Formale Sprachen

Inhalte

- Aufbau von Sprachen
- Grammatiken formaler Sprachen
- Endliche Automaten
- *Grenzen endlicher Automaten
- Werkzeuge

Lehrplan

Inf 12.1 Formale Sprachen (ca. 16 Std.)

Bisher kennen die Schüler Sprachen vor allem als Mittel zur Kommunikation zwischen Menschen. Ihnen ist bekannt, dass eindeutiges gegenseitiges Verstehen nur dann gewährleistet ist, wenn sich die Kommunikationspartner auf eine gemeinsame Sprache einigen und die zu deren Gebrauch festgelegten Regeln einhalten. Im Rückblick auf das bisherige Arbeiten mit dem Computer wird ihnen deutlich, dass die Verständigung zwischen Mensch und Maschine ebenfalls einen Kommunikationsprozess darstellt, der ähnlichen Regeln unterliegt. Daher betrachten sie zunächst den strukturellen Aufbau einer ihnen bereits bekannten natürlichen Sprache sowie den Aufbau einer künstlichen Sprache. Die Jugendlichen lernen dabei, die Begriffe Syntax und Semantik einer Sprache zu unterscheiden.

Anhand einfacher Beispiele wie Autokennzeichen, E-Mail-Adressen oder Gleitkommazahlen lernen die Schüler den Begriff der formalen Sprache als Menge von Zeichenketten kennen, die nach bestimmten Regeln aufgebaut sind. Zur Beschreibung dieser Regeln verwenden sie Textnotationen oder Syntaxdiagramme und können damit analog zu den natürlichen Sprachen Grammatiken für formale Sprachen definieren. Die Zweckmäßigkeit der streng formalen Beschreibung zeigt sich den Jugendlichen bei der automatischen Überprüfung der syntaktischen Korrektheit von Zeichenketten mithilfe von endlichen Automaten.

Den Schülern wird bewusst, dass nur Vorgänge, die sich mit Mitteln einer formalen Sprache ausdrücken lassen, von einem Computer bearbeitet werden können. Somit stoßen sie über die Theorie der formalen Sprachen auf eine prinzipielle Grenze des Computereinsatzes.

- einfache Beispiele für formale Sprachen über einem Alphabet; Zeichen, Zeichenvorrat (Alphabet), Zeichenkette
- Unterscheidung zwischen Syntax und Semantik, Vergleich zwischen natürlichen und formalen Sprachen
- syntaktischer Aufbau einer formalen Sprache: Grammatik (Terminal, Nichtterminal, Produktion, Startsymbol)
- Notation formaler Sprachen: Syntaxdiagramm, einfache Textnotation (z. B. Backus-Naur-Form)
- erkennender, endlicher Automat als geeignetes Werkzeug zur Syntaxprüfung für reguläre Sprachen; Implementierung eines erkennenden Automaten

Hinweis

Die Abbildungen sind dem Buch Informatik Oberstufe 1 entnommen und unterliegen dem Copyright des Oldenbourg Schulbuchverlags!

1 Aufbau von Sprachen

BANKS OF OHIO
TRADITIONAL 19TH CENTURY

COME MY LOVE LET'S TAKE A WALK
JUST A LITTLE WAY AWAY
WHILE WE WALK ALONG WE'LL TALK
TALK ABOUT OUR WEDDING DAY

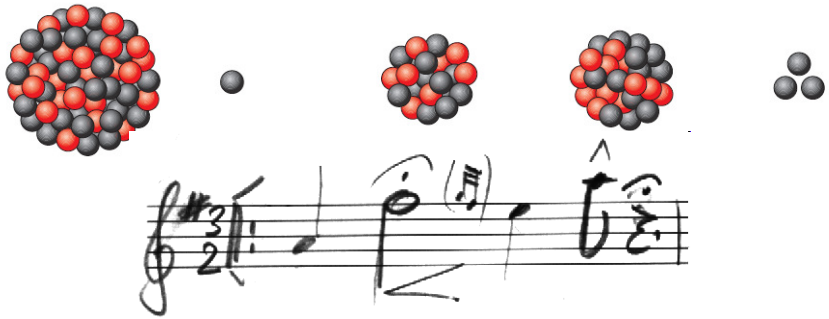
(CHORUS)
ONLY SAY THAT YOU'LL BE MINE
AND IN OUR HOME WE'LL HAPPY BE
DOWN BESIDE WHERE THE WATERS FLOW
DOWN ON THE BANKS OF THE OHIO
...

