Business Informatics Group

Vienna University of Technology

Towards Semantics-Aware Merge Support in Optimistic Model Versioning

Magdalena Widl

Business Informatics Group
Vienna University of Technology

Motivation and scope

Problem

Model evolution may have undesired effects

Motivation and scope

Problem

Model evolution may have undesired effects

Possible undesired effects

- Class diagram not instantiable
- Elements of one view are not modeled in another
- *Sequence diagram does not comply with state machine*

Motivation and scope

Problem

Model evolution may have undesired effects

Possible undesired effects

- Class diagram not instantiable
- Elements of one view are not modeled in another
- *Sequence diagram does not comply with state machine*

Vision

A formal framework to assist model evolution

Motivation and scope

For now, we look at *versioning*, more particularly at *semantic merging of sequence diagrams* We consider:

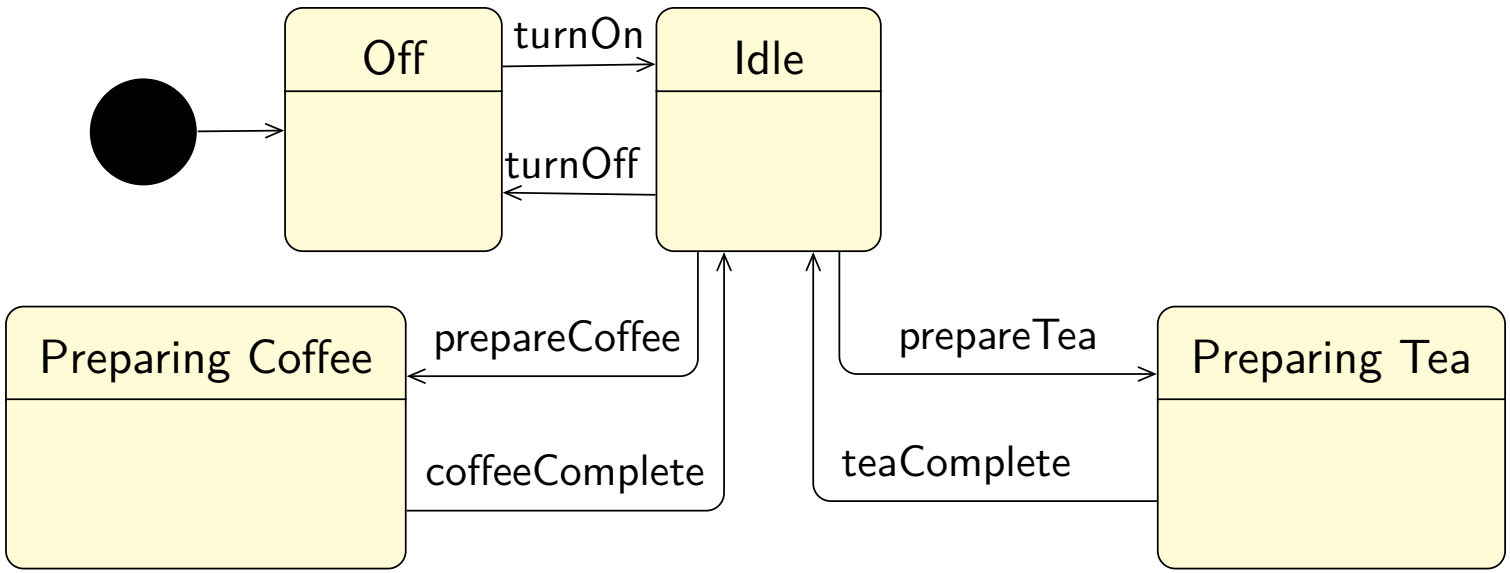
- State machines
- Sequence diagrams

Many complex constructs are omitted, but will be gradually added.

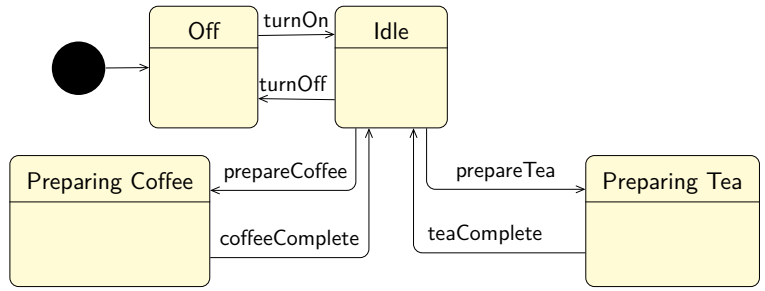
Example

Example

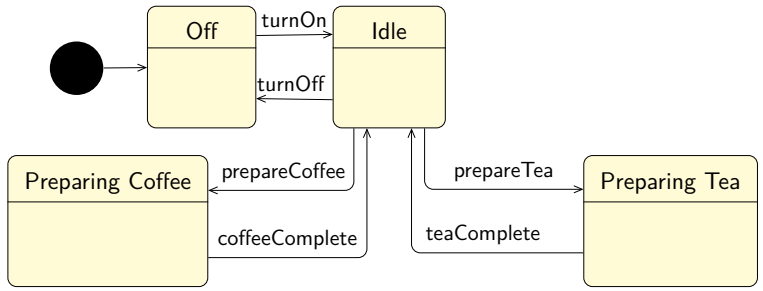
Example

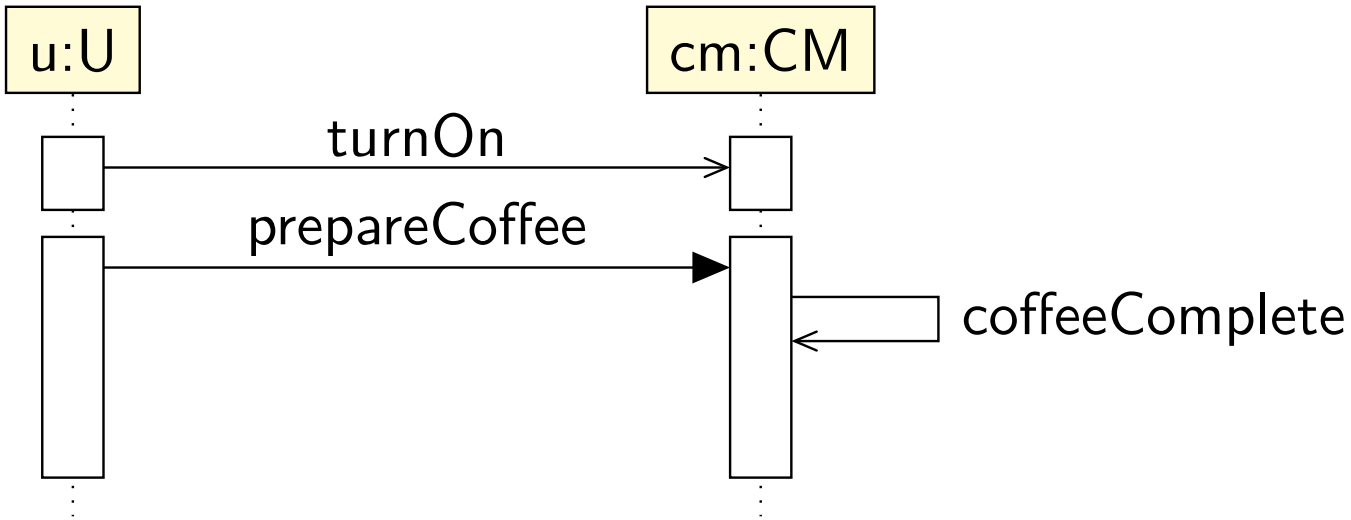

Off
turnOn
Idle
turnOff
Preparing Coffee
prepareCoffee
coffeeComplete
prepareTea
teaComplete
Preparing Tea

Example

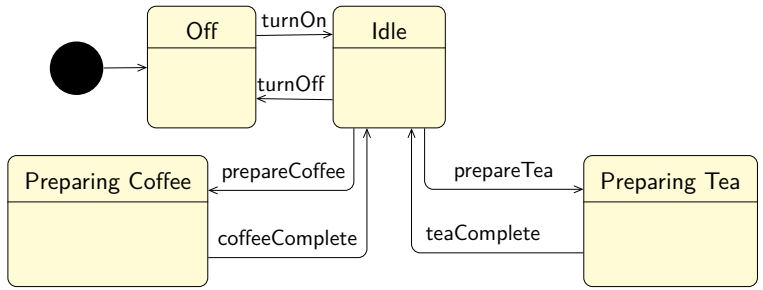

Off
turnOn
Idle
turnOff
Preparing Coffee
prepareCoffee
coffeeComplete
prepareTea
teaComplete
Preparing Tea

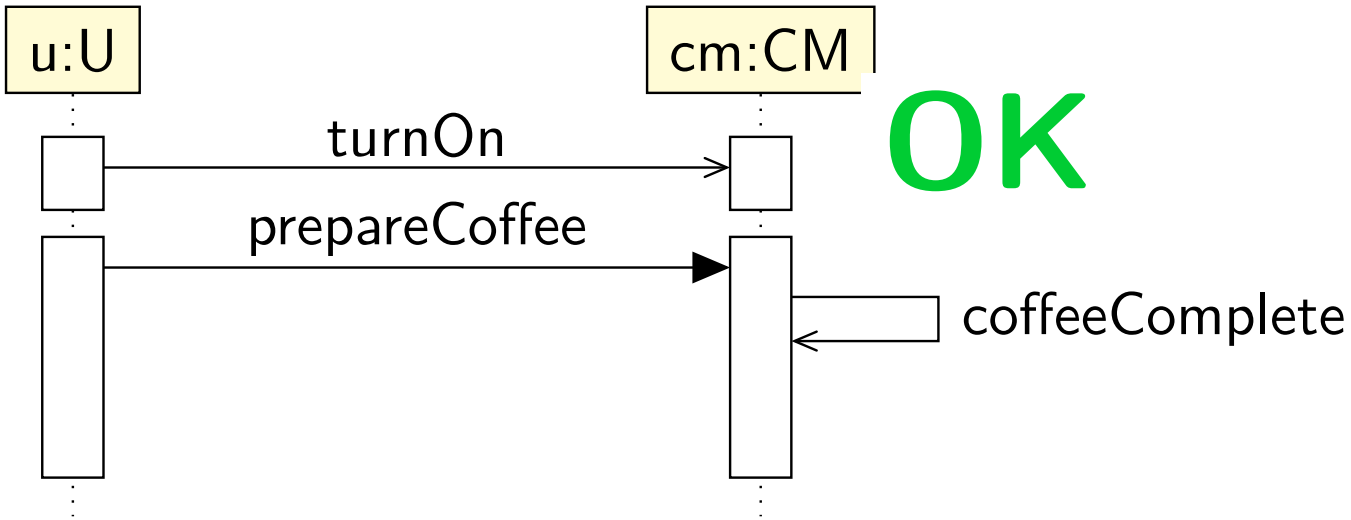
Example


Off
turnOn
Idle
turnOff
Preparing Coffee
prepareCoffee
coffeeComplete
prepareTea
teaComplete
Preparing Tea

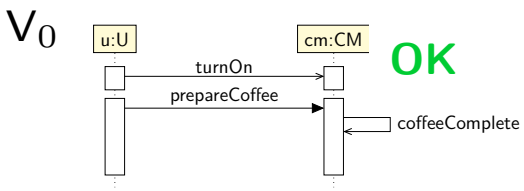

u:U
cm:CM
turnOn
prepareCoffee
coffeeComplete

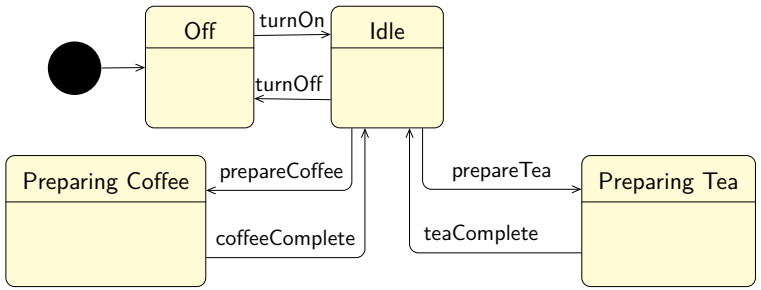
Example


Off
turnOn
Idle
turnOff
Preparing Coffee
prepareCoffee
coffeeComplete
prepareTea
teaComplete
Preparing Tea

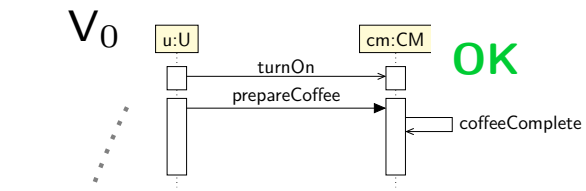

u:U
cm:CM
turnOn
prepareCoffee
coffeeComplete
OK

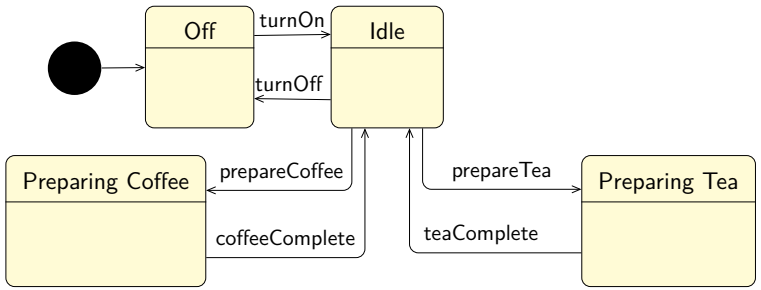
Example

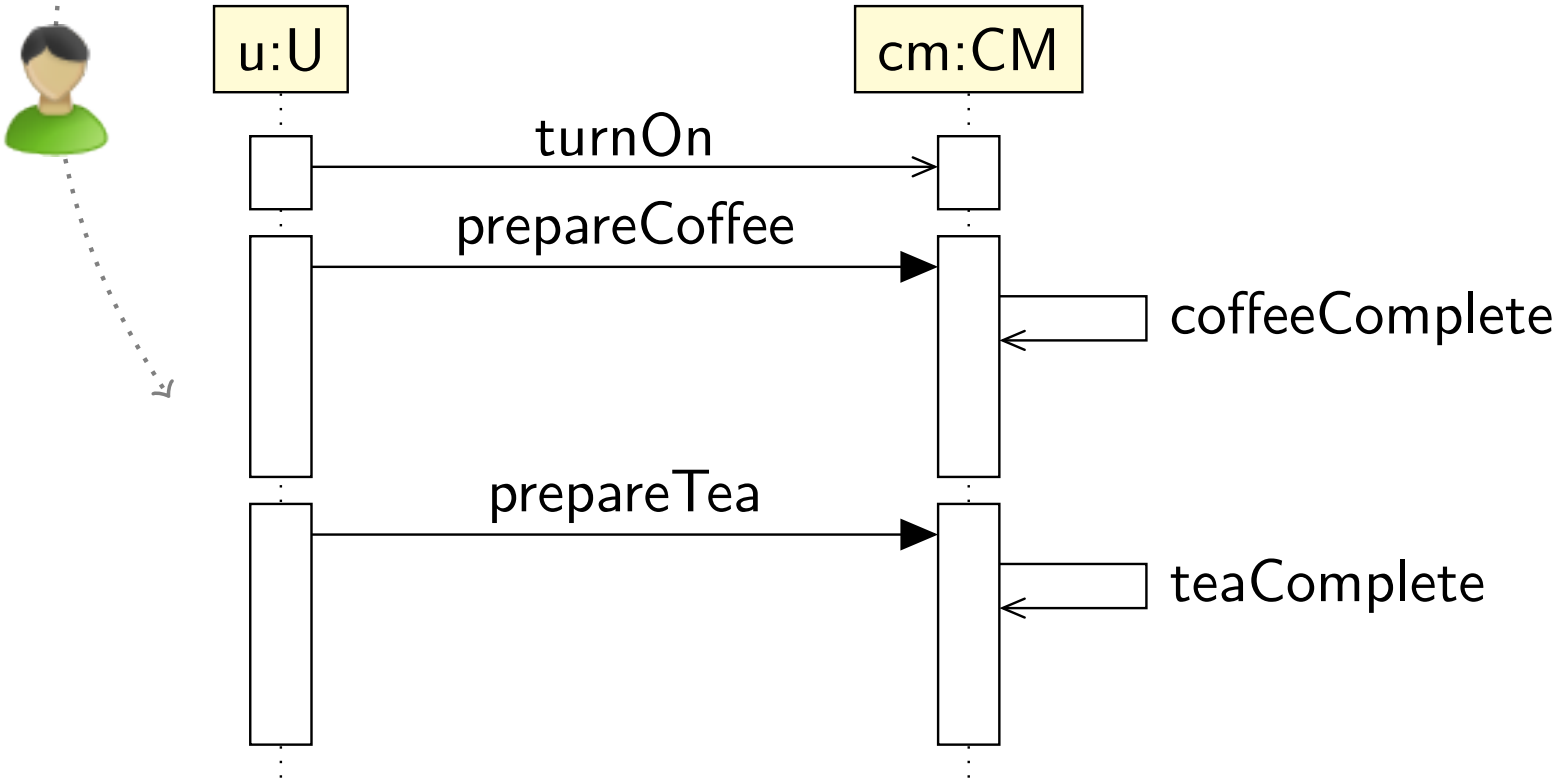

V_0
u:U
cm:CM
OK
turnOn
prepareCoffee
coffeeComplete


Off
turnOn
Idle
turnOff
Preparing Coffee
prepareCoffee
coffeeComplete
prepareTea
teaComplete
Preparing Tea

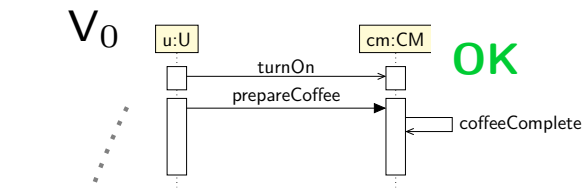
Example

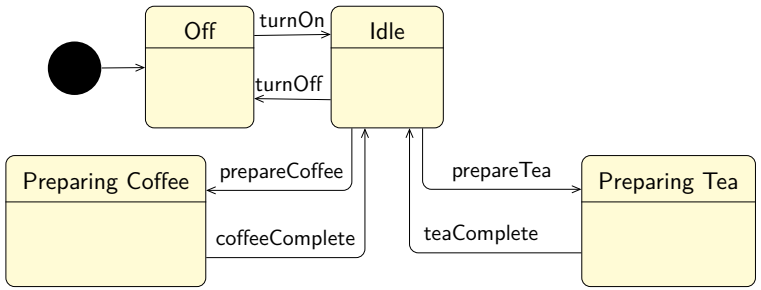

V0
u:U
cm:CM
OK
turnOn
prepareCoffee
coffeeComplete

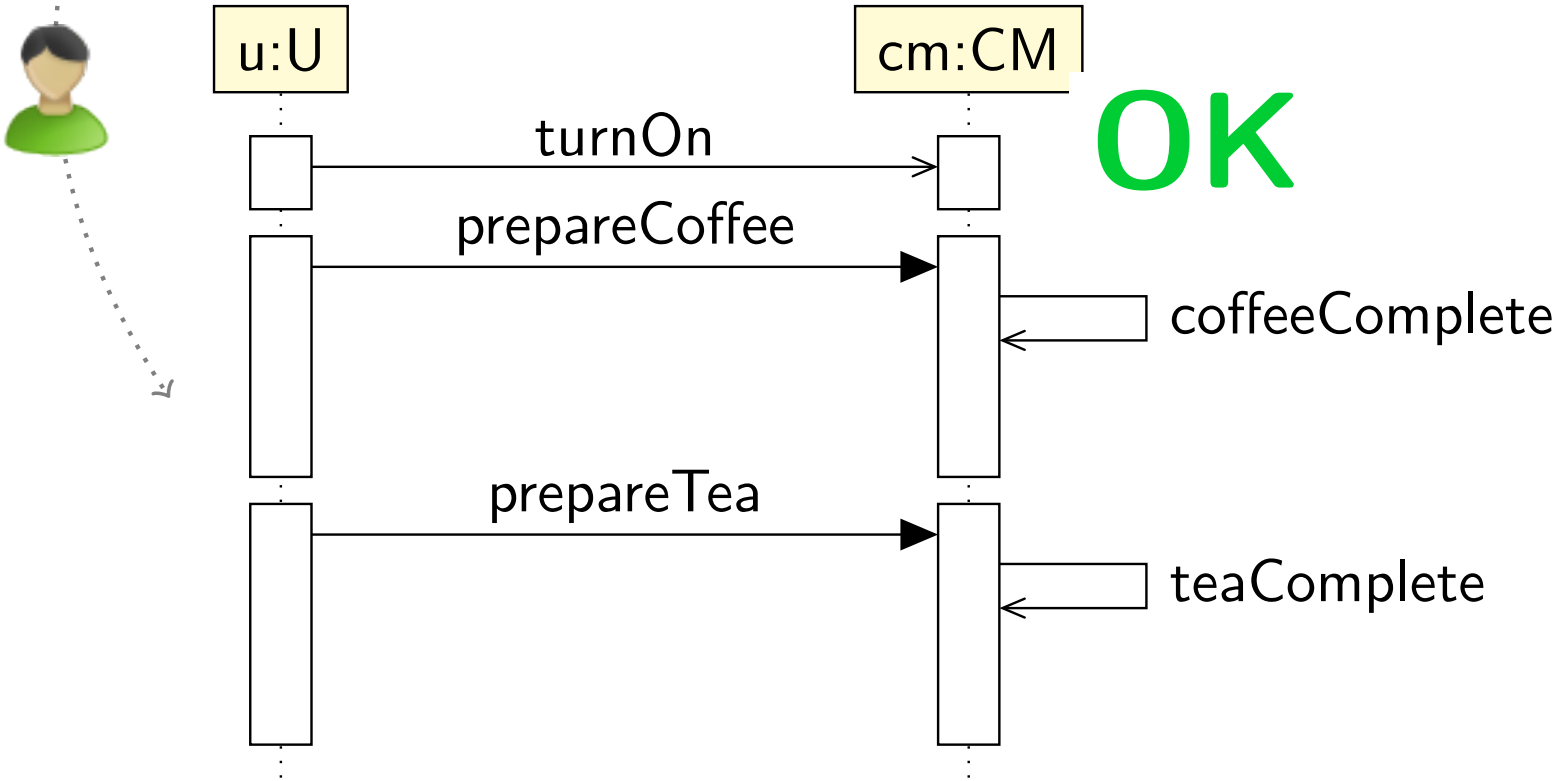

Off
turnOn
Idle
turnOff
Preparing Coffee
prepareCoffee
coffeeComplete
prepareTea
teaComplete
Preparing Tea


u:U
cm:CM
turnOn
prepareCoffee
coffeeComplete
prepareTea
teaComplete

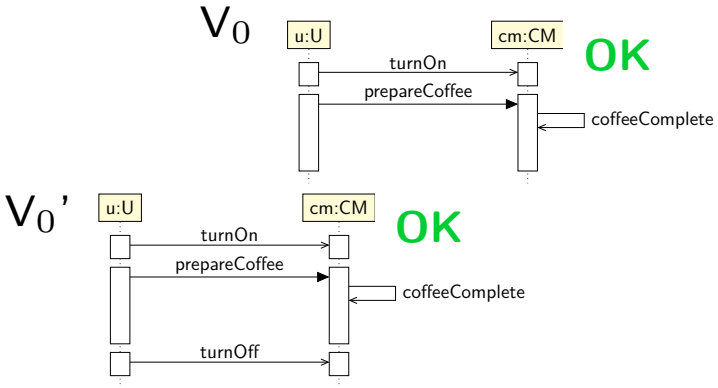
Example

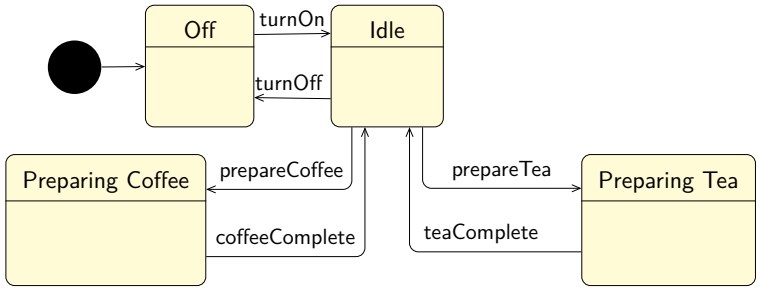

V0
u:U
cm:CM
OK
turnOn
prepareCoffee
coffeeComplete


Off
turnOn
Idle
turnOff
Preparing Coffee
prepareCoffee
coffeeComplete
prepareTea
teaComplete
Preparing Tea

