# FOUNDATIONS OF THE UML

Lecture 8: Statecharts

Joost-Pieter Katoen

MOVES

moves.rwth-aachen.de/i2/192

WS 2007/2008

December 19, 2007



## STATECHARTS

- MSCs: visual language for requirements
- Statecharts: graphical language to describe the behaviour of discrete-event systems
  (= automata + hierarchy + communication + time + concurrency)
- Developed by David Harel in 1987
  (professor at Weizmann Institute, and co-founder of i-Logix Inc.)
- Extensively used in embedded systems, automotive and avionics industry
- Supported by STATEMATE tool-set
- Adopted by the UML
- Variants like Stateflow used in MATLAB/Simulink (popular tool for embedded system design)

# WHAT ARE STATECHARTS?

[David Harel, 1987]:

" Statecharts constitute a visual formalism for:

- describing states and transitions in a modular way,
- enabling clustering [of states],
- orthogonality (i.e., concurrency),
- and refinement, and
- encouraging "zoom" capabilities for moving easily back and forth between levels of abstraction "

| Statecharts = | Mealy machines |
|---|---|
| + | state hierarchy |
| + | broadcast communication |
| + | orthogonality |



# MEALY MACHINES [Mealy, 1955]

A Mealy Machine (or: finite-state transducer) is an FSM that produces output on a transition, based on current input and state

A Mealy Machine $A = (Q, q_0, \Sigma, \Gamma, \delta, \omega)$

where:

- $Q$ is a finite set of states
- $q_0 \in Q$ is the initial state
- $\Sigma$ is the input alphabet
- $\Gamma$ is the output alphabet
- $\delta : Q \times \Sigma \longrightarrow 2^Q$ is transition function
- $\omega : Q \times \Sigma \longrightarrow \Gamma$ is output function

such that $|\delta(q,a)| \leq 1 \quad \forall_{\substack{q \in Q \\ a \in \Sigma}}$

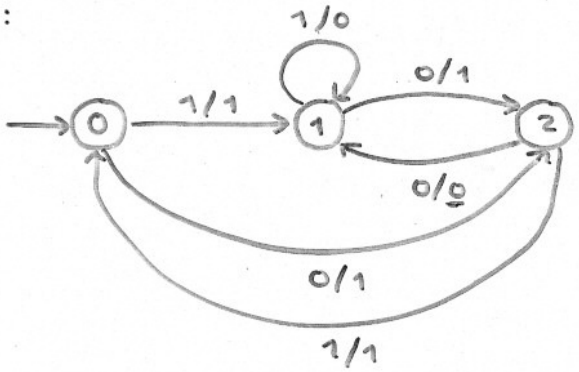
deterministic

# MEALY MACHINES

Mealy machines:
- no final states
- transitions produce output

  ⟹ no acceptance of input words, but generating output from input is important
- no nondeterministic states

Example:




$\omega(3,0) = 0$

(Note: in a Moore machine, output only depends on current state, i.e., $\omega: Q \to \Gamma$ ).

