Kurs OMSI
im WiSe 2013/14

Objektorientierte Simulation mit ODEMx

Prof. Dr. Joachim Fischer
Dr. Klaus Ahrens
Dipl.-Inf. Ingmar Eveslage

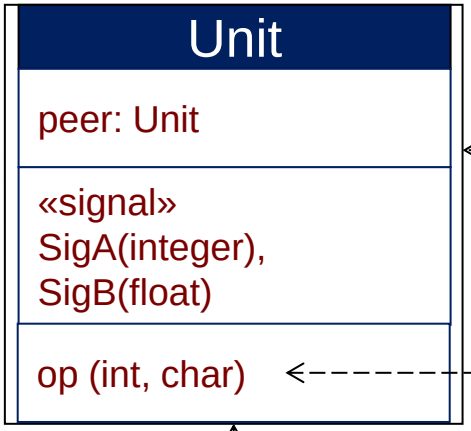
fischer|ahrens|eveslage@informatik.hu-berlin.de

1. Einführung

1. Systemsimulation – was ist das?
2. Ein Blick zurück in die Anfänge
3. Modelle und Originale
4. Modellierungssprachen, Simulationsumgebungen
5. Bespiele aus der aktuellen Forschung
6. Paradigma der objektorientierten Modellierung
7. Einordnung von UML
8. Klassifikation dynamischer Systeme
9. M&S eines Niedertemperaturofens

Klassen – Aktivitäten - Zustandsmaschinen

Aktive Klasse



S1:UnitSM

Instanz einer Zustandsmaschine

Verhaltensbeschreibungseinheit
- operiert über Datenstruktur des zugeordneten Kontext-Objektes,
- verfügt implizit über Puffer ausgelöster Trigger
- erwartet werden SigA- und SigB-Signale
- Zustandsübergänge können die Operation op einsetzen

Instanz einer Klasse

U: Unit

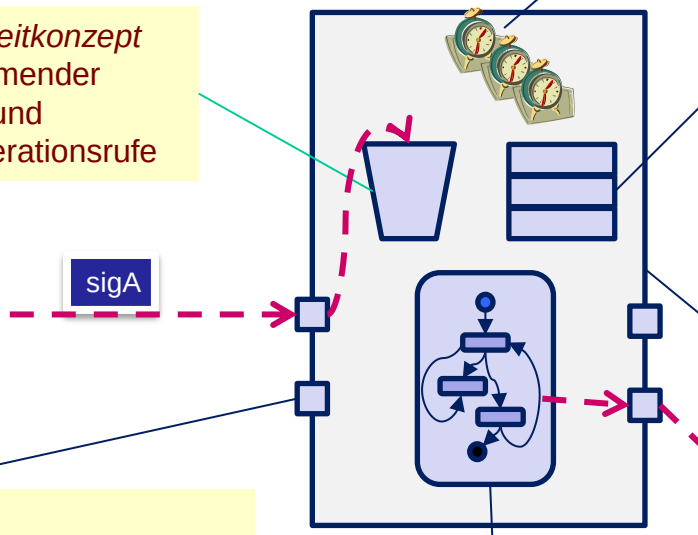
A1:UnitOp

Instanz einer Aktivität

Instanz der aktiven Klasse KommEinheit
stellt den Kontext zugeordneter Verhaltensbeschreibungen

Verhaltensbeschreibungseinheit
- beschreibt Verhalten von op
- operiert über Datenstruktur des zugeordneten Kontext-Objektes

Automaten zur Laufzeit~ *Ensemble verschiedener Entitäten*



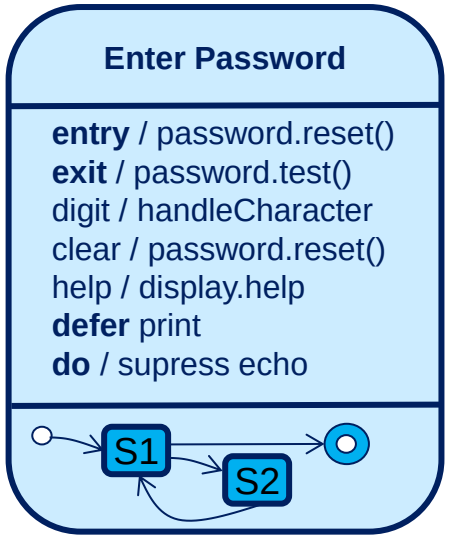
Zustand eines Agents zu einem Zeitpunkt:

- Stand der gestarteten Timer
- Belegung des Empfang-Pools (inklusive aktueller Parameter der Signale u. Op-Rufe)
- Wertebelegung der Attribute des zugeordneten Objektes
- **Zustand der Zustandsmaschine**
- Funktionsaufruf-Stack
- Befehlsregister (falls im Zustandsübergang)

Achtung (behandeln wir nicht):

- Spezialisierung (Vererbung) von Zustandsautomaten
- **hierarchische Zustände**
- orthogonale Zerlegung (in parallel) agierende Untermaschinen

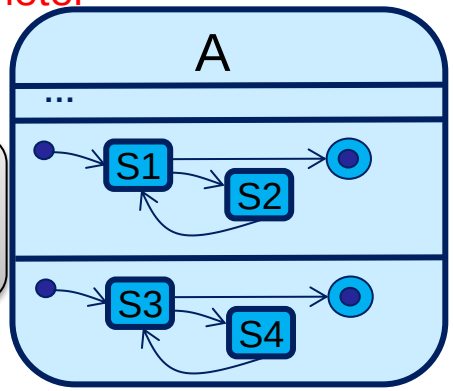
UML-Zustandsdefinition



Bestandteile/Charakteristik

- Name
  (optional), kein Name, dann anonymer Zustand
- Entry-/ Exit- Aktionen bei Eintritt bzw. Austritt
- interne Do-Aktivität
  durch externes Ereignis unterbrechbar,
  d.h. per Signal-, Call-, Time-, State-Event

ohne Parameter



- Substates/Unterzustände: verschachtelte Struktur
  - sequentiell aktive
  - gleichzeitig aktive

Beispiel *Heizung*

vertiefen wir hier aber nicht

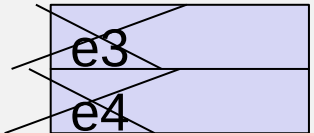
- interne Transitionen (ohne Wechsel des Zustandes)
  durch externes Ereignis unterbrechbar, d.h. per Signal-, Call-, Time-, State-Event
  semantischer Unterschied zu externen reflexiven Transitionen
- defer: Verzögerung von Ereignissen
  Liste von Signal- und Call-Request-Events,
  die in diesem Zustand nicht behandelt werden und trotzdem nicht verloren gehen sollen

Aktivitäten:= als Folge Aktionen bei Auszeichnung von elementaren nicht unterbrechbaren Aktionen → **Problem**: fehlende Action-Sprache

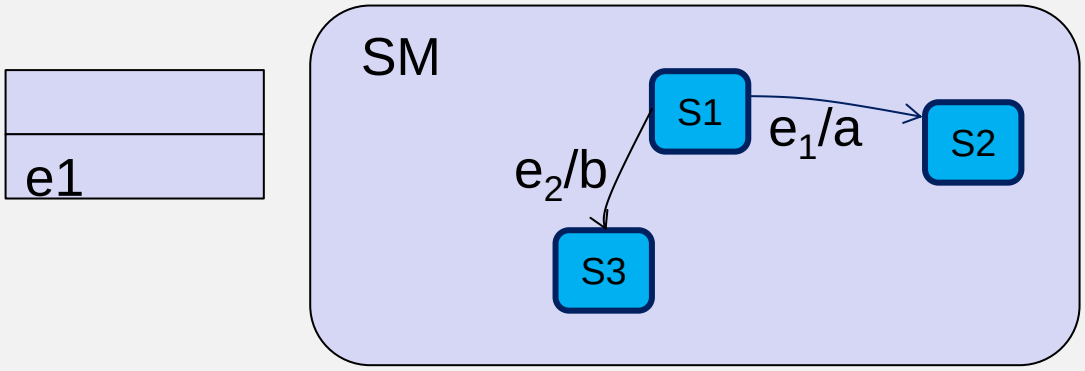
Aktionsreihenfolge beim Zustandsübergang

Ann.:

- Zustandsmaschine verharrt im Zustand S1
- bei Laufen einer do-Aktivität x und
- Eintreffen von e_1

