Grundlagen der Informatik

Zustandsautomaten

Prof. Dr. Peter Jüttner

Zustandsautomaten

Zustandsautomaten

Verhalten eines Getränkeautomaten ...

Verhalten des Automaten	Aktion des Kunden	Verhalten des Automaten

Zustandsautomaten

Verhalten eines Getränkeautomaten ...

Verhalten des Automaten	Aktion des Kunden	Verhalten des Automaten
Anzeige „Bitte Geld einwerfen“	Geld einwerfen	Ausgabe „Getränk wählen“ oder eingeworfenen Betrag anzeigen
Anzeige „Bitte Geld einwerfen“	Getränk wählen	Ausgabe „Bitte Geld einwerfen“
Anzeige „Bitte Geld einwerfen“	Rückgabeknopf drücken	Anzeige „Bitte Geld einwerfen“
Anzeige „Getränk wählen“	Getränk wählen	Getränk wird ausgegeben und Anzeige „Bitte Getränk entnehmen“
Anzeige „Getränk wählen	Rückgabeknopf drücken	Eingeworfenes Geld wird zurückgegeben

Zustandsautomaten

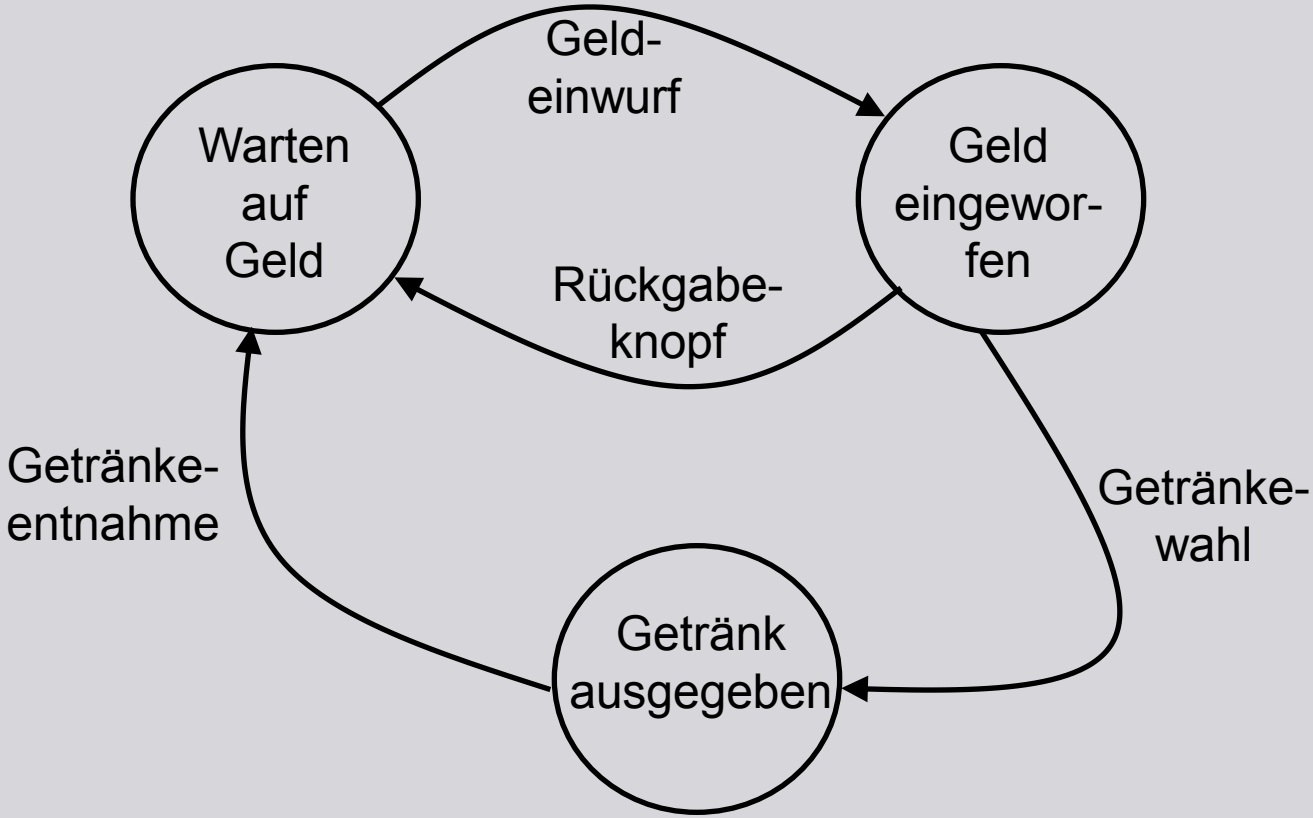
Beobachtungen des Verhaltens ...

