

# Model-Based Testing

(DIT848 / DAT261)

Spring 2017

## Lecture 5

### Extended Finite-State Machines

Gerardo Schneider

Department of Computer Science and Engineering  
Chalmers | University of Gothenburg

# So far...

We have seen

- Testing in general
  - Black box testing
  - White box testing
  - Beyond Unit testing
- A bit on EFSM

Today:

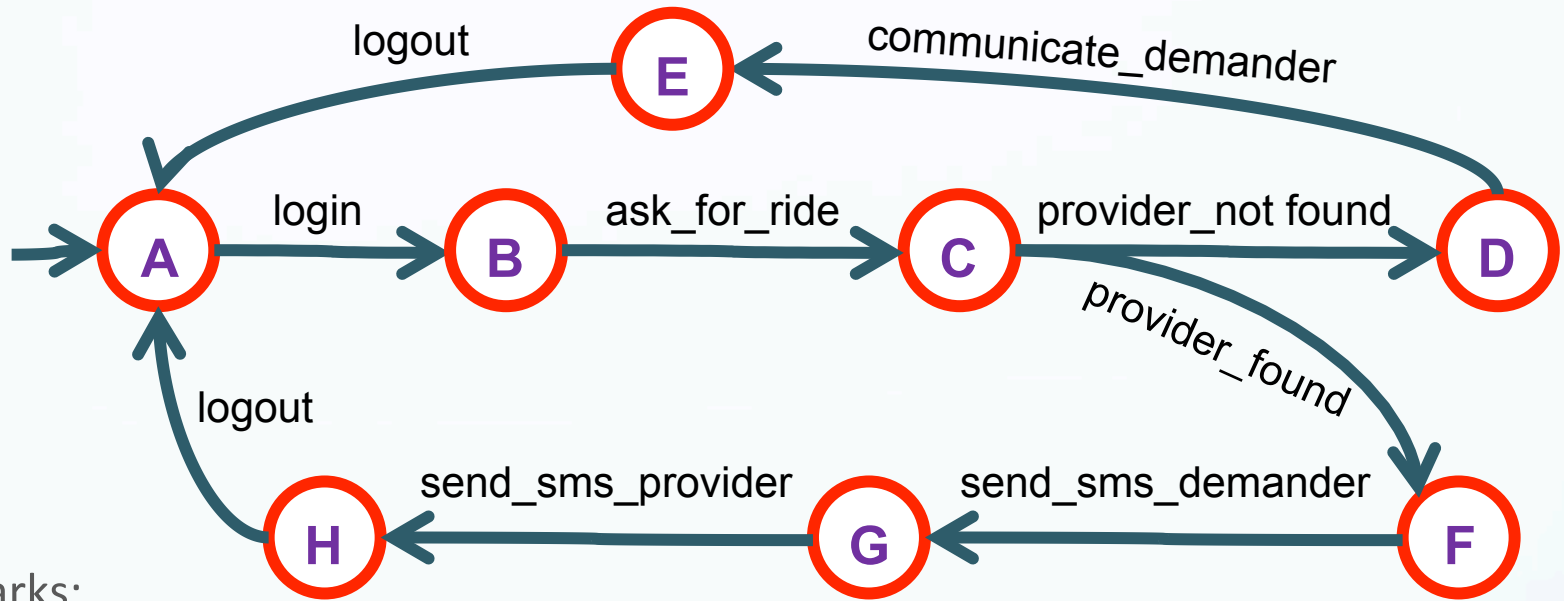
- Interactive exercises on EFSM

# Car sharing ride system (1)

- This task is concerned with part of a car sharing ride system where *demanders* (asking for a ride) log in to a web system asking for a *provider* (having a car and offering places in the car) to share a particular route.
- Your task is to **define a Finite-State Machine (FSM)** for the following specification:
  1. The demander logs into the system;
  2. the demander provides information on the particular route he/she wants and other information useful for the ride;
  3. the system checks whether there is a provider satisfying the demander's request;
  4. if a provider is found then an SMS is sent to both the provider and the demander confirming the ride, and the demander is logged out from the system;
  5. if no provider is found, this is communicated to the demander, who is automatically logged out.

# Car sharing ride system (1)

## Proposed Solution



Some remarks:

- Many other solutions depending on how much do you abstract
  - A "good" solution should be abstract enough as to capture the informal description (but not too much as to be useless)
- "logout" could be eliminated (as it is automatic)
- No check on whether login is correct or not (not in the specification)
- Implicit loop in state "C" on "look\_for\_provider"

# Car sharing ride system (2)

- Give **2** test cases that **can** be extracted from your FSM, and **2** that **cannot** be extracted from it.

