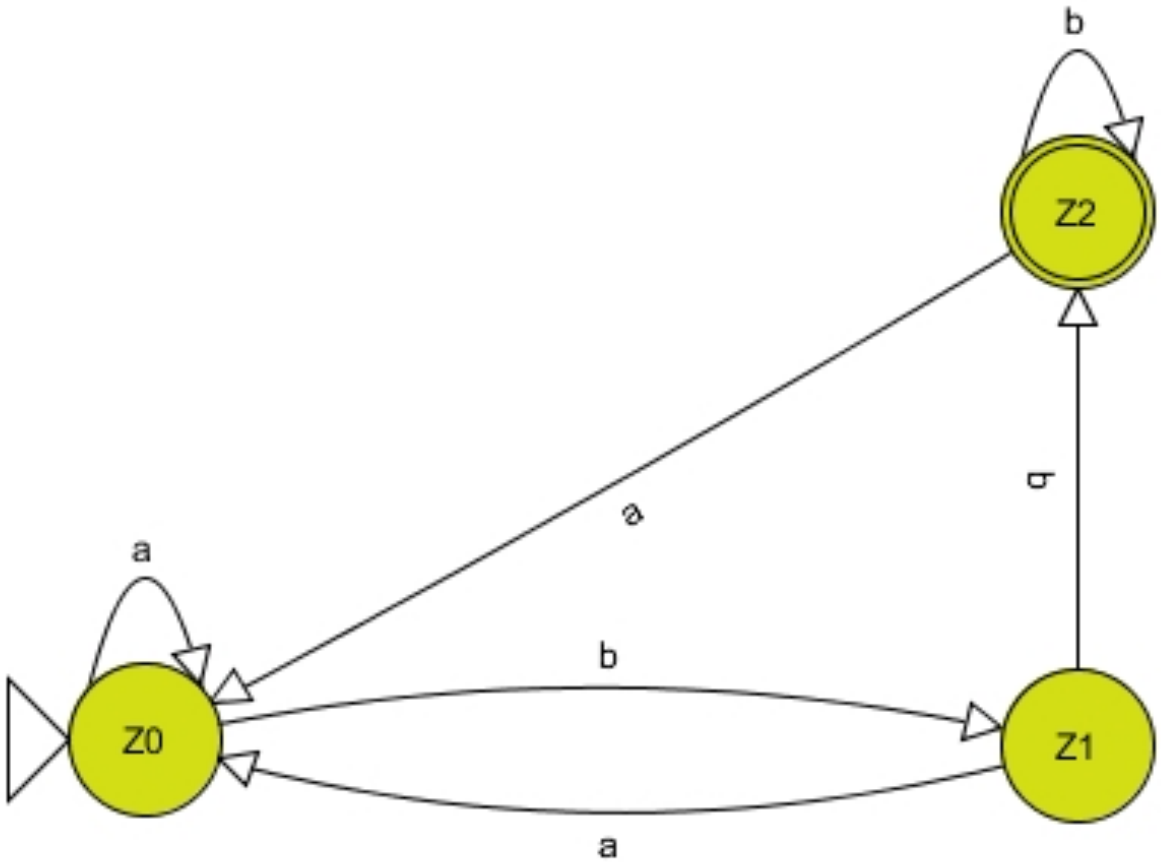
1. Formale Sprachen

1.3 Endliche Automaten

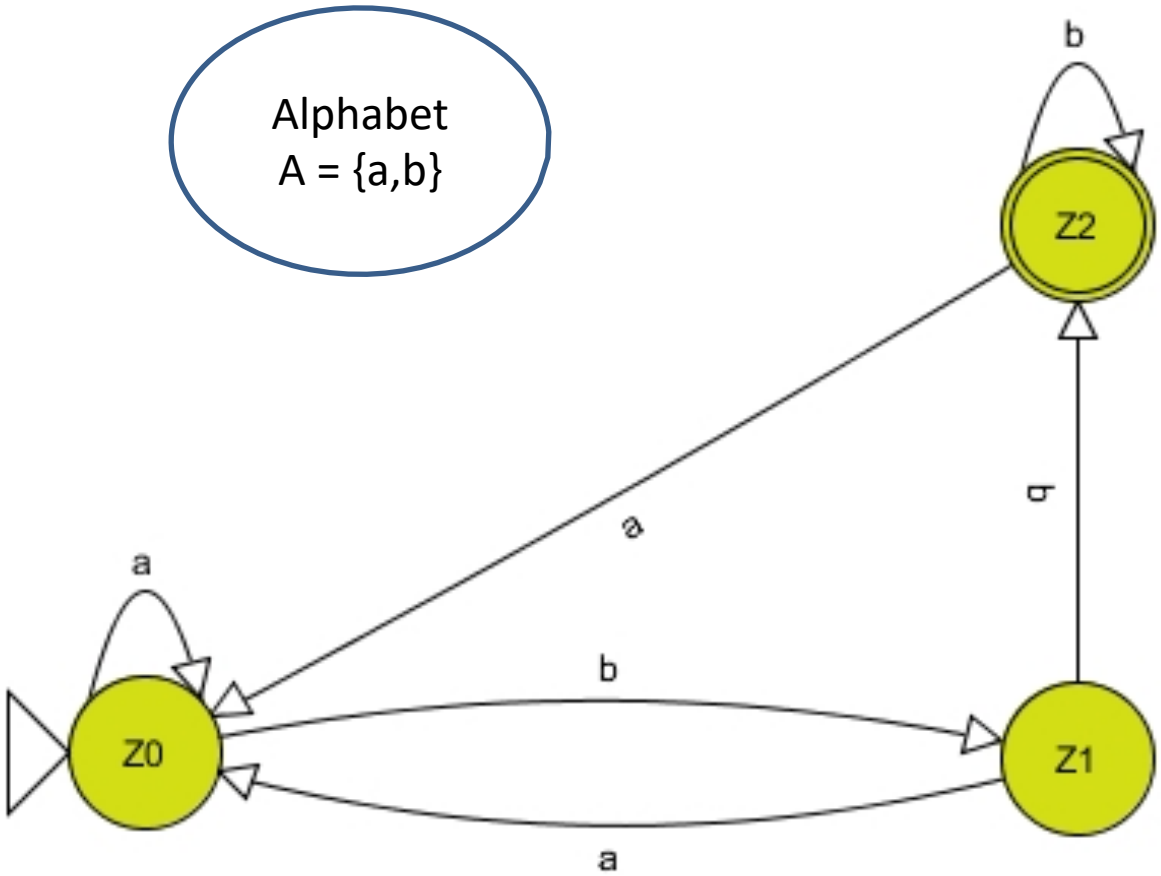
Ein endlicher Automat ist ein spezielles Zustandsdiagramm mit endlich vielen Zuständen.

