Computational Models for Embedded Systems

Vescan Andreea, PHD, Assoc. Prof.

Faculty of Mathematics and Computer Science
Babeș-Bolyai University
Cluj-Napoca

2025-2026

Lecture 9: Finite State Machines (b)

Software Systems Verification and Validation

"Tell me and I forget, teach me and I may remember, involve me and I learn."

(Benjamin Franklin)

Outline

- Hierarchical FSM models
- StateCharts
- Synchronous vs. Asynchronous FSMs

Hierarchical FSM models

- Problem: how to reduce the size of the representation?
- Harel's classical papers on StateCharts (language) and bounded concurrency (model): 3 orthogonal exponential reductions
- Hierarchy:
  - state **a** "encloses" an FSM
  - being in **a** means FSM in **a** is active
  - states of **a** are called OR states
  - used to model pre-emption and exceptions
- Concurrency:
  - two or more FSMs are simultaneously active
  - states are called AND states
- Non-determinism:
  - used to abstract behavior

Models Of Computation for reactive systems

- Main MOCs:
- **Communicating Finite State Machines**
- Dataflow Process Networks
- Petri Nets
- Discrete Event
- Codesign Finite State Machines

- Main languages:
- **StateCharts**
- Esterel
- Dataflow networks

StateCharts

- An extension of conventional FSMs
- Conventional FSMs are inappropriate for the behavioral description of complex control
  - flat and unstructured
  - inherently sequential in nature
- StateCharts supports repeated decomposition of states into sub-states in an AND/OR fashion, combined with a synchronous (instantaneous broadcast) communication mechanism.

State Decomposition

- OR-States have sub-states that are related to each other by exclusive-or
- AND-States have orthogonal state components (synchronous FSM composition)
  - AND-decomposition can be carried out on any level of states (more convenient than allowing only one level of communicating FSMs)
- Basic States have no sub-states (bottom of hierarchy)
- Root State : no parent states (top of hierarchy)

StateCharts Syntax

- The general syntax of an expression labeling a transition in a StateChart is

  e[c]/a ,where
  - **e** is the event that triggers the transition
  - **c** is the condition that guards the transition (cannot be taken unless c is true when e occurs)
  - **a** is the action that is carried out if and when the transition is taken
- For each transition label:
  - event condition and action are optional
  - an event can be the changing of a value
  - standard comparisons are allowed as conditions and assignment statements as actions

StateCharts Actions and Events

- An action **a** on the edge leaving a state may also appear as an event triggering a transition going into an orthogonal state:
  - a state transition broadcasts an event visible immediately to all other FSMs, that can make transitions immediately and so on
  - executing the first transition will immediately cause the second transition to be taken simultaneously
- Actions and events may be associated to the execution of orthogonal components : start(A) , stopped(B)

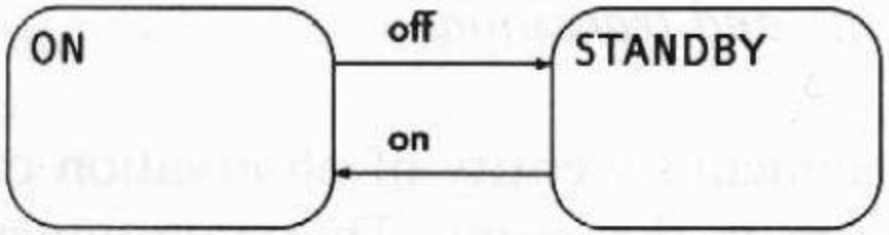
Graphical Hierarchical FSM Languages

- Multitude of commercial and non-commercial variants:
  - StateCharts, UML, StateFlow, …
- Easy to use for control-dominated systems
- Simulation (animated), SW and HW synthesis
- Original StateCharts have problems with causality loops and instantaneous events:
  - Circular dependencies can lead to paradoxes
  - Behavior is implementation-dependent
  - Not a truly synchronous language
- Hierarchical state necessary for complex reactive system specification

