamodel with OCL

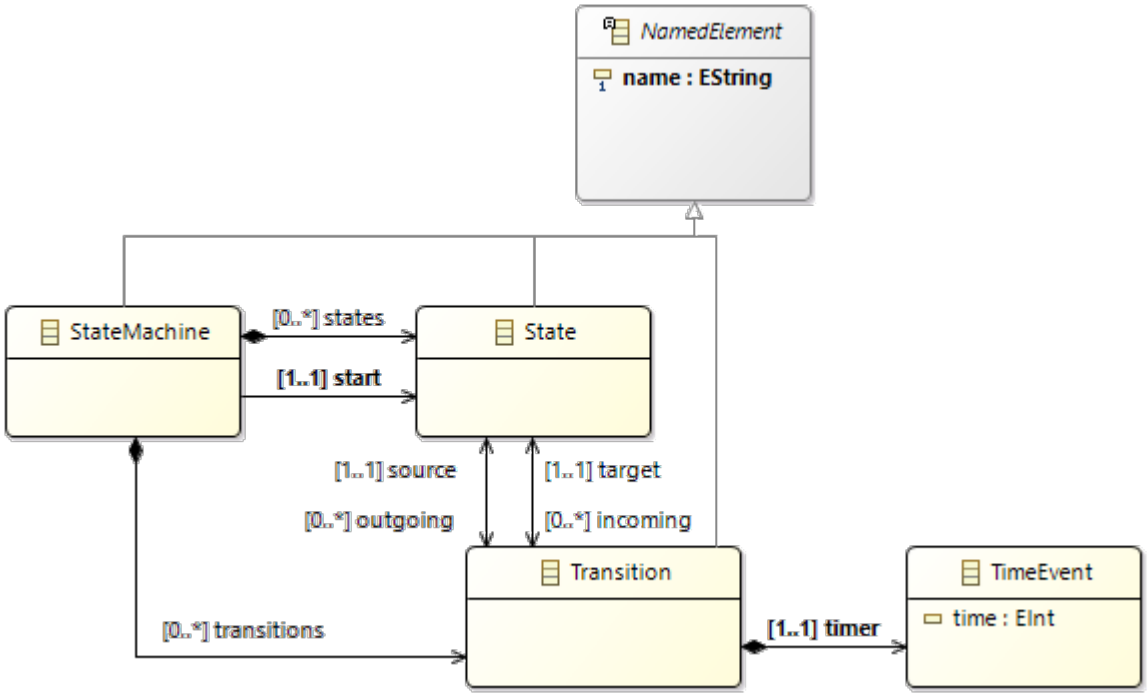
- **Step 2:** Add OCL constraints to metamodel

  3. The names of states have to start with lower case letter



Part 2: Constrain Metamodel with OCL

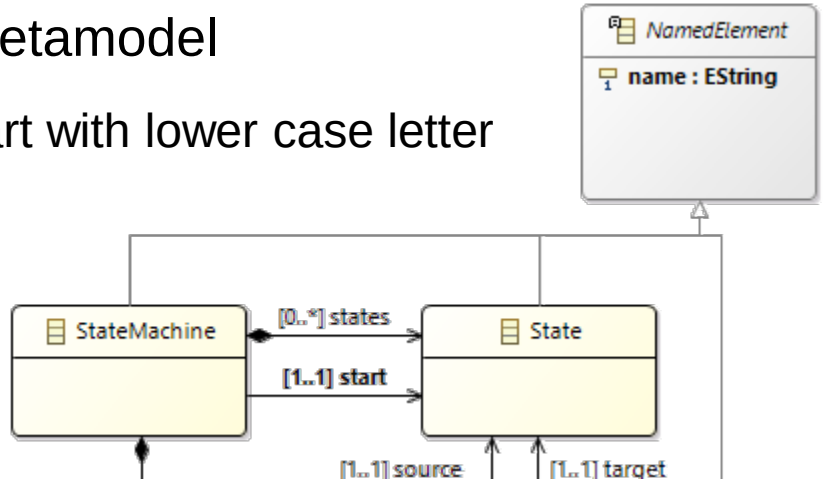
- **Step 2:** Add OCL constraints to metamodel
  3. The names of states have to start with lower case letter




```
context StateMachine
invariant self.states -> forAll(s : State | s.name.at(1) =
          s.name.at(1).toLowerCase())
```

Part 2: Constrain Metamodel with OCL

- **Step 2:** Add OCL constraints to metamodel
  3. The names of states have to start with lower case letter



Alternative Constraints:

```
context StateMachine
invariant self.states -> forAll(s : State | s.name.at(1) =
  s.name.at(1).toLowerCase())

context StateMachine
invariant self.states -> select(s : State | s.name.at(1) <>
  s.name.at(1).toLowerCase()) -> size() = 0

context State
invariant self.name.at(1) = self.name.at(1).toLowerCase()
```

Part 2: Constrain Metamodel with OCL

- **Step 2:** Add OCL constraints to metamodel (re-generate code)

```
package tsm : tsm = 'http://big.tuwien.ac.at/tsm' {
    class StateMachine extends NamedElement {
        property states : State[*|1] { ordered composes };
        property transitions : Transition[*|1] { ordered composes };
        property start : State[1];
        invariant StateNamesStartWithLowerCaseLetter : self.states -> forAll(s :
                State | s.name.at(1) = s.name.at(1).toLowerCase());
    }
    class State extends NamedElement {
        property incoming#target : Transition[*|1] { ordered };
        property outgoing#source : Transition[*|1] { ordered };
        invariant OneOutgoingEdge : self.outgoing -> size() = 1;
    }
    class TimeEvent {
        attribute time : ecore::EInt[1];
        invariant TimeGreaterZero : self.time > 0;
    }
    ...
}
```


Business Informatics Group

Part 3: Develop Textual Concrete Syntax with Xtext

Business Informatics Group

Institute of Software Technology and Interactive Systems
Vienna University of Technology

Favoritenstraße 9-11/188-3, 1040 Vienna, Austria
phone: +43 (1) 58801-18804 (secretary), fax: +43 (1) 58801-18896
office@big.tuwien.ac.at, www.big.tuwien.ac.at