Für bestimmte formale Sprachen (den sogenannten **regulären Sprachen**) kann man mit einem endlichen Automaten prüfen, ob ein Wort zu dieser Sprache gehört.

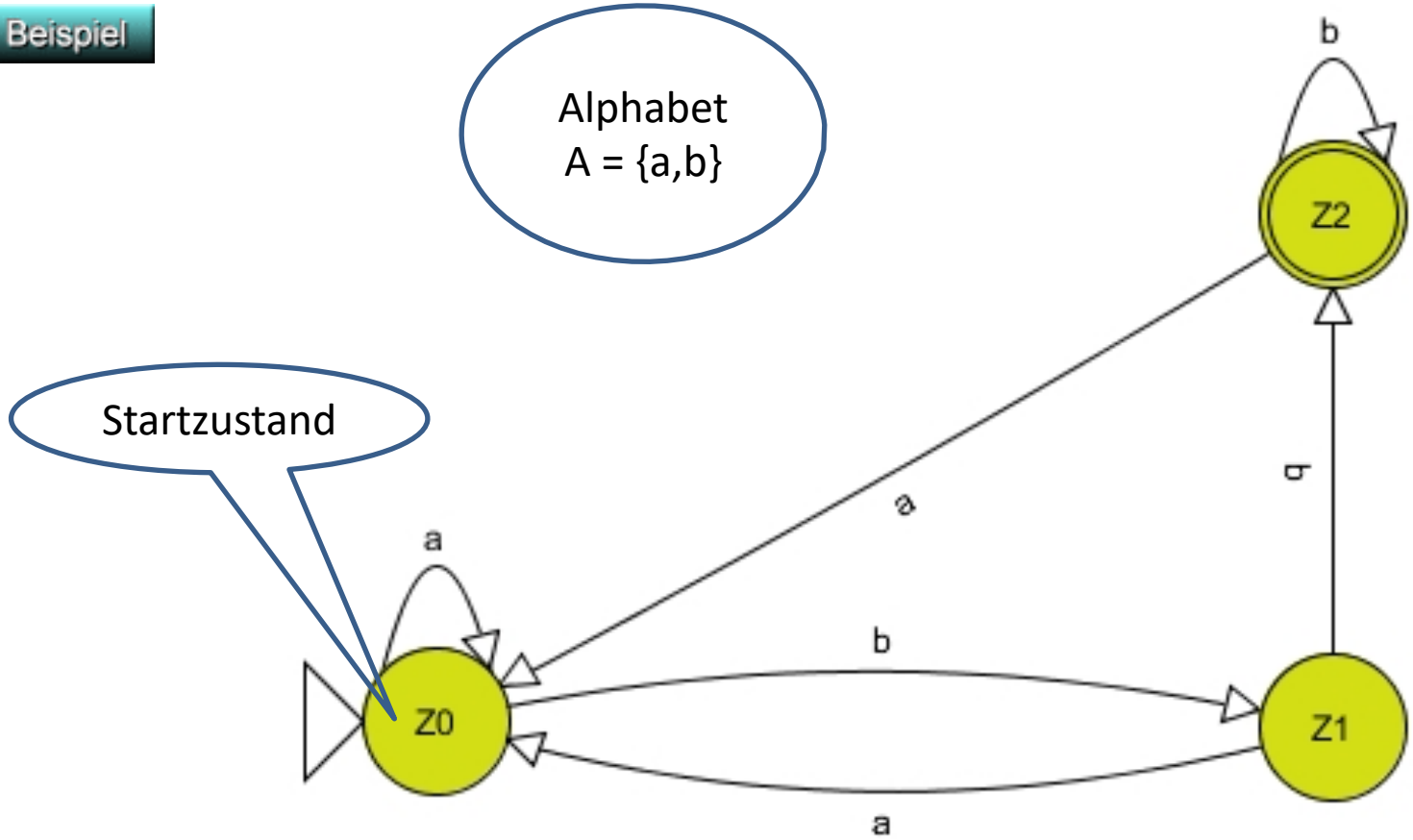
Beispiel



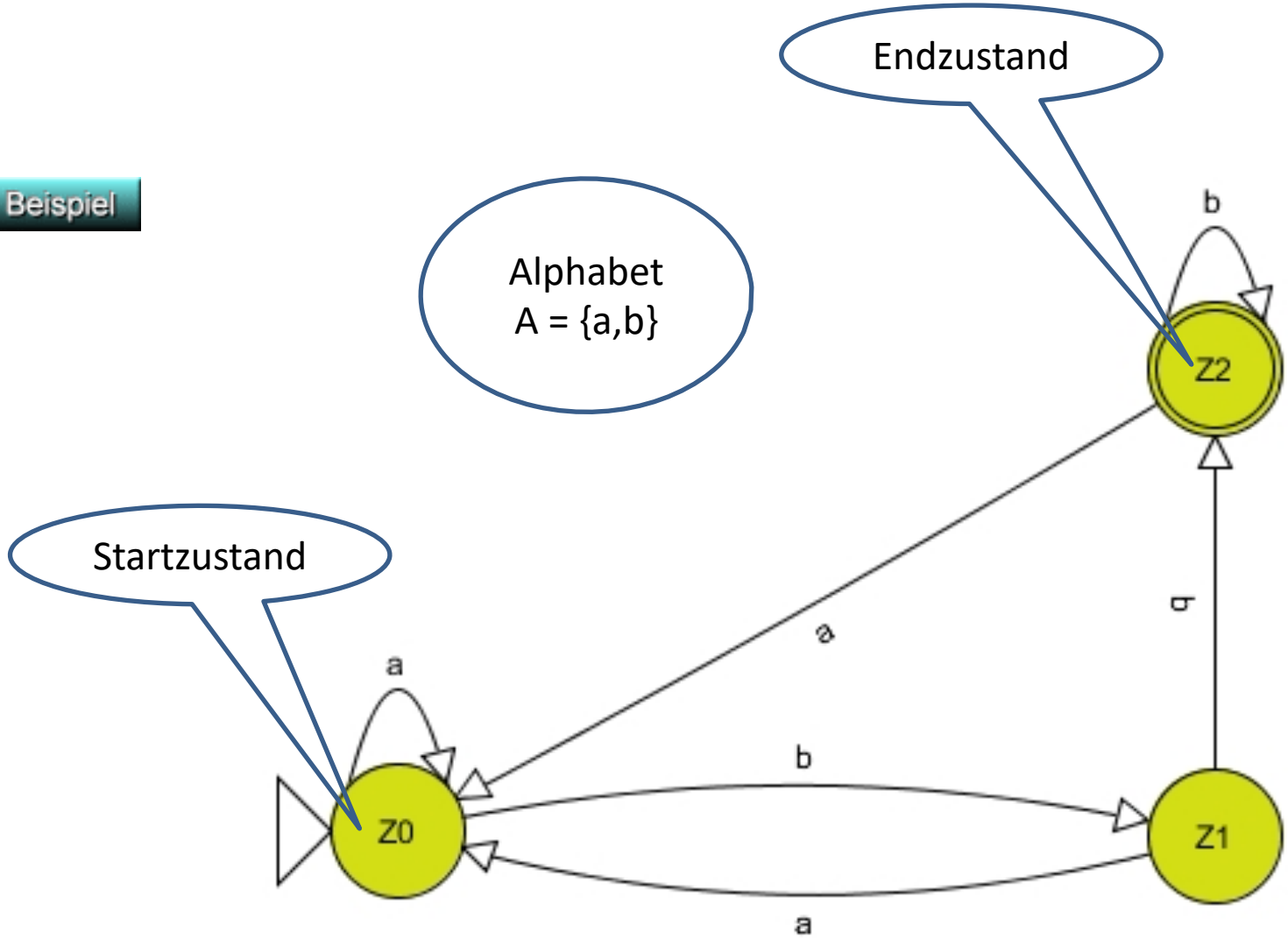
Beispiel



