Business Informatics Group

Vienna University of Technology

Objektorientierte Modellierung

Zustandsdiagramm

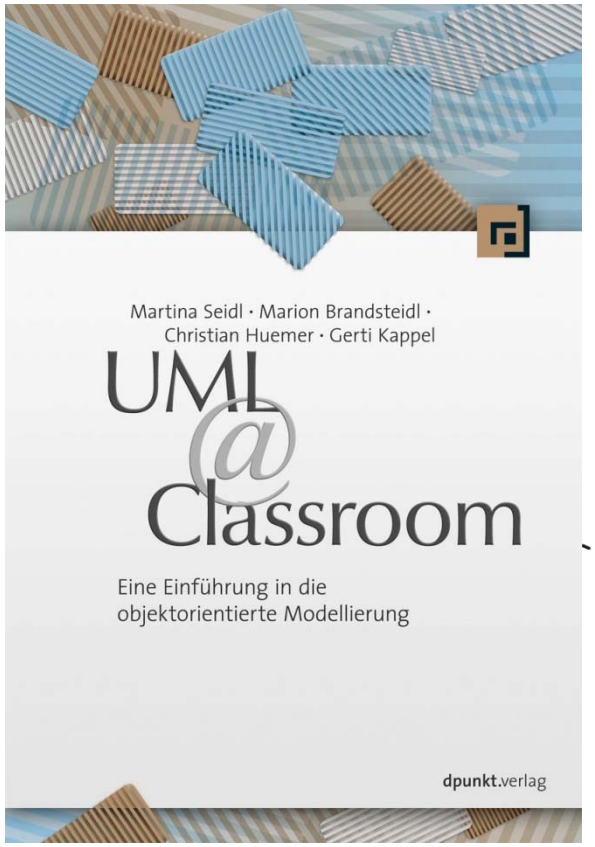
Business Informatics Group

Institute of Software Technology and Interactive Systems
Vienna University of Technology

Favoritenstraße 9-11/188-3, 1040 Vienna, Austria
phone: +43 (1) 58801-18804 (secretary), fax: +43 (1) 58801-18896
office@big.tuwien.ac.at, www.big.tuwien.ac.at

Literatur

- Die Vorlesung basiert auf folgendem Buch:



UML @ Classroom:
Eine Einführung in die objekt-orientierte Modellierung
Martina Seidl, Marion Brandsteidl, Christian Huemer und Gerti Kappel

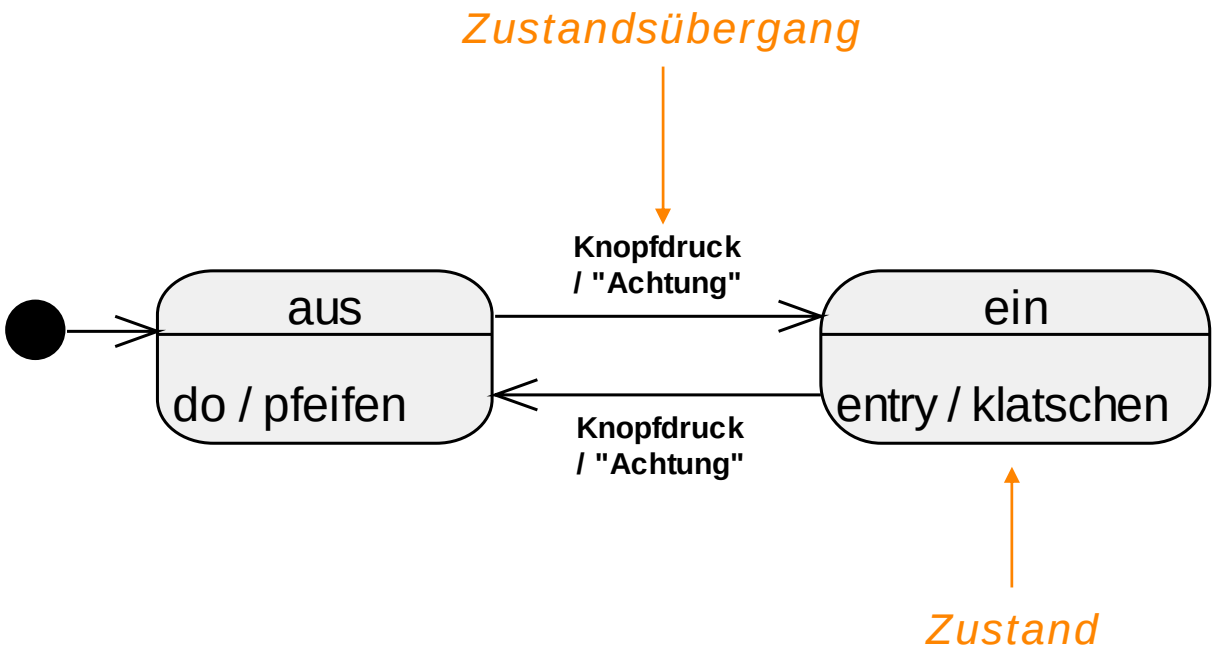
dpunkt.verlag

Juli 2012

ISBN *3898647765*

- *Anwendungsfalldiagramm*
- *Strukturmodellierung*
- ***Zustandsdiagramm***
- *Sequenzdiagramm*
- *Aktivitätsdiagramm*

Bsp.: Zustandsdiagramm einer Lampe



Inhalt

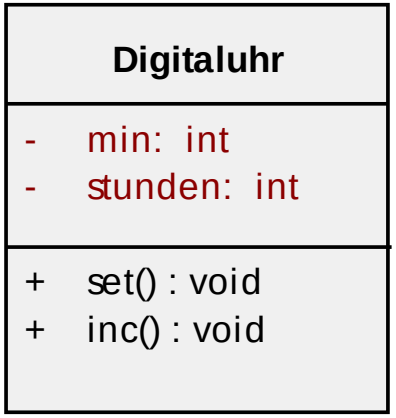
- Einführung
- Zustände und Aktivitäten
- Zustandsübergänge
- Ereignisarten
- Innere Transitionen
- Komplexe Zustände
  - Orthogonale Zustände
  - History-Zustände

Einführung

- Ein Zustandsdiagramm (State Machine Diagram) beschreibt die möglichen **Folgen von Zuständen** eines **Modell-Elements**, i.A. eines Objekts einer bestimmten Klasse
  - Während seines **Lebenslaufs** (Erzeugung bis Destruktion)
  - Während der **Ausführung** einer **Operation** oder **Interaktion**
- Modelliert werden
  - Die **Zustände**, in denen sich die Objekte einer Klasse befinden können
  - Die möglichen **Zustandsübergänge** (Transitionen) von einem Zustand zum anderen
  - Die **Ereignisse**, die Transitionen auslösen
  - **Aktivitäten**, die in Zuständen bzw. im Zuge von Transitionen ausgeführt werden

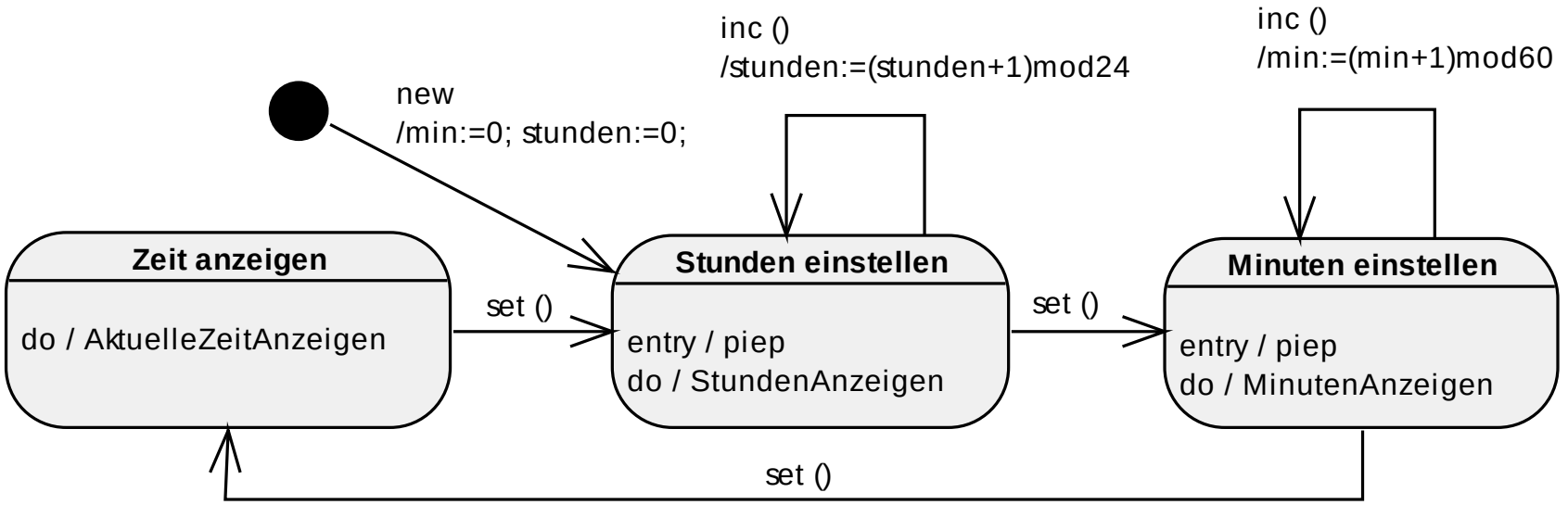
Einführung – Beispiel Digitaluhr (1/2)