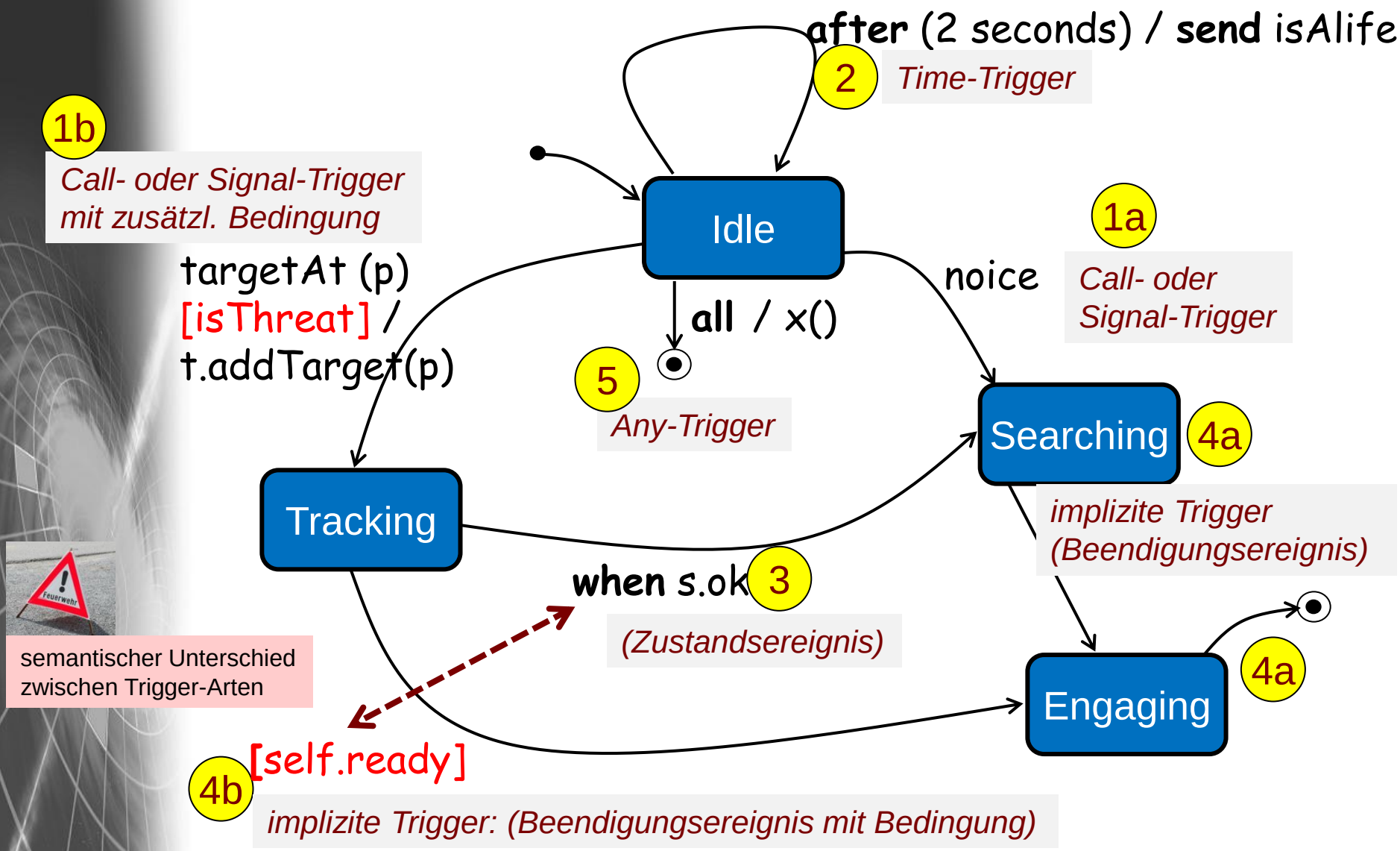


WICHTIG: Ereignisse, die im aktuellen Zustand keine Trigger darstellen, werden verworfen !