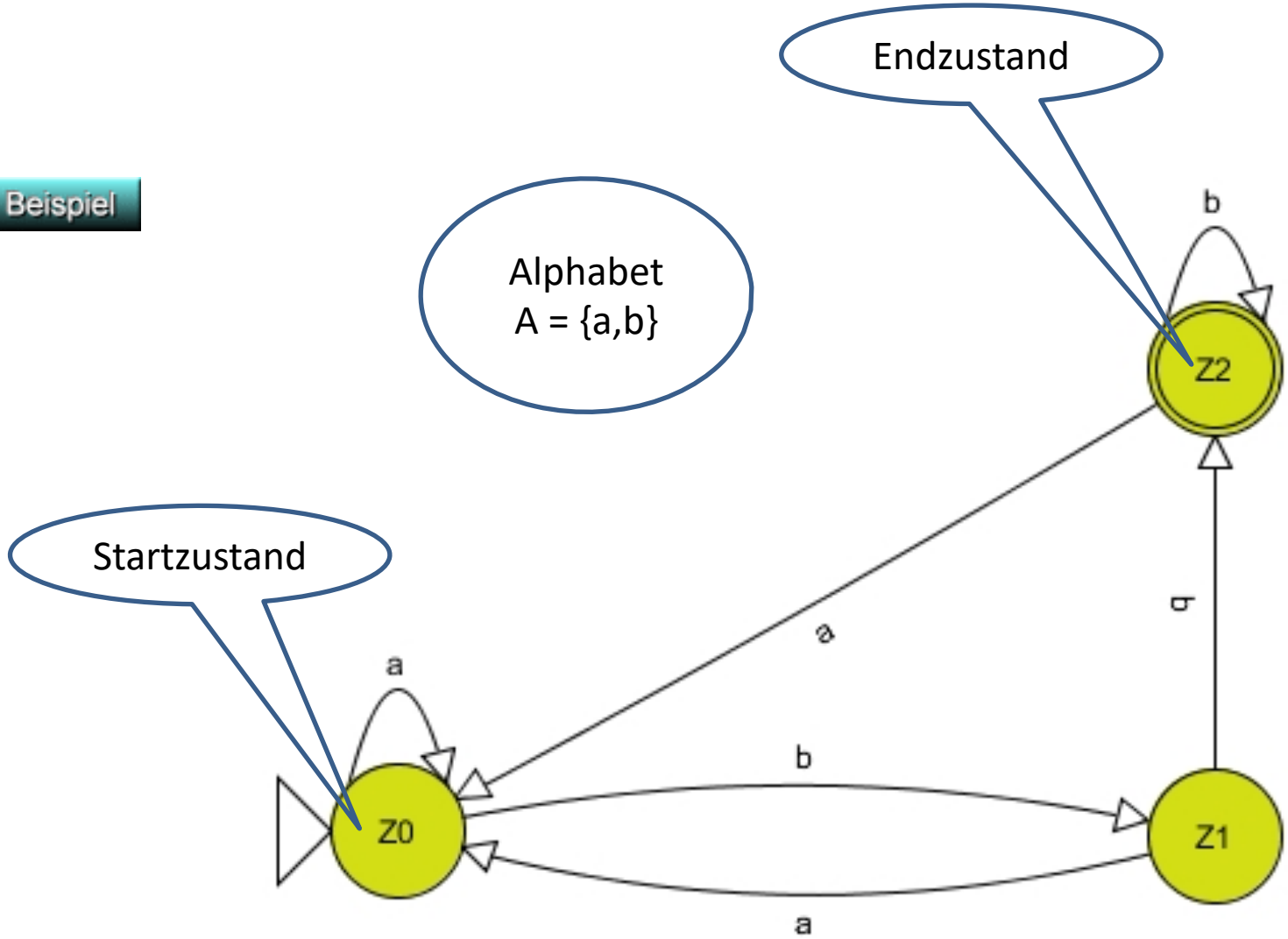
Beispiel



Beispiel

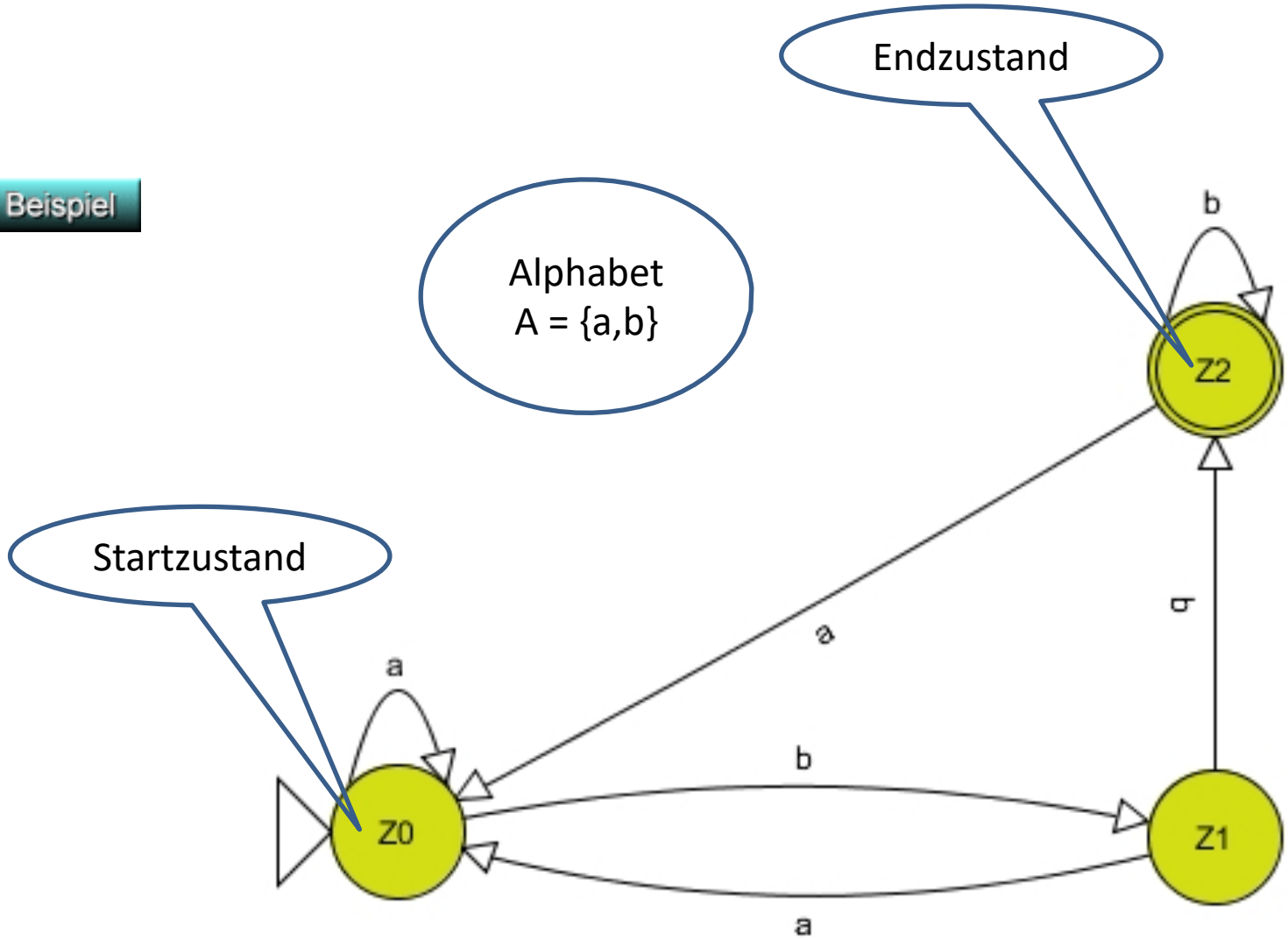


Beispiel



aabbabb ⇨ Abarbeitung ⇨ Zustand Z2 ⇨ Das Wort gehört zur Sprache

Beispiel

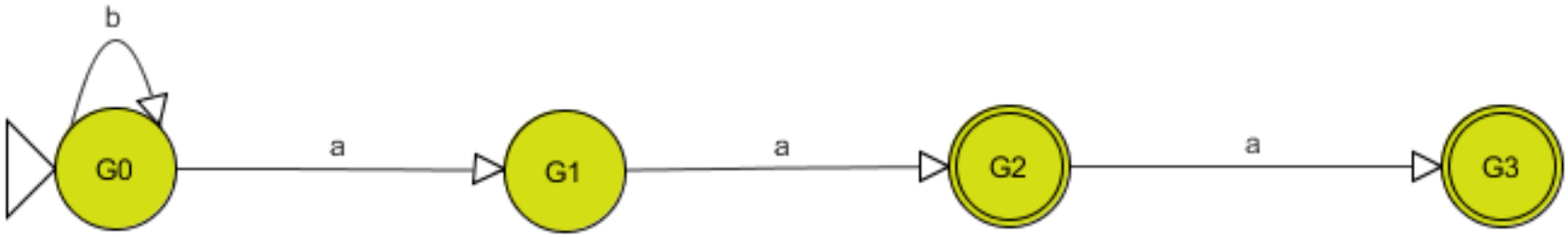


bbbab ⇨ Abarbeitung ⇨ Zustand Z1 ⇨ Das Wort gehört nicht zur Sprache

