

# Analyzing ActionGUI models using SMT solvers

## ActionGUI models

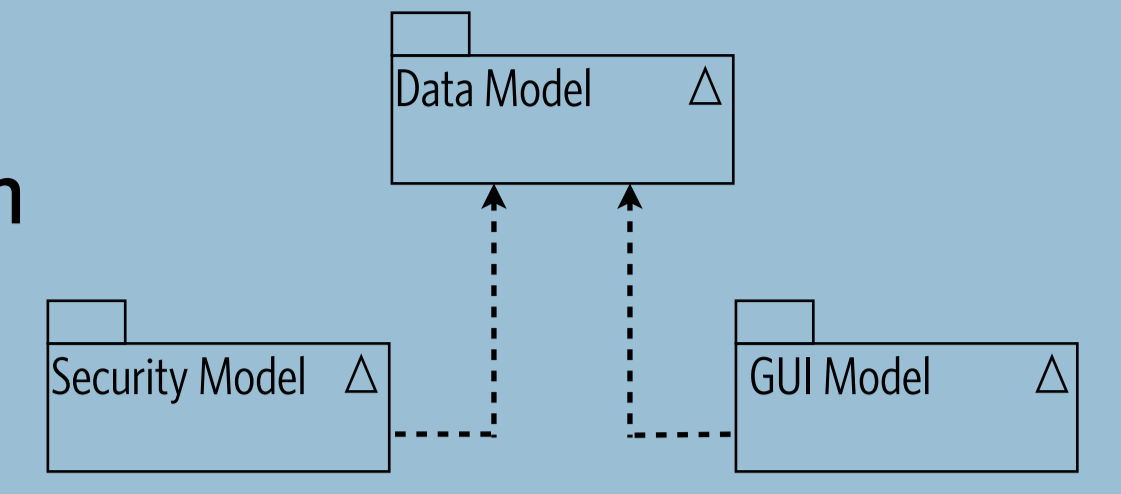
ActionGUI is a domain-specific language for modeling data-centric web applications with fine-grained access control policies.

An ActionGUI model consists of three models: a data model, a security model, and a GUI model.

Object Constraint Language (OCL) is used in ActionGUI models to specify data invariants, authorization constraints, and action conditions.

ActionGUI models have a well-defined semantics amenable to formal analysis.

Full data-centric web applications can be automatically generated from ActionGUI models.



**Data Model:** specifies the application domain (entities, attributes, associations, invariants).

```
Entity Employee {
  String id
  Set(Contract) contracts oppositeTo employee
}
Entity Contract {
  Real salary
  Employee employee oppositeTo contracts
}
```

```
context Contract inv
not(self.salary.oclIsUndefined())
self.employee->size() = 1

context Employee inv
not(self.id.oclIsUndefined())
Employee.allInstances()->excluding(self)->forall(e| e.id <> self.id)
self.contracts->notEmpty()
```

**Security Model:** specifies the application access control policy (roles, permissions, constraints).

```
Role Default {
  Contract {
    if (self.employee=caller)
      read salary
  }
  Employee {
    read id
  }
}
```

```
Role HHRR extends Default {
  Employee {
    create
    if (self.id.oclIsUndefined())
      update id
  }
  Contract {
    create
    read salary
    if (self.employee.isEmpty())
      update salary
  }
}
```

