Bedeutung der Produktionen / Semantik von KFGs

Jede Produktion einer KFG, etwa die Produktion
$A \to x$ kann man auffassen als

- "Strukturregel", die besagt "Ein A besteht aus x"

oder als

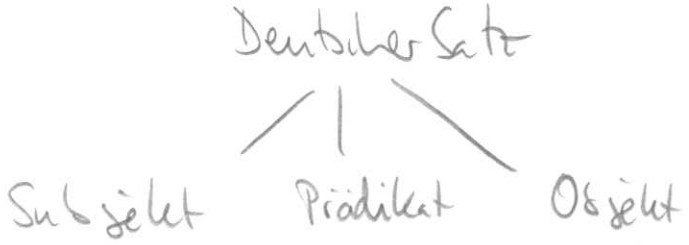
- "Ersetzungsregel", die besagt "A kann man durch x ersetzen"

Beispielsweise kann man die Produktion

Deutscher Satz → Subjekt Prädikat Objekt

verstehen als Aussage, die besagt:
"Ein Deutscher Satz ist aufgebaut aus Subjekt Prädikat Objekt"

Graphische Darstellung:



Das Grundkonzept für die Anwendung von Produktionen einer KFG ist die "Ableitung":

<u>Definition 6.7</u>: (Ableitung)

Sei $G = (T, N, S, P)$ eine KFG.

Falls $A \to x$ eine Produktion in P ist und $u \in V^*$ und $v \in V^*$ beliebige Worte über der Symbolmenge $V = T \cup N$ sind, so schreiben wir

$$uAv \underset{G}{\Rightarrow} uxv \quad (\text{bzw kurz: } uAv \Rightarrow uxv)$$

und sagen, dass uAv in einem <u>Ableitungsschritt</u> zu uxv umgeformt werden kann.

Eine <u>Ableitung</u> ist eine Folge von hintereinander angewendeten Ableitungsschritten.

Für Worte $w \in V^*$ und $w' \in V^*$ schreiben wir

$$w \underset{G}{\Rightarrow}^* w' \quad (\text{bzw kurz: } w \Rightarrow^* w'),$$

um auszusagen, dass es eine endliche Folge von Ableitungsschritten gibt, die w zu w' umformt.

<u>Spezialfall</u>: diese Folge darf auch aus 0 Ableitungsschritten bestehen, d.h. f.a. $w \in V^*$ gilt: $w \Rightarrow^* w$.

Beispiel 6.8

Sei G_K die "Klammer-Grammatik" aus Bsp 6.6:

Beispiele für einzelne Ableitungsschritte:

- (Liste) $\Rightarrow$ (Klammerung Liste)
- ((Liste) Liste) $\Rightarrow$ (( ) Liste)

Ein Beispiel für eine Ableitung in G_K:

$$
\begin{aligned}
\text{Klammerung} &\Rightarrow (\ \text{Liste}\ ) \\
&\Rightarrow (\ \text{Klammerung Liste}\ ) \\
&\Rightarrow (\ \text{Klammerung Klammerung Liste}\ ) \\
&\Rightarrow (\ \text{Klammerung } (\ \text{Liste}\ )\ \text{Liste}\ ) \\
&\Rightarrow (\ (\ \text{Liste}\ )\ (\ \text{Liste}\ )\ \text{Liste}\ ) \\
&\Rightarrow (\ (\ )\ (\ \text{Liste}\ )\ \text{Liste}\ ) \\
&\Rightarrow (\ (\ )\ (\ )\ \text{Liste}\ ) \\
&\Rightarrow (\ (\ )\ (\ )\ )
\end{aligned}
$$

In jedem Schritt wird jeweils eine Produktion auf ein Nichtterminal der vorangehenden Symbolfolge angewandt. Obige Kette von Ableitungsschritten zeigt, dass

$$\text{Klammerung} \Rightarrow^* (\ (\ )\ (\ )\ )$$

Definition 6.9 (Sprache einer KFG)

Sei $G = (T, N, S, P)$ eine KFG.

Die von G erzeugte Sprache $L(G)$ ist die Menge aller Worte über dem Alphabet T, die aus dem Startsymbol S abgeleitet werden können. D.h.:

$$L(G) := \{ w \in T^* : S \xrightarrow[G]{*} w \}$$

Beispiel 6.10

Die KFG G_K aus Bsp. 6.6 definiert die Sprache $L(G_K)$, die aus allen "wohlgeformten Klammerausdrücken" besteht, die von einem Klammerpaar umschlossen werden.

Beispielsweise gehören folgende Worte zu $L(G_K)$:

$()$, $(())$, $(()())$, $((()())())$,

aber die Worte $)($, $(()$, $()()$, (Liste)

gehören nicht zu $L(G_K)$.

