

Embedded Systems

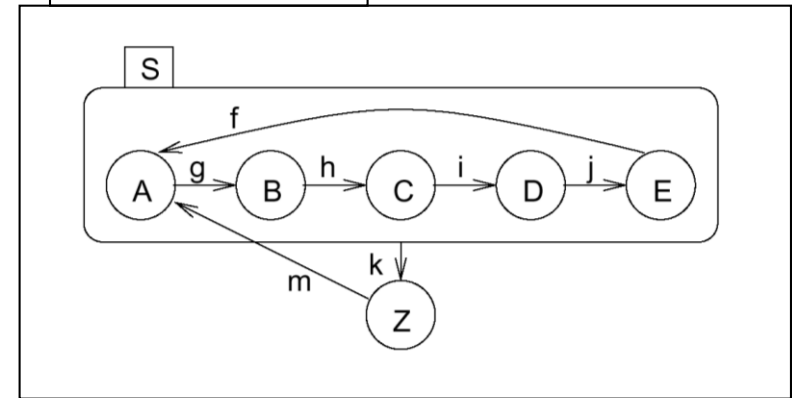
3



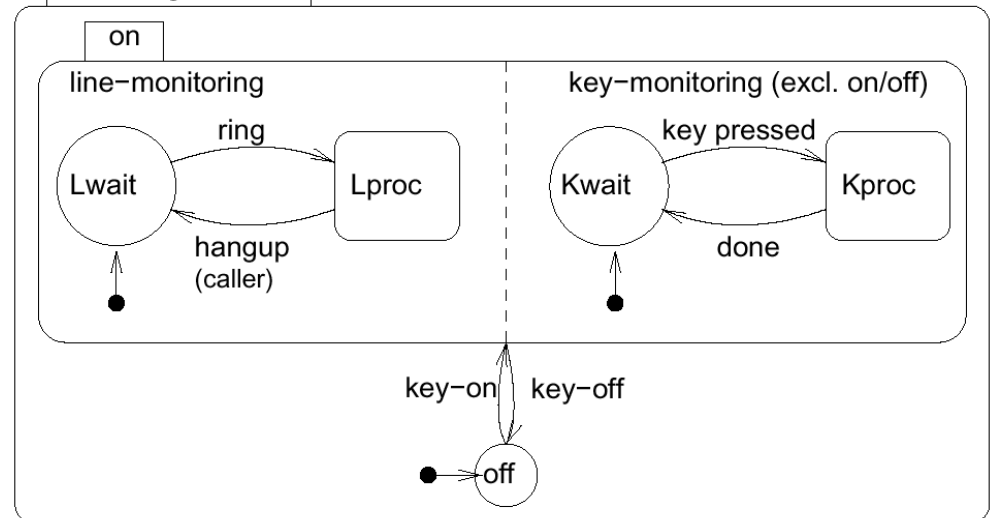
Statechart SC

REVIEW: StateCharts

- Hierarchy
- Concurrency



answering-machine



REVIEW: General form of edge labels



Meaning:

- Transition may be taken, if event occurred in last step and condition is true
- If transition is taken, then reaction is carried out.

Conditions:

- Refer to values of variables

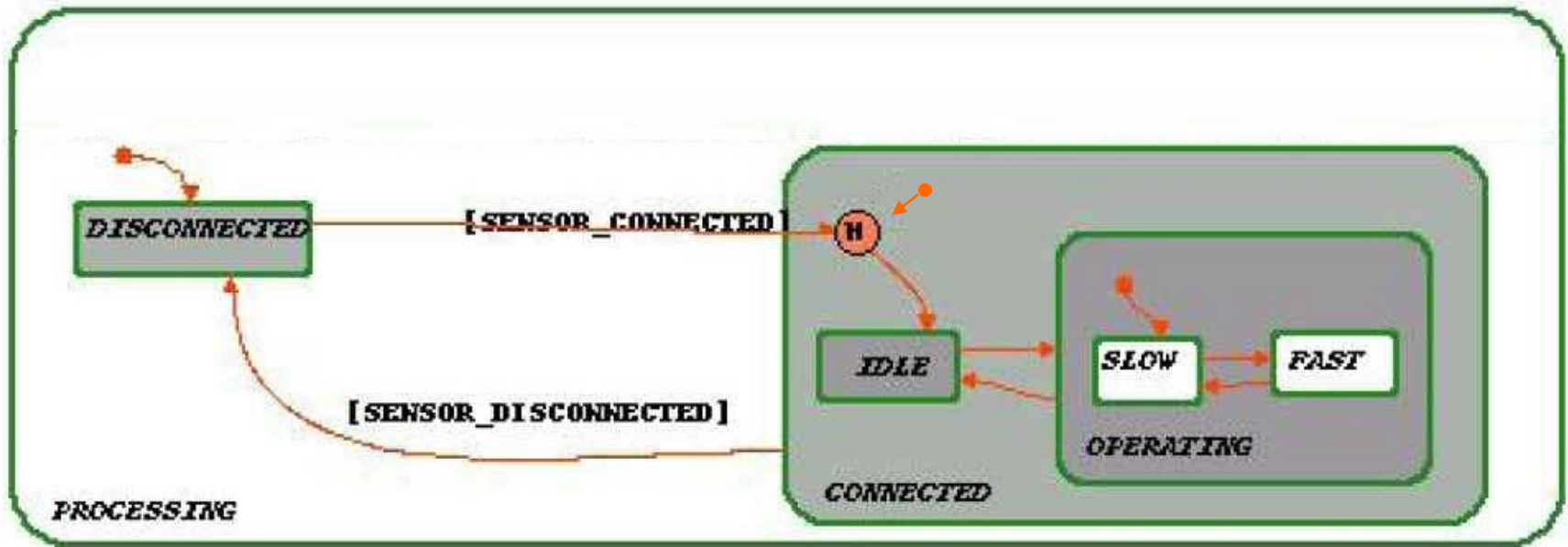
Actions:



- Can either be assignments for variables or creation of events

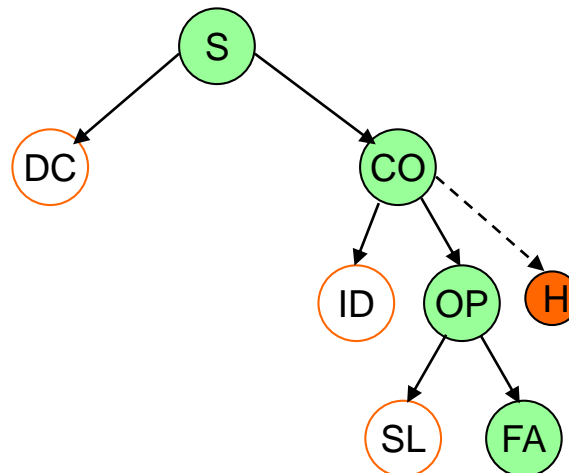
Example:

