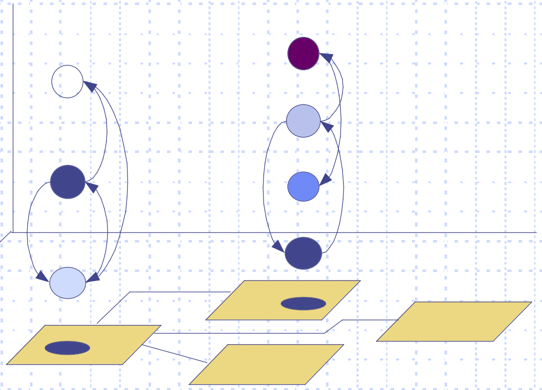
State Machines and Object Life Cycles

G. Falquet, L. Nerima

Another Modelling Dimension

- Dynamic view of the objects
- How objects evolve, change their state



I
S
I

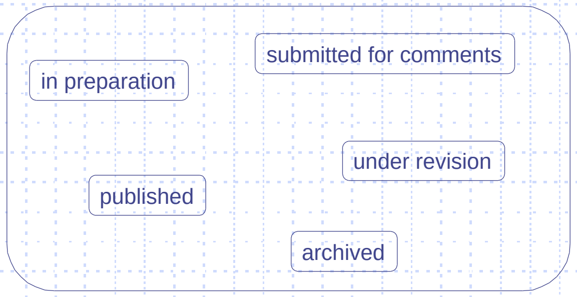
State Machine

- Specify a sequence of states that an object goes through in response to events
- Describe object life cycles
- Centred on one object
- Contains object states and state transitions
- Strong theoretical background: finite state automata, Petri nets, state charts (Harel). Real-time systems.

State (of an object)

- Situation during the life of an object
- During this situation the object
  - satisfies some condition
  - or performs some **activity**
  - or waits for some event
- A state can have substates (finer description)

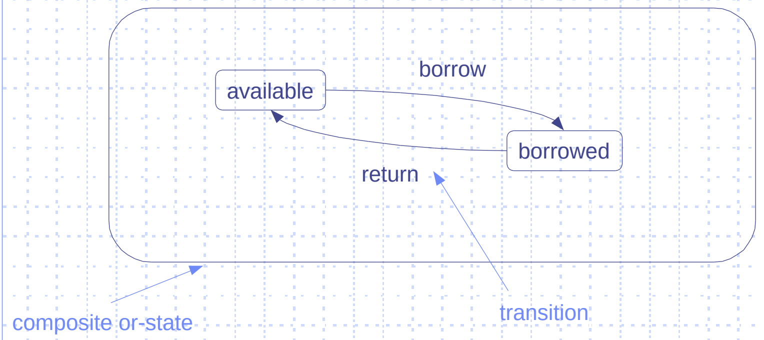
Example: Document



Event

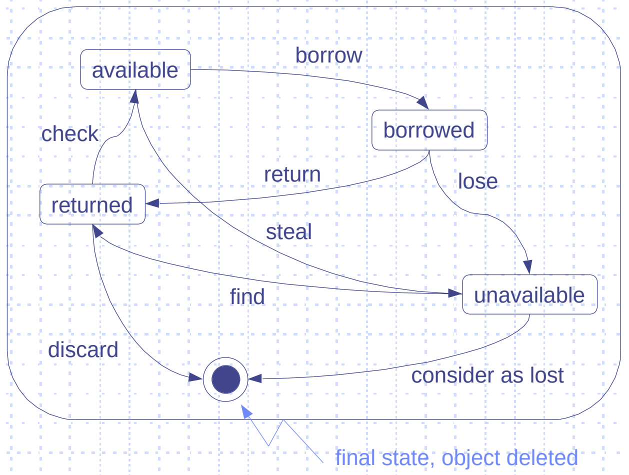
- "Noteworthy occurrence that has a location in time and space."
- Can trigger a state transition (+ actions).
- Kinds of event:
  - call (operation),
  - change (something has changed in the system),
  - signal (send an event to another object),
  - time (timer, timeout, ...)

