Kontextfreie Grammatiken

Was kann man mit kontextfreien Grammatiken anfangen?

Kontextfreie Grammatiken, kurz:

KFGs

werden zur Modellierung von

beliebig tief geschachtelten baumartigen Strukturen

eingesetzt.

Was kann man mit kontextfreien Grammatiken anfangen?

Kontextfreie Grammatiken, kurz:

KFGs

werden zur Modellierung von

beliebig tief geschachtelten baumartigen Strukturen

eingesetzt.

KFGs werden unter Anderen angewandt,

- um **Programmiersprachen** wie etwa Java, C oder Pascal zu beschreiben und spielen auch beim „Compilerbau“ eine zentrale Rolle,
- um **Datenaustauschformate**, d.h. Sprachen (wie etwa HTML oder XML) als Schnittstelle zwischen Software-Werkzeugen, zu beschreiben,
- um **Bäume** zur Repräsentation strukturierter Daten (z.B. XML) zu beschreiben.

Kontextfreie Grammatiken: Die formale Definition

Eine kontextfreie Grammatik

$$G = (\Sigma, V, S, P)$$

besteht aus

- einer endlichen Menge Σ von **Terminalen** und einer endlichen Menge V von **Nichtterminalen** (oder **Variablen**).

Kontextfreie Grammatiken: Die formale Definition

Eine kontextfreie Grammatik

$$G = (\Sigma, V, S, P)$$

besteht aus

- einer endlichen Menge Σ von **Terminalen** und einer endlichen Menge V von **Nichtterminalen** (oder **Variablen**).
  - Die Mengen Σ und V sind disjunkt, d.h. $\Sigma \cap V = \emptyset$ gilt.
  - Die Menge $W := \Sigma \cup V$ heißt **Vokabular**, die Elemente in W nennt man auch Symbole,

Kontextfreie Grammatiken: Die formale Definition

Eine kontextfreie Grammatik

$$G = (\Sigma, V, S, P)$$

besteht aus

- einer endlichen Menge Σ von **Terminalen** und einer endlichen Menge V von **Nichtterminalen** (oder **Variablen**).
  - Die Mengen Σ und V sind disjunkt, d.h. $\Sigma \cap V = \emptyset$ gilt.
  - Die Menge $W := \Sigma \cup V$ heißt **Vokabular**, die Elemente in W nennt man auch Symbole,
- einem Symbol $S \in V$, dem **Startsymbol**

Kontextfreie Grammatiken: Die formale Definition

Eine kontextfreie Grammatik

$$G = (\Sigma, V, S, P)$$

besteht aus

- einer endlichen Menge Σ von **Terminalen** und einer endlichen Menge V von **Nichtterminalen** (oder **Variablen**).
  - Die Mengen Σ und V sind disjunkt, d.h. $\Sigma \cap V = \emptyset$ gilt.
  - Die Menge $W := \Sigma \cup V$ heißt **Vokabular**, die Elemente in W nennt man auch Symbole,
- einem Symbol $S \in V$, dem **Startsymbol** und
- einer endlichen Menge

  $$P \subseteq V \times W^*$$

  von **Produktionen**.
  Für eine Produktion $(A, x) \in P$ schreiben wir meistens $A \to x$.

Arithmetische Ausdrücke (1/2)

Wir möchten alle „wohl-gebideten“ arithmetische Ausdrücke beschreiben,

- die über den Zahlen $1, 2, 3$ gebildet sind und
- die Operatoren $+, -, \cdot$ sowie Klammern $(,)$ benutzen.

Beispiele für wohl-gebildete arithmetische Ausdrücke sind

$$(1 + 3) \cdot (2 + 2 + 3) - 1$$

und

$$(1 + 3) \cdot ((2 + 2 + 3) - 1).$$

Arithmetische Ausdrücke (2/2)

Wir betrachten die KFG $G_{AA} := (\Sigma, V, S, P)$ mit

- Terminalalphabet $\Sigma := \{1, 2, 3, +, -, \cdot, (,)\}$

Arithmetische Ausdrücke (2/2)

Wir betrachten die KFG $G_{AA} := (\Sigma, V, S, P)$ mit

- Terminalalphabet $\Sigma := \{1, 2, 3, +, -, \cdot, (,)\}$
- Nichtterminalalphabet $V := \{\mathit{Ausdruck}, \mathit{Operator}\}$

Arithmetische Ausdrücke (2/2)

Wir betrachten die KFG $G_{AA} := (\Sigma, V, S, P)$ mit

- Terminalalphabet $\Sigma := \{1, 2, 3, +, -, \cdot, (,)\}$
- Nichtterminalalphabet $V := \{\mathit{Ausdruck}, \mathit{Operator}\}$
- Startsymbol $S := \mathit{Ausdruck}$

Arithmetische Ausdrücke (2/2)

Wir betrachten die KFG $G_{AA} := (\Sigma, V, S, P)$ mit

- Terminalalphabet $\Sigma := \{1, 2, 3, +, -, \cdot, (,)\}$
- Nichtterminalalphabet $V := \{\mathit{Ausdruck}, \mathit{Operator}\}$
- Startsymbol $S := \mathit{Ausdruck}$
- und der Produktionsmenge

$$P := \left\{ \begin{array}{lcl} \mathit{Ausdruck} & \to & 1\ , \\ \mathit{Ausdruck} & \to & 2\ , \\ \mathit{Ausdruck} & \to & 3\ , \end{array} \right.$$

Arithmetische Ausdrücke (2/2)

Wir betrachten die KFG $G_{AA} := (\Sigma, V, S, P)$ mit

- Terminalalphabet $\Sigma := \{1, 2, 3, +, -, \cdot, (,)\}$
- Nichtterminalalphabet $V := \{\mathit{Ausdruck}, \mathit{Operator}\}$
- Startsymbol $S := \mathit{Ausdruck}$
- und der Produktionsmenge

$$
\begin{aligned}
P := \quad \{ \quad & \mathit{Ausdruck} \rightarrow 1 \,, \\
& \mathit{Ausdruck} \rightarrow 2 \,, \\
& \mathit{Ausdruck} \rightarrow 3 \,, \\
& \mathit{Ausdruck} \rightarrow \mathit{Ausdruck}\ \mathit{Operator}\ \mathit{Ausdruck} \,,
\end{aligned}
$$

Arithmetische Ausdrücke (2/2)

Wir betrachten die KFG $G_{AA} := (\Sigma, V, S, P)$ mit

- Terminalalphabet $\Sigma := \{1, 2, 3, +, -, \cdot, (,)\}$
- Nichtterminalalphabet $V := \{\mathit{Ausdruck}, \mathit{Operator}\}$
- Startsymbol $S := \mathit{Ausdruck}$
- und der Produktionsmenge

$$
\begin{aligned}
P := \quad \{ \quad & \mathit{Ausdruck} \rightarrow 1 \ , \\
& \mathit{Ausdruck} \rightarrow 2 \ , \\
& \mathit{Ausdruck} \rightarrow 3 \ , \\
& \mathit{Ausdruck} \rightarrow \mathit{Ausdruck}\ \mathit{Operator}\ \mathit{Ausdruck} \ , \\
& \mathit{Ausdruck} \rightarrow (\ \mathit{Ausdruck}\ )
\end{aligned}
$$

Arithmetische Ausdrücke (2/2)

Wir betrachten die KFG $G_{AA} := (\Sigma, V, S, P)$ mit

- Terminalalphabet $\Sigma := \{1, 2, 3, +, -, \cdot, (,)\}$
- Nichtterminalalphabet $V := \{\mathit{Ausdruck}, \mathit{Operator}\}$
- Startsymbol $S := \mathit{Ausdruck}$
- und der Produktionsmenge

$$
\begin{aligned}
P := \quad \{ \quad & \mathit{Ausdruck} \rightarrow 1 \,, \\
& \mathit{Ausdruck} \rightarrow 2 \,, \\
& \mathit{Ausdruck} \rightarrow 3 \,, \\
& \mathit{Ausdruck} \rightarrow \mathit{Ausdruck}\ \mathit{Operator}\ \mathit{Ausdruck} \,, \\
& \mathit{Ausdruck} \rightarrow (\ \mathit{Ausdruck}\ ) \,, \\
& \\
& \mathit{Operator} \rightarrow
\end{aligned}
$$

Arithmetische Ausdrücke (2/2)

Wir betrachten die KFG $G_{AA} := (\Sigma, V, S, P)$ mit

- Terminalalphabet $\Sigma := \{1, 2, 3, +, -, \cdot, (,)\}$
- Nichtterminalalphabet $V := \{\mathit{Ausdruck}, \mathit{Operator}\}$
- Startsymbol $S := \mathit{Ausdruck}$
- und der Produktionsmenge

$$
\begin{aligned}
P := \quad \{ \quad & \mathit{Ausdruck} \rightarrow 1 \,, \\
& \mathit{Ausdruck} \rightarrow 2 \,, \\
& \mathit{Ausdruck} \rightarrow 3 \,, \\
& \mathit{Ausdruck} \rightarrow \mathit{Ausdruck}\ \mathit{Operator}\ \mathit{Ausdruck} \,, \\
& \mathit{Ausdruck} \rightarrow (\ \mathit{Ausdruck}\ ) \,, \\
& \\
& \mathit{Operator} \rightarrow + \,, \\
& \mathit{Operator} \rightarrow - \,, \\
& \mathit{Operatur} \rightarrow \cdot \quad \}
\end{aligned}
$$

Wir sparen Schreibarbeit

Wir fassen Zeilen, die das gleiche Nichtterminal auf der linken Seite des Pfeils aufweisen, zu einer einzigen Zeile zusammen.

Damit können wir die Produktionsmenge P auch kurz wie folgt beschreiben:

$$
\begin{array}{lllll}
P = \{ & \textit{Ausdruck} & \to & 1 \mid 2 \mid 3 & , \\
 & \textit{Ausdruck} & \to & \textit{Ausdruck Operator Ausdruck} \mid (\ \textit{Ausdruck}\ ) & , \\
 & \textit{Operator} & \to & + \mid - \mid \cdot & \}.
\end{array}
$$

Wir sparen Schreibarbeit

Wir fassen Zeilen, die das gleiche Nichtterminal auf der linken Seite des Pfeils aufweisen, zu einer einzigen Zeile zusammen.

Damit können wir die Produktionsmenge P auch kurz wie folgt beschreiben:

$$
\begin{array}{lll}
P = \{ & \textit{Ausdruck} \to 1 \mid 2 \mid 3 \ , & \\
& \textit{Ausdruck} \to \textit{Ausdruck Operator Ausdruck} \mid (\textit{Ausdruck}) \ , & \\
& \textit{Operator} \to + \mid - \mid \cdot & \}.
\end{array}
$$

Die Produktion $\textit{Ausdruck} \to \textit{Ausdruck Operator Ausdruck}$ können wir auffassen

- als **Strukturregel**, die besagt „Ein *Ausdruck* besteht aus einem *Ausdruck*, gefolgt von einem *Operator*, gefolgt von einem *Ausdruck* — oder als
- **Ersetzungsregel**, die besagt, dass das „Symbol *Ausdruck* durch das Wort *Ausdruck Operator Ausdruck* ersetzt werden kann.“

Ableitungen

Ableitungsschritte

Sei $G = (\Sigma, V, S, P)$ eine KFG.

Falls

$$A \to x$$

eine Produktion in P ist und $u \in W^*$ und $v \in W^*$ beliebige Worte über dem Vokabular $W = \Sigma \cup V$ sind, so schreiben wir

$$uAv \implies_G uxv \quad \text{(bzw. kurz:} \quad uAv \implies uxv\text{)}$$

und sagen, dass uAv in einem **Ableitungsschritt** zu uxv umgeformt werden kann.

Ableitungen

Eine **Ableitung** ist eine endliche Folge von hintereinander angewendeten Ableitungsschritten.

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

$$w \overset{*}{\Longrightarrow}_G w' \quad \text{(bzw. kurz:} \quad w \overset{*}{\Longrightarrow} w'\text{)},$$

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

Spezialfall: Diese Folge darf auch aus 0 Ableitungsschritten bestehen,

d.h. f.a. $w \in W^*$ gilt: $w \Longrightarrow_G w$.

Ableitungen: Ein Beispiel

Für die Grammatik $G_{AA} = (\Sigma, V, S, P)$ arithmetischer Ausdrücke gilt

$$\mathit{Ausdruck} \overset{*}{\Longrightarrow} (1 + 3) \cdot 2,$$

denn

$$\begin{array}{rcl} P = \{ & \mathit{Ausdruck} \to 1 \mid 2 \mid 3 \ , & \\ & \mathit{Ausdruck} \to \mathit{Ausdruck}\ \mathit{Operator}\ \mathit{Ausdruck} \mid (\ \mathit{Ausdruck}\ ) \ , & \\ & \mathit{Operator} \to + \mid - \mid \cdot & \}. \end{array}$$

und

$$\mathit{Ausdruck} \Longrightarrow \mathit{Ausdruck}\ \mathit{Operator}\ \mathit{Ausdruck}$$

Ableitungen: Ein Beispiel

Für die Grammatik $G_{AA} = (\Sigma, V, S, P)$ arithmetischer Ausdrücke gilt

$$\mathit{Ausdruck} \overset{*}{\Longrightarrow} (1 + 3) \cdot 2,$$

denn

$$\begin{array}{rcll} P = \{ & \mathit{Ausdruck} & \to & 1 \mid 2 \mid 3 \ , \\ & \mathit{Ausdruck} & \to & \mathit{Ausdruck}\ \mathit{Operator}\ \mathit{Ausdruck} \mid (\ \mathit{Ausdruck}\ ) \ , \\ & \mathit{Operator} & \to & + \mid - \mid \cdot \qquad \}. \end{array}$$

und

$$\begin{array}{rcl} \mathit{Ausdruck} & \Longrightarrow & \mathit{Ausdruck}\ \mathit{Operator}\ \mathit{Ausdruck} \\ & \Longrightarrow & (\ \mathit{Ausdruck}\ )\ \mathit{Operator}\ \mathit{Ausdruck} \end{array}$$

Ableitungen: Ein Beispiel

Für die Grammatik $G_{AA} = (\Sigma, V, S, P)$ arithmetischer Ausdrücke gilt

$$\mathit{Ausdruck} \overset{*}{\Longrightarrow} (1 + 3) \cdot 2,$$

denn

$$\begin{array}{lrcl} P = \{ & \mathit{Ausdruck} & \to & 1 \mid 2 \mid 3 \ , \\ & \mathit{Ausdruck} & \to & \mathit{Ausdruck}\ \mathit{Operator}\ \mathit{Ausdruck} \mid (\ \mathit{Ausdruck}\ ) \ , \\ & \mathit{Operator} & \to & + \mid - \mid \cdot \qquad \}. \end{array}$$

und

$$\begin{array}{rcl} \mathit{Ausdruck} & \Longrightarrow & \mathit{Ausdruck}\ \mathit{Operator}\ \mathit{Ausdruck} \\ & \Longrightarrow & (\ \mathit{Ausdruck}\ )\ \mathit{Operator}\ \mathit{Ausdruck} \\ & \Longrightarrow & (\ \mathit{Ausdruck}\ \mathit{Operator}\ \mathit{Ausdruck}\ )\ \mathit{Operator}\ \mathit{Ausdruck} \end{array}$$

Ableitungen: Ein Beispiel

Für die Grammatik $G_{AA} = (\Sigma, V, S, P)$ arithmetischer Ausdrücke gilt

$$\mathit{Ausdruck} \overset{*}{\Longrightarrow} (1 + 3) \cdot 2,$$

denn

$$\begin{array}{lllll} P = \{ & \mathit{Ausdruck} & \to & 1 \mid 2 \mid 3 & , \\ & \mathit{Ausdruck} & \to & \mathit{Ausdruck}\ \mathit{Operator}\ \mathit{Ausdruck} \mid (\ \mathit{Ausdruck}\ ) & , \\ & \mathit{Operator} & \to & + \mid - \mid \cdot & \}. \end{array}$$

und

$$\begin{array}{rcl} \mathit{Ausdruck} & \Longrightarrow & \mathit{Ausdruck}\ \mathit{Operator}\ \mathit{Ausdruck} \\ & \Longrightarrow & (\ \mathit{Ausdruck}\ )\ \mathit{Operator}\ \mathit{Ausdruck} \\ & \Longrightarrow & (\ \mathit{Ausdruck}\ \mathit{Operator}\ \mathit{Ausdruck}\ )\ \mathit{Operator}\ \mathit{Ausdruck} \\ & \Longrightarrow & (\ \mathit{Ausdruck}\ +\ \mathit{Ausdruck}\ )\ \mathit{Operator}\ \mathit{Ausdruck} \end{array}$$

Ableitungen: Ein Beispiel

Für die Grammatik $G_{AA} = (\Sigma, V, S, P)$ arithmetischer Ausdrücke gilt

$$\mathit{Ausdruck} \overset{*}{\Longrightarrow} (1 + 3) \cdot 2,$$

denn

$$\begin{array}{rcll} P = \{ & \mathit{Ausdruck} & \to & 1 \mid 2 \mid 3 \ , \\ & \mathit{Ausdruck} & \to & \mathit{Ausdruck}\ \mathit{Operator}\ \mathit{Ausdruck} \mid (\ \mathit{Ausdruck}\ ) \ , \\ & \mathit{Operator} & \to & + \mid - \mid \cdot \qquad \}. \end{array}$$

und

$$\begin{array}{rcl} \mathit{Ausdruck} & \Longrightarrow & \mathit{Ausdruck}\ \mathit{Operator}\ \mathit{Ausdruck} \\ & \Longrightarrow & (\ \mathit{Ausdruck}\ )\ \mathit{Operator}\ \mathit{Ausdruck} \\ & \Longrightarrow & (\ \mathit{Ausdruck}\ \mathit{Operator}\ \mathit{Ausdruck}\ )\ \mathit{Operator}\ \mathit{Ausdruck} \\ & \Longrightarrow & (\ \mathit{Ausdruck}\ +\ \mathit{Ausdruck}\ )\ \mathit{Operator}\ \mathit{Ausdruck} \\ & \Longrightarrow & (\ \mathit{Ausdruck}\ +\ \mathit{Ausdruck}\ )\ \cdot\ \mathit{Ausdruck} \end{array}$$