- abhängig von der Vorgeschichte
- unterschiedliche Reaktionen auf die selben Aktionen des Kunden
- identisches Verhalten bei gleicher Vorgeschichte und identischen Aktionen des Kunden
- unter Umständen keine Reaktion des Automaten (ist auch eine Reaktion, nämlich nichts tun)

Zustandsautomaten

grafische Darstellung

(vereinfachte Funktion des Automaten*))

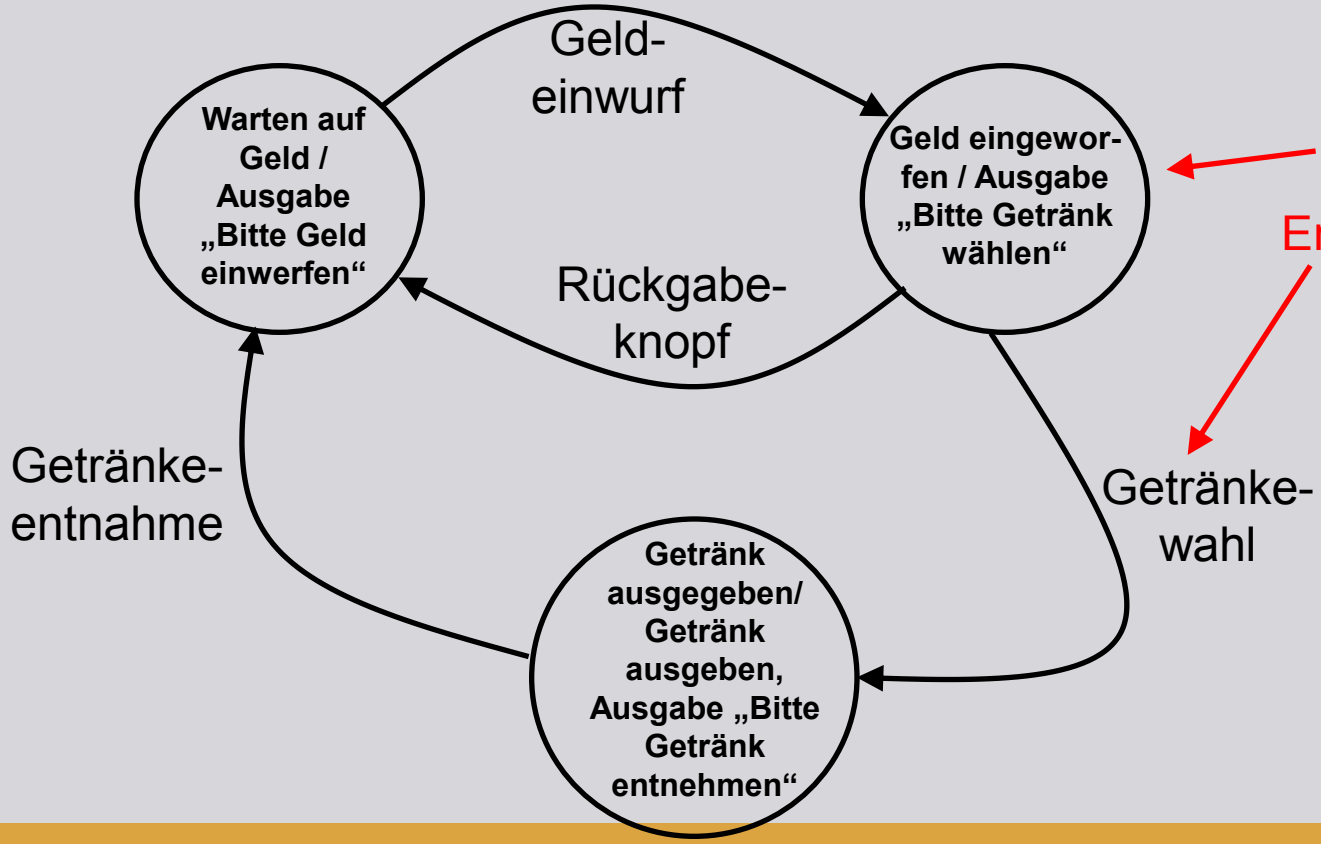


*) ohne Berücksichtigung von z.B.