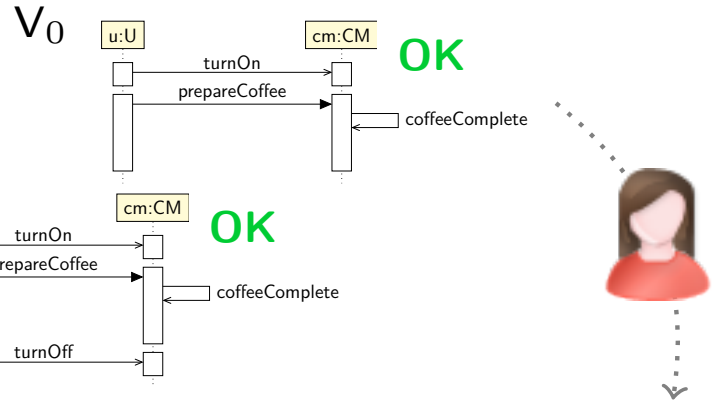

u:U
cm:CM
OK
turnOn
prepareCoffee
coffeeComplete
prepareTea
teaComplete

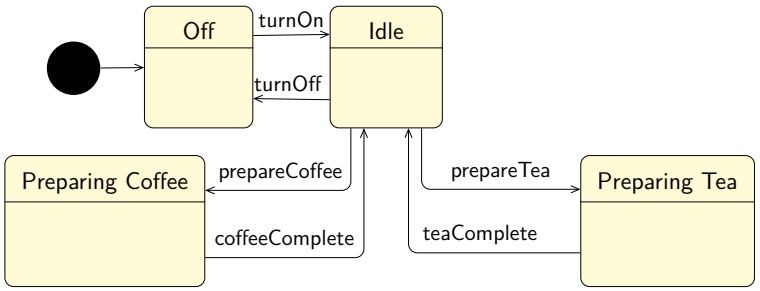
Example

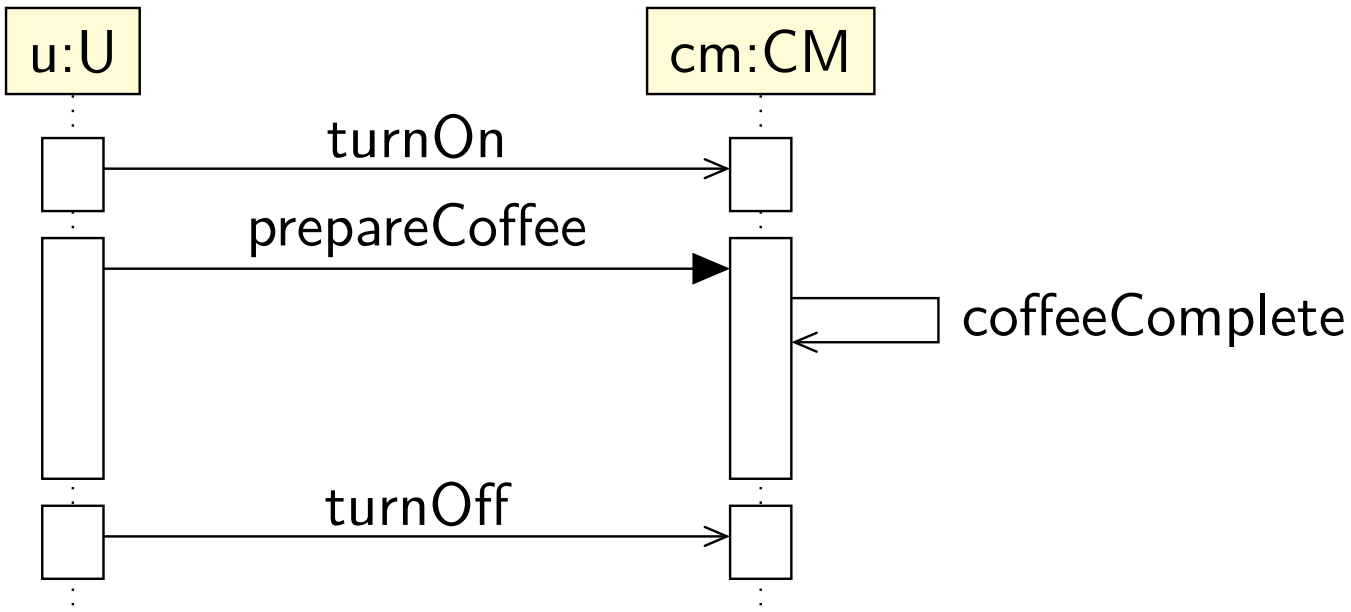

V0
u:U
cm:CM
OK
turnOn
prepareCoffee
coffeeComplete
V0'
u:U
cm:CM
OK
turnOn
prepareCoffee
coffeeComplete
turnOff


Off
turnOn
Idle
turnOff
Preparing Coffee
prepareCoffee
coffeeComplete
prepareTea
teaComplete
Preparing Tea

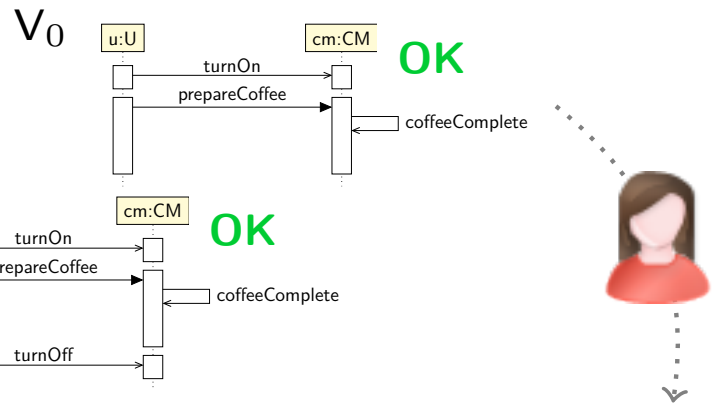
Example

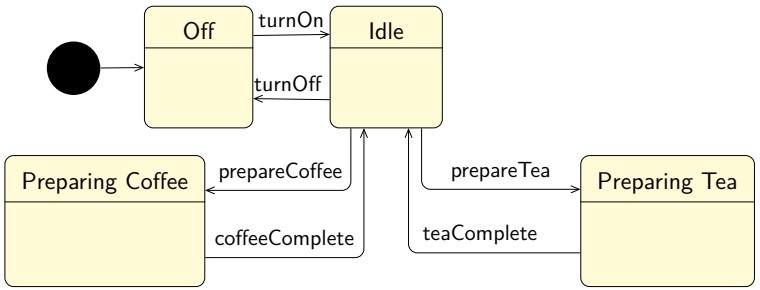

V0
u:U
cm:CM
OK
turnOn
prepareCoffee
coffeeComplete
V0'
u:U
cm:CM
OK
turnOn
prepareCoffee
coffeeComplete
turnOff

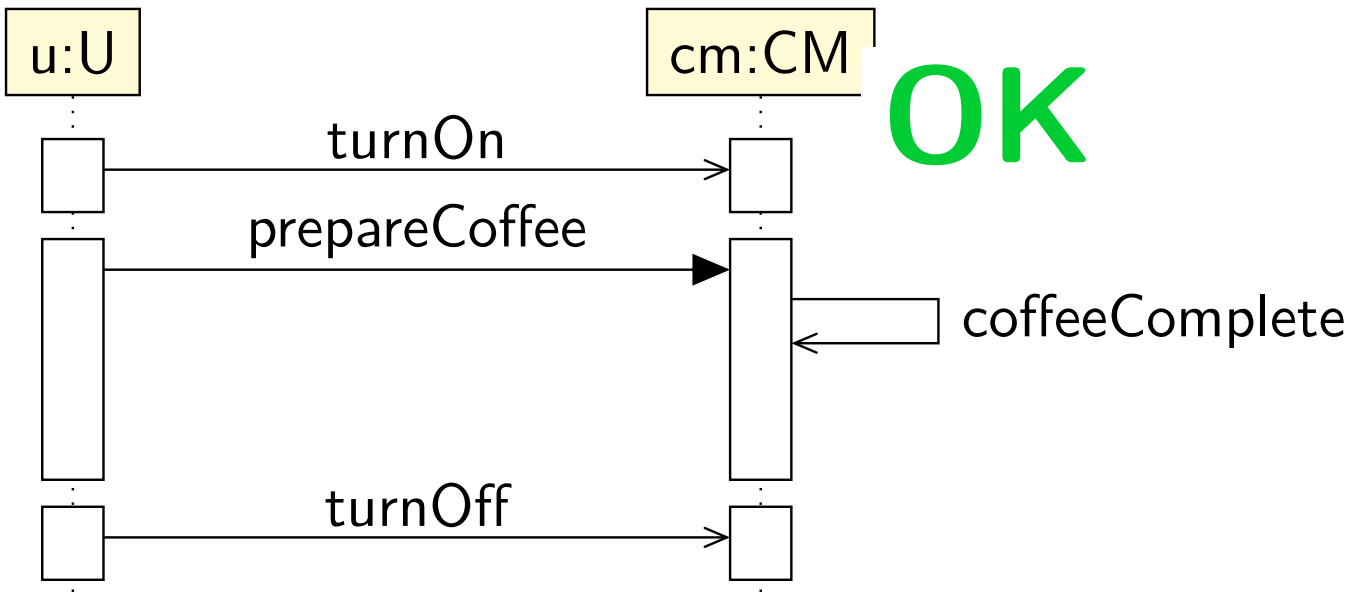

Off
turnOn
Idle
turnOff
Preparing Coffee
prepareCoffee
coffeeComplete
prepareTea
teaComplete
Preparing Tea


u:U
cm:CM
turnOn
prepareCoffee
coffeeComplete
turnOff

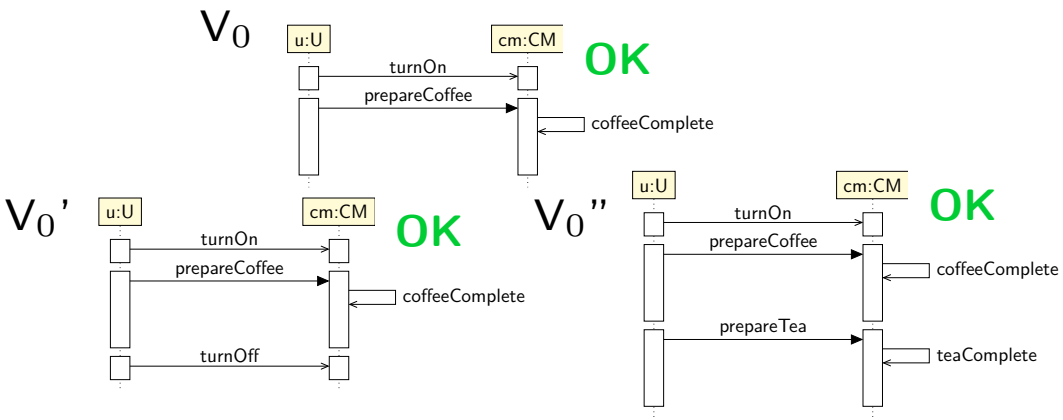
Example

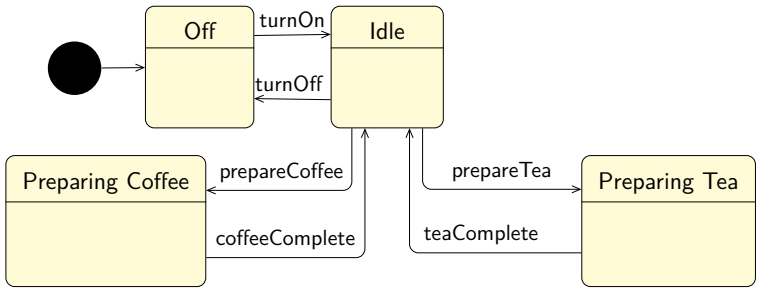

V0
u:U
cm:CM
OK
turnOn
prepareCoffee
coffeeComplete
V0'
u:U
cm:CM
OK
turnOn
prepareCoffee
coffeeComplete
turnOff


Off
turnOn
Idle
turnOff
Preparing Coffee
prepareCoffee
coffeeComplete
prepareTea
teaComplete
Preparing Tea

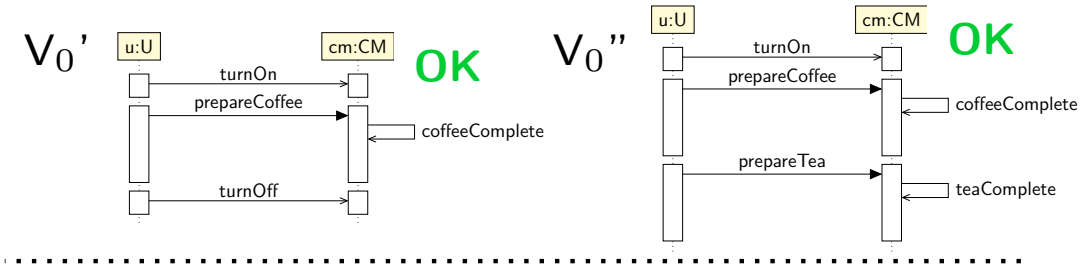

u:U
cm:CM
OK
turnOn
prepareCoffee
coffeeComplete
turnOff

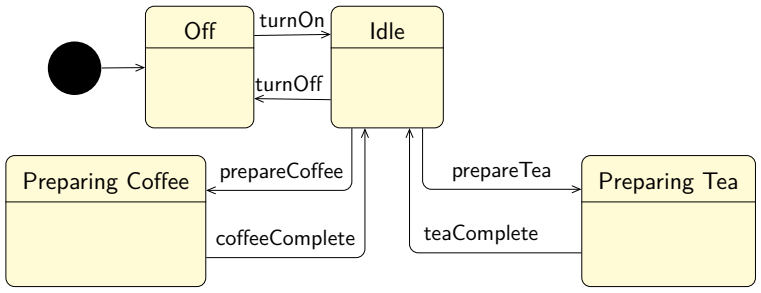
Example


V0
u:U
cm:CM
OK
turnOn
prepareCoffee
coffeeComplete
V0'
u:U
cm:CM
OK
turnOn
prepareCoffee
coffeeComplete
turnOff
V0''
u:U
cm:CM
OK
turnOn
prepareCoffee
coffeeComplete
prepareTea
teaComplete


Off
turnOn
Idle
turnOff
Preparing Coffee
prepareCoffee
coffeeComplete
prepareTea
teaComplete
Preparing Tea

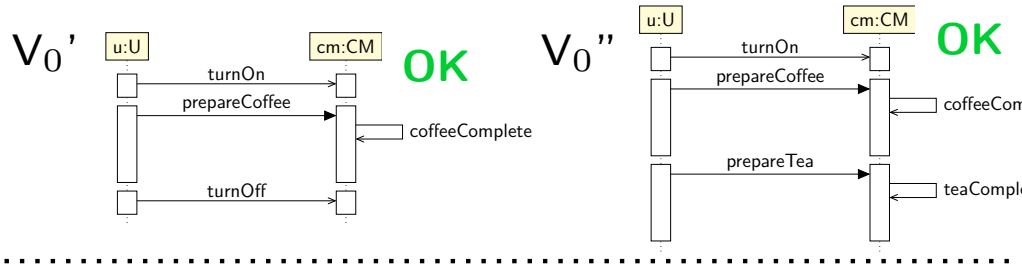
Example

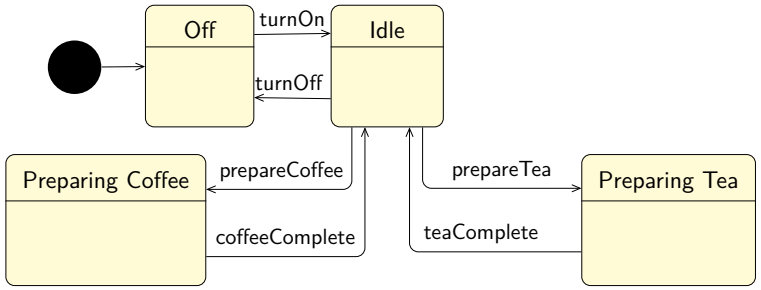

V0'
u:U
cm:CM
turnOn
prepareCoffee
coffeeComplete
turnOff
OK
V0"
u:U
cm:CM
turnOn
prepareCoffee
coffeeComplete
prepareTea
teaComplete
OK


Off
turnOn
Idle
turnOff
Preparing Coffee
prepareCoffee
coffeeComplete
prepareTea
teaComplete
Preparing Tea

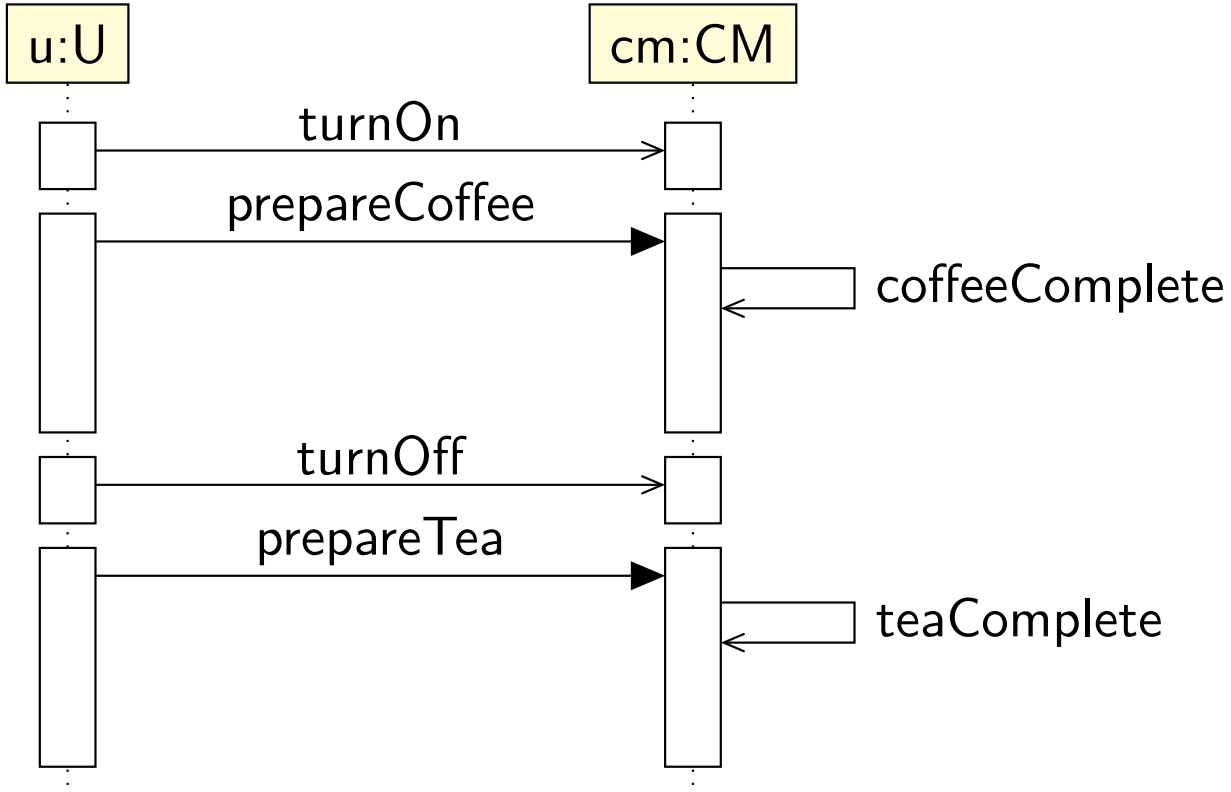
V_1?

Example

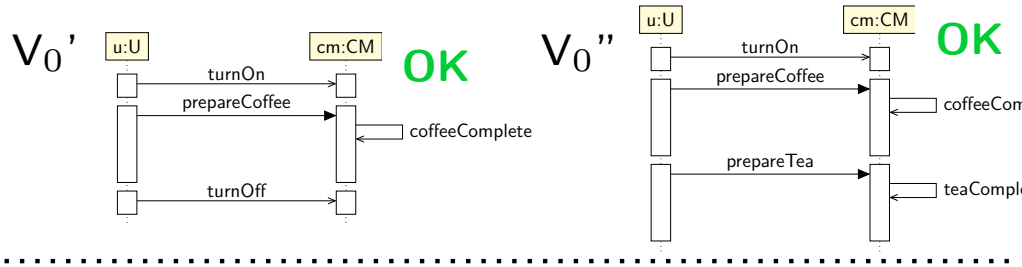

V0'
u:U
cm:CM
turnOn
prepareCoffee
coffeeComplete
turnOff
OK
V0''
u:U
cm:CM
turnOn
prepareCoffee
coffeeComplete
prepareTea
teaComplete
OK

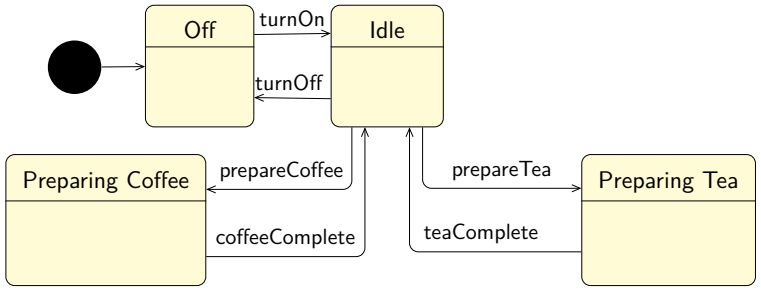

Off
turnOn
Idle
turnOff
Preparing Coffee
prepareCoffee
coffeeComplete
prepareTea
teaComplete
Preparing Tea

V1?

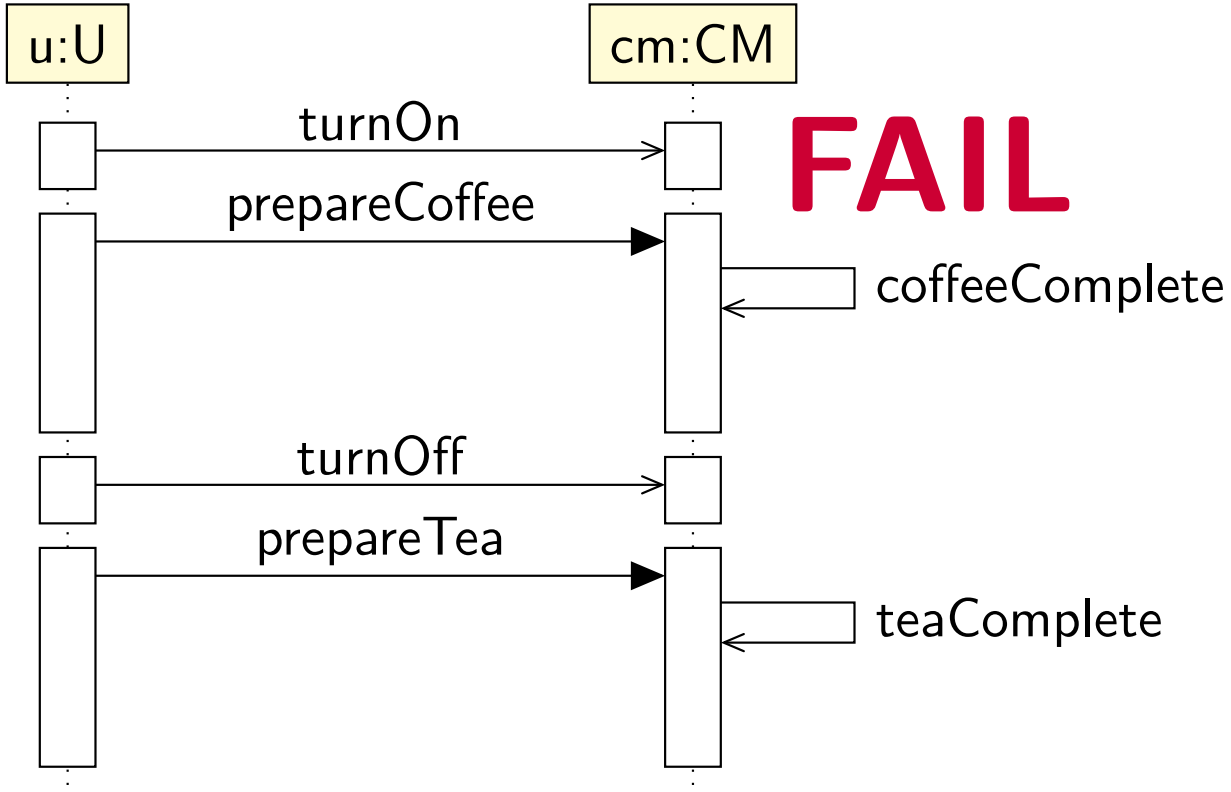

u:U
cm:CM
turnOn
prepareCoffee
coffeeComplete
turnOff
prepareTea
teaComplete

Example

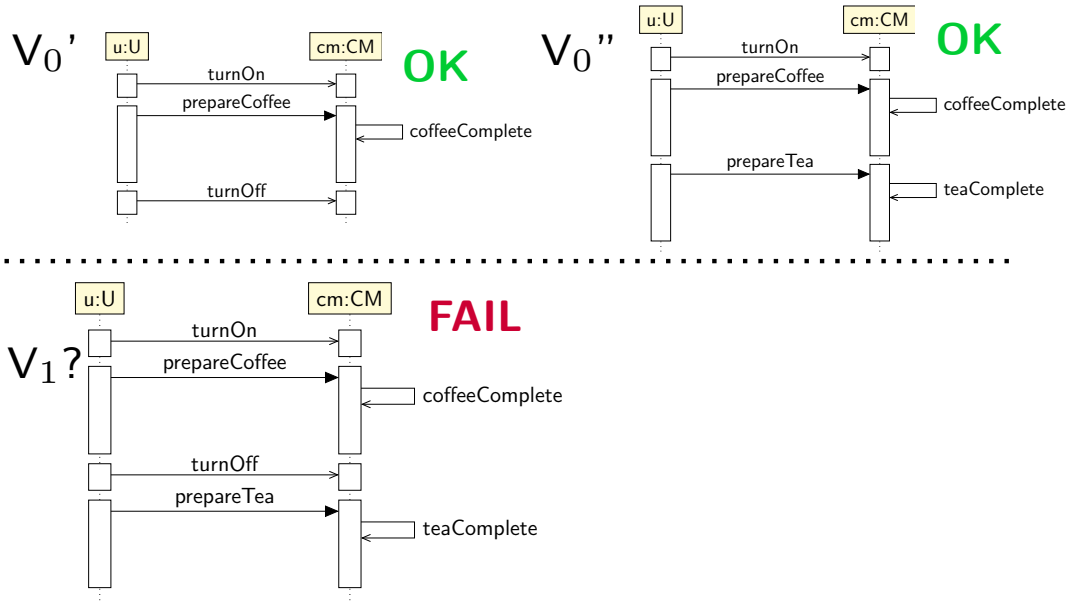

V0'
u:U
cm:CM
turnOn
prepareCoffee
coffeeComplete
turnOff
OK
V0''
u:U
cm:CM
turnOn
prepareCoffee
coffeeComplete
prepareTea
teaComplete
OK

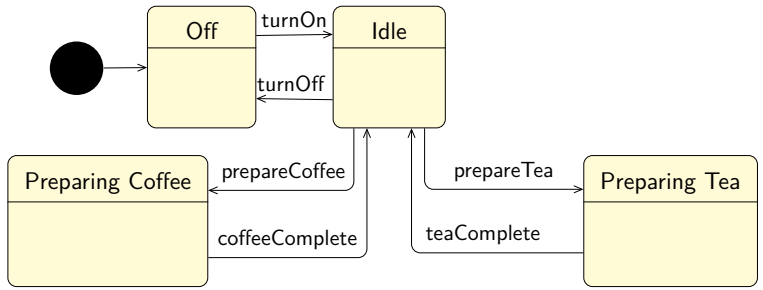

Off
turnOn
Idle
turnOff
Preparing Coffee
prepareCoffee
coffeeComplete
prepareTea
teaComplete
Preparing Tea

V1?

