Transforming Live Sequence Charts into Statecharts

Riandi Wiguna
rian.wiguna@mail.mcgill.ca
School of Computer Science
McGill University
April 13, 2004

Overview

1. Recap of Play-In/Play-Out
2. Recap of LSCs (Live Sequence Charts)
3. LSC Specification in AToM3
4. LSC to Statechart Transformation
5. Transformed Microwave Functions
6. DChart Demo

Recap of Play-In/Play-Out

The Play-In/Play-Out Approach is a way to easily generate and test LSCs (Live Sequence Charts). LSCs model all desired system reactions, providing a complete design for the system.

A full LSC specification of a system can be transformed into statecharts.

Recap of Play-In/Play-Out

This presentation is based off "Synthesizing State-Based Object Systems from LSC Specifications" by David Harel and Hillel Kugler, with additional information from:

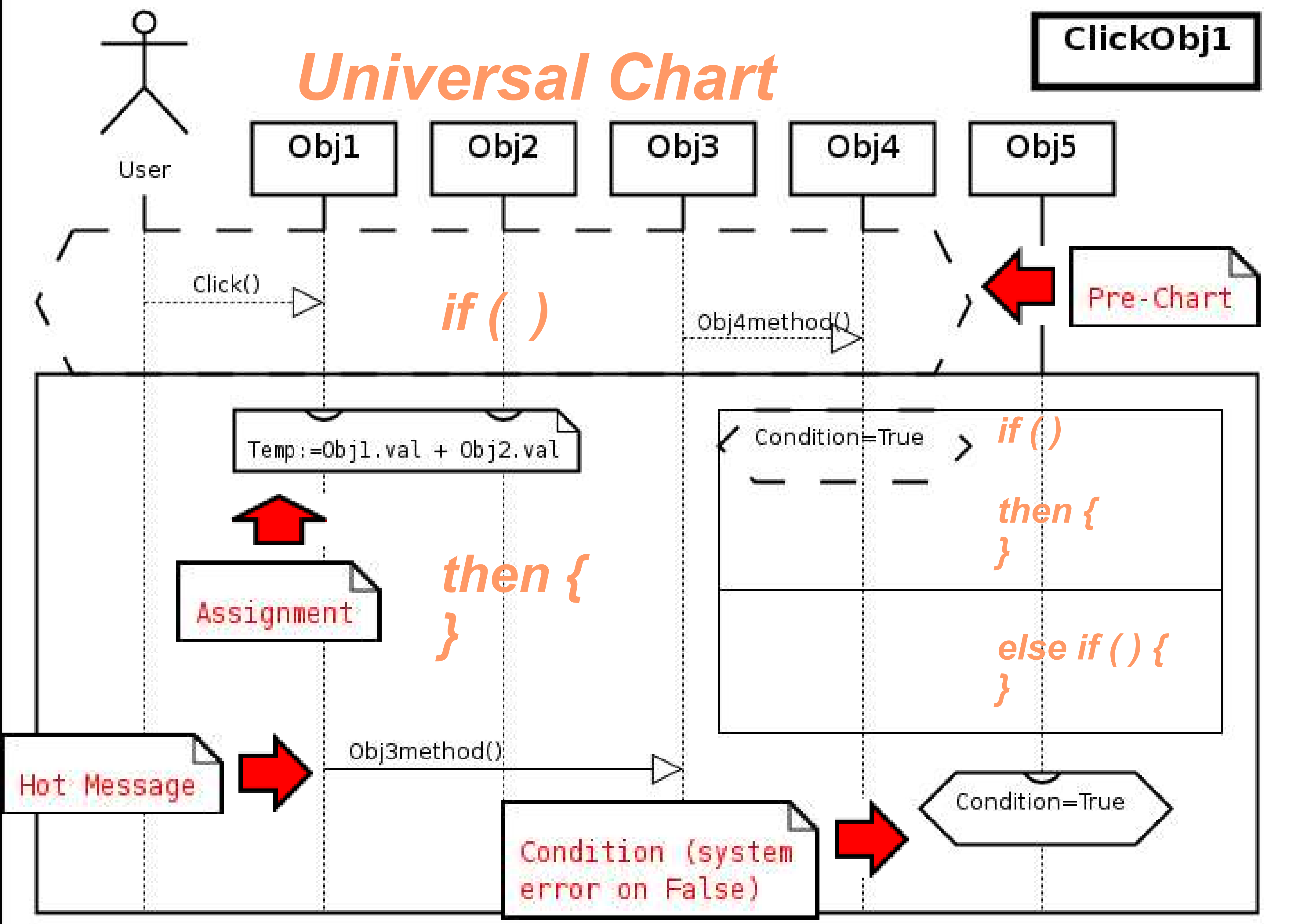
1. "DCharts, a Formalism for Modeling and Simulation Based Design for Reactive Software Systems" by Thomas Huning Feng
2. "Can Behavioral Requirements be Executed? (And why would we want to do so?)" by David Harel
3. "Specifying and Executing Behavioral Requirements: The Play-In/Play-Out Approach" by David Harel and Rami Marelly

(from pgs. 4-6 of "Specifying and Executing...")

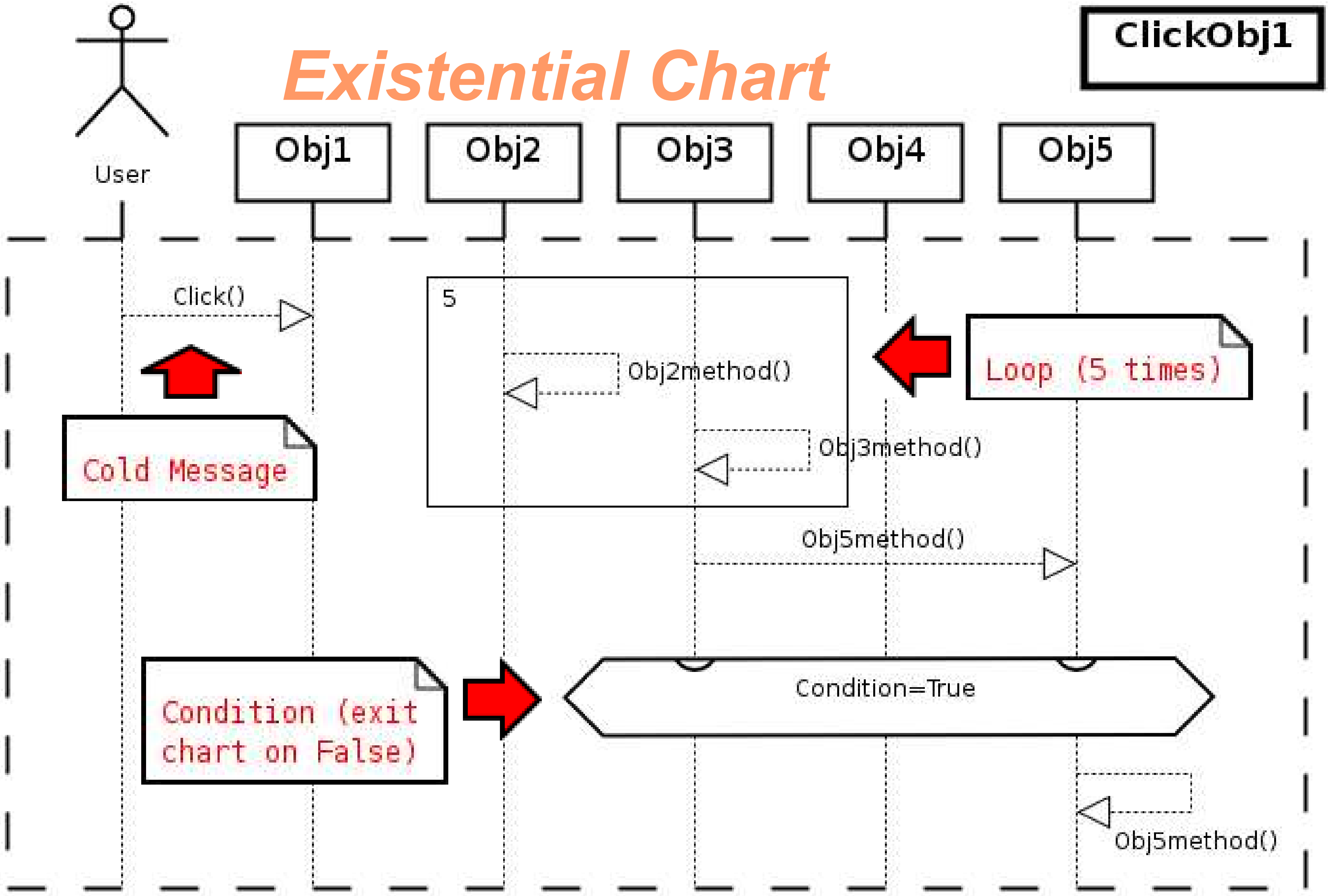
Recap of LSCs

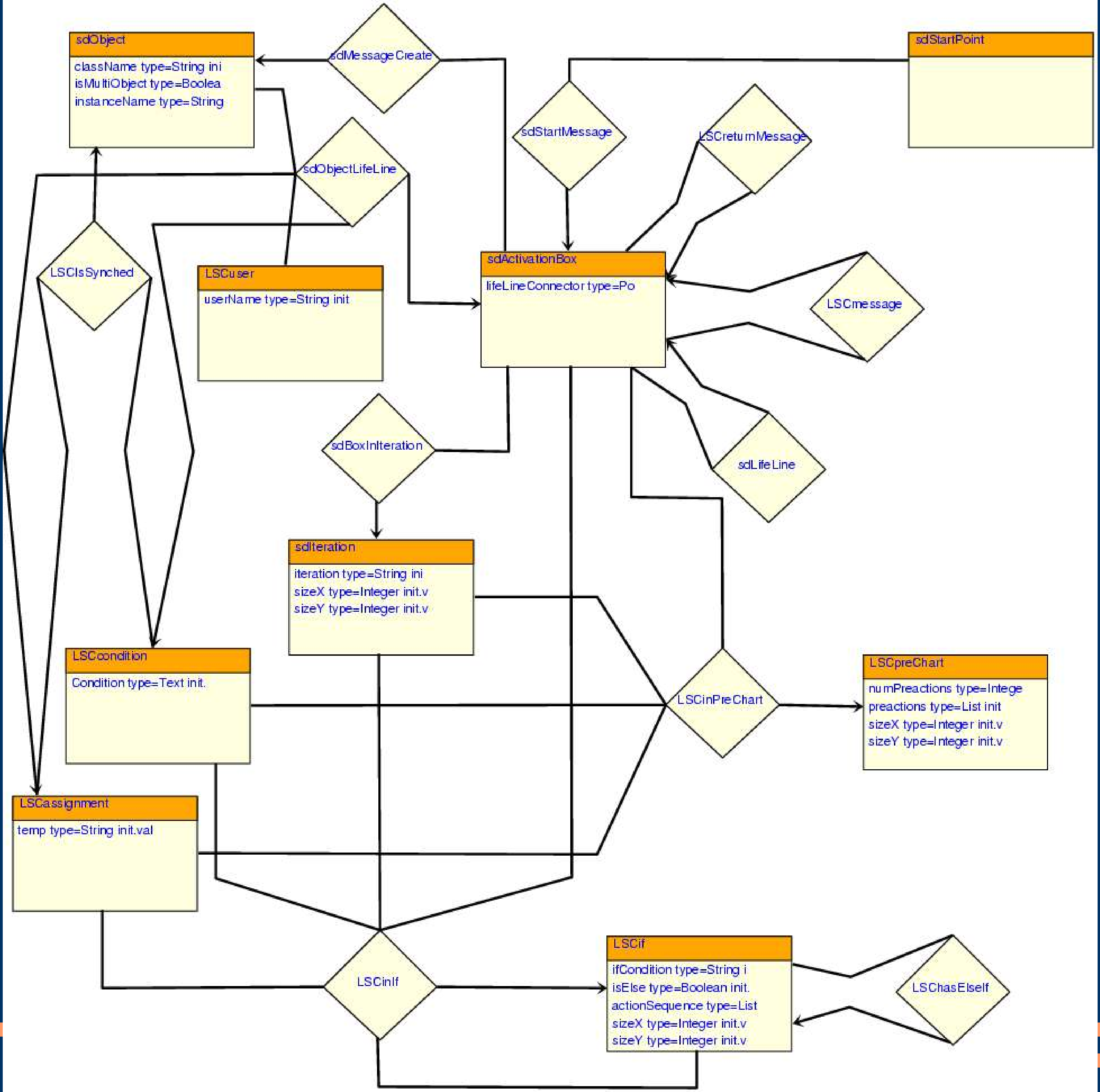
- Modified MSCs (Message Sequence Charts)

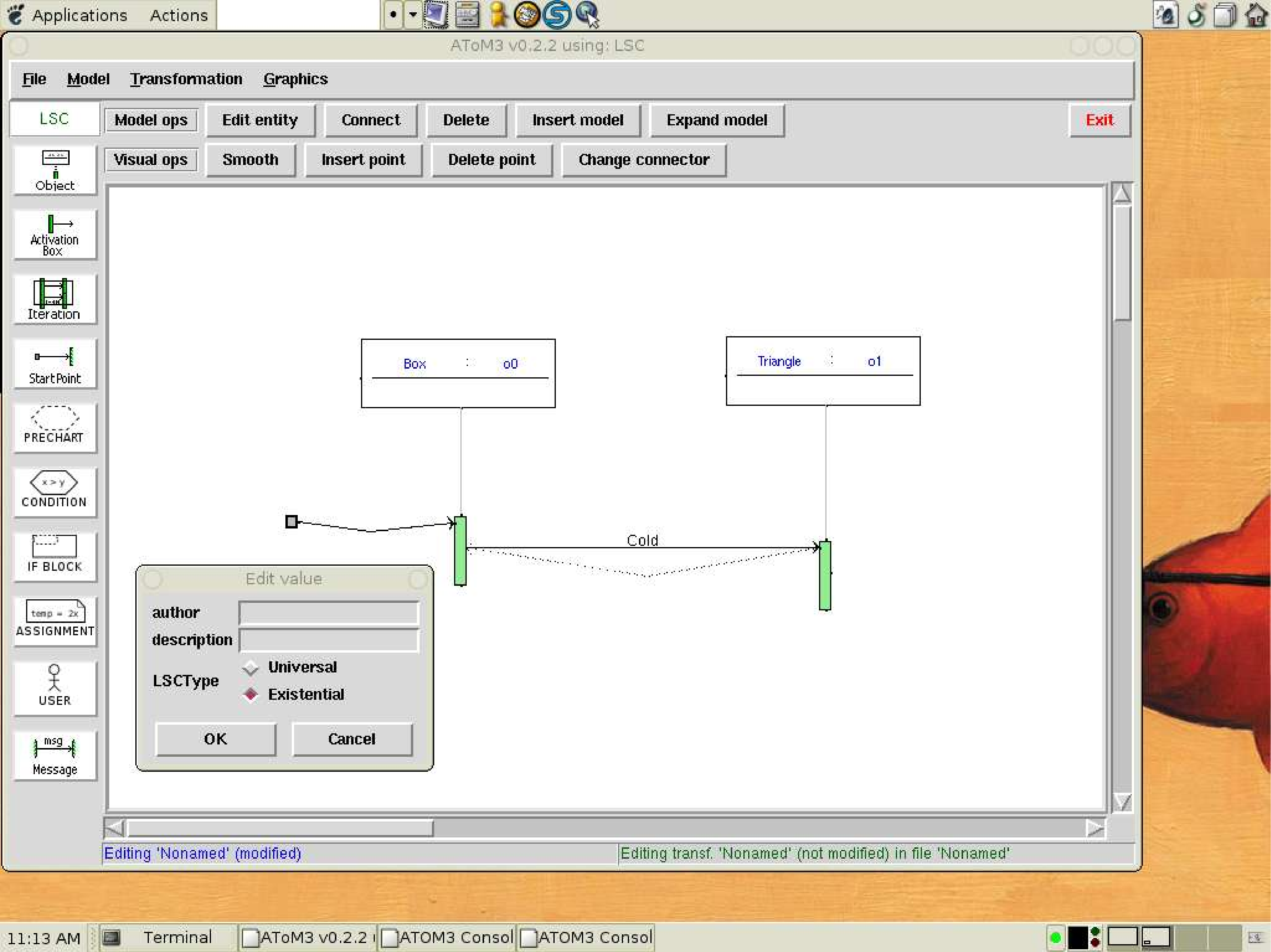
- Two different kinds of LSCs:
  - Universal (Solid Border)
    - Model system reactions that *must* happen
    - Pre-Chart is condition for main chart actions
    - Exiting these prematurely causes a system error/crash
    - Drive system execution during Play-Out
  - Existential (Dashed Border)
    - Model system reactions that *may* happen
    - Must be able to run to completion in at least one scenario
    - Monitored during Play-Out


