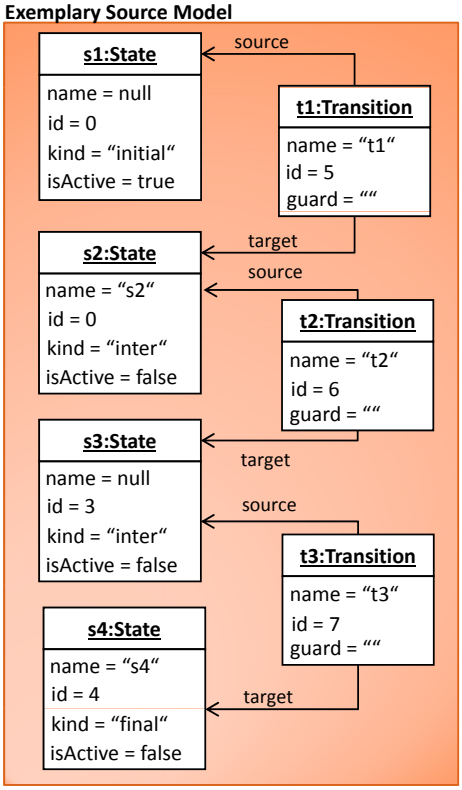
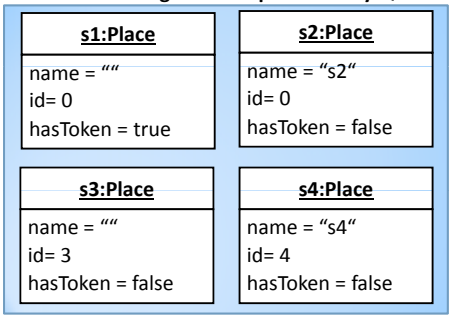
EXAMPLE 5 – QVT-O (1/2)

Exemplary Source Model



Target Model produced by QVT-O



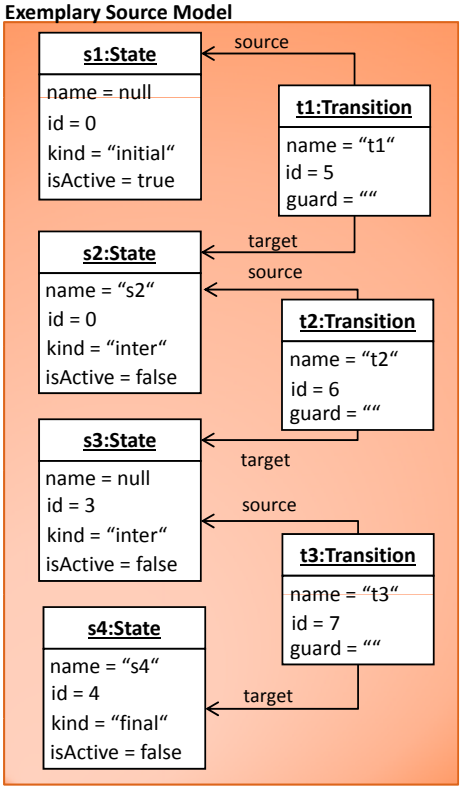
- **Results/Findings** (according to goals)
  - (1) Conditions are only evaluated to decide whether the code of the rule should be executed for a certain model element or not; no inheritance of conditions is performed

EXAMPLE 5 – QVT-O (2/2)

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

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

  mapping Statemachine::SM2Petri() : PetriNet {
    elements := self.elements[State] -> map State2Place();
  }

  abstract mapping ModelElem::ModelElem2Element() : Element
  when{self.name != null and self.name != ''}{
    dump('ModelElem2Element fuer ' + self.name);
    name := self.name;
  }

  abstract mapping SMElem::SMElem2PNElem() : PNElem
  when{self.id > 0}{
    dump('SMElem2PNElem fuer ' + self.id.toString());
    id := self.id;
  }