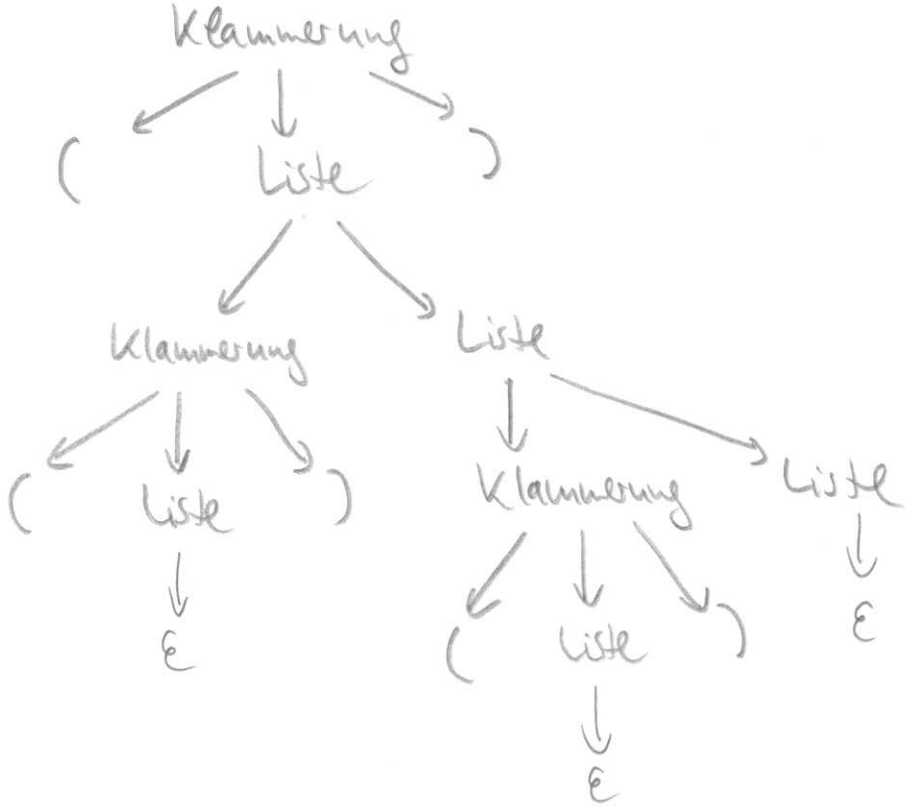
Beispiel 6.11:

Sei G_K die "Klammer-Grammatik" aus Bsp 6.6.

Die Ableitung

$$\begin{aligned}
\text{Klammerung} &\Rightarrow (\text{Liste}) \Rightarrow (\text{Klammerung Liste}) \\
&\Rightarrow ((\text{Liste})\ \text{Liste}) \Rightarrow (()\ \text{Liste}) \\
&\Rightarrow (()\ \text{Klammerung Liste}) \Rightarrow (()\ \text{Klammerung}) \\
&\Rightarrow (()\ (\text{Liste})) \Rightarrow (()())
\end{aligned}$$

wird durch den folgenden Ableitungsbaum dargestellt:



Ableitungsbäume:

Sei $G = (T, N, S, P)$ eine KFG.

Jede Ableitung $S \Rightarrow_G^* w$ lässt sich als gerichteter Baum darstellen, bei dem

- jeder Knoten mit einem Symbol aus $T \cup N \cup \{\varepsilon\}$ markiert ist und
- bei dem die Kinder jedes Knotens eine festgelegte Reihenfolge haben (in der Zeichnung eines Ableitungsbaums werden von links nach rechts zunächst das "erste Kind" dargestellt, dann das zweite, dritte etc.)

Die Wurzel des Baums ist mit dem Startsymbol S markiert. Jeder Knoten mit seinen Kindern repräsentiert die Anwendung einer Produktion aus P: Die Anwendung einer Produktion der Form $A \to x$ (mit $A \in N$ und $x \in V^*$) wird im Ableitungsbaum repräsentiert durch einen Knoten, der mit dem Symbol A markiert ist und der $|x|$ viele Kinder hat, so dass das i-te Kind mit dem i-ten Symbol von x markiert ist (f.a. $i \in \{1, \dots, |x|\}$. <u>Spezialfall</u>: Ist x das <u>leere</u> Wort ε, so wird eine Anwendung der Produktion $A \to \varepsilon$ repräsentiert durch einen mit A markierten Knoten, der genau ein Kind hat, das mit ε markiert ist.

<u>Beachte:</u>

Ein Ableitungsbaum kann mehrere Ableitungen repräsentieren

Beispielsweise repräsentiert der obige Ableitungsbaum auch die Ableitung

Klammerung $\Rightarrow$ (Liste) $\Rightarrow$ (Klammerung Liste)

$\Rightarrow$ (Klammerung Klammerung Liste)

$\Rightarrow$ ((Liste) Klammerung Liste)

$\Rightarrow$ ((Liste) (Liste) Liste)

$\Rightarrow$ (( ) (Liste) Liste) $\Rightarrow$ (( ) ( ) Liste)

$\Rightarrow$ (( ) ( )),

in der gegenüber der ursprünglichen Ableitung aus Bsp 6.11 einige Ableitungsschritte vertauscht sind.

Im Ableitungsbaum wird von der konkreten Reihenfolge, in der die einzelnen Ableitungsschritte vorkommen, abstrahiert.

Im Folgenden betrachten wir einige weitere Beispiele für kontextfreie Grammatiken.