**Note:** Consider test cases you might want to extract given a "full" specification of the system (consider that the FSM is given as a first step towards a full description of the system)

# Car sharing ride system (2)

## Proposed Solution

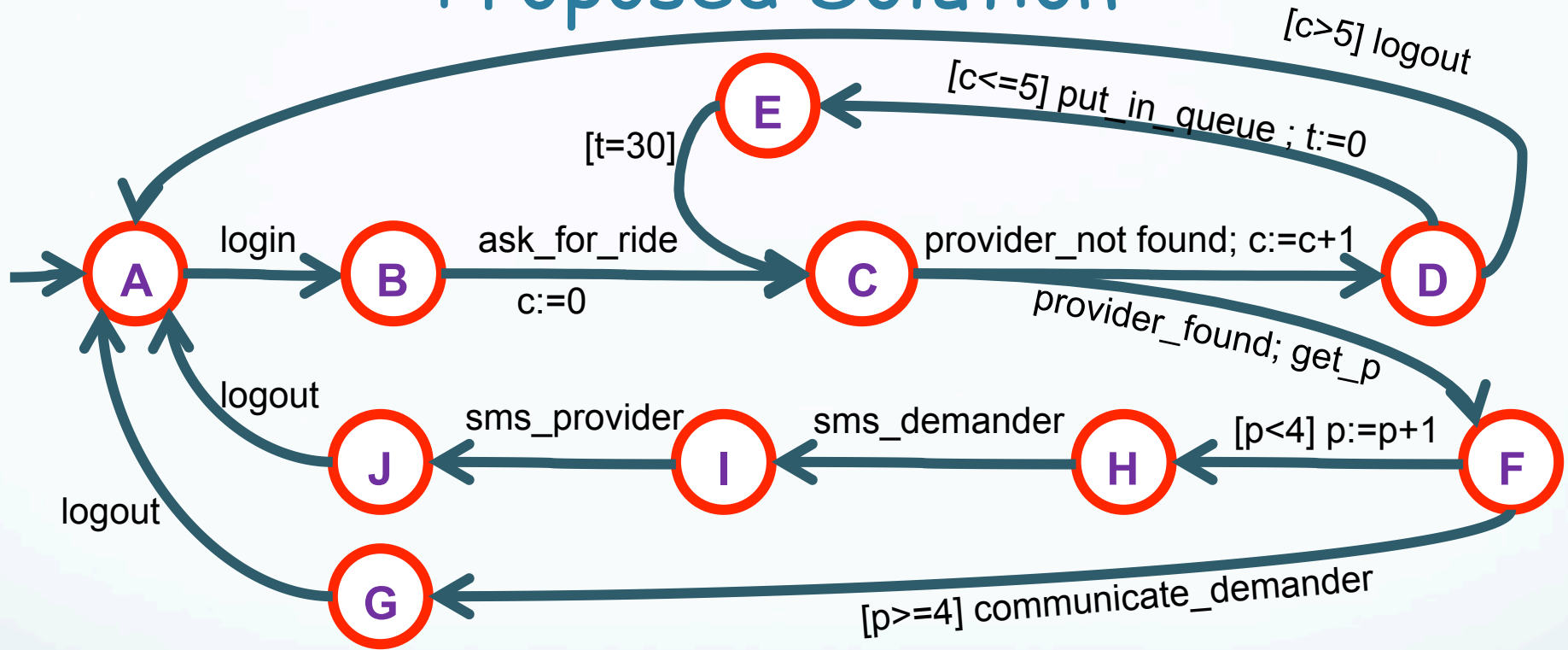
- Test cases you can extract:
  1. After login if there is a provider then the demander gets an sms indicating that
  2. If no provider exists for that ride then the user is logged out after getting a notification
  
- Test cases you cannot extract:
  1. If a provider does exist for the ride, the user may still not get a ride due to overbooking
  2. Any timing constraints in what concerns how much time to wait for getting a confirmation of a ride

# Car sharing ride system (3)

- Draw an *Extended Finite-State Machine (EFSM)* for a variation of the system of part (1). The new description of the system is as follows:
  1. The demander logs in into the system and asks for a ride as before.
  2. If a potential provider is found then it is first checked that the provider can offer the ride, which only happens if there are less than 4 confirmed demanders for that particular provider. If a provider is found but there is no place, then a communication is sent to the demander.
  3. If the ride request can be accepted, then an SMS is sent to both provider and demander confirming the ride, a counter counting the number of demanders for that particular provider is increased, and the demander is logged out.
  4. If a provider is not found, then the demander is put on a queue for 30 minutes after which the system checks again whether a provider for the requested ride is found; this is repeated at most 5 times, and if finally a provider is not found then the demander is automatically logged out.

# Car sharing ride system (3)

## Proposed Solution



Some remarks:

- Brackets (" $[.]$ ") are used as a short for "If ... then ..."
- $t$ : timer;  $c$ : number of times a demander may request a ride;  $p$ : nr of passengers (stored in the DB; get using "get\_p")
- Assumption: the timer is automatically incremented (implicit loop in state E)



# To Be Continued...

- More on EFSM in next lecture...