  mapping State::State2Place() : Place inherits ModelElem::ModelElem2Element, SMElem::SMElem2PNElem {
    dump('State2Place fuer ' + self.name);
    hasToken := self.isActive;
  }
```

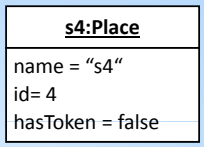
EXAMPLE 5 – TGGS (1/2)

Exemplary Source Model



Rule definitions see next slide

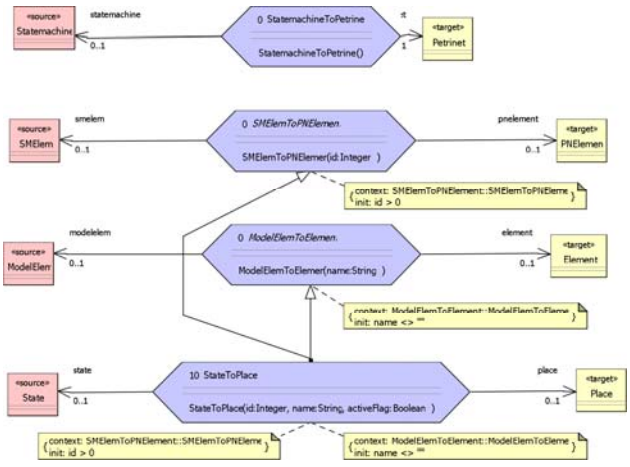
Target Model produced by TGGs



- **Results/Findings** (according to goals)
  - (1) TGGs support multiple inheritance of conditions. A subrule matches, if all (inherited) conditions are fulfilled.

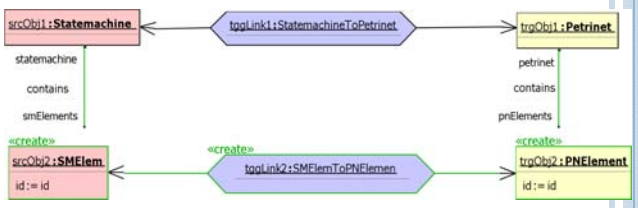
EXAMPLE 5 – TGGS (2/2)

TGG-Schema (type level)

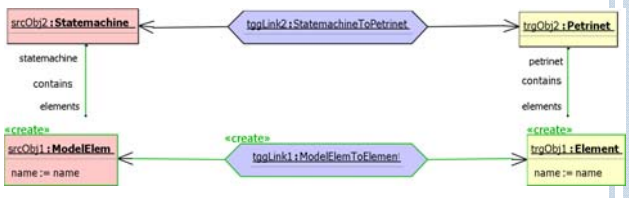


StatemachineToPetrinet(...)

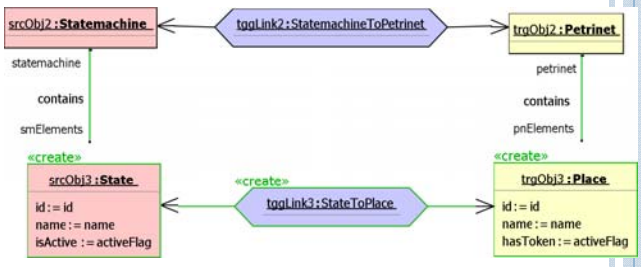
SMElemToPNElement(...)



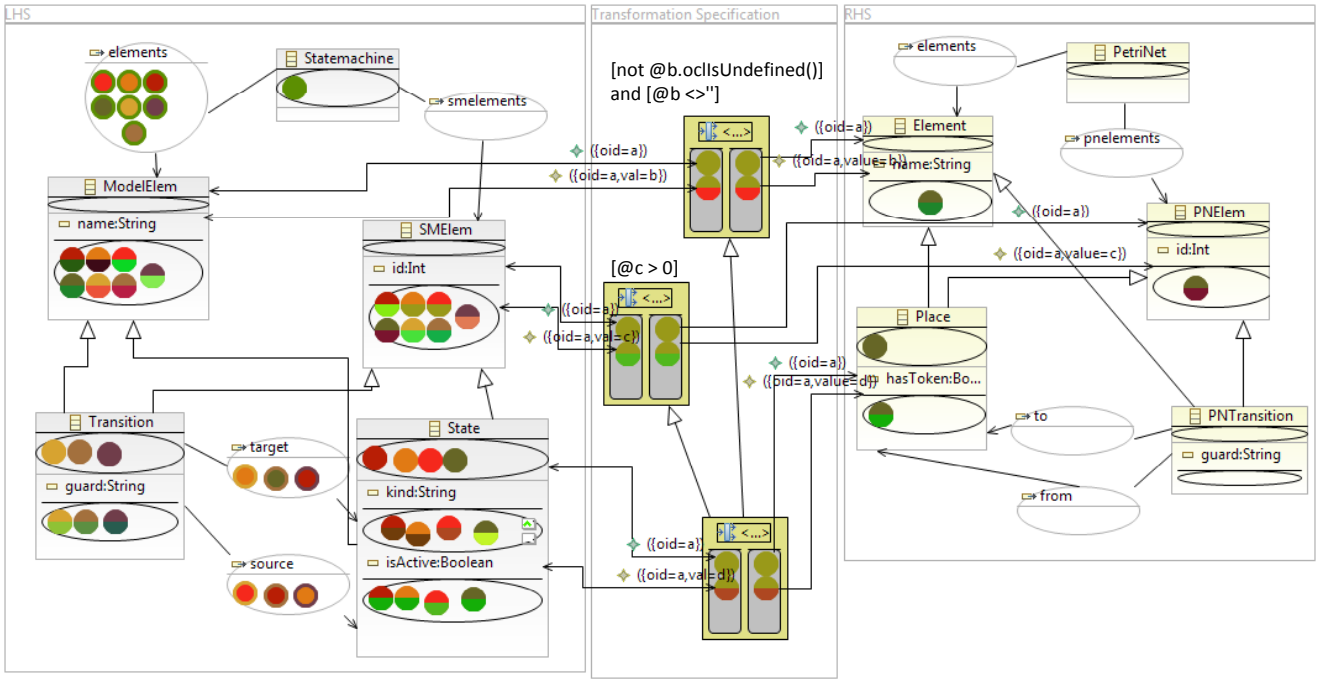
ModelElemToElemen(...)



StateToPlace

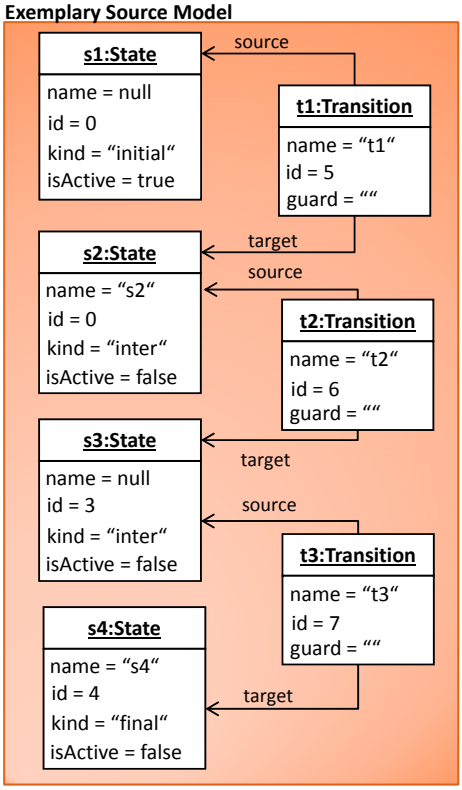


EXAMPLE 5 – TNS (1/2)

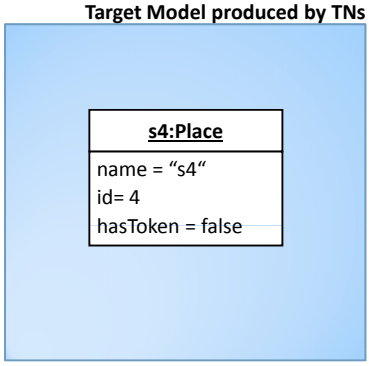


EXAMPLE 5 –TNS (2/2)

Exemplary Source Model



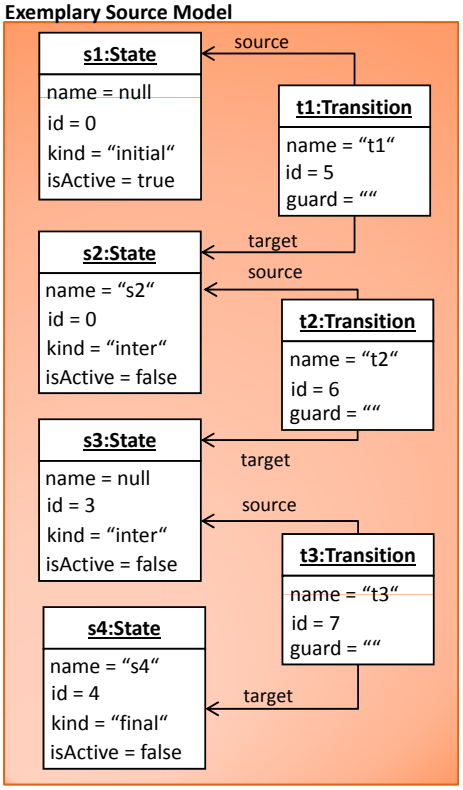
Target Model produced by TNs



- **Results/Findings** (according to goals)
  - (1) TNs support multiple inheritance of conditions. A subrule matches, if all (inherited) conditions are fulfilled.

Example 5 – ATL

Exemplary Source Model



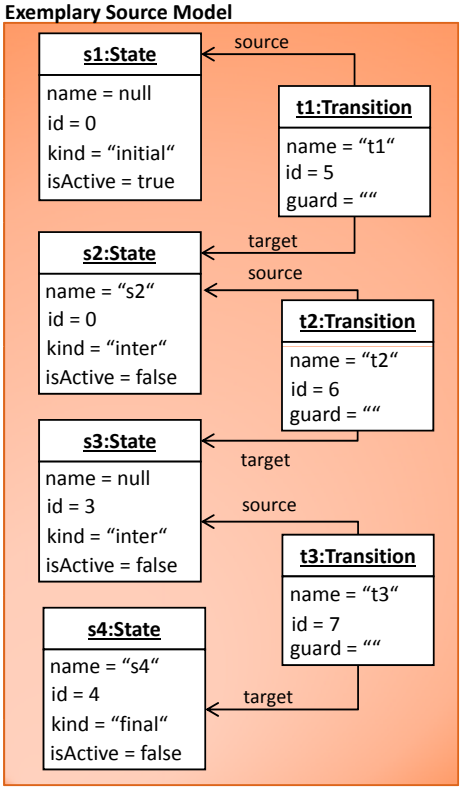
???

Target Model produced by ATL

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

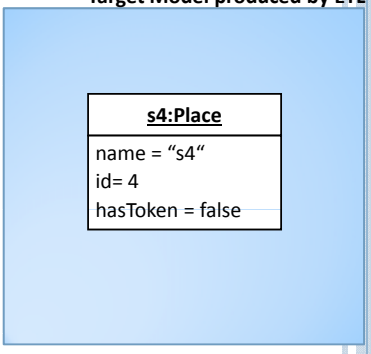
Example 5 – ETL

Exemplary Source Model




