Modelling III

UML state machines

Ketil Stølen

Based on slides prepared by Prof. Øystein Haugen, HiØ & SINTEF

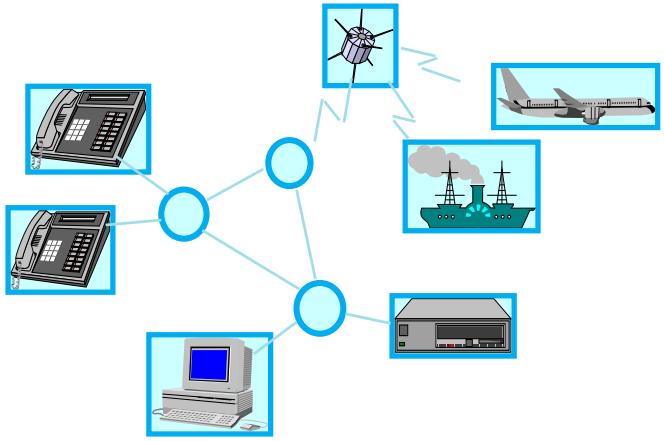
Overview of lecture

- State machines
- Consistency

State machines

Suitability of UML state machines

- reactive
- concurrent
- real-time
- distributed
- heterogeneous
- complex



UML State Machines

- Finite
  - a finite number of states
- State
  - a stable situation where the process awaits stimuli
  - a state in a state machine represents the history of the execution
- Machine
  - that only a stimulus (signal, message) triggers behavior
  - the behavior consists of executing transitions
  - may also have local data

Exercise

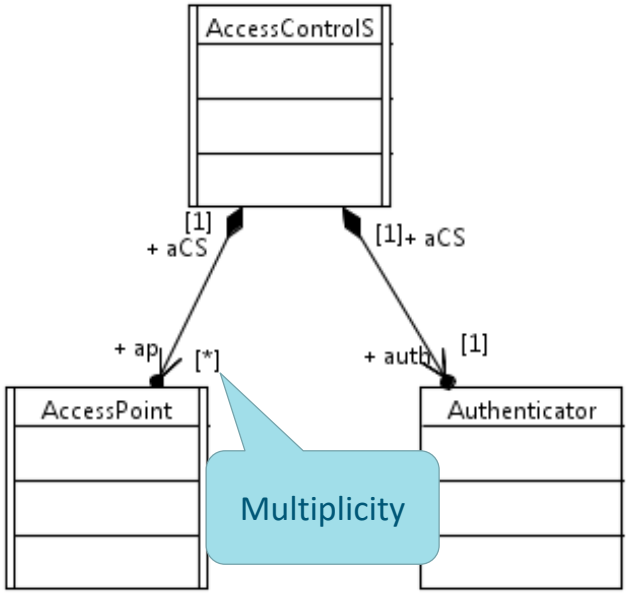
- What is a ***state*** in a programming language?
- What is a ***machine*** in a programming language?

Our example today

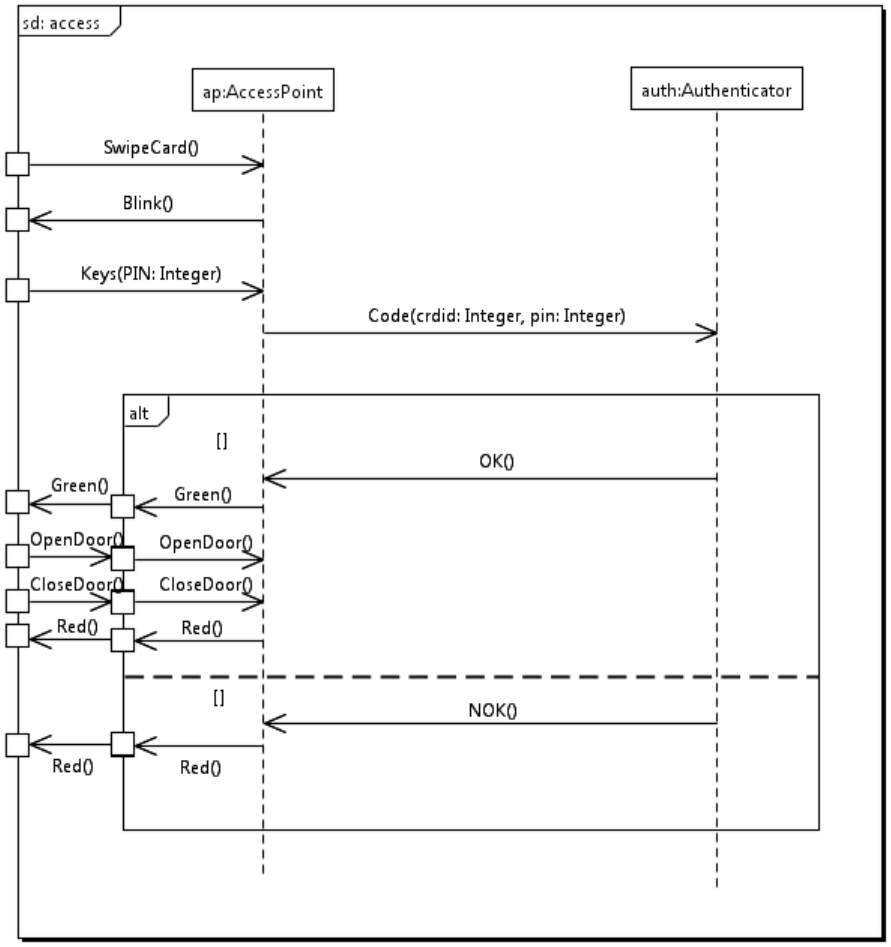
An Access Control System

- A set of Access Points are established to control the access to an area
- The Access Points controls the locking of a door
  - in a more abstract sense, access control systems may control bank accounts or any other asset that one wants to protect
- The Access Point access is granted when two pieces of correct identification is presented
  - A card
  - A PIN (Personal Identification Number)
- The access rights are awarded by a central Authentication service

