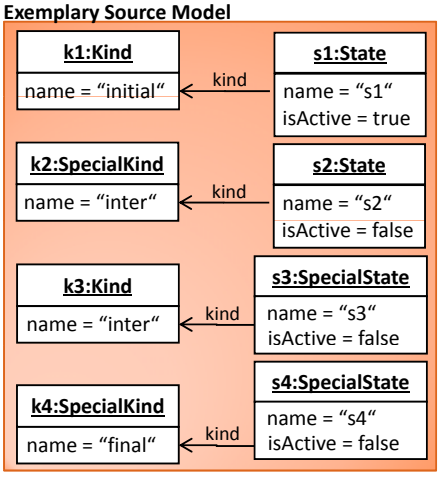
EXAMPLE 7 – ATL (2/2)

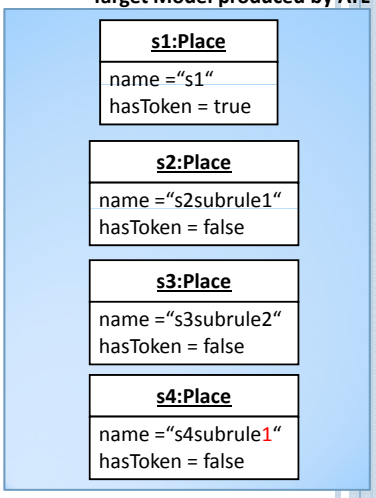
Exemplary Source Model



Please note: no rule definition for (SpecialStates, SpecialKinds) available -> therefore it is ambiguous which rule should match for such a combination!

```
rule StateKind2Place {
 from mElem : Statemachine!State,
        kind : Statemachine!Kind
        (mElem.kind = kind)
 to elem : Petrinet!Place (
      name <-  mElem.name,
      hasToken <- mElem.isActive
 )
}

rule StateSpecialKind2Place extends
StateKind2Place {
 from mElem : Statemachine!State,
        kind : Statemachine!SpecialKind
        (mElem.kind = kind)
 to elem : Petrinet!Place (
     name <- mElem.name + 'subrule1'
)
}

rule SpecialStateKind2Place extends
StateKind2Place {
 from mElem : Statemachine!SpecialState,
        kind : Statemachine!Kind
        (mElem.kind = kind)
 to elem : Petrinet!Place (
    name <- mElem.name + 'subrule2'
)
}
```

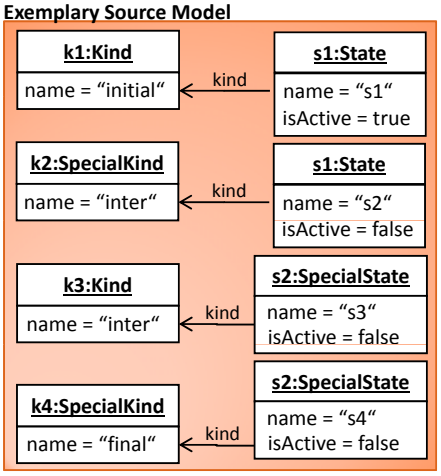
Target Model produced by ATL



- **Results/Findings** (according to goals)
  By duplicating the conditions, the behavior changes!
  Ambiguous rule definitions are not recognized; this is inferred from the fact, that for pair `(s4,k4)` the first matching method is applied (depends on the order in the file)

EXAMPLE 7 – ETL

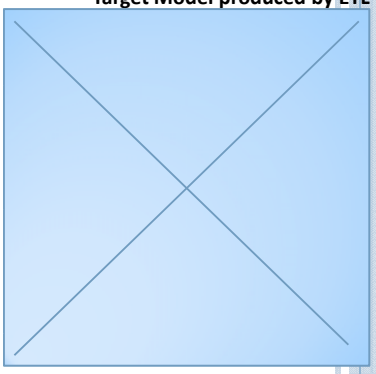
Exemplary Source Model



Please note: no rule definition for (SpecialStates, SpecialKinds) available -> therefore it is ambiguous which rule should match for such a combination!

```
rule StateKind2Place
 transform mElem : Statemachine!State,
                kind : Statemachine!Kind
 to elem : Petrinet!Place {
   guard : mElem.kind = kind
   elem.name := mElem.name;
  elem.hasToken := mElem.isActive;
 }

rule StateSpecialKind2Place
 transform mElem : Statemachine!State,
                kind : Statemachine!SpecialKind
 to elem : Petrinet!Place
 extends StateKind2Place {
  elem.name := mElem.name + 'subrule1';
}

rule SpecialStateKind2Place
 transform mElem : Statemachine!SpecialState,
                kind : Statemachine!Kind
 to elem : Petrinet!Place
 extends StateKind2Place {
  elem.name := mElem.name + 'subrule2';
}
```

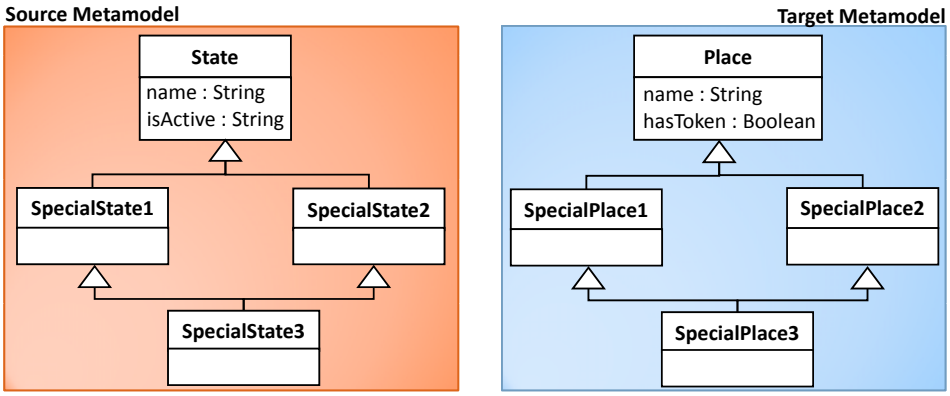
Target Model produced by ETL



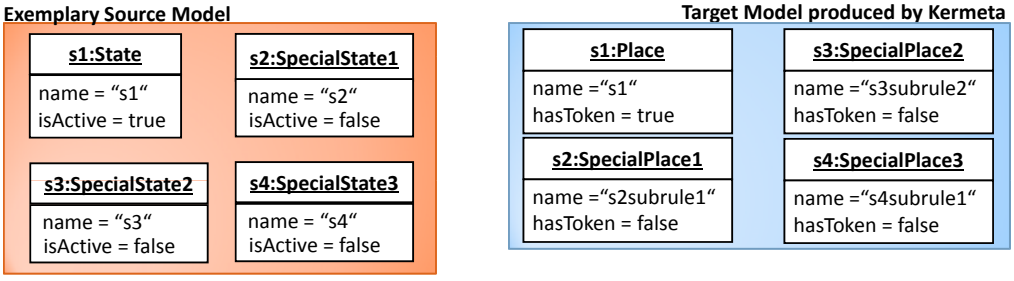
- **Results/Findings** (according to goals)
  (1) ETL does not allow for multiple input elements (syntax error)