- `a & [x = 1023] / overflow; x:=0`

REVIEW: History and deep history



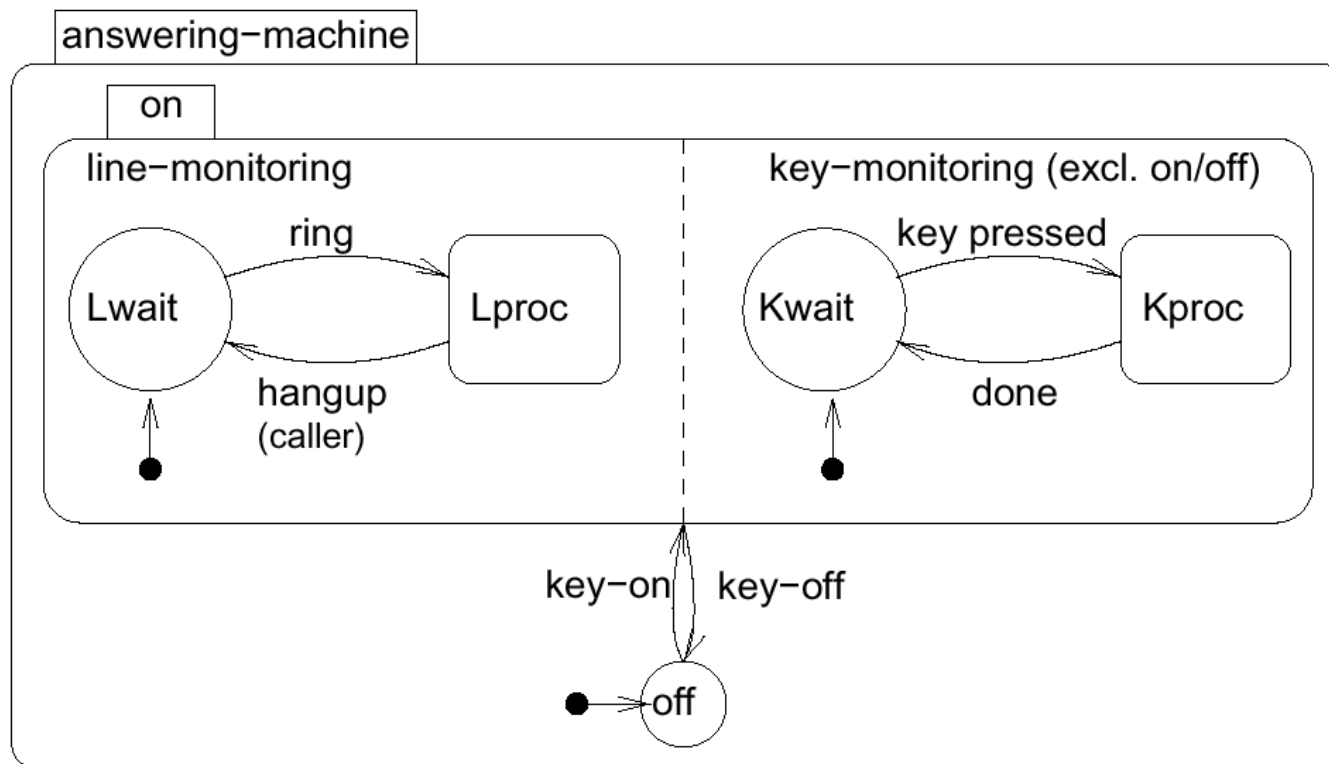
-  Default states
-  Active states



History connectors remember states **at the same level** as the history connector!

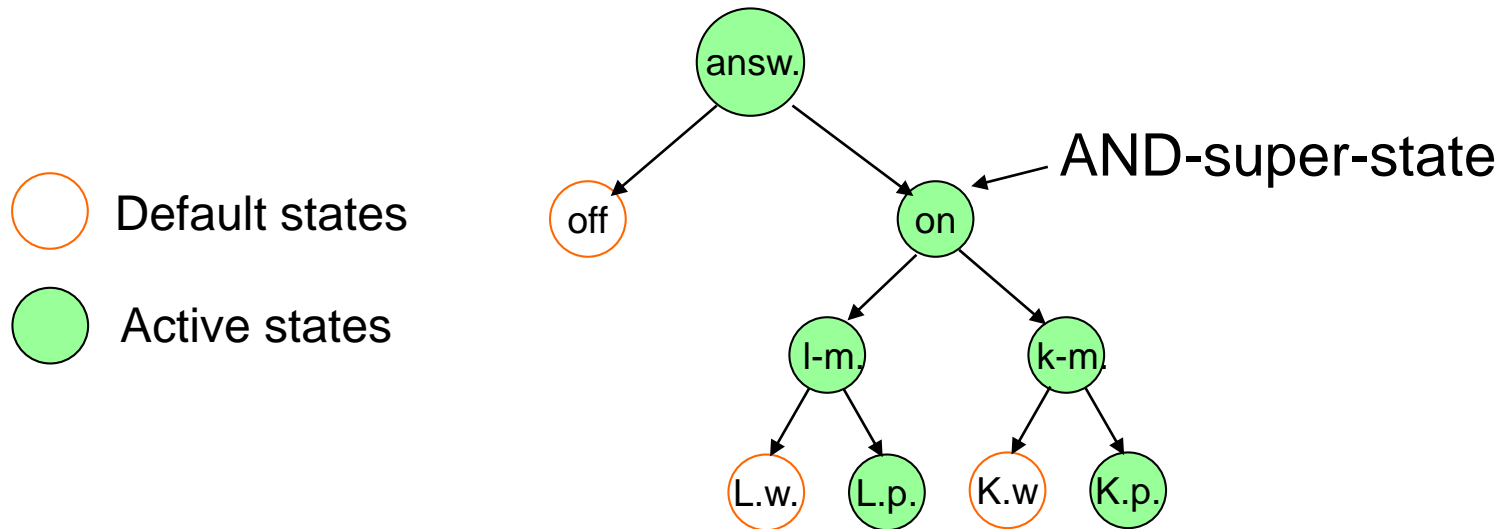
Concurrency

- **AND-super-states:** FSM is in **all** (immediate) sub-states of a AND-super-state; Example:

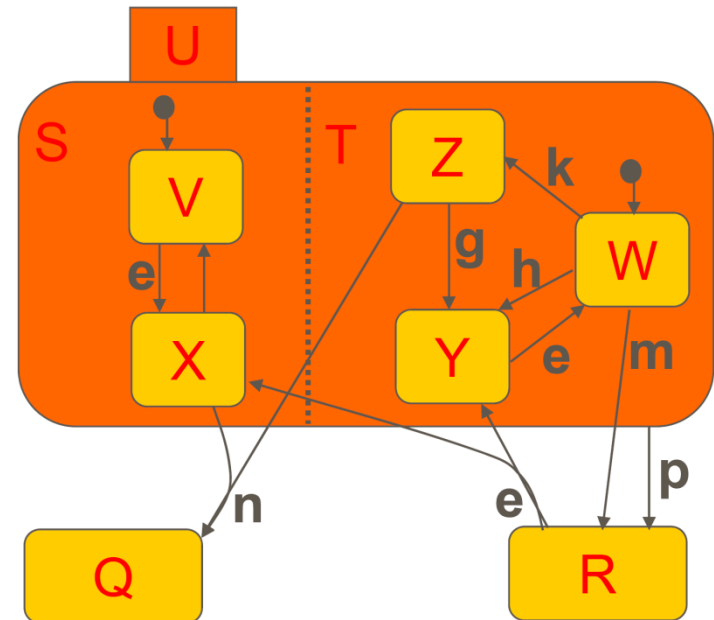
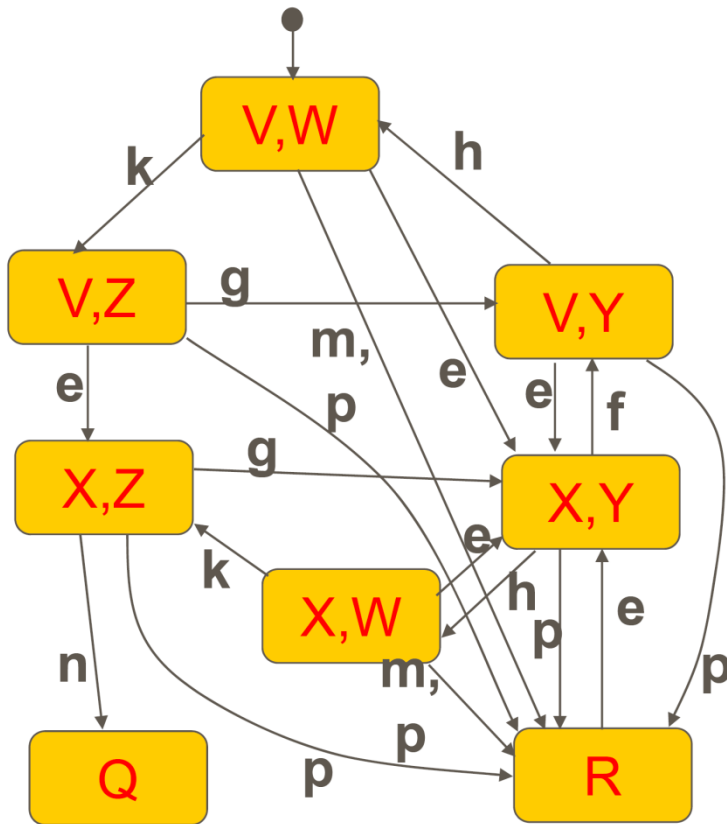