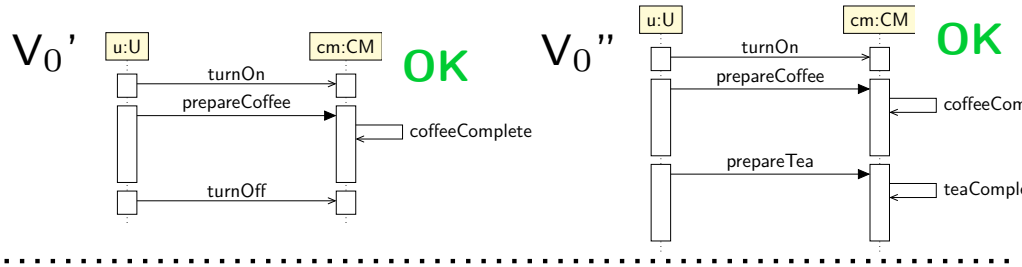

u:U
cm:CM
turnOn
prepareCoffee
coffeeComplete
turnOff
prepareTea
teaComplete
FAIL

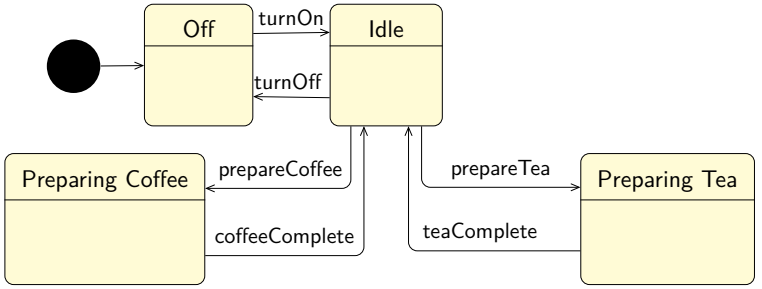
Example

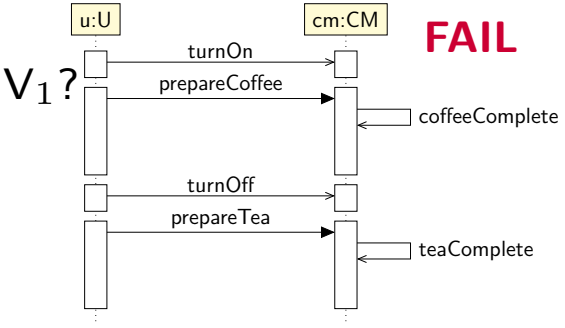

V0'
u:U
cm:CM
OK
turnOn
prepareCoffee
coffeeComplete
turnOff
V0''
u:U
cm:CM
OK
turnOn
prepareCoffee
coffeeComplete
prepareTea
teaComplete
u:U
cm:CM
FAIL
V1?
turnOn
prepareCoffee
coffeeComplete
turnOff
prepareTea
teaComplete

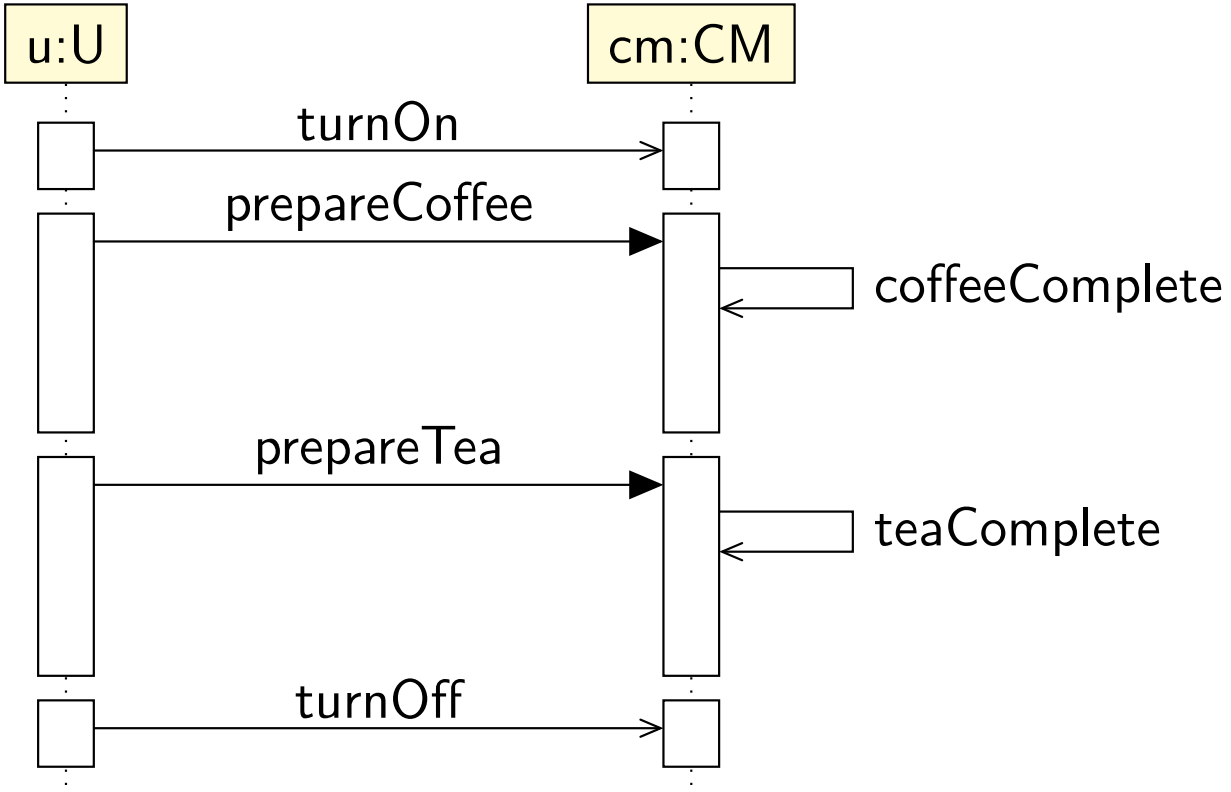

Off
turnOn
Idle
turnOff
Preparing Coffee
prepareCoffee
coffeeComplete
prepareTea
teaComplete
Preparing Tea

Example

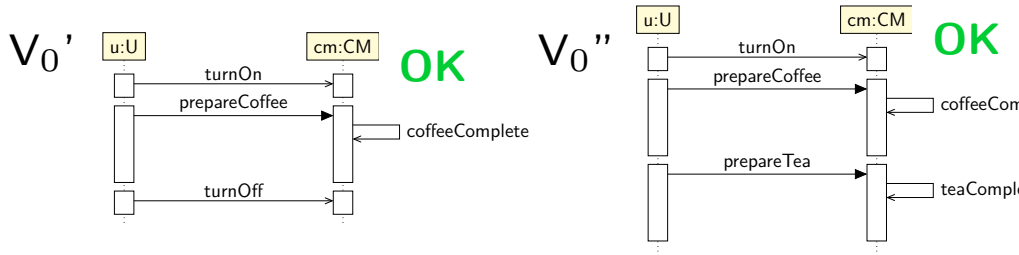

V0'
u:U
cm:CM
turnOn
prepareCoffee
coffeeComplete
turnOff
OK
V0''
u:U
cm:CM
turnOn
prepareCoffee
coffeeComplete
prepareTea
teaComplete
OK


Off
turnOn
Idle
turnOff
Preparing Coffee
prepareCoffee
coffeeComplete
prepareTea
teaComplete
Preparing Tea

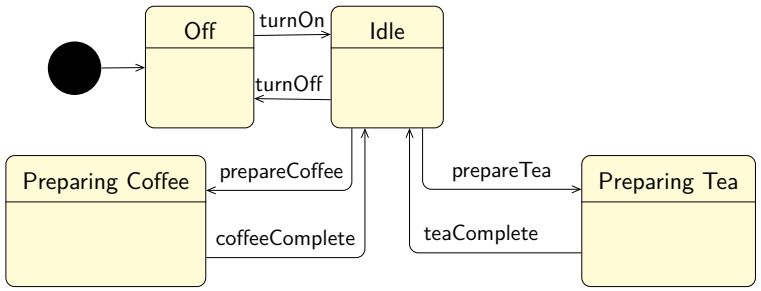

u:U
cm:CM
V1?
turnOn
prepareCoffee
coffeeComplete
turnOff
prepareTea
teaComplete
FAIL

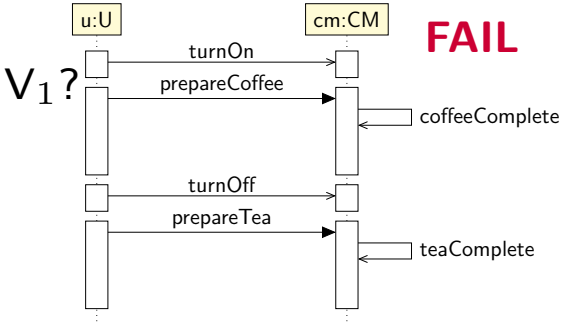

u:U
cm:CM
turnOn
prepareCoffee
coffeeComplete
prepareTea
teaComplete
turnOff

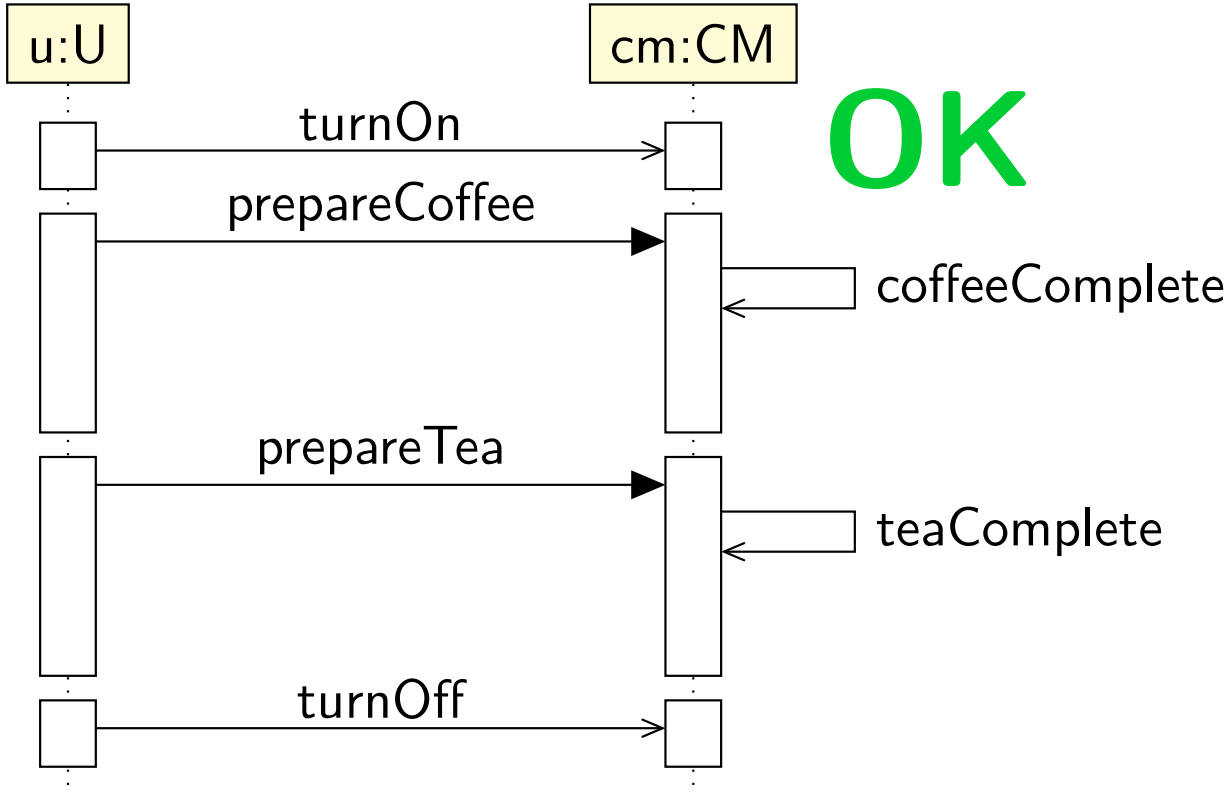
Example


V0'
u:U
cm:CM
OK
turnOn
prepareCoffee
coffeeComplete
turnOff

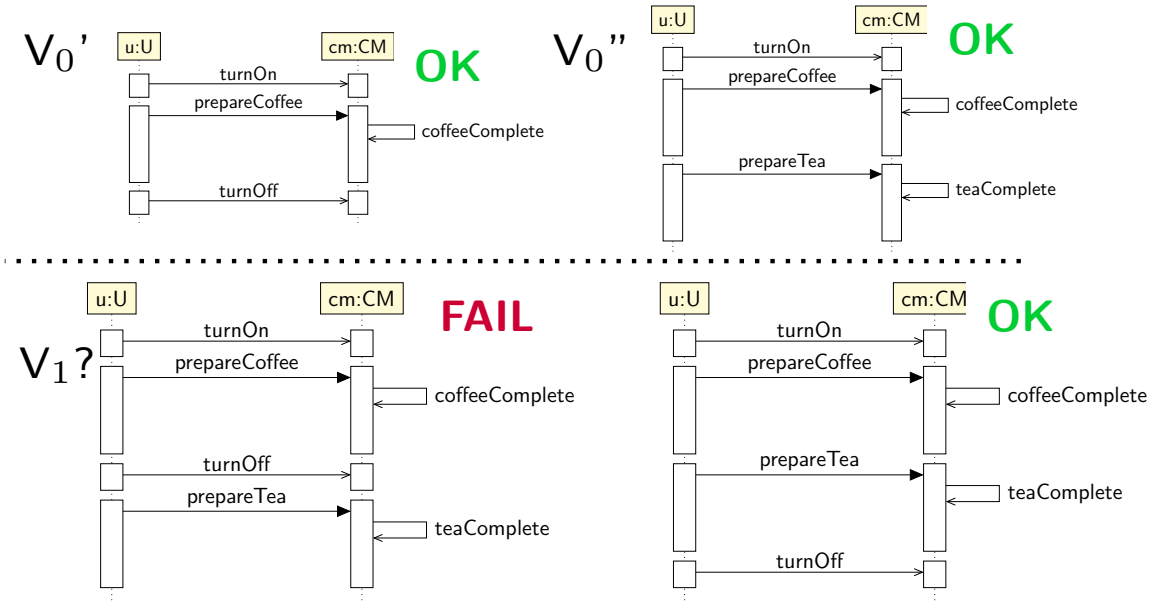
V0''
u:U
cm:CM
OK
turnOn
prepareCoffee
coffeeComplete
prepareTea
teaComplete

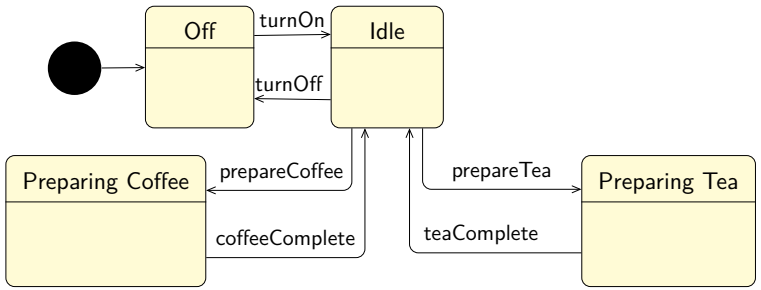

Off
turnOn
Idle
turnOff
Preparing Coffee
prepareCoffee
coffeeComplete
prepareTea
teaComplete
Preparing Tea


V1?
u:U
cm:CM
FAIL
turnOn
prepareCoffee
coffeeComplete
turnOff
prepareTea
teaComplete

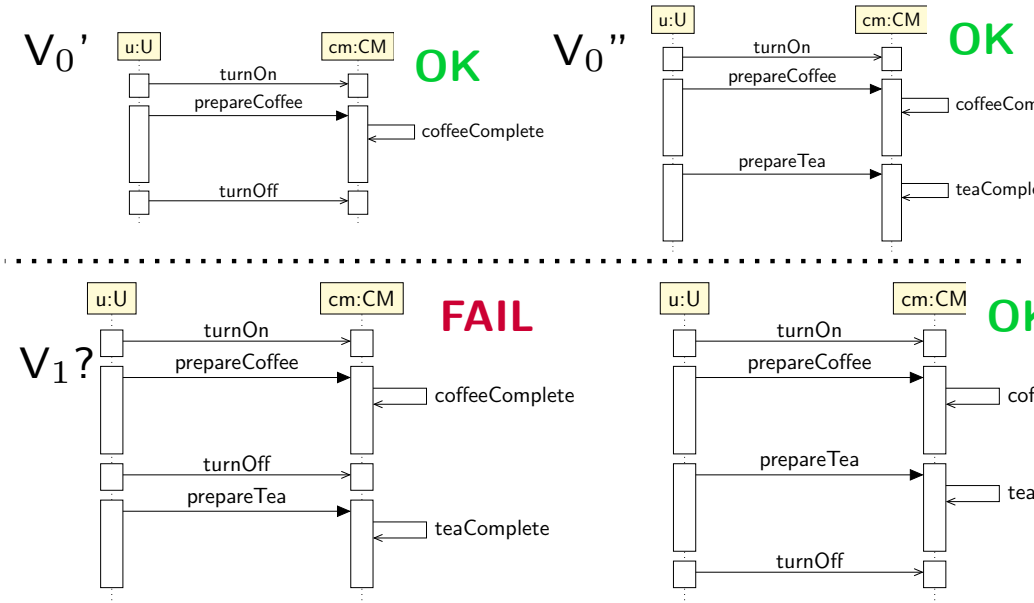

u:U
cm:CM
OK
turnOn
prepareCoffee
coffeeComplete
prepareTea
teaComplete
turnOff

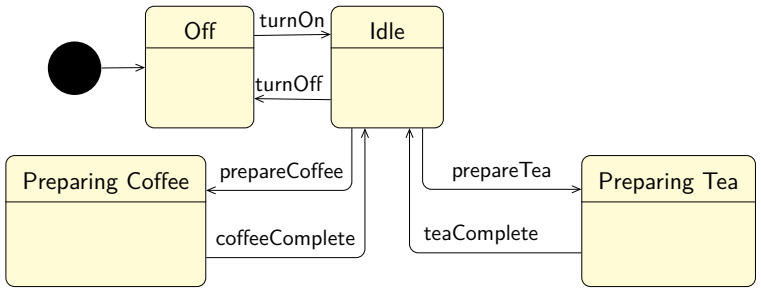
Example


V0'
u:U
cm:CM
OK
turnOn
prepareCoffee
coffeeComplete
turnOff
V0"
u:U
cm:CM
OK
turnOn
prepareCoffee
coffeeComplete
prepareTea
teaComplete
V1?
u:U
cm:CM
FAIL
turnOn
prepareCoffee
coffeeComplete
turnOff
prepareTea
teaComplete
u:U
cm:CM
OK
turnOn
prepareCoffee
coffeeComplete
prepareTea
teaComplete
turnOff


Off
turnOn
Idle
turnOff
Preparing Coffee
prepareCoffee
coffeeComplete
prepareTea
teaComplete
Preparing Tea

Example





Problem

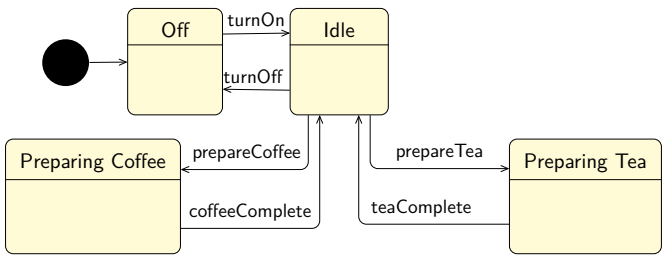
Many syntactically correct merges possible.

But how to avoid inconsistency with state machine?

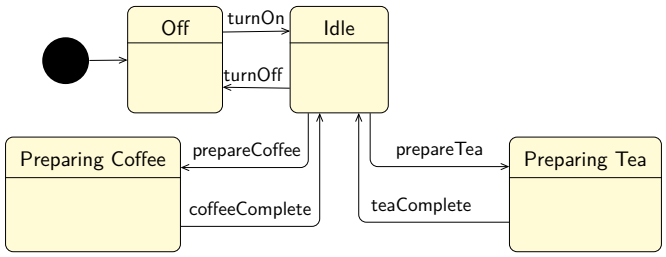
Related Work

- Model Versioning
  - *Brosch et al.* Model Versioning Survey
  - *Brosch et al.* Adaptable Model Versioning
- Model Verification
  - *Cabot et al.* UML/OCL operation contracts
  - HUGO: State machines and collaborations to SPIN, UML 1.x
  - CHARMY: Software architecture models to SPIN

Model Checking

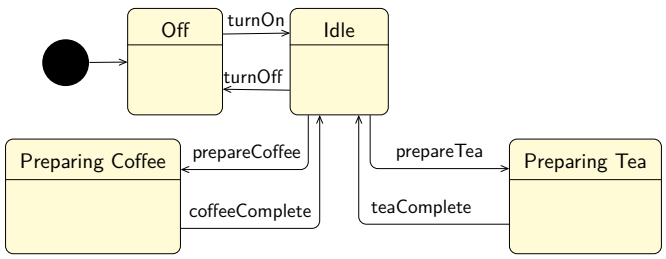


Model Checking

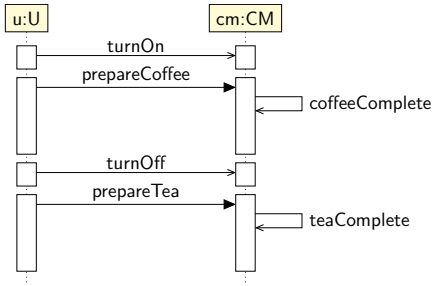


implements

Model Checking

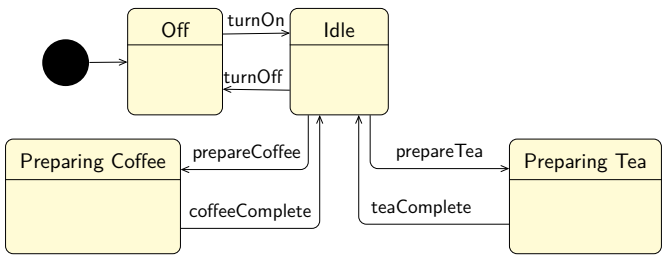


implements

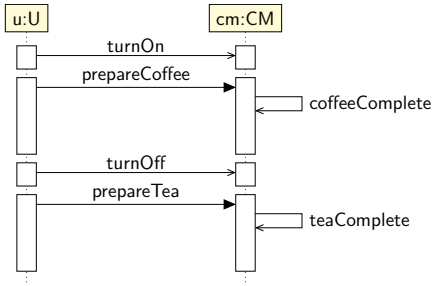


?

