HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL
FHO Fachhochschule Ostschweiz

Embedded Software Engineering
Zustandsbasierte Systeme

Prof. Reto Bonderer
HSR Hochschule für Technik Rapperswil
reto.bonderer@hsr.ch

Oktober 2019

Themen für heute

- Endliche Automaten
- Statecharts

Strategie bei der Modellierung

- UML beinhaltet viele verschiedene Diagramme, die untereinander nicht ohne Redundanz sind
- Sie müssen sich überlegen, mit welchen Diagrammen Sie arbeiten möchten
- Jedes einzelne Diagramm kann mit unzähligen Details ausgestattet werden (die Dialogboxen in einem CASE-Tool zeugen davon)
- Spezifizieren Sie nur die Details, die Ihnen etwas bringen. Im Normalfall ist weniger mehr.

ZUSTANDSBASIERTE SYSTEME

Motivation

- Viele Systeme, insbesondere Embedded Systems, können zustandsbasiert implementiert werden
- Die Theorie der *Endlichen Automaten* (*Finite State Machines*, *FSM*) ist in der Elektronik längstens etabliert
- Endliche Automaten können auf unterschiedliche Arten beschrieben werden (graphisch, tabellarisch, HDL)
- Endliche Automaten können auf viele Arten implementiert werden (mit Flip Flops und Logik, als (Full Custom) ASIC, auf einem FPGA, auf einem Controller in unterschiedlichen Programmiersprachen)
- Zur Ausführung auf einem Controller ist kein Betriebssystem notwendig

Asynchrone vs. synchrone FSMs

- Bei (elektronisch implementierten) asynchronen FSMs führen geänderte Inputsignale direkt zu Zustandsänderungen. Sie sind deshalb "schneller", jedoch äusserst empfindlich auf Glitches und kaum testbar.
- Bei synchronen FSMs werden die Inputsignale nur zu diskreten Zeitpunkten betrachtet, diese Systeme sind getaktet.
- Softwareimplementationen sind eigentlich immer synchron, da die Rechner getaktet sind. Wenn die Inputsignale mittels Interrupts behandelt werden, ist darauf zu achten, dass die Interrupts nicht fälschlicherweise gesetzt werden (ist Aufgabe der Elektronik)
- Rein softwareseitig besteht die Problematik der Asynchronizität nicht

Finite State Machines (FSM)

- Eine FSM befindet sich immer in einem definierten Zustand
- Die Inputs X bezeichnen üblicherweise Ereignisse (*Events*)
- Die Outputs Y werden oft auch *Actions* genannt
- Eine FSM benötigt immer Speicherelemente zur Speicherung des internen Zustands
- In der Praxis sind zwei FSM-Varianten bekannt:
  - der Mealy-Automat
  - der Moore-Automat
  - (Medvedev-Automat)

Zustands-Ereignis-Diagramm (State-Event-Diagram)

- Die Zustände werden mit einem Kreis gekennzeichnet
- Die Ereignisse werden mit Pfeilen zwischen den Zuständen bezeichnet (Transitionen)
- Die Aktionen werden entweder bei den Zuständen geschrieben oder bei den Transitionen (je nach Automatentyp)
- Die Ausführung einer Transition benötigt keine Zeit. Deshalb sind bei der Modellierung oft Zwischenzustände vorzusehen, z.B. "Closing", "Starting up", "Booting", etc.

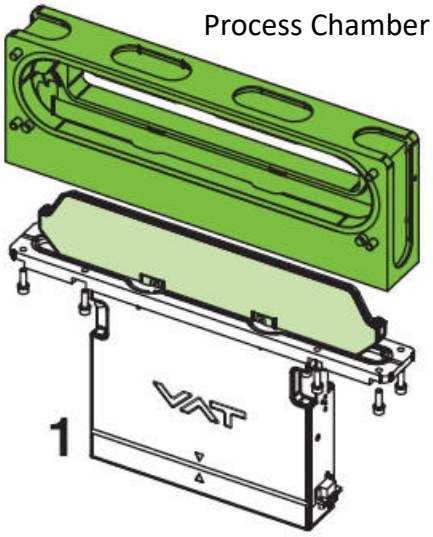
Mealy-Automat

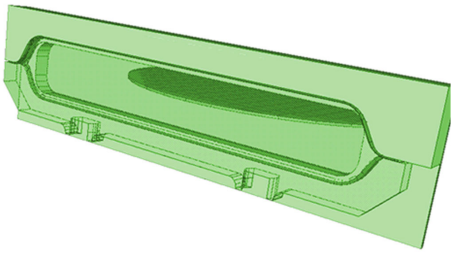
- Der nächste Zustand Z_{n+1} ist abhängig vom Input X und von Z_n:
  $Z_{n+1} = f(Z_n, X)$
- Der Output Y ist abhängig vom inneren Zustand Z_n und vom Input X:
  $Y = g(Z_n, X)$
- Die Actions liegen deshalb bei den Transitionen