A Simple State Machine for Books



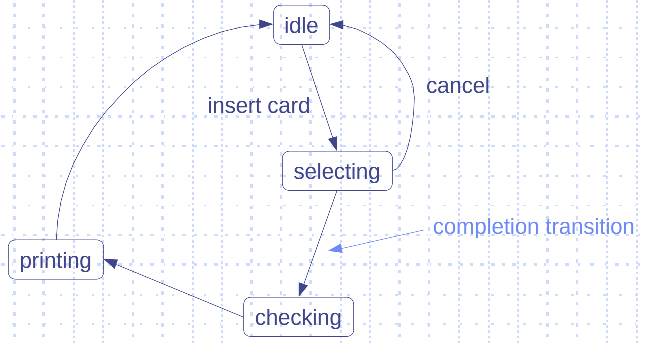
❖ A transition can be fired only if the object is in the source state and the event occurs.

Another State Machine for Books



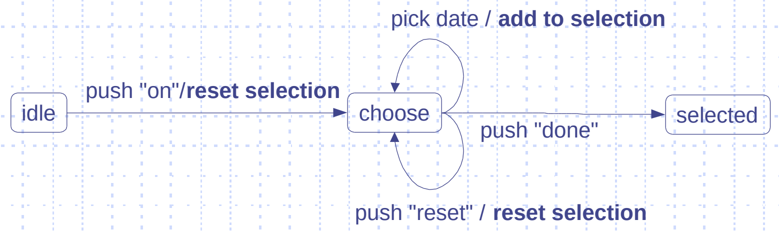
Completion transitions

- Triggered by the completion of the state's activity
- Notation: no event name



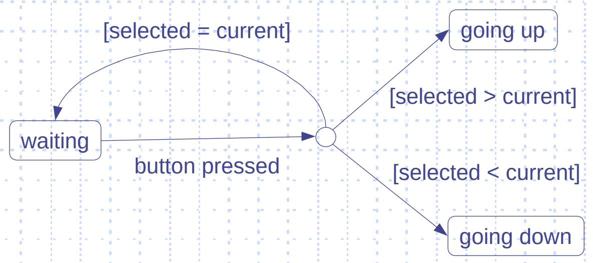
Event triggered actions

- A transition may execute an action
- Actions are atomic (assignment, send signal, create object, create link, ...)
- Actions must be fast (≠ complex computation)

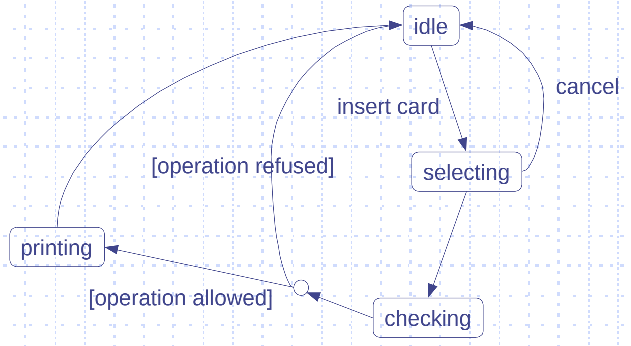

pick date / **add to selection**
idle
push "on"/**reset selection**
choose
push "done"
selected
push "reset" / **reset selection**

Guarded transitions

- Guard = additional condition on a transition
- Transition fired only if guard is true
- Only one transition fired if several guards are true
- Guards are queries (may not change values)



