Formale Sprachen

M. Jakob

Gymnasium Pegnitz

13. Oktober 2019

Inhaltsverzeichnis

Natürliche und künstliche Sprachen

- natürliche Sprachen (Deutsch, Englisch, Latein, . . .)
  meist historisch entwickelt, äußerst komplex
- künstliche Sprachen
  für spezielle Fachgebiete konstruiert, möglichst systematisch
  - Autokennzeichen
    

  - chemische Reaktionsgleichungen

    $$^{235}_{92}U + {}^{1}_{0}n \longrightarrow {}^{89}_{36}Kr + {}^{144}_{56}Ba + 3\,{}^{1}_{0}n$$
  - mathematische Terme

    $3x^2 - 7x + 3$
    $A \sin(\omega t + \varphi)$

Bestandteile einer Sprache

Jede künstliche und natürliche Sprache enthält . . .

Bestandteile einer Sprache

Jede künstliche und natürliche Sprache enthält ...

- ein Alphabet, aus dem die Wörter und Sätze der Sprache zusammengestellt werden.
- eine Syntax, die angibt, welche Wörter, Ausdrücke und Sätze zur Sprache gehören.
- eine Semantik, die festlegt, was die einzelnen Ausdrücke bedeuten.

Bestandteile einer Sprache

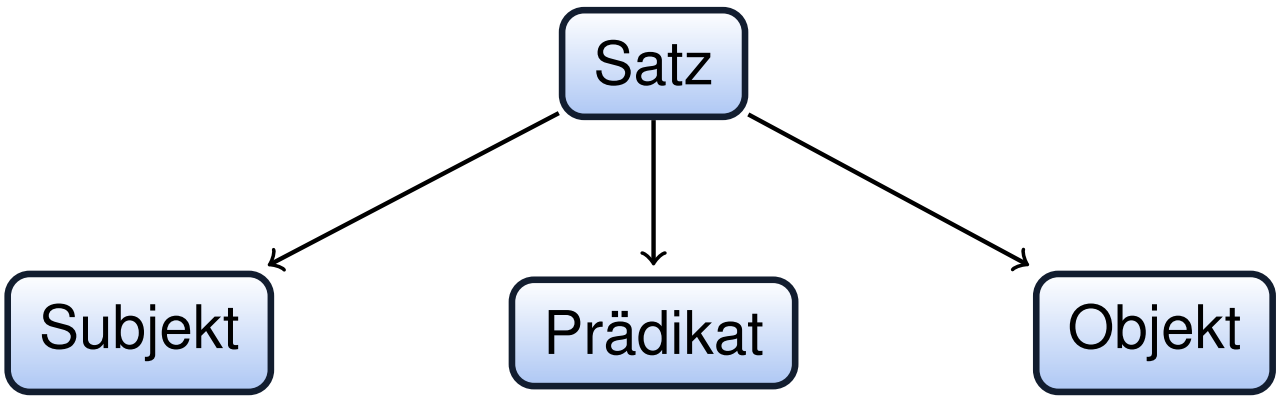
Jede künstliche und natürliche Sprache enthält ...

- ein Alphabet, aus dem die Wörter und Sätze der Sprache zusammengestellt werden.
- eine Syntax, die angibt, welche Wörter, Ausdrücke und Sätze zur Sprache gehören.
- eine Semantik, die festlegt, was die einzelnen Ausdrücke bedeuten.

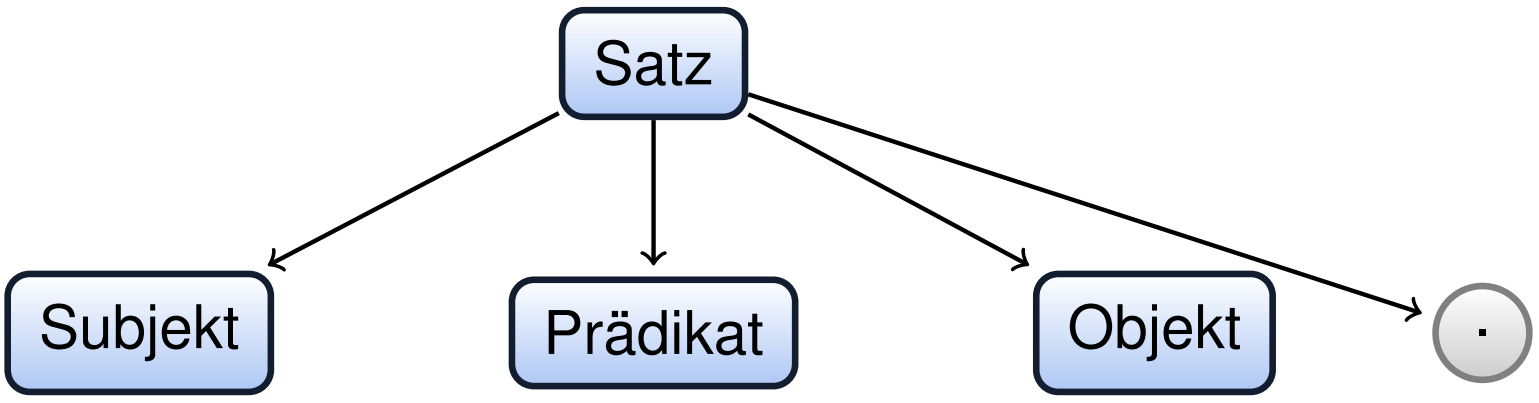
Die Grammatik stellt in diesem Zusammenhang ein Regelwerk dar, wie erreicht wird, dass ein Ausdruck syntaktisch korrekt ist.