Ableitungen: Ein Beispiel

Für die Grammatik $G_{AA} = (\Sigma, V, S, P)$ arithmetischer Ausdrücke gilt

$$\mathit{Ausdruck} \overset{*}{\Longrightarrow} (1 + 3) \cdot 2,$$

denn

$$\begin{array}{lllll} P = \{ & \mathit{Ausdruck} & \to & 1 \mid 2 \mid 3 & , \\ & \mathit{Ausdruck} & \to & \mathit{Ausdruck}\ \mathit{Operator}\ \mathit{Ausdruck} \mid (\ \mathit{Ausdruck}\ ) & , \\ & \mathit{Operator} & \to & + \mid - \mid \cdot & \}. \end{array}$$

und

$$\begin{array}{rcl} \mathit{Ausdruck} & \Longrightarrow & \mathit{Ausdruck}\ \mathit{Operator}\ \mathit{Ausdruck} \\ & \Longrightarrow & (\ \mathit{Ausdruck}\ )\ \mathit{Operator}\ \mathit{Ausdruck} \\ & \Longrightarrow & (\ \mathit{Ausdruck}\ \mathit{Operator}\ \mathit{Ausdruck}\ )\ \mathit{Operator}\ \mathit{Ausdruck} \\ & \Longrightarrow & (\ \mathit{Ausdruck}\ +\ \mathit{Ausdruck}\ )\ \mathit{Operator}\ \mathit{Ausdruck} \\ & \Longrightarrow & (\ \mathit{Ausdruck}\ +\ \mathit{Ausdruck}\ )\ \cdot\ \mathit{Ausdruck} \\ & \Longrightarrow & (\ 1\ +\ \mathit{Ausdruck}\ )\ \cdot\ \mathit{Ausdruck} \end{array}$$

Ableitungen: Ein Beispiel

Für die Grammatik $G_{AA} = (\Sigma, V, S, P)$ arithmetischer Ausdrücke gilt

$$\mathit{Ausdruck} \overset{*}{\Longrightarrow} (1 + 3) \cdot 2,$$

denn

$$\begin{array}{lllll} P = \{ & \mathit{Ausdruck} & \to & 1 \mid 2 \mid 3 & , \\ & \mathit{Ausdruck} & \to & \mathit{Ausdruck}\ \mathit{Operator}\ \mathit{Ausdruck} \mid (\ \mathit{Ausdruck}\ ) & , \\ & \mathit{Operator} & \to & + \mid - \mid \cdot & \}. \end{array}$$

und

$$\begin{array}{rcl} \mathit{Ausdruck} & \Longrightarrow & \mathit{Ausdruck}\ \mathit{Operator}\ \mathit{Ausdruck} \\ & \Longrightarrow & (\ \mathit{Ausdruck}\ )\ \mathit{Operator}\ \mathit{Ausdruck} \\ & \Longrightarrow & (\ \mathit{Ausdruck}\ \mathit{Operator}\ \mathit{Ausdruck}\ )\ \mathit{Operator}\ \mathit{Ausdruck} \\ & \Longrightarrow & (\ \mathit{Ausdruck}\ +\ \mathit{Ausdruck}\ )\ \mathit{Operator}\ \mathit{Ausdruck} \\ & \Longrightarrow & (\ \mathit{Ausdruck}\ +\ \mathit{Ausdruck}\ )\ \cdot\ \mathit{Ausdruck} \\ & \Longrightarrow & (\ 1\ +\ \mathit{Ausdruck}\ )\ \cdot\ \mathit{Ausdruck} \\ & \Longrightarrow & (\ 1\ +\ 3\ )\ \cdot\ \mathit{Ausdruck} \end{array}$$

Ableitungen: Ein Beispiel

Für die Grammatik $G_{AA} = (\Sigma, V, S, P)$ arithmetischer Ausdrücke gilt

$$\mathit{Ausdruck} \overset{*}{\Longrightarrow} (1 + 3) \cdot 2,$$

denn

$$\begin{array}{lllll} P = \{ & \mathit{Ausdruck} & \to & 1 \mid 2 \mid 3 & , \\ & \mathit{Ausdruck} & \to & \mathit{Ausdruck}\ \mathit{Operator}\ \mathit{Ausdruck} \mid (\ \mathit{Ausdruck}\ ) & , \\ & \mathit{Operator} & \to & + \mid - \mid \cdot & \}. \end{array}$$

und

$$\begin{array}{rcl} \mathit{Ausdruck} & \Longrightarrow & \mathit{Ausdruck}\ \mathit{Operator}\ \mathit{Ausdruck} \\ & \Longrightarrow & (\ \mathit{Ausdruck}\ )\ \mathit{Operator}\ \mathit{Ausdruck} \\ & \Longrightarrow & (\ \mathit{Ausdruck}\ \mathit{Operator}\ \mathit{Ausdruck}\ )\ \mathit{Operator}\ \mathit{Ausdruck} \\ & \Longrightarrow & (\ \mathit{Ausdruck}\ +\ \mathit{Ausdruck}\ )\ \mathit{Operator}\ \mathit{Ausdruck} \\ & \Longrightarrow & (\ \mathit{Ausdruck}\ +\ \mathit{Ausdruck}\ )\ \cdot\ \mathit{Ausdruck} \\ & \Longrightarrow & (\ 1\ +\ \mathit{Ausdruck}\ )\ \cdot\ \mathit{Ausdruck} \\ & \Longrightarrow & (\ 1\ +\ 3\ )\ \cdot\ \mathit{Ausdruck} \\ & \Longrightarrow & (\ 1\ +\ 3\ )\ \cdot\ 2 \end{array}$$

Kontextfreie Sprachen

Die von einer KFG G erzeugte Sprache $L(G)$

Sei $G = (\Sigma, V, S, P)$ eine KFG.

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

$$L(G) \;:=\; \left\{ w \in \Sigma^* \;\; : \;\; S \overset{*}{\Longrightarrow}_G w \right\}.$$

Die von einer KFG G erzeugte Sprache $L(G)$

Sei $G = (\Sigma, V, S, P)$ eine KFG.

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

$$L(G) := \left\{ w \in \Sigma^* \quad : \quad S \overset{*}{\Longrightarrow}_G w \right\}.$$

Achtung:
$L(G)$ ist eine Teilmenge von $\Sigma^* \Rightarrow$
In Worten aus $L(G)$ kommen keine Nichtterminale vor!

Wohlgeformte Klammerausdrücke

Sei D die Sprache aller wohl-geformten Klammerausdrücke über $\Sigma = \{ (,) \}$.

Wohlgeformte Klammerausdrücke

Sei D die Sprache aller wohl-geformten Klammerausdrücke über $\Sigma = \{ (,) \}$.

1. **Wohl-geformte Klammerausdrücke** sind
   - (), ((())), (())()()((())), (()())
2. **Nicht wohl-geformt** sind
   - (()(), ((())

Wohlgeformte Klammerausdrücke

Sei D die Sprache aller wohl-geformten Klammerausdrücke über $\Sigma = \{ (,) \}$.

1. **Wohl-geformte Klammerausdrücke** sind
   - (), ((())), (())()()((())), (()())
2. **Nicht wohl-geformt** sind
   - (()(), ((())

Es ist $D = L(G)$ für die kontextfreie Grammatik $G = (\Sigma, \{S\}, S, P)$ mit den Produktionen

$$S \rightarrow$$

Wohlgeformte Klammerausdrücke

Sei D die Sprache aller wohl-geformten Klammerausdrücke über $\Sigma = \{ (,) \}$.

1. **Wohl-geformte Klammerausdrücke** sind
   - (), ((())), (())()()((())), (()())
2. **Nicht wohl-geformt** sind
   - (()(), ((())

Es ist $D = L(G)$ für die kontextfreie Grammatik $G = (\Sigma, \{S\}, S, P)$ mit den Produktionen

$$S \to SS \mid$$

Wohlgeformte Klammerausdrücke

Sei D die Sprache aller wohl-geformten Klammerausdrücke über $\Sigma = \{ (,) \}$.

1. **Wohl-geformte Klammerausdrücke** sind
   - (), ((())), (())()()((())), (()())
2. **Nicht wohl-geformt** sind
   - (()(), ((())

Es ist $D = L(G)$ für die kontextfreie Grammatik $G = (\Sigma, \{S\}, S, P)$ mit den Produktionen

$$S \to SS \mid (S) \mid$$

Wohlgeformte Klammerausdrücke

Sei D die Sprache aller wohl-geformten Klammerausdrücke über $\Sigma = \{ (,) \}$.

1. **Wohl-geformte Klammerausdrücke** sind
   - (), ((())), (())()()((())), (()())
2. **Nicht wohl-geformt** sind
   - (()(), ((())

Es ist $D = L(G)$ für die kontextfreie Grammatik $G = (\Sigma, \{S\}, S, P)$ mit den Produktionen

$$S \to SS \mid (S) \mid \epsilon.$$

Die von G_{AA} erzeugte Sprache

Die Sprache $L(G_{AA})$ besteht aus allen über den Zahlen $1, 2, 3$, den Operatoren $+, -, \cdot$ und den Klammersymbolen $(,)$ wohl-geformten arithmetischen Ausdrücken.

$$
\begin{array}{lllll}
P = \{ & \mathit{Ausdruck} & \to & 1 \mid 2 \mid 3 & , \\
 & \mathit{Ausdruck} & \to & \mathit{Ausdruck}\ \mathit{Operator}\ \mathit{Ausdruck} \mid (\ \mathit{Ausdruck}\ ) & , \\
 & \mathit{Operator} & \to & + \mid - \mid \cdot & \}.
\end{array}
$$

Welche Worte gehören zu $L(G_{AA})$?

? 3

Die von G_{AA} erzeugte Sprache

Die Sprache $L(G_{AA})$ besteht aus allen über den Zahlen $1, 2, 3$, den Operatoren $+, -, \cdot$ und den Klammersymbolen $(,)$ wohl-geformten arithmetischen Ausdrücken.

$$
\begin{array}{rcl}
P = \{ & \mathit{Ausdruck} \rightarrow 1 \mid 2 \mid 3 \ , \\
& \mathit{Ausdruck} \rightarrow \mathit{Ausdruck}\ \mathit{Operator}\ \mathit{Ausdruck} \mid (\ \mathit{Ausdruck}\ ) \ , \\
& \mathit{Operator} \rightarrow + \mid - \mid \cdot \qquad \}.
\end{array}
$$

Welche Worte gehören zu $L(G_{AA})$?

- ? 3
- ? $(3 + 1)$,

Die von G_{AA} erzeugte Sprache

Die Sprache $L(G_{AA})$ besteht aus allen über den Zahlen $1, 2, 3$, den Operatoren $+, -, \cdot$ und den Klammersymbolen $(,)$ wohl-geformten arithmetischen Ausdrücken.

$$
\begin{array}{rcl}
P = \{ & \mathit{Ausdruck} \rightarrow 1 \mid 2 \mid 3 \ , \\
& \mathit{Ausdruck} \rightarrow \mathit{Ausdruck}\ \mathit{Operator}\ \mathit{Ausdruck} \mid (\ \mathit{Ausdruck}\ ) \ , \\
& \mathit{Operator} \rightarrow + \mid - \mid \cdot \qquad \}.
\end{array}
$$

Welche Worte gehören zu $L(G_{AA})$?

- ? 3
- ? $(3 + 1)$,
- ? $()$,

Die von G_{AA} erzeugte Sprache

Die Sprache $L(G_{AA})$ besteht aus allen über den Zahlen $1, 2, 3$, den Operatoren $+, -, \cdot$ und den Klammersymbolen $(,)$ wohl-geformten arithmetischen Ausdrücken.

$$
\begin{array}{lllll}
P = \{ & \textit{Ausdruck} & \rightarrow & 1 \mid 2 \mid 3 & , \\
 & \textit{Ausdruck} & \rightarrow & \textit{Ausdruck Operator Ausdruck} \mid (\ \textit{Ausdruck}\ ) & , \\
 & \textit{Operator} & \rightarrow & + \mid - \mid \cdot & \}.
\end{array}
$$

Welche Worte gehören zu $L(G_{AA})$?

- ? 3
- ? $(3+1)$,
- ? $()$,
- ? $(3+1$,

Die von G_{AA} erzeugte Sprache

Die Sprache $L(G_{AA})$ besteht aus allen über den Zahlen $1, 2, 3$, den Operatoren $+, -, \cdot$ und den Klammersymbolen $(,)$ wohl-geformten arithmetischen Ausdrücken.

$$
\begin{array}{lllll}
P = \{ & \textit{Ausdruck} & \to & 1 \mid 2 \mid 3 & , \\
 & \textit{Ausdruck} & \to & \textit{Ausdruck Operator Ausdruck} \mid (\ \textit{Ausdruck}\ ) & , \\
 & \textit{Operator} & \to & + \mid - \mid \cdot & \}.
\end{array}
$$

Welche Worte gehören zu $L(G_{AA})$?

- ? 3
- ? $(3+1)$,
- ? $()$,
- ? $(3+1$,
- ? $1+2\cdot 3$,

Die von G_{AA} erzeugte Sprache

Die Sprache $L(G_{AA})$ besteht aus allen über den Zahlen $1, 2, 3$, den Operatoren $+, -, \cdot$ und den Klammersymbolen $(,)$ wohl-geformten arithmetischen Ausdrücken.

$$
\begin{array}{rcl}
P = \{ & \mathit{Ausdruck} \rightarrow 1 \mid 2 \mid 3 \ , \\
& \mathit{Ausdruck} \rightarrow \mathit{Ausdruck}\ \mathit{Operator}\ \mathit{Ausdruck} \mid (\ \mathit{Ausdruck}\ ) \ , \\
& \mathit{Operator} \rightarrow + \mid - \mid \cdot \ \}.
\end{array}
$$

Welche Worte gehören zu $L(G_{AA})$?

- ? 3
- ? $(3+1)$,
- ? $()$,
- ? $(3+1$,
- ? $1+2\cdot 3$,
- ? $(3+1)\cdot(2+2+3)-1$,

Die von G_{AA} erzeugte Sprache

Die Sprache $L(G_{AA})$ besteht aus allen über den Zahlen $1, 2, 3$, den Operatoren $+, -, \cdot$ und den Klammersymbolen $(,)$ wohl-geformten arithmetischen Ausdrücken.

$$
\begin{array}{rcl}
P = \{ & \mathit{Ausdruck} \to 1 \mid 2 \mid 3 \ , & \\
& \mathit{Ausdruck} \to \mathit{Ausdruck}\ \mathit{Operator}\ \mathit{Ausdruck} \mid (\ \mathit{Ausdruck}\ ) \ , & \\
& \mathit{Operator} \to + \mid - \mid \cdot & \}.
\end{array}
$$

Welche Worte gehören zu $L(G_{AA})$?

- ? 3
- ? $(3+1)$,
- ? $()$,
- ? $(3+1$,
- ? $1+2\cdot 3$,
- ? $(3+1)\cdot(2+2+3)-1$,
- ? $2\cdot((3+1)\cdot(2+2+3)-1)$,

Die von G_{AA} erzeugte Sprache

Die Sprache $L(G_{AA})$ besteht aus allen über den Zahlen $1, 2, 3$, den Operatoren $+, -, \cdot$ und den Klammersymbolen $(,)$ wohl-geformten arithmetischen Ausdrücken.

$$
\begin{array}{rcl}
P = \{ & \textit{Ausdruck} \rightarrow & 1 \mid 2 \mid 3 \ , \\
& \textit{Ausdruck} \rightarrow & \textit{Ausdruck Operator Ausdruck} \mid (\ \textit{Ausdruck}\ ) \ , \\
& \textit{Operator} \rightarrow & + \mid - \mid \cdot \qquad \}.
\end{array}
$$

Welche Worte gehören zu $L(G_{AA})$?

- ? 3
- ? $(3+1)$,
- ? $()$,
- ? $(3+1$,
- ? $1+2\cdot 3$,
- ? $(3+1)\cdot(2+2+3)-1$,
- ? $2\cdot((3+1)\cdot(2+2+3)-1)$,
- ? $((3+1))$,

Die von G_{AA} erzeugte Sprache

Die Sprache $L(G_{AA})$ besteht aus allen über den Zahlen $1, 2, 3$, den Operatoren $+, -, \cdot$ und den Klammersymbolen $(,)$ wohl-geformten arithmetischen Ausdrücken.

$$
\begin{array}{lrcl}
P = \{ & \textit{Ausdruck} & \rightarrow & 1 \mid 2 \mid 3 \ , \\
 & \textit{Ausdruck} & \rightarrow & \textit{Ausdruck Operator Ausdruck} \mid (\ \textit{Ausdruck}\ ) \ , \\
 & \textit{Operator} & \rightarrow & + \mid - \mid \cdot \qquad \}.
\end{array}
$$

Welche Worte gehören zu $L(G_{AA})$?

- ? 3
- ? $(3+1)$,
- ? $()$,
- ? $(3+1$,
- ? $1+2\cdot 3$,
- ? $(3+1)\cdot(2+2+3)-1$,
- ? $2\cdot((3+1)\cdot(2+2+3)-1)$,
- ? $((3+1))$,
- ? *Ausdruck Operator Ausdruck*

Kontextfreie Sprachen

Eine Sprache $L \subseteq \Sigma^*$ heißt genau dann

kontextfrei,

wenn es eine kontextfreie Grammatik $G = (\Sigma, V, S, P)$ gibt mit

$$L = L(G).$$

KFGs und Programmiersprachen

Ein Fragment von Pascal

Wir beschreiben einen (allerdings sehr kleinen) Ausschnitt von Pascal durch eine kontextfreie Grammatik.

Ein Fragment von Pascal

Wir beschreiben einen (allerdings sehr kleinen) Ausschnitt von Pascal durch eine kontextfreie Grammatik.

- Wir benutzen das Alphabet $\Sigma = \{\texttt{a}, \ldots, \texttt{z}, \texttt{;}, :=, \texttt{begin}, \texttt{end}, \texttt{while}, \texttt{do}\}$ und
- die Variablen S, statements, statement, assign-statement, while-statement, variable, boolean, expression.

Ein Fragment von Pascal

Wir beschreiben einen (allerdings sehr kleinen) Ausschnitt von Pascal durch eine kontextfreie Grammatik.