1 Aufbau von Sprachen

```
void Weiterschalten()
{
    if (ampelphase == "rot")
    {
        RotgelbSetzen();
    }
    else
    {
        if (ampelphase == "rotgelb")
        {
            GruenSetzen();
        }
        else
        {
            if (ampelphase == "gruen")
            {
                GelbSetzen();
            }
            else
            {
                RotSetzen();
            }
        }
    }
}
```

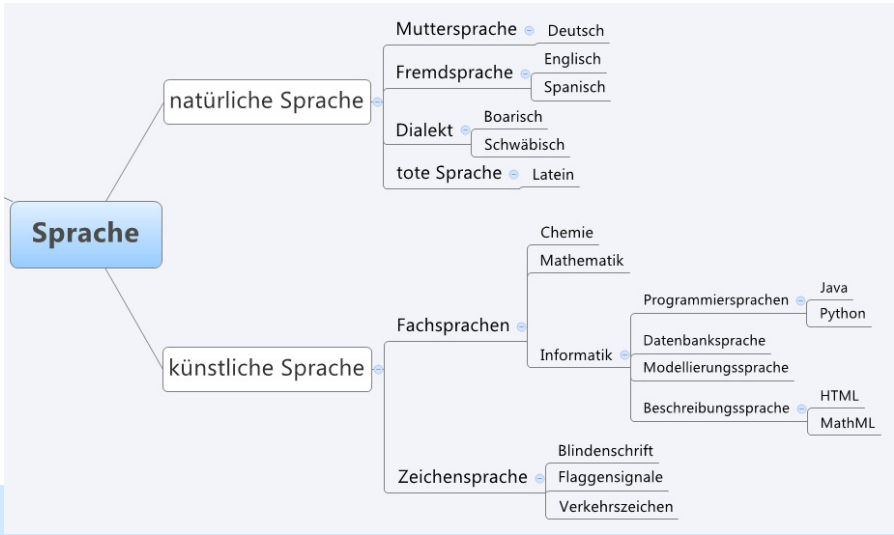
1 Aufbau von Sprachen

$$^{235}_{92}\mathrm{U} + {}^{1}_{0}\mathrm{n} \longrightarrow {}^{89}_{36}\mathrm{Kr}^{*} + {}^{144}_{56}\mathrm{Ba} + 3 \cdot {}^{1}_{0}\mathrm{n}$$



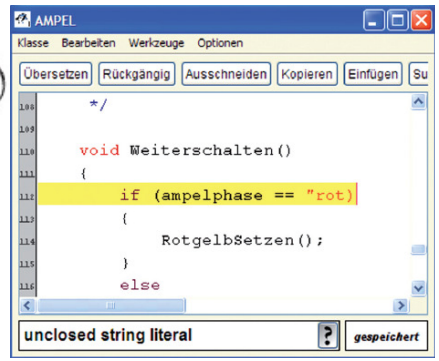
Welche Sprachen gibt es? Wie können sie klassifiziert werden?

1 Aufbau von Sprachen



1 Aufbau von Sprachen

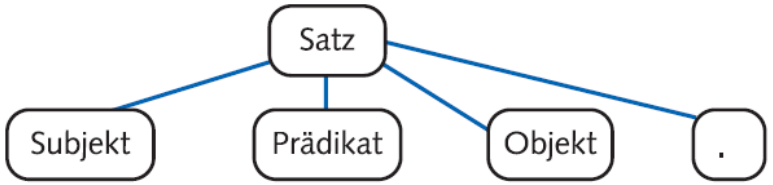
Welche Zweck haben Sprachen?