```
@abstract
rule ModelElem2Element
  transform mElem : Statemachine!ModelElem
  to elem : Petrinet!Element {
     guard : mElem.name <> null
             and mElem.name <> ''
     elem.name := mElem.name;
  }

@abstract
rule SMElem2PNElem
  transform mElem : Statemachine!SMElem
  to elem : Petrinet!PNElem {
     guard : mElem.id > 0
     elem.id := mElem.id;
  }

rule State2Place
  transform mElem : Statemachine!State
  to elem : Petrinet!Place
  extends ModelElem2Element,
SMElem2PNElem {
    elem.name := mElem.name;
    elem.hasToken := mElem.isActive;
  }
```

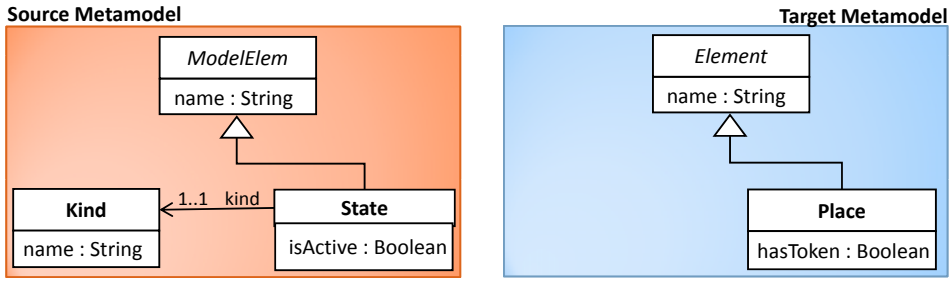
Target Model produced by ETL



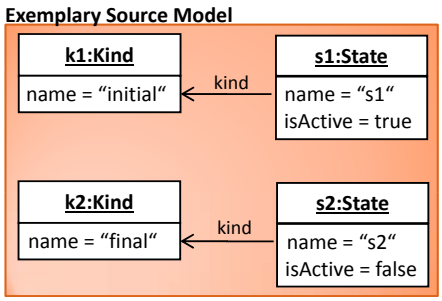
- **Results/Findings** (according to goals)
  - (1) Conditions of all subrules are evaluated (conjunction); therefore, a target `Place s4` gets generated, only

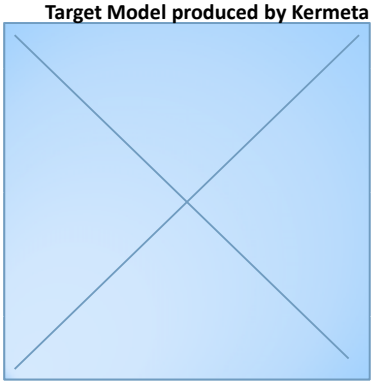
Example 6

- **Description**
  - In this example `ModelElems` should be transformed into `Elements` and `States` into `Places` by two inheriting rules in order to reuse the `name` assignment;
  - Moreover, only those `State` instances should be transformed, whose referenced `Kind` instance is unequal „`initial`“
- **Goal(s) of Evaluation**
  (1) Check, whether the number of input parameters may be extended in subrules



Example 6 – Kermeta (1/3)





- **Results/Findings** (according to goals)
  (1) Method signatures must not be changed -> compile-time errors, e.g.,
  `CONSTRAINT-CHECKER: 'conditionFulFilled' and 'conditionFulFilled' do not have the same number of parameters.`

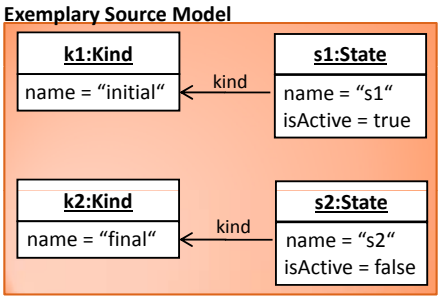
EXAMPLE 6 – KERMETA (2/3)

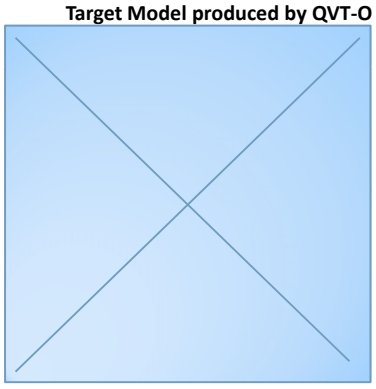
```
//transformation code for Statemachine2PetriNet
class Statemachine2PetriNet{