Concurrency

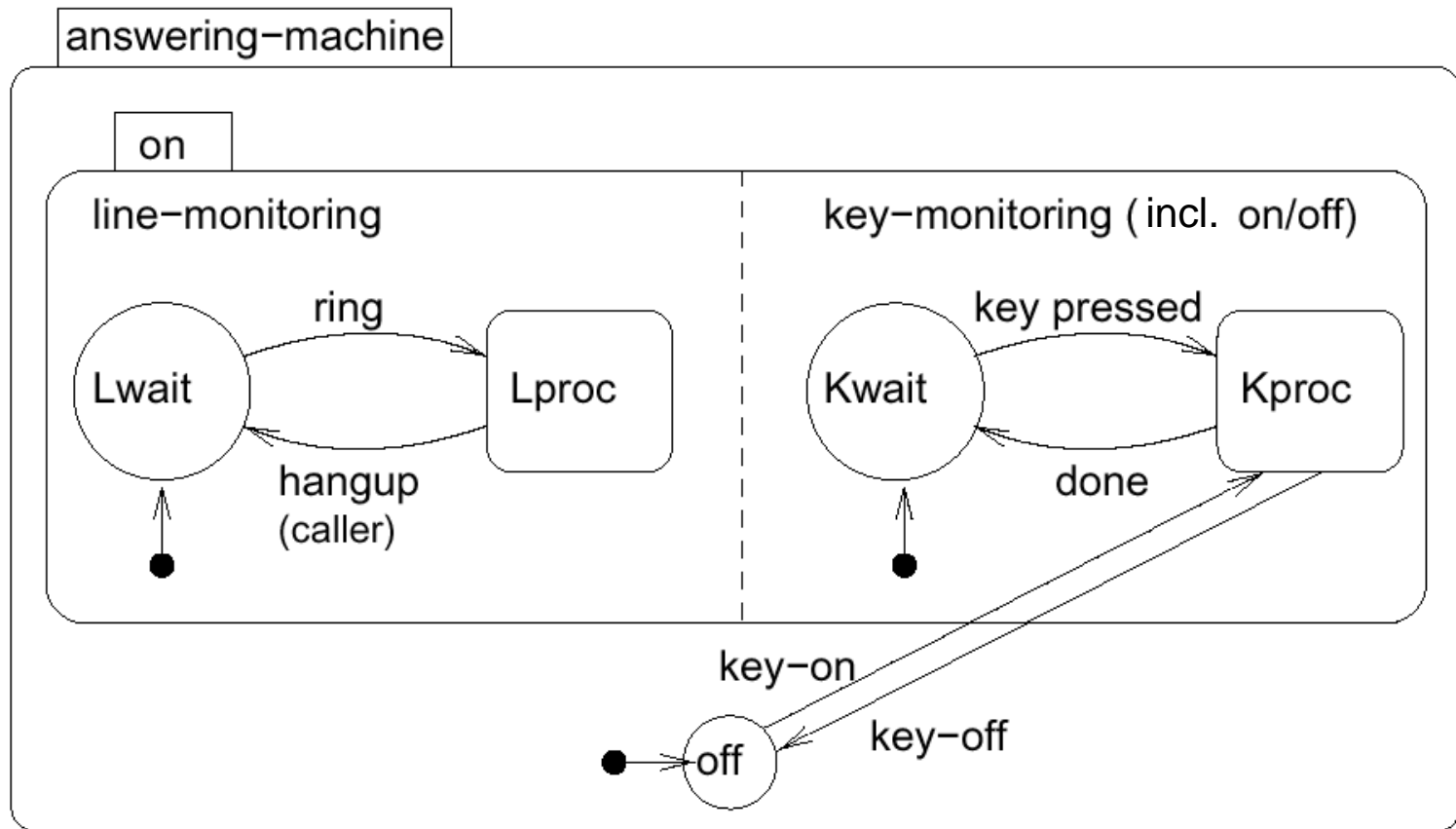
- Example for active states:



Benefits of AND-decomposition



Entering and leaving AND-super-states



- Line-monitoring and key-monitoring are entered and left, when key-on and key-off events occur.

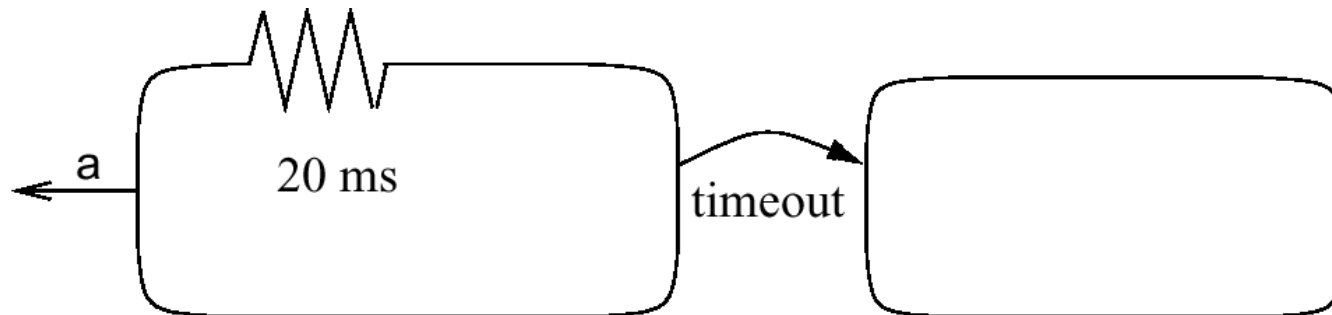
Types of states

In StateCharts, states are either

- **basic states**, or
- **AND-super-states**, or
- **OR-super-states**.