1 Aufbau von Sprachen

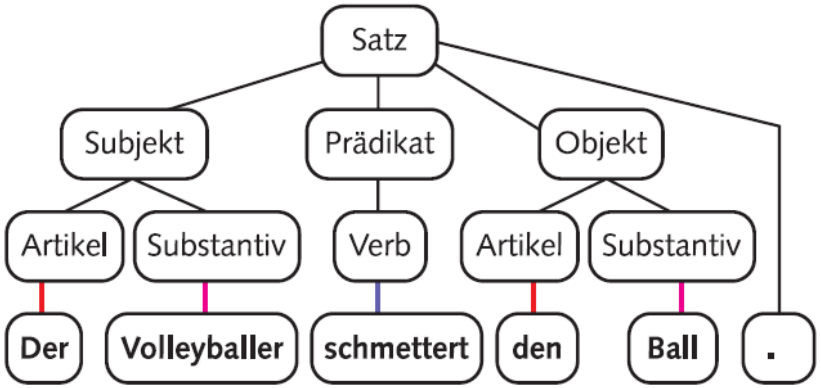
Wie ist Sprache aufgebaut?

R1 Satz → Subjekt Prädikat Objekt "."



1 Aufbau von Sprachen

Syntax



1 Aufbau von Sprachen

Semantik

1 Aufbau von Sprachen

formale Sprache

$A_5 = \{$ sin ; cos ; tan ; … $\}$

Zeichenkette der Länge 4 über dem Alphabet A_5

sin (x)

2 Grammatiken formaler Sprachen

jenni.zirbnich@simsoftlab.net

peter@cody@sim_soft_lab.at

jenni@zirb.nich.de jz@ssl.net

cody@-sim-soft-lab-.net

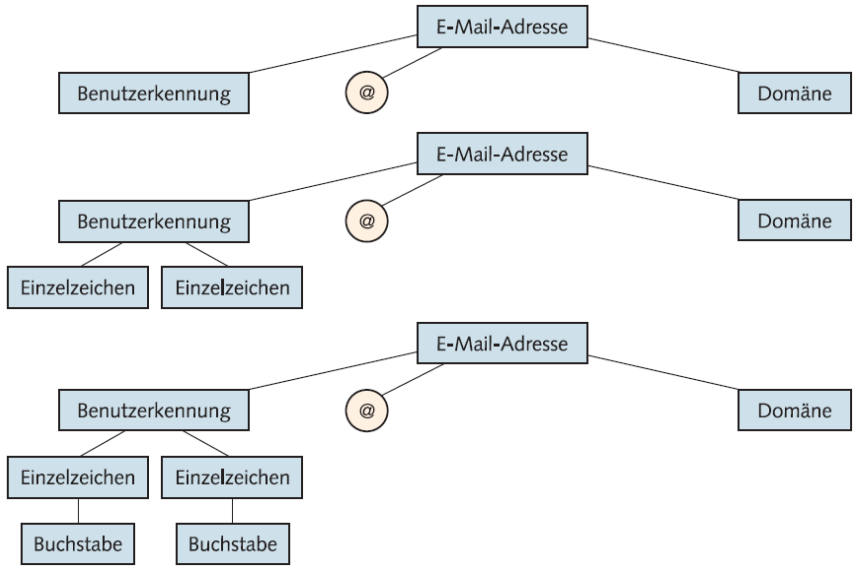
2 Grammatiken formaler Sprachen

R1 E-Mail-Adresse = Benutzerkennung "@" Domäne .
R2 Benutzerkennung = Einzelzeichen { Einzelzeichen } .
R3 Einzelzeichen = Buchstabe | Ziffer | "-" | "_" | "." | "!" .
R4 Ziffer = "0" | "1" | … | "9" .
R5 Buchstabe = "a" | "b" | … | "z" .
R6 Domäne = Unterdomäne { "." Unterdomäne } "." TLD .
R7 TLD = Buchstabe Buchstabe [Buchstabe] [Buchstabe] .
R8 Unterdomäne = (Buchstabe | Ziffer) { Buchstabe | Ziffer | "-" } (Buchstabe | Ziffer) .

