

Informatik I: Einführung in die Programmierung

19. Automaten, Zustandsmodellierung mit abstrakten und generischen Klassen

Albert-Ludwigs-Universität Freiburg



**UNI
FREIBURG**

Prof. Dr. Peter Thiemann

22.01.2025

1 Deterministische endliche Automaten



- Motivierendes Beispiel
- Formale Grundlagen
- Verhalten eines Automaten
- Teilwort-Erkennung

Deterministische
endliche
Automaten

Motivierendes
Beispiel
Formale
Grundlagen
Verhalten eines
Automaten
Teilwort-Erkennung

Transduktoren

Modellierung
von
Automaten

Welt &
Modell

Zusammenfassung &
Ausblick

Vor kurzem fand ich auf dem Dachboden eines Kollegen einen Würfel und einen Brief.



Lieber Finder dieses Briefes,

der Würfel ist gefallen:

5-1-5-6 ergibt x,

5-4-2-5 ergibt y.

*Bei N 48° 00, x' O 7° 50, y' in
15 Meter Tiefe wirst du
einen sagenhaften Schatz
finden, der das
Bernsteinzimmer wie eine
Studentenbude aussehen
läßt. Ich wünsche Dir viel
Glück bei deiner Suche !*

Emil Nebel

Freiburg, 1980

Deterministische
endliche
Automaten

Motivierendes
Beispiel

Formale
Grundlagen

Verhalten eines
Automaten

Teilwort-Erkennung

Transduktoren

Modellierung
von
Automaten

Welt &
Modell

Zusammenfassung &
Ausblick

- In dem Würfel gibt es einen Mechanismus, der die **Abfolge** der nach oben gerichteten Würfelseiten **erkennt**.
- Nachdem die richtige Folge „gewürfelt“ wurde, klopft von innen ein Hämmerchen die Koordinaten.
- Wie erkennt der Würfel solche Folgen von Ereignissen?
- Hierfür ist ein **endlicher Automat** geeignet.
- Ein endlicher Automat ist ein sehr einfaches und eingeschränktes **Berechnungsmodell**, das für viele Anwendungen adäquat ist.
- Wir können ihn durch eine Klasse definieren.

Deterministische
endliche
Automaten

Motivierendes
Beispiel

Formale
Grundlagen

Verhalten eines
Automaten

Teilwort-Erkennung

Transduktoren

Modellierung
von
Automaten

Welt &
Modell

Zusammenfassung &
Ausblick

Ein **deterministischer Automat** ist ein Quintupel (Q, E, δ, q_0, F)

```
@dataclass(frozen=True)
class Automaton[Q,
                E]:
    delta : Callable[[Q, str], Q] # Transitionsfunktion
    start : Q                    # Startzustand q0
    finals : frozenset[Q]       # Menge von Endzuständen F

    def accept (self, input: Iterable[E]) -> bool:
        state = self.start
        for c in input:
            state = self.delta(state, c)
        return state in self.finals
```

Die `accept` Methode nimmt ein Eingabewort, lässt den Automaten vom Startzustand bis zum Ende des Worts laufen. Das Wort wird akzeptiert, wenn der Automat einen Endzustand erreicht.

Deterministische
endliche
Automaten

Motivierendes
Beispiel

Formale
Grundlagen

Verhalten eines
Automaten

Teilwort-Erkennung

Transduktionen

Modellierung
von
Automaten

Welt &
Modell

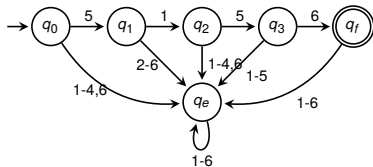
Zusammenfassung &
Ausblick

Die Transitionsfunktion wird entweder durch eine **Transitionstabelle** oder durch ein **Transitionsdiagramm** (Transitionsdiagramm) angegeben.

Beispiel: Transitionsfunktion zur Erkennung von 5156

(q_e bezeichnet einen Fehlerzustand und $F = \{q_f\}$).