Part 3: Develop Textual Concrete Syntax with Xtext

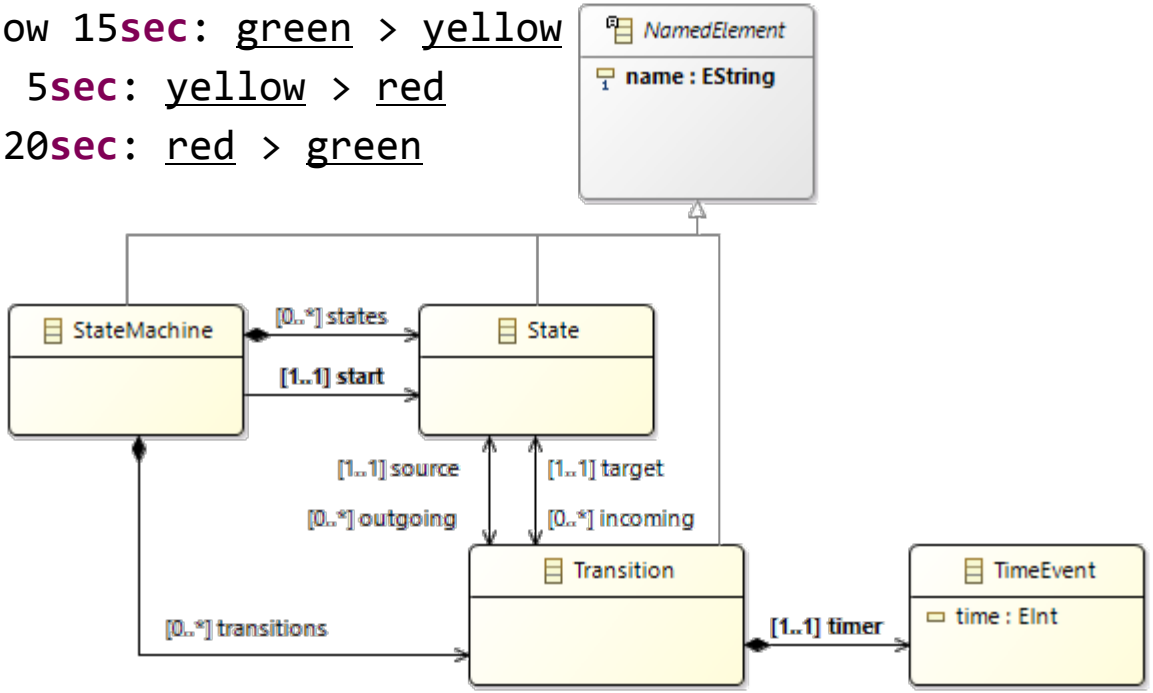
- **Goal:** Develop a textual concrete syntax for TSM so that TSM models can be defined textually
- **Step 1:** Identify keywords, scope borders, separation characters, attributes, references
- **Step 2:** Define one type rule for each metaclass including definitions of all textual syntax elements identified in Step 1
- **Step 3:** Test the textual concrete syntax by creating example models

Part 3: Develop Textual Concrete Syntax with Xtext

- **Step 1:** Identify keywords, scope borders, separation characters, attributes, references

Example Model

```
state-machine TrafficLightStateMachine starts-at green {
   state green in red-to-green out green-to-yellow
   state yellow in green-to-yellow out yellow-to-green
   state red in yellow-to-red out red-to-green
   transition green-to-yellow 15sec: green > yellow
   transition yellow-to-red 5sec: yellow > red
   transition red-to-green 20sec: red > green
}
```



Part 3: Develop Textual Concrete Syntax with Xtext

- **Step 1:** Identify keywords, scope borders, separation characters, attributes, references

Example Model

```
state-machine TrafficLightStateMachine starts-at green {
    state green in red-to-green out green-to-yellow
    state yellow in green-to-yellow out yellow-to-red
    state red in yellow-to-red out red-to-green
    transition green-to-yellow 15sec: green > yellow
    transition yellow-to-red 5sec: yellow > red
    transition red-to-green 20sec: red > green
}
```