Guarded completion

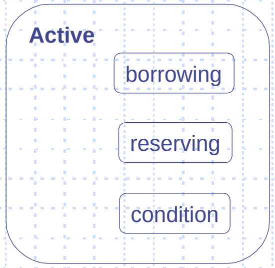


Concurrent states

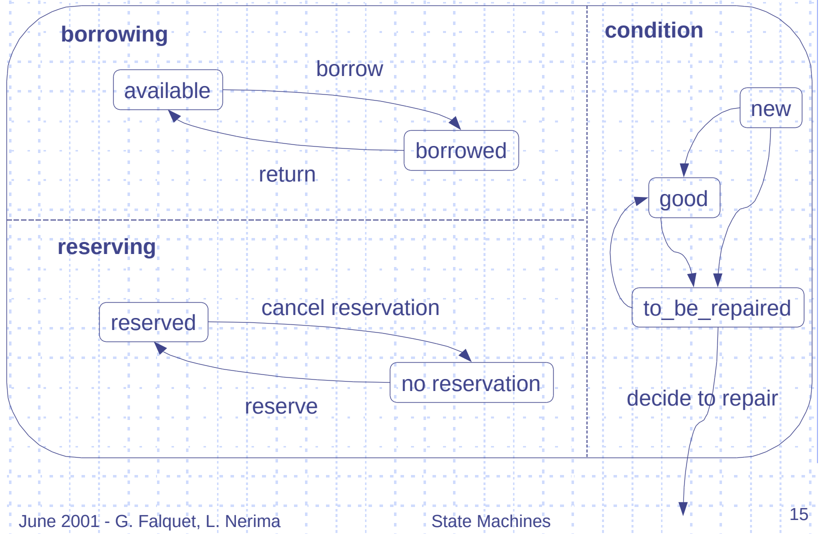
- An object may be in two (or more) states at the same time.
  - Book: (borrowed + reserved)
  - Book: (borrowed + not reserved)
- => Large number of combinations
- Composite concurrent states

Concurrent states (*and* states)

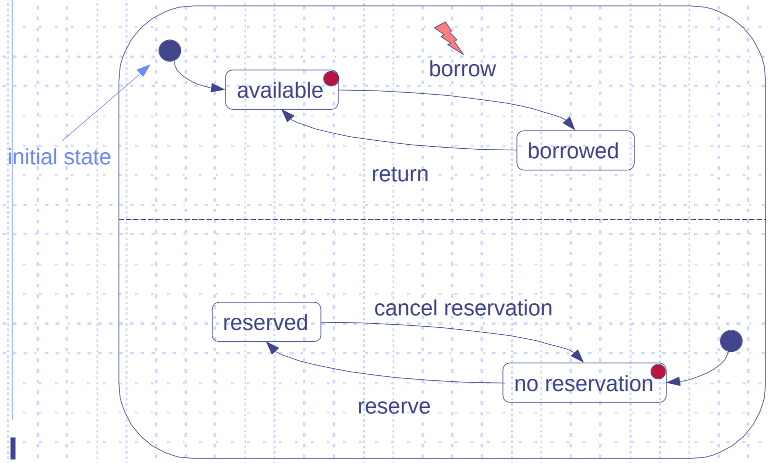
- Being in state Active means being simultaneously in states condition, borrowing, reserving.