EXAMPLE 8

- **Description**
  - In this example; `States` should be transformed into corResponding `Places`, whereby different assignments should take place according to the matched subtype, in order to be able to differentiate the rules
- **Goal(s) of Evaluation**
  (1) Check how diamond inheritance issues are handled



EXAMPLE 8 – KERMETA (1/4)



- **Results/Findings** (according to goals)
  (1) Diamonds per se are not detected, but the problem of inheriting several equally named methods is detected at compile time and the user is asked to explicitly decide for a certain method with the `from` keyword; consequently the result depends on the user's decision
  Error Message:

```
Several super operations found for method assignments in class
definition Statemachine2Petrinet::SpecialState32SpecialPlace3
Definitions coming from
Statemachine2Petrinet::SpecialState12SpecialPlace1,
Statemachine2Petrinet::SpecialState22SpecialPlace2.
```

EXAMPLE 8 – KERMETA (2/4)

```
//transformation code for Statemachine2PetriNet
class Statemachine2PetriNet{

  operation conditionFulFilled(s : Statemachine) : kermeta::standard::Boolean is do
    result := true
  end

  operation assignments(s : Statemachine, p : PetriNet) is do
  end

  operation referenceAssignments(s : Statemachine, p : PetriNet, trace: Trace<Object, Object>) is do
    s.elements.each{ e |
      if trace.getTargetElem(e) != void then
        p.elements.add(trace.getTargetElem(e).asType(Place))
      end
    }
  end
}

//transformation code for State2Place
class State2Place{

  operation conditionFulFilled(s : State) : kermeta::standard::Boolean is do
    result := true
  end

  operation assignments(s : State, p : Place) is do
    p.name := s.name
    p.hasToken := s.isActive
  end

  operation referenceAssignments(s : State, p : Place, trace: Trace<Object, Object>) is do
  end
}
```

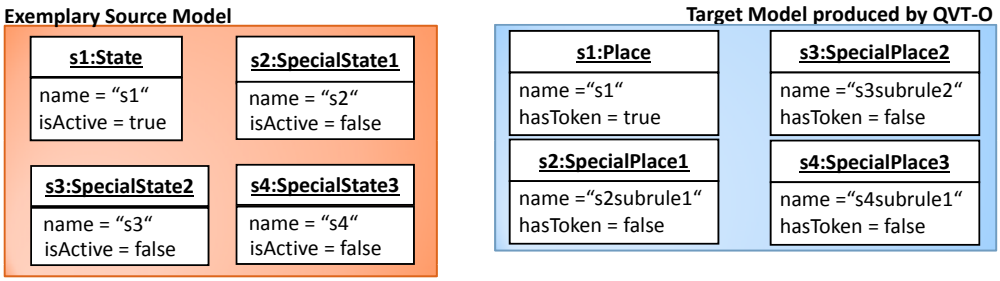
EXAMPLE 8 – KERMETA (3/4)

```
//transformation code for SpecialState12SpecialPlace1
class SpecialState12SpecialPlace1 inherits State2Place{

  method conditionFulFilled(s : State) : kermeta::standard::Boolean is do
    result := super(s)
  end

  method assignments(s : State, p : Place) is do
    result := super(s,p)
    p.name := s.name + "subrule1"
  end

  method referenceAssignments(s : State, p : Place, trace: Trace<Object, Object>) is do
    super(s,p,trace)
  end
}

//transformation code for SpecialState22SpecialPlace2
class SpecialState22SpecialPlace2 inherits State2Place{

  method conditionFulFilled(s : State) : kermeta::standard::Boolean is do
    result := super(s)
  end

  method assignments(s : State, p : Place) is do
    result := super(s,p)
    p.name := s.name + "subrule2"
  end

  method referenceAssignments(s : State, p : Place, trace: Trace<Object, Object>) is do
    super(s,p,trace)
  end
}
```

EXAMPLE 8 – KERMETA (4/4)

```
//transformation code for SpecialState22SpecialPlace2
class SpecialState32SpecialPlace3 inherits SpecialState12SpecialPlace1, SpecialState22SpecialPlace2{

  method conditionFulFilled(s : State) : kermeta::standard::Boolean from SpecialState12SpecialPlace1 is do
    result := super(s)
  end

  method assignments(s : State, p : Place) from SpecialState12SpecialPlace1 is do
    result := super(s,p)
  end

  method referenceAssignments(s : State, p : Place, trace: Trace<Object, Object>)
  from SpecialState12SpecialPlace1 is do
    super(s,p,trace)
  end
}
```


EXAMPLE 8 – QVT-O (1/2)

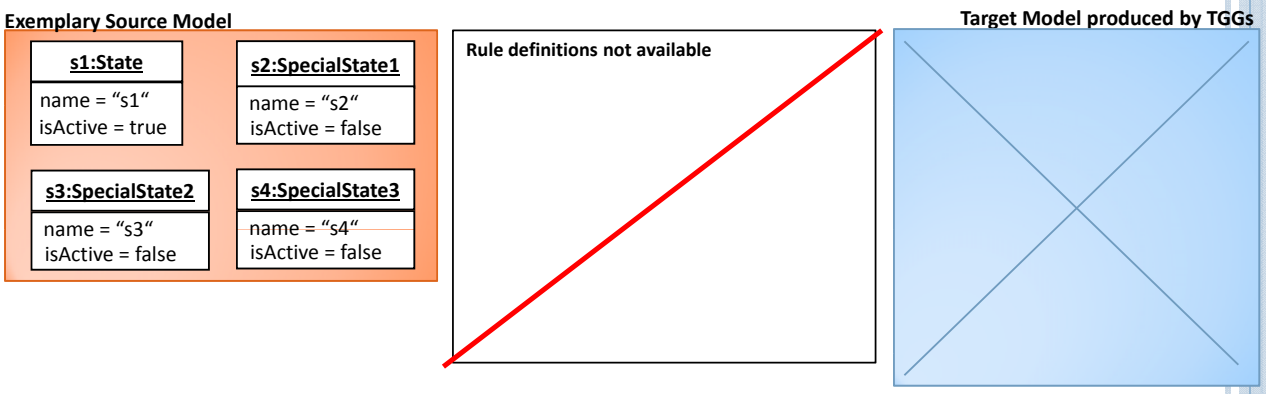


- **Results/Findings** (according to goals)
  (1) Diamonds are not detected, neither at compile-time nor at run-time; instead, those assignments win, which exist in the latest specified inherited rule