	1	2	3	4	5	6
q_0	q_e	q_e	q_e	q_e	q_1	q_e
q_1	q_2	q_e	q_e	q_e	q_e	q_e
q_2	q_e	q_e	q_e	q_e	q_3	q_e
q_3	q_e	q_e	q_e	q_e	q_e	q_f
q_f	q_e	q_e	q_e	q_e	q_e	q_e
q_e	q_e	q_e	q_e	q_e	q_e	q_e



Beachte: Im Transitionsdiagramm wird der **absorbierende Fehlerzustand** q_e mit allen Übergängen dorthin in der Regel nicht angegeben.

Deterministische
endliche
Automaten

Motivierendes
Beispiel

Formale
Grundlagen

Verhalten eines
Automaten

Teilwort-Erkennung

Transduktoren

Modellierung
von
Automaten

Welt &
Modell

Zusammenfassung &
Ausblick

Implementierung der Transitionsfunktion



UNI
FREIBURG

```
class State (Enum):
    Q0 = auto(); Q1 = auto(); Q2 = auto(); Q3 = auto()
    Qf = auto(); Qe = auto()
type Alphabet = Literal['1','2','3','4','5','6']
def delta_dice (q:State, c:Alphabet) -> State:
    match (q, c):
        case (State.Q0, '5'):
            return State.Q1
        case (State.Q1, '1'):
            return State.Q2
        case (State.Q2, '5'):
            return State.Q3
        case (State.Q3, '6'):
            return State.Qf
        case _:
            return State.Qe
```

Deterministische
endliche
Automaten

Motivierendes
Beispiel

Formale
Grundlagen

Verhalten eines
Automaten

Teilwort-Erkennung

Transduktoren

Modellierung
von
Automaten

Welt &
Modell

Zusammenfassung &
Ausblick

Der Typ `Literal['1','2','3','4','5','6']` bestimmt das

Eingabealphabet!

- Anfänglich befindet sich der Automat im **Startzustand** q_0 .
- Der Automat erhält ein Wort $w = "a_1a_2 \dots a_n"$ als **Eingabe** (darf auch leer sein, d.h. $n \geq 0$).
- Der Automat liest im Zustand q_i das **Eingabesymbol** a_{i+1} und wechselt in den **Folgezustand** $q_{i+1} = \delta(q_i, a_{i+1})$.
- Das macht der Automat, so lange Eingabezeichen gelesen werden können, das heißt für $i \in \{0, 1, \dots, n-1\}$.
- Am Ende der Eingabe befindet sich der Automat in einem Zustand q_n . Das Eingabewort w wird genau dann **akzeptiert**, wenn $q_n \in F$ ein **Endzustand** ist.
- Die **Sprache von A**, $\mathcal{L}(A)$, ist die Menge aller von A akzeptierten Worte.

Deterministische
endliche
Automaten

Motivierendes
Beispiel

Formale
Grundlagen

Verhalten eines
Automaten

Teilwort-Erkennung

Transduktoren

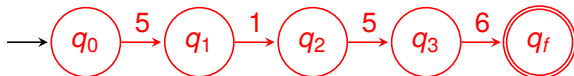
Modellierung
von
Automaten

Welt &
Modell

Zusammenfassung &
Ausblick

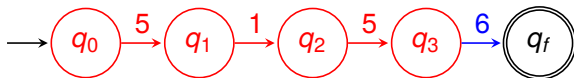
Eingabe: 5156 5156 156 156 56 56 6 6

Eingabe akzeptiert



Eingabe: 515156 515156 15156 15156 5156 5156 156 156
156

Kein Transition von q_3 aus möglich! Eingabe nicht akzeptiert.