ClickObj1
Universal Chart
User
Obj1
Obj2
Obj3
Obj4
Obj5
Click()
if ( )
Obj4method()
Pre-Chart
Temp:=Obj1.val + Obj2.val
Assignment
then {
}
Condition=True
if ( )
then {
}
else if ( ) {
}
Hot Message
Obj3method()
Condition (system error on False)
Condition=True

Existential Chart




sdObject
className type=String ini
isMultiObject type=Boolea
instanceName type=String
sdMessageCreate
sdStartPoint
sdStartMessage
LSCreturnMessage
sdObjectLifeLine
LSCIsSynched
LSCuser
userName type=String init
sdActivationBox
lifeLineConnector type=Po
LSCmessage
sdBoxInIteration
sdLifeLine
sdIteration
iteration type=String ini
sizeX type=Integer init.v
sizeY type=Integer init.v
LSCcondition
Condition type=Text init.
LSCinPreChart
LSCpreChart
numPreactions type=Intege
preactions type=List init
sizeX type=Integer init.v
sizeY type=Integer init.v
LSCassignment
temp type=String init.val
LSCinIf
LSCif
ifCondition type=String i
isElse type=Boolean init.
actionSequence type=List
sizeX type=Integer init.v
sizeY type=Integer init.v
LSChasElseIf



Edit value

author

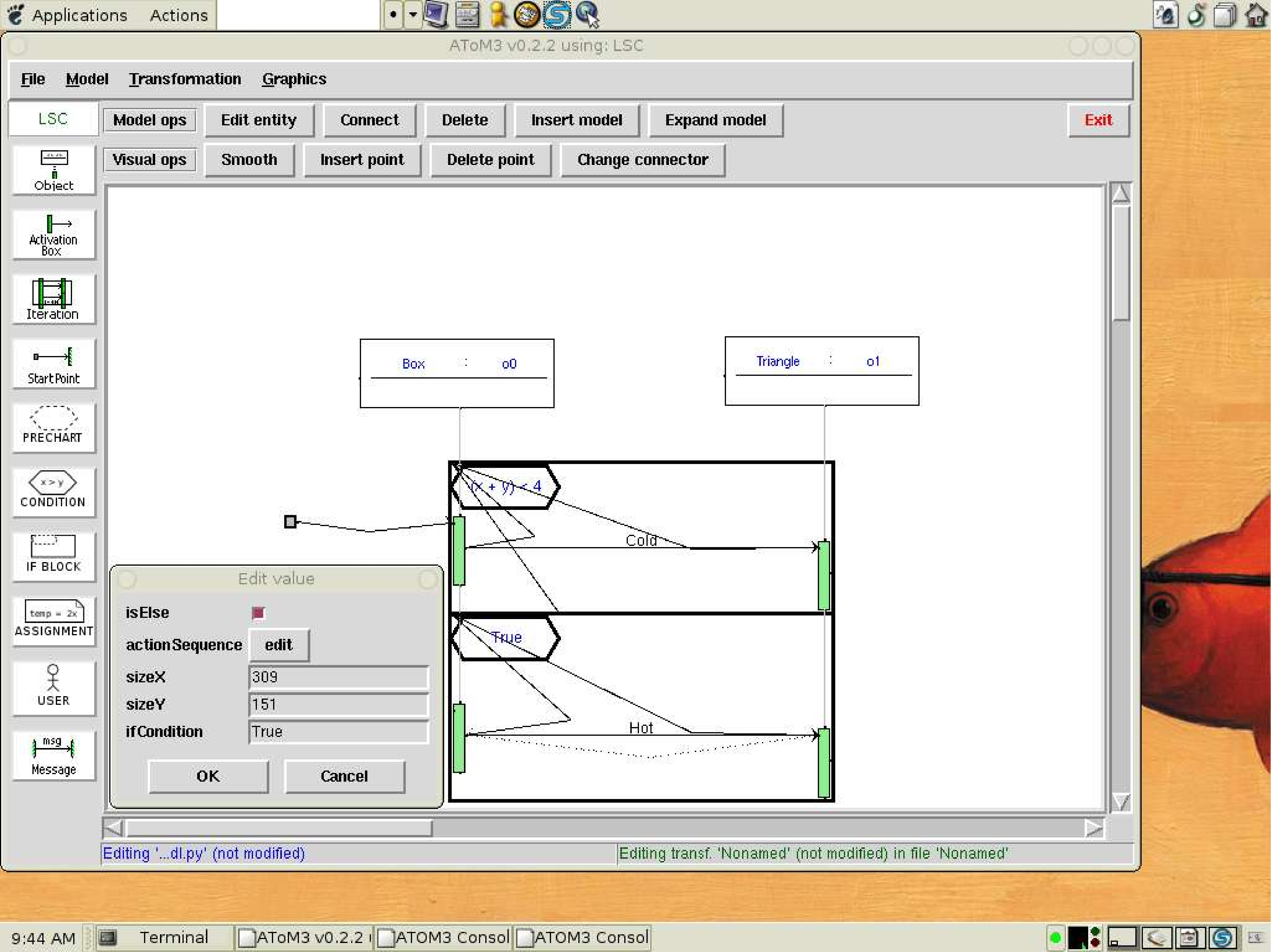
description

LSCType: Universal [✓ Existential]

OK Cancel

Editing 'Nonamed' (modified)

Editing transf. 'Nonamed' (not modified) in file 'Nonamed'

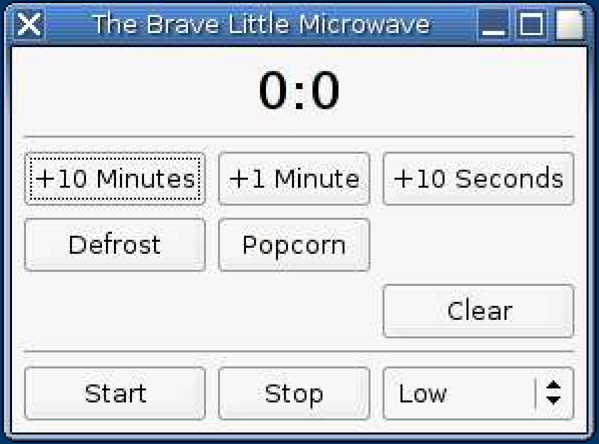


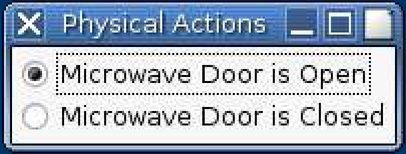
isElse	🟥
actionSequence	edit
sizeX	309
sizeY	151
ifCondition	True

Editing '...dl.py' (not modified)

Editing transf. 'Nonamed' (not modified) in file 'Nonamed'

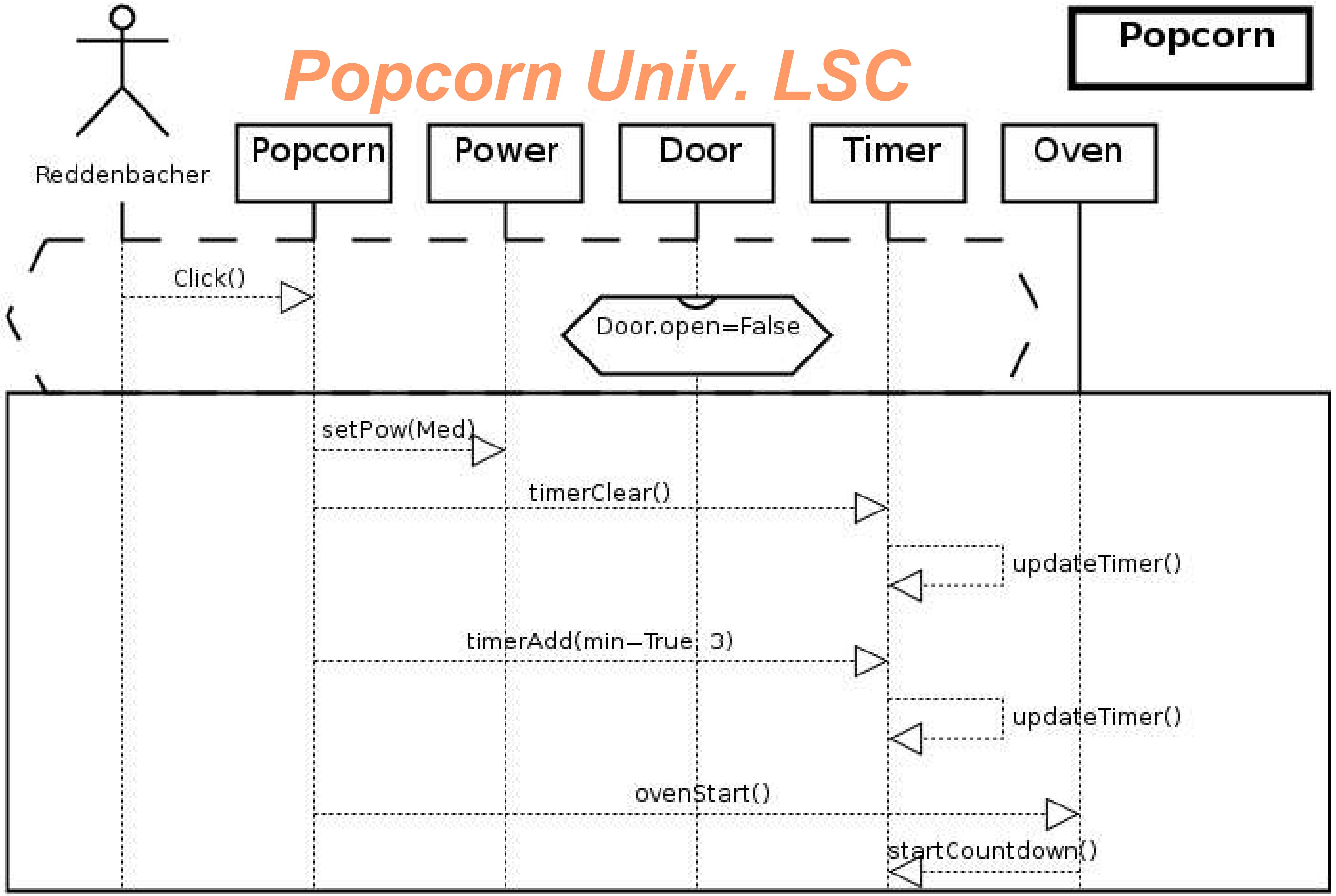
LSC to Statechart Transformation

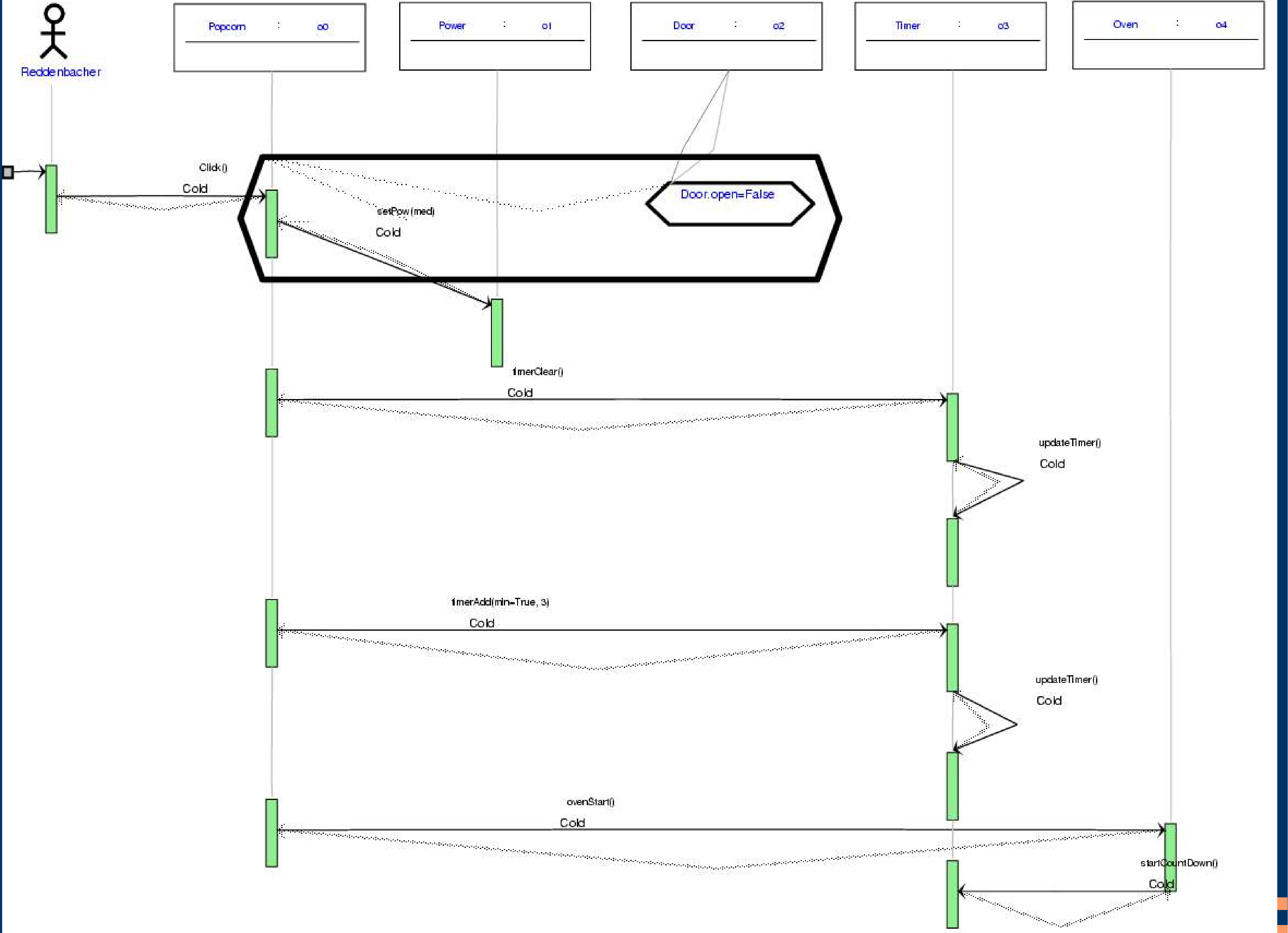




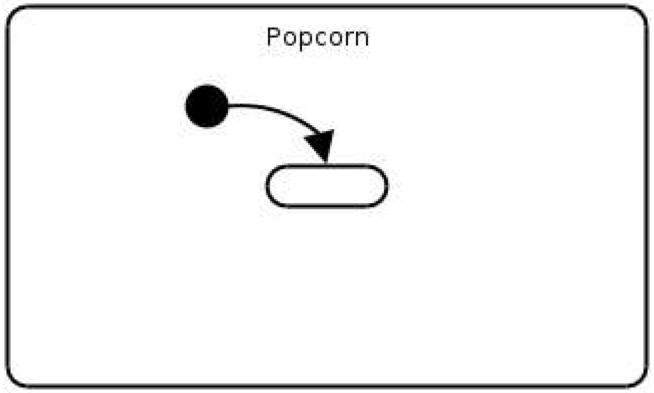
- Goal of this example: transform LSC for the 'Popcorn' button into language of Statecharts
- We'll use multiple Statecharts