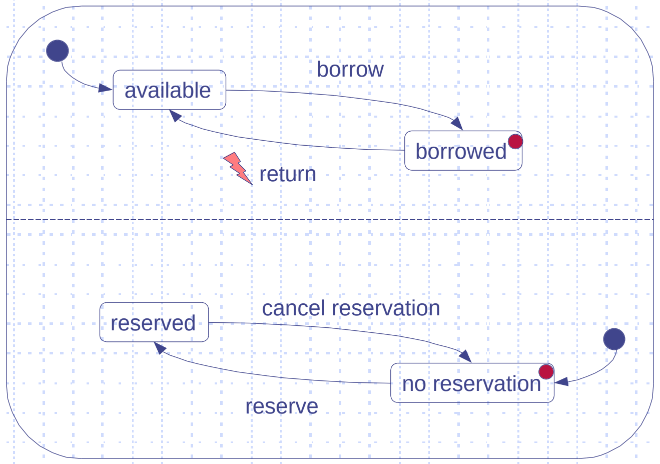
Composite concurrent states



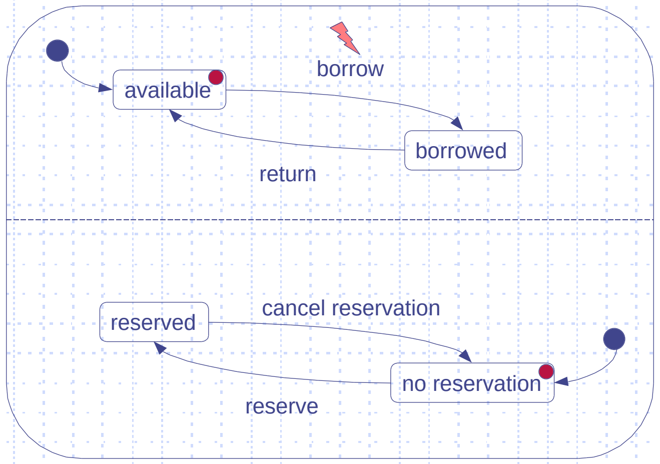
Concurrent active states



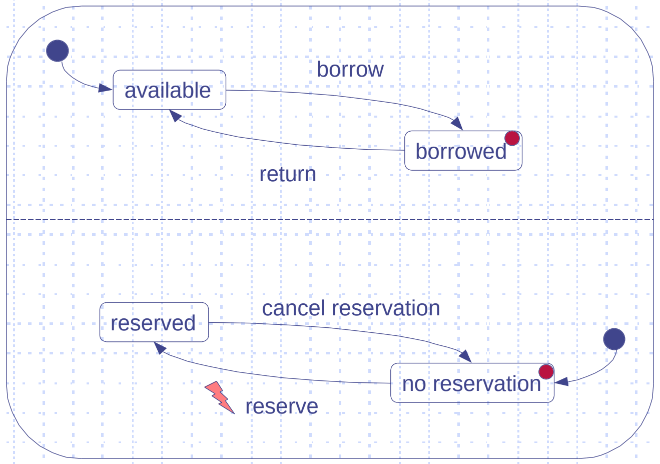
Concurrent active states



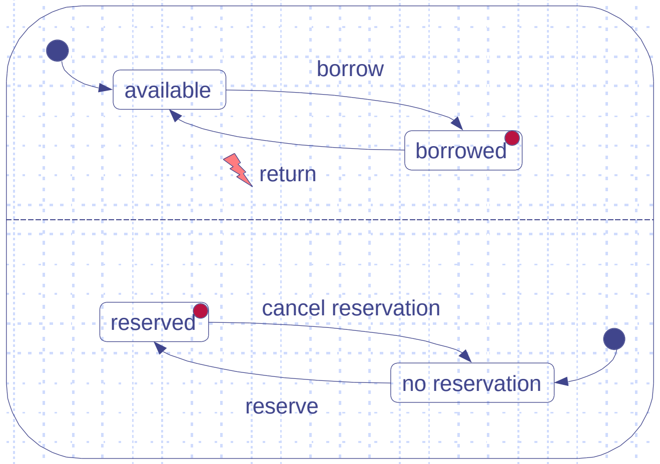
Concurrent active states



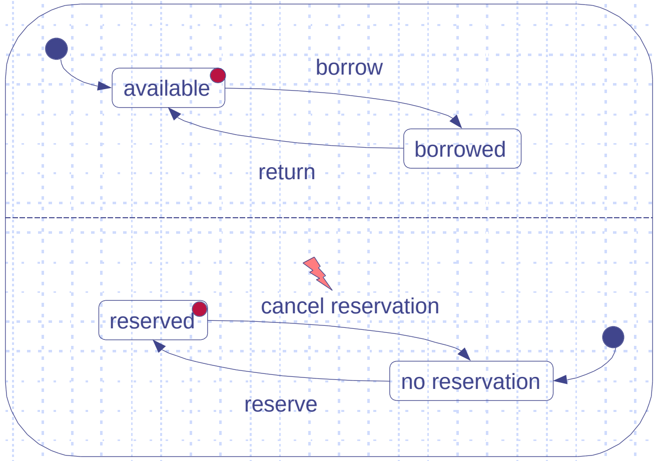
Concurrent active states



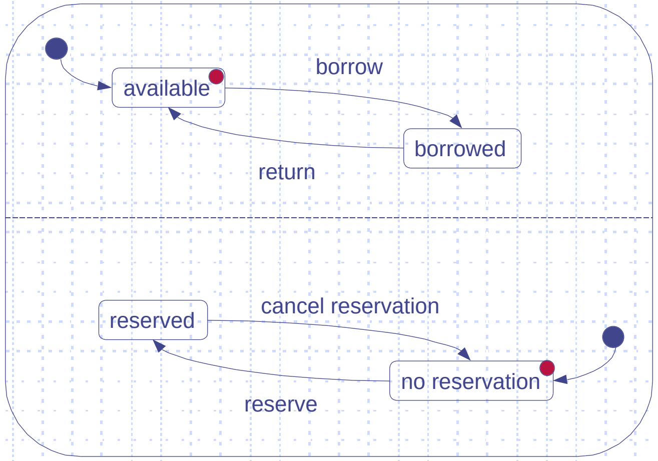