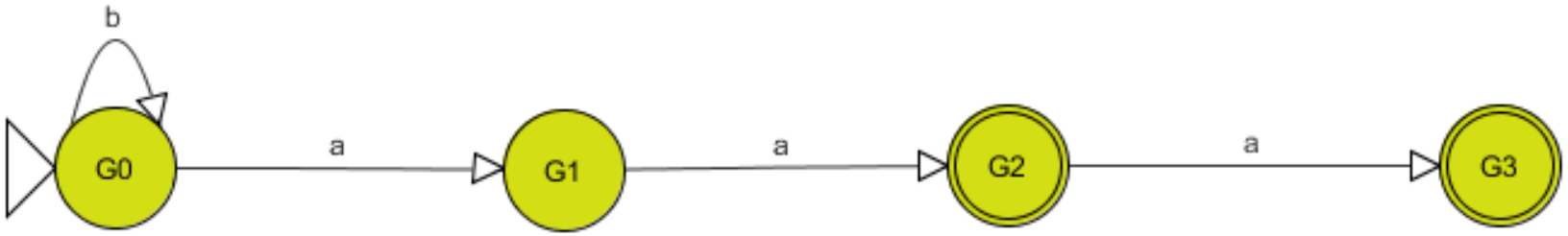
Ein Automat soll alle Wörter über dem Alphabet A = {a,b} erkennen, die mit mit **beliebig vielen Zeichen b beginnen und auf zwei oder drei a enden.** Weitere Zeichen a oder b sollen in einem Wort nicht vorkommen.