Model Checking



implements

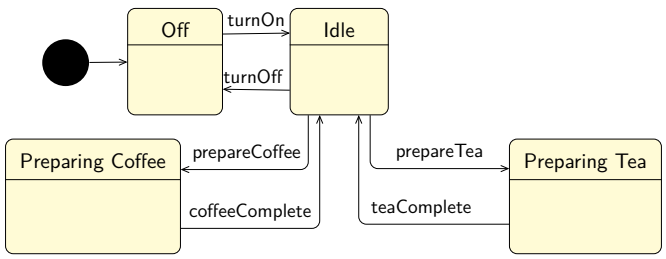


?

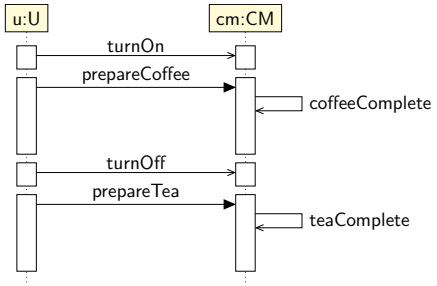
$\mathcal{K}$

Kripke structure

Model Checking



implements

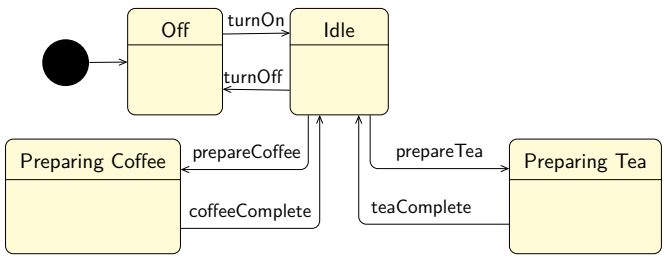


?

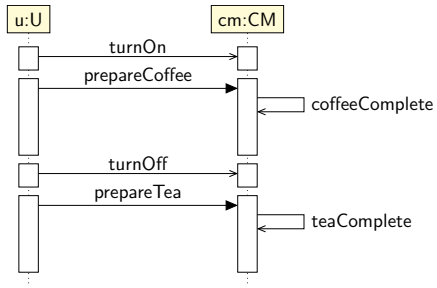
$\mathcal{K}$ $\models$

Kripke structure

Model Checking



implements



?

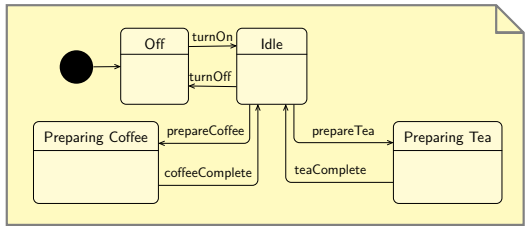
$\mathcal{K}$ $\models$ ϕ

Kripke structure

Temporal logic

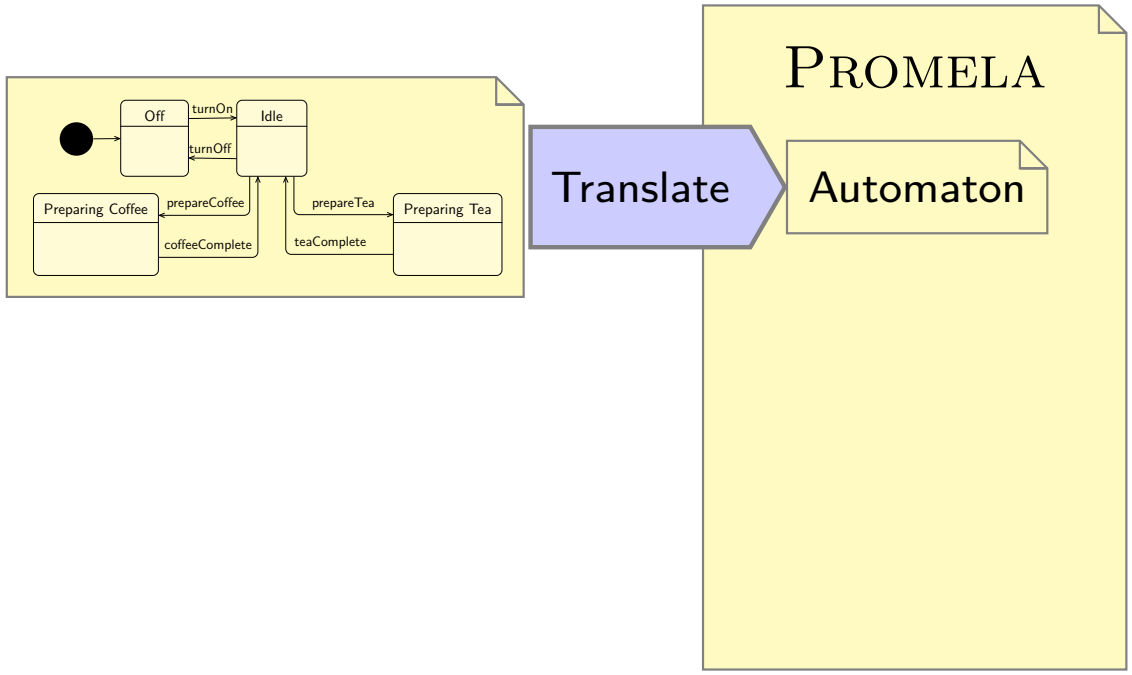
Semantics-aware model versioning

Using the SPIN model checker



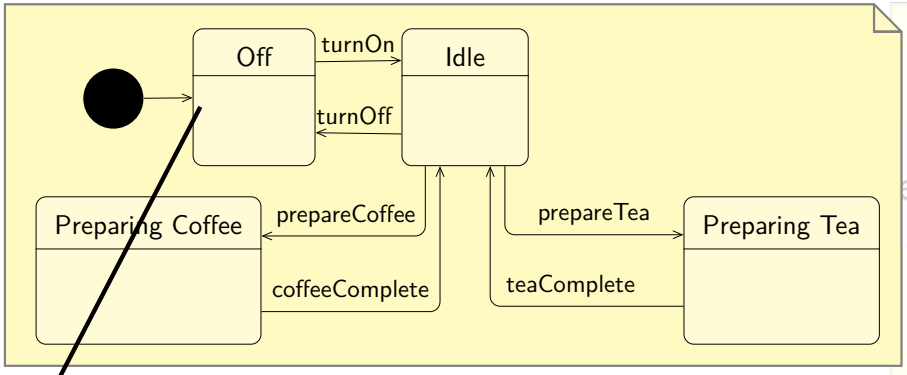
Semantics-aware model versioning

Using the SPIN model checker



Semantics-aware model versioning

Using the SPIN model checker



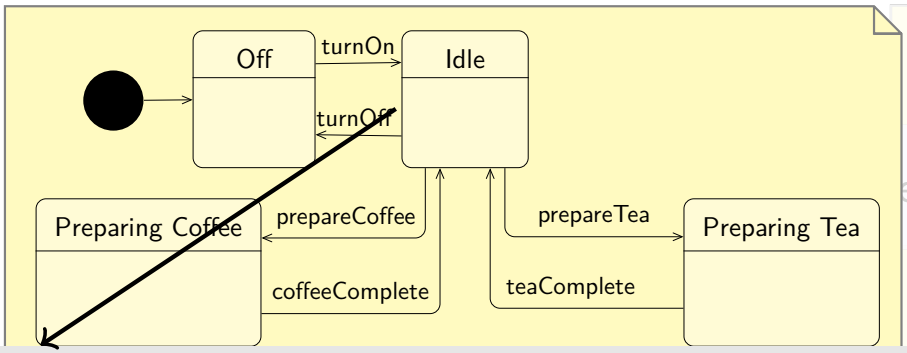
PROMELA

Automaton

```
 Off:
printf("Off", CM[h]);
if
::  CM[h] == turnOn -> h++; goto Idle
::  CM[h] == acc -> goto end
fi;
```

Semantics-aware model versioning

Using the SPIN model checker



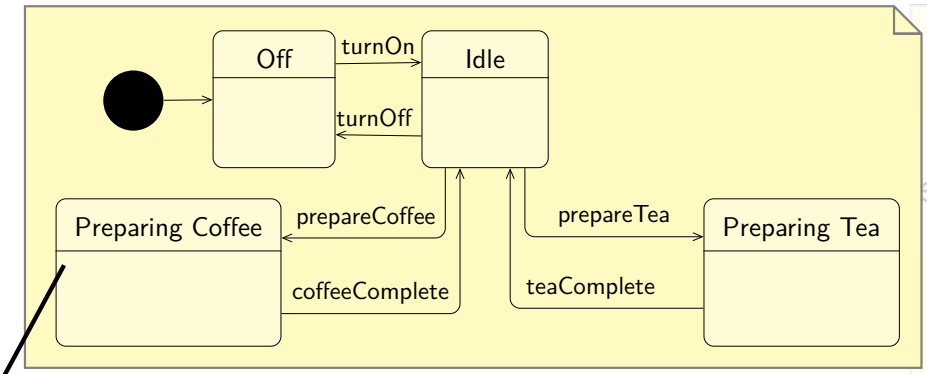
PROMELA

Automaton

```
 Idle:
printf("Idle", CM[h]);
if
::  CM[h] == prepareCoffee -> h++; goto PrepareCoffee
::  CM[h] == prepareTea -> h++; goto PrepareTea
::  CM[h] == turnOff -> h++; goto Off
::  CM[h] == acc -> goto end
fi;
```


Semantics-aware model versioning

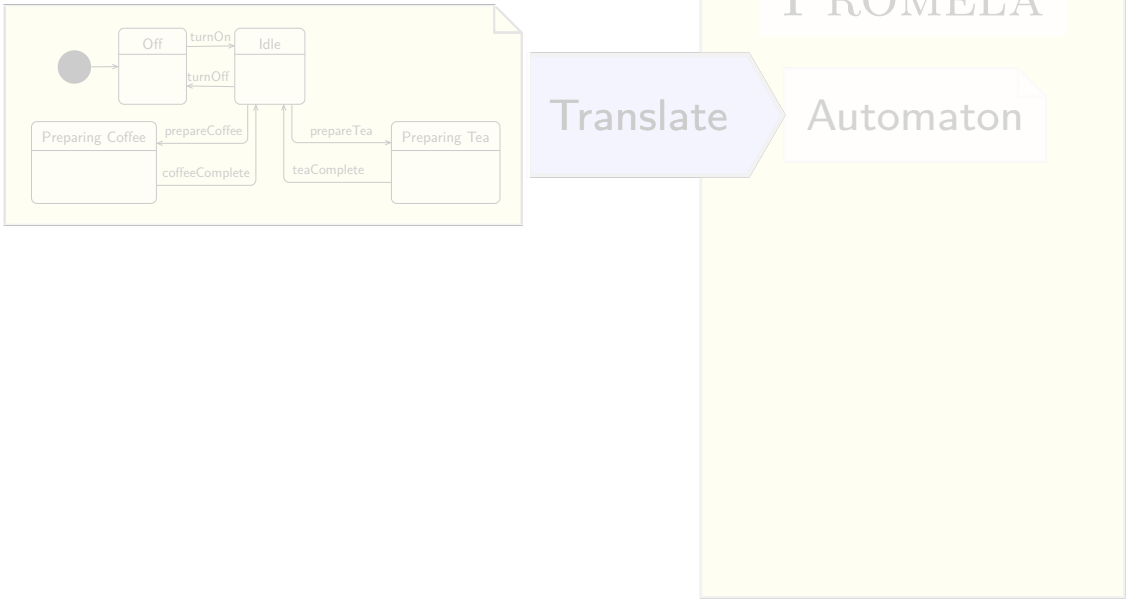
Using the SPIN model checker

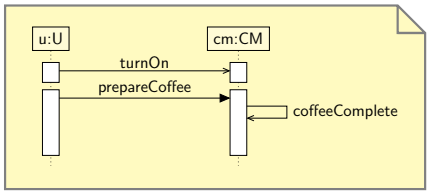



```
 PreparingCoffee:
printf("PrepareCoffee", CM[h]);
if
::  CM[h] == coffeeComplete -> h++; goto Idle
::  CM[h] == acc -> goto end
fi;
```