EXAMPLE 8 – QVT-O (2/2)

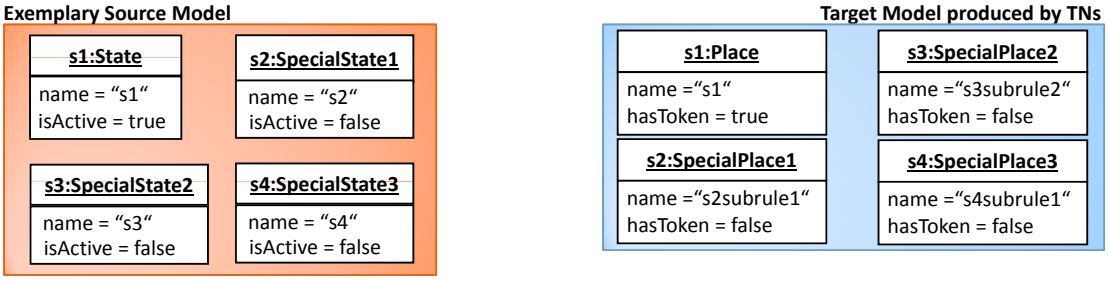
```
transformation testTrafo(in inModel : sm, out outModel : pn);

  main() {
    inModel.rootObjects()[Statemachine] -> map SM2Petri();
  }

  mapping Statemachine::SM2Petri() : PetriNet {
    //please note that specific rules must be called first!
    elements := self.elements[SpecialState3] -> map SpecialState32SpecialPlace3();
    elements += self.elements[SpecialState2] -> map SpecialState22SpecialPlace2();
    elements += self.elements[SpecialState1] -> map SpecialState12SpecialPlace1();
    elements += self.elements[State] -> map State2Place();
  }

  mapping State::State2Place() : Place {
    dump('Rule0 with ' + self.name);
    name := self.name;
    hasToken := self.isActive;
  }

  mapping SpecialState1::SpecialState12SpecialPlace1() : SpecialPlace1 inherits State::State2Place {
    dump('Rule1 with ' + self.name);
    name := self.name + 'subrule1';
}

  mapping SpecialState2::SpecialState22SpecialPlace2() : SpecialPlace2 inherits State::State2Place {
    dump('Rule2 with ' + self.name);
    name := self.name + 'subrule2';
  }

  mapping SpecialState3::SpecialState32SpecialPlace3() : SpecialPlace3 inherits
  SpecialState2::SpecialState22SpecialPlace2, SpecialState1::SpecialState12SpecialPlace1  {
    dump('Rule3 with ' + self.name);
  }
```

EXAMPLE 8 – TGGS



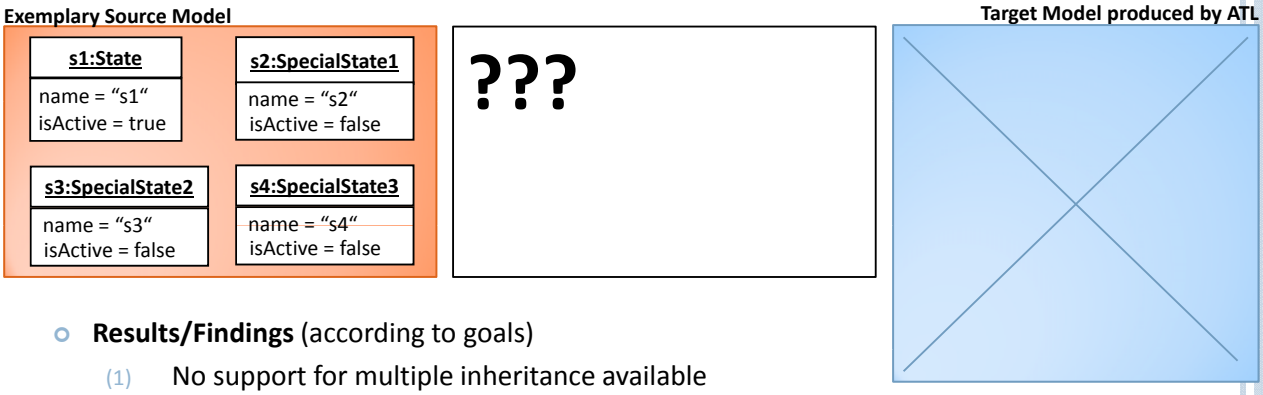
- **Results/Findings** (according to goals)
  - (1) Static analysis prohibits to inherit from two or rules that have different assignment statements for the same parameter. It remains unclear for the subrule, how to satisfy both assignments simultaneously. (Without such static analyses, it is possible to rely on user-driven inheritance conflict resolution).

EXAMPLE 8 - TNS



- **Results/Findings** (according to goals)
  (1) The current implementation of TNs does not provide warnings or resolution strategies for diamonds; the current implementation simply executes the assignments of the first super-transition found in the compilation process.

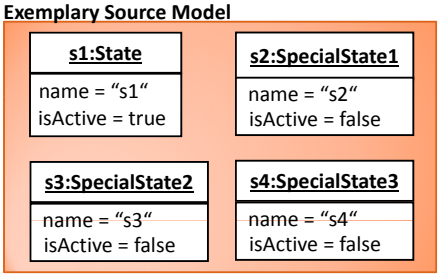
EXAMPLE 8 – ATL



- **Results/Findings** (according to goals)
  (1) No support for multiple inheritance available

EXAMPLE 8 – ETL

Exemplary Source Model




```
rule State2Place
 transform mElem : Statemachine!State
 to elem : Petrinet!Place {
  elem.name := mElem.name;
  elem.hasToken := mElem.isActive;
 }

rule SpecialState12Place
 transform mElem : Statemachine!State1
 to elem : Petrinet!SpecialPlace1
 extends State2Place {
  elem.name := mElem.name + 'subrule1';
}

rule SpecialState22Place
 transform mElem : Statemachine!SpecialState2
 to elem : Petrinet!SpecialPlace2
 extends State2Place {
  elem.name := mElem.name + 'subrule2';
}

rule SpecialState32Place
 transform mElem : Statemachine!SpecialState3
 to elem : Petrinet!SpecialPlace3
 extends SpecialState12Place,
         SpecialState22Place {
 }
```