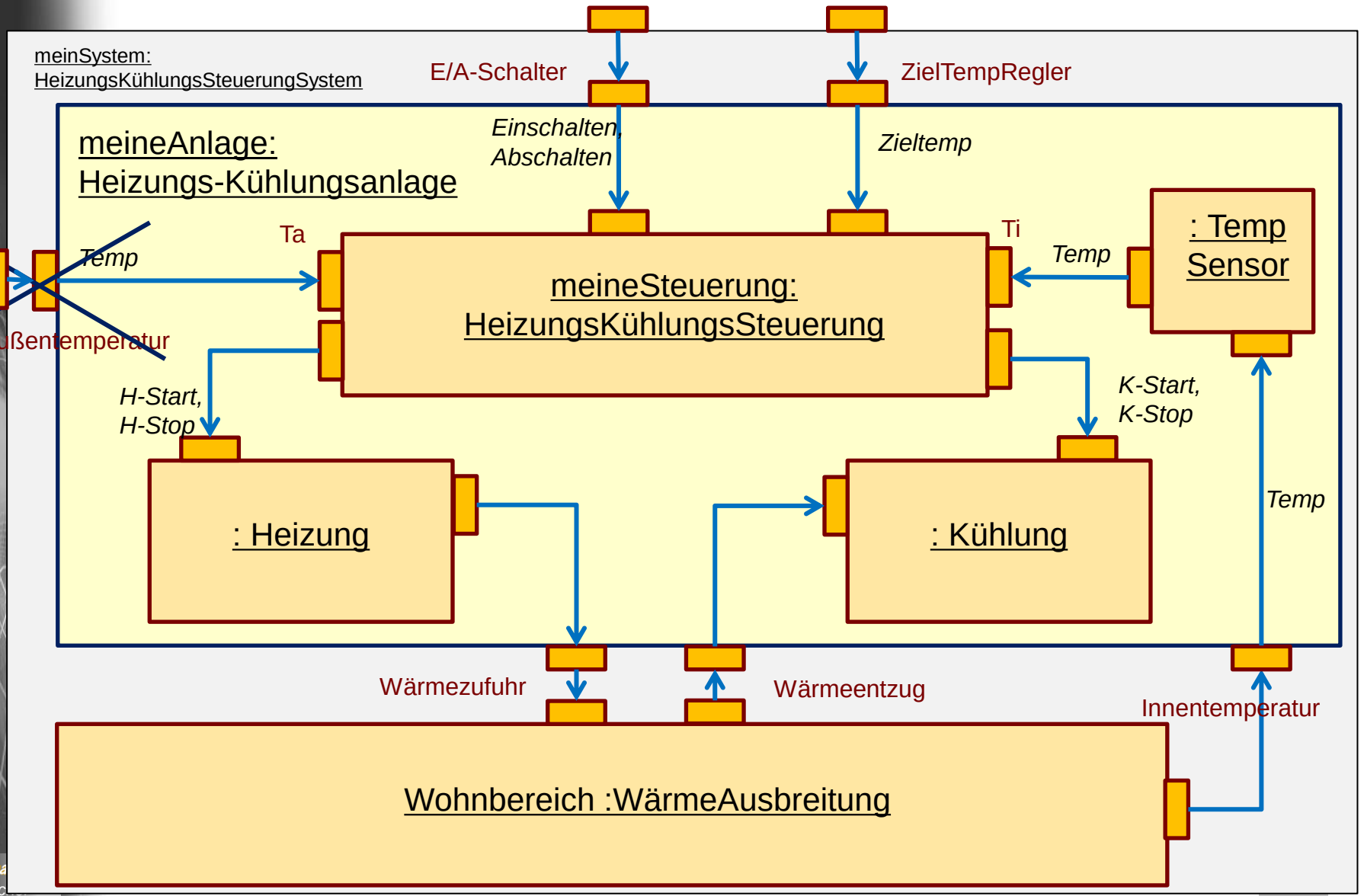
(0) Entfernen von e1 aus dem Empfangspool der SM-Instanz
(1) Unterbrechung von x
(2) exit-Op von S1
(3) Aktion a
(4) Zustandswechsel der SM-Instanz: S1 → S2
(5) entry-Op von S2
(6) Start der evtl. do-Aktivität von S2