Semantics-aware model versioning

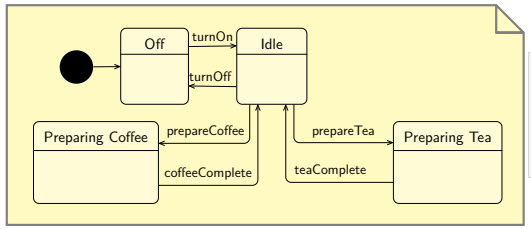
Using the SPIN model checker





Semantics-aware model versioning

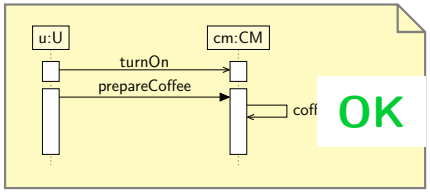
Using the SPIN model checker



Translate

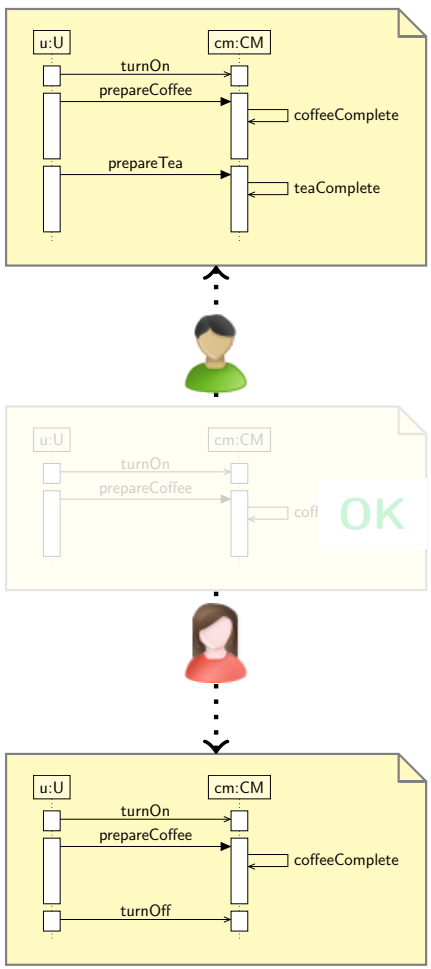
PROMELA

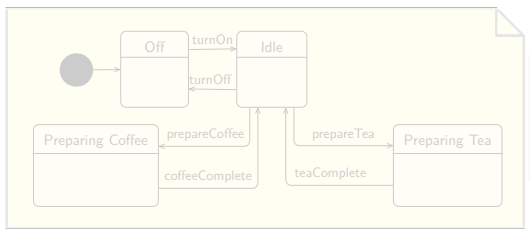
Automaton



Semantics-aware model versioning

Using the SPIN model checker





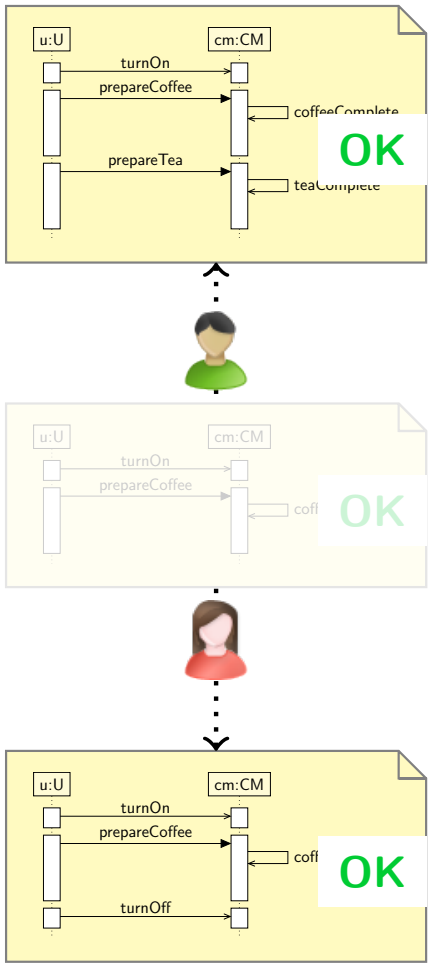
Translate

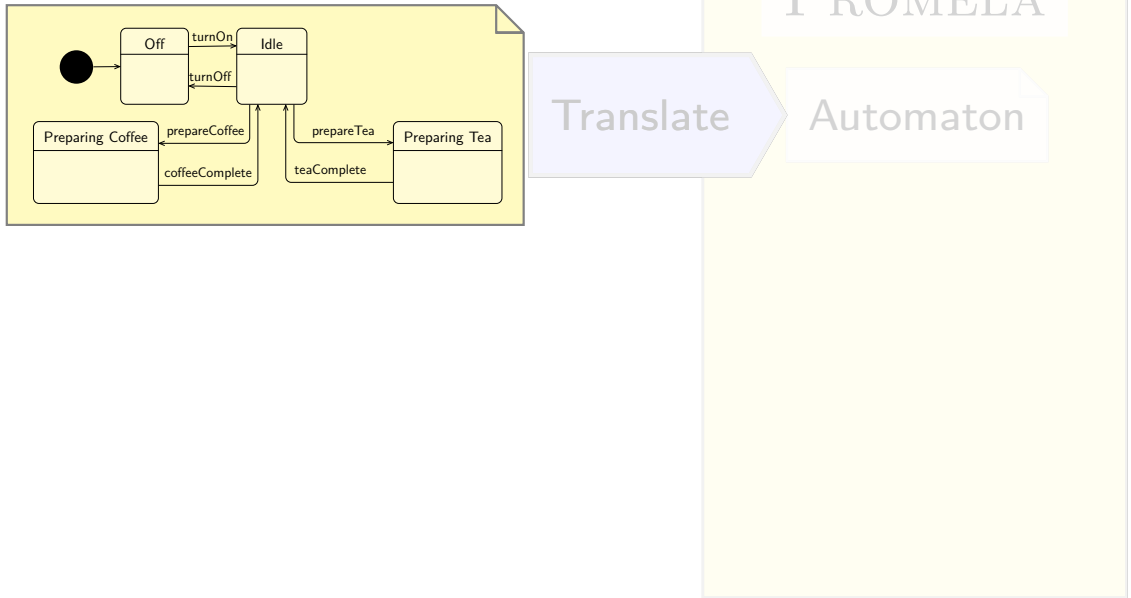
PROMELA

Automaton

Semantics-aware model versioning

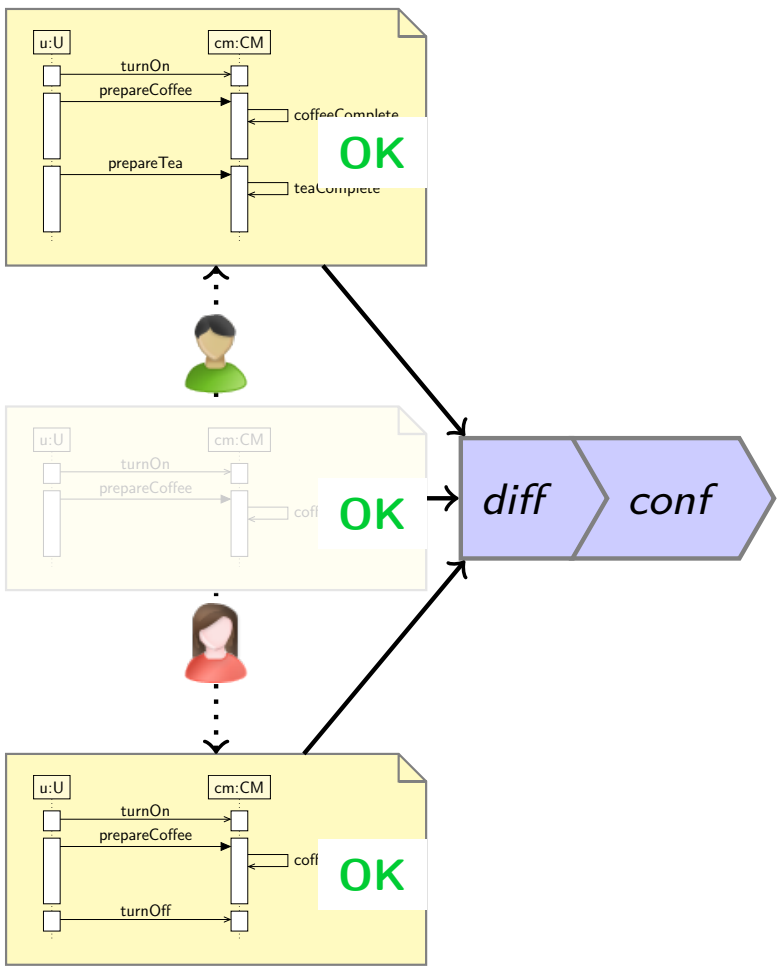
Using the SPIN model checker

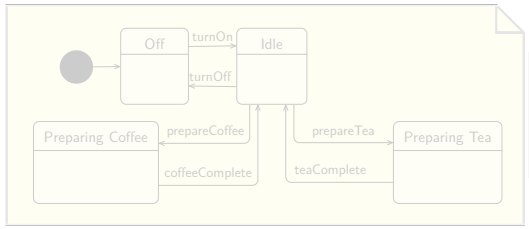




Semantics-aware model versioning

Using the SPIN model checker





Translate

PROMELA

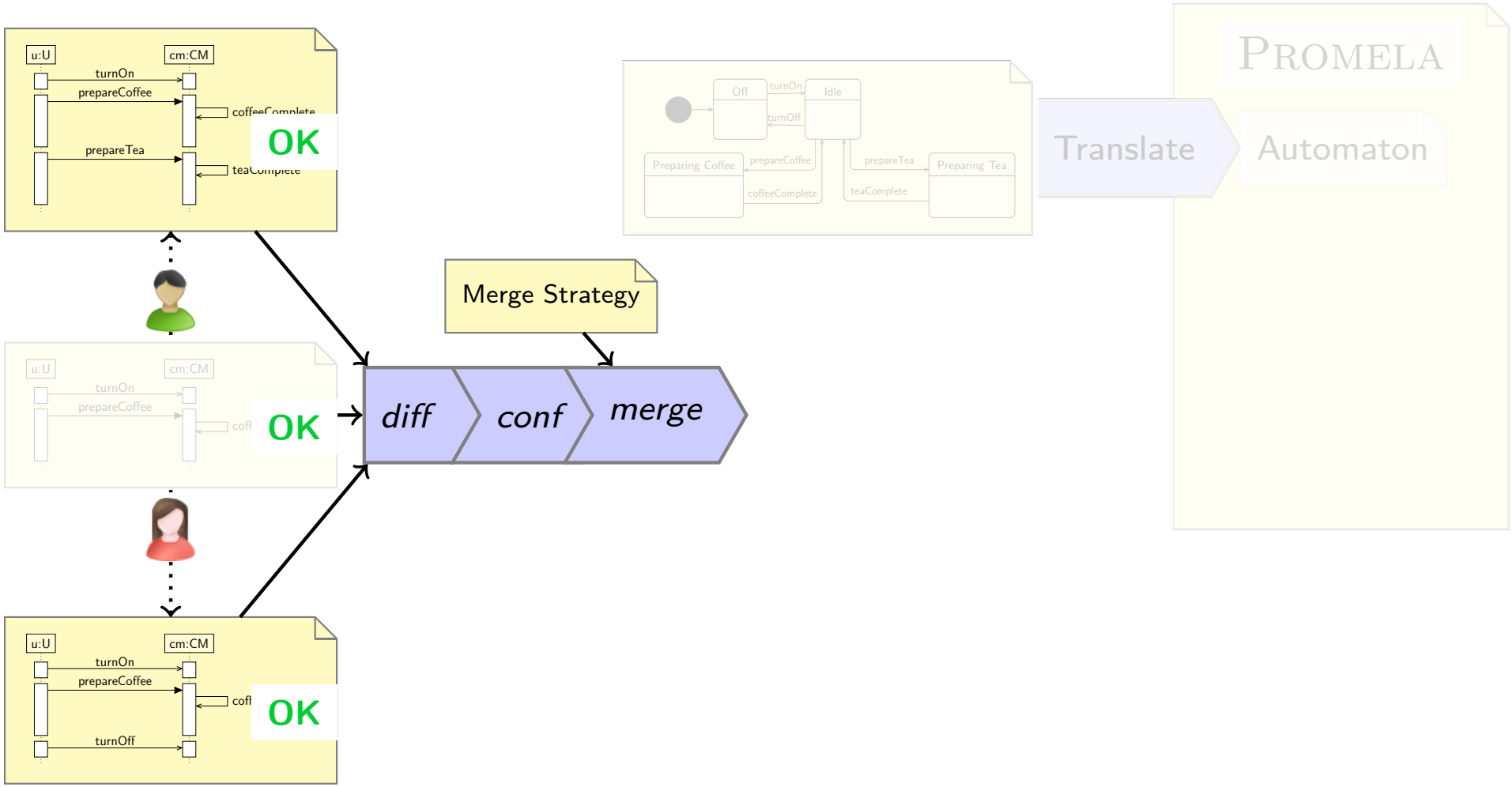
Automaton

diff

conf

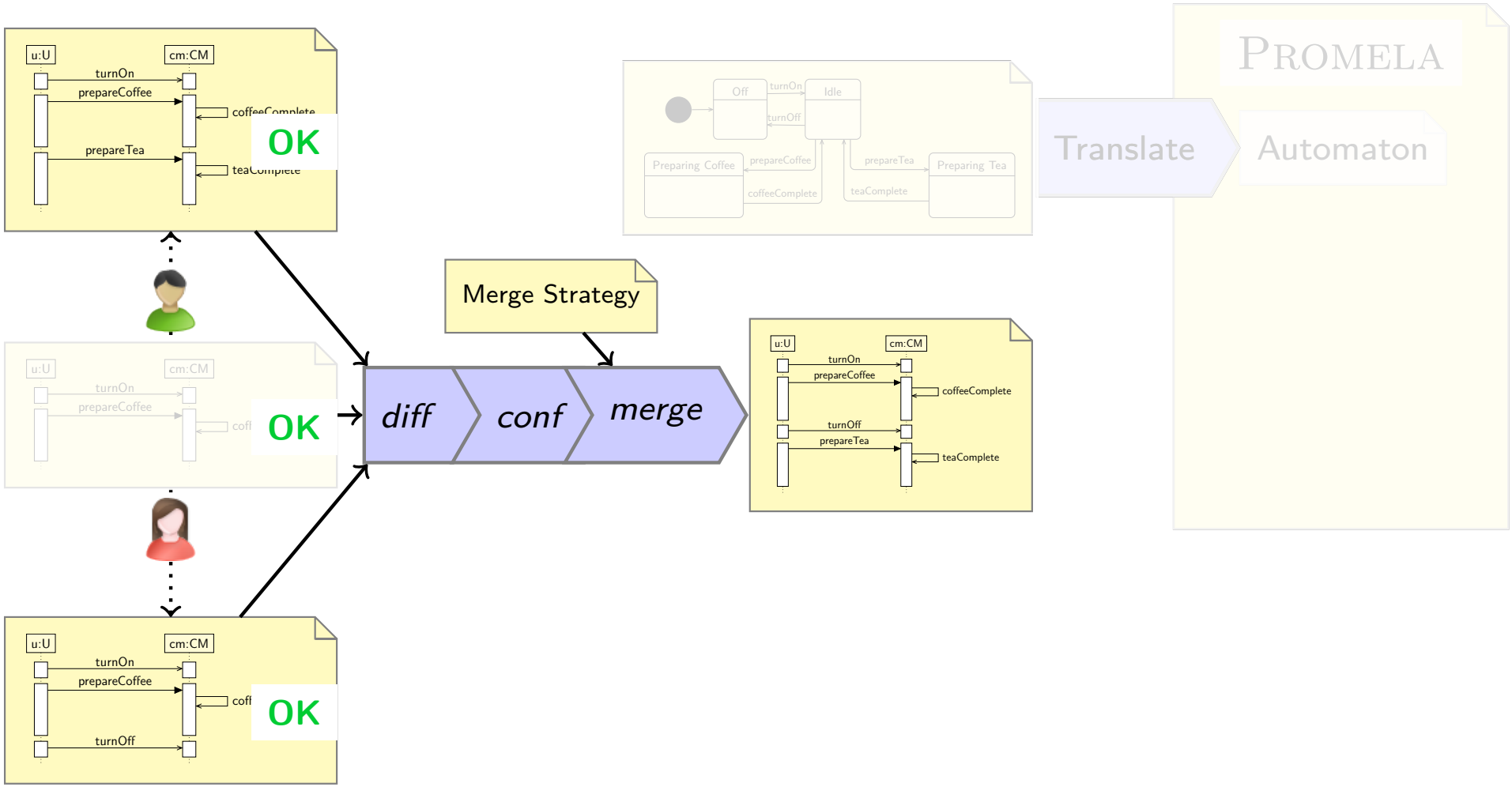
Semantics-aware model versioning

Using the SPIN model checker



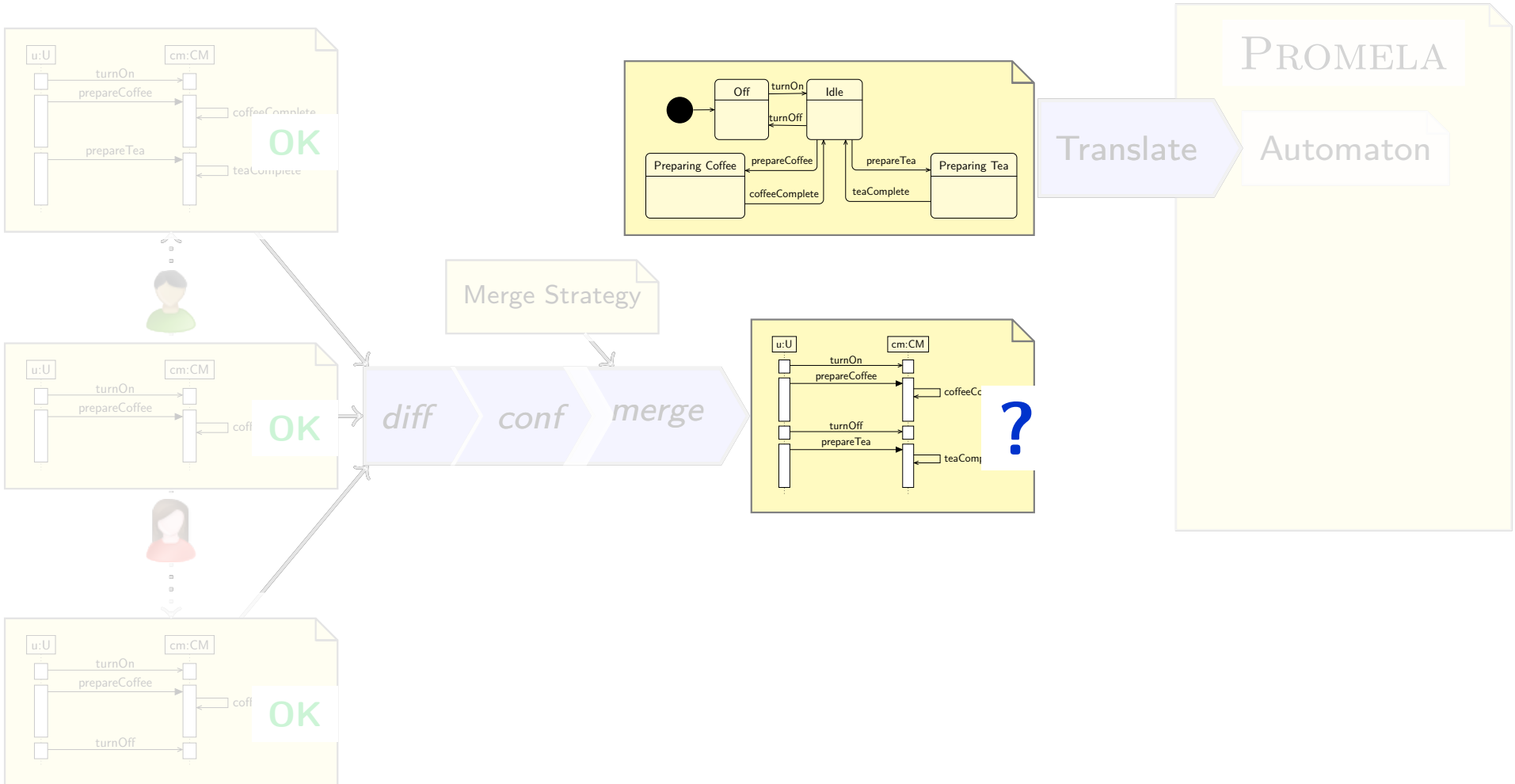
Semantics-aware model versioning

Using the SPIN model checker



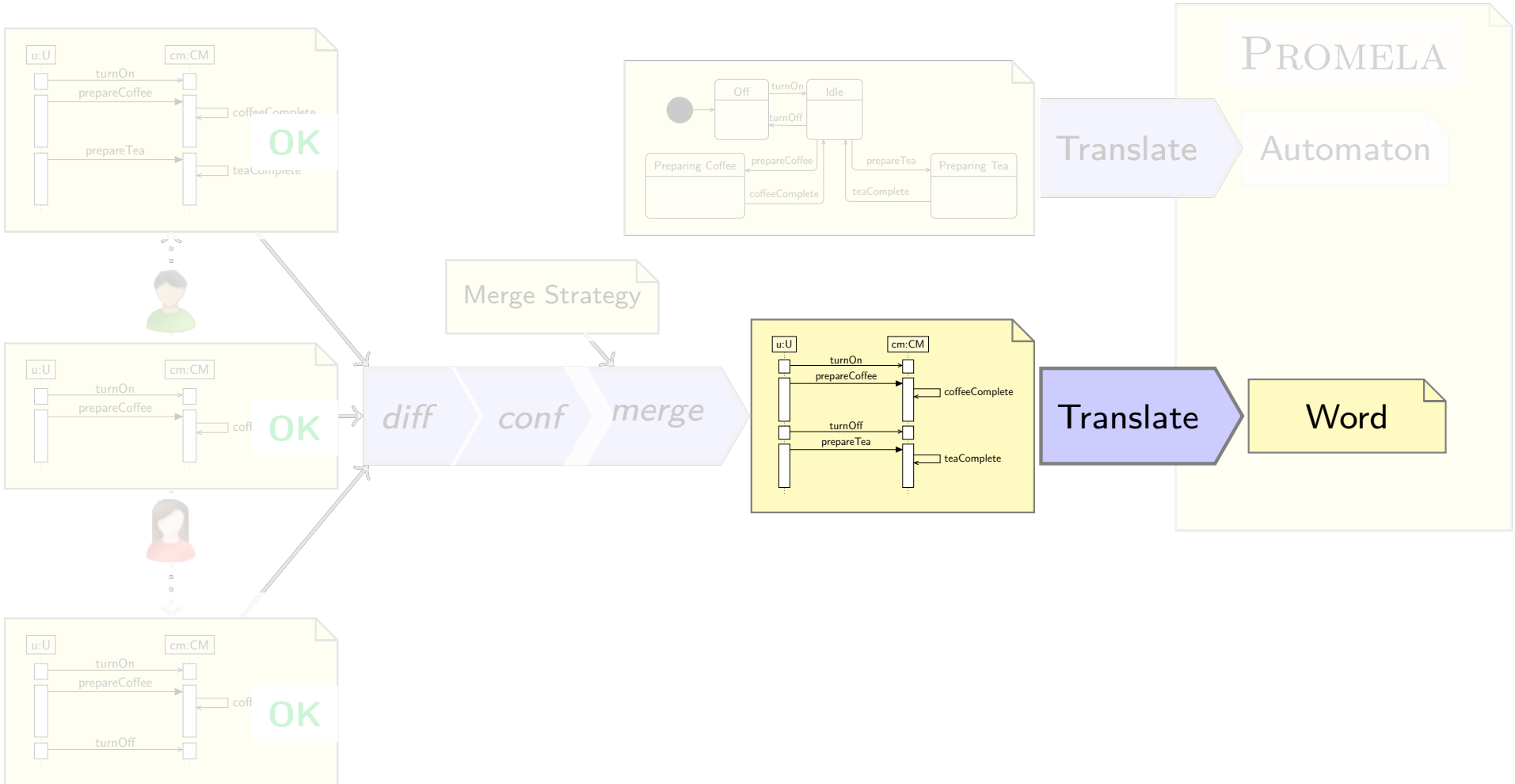
Semantics-aware model versioning

Using the SPIN model checker



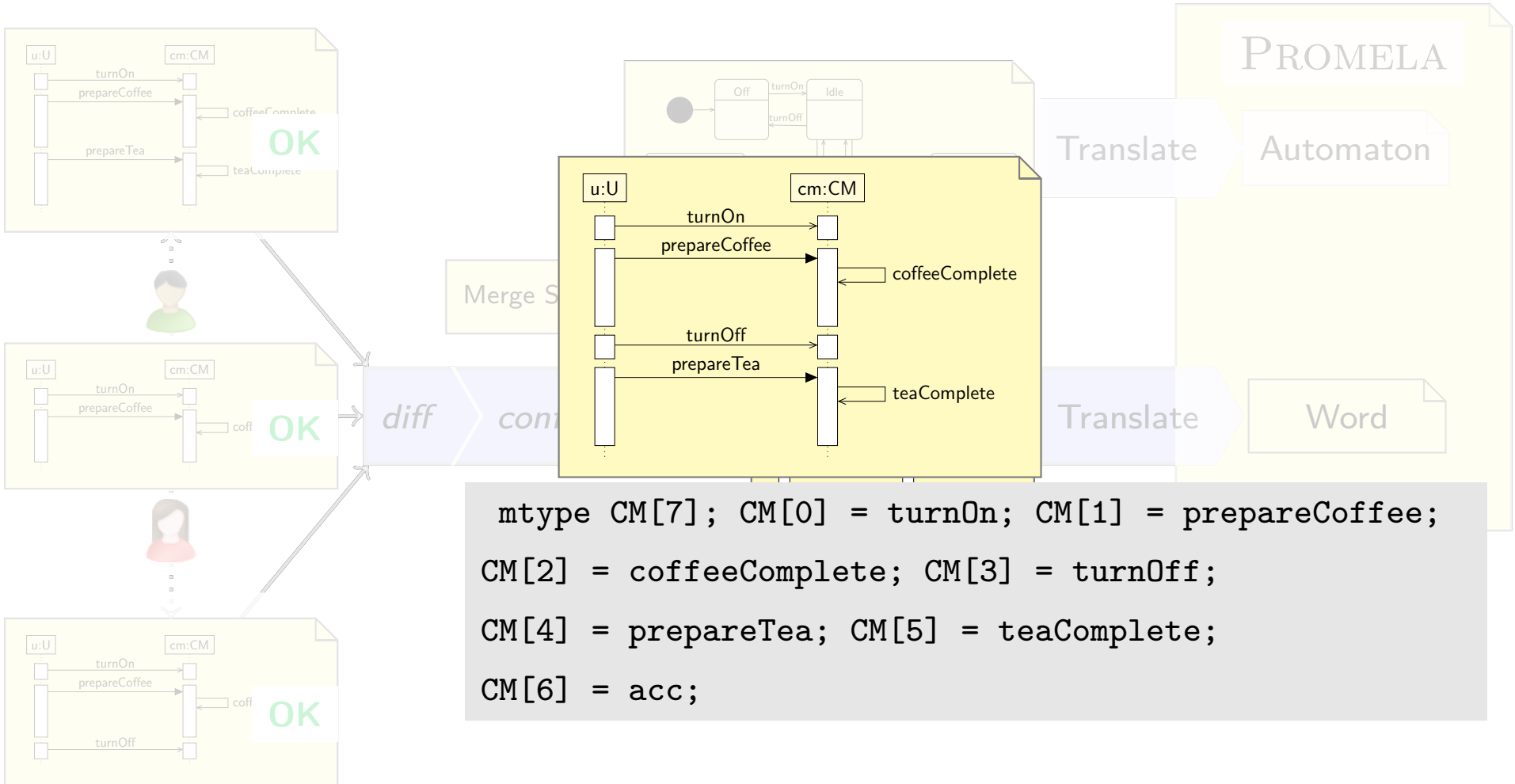
Semantics-aware model versioning

Using the SPIN model checker



Semantics-aware model versioning

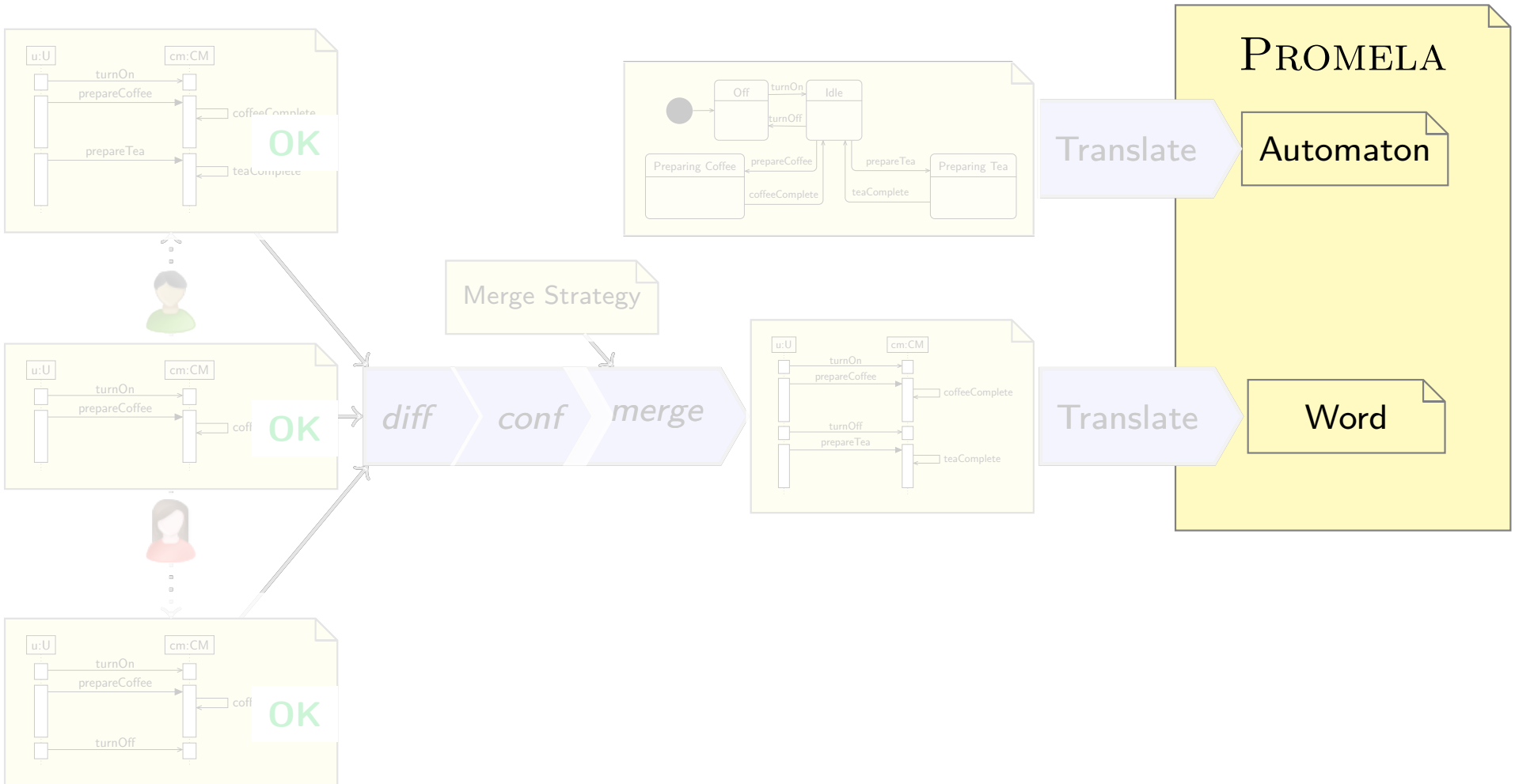
Using the SPIN model checker




```
mtype CM[7]; CM[0] = turnOn; CM[1] = prepareCoffee;
CM[2] = coffeeComplete; CM[3] = turnOff;
CM[4] = prepareTea; CM[5] = teaComplete;
CM[6] = acc;
```

