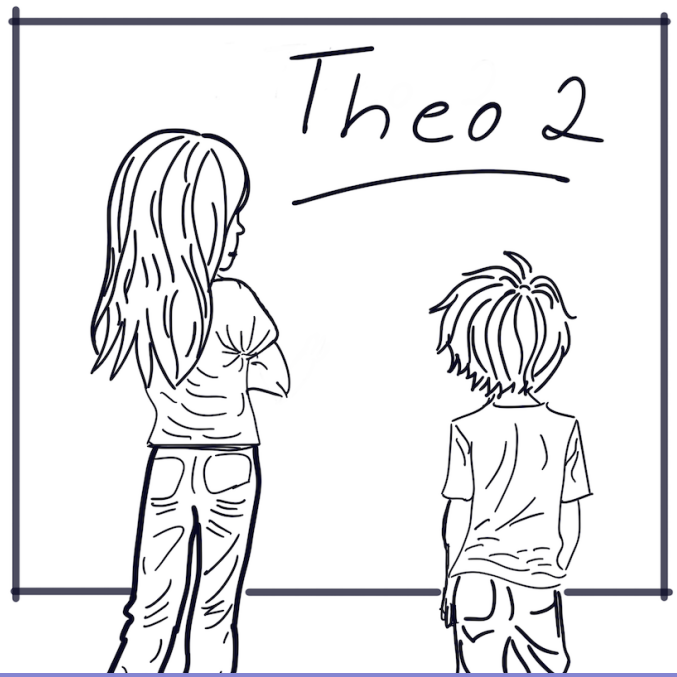
Theoretische Informatik 2

SoSe 2024

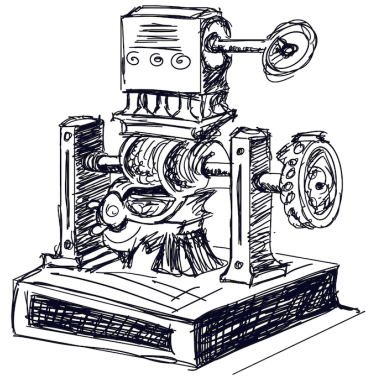
Berechenbarkeit und Komplexität

Prof. Dr. Sebastian Siebertz
AG Theoretische Informatik
MZH, Raum 3160
siebertz@uni-bremen.de

Universität Bremen



Was haben wir erreicht?



- Zentrale Frage: Was kann ein Computer berechnen?
- Sind für verschiedene Programmiersprachen oder für verschiedene Rechnerarchitekturen verschiedene Funktionen berechenbar?
- Wir brauchen formale Modelle der Berechenbarkeit!
- Dann können wir untersuchen
  - welche Funktionen in welchem Modell berechenbar sind,
  - ob ein Modell mächtiger ist als ein anderes, oder
  - ob es einen universellen Begriff der Berechenbarkeit gibt.

Was haben wir erreicht?

- Die Essenz einer Berechnung:
  - eine Eingabe liegt in einem Speicher;
  - es wird eine feste endliche Folge von Befehlen aus einem festen endlichen Befehlssatz ausgeführt;
  - jeder Befehl manipuliert den Speicher nach festen Regeln;
  - das Ergebnis der Berechnung wird aus dem Speicher abgelesen.
- Wir haben Turingmaschinen eingeführt:
  - Unglaublich einfaches Modell für die endliche Kontrolle eines Speichers.
  - Intuitiv ist klar, dass jedes endliche, deterministische System zur Speichermanipulation simuliert werden kann!

Was haben wir erreicht?

- Varianten von Turingmaschinen, die alle äquivalent sind.
  - Statt q_a und q_r Menge von akzeptierenden Zuständen F.
  - Speicherband in beide Richtungen unendlich.
  - Mehrere Spuren zur Berechnung: Mehrspur-Turingmaschinen.
  - Mehrere Speicherbänder mit unabhängigen Lese- und Schreibköpfen: Mehrband-Turingmaschinen.
  - Nichtdeterministische Maschinen.

Was bleibt offen?

- Kann wirklich jede Rechnerarchitektur und jede Programmiersprache simuliert werden?