- Wir benutzen das Alphabet $\Sigma = \{\texttt{a}, \ldots, \texttt{z}, \texttt{;}, :=, \texttt{begin}, \texttt{end}, \texttt{while}, \texttt{do}\}$ und
- die Variablen S, statements, statement, assign-statement, while-statement, variable, boolean, expression.
- variable, boolean und expression sind im Folgenden nicht weiter ausgeführt.

Ein Fragment von Pascal

Wir beschreiben einen (allerdings sehr kleinen) Ausschnitt von Pascal durch eine kontextfreie Grammatik.

- Wir benutzen das Alphabet $\Sigma = \{\texttt{a}, \ldots, \texttt{z}, \texttt{;}, :=, \texttt{begin}, \texttt{end}, \texttt{while}, \texttt{do}\}$ und
- die Variablen S, statements, statement, assign-statement, while-statement, variable, boolean, expression.
- variable, boolean und expression sind im Folgenden nicht weiter ausgeführt.

$$S \quad \rightarrow$$

Ein Fragment von Pascal

Wir beschreiben einen (allerdings sehr kleinen) Ausschnitt von Pascal durch eine kontextfreie Grammatik.

- Wir benutzen das Alphabet $\Sigma = \{\texttt{a}, \ldots, \texttt{z}, \texttt{;}, :=, \texttt{begin}, \texttt{end}, \texttt{while}, \texttt{do}\}$ und
- die Variablen S, statements, statement, assign-statement, while-statement, variable, boolean, expression.
- variable, boolean und expression sind im Folgenden nicht weiter ausgeführt.

$$
\begin{aligned}
S &\to \texttt{begin}\ \text{statements}\ \texttt{end} \\
\text{statements} &\to
\end{aligned}
$$

Ein Fragment von Pascal

Wir beschreiben einen (allerdings sehr kleinen) Ausschnitt von Pascal durch eine kontextfreie Grammatik.

- Wir benutzen das Alphabet $\Sigma = \{\texttt{a}, \ldots, \texttt{z}, \texttt{;}, :=, \texttt{begin}, \texttt{end}, \texttt{while}, \texttt{do}\}$ und
- die Variablen S, statements, statement, assign-statement, while-statement, variable, boolean, expression.
- variable, boolean und expression sind im Folgenden nicht weiter ausgeführt.

$$
\begin{aligned}
S &\to \texttt{begin}\ \text{statements}\ \texttt{end} \\
\text{statements} &\to \text{statement} \mid \text{statement}\ \texttt{;}\ \text{statements} \\
\text{statement} &\to
\end{aligned}
$$

Ein Fragment von Pascal

Wir beschreiben einen (allerdings sehr kleinen) Ausschnitt von Pascal durch eine kontextfreie Grammatik.

- Wir benutzen das Alphabet $\Sigma = \{\texttt{a}, \ldots, \texttt{z}, \texttt{;}, :=, \texttt{begin}, \texttt{end}, \texttt{while}, \texttt{do}\}$ und
- die Variablen S, statements, statement, assign-statement, while-statement, variable, boolean, expression.
- variable, boolean und expression sind im Folgenden nicht weiter ausgeführt.

$$
\begin{aligned}
S &\to \texttt{begin}\ \text{statements}\ \texttt{end} \\
\text{statements} &\to \text{statement} \mid \text{statement}\ \texttt{;}\ \text{statements} \\
\text{statement} &\to \text{assign-statement} \mid \text{while-statement} \\
\text{assign-statement} &\to
\end{aligned}
$$

Ein Fragment von Pascal

Wir beschreiben einen (allerdings sehr kleinen) Ausschnitt von Pascal durch eine kontextfreie Grammatik.

- Wir benutzen das Alphabet $\Sigma = \{\texttt{a}, \ldots, \texttt{z}, \texttt{;}, :=, \texttt{begin}, \texttt{end}, \texttt{while}, \texttt{do}\}$ und
- die Variablen S, statements, statement, assign-statement, while-statement, variable, boolean, expression.
- variable, boolean und expression sind im Folgenden nicht weiter ausgeführt.

$$
\begin{array}{rcl}
S & \to & \texttt{begin}\ \text{statements}\ \texttt{end} \\
\text{statements} & \to & \text{statement} \mid \text{statement ; statements} \\
\text{statement} & \to & \text{assign-statement} \mid \text{while-statement} \\
\text{assign-statement} & \to & \text{variable} := \text{expression} \\
\text{while-statement} & \to &
\end{array}
$$

Ein Fragment von Pascal

Wir beschreiben einen (allerdings sehr kleinen) Ausschnitt von Pascal durch eine kontextfreie Grammatik.

- Wir benutzen das Alphabet $\Sigma = \{\texttt{a}, \ldots, \texttt{z}, \texttt{;}, :=, \texttt{begin}, \texttt{end}, \texttt{while}, \texttt{do}\}$ und
- die Variablen S, statements, statement, assign-statement, while-statement, variable, boolean, expression.
- variable, boolean und expression sind im Folgenden nicht weiter ausgeführt.

$$
\begin{array}{rcl}
S & \to & \texttt{begin}\ \text{statements}\ \texttt{end} \\
\text{statements} & \to & \text{statement} \mid \text{statement ; statements} \\
\text{statement} & \to & \text{assign-statement} \mid \text{while-statement} \\
\text{assign-statement} & \to & \text{variable} := \text{expression} \\
\text{while-statement} & \to & \texttt{while}\ \text{boolean}\ \texttt{do}\ \text{statements}
\end{array}
$$

Programmiersprachen und kontextfreie Sprachen

Lassen sich die syntaktisch korrekten Programme einer modernen Programmiersprache durch eine kontextfreie Sprache definieren?

Programmiersprachen und kontextfreie Sprachen

Lassen sich die syntaktisch korrekten Programme einer modernen Programmiersprache durch eine kontextfreie Sprache definieren?

- 1. Antwort: Nein. In Pascal muss zum Beispiel sichergestellt werden, dass Anzahl und Typen der formalen und aktuellen Parameter übereinstimmen.

Programmiersprachen und kontextfreie Sprachen

Lassen sich die syntaktisch korrekten Programme einer modernen Programmiersprache durch eine kontextfreie Sprache definieren?

- 1. Antwort: Nein. In Pascal muss zum Beispiel sichergestellt werden, dass Anzahl und Typen der formalen und aktuellen Parameter übereinstimmen.
  - Die Sprache $\{ww : w \in \Sigma^*\}$ wird sich als nicht kontextfrei herausstellen.

Programmiersprachen und kontextfreie Sprachen

Lassen sich die syntaktisch korrekten Programme einer modernen Programmiersprache durch eine kontextfreie Sprache definieren?

- 1. Antwort: Nein. In Pascal muss zum Beispiel sichergestellt werden, dass Anzahl und Typen der formalen und aktuellen Parameter übereinstimmen.
  - Die Sprache $\{ww : w \in \Sigma^*\}$ wird sich als nicht kontextfrei herausstellen.
- 2. Antwort: Im Wesentlichen ja, wenn man „Details“ wie Typ-Deklarationen und Typ-Überprüfungen ausklammert:

Programmiersprachen und kontextfreie Sprachen

Lassen sich die syntaktisch korrekten Programme einer modernen Programmiersprache durch eine kontextfreie Sprache definieren?

- 1. Antwort: Nein. In Pascal muss zum Beispiel sichergestellt werden, dass Anzahl und Typen der formalen und aktuellen Parameter übereinstimmen.
  - Die Sprache $\{ww : w \in \Sigma^*\}$ wird sich als nicht kontextfrei herausstellen.
- 2. Antwort: Im Wesentlichen ja, wenn man „Details“ wie Typ-Deklarationen und Typ-Überprüfungen ausklammert:
  - Man beschreibt die Syntax durch eine kontextfreie Grammatik, die alle syntaktisch korrekten Programme erzeugt.

Programmiersprachen und kontextfreie Sprachen

Lassen sich die syntaktisch korrekten Programme einer modernen Programmiersprache durch eine kontextfreie Sprache definieren?

- 1. Antwort: Nein. In Pascal muss zum Beispiel sichergestellt werden, dass Anzahl und Typen der formalen und aktuellen Parameter übereinstimmen.
  - Die Sprache $\{ww : w \in \Sigma^*\}$ wird sich als nicht kontextfrei herausstellen.
- 2. Antwort: Im Wesentlichen ja, wenn man „Details“ wie Typ-Deklarationen und Typ-Überprüfungen ausklammert:
  - Man beschreibt die Syntax durch eine kontextfreie Grammatik, die alle syntaktisch korrekten Programme erzeugt.
  - Allerdings werden auch syntaktisch inkorrekte Progamme (z.B. aufgrund von Typ-Inkonsistenzen) erzeugt.

Die Backus-Naur Normalform

Die Backus-Naur Normalform (BNF) wird zur Formalisierung der Syntax von Programmiersprachen genutzt.

- Sie ist ein „Dialekt“ der kontextfreien Grammatiken.

Die Backus-Naur Normalform

Die Backus-Naur Normalform (BNF) wird zur Formalisierung der Syntax von Programmiersprachen genutzt.

- Sie ist ein „Dialekt“ der kontextfreien Grammatiken.

  Produktionen der Form

  $$X \to aYb$$

  (mit $X, Y \in V$ und $a, b \in \Sigma$) werden in BNF notiert als

  ```
  <X> ::= a <Y> b
  ```

Die Backus-Naur Normalform

Die Backus-Naur Normalform (BNF) wird zur Formalisierung der Syntax von Programmiersprachen genutzt.

- Sie ist ein „Dialekt“ der kontextfreien Grammatiken.

  Produktionen der Form

  $$X \to aYb$$

  (mit $X, Y \in V$ und $a, b \in \Sigma$) werden in BNF notiert als

  ```
  <X> ::= a <Y> b
  ```

- **Beispiel:** Eine Beschreibung der Syntax von Java in einer Variante der BNF auf `http://docs.oracle.com/javase/specs/jls/se8/html/index.html`
  (zuletzt besucht am 29.01.2015)

Die Backus-Naur Normalform

Die Backus-Naur Normalform (BNF) wird zur Formalisierung der Syntax von Programmiersprachen genutzt.

- Sie ist ein „Dialekt“ der kontextfreien Grammatiken.

  Produktionen der Form

  $$X \to aYb$$

  (mit $X, Y \in V$ und $a, b \in \Sigma$) werden in BNF notiert als

  ```
  <X> ::= a <Y> b
  ```

- **Beispiel:** Eine Beschreibung der Syntax von Java in einer Variante der BNF auf `http://docs.oracle.com/javase/specs/jls/se8/html/index.html`
  (zuletzt besucht am 29.01.2015)

Eine effiziente Syntaxanalyse ist möglich.

Frage: Was ist eine Syntaxanalyse?

Die Backus-Naur Normalform

Die Backus-Naur Normalform (BNF) wird zur Formalisierung der Syntax von Programmiersprachen genutzt.

- Sie ist ein „Dialekt“ der kontextfreien Grammatiken.

  Produktionen der Form

  $$X \to aYb$$

  (mit $X, Y \in V$ und $a, b \in \Sigma$) werden in BNF notiert als

  ```
  <X> ::= a <Y> b
  ```

- **Beispiel:** Eine Beschreibung der Syntax von Java in einer Variante der BNF auf `http://docs.oracle.com/javase/specs/jls/se8/html/index.html`
  (zuletzt besucht am 29.01.2015)

Eine effiziente Syntaxanalyse ist möglich.

Frage: Was ist eine Syntaxanalyse?

Antwort: Die Bestimmung einer Ableitung bzw. eines Ableitungsbaums.

Die Backus-Naur Normalform

Die Backus-Naur Normalform (BNF) wird zur Formalisierung der Syntax von Programmiersprachen genutzt.

- Sie ist ein „Dialekt“ der kontextfreien Grammatiken.

  Produktionen der Form

  $$X \to aYb$$

  (mit $X, Y \in V$ und $a, b \in \Sigma$) werden in BNF notiert als

  ```
  <X> ::= a <Y> b
  ```

- **Beispiel:** Eine Beschreibung der Syntax von Java in einer Variante der BNF auf `http://docs.oracle.com/javase/specs/jls/se8/html/index.html`
  (zuletzt besucht am 29.01.2015)

Eine effiziente Syntaxanalyse ist möglich.

Frage: Was ist eine Syntaxanalyse?

Antwort: Die Bestimmung einer Ableitung bzw. eines Ableitungsbaums.

Und was ist ein Ableitungsbaum?

Ableitungsbäume

Ableitungen und Ableitungsbäume (1/2)

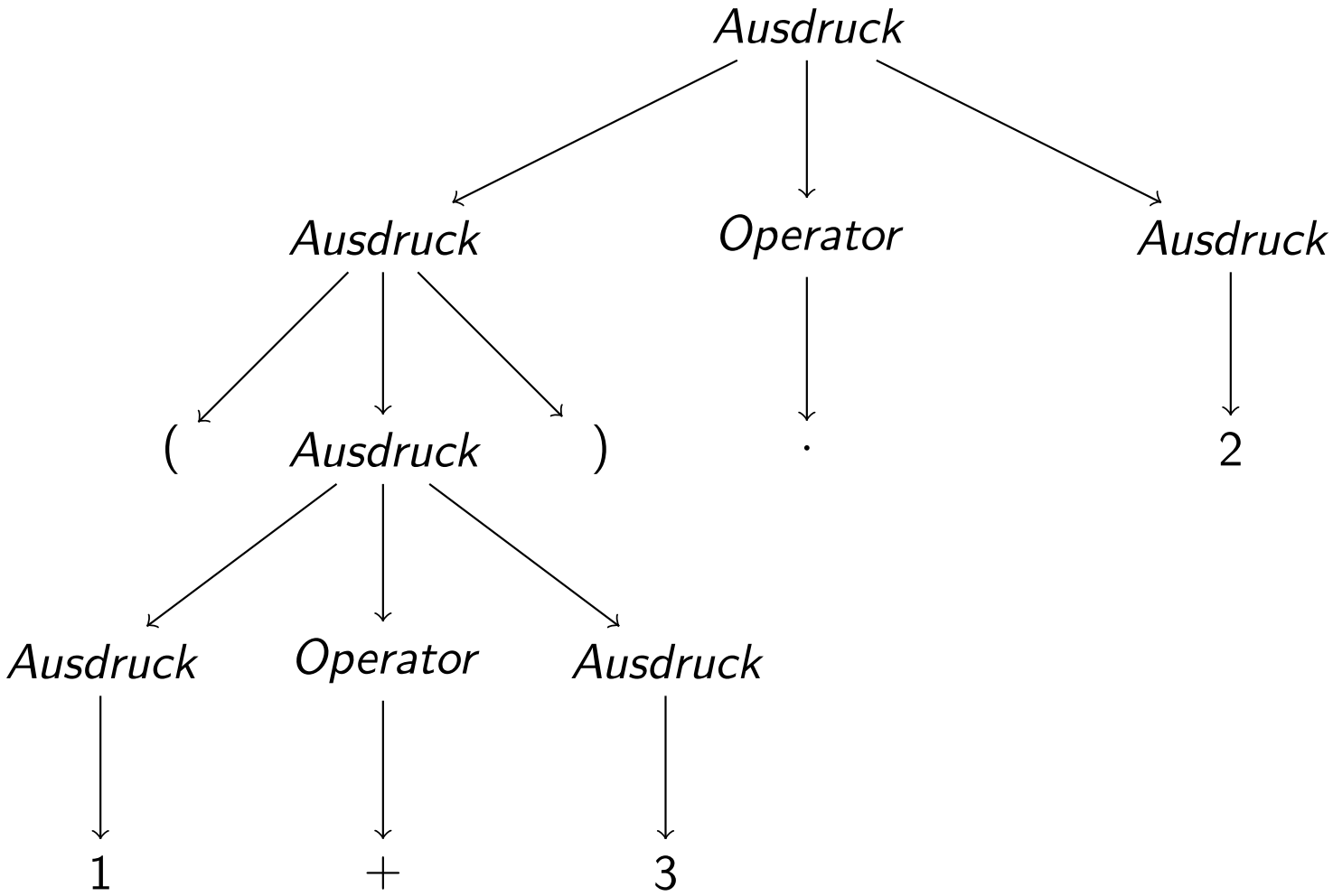
Ableitungen lassen sich am besten mit **Ableitungsbäumen** veranschaulichen.

Betrachte dazu in G_{AA} die Ableitung

$$
\begin{aligned}
\mathit{Ausdruck} &\Longrightarrow \mathit{Ausdruck}\ \mathit{Operator}\ \mathit{Ausdruck} \\
&\Longrightarrow (\ \mathit{Ausdruck}\ )\ \mathit{Operator}\ \mathit{Ausdruck} \\
&\Longrightarrow (\ \mathit{Ausdruck}\ \mathit{Operator}\ \mathit{Ausdruck}\ )\ \mathit{Operator}\ \mathit{Ausdruck} \\
&\Longrightarrow (\ \mathit{Ausdruck}\ +\ \mathit{Ausdruck}\ )\ \mathit{Operator}\ \mathit{Ausdruck} \\
&\Longrightarrow (\ \mathit{Ausdruck}\ +\ \mathit{Ausdruck}\ )\ \cdot\ \mathit{Ausdruck} \\
&\Longrightarrow (\ 1\ +\ \mathit{Ausdruck}\ )\ \cdot\ \mathit{Ausdruck} \\
&\Longrightarrow (\ 1\ +\ 3\ )\ \cdot\ \mathit{Ausdruck} \\
&\Longrightarrow (\ 1\ +\ 3\ )\ \cdot\ 2\ ,
\end{aligned}
$$

Ableitungen und Ableitungsbäume (2/2)

Diese Ableitung hat den folgenden **Ableitungsbaum**:



Beachte: Ein Ableitungsbaum kann mehrere Ableitungen repräsentieren.

Ableitungsbäume: Die formale Definition (1/2)

Sei $G = (\Sigma, V, S, P)$ eine KFG und sei $w \in L(G)$.

Ableitungsbäume: Die formale Definition (1/2)

Sei $G = (\Sigma, V, S, P)$ eine KFG und sei $w \in L(G)$.

Jede Ableitung

$$S \overset{*}{\Longrightarrow}_G w$$

lässt sich als gerichteter Baum darstellen, bei dem

1. jeder Knoten mit einem Symbol aus $\Sigma \cup V \cup \{\varepsilon\}$ markiert ist

Ableitungsbäume: Die formale Definition (1/2)

Sei $G = (\Sigma, V, S, P)$ eine KFG und sei $w \in L(G)$.