Deterministische
endliche
Automaten

Motivierendes
Beispiel
Formale
Grundlagen

Verhalten eines
Automaten

Teilwort-Erkennung

Transduktoren

Modellierung
von
Automaten

Welt &
Modell

Zusammenfassung &
Ausblick

```
dice : Automaton[State, Alphabet] = Automaton(  
    delta = delta_dice,  
    start = State.Q0,  
    finals = frozenset ({State.Qf})  
)  
print(dice.accept("5156"))
```

Ausgabe: True

```
print(dice.accept("5155"))
```

Ausgabe: False

Deterministische
endliche
Automaten

Motivierendes
Beispiel
Formale
Grundlagen

Verhalten eines
Automaten

Teilwort-Erkennung

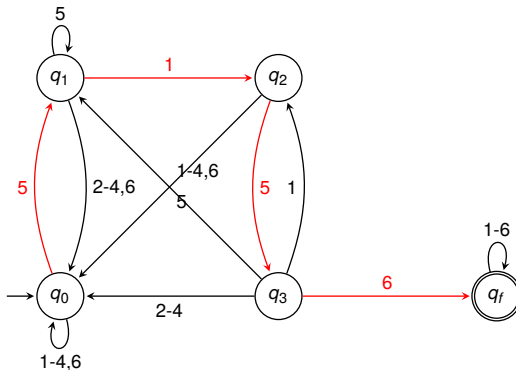
Transduktoren

Modellierung
von
Automaten

Welt &
Modell

Zusammenfassung &
Ausblick

Der Würfel soll letztlich alle Folgen akzeptieren, die 5156 als **Teilwort** enthalten, z.B. auch 55156, oder 5155156, oder 515156 oder ...5156...



Deterministische
endliche
Automaten

Motivierendes
Beispiel
Formale
Grundlagen
Verhalten eines
Automaten

Teilwort-Erkennung

Transduktoren

Modellierung
von
Automaten

Welt &
Modell

Zusammenfassung &
Ausblick

Zusammenfassung & Ausblick



- Moore-Automat
- Umsetzung
- Python-Skript für Beispiel

Deterministische
endliche
Automaten

Transduktoren

Moore-Automat
Umsetzung
Python-Skript für
Beispiel

Modellierung
von
Automaten

Welt &
Modell

Zusammenfassung &
Ausblick

Nach der Akzeptanz ist vor der Akzeptanz!



- Der bisher konstruierte Automat akzeptiert alle Wörter, die 5156 oder 5425 als Teilwort enthalten.
- Eigentlich ist eine Maschine gesucht, die „ewig“ läuft und die jeweils nach einem akzeptierten Teilwort eine **Ausgabe** macht.
- Eine solche Maschine heißt **Transduktor**, das ist ein Automat, der auch Ausgaben macht.
- Transduktoren werden oft verwendet um das Verhalten **eingebetteter Systeme** zu beschreiben.

Deterministische
endliche
Automaten

Transduktoren

Moore-Automat

Umsetzung

Python-Skript für
Beispiel

Modellierung
von
Automaten

Welt &
Modell

Zusammenfassung &
Ausblick

```
@dataclass(frozen=True)
class MooreAutomaton[Q,           # Menge von Zuständen
                    E,           # Eingabealphabet
                    A]:          # Ausgabealphabet
    dl : Callable[[Q, E], Q]      # Transitionsfunktion
    lm : Callable[[Q], A]         # Ausgabefunktion
    st : Q                       # Startzustand

    def translate (self, input: Iterable[E]) -> Iterator[A]:
        state = self.st
        yield self.lm(state)
        for c in input:
            state = self.dl(state, c)
            yield self.lm(state)
        return
```

Deterministische
endliche
Automaten

Transduktoren

Moore-Automat

Umsetzung

Python-Skript für
Beispiel

Modellierung
von
Automaten

Welt &
Modell

Zusammenfassung &
Ausblick

- Im **Startzustand** q_0 gibt der Automat $\lambda(q_0)$ aus.
- Der Automat erhält ein Wort $w = "a_1 a_2 \dots"$ als **Eingabe**.
- Der Automat liest im Zustand q_i das **Eingabesymbol** a_{i+1} , wechselt in den **Folgezustand** $q_{i+1} = \delta(q_i, a_{i+1})$ und gibt dann $\lambda(q_{i+1})$ aus.
- Das macht der Automat, so lange wie Eingabezeichen verfügbar sind. (Ggf. mit offenem Ende.)
- Dabei wird das Eingabewort w in das Ausgabewort $\lambda(q_0)\lambda(q_1)\dots$ **übersetzt**. Daher auch der Name Transduktor.