<u>Beispiel 6.12</u> (Aussagenlogik)

Wir konstruieren eine KFG

$$G_{AL} = (T, N, S, P),$$

deren Sprache $L(G_{AL})$ gerade die Menge aller aussagenlogischen Formeln ist, in denen nur Variablen aus $\{V_0, V_1, V_2\}$ vorkommen:

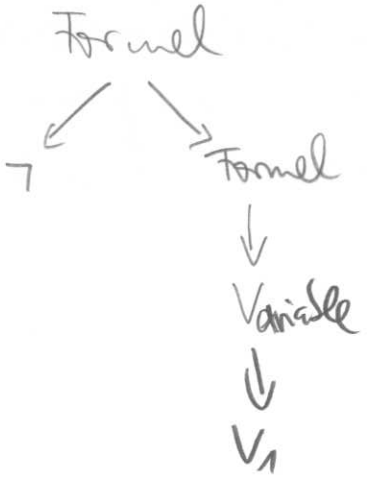
- Terminalsymbole
  $T := \{V_0, V_1, V_2, 0, 1, \neg, \wedge, \vee, \rightarrow, \leftrightarrow, (,)\}$

- Nichtterminalsymbole
  $N := \{\text{Formel}, \text{Variable}, \text{Junktor}\}$

- Startsymbol $S :=$ Formel

- Produktionen $P := \{$
  Formel $\rightarrow 0$, Formel $\rightarrow 1$, Formel $\rightarrow$ Variable
  Formel $\rightarrow \neg$ Formel,
  Formel $\rightarrow$ (Formel Junktor Formel)
  Variable $\rightarrow V_0$, Variable $\rightarrow V_1$, Variable $\rightarrow V_2$,
  Junktor $\rightarrow \wedge$, Junktor $\rightarrow \vee$, Junktor $\rightarrow \rightarrow$, Junktor $\rightarrow \leftrightarrow$ $\}$

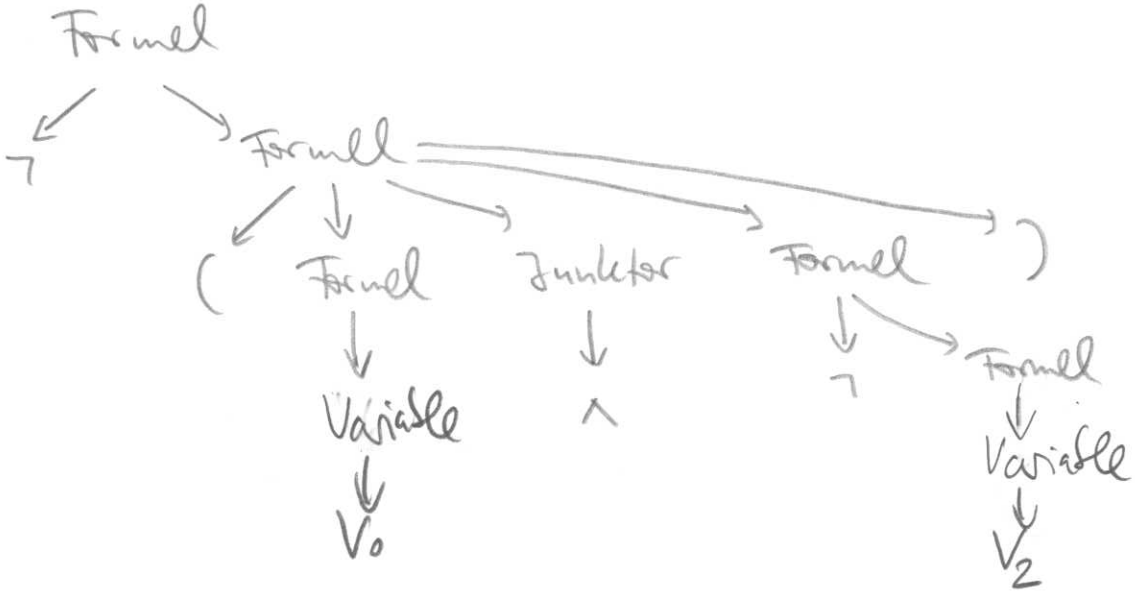
Beispiele für Ableitungsbäume:



Dieser Ableitungsbaum repräsentiert die Ableitung

Formel $\Rightarrow$ ¬Formel $\Rightarrow$ ¬Variable $\Rightarrow$ $\neg V_1$

Das durch diese(n) Ableitung(sbaum) erzeugte Wort in der Sprache $L(G_{AL})$ ist die Formel $\neg V_1$



Dieser Ableitungsbaum repräsentiert die Ableitung

Formel $\Rightarrow$ ¬Formel $\Rightarrow$ ¬(Formel Junktor Formel)

$\Rightarrow$ ¬(V_0 Junktor Formel) $\Rightarrow$ ¬($V_0 \wedge$ Formel)

$\Rightarrow$ ¬($V_0 \wedge$ ¬Formel) $\Rightarrow$ ¬($V_0 \wedge \neg V_2$)

Das durch diese(n) Ableitung(sbaum) erzeugte Wort in der Sprache $L(G_{AL})$ ist die Formel $\neg(V_0 \wedge \neg V_2)$

Beispiel 6.13 (Menü-Struktur)

In der graphischen Benutzungsoberfläche von vielen Software-Systemen werden oftmals "Menüs" verwendet. Ein Menü besteht aus einem Menünamen und einer Folge von Einträgen. Jeder einzelne Eintrag besteht dabei aus einem Operationsnamen oder selbst wieder einem Menü.