  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(Element))
      end
    }
  end
}

//transformation code for ModelElem2Element
abstract class ModelElem2Element{

  operation conditionFulFilled(m : ModelElem) : kermeta::standard::Boolean is do
    result := true
  end

  operation assignments(m : ModelElem, e : Element) is do
    e.name := m.name
  end

  operation referenceAssignments(m : ModelElem, e : Element, trace: Trace<Object, Object>) is do
  end
}
```

EXAMPLE 6 – KERMETA (3/3)

```
//Note that type Kind is for some reason not recognized -> maybe a keyword?
//transformation code for State2Place
class State2Place inherits ModelElem2Element{

  method conditionFulFilled(m : ModelElem, k : Kind) : kermeta::standard::Boolean is do
    result := super(m)
    result := result and (m.asType(State)).kind != "initial"
  end

  method assignments(m : ModelElem, k : Kind, e : Element) is do
    super(m,e)
    (e.asType(Place)).hasToken := (m.asType(State)).isActive
  end


  method referenceAssignments(m : ModelElem, k : Kind, e : Element, trace: Trace<Object, Object>) is do
    super(m,e,trace)
  end
}
```

EXAMPLE 6 – QVT-O (1/2)





- **Results/Findings** (according to goals)
  (1) Number of input elements must not be extended -> compile-time error

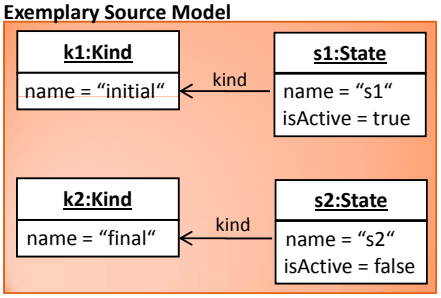
```
Mapping operation
'statemachine_1::ModelElem::ModelElem2Element' has non-
conformant signature for 'inherits' in
'statemachine_1::State::State2Place'
```


EXAMPLE 6 – QVT-O (2/2)

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

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

  mapping Statemachine::SM2Petri() : PetriNet {
    elements := self.elements[State] -> map State2Place();
    elements += self.elements[ModelElem] -> map ModelElem2Element();
  }

  mapping ModelElem::ModelElem2Element() : Element
  when{self.name != null and self.name != ''}{
    dump('ModelElem2Element fuer ' + self.name);
    name := self.name;
  }

  mapping State::State2Place(in k : Kind) : Place inherits ModelElem::ModelElem2Element {
    dump('State2Place fuer ' + self.name);
    hasToken := self.isActive;
  }
```

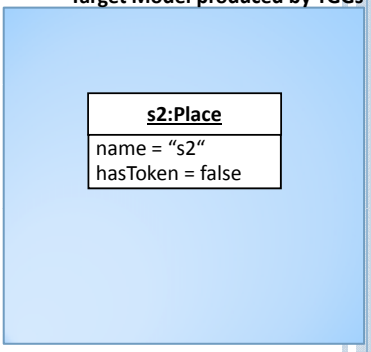
EXAMPLE 6 – TGGS (1/2)

Exemplary Source Model



Rule definitions see next slide

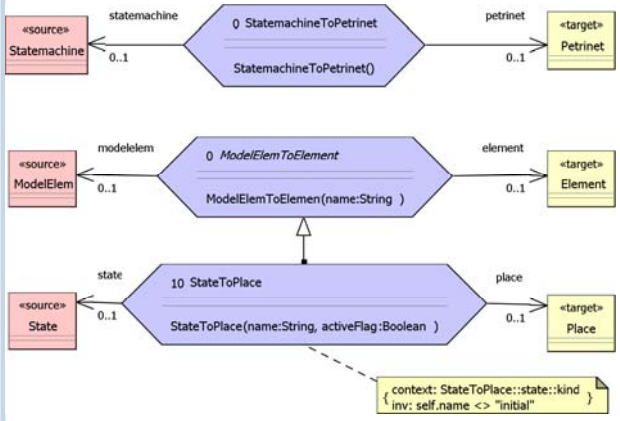
Target Model produced by TGGs



- **Results/Findings** (according to goals)
  - (1) TGGs support extended parameter lists in subrules; Nevertheless, a subrules must inherit all parameters of its superrules.

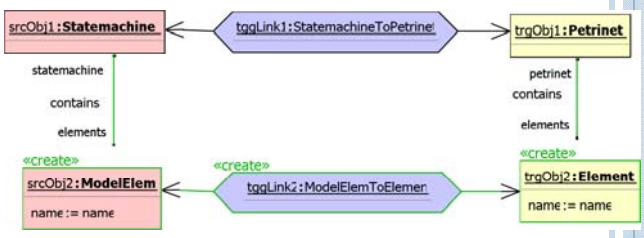
EXAMPLE 6 – TGGS (2/2)

TGG-Schema (type level)

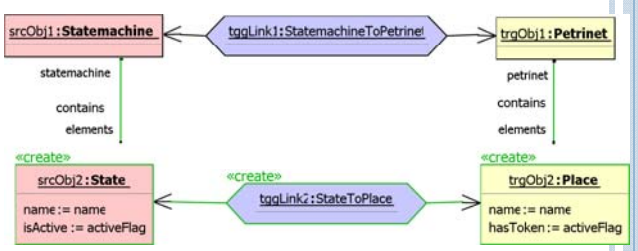


StatemachineToPetrinet(…)