Deterministische
endliche
Automaten

Transduktoren

Moore-Automat

Umsetzung

Python-Skript für
Beispiel

Modellierung
von
Automaten

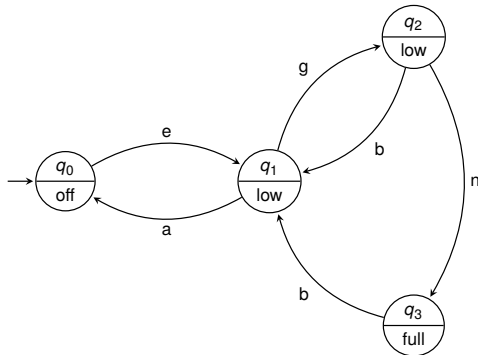
Welt &
Modell

Zusammenfassung &
Ausblick

Beispiel: Ein hypothetische Motorsteuerung



$E = \{e, a, g, b, n\}$, wobei e für „ein“, a für „aus“, g für „Gas geben“, b für „bremsen“, n für „nicht drehende Räder“ steht.
 $A = \{\text{off}, \text{low}, \text{full}\}$.



Deterministische
endliche
Automaten

Transduktoren

Moore-Automat

Umsetzung

Python-Skript für
Beispiel

Modellierung
von
Automaten

Welt &
Modell

Zusammenfassung &
Ausblick

```
class MState (Enum):
    Q0 = auto(); Q1 = auto(); Q2 = auto(); Q3 = auto()
type Input = Literal['e', 'g', 'a', 'b', 'n']
type Output = Literal['off', 'low', 'full']

def delta_m (q : MState, c : Input) -> MState:
    match (q, c):
        case (MState.Q0, "e"): return MState.Q1
        case (MState.Q1, "g"): return MState.Q2
        case (MState.Q1, "a"): return MState.Q0
        case (MState.Q2, "b"): return MState.Q1
        case (MState.Q2, "n"): return MState.Q3
        case (MState.Q3, "b"): return MState.Q1
        case _: return q

def lambda_m (q : MState) -> Output:
    match q:
        case MState.Q0: return "off"
        case MState.Q1 | MState.Q2: return "low"
        case MState.Q3: return "full"
```

Deterministische
endliche
Automaten

Transduktoren

Moore-Automat

Umsetzung

Python-Skript für
Beispiel

Modellierung
von
Automaten

Welt &
Modell

Zusammenfassung &
Ausblick

Implementierung der Motorsteuerung (2)



```
mcontrol : MooreAutomaton[MState, Input, Output] = MooreAutomaton(  
    dl= delta_m,  
    lm= lambda_m,  
    st= MState.Q0  
)
```

Ein Testlauf:

```
output = mcontrol.translate ("egbnba")  
print (list (output))
```

Ausgabe:

```
['off', 'low', 'low', 'low', 'low', 'full', 'low', 'off']
```

Deterministi-
sche
endliche
Automaten

Transduktio-
nen

Moore-Automat

Umsetzung

Python-Skript für
Beispiel

Modellierung
von
Automaten

Welt &
Modell

Zusammen-
fassung &
Ausblick

Deterministische endliche Automaten

Moore-Automat

Umsetzung
Python-Skript für
Beispiel

Welt & Modell

Zusammenfassung & Ausblick



- Wie sieht die **Implementierung** eines solchen abstrakten Automaten aus?
- Das Innere des Würfels:



- Batterien (4×AA-Akkus, also 4,5-6 Volt),
- Servomotor,
- pyboard (mit einem ARM-5 Prozessor, Beschleunigungsmesser, usw.), auf dem Micropython läuft

Deterministische
endliche
Automaten

Transduktoren

Moore-Automat

Umsetzung

Python-Skript für
Beispiel

Modellierung
von
Automaten

Welt &
Modell

Zusammenfassung &
Ausblick

- `side_up()`:
Bestimmt mit Hilfe des Beschleunigungsmessers, welche Seite oben liegt. Bei unklaren Werten wartet die Funktion, bis eine stabile Lage eingetreten ist.
- `new_input()`:
Erzeugt ein neues Eingabesymbol (Zahl zwischen 1 und 6), wenn der Würfel 500 Millisekunden stabil lag.
- `next_state(state, input)`:
Die Transitionsfunktion.
- `output_symbol(state)`:
Berechnet das zum Zustand gehörige Ausgabesymbol.
- `automaton()`:
Die Endlosschleife zur Ausführung des Automaten.
- `code_knock(code)`:
Klopft entsprechend dem angeforderten Code.

Deterministische
endliche
Automaten

Transduktoren

Moore-Automat
Umsetzung
Python-Skript für
Beispiel

Modellierung
von
Automaten

Welt &
Modell

Zusammenfassung &
Ausblick

Der Seitenerkenner mittels Beschleunigungssensor



Die Erdbeschleunigung von 1g entspricht einem Messwert von rund 20.

Seitenerkenner

```
thres = 12
def side_up():
    while True:
        x = acc.x(); y = acc.y(); z = acc.z()
        if x > thres: return 5 #x up
        if x < -thres: return 2 #x down
        if y > thres: return 6 #y up
        if y < -thres: return 1 #y down
        if z > thres: return 3 #z up
        if z < -thres: return 4 #z down
        # no stable situation yet
```

Deterministische
endliche
Automaten

Transduktoren

Moore-Automat

Umsetzung

Python-Skript für
Beispiel

Modellierung
von
Automaten

Welt &
Modell

Zusammenfassung &
Ausblick

Symbolerzeugung

```
def new_input():  
    while True:  
        curr = side_up()  
        new = curr  
        start = pyb.millis()  
        while (curr == new and  
               pyb.elapsed_millis(start) <= 500):  
            new = side_up()  
        if curr == new:  
            return curr
```

Erzeugt ca. alle 0,5 Sekunden ein neues Eingabesymbol.
Nicht nur, wenn die Seite gewechselt wird. Daher muss der
Automat etwas anders aussehen!

Deterministi-
sche
endliche
Automaten

Transdukto-
ren

Moore-Automat

Umsetzung

Python-Skript für
Beispiel

Modellierung
von
Automaten

Welt &
Modell

Zusammen-
fassung &
Ausblick

Transitionsfunktion

```
def next_state(state, input):  
    if state == 0: # initial state  
        if input == 5: return 1  
        return 0  
    elif state == 1: # '5' read  
        if input == 5: return 1  
        if input == 1: return 2  
        if input == 4: return 4  
        return 0  
    elif state == 2: # '51' read  
        if input == 1: return 2 # repetition!  
        if input == 5: return 3  
        return 0  
    elif ...
```

Beachte: Jeder Zustand hat eine Schleife für das Zeichen, das dafür notwendig war, in den Zustand zu kommen.