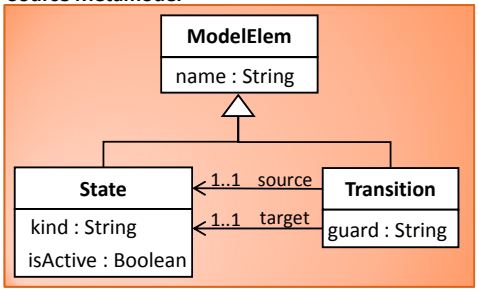
Target Model produced by ETL

- **Results/Findings** (according to goals)
  (1) Compile Error: „`Circular extension detected in rule State2Place`"; therefore diamond rule inheritance is not allowed, even if no problems arises in the definition (which would be the case, if, e.g., `SpecialState32Place` would override the `name` assignment)
  This is inferred from the fact that a compile-time error arises in any case of diamond rule inheritance

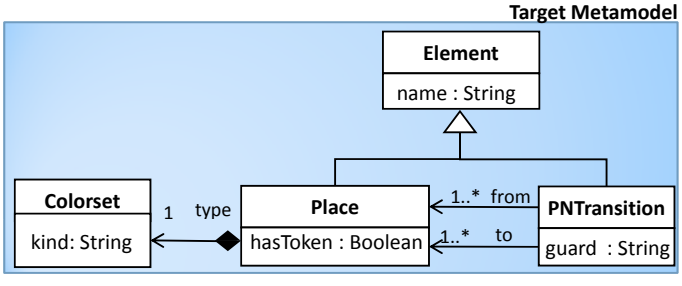
EXAMPLE 9

- **Description**
  - In this example `ModelElems` should be transformed into `Elements` and `States` into `Places` and a `Colorset`
- **Goal(s) of Evaluation**
  (1) Check, if the number of target elements may be extended in subrules

Source Metamodel

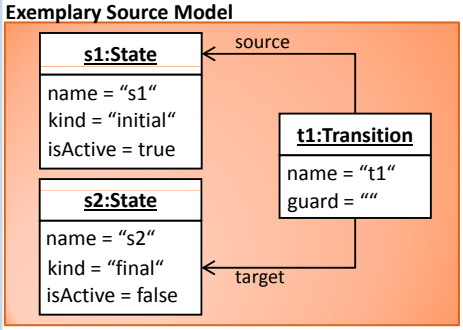


Target Metamodel



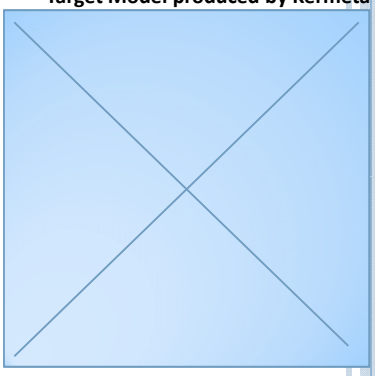
EXAMPLE 9 - KERMETA

Exemplary Source Model



See Example 6

Target Model produced by Kermeta

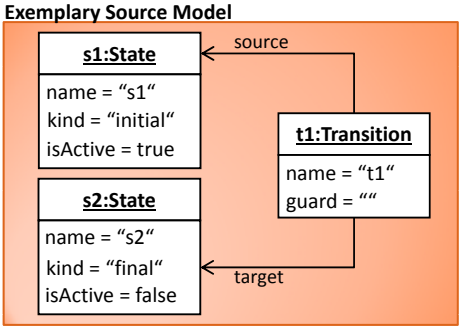


- **Results/Findings** (according to goals)
  - (1) Method signatures must not be changed -> compile-time error

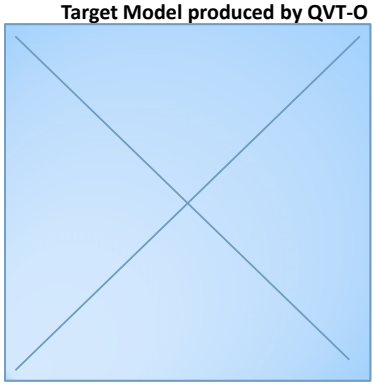
```
CONSTRAINT-CHECKER: 'conditionFulFilled' and
'conditionFulFilled' do not have the same number of
parameters.
```


EXAMPLE 9 – QVT-O (1/2)

Exemplary Source Model



Target Model produced by QVT-O



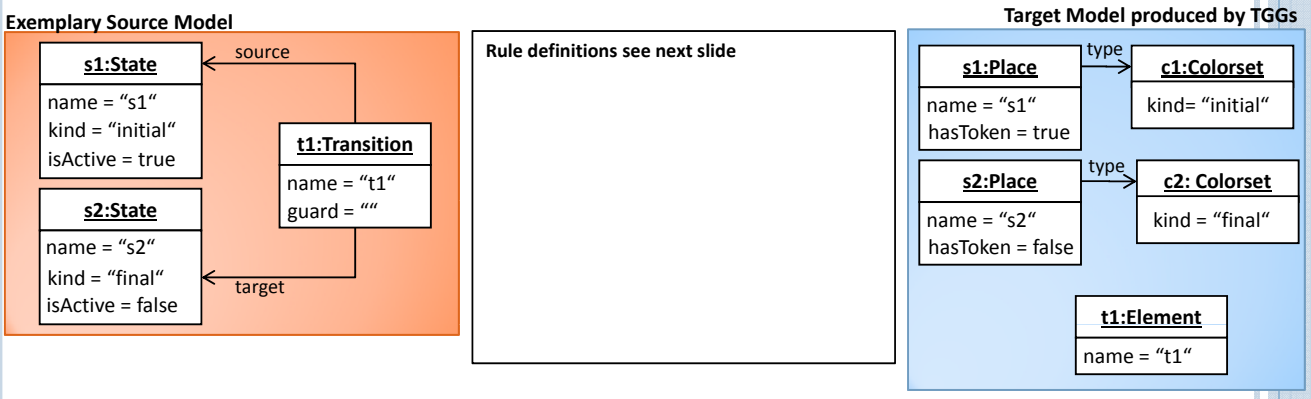
- **Results/Findings** (according to goals)
  - (1) QVT-O does not allow to extend the number of output parameters

```
Mapping operation 'statemachine_1::ModelElem::ModelElem2Element'
has non-conformant signature for 'inherits' in
'statemachine_1::State::State2Place'
```

EXAMPLE 9 – QVT-O (2/2)

```
transformation testTrafo(in inModel : sm, out outModel : pn);

  main() {
    inModel.rootObjects()[Statemachine] -> map SM2Petri();
  }

  mapping Statemachine::SM2Petri() : PetriNet {
    //please note that specific rules must be called first!
    elements := self.elements[State] -> map State2Place().p;
    elements += self.elements[ModelElem] -> map ModelElem2Element();
  }

  mapping ModelElem::ModelElem2Element() : Element {
    name := self.name;
  }

  mapping State::State2Place() : p : Place, c : Colorset inherits ModelElem::ModelElem2Element {
    p.hasToken := self.isActive;
    p.type := c;
    c.kind := self.kind;
  }
```