Syntaxbaum (Ableitungsbaum) einer Grammatik der deutschen Sprache

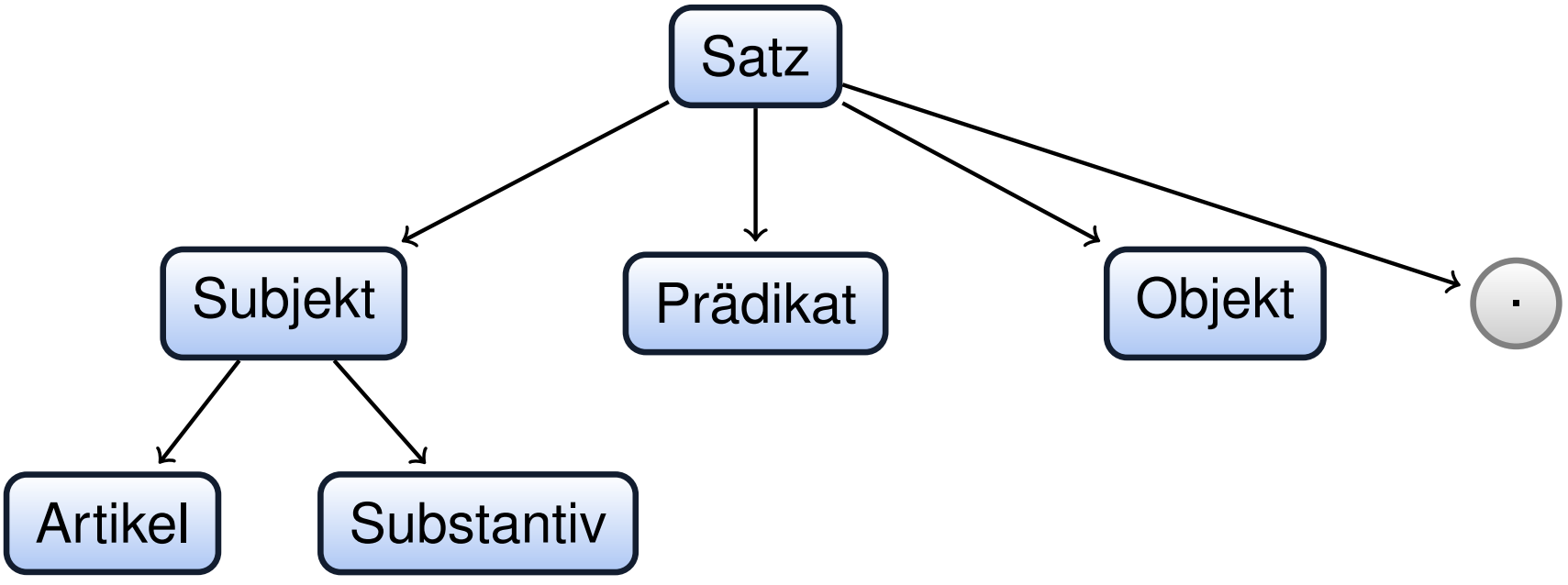
Syntaxbaum (Ableitungsbaum) einer Grammatik der deutschen Sprache



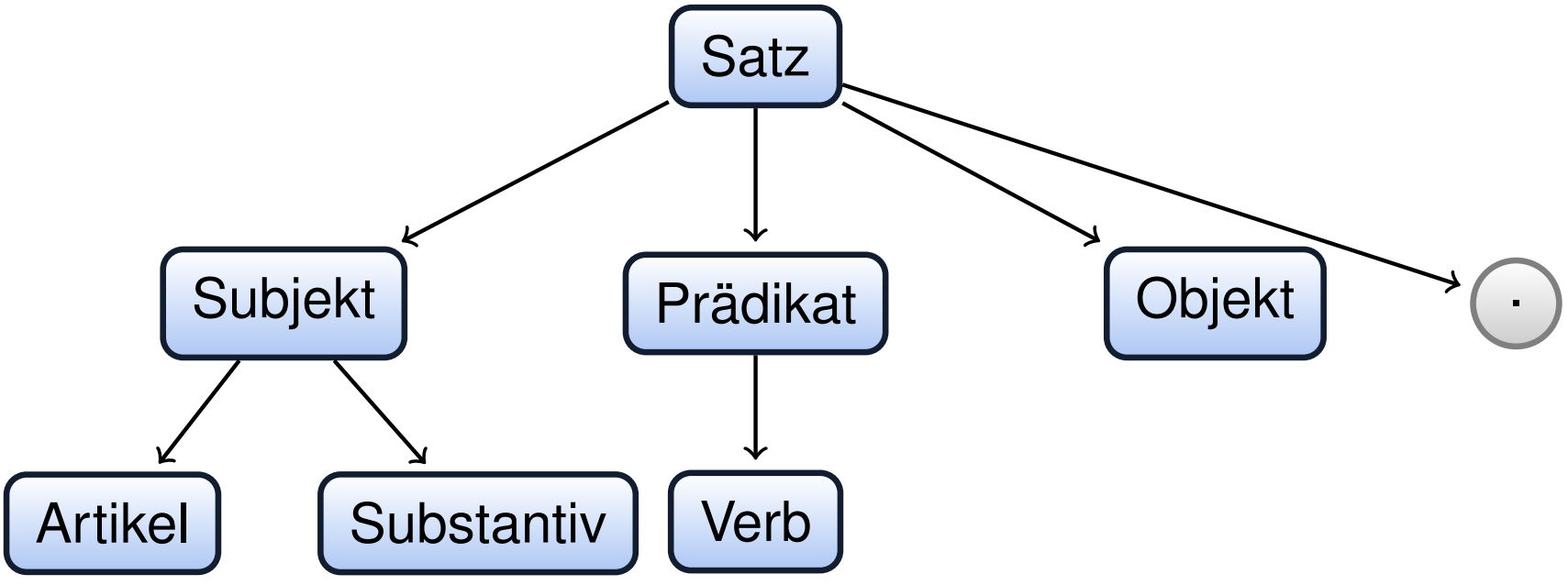
Syntaxbaum (Ableitungsbaum) einer Grammatik der deutschen Sprache