Example
Television set with remote control

First Concept: Hierarchy

- Hierarchy or depth in states, and interrupts.
- This is achieved by drawing states as boxes that contain other boxes as sub-states.
- The television set can be in two states: **ON** and **STANDBY**. Switching between them is done by pushing the **on** and **off** buttons, generating the **on** and **off** events:



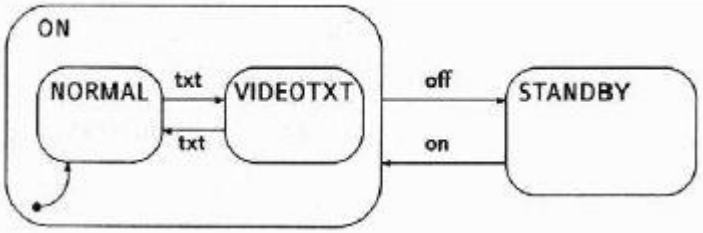
Example

Television set with remote control

First Concept: Hierarchy

Zooming into ON

- In state ON, the tv set can be in two sub-states: NORMAL and VIDEOTEXT:



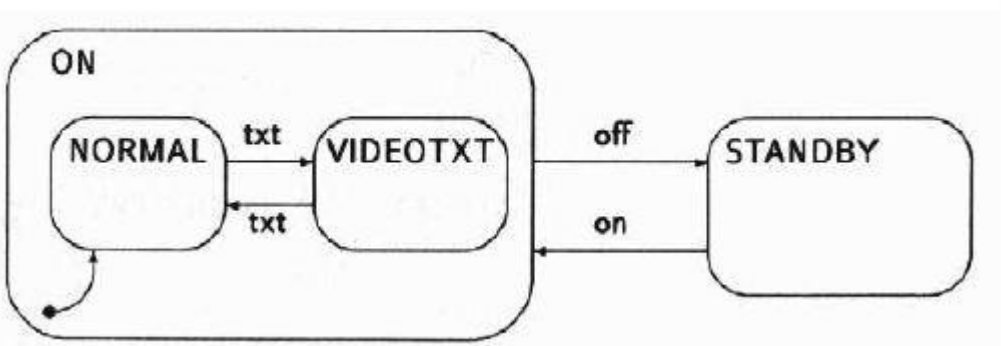
- The → arrow leading to NORMAL specifies which sub-state should be entered when the higher level state ON is entered, namely NORMAL; this state is also called the initial state (within ON)
- Note: To be complete, one of the states ON and OFF would also need to be labeled as initial state (at top-level).

Example

Television set with remote control

First Concept: Hierarchy

Superstates



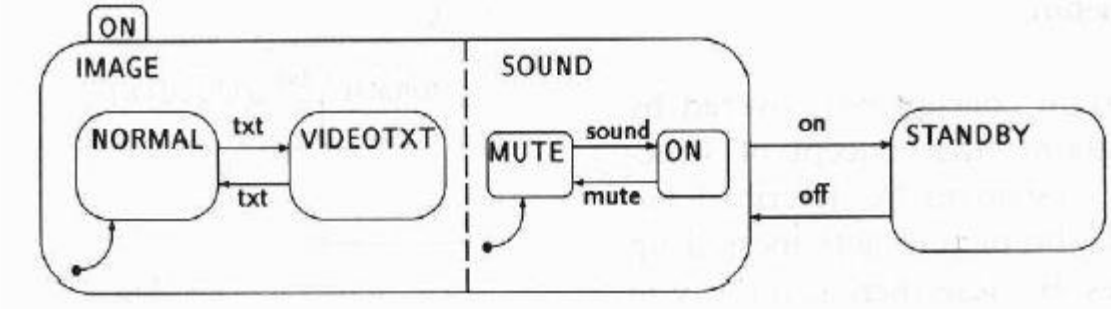
- When in ON, and an event off is generated:
  - 1. State ON (incl. all its sub-states) is left - so this acts like an interrupt
  - 2. Control switches to state STANDBY.
- In this way interrupts are handled without cluttering the picture with arrows