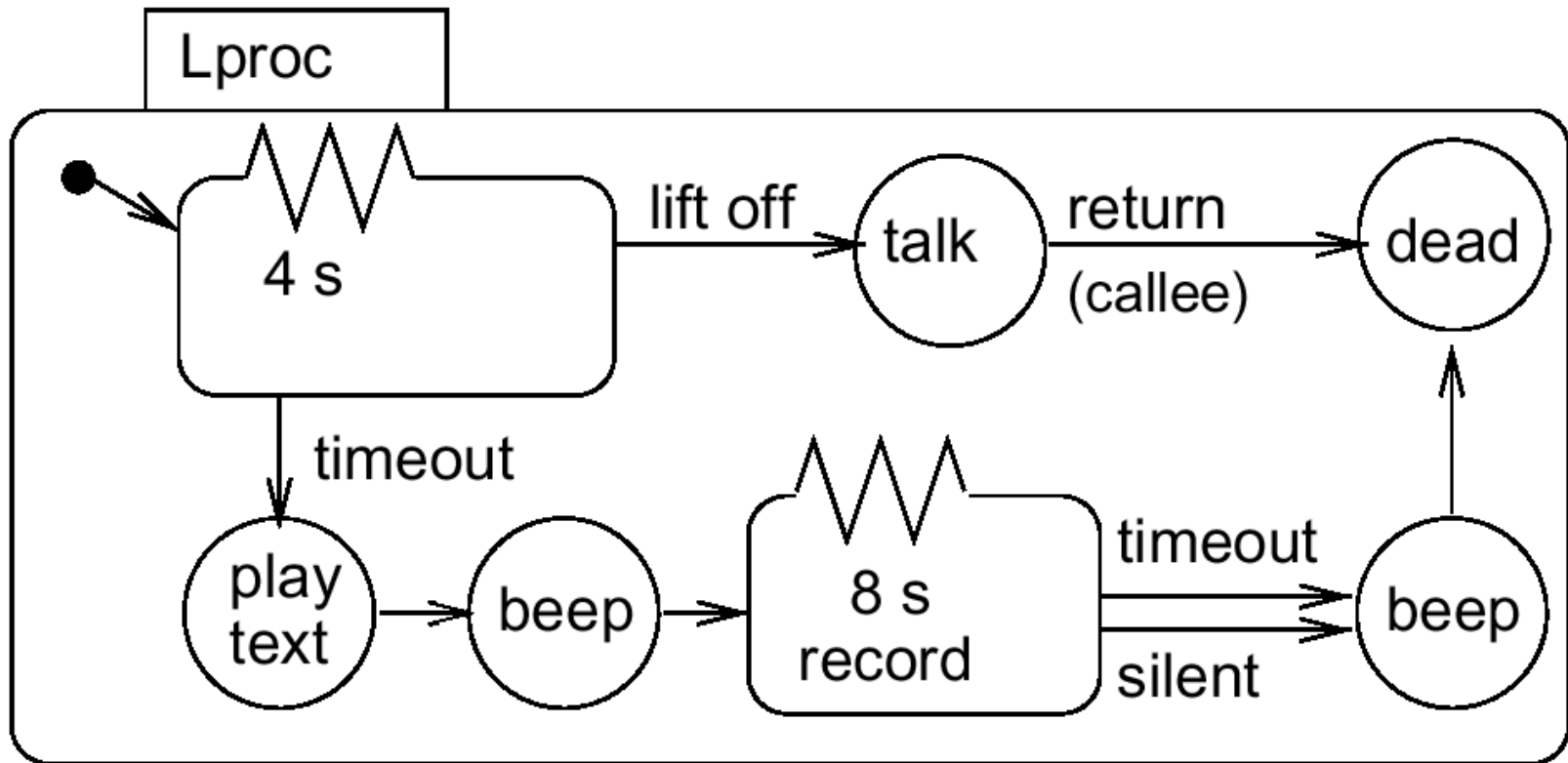
Timers

- In StateCharts, special edges can be used for timeouts.

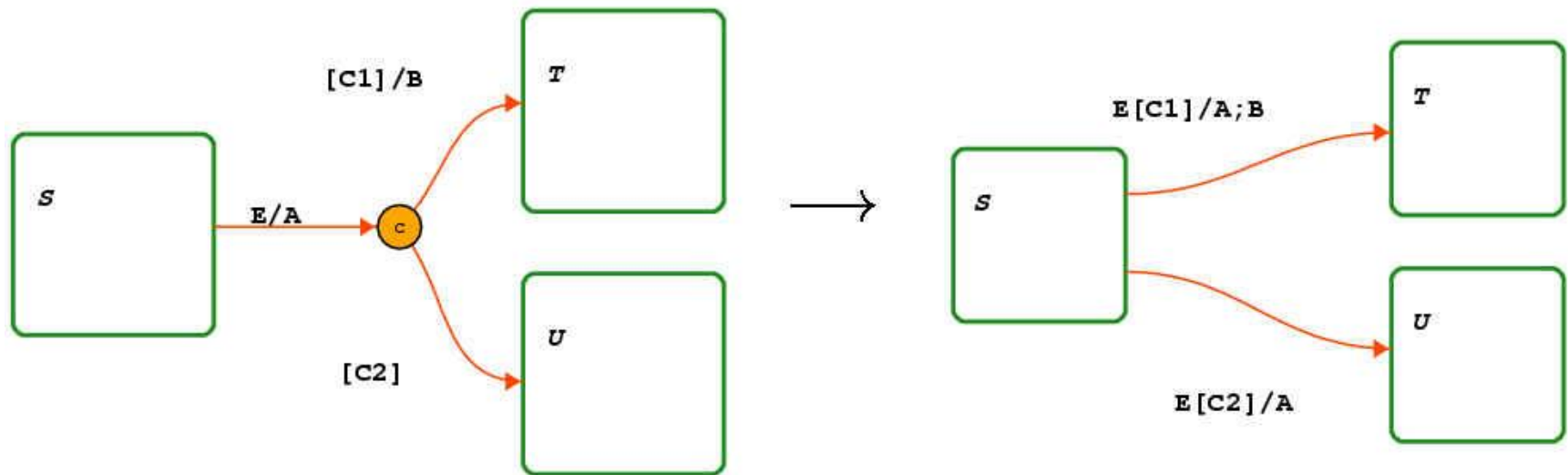


If event a does not happen while the system is in the left state for 20 ms, a timeout will take place.

Using timers in answering machine

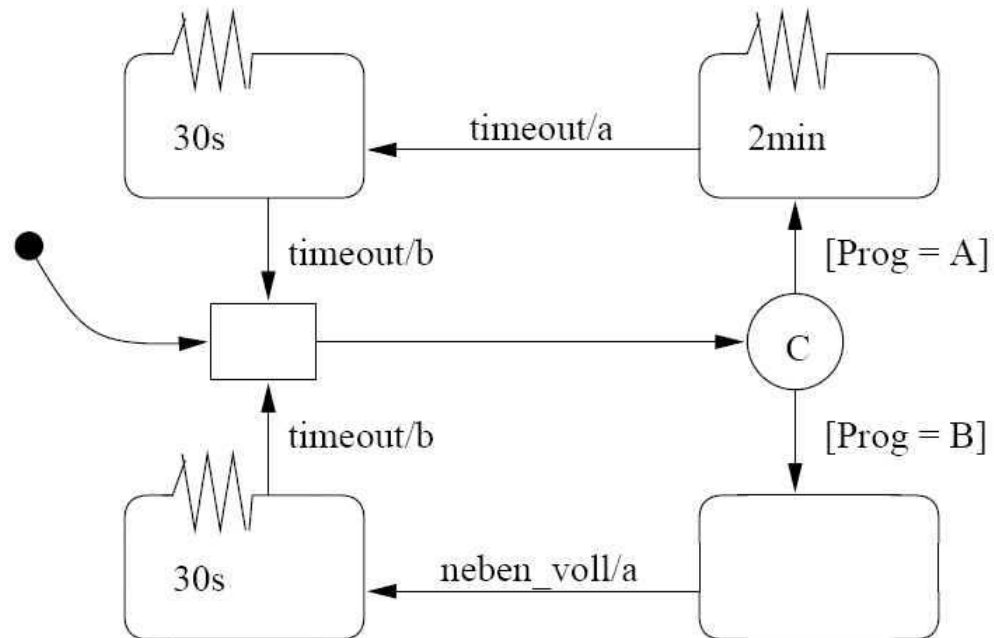


Condition connector

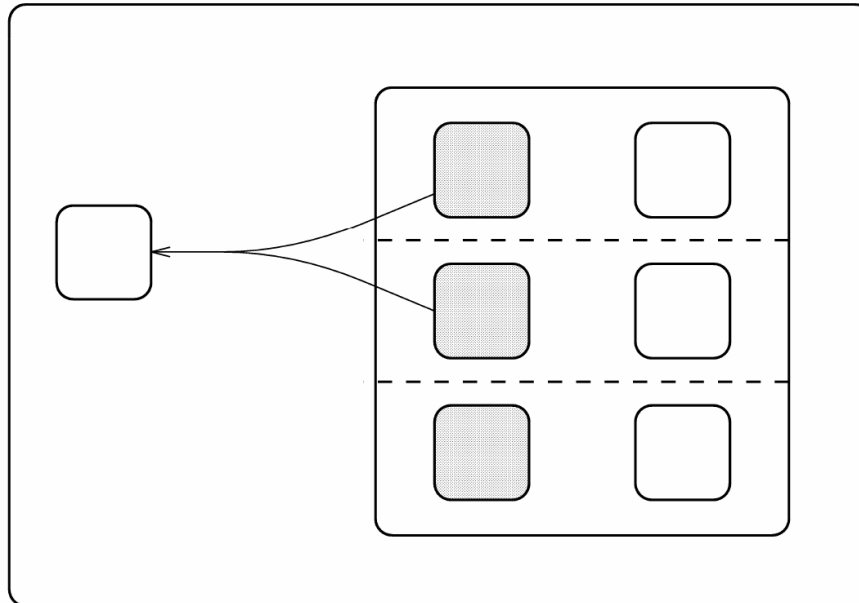
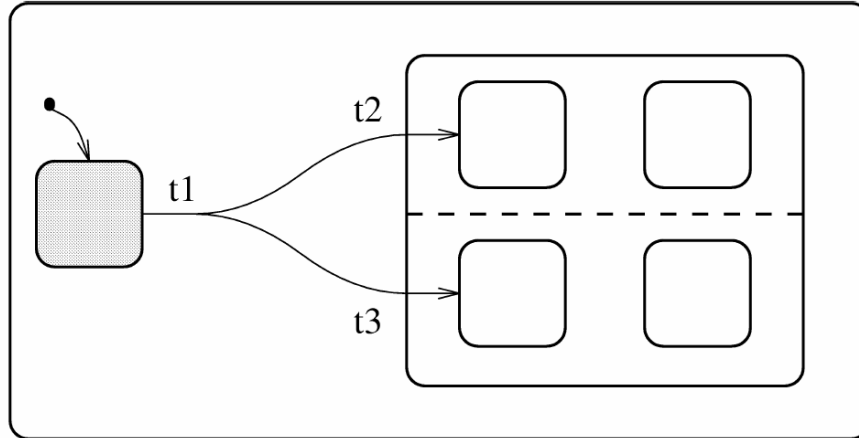