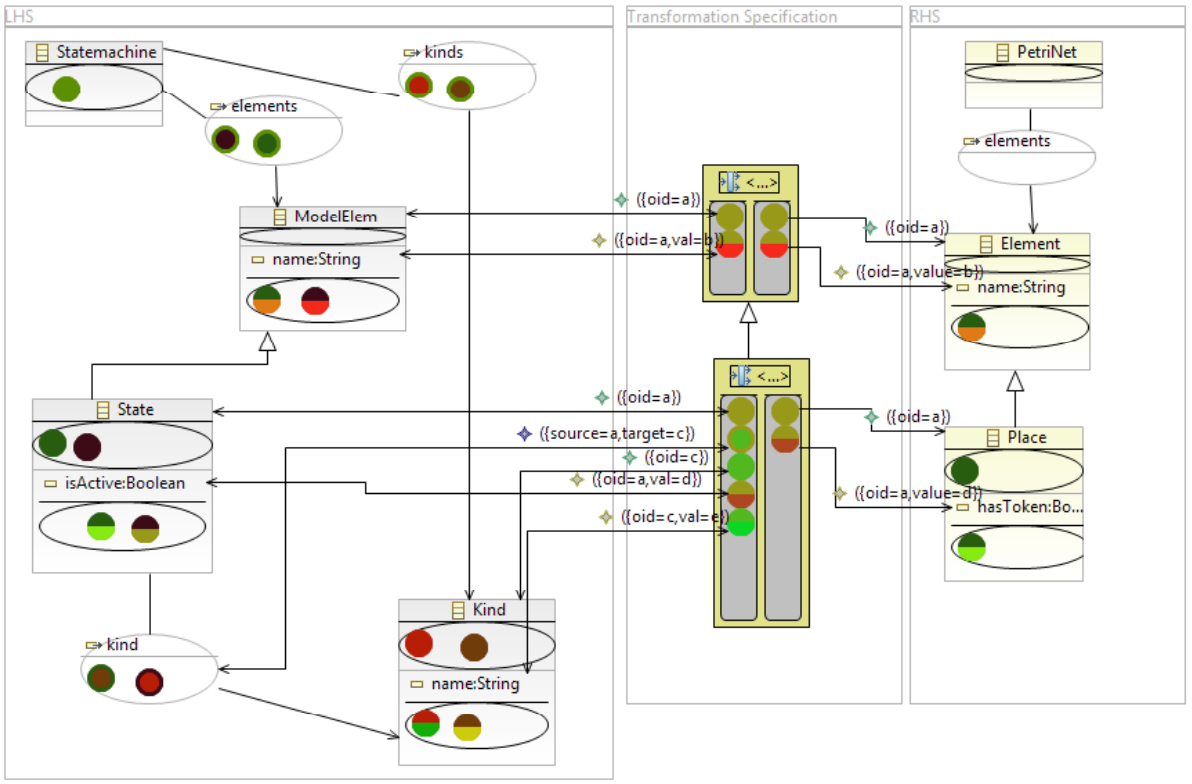
ModelElemToElement(…)



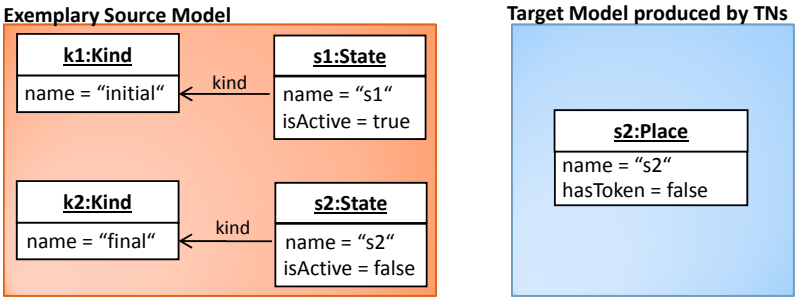
StateToPlace(…)



EXAMPLE 6 – TNS (1/2)



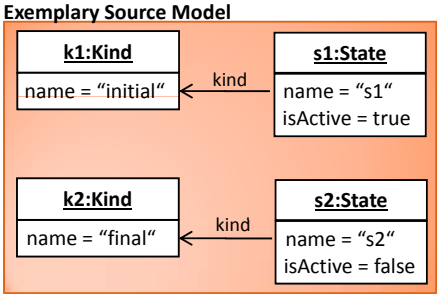
EXAMPLE 6 – TNS (2/2)



- **Results/Findings** (according to goals)
  - (1) TNs support extended parameter lists for subrules; Nevertheless, a subrule inherits all parameters of its superrules.

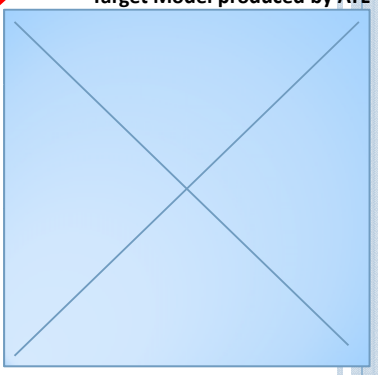
EXAMPLE 6 – ATL

Exemplary Source Model




```
abstract rule ModelElem2Element{
 from mElem : Statemachine!ModelElem
 to elem : Petrinet!Element (
   name <- mElem.name
 )
}

rule State2Place extends ModelElem2Element {
 from mElem : Statemachine!State,
        kind : Statemachine!Kind
        (kind.name <> 'initial')
 to elem : Petrinet!Place (
   hasToken <- mElem.isActive
 )
}
```

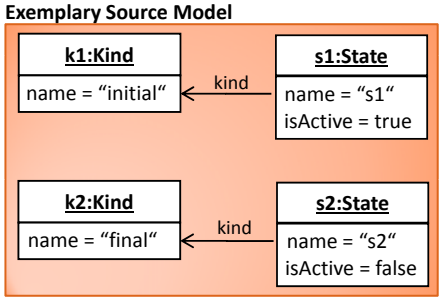
Target Model produced by ATL



- **Results/Findings** (according to goals)
  - (1) Input pattern in a subrule must not be changed -> run-time error (`org.eclipse.m2m.atl.engine.emfvm.VMException: Unable to access __xmiID__ on OclUndefined`)

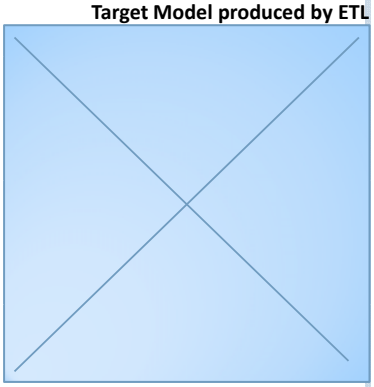
EXAMPLE 6 – ETL

Exemplary Source Model




```
@abstract
rule ModelElem2Element
 transform mElem : Statemachine!ModelElem
 to elem : Petrinet!Element {
    elem.name := mElem.name;
 }

rule State2Place
 transform mElem : Statemachine!State,
             kind : Statemachine!Kind
 to elem : Petrinet!Place
 extends ModelElem2Element {
  guard : kind.name <> 'initial'
  elem.hasToken := mElem.isActive;
 }
```