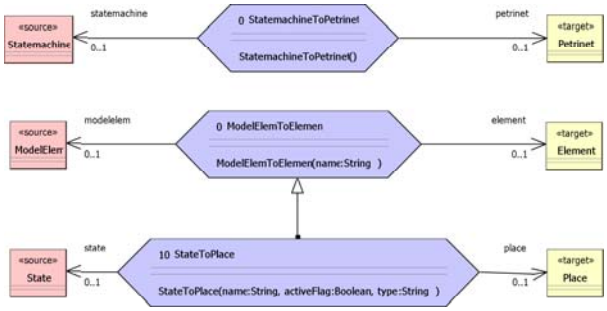

EXAMPLE 9 – TGGS (1/2)



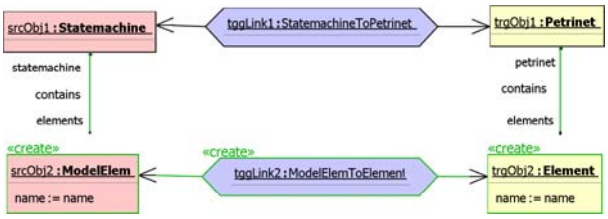
- **Results/Findings** (according to goals)
  - (1) Subrules must create at least the same elements (or subelements) compared to the superrule. To extend the number of elements to be created in a subrule is allowed.

EXAMPLE 9 – TGGS (2/2)

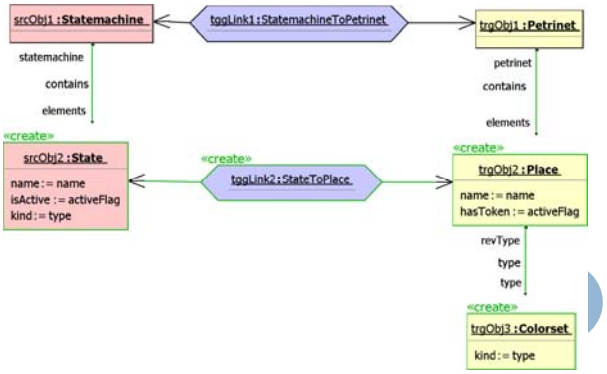
TGG-Schema (type level)



ModelElemToElement(...)

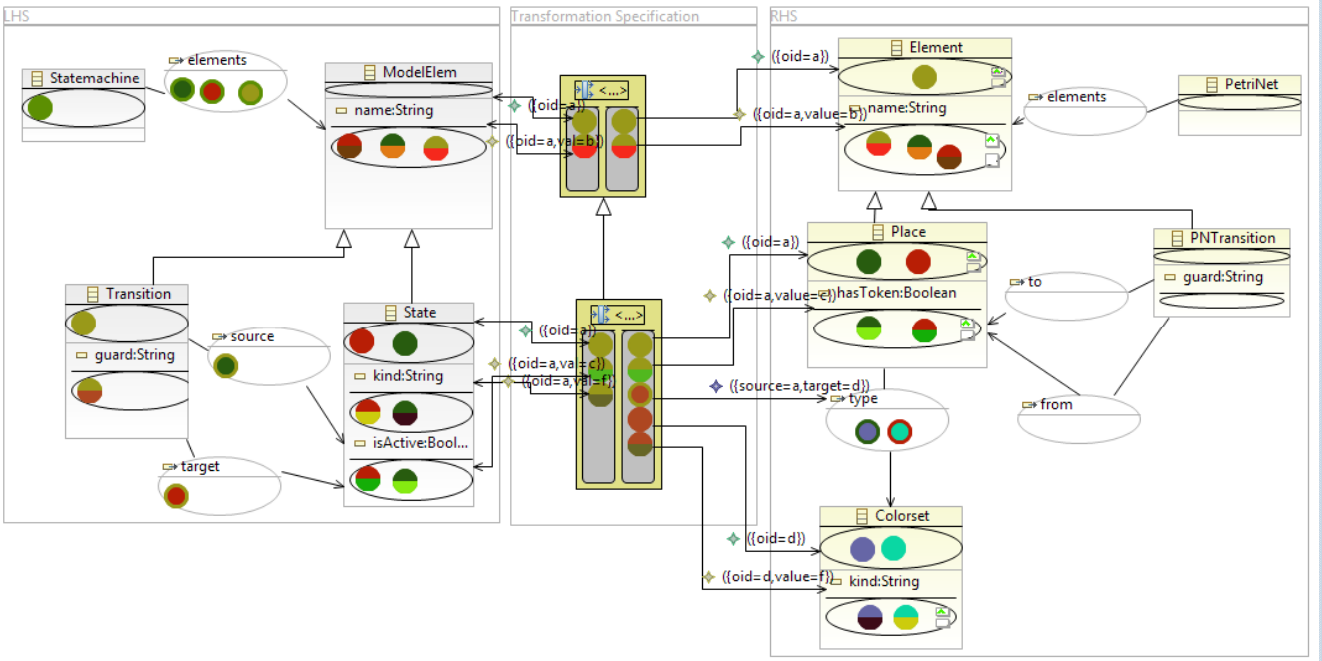


StateToPlace(...)

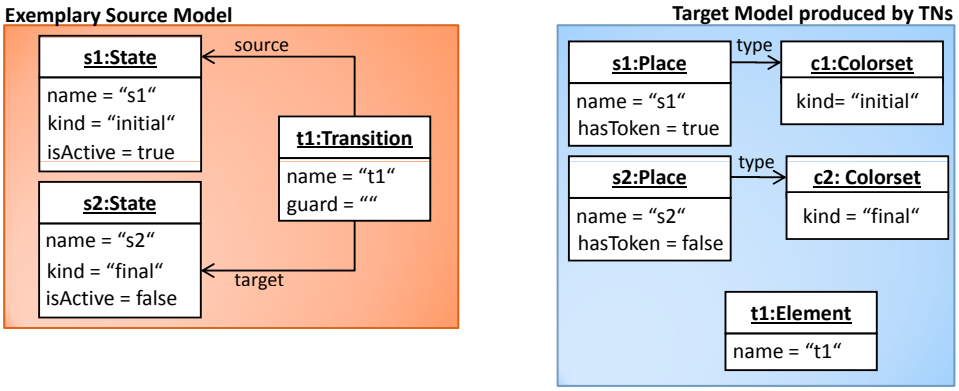


StatemachineToPetrinet(...)