Church-Turing These

Die Klasse der Turing-berechenbaren Funktionen stimmt mit der Klasse der intuitiv berechenbaren Funktionen überein.

- Verstehen wir wirklich, was berechenbar ist?
- Was ist semi-entscheidbar und was ist entscheidbar?
  - Gibt es Probleme, die semi-entscheidbar, aber nicht entscheidbar sind?
  - Gibt es Probleme, die nicht semi-entscheidbar sind?

Vorschau

- Weitere äquivalente Formalismen, die uns mehr Intuition geben, was berechenbar ist und die die Church-Turing These stützen.
  - Grammatiken
  - WHILE-Programme als Abstraktion imperativer Programmiersprachen
- Grenzen des Berechenbaren, unentscheidbare Probleme.

Grammatiken

- Parallel zu den Entwicklungen in

  der Berechenbarkeitstheorie:

  - Formalisierung der Grammatik und Sprache.
- Hierarchie von Sprachklassen auf Basis von Grammatiken verschiedener Komplexität.
  - Chomsky-Hierarchie.
  - Entwickelt von Noam Chomsky.
- Zu jeder Sprachklasse korrespondiert ein Automaten/Maschinenmodell.
  - Grammatiken bieten eine intuitive alternative Sicht auf Berechnungen.
  - Grammatiken motivieren Nichtdeterminismus.

Grammatiken intuitiv

- Grammatiken beschreiben oder definieren formale Sprachen:
  - Wir beginnen mit einem Startsymbol.
  - Regeln erlauben es, wiederholt (Teil-)Wörter durch andere Wörter zu ersetzen.

Beispiel

- Startsymbol: S
- Regeln:
  - $S \longrightarrow aSb$
  - $S \longrightarrow \varepsilon$
- Eine mögliche Ableitung eines Wortes ist:

$$S \underset{1}{\vdash} aSb \underset{1}{\vdash} aaSbb \underset{1}{\vdash} aaaSbbb \underset{2}{\vdash} aaabbb$$

- Symbol S: nichtterminales Symbol
- Wir sind nur an erzeugten Wörtern interessiert, die keine nichtterminalen Symbole enthalten: Terminalwörter.
- Die erzeugte Sprache: $\{a^n b^n : n \geq 0\}$

Grammatiken

Definition Grammatik

Eine Grammatik ist von der Form $G = (N, \Sigma, P, S)$, wobei

- N und Σ endliche, disjunkte Alphabete von Nichtterminalsymbolen bzw. Terminalsymbolen sind,
- $S \in N$ das Startsymbol ist,
- $P \subseteq (N \cup \Sigma)^+ \times (N \cup \Sigma)^*$ eine endliche Menge von Regeln ist,
  - ▷ wobei für jedes $(u, v) \in P$ gilt, dass u mindestens ein Nichtterminalsymbol enthält.

Grammatiken Notation

- Wir schreiben Regeln $(u, v) \in P$ gewöhnlich als $u \longrightarrow v$ und mehrere Regeln $(u, v_1), (u, v_2), \ldots, (u, v_k)$ mit der gleichen linken Seite meist als $u \longrightarrow v_1 \mid v_2 \mid \ldots \mid v_k$.
- Wir schreiben Elemente von N meistens mit Großbuchstaben und Elemente von Σ meistens mit Kleinbuchstaben.

Ableitbarkeit, erzeugte Sprache

Definition Ableitbarkeit, erzeugte Sprache

Sei $G = (N, \Sigma, P, S)$ eine Grammatik und v, w Wörter aus $(N \cup \Sigma)^*$.

1) w aus v direkt ableitbar:
   $v \vdash_G w \Leftrightarrow v = v_1 x v_2$ und $x \longrightarrow y \in P$ und $w = v_1 y v_2$ mit $v_1, v_2 \in (N \cup \Sigma)^*$
2) w aus v in n Schritten ableitbar:
   $v \vdash_G^n w \Leftrightarrow v \vdash_G v_1 \vdash_G \cdots \vdash_G v_{n-1} \vdash_G w$ für $v_1, \ldots, v_{n-1} \in (N \cup \Sigma)^*$
3) w aus v ableitbar:
   $v \vdash_G^* w \Leftrightarrow v \vdash_G^n w$ für ein $n \geq 0$
4) Die durch G erzeugte Sprache ist
   $L(G) := \{w \in \Sigma^* \mid S \vdash_G^* w\}$.