Example

Television set with remote control

Second Concept: Orthogonality

- Two independent components can be put together into an AND-state, separated by a dotted line



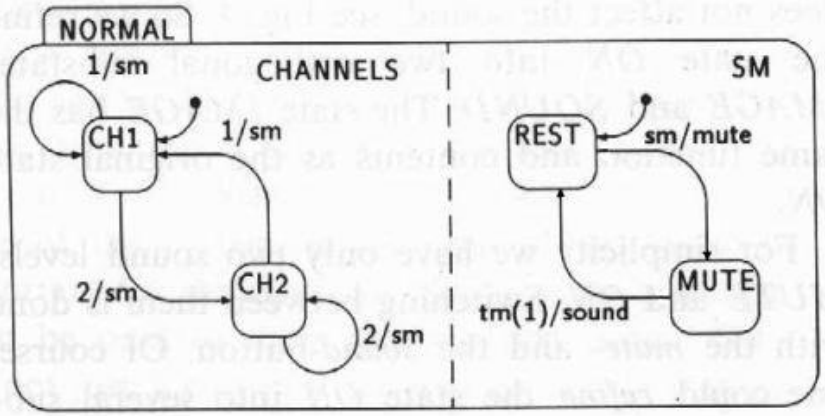
- Being in an AND-state means being in all of its immediate sub-states at the same time.
- **This prevents the exponential blow-up familiar from composing FSMs in parallel.**

Example

Television set with remote control

Third Concept: Broadcast

- In our case we split state **NORMAL** in two orthogonal components **CHANNEL**, for selecting channels, and **SM** for switching to mute:



- Note special time-out event tm(1)

Example

Television set with remote control

Third Concept: Broadcast

Actions and Transitions

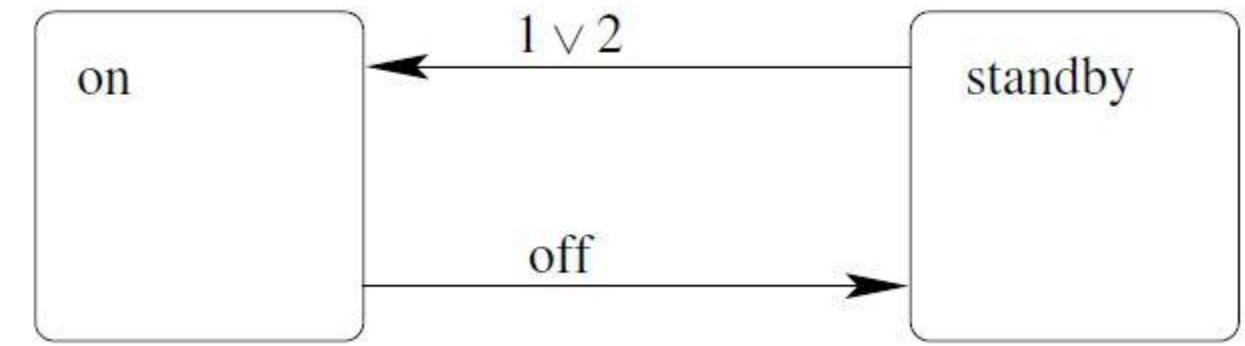
- Orthogonal components can communicate by generating events which are broadcast.
- This can be done in a time-dependent manner: introducing the generation of events e/a1; ... ; and time-out events tm(1), . . .

- In general, the label of a transition consists of two parts:
  - 1. Trigger, determining if and when a transition will be taken
  - 2. Action, performed when a transition is taken.
- This action is the generation of a set of events.