2 Grammatiken formaler Sprachen

E-Mail-Adresse
↓ R1 — Begonnen wird mit dem Startsymbol.
Benutzerkennung "@" Domäne
↓ R2 — Bei der Wiederholung in R2 wird der Ausdruck { Einzelzeichen } durch Einzelzeichen ersetzt, sodass das Nichtterminal insgesamt zweimal auftritt.
Einzelzeichen Einzelzeichen "@" Domäne
↓ R3, R3 — Die Regel R3 wird zweimal angewandt.
Buchstabe Buchstabe "@" Domäne
↓ R5, R5 — Die Regel R5 wird zweimal angewandt.
"j" "z" "@" Domäne
↓ R6 — Da es nur eine Unterdomäne gibt, wird der Ausdruck

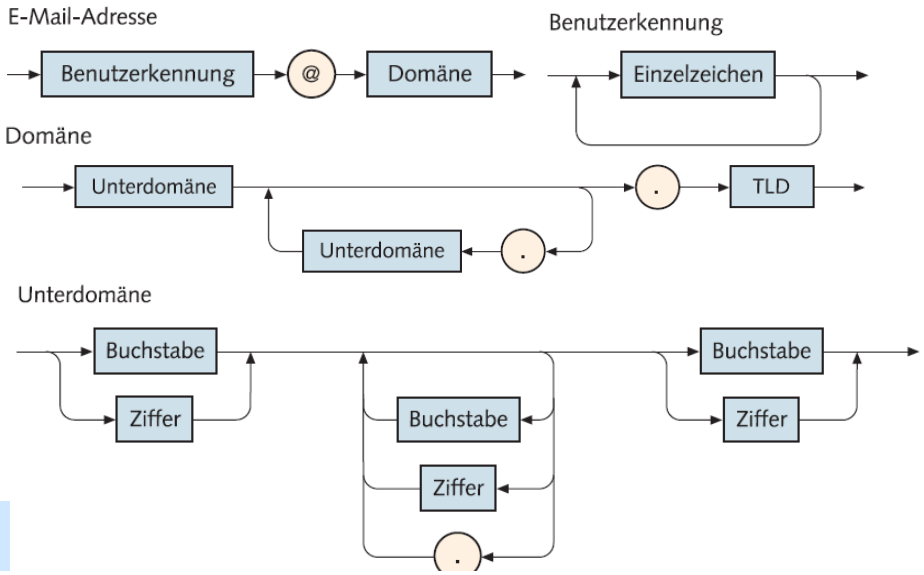
2 Grammatiken formaler Sprachen



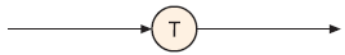
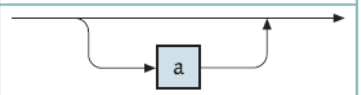
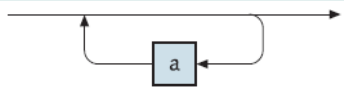
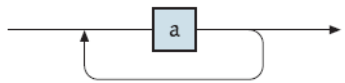
2 Grammatiken formaler Sprachen