- Wir schreiben $\vdash$ statt $\vdash_G$ wenn klar ist in welcher Grammatik wir ableiten. Stattdessen kann die angewendete Regel als Subskript geschrieben werden.

Beispiel

- Ableitungen in der Grammatik $G = (N, \Sigma, P, S)$ mit
  - $N = \{S, B\}$
  - $\Sigma = \{a, b, c\}$
  - $P = \{S \longrightarrow aSBc \mid abc,$
    $cB \longrightarrow Bc,$
    $bB \longrightarrow bb\}$:
    - $S \underset{1.2}{\vdash} abc$
    - $S \underset{1.1}{\vdash} aSBc \underset{1.1}{\vdash} aaSBcBc \underset{1.2}{\vdash} aaabcBcBc$
      $\underset{2}{\vdash} aaabBccBc \underset{2}{\vdash}^{2} aaabBBccc \underset{3}{\vdash}^{2} aaabbbccc$
- $L(G) = \{a^n b^n c^n \mid n \geq 1\}$.

Die Chomsky-Hierarchy

Definition Chomsky-Hierarchy

Es sei $G = (N, \Sigma, P, S)$ eine Grammatik.

- G ist Grammatik vom Typ 3 (rechtslinear), falls alle Regeln die Form $A \longrightarrow uB$ oder $A \longrightarrow u$ haben mit $A, B \in N$, $u \in \Sigma^*$.
- G ist Grammatik vom Typ 2 (kontextfrei), falls alle Regeln die Form $A \longrightarrow w$ haben mit $A \in N, w \in (\Sigma \cup N)^*$.
- G ist Grammatik vom Typ 1 (nicht verkürzend, kontextsensitiv), falls alle Regeln nicht verkürzend sind, also die Form $w \longrightarrow u$ haben wobei $w, u \in (\Sigma \cup N)^+$ und $|u| \geq |w|$.

  Ausnahme: Die Regel $S \longrightarrow \varepsilon$ ist erlaubt, wenn S in keiner Regel auf der rechten Seite vorkommt.
- Jede Grammatik G ist Grammatik vom Typ 0.

Kontextfrei versus kontextsensitiv

- Typ 2 heißt kontextfrei, weil in kontextfreien Regeln

$$A \longrightarrow w$$

  die linke Seite nur aus Nichterminalsymbol A besteht.
- Ersetzung daher unabhängig vom Kontext im Wort.
- Grammatiken der Typen 0 und 1 erlauben Regeln

$$u_1 A u_2 \longrightarrow u_1 w u_2.$$

- Ersetzung von A durch w abhängig vom Kontext u_1 links und u_2 rechts (und noch allgemeinere Ersetzungen).