The concepts in a class diagram



Happy Day Scenario

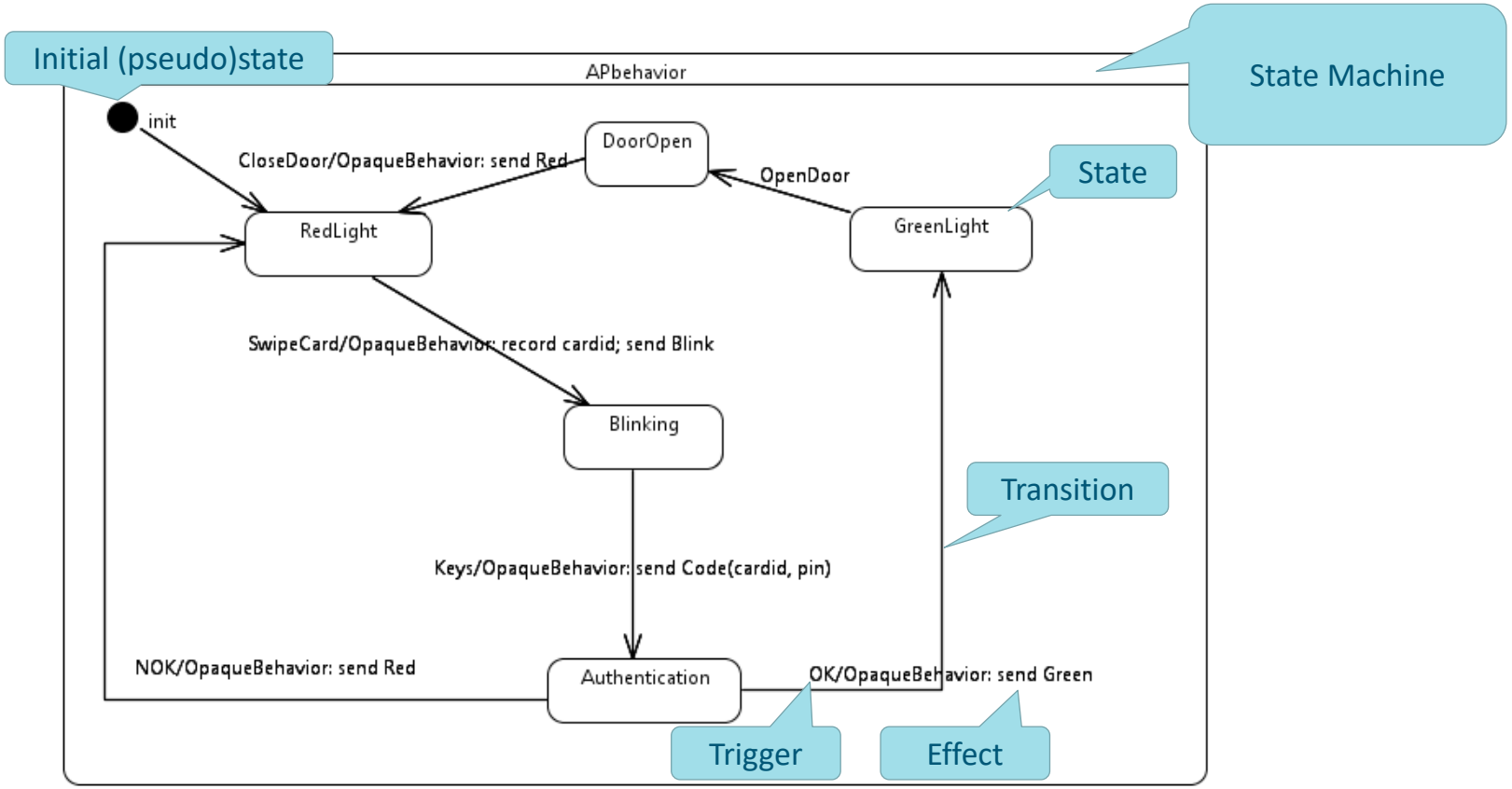


Exercise: Describe the set of traces representing the semantics of access

OpaqueBehavior is a UML behavior defined in another language

In this course we are flexible wrt how behaviors are expressed
Hence, using the OpaqueBehavior construct is not important