Jede Ableitung

$$S \overset{*}{\Longrightarrow}_G w$$

lässt sich als gerichteter Baum darstellen, bei dem

1. jeder Knoten mit einem Symbol aus $\Sigma \cup V \cup \{\varepsilon\}$ markiert ist und
2. 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.
   - Der Ableitungsbaum ist also ein geordneter Baum.

Ableitungsbäume: Die formale Definition (2/2)

3. Die Wurzel des Baums ist mit dem Startsymbol S markiert.

Ableitungsbäume: Die formale Definition (2/2)

3. Die Wurzel des Baums ist mit dem Startsymbol S markiert.
4. Jeder Knoten mit seinen Kindern repräsentiert die Anwendung einer Produktion aus P, also einer Produktion

$$A \to x \text{ mit } A \in V, x \in (V \cup \Sigma)^+.$$

Die Anwendung der Produktion wird im Ableitungsbaum repräsentiert durch einen Knoten v, der mit dem Symbol A markiert ist.

- Wenn $x \in (V \cup \Sigma)^+$, dann hat v genau $|x|$ viele Kinder, so dass das i-te Kind mit dem i-ten Symbol von x markiert ist (f.a. $i \in \{1, \ldots, |x|\}$).

Ableitungsbäume: Die formale Definition (2/2)

3. Die Wurzel des Baums ist mit dem Startsymbol S markiert.
4. Jeder Knoten mit seinen Kindern repräsentiert die Anwendung einer Produktion aus P, also einer Produktion

$$A \to x \text{ mit } A \in V, x \in (V \cup \Sigma)^+.$$

Die Anwendung der Produktion wird im Ableitungsbaum repräsentiert durch einen Knoten v, der mit dem Symbol A markiert ist.

- Wenn $x \in (V \cup \Sigma)^+$, dann hat v genau $|x|$ viele Kinder, so dass das i-te Kind mit dem i-ten Symbol von x markiert ist (f.a. $i \in \{1, \ldots, |x|\}$).
- Wenn $x = \varepsilon$, dann hat v genau ein Kind, das mit ε markiert ist.

Wie versteht ein Compiler ein syntaktisch korrektes Programm p?

Indem der Compiler den Ableitungsbaum von p bestimmt.

Und wenn es mehrere Ableitungsbäume für p gibt?

Die Spezifikation der Pogrammiersprache, –also die KFG im Fall von Java– **muss garantieren**, dass es stets nur einen Ableitungsbaum gibt.

*Solche KFGs heißen **eindeutig**.*

Details: In der Veranstaltung „**Theoretische Informatik 2**“.

Die Aussagenlogik als kontextfreie Sprache (1/3)

Wir konstruieren eine KFG

$$G_{\mathrm{AL}} \;=\; (\Sigma, V, S, P),$$

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

Die Aussagenlogik als kontextfreie Sprache (1/3)

Wir konstruieren eine KFG

$$G_{\mathrm{AL}} \;=\; (\Sigma, V, S, P),$$

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

- **Terminale**:

Die Aussagenlogik als kontextfreie Sprache (1/3)

Wir konstruieren eine KFG

$$G_{\mathrm{AL}} \;=\; (\Sigma, V, S, P),$$

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

- **Terminale**: $\Sigma \;:=\; \{\; V_0,\; V_1,\; V_2,\; \mathbf{0},\; \mathbf{1},\; \neg,\; \wedge,\; \vee,\; \rightarrow,\; \leftrightarrow,\; (,\;)\;\}$,
- **Nichtterminale:**

Die Aussagenlogik als kontextfreie Sprache (1/3)

Wir konstruieren eine KFG

$$G_{\mathrm{AL}} \;=\; (\Sigma, V, S, P),$$

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

- **Terminale**: $\Sigma \;:=\; \{\; V_0,\; V_1,\; V_2,\; \mathbf{0},\; \mathbf{1},\; \neg,\; \wedge,\; \vee,\; \rightarrow,\; \leftrightarrow,\; (,\;)\;\}$,
- **Nichtterminale:** $V := \{\; \mathit{Formel},\; \mathit{Variable},\; \mathit{Junktor}\;\}$,
- **Startsymbol:** $S :=$

Die Aussagenlogik als kontextfreie Sprache (1/3)

Wir konstruieren eine KFG

$$G_{\mathrm{AL}} \;=\; (\Sigma, V, S, P),$$

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

- **Terminale**: $\Sigma \;:=\; \{\; V_0,\; V_1,\; V_2,\; \mathbf{0},\; \mathbf{1},\; \neg,\; \wedge,\; \vee,\; \rightarrow,\; \leftrightarrow,\; (,\;)\;\}$,
- **Nichtterminale:** $V := \{\; \mathit{Formel},\; \mathit{Variable},\; \mathit{Junktor}\;\}$,
- **Startsymbol:** $S := \mathit{Formel}$,
- **Produktionsmenge** $P :=$

$$\{\quad \mathit{Formel} \quad \rightarrow$$

Die Aussagenlogik als kontextfreie Sprache (1/3)

Wir konstruieren eine KFG

$$G_{\mathrm{AL}} \;=\; (\Sigma, V, S, P),$$

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

- **Terminale**: $\Sigma \;:=\; \{\; V_0,\; V_1,\; V_2,\; \mathbf{0},\; \mathbf{1},\; \neg,\; \wedge,\; \vee,\; \rightarrow,\; \leftrightarrow,\; (,\;)\;\}$,
- **Nichtterminale:** $V := \{\; \mathit{Formel},\; \mathit{Variable},\; \mathit{Junktor}\;\}$,
- **Startsymbol:** $S := \mathit{Formel}$,
- **Produktionsmenge** $P :=$

$$\left\{ \begin{array}{lcl} \mathit{Formel} & \rightarrow & \mathbf{0} \mid \mathbf{1} \mid \mathit{Variable}\;, \\ \mathit{Formel} & \rightarrow & \end{array} \right.$$

Die Aussagenlogik als kontextfreie Sprache (1/3)

Wir konstruieren eine KFG

$$G_{\mathrm{AL}} \;=\; (\Sigma, V, S, P),$$

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

- **Terminale**: $\Sigma \;:=\; \{\; V_0,\; V_1,\; V_2,\; \mathbf{0},\; \mathbf{1},\; \neg,\; \wedge,\; \vee,\; \rightarrow,\; \leftrightarrow,\; (,\;)\;\}$,
- **Nichtterminale:** $V := \{\; \mathit{Formel},\; \mathit{Variable},\; \mathit{Junktor}\;\}$,
- **Startsymbol:** $S := \mathit{Formel}$,
- **Produktionsmenge** $P :=$

$$\left\{\begin{array}{rcl} \mathit{Formel} & \rightarrow & \mathbf{0} \mid \mathbf{1} \mid \mathit{Variable}\ , \\ \mathit{Formel} & \rightarrow & \neg\, \mathit{Formel} \mid (\ \mathit{Formel}\ \mathit{Junktor}\ \mathit{Formel}\ )\ , \\ \mathit{Variable} & \rightarrow & \end{array}\right.$$

Die Aussagenlogik als kontextfreie Sprache (1/3)

Wir konstruieren eine KFG

$$G_{\mathrm{AL}} \;=\; (\Sigma, V, S, P),$$

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

- **Terminale**: $\Sigma \;:=\; \{\; V_0,\; V_1,\; V_2,\; \mathbf{0},\; \mathbf{1},\; \neg,\; \wedge,\; \vee,\; \rightarrow,\; \leftrightarrow,\; (,\;)\;\}$,
- **Nichtterminale:** $V := \{\; \mathit{Formel},\; \mathit{Variable},\; \mathit{Junktor}\;\}$,
- **Startsymbol:** $S := \mathit{Formel}$,
- **Produktionsmenge** $P :=$

$$\left\{\begin{array}{rcl} \mathit{Formel} & \rightarrow & \mathbf{0} \mid \mathbf{1} \mid \mathit{Variable}\ , \\ \mathit{Formel} & \rightarrow & \neg\, \mathit{Formel} \mid (\ \mathit{Formel}\ \mathit{Junktor}\ \mathit{Formel}\ )\ , \\ \mathit{Variable} & \rightarrow & V_0 \mid V_1 \mid V_2\ , \\ \mathit{Junktor} & \rightarrow & \end{array}\right.$$

Die Aussagenlogik als kontextfreie Sprache (1/3)

Wir konstruieren eine KFG

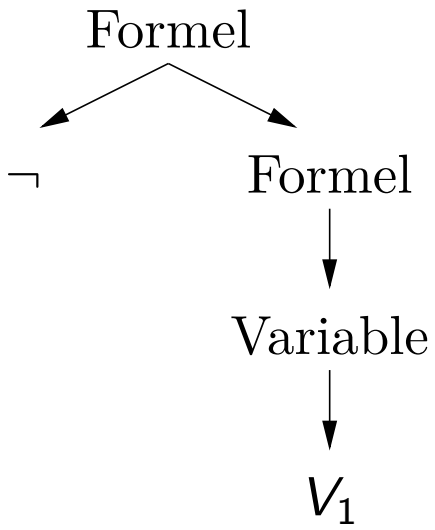
$$G_{\mathrm{AL}} \;=\; (\Sigma, V, S, P),$$

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

- **Terminale**: $\Sigma \;:=\; \{\; V_0,\; V_1,\; V_2,\; \mathbf{0},\; \mathbf{1},\; \neg,\; \wedge,\; \vee,\; \rightarrow,\; \leftrightarrow,\; (,\;)\;\}$,
- **Nichtterminale:** $V := \{\; \mathit{Formel},\; \mathit{Variable},\; \mathit{Junktor}\;\}$,
- **Startsymbol:** $S := \mathit{Formel}$,
- **Produktionsmenge** $P :=$

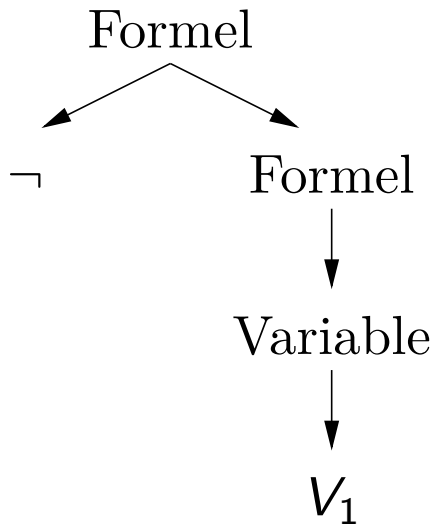
$$\left\{\begin{array}{rcl} \mathit{Formel} & \rightarrow & \mathbf{0} \mid \mathbf{1} \mid \mathit{Variable}\ , \\ \mathit{Formel} & \rightarrow & \neg\, \mathit{Formel} \mid (\ \mathit{Formel}\ \mathit{Junktor}\ \mathit{Formel}\ )\ , \\ \mathit{Variable} & \rightarrow & V_0 \mid V_1 \mid V_2\ , \\ \mathit{Junktor} & \rightarrow & \wedge \mid \vee \mid \rightarrow \mid \leftrightarrow \end{array}\right\}.$$

Die Aussagenlogik als kontextfreie Sprache (2/3)



Der Ableitungsbaum repräsentiert die Ableitung

Die Aussagenlogik als kontextfreie Sprache (2/3)

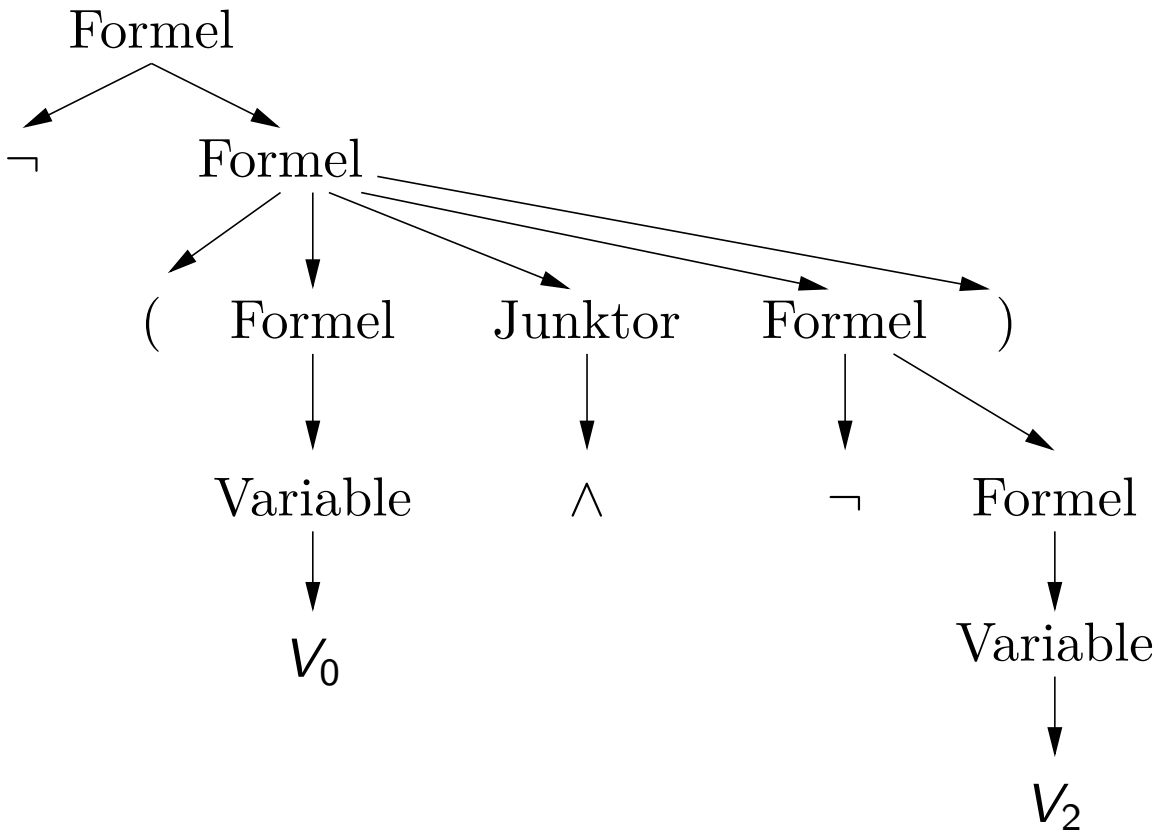


Der Ableitungsbaum repräsentiert die Ableitung

$$\begin{aligned} \textit{Formel} &\Longrightarrow \neg\, \textit{Formel} \\ &\Longrightarrow \neg\, \textit{Variable} \\ &\Longrightarrow \neg V_1 \end{aligned}$$

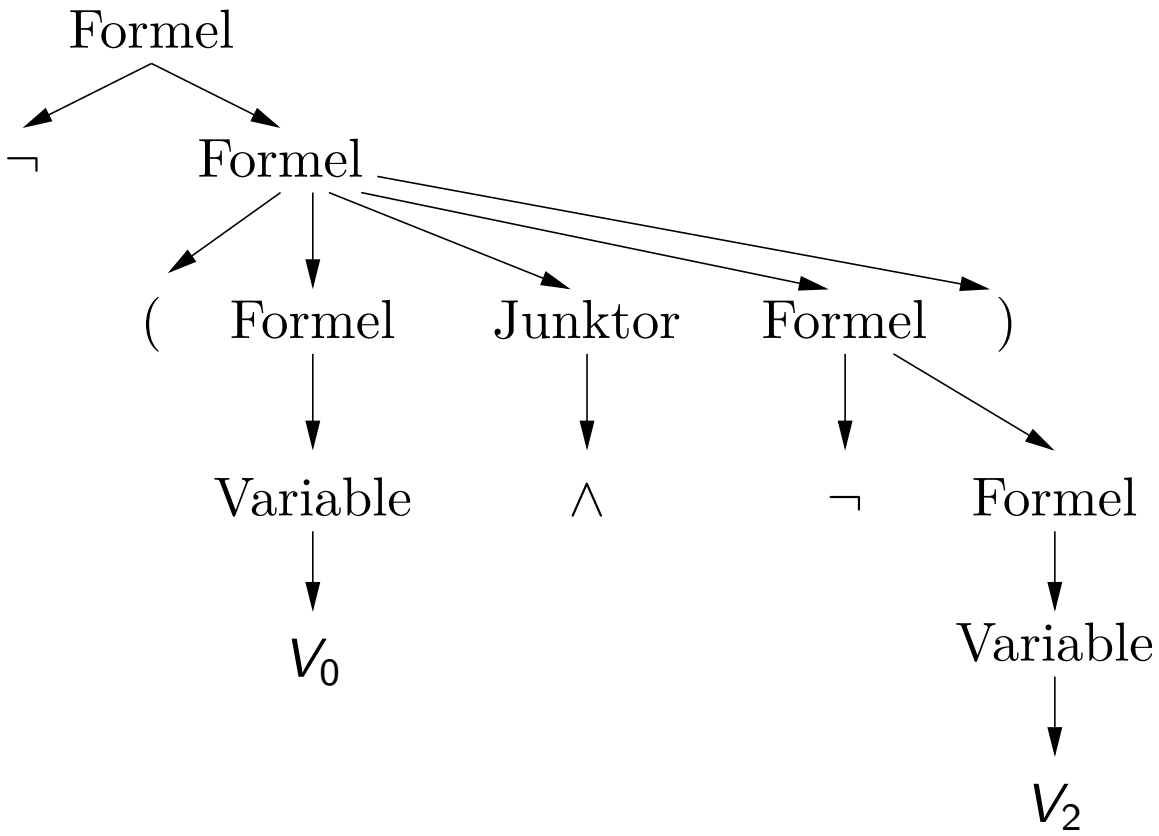
Das durch diese(n) Ableitung(sbaum) erzeugte Wort in der Sprache $L(G_{\mathrm{AL}})$ ist die Formel $\neg V_1$.