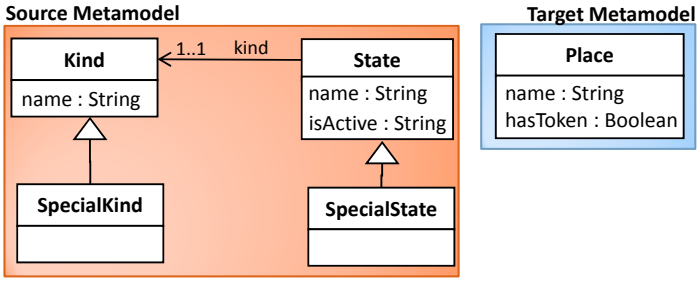
Target Model produced by ETL



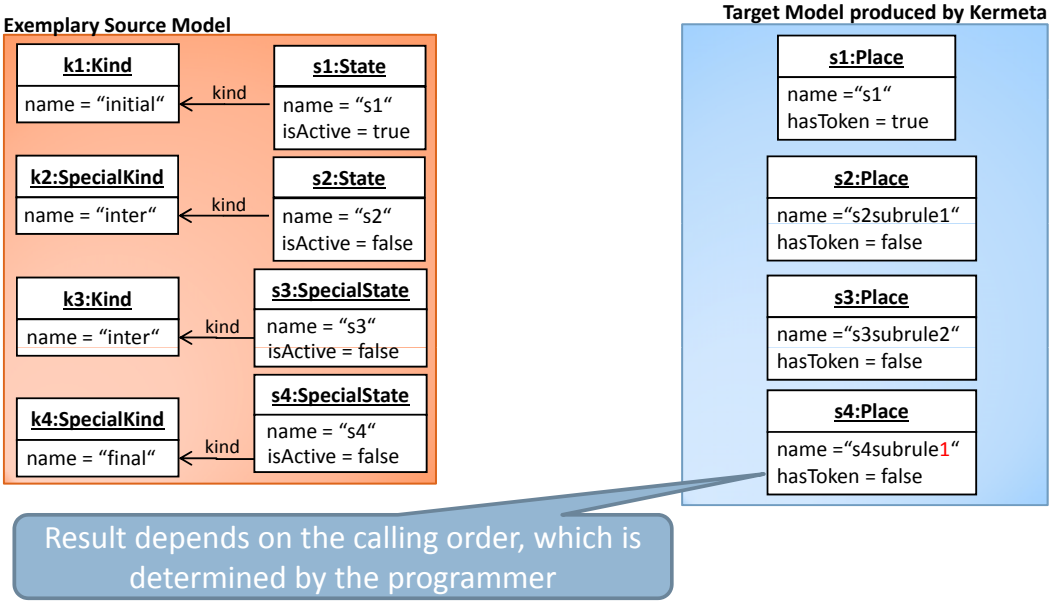
- **Results/Findings** (according to goals)
  - (1) ETL does not allow for multiple input elements (syntax error)

EXAMPLE 7

- **Description**
  - In this example `States` should be transformed into `Places`, whereby different assignments should take place according to the matched subtype combination
- **Goal(s) of Evaluation**
  (1) Check whether ambiguities in rule definitions are detected and/or resolved



EXAMPLE 7 – KERMETA (1/3)



- **Results/Findings** (according to goals)
  Ambigious Rule definitions may not be recognized in Kermeta, since the inheriting methods must exhibit the same signature anyway; therefore the result depends on the calling order determined by the programmer

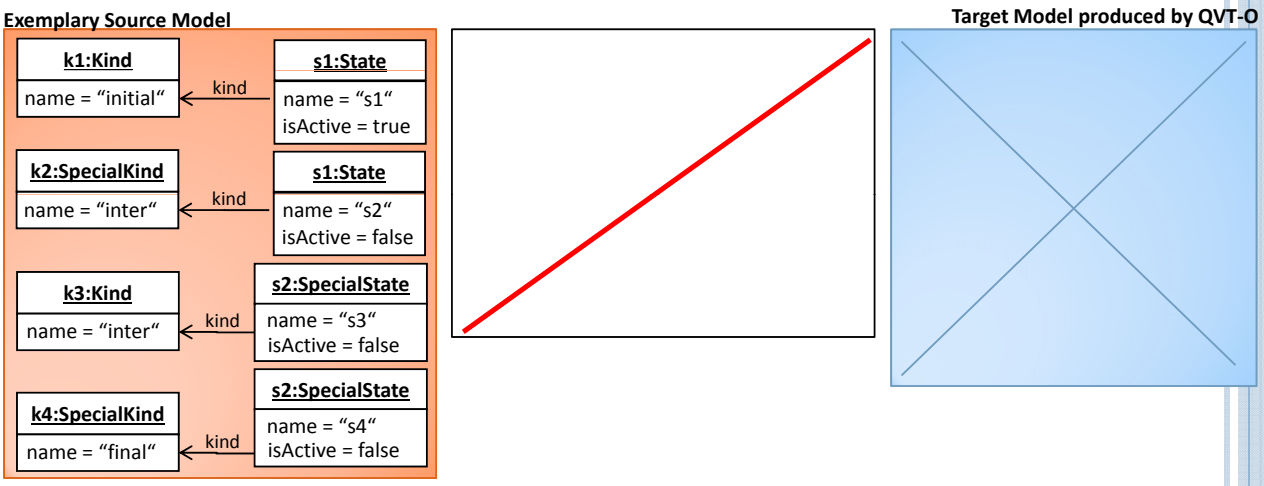
EXAMPLE 7 – KERMETA (2/3)

```
//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 StateKind2Place
class StateKind2Place{

  operation conditionFulFilled(s : State, k : Kind) : kermeta::standard::Boolean is do
    result := (s.kind == k)
  end

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

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

EXAMPLE 7 – KERMETA (3/3)

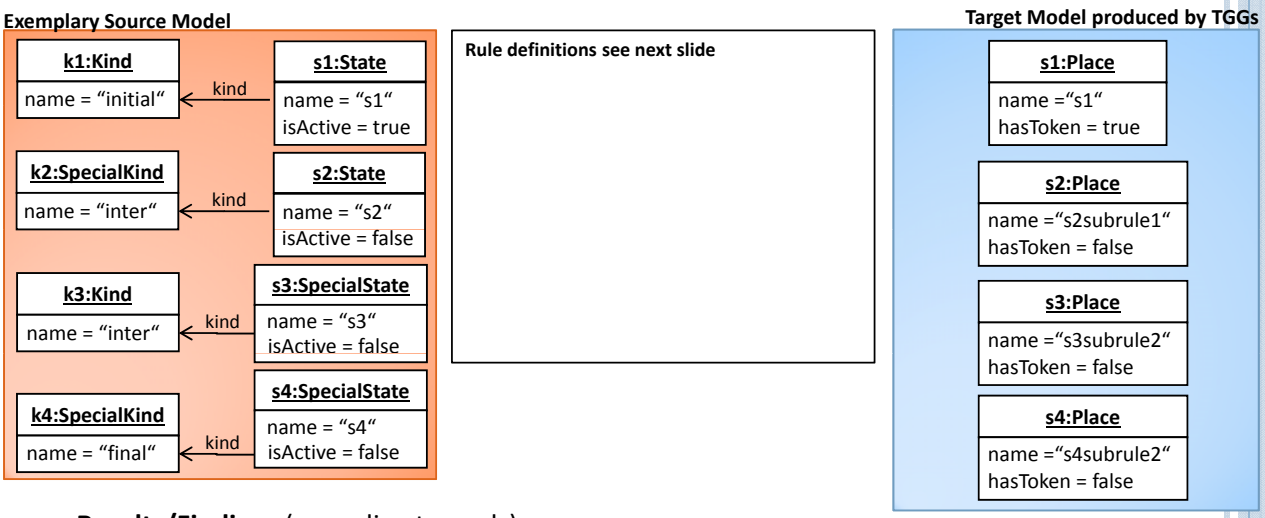
```
//transformation code for StateSpecialKind2Place
class StateSpecialKind2Place inherits StateKind2Place{