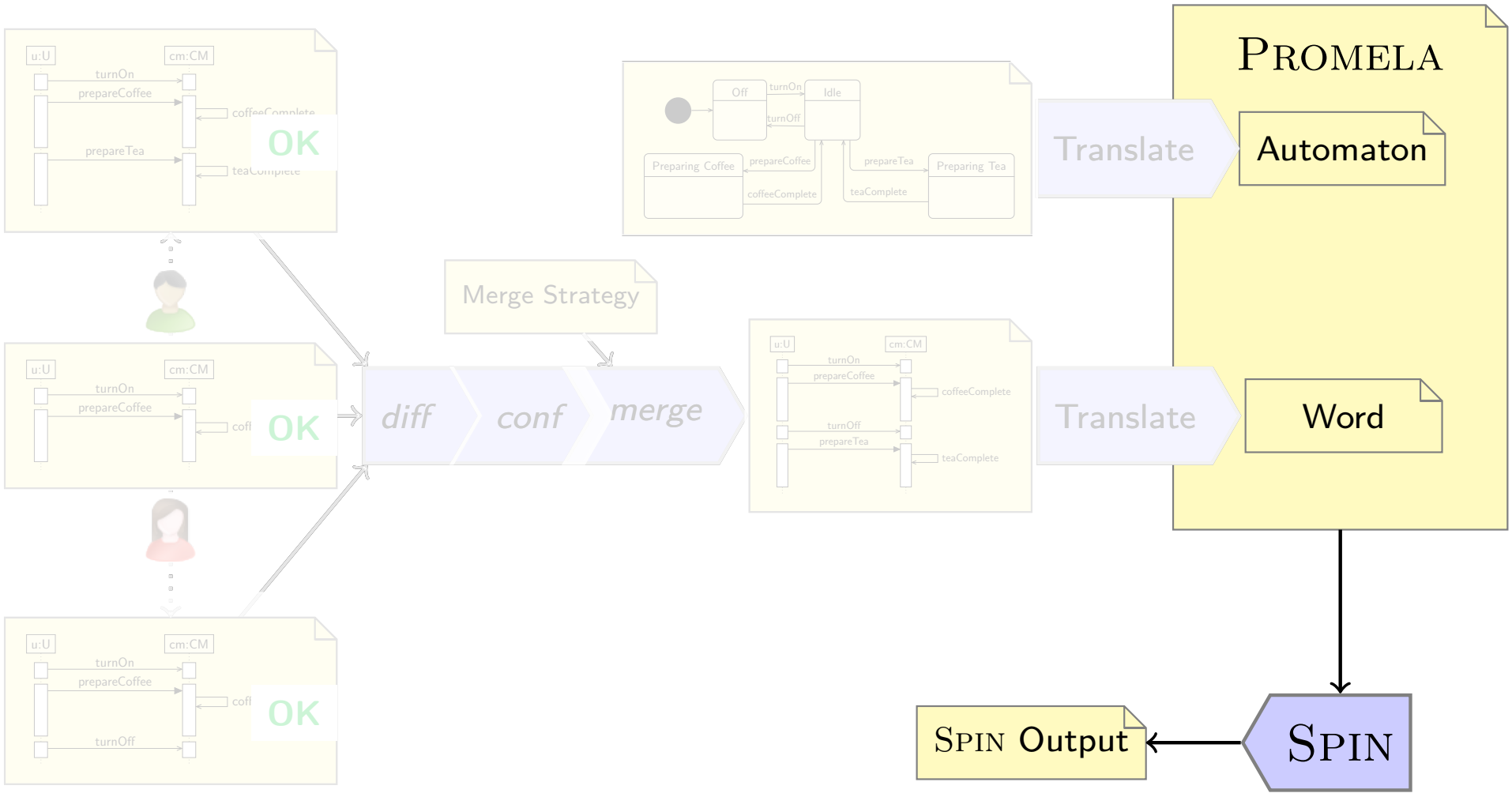
Semantics-aware model versioning

Using the SPIN model checker



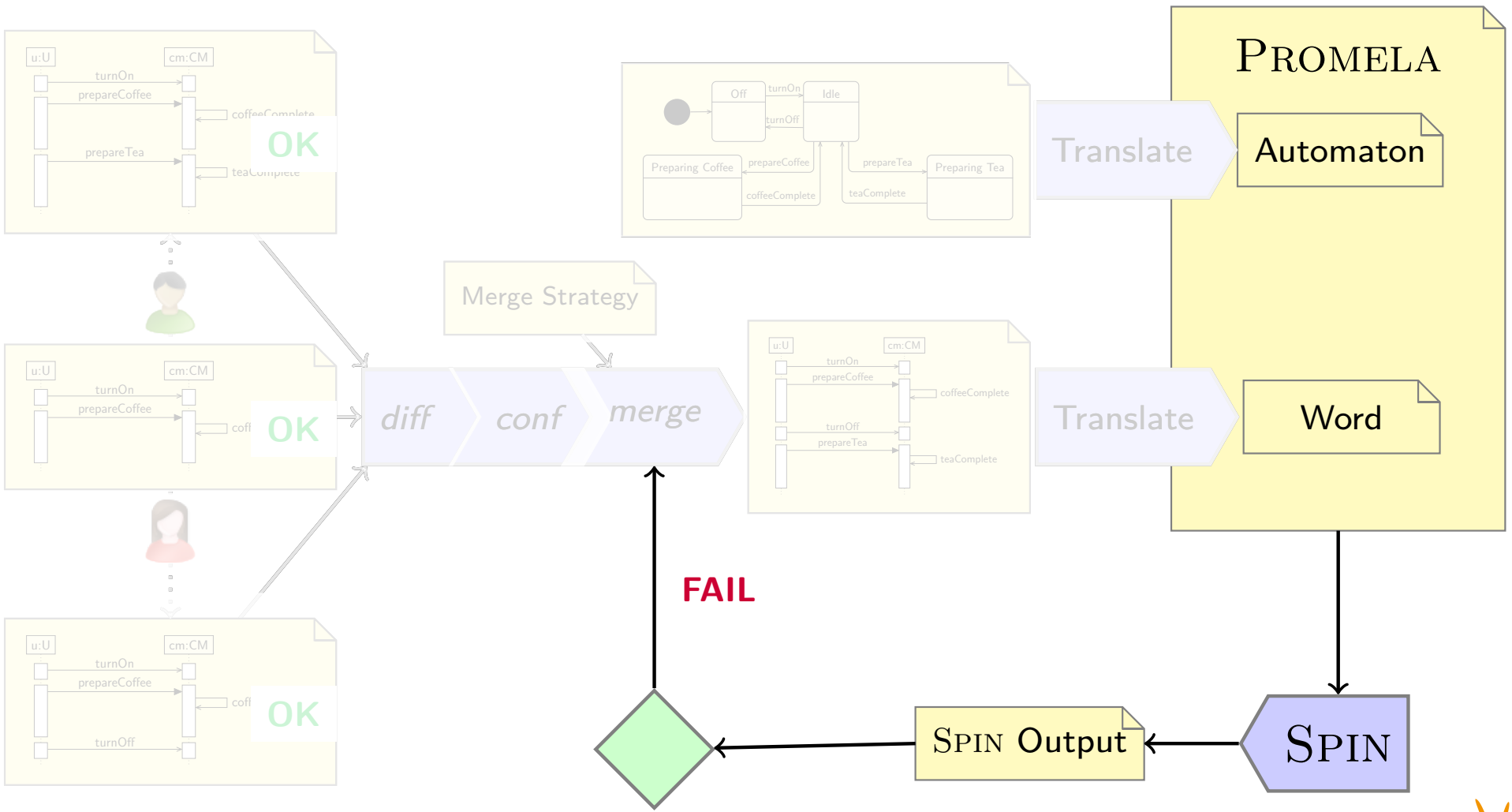
Semantics-aware model versioning

Using the SPIN model checker



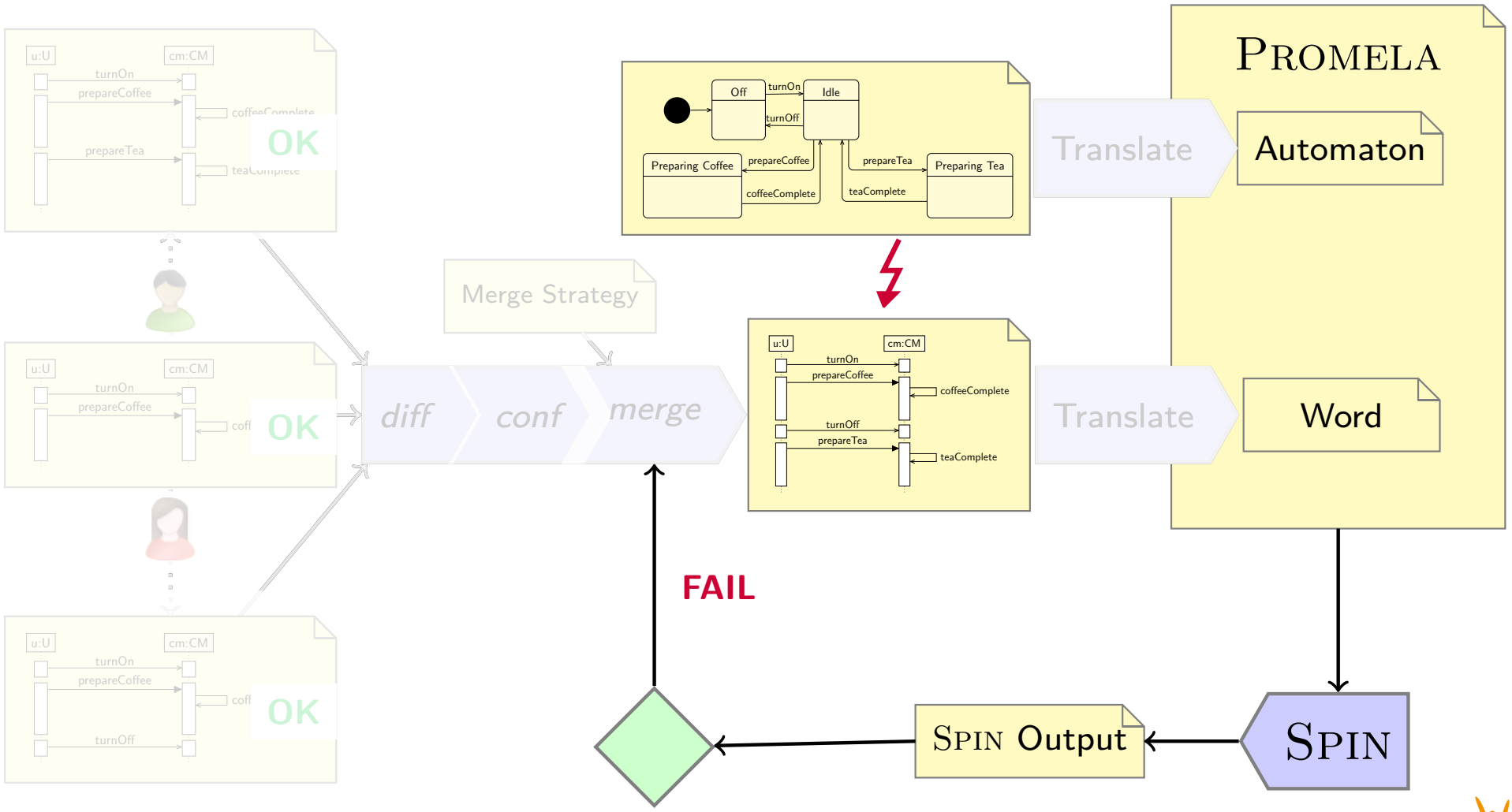
Semantics-aware model versioning

Using the SPIN model checker



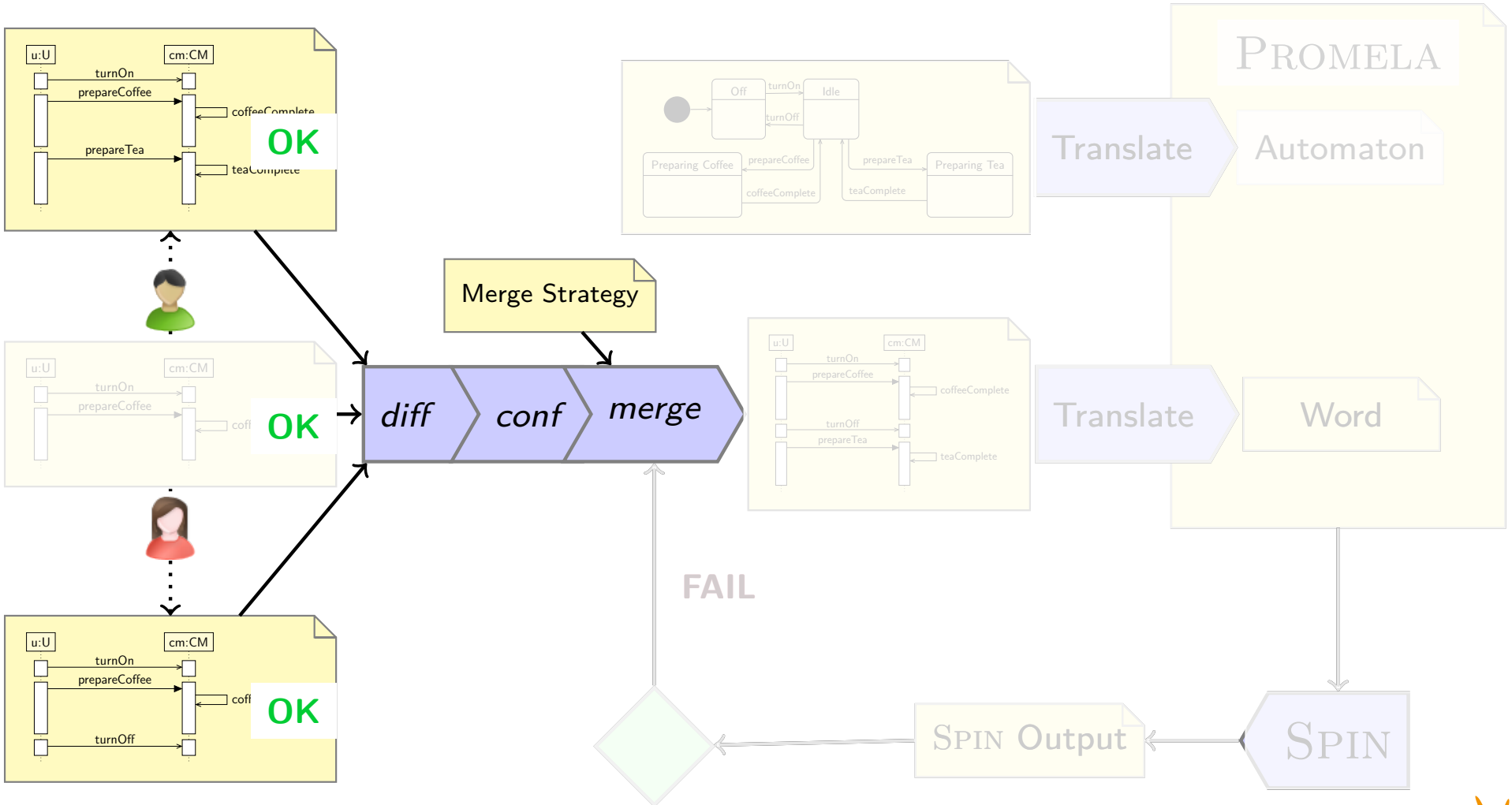
Semantics-aware model versioning

Using the SPIN model checker



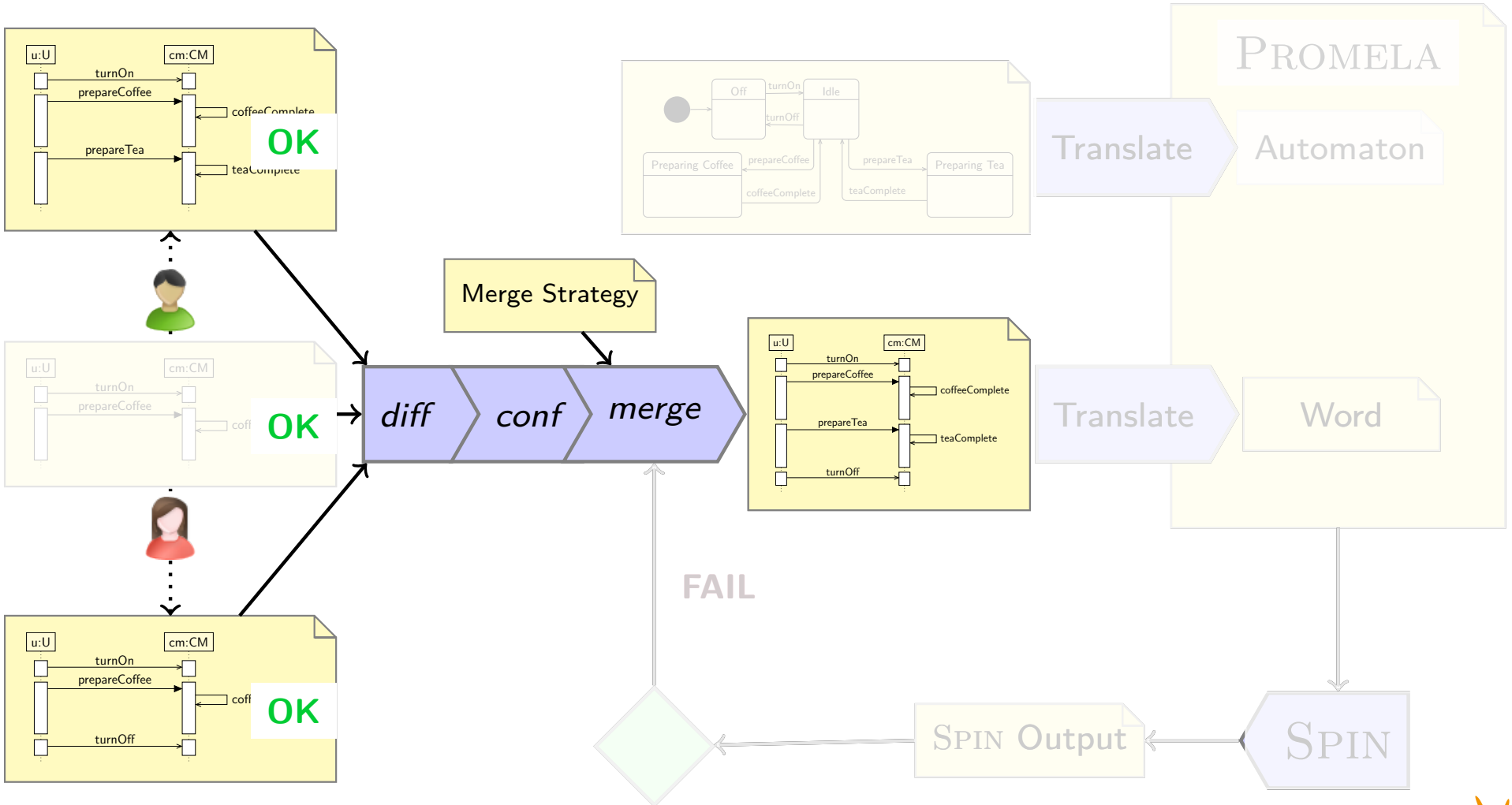
Semantics-aware model versioning

Using the SPIN model checker



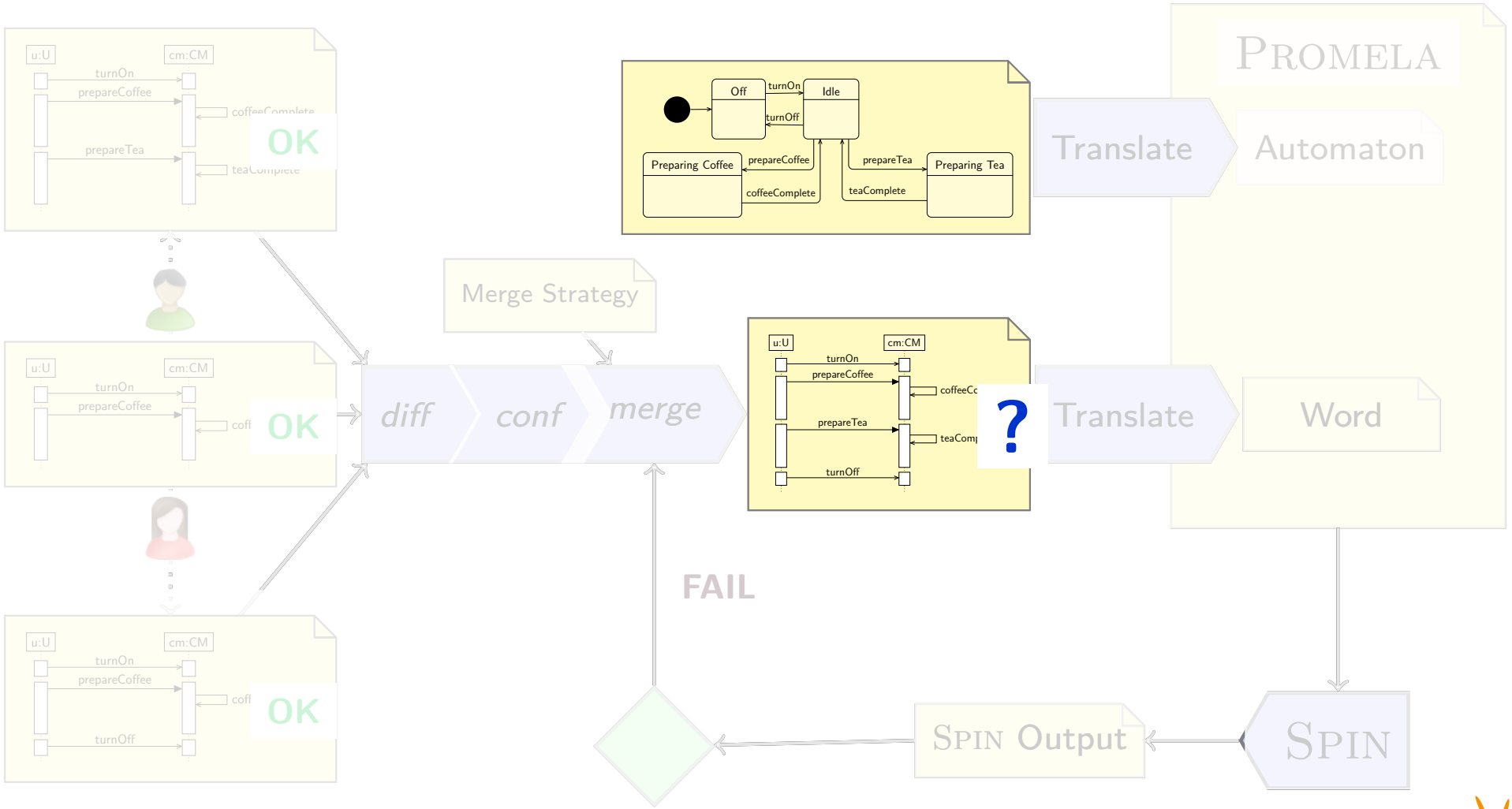
Semantics-aware model versioning

Using the SPIN model checker



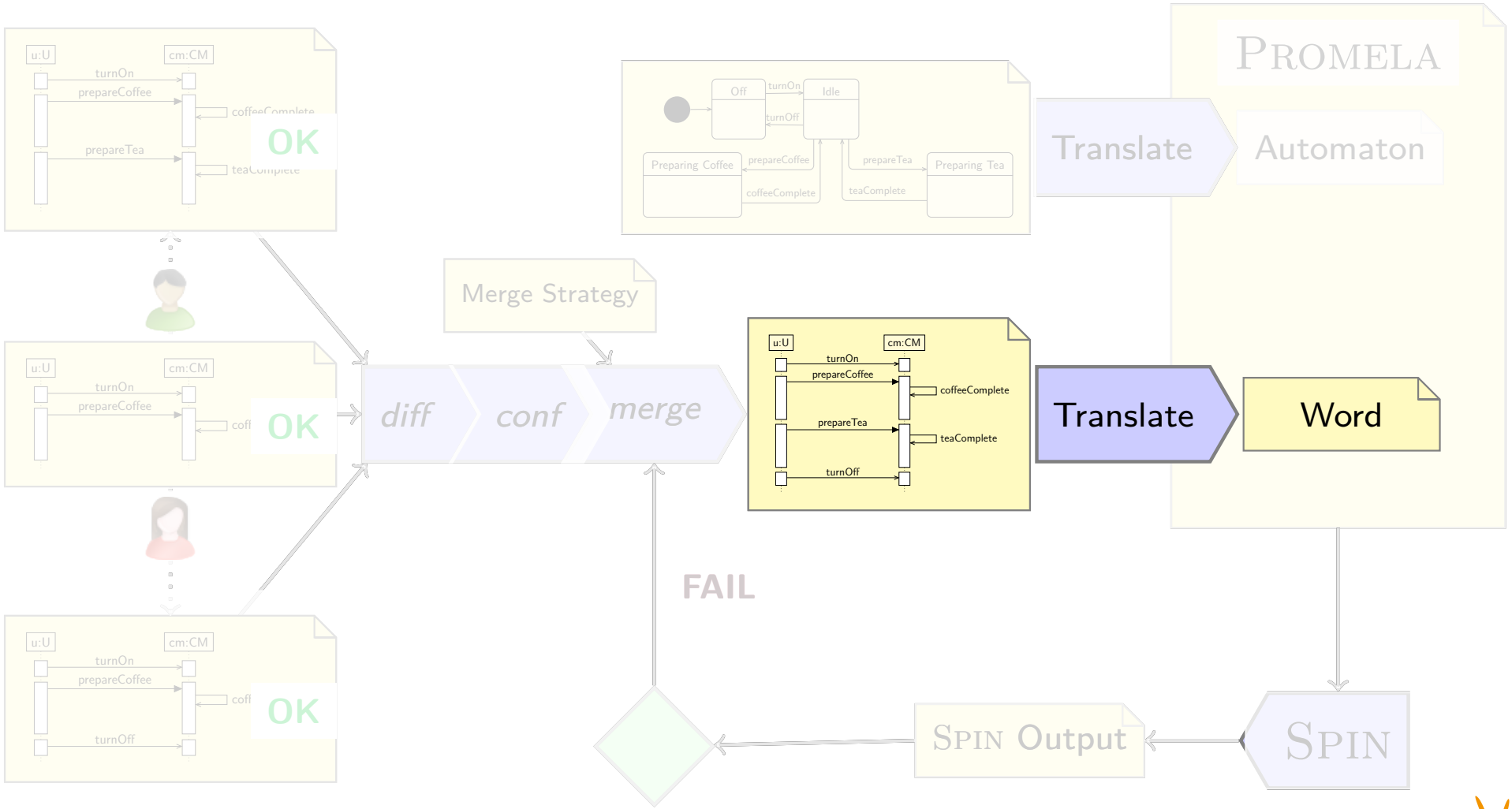
Semantics-aware model versioning

Using the SPIN model checker