- unterschiedlichen Preisen
- Geldeinwurf mit unterschiedlichen Münzen
- Verfügbarkeit eines Getränks

Zustandsautomaten

grafische Darstellung mit Aktionen (vereinfachte Funktion des Automaten*))

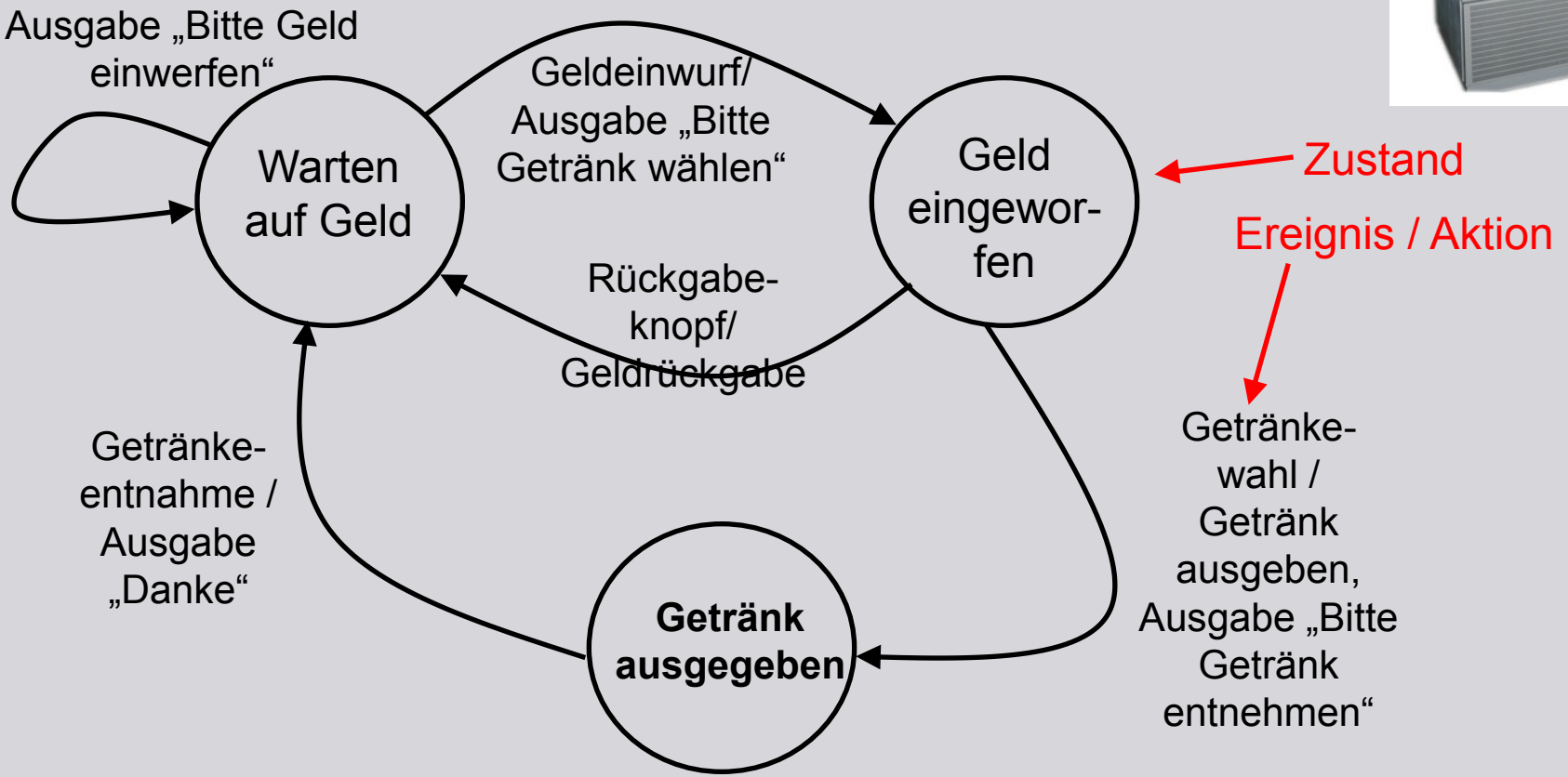


*) ohne Berücksichtigung von z.B.