EXAMPLE 9 – TNS (1/2)

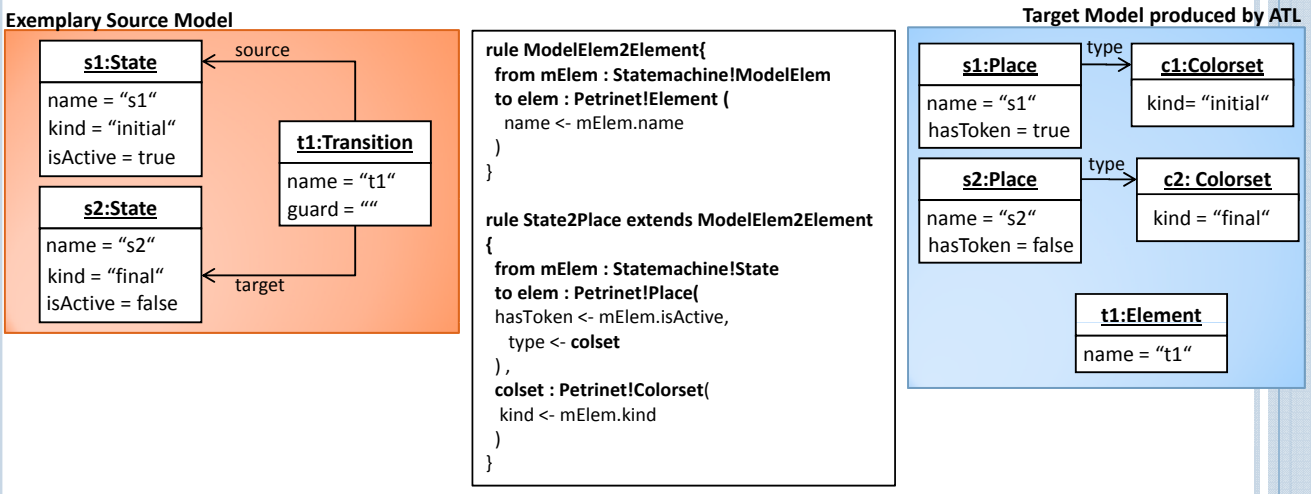


Example 9 – TNs (2/2)



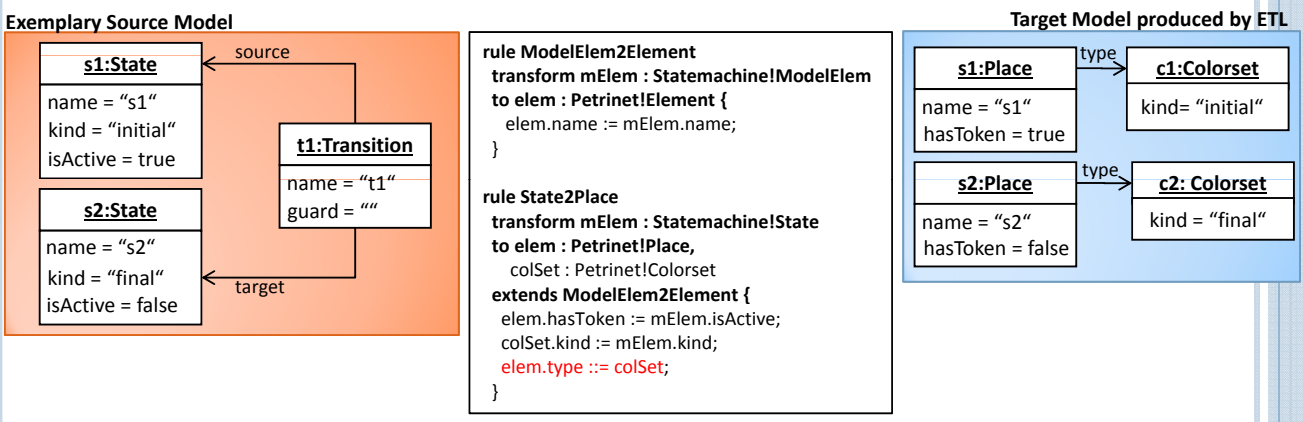
- **Results/Findings** (according to goals)
  - (1) Subrules must create at least the same elements (or subelements) compared to the superrule. It is allowed to add arbitrary extensions to subrules.

Example 9 – ATL



- **Results/Findings** (according to goals)
  - (1) ATL allows to extend the number of output parameters

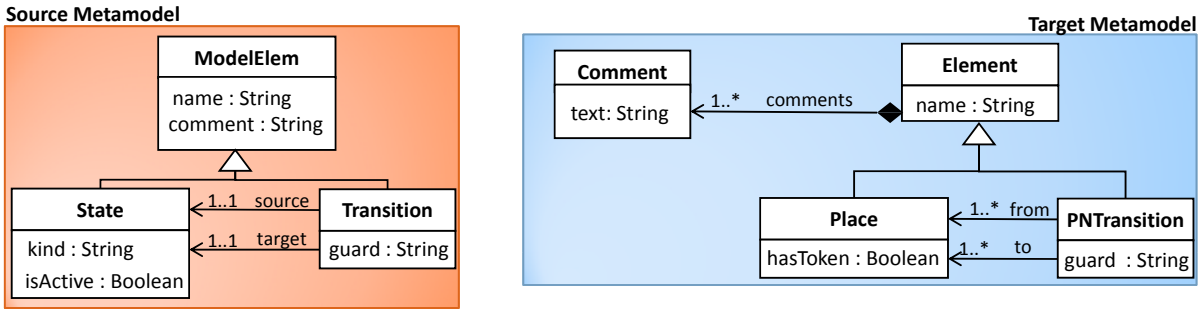
EXAMPLE 9 – ETL



- **Results/Findings** (according to goals)
  (1) ETL allows to extend the number of output parameters

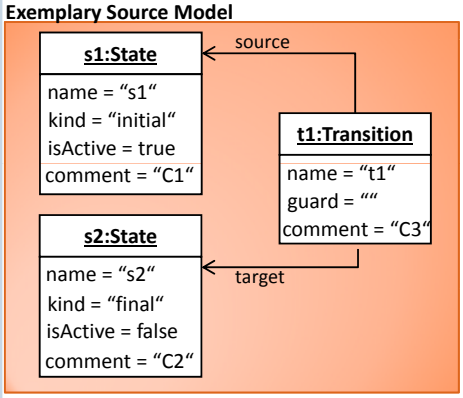
EXAMPLE 10

- **Description**
  - In this example, `ModelElems` should be transformed into `Elements` and `Comments`
- **Goal(s) of Evaluation**
  - Check, if the number of target elements may be reduced in subrules