Popcorn Univ. LSC

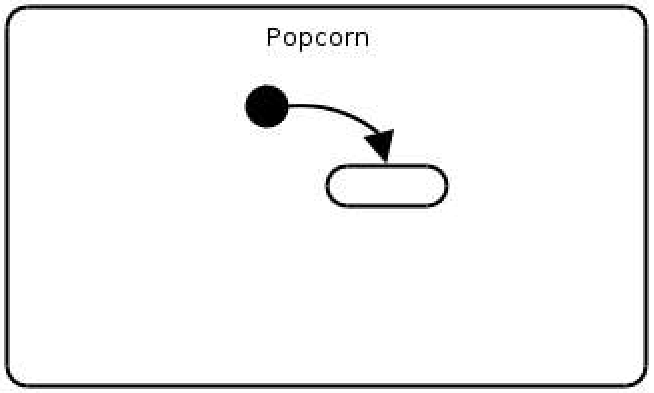



Reddenbacher
Popcorn : o0
Power : o1
Door : o2
Timer : o3
Oven : o4
Click()
Cold
Door.open=False
setPow(med)
Cold
timerClear()
Cold
updateTimer()
Cold
timerAdd(min=True, 3)
Cold
updateTimer()
Cold
ovenStart()
Cold
startCountDown()
Cold

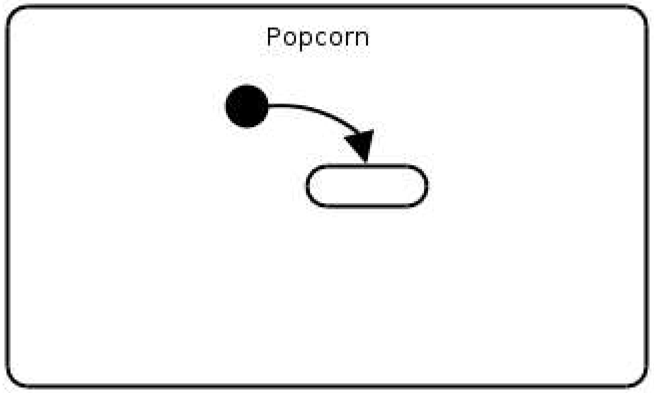
Popcorn



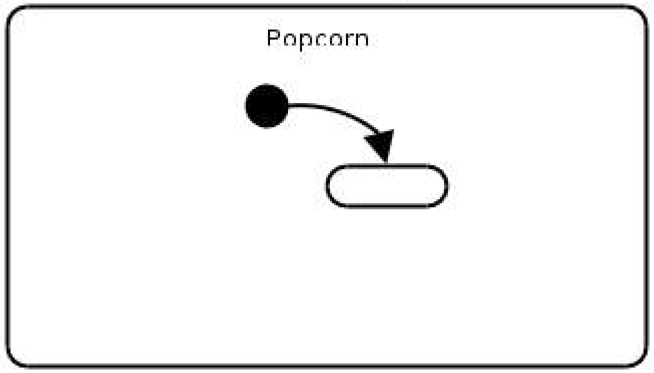
Power



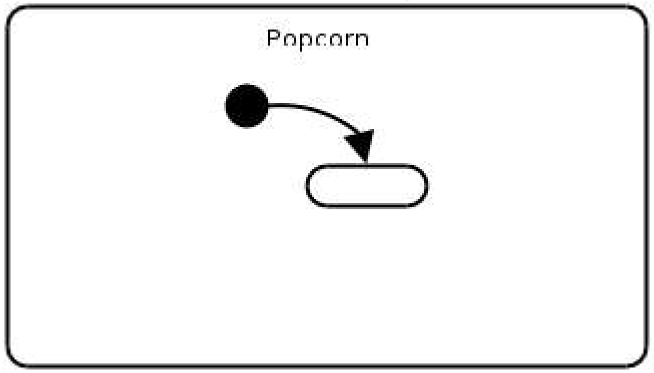
Timer

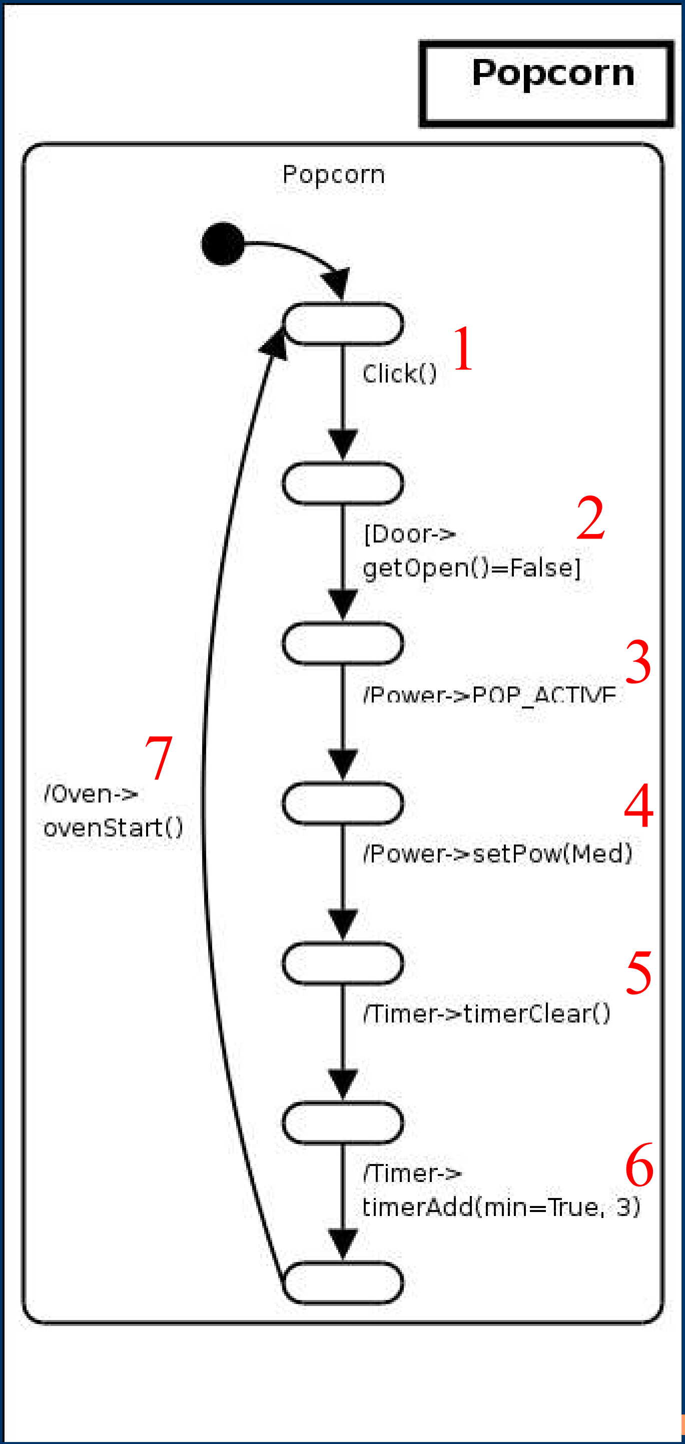


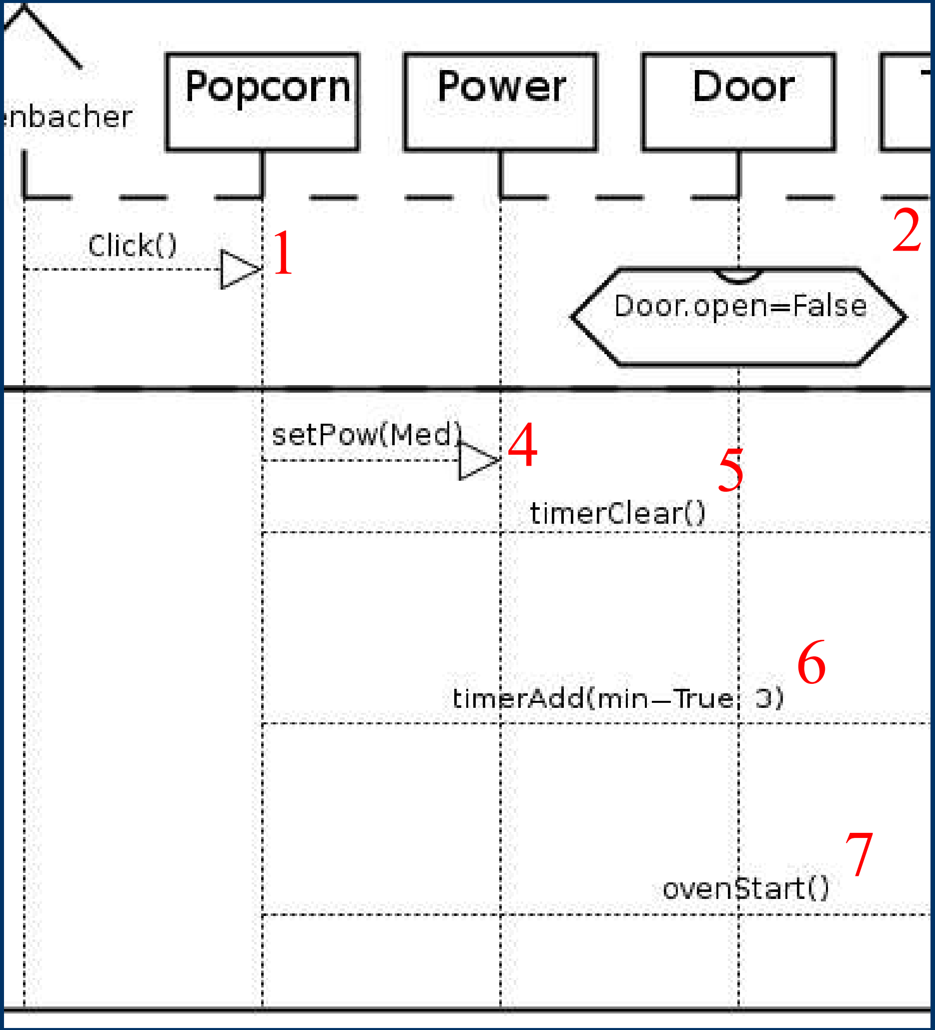
Door



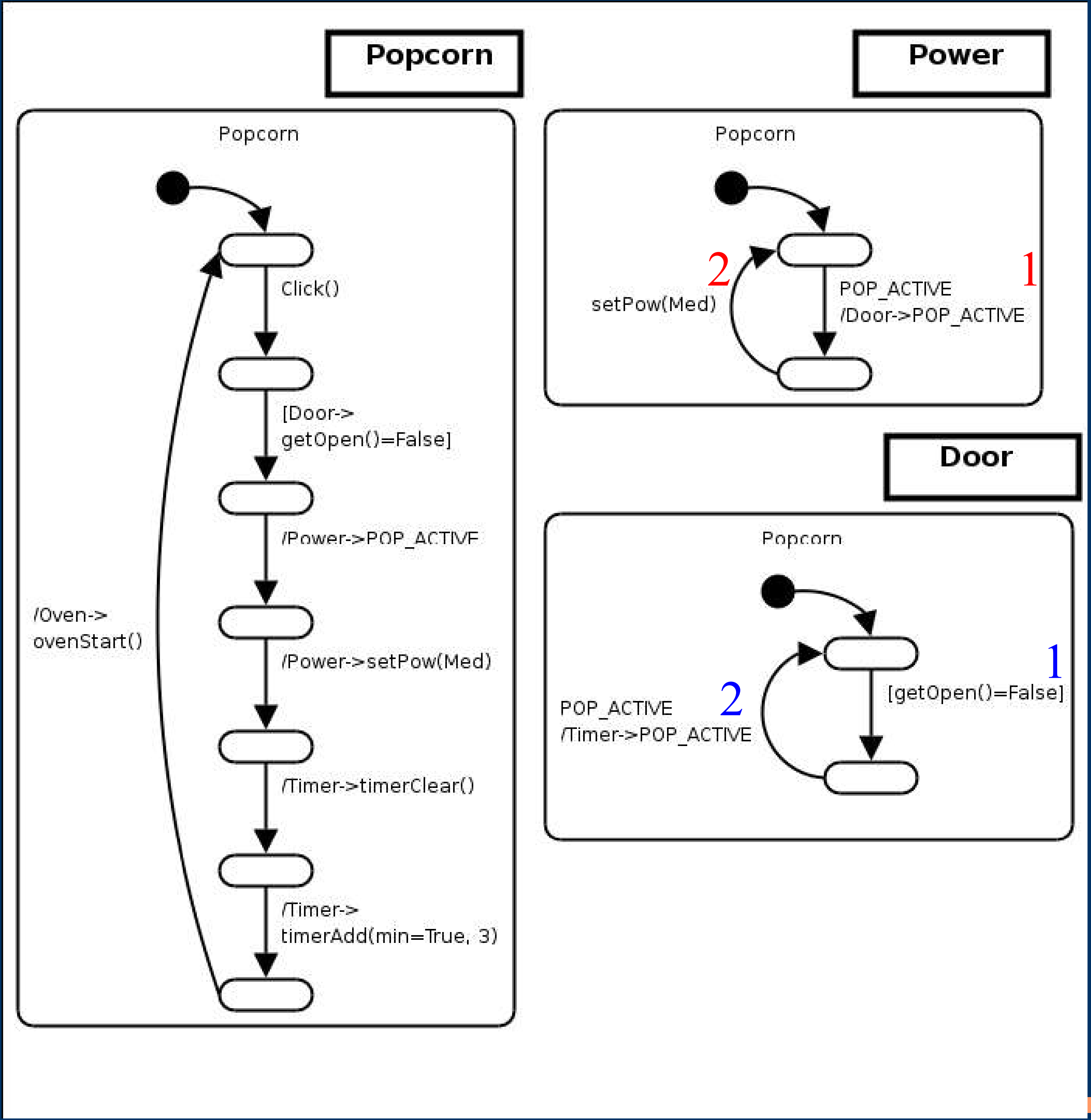
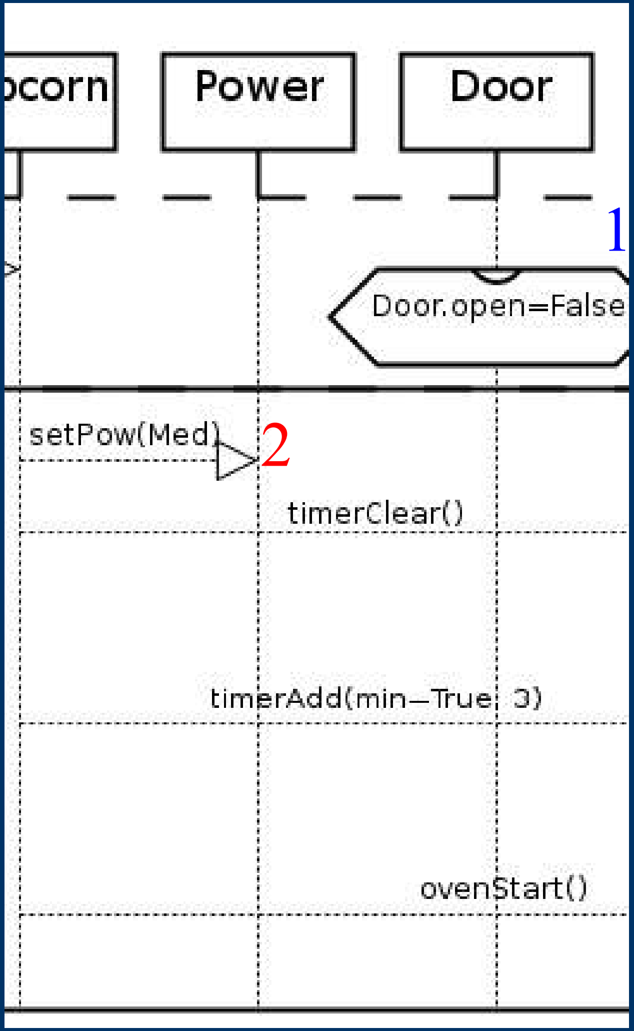
Oven