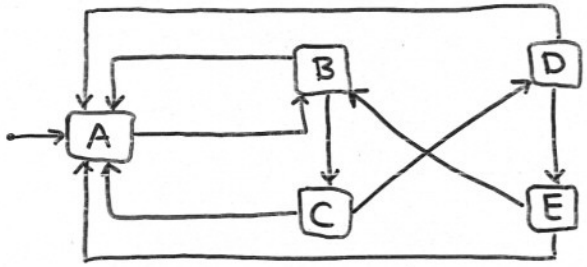


# LIMITATIONS OF MEALY MACHINES

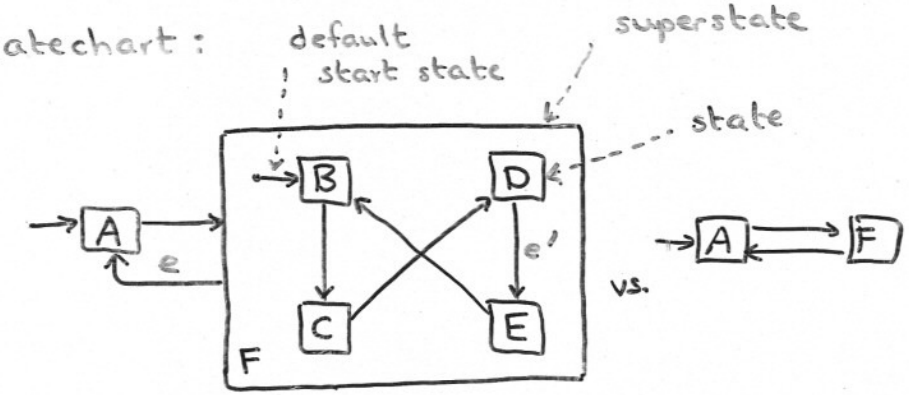
- No support for hierarchy
  - all states are arranged in "flat" fashion
  - no notion of substates
- For realistic systems, a complex transition structure and huge number of states are needed (= scalability problems)
  - yields unstructured state diagrams
- No notion of concurrency
  - need for modeling independent components
- No notion of communication
- No direct support for data

# SCALABILITY

State Machine:




Statechart:




State hierarchy yield modular, hierarchical and more structured models.

F is an ancestor of B, C, D, E

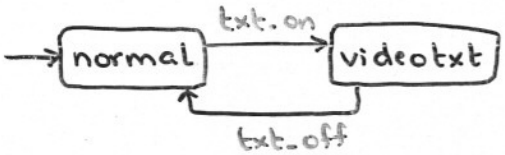
B, C, D, E are descendants of F



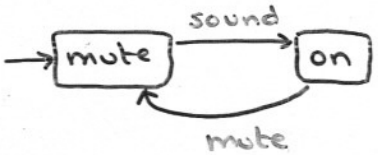
# ORTHOGONALITY

- Two independent components:

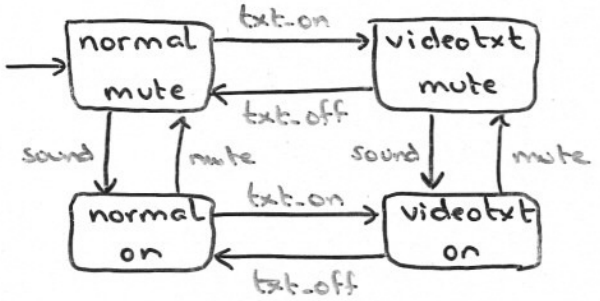
Image:




Sound:

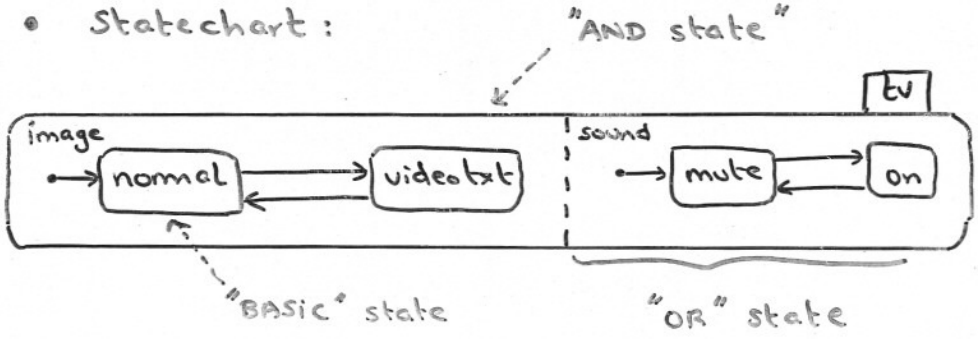



- Mealy machine:




number of states is exponential

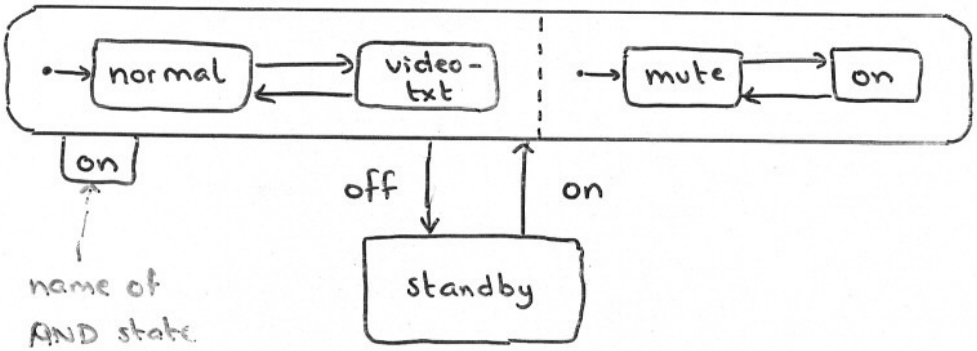
- Statechart:




Concurrency modeled by independence

# HIERARCHY

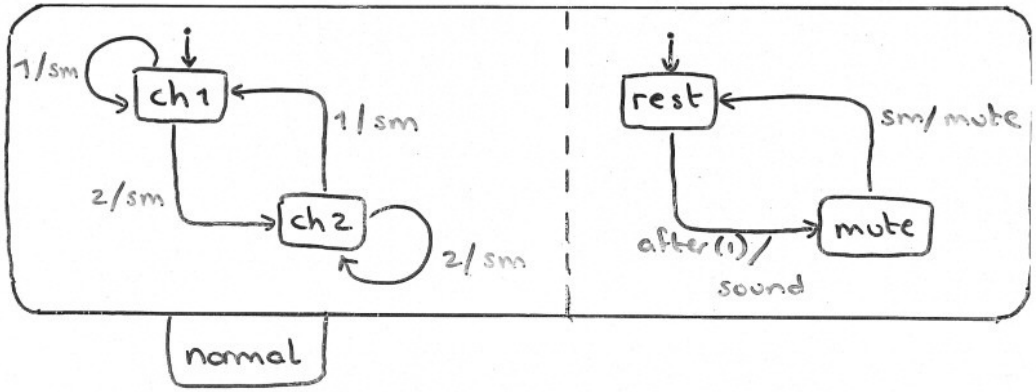
Switching on and off the television:




# BROADCAST

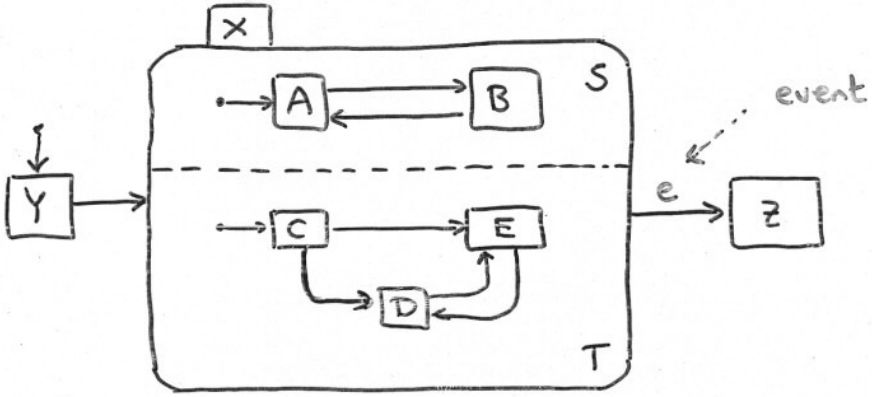
Output is a broadcast that can be received by any other component.

E.g., turn off sound on changing a channel:



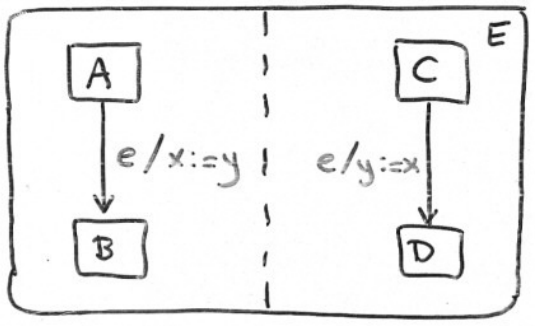



# CONCURRENCY




- As long as X is "active", S and T are "active"
- S is "active" when either A or B is "active"
- T is "active" if either C, D, or E is "active" (one of them)
- When X exits, both S and T exit
- When Y exits, X starts, S in A, T in C
- On the occurrence of event e, X exits (regardless of current state in S or T)