Moore-Automat

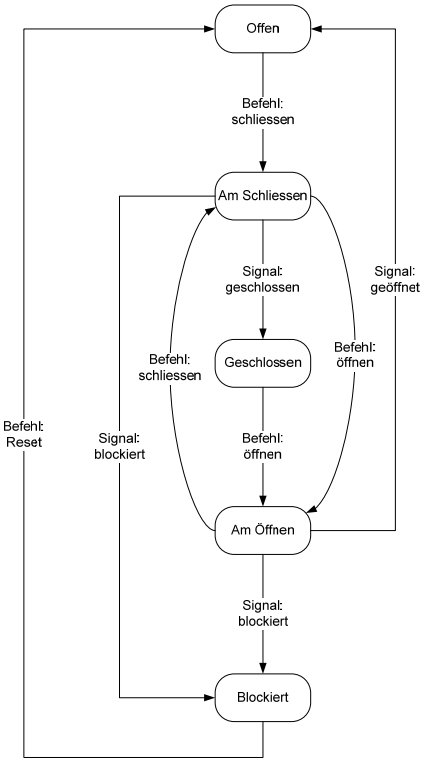
- Der nächste Zustand Z_{n+1} ist abhängig vom Input X und von Z_n:
  $Z_{n+1} = f(Z_n, X)$
- Der Output Y ist nur abhängig vom inneren Zustand Z_n:
  $Y = g(Z_n)$
- Die Actions liegen deshalb bei den Zuständen

Hochvakuumventil zum Einschleusen von Si-Wafern (Loadlock)





HV-Ventil als Moore-Automat



Ventil grün

Ventil weiss blinkend
Ventil schliessen

Ventil weiss

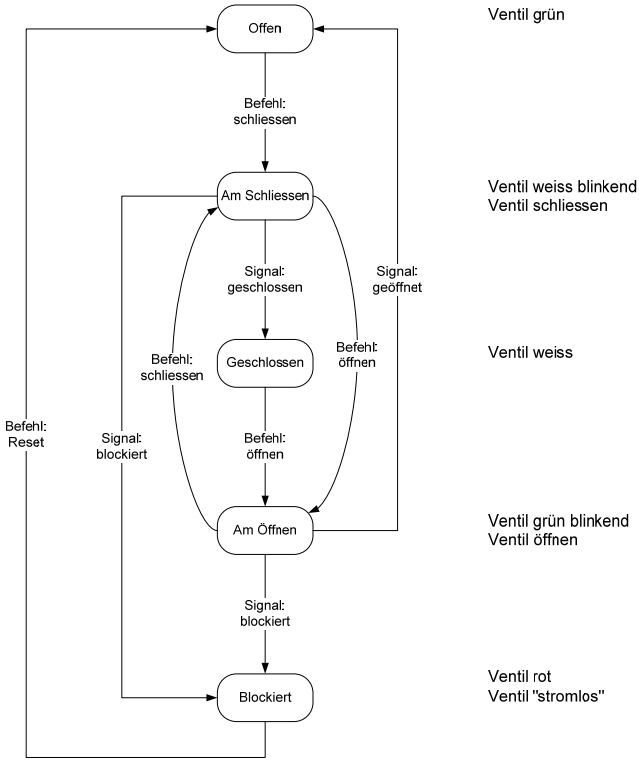
Ventil grün blinkend
Ventil öffnen

Ventil rot
Ventil "stromlos"

HV-Ventil als Moore-Automat

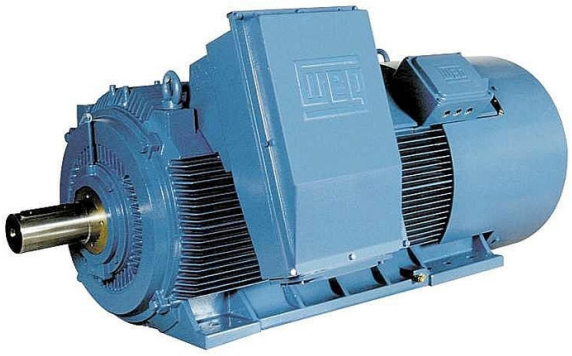
- States:
  - Offen
  - Am Schliessen
  - Geschlossen
  - Am Öffnen
  - Blockiert
- Befehle:
  - Schliessen
  - Öffnen
  - Reset
- Signale
  - Geöffnet
  - Geschlossen
  - Blockiert