**GUI Model:** specifies the application GUI (widgets, events, actions).

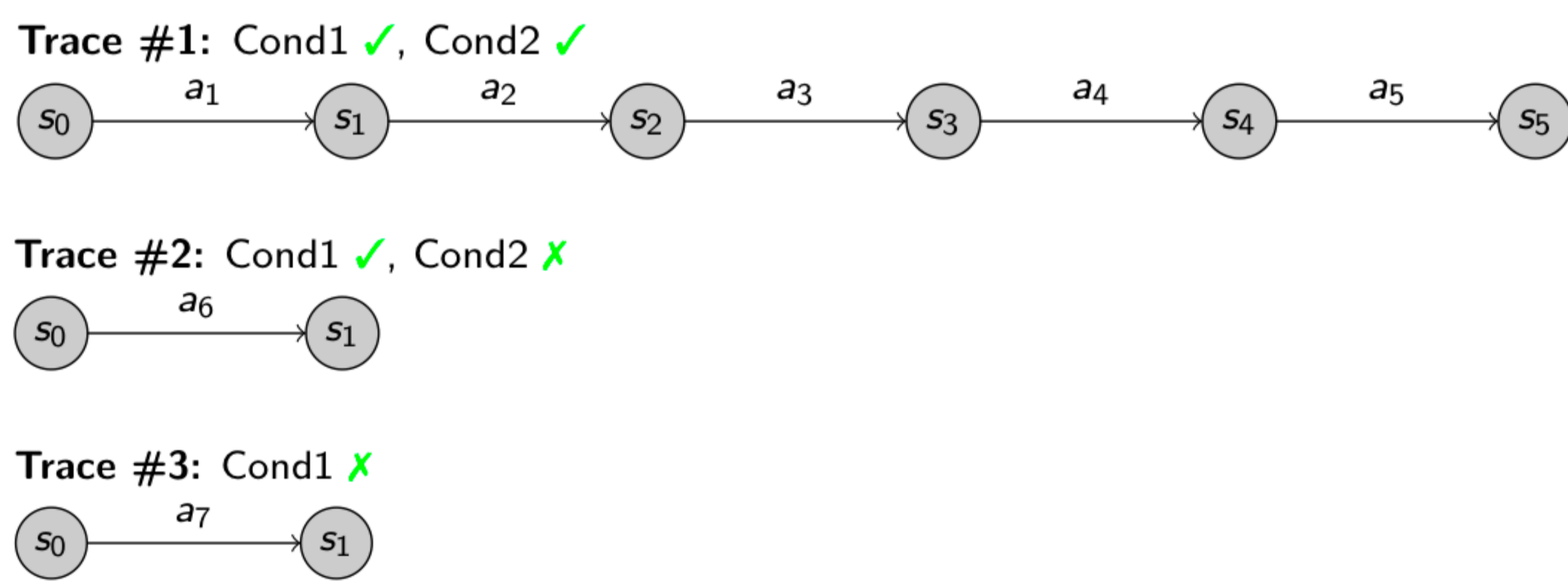
```
Window CreateEmployee
{
  Label Id
  TextField NewId
  Label Salary
  TextField NewSalary
  Button NewEmployee
}
```

```
CreateEmployee.NewEmployee. onClick {
  if (not([NewId.text].oclIsUndefined()) and
    not([NewSalary.text].oclIsUndefined())) then
    if (Employee.allInstances()->forall(e|e.id <> [NewId.text])) then
      newEmployee := new Employee // create
      [newEmployee].id := [NewId.text] // update
      newContract := new Contract // create
      [newContract].salary := [NewSalary.text] // update
      [newContract].employee += [newEmployee] // add
    else error := 'There exists an employee with this id.' // set
  else error := 'Some information is missing.' // set
}
```

## ActionGUI model properties

### Invariants preservation

for any possible sequence of action triggered by an event, if all the data model invariants are satisfied **before** the event, then they will be satisfied afterwards.



### Security awareness

(for each role) for any possible sequence of action triggered by an event, if all the data model invariants are satisfied **before** the event, then, for each action in the sequence, the authorization constraint will be satisfied **before** its execution is attempted.

```
Cond1 = Employee.allInstances()->forall(e|e.id <> newId)
Cond2 = not(NewId.oclIsUndefined()) and not(NewSalary.oclIsUndefined())
```

Authorization constraints, Auth()

```

a1 = [NewEmployee] := new Employee
a2 = [NewEmployee].id := [NewId.text]
a3 = [newContract] := new Contract
a4 = [newContract].salary := [NewSalary.text]
a5 = [newContract].employee := [NewEmployee]
a6 = error := 'There exists an employee...'
a7 = error := 'Some information...'