- unterschiedlichen Preisen
- Geldeinwurf mit unterschiedlichen Münzen
- Verfügbarkeit eines Getränks

Zustandsautomaten

grafische Darstellung mit Aktionen nach Ereignissen (Alternative)



Zustandsautomaten

Definition: Mealy-Automat

Ein Mealy-Automat ist durch (Z, z_0, E, A, T) gegeben, wobei gilt

Z	ist eine endliche nichtleere Menge von Zuständen
z_0	$z_0 \in Z$ ist der Anfangszustand
E	ist die endliche Menge der möglichen Eingaben
A	ist die endliche Menge der möglichen Ausgaben
T	ist die Menge der Transitionen (Zustandsübergänge).

Jede Transition $t \in T$ ordnet einem Ausgangszustand $z_a \in Z$ und einer Eingabe $e \in E$ einen Folgezustand $z_f \in Z$ und eine Ausgabe $a \in A$ zu: $(z_a, e) \rightarrow (z_f, a)$

Zustandsautomaten

Beispiel Getränkeautomat

Z = { Warten auf Geld, Geld eingeworfen, Getränk ausgegebe

Z_0 = Warten auf Geld“

E = { Geldeinwurf, Getränkewahl, Rückgabeknopf, Getränkeentnahme }

A = {Geldrückgabe, Ausgabe „Bitte Geld einwerfen“, Getränk ausgeben+Ausgabe „Bitte Getränk entnehmen“, Ausgabe „Bitte Getränk wählen“, Ausgabe „Danke“ }

Zustandsautomaten

Beispiel Getränkeautomat

T =

Ausgangszustand	Ereignis	Folgezustand	Aktion
Warten auf Geld	-	Warten auf Geld	Ausgabe „Bitte Geld einwerfen“
Warten auf Geld	Geldeinwurf	Geld eingeworfen	Ausgabe „Bitte Getränk wählen“
Geld eingeworfen	Getränkewahl	Getränk ausgegeben	Ausgabe „Bitte Getränk entnehmen“
Geld eingeworfen	Rückgabeknopf	Warten auf Geld	Geldrückgabe
Getränk ausgegeben	Getränkeentnahme	Warten auf Geld	Ausgabe „Danke“

Zustandsautomaten

Automatenbeschreibung ➔ grafische Darstellung (nicht UML)

- Jedem Zustand $z \in Z$ wird ein Knoten (gezeichnet als Kreis) zugeordnet.
- Jeder Knoten wird mit dem Namen des zugehörigen Zustandes beschriftet.
- Jeder Transition $t \in T$ mit $t=(z_a, e, z_f, a)$ wird eine gerichtete Kante (gezeichnet als Pfeil) vom Zustand z_a nach Zustand z_f zugeordnet.
- Jede Transition wird mit ihrer Eingabe e und Ausgabe a beschriftet.

Zustandsautomaten

Anwendungen von Automaten

- Modellierung technischer Systeme, z.B.
  - Verkaufsautomaten
  - Karosseriefunktionen im Kfz (z. B. Blinker, Zentralverriegelung, Scheibenwischer)
  - Waschmaschine