Sprachklassen der Chomsky-Hierarchie

Definition Sprachklassen

Für $i = 0, 1, 2, 3$ ist die Klasse der Typ-i-Sprachen definiert als

$$\mathcal{L}_i := \{L(G) \mid G \text{ ist Grammatik vom Typ } i\}.$$

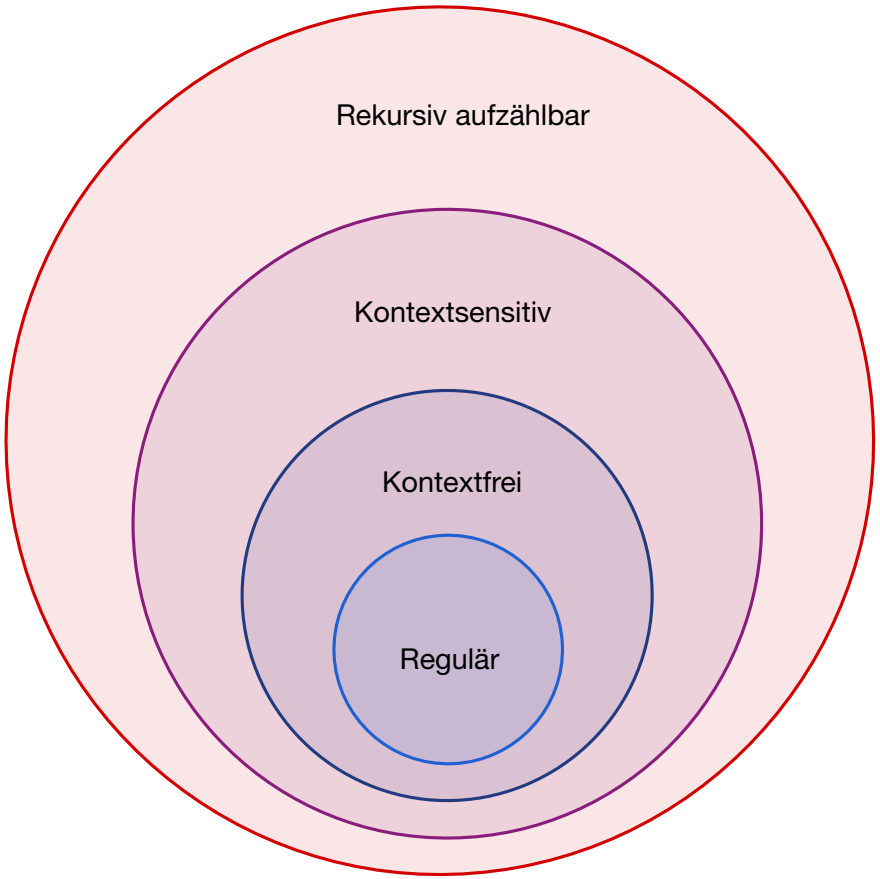
- Jede Typ 3 Grammatik ist auch eine Typ 2 Grammatik.
- Jede Typ 1 Grammatik ist auch eine Typ 0 Grammatik.
- Typ 2 und Typ 3 Grammatiken erlauben das Verkürzen des abgeleiteten Wortes, sie sind also nicht immer Typ 1 Grammatiken!
- Aber jede Typ-2-Grammatik kann in äquivalente Typ-1 Grammatik umgewandelt werden.

Sprachklassen der Chomsky-Hierarchie

- Also bilden die Sprachklassen eine Hierarchie.
- Diese ist sogar strikt.

Satz

$\mathcal{L}_3 \subsetneq \mathcal{L}_2 \subsetneq \mathcal{L}_1 \subsetneq \mathcal{L}_0$



Automaten/Maschinenmodelle

- Jeder Grammatiktyp korrespondiert zu einem Maschinen/Automatenmodell.
  - Typ 3: endliche Automaten
  - Typ 2: Kellerautomaten
  - Typ 1: linear beschränkte NTM
  - Typ 0: Turingmaschinen

Automaten/Maschinenmodelle

- Grammatiken erklären die Nutzung des Speichers!
  - Endlicher Automat: Turingmaschine ohne Speicherzugriff
  - Kellerautomat: Turingmaschine mit Kellerspeicher (Stack, Zugriff nur auf das letzte/oberste Element des Speichers)
  - Linear beschränkte NTM: Nichtdeterministische Turingmaschine mit linearer Speicherbeschränkung (linear in der Eingabegröße)
  - Unbeschränkte Turingmaschine

Grammatiken vs Turingmaschinen

Satz

Sei L eine Sprache. Dann ist L semi-entscheidbar genau dann, wenn $L \in \mathcal{L}_0$.

Beweis.

- Sei $M = (Q, \Sigma, \Gamma, [, \sqcup, \delta, q_s, q_a, q_r)$ eine deterministische 1-Band Turingmaschine mit $L(M) = L$.
- Annahme: M akzeptiert ein Eingabewort $w \in L(M)$ nur in der Konfiguration $([w\sqcup\sqcup\ldots, 1)$, d.h.
  - Das Band ist nur mit $[w\sqcup\sqcup\ldots$ beschriftet.
  - Der Kopf befindet sich an Position 1.