Werkzeug AtoCC
kfG Edit

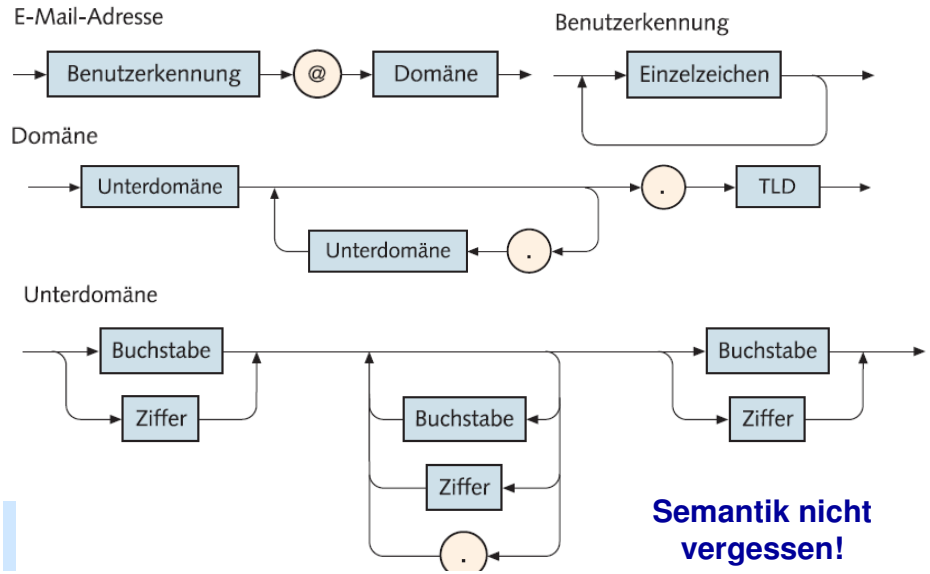
2 Grammatiken formaler Sprachen



2 Grammatiken formaler Sprachen

EBNF	Erläuterung	Syntaxdiagramm
Terminalsymbol T	Das Terminal ist von einer Ellipse umrandet.	
Nichtterminalsymbol N	Das Nichtterminal ist von einem Rechteck umrandet.	
Optionsklammer [a]	a kommt keinmal oder einmal vor.	
Wiederholungsklammer {a}	a kommt keinmal, einmal oder mehrfach vor.	
Variante der Wiederholungsklammer a {a}	a kommt mindestens einmal vor, d. h. einmal oder mehrfach.	

2 Grammatiken formaler Sprachen

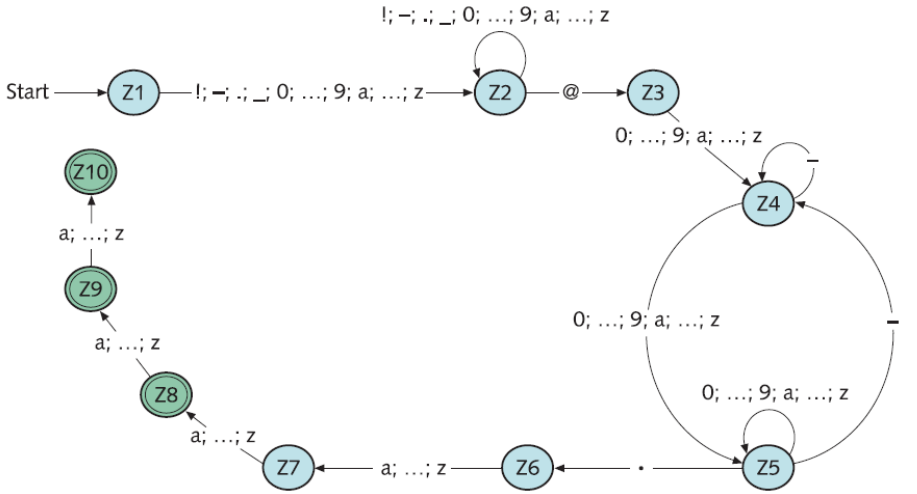


Semantik nicht vergessen!