Die Aussagenlogik als kontextfreie Sprache (3/3)



Dieser Ableitungsbaum repräsentiert die Ableitung der Formel

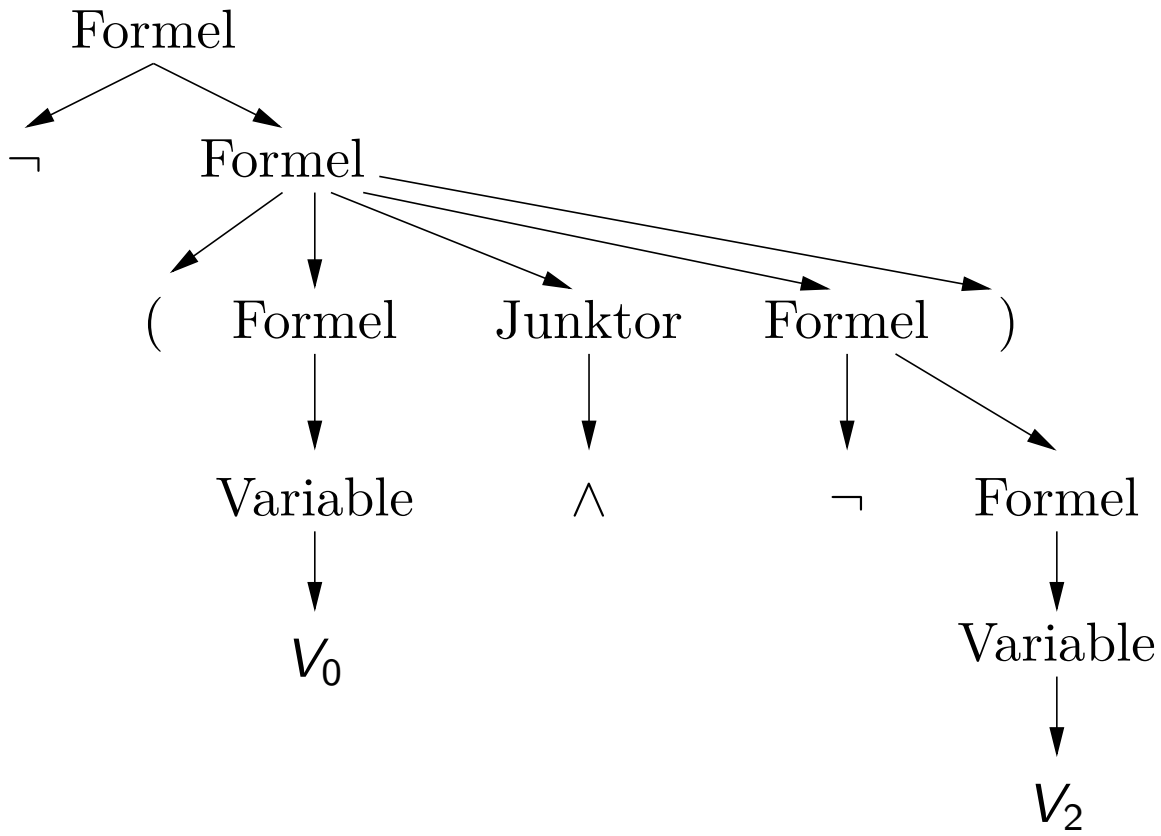
Die Aussagenlogik als kontextfreie Sprache (3/3)



Dieser Ableitungsbaum repräsentiert die Ableitung der Formel $\neg(V_0 \wedge \neg V_2)$

Formel $\Longrightarrow$

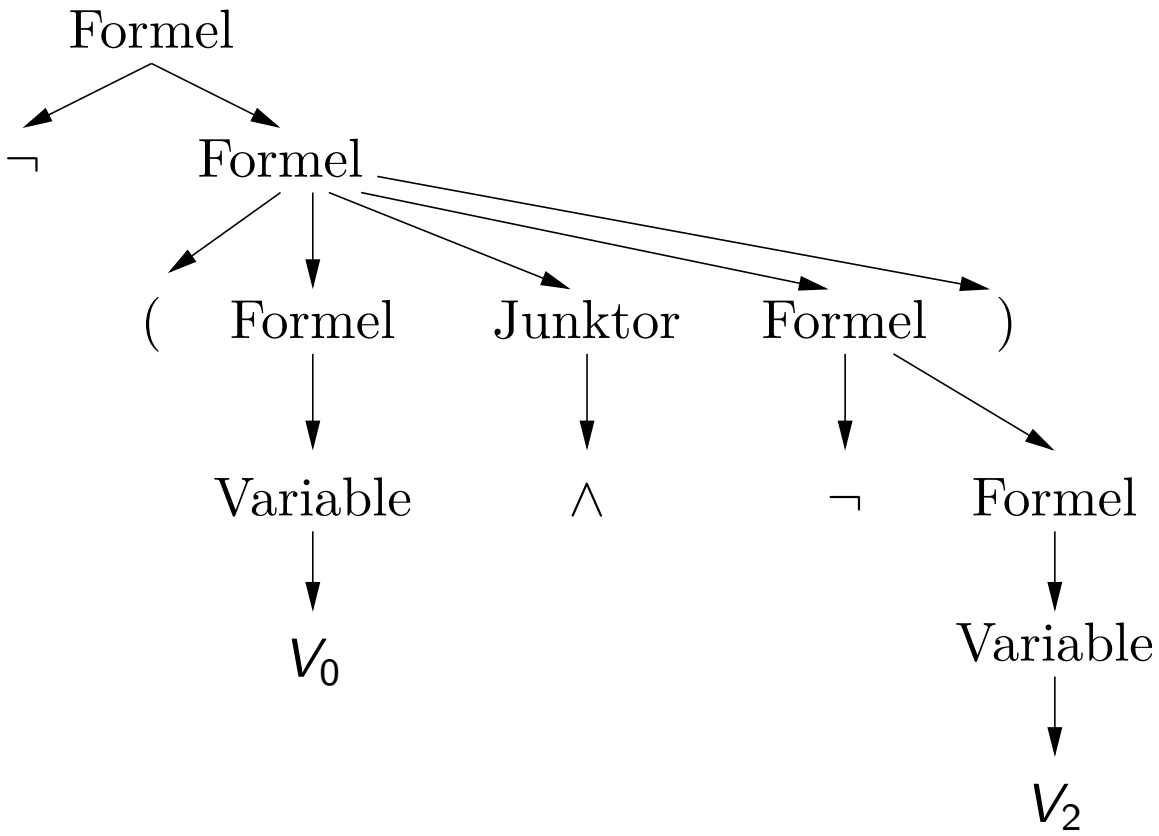
Die Aussagenlogik als kontextfreie Sprache (3/3)



Dieser Ableitungsbaum repräsentiert die Ableitung der Formel $\neg(V_0 \wedge \neg V_2)$

$$\textit{Formel} \implies \neg\, \textit{Formel} \implies$$

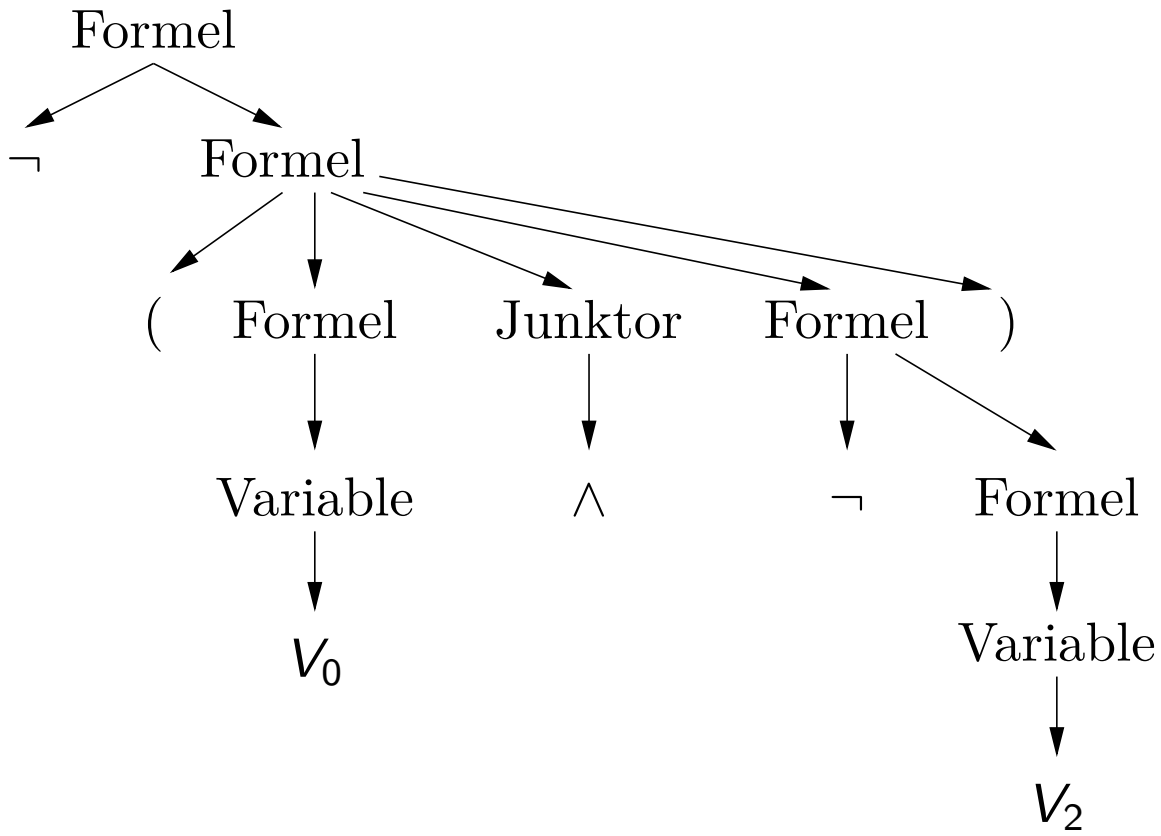
Die Aussagenlogik als kontextfreie Sprache (3/3)



Dieser Ableitungsbaum repräsentiert die Ableitung der Formel $\neg(V_0 \wedge \neg V_2)$

$$\textit{Formel} \implies \neg\, \textit{Formel} \implies \neg\,(\textit{Formel Junktor Formel})$$
$$\implies$$

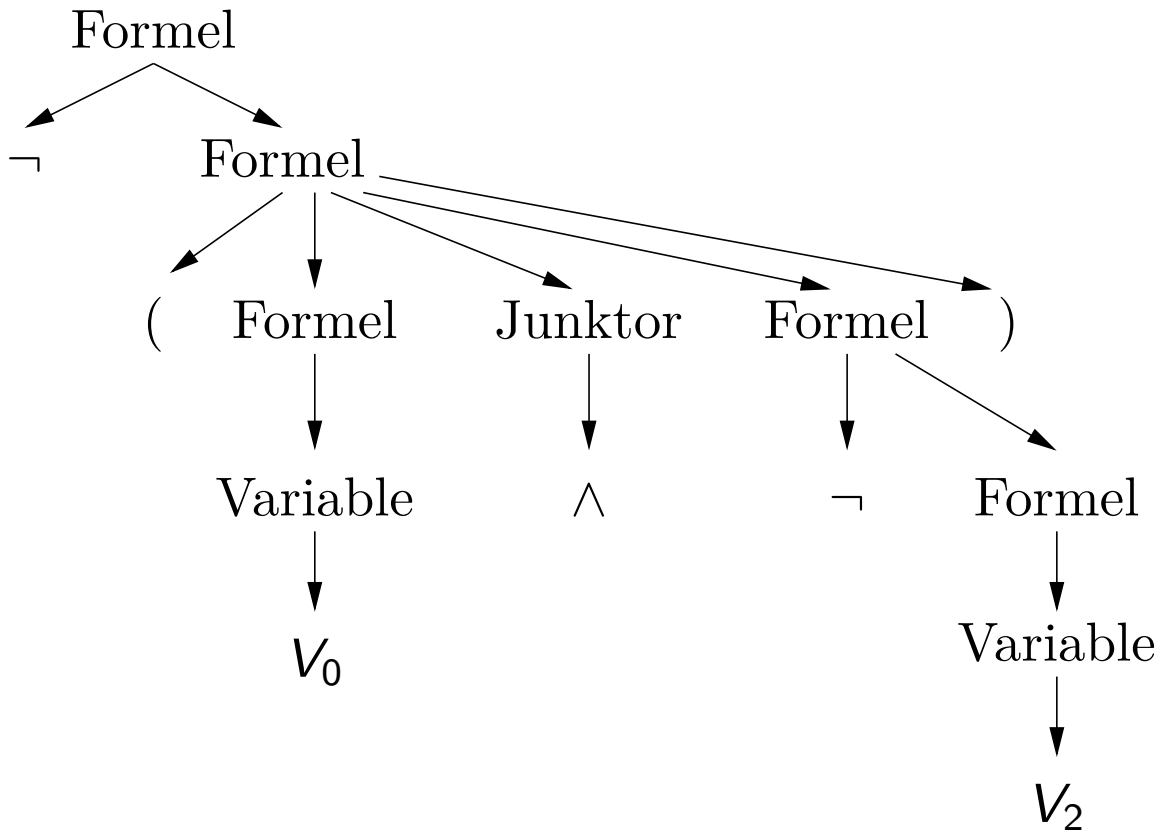
Die Aussagenlogik als kontextfreie Sprache (3/3)



Dieser Ableitungsbaum repräsentiert die Ableitung der Formel $\neg(V_0 \wedge \neg V_2)$

$$\begin{aligned} \textit{Formel} &\Longrightarrow \neg\,\textit{Formel} \Longrightarrow \neg\,(\textit{Formel Junktor Formel}) \\ &\Longrightarrow \neg\,(\textit{Variable Junktor Formel}) \Longrightarrow \end{aligned}$$

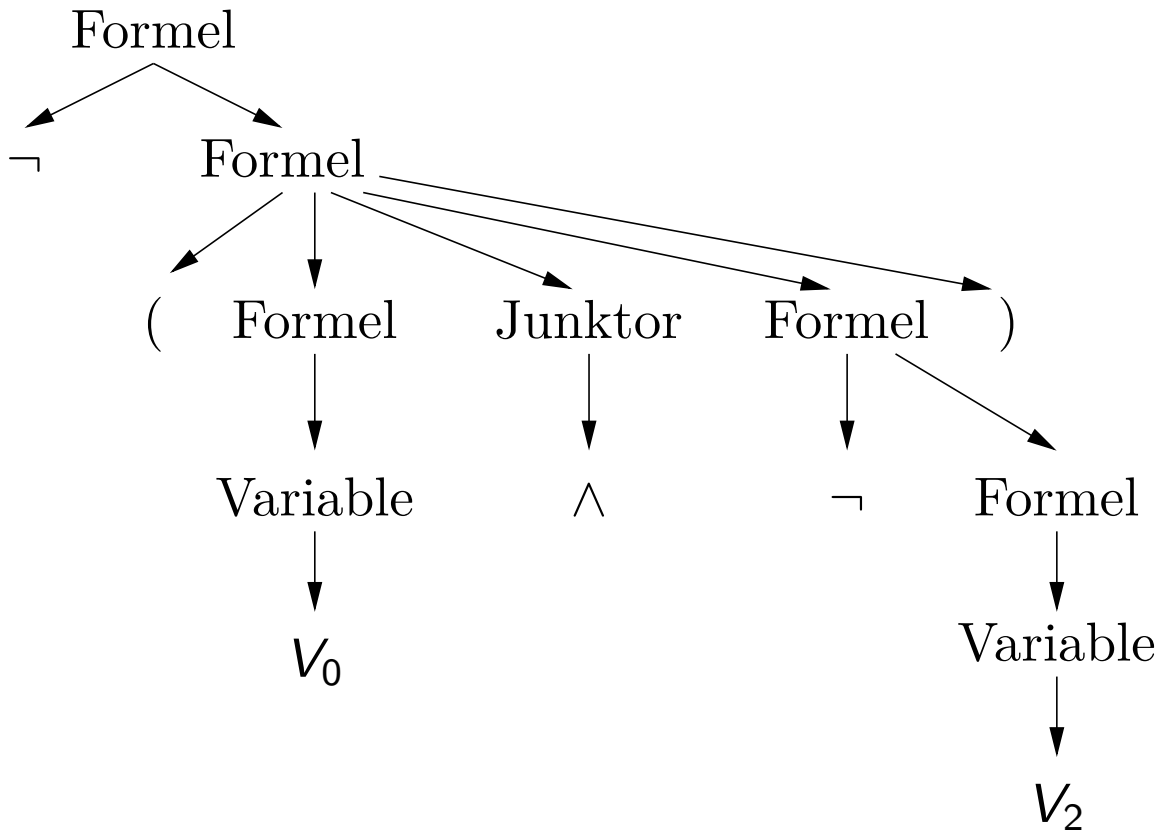
Die Aussagenlogik als kontextfreie Sprache (3/3)



Dieser Ableitungsbaum repräsentiert die Ableitung der Formel $\neg(V_0 \wedge \neg V_2)$

$$\begin{aligned} \textit{Formel} &\Longrightarrow \neg\,\textit{Formel} \Longrightarrow \neg\,(\textit{Formel Junktor Formel}) \\ &\Longrightarrow \neg\,(\textit{Variable Junktor Formel}) \Longrightarrow \neg\,(V_0\ \textit{Junktor Formel}) \\ &\Longrightarrow \end{aligned}$$