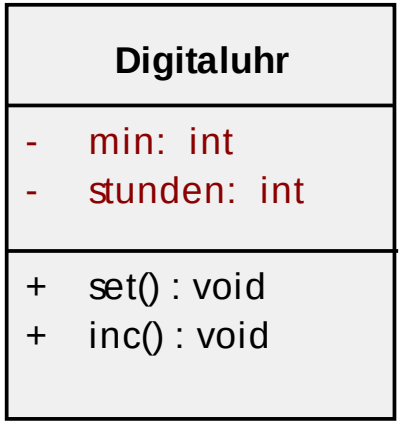
- Modelliert werden sollen die Zustände, die eine Digitaluhr beim Stellen der Uhr einnehmen kann.
- Die Uhr kann 3 Zustände einnehmen:
  - Zeit anzeigen
  - Stunden einstellen
  - Minuten einstellen



- Durch die Betätigung des Einstellungsknopfes wird von "Zeit anzeigen" in "Stunden einstellen" gewechselt, von "Stunden einstellen" in "Minuten einstellen“ und schließlich von "Minuten einstellen" wieder in "Zeit anzeigen".
- Wird in "Stunden einstellen" gewechselt, piepst die Uhr und zeigt die Stunden an. Durch die Betätigung des inc-Buttons, wird die Stundenanzahl um 1 erhöht.
- "Minuten einstellen" funktioniert analog.
- Am Anfang befindet sich die Uhr im Zustand "Stunden einstellen".

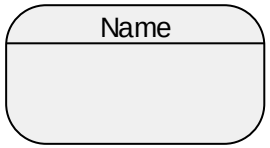
Einführung – Beispiel Digitaluhr (2/2)





Zustand

- („echter“) Zustand (state)
  System kann sich dauerhaft im Zustand befinden
  - Zustand im eigentlichen Sinn

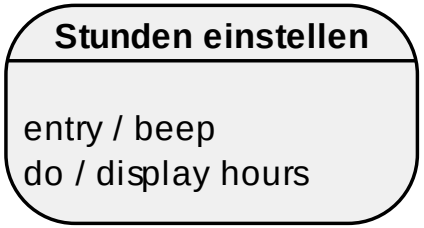


  - Endzustand

- Pseudozustand
  transient (System kann nicht dauerhaft in einem Pseudozust. sein)
  - Startzustand 
  - Flacher/Tiefer History-Zustand
  - Parallelisierungsknoten & Synchronisierungsknoten
  - Terminierungsknoten
  - Entscheidungsknoten

Aktivitäten innerhalb eines Zustands

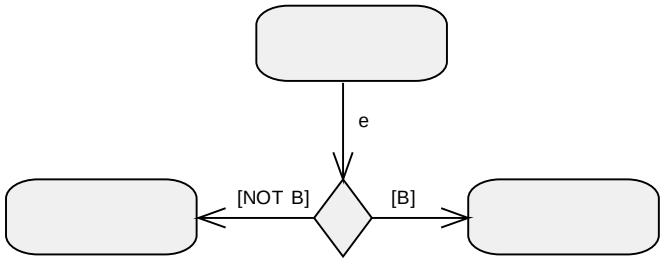
- **entry** / aktivität
  - Wird beim Eingang in den Zustand ausgeführt
- **exit** / aktivität
  - Wird beim Verlassen des Zustands ausgeführt
- **do** / aktivität
  - Wird ausgeführt, Parameter sind erlaubt

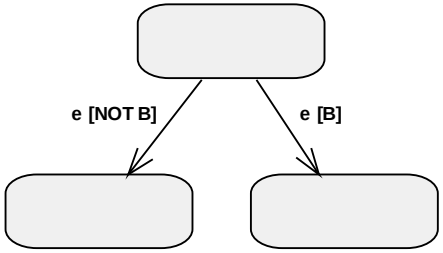


- **event** / aktivität
  Aktivität behandelt Ereignis innerhalb des Zustands
  - Wird ausgeführt, wenn sich das System in dem Zustand befindet und das Ereignis eintritt

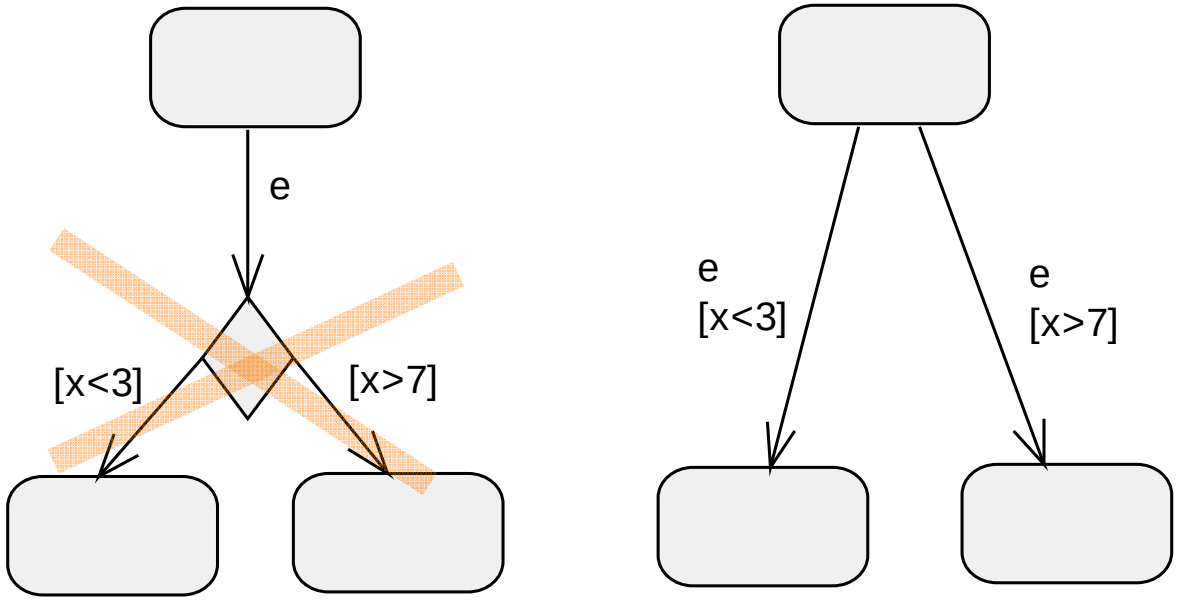
Zustandsübergang (1/2)

- Ein Zustandsübergang (Transition) erfolgt, wenn
  - das **Ereignis** eintritt
    - eine evt. noch andauernde **Aktivität** im Vorzustand wird **unterbrochen**!
  - und die **Bedingung** (guard) **erfüllt** ist
    - bei Nicht-Erfüllung geht das nicht »konsumierte« Ereignis verloren
      => wenn die Bedingung erst zu einem späteren Zeitpunkt erfüllt wird, kann die Transition ohne neuerliches Ereignis nicht durchgeführt werden
- Durch entsprechende Bedingungen können **Entscheidungsbäume** modelliert werden





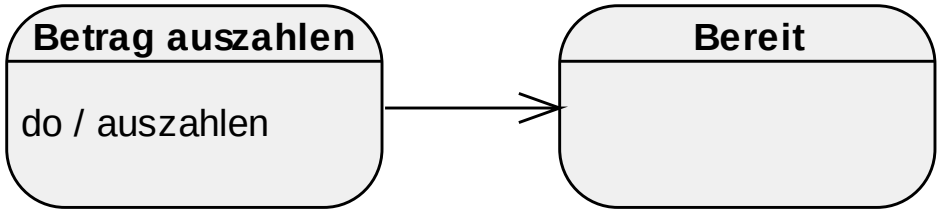
Zustandsübergang (2/2)



Default Werte für Zustandsübergänge

- Default-Werte
  - Fehlendes Ereignis entspricht dem Ereignis »Aktivität ist abgeschlossen«
  - Fehlende Bedingung entspricht der Bedingung [true]