In EBNF :
wort = {b}aa[a]

wort = {b}aa[a]



wort = {b}aa[a]



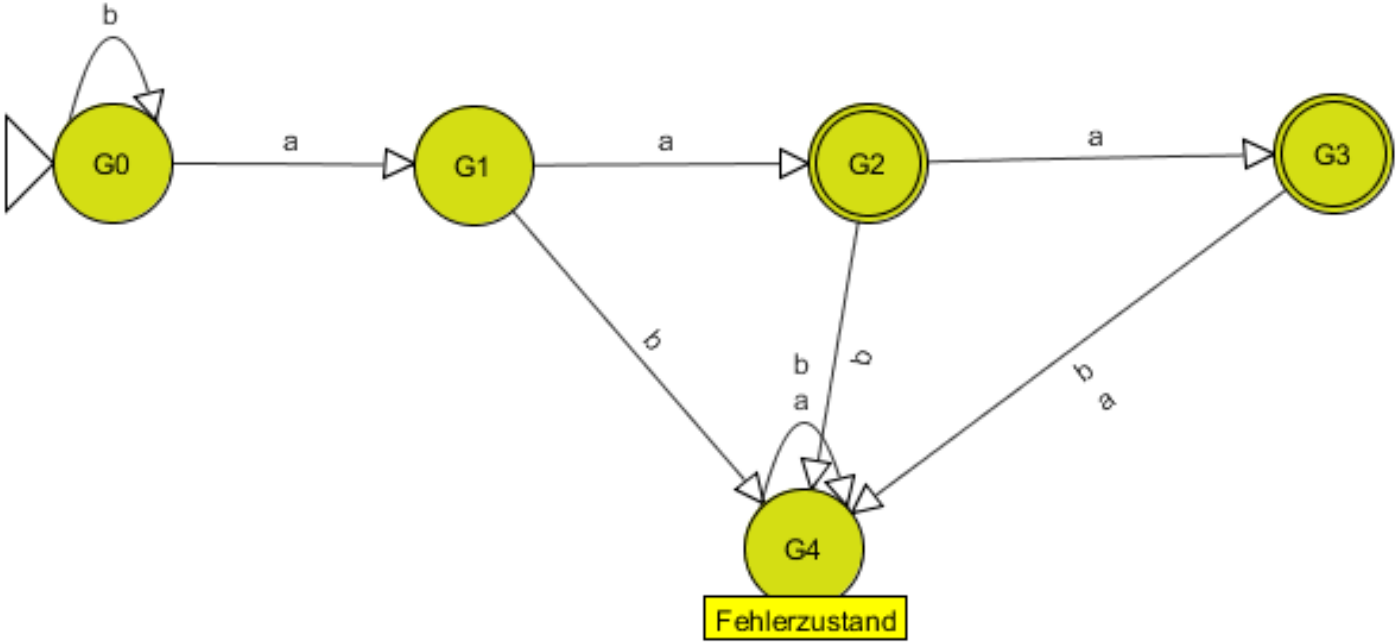
bbaaaa

bbbaab

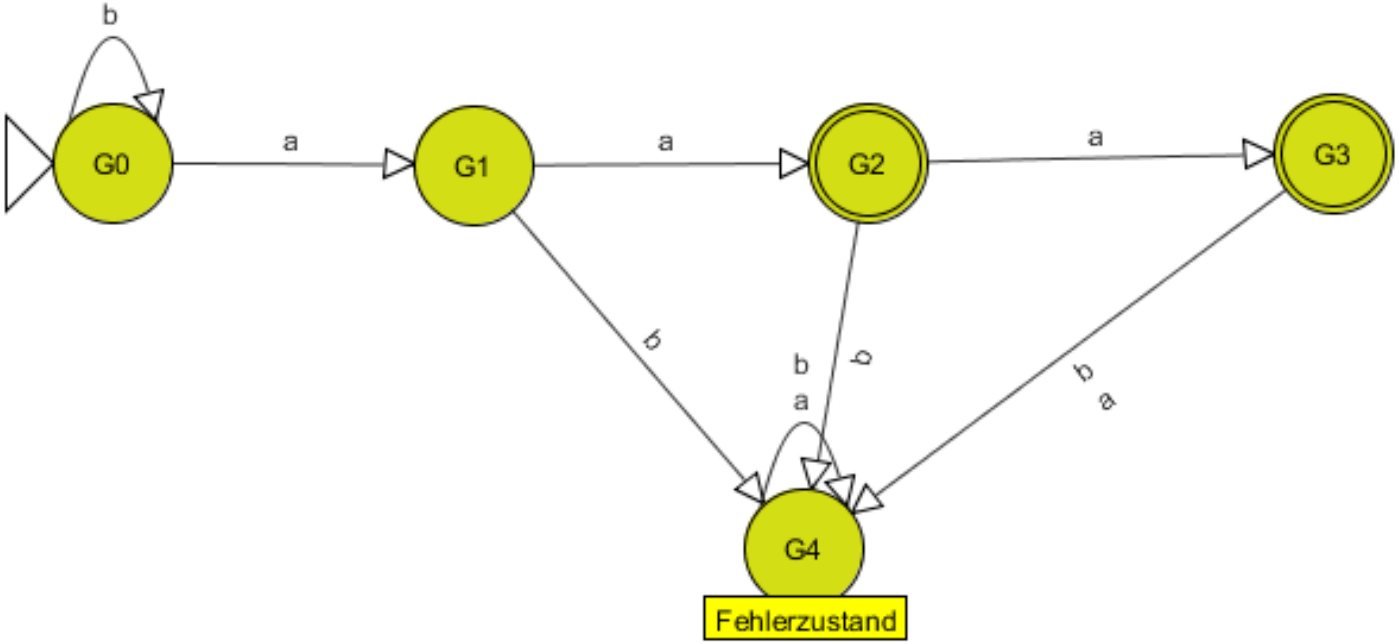
⇨ Abarbeitung nicht voll-ständig möglich

⇨ Das Wort gehört nicht zur Sprache

Automat mit einem Fehlerzustand:

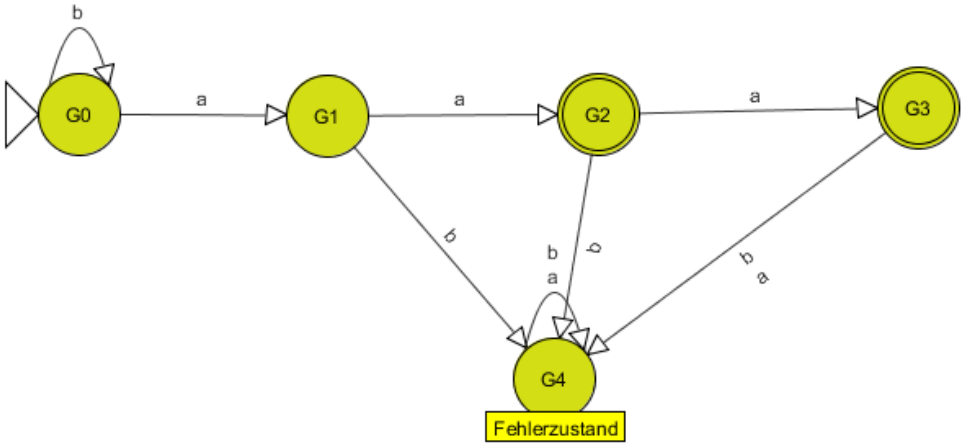


Automat mit einem Fehlerzustand:

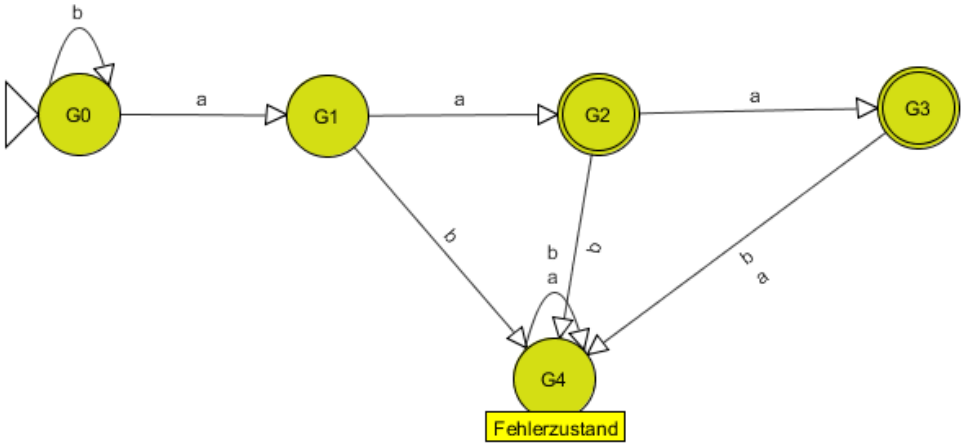


bbaaaa
bbbaab

⇨ vollständige Abarbeitung ⇨ Zustand G4 ⇨ Das Wort gehört nicht zur Sprache



Zu jedem Zustand gibt es bei jedem Eingabezeichen einen Übergang.
Der Automat heißt in so einem Fall **vollständig** .



Zu jedem Zustand gibt es bei jedem Eingabezeichen einen Übergang.
Der Automat heißt in so einem Fall **vollständig** .

Gibt es von jedem Zustand für ein bestimmtes Zeichen höchstens einen Übergang, heißt der Automat **deterministisch**.

Formale Definition:

Ein vollständiger deterministischer endlicher Automat (DEA) ist festgelegt durch:

1) eine endliche Menge Z von Zuständen
   $Z = \{Z_0, Z_1, Z_2 \ldots, Z_n\}$
2) ein endliches Eingabealphabet $\mathrm{A} = \{t_0, t_1, t_2 \ldots, t_m\}$
3) einen Startzustand $S \in Z$
4) eine Menge E von Endzuständen ($\mathrm{E} \subseteq Z$)
5) eine Übergangsfunktion f, die jedem möglichen Paar aus Zustand und Eingabezeichen einen Folgezustand zuordnet.

Übung 1

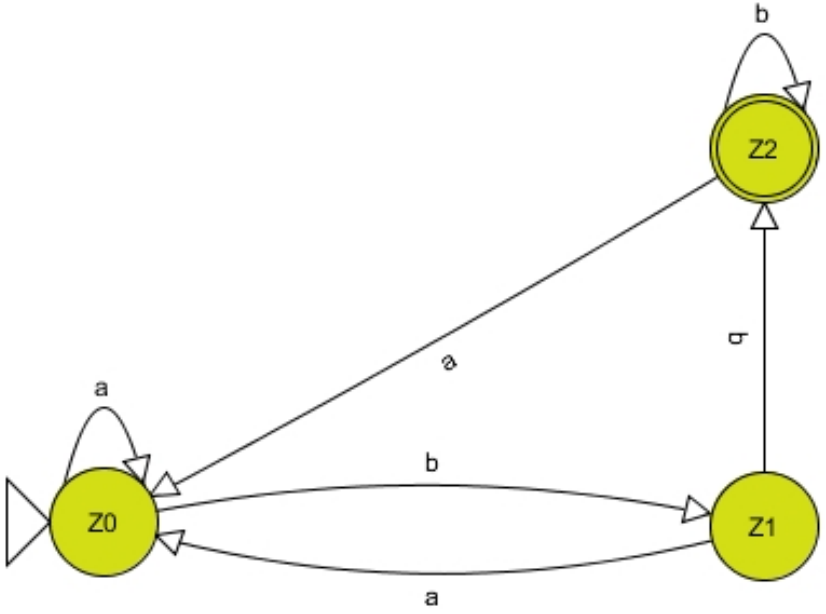
Starte das Programm jflap.jar (Internet: http://www.jflap.org).
Gib die Automaten aus dem Skript ein und teste sie.

Erstelle einen neuen Automaten und gib ihn deinem Nachbarn, der herausfinden und beschreiben soll, welche Sprache der Automat erkennt.