# SWAPPING TWO VARIABLES



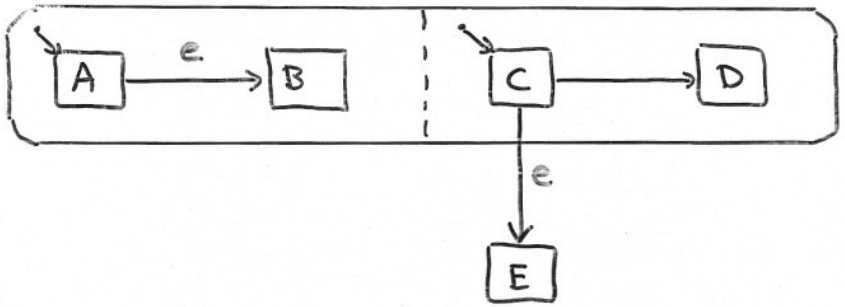

x, y are global variables

- if A and C are "active", x = 1, y = 2
- and event e occurs
- next status: B and D are "active"
  x = 2 and y = 1
- variables x and y are thus swapped

⇒ memory is <u>shared</u>, i.e., concurrent processes see changes to variables immediately

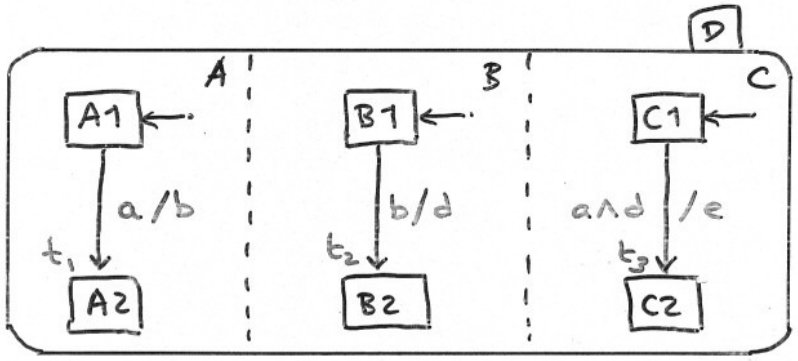


# PRIORITY




- What happens if event e occurs and states A and C are active?
- Add <u>priority</u> mechanism that decides whether inter-level (e.g. C → E)
  or intra-level (e.g. A → B)
  transitions prevail

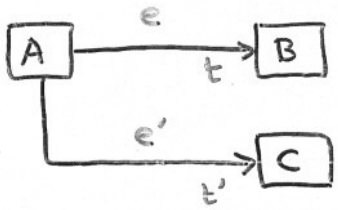
# ZERO RESPONSE TIME




- Let A1, B1 and C1 be active
- On occurrence of event a, a chain of reactions occurs: $t_1$ triggers $t_2$, and $t_2$ triggers $t_3$
- But: transitions $t_1, t_2, t_3$ occur at the same time as events do not take time (exception: after(d) event with $d \in \mathbb{R}_{>0}$)



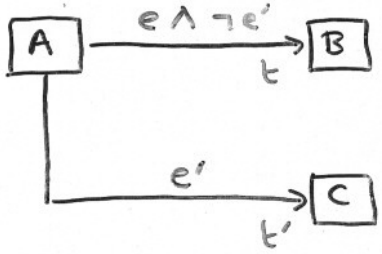
# NONDETERMINISM




when A is "active" and events e and e' are alive, next status is either B or C, but not both.

# NEGATION OF EVENTS

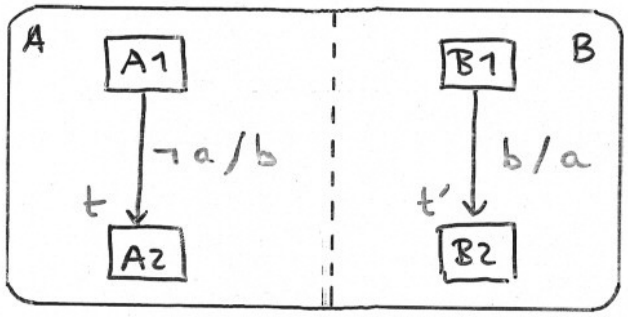
Priority of t' over t can be specified by negated events, i.e., the absence of events.