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

  method assignments(s : State, k : Kind, p : Place) is do
    super(s,k,p)
    //if specific features should be accessed, cast would be needed
    p.name := s.name + "subrule1"
  end

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

//transformation code for SpecialStateKind2Place
class SpecialStateKind2Place inherits StateKind2Place{

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

  method assignments(s : State, k : Kind, p : Place) is do
    super(s,k,p)
     //if specific features should be accessed, cast would be needed
    p.name := s.name + "subrule2"
  end

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

EXAMPLE 7 – QVT-O



- **Results/Findings** (according to goals)
  - (1) QVT-O does not allow for multiple input elements (syntax error), i.e., multiple dispatching is not supported; however, multiple input elements may be simulated with in parameters

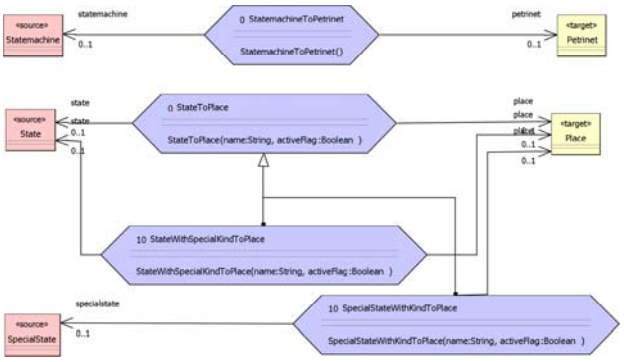
EXAMPLE 7 – TGGS (1/2)



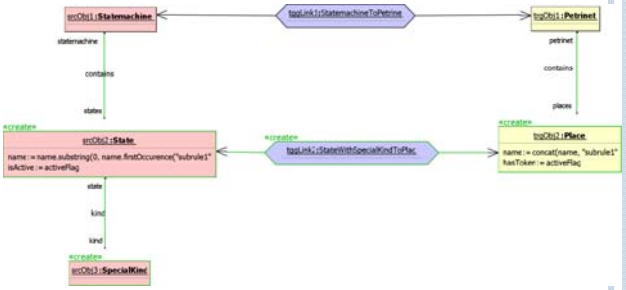
- **Results/Findings** (according to goals)
  - (1) TGGs do not provide ambiguity resolution; Nevertheless, in this example, it is unambiguous that the translation of `s4:SpecialState` should be done by executing rule `SpecialStateWithKindToPlace`. Since `k4:SpecialKind` is also instanceOf `Kind`, this rule matches and `s4:Place` is created.

EXAMPLE 7 – TGGS (2/2)

TGG-Schema (type level)

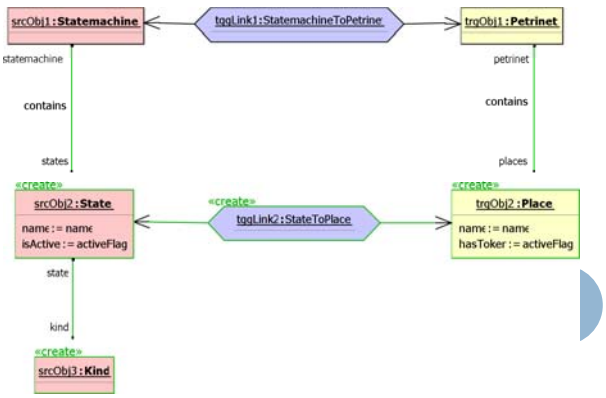


StateWithSpecialKindToPlace(…)

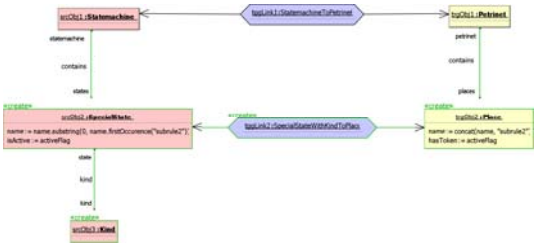


StatemachineToPetrinet(…)