Name Attribute

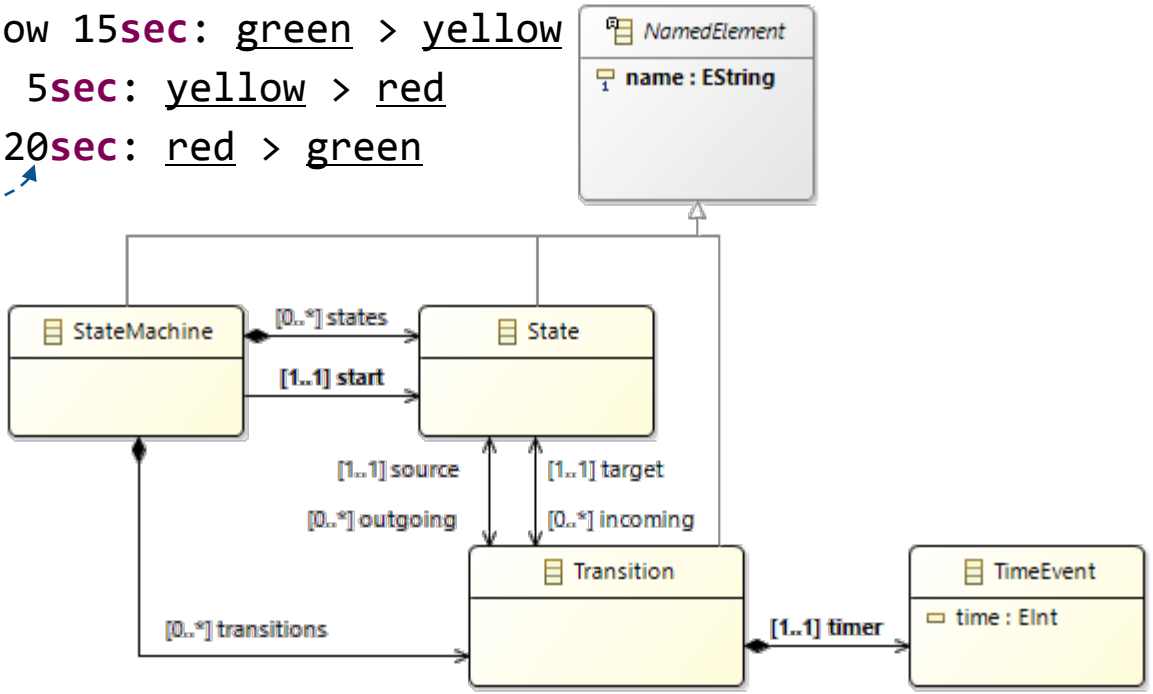
Scope Border

Keyword

Reference

Containment Reference

Time Attribute



Part 3: Develop Textual Concrete Syntax with Xtext

- **Step 2:** Define one type rule for each metaclass

Example Model

Name Attribute

Scope Border

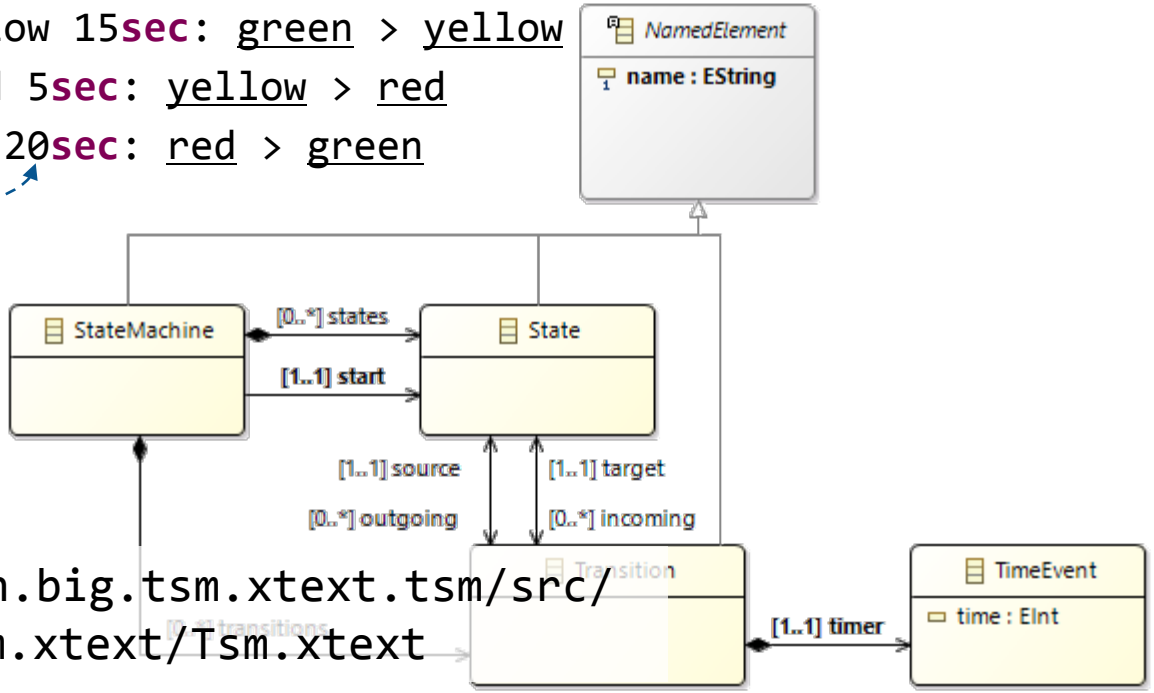
Keyword

Reference

```
state-machine TrafficLightStateMachine starts-at green {
    state green in red-to-green out green-to-yellow
    state yellow in green-to-yellow out yellow-to-red
    state red in yellow-to-red out red-to-green
    transition green-to-yellow 15sec: green > yellow
    transition yellow-to-red 5sec: yellow > red
    transition red-to-green 20sec: red > green
}
```

Containment Reference

Time Attribute



Xtext grammar: `at.ac.tuwien.big.tsm.xtext.tsm/src/at.ac.tuwien.big.tsm.xtext/Tsm.xtext`

Part 3: Develop Textual Concrete Syntax with Xtext

- **Step 2:** Define one type rule for each metaclass

```
StateMachine returns StateMachine:
    'state-machine‘ name=ID 'starts-at' start=[State|QualifiedName] '{'
       (states+=State)*
       (transitions+=Transition)*
    '}';

State returns State:
    'state'       name=ID
    'in' incoming+=[Transition|QualifiedName]
    'out' outgoing+=[Transition|QualifiedName];

Transition returns Transition:
    'transition' name=ID timer=TimeEvent 'sec' ':'
    source=[State|QualifiedName] '>' target=[State|QualifiedName];

TimeEvent returns TimeEvent:
    time=EInt;
```


Part 3: Develop Textual Concrete Syntax with Xtext

- **Step 3:** Test the textual concrete syntax by creating example models

TrafficLight.tsmtxt

```
state-machine TrafficLightStateMachine starts-at green {
    state green in red2green out green2yellow
    state yellow in green2yellow out yellow2red
    state red in yellow2red out red2green
    transition green2yellow 15sec: green > yellow
    transition yellow2red 5sec: yellow > red
    transition red2green 20sec: red > green
}
```

Example model: `at.ac.tuwien.big.tsm.xtext.examples/TrafficLight.tsmtxt`

Business Informatics Group

Part 4: Develop Model Transformation with ATL

Business Informatics Group

Institute of Software Technology and Interactive Systems
Vienna University of Technology

Favoritenstraße 9-11/188-3, 1040 Vienna, Austria
phone: +43 (1) 58801-18804 (secretary), fax: +43 (1) 58801-18896
office@big.tuwien.ac.at, www.big.tuwien.ac.at