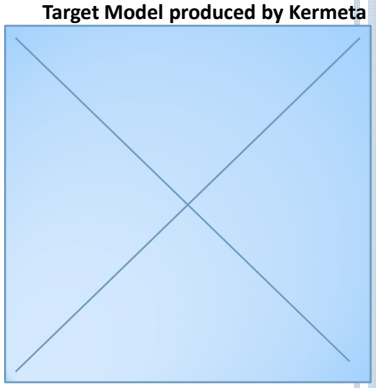
Example 10 - Kermeta

Exemplary Source Model



See Example 6

Target Model produced by Kermeta

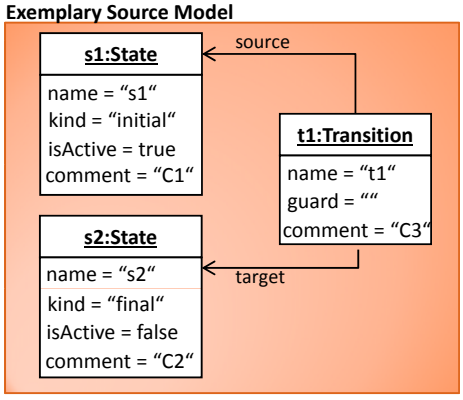


- **Results/Findings** (according to goals)
  - (1) Method signatures must not be changed -> compile-time error

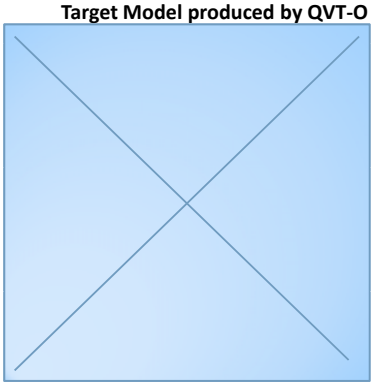
```
CONSTRAINT-CHECKER: 'conditionFulFilled' and
'conditionFulFilled' do not have the same number of
parameters.
```


Example 10 – QVT-O (1/2)

Exemplary Source Model



Target Model produced by QVT-O



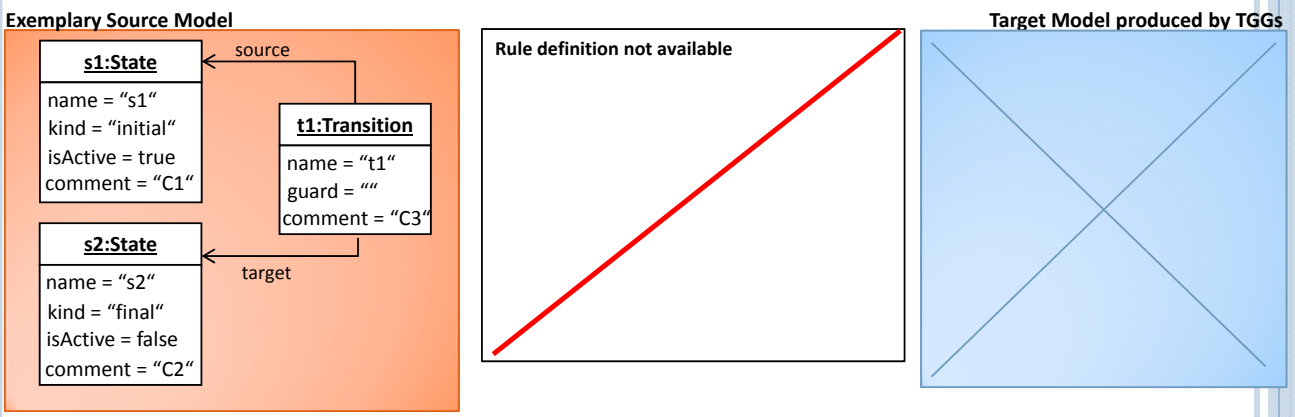
- **Results/Findings** (according to goals)
  - (1) QVT-O does not allow to remove output elements

```
Mapping operation
'statemachine_1::ModelElem::ModelElem2Element' has
non-conformant signature for 'inherits' in
'statemachine_1::State::State2Place'
```

EXAMPLE 10 – QVT-O (2/2)

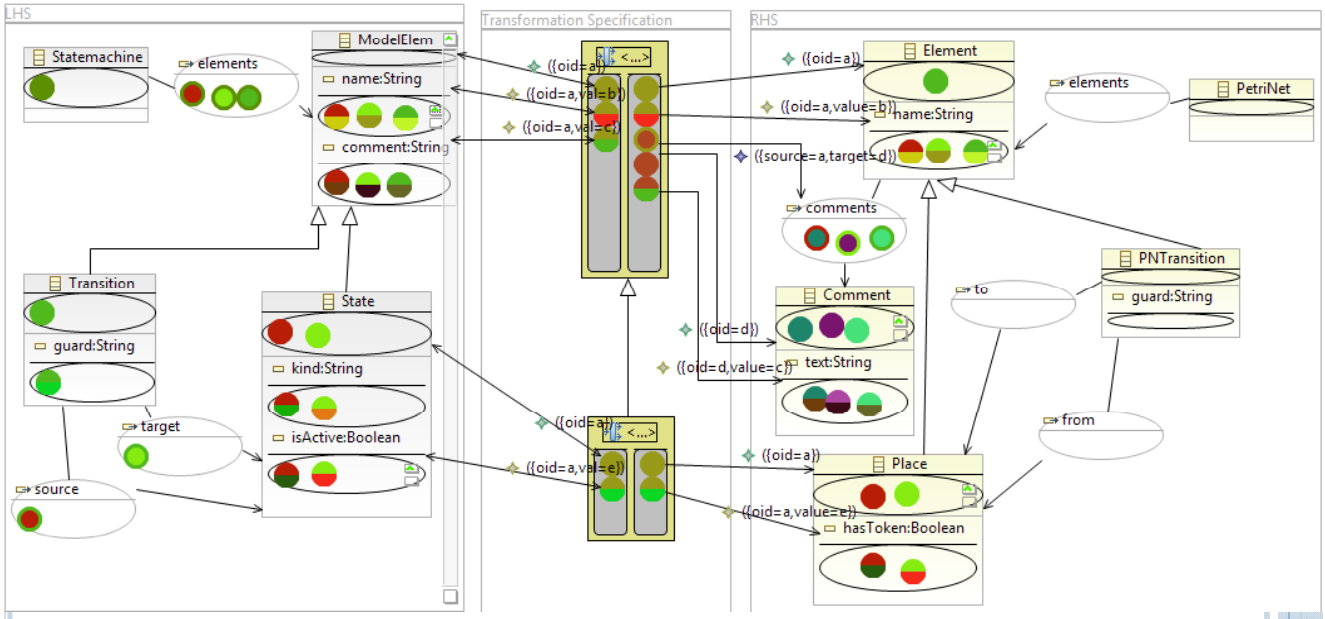
```
transformation testTrafo(in inModel : sm, out outModel : pn);

  main() {
    inModel.rootObjects()[Statemachine] -> map SM2Petri();
  }

  mapping Statemachine::SM2Petri() : PetriNet {
    //please note that specific rules must be called first!
    elements := self.elements[State] -> map State2Place();
    elements += self.elements[ModelElem] -> map ModelElem2Element().e;
  }

  mapping ModelElem::ModelElem2Element() : e: Element, c : Comment {
    e.name := self.name;
    e.comments := c;
    c.text := self.comment;
  }

  mapping State::State2Place() : Place inherits ModelElem::ModelElem2Element {
    hasToken := self.isActive;
  }
```