Beispiel:

- Hauptmenü:
  - Datei
    - Laden
    - Speichern
    - Exportieren als
      - HTML
      - PDF
  - Bearbeiten
    - Kopieren
    - Einfügen

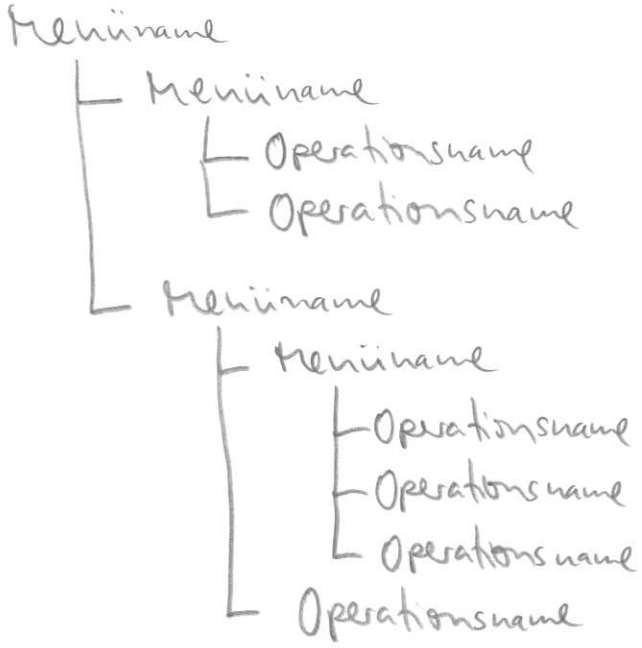
Zur Spezifizierung der Grund-Struktur solcher Menüs kann man folgende Grammatik $G_{Menü}$ verwenden:

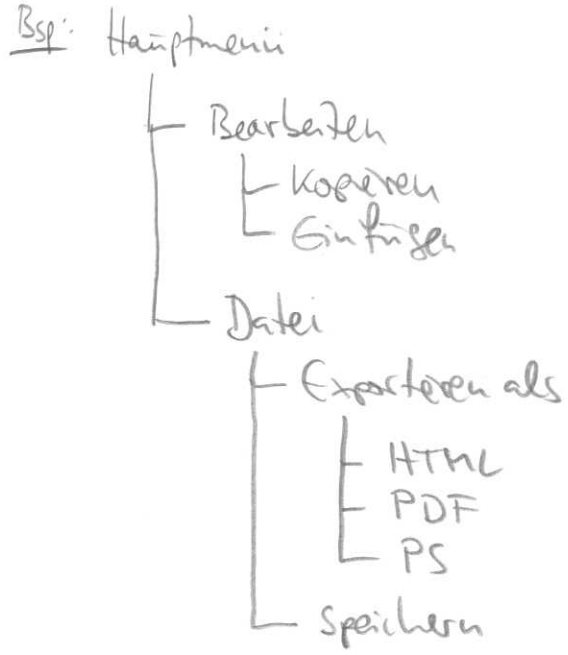
$$G_{Menü} = (T, N, S, P) \quad \text{mit}$$

- $T := \{$ Menüname, Operationsname $\}$
- $N := \{$ Menü, Eintragsfolge, Eintrag $\}$
- $S :=$ Menü