Example
Television set with remote control

Fourth concept: Compound events

- When in state STANDBY, dependent on whether one presses button 1 or 2, one makes sure to switch to states CH1 or CH2 in ON. This is indicated as follows:



Example

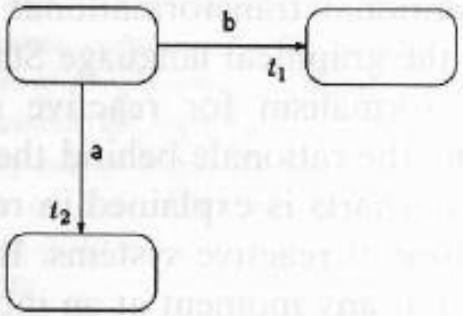
Television set with remote control

Fourth concept: Compound events

Transition Labels

- In general, one can label transitions by compound events such as

$$(\neg a \wedge b) \vee c, a \wedge b, c \vee d, \neg a, etc$$



- To express priority of event **b** over event **a** in this statechart: can replace **a** by

$$a \wedge \neg b$$

Example
Television set with remote control

Summa Summarum

- In a nutshell, one may say with David Harel:

- Statecharts = Mealy Machines
  - + depth
  - + orthogonality
  - + broadcast
  - + data

Example
Time in Statecharts

Characteristics of Real-Time Systems

- The **environment** can deliver data continuously, for example via temperature sensor.
- Data can be delivered from different sources simultaneously and must therefore be processed in parallel.
- The time scale is fast by human standard (milli seconds instead of seconds).
- The system must react in time and accurately on input from the environment.

Example
Time in Statecharts

Time

- The elementary unit of observation in a reactive system is the event.
- The environment sends events to the system to trigger computations, the system reacts to the environment by sending, or generating, events.
- Events are also means of communication between parts of a system.

Example
Time in Statecharts

Time

- Because one wants to specify reactive systems at the highest level of abstraction in a discrete fashion, events are discrete signals, occurring at a point in time.
- Events have no duration; they are generated from one state to another.
- Hence, transitions have a discrete uninterruptable nature and all time is spent in states.

Example
Time in Statecharts

Reason

- In a reactive system new inputs may arrive at any moment. Therefore the current state it is in should be always clear. Since transitions have no duration, there are no “transient" periods in between states.

- Therefore, the reaction on a possible input is always well defined.
  - Of course this is an abstraction from reality. (At deep levels of electronic implementations, one encounters levels where discrete reasoning makes no sense anymore)

- Statecharts is meant to be a high level specification language, where this abstraction can be maintained and is appropriate.

Example

Time in Statecharts

Reaction Time

- We know that transitions have no duration, but when do they take place, relative to the trigger? And:

  How long does it take the system to compute a reaction upon an external event?

- For transformational systems this is easy - the only important distinction is between finite and infinite values (corresponding to a final state or no final state)
- For reactive systems this is not enough:

  We have to know when an output occurs relative to the events in the input sequence ➔

  One has to determine *the reaction time of a sequence*.

Example
Time in Statecharts

Reaction Time in Statecharts

Question: What's the reaction time of a reactive system upon an external event in Statecharts?

- Possibility 1
- Specify a concrete amount of time for each situation.
  - This forces us to quantify time right from the beginning.
  - This is clumsy, and not appropriate at this stage of specification where one is only interested in the relative order and coincidence of events.

Example
Time in Statecharts

Reaction Time in Statecharts

Question: What's the reaction time of a reactive system upon an external event in Statecharts?

- Possibility 2
- Fix reaction time between trigger a and corresponding action a within e/a (the label of a transition) upon 1 time unit.
  - Doesn't work: Upon refining question/answer to a question/consult and a consult/answer transition, there's a change of time, which may have far reaching consequences (e. g., because of tm(n)-events)
- ➔ A fixed execution time for syntactic entities (transitions, statements, etc.) is not flexible enough.

Example
Time in Statecharts

Reaction Time in Statecharts

Question: What's the reaction time of a reactive system upon an external event in Statecharts?

- Possibility 3
- Leave things open
  - Say only that execution of a reaction takes some positive amount of time, and see at a later stage (closer to the actual implementation) how much time things take.
  - Clumsy, introduces far too much nondeterminism.

Example
Time in Statecharts

Solution

Summary: We want the execution time associated to reactions to have following properties:

- 1. It should be accurate, but not depending on the actual implementation.
- 2. It should be as short as possible, to avoid artificial delays.
- 3. It should be abstract in the sense that the timing behavior must be orthogonal to the functional behavior.