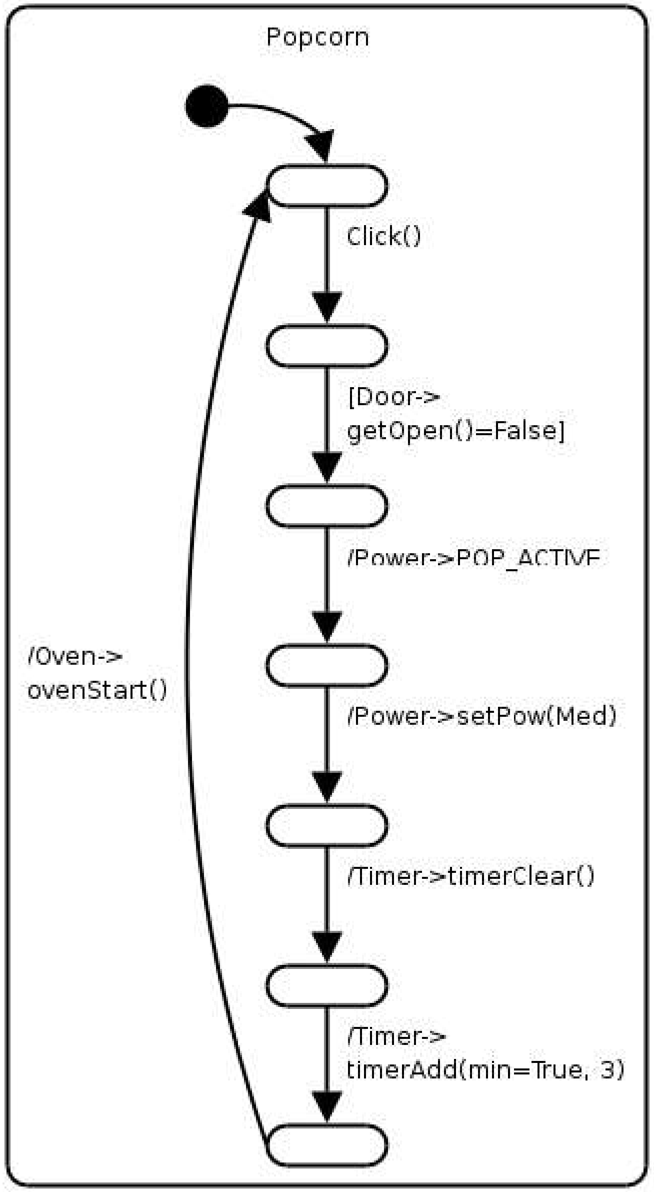




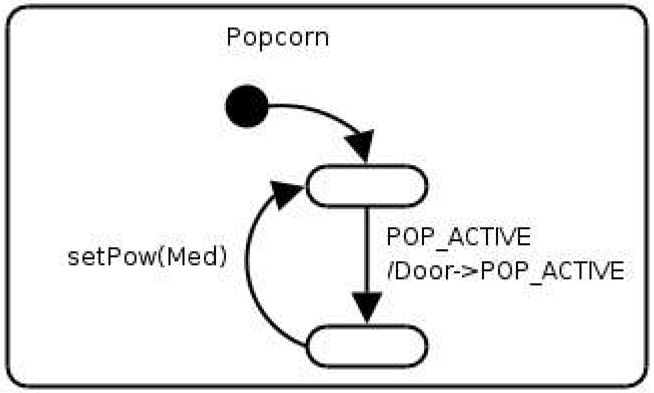
Note transition 3: “/Power->POP_ACTIVE” starts chain of object notification


Popcorn
Popcorn
Click()
[Door->getOpen()=False]
/Power->POP_ACTIVE
/Power->setPow(Med)
/Timer->timerClear()
/Timer->timerAdd(min=True, 3)
/Oven->ovenStart()
Power
Popcorn
2
1
setPow(Med)
POP_ACTIVE /Door->POP_ACTIVE
Door
Popcorn
1
2
[getOpen()=False]
POP_ACTIVE /Timer->POP_ACTIVE

pcorn
Power
Door
1
Door.open=False
setPow(Med)
2
timerClear()
timerAdd(min=True, 3)
ovenStart()

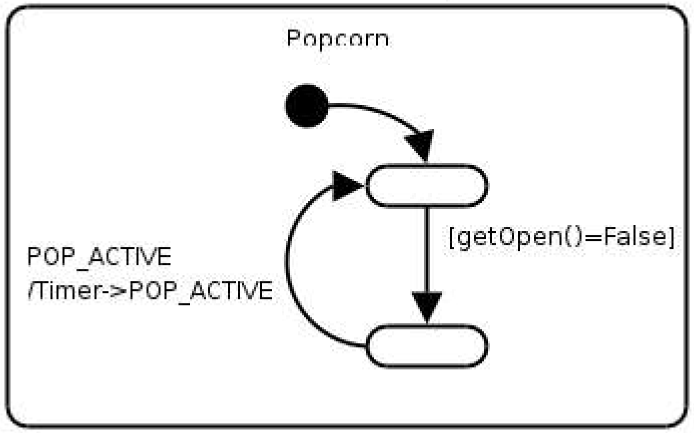
Popcorn



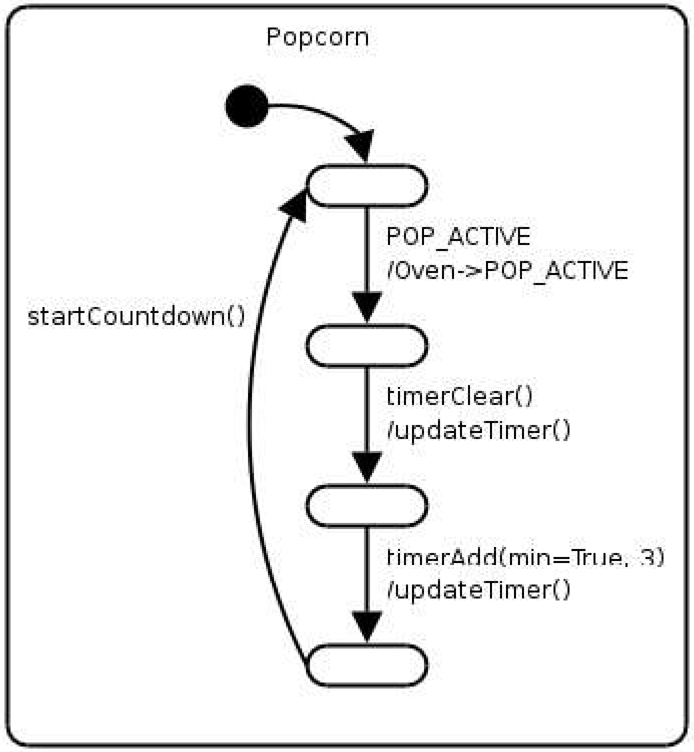
Power



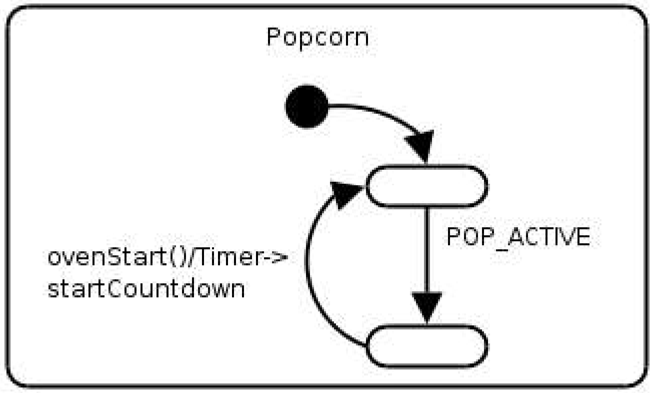
Door



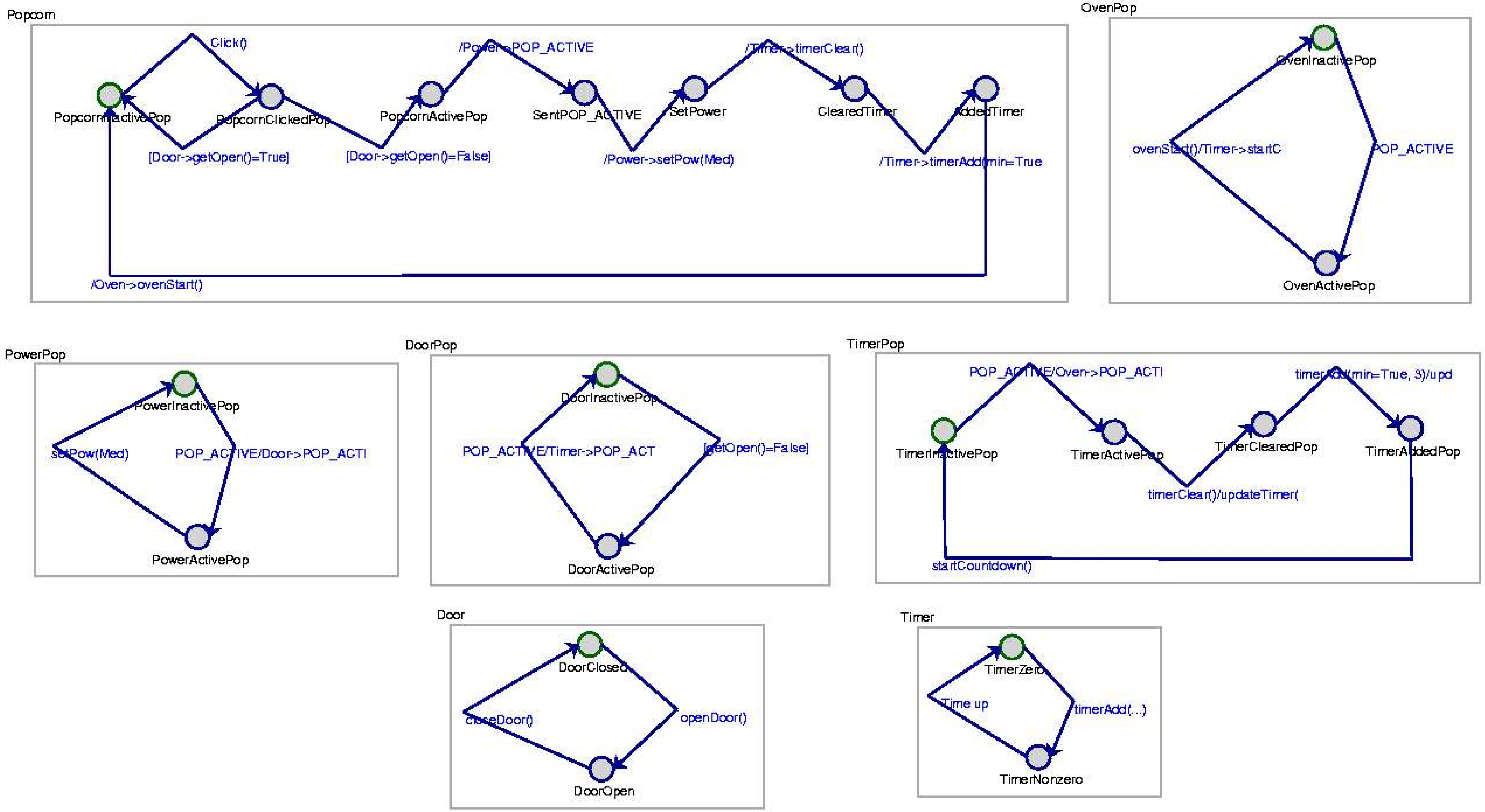
Timer


Popcorn
POP_ACTIVE /Oven->POP_ACTIVE
timerClear() /updateTimer()
timerAdd(min=True, 3) /updateTimer()
startCountdown()

Oven



Popcorn DChart



LSC to Statechart Transformation

1. Create one statechart for each unique object in Universal LSC
2. For each statechart:
    1. Create default state
    2. Create one state for each action requiring the object
    3. Chain states together with transitions.
    4. Create one transition from state at end of chain to default state
    5. Label transitions with above actions and “ACTIVE” notification

LSC to Statechart Transformation

3. Use orthogonal components if object is in more than one Universal LSC*
4. Check Bad_{max}, set of all supercuts without successors or that lead to those without successors*

*We didn't do these in the example

Transformed Microwave Functions

LSCs:

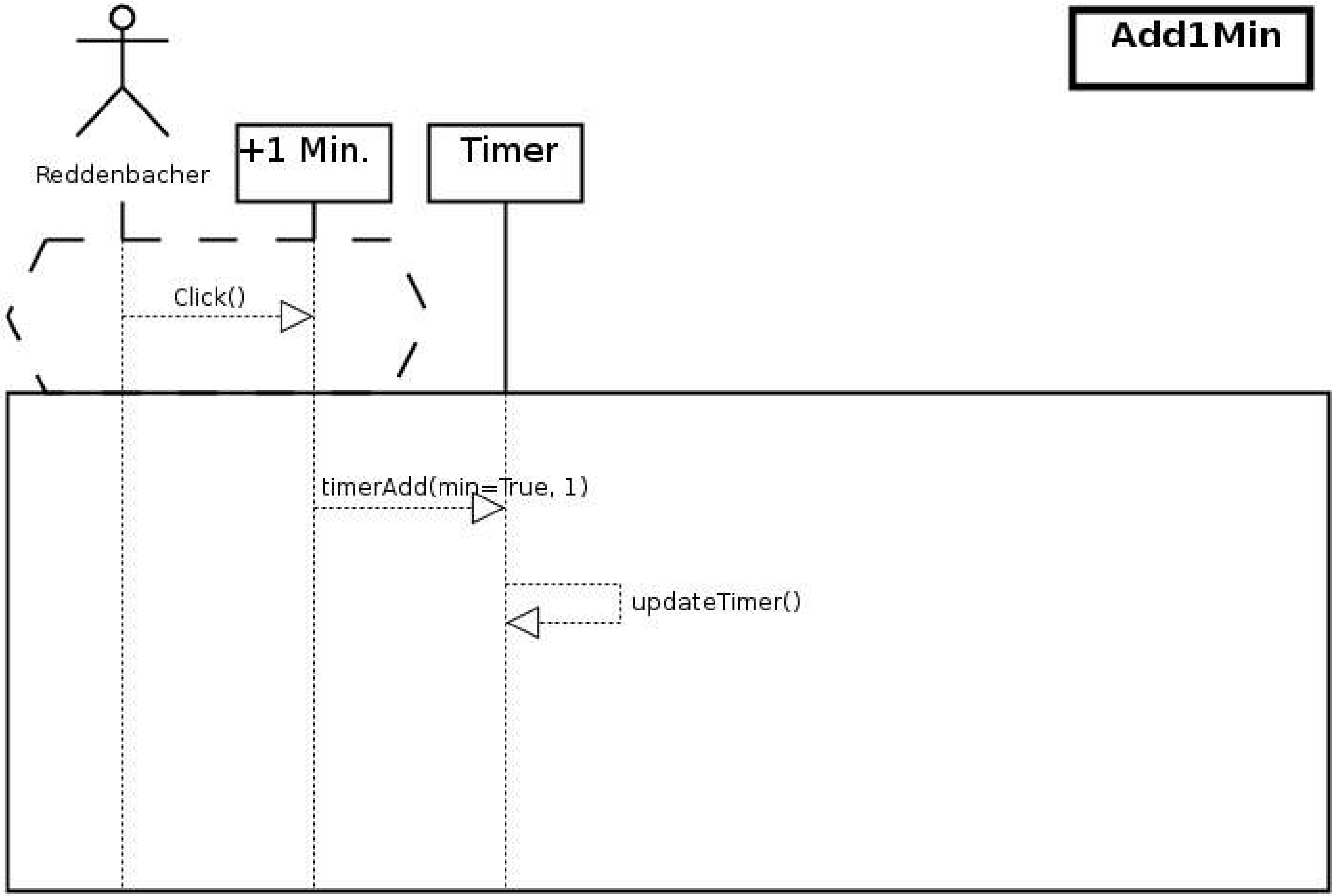
1. Add1Min
2. Clear
3. Popcorn
4. Defrost
5. Start
6. Stop
7. OpenDoorWhileOven Active