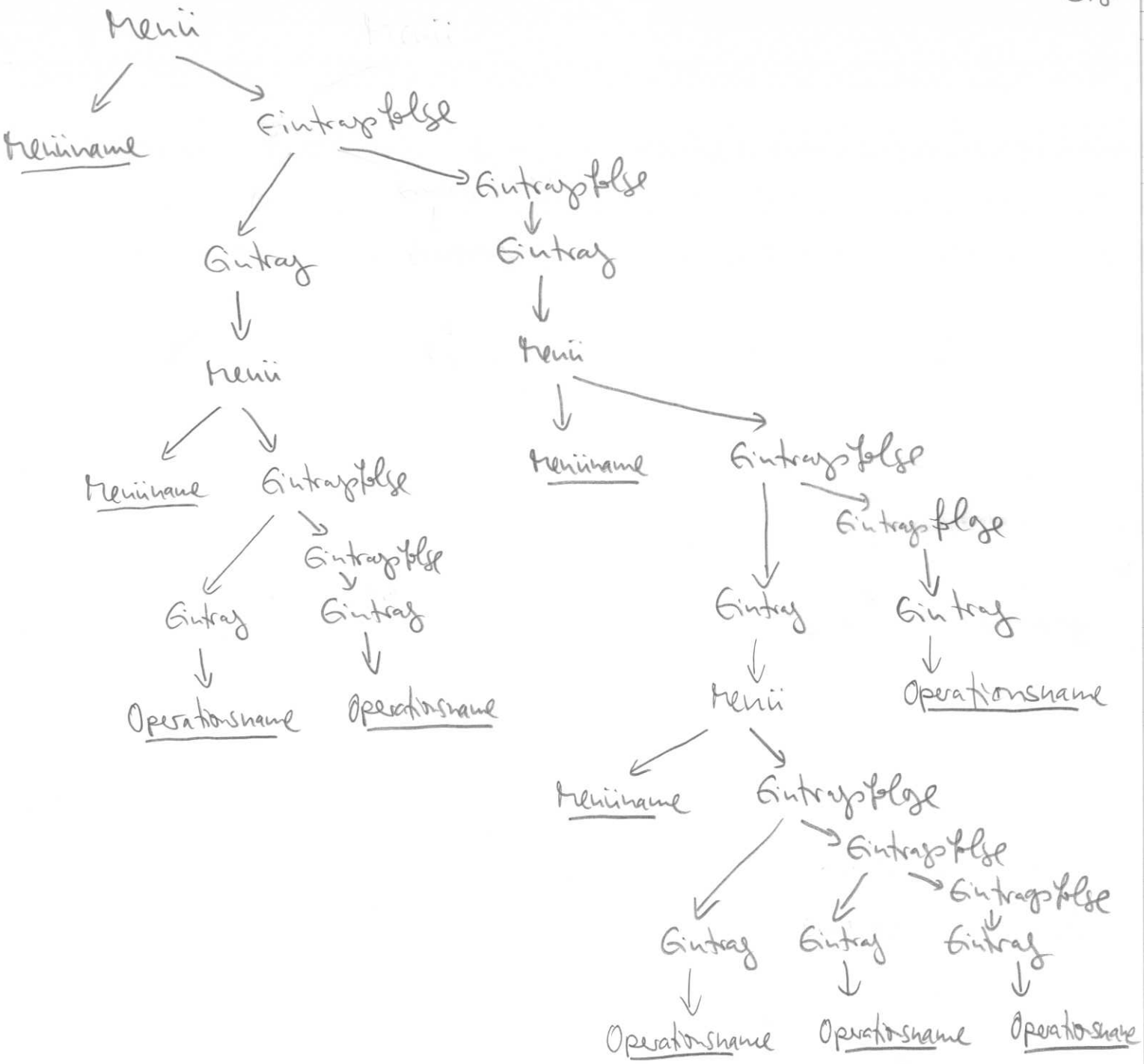
- $P := \{$ Menü $\rightarrow$ Menüname Eintragsfolge,
  Eintragsfolge $\rightarrow$ Eintrag,
  Eintragsfolge $\rightarrow$ Eintrag Eintragsfolge,
  Eintrag $\rightarrow$ Operationsname
  Eintrag $\rightarrow$ Menü $\}$

Jeder Ableitungsbaum repräsentiert die Struktur eines Menüs. Ein Menü der Struktur





wird z.B. von folgendem Ableitungsbaum repräsentiert:


Menü
Menüname
Eintragsfolge
Eintrag
Menü
Menüname
Eintragsfolge
Eintrag
Operationsname
Eintragsfolge
Eintrag
Operationsname
Eintragsfolge
Eintrag
Menü
Menüname
Eintragsfolge
Eintrag
Menü
Menüname
Eintragsfolge
Eintrag
Operationsname
Eintragsfolge
Eintrag
Operationsname
Eintragsfolge
Eintrag
Operationsname
Eintrag
Operationsname
Eintrag
Operationsname

Beispiel 6.14: (HTML-Tabellen)

HTML: Format zur Beschreibung von verzweigten Dokumenten im WWW.

Ein Bestandteil, der oft im Quell-Code von Internet-Seiten vorkommt, sind Tabellen. Z.B. wird der Eintrag

Tag	Zeit	Raum
Mi	8:00 – 11:00	Magnus-Hörsaal
Do	14:00 – 16:00	NM 10

durch HTML-Quelltext der folgenden Form erzeugt:

```
<table>
  <tr>
    <td> Tag </td>
    <td> Zeit </td>
    <td> Raum </td>
  </tr>
  <tr>
    <td> Mi </td>
    <td> 8:00 – 11:00 </td>
    <td> Magnus-Hörsaal </td>
  </tr>
  <tr>
    <td> Do </td>
    <td> 14:00 – 16:00 </td>
    <td> NM10 </td>
  </tr>
</table>
```

<tr> steht für den Anfang einer Zeile der Tabelle

</tr> steht für das Ende einer Zeile der Tabelle

<td> steht für den Anfang eines Eintrags,

</td> für das Ende eines Eintrags

Als Einträge in einzelnen Zellen der Tabelle kann z.B. Text stehen oder eine weitere Tabelle.

Im Folgenden konstruieren wir eine Grammatik

$$G_{\text{HTML-Tabellen}} = (T, N, S, P),$$

so dass die von $G_{\text{HTML-Tabellen}}$ erzeugte Sprache aus (möglicherweise geschachtelten) HTML-Tabellen besteht:

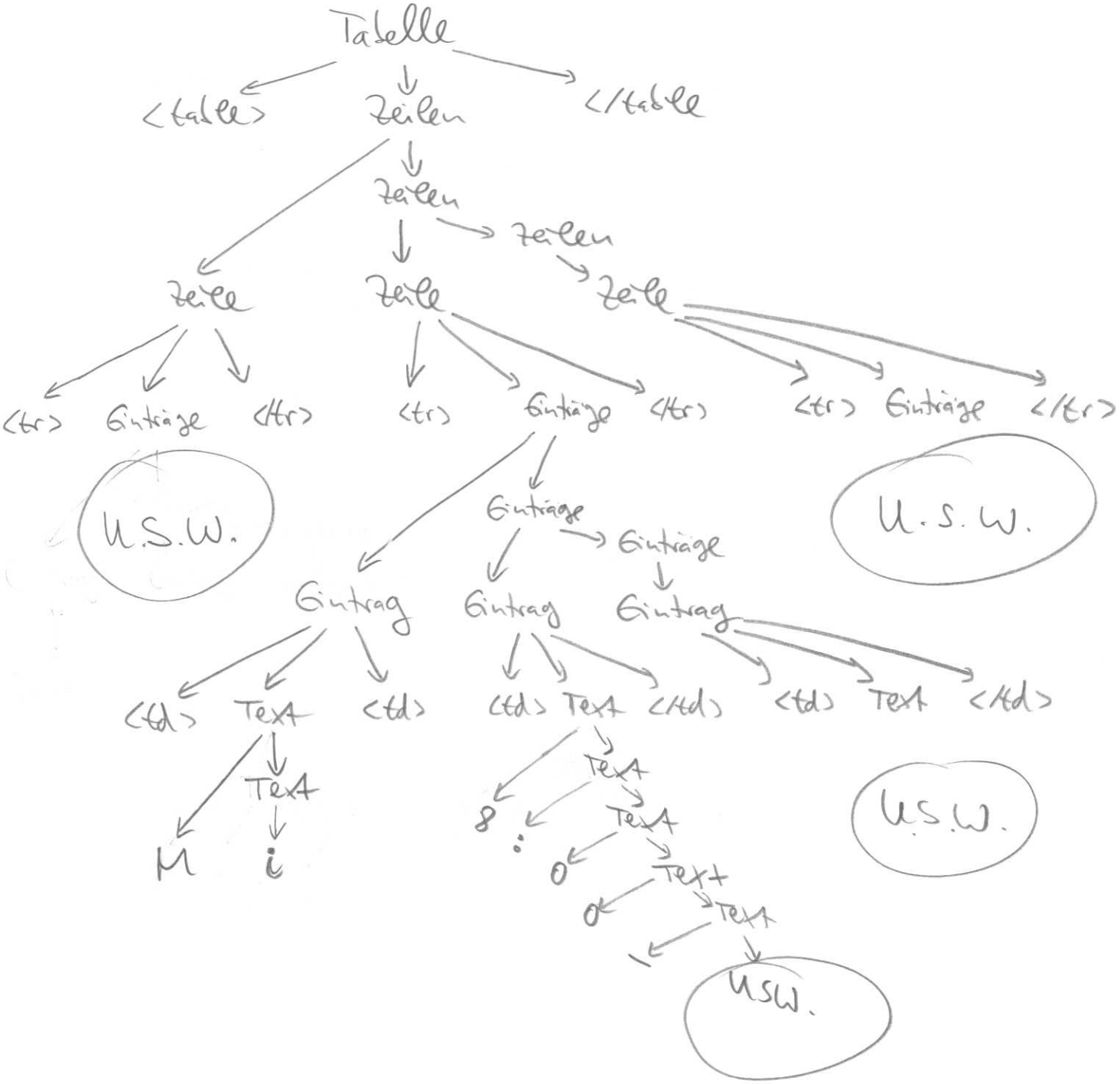
$T := \{$ <table>, </table>,
<tr>, </tr>, <td>, </td>,
a, ..., z, A, ..., Z, 0, 1, ..., 9, :, -, ␣,
ö, ä, ü, ß, Ö, Ä, Ü $\}$

$N := \{$ Tabelle, Zeile, Eintrag, Text, Zeilen, Einträge $\}$

$S :=$ Tabelle

$P := \{$ Tabelle → <table> Zeilen </table>,
Zeilen → Zeile, Zeilen → Zeile Zeilen,
Zeile → <tr> Einträge </tr>,
Einträge → Eintrag, Einträge → Eintrag Einträge,
Eintrag → <td> Text </td>, Eintrag → <td> Tabelle </td>
Text → a, ..., Text → z, Text → A, ..., Text → Ü,
Text → a Text, ..., Text → z Text, Text → A Text, ..., Text → Ü Text $\}$

Die oben angegebene Beispiel-HTML-Tabelle wird z.B. durch eine Ableitung erzeugt, die durch folgenden Ableitungsbaum repräsentiert wird:



Bemerkung 6.15

Typische Fragen bzgl. kontextfreien Grammatiken:

(a) Welche Sprachen können prinzipiell durch KFGs erzeugt werden, welche nicht?

Bsp: Die Sprache $L_1 = \{a^n b^n : n \in \mathbb{N}\}$ wird von der KGF $G_1 = (T, N, S, P)$ mit $T = \{a, b\}$, $N = \{S\}$ und $P = \{S \to aSb, \quad S \to \varepsilon\}$ erzeugt, dh $L_1 = L(G_1)$.

Notation: $a^n b^n$ ist eine Abkürzung für $\underbrace{aa\cdots a}_{n}\underbrace{bb\cdots b}_{n}$

Satz: Es gibt keine KFG, die genau die Sprache $L_2 := \{a^n b^n c^n : n \in \mathbb{N}\}$ erzeugt

Beweis: In der Vorlesung GL-2.