- Beispiel: Bankomat

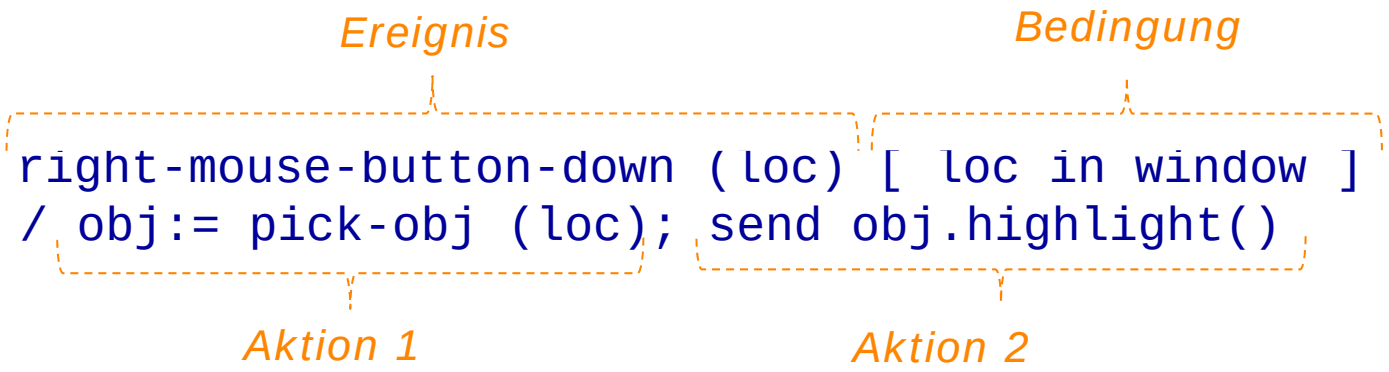


Syntax von Zustandsübergängen

- Ereignisse, Bedingungen und Aktivitäten auf Zustandsübergängen möglich
- Notation: `Ereignis(Argumente) [Bedingung] / Aktivität`
  - Die Aktivität kann aus mehreren Aktionen bestehen
  - Spezielle Aktivität: Nachricht an anderes Objekt senden

    send empfänger.nachricht()

- Beispiel:



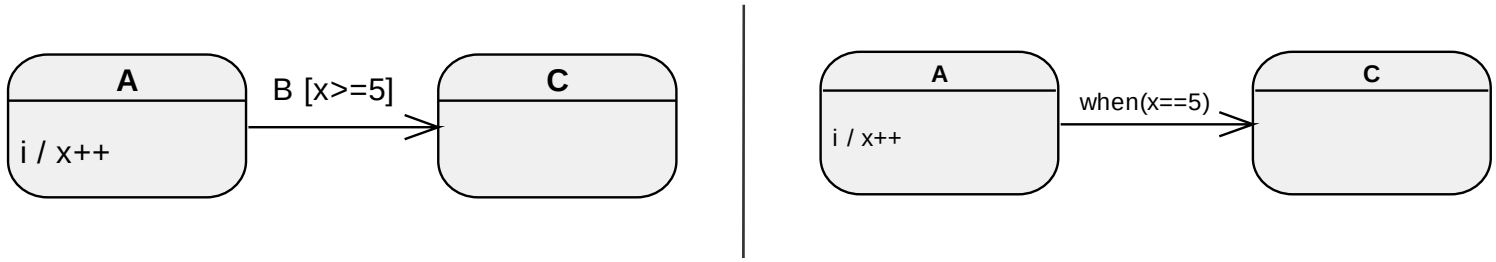
Zustandsübergang: Ereignistypen (1/2)

- **CallEvent**
  Empfang einer Nachricht (Operationsaufruf)
  - Bsp.: stornieren(), kollidiertMit(Termin)
- **SignalEvent**
  Empfang eines Signals
  - Bsp.: right-mouse-button-down, ok-Taste-gedrueckt
- **ChangeEvent**
  Eine Bedingung wird wahr
  - Bsp.: when(x<y), when(a=1), when(terminBestaetigt)
- **TimeEvent**
  Zeitablauf oder Zeitpunkt
  - Bsp.: after(5 sec.), when(date=31.01.2008)

Zustandsübergang: Ereignistypen (2/2)

- Unterschied ChangeEvent und Bedingung

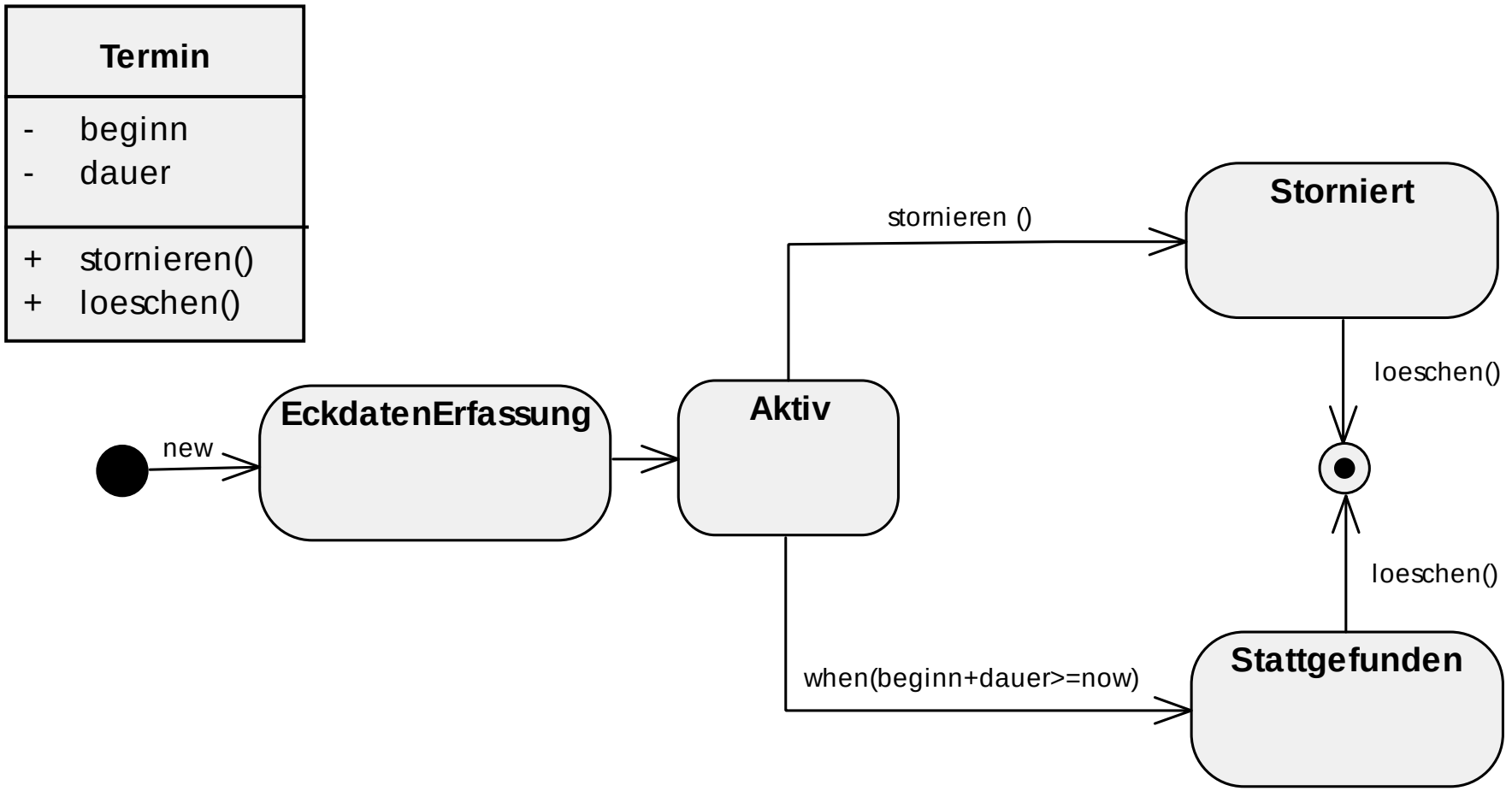
- **ChangeEvent**:
  - Bedingung wird permanent geprüft
  - wenn Bedingung wahr ist, kann zugehöriger Zustandsübergang ausgelöst werden (falls nicht durch zugehörige Überwachungsbedingung blockiert)
- **Bedingung**:
  - wird nur geprüft, wenn zugeordnetes Ereignis eintritt
  - kann selbst keinen Zustandsübergang auslösen