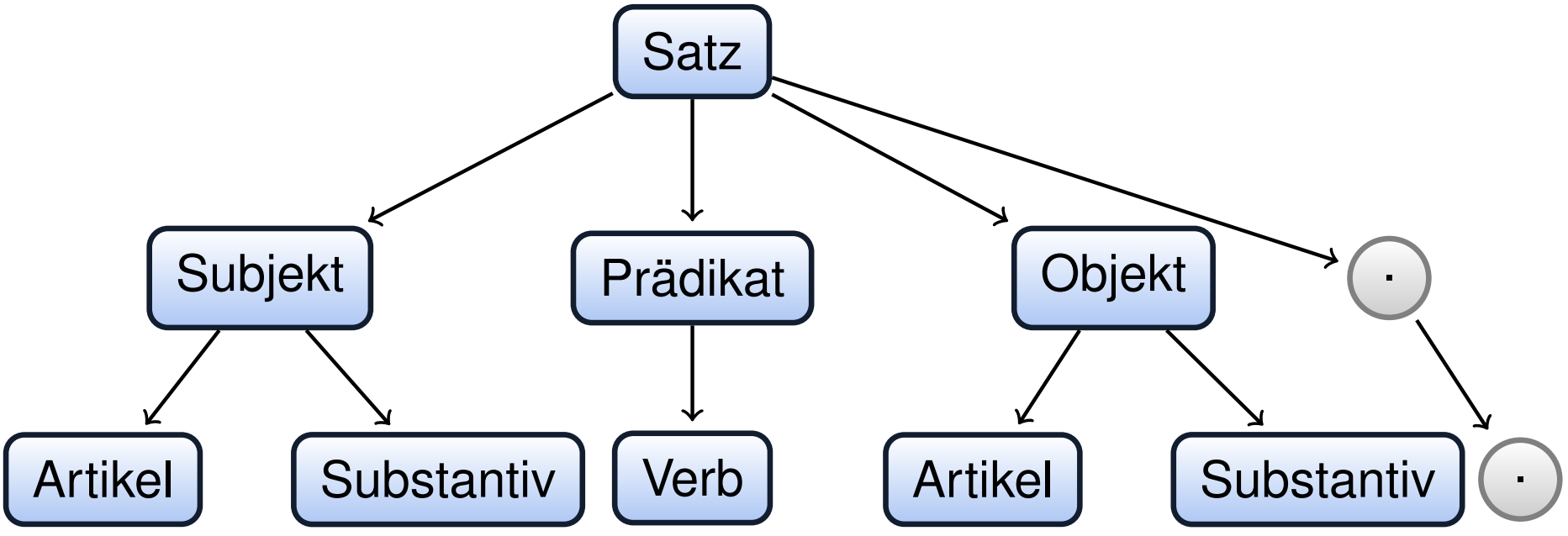
Syntaxbaum (Ableitungsbaum) einer Grammatik der deutschen Sprache



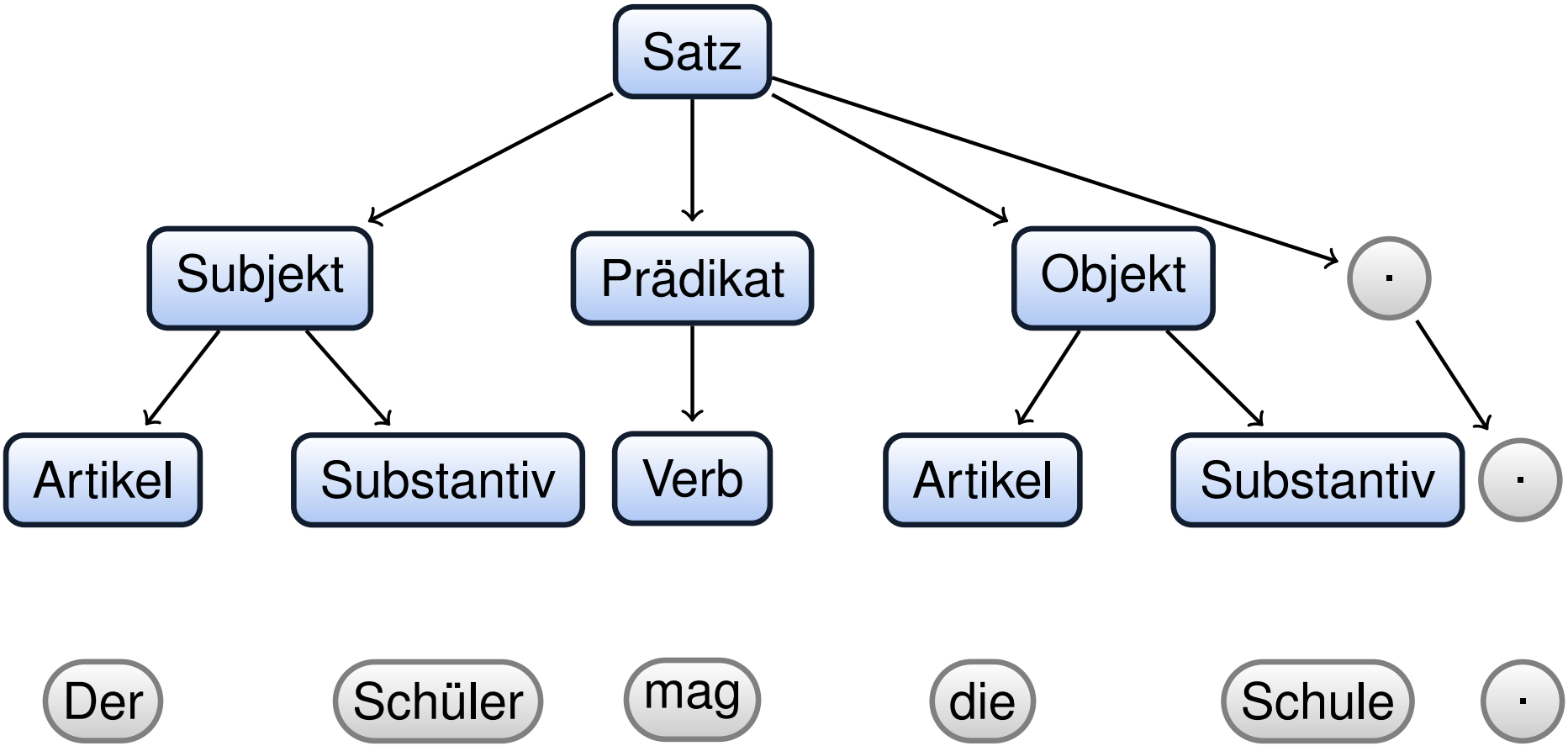
Syntaxbaum (Ableitungsbaum) einer Grammatik der deutschen Sprache