- ➔ Only choice that meets all wishes is zero reaction time.

Example
Time in Statecharts

Problems Disappear

- As a result all objections raised w. r. t. the possibilities mentioned on the previous page are met!
  - Now, for instance, upon refining transition *question/answer* from previous page into two transitions, the reaction time of this refinement is the same as that of the original transition.
  - Objection 3 is resolved, too.
  - Finally, also objection 1 is met, because 0 + 0 = 0!

Example
Time in Statecharts

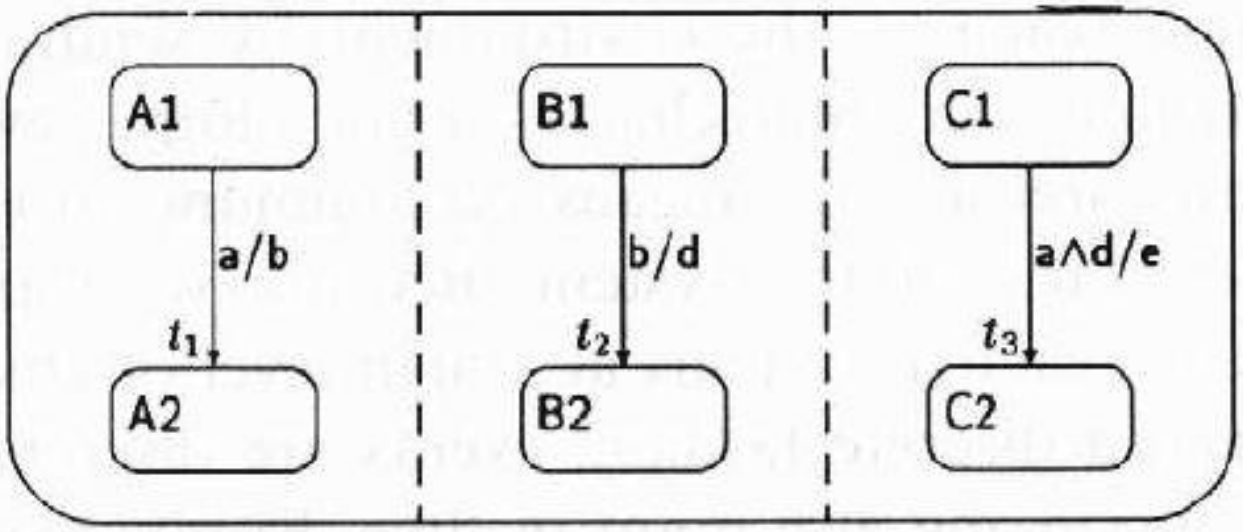
Berry's synchrony hypothesis

- This choice, that the reaction time between a trigger and its event is zero, is called **Berry's synchrony hypothesis.**
- Is this implementable? No, a real computation takes time.
- However, in actual implementation this means:
  - **The reaction comes before the next input arrives**, or, put another way,
  - **Reactions are not infinitely fast, but fast enough**.

Example
Time in Statecharts

Example

See the following figure:



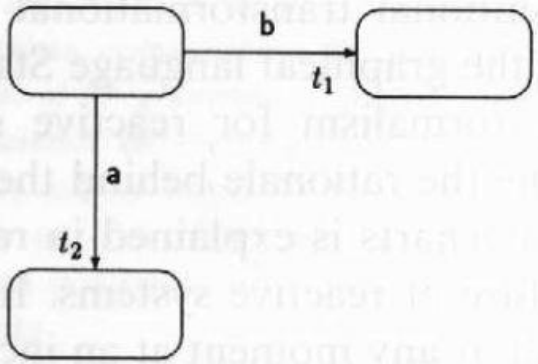
- A consequence is that transition t3 is taken!!