Concurrent active states



Concurrent active states

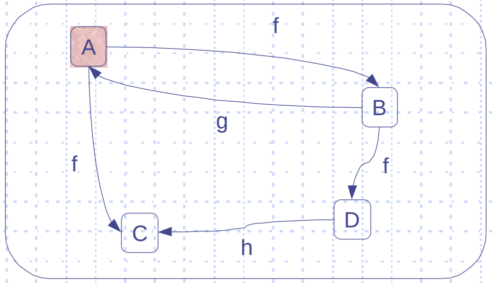


Concurrent active states



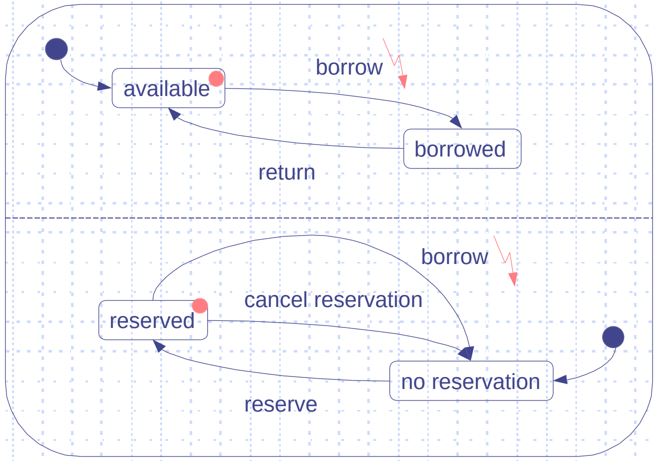
Transitions on the same event

- What happens if A is active and f occurs ?
- Non determinism in an or-state



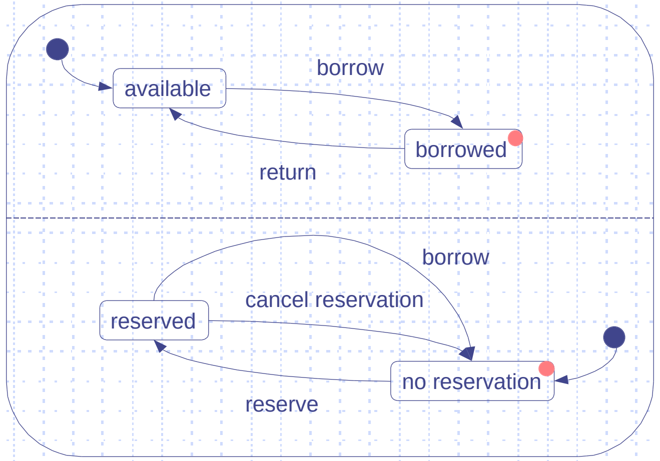
I S I

In concurrent composite states

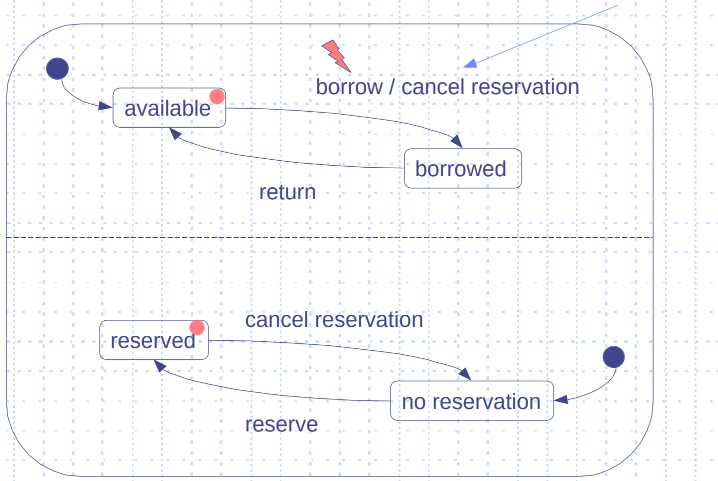