HV-Ventil: Implementation mit Testprogramm

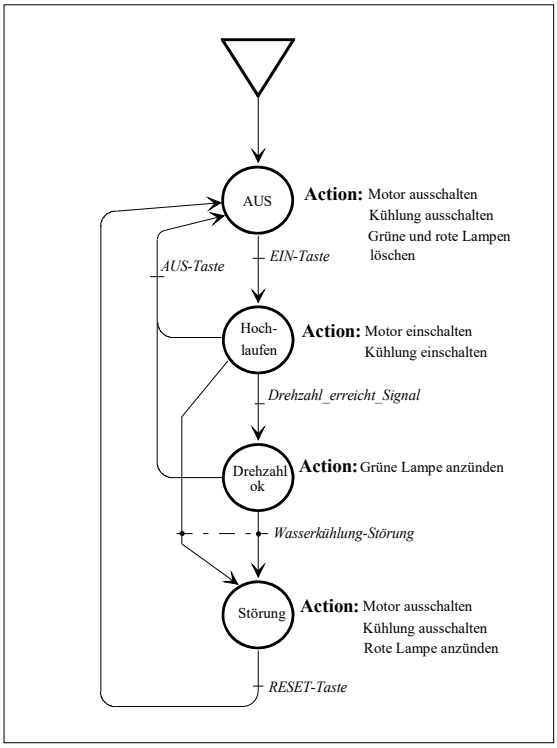


Beispiel: Elektromotor

Ein Elektromotor wird durch Betätigung eines Startknopfes eingeschaltet. Gleichzeitig muss die für den Motor benötigte Wasserkühlung eingeschaltet werden. Sobald die Soll-Drehzahl erreicht ist, wird eine grüne Lampe eingeschaltet (andernfalls immer ausgeschaltet). Weiters existiert ein AUS-Knopf, um den Motor zu jedem Zeitpunkt auszuschalten. Fällt das Kühlsystem aus oder sinkt die Ist-Drehzahl unter den Sollwert, so wird der Motor aus Sicherheitsgründen ausgeschaltet, und eine Lampe leuchtet rot. Um den Motor aus dem Fehlerzustand herauszubringen dient ein Reset-Knopf.



State-Event Diagramm für den Elektromotor



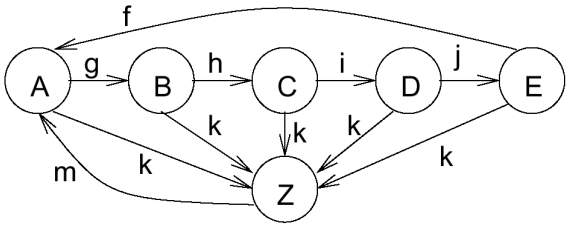
Darstellung als Zustandstabelle

Momentaner Zustand	Ereignis	Nächster Zustand	Aktionen
AUS	EIN-Taste	Hochlaufen	Motor ausschalten Kühlung ausschalten Grüne Lampe aus Rote Lampe aus
Hochlaufen	Drehzahl_erreicht Signal	Drehzahl_ok	Motor einschalten Kühlung einschalten
	Aus-Taste	AUS	
	Wasserkühlung Störung	Störung	
Drehzahl_ok	Wasserkühlung Störung	Störung	Grüne Lampe anzeigen
	AUS-Taste	AUS	
Störung	RESET-Taste	AUS	Motor ausschalten Kühlung ausschalten Rote Lampe anzeigen

(Folien teilweise von Prof. Marwedel)

STATECHARTS VON HAREL [1987]