2 Grammatiken formaler Sprachen

Werkzeug EBNF Visualizer

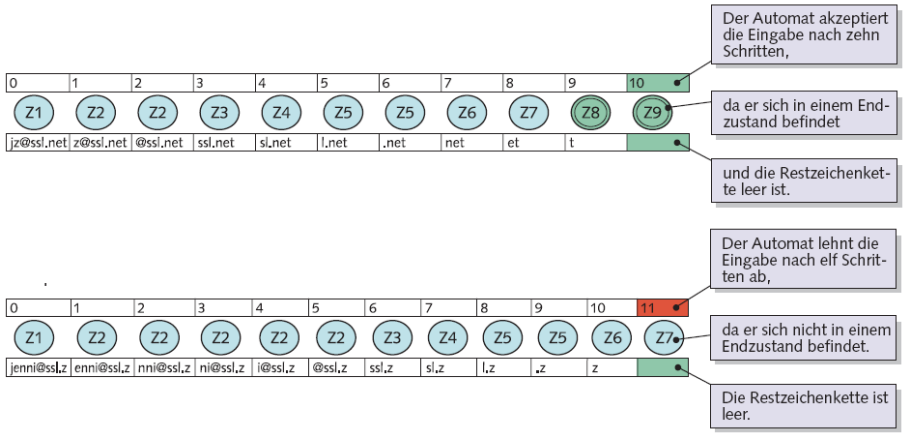
3 Endliche Automaten



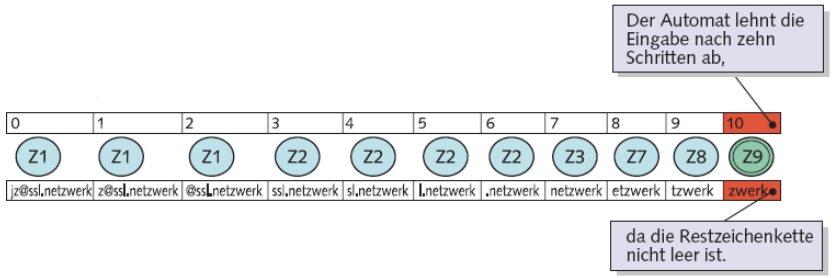
3 Endliche Automaten

aktueller Zustand	zu überprüfende Restzeichenkette	erstes Zeichen in der Rest-zeichenkette	passender Zustandsübergang und Bemerkung
Z1	jz@ssl.net	j	Z1 —!; –; .; _; 0; …; 9; a; …; z→ Z2 Der neue Zustand ist Z2. Die zu überprüfende Restzeichenkette wird um das eingelesene j zu z@ssl.net verkürzt.
Z2	z@ssl.net	z	Der Zustand bleibt Z2. Die zu überprüfende Restzeichenkette wird um das eingelesene z zu @ssl.net verkürzt. !; –; .; _; 0; …; 9; a; …; z → Z2
Z2	@ssl.net	@	Z2 —@→ Z3 Der neue Zustand ist Z3. Die zu überprüfende Restzeichenkette wird um das eingelesene @ zu ssl.net verkürzt.

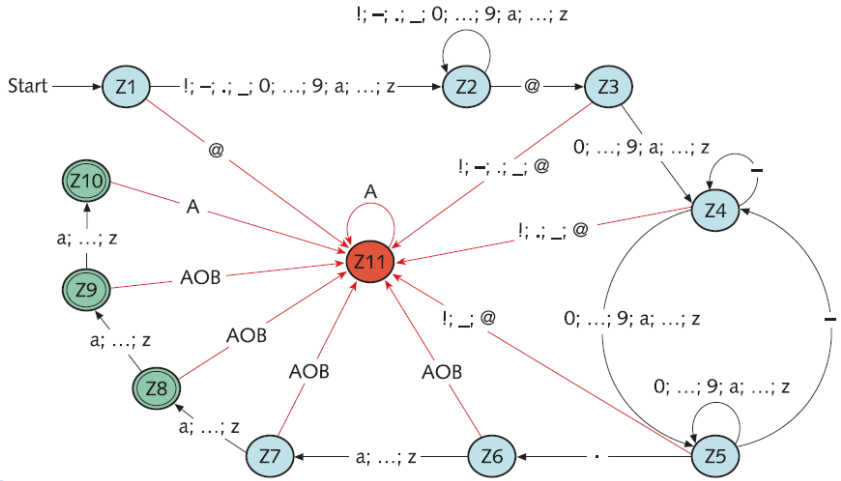
3 Endliche Automaten



3 Endliche Automaten

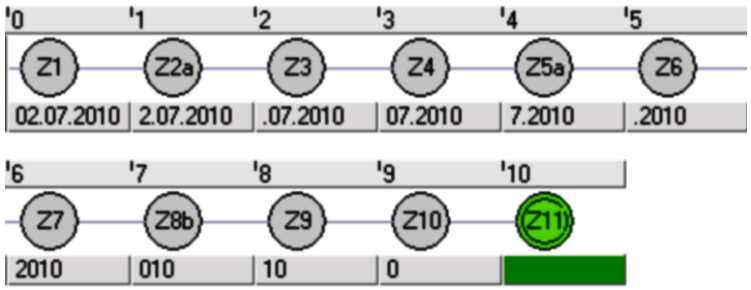


3 Endliche Automaten



2 Grammatiken formaler Sprachen

Werkzeug AtoCC
AutoEdit



3 Endliche Automaten

Implementierung eines endlichen Automaten
1) Eingrenzen der möglichen Zustandsübergänge aufgrund des aktuellen Zustands – nur die Zustandsübergänge kommen in Betracht, die vom aktuellen Zustand ausgehen.
2) Durchführen des Zustandsübergangs abhängig vom aktuellen Zeichen.

Z1	Z2	…	Z11
Z1Zeicheneingabe Bearbeiten(zeichen)	Z2Zeicheneingabe Bearbeiten(zeichen)	…	Z11Zeicheneingabe Bearbeiten(zeichen)

10 Struktogramm der Methode ZeicheneingabeWeiterleiten(char zeichen)

3 Endliche Automaten

Implementierung eines endlichen Automaten
1) Eingrenzen der möglichen Zustandsübergänge aufgrund des aktuellen Zustands – nur die Zustandsübergänge kommen in Betracht, die vom aktuellen Zustand ausgehen.
2) Durchführen des Zustandsübergangs abhängig vom aktuellen Zeichen.