Example

Time in Statecharts

Negations and paradoxes

- Idea of immediate reaction works fine as long as transitions only triggered by primitive events, or conjunctions and disjunctions of them.
- However, one also needs **negations of events** to trigger a transition.
- Example: To specify priority between (reacting on) event **a** and event **b**

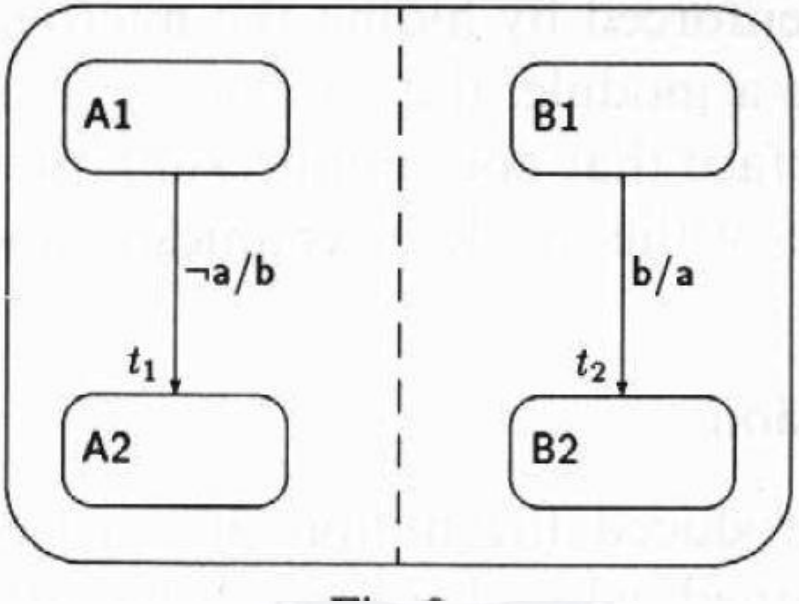


Example

Time in Statecharts

Problem: Grandfather Paradoxon

What semantics to give to this Statechart?



Example

Time in Statecharts

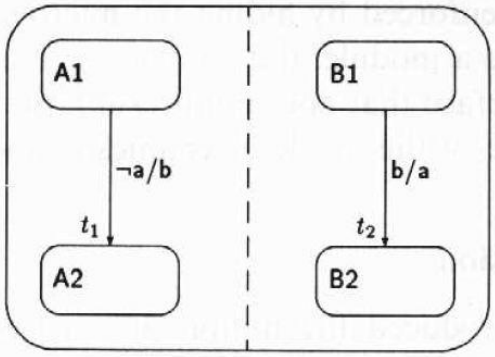
Problem: Grandfather Paradoxon

What semantics to give to this Statechart?

- It's solution is to order event occurrences causally, with later events not influencing earlier events:

$$\neg a \leq b \leq a$$

- Note here: This causal order has nothing to do with the passage of time; it merely refers to causal chains within one time step.



Example
Time in Statecharts

Solution

- Introduce two levels of time
  - Macro steps, for counting time, (these are observable) time steps, and
  - Micro steps, which describe the causal chain within reactions. Every macro-step is then divided in an arbitrary but finite number of micro-steps.