Weitere Übungsmöglichkeit:

https://flaci.com/home/

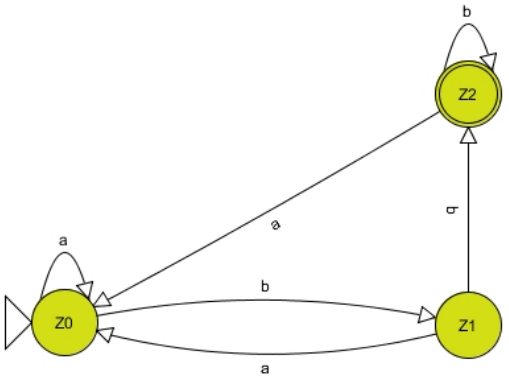
Beispiel



$$Z = \{Z_0, Z_1, Z_2\}$$
$$A = \{a, b\}$$
$$S = Z_0$$
$$E = \{Z_2\}$$

$$f: Z \times A \mapsto Z$$

$$f(Z_0, a) = Z_0$$
$$f(Z_0, b) = Z_1$$
$$f(Z_1, a) = Z_0$$
$$f(Z_1, b) = Z_2$$
$$f(Z_2, a) = Z_0$$
$$f(Z_2, b) = Z_2$$



Erstellen einer Grammatik:

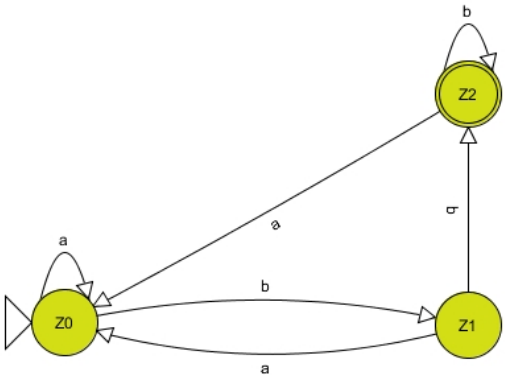
$Z = \{Z_0, Z_1, Z_2\}$
$A = \{a, b\}$
$S = Z_0$
$E = \{Z_2\}$

$f: Z \times A \mapsto Z$
$f(Z_0, a) = Z_0$
$f(Z_0, b) = Z_1$
$f(Z_1, a) = Z_0$
$f(Z_1, b) = Z_2$
$f(Z_2, a) = Z_0$
$f(Z_2, b) = Z_2$

Menge der syntaktischen Variablen:
$V = \{\langle Z_0 \rangle, \langle Z_1 \rangle, \langle Z_2 \rangle\}$

Alphabet:
$A = \{a, b\}$

Startvariable:
$S = \langle Z_0 \rangle$



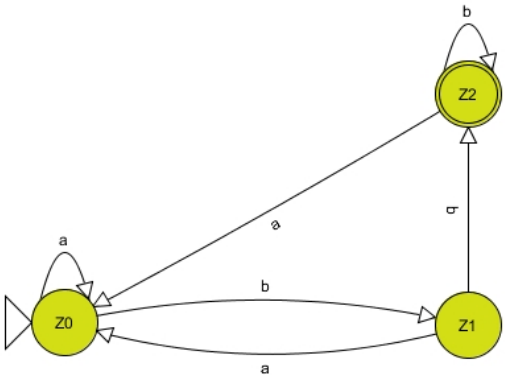
Erstellen einer Grammatik:

$$Z = \{Z_0, Z_1, Z_2\}$$
$$A = \{a, b\}$$
$$S = Z_0$$
$$E = \{Z_2\}$$

$$f: Z \times A \mapsto Z$$
$$f(Z_0, a) = Z_0$$
$$f(Z_0, b) = Z_1$$
$$f(Z_1, a) = Z_0$$
$$f(Z_1, b) = Z_2$$
$$f(Z_2, a) = Z_0$$
$$f(Z_2, b) = Z_2$$

Produktionsregeln:

$$\langle Z_0 \rangle \to a\langle Z_0 \rangle$$
$$\langle Z_0 \rangle \to b\langle Z_1 \rangle$$
$$\langle Z_1 \rangle \to a\langle Z_0 \rangle$$
$$\langle Z_1 \rangle \to b\langle Z_2 \rangle$$
$$\langle Z_2 \rangle \to a\langle Z_0 \rangle$$
$$\langle Z_2 \rangle \to b\langle Z_2 \rangle$$



Erstellen einer Grammatik:

$$Z = \{Z_0, Z_1, Z_2\}$$
$$A = \{a, b\}$$
$$S = Z_0$$
$$E = \{Z_2\}$$

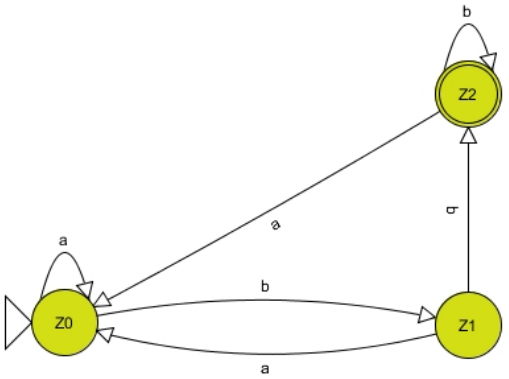
$$f: Z \times A \mapsto Z$$
$$f(Z_0, a) = Z_0$$
$$f(Z_0, b) = Z_1$$
$$f(Z_1, a) = Z_0$$
$$f(Z_1, b) = Z_2$$
$$f(Z_2, a) = Z_0$$
$$f(Z_2, b) = Z_2$$

Zusammenfassung:

$$\langle Z_0 \rangle \rightarrow a\langle Z_0 \rangle | b\langle Z_1 \rangle$$

$$\langle Z_1 \rangle \rightarrow a\langle Z_0 \rangle | b\langle Z_2 \rangle$$

$$\langle Z_2 \rangle \rightarrow a\langle Z_0 \rangle | \ b\langle Z_2 \rangle$$



Erstellen einer Grammatik:

$$Z = \{Z_0, Z_1, Z_2\}$$
$$A = \{a, b\}$$
$$S = Z_0$$
$$E = \{Z_2\}$$

$$f: Z \times A \mapsto Z$$
$$f(Z_0, a) = Z_0$$
$$f(Z_0, b) = Z_1$$
$$f(Z_1, a) = Z_0$$
$$f(Z_1, b) = Z_2$$
$$f(Z_2, a) = Z_0$$
$$f(Z_2, b) = Z_2$$