I S I

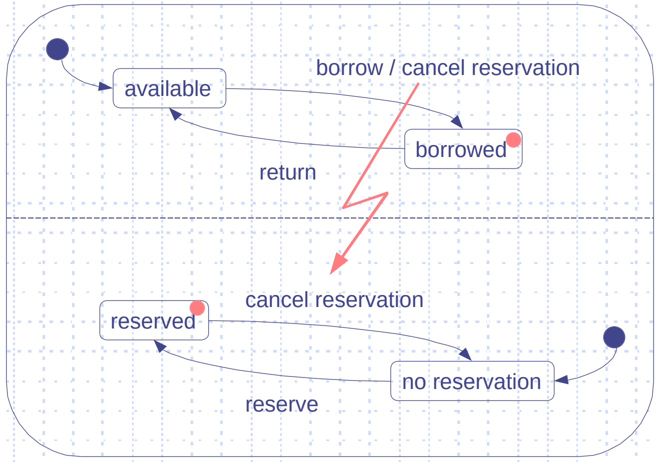
In concurrent composite states



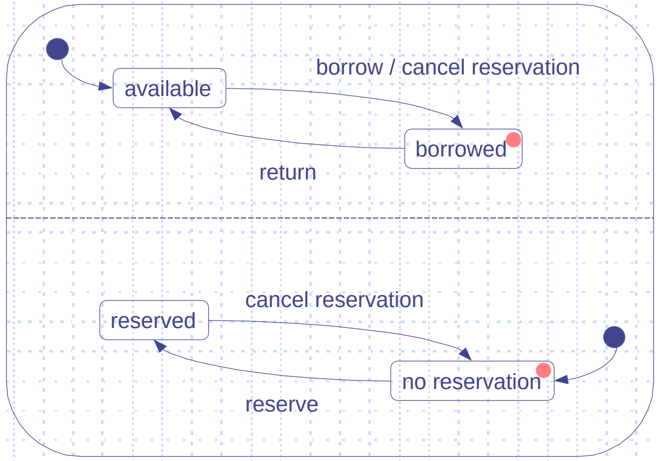
Events triggered by other events



Events triggered by other events

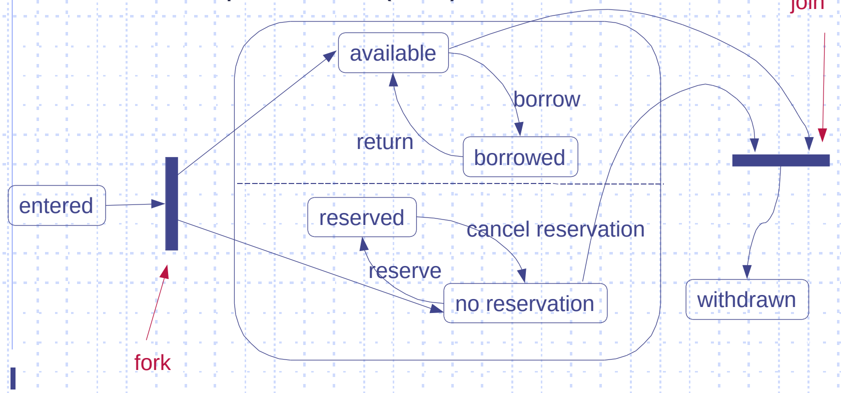


Events triggered by other events



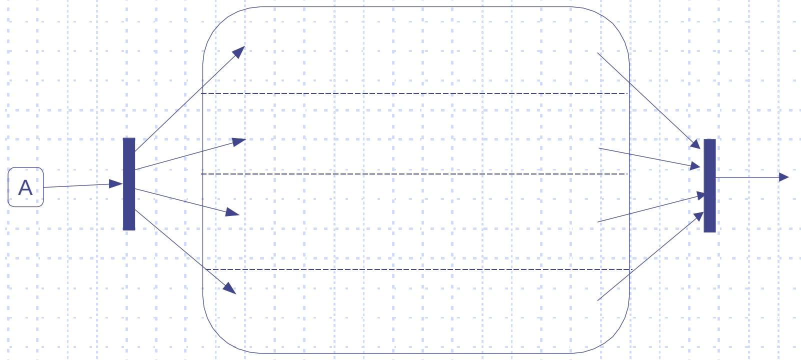
Complex transitions

- Create parallelism
- Terminate parallelism (wait)