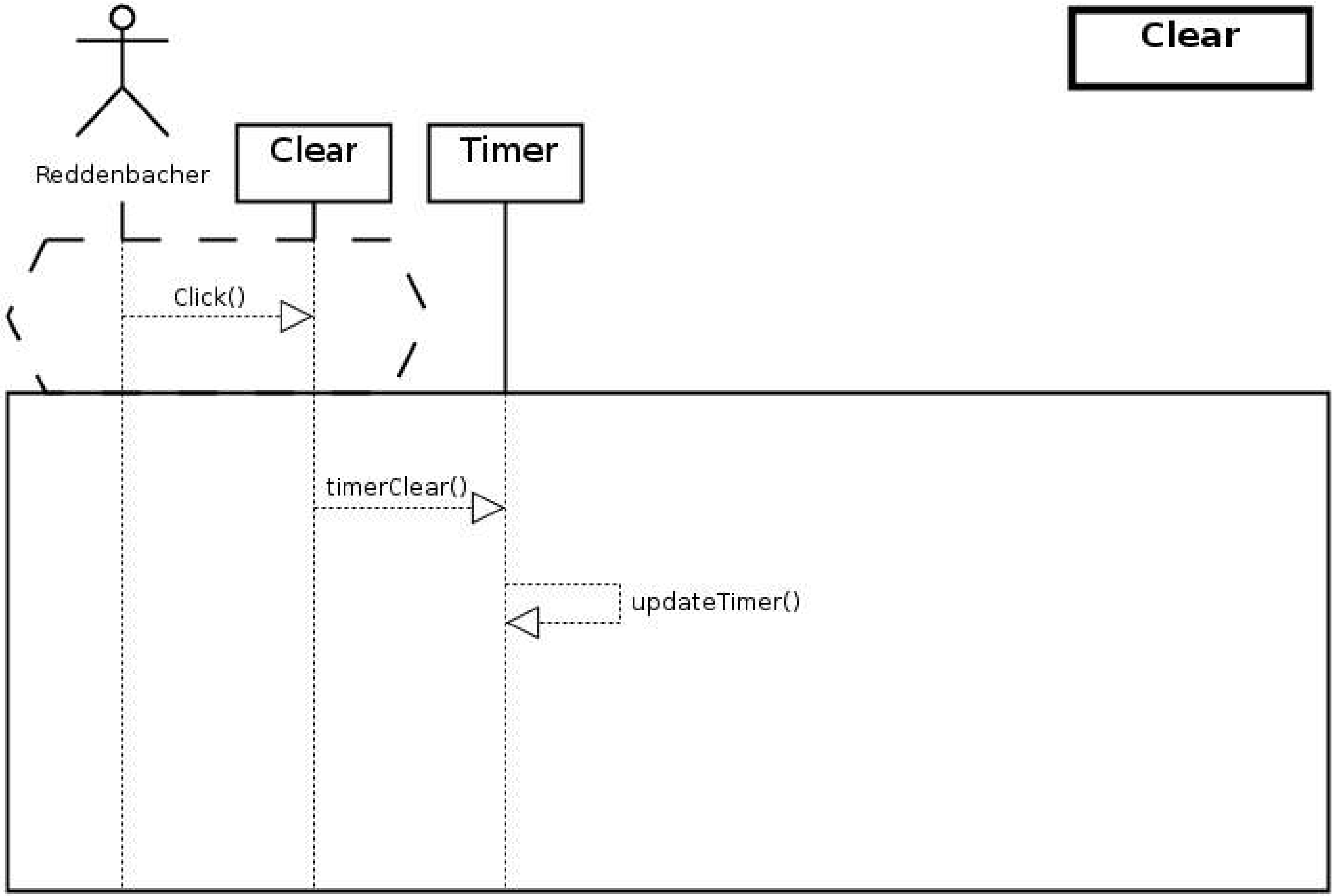
Statecharts:

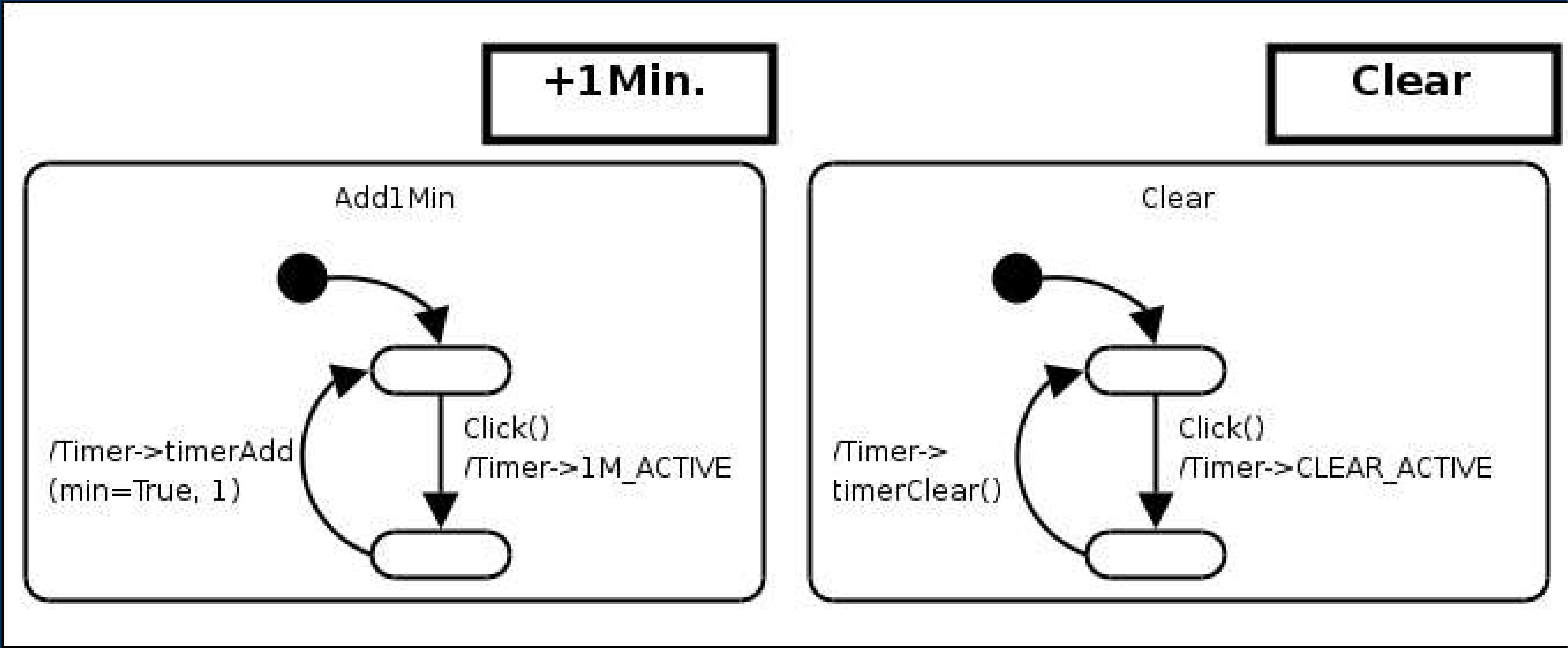
1. +1Min.
2. Clear
3. Popcorn
4. Defrost
5. Start
6. Stop
7. Power
8. Door
9. Timer
10. Oven

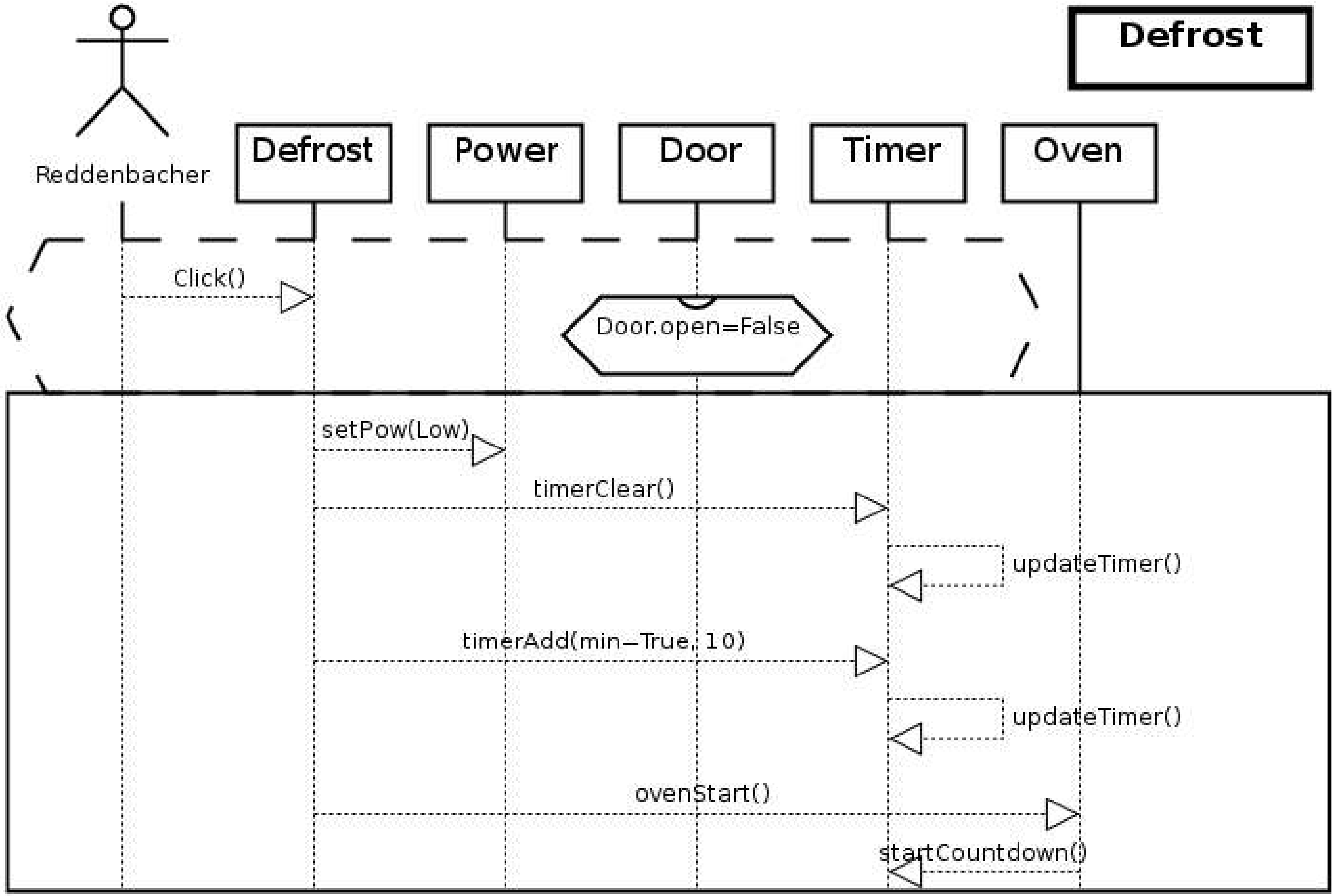
Results of Transformation

In the following statecharts, please assume all states before “ACTIVE” notices have transitions to the default states.


Add1Min
Reddenbacher
+1 Min.
Timer
Click()
timerAdd(min=True, 1)
updateTimer()

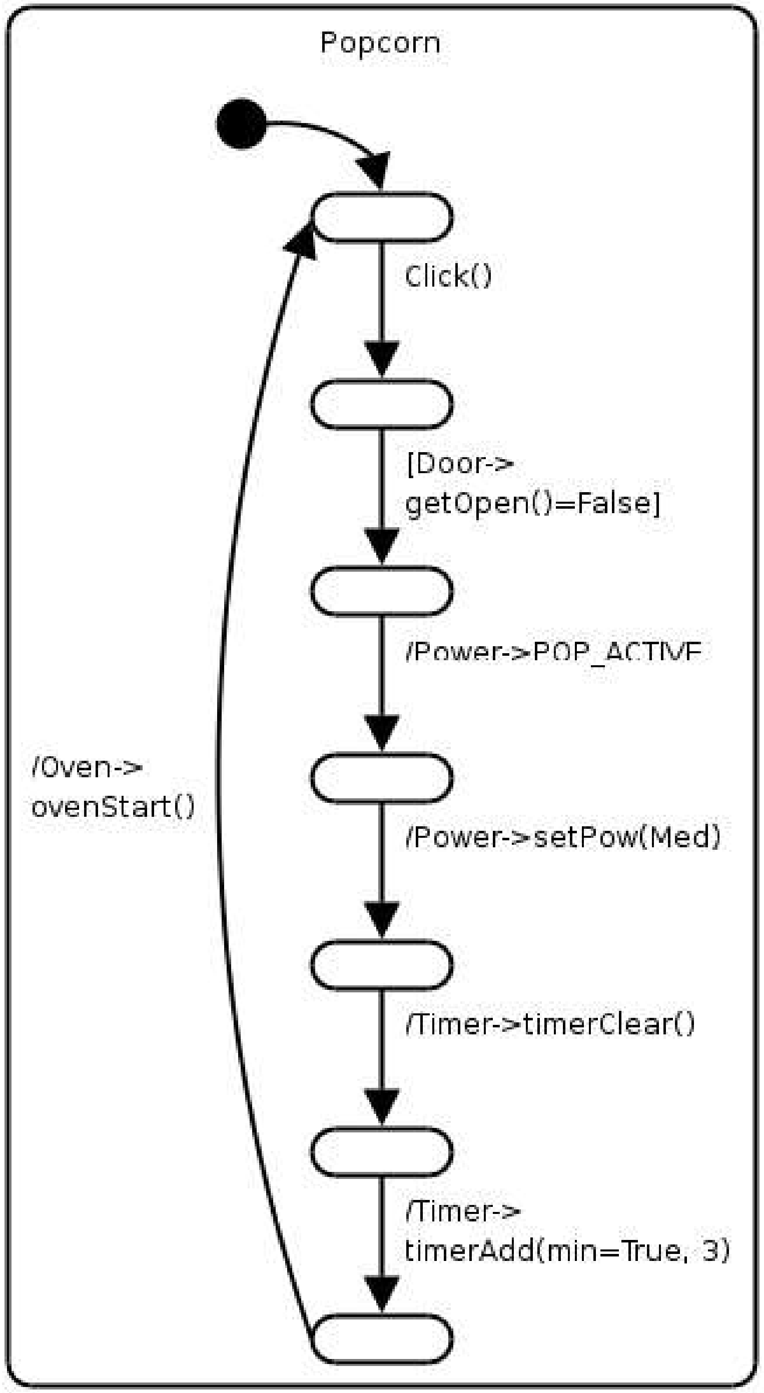

Clear
Reddenbacher
Clear
Timer
Click()
timerClear()
updateTimer()