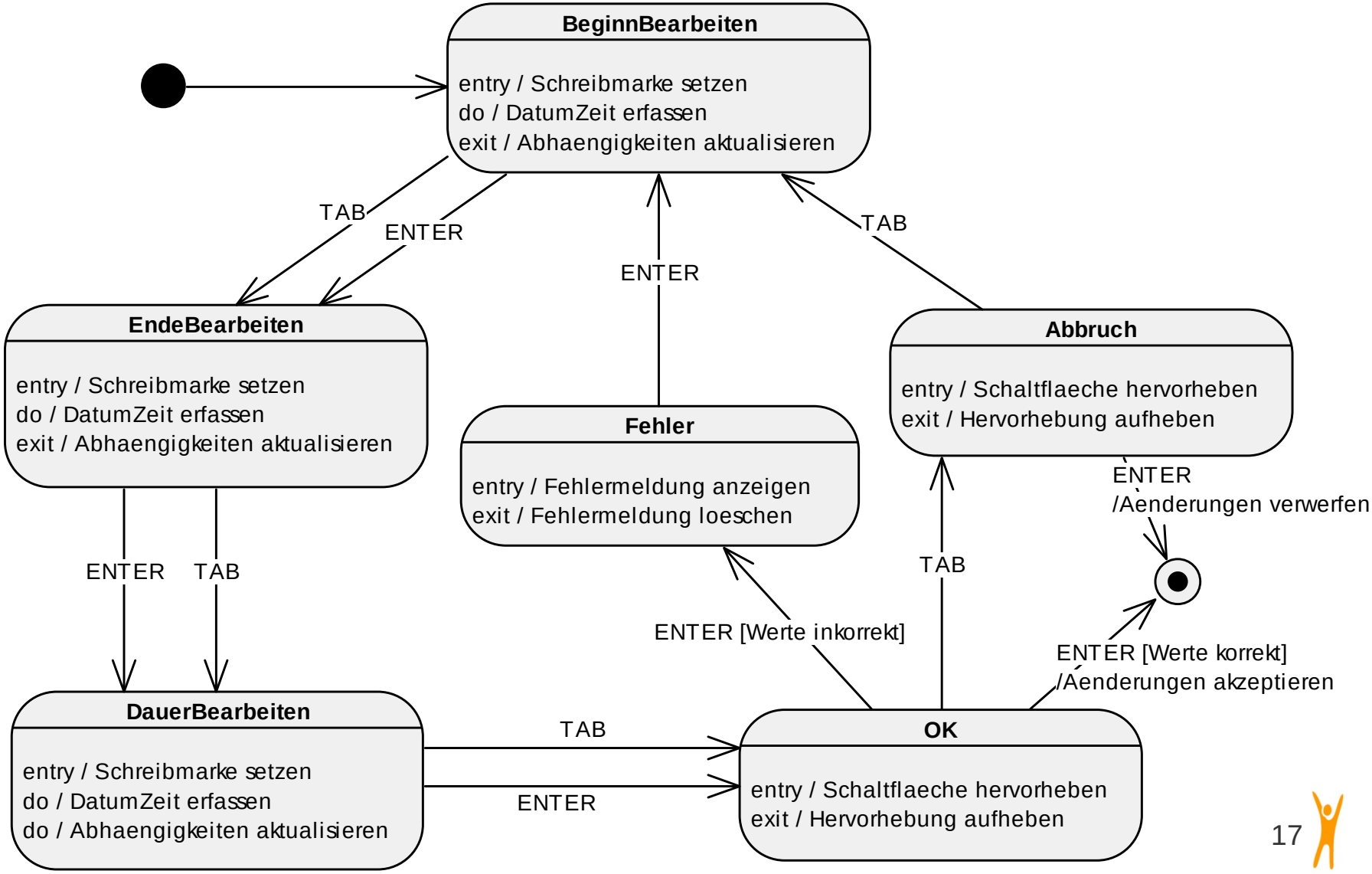
Start- u. Endzustand, Terminierungsknoten

- **Startzustand**
  - "Beginn" des Zustandsdiagramms
  - keine eingehenden Transitionen
  - genau eine ausgehende Transition
    - wird sofort ausgelöst, wenn sich das System im Startzustand befindet
    - keine Bedingungen und Ereignisse (Ausnahme: Ereignis zur Erzeugung des betrachteten Objekts)
    - Angabe von Aktivitäten ist erlaubt
- **Endzustand**
  - keine ausgehenden Transitionen
  - kein Pseudozustand!
- **Terminierungsknoten**
  - Objekt, dessen Verhalten modelliert wird, hört auf zu existieren

Bsp.: Lebenszyklus eines Termins im CALENDARIUM



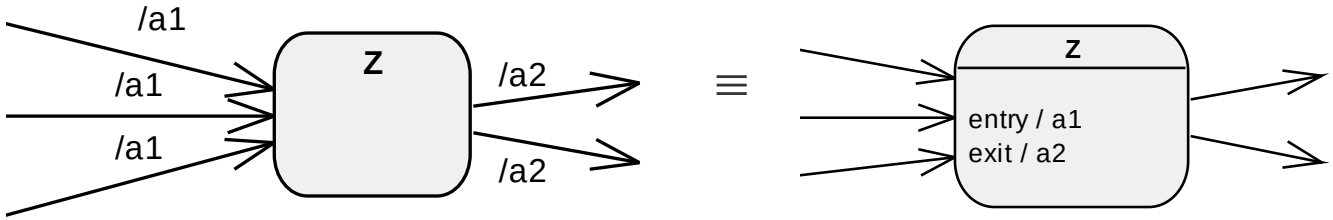
Bsp.: Lebenszyklus einer Eingabemaske im CALENDARIUM



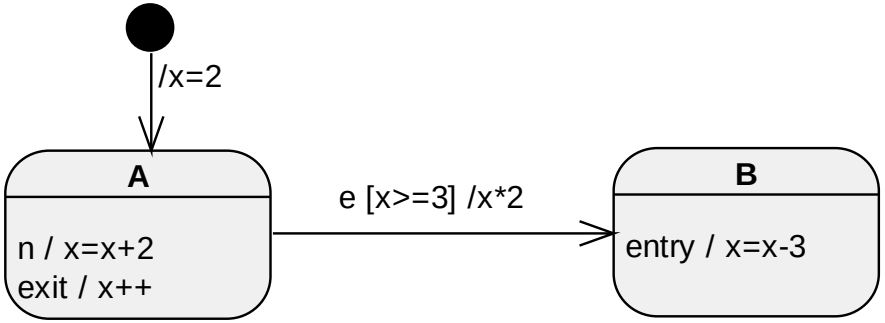
Zustandsübergang: Innere Transitionen

- Werden wie »äußere« Transitionen von Ereignissen ausgelöst, **verlassen** aber den **aktuellen Zustand nicht**
- Äquivalent zu Selbsttransition, sofern keine entry / exit-Aktivitäten vorhanden

- Gleiche Aktivitäten können in den Zustand hineingezogen werden:



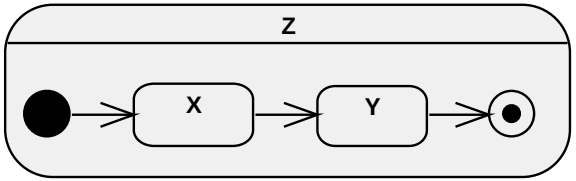
Ausführungsreihenfolge von Aktivitäten – Bsp.



Event	Zustand	Variable
„Start“	A	x=2
e	A	x=2
n	A	x=4
e	B	x=7

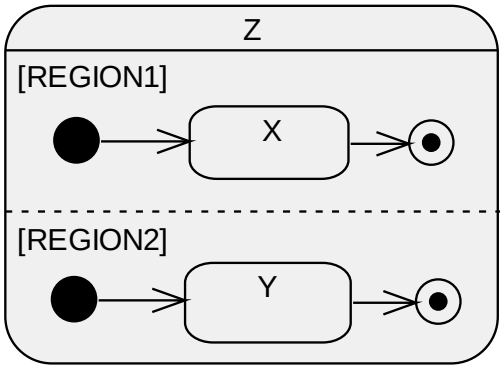
Komplexe Zustände (1/2)

- = Zustände, die aus mehreren Subzuständen zusammengesetzt sind
  - ⇨ geschachteltes Zustandsdiagramm
- Die Subzustände sind disjunkt, d.h. genau ein Subzustand ist aktiv, wenn der komplexe Zustand aktiv ist



Zu einem Zeitpunkt kann nur X ODER Y aktiv sein!

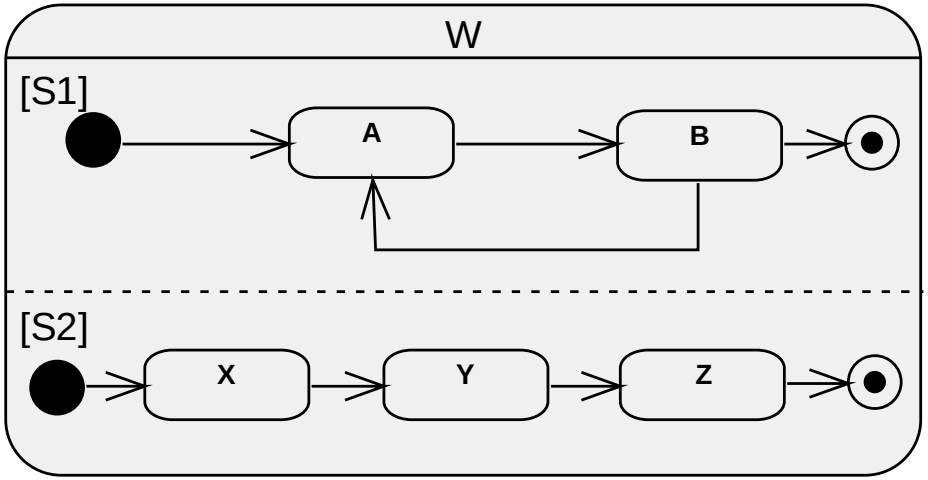
- Teilung des Superzustandes in mehrere Regionen
  - → die Subzustände sind nebenläufig, gleichzeitig aktiv
  - Z = „orthogonaler Zustand“



Zu einem Zeitpunkt sind X UND Y aktiv!