Deterministische
endliche
Automaten

Transduktoren

Moore-Automat

Umsetzung

Python-Skript für
Beispiel

Modellierung
von
Automaten

Welt &
Modell

Zusammenfassung &
Ausblick

Der Automat & die Ausgabefunktion

```
def automaton():  
    state = 0  
    while True:  
        if sw(): return # if switch is pressed, exit  
        state = next_state(state, new_input())  
        code_knock(output_symbol(state))  
  
def output_symbol(state):  
    if state == 10:  
        return "north"  
    elif state == 11:  
        return "east"  
    else:  
        return None
```

Deterministische
endliche
Automaten

Transduktoren

Moore-Automat

Umsetzung

Python-Skript für
Beispiel

Modellierung
von
Automaten

Welt &
Modell

Zusammenfassung &
Ausblick

3 Modellierung von Automaten



Deterministische
endliche
Automaten

Transduktoren

**Modellierung
von
Automaten**

Welt &
Modell

Zusammenfassung &
Ausblick

- Im Vergleich zu Standard-Python hat Micropython einige Einschränkungen.
- Die Modellierung der Zustände durch Zahlen ist kompakt und effizient, aber fehleranfällig.
- Besserer Ansatz: Verwende Aufzählungstypen.

Deterministische
endliche
Automaten

Transduktoren

Modellierung
von
Automaten

Welt &
Modell

Zusammenfassung &
Ausblick

```
from enum import Enum, auto
```

```
class State(Enum):  
    INIT = auto()  
    AFTER_5 = auto()  
    AFTER_51 = auto()  
    AFTER_515 = auto()  
    AFTER_5156 = auto()  
    AFTER_54 = auto()  
    AFTER_542 = auto()  
    AFTER_5425 = auto()
```

- Bessere Dokumentation für die Zustände.
- `enum.auto()` erzeugt automatisch einen neuen internen Code, sodass die Zustände durchnummeriert werden.

Deterministische
endliche
Automaten

Transduktoren

Modellierung
von
Automaten

Welt &
Modell

Zusammenfassung &
Ausblick

S = State

```
def next_state(q: State, i: int) -> State:
    match (q, i):
        case (S.INIT,      5): return S.AFTER_5
        case (S.AFTER_5,   1): return S.AFTER_51
        case (S.AFTER_5,   4): return S.AFTER_54
        case (S.AFTER_5,   5): return S.AFTER_5
        case (S.AFTER_51,  5): return S.AFTER_515
        case (S.AFTER_515, 6): return S.AFTER_5156
        case (S.AFTER_515, 1): return S.AFTER_51
        case (S.AFTER_515, 4): return S.AFTER_54
        case (S.AFTER_54,  2): return S.AFTER_542
        case (S.AFTER_542, 5): return S.AFTER_5425
    return S.INIT
```

■ Hier ist ein Fehler!

Deterministische
endliche
Automaten

Transduktoren

Modellierung
von
Automaten

Welt &
Modell

Zusammenfassung &
Ausblick

- Falls die Zustände selbst noch weitere Information beinhalten, können sie als “richtige” Objekte betrachtet werden.
- Die Zustandsfunktion wird nun als Methode implementiert.
- So ist auch die Menge der Zustände einfach erweiterbar.
- Wegen des Speicherbedarfs allerdings für eingebettete Systeme weniger geeignet.
- Das Zustandsmuster ist ein sogenanntes **design pattern**.
- Siehe *Design Patterns: Elements of Reusable Object-Oriented Software* von der “Gang of Four” (Gamma, Helm, Johnson, Vlissides).

Deterministische
endliche
Automaten

Transduktoren

Modellierung
von
Automaten

Welt &
Modell

Zusammenfassung &
Ausblick

```
# base class for all states
@dataclass
class State:
    def next(self, input: int) -> 'State':
        return self
    def output(self) -> str:
        return ""
```

- Die Basisklasse State dient nur als Superklasse, von der konkrete Zustandsklassen erben.
- Sie soll nicht instanziiert werden, d.h. State() soll nicht verwendet werden.
- Auch die Methoden sollen eigentlich nicht verwendet werden, sondern nur anzeigen, was in den Subklassen implementiert werden soll.
- Eine solche Klasse heißt **abstrakte Klasse** und die Methoden **abstrakte Methoden**.