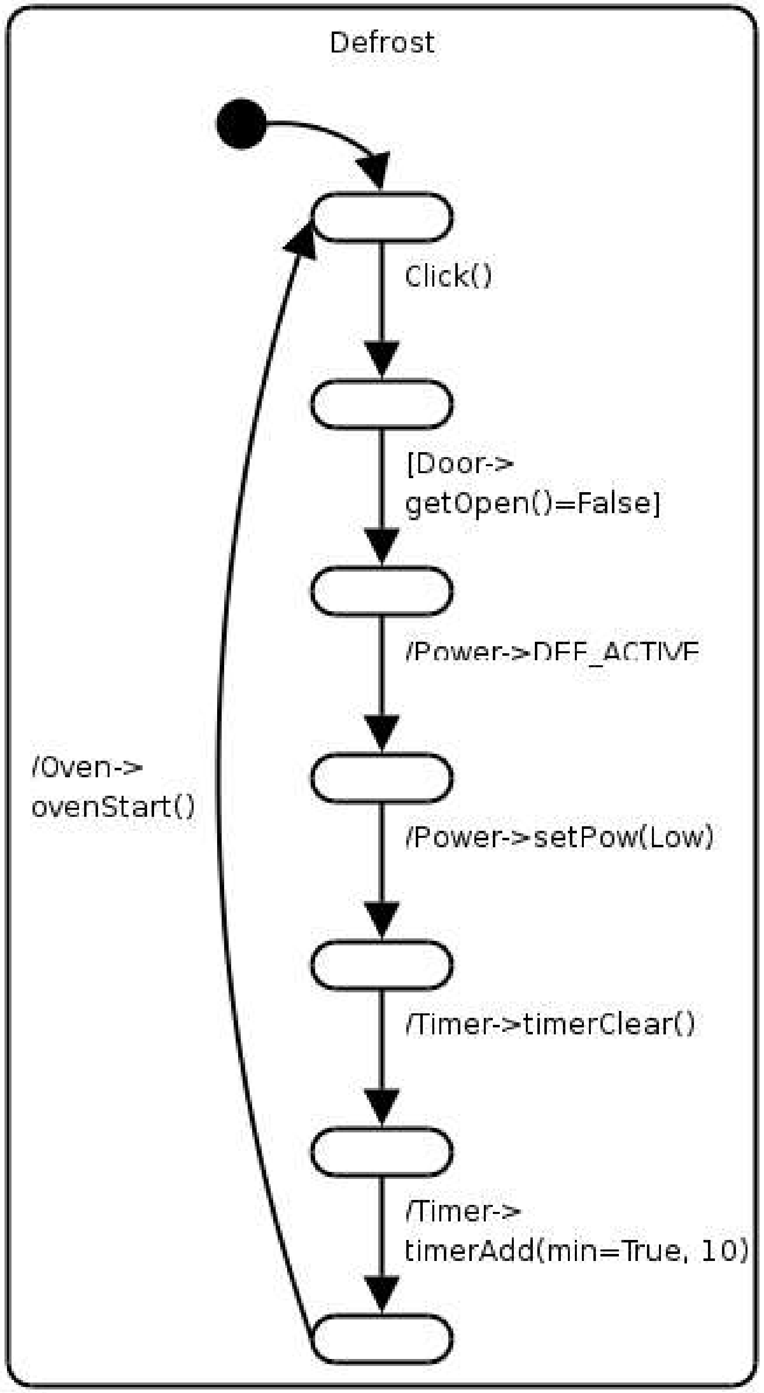

+1Min.
Clear
Add1Min
Click()
/Timer->1M_ACTIVE
/Timer->timerAdd
(min=True, 1)
Clear
Click()
/Timer->CLEAR_ACTIVE
/Timer->
timerClear()

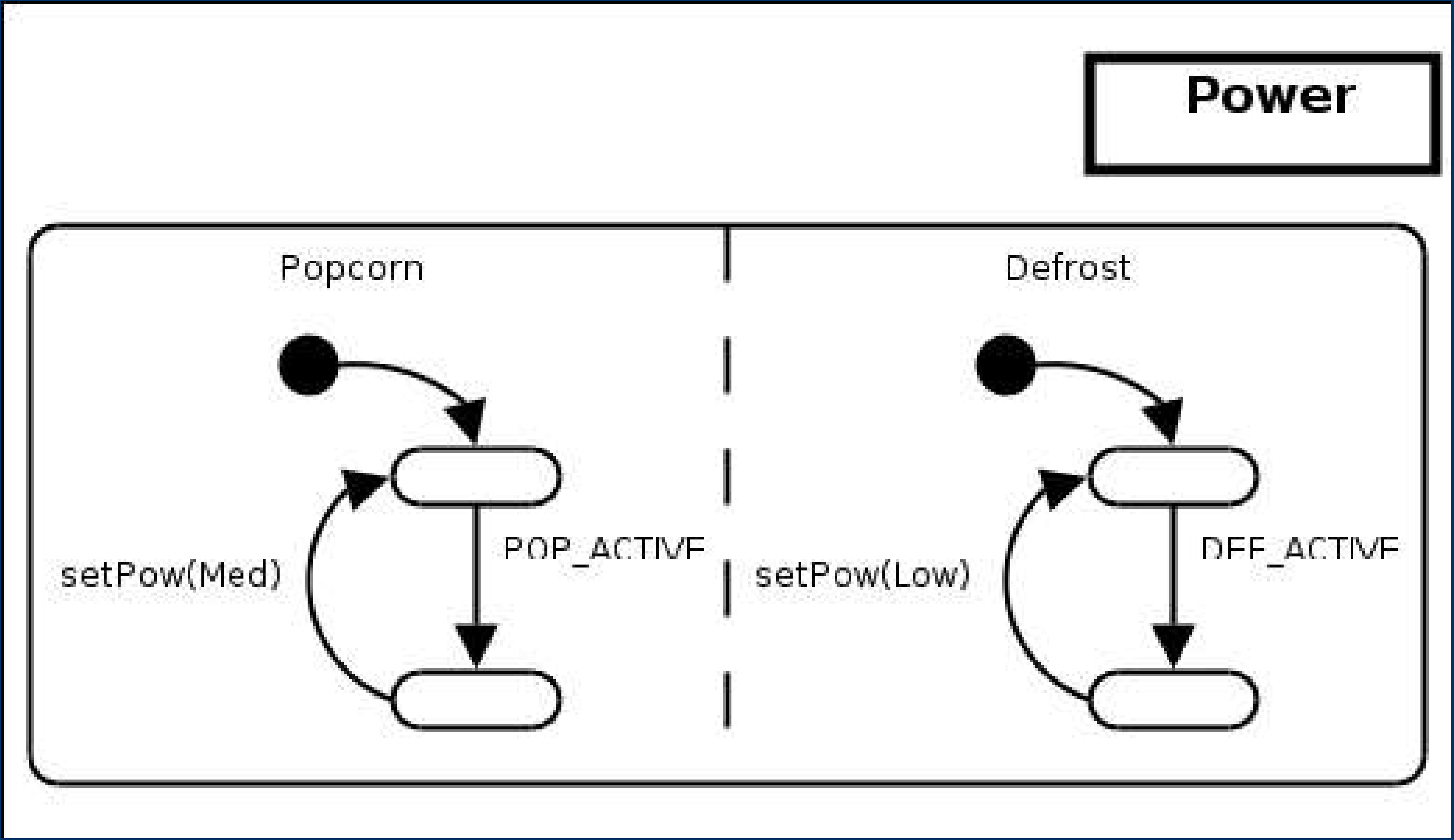

Defrost
Reddenbacher
Defrost
Power
Door
Timer
Oven
Click()
Door.open=False
setPow(Low)
timerClear()
updateTimer()
timerAdd(min=True, 10)
updateTimer()
ovenStart()
startCountdown()