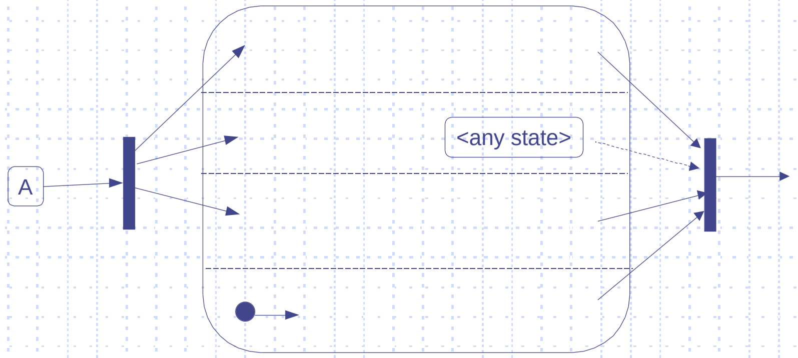
Formation rules for complex transitions

- Enter all concurrent subregions
- Leave all concurrent subregions



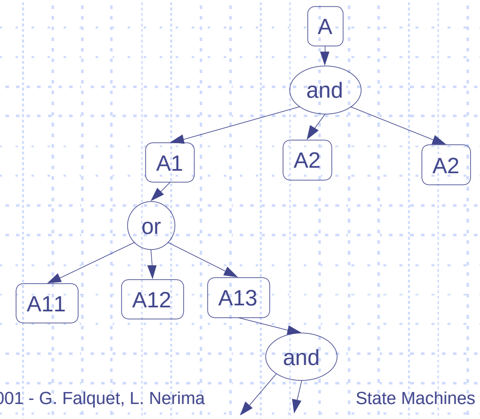
Default rules for complex transitions

- Activate the initial state.
- Leave the active substate.

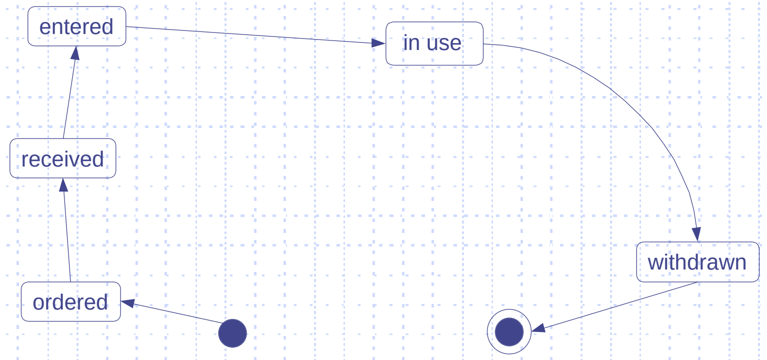


Structured design of state machines

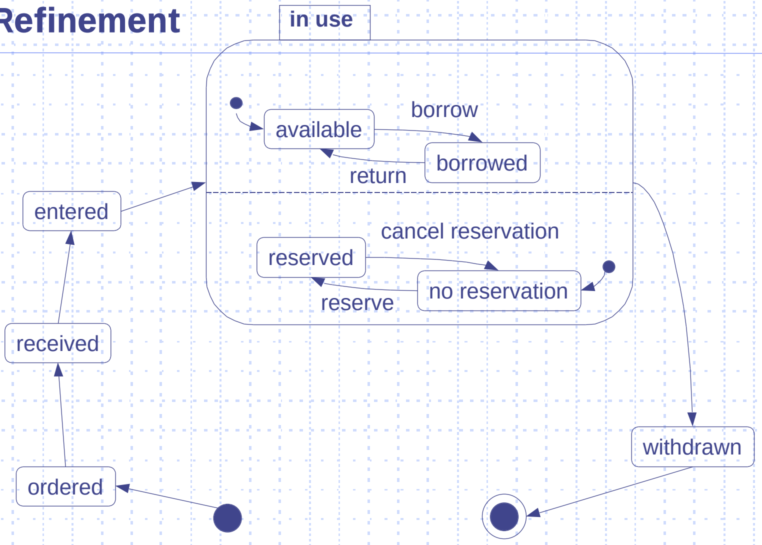
- Composition of and/or states (refinement)
- => no synchronisation problems (deadlocks, etc.)
- => expressiveness slightly limited