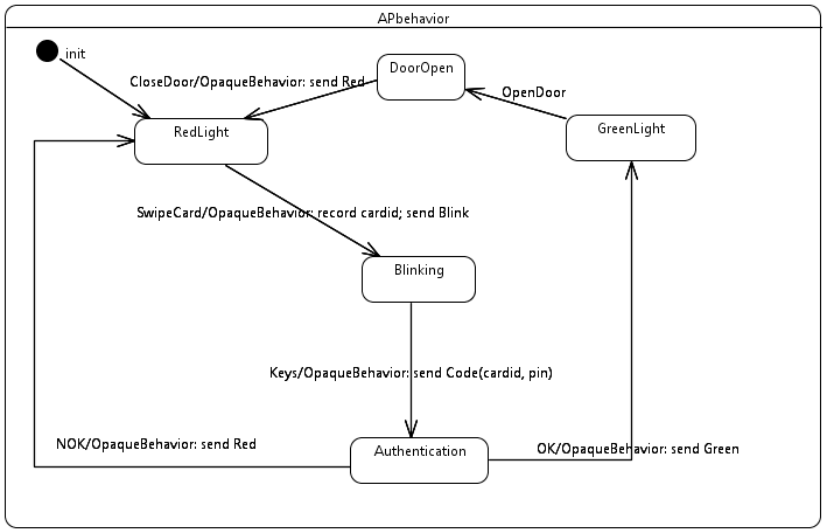
The behavior of the AccessPoint

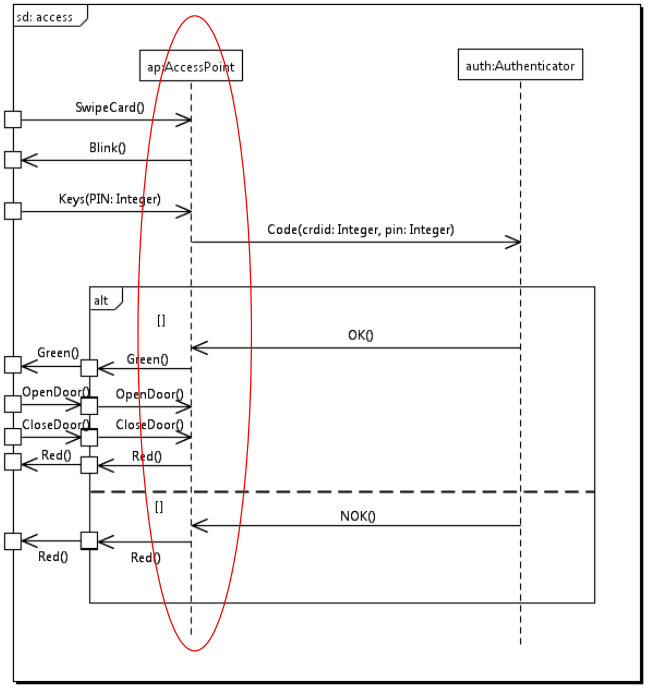


Exercise: Make a state machine for the Authenticator

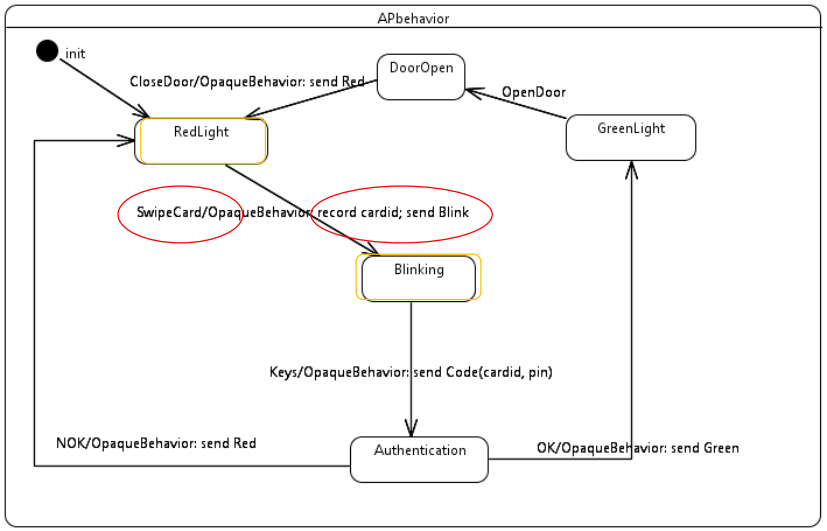
Consistency

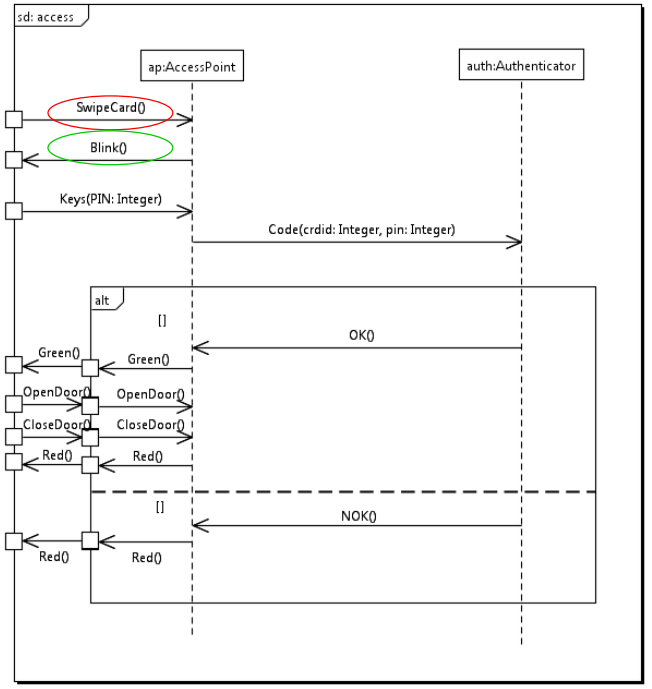
Runtime consistency – behaviors corresponding