Komplexe Zustände (2/2)

- Beispiel



Zu einem Zeitpunkt kann nur A ODER B aktiv sein!

Zu einem Zeitpunkt kann nur X ODER Y ODER Z aktiv sein!

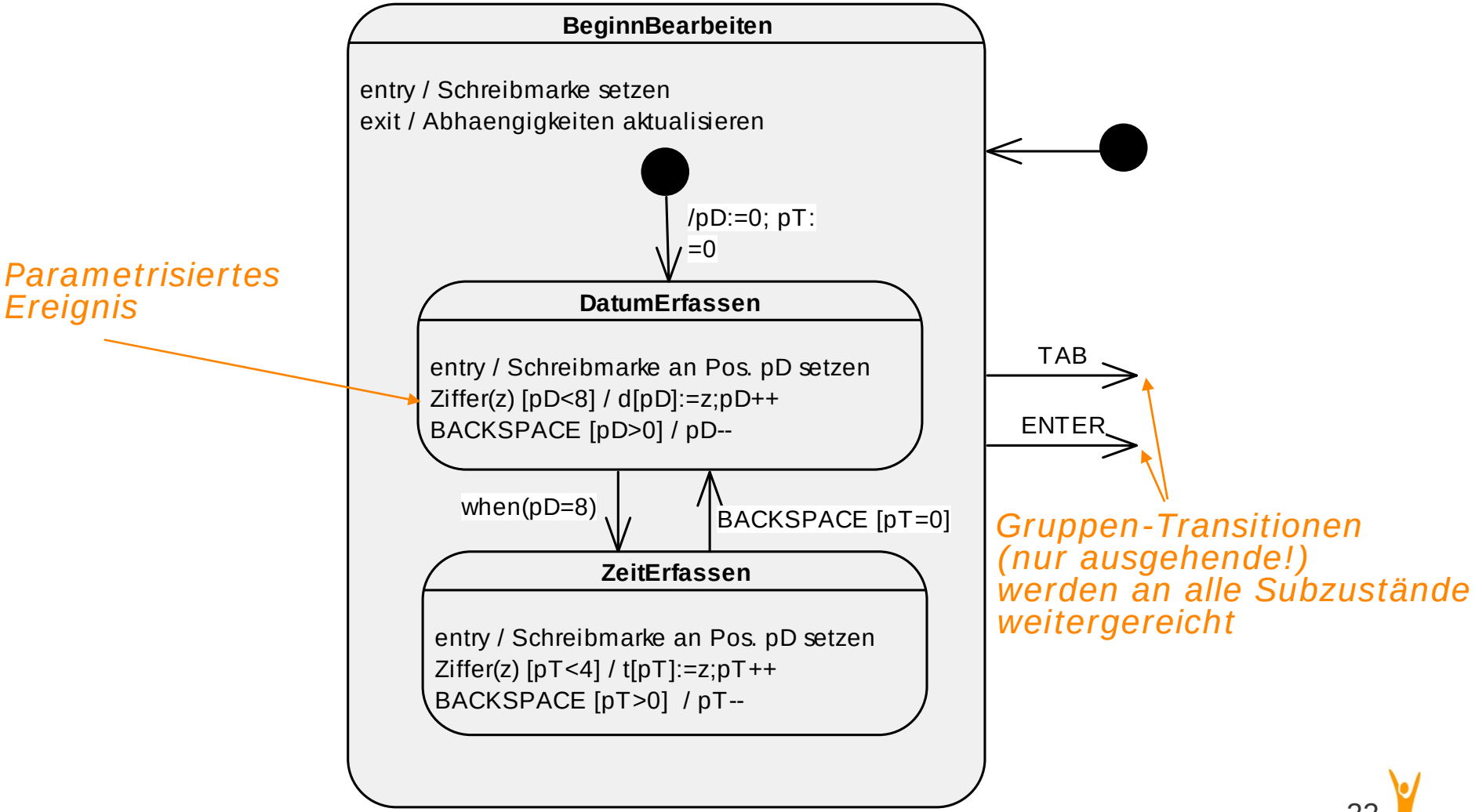
Zu einem Zeitpunkt ist jeweils ein Subzustand jeder der beiden orthogonalen (=parallelen) Regionen von W aktiv!

- Mögliche Kombinationen von gleichzeitig aktiven Zuständen:

  A & X *oder* A & Y *oder* A & Z *oder* A & Endzustand von [S2]
  B & X *oder* B & Y *oder* B & Z *oder B &* Endzustand von [S2]
  Endzustand von [S1] & X *oder* Endzustand von [S1] & Y *oder* Endzustand von [S1] & Z *oder* Endzustand von [S1] & Endzustand von [S2]

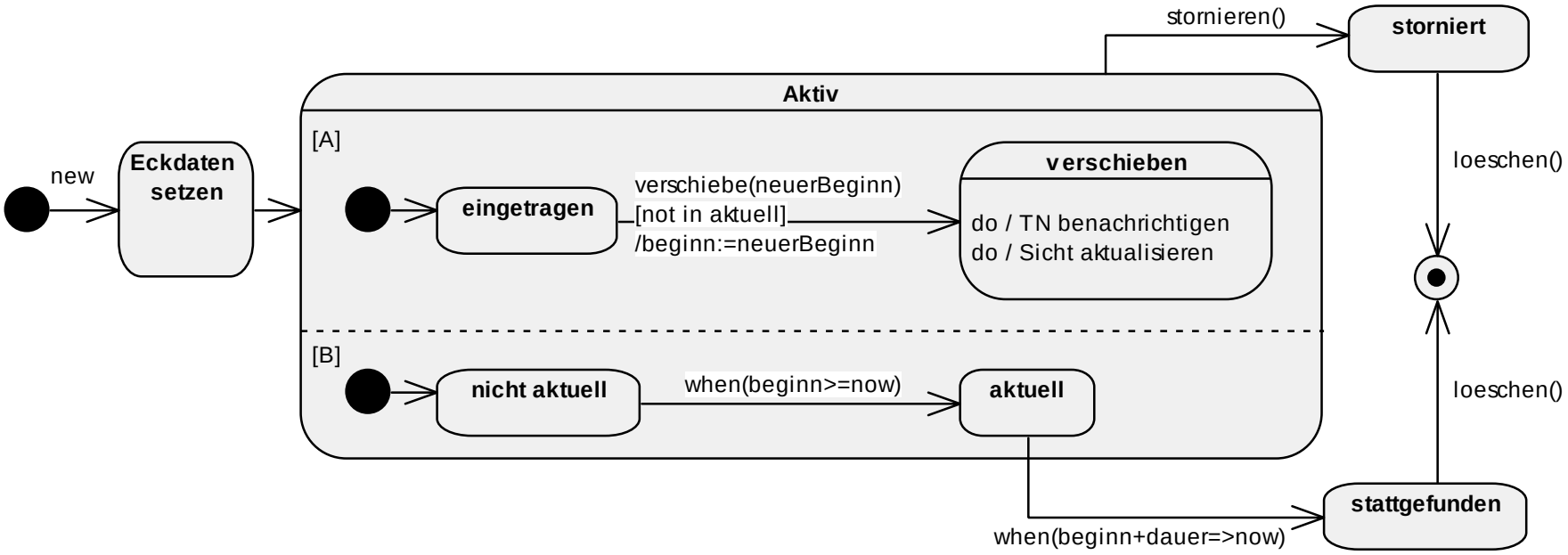
Bsp.: Komplexer Zustand »BeginnBearbeiten«

(*CALENDARIUM-Bsp*)