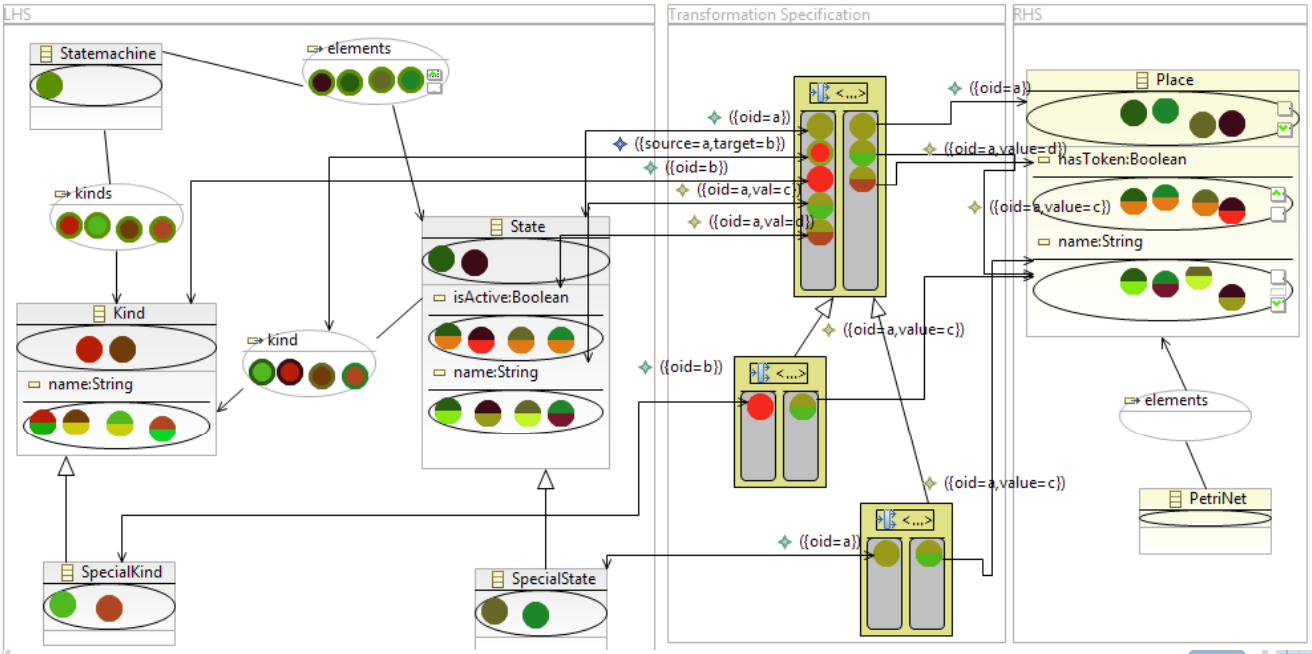
StateToPlace(…)



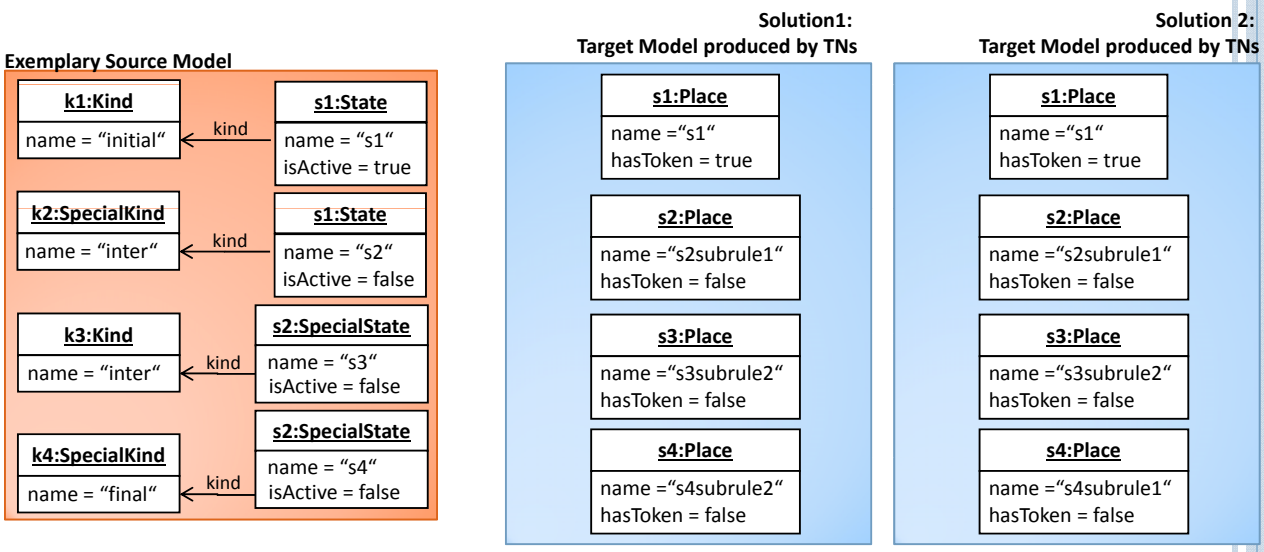
SpecialStateWithKindToPlace(…)



EXAMPLE 7 – TNS (1/2)

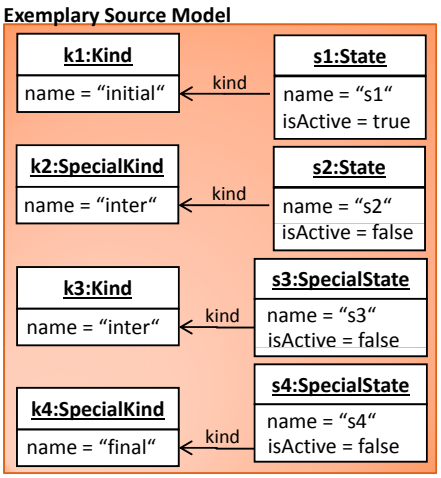


EXAMPLE 7 – TNS (2/2)



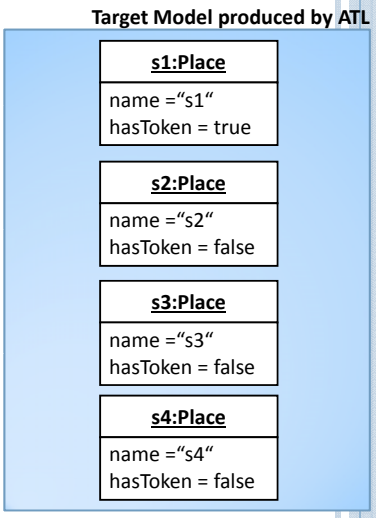
- **Results/Findings** (according to goals)
  - (1) The current implementation of TNs does not provide ambiguity warnings or resolutions; In this example this means that both subtransitions may fire in a non-deterministic manner for the combination `(s4,k4)`

EXAMPLE 7 – ATL (1/2)



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
  to elem : Petrinet!Place (
      name <- mElem.name + 'subrule1'
)
}

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



- **Results/Findings** (according to goals)
  - (1) Strange behavior; although there would be perfect matches for (s2, k2) and (s3, k3), these subrules are never executed