```

```

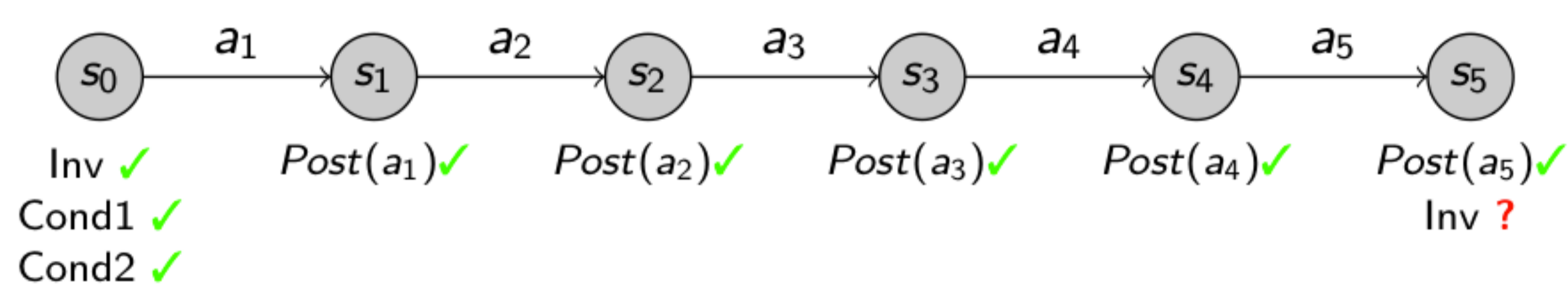
true
self.id.oclIsUndefined()
true
self.employee.isEmpty()
target<caller
true
true

```

## Checking ActionGUI model properties

- **STEP 1:** we formalize in OCL the data actions' post-conditions
- **STEP 2:** we use OCL2FOL to map OCL into first-order logic
- **STEP 3:** we use SMT solvers to solve the resulting satisfiability problems

E.g. Is the sequence of actions triggered by clicking on the button NewEmployee "invariants preserving"?

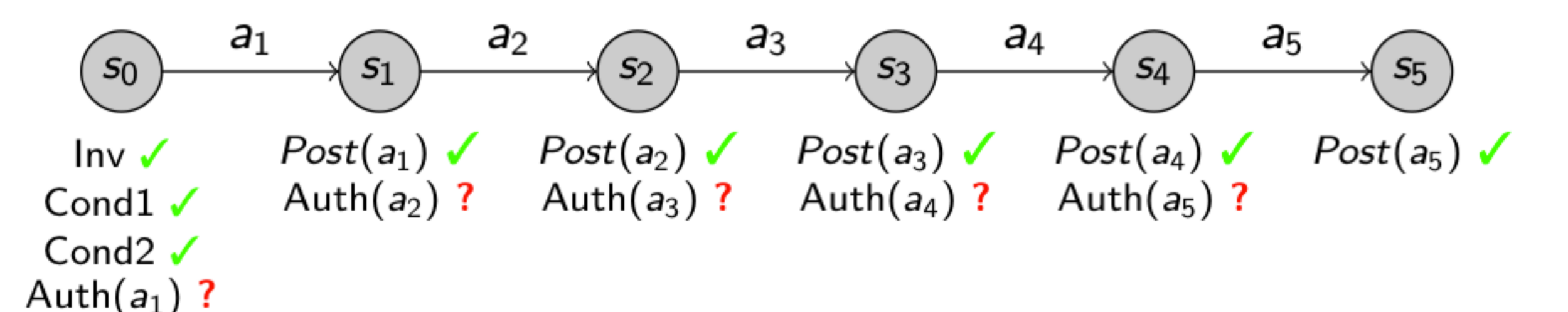


$$Inv(s_0) \wedge Cond1(s_0) \wedge Cond2(s_0) \wedge \left( \bigwedge_{j=1}^5 Post(a_j, s_{j-1}, s_j) \right) \wedge \neg Inv(s_j)$$

OCL2FOL

FOL satisfiability problem

E.g. Is the sequence of actions triggered by clicking on the button NewEmployee "security aware"?



For  $i = 1..5$

$$Inv(s_0) \wedge Cond1(s_0) \wedge Cond2(s_0) \wedge \left( \bigwedge_{j=1}^i Post(a_j, s_{j-1}, s_j) \right) \wedge \neg Auth(s_{j-1}, a_j)$$

OCL2FOL

FOL satisfiability problem

SMT solvers



Carolina Dania ([carolina.dania@imdea.org](mailto:carolina.dania@imdea.org)), Manuel Clavel ([manuel.clavel@imdea.org](mailto:manuel.clavel@imdea.org))