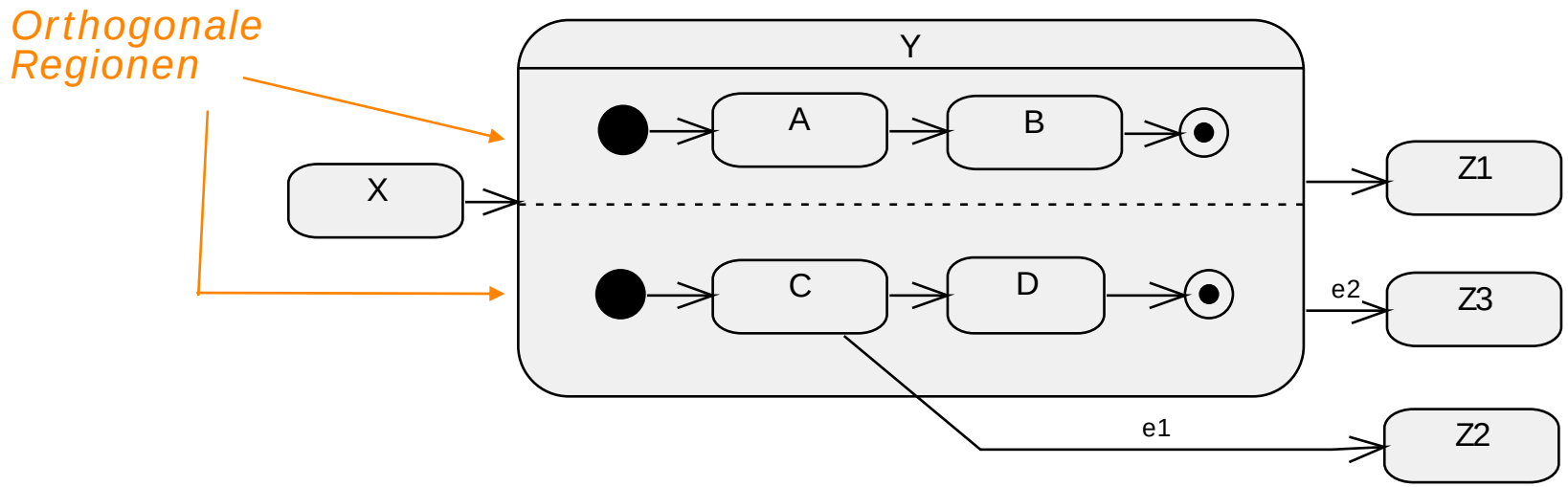


Bsp.: Komplexer Zustand »Aktiv«

- …vom Lebenslauf eines Termins



Komplexer Zustand – Verlassen von komplexen Zuständen



- Der komplexe Zustand Y wird verlassen, wenn
  - B und D verlassen worden sind (Folgezustand Z1)
    [= die Subzustandsfolgen beendet sind] *oder*
  - im Zustand C Ereignis e1 eintritt (Folgezustand Z2) *oder*
  - in irgendeinem Subzustand Ereignis e2 eintritt (Folgezustand Z3)

Historischer Zustand

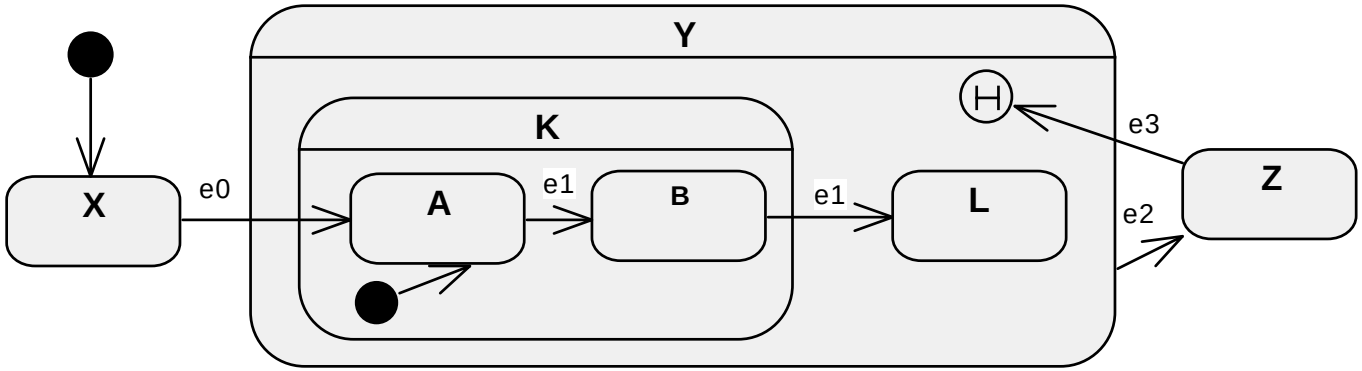
- Historische Zustände können sich jenen internen **Zustand** in einem komplexen Zustand **merken**, von dem die **letzte Transition** (vor einer Unterbrechung) **ausgegangen** ist

- Zu einem späteren Zeitpunkt kann zu diesem Zustand über Transitionen **aus übergeordneten Zuständen zurückgekehrt** werden
  - alle Entry-Aktivitäten werden wiederum ausgeführt

- **Flacher History-Zustand** merkt sich **eine Ebene** 

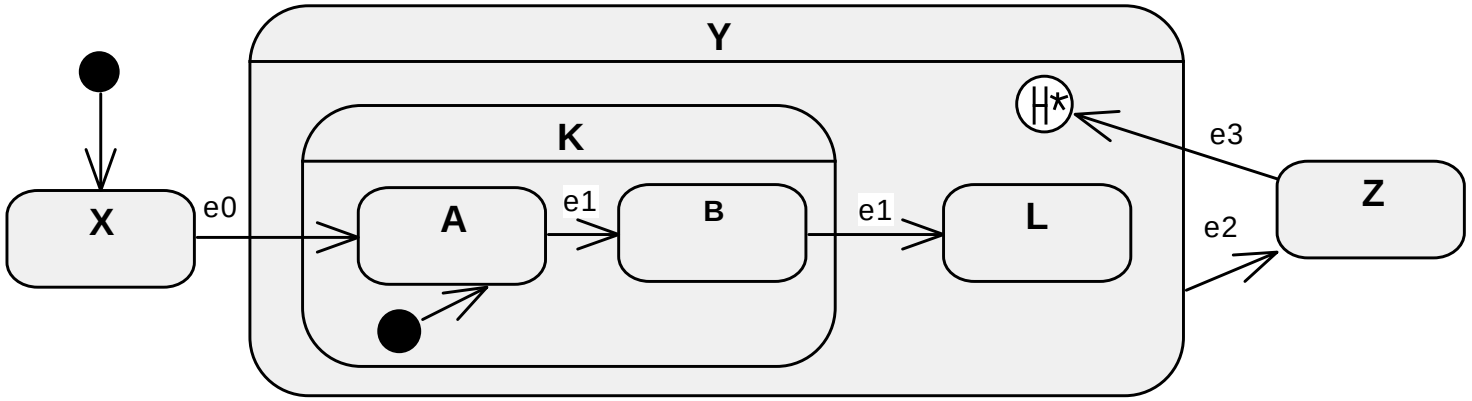
- Über einen **tiefen History-Zustand** »**H***« werden alle Zustände über die **gesamte Schachtelungstiefe** hinweg gesichert 

Bsp.: H vs. H* (1/2)



Event	Zustand
„Start“	X
e0	Y/K/A
e2	Z
e3	(H→) Y/K/A
e1	Y/K/B
e1	Y/L
e2	Z
e3	(H→) Y/L

Bsp.: H vs. H* (2/2)



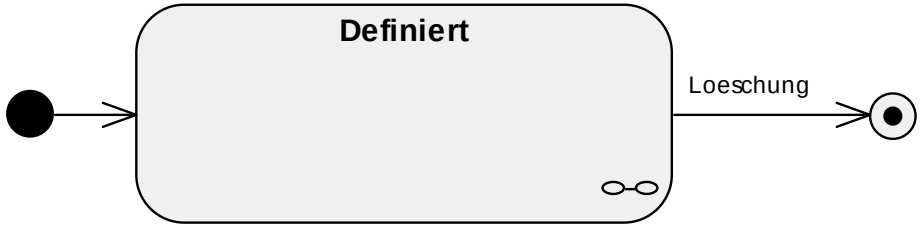
Event	Zustand
„Start“	X
e0	Y/K/A
e1	Y/K/B
e2	Z
e3	(H*→) Y/K/B

Bsp.: Lebenslauf eines to-do-Eintrags

- Lebenslauf eines to-do-Eintrags kann auf folgende Weise vereinfacht dargestellt werden:
  - **Vereinfachung 1**: Markierung der Transition vom Superzustand zum äußeren Endzustand mit dem Ereignis »Löschung« - alle inneren Löschung-Transitionen und alle inneren Endzustände entfallen
  - **Vereinfachung 2**: Zusätzliches **Ausblenden** der Verfeinerung

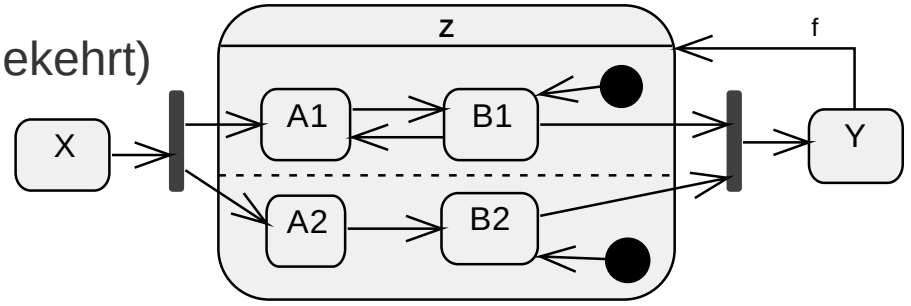
    d.h. Verfeinerung wird an anderer Stelle dargestellt –