Verschiedene Trigger-Arten

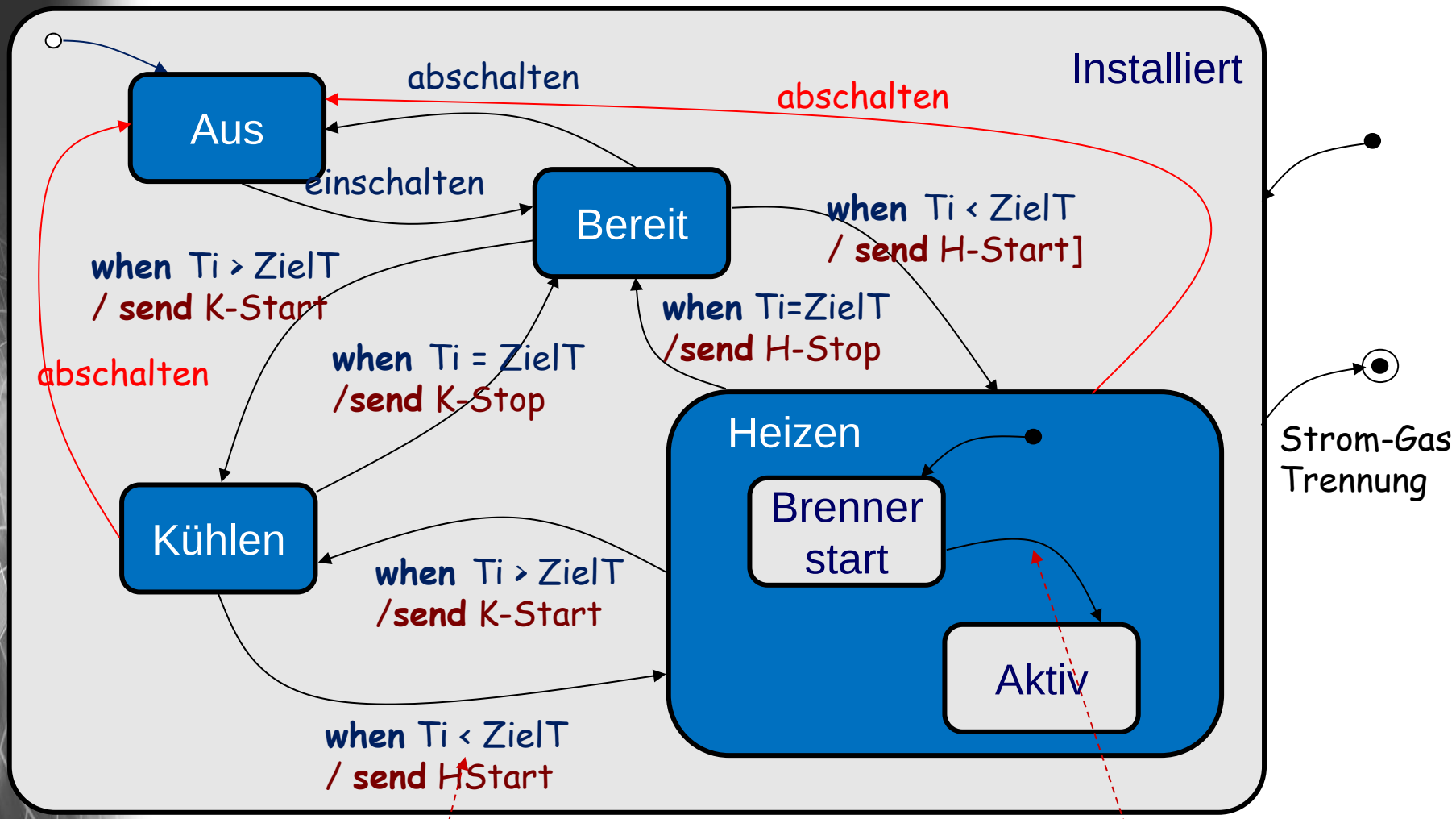


semantischer Unterschied
zwischen Trigger-Arten

Installierte Heizungs-Kühlungsanlage (Systemkonfiguration)



HeizungsKühlungsSteuerung (Zustandsautomat)



- Ereignisse lösen Transitionen aus, wenn dafür Trigger vorhanden sind
- Trigger können explizit angegeben sein, oder sie ergeben sich implizit