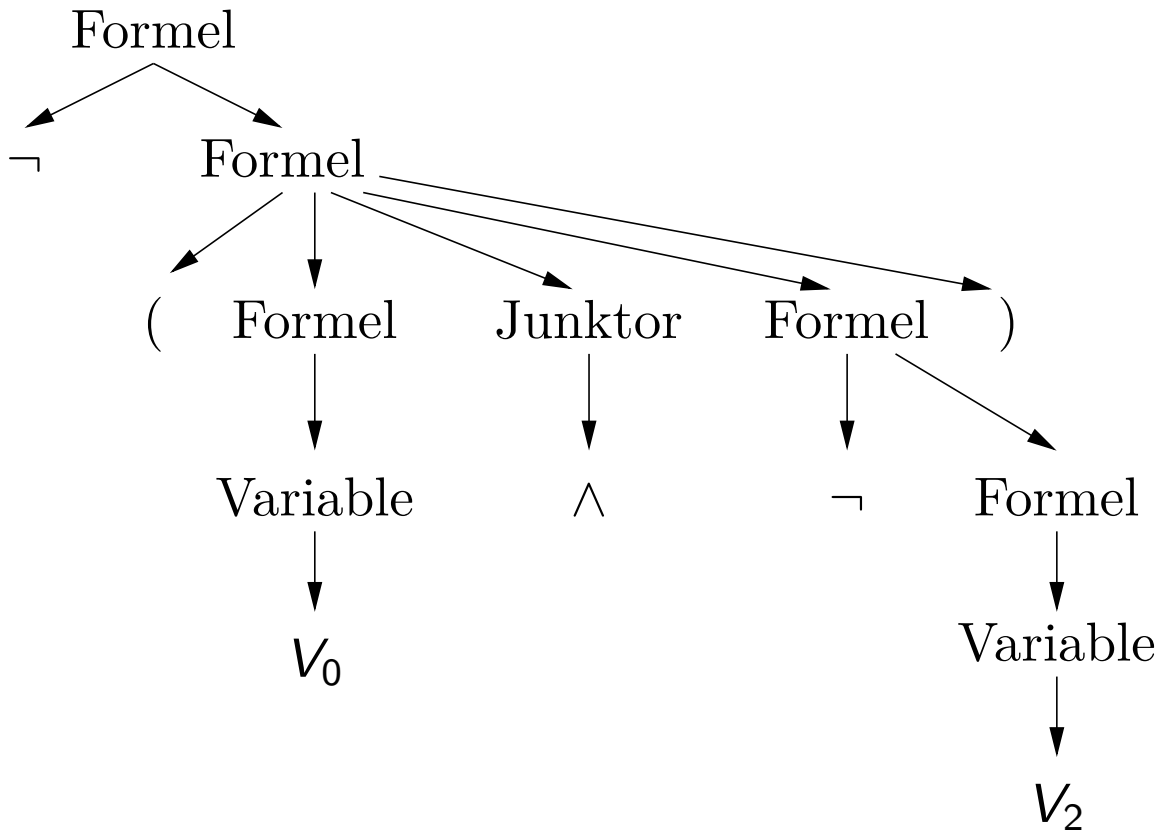
Die Aussagenlogik als kontextfreie Sprache (3/3)



Dieser Ableitungsbaum repräsentiert die Ableitung der Formel $\neg(V_0 \wedge \neg V_2)$

$$
\begin{aligned}
\mathit{Formel} &\Longrightarrow \neg\,\mathit{Formel} \Longrightarrow \neg\,(\mathit{Formel}\ \mathit{Junktor}\ \mathit{Formel}) \\
&\Longrightarrow \neg\,(\mathit{Variable}\ \mathit{Junktor}\ \mathit{Formel}) \Longrightarrow \neg\,(V_0\ \mathit{Junktor}\ \mathit{Formel}) \\
&\Longrightarrow \neg\,(V_0 \wedge \mathit{Formel}) \Longrightarrow
\end{aligned}
$$

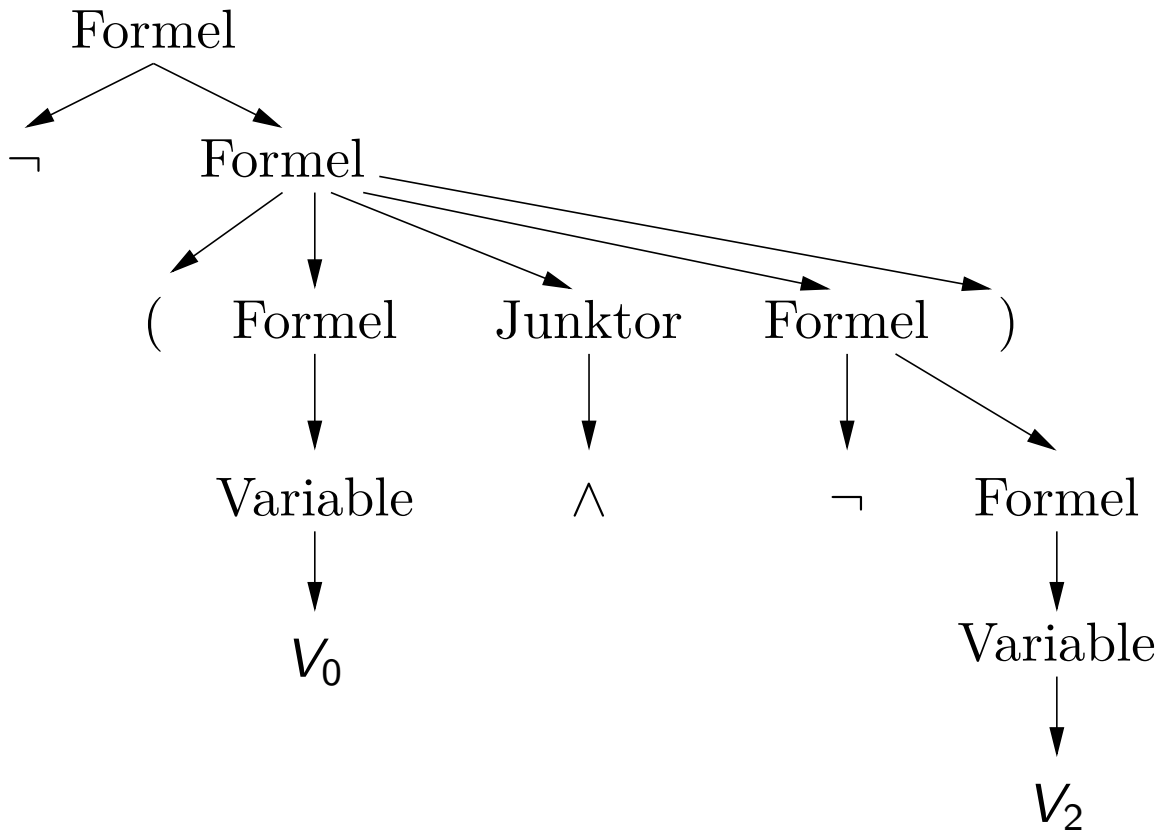
Die Aussagenlogik als kontextfreie Sprache (3/3)



Dieser Ableitungsbaum repräsentiert die Ableitung der Formel $\neg(V_0 \wedge \neg V_2)$

$$
\begin{aligned}
\mathit{Formel} &\Longrightarrow \neg\,\mathit{Formel} \Longrightarrow \neg\,(\mathit{Formel}\ \mathit{Junktor}\ \mathit{Formel}) \\
&\Longrightarrow \neg\,(\mathit{Variable}\ \mathit{Junktor}\ \mathit{Formel}) \Longrightarrow \neg\,(V_0\ \mathit{Junktor}\ \mathit{Formel}) \\
&\Longrightarrow \neg\,(V_0 \wedge \mathit{Formel}) \Longrightarrow \neg\,(V_0 \wedge \neg \mathit{Formel}) \Longrightarrow
\end{aligned}
$$

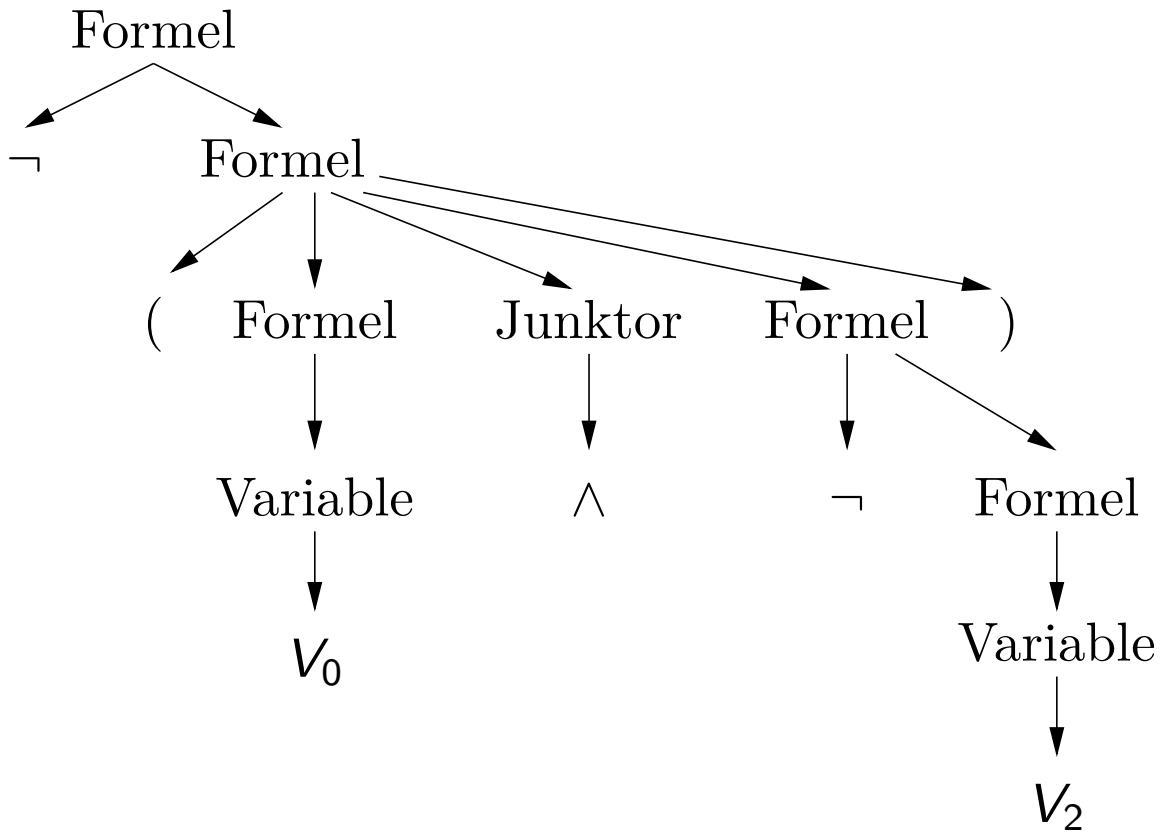
Die Aussagenlogik als kontextfreie Sprache (3/3)



Dieser Ableitungsbaum repräsentiert die Ableitung der Formel $\neg(V_0 \wedge \neg V_2)$

$$
\begin{aligned}
\mathit{Formel} &\Longrightarrow \neg\,\mathit{Formel} \Longrightarrow \neg\,(\mathit{Formel}\ \mathit{Junktor}\ \mathit{Formel}) \\
&\Longrightarrow \neg\,(\mathit{Variable}\ \mathit{Junktor}\ \mathit{Formel}) \Longrightarrow \neg\,(V_0\ \mathit{Junktor}\ \mathit{Formel}) \\
&\Longrightarrow \neg\,(V_0 \wedge \mathit{Formel}) \Longrightarrow \neg\,(V_0 \wedge \neg \mathit{Formel}) \Longrightarrow \neg\,(V_0 \wedge \neg \mathit{Variable}) \\
&\Longrightarrow
\end{aligned}
$$

Die Aussagenlogik als kontextfreie Sprache (3/3)



Dieser Ableitungsbaum repräsentiert die Ableitung der Formel $\neg(V_0 \wedge \neg V_2)$

$$
\begin{aligned}
\mathit{Formel} &\Longrightarrow \neg\,\mathit{Formel} \Longrightarrow \neg\,(\mathit{Formel}\ \mathit{Junktor}\ \mathit{Formel}) \\
&\Longrightarrow \neg\,(\mathit{Variable}\ \mathit{Junktor}\ \mathit{Formel}) \Longrightarrow \neg\,(V_0\ \mathit{Junktor}\ \mathit{Formel}) \\
&\Longrightarrow \neg\,(V_0 \wedge \mathit{Formel}) \Longrightarrow \neg\,(V_0 \wedge \neg \mathit{Formel}) \Longrightarrow \neg\,(V_0 \wedge \neg \mathit{Variable}) \\
&\Longrightarrow \neg\,(V_0 \wedge \neg V_2).
\end{aligned}
$$

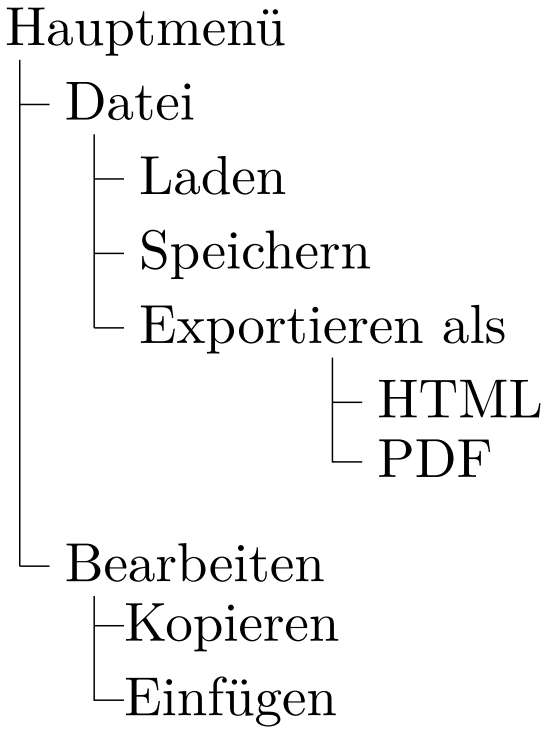
Menüs in Benutzungsoberflächen (1/4)

Ein **Menü** besteht aus einem Menünamen und einer Folge von Einträgen:
Ein Eintrag besteht aus einem Operationsnamen oder selbst wieder einem Menü.

Menüs in Benutzungsoberflächen (1/4)

Ein **Menü** besteht aus einem Menünamen und einer Folge von Einträgen:
Ein Eintrag besteht aus einem Operationsnamen oder selbst wieder einem Menü.

Beispiel:



Menüs in Benutzungsoberflächen (2/4)

Zur Spezifizierung solcher Menüs kann man die Grammatik $G_{\text{Menü}} = (\Sigma, V, S, P)$ verwenden, wobei

- $\Sigma :=$

Menüs in Benutzungsoberflächen (2/4)

Zur Spezifizierung solcher Menüs kann man die Grammatik $G_{\text{Menü}} = (\Sigma, V, S, P)$ verwenden, wobei

- $\Sigma := \{ \texttt{Menüname}, \texttt{Operationsname} \}$,
- $V :=$

Menüs in Benutzungsoberflächen (2/4)

Zur Spezifizierung solcher Menüs kann man die Grammatik $G_{\text{Menü}} = (\Sigma, V, S, P)$ verwenden, wobei

- $\Sigma := \{ \texttt{Menüname}, \texttt{Operationsname} \}$,
- $V := \{ \textit{Menü}, \textit{Eintragsfolge}, \textit{Eintrag} \}$,
- $S :=$

Menüs in Benutzungsoberflächen (2/4)

Zur Spezifizierung solcher Menüs kann man die Grammatik $G_{\text{Menü}} = (\Sigma, V, S, P)$ verwenden, wobei

- $\Sigma := \{ \texttt{Menüname}, \texttt{Operationsname} \}$,
- $V := \{ \textit{Menü}, \textit{Eintragsfolge}, \textit{Eintrag} \}$,
- $S := \textit{Menü}$,
- $P := \{ \quad \textit{Menü} \quad \rightarrow$

Menüs in Benutzungsoberflächen (2/4)

Zur Spezifizierung solcher Menüs kann man die Grammatik $G_{\text{Menü}} = (\Sigma, V, S, P)$ verwenden, wobei

- $\Sigma := \{ \texttt{Menüname}, \texttt{Operationsname} \}$,
- $V := \{ \textit{Menü}, \textit{Eintragsfolge}, \textit{Eintrag} \}$,
- $S := \textit{Menü}$,
- $P := \{ \quad \textit{Menü} \rightarrow \texttt{Menüname}\ \textit{Eintragsfolge}\ ,$
  $\textit{Eintragsfolge} \rightarrow$

Menüs in Benutzungsoberflächen (2/4)

Zur Spezifizierung solcher Menüs kann man die Grammatik $G_{\text{Menü}} = (\Sigma, V, S, P)$ verwenden, wobei

- $\Sigma := \{ \texttt{Menüname}, \texttt{Operationsname} \}$,
- $V := \{ \textit{Menü}, \textit{Eintragsfolge}, \textit{Eintrag} \}$,
- $S := \textit{Menü}$,
- $P := \{$
$$\begin{aligned} \textit{Menü} &\to \texttt{Menüname}\ \textit{Eintragsfolge}\ , \\ \textit{Eintragsfolge} &\to \textit{Eintrag} \mid \textit{Eintrag}\ \textit{Eintragsfolge}\ , \\ \textit{Eintrag} &\to \end{aligned}$$

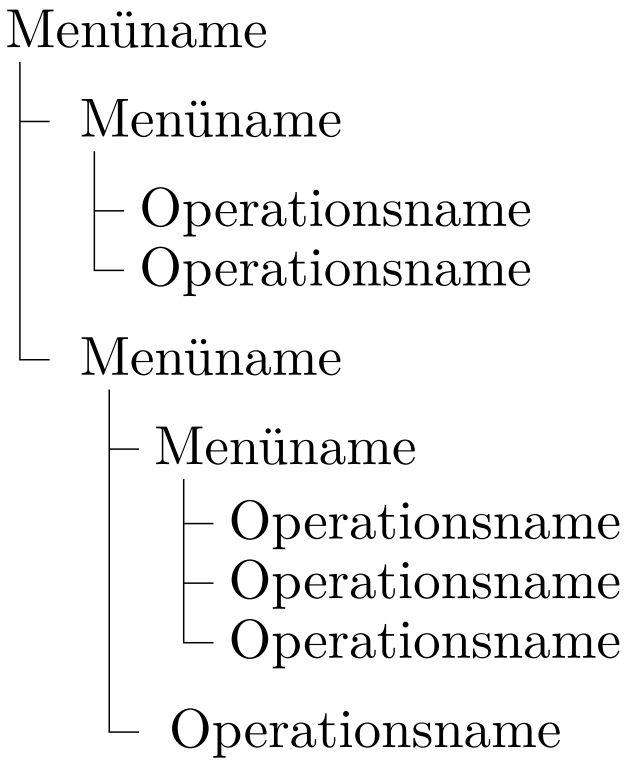
Menüs in Benutzungsoberflächen (2/4)

Zur Spezifizierung solcher Menüs kann man die Grammatik $G_{\text{Menü}} = (\Sigma, V, S, P)$ verwenden, wobei