    Notation:



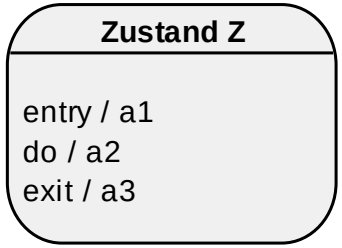
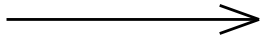
Komplexe Transition für Orthogonale Zustände

- Wird ein orthogonaler Zustand aktiviert, so werden **alle** seine **nebenläufigen Regionen aktiviert**
- Möchte man den Kontrollfluss jedoch anders aufspalten und nicht in allen Regionen die Startzustände aktivieren
  - Verwendung einer komplexen Transition in Form Parallelisierungsknoten bzw. Synchronisierungsknoten
  - Dieser kann eine Transitionsspezifikation tragen, die Pfeile der Zustandsübergänge selbst sind unmarkiert
  - Forderung bei Parallelisierungsknoten: Nachzustände müssen unterschiedlichen Regionen angehören und ihr Vorzustand muss außerhalb liegen
    (Bei Synchronisierungsknoten umgekehrt)

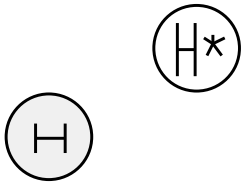


- Eine komplexe Transition ändert damit den »Grad an Nebenläufigkeit«

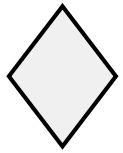
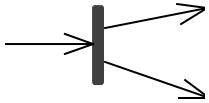
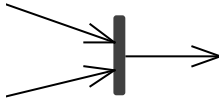
Basiselemente (1/3)

Name	Syntax	Beschreibung
Zustand		Bei Erreichen des Zustands Z wird die Aktivität a1 ausgeführt, während Z der aktuelle Zustand ist, wird a2 ausgeführt und beim Verlassen von Z wird a3 ausgeführt.
Transition		Zustandsübergang
Startzustand		Beginn des Zustandsdiagramms

Basiselemente (2/3)

Name	Syntax	Beschreibung
Endzustand		Ende
Terminierungs-knoten		Das modellierte Objekt hört auf zu existieren.
Flacher/tiefer History-Zustand		"Rücksprungadresse"

Basiselemente (3/3)

Name	Syntax	Beschreibung
Entscheidungs-knoten		Knoten, von dem mehrere alternative Transitionen ausgehen können.
Parallelisierungs-knoten		Aufspaltung des Kontrollflusses in mehrere parallele Zustände
Synchronisierungs-knoten		Zusammenführung des Kontrollflusses von mehreren parallelen Zuständen

Zusammenfassung

- Sie haben diese Lektion verstanden, wenn Sie wissen..

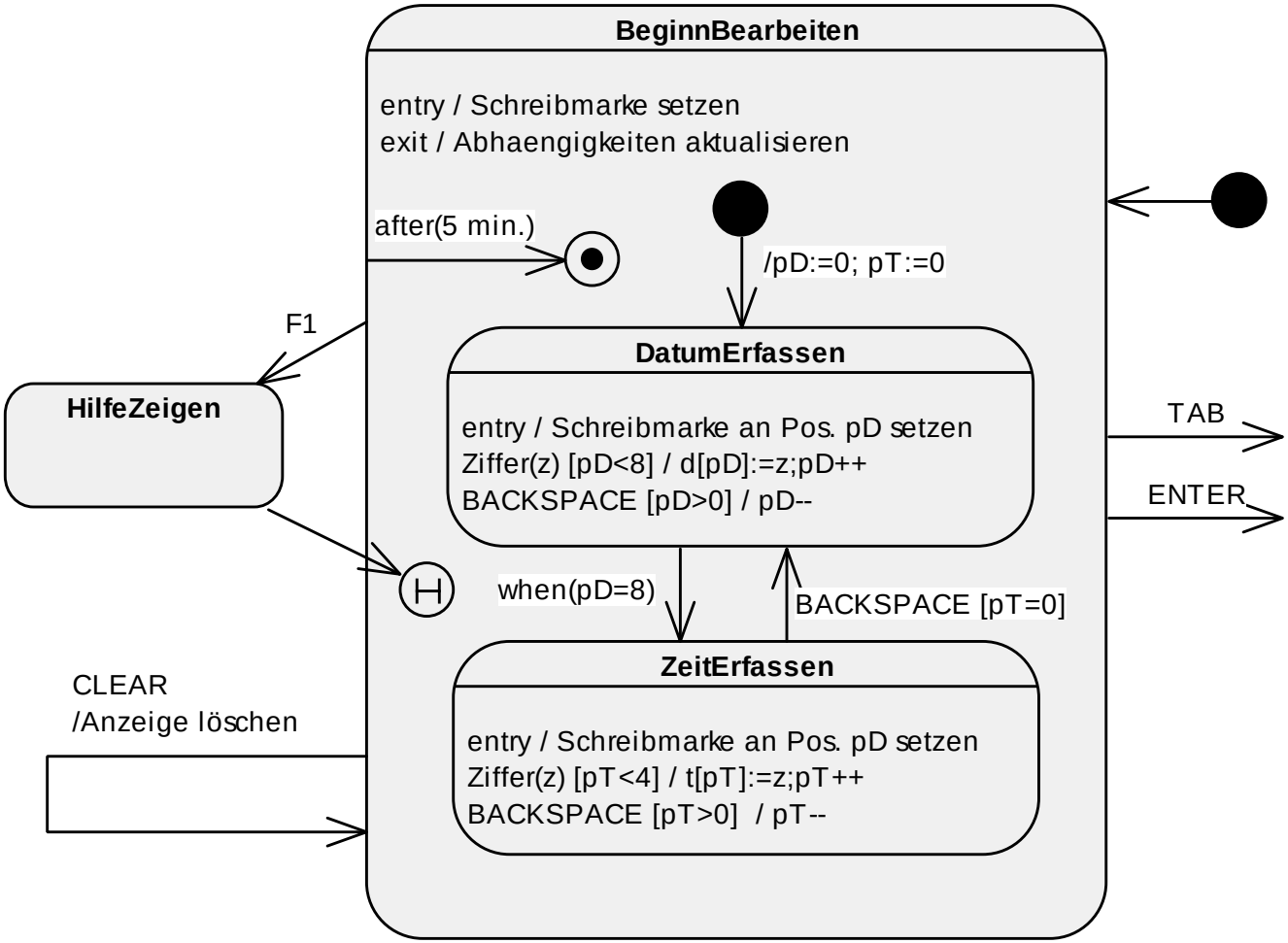
- Was mit dem Zustandsdiagramm modelliert wird
- Was Ereignisse und Aktivitäten sind und wie sie eingesetzt werden
- Wozu Bedingungen benötigt werden und was der Unterschied zu Ereignissen ist
- Welche Aktivitäten es innerhalb eines Zustands gibt
- Wozu und wie ein Historischer Zustand eingesetzt wird
- Was komplexe und orthogonale Zustände sind