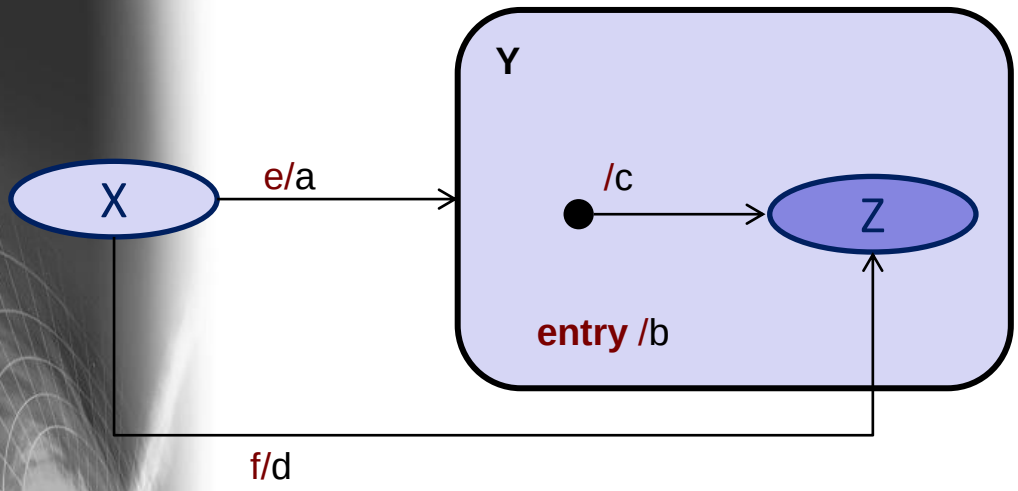
Besonderheiten interner Zustandsaktivitäten

- entry- und exit-Aktivitäten dürfen
  - keine Parameter besitzen
    Ausnahme: Entry-Aktivität für ersten äußersten Zustand
    (Parameter des Konstrukturs des zugehörigen Classifiers)
  - Keine Randbedingungen besitzen

- interne Transitionen können
  - über Ereignisse mit Parametern und Randbedingungen verfügen
  - Unterschied zu einer entsprechenden externen Transition:
    keine Ausführung der entry- und exit-Aktivitäten

- befindet sich ein Objekt in einem Zustand,
  - dann befindet sich das Objekt i.allg. im „Leerlauf“
    (wartet lediglich auf externes Ereignis)
  - es kann aber auch eine Aktivität laufen mit
    a) endlicher oder
    b) unendlicher Laufzeit

Zustandsübergänge

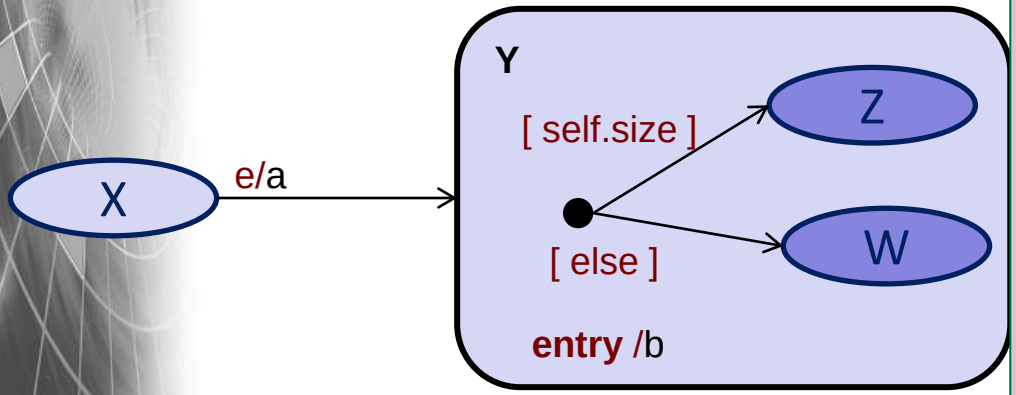


Verharren in X

(1) bei Eintritt von e feuert die Transition
- Aktion a wird ausgeführt
- Übergang nach Y

• Eintrittsaktion b wird ausgeführt
  Startzustand von Y wird aktiv
• Übergang nach Z wird unmittelbar ausgeführt
• - Ausführung der Aktion c
  - Übergang in Z (bleiben auch Y)

(2) Bei Eintritt von f feuert die andere Transition bei Ausführung von d und b (ohne c)