Technische Annahme

- Warum dürfen wir das annehmen?
- Falls M diese Eigenschaft nicht hat, betrachten wir M':
  - M' schreibt zuerst ein neues Trennzeichen $[']$ und eine Kopie des Eingabewortes w hinter das Eingabewort w.
  - M' markiert mit einem weiteren neuen Symbol $]$ die letzte Speicherstelle, die sie bisher für ihre Berechnung genutzt hat.
    - M' stellt zu Beginn also die Bandinschrift $[\,w[\,'w\,]\sqcup\sqcup\ldots$ her.
  - Dann arbeitet M' wie M auf dem hinteren, abgetrennten Teil des Bandes und schiebt das Zeichen $]$ immer ein Stück weiter, wenn sie weiteren Speicher belegt.
  - Bevor M' in den akzeptierenden Zustand wechselt, löscht sie den Teil des Bandes, der durch $[']$ und $]$ eingerahmt ist und läuft mit dem Kopf auf das Zeichen $[$.
    - Bandinhalt $[\,w\sqcup\sqcup\ldots$ wieder hergestellt.

Konstruktion der Grammatik

- Konstruiere Grammatik G, deren Ableitungen den Berechnungen von M entsprechen.
- Sei

$$\alpha = (q, v, k) \in Q \times \Gamma^{\mathbb{N}} \times \mathbb{N}$$

eine Konfiguration und sei $n \geq k$ minimal mit $v(i) = \sqcup$ für alle $i > n$.

- Für $\ell \geq n$ definiere

$$\kappa(\alpha, \ell) := v_1 \dots v_{k-1} q v_k \dots v_n \sqcup \dots \sqcup \in (Q \cup \Gamma)^{\ell+1}.$$

  - $\kappa(\alpha, \ell)$ ist also das Wort, das aus der Bandinschrift v entsteht, indem wir das Band nach der Stelle ℓ abschneiden
  - und den Zustand q (als Zeichen) vor das Zeichen v_k schreiben, wobei k die Kopfposition ist (das Wort mit q eingefügt hat Länge $\ell + 1$).

Konstruktion der Grammatik: Intuition

- Angenommen $w \in L(M)$.
- Dann existiert die akzeptierende Berechnung

$$\alpha_1 \vdash_M \alpha_2 \vdash_M \dots$$

  wobei α_1 die Startkonfiguration von M auf w ist und α_i die akzeptierende Konfiguration $(q_a, [\, w \sqcup \sqcup \dots, 1)$ für ein $i \in \mathbb{N}$ ist.
- Sei ℓ maximal, so dass der Kopf die Position ℓ während der Berechnung besucht.
- Die Grammatik soll zunächst das Wort $\kappa(\alpha_1, \ell)$ erzeugen.
- Dann soll sie $\kappa(\alpha_1, \ell) \vdash_G \kappa(\alpha_2, \ell) \vdash_G \dots \vdash_G \kappa(\alpha_i, \ell)$ ableiten.
- Dann soll sie alle $\sqcup$-Symbole, den Zustand q_a und das Zeichen $[$ aus $\kappa(\alpha_i, \ell)$ löschen um das Terminalwort w zu erhalten.

Konstruktion der Grammatik

- Formal definieren wir $G = (N, \Sigma, P, S)$ mit $N = Q \cup \Gamma \setminus \Sigma \cup \{S, W, B, A\}$ und den folgenden Regeln.
- Regeln für die Ableitung von $\kappa(\alpha_1, \ell)$ für die Startkonfiguration von M auf w für beliebiges $w \in \Sigma^*$ und beliebiges $\ell \in \mathbb{N}$.
  - $S \longrightarrow q_s \llbracket W$,
  - $W \longrightarrow aW$ für $a \in \Sigma$,
  - $W \longrightarrow B$,
  - $B \longrightarrow \sqcup B$,
  - $B \longrightarrow \varepsilon$.