'@'	'_'	…	'z'	sonst
aktuellerZustand = 11	aktuellerZustand = 2	…	aktuellerZustand = 2	konsole.Ausgeben ("Unerlaubtes Zeichen wurde eingegeben.") aktuellerZustand = 11

11 Struktogramm der Methode void Z1ZeicheneingabeBearbeiten(char zeichen)

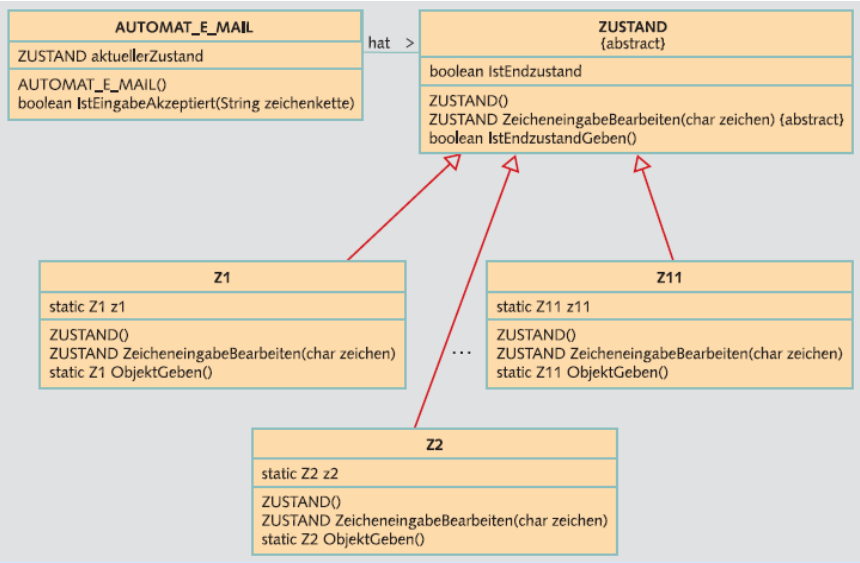
3 Endliche Automaten

Klasse AUTOMAT_E_MAIL

Methode boolean IstEingabeAkzeptiert(String zeichenkette)

```
// lokales Attribut zur Speicherung der noch zu überprüfenden Zeichen
String restzeichenkette = zeichenkette

// Abarbeitung der Zeichenkette Zeichen für Zeichen
zähle i von 1 bis zeichenkette.länge()
   ZeicheneingabeWeiterleiten(restzeichenkette.ZeichenGebenAnDerStelle(0))
   // die Restzeichenkette wird um das vorderste Zeichen verkürzt
   restzeichenkette = restzeichenkette.TeilzeichenketteGebenAbDerStelle(1)
endezähle

// Überprüfung, ob sich der Automat in einem Endzustand befindet
wenn (aktuellerZustand == 8 || aktuellerZustand == 9 || aktuellerZustand == 10)
   dann
      return true
   sonst
      return false
endewenn
```


* Entwurfsmuster Zustand



* Entwurfsmuster Zustand

'@'	'-'	':'	...
return Z11.ObjektGeben()	return Z2.ObjektGeben()	return Z2.ObjektGeben()	...

'z'	sonst
return Z2.ObjektGeben()	konsole.Ausgeben ("Unerlaubtes Zeichen wurde eingegeben.") return Z11.ObjektGeben()