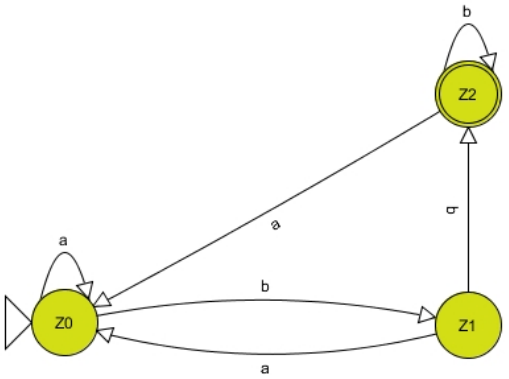
Endzustände mit ε:

$$\langle Z_0 \rangle \rightarrow a\langle Z_0 \rangle | b\langle Z_1 \rangle$$

$$\langle Z_1 \rangle \rightarrow a\langle Z_0 \rangle | b\langle Z_2 \rangle$$

$$\langle Z_2 \rangle \rightarrow a\langle Z_0 \rangle | b\langle Z_2 \rangle | \varepsilon$$

Erstellen einer Grammatik:



$$Z = \{Z_0, Z_1, Z_2\}$$
$$A = \{a, b\}$$
$$S = Z_0$$
$$E = \{Z_2\}$$

$$f: Z \times A \mapsto Z$$
$$f(Z_0, a) = Z_0$$
$$f(Z_0, b) = Z_1$$
$$f(Z_1, a) = Z_0$$
$$f(Z_1, b) = Z_2$$
$$f(Z_2, a) = Z_0$$
$$f(Z_2, b) = Z_2$$

Endzustände ohne ε:

$$\langle Z_0 \rangle \rightarrow a\langle Z_0 \rangle | b\langle Z_1 \rangle$$

$$\langle Z_1 \rangle \rightarrow a\langle Z_0 \rangle | b\langle Z_2 \rangle | b$$

$$\langle Z_2 \rangle \rightarrow a\langle Z_0 \rangle | b\langle Z_2 \rangle | b$$

Die Produktionsregeln haben alle die Form

$\langle Z_i \rangle \rightarrow t\langle Z_j \rangle$ oder $\langle Z_i \rangle \rightarrow t$

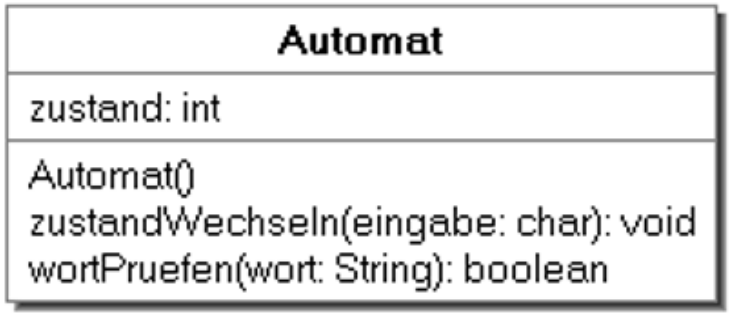
Die Sprachen, die durch solche Grammatiken beschrieben werden nennt man **reguläre Sprachen**.

Ein wichtiger Satz aus der theoretischen Informatik lautet:

Eine Sprache ist genau dann regulär, wenn sie von einem endlichen Automaten akzeptiert wird.

Implementieren eines endlichen Automaten

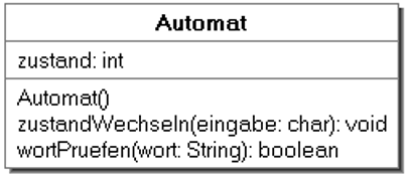
Ein deterministischer endlicher Automat kann in Java sehr einfach mit Hilfe der Mehrfachauswahl implementiert werden.
Das folgende Klassendiagramm modelliert einen Automaten, dessen Sprache einfache Zeichen (char) als Alphabet besitzt.



Klasse Automat
Methode void zustandWechseln(char eingabe)

Mit einer Mehrfachauswahl werden die möglichen Werte für das Attribut *zustand* erfasst. Für jeden möglichen Wert von zustand gibt es dann wieder eine Mehrfachauswahl, die die möglichen Werte des zu prüfenden Zeichen erfasst.

```
wenn zustand = 0, dann
  wenn eingabe = 'a', dann zustand = neuer Wert
  wenn eingabe = 'b', dann zustand = neuer Wert
  usw.
wenn zustand = 1, dann
  wenn eingabe = 'a', dann zustand = neuer Wert
  wenn eingabe = 'b', dann zustand = neuer Wert
  usw.
usw.
```



Klasse Automat
Methode boolean wortPruefen(String wort)

Die Zeichenkette wird beginnend mit dem ersten Zeichen abgearbeitet, es wird jeweils die Methode *zustandWechseln(char eingabe)* für das aktuelle Zeichen aufgerufen.
Ist das Wort vollständig abgearbeitet, wird geprüft, ob der erreichte Zustand akzeptierend ist und entsprechend *true* oder *false* zurückgegeben.

Übung 2

Wiederhole z.B. im Javahandbuch die Mehrfachauswahl mit der Konstruktion *switch* und Methoden der Klasse *String*.

Implementiere dann einen Automaten über eine ab-Sprache und teste ihn.

Zusatzaufgabe: Entwickle eine GUI sowie eine Methode *schrittweisePruefen(String wort)*, mit der das Wort schrittweise abgearbeitet wird und die entsprechenden Zustandswechsel ausgegeben werden.

Übung 3

Die folgende Javamethode führt die Zustandsänderungen eines vollständigen endlichen Automaten über dem Alphabet {a,b} aus. Der Startzustand enspricht dem Wert zustand = 0.
Ein Wort wird nur dann akzeptiert, wenn *zustand* nach Abarbeitung des Wortes den Wert 3 hat.

```
 public void zustandWechseln(char eingabe){
   switch(zustand) {
     case 0:{
       switch(eingabe){
         case 'a': zustand = -1 ;   break;
         case 'b': zustand = 1 ;   break;
       }
     } break;

     case 1:{
       switch(eingabe){
         case 'a': zustand = 2 ;   break;
         case 'b': zustand = 1 ;   break;
       }
     } break;

     case 2:{
       switch(eingabe){
         case 'a': zustand = 3 ;   break;
         case 'b': zustand = 1 ;   break;
       }
     } break;

     case 3:{
       switch(eingabe){
         case 'a': zustand = 2 ;   break;
         case 'b': zustand = 1 ;   break;
       }
     } break;

   }
}
```

Zeichne den zugehörigen deterministischen endlichen Automaten.

Welche Sprache wird durch den Automaten beschrieben?

Prüfe deine Vermutung mit dem Programm aus Übung 2.