Syntaxbaum (Ableitungsbaum) einer Grammatik der deutschen Sprache



Syntaxbaum (Ableitungsbaum) einer Grammatik der deutschen Sprache



Syntaxbaum in Regeln gefasst

Regeln der Grammatik

R1	Satz	→	Subjekt Prädikat Objekt ".“
R2	Subjekt	→	Substantiv | Artikel Substantiv
R3	Prädikat	→	Verb
R4	Objekt	→	Substantiv | Artikel Substantiv
R5	Artikel	→	“der“ | “den“ | “einen“
R6	Substantiv	→	“Volleyballer“ | “Ball“ | “Schüler“ | “Pass“ | “Saxofon“
R7	Verb	→	“spielt“ | “mag“

Zusammenfassung

Regeln, die auf Platzhalter (bestehend aus **Nichtterminalsymbolen**) enden

R1	Satz	→	Subjekt Prädikat Objekt "."
R2	Subjekt	→	Substantiv | Artikel Substantiv
R3	Prädikat	→	Verb
R4	Objekt	→	Substantiv | Artikel Substantiv

Regeln, die auf Wörtern (**Terminalsymbolen**) enden

R5	Artikel	→	“der“ | “den“ | “einen“
R6	Substantiv	→	“Volleyballer“ | “Ball“ | “Schüler“ | “Pass“ | “Saxofon“
R7	Verb	→	“spielt“ | “mag“

Unterschied zwischen Syntax und Semantik

Mit dieser Grammatik lassen sich folgende syntaktisch richtigen Sätze bilden:

der Schüler mag den Schule.

der Volleyballer spielt den Schüler.

der Schüler spielt Saxofon.

Unterschied zwischen Syntax und Semantik

Mit dieser Grammatik lassen sich folgende syntaktisch richtigen Sätze bilden:

der Schüler mag den Schule.

der Volleyballer spielt den Schüler.

der Schüler spielt Saxofon.

Dabei ist nur der letzte Satz semantisch korrekt.

Übung

Ü 1.1: Buch S.14 / 5
Ü 1.2: Buch S.14 / 6
Ü 1.3: Buch S.14 / 7

Bestandteile einer Grammatik

Definition (Grammatik)

Eine Grammatik (V, Σ, P, S) besteht aus ...

- Einer Menge V von Nichtterminalsymbolen (Variablen),
- Einer Menge Σ von Terminalsymbolen (Alphabet).
- Einer Menge P von Produktionen (auch Regeln genannt).
- Einem Startsymbol S aus der Menge der Nichtterminalsymbole.

Aufbau formaler Sprachen — Bemerkungen

- Σ und V dürfen kein gemeinsames Element besitzen ($\Sigma \cap V = \{\}$).
- Die Grammatik legt eine formale Sprache fest, deren Worte nur aus Terminalsymbolen aus Σ bestehen und die ausgehend vom Startsymbol S mithilfe einer endlichen Anzahl von Anwendungen von Regeln aus P abgeleitet werden können.

Bespiel E-Mail-Adresse: groessterDepp@web.de

R1 E-Mail-Adresse = Benutzerkennung "@"Domäne

Bespiel E-Mail-Adresse: groessterDepp@web.de

R1 E-Mail-Adresse = Benutzerkennung "@"Domäne

R2 Benutzerkennung = Einzelzeichen {Einzelzeichen}

Bespiel E-Mail-Adresse: groessterDepp@web.de

R1 E-Mail-Adresse = Benutzerkennung "@"Domäne

R2 Benutzerkennung = Einzelzeichen {Einzelzeichen}

R3 Domäne = Unterdomäne {"."Unterdomäne} "."TLD

Bespiel E-Mail-Adresse: groessterDepp@web.de

R1 E-Mail-Adresse = Benutzerkennung "@"Domäne

R2 Benutzerkennung = Einzelzeichen {Einzelzeichen}

R3 Domäne = Unterdomäne {"."Unterdomäne} "."TLD

R4 TLD = Buchstabe Buchstabe [Buchstabe] [Buchstabe]

Bespiel E-Mail-Adresse: groessterDepp@web.de

R1 E-Mail-Adresse = Benutzerkennung "@"Domäne

R2 Benutzerkennung = Einzelzeichen {Einzelzeichen}

R3 Domäne = Unterdomäne {"."Unterdomäne} "."TLD

R4 TLD = Buchstabe Buchstabe [Buchstabe] [Buchstabe]

R5 Unterdomäne = (Buchstabe | Ziffer) {Buchstabe | Ziffer | "-"} (Buchstabe | Ziffer)

Bespiel E-Mail-Adresse: groessterDepp@web.de

R1 E-Mail-Adresse = Benutzerkennung "@"Domäne

R2 Benutzerkennung = Einzelzeichen {Einzelzeichen}

R3 Domäne = Unterdomäne {"."Unterdomäne} "."TLD

R4 TLD = Buchstabe Buchstabe [Buchstabe] [Buchstabe]

R5 Unterdomäne = (Buchstabe | Ziffer) {Buchstabe | Ziffer | "-"} (Buchstabe | Ziffer)

R6 Einzelzeichen = Buchstabe | Ziffer | "-"| "_"| "."| "!"

Bespiel E-Mail-Adresse: groessterDepp@web.de

R1 E-Mail-Adresse = Benutzerkennung "@"Domäne

R2 Benutzerkennung = Einzelzeichen {Einzelzeichen}

R3 Domäne = Unterdomäne {"."Unterdomäne} "."TLD

R4 TLD = Buchstabe Buchstabe [Buchstabe] [Buchstabe]

R5 Unterdomäne = (Buchstabe | Ziffer) {Buchstabe | Ziffer | "-"} (Buchstabe | Ziffer)

R6 Einzelzeichen = Buchstabe | Ziffer | "-"| "_"| "."| "!"