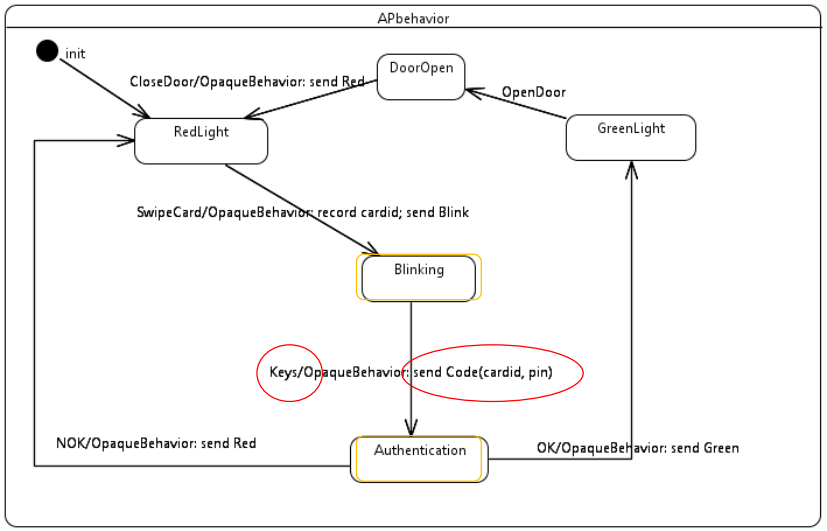


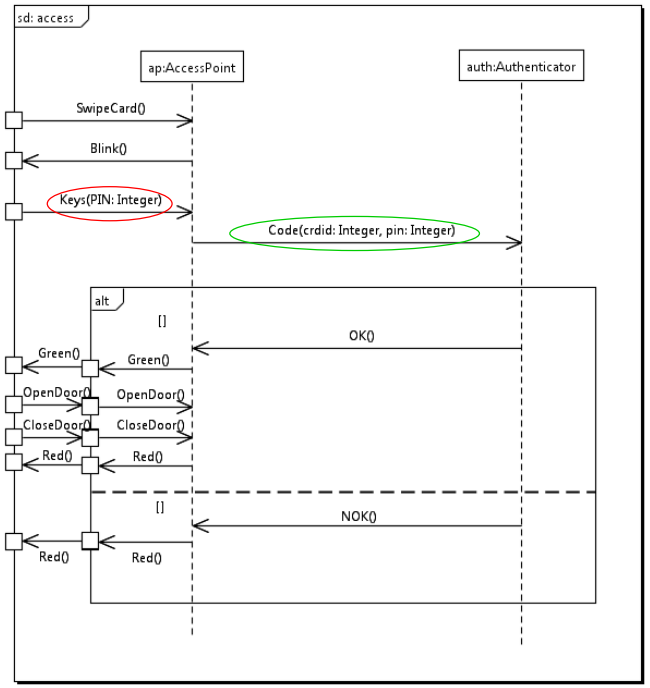
Let's execute the state machine according to the sequence diagram



