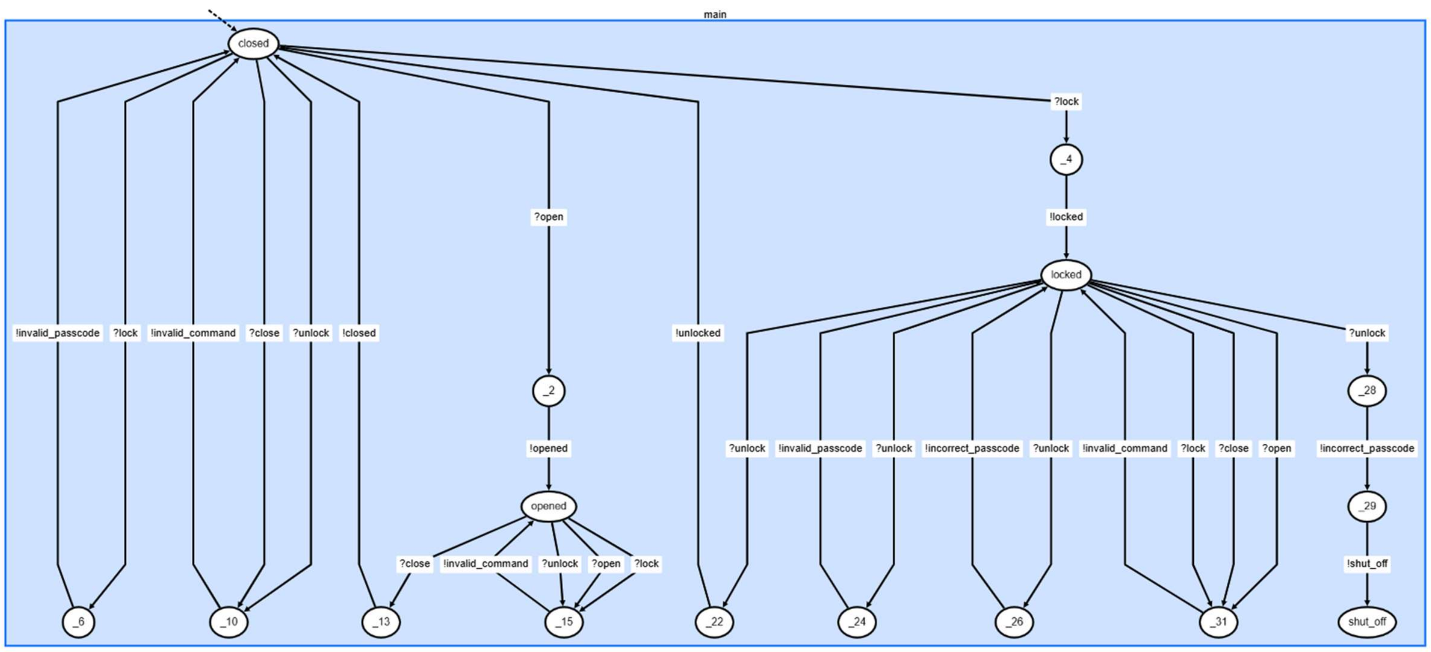
uva_group_20@axini.com

Final AML model:



Bugs:

Axini:

- 1. When 3 incorrect passcodes are entered
  2. 'incorrect_passcode' should be sent, and quiescence should be observed
  3. Besides 'incorrect_passcode', also 'shut_off' is expected to be sent before observing quiescence. This is however against DIALOG-01, which states one command should only get one response.
  4. Mistake in the specifications most likely, as it would be desired that you are informed that the door is shut off

Besto:

- 1. When a total of 3 invalid and incorrect passcodes are entered, ending with a invalid passcode
  2. 'invalid_passcode' should be sent to the controller, and quiescence should be observed
  3. 'invalid_passcode' is returned, and then 'shut_off' is returned
  4. Entering an invalid passcode is also counted as a failed try, and adds up to the 3 total tries

Logica:

- 1. When 3 incorrect passcodes are entered
  2. 'incorrect_passcode' and 'shut_off' should be sent, and quiescence should be observed
  3. This model does not want 'shut_off' to be sent, which is in line with the specification but differs from the Axini model
  4. The incorrect specifications were followed

uva_group_20@axini.com

OnTarget:

- 1. When ‘unlock’ is sent to the door in the state ‘locked’ with a different passcode
  2. ‘incorrect_passcode’ should be sent as the set password is not the same as the one received during ‘lock’
  3. ‘unlocked’ is sent instead of ‘incorrect_passcode’
  4. It performs no check on the passwords it receives during ‘lock’ and ‘unlock’

Quickerr:

- 1. When a command is sent and a response is expected
  2. A response should be sent within 0.5 seconds
  3. a ‘tick’ is observed before the response is sent,
  4. The response is sent too late (after 0.5 seconds)

SmartSoft

- 1. When ‘lock’ is sent to the door in the state ‘closed’ with a passcode longer than 4 numbers
  2. ‘invalid_passcode’ should be sent as the password does not comply to the maximum password length
  3. ‘locked’ is sent back to the controller and the door goes to the state ‘locked’
  4. There exists no check on the (max) length of the passcode