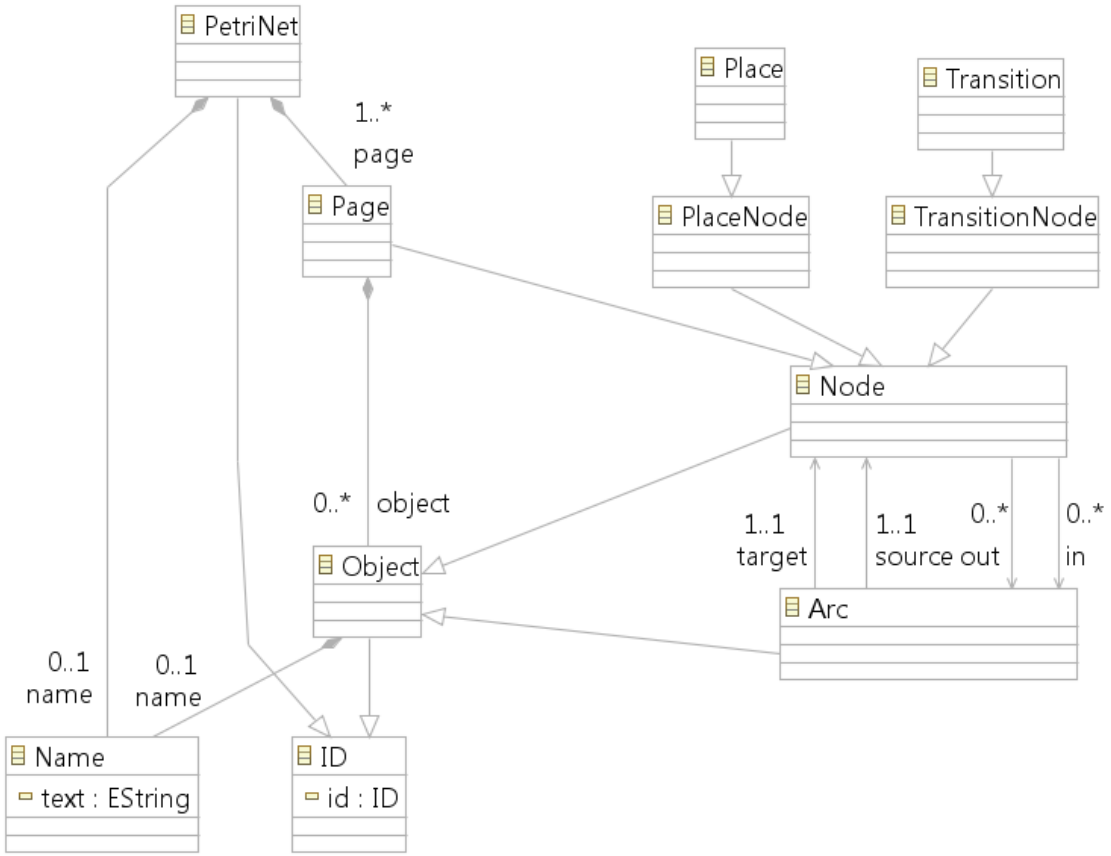
Part 4: Develop Model Transformation with ATL

- **Goal:** Develop a model transformation from TSM to Petri nets
- **Step 1:** Identify mappings from TSM to Petri nets
- **Step 2:** Implement mapping in ATL transformation (declarative rules preferred)
- **Step 3:** Test transformation by applying it on example models

Part 4: Develop Model Transformation with ATL

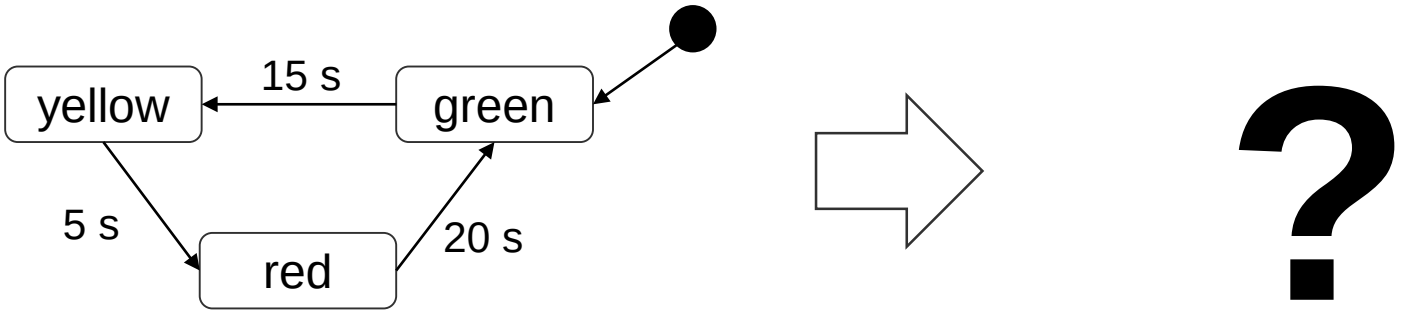
- **Step 1:** Identify mappings from TSM to Petri nets

Petri nets metamodel



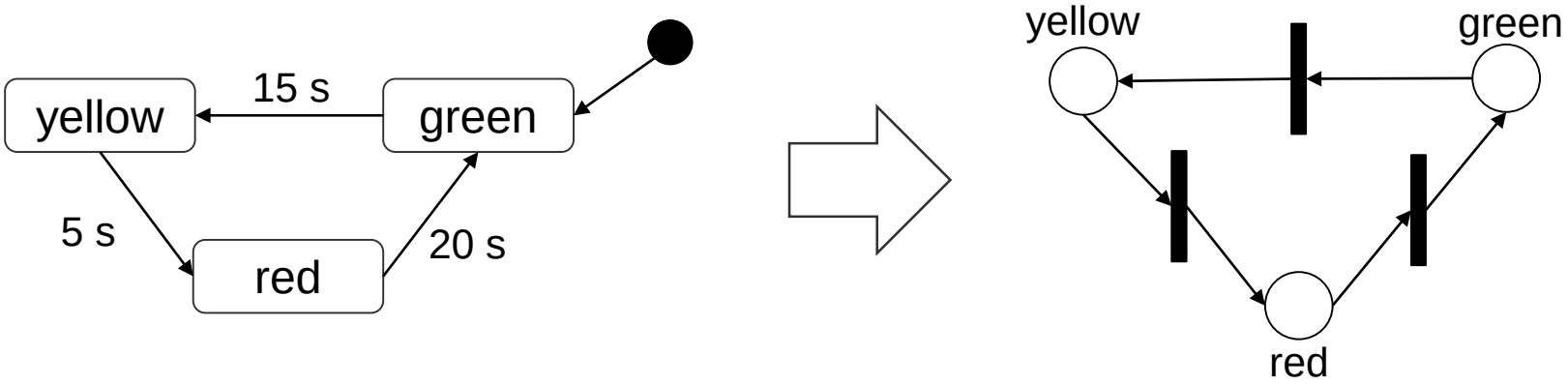
Part 4: Develop Model Transformation with ATL

- **Step 1:** Identify mappings from TSM to Petri nets



Part 4: Develop Model Transformation with ATL

- **Step 1:** Identify mappings from TSM to Petri nets



Part 4: Develop Model Transformation with ATL

- **Step 1:** Identify mappings from TSM to Petri nets

TSM	**Petri Nets**
State Machine	PetriNetDoc PetriNet + Name Page + Name
State	Place + Name
Transition	Transition + Name Arc (incoming) Arc (outgoing)
TimeEvent	-