Popcorn

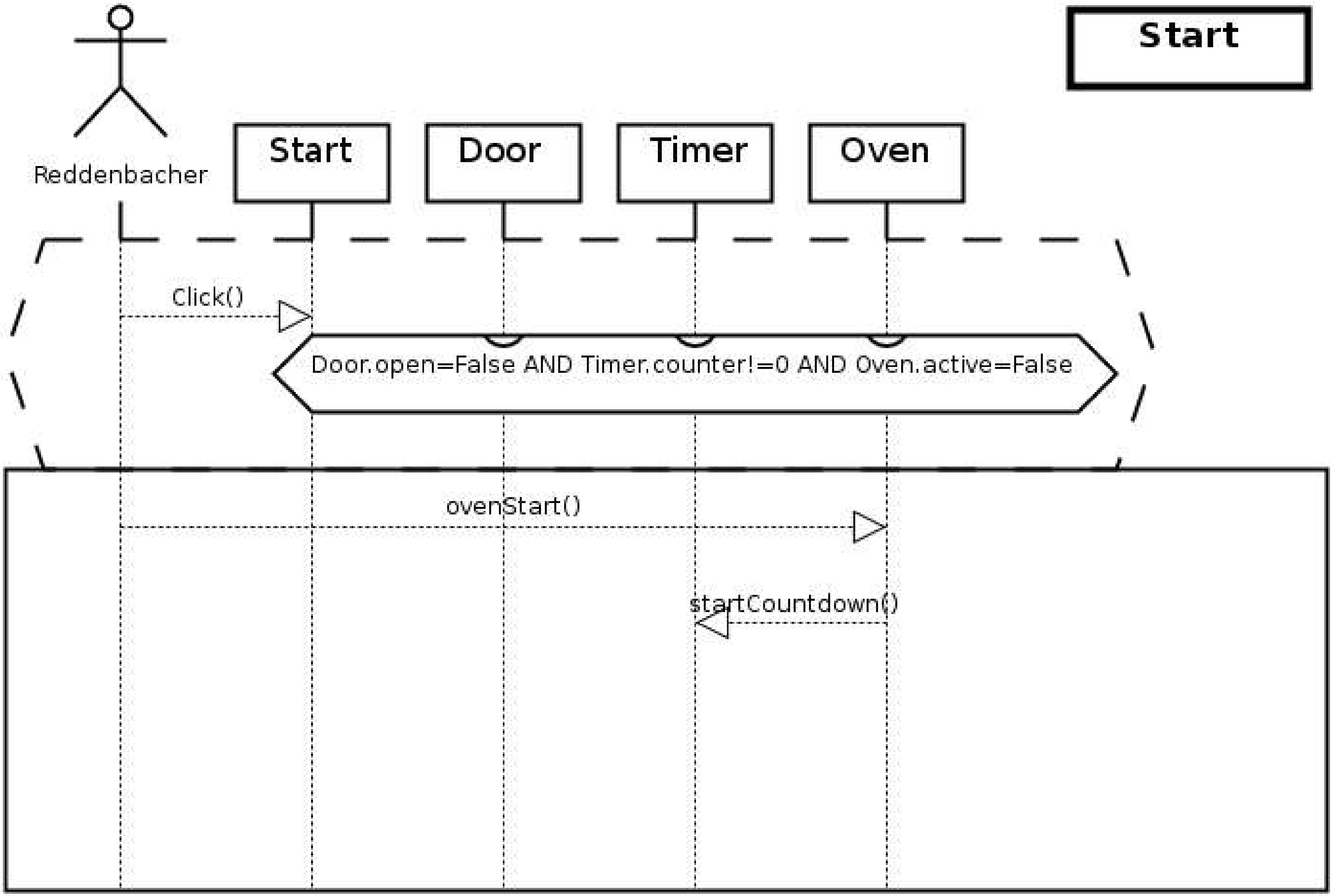
Defrost

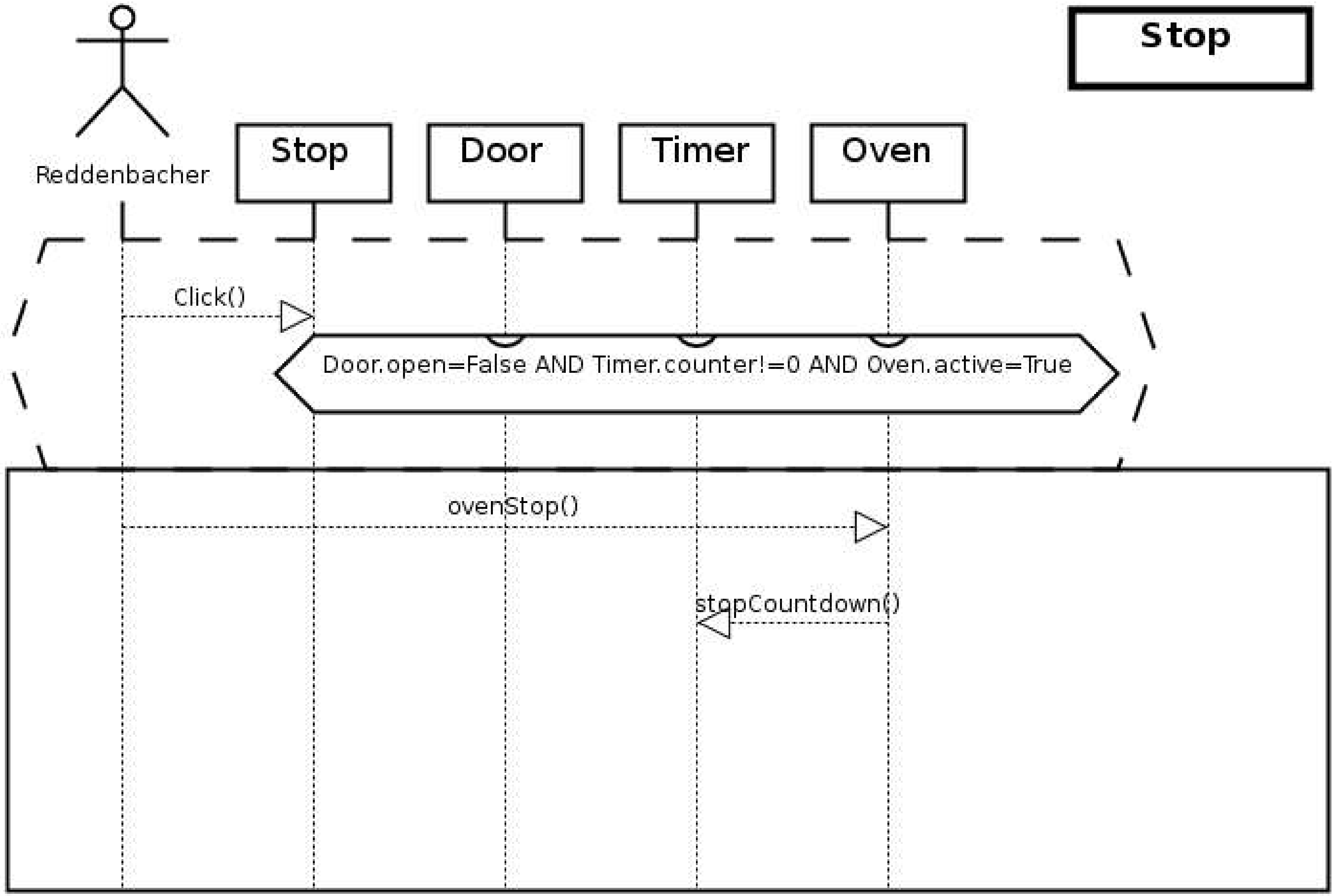




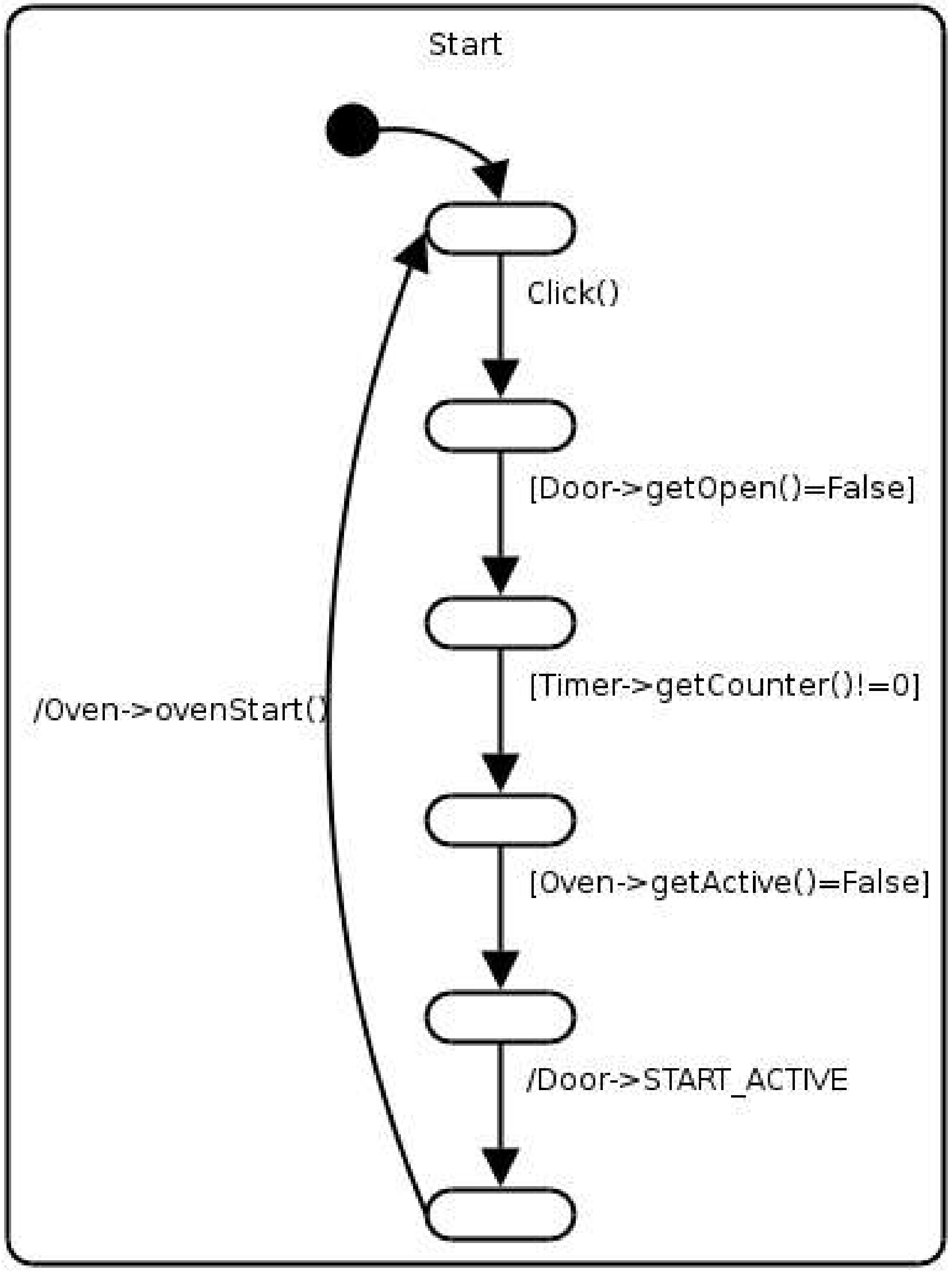

Power
Popcorn
Defrost
POP_ACTIVE
setPow(Med)
DEF_ACTIVE
setPow(Low)

Start

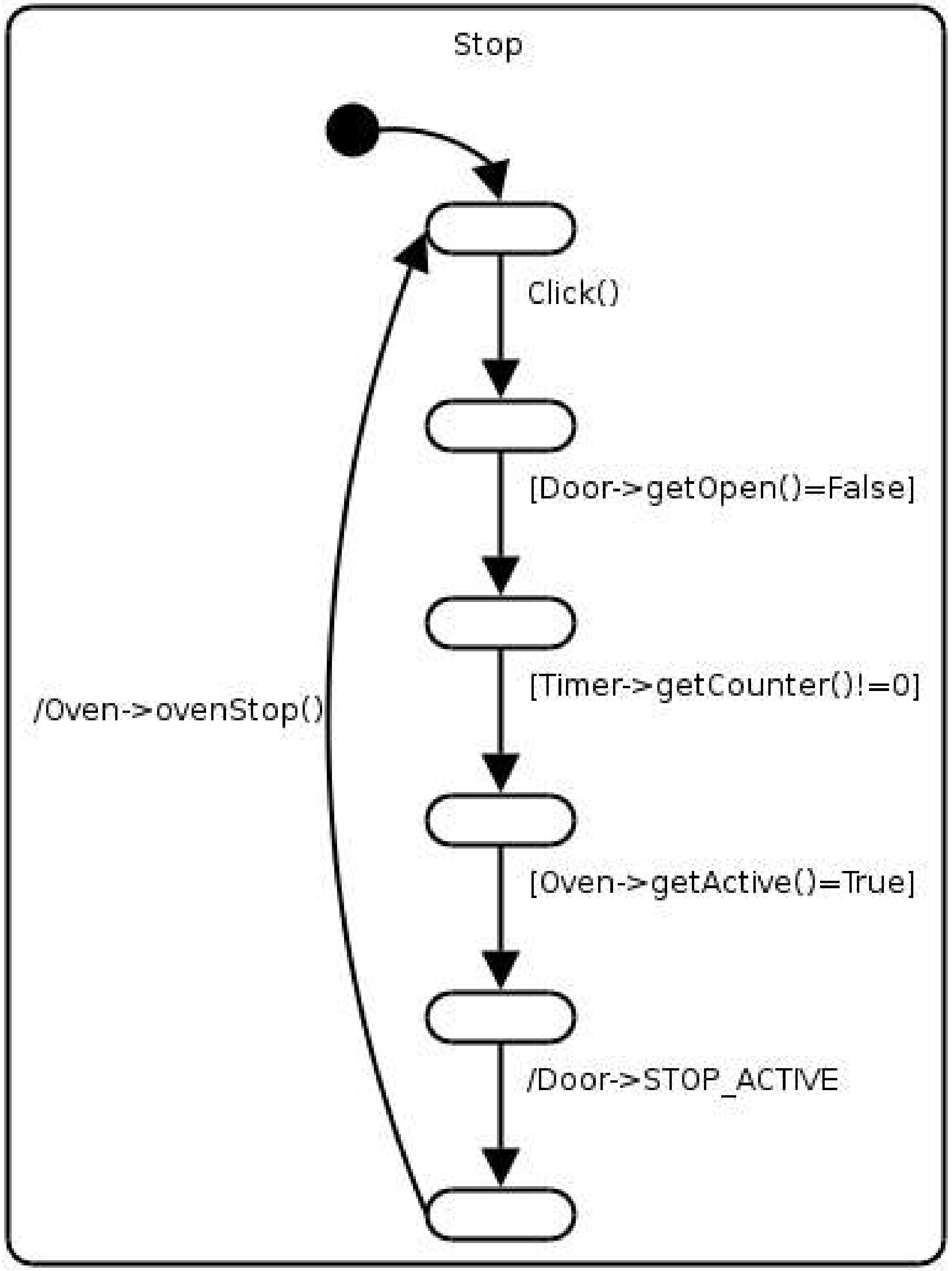

Reddenbacher
Start
Door
Timer
Oven
Click()
Door.open=False AND Timer.counter!=0 AND Oven.active=False
ovenStart()
startCountdown()

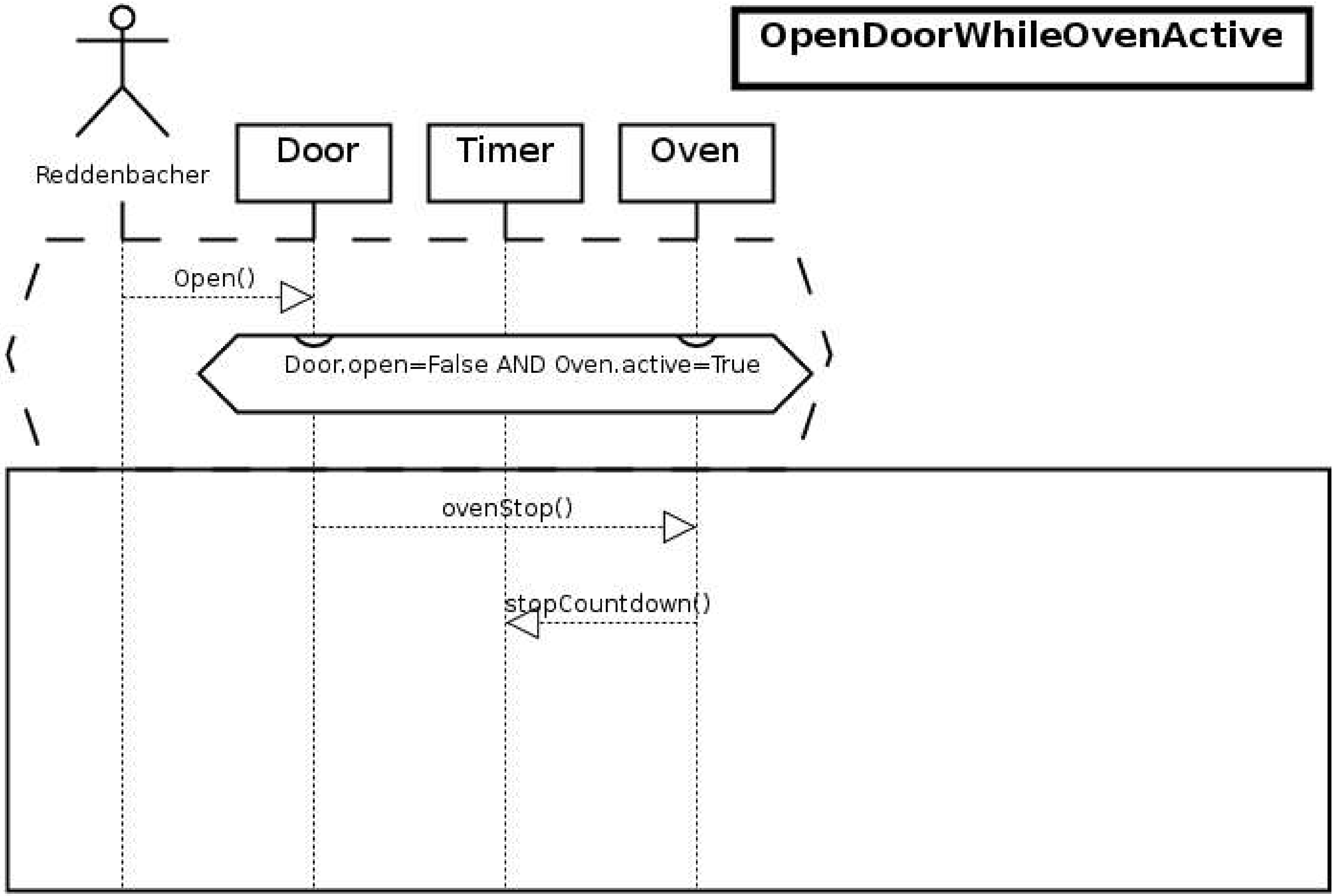

Stop
Reddenbacher
Stop
Door
Timer
Oven
Click()
Door.open=False AND Timer.counter!=0 AND Oven.active=True
ovenStop()
stopCountdown()