- 1. When ‘close’ is sent to the door which is in the state ‘open’
  2. ‘closed’ should be returned and the door should go to state ‘closed’
  3. ‘opened’ is returned instead of ‘closed’, so the controller receives the wrong message
  4. Either the only the wrong response is sent and it actually goes to the state ‘closed’, or both the wrong response is sent and the door goes to the wrong door

- 1. When ‘unlock’ is sent to the door in the state ‘locked’ with a passcode longer than 4 digits
  2. ‘invalid_passcode’ should be sent as the passcode does not comply to the maximum password length
  3. ‘incorrect_passcode’ is sent to the controller, as it is not the same as the passcode set earlier
  4. There exists no check on the (max) length of the passcode

TrustedTechnologies

- 1. When ‘lock is sent to the door in the state ‘closed’ with a passcode of length 1
  2. ‘locked’ should be sent as a response, as the passcode is between 0 and 9999
  3. ‘invalid_passcode’ is sent as a response.
  4. Passwords of a length below 4 are seen as invalid, but given that it is an integer, zeros should be imagined at the front, so a 4-digit number is formed. This model does not take this into consideration

uva_group_20@axini.com

- 1. When ‘unlock’ is sent to the door in the state ‘locked’ with an integer of length 1
  2. ‘unlocked’ should be sent as a response, as the passcode is between 0 and 9999
  3. ‘invalid_passcode’ is sent as a response.
  4. Passwords of a length below 4 are seen as invalid, but given that it is an integer, zeros should be imagined at the front, so a 4-digit number is formed. This model does not take this into consideration

univerSolutions

-

XtraSafe

- 1. ‘close’ is sent to the door which is in the state ‘closed’
  2. ‘invalid_command’ should be returned to the controller
  3. ‘closed’ was sent to the controller
  4. This model accepts self loops

- 1. ‘open is sent to the door which is in the state ‘opened
  2. ‘invalid_command’ should be returned to the controller
  3. ‘opened’ was sent to the controller
  4. This model accepts self loops

- 1. ‘unlock’ is sent to the door which is in the state ‘unlocked’
  2. ‘invalid_command’ should be returned to the controller
  3. ‘unlocked’ was sent to the controller
  4. This model accepts self loops

Code:

```
# Set the default timeout for responses (in seconds)

timeout 0.5

# Define an external channel
external 'door'

# Define a process describing the behavior of the coffee machine.
process('main') {

  var 'attempts', :integer, 0
  var 'code', :integer
```

uva_group_20@axini.com

```
 # Define stimuli (input) and responses (output) the process can perform on
  # this channel.
  channel('door') {
    stimulus 'open'
    stimulus 'close'
    stimulus 'lock', {'passcode' => :integer}
    stimulus 'unlock', {'passcode' => :integer}

    response 'shut_off'
    response 'opened'
    response 'closed'
    response 'locked'
    response 'unlocked'
    response 'invalid_command'
    response 'invalid_passcode'
    response 'incorrect_passcode'
  }

  # Describe the behavior of this process. Statements are read top to bottom,
  # similar to an imperative program.

# state representing the door being closed and unlocked
 state 'closed'
 repeat {
    o { receive 'open';  send 'opened'; goto 'opened' }
    o { receive 'lock', constraint: 'passcode == passcode && passcode > 0000
&& passcode < 9999', update: 'code = passcode'; send 'locked'; goto 'locked' }
    o { receive 'lock', constraint: 'passcode < 0000 || passcode > 9999'; send
'invalid_passcode'}
    o { receive 'unlock', constraint: 'passcode == passcode && (passcode
==  code || passcode != code)'; send 'invalid_command'}
    o { receive 'close'; send 'invalid_command' }
    }

# state representing the door being open
 state 'opened'
 repeat {
    o { receive 'close'; send 'closed'; goto 'closed' }
    o { receive 'lock', constraint: 'passcode == passcode && (passcode == 0 ||
passcode != 0)'; send 'invalid_command'}
    o { receive 'open'; send 'invalid_command'}
    o { receive 'unlock', constraint: 'passcode == passcode && (passcode
==  code || passcode != code)'; send 'invalid_command'}
 }
```

uva_group_20@axini.com

```
 # state representing the door being closed and locked
 state 'locked'
 repeat {
    o { receive 'unlock', constraint: 'passcode == passcode && passcode ==
code', update: 'attempts = 0'
        send 'unlocked'
        goto 'closed' }
   o {receive 'unlock', constraint: 'passcode == passcode && (passcode < 0000
|| passcode > 9999)'
       send 'invalid_passcode' }
    o {receive 'unlock', constraint: 'passcode == passcode && (passcode >
0000) && (passcode < 9999) && (passcode != code) && (attempts < 2)', update:
'attempts = attempts + 1'
       send 'incorrect_passcode' }
    o {receive 'unlock', constraint: 'passcode == passcode && (passcode >
0000) && (passcode < 9999) && (passcode != code) && (attempts == 2)', update:
'attempts = 3'
       send 'incorrect_passcode'
       send 'shut_off'
       goto 'shut_off'}
    o { receive 'open'; send 'invalid_command'}
    o { receive 'close'; send 'invalid_command'}
    o { receive 'lock', constraint: 'passcode == passcode && (passcode == 0 ||
passcode != 0)'; send 'invalid_command'}
   }

 state 'shut_off'
 repeat { }

}
```