Connectors

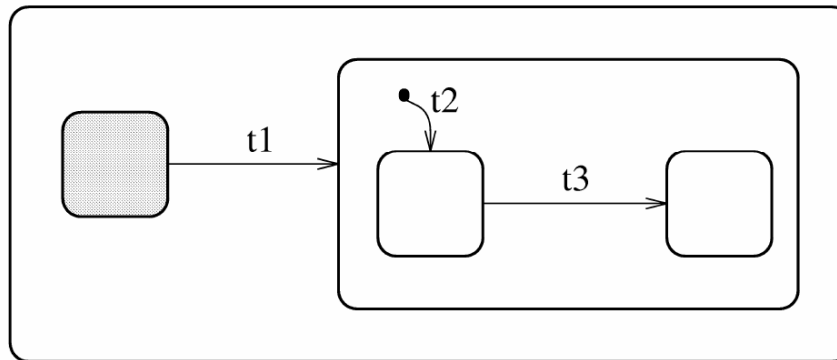
- Example: Traffic light control with two programs



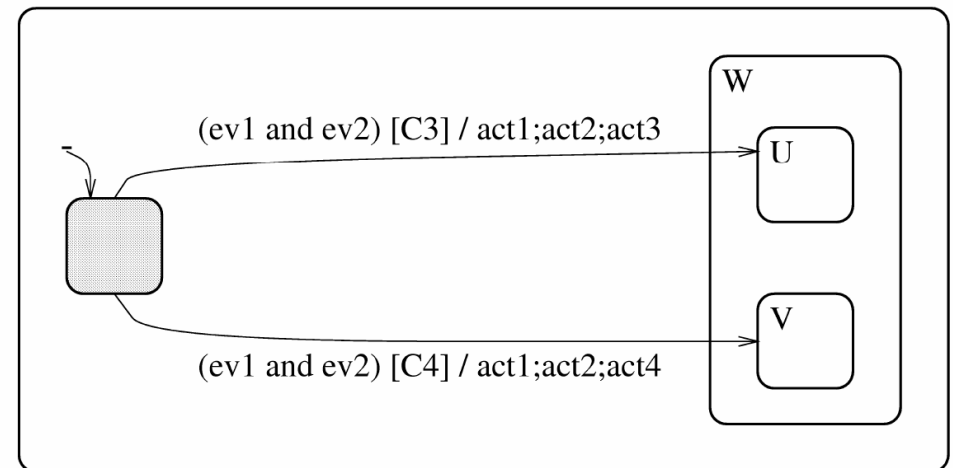
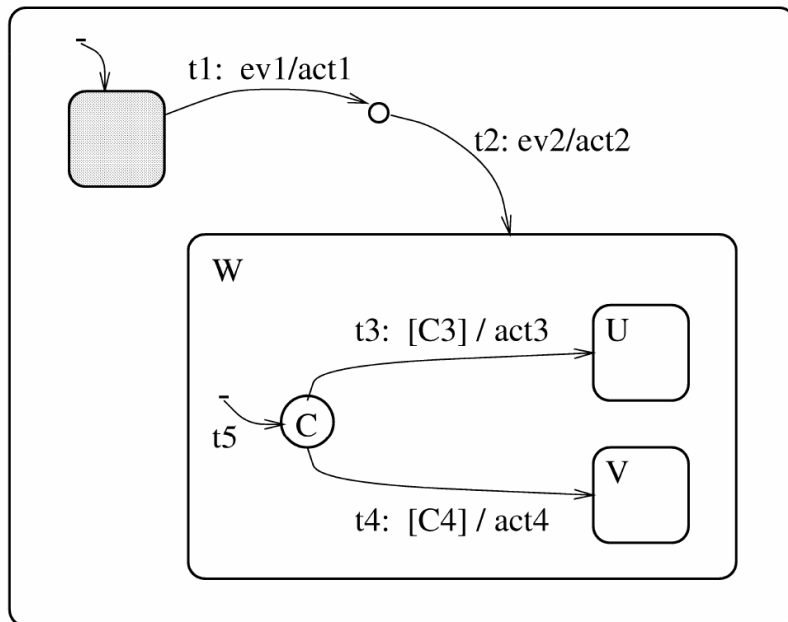
Join and Fork Connectors



Compound transitions

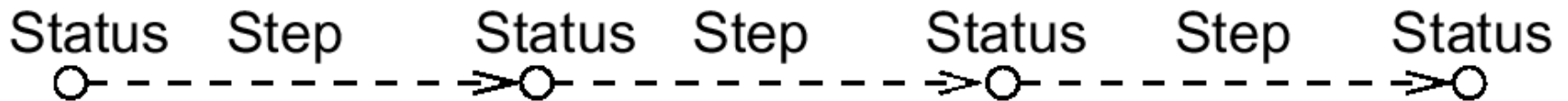


$t1$ and $t2$ must be executed together



Semantics of StateCharts

- Execution of a StateChart model consists of a sequence of **steps**
- A step leads from one **status** to another



- One step:
 - Given:
 - Current system status s_i
 - Current time t
 - External changes Δ
 - Find:
 - New status s_{i+1}

External changes

- External data and external events constitute the interface between system and environment.
- The environment provides external events at certain times and changes external data at certain times.
- External events not yet seen in the previous step and changes of external data not seen in the previous step are called **external changes** for the current step.

Status of the system

The current status of the system is given by

- set of active states
- current values of variables
- the generated events from previous step
- the values of the history connectors
- set of all timeout events $\langle tm(e, d), n \rangle$ in the state chart with „emission times“ n (times n are initially set to 1)
- set of currently scheduled actions $\langle sc(a, d), n \rangle$ with their times n

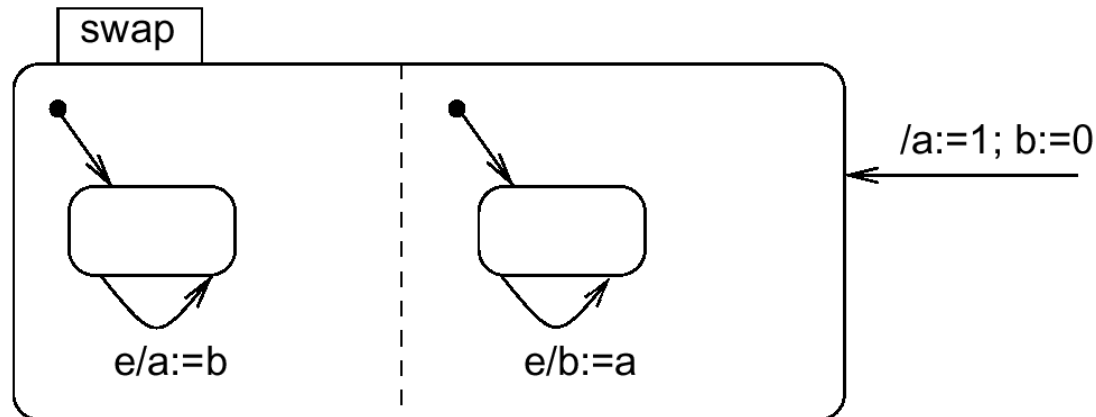
StateMate Semantics

Three phases

1. Effect of external changes on events and conditions is evaluated
2. The set of transitions to be made in the current step and right-hand side of assignments are computed
3. Transitions become effective, variables obtain new values

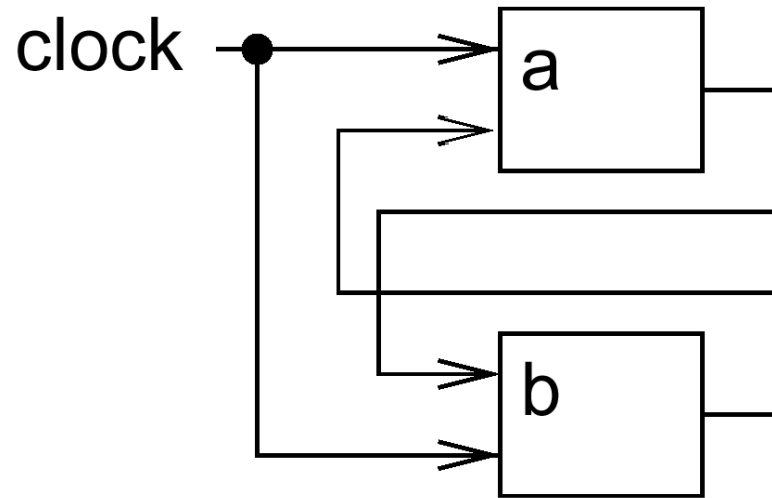
<http://www-03.ibm.com/software/products/en/ratistat>

Example



- In part 2, variables a and b are assigned to temporary variables. In part 3, these are assigned to a and b . As a result, variables a and b are swapped.
- Without this separation, executing the left state first would assign the old value of b ($=0$) to a and b . Executing the right state first would assign the old value of a ($=1$) to a and b . The execution of parallel assignment would be nondeterministic.

