# Formale Sprachen - Einführung

Eine natürliche Sprache wie Deutsche, Englisch oder gar Chinesisch hat äußerst komplexe Regeln, die über Jahrhunderte gewachsen sind. Ob ein Satz in einer Sprache korrekt ist, entscheidet ein geübter Sprecher der Sprache meist intuitiv.

Jemand, der die Sprache erst lernt, müsste anhand von Regeln ableiten, ob ein Satz der „Grammatik“ der Sprache entspricht, also ob der Satz in der Sprache akzeptiert wird.

Da natürliche Sprachen viel zu komplexen Regeln folgen, betrachten wir im Folgenden nur „künstliche“ Sprachen, was im Zusammenhang der Informatik auch deswegen Sinn macht, da es sich bei den meisten „Programmiersprachen“ um künstliche Sprachen handelt, die von Autiomaten (den Compilern) verarbeitet werden.

**Frage:** Was ist nötig, um eine künstliche Sprache formal zu definieren?

## Hunde, die bellen, rennen (nicht)?

Wir betrachten eine sehr einfache Sprache, diese besteht aus Subjekten und Objekten. Insgesamt kann man Sätze bilden aus den Bestandteilen `{Higgs, Emil, bellen, rennen}`.

Zwei der Elemente sind Subjekte (Higgs und Emil), zwei sind Prädikate (bellen, rennen), man kann auf genau eine Weise einen Satz bilden:

```
<Subjekt> <Prädikat>
```

---

**(A1)**

Unsere Sprache hat vier verschiedene „Sätze“ - welche?

## Bestandteile unserer Sprache

### Alphabet

Um eine Sprache formal zu definieren, muss man zunächst ein **Alphabet Σ**[1] festlegen. Das Alphabet umfasst alle Symbole, aus denen Wörter/Sätze der Sprache gebildet werden können. Diese Symbole heißen auch „Terminalsymbole“

In unserem Beispiel besteht das Alphabet aus den Symbolen „Higgs“, „Emil“, „bellen“ und „rennen“. Man schreibt kurz:

**Σ={Higgs, Emil, bellen, rennen}**

Vorsicht Falle: Im normalen Sprachgebrauch bezeichnet ein Alphabet eine Menge aus einzelnen Zeichen. Bei formalen Sprachen kann ein Alphabet auch Zeichenfolgen (= Symbole) enthalten. Die Zeichenfolge „Higgs“ ist also ein Symbol unseres Alphabets.

Das Alphabet wird häufig auch als die Menge der Terminale bezeichnet.

## Regeln

Außer dem Alphabet benötigt man eine Menge von Regeln, die festlegen, wie in der Sprache ein gültiger Satz gebildet wird. Man draf also nicht einfach Symbole des Alphabets beliebig aneinanderreihen, um gültige Sätze zu bekommen, sondern man muss diese Regeln beachten.

Die Menge an Regeln, nach der sie Sätze unserer Sprache entstehen, wird mit **P** bezeichnet[2].

Aufgeschrieben werden die Regeln zum Beispiel so:

```
S -> H T
```

S, H und T sind dabei „syntaktische Variablen“, also Platzhalter für Symbole des Alphabets[3]. S hat eine besondere Rolle, und bezeichnet die Startvariable, also diejenige Regeln, an der die Satzbildung beginnt.

In unserem Beispiel kann man jetzt die folgenden Regelmenge **P** festlegen:

```
S -> H T       // Start, dann etwas, das im Platzhalter H steht, dann etwas
das im Platzhalter T steht
H -> Higgs     // Im Platzehalter H kann Higgs stehen
H -> Emil      // Im Platzhalter H kann Emil stehen
T -> bellt     // Im Platzhalter T kann bellen stehen
T -> rennt     // Im Platzhalter T kann rennen stehen
```

Dabei haben wir die **Variablenmenge V** verwendet: **V={S,H,T}**, wobei S die besondere Rolle der Startvariablen zukommt. Die Variablen heißen auch Nichtterminale, sie anders als die Terminale des

Alphabets, bei der Bildung von Worten der Sprache solange ersetzt werden, bis sie „verschwinden“.

Da es - wie in diesem Beispiel - häufig vorkommt, dass eine Variable alternativ für mehrere unterschiedliche Ersetzungen stehen kann, kürzt man das häufig folgendermaßen ab:

```
S -> H T             // Start, ein Element aus H dann ein Element aus T
H -> Higgs | Emil    // In H kann Higgs oder Emil stehen (der senkrechte
Strich steht also für "oder")
T -> bellt | rennt   // In T kann bellen oder rennen stehen
```

---

**(A2)**

- Mache dir klar, dass du unter Verwendung von Alphabet Σ, Produktionen P und Variablen V alle vier Sätze unserer Sprache bilden kannst, wenn du weißt, wo deine Regeln beginnen (S).
- Entwerfe einen endlichen Automaten, der nur korrekte Sätze der Sprache akzeptiert.
- Was muss du alles anpassen, damit die Hunde auch beide fressen können?