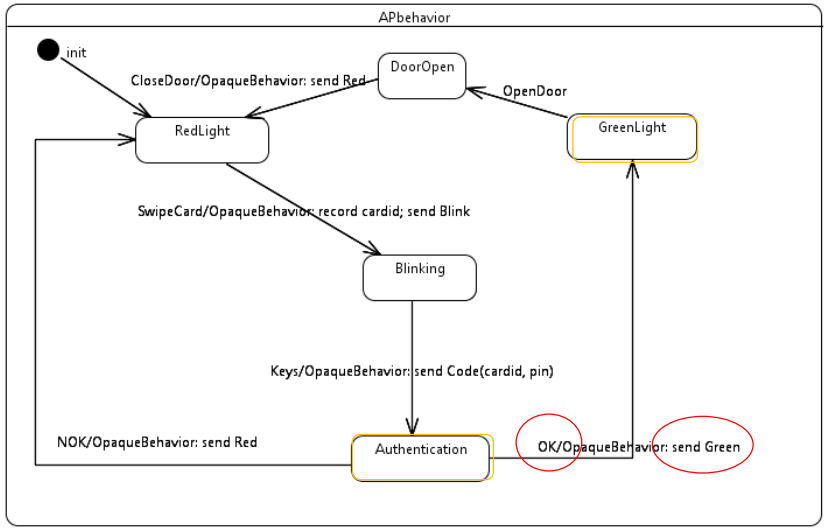


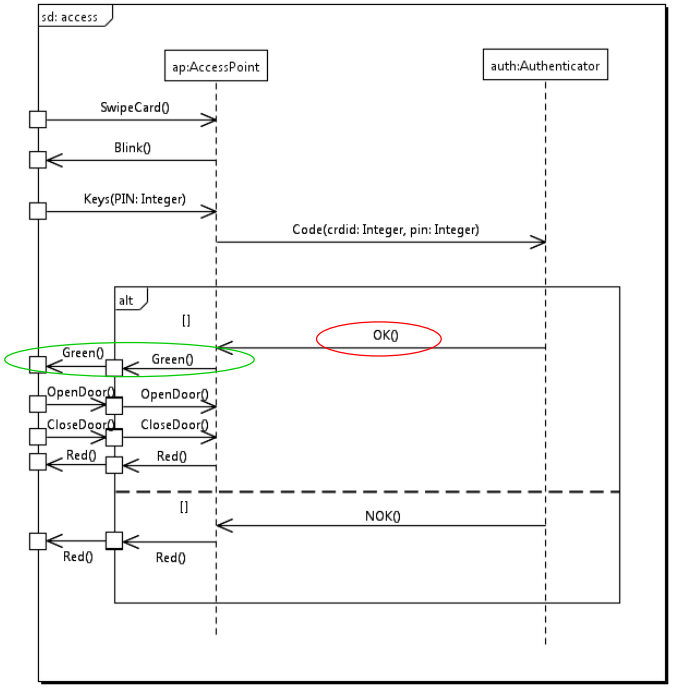
Play it again Sam



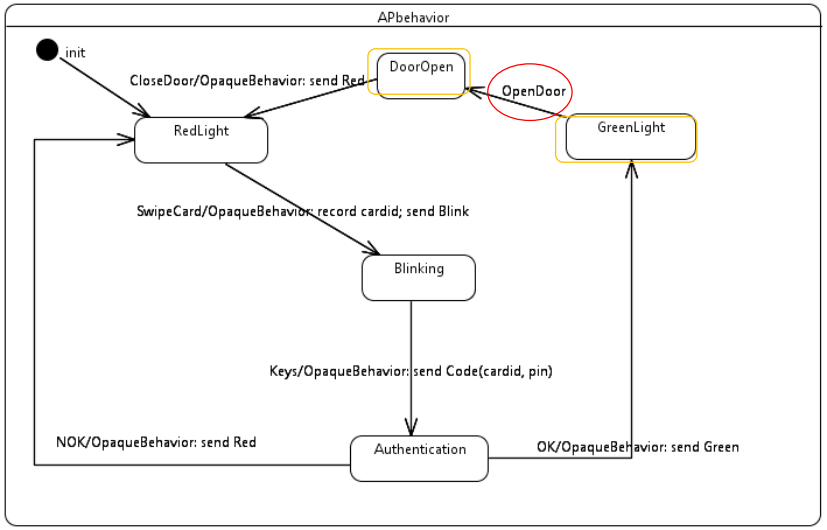


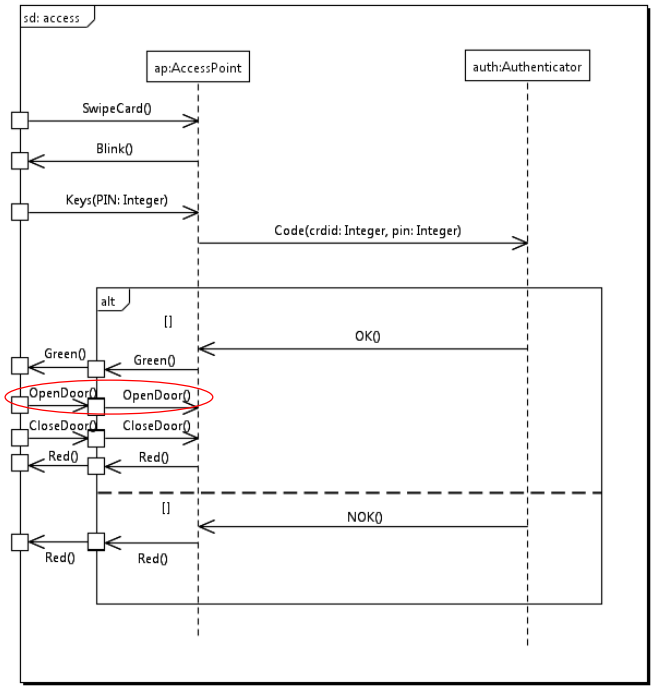
Access granted (one out of two alternatives)