Reflects model of clocked hardware



- In an actual clocked (synchronous) hardware system, both registers would be swapped as well.

Same separation into phases found in other languages as well, especially those that are intended to model hardware.

Other semantics

- Several other specification languages for hierarchical state machines (e.g., UML) do not include the three simulation phases
- Corresponds more to a software point of view without synchronous clocks.
- Some simulation tools can be run with optional multi-phased simulation.

Broadcast mechanism

- Values of variables are visible to all parts of the StateChart model.
- New values become effective in part 3 of the execution stage for the current step and are obtained by all parts of the model in the following step.

- ☞ StateCharts implicitly assumes a **broadcast** mechanism for variables.
- ☞ StateCharts is appropriate for local control systems (☺), but not for distributed applications for which updating variables might take some time (☹).

Time models

- External events and external changes of variables are associated with physical times.
- But how does time proceed internally?
- How many steps are performed before external changes are evaluated?

The synchronous time model

- A single step every time unit.
- If the current step is executed at time t , then the next step is executed at time $t+1$.
- Events and variable changes are communicated between different states during one time unit.
- External changes are only accumulated during one time unit.

Models of time in statecharts

- Synchronous time model
 - Assume single step executed every time unit
 - Reacts to external changes occurred since end of previous step
- Asynchronous time model
 - Reacts when external changes occur
 - Allows several changes to occur simultaneously
 - Allows several steps at once – superstep
- In both models, execution of step takes zero time

The super-step time model (1)

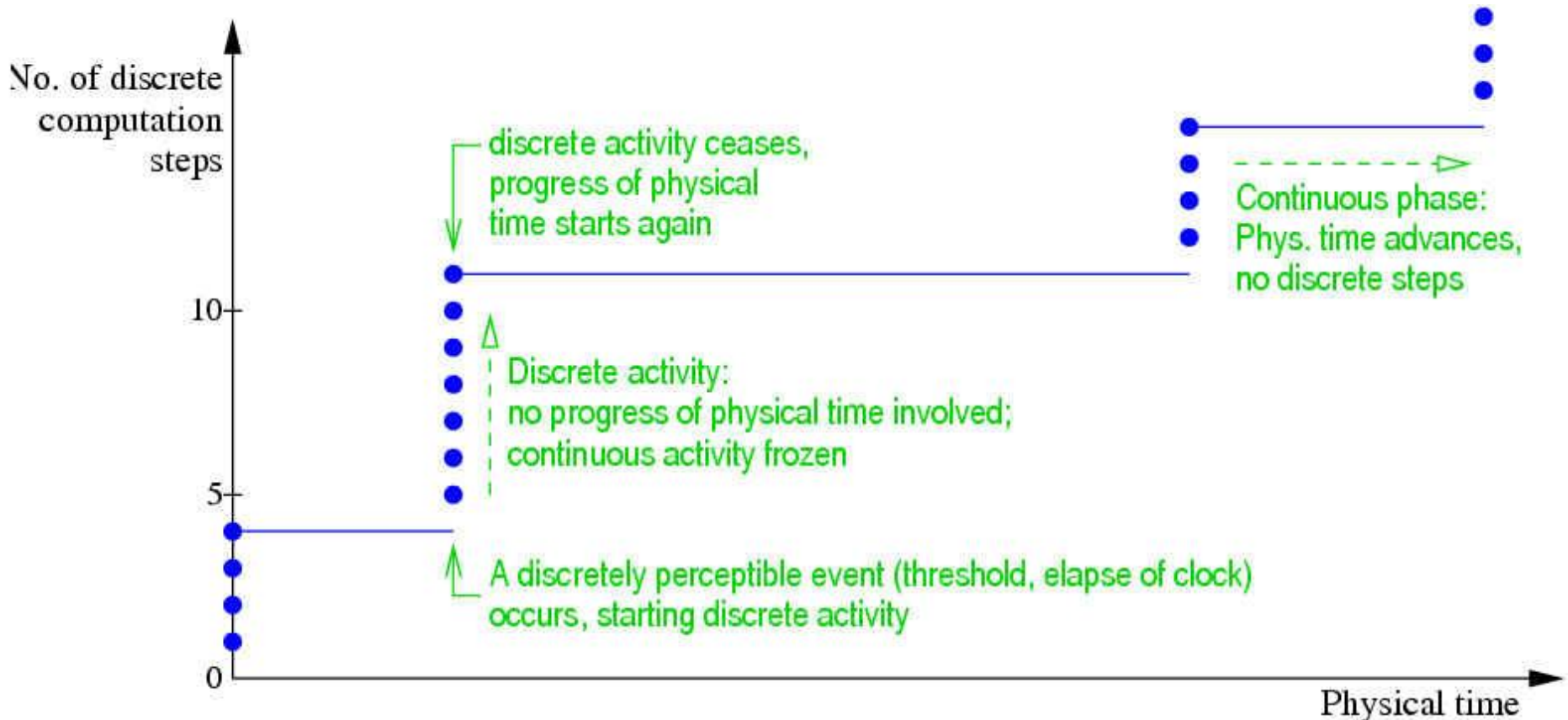
- A step of the statechart does not need time.
- Super-steps are performed:
 - A super-step is a sequence of steps.
 - A super-step terminates when the status of the system is stable.
 - During a super-step the time does not proceed and thus external changes are not considered.
- After a super-step, physical time restarts running, i.e. activity of the environment will be possible again.
- The computation of the statechart is resumed when
 - external changes enable transitions in the statechart
 - Timeout events enable transitions of the statechart

The super-step time model (2)

- In STATEMATE, GO-REPEAT command executes superstep
- Semantics for GO-REPEAT
 - Execute external changes since end of last step
 - Execute timeout and scheduled actions that are due
 - Execute basic step algorithm until system in stable state (no generated events, no enabled compound transitions, static reactions)
- GO-REPEAT may result in infinite loop

The super-step time model (3)

- Two-dimensional time:



- Assumption: Computation time is negligible compared to dynamics of the environment.

The super-step time model (4)

- During one super-step the number of communications between different states is not restricted. All communications are assumed to be performed in zero time.
- Simplified model of reality.
- Can only be realistic, if
 - Discrete computations are fast compared to dynamics of the environment.
 - Discrete computations will be stable after a restricted number of steps.
- Timeout events can reactivate a statechart
 - ⇒ Possible to specify statecharts which permit progress of physical time after a limited number of steps and reactivate themselves via timeout events

Evaluation of StateCharts (1)

Pros:

- Hierarchy allows arbitrary nesting of AND- and OR-superstates.
- Formal semantics (defined in a follow-up paper to original paper).
- Large number of commercial simulation tools available (StateMate, StateFlow, BetterState, ...)
- Available „back-ends“ translate StateCharts into C or VHDL, thus enabling software or hardware implementations.

Evaluation of StateCharts (2)

Cons:

- Generated C programs frequently inefficient,
- Not useful for distributed applications,
- No program constructs,
- No description of non-functional behavior,
- No object-orientation,
- No description of structural hierarchy.