Nachteile von Zustandsdiagrammen

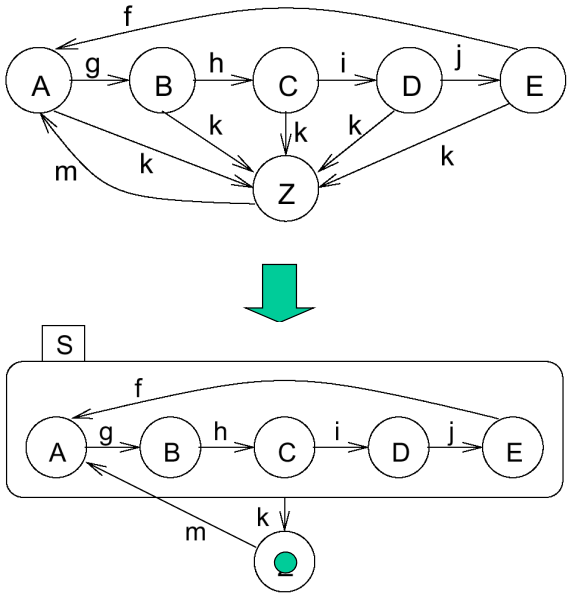


- Die Transition k führt von jedem Zustand in den Zustand Z. Das ist typischerweise ein Fehlerevent, Reset, etc. Das Diagramm wird dadurch unnötigerweise kompliziert.
- Da Zustandsdiagramme flach sind (es gibt keine Hierarchie), werden sie bei praktisch relevanten Aufgaben schnell unübersichtlich
- In Zustandsdiagrammen kann keine zeitliche Parallelität modelliert werden

Statecharts

Zustandsdiagramme +
Hierarchie +
Nebenläufige Prozesse +
Broadcastkommunikation

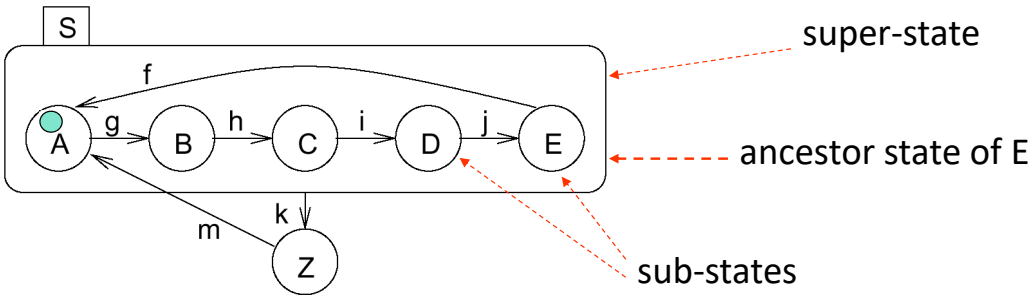
Introducing hierarchy



FSM will be **in** exactly one of the substates of S if S is **active** (either in A or in B or ..)

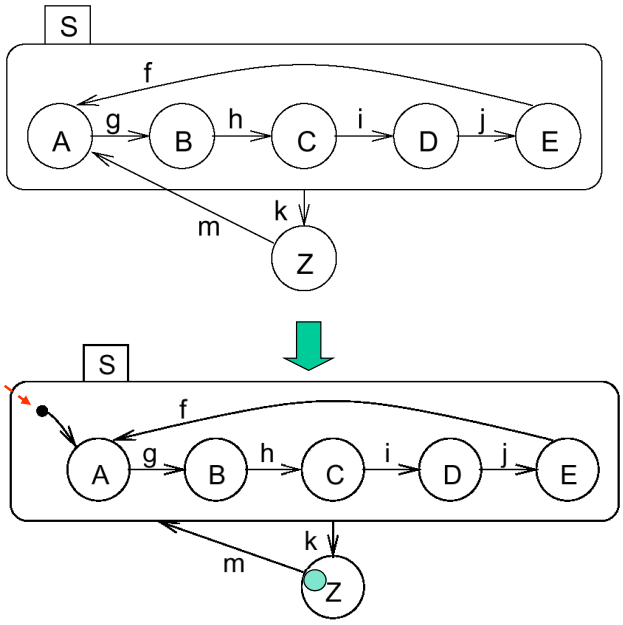
Definitions

- Current states of FSMs are also called **active** states.
- States which are not composed of other states are called **basic states.**
- States containing other states are called **super-states**.
- For each basic state s, the super-states containing s are called **ancestor states**.
- Super-states S are called **OR-super-states**, if exactly one of the sub-states of S is active whenever S is active.

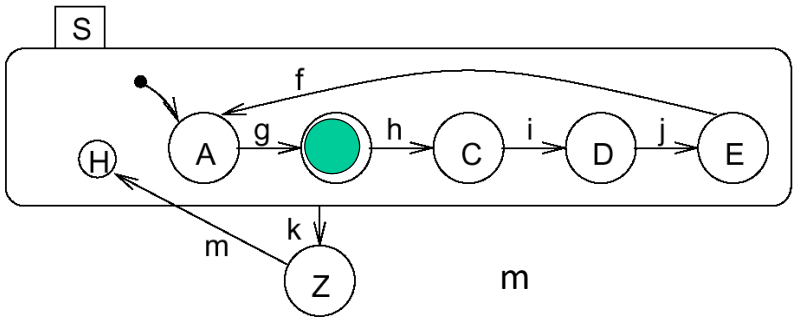


Default state mechanism

- Try to hide internal structure from outside world!
- ☞ Default state
- Filled circle indicates sub-state entered whenever super-state is entered.
- Not a state by itself!