Konstruktion der Grammatik

- Regeln zur Simulation der Berechnung.
  - Für alle $q, q' \in Q, a, b, c \in \Gamma$ mit $\delta(q, a) = (q', b, -1)$ die Regel

$$cqa \longrightarrow q'cb.$$

  - Für alle $q, q' \in Q, a, b \in \Gamma$ mit $\delta(q, a) = (q', b, 0)$ die Regel

$$qa \longrightarrow q'b.$$

  - Für alle $q, q' \in Q, a, b \in \Gamma$ mit $\delta(q, a) = (q', b, +1)$ die Regel

$$qa \longrightarrow bq'.$$

- $\alpha_i \vdash_M \alpha_{i+1} \Leftrightarrow \kappa(\alpha_i, \ell) \vdash_G \kappa(\alpha_{i+1}, \ell)$.

Konstruktion der Grammatik

- Letzte Regeln: Aufräumphase
- Wenn M das Wort w akzeptiert, so terminiert sie in Konfiguration $(q_a, [\, w \sqcup \sqcup \ldots, 1)$
- Hieraus muss G noch w ableiten.
  - $q_a [\longrightarrow A$,
  - $Aa \longrightarrow aA$ für $a \in \Sigma$,
  - $A \sqcup \longrightarrow A$,
  - $A \longrightarrow \varepsilon$.

Beweis der Korrektheit

- Wir zeigen nun, dass $L(M) = L(G)$.
- Sei $w \in L(M)$. Dann führt die Berechnung

$$\alpha_1 \vdash_M \alpha_2 \vdash_M \dots,$$

wobei $\alpha_1 = (q_s, [w\sqcup\sqcup\dots, 1)$ die Startkonfiguration von M auf w ist, in die akzeptierende Konfiguration $\alpha_i = (q_a, [w\sqcup\sqcup\dots, 1)$ für ein $i \in \mathbb{N}$.

- Sei ℓ maximal, so dass der Kopf die Position ℓ während der Berechnung besucht.

Beweis der Korrektheit

- Wir leiten zunächst das Wort $\kappa(\alpha_1, \ell)$ in G ab:

$$
\begin{aligned}
S &\vdash_G q_s[\,W \\
&\vdash_G q_s[\,w_1\,W \\
&\vdash_G \dots \\
&\vdash_G q_s[\,w_1 \dots w_{|w|}\,W \\
&\vdash_G q_s[\,w_1 \dots w_{|w|}\,B \\
&\vdash_G q_s[\,w_1 \dots w_{|w|}\,\sqcup\,B \\
&\vdash_G \dots \\
&\vdash_G q_s[\,w_1 \dots w_{|w|}\,\sqcup \dots \sqcup\,B \\
&\vdash_G q_s[\,w_1 \dots w_{|w|}\,\sqcup \dots \sqcup \quad = \kappa(\alpha_1, \ell).
\end{aligned}
$$

Beweis der Korrektheit

- Weiter geht's: da ℓ maximal ist, so dass der Kopf die Position während der Berechnung besucht, haben wir für $1 \leq j < i$ eine Regel, die die Ableitung $\kappa(\alpha_j, \ell) \vdash_G \kappa(\alpha_{j+1}, \ell)$ erlaubt.
- Also können wir weiter ableiten:

$$\kappa(\alpha_1, \ell) \vdash_G \kappa(\alpha_2, \ell) \vdash_G \ldots \vdash_G \kappa(\alpha_i, \ell).$$

- Es gilt $\kappa(\alpha_i, \ell) = q_a [w \sqcup \ldots \sqcup$.
- Wir wenden nacheinander die Regeln $q_a [\longrightarrow A$, $A w_i \longrightarrow w_i A$ für $i = 1, \ldots, |w|$ und $A \sqcup \longrightarrow A$ an, bis keine $\sqcup$ Symbole mehr vorhanden sind.
- Dann leiten wir mit $A \longrightarrow \varepsilon$ das Wort w ab.
- Also $w \in L(G)$.