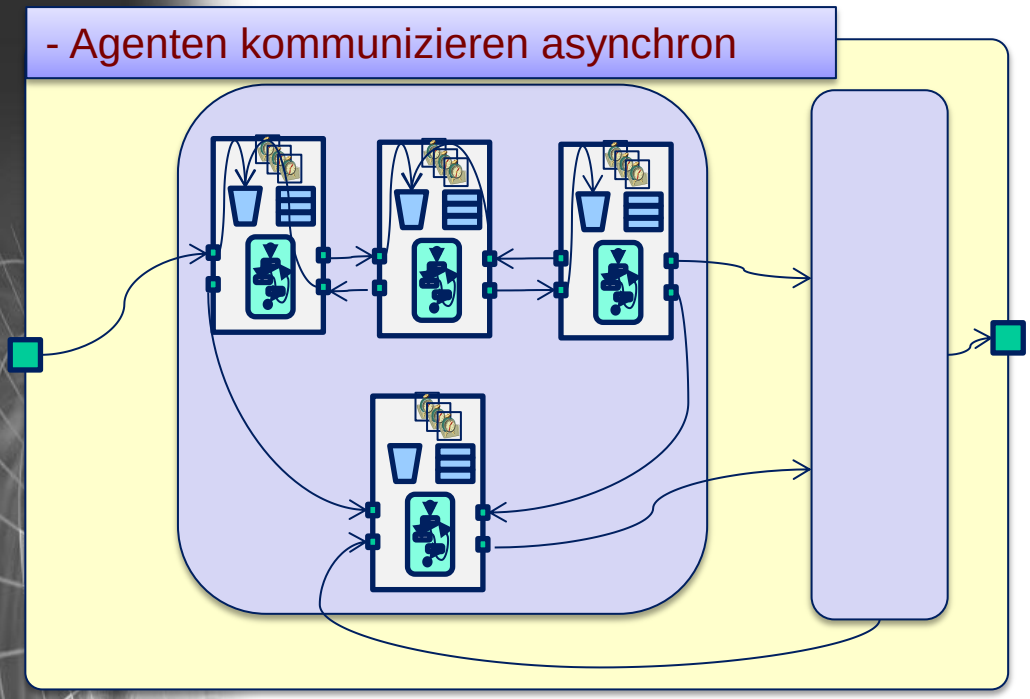


Verharren in X

bei Eintritt von e feuert die Transition
- Aktion a wird ausgeführt
- Übergang nach Y

• Eintrittsaktion b wird ausgeführt
  Startzustand von Y wird aktiv
• Übergang nach Z oder W in Abh. Vom Wert von size als Attribut des zugehörigen Objekts