Deterministische
endliche
Automaten

Transduktoren

Modellierung
von
Automaten

Welt &
Modell

Zusammenfassung &
Ausblick


```
@dataclass
class S_Init(State):
    def next(self, input: int) -> State:
        match input:
            case 5:
                return S_After([5])
        return self
```

- Subklasse von State.
- Methode `next()` wird überschrieben.
- Methode `output()` wird aus der Superklasse übernommen (keine Ausgabe).

Deterministische
endliche
Automaten

Transduktoren

Modellierung
von
Automaten

Welt &
Modell

Zusammenfassung &
Ausblick

```
@dataclass
class S_After(State):
    prefix: list[int]
    def next(self, input: int) -> State:
        self.prefix = self.prefix + [input]
        match self.prefix:
            case ([5, 1] | [5, 1, 5] | [5, 1, 5, 6] |
                  [5, 4] | [5, 4, 2] | [5, 4, 2, 5]):
                return self
        match self.prefix[-1:]:
            case [5]:
                return S_After(self.prefix[-1:])
        match self.prefix[-2:]:
            case [5, 1] | [5, 4]:
                return S_After(self.prefix[-2:])
        return S_Init()
```

Deterministische
endliche
Automaten

Transduktoren

Modellierung
von
Automaten

Welt &
Modell

Zusammenfassung &
Ausblick

```
def output(self) -> str:
    match self.prefix:
        case [5, 1, 5, 6]:
            return "north"
        case [5, 4, 2, 5]:
            return "east"
        case _:
            return ""
```

- Die Klasse `S_After` verwaltet das bisher gelesene Präfix der Eingabe im Feld `prefix`.
- Zustandsübergang durch Änderung des Präfixes (soweit möglich).

Deterministische
endliche
Automaten

Transduktoren

Modellierung
von
Automaten

Welt &
Modell

Zusammenfassung &
Ausblick

4 Welt & Modell



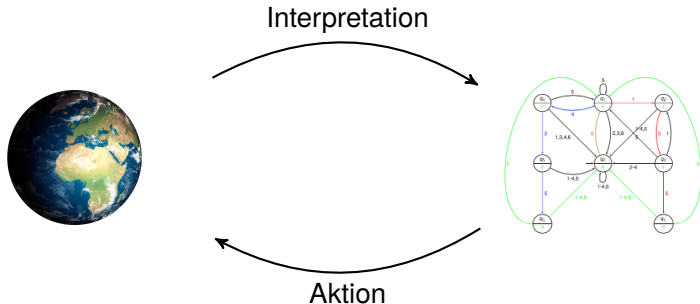
Deterministische
endliche
Automaten

Transduktoren

Modellierung
von
Automaten

**Welt &
Modell**

Zusammenfassung &
Ausblick



Deterministische
endliche
Automaten

Transduktoren

Modellierung
von
Automaten

Welt &
Modell

Zusammenfassung &
Ausblick

Vor dem Einsatz **formaler Modelle** (wie Moore-Automaten) müssen die Messwerte/Eingaben **interpretiert** und in **Symbole** umgesetzt werden. Die **Interpretation** und das **Modell** beeinflussen sich dabei gegenseitig (Beispiel: Würfelseitenerkennung und Automat).
⇒ Siehe Vorlesung Embedded Systems.

5 Zusammenfassung & Ausblick



Deterministische
endliche
Automaten

Transduktoren

Modellierung
von
Automaten

Welt &
Modell

**Zusammen-
fassung &
Ausblick**

- Eine **formale Sprache** ist eine Menge von Wörtern.
- Automaten sind ein einfaches **Berechnungsmodell**.
- Sie können Sprachen erkennen (Akzeptoren) und übersetzen (Transduktoren).
- **Deterministische endliche Automaten (DEAs)** und **Transduktoren** (Moore-Automaten) werden zur Beschreibung von eingebetteten Systemen verwendet.
- Einfache Implementierung.

Deterministische
endliche
Automaten

Transduktoren

Modellierung
von
Automaten

Welt &
Modell

Zusammenfassung &
Ausblick