History mechanism



m

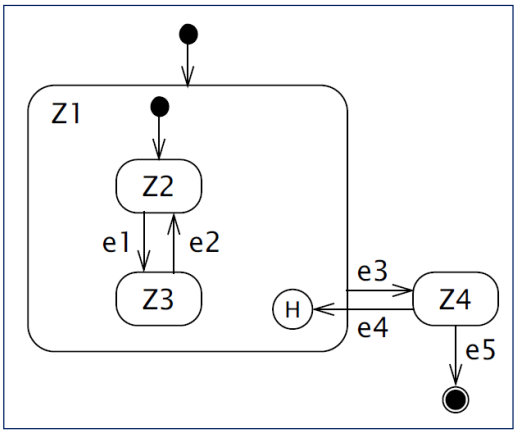
(behavior different from last slide)

- For input m, S enters the state it was in before S was left (can be A, B, C, D, or E). If S is entered for the very first time, the default mechanism applies.
- History and default mechanisms can be used hierarchically.

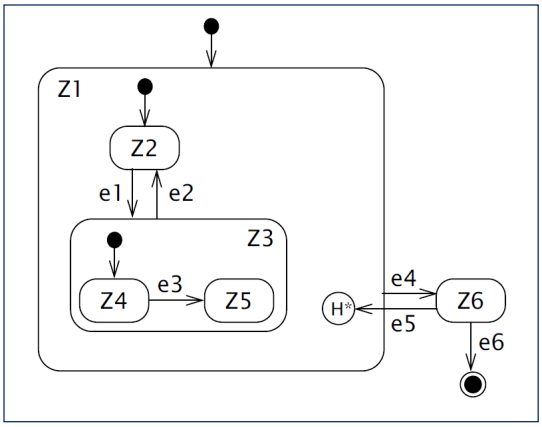
History-Mechanismus

- Der History-Mechanismus merkt sich frühere Zustände
- In diesem Statechart ist die Shallow History gezeigt (shallow = flach), sie wird mit einem **H** bezeichnet

- Beispiel
  - Im Zustand Z3 bewirkt der Event e3 einen Übergang zu Z4
  - Der Event e4 führt nun zu einem Übergang zum früheren Zustand Z3



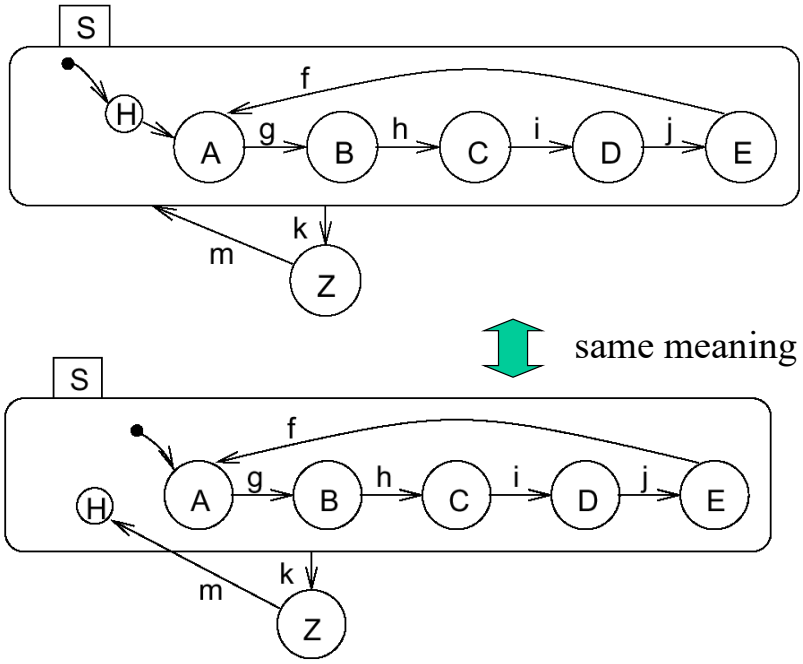
Deep History



- Der Deep History-Mechanismus merkt sich frühere Zustände bis in die unterste Hierarchie
- Im Statechart links ist die Deep History gezeigt, sie wird mit einem **H*** bezeichnet