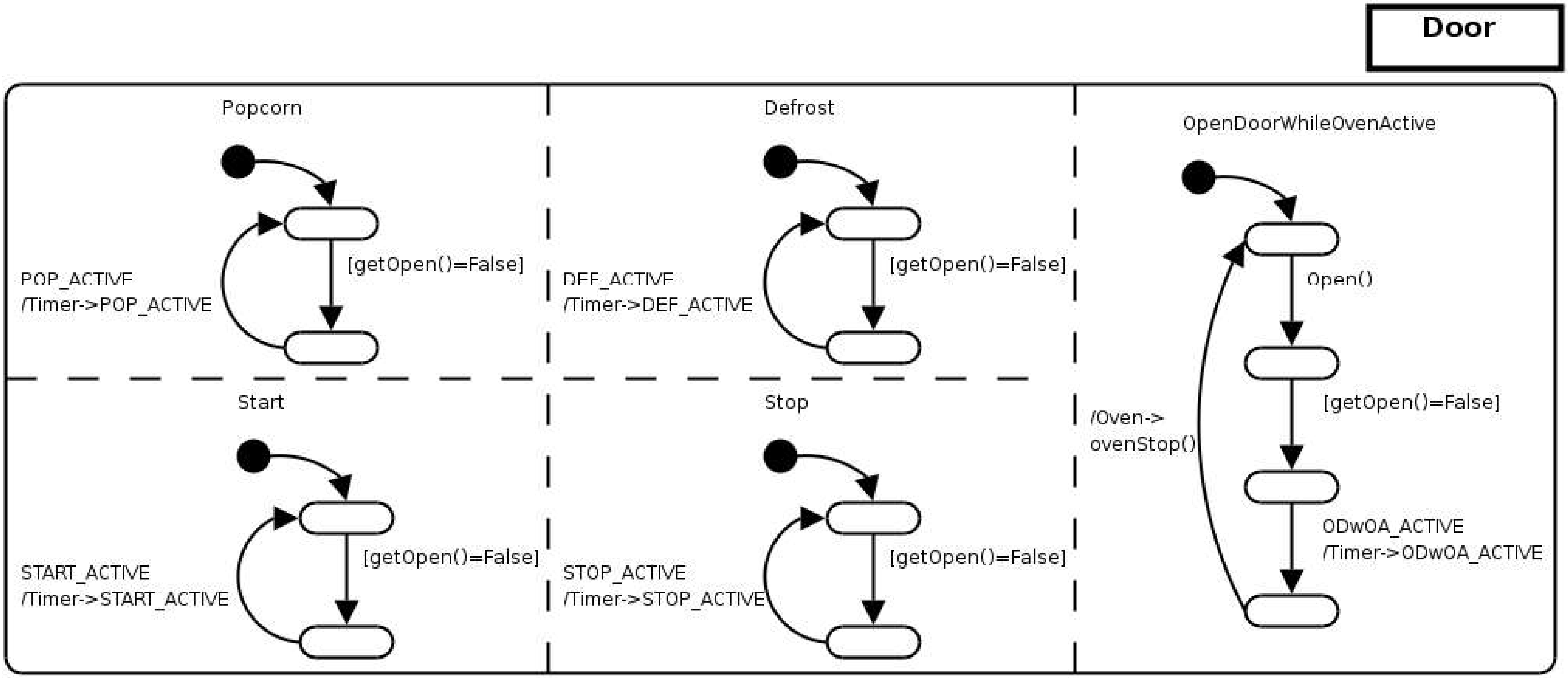
Start

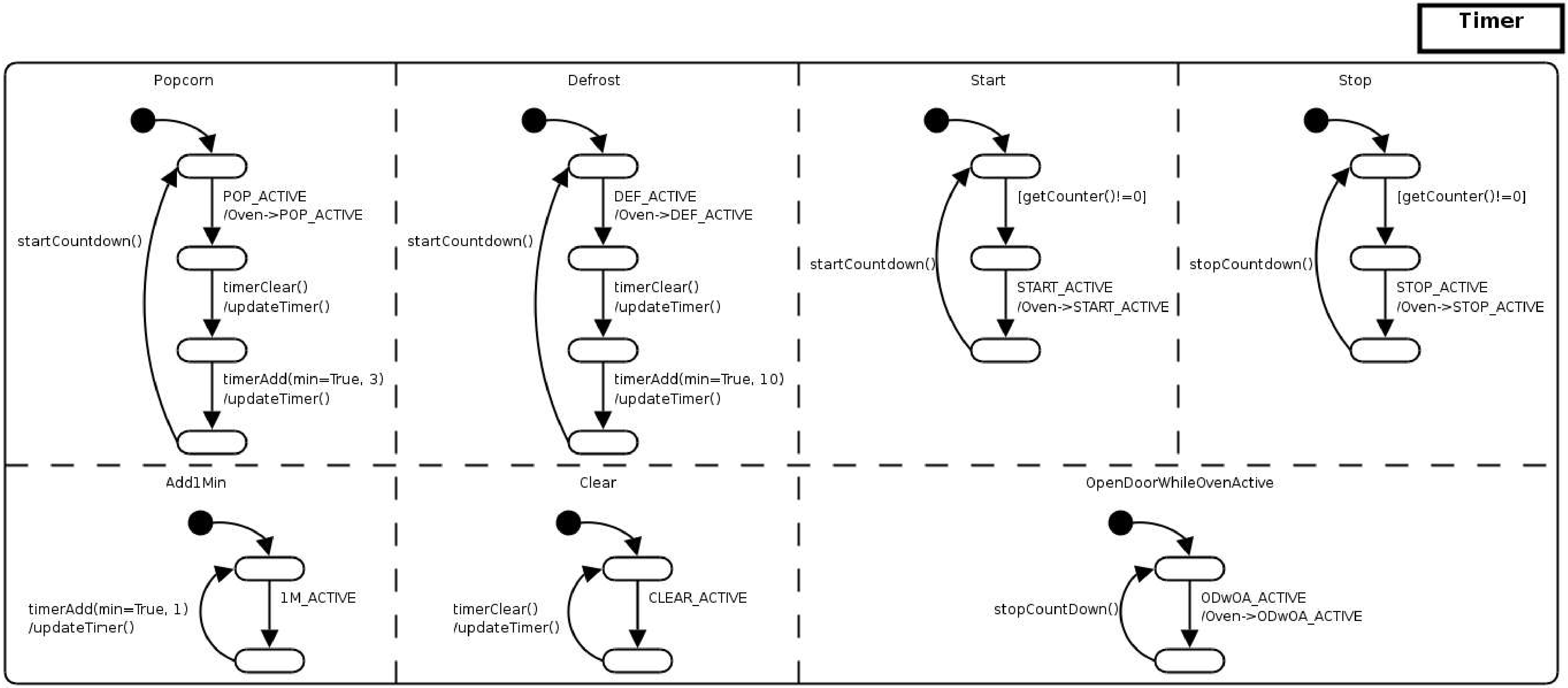

Start
Click()
[Door->getOpen()=False]
[Timer->getCounter()!=0]
[Oven->getActive()=False]
/Door->START_ACTIVE
/Oven->ovenStart()

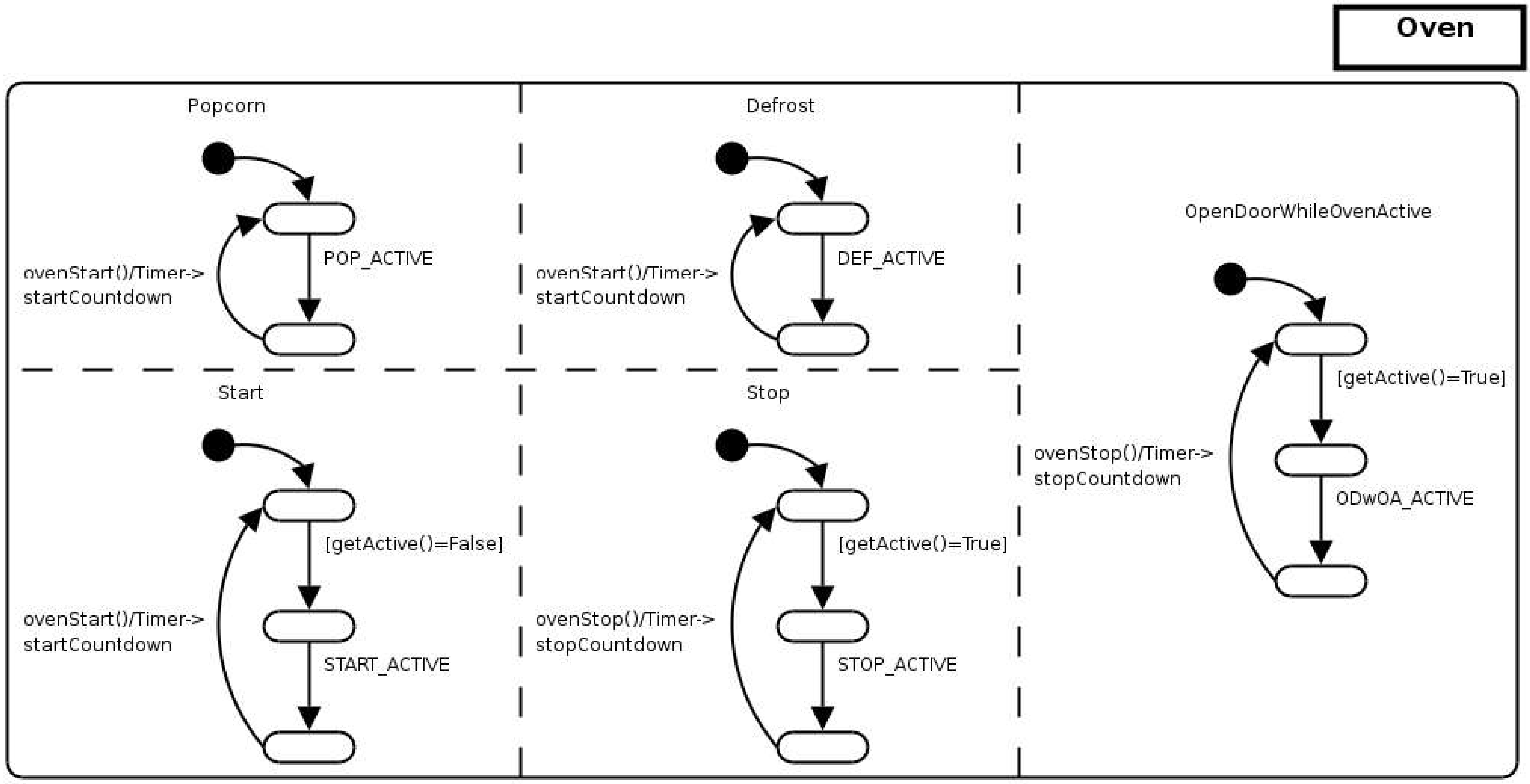
Stop


Stop
Click()
[Door->getOpen()=False]
[Timer->getCounter()!=0]
[Oven->getActive()=True]
/Door->STOP_ACTIVE
/Oven->ovenStop()

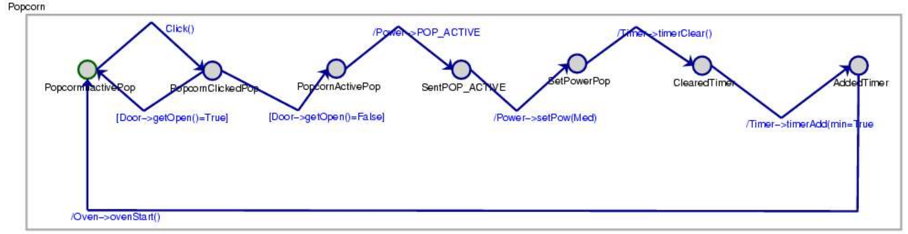

OpenDoorWhileOvenActive
Reddenbacher
Door
Timer
Oven
Open()
Door.open=False AND Oven.active=True
ovenStop()
stopCountdown()

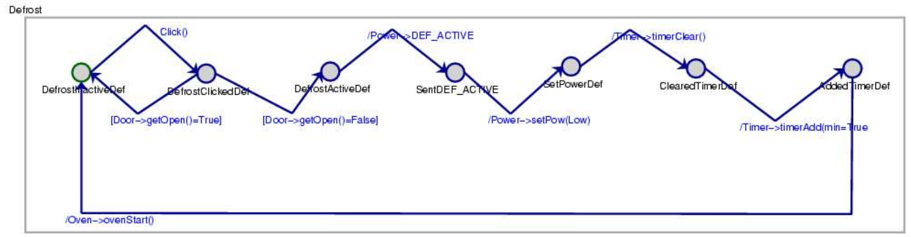

Door
Popcorn
[getOpen()=False]
POP_ACTIVE
/Timer->POP_ACTIVE
Defrost
[getOpen()=False]
DEF_ACTIVE
/Timer->DEF_ACTIVE
OpenDoorWhileOvenActive
Open()
[getOpen()=False]
/Oven->
ovenStop()
ODwOA_ACTIVE
/Timer->ODwOA_ACTIVE
Start
[getOpen()=False]
START_ACTIVE
/Timer->START_ACTIVE
Stop
[getOpen()=False]
STOP_ACTIVE
/Timer->STOP_ACTIVE


Timer
Popcorn
POP_ACTIVE
/Oven->POP_ACTIVE
startCountdown()
timerClear()
/updateTimer()
timerAdd(min=True, 3)
/updateTimer()
Defrost
DEF_ACTIVE
/Oven->DEF_ACTIVE
startCountdown()
timerClear()
/updateTimer()
timerAdd(min=True, 10)
/updateTimer()
Start
[getCounter()!=0]
startCountdown()
START_ACTIVE
/Oven->START_ACTIVE
Stop
[getCounter()!=0]
stopCountdown()
STOP_ACTIVE
/Oven->STOP_ACTIVE
Add1Min
timerAdd(min=True, 1)
/updateTimer()
1M_ACTIVE
Clear
timerClear()
/updateTimer()
CLEAR_ACTIVE
OpenDoorWhileOvenActive
stopCountDown()
ODwOA_ACTIVE
/Oven->ODwOA_ACTIVE

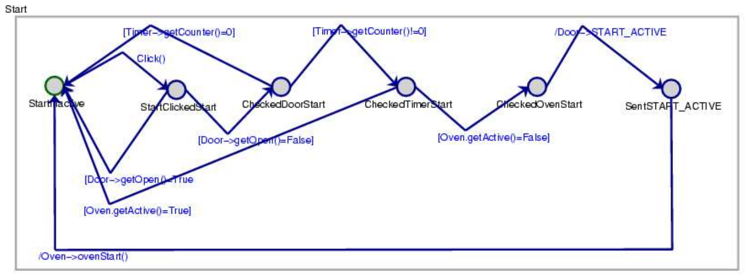


Oven
Popcorn
POP_ACTIVE
ovenStart()/Timer->startCountdown
Defrost
DEF_ACTIVE
ovenStart()/Timer->startCountdown
OpenDoorWhileOvenActive
[getActive()=True]
ODwOA_ACTIVE
ovenStop()/Timer->stopCountdown
Start
[getActive()=False]
START_ACTIVE
ovenStart()/Timer->startCountdown
Stop
[getActive()=True]
STOP_ACTIVE
ovenStop()/Timer->stopCountdown

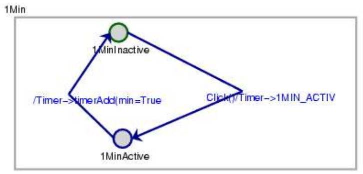




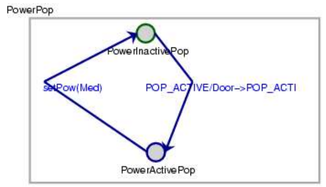
Popcorn, Defrost

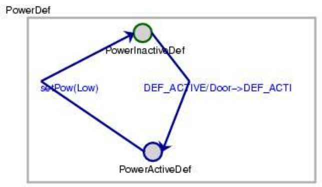


Start

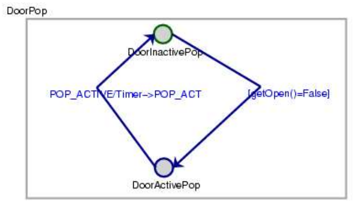


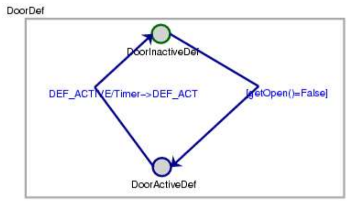
+1Min.

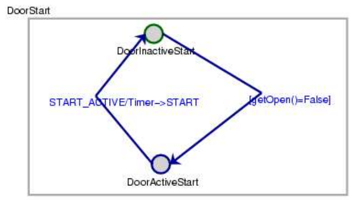


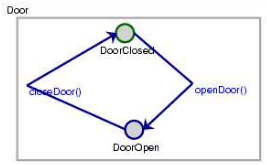


Power

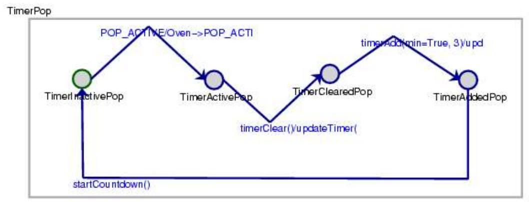


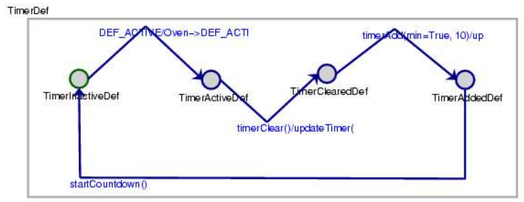


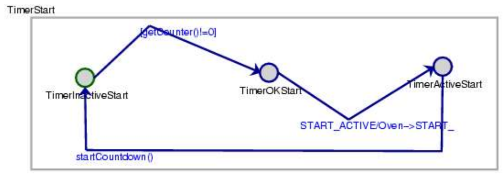


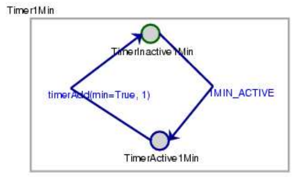


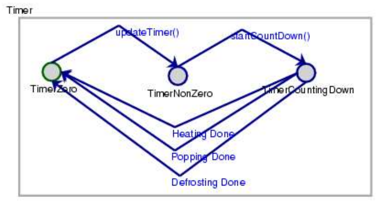
Door



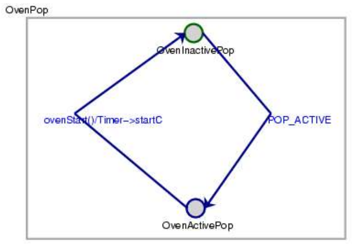


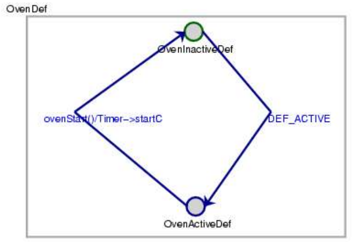


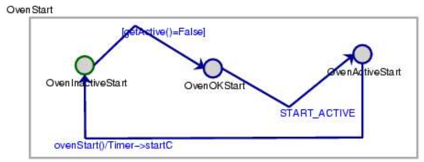


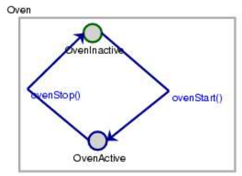


Timer









Oven

Demonstration & Questions

Final Questions?

References

1. Feng, Thomas Huining. “Charts, a Formalism for Modeling and Simulation Based Design of Reactive Software Systems”. http://moncs.cs.mcgill.ca/people/tfeng/thesis/thesis.html. Feb. 2004.

2. Harel, David. “Can Behavioral Requirements be Executed? (And why would we want to do so?)”

3. Harel, David and Hillel Kugler. “Synthesizing State-Based Object Systems from LSC Specifications”.

4. Harel, David and Rami Marelly. “Specifying and Executing Behavioral Requirements: The Play-In/Play-Out Approach”. September 10, 2002.