Part 4: Develop Model Transformation with ATL

- **Step 2:** Implement mapping in ATL transformation

TSM	**Petri Nets**
State Machine	PetriNetDoc PetriNet + Name Page + Name
State	Place + Name
Transition	Transition + Name Arc (incoming) Arc (outgoing)
TimeEvent	-

ATL file: `at.ac.tuwien.big.tsm.atl/transformation/TSM2PN.atl`

Part 4: Develop Model Transformation with ATL

- **Step 2:** Implement mapping in ATL transformation

```
-- @path TSM=/at.ac.tuwien.big.tsm/model/tsm.ecore
-- @path PN=/org.pnml.tools.epnk/model/PNMLCoreModel.ecore
module TSM2PN;
create OUT : PN from IN : TSM;

rule StateMachine2PN {
    from   sm : TSM!StateMachine
    to     doc : PN!PetriNetDoc ( net <- pn ),
                pn : PN!PetriNet (
                page <- page,
                name <- pn_name,
                id <- 'PN1'
           ),
           page : PN!Page (
                name <- page_name,
                id <- 'P1',
                object <- sm.states,
                object <- sm.transitions,
                object <- sm.transitions -> collect(t | thisModule.resolveTemp(t, 'arc_in')),
                object <- sm.transitions -> collect(t | thisModule.resolveTemp(t, 'arc_out'))
           ),
           pn_name : PN!Name ( text <- sm.name ),
           page_name : PN!Name ( text <- 'MainPage'    )
}
```

(Continued on next slide)

Part 4: Develop Model Transformation with ATL

```
rule State2Place {
    from   state : TSM!State
    to     place : PN!Place (
               name <- nodename,
               id <- state.name.toLower()
           ),
           nodename : PN!Name ( text <- state.name )
}

rule Transition2Transition {
    from   tsm_transition : TSM!Transition
    to     pn_transition : PN!Transition (
               name <- pn_transition_name,
               id <- tsm_transition.name.toLower(),
               "in" <- arc_in,
               out <- arc_out
           ),
         pn_transition_name : PN!Name ( text <- tsm_transition.name ),
         arc_in : PN!Arc (
           id <- tsm_transition.name.toLower() + '-in',
           source <- tsm_transition.source,
           target <- pn_transition
         ),
         arc_out : PN!Arc (
           id <- tsm_transition.name.toLower() + '-out',
           source <- pn_transition,
           target <- tsm_transition.target
         )
}
```


Business Informatics Group

Part 5: Develop Model Transformation with Henshin

Business Informatics Group

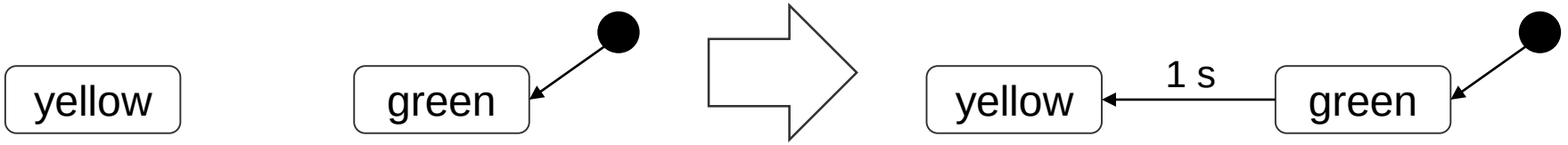
Institute of Software Technology and Interactive Systems
Vienna University of Technology

Favoritenstraße 9-11/188-3, 1040 Vienna, Austria
phone: +43 (1) 58801-18804 (secretary), fax: +43 (1) 58801-18896
office@big.tuwien.ac.at, www.big.tuwien.ac.at

Part 5: Develop Model Transformation with Henshin

- **Goal:** Develop a refactoring transformation rule for TSM models that connects unconnected states with start state

- **Example:**



Part 5: Develop Model Transformation with Henshin

- **Goal:** Develop a refactoring transformation rule for TSM models that connects unconnected states with start state

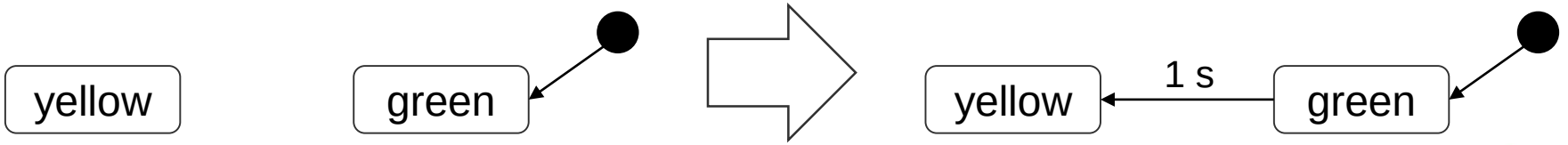
- **Step 1:** Identify
  - Conditions when rule should be applied *(application condition)*
  - Conditions when rule should not be applied *(negative application condition)*
  - Changes that should be applied on model if conditions are met (*effect*)

- **Step 2:** Implement application conditions («preserve», «delete»), negative application conditions («forbid»), changes («delete», «create»)

- **Step 3:** Test transformation rule by applying it on example models

Part 5: Develop Model Transformation with Henshin

- **Step 2:** Define application condition

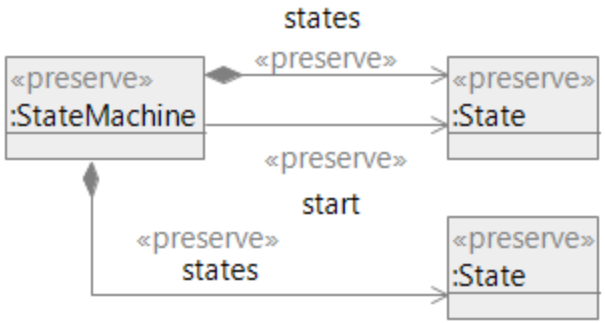


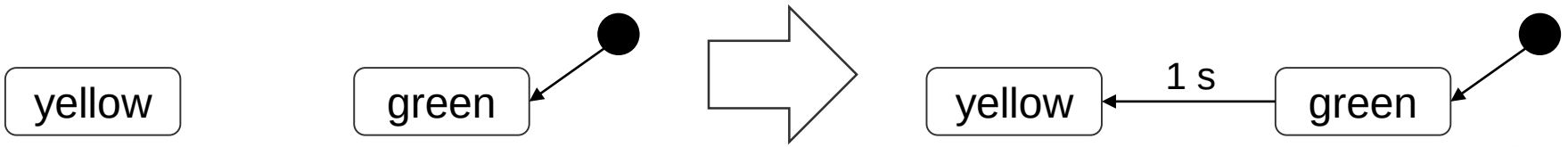
Henshin file: `at.ac.tuwien.big.tsm.henshin/henshin/tsm-connect-state.henshin_diagram`