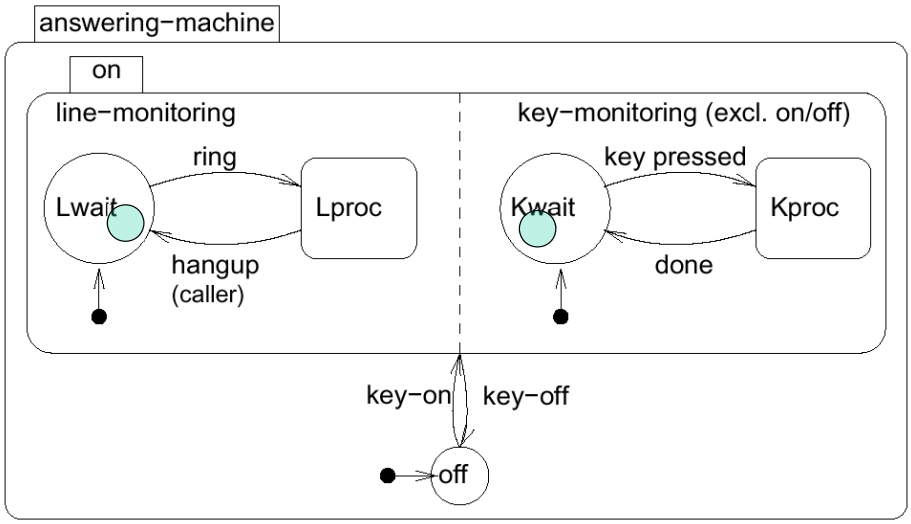
- Beispiel
  - Im Zustand Z5 bewirkt der Event e4 einen Übergang zu Z6
  - Der Event e5 führt nun zu einem Übergang zum früheren Zustand Z5
  - Eine Shallow History würde bei e5 nur in den Zustand Z3, und damit in Z4, wechseln

Combining history and default state mechanism

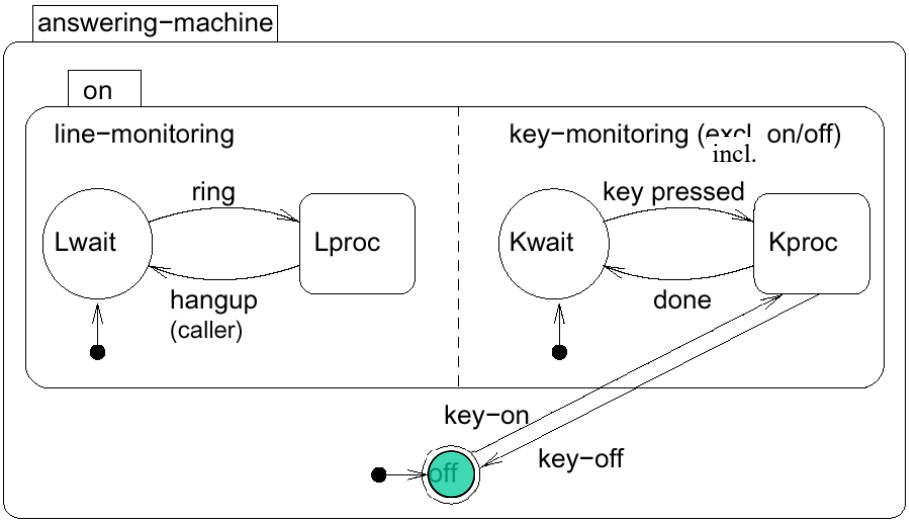


Concurrency

- Convenient ways of describing concurrency are required.
- **AND-super-states**: FSM is in **all** (immediate) sub-states of a super-state; Example:



Entering and leaving AND-super-states



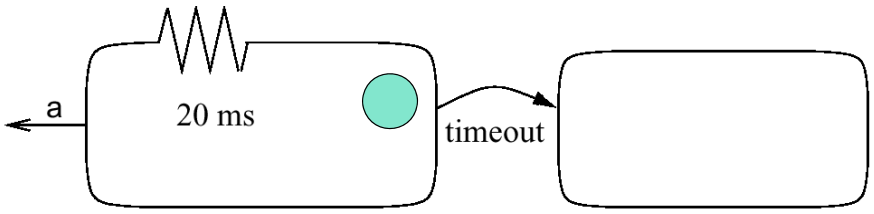
- Line-monitoring and key-monitoring are entered and left, when service switch is operated.