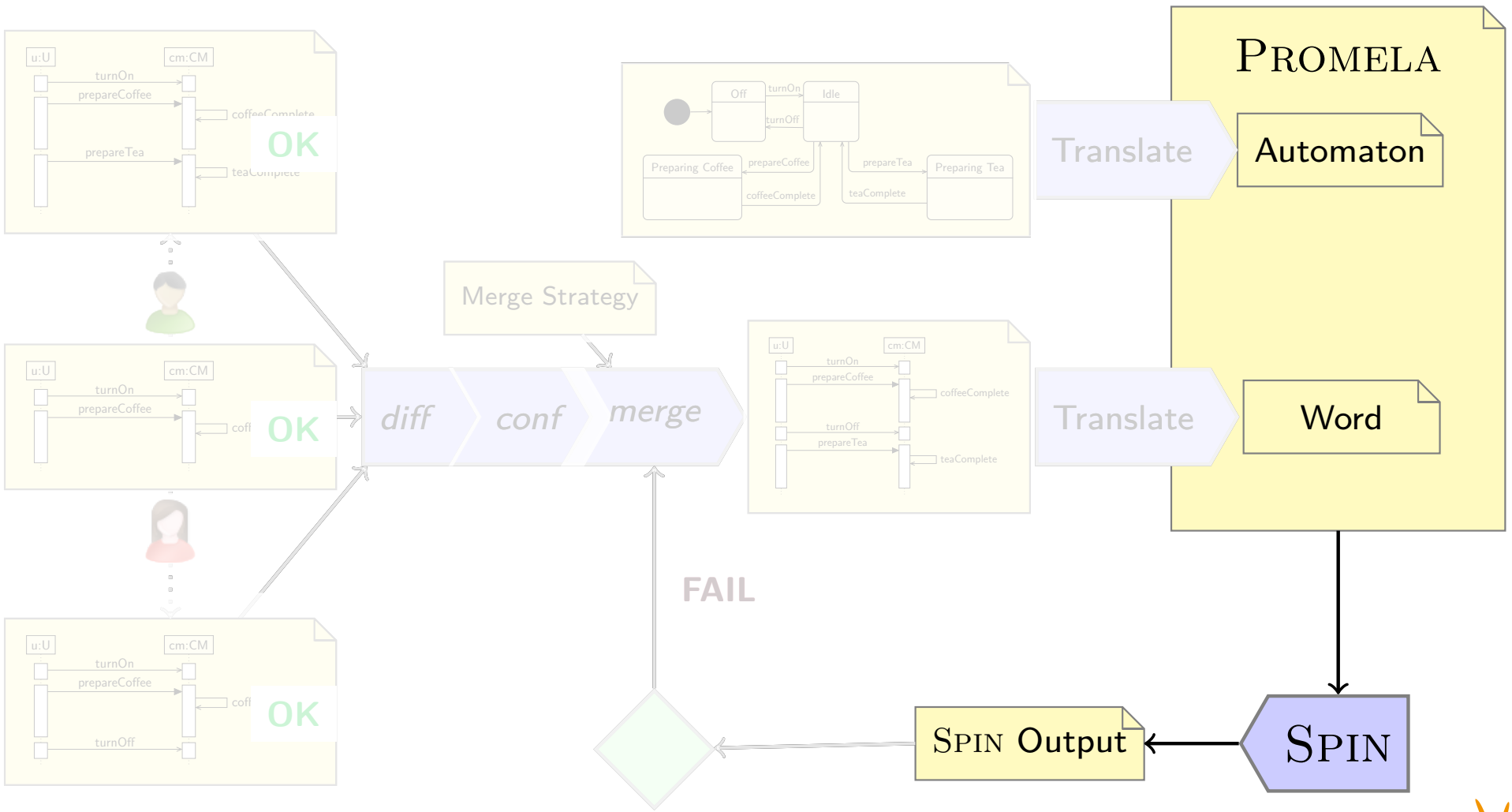
Semantics-aware model versioning

Using the SPIN model checker



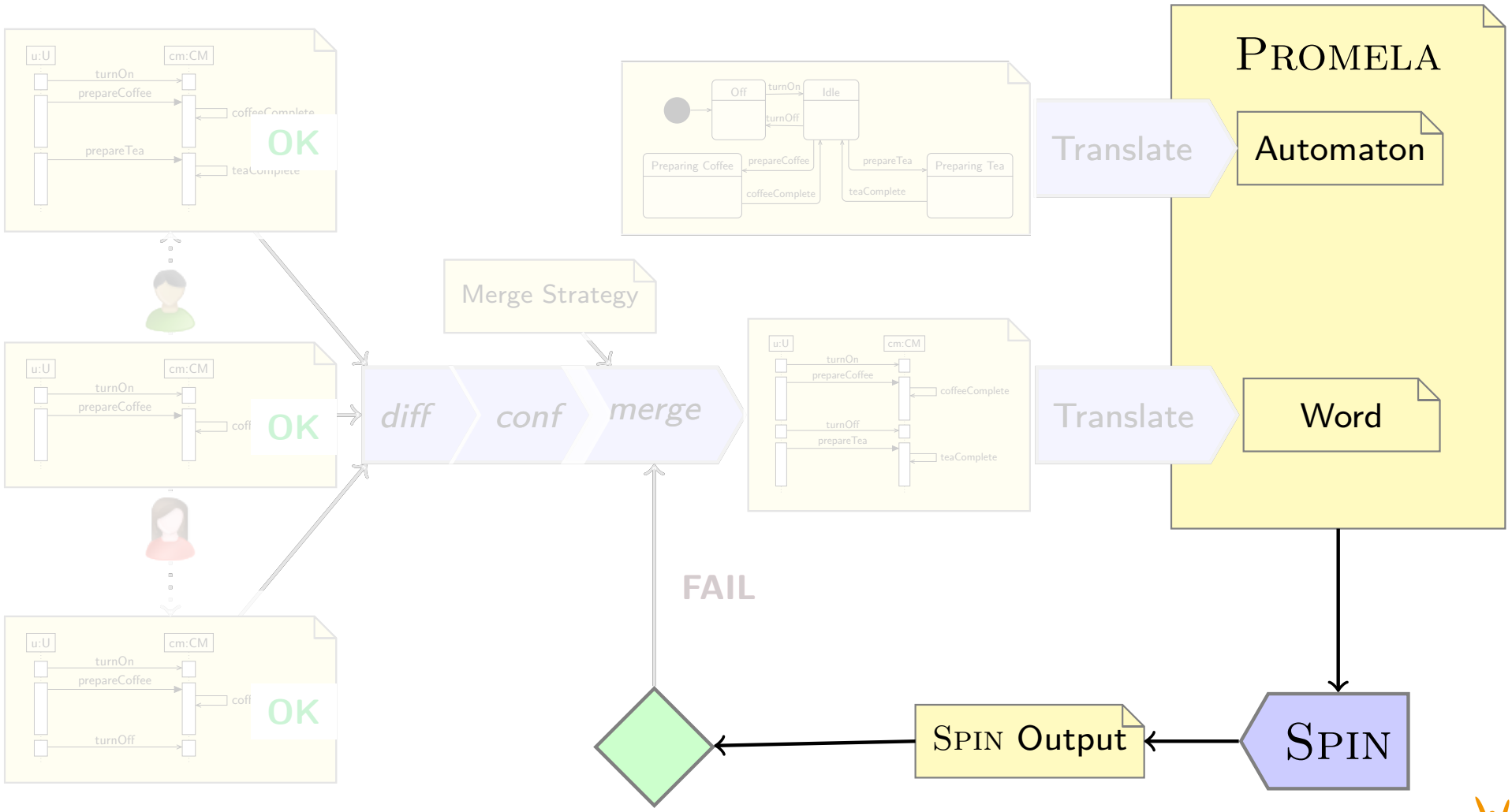
Semantics-aware model versioning

Using the SPIN model checker



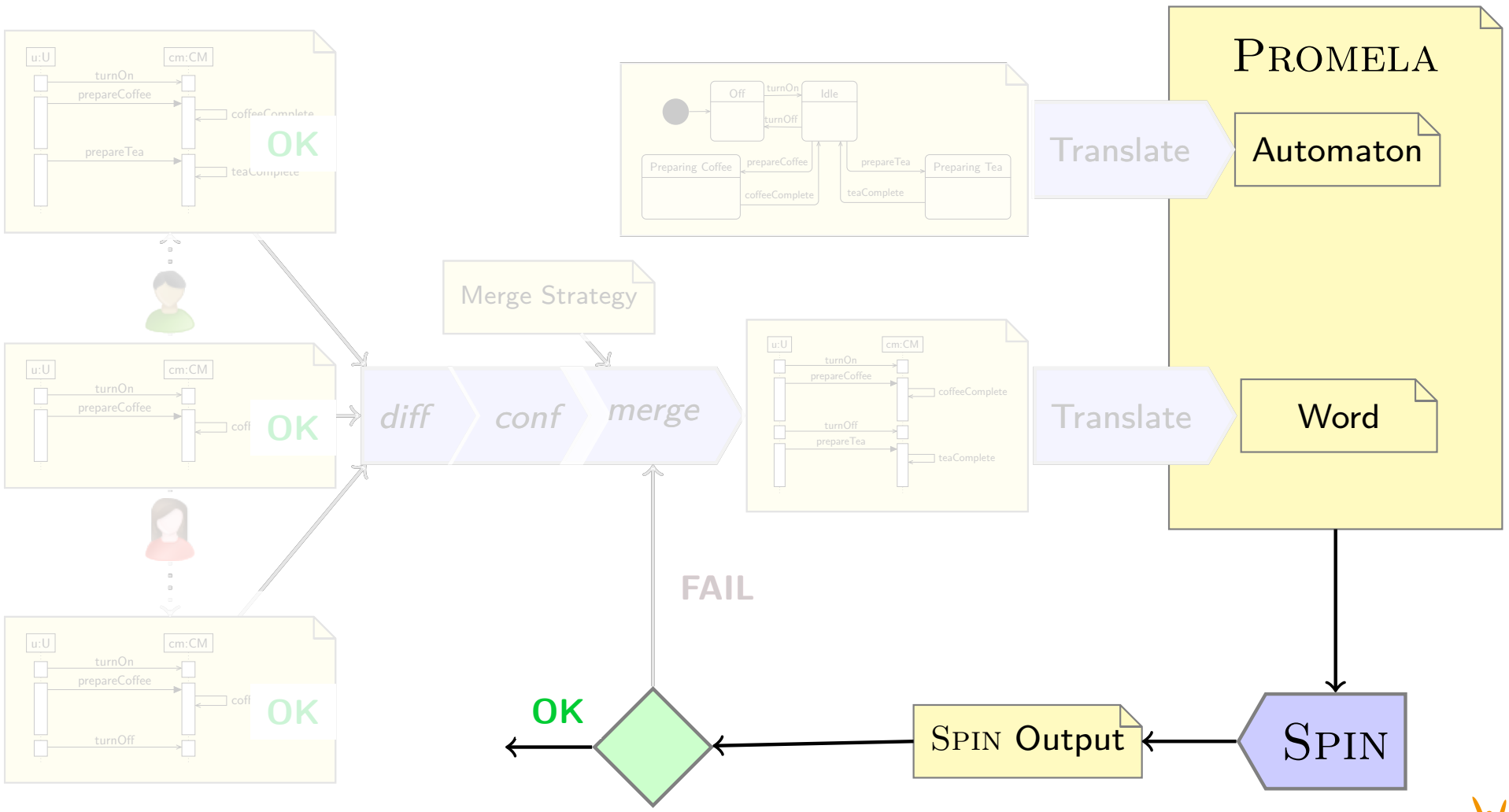
Semantics-aware model versioning

Using the SPIN model checker



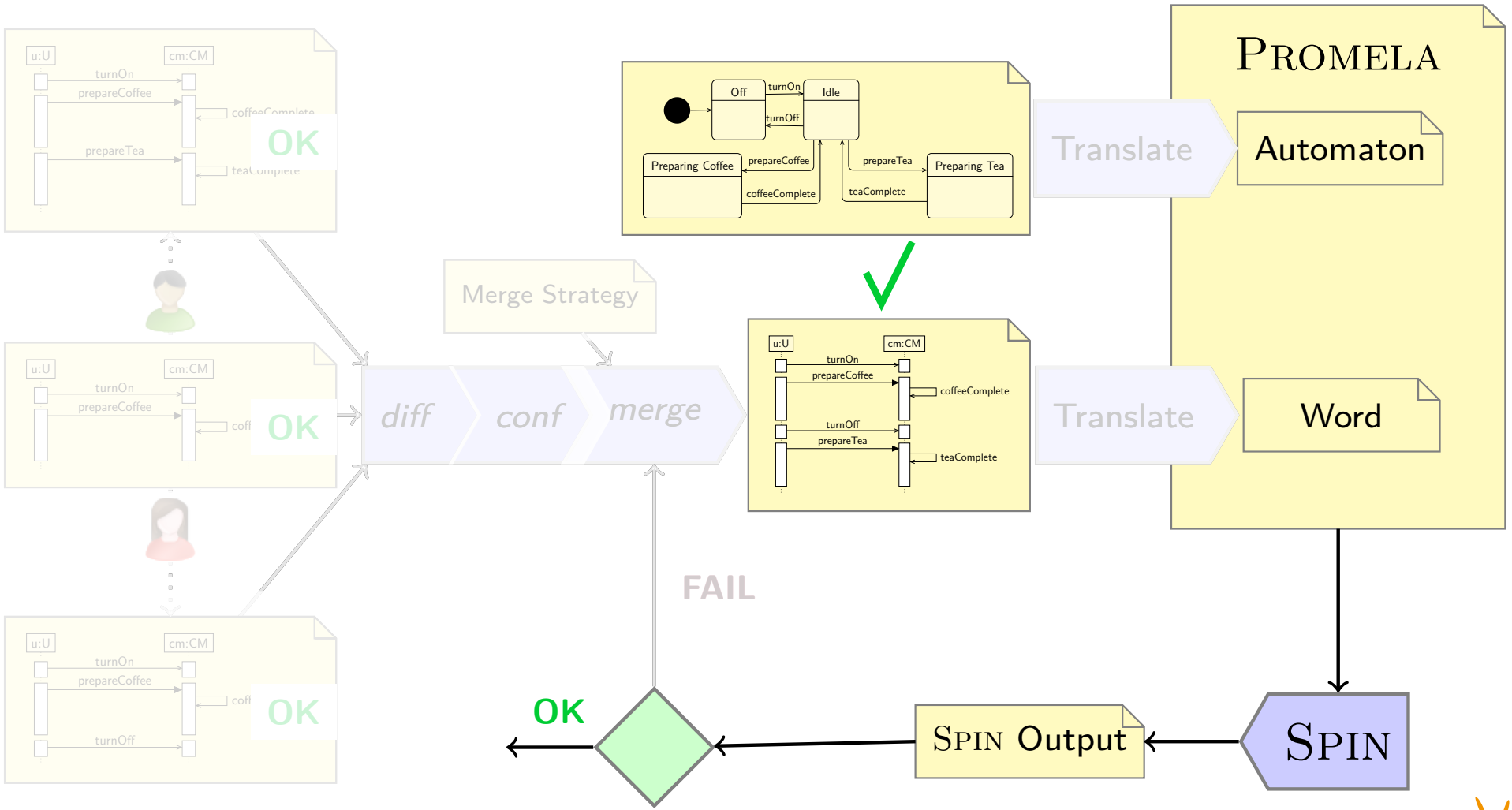
Semantics-aware model versioning

Using the SPIN model checker



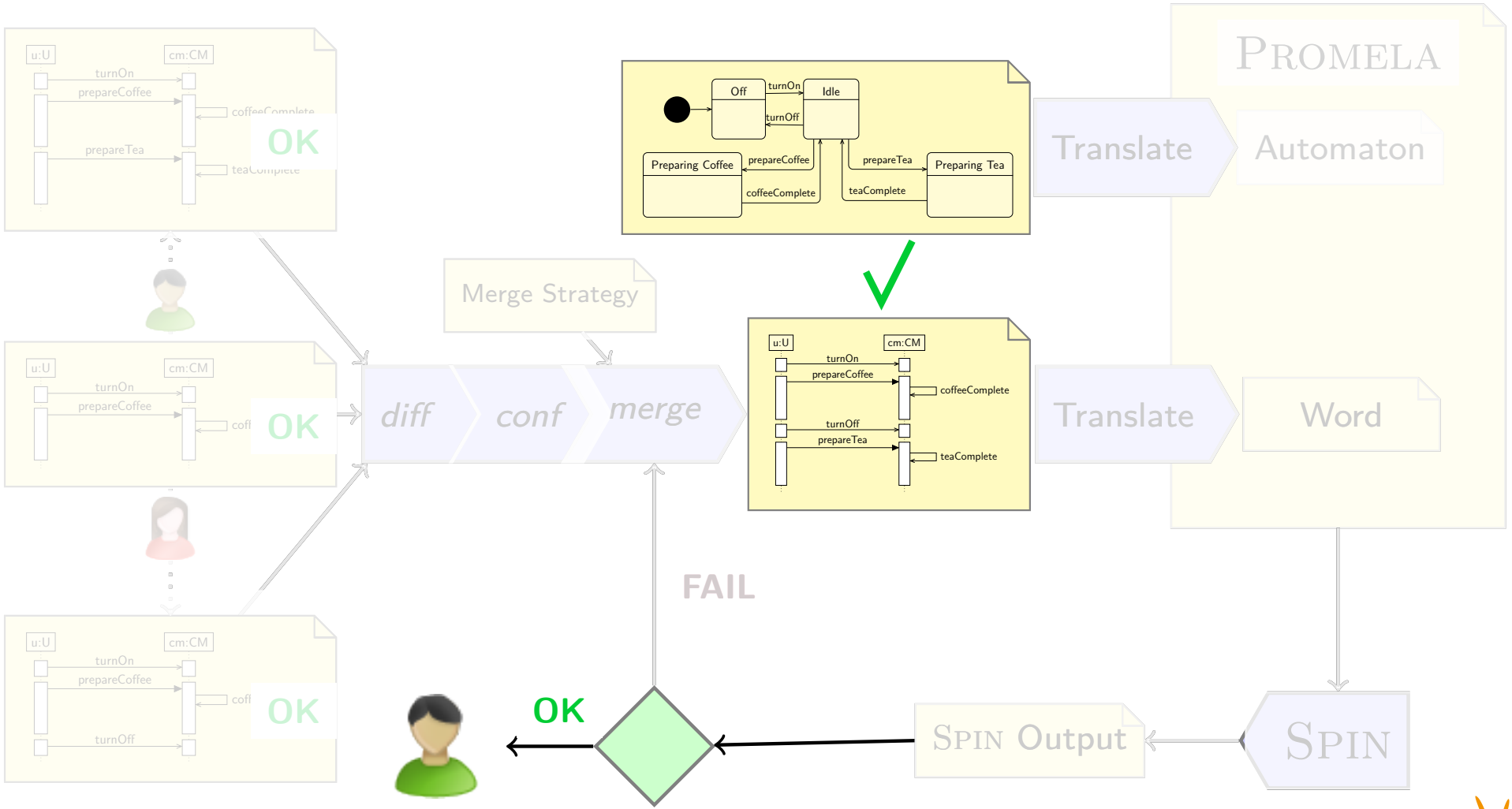
Semantics-aware model versioning

Using the SPIN model checker



Semantics-aware model versioning

Using the SPIN model checker

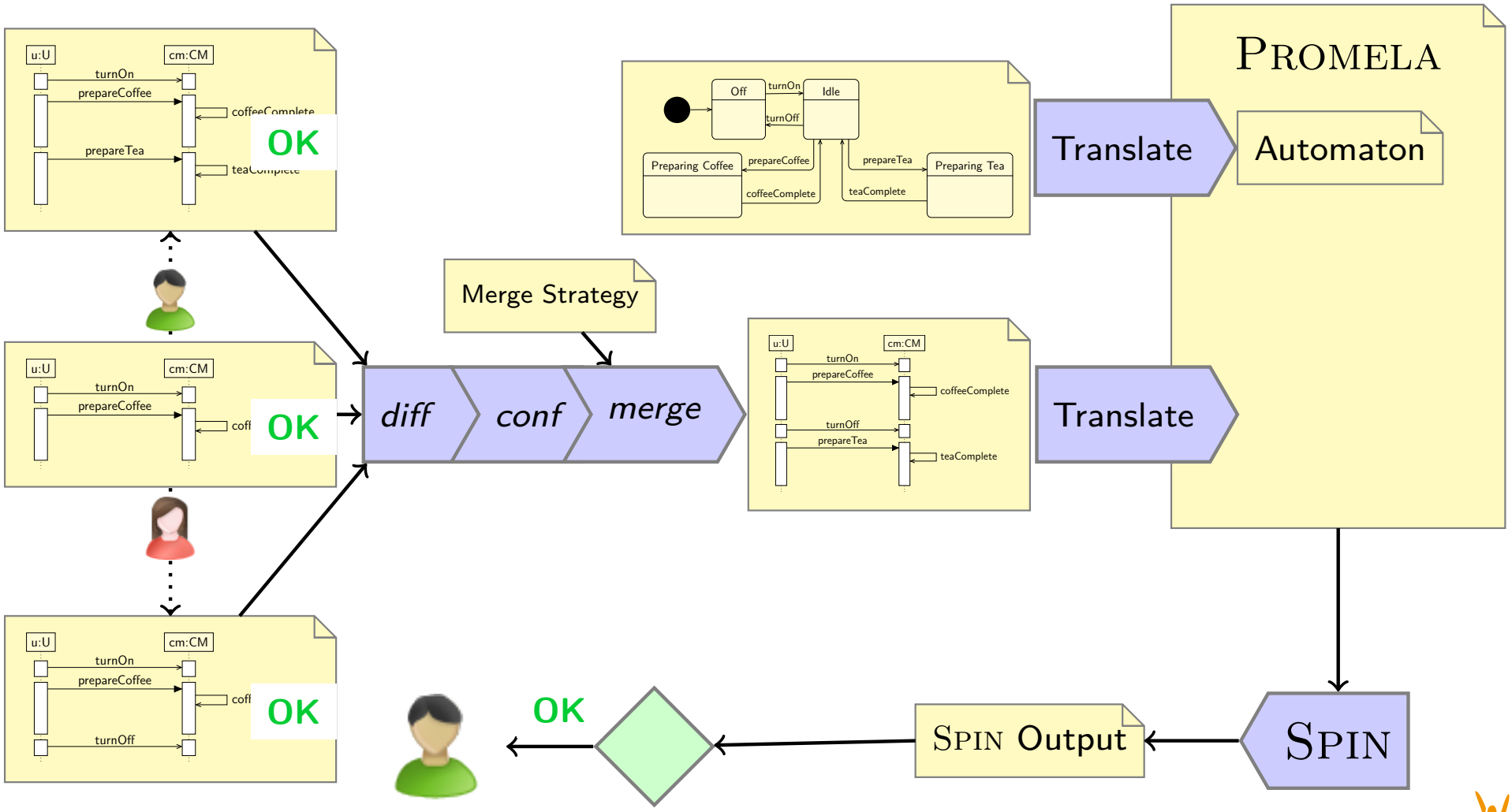


What's next?

Multiple State Machines and more complex Sequence Diagrams

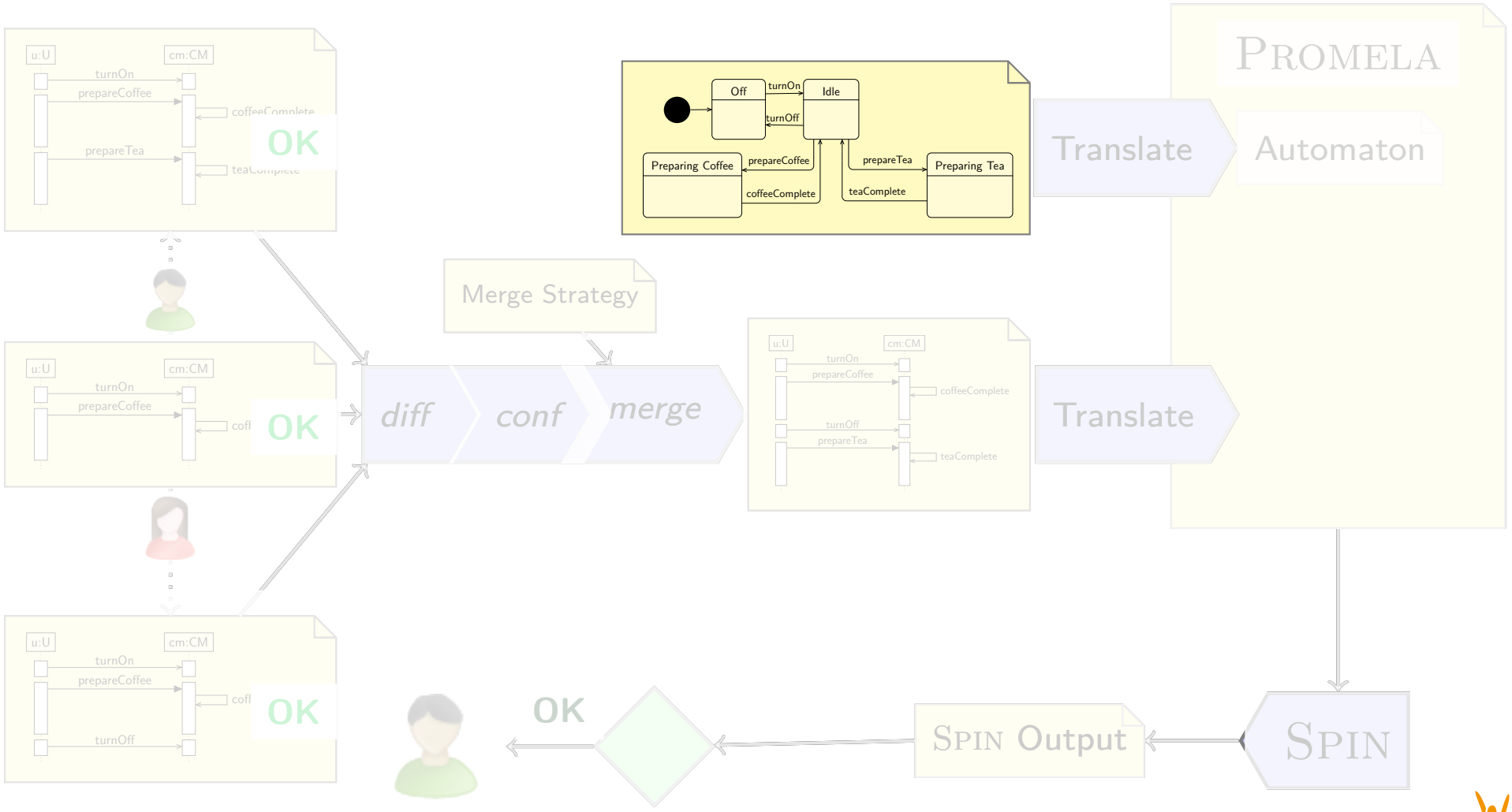
What's next?