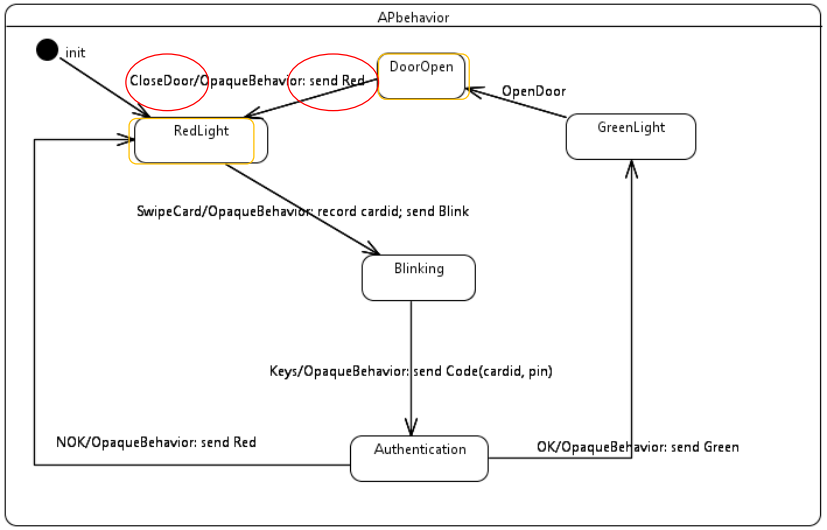


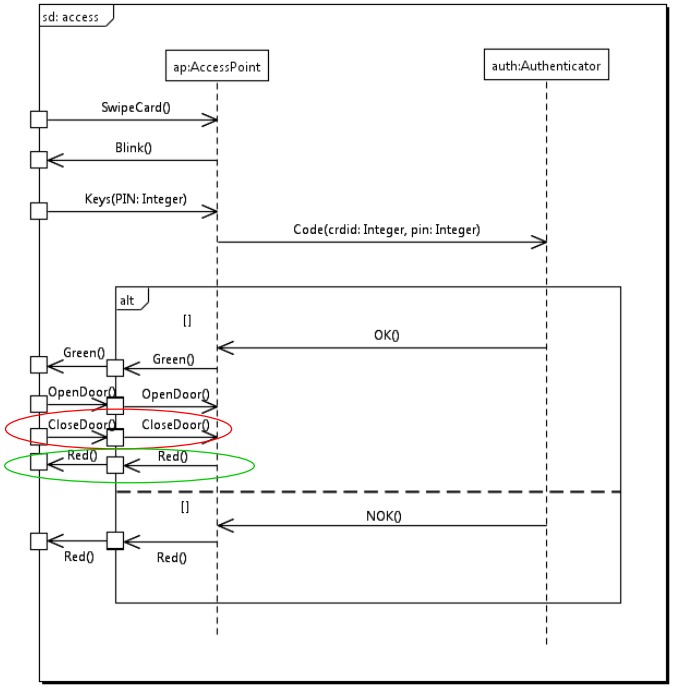
User opens the door



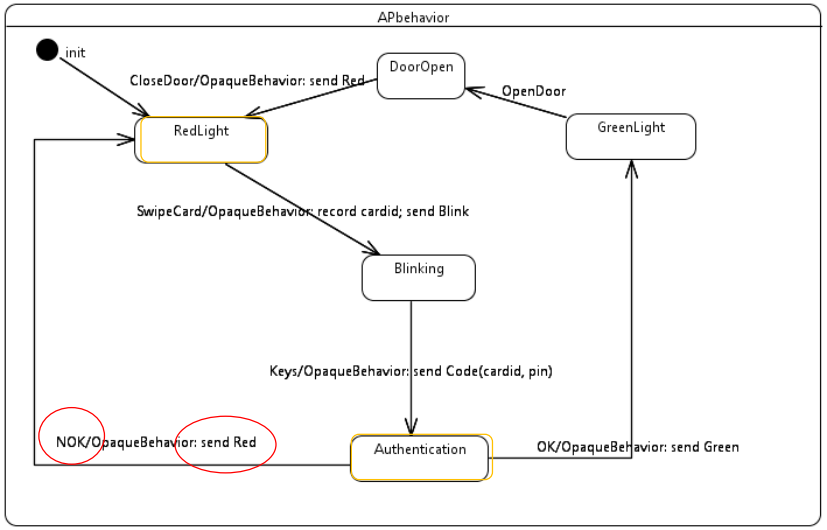


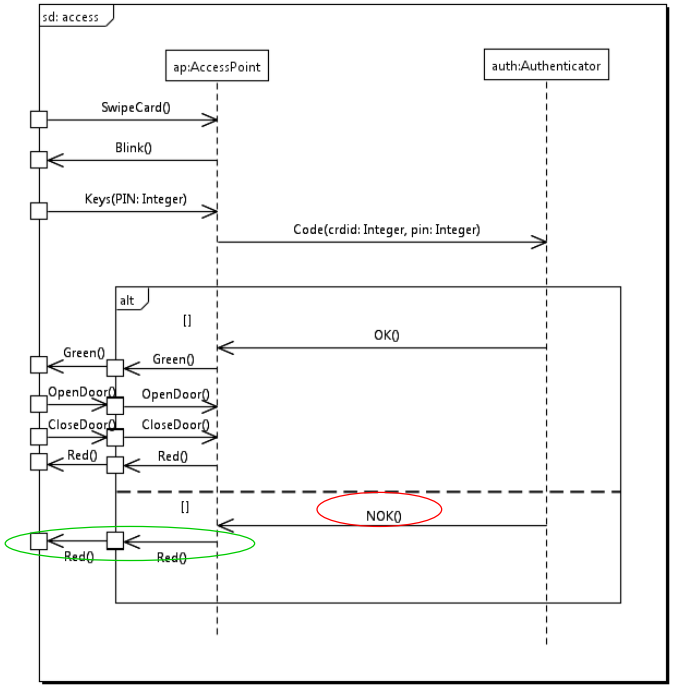
User closes the door again





Access not granted (second of two alternatives)





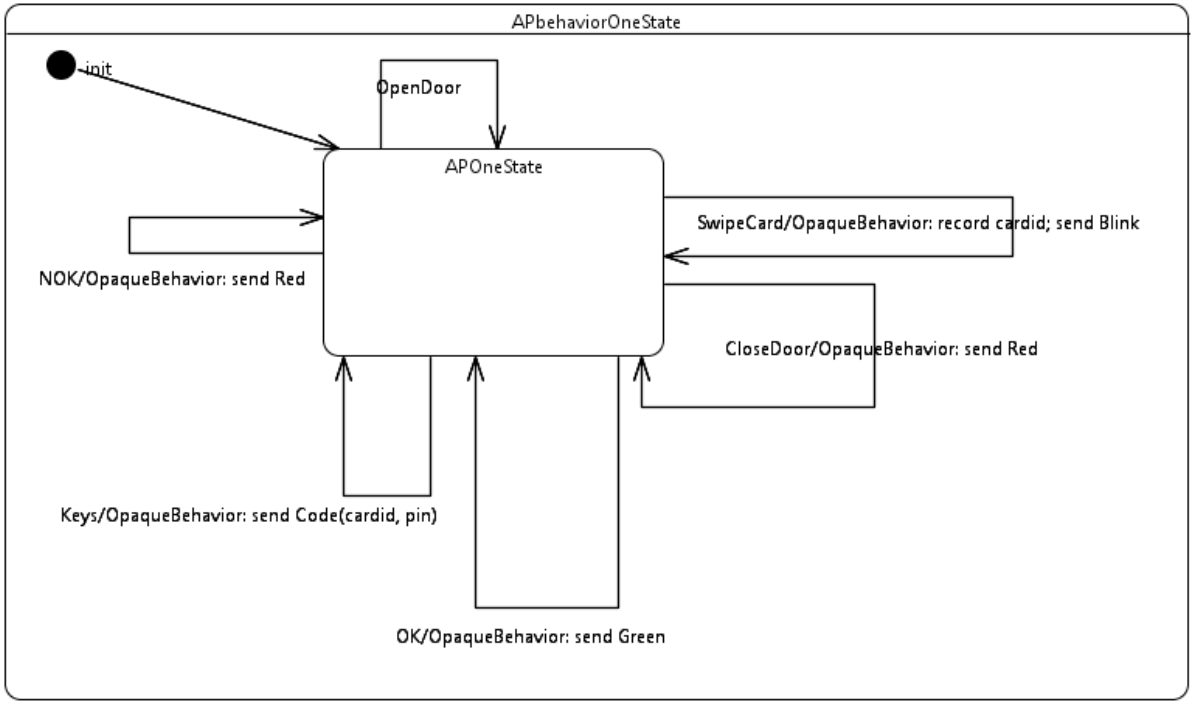
Concluding the runtime consistency check

- The APbehavior state machine satisfies all traces of the sequence diagram access
- Thus these behaviors are consistent

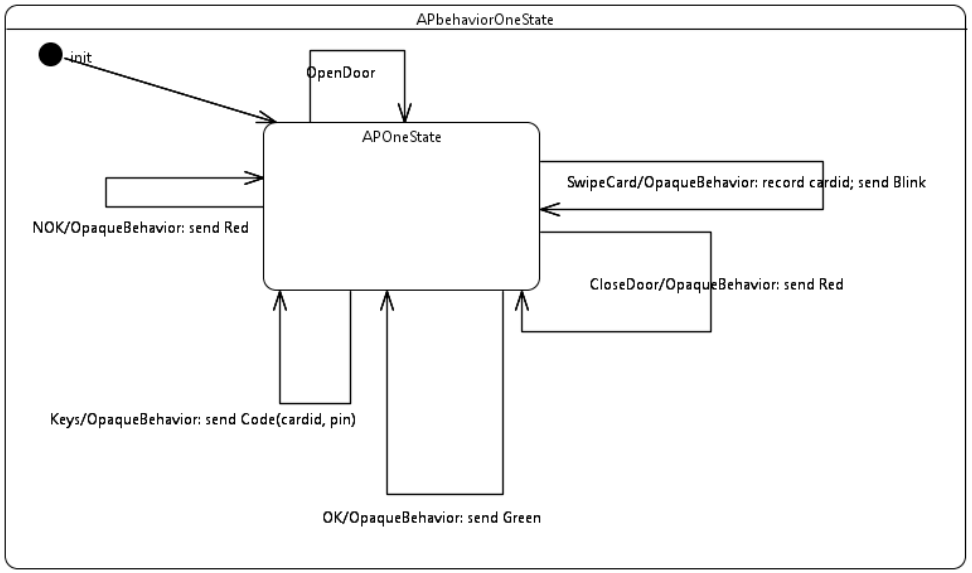
Exercise: Are we then perfectly happy?