# A PARADOX



Negated events and zero response time yield the following paradox:

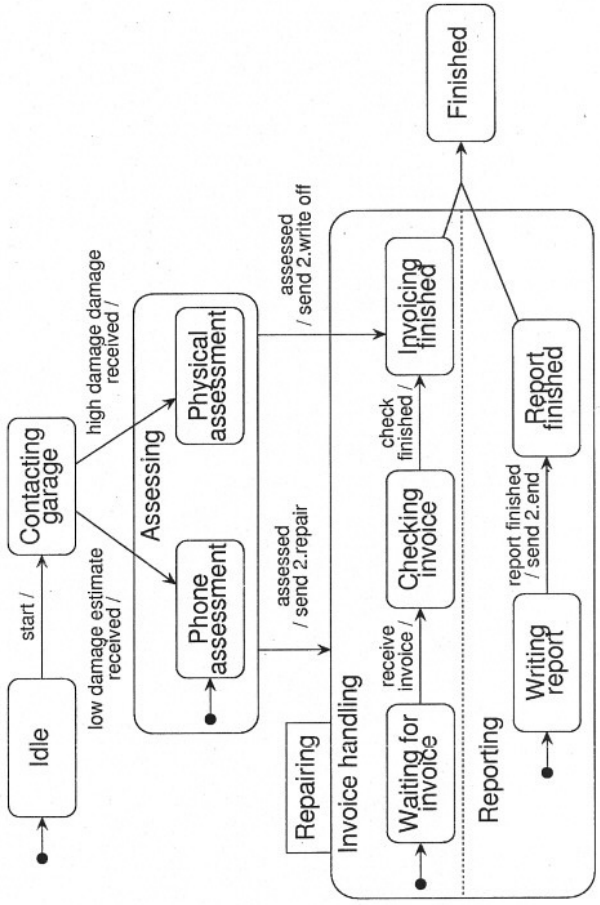



- Assume events a and b are not alive
- Transition t can be taking, generating event b
- Transition t' can be taken, generating a
- But then t should not have taken place since it is not enabled
- But then t' cannot be taken since b does not occur. Hence a not occurs and t





## A UML StateChart



17



# What is a UML StateChart?

A UML StateChart $SC = (\mathit{Nodes}, \mathit{Ev}, \mathit{Edges})$ with:

- A set *Nodes* of *nodes* structured in a tree
- A (finite) set *Ev* of *events*
  - there is a pseudo-event *after*$(d)$ denoting a delay of $d$ time units
  - $\perp \notin \mathit{Ev}$ stands for "no event required" (for this edge)
- A (finite) set *Edges* of edges (in fact, *hyperedges*)



18



# Syntactic sugar

*this is an elementary form; the UML allows more constructs that can be defined in terms of these basic elements*

| | |
|---|---|
| • Deferred events | simulate by regeneration |
| • Parametrised events | simulate by set of parameter-less events |
| • Activities that take time | simulate by start and end event |
| • Dynamic choice points | simulate by intermediate state |
| • Synchronization states | use a hyperedge with a counter |
| • History states | (re)define an entry point |

# More about nodes

- Nodes are structured in a *tree*
  - *children*$(x)$ is the set of children of node $x$
  - *root* is the unique root node
  - $x \trianglelefteq y$ means node $x$ is a descendant of node $y$
- Nodes are *typed*, *type*$(x) \in \{$ BASIC, AND, OR $\}$ such that:
  - the root node is of type OR
  - each leaf node is of type BASIC
  - each child of an AND-node is of type OR
  - each OR-node has a *default* (initial) node





# Edges

An *edge* is a tuple $(X, e, g, A, Y)$, notation $X \xrightarrow{e[g]/A} Y$, where

- $X$ and $Y$ are non-empty sets of nodes
- $X$ is the source and $Y$ the target
- $e \in$ *Ev* is the trigger event
- $g$ is a guard, i.e., a Boolean expression
- $A$ is a set of actions (such as *send* $j.e$ or $v :=$ *expr*)

"if currently in $X$ and $g$ holds, on occurrence of $e$,
actions $A$ can be performed while evolving into $Y$"