R7 Ziffer = "0 | "1"| ... | "9"

Bespiel E-Mail-Adresse: groessterDepp@web.de

R1 E-Mail-Adresse = Benutzerkennung "@"Domäne

R2 Benutzerkennung = Einzelzeichen {Einzelzeichen}

R3 Domäne = Unterdomäne {"."Unterdomäne} "."TLD

R4 TLD = Buchstabe Buchstabe [Buchstabe] [Buchstabe]

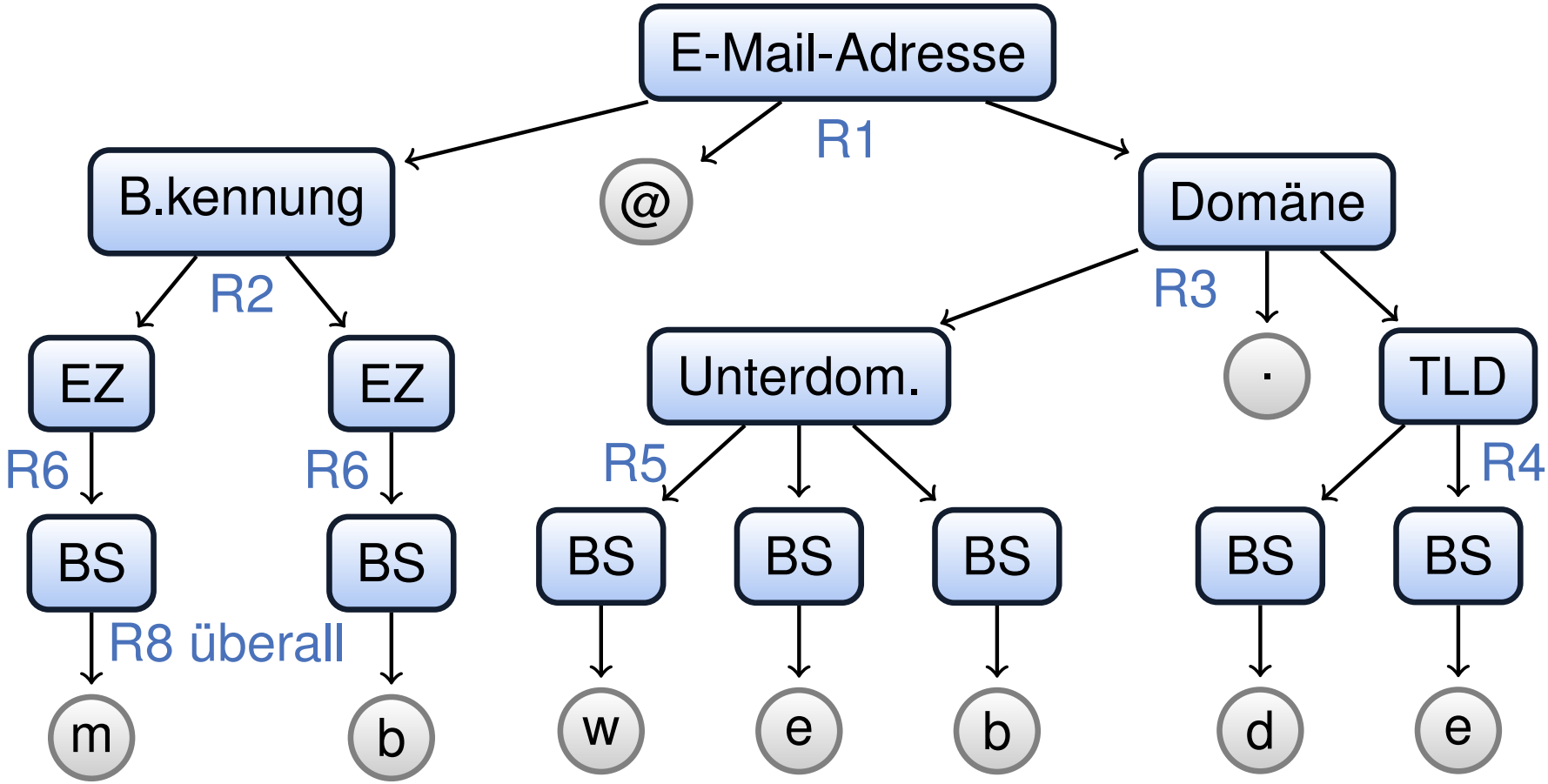
R5 Unterdomäne = (Buchstabe | Ziffer) {Buchstabe | Ziffer | "-"} (Buchstabe | Ziffer)

R6 Einzelzeichen = Buchstabe | Ziffer | "-"| "_"| "."| "!"

R7 Ziffer = "0 | "1"| ... | "9"

R8 Buchstabe = "a"| "b"| "c"| ... | "z"

Ableitungsbaum für eine E-Mail-Adresse



In diesem Abschnitt

Die Backus-Naur-Form (BNF)

Wie Syntaxregeln aufgeschrieben werden, legt z.B. die Backus-Naur-Form fest.
Beispiel:

```
1 <Ziffer>                  ::= 0 | <ZifferNNull>
2 <ZweistZahl>              ::= <ZifferNNull> <Ziffer>
3 <Zehn bis Neunzehn>       ::= 1 <Ziffer>
4 <Zweiundvierzig>          ::= 42
```

Die Backus-Naur-Form

hat zwei große Nachteile: Sie

- ist nicht standardisiert (viele unterschiedliche Dialekte).
- besitzt keine Symbole für optionale Zeichen und Wiederholungen.

Erweiterte Backus-Naur-Form (EBNF)

EBNF

Unsere bisherigen Beispiele verwendet einige Schreibvereinfachungen, die in der Erweiterten Backus-Naur-Form (EBNF) standardisiert sind.