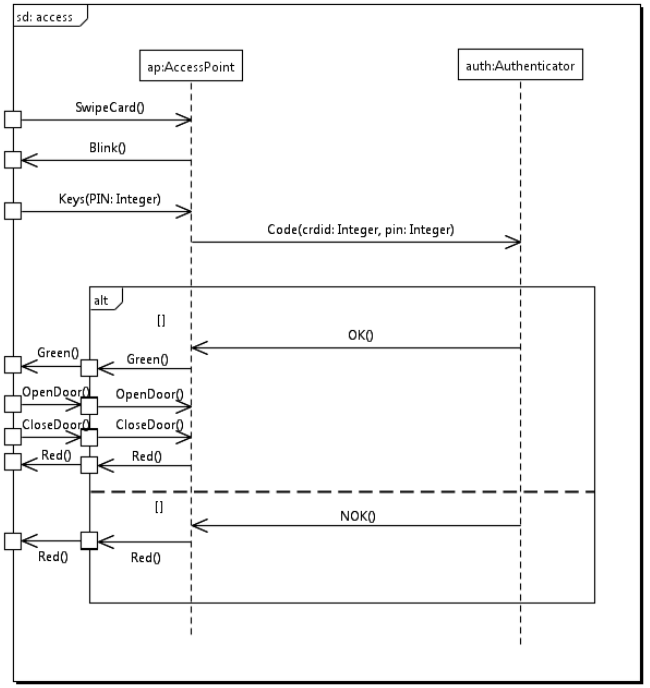
Exercise: Describe a trace of the state machine that is not in the semantics of the sequence diagram

Another attempt to define the state machine

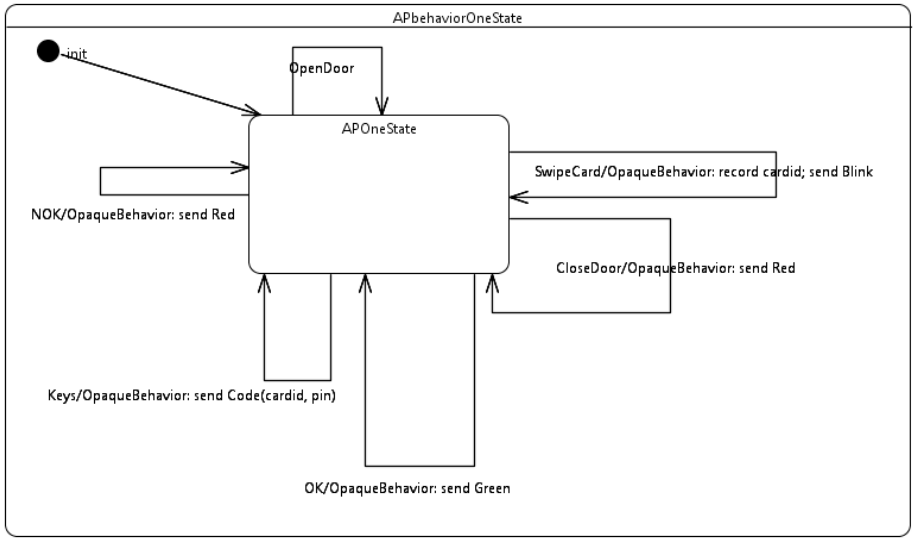


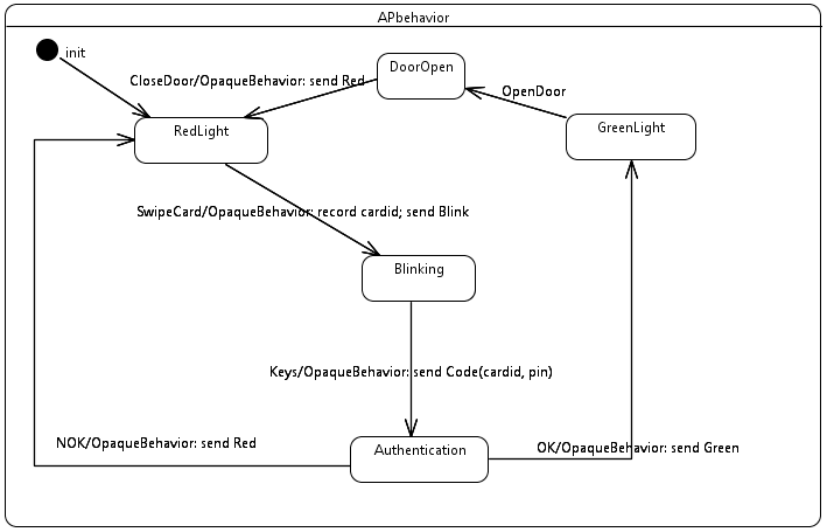
Exercise: Are these behaviors consistent?





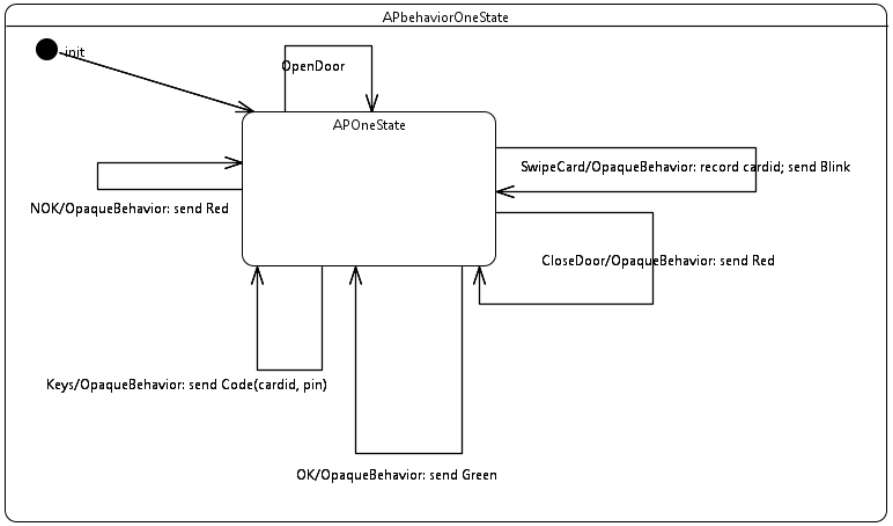
Exercise: Which state machine is the better description?