Types of states

- In StateCharts, states are either
  - **basic states, or**
  - **AND-super-states, or**
  - **OR-super-states.**

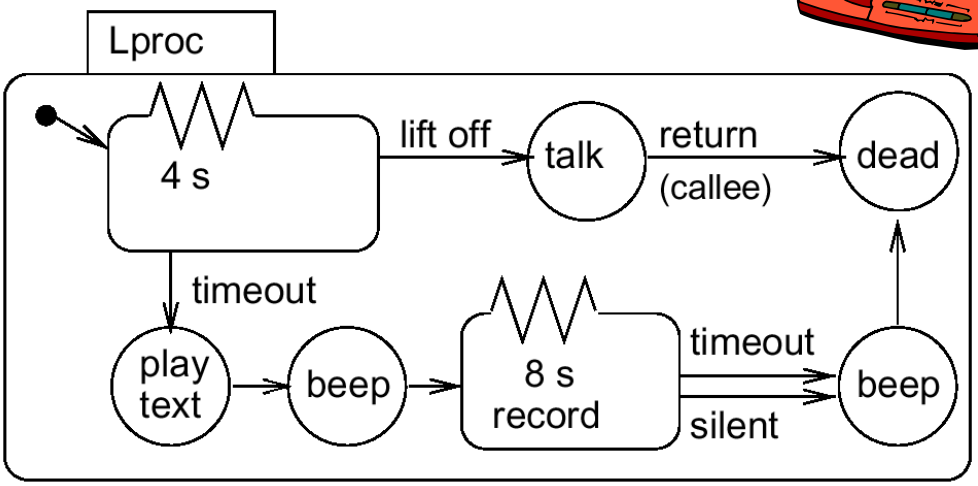
Timers

- Since time needs to be modeled in embedded systems,
- timers need to be modeled.
- 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 an answering machine



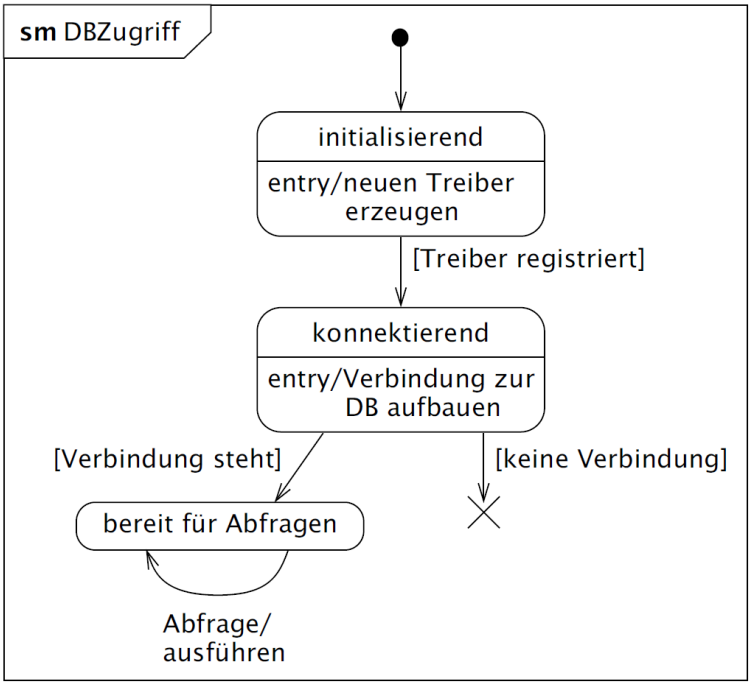
Vergleiche auch

- Statecharts
  - LE 4: pp 87-95
  - LE 8: pp 197-205 (Checkliste)
  - LE 13: pp 339-348

Elemente der Statecharts

- Anfangszustand *(initial pseudo state)*
- Entscheidung *(choice)*
- Kreuzung *(junction)*
- Terminator *(terminate pseudo state)*
- flache Historie *(shallow history)*
- tiefe Historie *(deep history)*
- Eintrittspunkt *(entry point)*
- Austrittspunkt *(exit point)*