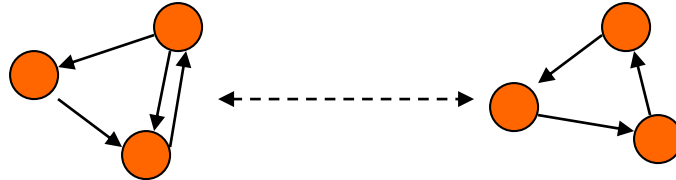
Some general properties of languages

1. Synchronous vs. asynchronous languages

- Description of several (concurrent) processes in many languages non-deterministic:
The order in which executable tasks are executed is not specified (may affect result).
- Synchronous languages: based on automata models. They describe concurrently operating automata. When automata are composed in parallel, a transition of the product is made of the "simultaneous" transitions of all of them.
- Synchronous languages implicitly assume the presence of a (global) clock. Each clock tick, all inputs are considered, new outputs and states are calculated and then the transitions are made.

Some general properties of languages


1. Synchronous vs. asynchronous languages



- This requires a broadcast mechanism for all parts of the model.
- Idealistic view of concurrency.
- Has the advantage of guaranteeing deterministic behavior.
- Statechart steps work synchronously.
 - Broadcast of events and variable changes during each step.
 - StateCharts are deterministic, if priority rules are introduced for transitions enabled at the same time.

Some general properties of languages

2. Properties of processes

- **Number of processes**
static (suitable for hardware);
dynamic (dynamically changed hardware architecture)
- **Nested declaration of processes**
or all declared at the same level
-  **StateCharts comprises a static number of processes and nested declaration of processes.**

Some general properties of languages

3. Communication paradigms

▪ **Message passing**

- **Asynchronous message passing = non-blocking communication**

Sender does not have to wait until message has arrived; potential problem: buffer overflow

- **Synchronous message passing = blocking communication, *rendez-vous*-based communication**

Sender will wait until receiver is ready for receiving message (“point of communication”)

- **Extended *rendez-vous***

Explicit acknowledge from receiver required. Receiver can do checking before sending acknowledgement.


Some general properties of languages

3. Communication paradigms

- **Shared memory**

Variables accessible to several tasks

- Problem: Concurrent write.
- Critical sections = sections at which exclusive access to some resource r must be guaranteed.

 **StateCharts uses shared memory for communication between processes.**

Some general properties of languages

4. Specifying timing

4 types of timing specs required [Burns, 1990]:

- **Measure elapsed time**

Check, how much time has elapsed since last call


- **Means for delaying processes**

- **Possibility to specify timeouts**

We would like to be in a certain state only a certain maximum amount of time.

- **Methods for specifying deadlines**

With current languages not available or specified in separate control file.

 **StateCharts comprises a mechanism for specifying timeouts. Other types of timing specs are not supported.**

Synchronous Composition

Lee/Seshia
Section 6.2

- Important semantic model for concurrent composition
- Here: composition of actors
- Foundation of Statecharts, Simulink, synchronous programming languages
 - Esterel
 - Lustre
 - Scade
- Idealistic view of concurrency, not adequate for distributed systems (Implicit assumption: presence of global clock and instant communication; requires broadcast mechanism)

Goal: deterministic behavior

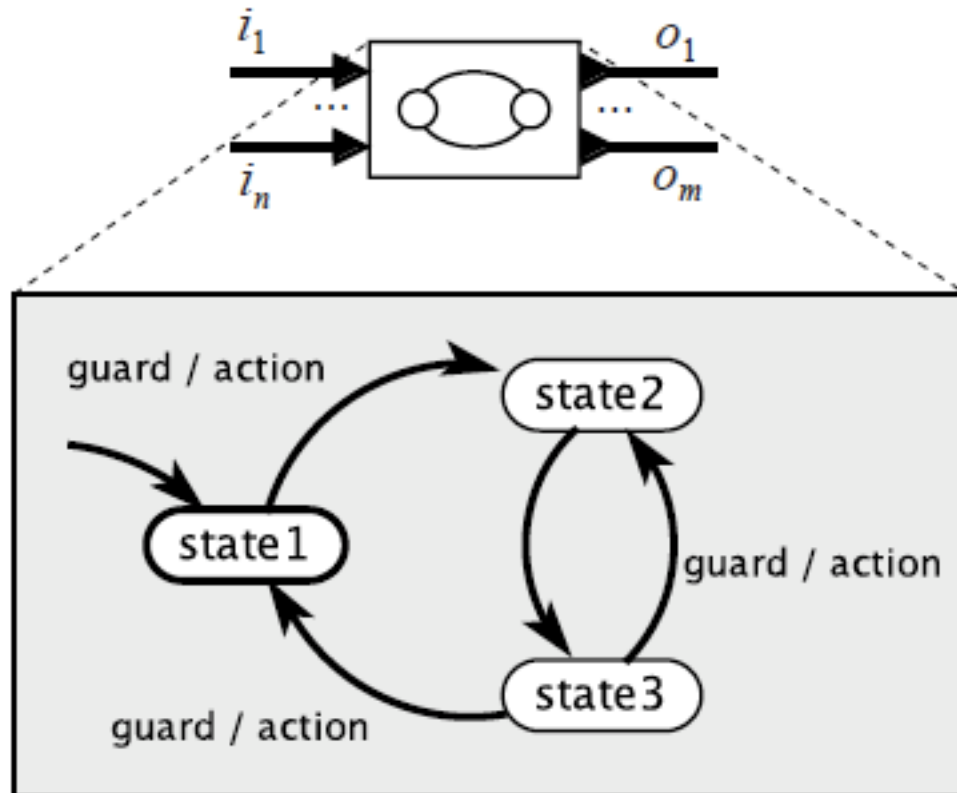
An important advantage of synchronous over asynchronous composition is that determinacy can be preserved.

In the following, we'll assume that the individual actors are deterministic, and ensure that the composition remains deterministic.

For example, StateCharts are deterministic, if priority rules are introduced for transitions enabled at the same time (see, for example, the Stateflow semantics.)

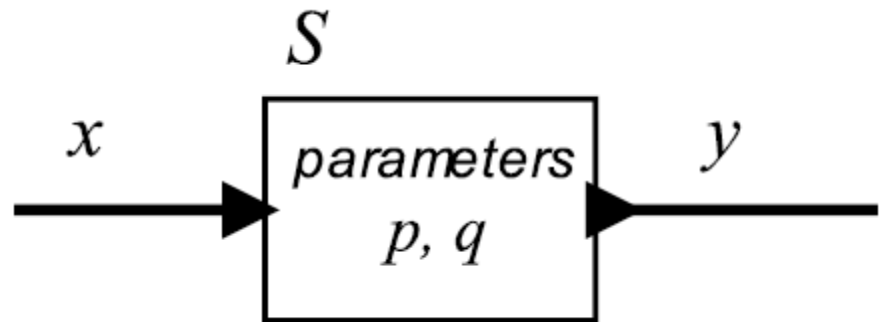
REVIEW: Actor Model for State Machines

Expose inputs and outputs, enabling composition:



REVIEW: Actor Model of Continuous-Time Systems

- A *system* is a function that accepts an input *signal* and yields an output signal.
- The domain and range of the system function are sets of signals, which themselves are functions.
- Parameters may affect the definition of the function S .