## Definition Grammatik einer formalen Sprache

Die Einzelteile (das 4-Tupel)

```
 V: Variablenmenge (Menge der Nichtterminale)
Σ: Alphabet (Menge der Terminale)
 P: Produktionen (Ersetzungsregeln)
 S: Startvariable
```

Bilden zusammen eine **Grammatik G**, welche die Sprache **L** beschreibt. man schreibt kurz:

```
G=(V,Σ,P,S)
```

Die Sprache **L** ist die Menge aller **Wörter**, die von der Startvariablen S aus anhand der Regeln P der Grammatik **abgeleitet** werden können.

**Achtung:** Obwohl man „Higgs rennt“ im normalen Sprachgebrauch als Satz bezeichnen würde, ist das im Sinne der formalen Sprachen ein Wort - das war oben wie ganze Zeit so, wir haben also die ganze Zeit „Worte“ unserer Sprache gebildet, keine Sätze, aber um euch nicht zu verwirren…

**Ableiten** bedeutet im Zusammenhang der formalen Sprachen, dass die linke Seite einer Regel durch die entsprechende rechte Seite ersetzt wird.

**(A3)**

Die Mengen

```
Σ = { der, die, das, Hund, Katze, jagt, kleine, bissige, große }
V = { <Satz>, <Subjekt>, <Prädikat>, <Objekt>, <Artikel>, <Attribut>,
<Adjektiv>, <Substantiv> }
    // <Satz> ist Startvariable
P = {
    <Satz>       ->   <Subjekt> <Prädikat> <Objekt>,
    <Subjekt>    ->   <Artikel> <Attribut> <Substantiv>,
    <Objekt>     ->   <Artikel> <Attribut> <Substantiv>,
    <Artikel>    ->   der | die | das,
    <Attribut>   ->   <Adjektiv> | <Adjektiv> <Attribut>,
    <Adjektiv>   ->   kleine | bissige | große,
    <Substantiv> ->   Hund | Katze,
    <Prädikat>   ->   jagt }
    }
```

Leite 5 gültige Worte der Sprache L ab, die durch die Grammatik G=(V,Σ,P,S) definiert ist und notiere jeweils den gesamten Weg der Ableitung.

## Syntaxdiagramm

Man kann eine Grammatik auch als **Syntaxdiagramm** darstellen. Für unsere Hundesprache würde das so aussehen:

Die Symbolformen haben dabei die folgende Bedeutung:

- Rechtecke mit scharfen Ecken sind Nichtterminalsymbole (Variablen). Diese müssen also an anderer Stelle durch Terminalsymbole definiert werden.
- Die Rechtecke mit runden Ecken sind Terminalsymbole
- Gültige Wörter der Sprache findet man, indem man wie eine Zug auf den Linien „fährt“, Alternativen stellen sich als „Weichen“ dar, Nichtterminale werden auf die gleiche Weise ersetzt bis das Wort nur noch aus Terminalsymbolen besteht. Wegen dieses Vorgehens nennt mal solche Syntaxdiagramme oft auch **Railroad-Diagramm**.

Mit dem Tool "Bottlecaps" kann man eine Grammatik eingeben und erhält dann ein Syntaxdiagramm. Dabei gelten folgende Konventionen:

- Der Ersetzungspfeil → wird in Bottlecaps als ::= geschrieben. Aus A→B wird also A::=B.

- Terminalsymbole werden in einfachen Hochhkommas geschrieben. Aus A→ B k wird `A::=B 'k'`
- Alternativen werden wie in unserer Schreibweise durch einen senkrechten Strich dargestellt: A→ B|C wird `A::=B|C`

---

**(A4)**

Überführe die Grammatik aus Aufgabe 3 mit Bottlecaps in ein Syntaxdiagramm

Lösung

```
Satz ::=   Subjekt Prädikat Objekt
Subjekt  ::= Artikel Attribut Substantiv
Objekt     ::=   Artikel Attribut Substantiv
Artikel    ::=   'der' | 'die' | 'das'
Attribut   ::=   Adjektiv | Adjektiv Attribut
Adjektiv   ::=   'kleine' | 'bissige' | 'große'
Substantiv ::=   'Hund' | 'Katze'
Prädikat   ::=   'jagt'
```

## Ein leeres Symbol

Zusätzlich zu Variablen und Alphabetsymbolen kann auf der rechten Seite einer Regel das „leere Wort“ ε[4] stehen. Das leere Wort steht für „kein Symbol“, manchmal ist seine Verwendung praktisch.

---

**(A5)**

Gegeben ist

```
Σ = {a, b}
V = { S }
P = { S -> aSb | ε }
```

Welche Worte hat die Sprache, die durch G=(V,Σ,P,S) erzeugt wird?[5]

# Material

[n/a: Keine Treffer]

1)

Sigma

2)

P steht für „productions“ oder „Produktionen“

3)

oder bereits nach anderen Regeln gebildete Satzgebilde, aber das dann später

4)

Epsilon, ε ist vergleichbar mit „“ in Java

5)

Achtung - man kann nicht alle Worte angeben, aber man kann angeben, wie alle Worte aufgebaut sind.

From:
https://wiki.qg-moessingen.de/ - **QG Wiki**

Permanent link:
**https://wiki.qg-moessingen.de/faecher:informatik:oberstufe:automaten:formale_sprachen:einfuehrung:start?rev=1654152737**

Last update: **02.06.2022 08:52**