Multiple State Machines and more complex Sequence Diagrams



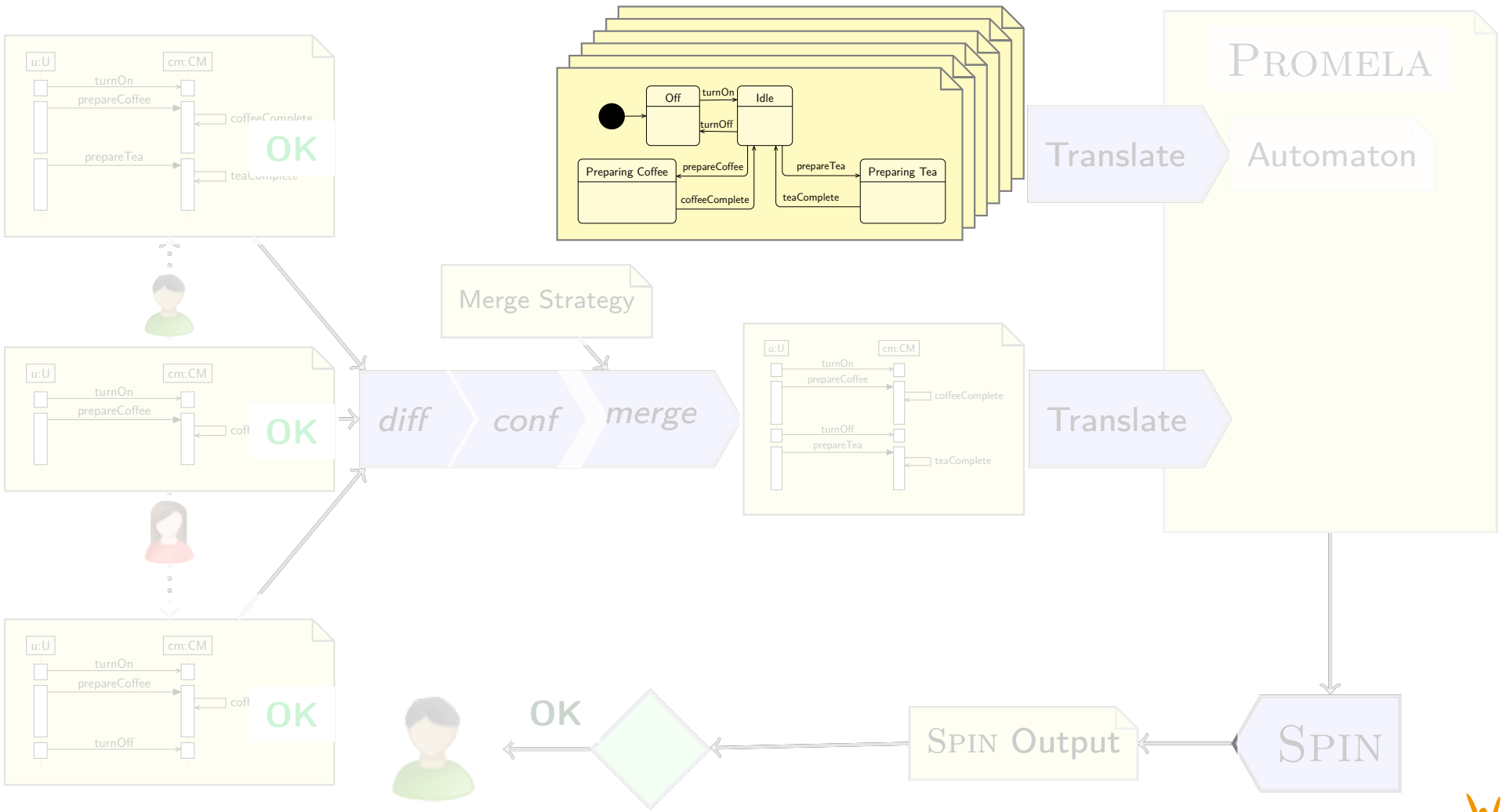
What's next?

Multiple State Machines and more complex Sequence Diagrams



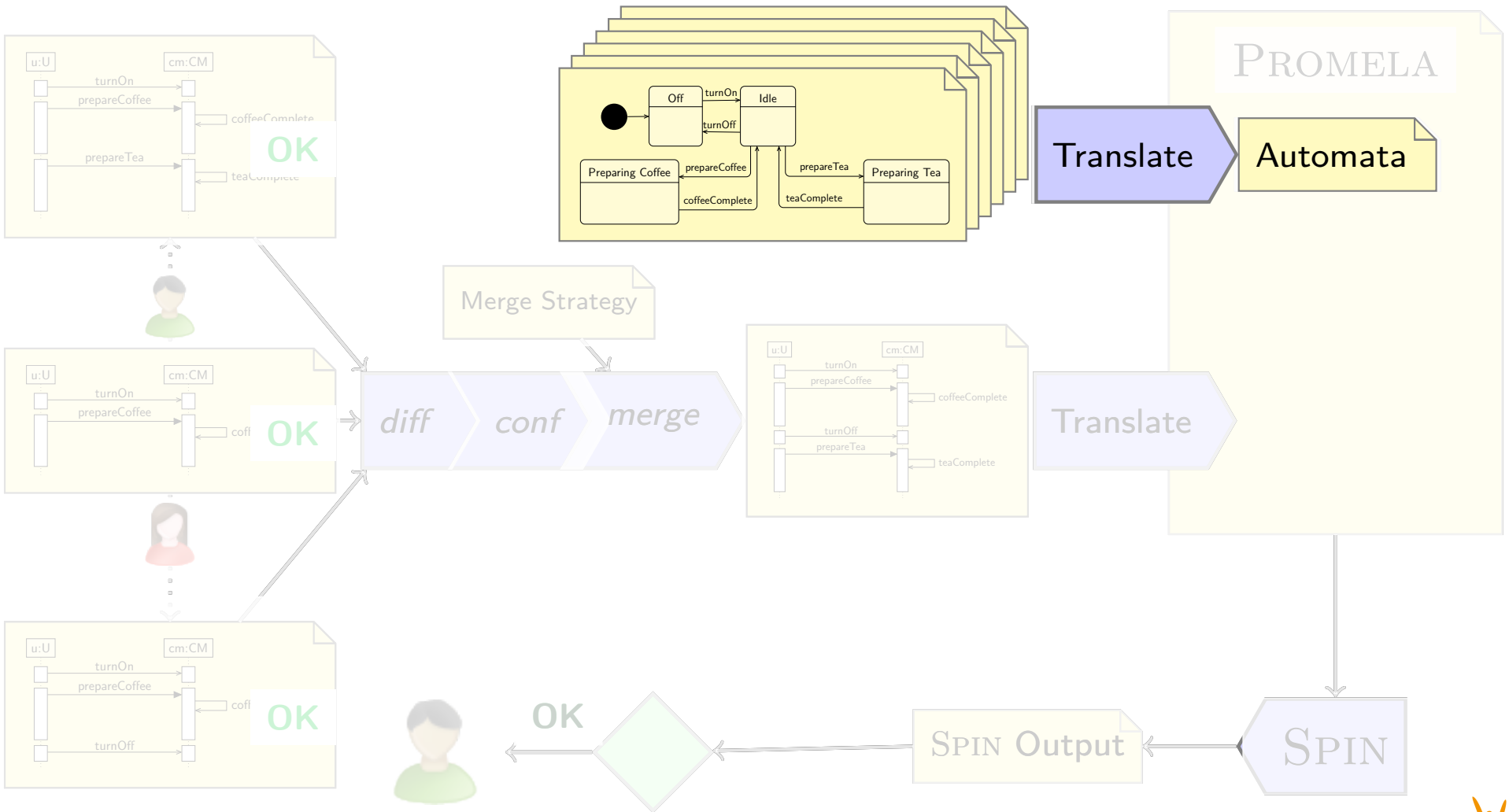
What's next?

Multiple State Machines and more complex Sequence Diagrams



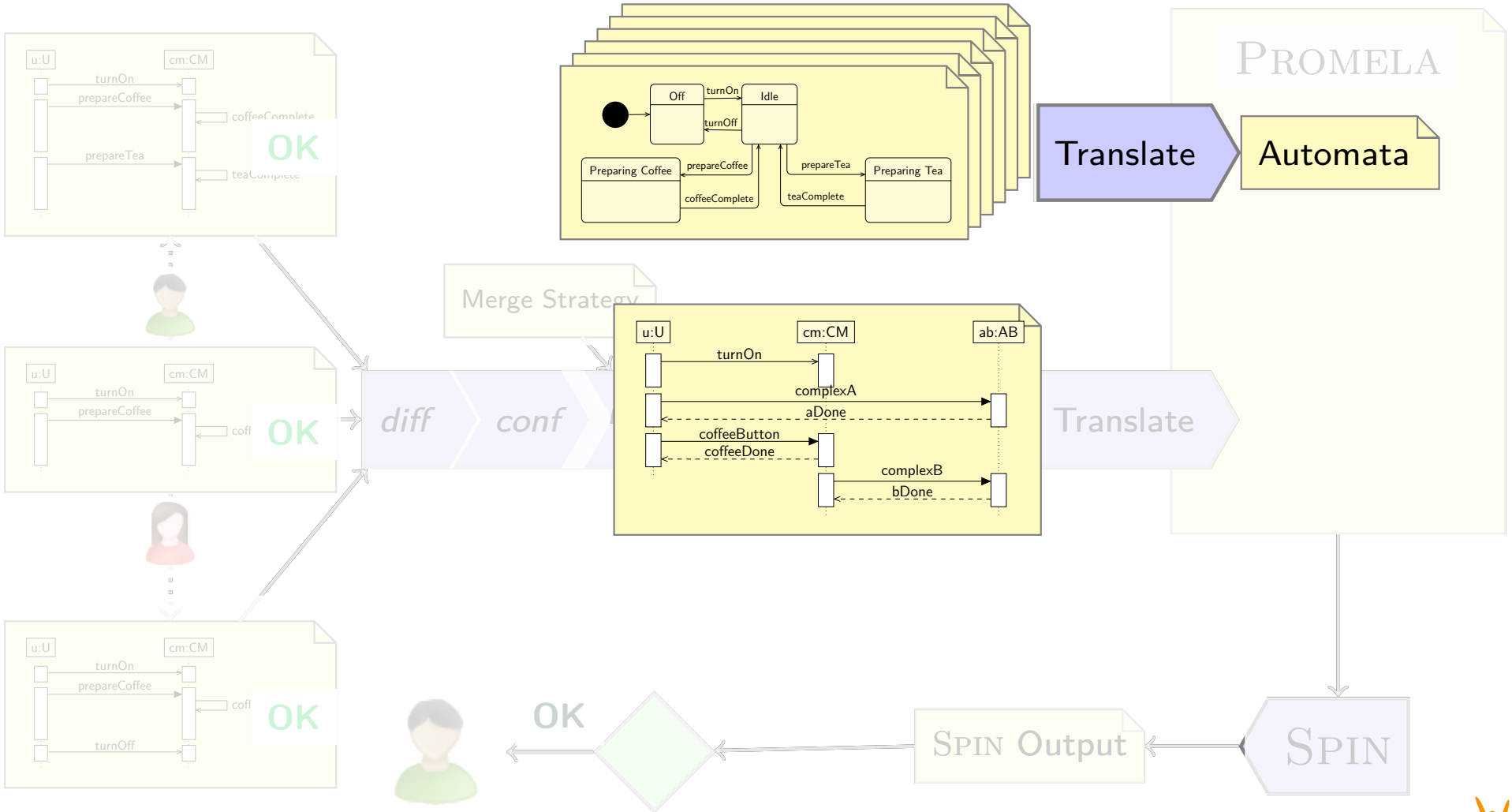
What's next?

Multiple State Machines and more complex Sequence Diagrams



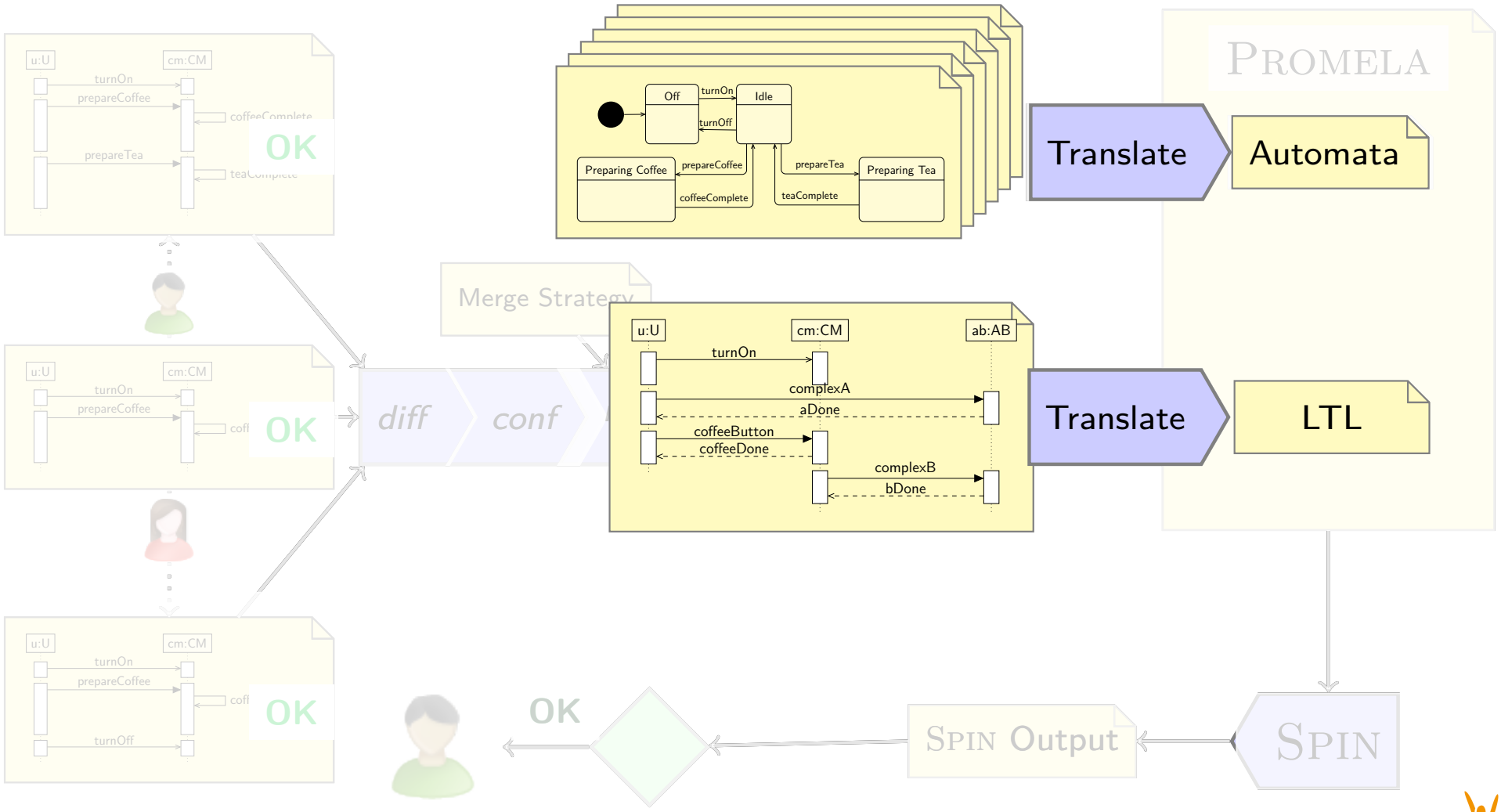
What's next?

Multiple State Machines and more complex Sequence Diagrams



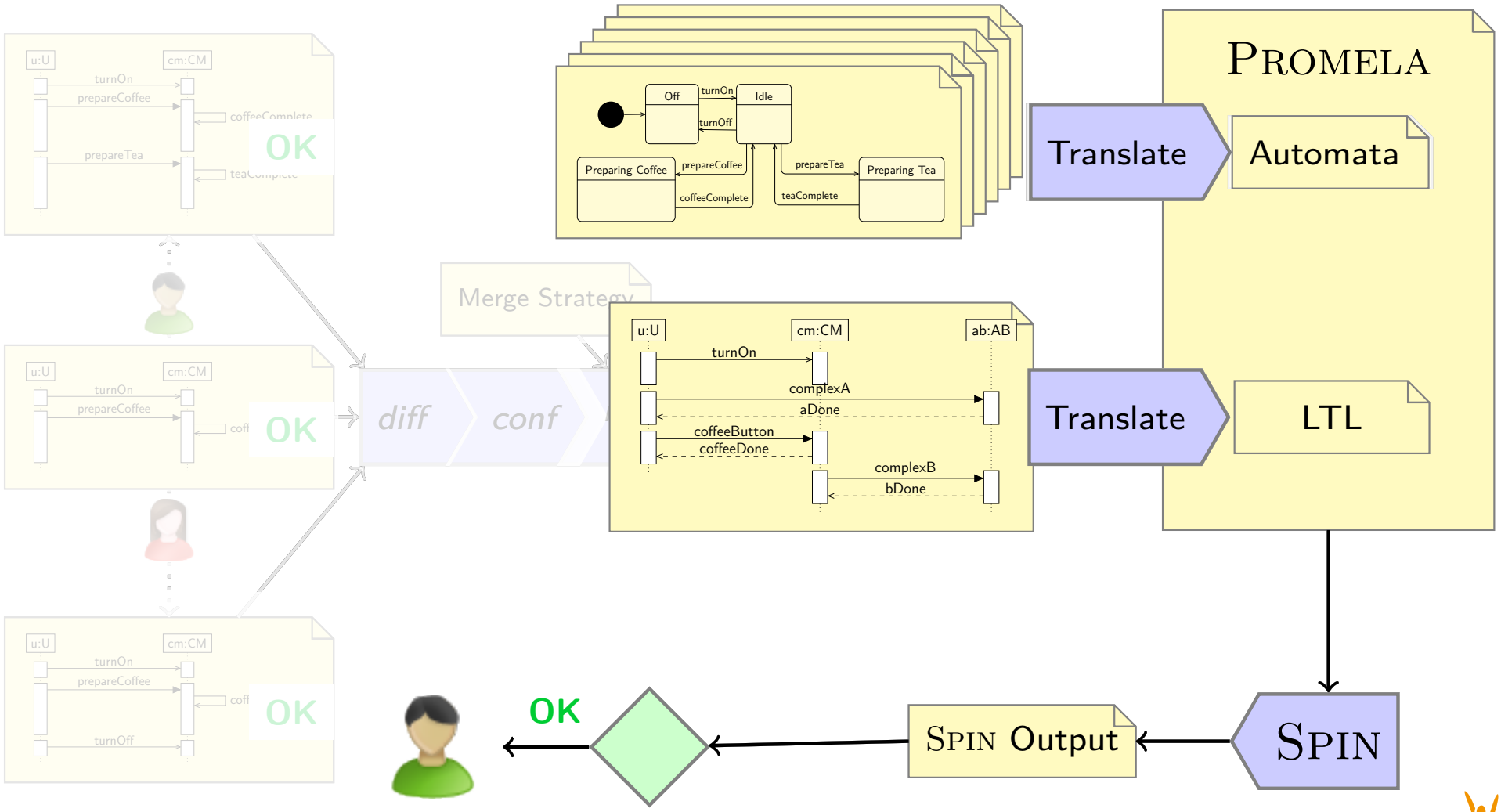
What's next?

Multiple State Machines and more complex Sequence Diagrams



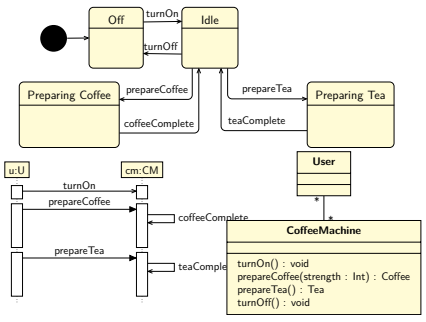
What's next?

Multiple State Machines and more complex Sequence Diagrams



Outlook

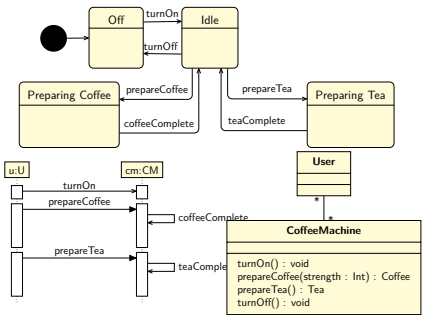
Model verification




```
package org.modelevolution.multiview.java.coffee;
public enum CoffeeMachineState implements ICoffeeMachineTransition {
        OFF {
                public void busy(CoffeeMachine cm) {
                        throw new IllegalStateException("busy not allowed in state OFF.");
                }
                public void turnOn(CoffeeMachine cm) {
                        cm.setCurrentState(IDLE);
                }
                public void turnOff(CoffeeMachine cm) {
                        throw new IllegalStateException("turnOff not allowed in state OFF");
                }
                public void done(CoffeeMachine cm) {
                        throw new IllegalStateException("done not allowed in state OFF");
                }
        },
        IDLE {
                public void busy(CoffeeMachine cm) {
                        cm.setCurrentState(BUSY);
                }
                public void turnOn(CoffeeMachine cm) {
                        throw new IllegalStateException("turnOn not allowed in state IDLE.");
                }
                public void turnOff(CoffeeMachine cm) {
                        cm.setCurrentState(OFF);
                }
                public void done(CoffeeMachine cm) {
                        throw new IllegalStateException("done not allowed in state IDLE.");
                }
        },
        BUSY() {
                {
                        done(null);
                }
```

Outlook

Model verification

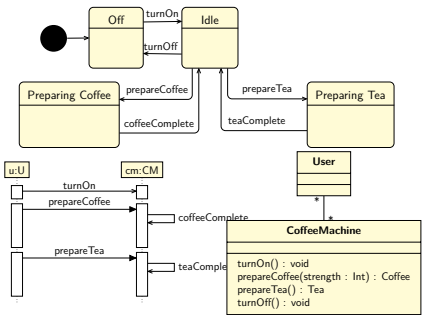



```
package org.modelevolution.multiview.java.coffee;
public enum CoffeeMachineState implements ICoffeeMachineTransition {
        OFF {
                public void busy(CoffeeMachine cm) {
                        throw new IllegalStateException("busy not allowed in state OFF.");
                }
                public void turnOn(CoffeeMachine cm) {
                        cm.setCurrentState(IDLE);
                }
                public void turnOff(CoffeeMachine cm) {
                        throw new IllegalStateException("turnOff not allowed in state OFF");
                }
                public void done(CoffeeMachine cm) {
                        throw new IllegalStateException("done not allowed in state OFF");
                }
        },
        IDLE {
                public void busy(CoffeeMachine cm) {
                        cm.setCurrentState(BUSY);
                }
                public void turnOn(CoffeeMachine cm) {
                        throw new IllegalStateException("turnOn not allowed in state IDLE.");
                }
                public void turnOff(CoffeeMachine cm) {
                        cm.setCurrentState(OFF);
                }
                public void done(CoffeeMachine cm) {
                        throw new IllegalStateException("done not allowed in state IDLE.");
                }
        },
        BUSY() {
                {
                        done(null);
                }
```

Software verification, testing

Outlook

Model verification



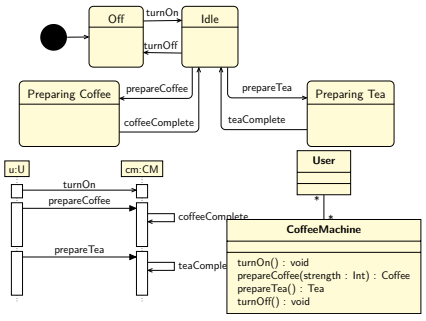

```
package org.modelevolution.multiview.java.coffee;
public enum CoffeeMachineState implements ICoffeeMachineTransition {
        OFF {
                public void busy(CoffeeMachine cm) {
                        throw new IllegalStateException("busy not allowed in state OFF.");
                }
                public void turnOn(CoffeeMachine cm) {
                        cm.setCurrentState(IDLE);
                }
                public void turnOff(CoffeeMachine cm) {
                        throw new IllegalStateException("turnOff not allowed in state OFF");
                }
                public void done(CoffeeMachine cm) {
                        throw new IllegalStateException("done not allowed in state OFF");
                }
        },
        IDLE {
                public void busy(CoffeeMachine cm) {
                        cm.setCurrentState(BUSY);
                }
                public void turnOn(CoffeeMachine cm) {
                        throw new IllegalStateException("turnOn not allowed in state IDLE.");
                }
                public void turnOff(CoffeeMachine cm) {
                        cm.setCurrentState(OFF);
                }
                public void done(CoffeeMachine cm) {
                        throw new IllegalStateException("done not allowed in state IDLE.");
                }
        },
        BUSY() {
                {
                        done(null);
                }
```

Model verification?

Software verification, testing

Outlook

Model verification



```
package org.modelevolution.multiview.java.coffee;
public enum CoffeeMachineState implements ICoffeeMachineTransition {
        OFF {
                public void busy(CoffeeMachine cm) {
                        throw new IllegalStateException("busy not allowed in state OFF.");
                }
                public void turnOn(CoffeeMachine cm) {
                        cm.setCurrentState(IDLE);
                }
                public void turnOff(CoffeeMachine cm) {
                        throw new IllegalStateException("turnOff not allowed in state OFF");
                }
                public void done(CoffeeMachine cm) {
                        throw new IllegalStateException("done not allowed in state OFF");
                }
        },
        IDLE {
                public void busy(CoffeeMachine cm) {
                        cm.setCurrentState(BUSY);
                }
                public void turnOn(CoffeeMachine cm) {
                        throw new IllegalStateException("turnOn not allowed in state IDLE.");
                }
                public void turnOff(CoffeeMachine cm) {
                        cm.setCurrentState(OFF);
                }
                public void done(CoffeeMachine cm) {
                        throw new IllegalStateException("done not allowed in state IDLE.");
                }
        },
        BUSY() {
                {
                        done(null);
                }
```

Model verification?

Software verification, testing

Idea

Find errors on *model* level

Then work only on *consistent* models

Summary

- Working towards a uniform framework
- First step: Semantic merging
- Integration into an automatic versioning tool
- Ongoing work on more complex state machine / sequence diagrams
- Might be useful for modeling test cases