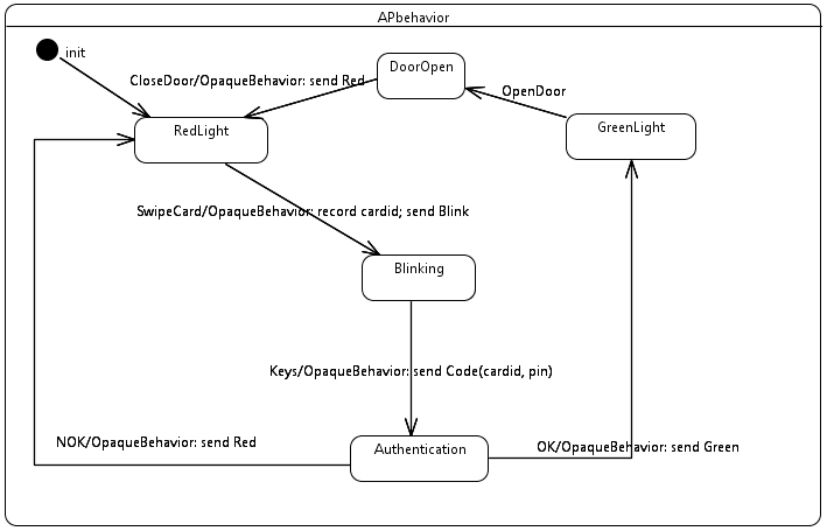




and why?

What if the user started keying the PIN at once?





***APbehavior* may spot the problem**

***APbehaviorOneState* will go on in error**

Why using different states?

- Several different states distinguishes between different situations
- In different situations, different reactions may be desirable to the same trigger
- A specific state represents in a compact way the whole history of behavior that led to reaching that state

Exercise: Explain the difference between the two machines in terms of a dataprogram

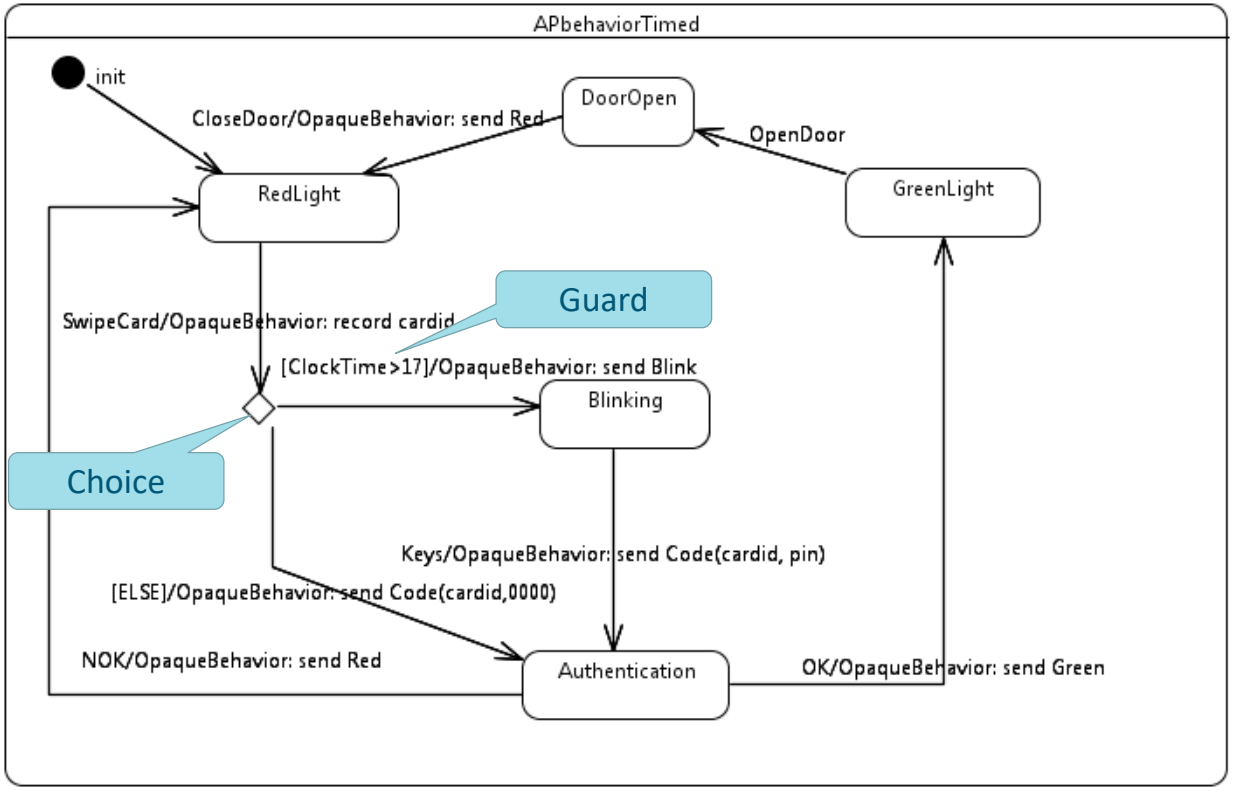
Guidelines and Reminders

- Even though the state machine was consistent with the sequence diagram, the state machine was flawed
  - The reason was that sequence diagrams are only partial descriptions of the whole, while state machines are complete descriptions of a part of the whole
- Use several states if you can
  - Each state representing a stable, recognizable situation
- We should supplement our state machine with all the possible different transitions
  - This would help us consider and handle most error situations

What if we need to modify a state machine?

- Our access control system should possibly be acting differently during working hours than at other times
- How well do state machines cope with modifications?

Enhancing the state machine



Summarizing

- State machines describe behavior of independently acting components
- Reactive systems are suitable for state machines
- Consistency checks between sequence diagrams and state machines are very useful
  - but not sufficient
- State machines are robust in as much as additional functionality can often be included without ripple effects on other parts of the behavior