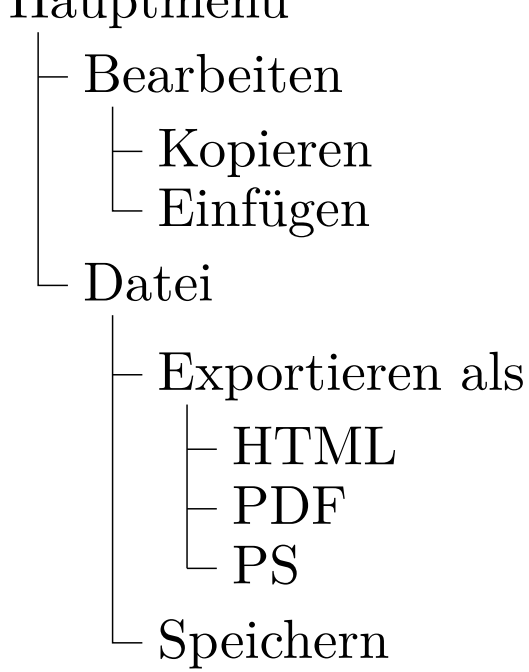
- $\Sigma := \{ \texttt{Menüname}, \texttt{Operationsname} \}$,
- $V := \{ \textit{Menü}, \textit{Eintragsfolge}, \textit{Eintrag} \}$,
- $S := \textit{Menü}$,
- $P := \{$
$$\begin{aligned} \textit{Menü} &\to \texttt{Menüname}\ \textit{Eintragsfolge}\ , \\ \textit{Eintragsfolge} &\to \textit{Eintrag} \mid \textit{Eintrag}\ \textit{Eintragsfolge}\ , \\ \textit{Eintrag} &\to \texttt{Operationsname} \mid \textit{Menü} \quad \}. \end{aligned}$$

Menüs in Benutzungsoberflächen (3/4)

Die Struktur eines Menüs:

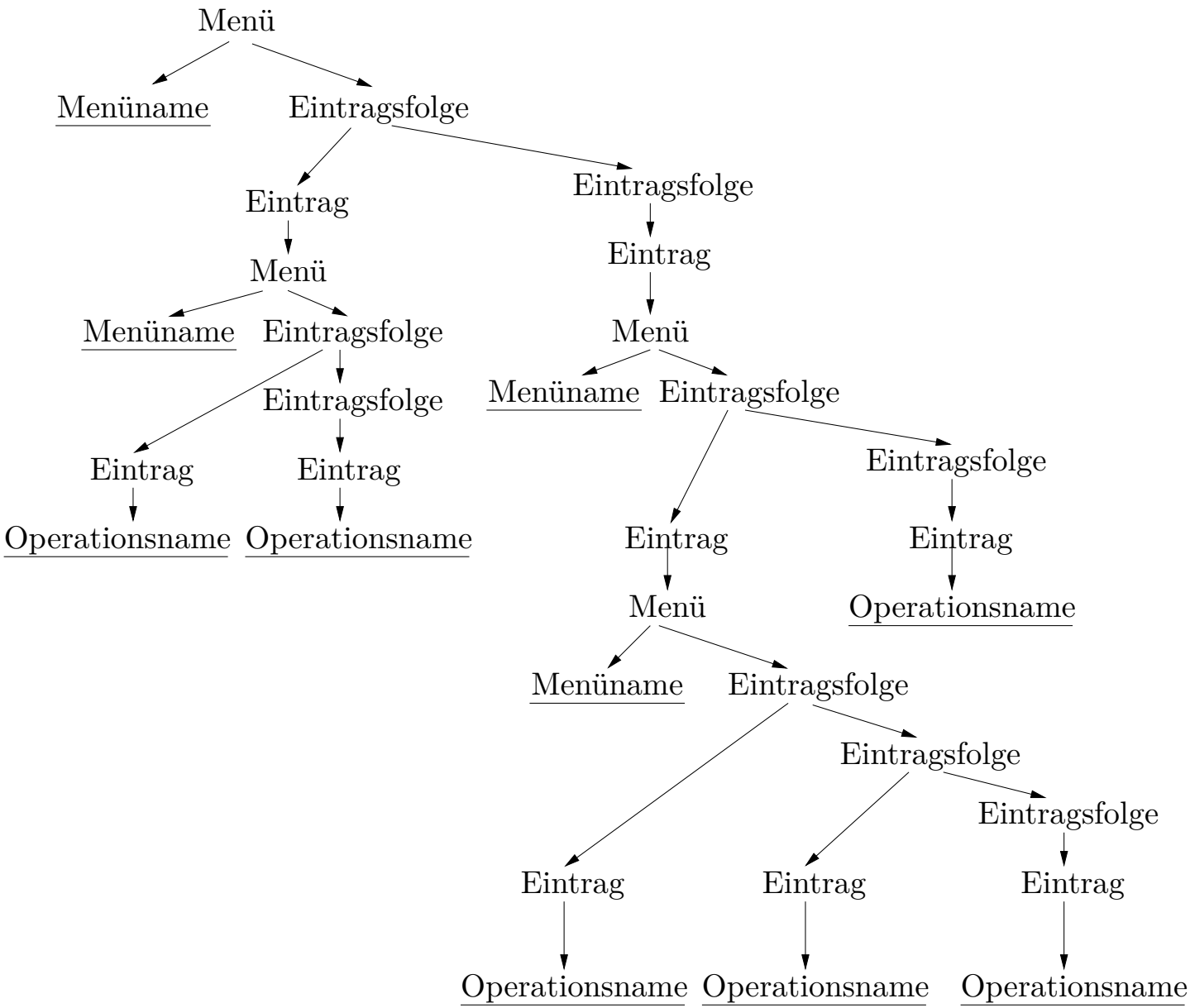


Beispiel für Namen:



Und wie sieht der Ableitungsbaum für das Beispiel aus?

Menüs in Benutzungsoberflächen (4/4)



HTML-Tabellen und KFGs

HTML (HyperText Markup Language)
ist ein Format zur Beschreibung von verzweigten Dokumenten im Internet.

Uns interessiert hier der HTML-Code zur Erzeugung von HTML-Tabellen.

HTML-Tabellen: Ein Beispiel

Die HTML-Tabelle

Tag	Zeit	Raum
Di	8:00-10:00	Hoersaal VI
Do	8:00-10:00	Hoersaal VI

besitzt den HTML-Code:

```
<table>
  <tr>
    <td> Tag </td>
    <td> Zeit </td>
    <td> Raum </td>
  </tr>
  <tr>
    <td> Di </td>
    <td> 8:00-10:00 </td>
    <td> Hoersaal VI </td>
  </tr>
  <tr>
    <td> Do </td>
    <td> 8:00-10:00 </td>
    <td> Hoersaal VI </td>
  </tr>
</table>
```

Die Symbole `<table>` und `</table>`, `<tr>` und `</tr>` bzw. `<td>` und `</td>` stehen für Anfang und Ende einer **Tabelle**, für Anfang und Ende einer **Zeile der Tabelle** bzw. für Anfang und Ende eines **Eintrags in einer Zelle der Tabelle**.

Eine KFG zur Erzeugung von HTML-Tabellen

Wir konstruieren eine Grammatik $G_{\mathrm{HTML}} = (\Sigma, V, S, P)$, so dass G_{HTML} genau die (möglicherweise geschachtelten) HTML-Tabellen erzeugt.

Eine KFG zur Erzeugung von HTML-Tabellen

Wir konstruieren eine Grammatik $G_{\mathrm{HTML}} = (\Sigma, V, S, P)$, so dass G_{HTML} genau die (möglicherweise geschachtelten) HTML-Tabellen erzeugt.

- $\Sigma := \{$

Eine KFG zur Erzeugung von HTML-Tabellen

Wir konstruieren eine Grammatik $G_{\mathrm{HTML}} = (\Sigma, V, S, P)$, so dass G_{HTML} genau die (möglicherweise geschachtelten) HTML-Tabellen erzeugt.

- $\Sigma := \{$ `<table>`, `</table>`, `<tr>`, `</tr>`, `<td>`, `</td>`, $\mathtt{a}, \ldots, \mathtt{z}$, $\mathtt{A}, \ldots, \mathtt{Z}$, $0, 1, \ldots, 9$, :, -, $\sqcup$ $\}$
- $V := \{$

Eine KFG zur Erzeugung von HTML-Tabellen

Wir konstruieren eine Grammatik $G_{\mathrm{HTML}} = (\Sigma, V, S, P)$, so dass G_{HTML} genau die (möglicherweise geschachtelten) HTML-Tabellen erzeugt.

- $\Sigma := \{$ `<table>`, `</table>`, `<tr>`, `</tr>`, `<td>`, `</td>`, $\mathtt{a}, \ldots, \mathtt{z}$, $\mathtt{A}, \ldots, \mathtt{Z}$, $\mathtt{0}, \mathtt{1}, \ldots, \mathtt{9}$, :, -, $\sqcup$ $\}$
- $V := \{$ *Tabelle*, *Zeilen*, *Zeile*, *Einträge*, *Eintrag*, *Text*, $\}$
- $S :=$

Eine KFG zur Erzeugung von HTML-Tabellen

Wir konstruieren eine Grammatik $G_{\mathrm{HTML}} = (\Sigma, V, S, P)$, so dass G_{HTML} genau die (möglicherweise geschachtelten) HTML-Tabellen erzeugt.

- $\Sigma := \{$ `<table>`, `</table>`, `<tr>`, `</tr>`, `<td>`, `</td>`, $\mathtt{a}, \ldots, \mathtt{z}$, $\mathtt{A}, \ldots, \mathtt{Z}$, $0, 1, \ldots, 9$, :, -, $\sqcup$ $\}$
- $V := \{$ *Tabelle*, *Zeilen*, *Zeile*, *Einträge*, *Eintrag*, *Text*, $\}$
- $S :=$ *Tabelle*
- $P :=$
  $\{$ *Tabelle* $\rightarrow$

Eine KFG zur Erzeugung von HTML-Tabellen

Wir konstruieren eine Grammatik $G_{\mathrm{HTML}} = (\Sigma, V, S, P)$, so dass G_{HTML} genau die (möglicherweise geschachtelten) HTML-Tabellen erzeugt.

- $\Sigma := \{$ `<table>`, `</table>`, `<tr>`, `</tr>`, `<td>`, `</td>`, $\mathtt{a}, \ldots, \mathtt{z}$, $\mathtt{A}, \ldots, \mathtt{Z}$, $0, 1, \ldots, 9$, :, -, $\sqcup$ $\}$
- $V := \{$ *Tabelle*, *Zeilen*, *Zeile*, *Einträge*, *Eintrag*, *Text*, $\}$
- $S :=$ *Tabelle*
- $P :=$
  $\{$ *Tabelle* $\rightarrow$ `<table>` *Zeilen* `</table>` ,
  Zeilen $\rightarrow$

Eine KFG zur Erzeugung von HTML-Tabellen

Wir konstruieren eine Grammatik $G_{\mathrm{HTML}} = (\Sigma, V, S, P)$, so dass G_{HTML} genau die (möglicherweise geschachtelten) HTML-Tabellen erzeugt.

- $\Sigma := \{$ `<table>`, `</table>`, `<tr>`, `</tr>`, `<td>`, `</td>`, $\mathtt{a}, \ldots, \mathtt{z}$, $\mathtt{A}, \ldots, \mathtt{Z}$, $0, 1, \ldots, 9$, :, -, $\sqcup$ $\}$
- $V := \{$ *Tabelle*, *Zeilen*, *Zeile*, *Einträge*, *Eintrag*, *Text*, $\}$
- $S :=$ *Tabelle*
- $P :=$

$$\left\{ \begin{array}{rcl} \textit{Tabelle} & \rightarrow & \texttt{<table>}\ \textit{Zeilen}\ \texttt{</table>}\ , \\ \textit{Zeilen} & \rightarrow & \textit{Zeile} \mid \textit{Zeile}\ \textit{Zeilen}\ , \\ \textit{Zeile} & \rightarrow & \end{array} \right.$$

Eine KFG zur Erzeugung von HTML-Tabellen

Wir konstruieren eine Grammatik $G_{\mathrm{HTML}} = (\Sigma, V, S, P)$, so dass G_{HTML} genau die (möglicherweise geschachtelten) HTML-Tabellen erzeugt.

- $\Sigma := \{$ `<table>`, `</table>`, `<tr>`, `</tr>`, `<td>`, `</td>`, $\mathtt{a}, \ldots, \mathtt{z}$, $\mathtt{A}, \ldots, \mathtt{Z}$, $0, 1, \ldots, 9$, :, -, $\sqcup$ $\}$
- $V := \{$ *Tabelle*, *Zeilen*, *Zeile*, *Einträge*, *Eintrag*, *Text*, $\}$
- $S :=$ *Tabelle*
- $P :=$

$$\left\{ \begin{array}{rcl} \textit{Tabelle} & \to & \texttt{<table>}\ \textit{Zeilen}\ \texttt{</table>}\ , \\ \textit{Zeilen} & \to & \textit{Zeile} \mid \textit{Zeile}\ \textit{Zeilen}\ , \\ \textit{Zeile} & \to & \texttt{<tr>}\ \text{Einträge}\ \texttt{</tr>}\ , \\ \textit{Einträge} & \to & \end{array} \right.$$

Eine KFG zur Erzeugung von HTML-Tabellen

Wir konstruieren eine Grammatik $G_{\mathrm{HTML}} = (\Sigma, V, S, P)$, so dass G_{HTML} genau die (möglicherweise geschachtelten) HTML-Tabellen erzeugt.

- $\Sigma := \{$ `<table>`, `</table>`, `<tr>`, `</tr>`, `<td>`, `</td>`, $\mathtt{a}, \ldots, \mathtt{z}$, $\mathtt{A}, \ldots, \mathtt{Z}$, $0, 1, \ldots, 9$, :, -, $\sqcup$ $\}$
- $V := \{$ *Tabelle*, *Zeilen*, *Zeile*, *Einträge*, *Eintrag*, *Text*, $\}$
- $S :=$ *Tabelle*
- $P :=$

$$\left\{ \begin{array}{rcl} \textit{Tabelle} & \to & \texttt{<table>}\ \textit{Zeilen}\ \texttt{</table>}\ , \\ \textit{Zeilen} & \to & \textit{Zeile} \mid \textit{Zeile}\ \textit{Zeilen}\ , \\ \textit{Zeile} & \to & \texttt{<tr>}\ \text{Einträge}\ \texttt{</tr>}\ , \\ \textit{Einträge} & \to & \textit{Eintrag} \mid \textit{Eintrag}\ \textit{Einträge}\ , \\ \textit{Eintrag} & \to & \end{array} \right.$$

Eine KFG zur Erzeugung von HTML-Tabellen

Wir konstruieren eine Grammatik $G_{\mathrm{HTML}} = (\Sigma, V, S, P)$, so dass G_{HTML} genau die (möglicherweise geschachtelten) HTML-Tabellen erzeugt.

- $\Sigma := \{$ `<table>`, `</table>`, `<tr>`, `</tr>`, `<td>`, `</td>`, $\mathtt{a}, \ldots, \mathtt{z}$, $\mathtt{A}, \ldots, \mathtt{Z}$, $0, 1, \ldots, 9$, :, -, $\sqcup$ $\}$
- $V := \{$ *Tabelle*, *Zeilen*, *Zeile*, *Einträge*, *Eintrag*, *Text*, $\}$
- $S :=$ *Tabelle*
- $P :=$

$$\left\{ \begin{array}{rcl} \textit{Tabelle} & \to & \texttt{<table>}\ \textit{Zeilen}\ \texttt{</table>}\ , \\ \textit{Zeilen} & \to & \textit{Zeile} \mid \textit{Zeile}\ \textit{Zeilen}\ , \\ \textit{Zeile} & \to & \texttt{<tr>}\ \text{Einträge}\ \texttt{</tr>}\ , \\ \textit{Einträge} & \to & \textit{Eintrag} \mid \textit{Eintrag}\ \textit{Einträge}\ , \\ \textit{Eintrag} & \to & \texttt{<td>}\ \textit{Text}\ \texttt{</td>} \mid \texttt{<td>}\ \textit{Tabelle}\ \texttt{</td>}\ , \\ \textit{Text} & \to & \end{array} \right.$$

Eine KFG zur Erzeugung von HTML-Tabellen

Wir konstruieren eine Grammatik $G_{\mathrm{HTML}} = (\Sigma, V, S, P)$, so dass G_{HTML} genau die (möglicherweise geschachtelten) HTML-Tabellen erzeugt.