- Zeichenkettenverarbeitung
  - Compiler
  - Kommandointerpreter

Zustandsautomaten

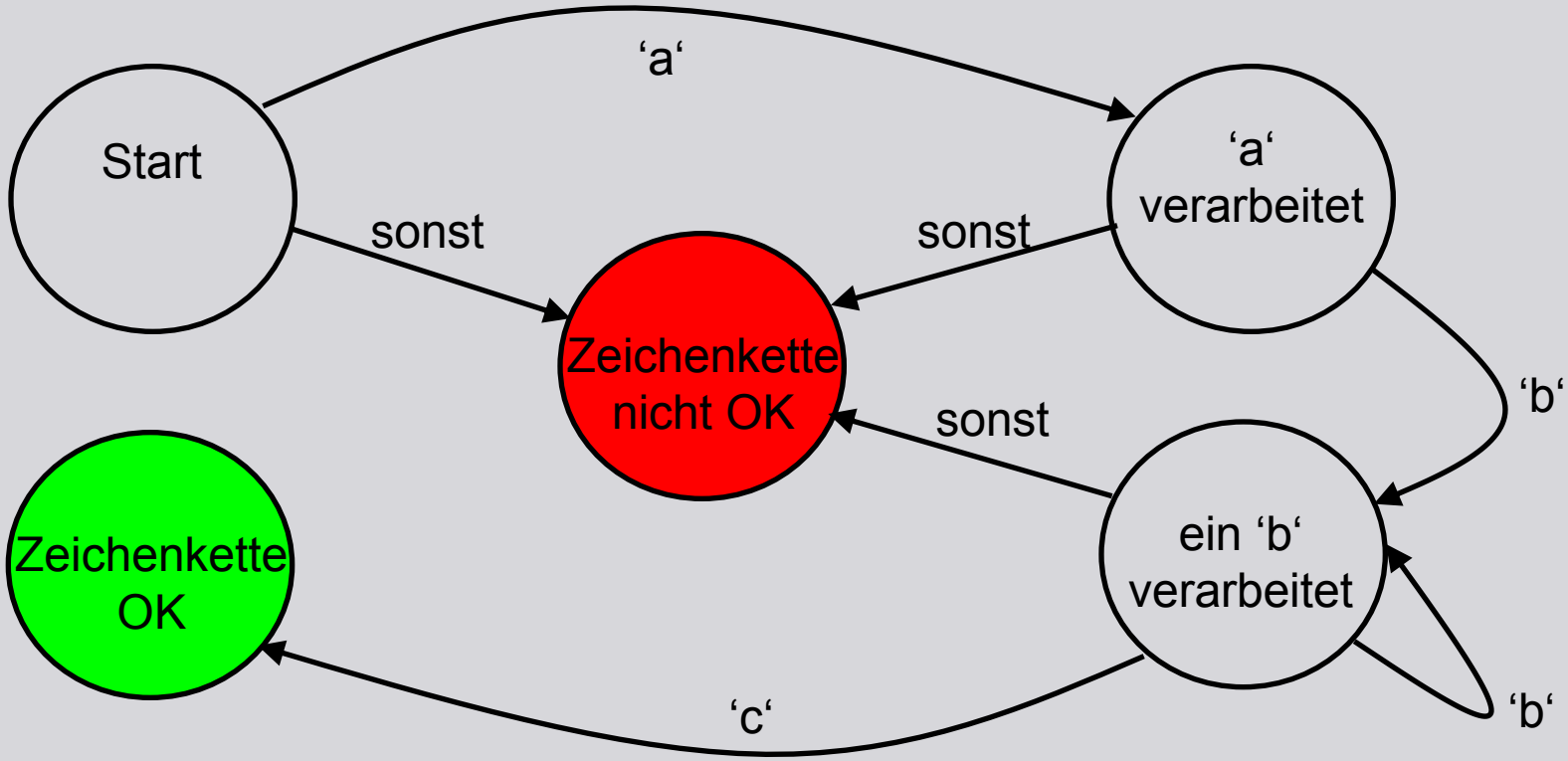
Zeichenkettenverarbeitung

- Erkennen einer Zeichenkette, die mit einem ‘a‘ anfängt, dann beliebig viele ‘b‘s (mindestens eins) enthält und mit einem ‘c‘ endet, d.h.
  - „abc“ OK
  - „abbbbbbbbbc“ OK
  - „abbbbbbd“ nicht OK

Zustandsautomaten

Zeichenkettenverarbeitung

- Erkennen einer Zeichenkette, die mit einem ‘a‘ anfängt, dann beliebig viele ‘b‘s (mindestens eins) enthält und mit einem ‘c‘ endet.



Zustandsautomaten

Umsetzung ins Programm

- systematische Umsetzung in C-Code möglich
- verschiedene äquivalente Lösungsmöglichkeiten
  - switch ... case
  - goto ...
  - Tabelle

Zustandsautomaten

Umsetzung ins Programm mittels switch ... case

- für die Zustände wird ein Aufzählungstyp mittels enum definiert
- der Zustand des Automaten wird in einer Variablen dieses Typs gespeichert (Initialisierung mit Anfangszustand)
- jedem Zustand $z \in Z$ wird eine case-Marke im Programm zugeordnet
- hinter jeder case-Marke zu einem Zustand $z \in Z$ wird die nächste Eingabe $e \in E$ eingelesen
- nach dem Lesen der Eingabe e hinter der Marke für z_a erfolgt eine Verzweigung, wobei jeder Zweig einer anwendbaren Transition $(z_a, e) \rightarrow (z_f, a)$ entspricht
- in jedem Zweig wird die Ausgabe $a \in A$ der Transition getätigt, anschließend wird per Zuweisung an die Zustandsvariable der Folgezustand definiert.
- der Automat wird in einer while-Schleife abgearbeitet, so lange bis ein Endzustand erreicht wird (andernfalls endlos).