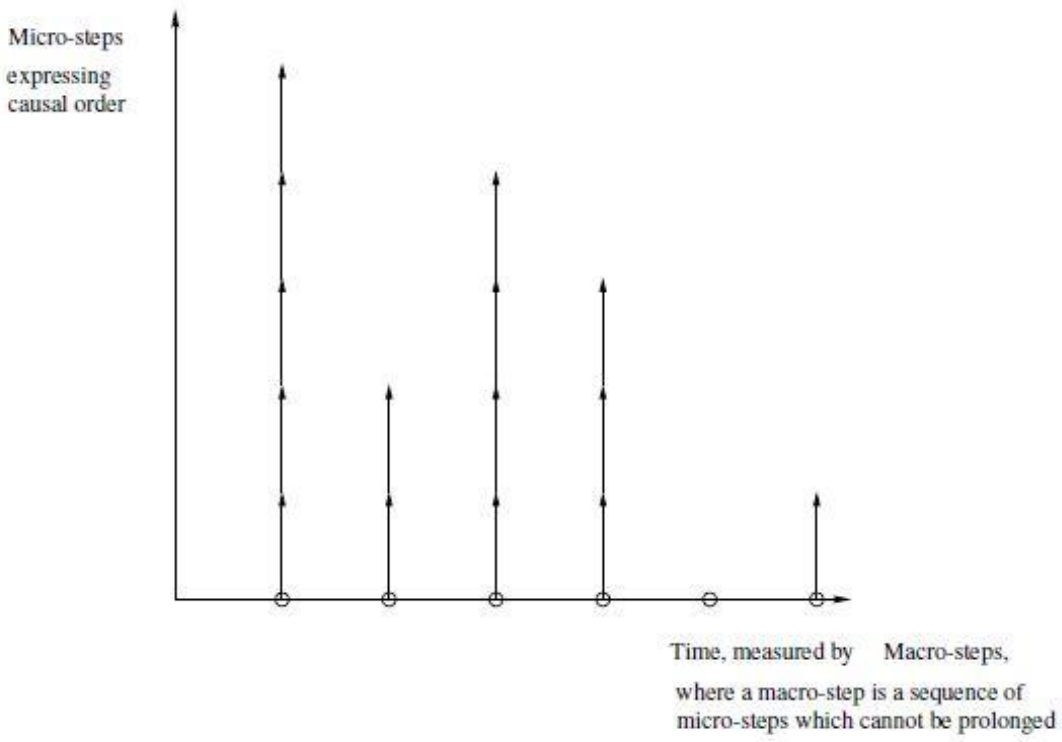
- This sequence of micro-steps has only an operational meaning.

Example

Time in Statecharts

The STATEMATE Semantics

- This leads to a semantics of the following form:



Example

Time in Statecharts

The STATEMATE Semantics

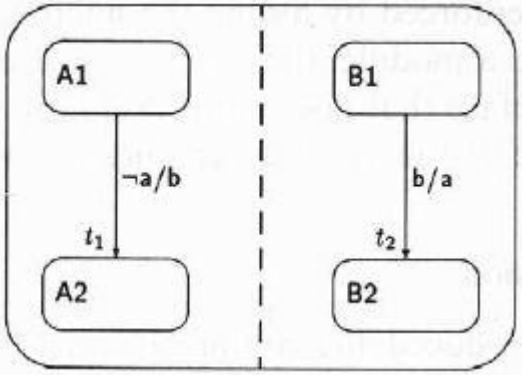
- Macro-steps are observable steps ➔ $^{O}_{I}$
- Each macro-step is a sequence of micro-steps, that is ordered causally; one micro-step can never influence previous micro-steps.
- In Statecharts as implemented by STATEMATE (“Harel-Statecharts"):
  - Causality is trivially obtained because in STATEMATE events generated in one step are only available in the next step, and only for that one.
  - i. e., there is no causality within one step.

Example

Time in Statecharts

Problems with STATEMATE Semantics

- The problem with macro-steps is that they lead to a globally inconsistent semantics.



$$S_1 \Rightarrow_{\phi}^{b} S_2 \Rightarrow_{b}^{a} S_3$$

- Here absence of triggers generates presence of triggers, which violates their absence within the same step.

Synchronous vs. Asynchronous FSMs

- Synchronous (Esterel, StateCharts):
  - communication by shared variables that are read and written in zero time
  - communication and computation happens instantaneously at discrete time instants
  - all FSMs make a transition simultaneously (lock-step)
  - may be difficult to implement
  - multi-rate specifications
  - distributed/heterogeneous architectures
- A-synchronous FSMs:
  - free to proceed independently
  - do not execute a transition at the same time (except for CSP rendezvous)
  - may need to share notion of time: synchronization
  - easy to implement