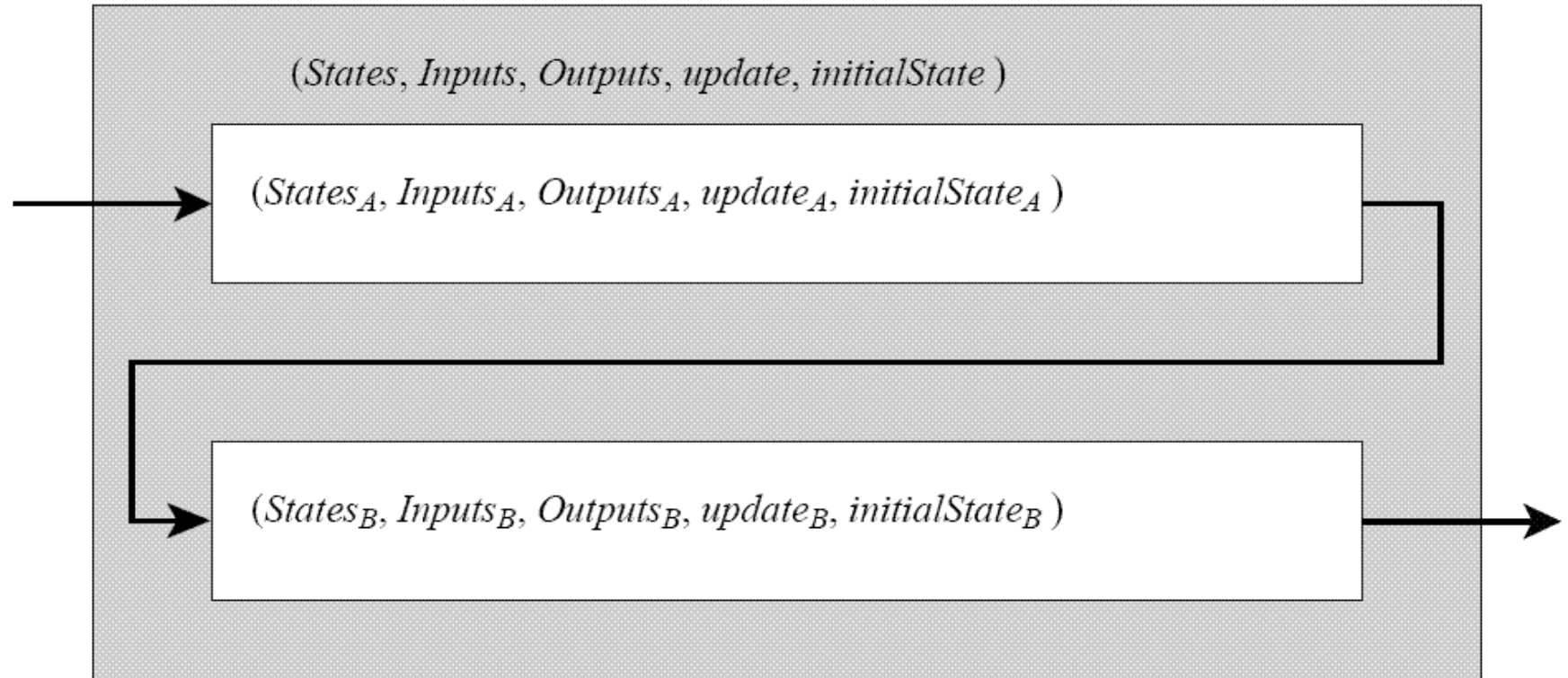


$$x: \mathbb{R} \rightarrow \mathbb{R}, \quad y: \mathbb{R} \rightarrow \mathbb{R}$$

$$S: X \rightarrow Y$$

$$X = Y = (\mathbb{R} \rightarrow \mathbb{R})$$

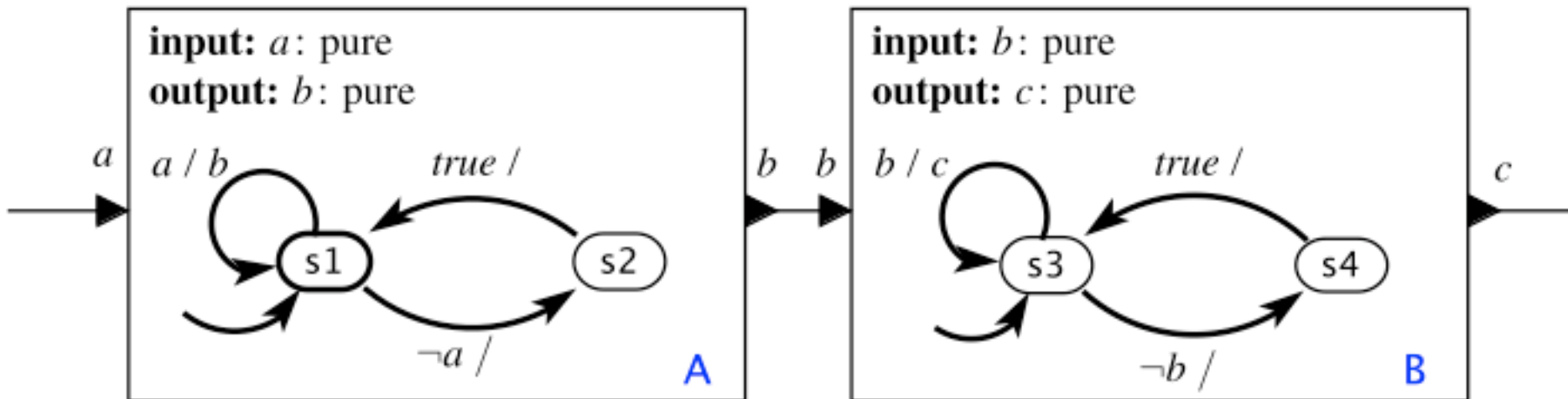
Synchronous composition



Synchronous composition: the machines react simultaneously and instantaneously, despite the apparent causal relationship!

Synchronous composition:

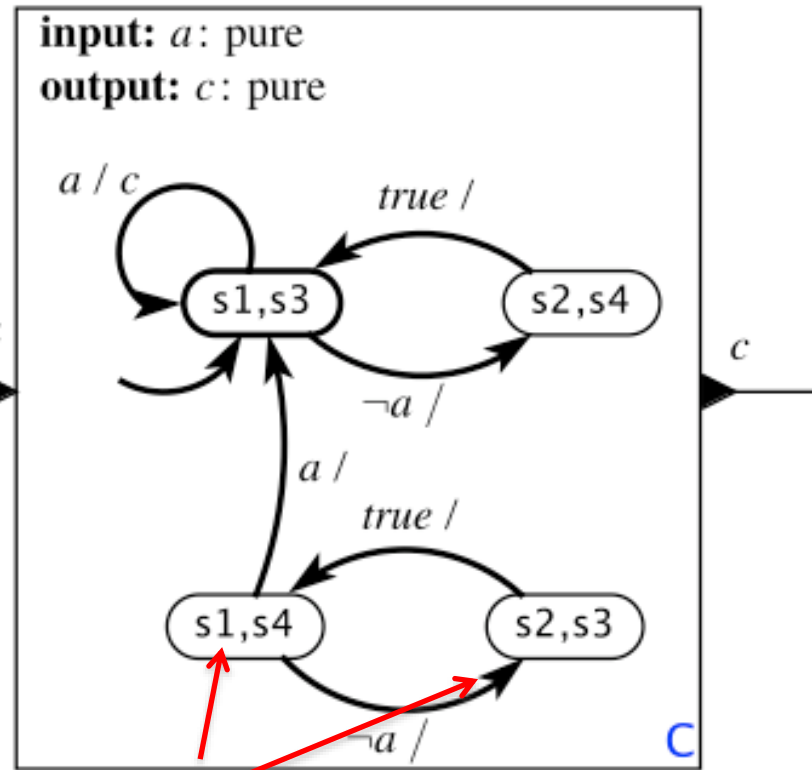
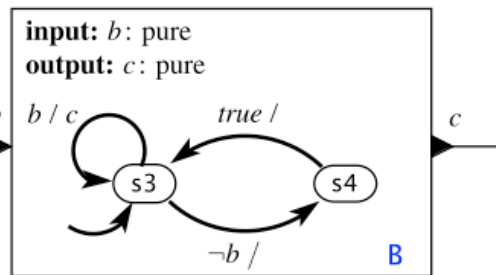
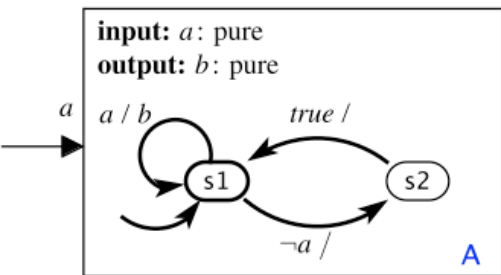
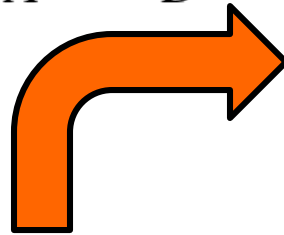
Reactions are *simultaneous* and *instantaneous*



Synchronous composition:

Reactions are *simultaneous* and *instantaneous*

$$S_C = S_A \times S_B$$



unreachable