(b) Gegeben sei eine KFG $G = (T, N, S, P)$ und ein Wort $w \in T^*$. Wie kann man herausfinden, ob $w \in L(G)$ ist, dh. ob das Wort w zu der von G erzeugten Sprache gehört?

Dies ist das so genannte Wortproblem.

Einen Algorithmus zum Lösen des Wortproblems für KFGs werden Sie in der Vorlesung GL-2 kennenlernen (den sogenannten CYK-Algorithmus, der nach seinen Erfindern Cocke, Younger und Kasami benannt ist).

7. Modellieren von Abläufen

In diesem Kapitel geht es darum, das dynamische Verhalten von Systemen zu beschreiben, z.B.

- die Wirkung von Bedienoperationen auf reale Automaten oder auf die Benutzungsoberflächen von Software-Systemen
- Schaltfolgen von Ampelanlagen
- Abläufe von Geschäftsprozessen in Firmen
- Steuerung von Produktionsanlagen.

Solche Abläufe werden modelliert, indem man die Zustände angibt, die ein System annehmen kann, und beschreibt, unter welchen Bedingungen es aus einem Zustand in einen anderen übergehen kann (vgl. das Flussübergabe-Problem aus Beispiel 1.1).

In diesem Kapitel werden zwei grundlegende Kalküle vorgestellt, mit denen man solche Abläufe beschreiben kann:

1) <u>endliche Automaten</u>, die sich gut für Modellierung sequentieller Abläufe eignen, und
2) <u>Petri-Netze</u>, mit denen nebenläufige Prozesse beschrieben werden können, bei denen Ereignisse gleichzeitig an mehreren Stellen des Systems Zustandsänderungen bewirken können.

7.1 Endliche Automaten

Endliche Automaten sind ein formaler Kalkül, der zur Spezifikation von realen oder abstrakten Maschinen genutzt werden kann. Endliche Automaten

- reagieren auf äußere Ereignisse
- ändern ggf. seinen "inneren Zustand"
- produzieren ggf. eine Ausgabe.

Sie werden z.B eingesetzt um

- das Verhalten realer Maschinen zu spezifizieren
  (Bsp: Getränkeautomat)
- das Verhalten von Software-Komponenten zu beschreiben
  (Bsp: das Wirken von Bedienoperationen auf Benutzungs-Oberflächen von Software-Systemen)
- Sprachen zu spezifizieren, d.h. die Menge aller Ereignisfolgen, die den Automat von seinem "Startzustand" in einen "akzeptierenden Zustand" überführen
  (Bsp: "Flüssigüberquerung" aus Bsp 1.1: alle Folgen von "Flüssigüberquerungsschritten", mit denen man vom "Startzustand" 

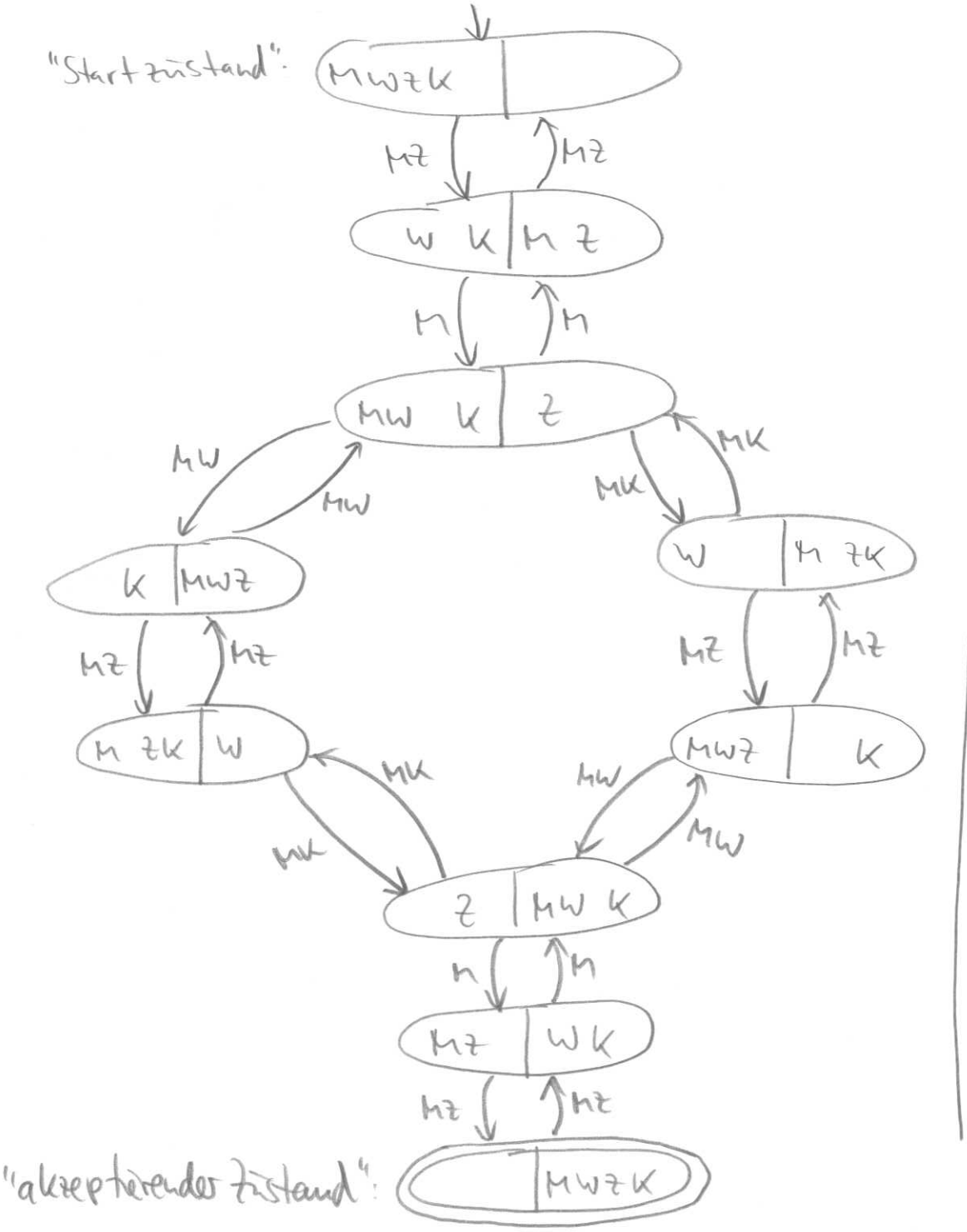
zum "Zielzustand" 