Anhang

Bsp.: Lebenslauf einer Termin-Eingabemaske

- Variante des komplexen Zustands »BeginnBearbeiten«

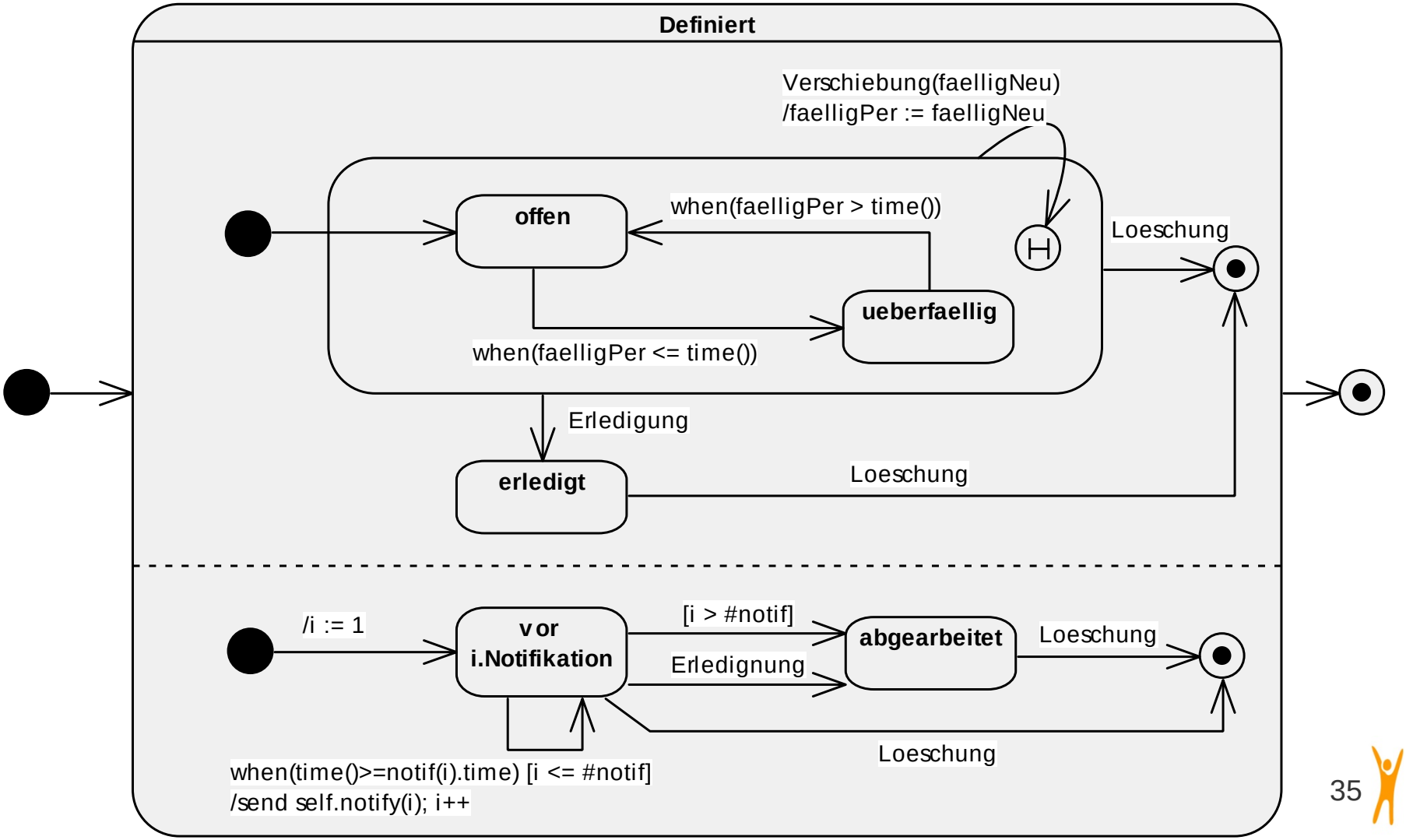
(*CALENDARIUM-Bsp*)



Bsp.: Lebenslauf eines to-do-Eintrags

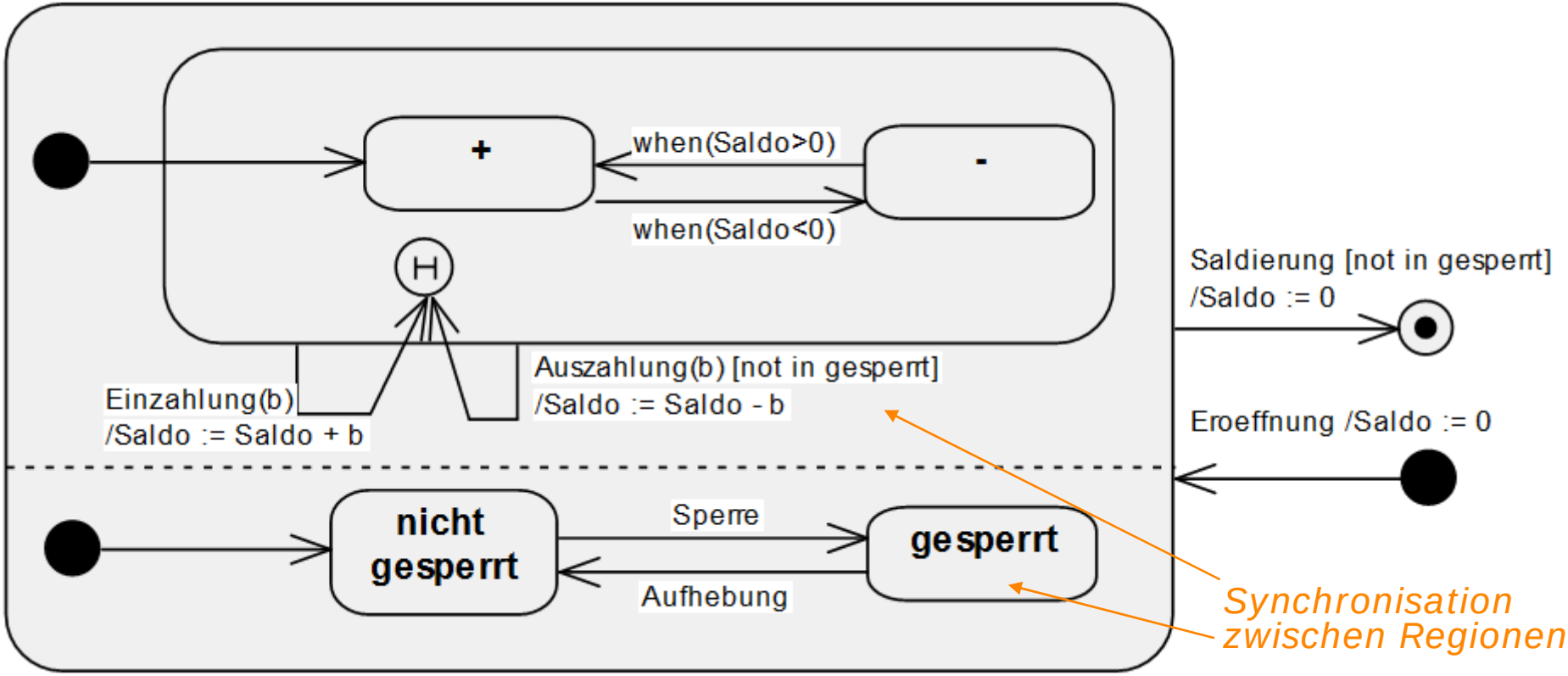
(*CALENDARIUM-Bsp*)

- Komplexer Zustand »Definiert«



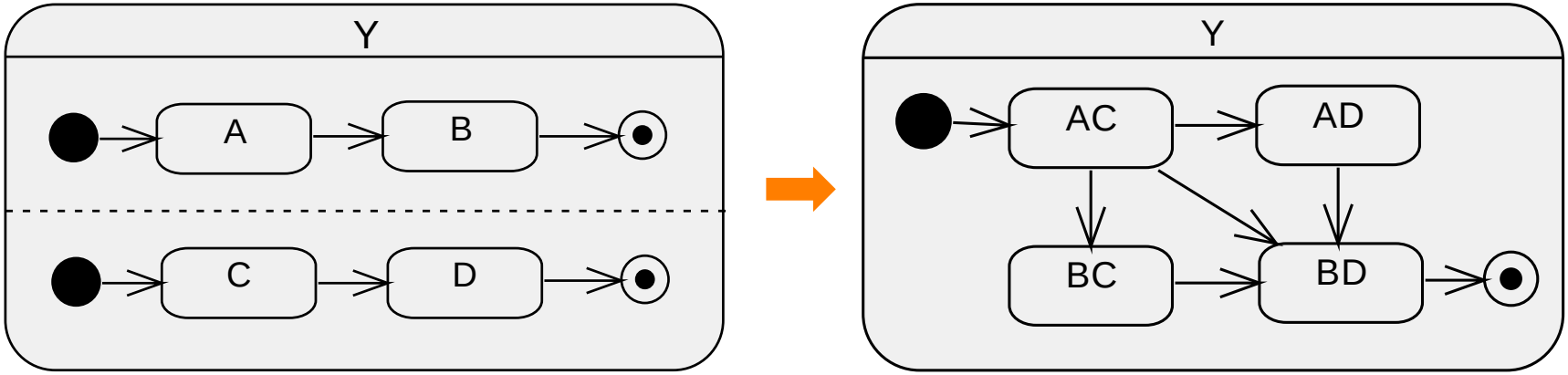
Bsp.: Konto

- Modellierung von unabhängigen Zustandsmengen eines Objektes ("mehrdimensionale Modellierung") durch orthogonale Regionen



Orthogonale Zustände vs. sequentielle Form

- Orthogonale Zustände können durch die Erzeugung von Produktautomaten auf nicht orthogonale Zustände abgebildet werden
- Für jede mögliche Kombination von orthogonalen Zuständen wird ein eigener Zustand definiert – Transitionen werden entsprechend dupliziert
- Beispiel:



Bsp.: orthogonale Zustände vs. sequent. Form

- Transformation eines orthogonalen Zustands in eine sequentielle Form ist **umständlich** und **semantisch nicht äquivalent**
  - Erzeugung von **Produktautomaten**
  - Beispiel: Konto
  - »G« und »NG« stehen für »gesperrt« bzw. »nicht gesperrt«