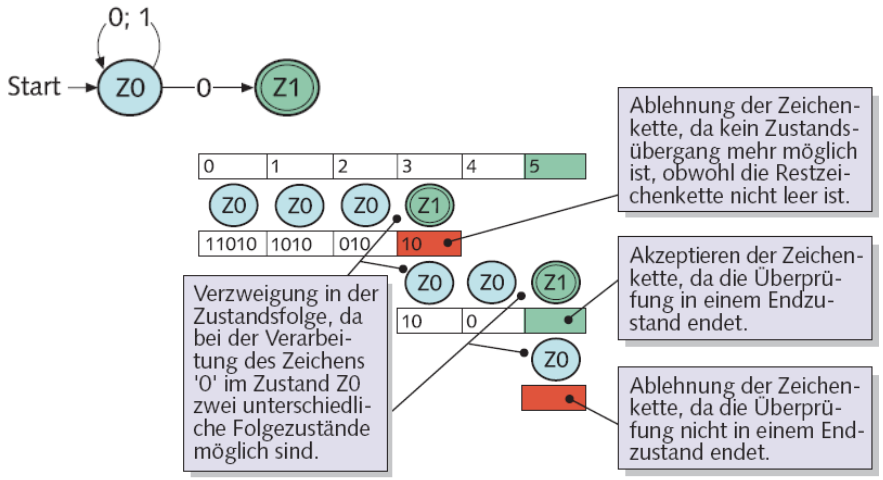
* Entwurfsmuster Zustand

Klasse AUTOMAT_E_MAIL

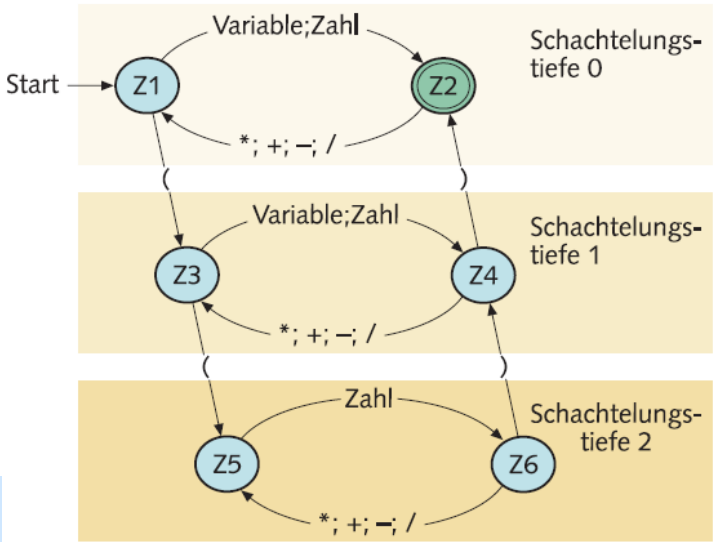
Methode boolean IstEingabeAkzeptiert(String zeichenkette)

```
// lokales Attribut zur Speicherung der noch zu überprüfenden Zeichen
String restzeichenkette = zeichenkette

// Abarbeitung der Zeichenkette Zeichen für Zeichen
zähle i von 1 bis zeichenkette.länge()
   aktuellerZustand = aktuellerZustand.
        ZeicheneingabeBearbeitung(restzeichenkette.ZeichenGebenAnDerStelle(0))
   // die Restzeichenkette wird um das vorderste Zeichen verkürzt
   restzeichenkette = restzeichenkette.TeilzeichenketteGebenAbDerStelle(1)
endezähle

// Überprüfung, ob sich der Automat in einem Endzustand befindet
wenn (aktuellerZustand.IstEndzustandGeben())
   dann
      return true
   sonst
      return false
endewenn
```

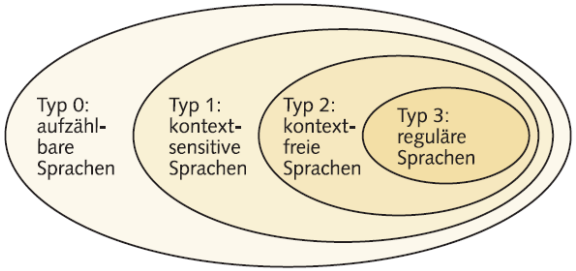
* Nichtdet. endl. Automaten



*4 Grenzen endlicher Automaten



*4 Grenzen endlicher Automaten



Grammatikart	Sprachtyp	akzeptierende Maschine
unbeschränkt	aufzählbare Sprachen	Turingmaschine
kontextsensitiv	kontextsensitive Sprachen	eine lineare beschränkte Turingmaschine
kontextfrei	kontextfreie Sprachen	Kellerautomat
rechtslinear	reguläre Sprachen	endlicher Automat

Vielen Dank für
Ihre Aufmerksamkeit!