Part 5: Develop Model Transformation with Henshin

- **Step 2:** Define application condition

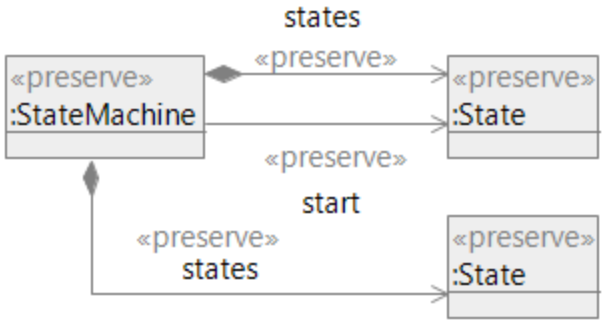
  There exists a state and a start state in a state machine:

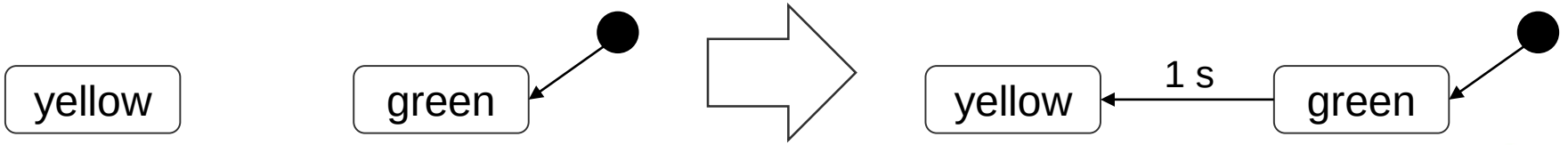




Part 5: Develop Model Transformation with Henshin

- **Step 2:** Define negative application condition

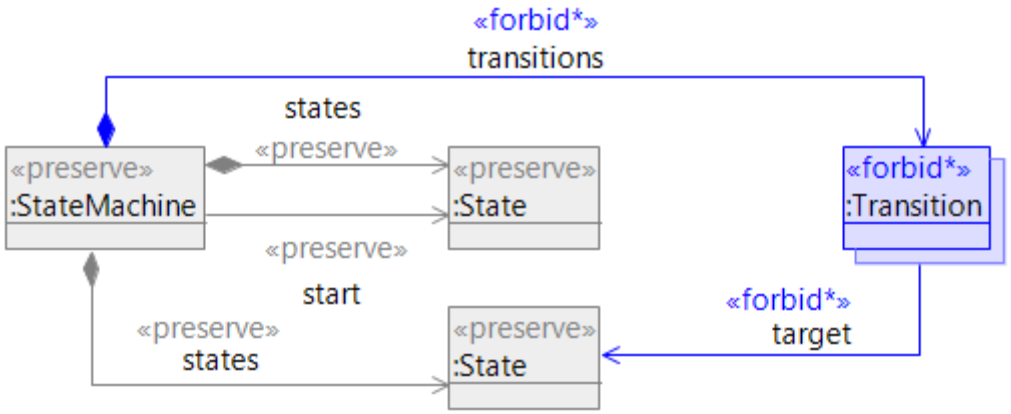


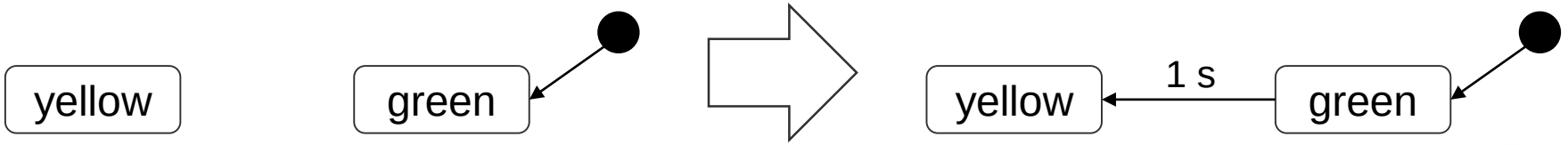


Part 5: Develop Model Transformation with Henshin

- **Step 2:** Define negative application condition

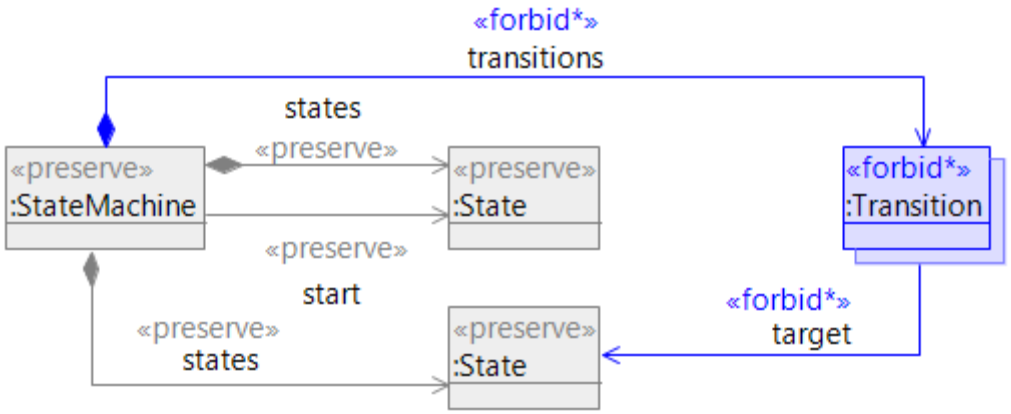
  The state has no incoming transition:

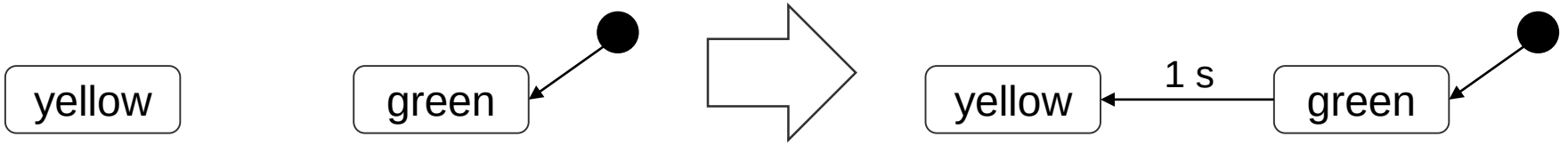




Part 5: Develop Model Transformation with Henshin

- **Step 2:** Define effect

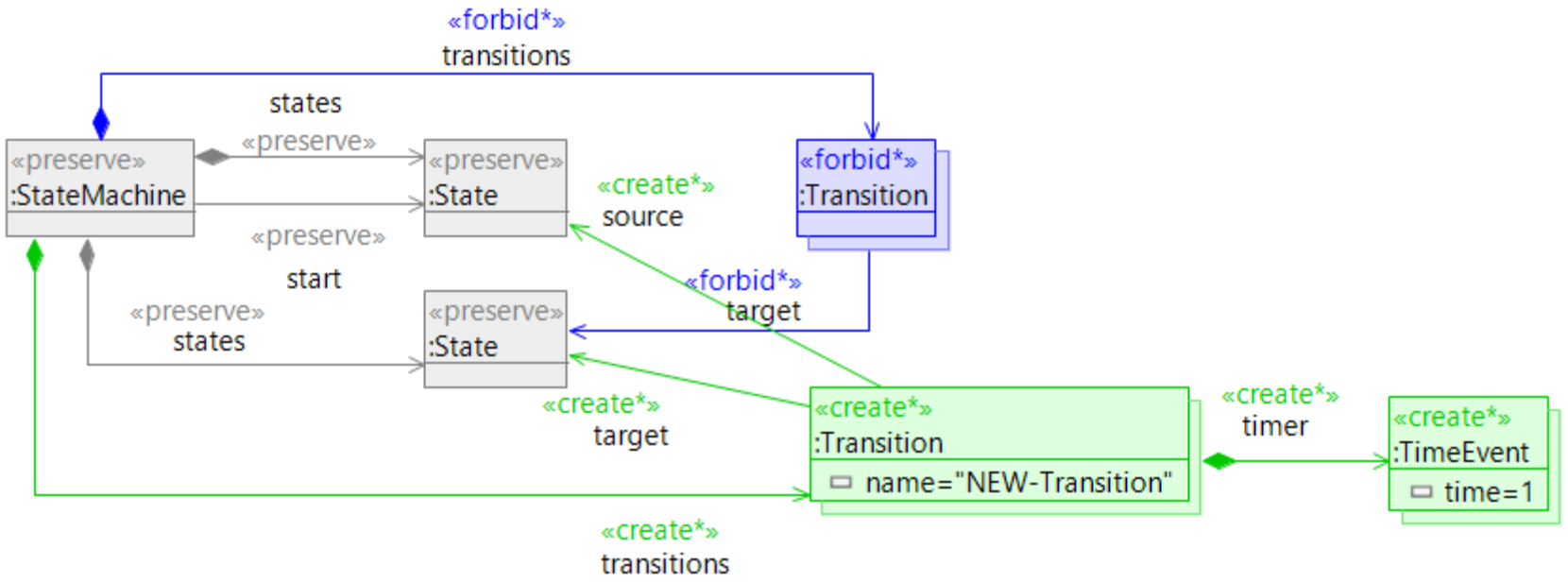


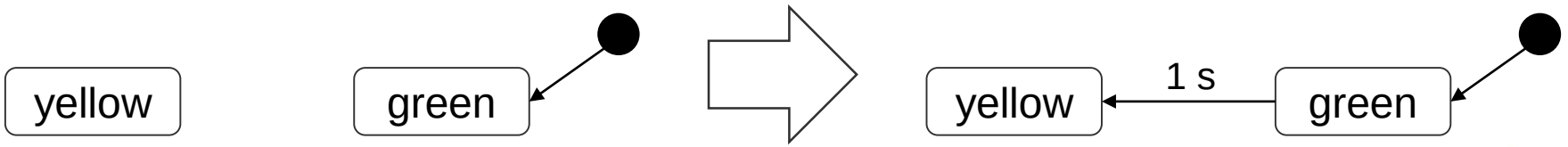


Part 5: Develop Model Transformation with Henshin

- **Step 2:** Define effect

  Create a new transition:





Business Informatics Group

Part 6: Develop Code Generator with Xtend

Business Informatics Group

Institute of Software Technology and Interactive Systems
Vienna University of Technology

Favoritenstraße 9-11/188-3, 1040 Vienna, Austria
phone: +43 (1) 58801-18804 (secretary), fax: +43 (1) 58801-18896
office@big.tuwien.ac.at, www.big.tuwien.ac.at

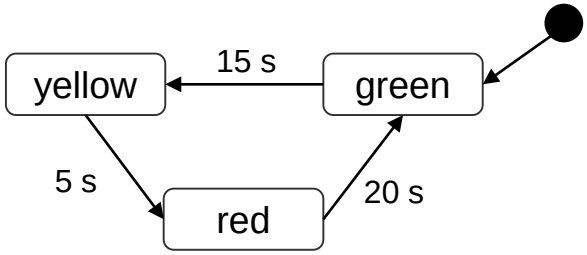
Part 6: Develop Code Generator with Xtend

- **Goal:** Generate Java classes simulating TSM models
- **Step 1:** Determine code that should be generated
  - Investigate reference code
    (may be given; otherwise, develop manually for examples)
- **Step 2:** Identify static code and dynamic code (computed from model)
- **Step 3:** Implement code generation template
  - Static code = text fragments
  - Dynamic code = meta-markers

Part 6: Develop Code Generator with Xtend

- **Step 1:** Determine code that should be generated

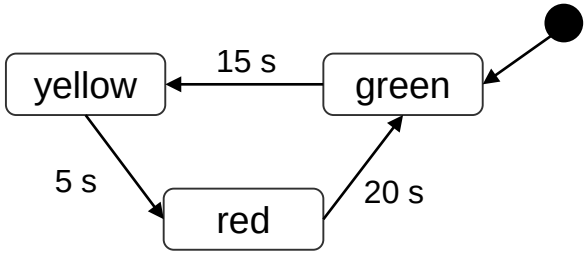
```
package at.ac.tuwien.big.tsm.trafficlightstatemachine;
public class TrafficLightStateMachine {
    static TrafficLightStateMachineStates prevState = null;
    static TrafficLightStateMachineStates state = TrafficLightStateMachineStates.GREEN;
    static TimeWatch watch = null;
    public static void main(String[] args) {
        watch = TimeWatch.start();
        while (true) { runStateMachine(); }
    }
    private static void runStateMachine() {
        if (state == TrafficLightStateMachineStates.GREEN) {
            if (prevState != state) {
                System.out.println("green");
                prevState = state;
            }
            if (watch.time(TimeUnit.SECONDS) > 15) {
                System.out.println("green-to-yellow");
                state = TrafficLightStateMachineStates.YELLOW;
                watch.reset();
            }
        }
        ...
    }
    enum TrafficLightStateMachineStates { GREEN, YELLOW, RED }
}
```