EXAMPLE 10 – TGGS



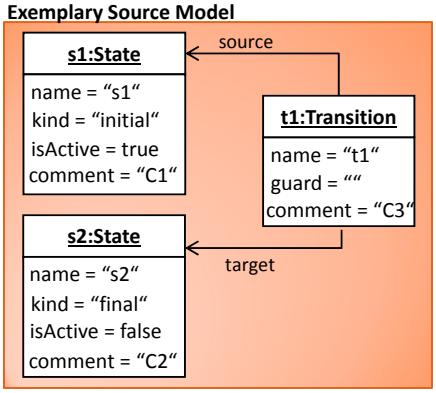
- **Results/Findings** (according to goals)
  - (1) Static analysis prohibit to reduce the number of elements to be created. Subrules must create a superset of elements compared to a superrule.

EXAMPLE 10 – TNS (1/2)

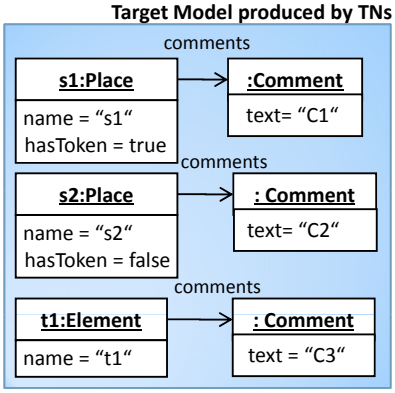


EXAMPLE 10 – TNS (2/2)

Exemplary Source Model

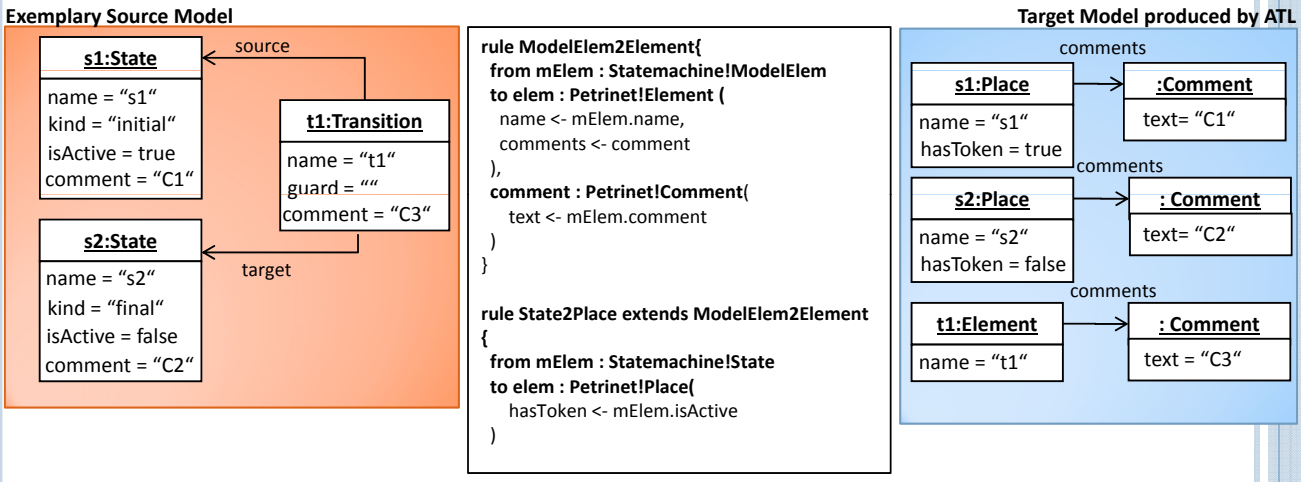


Target Model produced by TNs



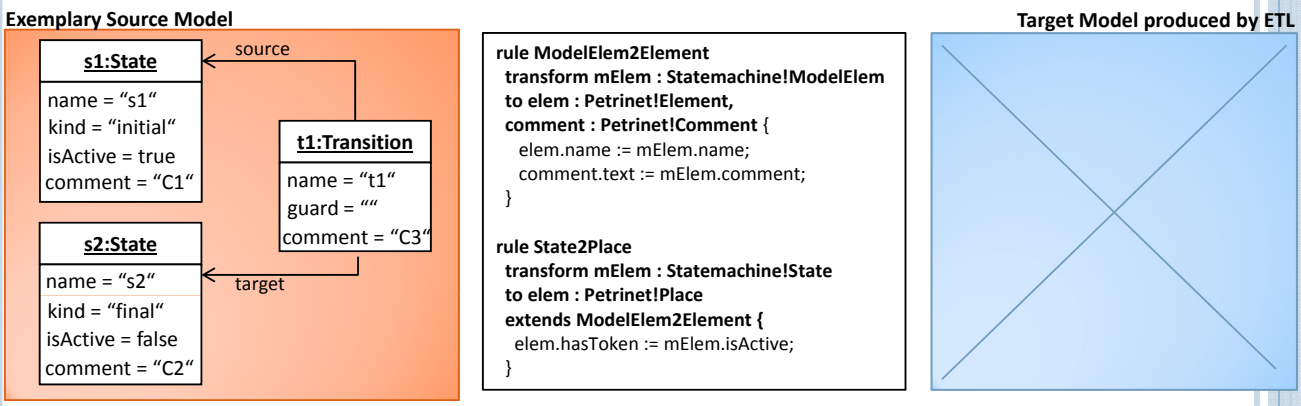
- **Results/Findings** (according to goals)
  - (1) TNs allow to reduce the number of output parameters in the syntax, but the inherited output elements are still produced (even if not specified again). Thus, the number cannot be restricted.

EXAMPLE 10 – ATL



- **Results/Findings** (according to goals)
  - (1) ATL allows to reduce the number of output parameters in the syntax, but the inherited output elements are still produced (even if not specified again). Thus, the number cannot be restricted.

EXAMPLE 10 – ETL



- **Results/Findings** (according to goals)
  - (1) ETL does NOT allow to reduce the number of output parameters; This is inferred from the fact, that a run-time exception arises