Example



Refinement



Design consideration

- State machines form the highest level of abstraction in the "dynamics" dimension
- State machines are not flow charts !
  - do not try to express algorithms, methods, computations, etc. with StM
- States must correspond to specific behaviour, conditions, etc. (avoid infinite modelling)
- Local dynamics: state machine of an object
- Global dynamics: all the state machines (with signals)

Implementation

- State machines are executable
  - => implementation with a state monitor/controller
  - (transaction monitors, real-time systems, ...)
- Generally: transformation into data structures and program code
- State ---> attribute value or link.
  - Book:borrowed == linked to a Loan object.
  - Report:approved == status = 'a'
- Transitions ---> execution of an operation/method.

State Machines and Use Cases (Douglass 2000)

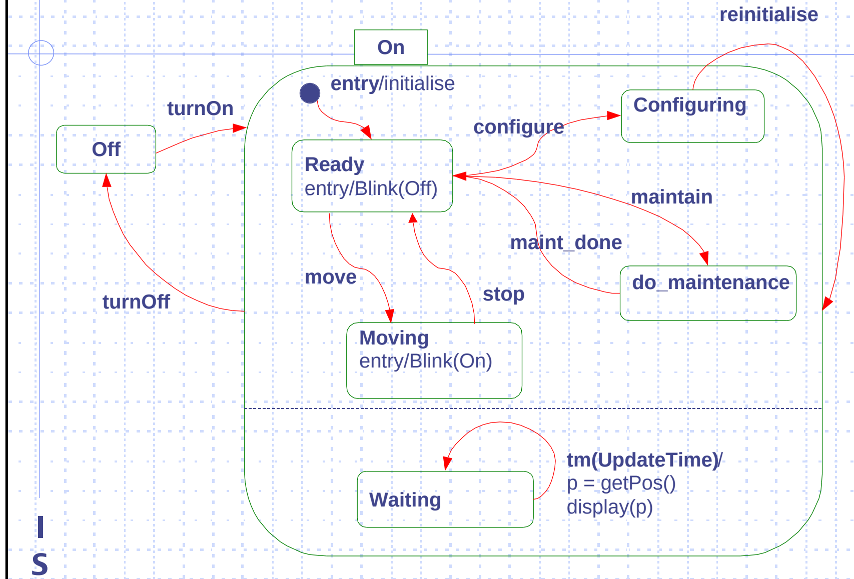
- Can represent all possible scenarios on a single diagram.
  - A scenario is a path through the state machine.
- Useful to elaborate complex protocols of actor-system interaction.
  - represent actor -> system messages as triggering events and conditions
  - represent system -> actor messages as actions
- Possible to execute (simulate) the state machine to check accuracy and completeness.

http://iamwww.unibe.ch/CHOOSE/Events/forum2k/douglass.pdf

Example

Interaction: The operator can enter commands to control a telescope system, subject to a number of constraints

- When the telescope is idle, the system may be configured, maintained, or commanded to move.
- The system will not accept a command when the telescope is currently moving except stop or turn off.
- Whenever the telescope is moving, monitored position is displayed in a blinking form.
- When the telescope is stopped, the monitored position is non-blinking.
- After configuration is complete, the user must reinitialise the system.
- Telescope position is displayed on a user-defined periodic basis.


reinitialise
On
entry/initialise
turnOn
Off
Configuring
configure
Ready
entry/Blink(Off)
maintain
maint_done
move
stop
do_maintenance
turnOff
Moving
entry/Blink(On)
Waiting
tm(UpdateTime)/
p = getPos()
display(p)