Part 6: Develop Code Generator with Xtend

- **Step 2:** Identify static code and dynamic code

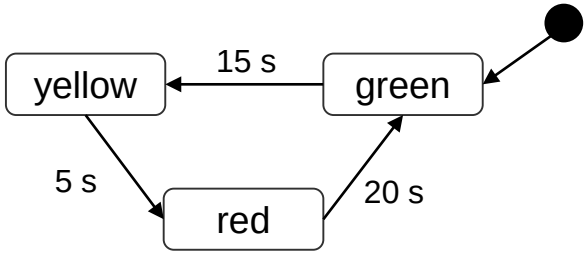
```
package at.ac.tuwien.big.tsm.trafficlightstatemachine;
public class TrafficLightStateMachine {
    static TrafficLightStateMachineStates prevState = null;
    static TrafficLightStateMachineStates state = TrafficLightStateMachineStates.GREEN;
    static TimeWatch watch = null;
    public static void main(String[] args) {
        watch = TimeWatch.start();
        while (true) { runStateMachine(); }
    }
    private static void runStateMachine() {
        if (state == TrafficLightStateMachineStates.GREEN) {
            if (prevState != state) {
                System.out.println("green");
                prevState = state;
            }
            if (watch.time(TimeUnit.SECONDS) > 15) {
                System.out.println("green-to-yellow");
                state = TrafficLightStateMachineStates.YELLOW;
                watch.reset();
            }
        }
        ...
    }
    enum TrafficLightStateMachineStates { GREEN, YELLOW, RED }
}
```



Part 6: Develop Code Generator with Xtend

- **Step 2:** Identify static code and **dynamic code**

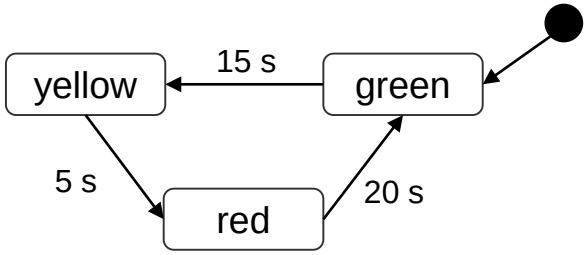
```
package at.ac.tuwien.big.tsm.trafficlightstatemachine;
public class TrafficLightStateMachine {
    static TrafficLightStateMachineStates prevState = null;
    static TrafficLightStateMachineStates state = TrafficLightStateMachineStates.GREEN;
    static TimeWatch watch = null;
    public static void main(String[] args) {
        watch = TimeWatch.start();
        while (true) { runStateMachine(); }
    }
    private static void runStateMachine() {
        if (state == TrafficLightStateMachineStates.GREEN) {
            if (prevState != state) {
                System.out.println("green");
                prevState = state;
            }
            if (watch.time(TimeUnit.SECONDS) > 15) {
                System.out.println("green-to-yellow");
                state = TrafficLightStateMachineStates.YELLOW;
                watch.reset();
            }
        }
        ...
    }
    enum TrafficLightStateMachineStates { GREEN, YELLOW, RED }
}
```



Part 6: Develop Code Generator with Xtend

- **Step 3:** Implement code generation template

```
package at.ac.tuwien.big.tsm.trafficlightstatemachine;
public class TrafficLightStateMachine {
    static TrafficLightStateMachineStates prevState = null;
    static TrafficLightStateMachineStates state = TrafficLightStateMachineStates.GREEN;
    static TimeWatch watch = null;
    public static void main(String[] args) {
        watch = TimeWatch.start();
        while (true) { runStateMachine(); }
    }
    private static void runStateMachine() {
        if (state == TrafficLightStateMachineStates.GREEN) {
            if (prevState != state) {
                System.out.println("green");
                prevState = state;
            }
            if (watch.time(TimeUnit.SECONDS) > 15) {
                System.out.println("green-to-yellow");
                state = TrafficLightStateMachineStates.YELLOW;
                watch.reset();
            }
        }
        ...
    }
    enum TrafficLightStateMachineStates { GREEN, YELLOW, RED }
}
```



Xtend file:
`at.ac.tuwien.big.tsm.codegen/src/TSMJavaGenerator.xtend`

Part 6: Develop Code Generator with Xtend

- **Step 3:** Implement code generation template

```
'''
package «stateMachine.packageName»;
import java.util.concurrent.TimeUnit;
import at.ac.tuwien.big.stl.codegen.lib.TimeWatch;
public class «stateMachine.className» {
    static «stateMachine.stateEnumerationName» prevState = null;
    static «stateMachine.stateEnumerationName» state =
              «stateMachine.stateEnumerationName».«stateMachine.start.stateEnumValue»;
    static TimeWatch watch = null;
    public static void main(String[] args) {
        watch = TimeWatch.start();
        while (true) {
            runStateMachine();
        }
    }
```

(Continued on next slide)

Part 6: Develop Code Generator with Xtend

```
    private static void runStateMachine() {
        «FOR State state : stateMachine.states»
        if (state == «stateMachine.stateEnumerationName».«state.stateEnumValue») {
            if (prevState != state) {
             System.out.println("«state.name»");
             prevState = state;
            }
            if (watch.time(TimeUnit.SECONDS) > «state.outgoing.get(0).timer.time») {
             System.out.println("«state.outgoing.get(0).name»");
             state = «stateMachine.stateEnumerationName».«state.outgoing.get(0).target.stateEnumValue»;
             watch.reset();
            }
        }
        «ENDFOR»
    }
    enum «stateMachine.name.toAlphaNumerical»States {
        «FOR State state : stateMachine.states SEPARATOR ', '»
        «state.stateEnumValue»
        «ENDFOR»
    }
 }
'''
```


Congratulations!

You have just developed a new modeling language!