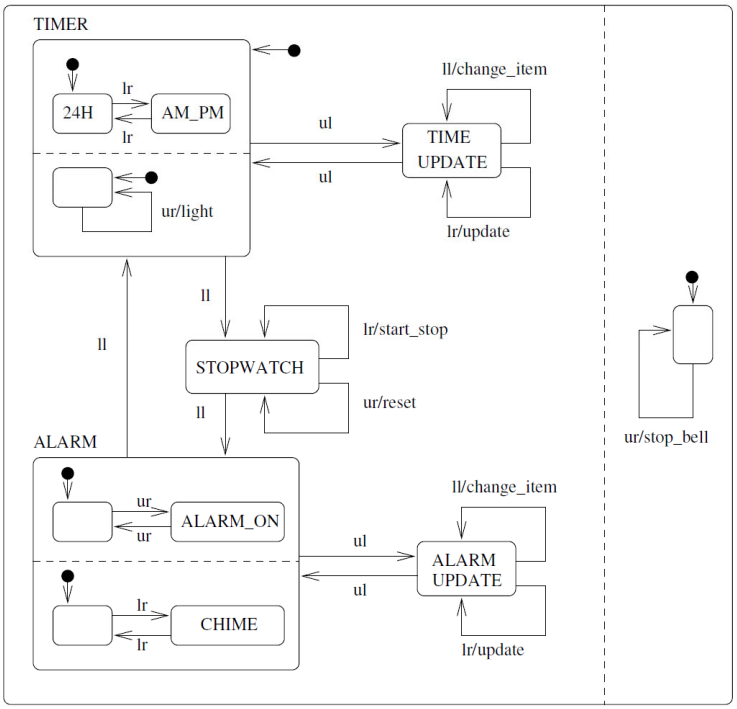
Beispiel: DB-Zugriff

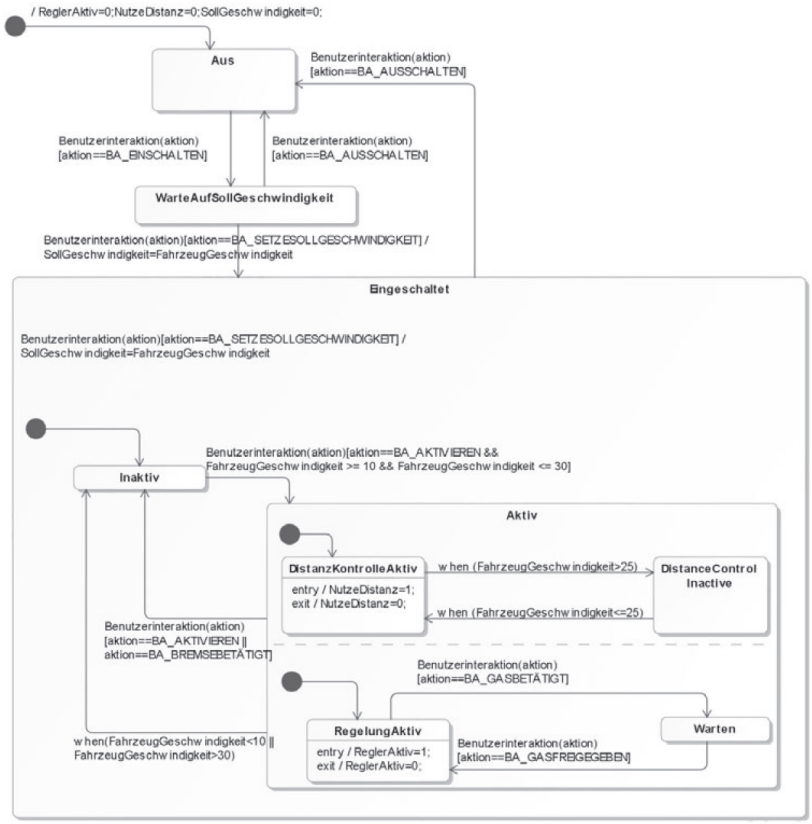


Beispiel: Armbanduhr als Statechart

- *TIMER: In dieser Betriebsart (Grundzustand) wird nur die Uhrzeit dargestellt. In diesem Zustand bedeuten Knopfdruck*
  - *ll: Wechsel in den Zustand STOPWATCH;*
  - *ul: Wechsel in den Zustand TIME_UPDATE;*
  - *lr: Wechsel des Displayzustands (24H oder AM-PM); und*
  - *ur: Einschalten der Beleuchtung.*
- *TIME_UPDATE (Ändern der Zeit):*
  - *ll: Umschalten des zu verändernden Datums (Sekunden, Stunden, Minuten, Tag, etc.);*
  - *lr: Update des selektierten Datums;*
  - *ul: Zurück in den Zustand TIMER;*
- *STOPWATCH (Stoppuhrfunktion):*
  - *ll: Wechsel in Zustand ALARM;*
  - *lr: Start-Stop der Stoppuhr;*
  - *ur: Reset der Stoppuhr;*
- *ALARM (Weckfunktion):*
  - *ll: Wechsel in Zustand TIMER;*
  - *ul: Wechsel in Zustand ALARM_UPDATE;*
  - *lr: Ein- und Ausschalten des Stundensignals (chime);*
  - *ur: Ein- und Ausschalten des Alarms.*

Beispiel: Armbanduhr als Statechart





Ausblick: Themen von nächster Woche

- Realisierung von Finite State Machines