gelangen kann)

Vor der formalen Definition endlicher Automaten zunächst zwei einführende Beispiele:

Beispiel 7.1

Graphische Darstellung eines endlichen Automaten zum Flussüberquerungs-Problem aus Bsp 1.1:



Dieser endliche Automat "akzeptiert" genau diejenigen Folgen von einzelnen Flussüberquerungen, die vom Startzustand in den akzeptierenden Zustand führen

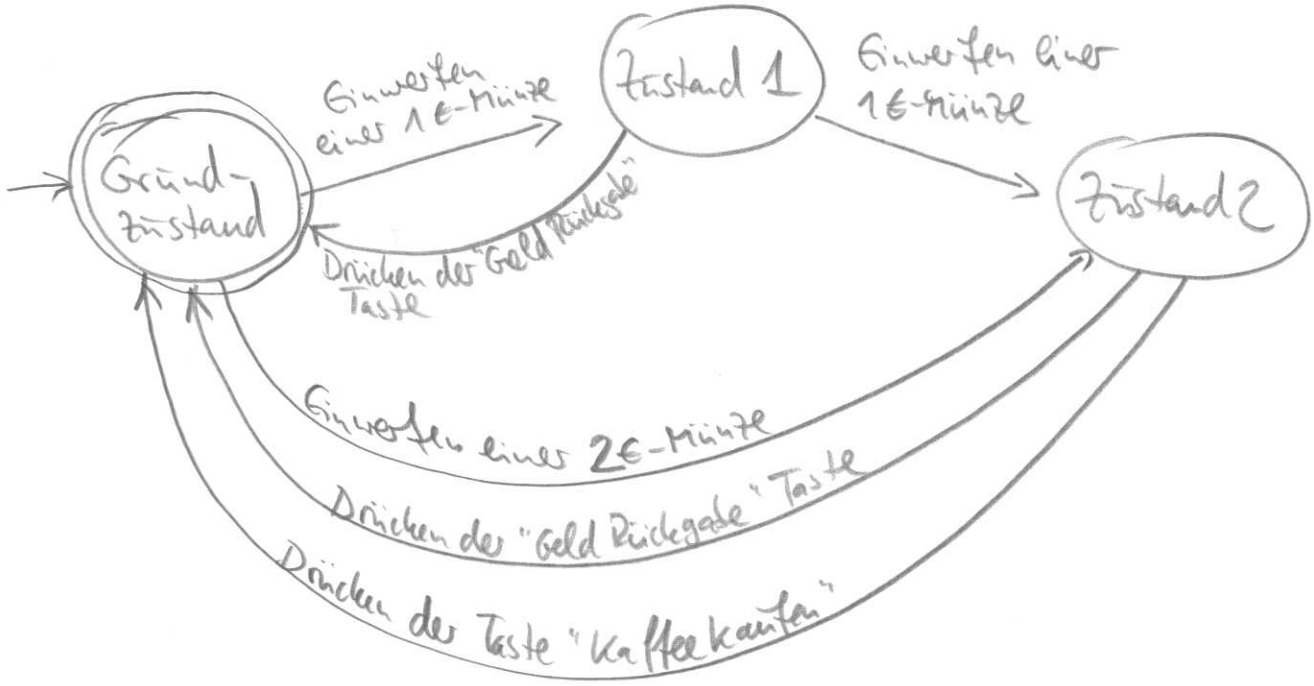
Beispiel 7.2:

Betrachte einen einfachen Getränkeautomat, der folgende Bedienoptionen hat:

- Einwerfen einer 1€-Münze
- Einwerfen einer 2€-Münze
- Taste "Geld Rückgabe" drücken
- Taste "Kaffee kaufen" drücken

und bei dem man ein einziges Getränk kaufen kann, das 2€ kostet.

Dieser Getränkeautomat kann durch folgenden endlichen Automaten modelliert werden:



Dieser endliche Automat "akzeptiert" genau diejenigen Folgen von Bedienoperationen, die vom Grundzustand aus wieder in den Grundzustand führen.