- Das Zeichen | steht für eine Alternative (eines davon).
- Die Klammern [] stehen für eine Option (kann, muss aber nicht).
- Die Klammern {} stehen für eine Wiederholung (beliebig oft - auch keinmal).
- Die Klammern () stehen für logische Gruppierungen.
- Die Ableitungspfeile werden durch =-Zeichen ersetzt.
- Am Ende einer Anweisung stets ein „Ende“-Zeichen (z.B. ;)

Hinweis ∗ als Wiederholungszeichen

In diesem Abschnitt

Syntaxdiagramme

Syntaxdiagramme

stellen die Regeln einer Grammatik grafisch dar. Für jede Regel gibt es ein eigenes Syntaxdiagramm.
Terminale werden durch Kreise, Nichtterminale durch Rechtecke repräsentiert. Verfolgt man einen Pfad vom Start- zum Endpfeil, so erhält man die entsprechenden Regel der Grammatik.

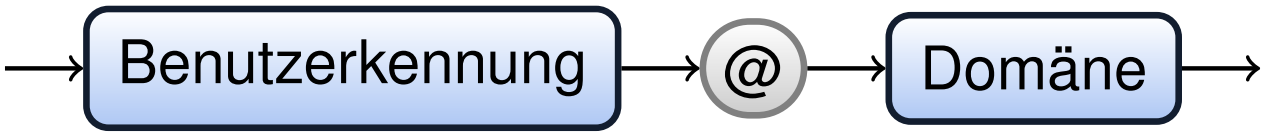
Beispiele

EBNF

E-Mail-Adresse = Benutzerkennung ’@’ Domäne

Syntaxdiagramm

E-Mail-Adresse:



Beispiele

EBNF

Benutzerkennung = Einzelzeichen {Einzelzeichen}

Syntaxdiagramm

Benutzerkennung:

Beispiele

EBNF

Domäne = Unterdomäne {’.’ Unterdomäne} ’.’ TLD

Syntaxdiagramm

Domäne:

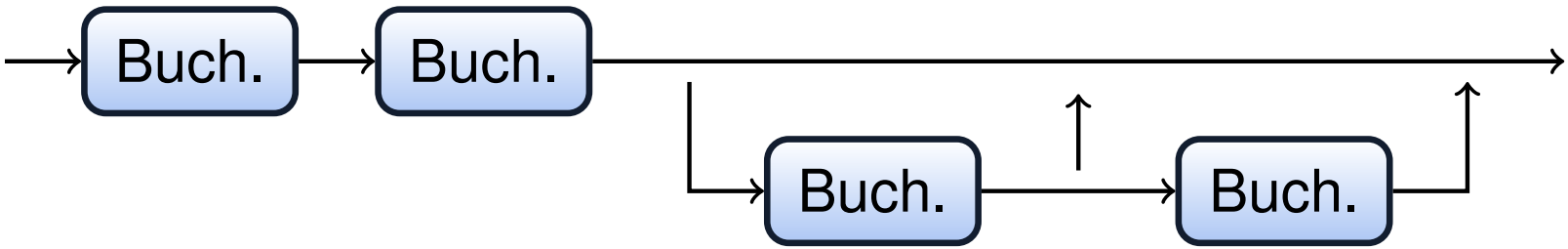
Beispiele

EBNF

TLD = Buchstabe Buchstabe [Buchstabe] [Buchstabe]

Syntaxdiagramm

TDL:



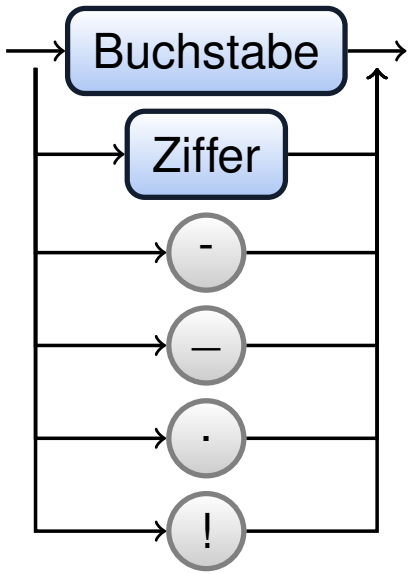
Beispiele

EBNF

Einzelzeichen = Buchstabe | Ziffer | ’-’ | ’_’ | ’.’ | ’!’

Syntaxdiagramm

Einzelzeichen:



Übung

Überblick Syntaxdiagramme S. 24ff
Ü 3.1: S. 28/1 (gem.)
Ü 3.2: S. 28/2
Ü 3.3: S. 29/5

Übung

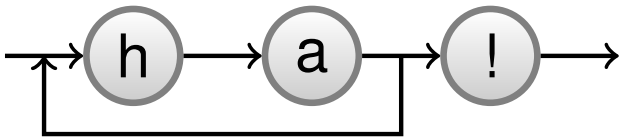
Ü 3.4: S. 29/6
Ü 3.5: S. 29/7a
Ü 3.6: S. 31/10
Ü 3.7: Abi 2012, III,1a und smilies mit EBNF-Visualizer oder Online-RRG

➥EBNF-Visualizer
➥Online-RRG

Beispiel: Lachautomat

Syntaxdiagramm

Lach:



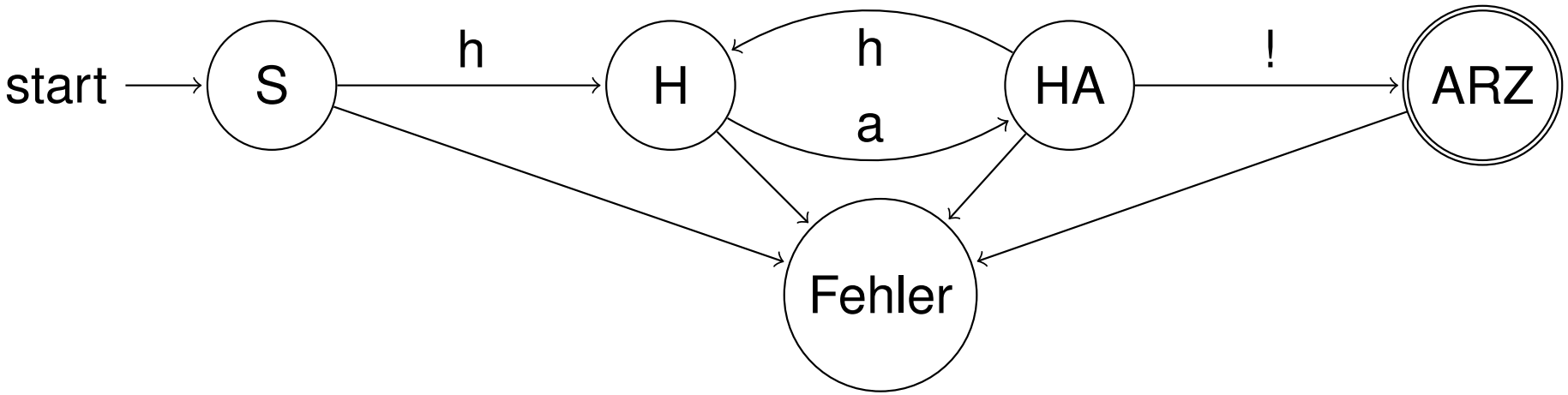
gesucht

Algorithmus, der erkennt, ob ein Wort zu einer Grammatik gehört, oder nicht.

an Tafel in DFA konvertieren

Beispiel: Der Lachautomat

Zustandsdiagramm



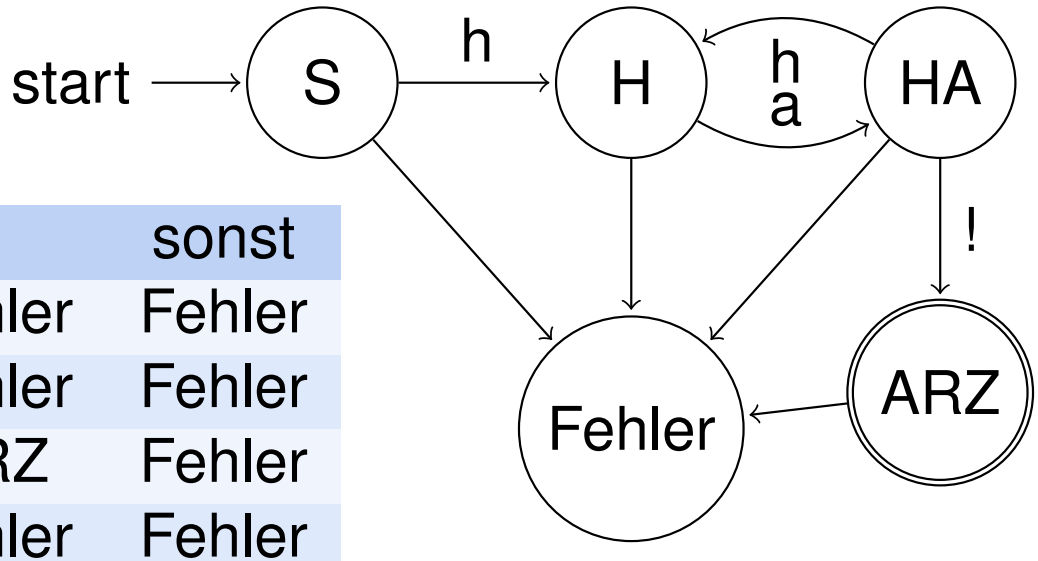
- Jeder Automat muss einen Startzustand haben (hier S).
- Wenn ein bestimmtes Zeichen eingelesen wird, springt der Automat evtl. in einen anderen Zustand.
- Ist ein Ausdruck syntaktisch korrekt, wird er akzeptiert, d.h. der Automat springt in den Endzustand (hier ARZ), wenn nicht , in einen Fehlerzustand, aus dem kein Übergang mehr herausführen darf.

Übung

Ü 4.1: Lachautomat mit AutoEdit von AtoCC
Ü 4.2: S. 41/2
Ü 4.3: S. 41/5
➥Ü 4.4: Exorciser

Die Übergangsfunktion — Matrixdarstellung

Die Übergangsfunktion lässt sich als Matrix darstellen, in der für jeden Zustand angegeben wird, durch welche eingelesenen Zeichen, welche Zustandsübergänge ausgelöst werden.

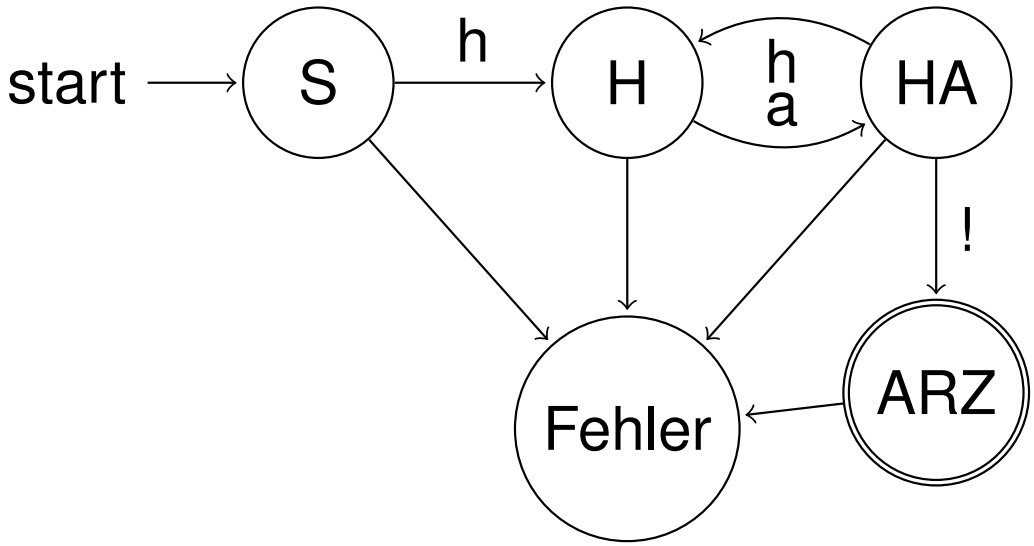


	h	a	!	sonst
S	H	Fehler	Fehler	Fehler
H	Fehler	HA	Fehler	Fehler
HA	H	Fehler	ARZ	Fehler
ARZ	Fehler	Fehler	Fehler	Fehler
Fehler	Fehler	Fehler	Fehler	Fehler