Herausforderung: System als Agent-Ensemble



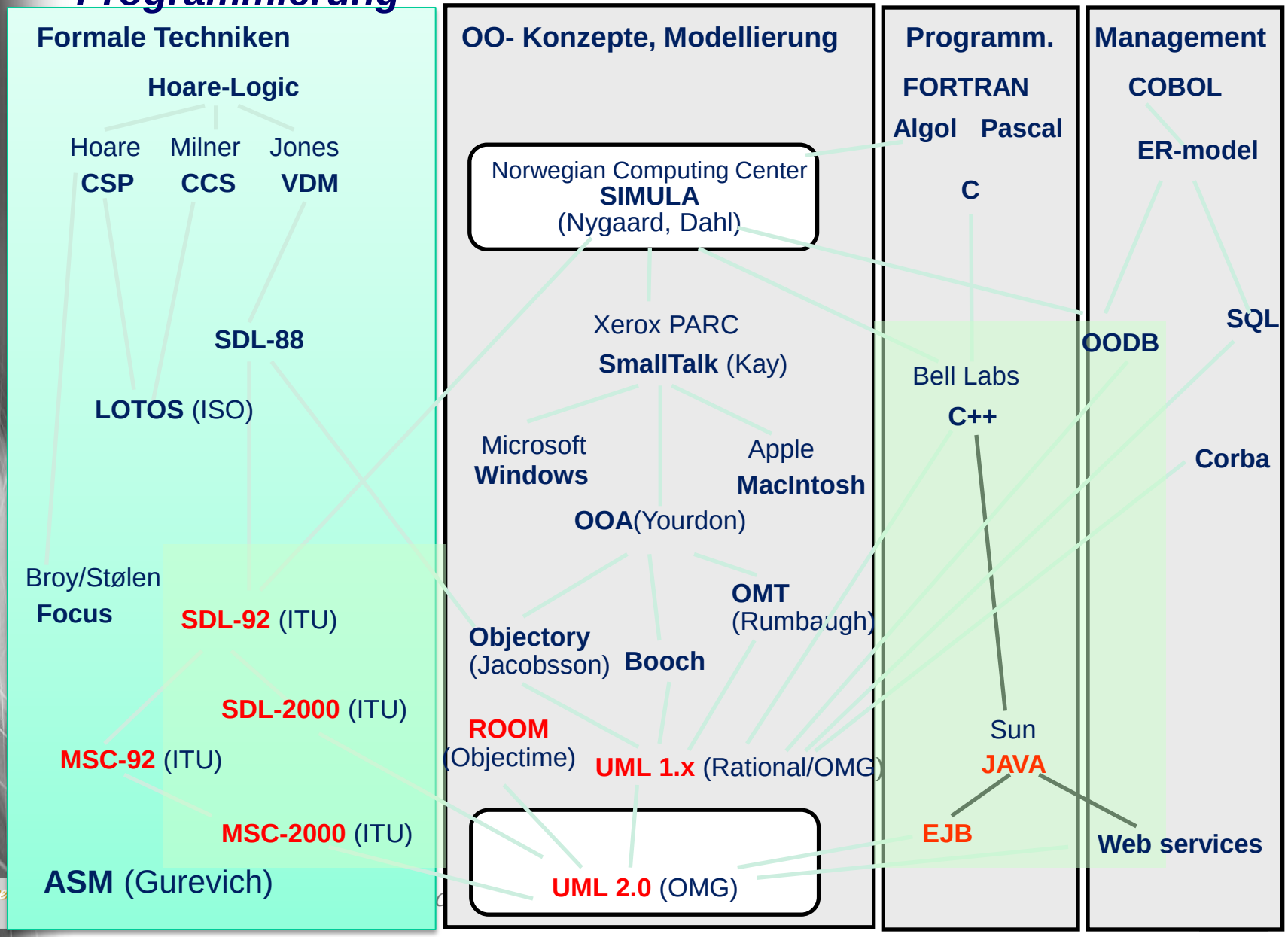
UML-Probleme

- Pool-bearbeitung:
  sog. semantischer Variationspunkt
  (Reihenfolge, …)
- Adressierung von Signalen
  (nicht komplett gelöst,
  Bekanntmachung der Agenten/Ports)
- Übertragung von Signalen
  (ungelöst: Zeitverbrauch, Sicherheit)
- semantischer Variationspunkt
  Offenheit bzgl. Action-Sprache
  Datentypen, Anweisungen
- Do-Aktiviät kann sowohl zeitdiskret als auch zeitkontinuierlich sein

ODEMx: als geeignete Basis für eine Agent-Implementierung

Klasse Process ~ Basis für UML/SDL-Agent

Wurzeln der Objektorientierten Modellierung/ Programmierung



Wurzeln der Objektorientierten Modellierung/ Programmierung

Formale Techniken

Hoare-Logic

Hoare Milner Jones

CSP **CCS** **VDM**

OO- Konzepte, Modellierung

Programm.

FORTRAN

Algol **Pascal**

C

Management

COBOL

ER-model

Norwegian Computing Center
SIMULA
(Nygaard, Dahl)

Xerox PARC
SmallTalk (Kay)

Apple
MacInt

OOA(Yourdon)

OMT
(Rumbaugh)

Booch

UML 1.x (Rational/OMG)

UML 2.0 (OMG)

Sun
JAVA

EJB

Web services

MSC-2000 (ITU)

ASM (Gurevich)

Konzeptuelle Basis von OO:
- Klassen mit Vererbung,
- Objektreferenzen,
- Polymorphie, Virtualität,
- Co-Routinen,
- Garbage Collection
- Standardklassenbibliothek

Simula als erste OO-Sprache
- Implementationssprache
- Modellierungssprache
- Byte-Code

Wurzeln der Objektorientierten Modellierung/ Programmierung

Formale Techniken

Hoare-Logic

Hoare Milner Jones

CSP **CCS** **VDM**

SDL-88

LOTOS (ISO)

Broy/Stølen

Focus

SDL-92 (ITU)

SDL-2000 (ITU)

MSC-92 (ITU)

MSC-2000 (ITU)

ASM (Gurevich)

OO- Konzepte, Modellierung

Norwegian Computing C
SIMULA
(Nygaard, Dahl)

Xerox PARC
SmallTalk (Kay)

Microsoft
Windows

Apple
MacIntosh

OOA(Yourdon)

OMT
(Rumbaugh)

Objectory
(Jacobsson) **Booch**

ROOM
(Objectime)

UML 1.x (Rational/OMG)

UML 2.0 (OMG)

Programm.

effiziente Programmausführung
(explizite Speicherverwaltung)
C with Classes

Bell Labs
++

Management

SQL

OODB

Corba

EJB

Web services