Zustandsautomaten

Umsetzung ins Programm mittels switch ... case

Beispiel Getränkeautomat (Teil 1)

```
#include <stdio.h>
#include <stdlib.h>

int main(void)
{ typedef enum Zustaende { WaG /* Warten auf Geld */,
                           Ge /* Geld eingeworfen */,
                           Ga /* Getränk ausgegeben */};
  Zustaende zustand = WaG;
  int ereignis = 0; /* Startzustand */
```

... siehe nächste Folie ...

Zustandsautomaten

Teil 2

```
while (1)/* Automat läuft endlos */

 { printf("Bitte Ereignis eingeben: \n");
   printf("1 = Geldeinwurf\n2 = Getraenk gewählt\n3 = Getraenk entnommen\n4 =
Rueckgabeknopf\n");
   switch(zustand)
   { case WaG:  printf("\nBitte Geld einwerfen\n");
                scanf("%d",&ereignis);
                if (ereignis == 1)
                  zustand = Ge;
                break;
     case Ge:   printf("\nBitte Getraenk wählen\n");
                scanf("%d",&ereignis);
                if (ereignis == 2)
                  zustand = Ga;
                else if (ereignis == 4)
                  zustand = WaG;
                break;
     case Ga:   printf("\nBitte Getraenk entnehmen\n");
                scanf("%d",&ereignis);
                if (ereignis == 3)
                  zustand = WaG;
                break;
     default: printf("ungültiger Zustand\n");
   };
 };
}
```

Zustandsautomaten

Umsetzung ins Programm mittels goto‘s

- jedem Zustand $z \in Z$ wird eine Sprung-Marke im Programm zugeordnet
- hinter jeder Sprung-Marke zu einem Zustand $z \in Z$ wird die nächste Eingabe $e \in E$ eingelesen
- nach dem Lesen der Eingabe e hinter der Marke für z_a erfolgt eine Verzweigung, wobei jeder Zweig einer anwendbaren Transition $(z_a, e) \rightarrow (z_f, a)$ entspricht
- in jedem Zweig wird die Ausgabe $a \in A$ der Transition getätigt, anschließend wird per goto an die Sprungmarke des Folgezustands gesprungen.

Zustandsautomaten

Umsetzung ins Programm mittels goto‘s

Beispiel Getränkeautomat (Teil 1)

```
#include <stdlib.h>

int main(void)
{ int ereignis = 0; /* Startzustand */
  { printf("Bitte Ereignis eingeben: \n");
    printf("1 = Geldeinwurf\n2 = Getraenk gewaehlt\n3 = Getraenk entnommen\n4 = Rueckgabeknopf\n");
    WaG:  printf("\nBitte Geld einwerfen\n");
              scanf("%d",&ereignis);
              if (ereignis == 1)
                goto Ge;
              else goto WaG;
```

... siehe nächste Folie ...

Zustandsautomaten

Umsetzung ins Programm mittels goto‘s

Beispiel Getränkeautomat (Teil 2)

```
Ge:   printf("\nBitte Getraenk wählen\n");
        scanf("%d",&ereignis);
        if (ereignis == 2)
          goto Ga;
        else if (ereignis == 4)
              goto WaG;
        else goto Ge;
   Ga:   printf("\nBitte Getraenk entnehmen\n");
        scanf("%d",&ereignis);
        if (ereignis == 3)
          goto WaG;
        else goto Ga;
  };
}
```

Zustandsautomaten

Umsetzung ins Programm mittels Tabelle

- in einer Tabelle wird für jeden möglichen Zustand abhängig vom Ereignis der Folgezustand und die auszuführenden Aktionen definiert.
- Der Zustandsautomat wird in einer Schleife realisiert, die bei jedem Ereignis aus Basis des aktuellen Zustands den entsprechenden Tabelleneintrag auswählt, die Aktion ausführt und den nachfolgenden Zustand einstellt.