Beweis der Korrektheit

- Sei umgekehrt $S \vdash_G x_1 \vdash_G \ldots \vdash_G x_s = w$ für $x_i \in (N \cup \Sigma)^*$ eine Ableitung eines Wortes w in G.
- Wir beobachten zunächst, dass wir annehmen dürfen, dass $m, n \in \mathbb{N}$ und für $1 \leq i \leq m$ Symbole $v_i \in \Sigma$ existieren, so dass gilt
  - $x_1 = q_s \llbracket W$,
  - $x_{i+1} = q_s \llbracket v_1 \ldots v_i W$ für $1 \leq i \leq m$,
  - $x_{m+2} = q_s \llbracket v_1 \ldots v_m B$,
  - $x_{m+2+j} = q_s \llbracket v_1 \ldots v_m \llcorner\!\!\lrcorner^j B$ für $1 \leq j \leq m$ und
  - $x_{m+2+n} = q_s \llbracket v_1 \ldots v_m \llcorner\!\!\lrcorner^n$.

Beweis der Korrektheit

- Warum dürfen wir das annehmen?
  - $S \vdash_M q_s \llbracket W$ ist die einzige Regel, die auf das Startsymbol S anwendbar ist.
  - Wenn die Nichtterminalsymbole W und B in x_i vorkommen, so stehen sie ganz rechts in x_i, B kann nur aus W abgeleitet werden und wenn W (bzw. B) in x_i nicht vorkommt, so kommt es in keinem x_j für $j > i$ vor.
  - Ein Nichtterminalsymbol $q \in Q$ kann nur ersetzt werden, wenn das Symbol nach q ein Symbol aus Γ ist, insbesondere niemals, wenn das Symbol nach q das Symbol W oder B ist.

    Die Ableitung von Symbolen $q \in Q$ setzt also voraus, dass die Ableitung von W bzw. B hinreichend weit fortgeschritten ist, kann diese aber niemals "überholen". Ansonsten sind die Ableitungen der Symbole unabhängig voneinander und können in der Reihenfolge vertauscht werden.

Beweis Korrektheit

- Sei $v = v_1 \dots v_m$ und $\ell = m + n$.
- Dann ist nach unserer Annahme $x_{m+n+2} = \kappa(\alpha_1, \ell)$, wobei α_1 die Startkonfiguration von M auf v ist.
- Sei p maximal, so dass $x_{m+n+1+p}$ nicht das Nichtterminal A enthält.
- Dann gilt $x_{m+n+1+j} = \kappa(\alpha_j, \ell)$ für $1 \leq j \leq p$.
- Also können wir mit M die Berechnung $\alpha_1 \vdash_M \alpha_2 \vdash_M \dots \vdash_M \alpha_p$ ausführen.

Beweis Korrektheit

- Da am Ende w durch die Ableitung erzeugt wird, muss die Regel $q_a[\longrightarrow A$ angewendet werden.
- Es folgt, dass $\alpha_p = \alpha_i$ die akzeptierende Konfiguration $(q_a, [w\sqcup\sqcup\ldots, 1)$ ist.
- Also gilt $v = w$ und wir haben eine akzeptierende Berechnung von M auf w gefunden.
- Also $w \in L(M)$. □

Die andere Richtung

- $L \in \mathcal{L}_0$ und sei $G = (N, \Sigma, P, S)$ eine Grammatik mit $L(G) = L$.
- Konstruiere nichtdeterministische 2-Band Turingmaschine M mit $L(M) = L(G)$.
- M speichert auf dem ersten Band die Eingabe $w \in \Sigma^*$ und simuliert auf dem zweiten Band nichtdeterministisch eine Ableitung von G.
- Spezieller Zustand q_x, in den sie wechselt nachdem sie einen Ableitungsschritt simuliert hat.

Die andere Richtung

- Initialisierung: M schreibt S auf das zweite Band und wechselt in den Zustand q_x.
- Invariante: Wenn M im Zustand q_x ist, so steht auf dem zweiten Band ein Wort $x_i \in (N \cup \Sigma)^*$ eingerahmt von $\llcorner\lrcorner$, so dass eine Ableitung $S \vdash_G^* x_i$ existiert.