Die Übergangsfunktion — Darstellung mittels Funktionsaufruf

Die Übergangsfunktion δ lässt mittels Funktionsaufrufe darstellen. Eingabeparameter sind Tupel von Zuständen und Zeichen, Ausgabeparameter die dazugehörigen Zielzustände.

$$
\begin{aligned}
\delta(S; h) &= H \\
\delta(H; a) &= HA \\
\delta(HA; h) &= H \\
\delta(HA; !) &= ARZ \\
\delta(\text{sonst}; \text{sonst}) &= \text{Fehler}
\end{aligned}
$$



Definition Endlicher Automat

Definition

Ein erkennender endlicher Automat besteht aus folgenden Teilen:

- Eine Menge der Zustände: $Z = \{S, H, HA, ARZ, Fehler\}$
- Eine Menge der Eingabezeichen (Alphabet): $T = \{h, a, !\}$
- Eine Übergangsfunktion: $\delta : Z \times T \longrightarrow Z$
  Das bedeutet: Wenn man einen Zustand mit einem Eingabezeichen kombiniert, ergibt sich wieder ein (neuer) Zustand.
- Ein Startsymbol: $S \in Z$
- Eine Menge von Endzuständen: $F \subset Z$, $F = \{ARZ\}$

Ein Wort gilt als erkannt, wenn sich der Automat nach Einlesen aller Eingabezeichen in einem Endzustand befindet.

Grenzen endlicher Automaten

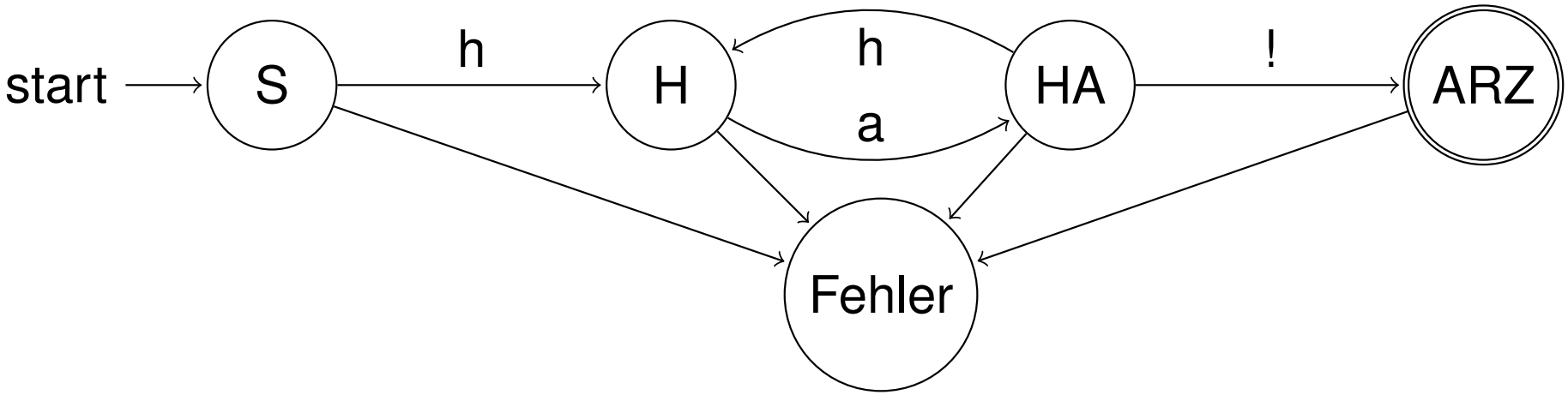
Es lässt sich zeigen:

Zu jedem endlichen Automaten existiert eine wie oben definierte Grammatik aber nicht umgekehrt.

Beispiel:
Multiklammern:

S = (S) | a

Beispiel: Lachautomat-Implementation



Idee:

- Es wird eine Variable `Zustand` definiert.
- Das zu testende Wort wird buchstabenweise ausgewertet und die Zustandsvariable entsprechend geändert.
- Befindet sich die Zustandsvariable am am Ende im Endzustand gehört das eingegeben Wort zur Grammatik.

Lachautomat — Variablendeklaration

```
1    private String eingabe;
2    private int zustand;
```

XX

Lachautomat — Zustandsdeklaration

```
1  // Codierung der Zustände als Konstanten
2  private final int START  = 0;
3  private final int H      = 1;
4  private final int HA     = 2;
5  private final int ARZ    = 3;
6  private final int FEHLER = 99;
7
8  // Menge der Endzustände
9  private int F[] = { ARZ };
```

Lachautomat — Konstruktor und Parser starten

```
1       public Lachautomat() { }
2
3       public void setEingabe(String e)
4       {
5           eingabe = e;
6           parsen();
7       }
```

Lachautomat — parsen (lokale Variablen)

```
1    private void parsen(){
2        char z;
3        zustand = START;
4        System.out.print('\u000c');
5        anzeigen(' ');
```

Lachautomat — parsen (Kern und Ausgabe)

```
6           for (int i = 0; i < eingabe.length(); i =
                i + 1){
7               z = eingabe.charAt(i);
8               switch (z){
9                   case 'h': { h();      break; }
10                  case 'a': { a();      break; }
11                  case '!': { arz();    break; }
12                  default:  { sonst(); break; }
13               }
14               anzeigen(z);
15           }
16           ergebnisAusgeben(isEndzustand());
17      }
```

Lachautomat — parsen (Zustandsübergänge)

```
1    private void h()
2    {
3        switch (zustand)
4        {
5            case START: { zustand = H;       break
                 ; }
6            case HA:    { zustand = H;       break
                 ; }
7            default:    { zustand = FEHLER; break
                 ; }
8        }
9    }
```

Übung

Ü 5.1: Fehlende Methoden implementieren
Ü 5.2: Funktion der Methode `isEndzustand()` erklären
Ü 5.3: Implementation Lachautomat möglichst weit vereinfachen
Ü 5.4: Implementation Lachautomat zuerst nach Zuständen switchen
Ü 5.5: parser für binäre Rechenoperationen erstellen
Ü 5.6: S 44/8
Ü 5.7: S 44/10