Zustandsautomaten

Umsetzung ins Programm mittels Tabelle

Zustände / Ereignisse	Warten auf Geld	Geld eingeworfen	Getränk ausgegeben
Geldeinwurf	Geld Eingeworfen „Bitte Getränk wählen“	Geld Eingeworfen „Bitte Getränk wählen“	Getränk ausgegeben „Bitte Getränk entnehmen“
Rückgabeknopf	Warten auf Geld „Bitte Geld einwerfen“	Warten auf Geld „Bitte Geld einwerfen“	Getränk ausgegeben „Bitte Getränk entnehmen“
Getränkewahl	Warten auf Geld „Bitte Geld einwerfen“	Getränk ausgegeben „Bitte Getränk entnehmen“	Getränk ausgegeben „Bitte Getränk entnehmen“
Getränkentnahme	Warten auf Geld „Bitte Geld einwerfen“	Geld Eingeworfen „Bitte Getränk wählen“	Warten auf Geld „Bitte Geld einwerfen“

Zustandsautomaten

Umsetzung ins Programm mittels Tabelle

Beispiel Getränkeautomat (Teil 1)

```
#include <stdio.h>
#include <stdlib.h>

int main(void)
{
  typedef enum Zustaende { WaG /* Warten auf Geld */,
                           Ge /* Geld eingeworfen */,
                           Ga /* Getränk ausgegeben */};

  typedef enum Aktionen {Geld, Wahl, Rueck, Entnahme };
```

... siehe nächste Folie ...

Zustandsautomaten

Umsetzung ins Programm mittels Tabelle

Beispiel Getränkeautomat (Teil 2)

```
typedef enum Ereignisse {GI, Wa, En, Ru};

Zustaende zustand = WaG;
Aktionen aktion = Geld;
Ereignisse ereignis = GI; /* Startzustand */

typedef struct Tab_element {  Zustaende fz; Aktionen ak; };

Tab_element z_tabelle [3][4];

/* Besetzen der Tabellenelemente */
```

... siehe nächste Folie ...

Zustandsautomaten

Umsetzung ins Programm mittels Tabelle

Beispiel Getränkeautomat (Teil 3)

```
z_tabelle[WaG][Gl].fz=Ge;
  z_tabelle[WaG][Gl].ak=Wahl;
  z_tabelle[WaG][Wa].fz=WaG;
  z_tabelle[WaG][Wa].ak=Geld;
  z_tabelle[WaG][Ru].fz=WaG;
  z_tabelle[WaG][Ru].ak=Geld;
  z_tabelle[WaG][En].fz=WaG;
  z_tabelle[WaG][En].ak=Geld;
  z_tabelle[Ge][Gl].fz=Ge;
  z_tabelle[Ge][Gl].ak=Wahl;
  z_tabelle[Ge][Wa].fz=Ga;
  z_tabelle[Ge][Wa].ak=Entnahme;
  /* ... Tabelle hier nicht vollständig */
```

... siehe nächste Folie ...

Zustandsautomaten

Umsetzung ins Programm mittels Tabelle

Beispiel Getränkeautomat (Teil 4)

```
while (1)/* Automat läuft endlos */

{ printf("Bitte Ereignis eingeben: \n"); printf("1 = Geldeinwurf\n"); printf("2 = Getraenk gewählt\n");
  printf("3 = Getraenk entnommen\n"); printf("4 = Rueckgabeknopf\n");

  switch(aktion)
  { case Geld: printf("\nBitte Geld einwerfen\n"); break;
    case Wahl: printf("\nBitte Getraenk wählen\n"); break;
    case Rueck: printf("\nBitte Geld einwerfen\n"); break;
    case Entnahme: printf("\nBitte Getraenk entnehmen\n"); break;
    default: printf("Programmfehler 1\n");  break;
  };
```

... siehe nächste Folie ...

Zustandsautomaten

Umsetzung ins Programm mittels Tabelle

Beispiel Getränkeautomat (Teil 5)

```
scanf("%d",&ereignis);
    if (ereignis > Ru)
      printf("Programmfehler 2\n");

    aktion = z_tabelle[zustand][ereignis-1].ak;
    zustand = z_tabelle[zustand][ereignis-1].fz;
  };

  system("PAUSE");
  return 0;
}
```

Zustandsautomaten

Vor-/Nachteile

- goto
  - übersichtlich
  - Änderungen ggf. an vielen Stellen
  - goto fragwürdig
- switch-case-Lösung
  - übersichtlich
  - Änderungen ggf. an vielen Stellen
- Tabelle:
  - übersichtlich
  - Änderungen ggf. nur an der Tabelle
  - eigentliche Ausführung unabhängig von Änderungen

Zum Schluss dieses Abschnitts …

Noch Fragen ??