Finite State Machines

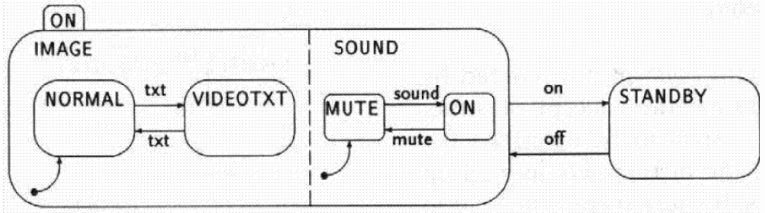
Composition of State Machines

- Concurrent composition:
  - Side-by-side
  - Cascade
  - Feedback

[LS12] Book: Lee,Seshia, Chapter 5, Composition of State Machines

Hierarchical FSMs + Synchronous Composition: Statecharts [Harel 87]

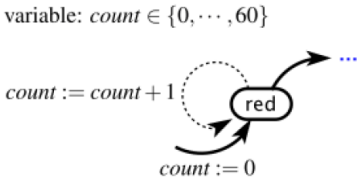
- Modeling with
  - Hierarchy (OR states)
  - Synchronous composition (AND states)
  - Broadcast (for communication)



Example due to Reinhard von Hanxleden

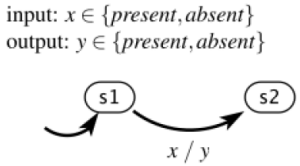
When does a reaction occur?

- When a reaction occurs is not specified in the state machine itself. It is up to the environment.



- This traffic light controller design assumes one reaction per second. This is a *time-triggered model*.

When does a reaction occur?



- Suppose all inputs are discrete and a reaction occurs when any input is present. Then the above transition will be taken whenever the current state is s_1 and x is present.
- This is an *event-triggered model*.

References

- **System Modelling (IL2202),** Axel Jantsch
  - https://www.kth.se/student/kurser/kurs/IL2202
  - http://jantsch.se/AxelJantsch/
- **Design of Embedded Systems: Models, Validation and Synthesis,** Alberto Sangiovanni-Vincentelli
  - https://inst.eecs.berkeley.edu/~ee249/fa07/

- [LS12] E.Lee, S. Seshia, Introduction to Embedded Systems - A Cyber-Physical Systems Approach, 2012 (Chapter 5).
- [Har87] David Harel, Statecharts: a visual formalism for complex systems, David Harel, Science of Computer Programming, 1987
- [HN96] David HAREL, Amnon NAAMAD, The STATEMATE semantics of Statecharts, ACM Transactions on Software Engineering and Methodology (TOSEM), Vol 5 (4), pp. 293-333, 1996

CMES – Today Bring it All Together

Finite State Machines (b)

- Hierarchical FSM models
  - Hierarchy:
    - state **a** "encloses" an FSM
    - being in **a** means FSM in **a** is active
    - states of **a** are called OR states
    - used to model pre-emption and exceptions
  - Concurrency:
    - two or more FSMs are simultaneously active
    - states are called AND states
  - Non-determinism:
    - used to abstract behavior

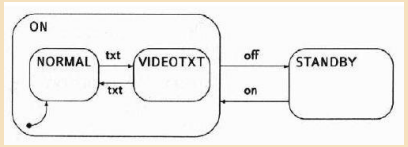
StateCharts

An extension of conventional FSMs mechanism.

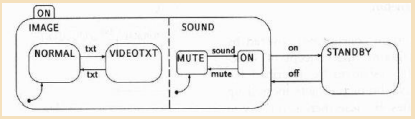
- StateCharts supports repeated decomposition of states into sub-states in an AND/OR fashion, combined with a synchronous (instantaneous broadcast) communication mechanism.

Television set with remote control

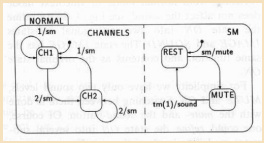
First Concept: Hierarchy



Second Concept: Orthogonality



Third Concept: Broadcast



Thank You For Your Attention!

- ExitTicket
- Mentimeter
  - menti.com
  - Code:

Next Lecture

- Petri nets

Software Systems Verification and Validation

"Tell me and I forget, teach me and I may remember, involve me and I learn."

(Benjamin Franklin)