Die andere Richtung

- Wenn M in q_x ist, so führt sie danach folgende Berechnungsfolge aus:
  - Laufe über das zweite Band und wähle nichtdeterministisch eine Position j mit $0 \leq j \leq |x_i|$ aus.
  - Wähle nichtdeterministisch eine Regel $u \longrightarrow v$ aus P aus.
  - Verifiziere, dass $x_j \ldots x_{j+|u|} = u$ und ersetze u durch v (verschiebe das Restwort $x_{j+|u|+1} \ldots$ falls nötig).
    - Das erhaltene x_{i+1} erfüllt $x_i \vdash_G x_{i+1}$
  - Prüfe, ob das Wort v auf Band 2 ein Wort aus Σ^* ist.
    - Nein $\rightarrow$ wechsele wieder in q_x und fahre mit Berechnung fort.
    - Ja $\rightarrow$ überprüfe ob $v = w$ und wechsele entsprechend in akzeptierenden Zustand oder verwerfenden Zustand.

Beweis Korrektheit

- Für $w \in \Sigma^*$ gilt

  $w \in L(M)$

  $\Leftrightarrow$ es existiert eine akzeptierende Berechnung $\alpha_1 \vdash_M \ldots \vdash_M \alpha_i$ von M auf w

  $\Leftrightarrow$ es existiert eine akzeptierende Berechnung $\alpha_1 \vdash_M \ldots \vdash_M \alpha_i$ in der auf Band 2 das Wort w geschrieben wird

  $\Leftrightarrow$ es gilt $S \vdash_G^* w$

  $\Leftrightarrow$ es gilt $w \in L(G)$.

Zusammenfassung

Satz

Sei L eine Sprache. Dann ist L semi-entscheidbar genau dann, wenn $L \in \mathcal{L}_0$.

- Berechnungen von Turingmaschinen und Ableitungen in Grammatiken sind sehr ähnlich!
- Der Nichtdeterminismus von Grammatiken "Ersetze ein Teilwort durch dieses oder jenes Teilwort" ist vielleicht intuitiver als der Nichtdeterminismus von Turingmaschinen.

Entscheidbare Sprachen

Satz

Eine Sprache $L \subseteq \Sigma^*$ ist entscheidbar genau dann, wenn L und $\bar{L}$ semi-entscheidbar sind.

Beweis.

- Sei L entscheidbar.
- Dann existiert eine deterministische Turingmaschine M, die auf jedem $w \in \Sigma^*$ terminiert und w akzeptiert genau dann, wenn $w \in L$.
- Sei M' die Maschine, die wir aus M erhalten indem wir q_a und q_r vertauschen.
- Dann gilt $L(M') = \bar{L}$. Also sind sowohl L als auch $\bar{L}$ entscheidbar, insbesondere also semi-entscheidbar.

Beweis fortgesetzt

- Seien umgekehrt L und $\bar{L}$ semi-entscheidbar und seien M und M' deterministische Turingmaschinen mit $L(M) = L$ und $L(M') = \bar{L}$.
- Konstruiere 2-Band Turingmaschine M'', die für Eingabe $w \in \Sigma^*$ auf dem ersten Band die Berechnung von M auf w und auf dem zweiten Band die Berechnung von M' auf w simuliert.
- Falls die Berechnung auf dem ersten Band in einem akzeptierenden Zustand terminiert, so akzeptiert M'', falls die Berechnung auf dem zweiten Band in einem akzeptierenden Zustand terminiert, so verwirft M''.

Beweis fortgesetzt

Es gilt

$w \in L$
$\Rightarrow$ Berechnung von M auf w terminiert in akzeptierenden Zustand
$\Rightarrow$ Berechnung von M'' auf w auf dem ersten Band terminiert in akzeptierenden Zustand
$\Rightarrow$ M'' akzeptiert w

und

$w \notin L$
$\Rightarrow$ Berechnung von M' auf w terminiert in akzeptierenden Zustand
$\Rightarrow$ Berechnung von M'' auf w auf dem zweiten Band terminiert in akzeptierenden Zustand
$\Rightarrow$ M'' verwirft w.

Also gilt $w \in L \Leftrightarrow w \in L(M'')$.

Korollar

Korollar

Eine Sprache $L \subseteq \Sigma^*$ ist entscheidbar genau dann, wenn es eine Grammatik G und eine Grammatik $\bar{G}$ gibt mit $L(G) = L$ und $L(\bar{G}) = \bar{L}$.