- $\Sigma := \{$ `<table>`, `</table>`, `<tr>`, `</tr>`, `<td>`, `</td>`, $\mathtt{a}, \ldots, \mathtt{z}$, $\mathtt{A}, \ldots, \mathtt{Z}$, $\mathtt{0}, \mathtt{1}, \ldots, \mathtt{9}$, `:`, `-`, $\mathtt{\sqcup}$ $\}$
- $V := \{$ *Tabelle*, *Zeilen*, *Zeile*, *Einträge*, *Eintrag*, *Text*, $\}$
- $S :=$ *Tabelle*
- $P :=$

$$
\begin{aligned}
\{ \quad \textit{Tabelle} &\to \texttt{<table>}\ \textit{Zeilen}\ \texttt{</table>}\ , \\
\textit{Zeilen} &\to \textit{Zeile} \mid \textit{Zeile}\ \textit{Zeilen}\ , \\
\textit{Zeile} &\to \texttt{<tr>}\ \text{Einträge}\ \texttt{</tr>}\ , \\
\textit{Einträge} &\to \textit{Eintrag} \mid \textit{Eintrag}\ \textit{Einträge}\ , \\
\textit{Eintrag} &\to \texttt{<td>}\ \textit{Text}\ \texttt{</td>} \mid \texttt{<td>}\ \textit{Tabelle}\ \texttt{</td>}\ , \\
\textit{Text} &\to \mathtt{a} \mid \mathtt{b} \mid \cdots \mid \mathtt{z} \mid \mathtt{A} \mid \mathtt{B} \mid \cdots \mid \mathtt{Z} \mid \mathtt{0} \mid \cdots \mid \mathtt{9} \mid \mathtt{:} \mid \mathtt{-} \mid \mathtt{\sqcup} \\
\textit{Text} &\to
\end{aligned}
$$

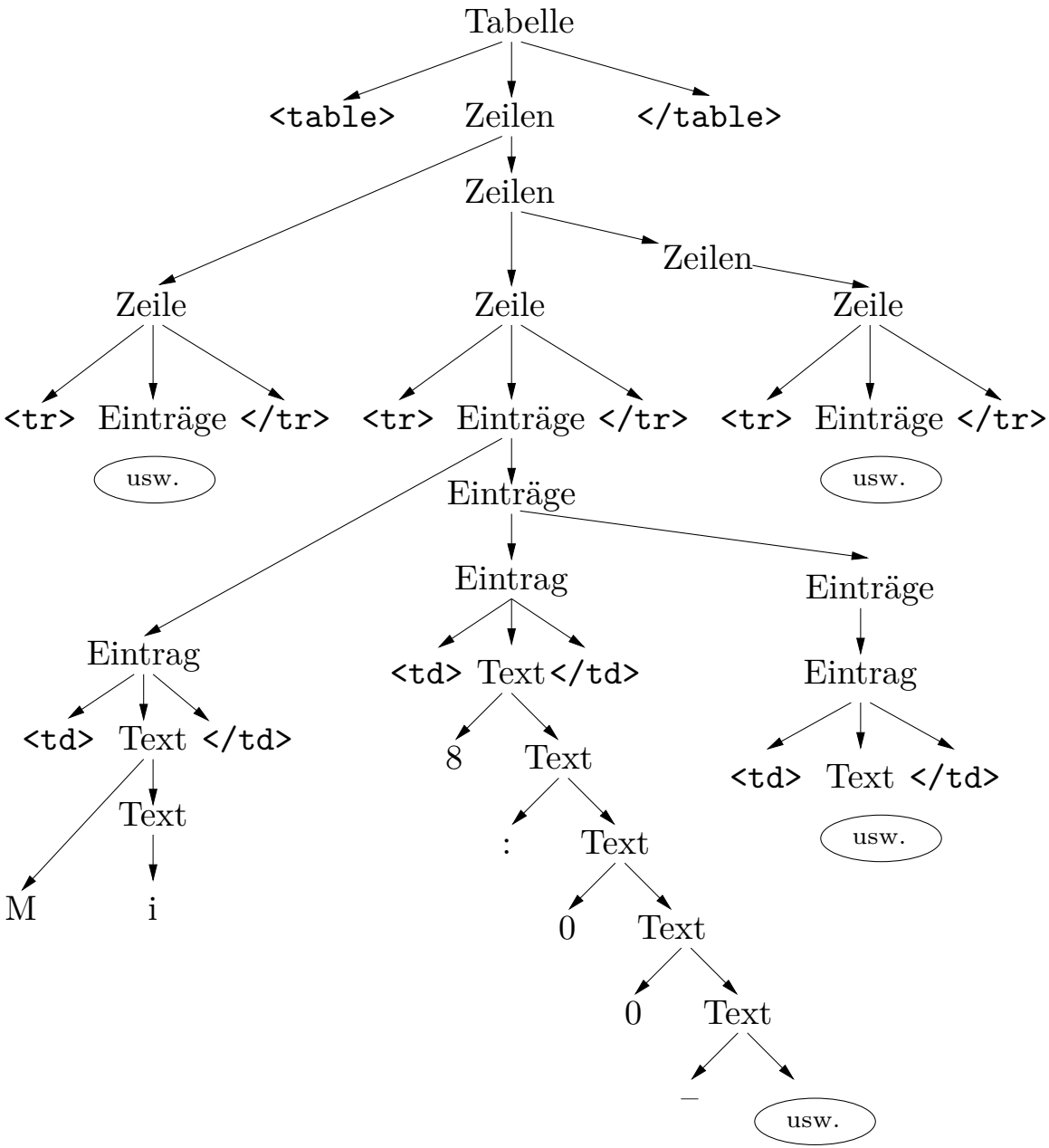
Eine KFG zur Erzeugung von HTML-Tabellen

Wir konstruieren eine Grammatik $G_{\mathrm{HTML}} = (\Sigma, V, S, P)$, so dass G_{HTML} genau die (möglicherweise geschachtelten) HTML-Tabellen erzeugt.

- $\Sigma := \{$ `<table>`, `</table>`, `<tr>`, `</tr>`, `<td>`, `</td>`, $\mathtt{a}, \ldots, \mathtt{z}$, $\mathtt{A}, \ldots, \mathtt{Z}$, $\mathtt{0}, \mathtt{1}, \ldots, \mathtt{9}$, `:`, `-`, $\mathtt{\sqcup}$ $\}$
- $V := \{$ *Tabelle*, *Zeilen*, *Zeile*, *Einträge*, *Eintrag*, *Text*, $\}$
- $S :=$ *Tabelle*
- $P :=$

$$
\begin{aligned}
\{ \quad \mathit{Tabelle} &\to \texttt{<table>}\ \mathit{Zeilen}\ \texttt{</table>}\ , \\
\mathit{Zeilen} &\to \mathit{Zeile} \mid \mathit{Zeile}\ \mathit{Zeilen}\ , \\
\mathit{Zeile} &\to \texttt{<tr>}\ \text{Einträge}\ \texttt{</tr>}\ , \\
\mathit{Einträge} &\to \mathit{Eintrag} \mid \mathit{Eintrag}\ \mathit{Einträge}\ , \\
\mathit{Eintrag} &\to \texttt{<td>}\ \mathit{Text}\ \texttt{</td>} \mid \texttt{<td>}\ \mathit{Tabelle}\ \texttt{</td>}\ , \\
\mathit{Text} &\to \mathtt{a} \mid \mathtt{b} \mid \cdots \mid \mathtt{z} \mid \mathtt{A} \mid \mathtt{B} \mid \cdots \mid \mathtt{Z} \mid \mathtt{0} \mid \cdots \mid \mathtt{9} \mid \mathtt{:} \mid \mathtt{-} \mid \mathtt{\sqcup} \\
\mathit{Text} &\to \mathtt{a}\,\mathit{Text} \mid \mathtt{b}\,\mathit{Text} \mid \cdots \mid \mathtt{z}\,\mathit{Text} \mid \mathtt{A}\,\mathit{Text} \mid \mathtt{B}\,\mathit{Text} \mid \cdots \mid \mathtt{Z}\,\mathit{Text} \\
\mathit{Text} &\to \mathtt{0}\,\mathit{Text} \mid \cdots \mathtt{9}\,\mathit{Text} \mid \mathtt{:}\,\mathit{Text} \mid \mathtt{-}\,\mathit{Text} \mid \mathtt{\sqcup}\,\mathit{Text} \qquad \}.
\end{aligned}
$$

Der Ableitungsbaum der HTML-Tabelle



Links- und Rechtsreguläre Grammatiken

Kann jede reguläre Sprache von einer konextfreien Grammatik erzeugt werden?

Sei $G = (\Sigma, V, S, P)$ eine kontextfreie Grammatik.

(a) G heißt **rechtsregulär**, wenn alle Produktionen die Form

$$A \to aB \text{ oder } A \to \varepsilon$$

für Variabeln $A, B \in V$ und ein Terminal $a \in \Sigma$ besitzen.

Links- und Rechtsreguläre Grammatiken

Kann jede reguläre Sprache von einer konextfreien Grammatik erzeugt werden?

Sei $G = (\Sigma, V, S, P)$ eine kontextfreie Grammatik.

(a) G heißt **rechtsregulär**, wenn alle Produktionen die Form

$$A \to aB \text{ oder } A \to \varepsilon$$

für Variabeln $A, B \in V$ und ein Terminal $a \in \Sigma$ besitzen.

(b) G heißt **linksregulär**, wenn alle Produktionen die Form

$$A \to Ba \text{ oder } A \to \varepsilon$$

für Variabeln $A, B \in V$ und ein Terminal $a \in \Sigma$ besitzen.

Reguläre und kontextfreie Sprachen

(a) Jede reguläre Sprache wird von einer rechtsregulären Grammatik erzeugt.

- Siehe **Tafel**.

Reguläre und kontextfreie Sprachen

(a) Jede reguläre Sprache wird von einer rechtsregulären Grammatik erzeugt.

- Siehe **Tafel**.

(b) Die Klasse der kontextfreien Sprachen ist eine echte Obermenge der regulären Sprachen, denn

$$L = \{ a^n b^n \ : \ n \in \mathbb{N} \}$$

- ist kontextfrei: Die kontextfreie Grammatik $G = (\{a, b\}, \{S\}, S, P)$ mit den Produktionen

Reguläre und kontextfreie Sprachen

(a) Jede reguläre Sprache wird von einer rechtsregulären Grammatik erzeugt.

- Siehe **Tafel**.

(b) Die Klasse der kontextfreien Sprachen ist eine echte Obermenge der regulären Sprachen, denn

$$L = \{ a^n b^n \ : \ n \in \mathbb{N} \}$$

- ist kontextfrei: Die kontextfreie Grammatik $G = (\{a, b\}, \{S\}, S, P)$ mit den Produktionen

  $$S \to aSb \mid \epsilon$$

  erzeugt L.
- aber nicht regulär.

Reguläre und kontextfreie Sprachen

(a) Jede reguläre Sprache wird von einer rechtsregulären Grammatik erzeugt.
- Siehe **Tafel**.

(b) Die Klasse der kontextfreien Sprachen ist eine echte Obermenge der regulären Sprachen, denn

$$L = \{ a^n b^n \ : \ n \in \mathbb{N} \}$$

- ist kontextfrei: Die kontextfreie Grammatik $G = (\{a, b\}, \{S\}, S, P)$ mit den Produktionen

$$S \to aSb \mid \epsilon$$

  erzeugt L.
- aber nicht regulär.

(c) Eine Sprache L ist genau dann regulär, wenn es links- oder rechtsreguläre Grammatik G gibt mit

$$L = L(G).$$

Details in der Veranstaltung „**Theoretische Informatik 2**“.

Zusammenfassung und Ausblick

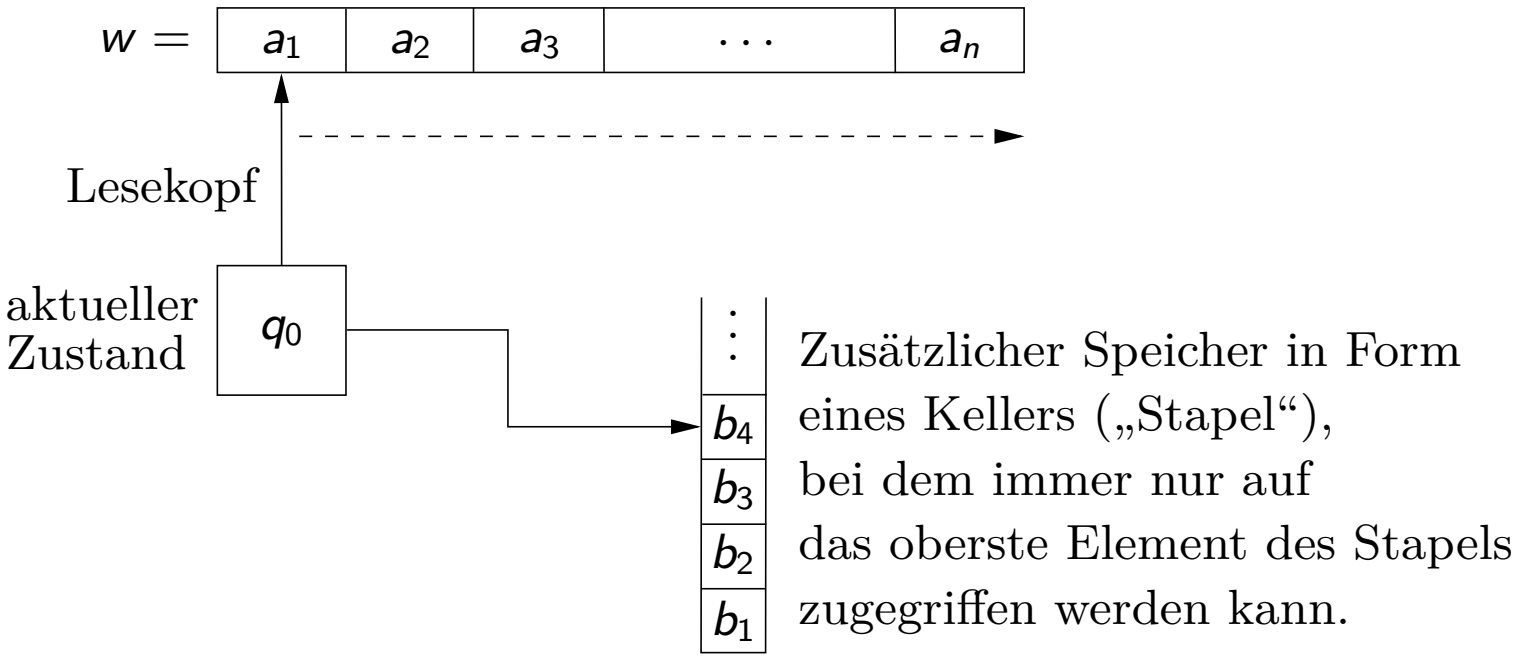
KFGs werden eingesetzt, um

rekursiv definierte Strukturen

zu modellieren.

Kellerautomaten

Schematische Darstellung der Verarbeitung eines Eingabeworts durch einen **Kellerautomaten**:



In der „Theoretischen Informatik 2“ wird gezeigt, dass kontextfreie Grammatiken und nichtdeterministische Kellerautomaten genau die Klasse der kontextfreien Sprachen erzeugen, bzw. akzeptieren.

Deterministisch kontextfreie Sprachen

Eine Sprache L heißt **deterministisch kontextfrei**, wenn L von einem deterministischen Kellerautomaten akzeptiert wird.

- Die Sprache

$$L_1 = \{ a^n b^n \ : \ n \in \mathbb{N} \}$$

  ist

Deterministisch kontextfreie Sprachen

Eine Sprache L heißt **deterministisch kontextfrei**, wenn L von einem deterministischen Kellerautomaten akzeptiert wird.

- Die Sprache

$$L_1 = \{ a^n b^n \ : \ n \in \mathbb{N} \}$$

  ist deterministisch kontextfrei,

- die Sprache

$$L_2 = \{ a^n b^n c^m \ : \ n, m \in \mathbb{N} \} \cup \{ a^m b^n c^n \ : \ n, m \in \mathbb{N} \}$$

  ist

Deterministisch kontextfreie Sprachen

Eine Sprache L heißt **deterministisch kontextfrei**, wenn L von einem deterministischen Kellerautomaten akzeptiert wird.

- Die Sprache

$$L_1 = \{ a^n b^n \ : \ n \in \mathbb{N} \}$$

  ist deterministisch kontextfrei,

- die Sprache

$$L_2 = \{ a^n b^n c^m \ : \ n, m \in \mathbb{N} \} \cup \{ a^m b^n c^n \ : \ n, m \in \mathbb{N} \}$$

  ist nicht deterministisch kontextfrei.

Details in der Veranstaltung „**Theoretische Informatik 2**“.

Das Wort-Problem

Das Wortproblem:

$$\text{„Erzeugt eine KFG } G = (\Sigma, V, S, P) \text{ das Wort } w?\text{“}$$

sprich: „Ist das Programm w syntaktisch korrekt?“

Ein Parser muss das Wortproblem für jedes Eingabeprogramm w lösen, deshalb kommt dem Wortproblem eine große Bedeutung zu.

Das Wort-Problem

Das Wortproblem:

$$\text{„Erzeugt eine KFG } G = (\Sigma, V, S, P) \text{ das Wort } w?\text{“}$$

sprich: „Ist das Programm w syntaktisch korrekt?“

Ein Parser muss das Wortproblem für jedes Eingabeprogramm w lösen, deshalb kommt dem Wortproblem eine große Bedeutung zu.

(a) Der CYK-Algorithmus löst das Wortproblem für eine Grammatik $G = (\Sigma, V, S, P)$ in Zeit proportional zu $|w|^3 \cdot |P|$.

(b) **Kubische** Laufzeit ist völlig inakzeptabel, das Wortproblem für deterministisch kontextfreie Sprachen ist hingegen in **Linearzeit** lösbar.

- Die Syntax vieler Programmiersprachen wird deshalb von deterministisch kontextfreien Grammatiken definiert.

Details in der Veranstaltung „**Theoretische Informatik 2**“.

Eindeutige Grammatiken

Viele Parser übersetzen ein Anwenderprogramm in einen Ableitungsbaum, um damit die Semantik des Programms auszudrücken.

Eindeutige Grammatiken

Viele Parser übersetzen ein Anwenderprogramm in einen Ableitungsbaum, um damit die Semantik des Programms auszudrücken.

(a) Nenne eine kontextfreie Grammatik

eindeutig,

wenn jedes erzeugbare Wort genau einen Ableitungsbaum besitzt.

(b) Ein deterministischer Kellerautomat „besitzt“ eine äquivalente eindeutige Grammatik.

Details in der Veranstaltung „**Theoretische Informatik 2**“.

Eindeutige Grammatiken

Viele Parser übersetzen ein Anwenderprogramm in einen Ableitungsbaum, um damit die Semantik des Programms auszudrücken.

(a) Nenne eine kontextfreie Grammatik

eindeutig,

wenn jedes erzeugbare Wort genau einen Ableitungsbaum besitzt.

(b) Ein deterministischer Kellerautomat „besitzt“ eine äquivalente eindeutige Grammatik.

Details in der Veranstaltung „**Theoretische Informatik 2**“.

Die Syntax von Programmiersprachen und deterministisch kontextfreie Sprachen:

„A marriage made in heaven“.

Eine nicht-kontextfreie Sprache

(a) Die Sprache

$$L = \{a^n b^n c^n \ : \ n \in \mathbb{N}\}$$

ist **nicht** kontextfrei,

(b) aber ihre Komplement-Sprache

$$\overline{L} = \{\, w \in \{a, b, c\}^* \ : \ w \notin L \,\}$$

ist kontextfrei!

Eine nicht-kontextfreie Sprache

(a) Die Sprache

$$L = \{a^n b^n c^n \ : \ n \in \mathbb{N}\}$$

ist **nicht** kontextfrei,

(b) aber ihre Komplement-Sprache

$$\overline{L} = \{\, w \in \{a, b, c\}^* \ : \ w \notin L \,\}$$

ist kontextfrei!

Details in der Veranstaltung „**Theoretische Informatik 2**“.

Kontextfreie Sprachen sind nicht unter Komplementbildung abgeschlossen!