

FGI 1

Automaten, Formale Sprachen, Berechenbarkeit

Christopher Habel & Matthias Jantzen

Kap 19

(Berechenbarkeit, Turingmaschinen und kontextsensitive Grammatiken)

Matthias Jantzen

M. Jantzen, FGI-1, SoSe 2009: 1

Donnerstag, 25. Juni 2009

1

Wortfunktion versus ganzzahlige Berechnungen

Zur Erinnerung: (Turing-)berechenbare Wortfunktionen:

Definition 50:

Seien Σ und Λ Alphabete.

Eine partielle (Wort-)Funktion $f : \Sigma^* \rightarrow \Lambda^*$ heißt **Turing-berechenbar** oder **partiell rekursiv** genau dann, wenn es eine *DTM* $T = (Q, \Sigma, \Gamma, \delta, q_0, \#, F)$ gibt mit:

$$q_0 w \vdash_T^* q_e v, \quad \text{für } q_e \in F \text{ und } v \in \Lambda^* \text{ mit } f(w) = v.$$

...der Funktionswert steht *ab der Kopfposition* auf dem Band (abgesehen von nachfolgenden $\#$'s)!

Wenn die *DTM* T die Eingabe verwirft, oder nie anhält, so soll $f(w) := \perp$ die Undefiniertheit von f anzeigen.

M. Jantzen, FGI-1, SoSe 2009: 2

Donnerstag, 25. Juni 2009

2

TM-berechenbare Funktionen

Wie definieren Vossen/Witt (Turing-)berechenbare Zahlenfunktionen?

Definition 53:

Seien $r, s \in \mathbb{N}$ und $r, s \geq 1$.

Eine (partielle) Funktion $f : \mathbb{N}^r \rightarrow \mathbb{N}^s$ heißt **Turingberechenbar** oder **partiell rekursiv** gdw. eine DTM $T = (Q, \{|\, 0\}, \Gamma, \delta, q_0, \#, F)$ existiert mit

$$q_0 |^{m_1} 0 |^{m_2} 0 \dots |^{m_r} \xrightarrow{*}_T p |^{n_1} 0 |^{n_2} 0 \dots |^{n_s}$$

mit $p \in F$, sowie

$$f(m_1, m_2, \dots, m_r) = (n_1, n_2, \dots, n_s)$$

gdw. der Funktionswert definiert ist.

einige berechenbare Funktionen

- $f : \mathbb{N} \rightarrow \mathbb{N}$ mit $f(x) := x + 1$ (Nachfolger)
- $f : \{1\}^* \rightarrow \{0, 1\}^*$ mit $f(w) := v$ mit $[w]_1 = [v]_2$
(unär \rightarrow binär Konversion)
- $f : \mathbb{N}^2 \rightarrow \mathbb{N}$ mit $f(x, y) := x + y$ (Summe)
- $f : \mathbb{N}^2 \rightarrow \mathbb{N}$ mit $f(x, y) := x \cdot y$ (Produkt)
- $f : \mathbb{N}^2 \rightarrow \mathbb{N}$ mit $f(x, y) := x^y$ (Exponentiation)

Die Darstellung der Zahlen (binär, unär oder dezimal) ist für die Turingberechenbarkeit nicht von Bedeutung. Jedoch ist die unäre Darstellung nach Vossen/Witt schlecht gewählt, denn Sie können die Zahl 0 nicht eindeutig kodieren!

Besser ist die übliche Codierung von n durch $|^{n+1}$!

Ackermannfunktion

Definition 54:

Die Funktion $ack : \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}$ sei erklärt durch:

$$ack(x, y) := \begin{cases} y + 1, & \text{falls } x = 0 \\ ack(x - 1, y), & \text{falls } x \geq 1, y = 0 \\ ack(x - 1, ack(x, y - 1)), & \text{falls } x, y \geq 1 \end{cases}$$

Was ist das Besondere an der Ackermannfunktion?

1. Sie ist **total**, d.h., überall definiert
2. Sie ist **eine der kompliziertesten Funktionen**, denn:
3. Sie ist **nicht primitiv rekursiv**, d.h., man kann sie nicht mit Programmen berechnen, die ausschließlich „for-Schleifen“ als Wiederholungsanweisung kennen, sog. LOOP-Programme!

M. Jantzen, FGI-1, SoSe 2009: 5

Donnerstag, 25. Juni 2009

5

Diagonalisierungstechnik

Existenz nicht berechenbarer Funktionen:

- DTM als endliche Zeichenkette darstellbar
 \Rightarrow Menge aller DTM abzählbar.
- Ist die Menge aller Funktionen $f : \mathbb{N} \rightarrow \{0, 1\}$ abzählbar? (**Annahme: JA!** $\Rightarrow f_1, f_2, f_3, \dots$)
- Definiere $g : \mathbb{N} \rightarrow \{0, 1\}$ durch

$$g(x) := \begin{cases} 0 & \text{falls } f_x(x) = 1 \\ 1 & \text{falls } f_x(x) = 0 \end{cases}$$

- Dann gilt: $\forall n \in \mathbb{N} : g \neq f_n$ **Widerspruch!!!**

Also muss es nicht berechenbare Funktion geben!

M. Jantzen, FGI-1, SoSe 2009: 6

Donnerstag, 25. Juni 2009

6

Existenz nicht abzählbarer Mengen

Sei $r \in \mathbb{R}$ mit $0 \leq r \leq 1$ beliebig. Definiere Funktion $f_r : \mathbb{N} \rightarrow \mathbb{N}$ durch:

$$f_r(n) := \begin{cases} 1 & \text{falls } n \text{ Präfix des Nachkommanteils von } r \text{ ist,} \\ 0 & \text{sonst.} \end{cases}$$

Die Menge $\{f_r \mid r \in]0, 1[, r \text{ irrational}\}$ ist nicht abzählbar.

Die Menge aller Computerprogramme ist jedoch abzählbar, weil jedes Programm in endlichem Text notiert.

Turingsche These

Turingsche These oder Churchsche These

Jede intuitiv berechenbare Funktion kann auch von einer Turingmaschine berechnet werden kann.

Was intuitiv berechenbar ist kann nicht definiert werden!

Daher ist die Turingmaschine das Standardmodell mit dem Berechenbarkeit und der Algorithmusbegriff definiert wird!

Ein *Algorithmus* ist eine präzise, das heißt in einer festgelegten Sprache abgefasste endliche Beschreibung eines allgemeinen Verfahrens unter Verwendung effektiv ausführbarer, elementarer Verarbeitungsschritte.

Eigenschaften von Algorithmen

Ein Algorithmus soll:

- schrittweise arbeiten (**Diskretheit**),
 - nach jedem Schritt eindeutig bestimmen, was der nächste Schritt ist (**Determiniertheit**),
 - einfache Schritte enthalten (**Elementarität**).
 - auf eine hinreichend große Klasse von Instanzen anwendbar sein (**Generalität**).
 - sich mit endlichen Mitteln beschreiben lassen (**endliche Beschreibbarkeit**)
- nach endlichen vielen Schritten zu einer Lösung führen (**Konklusivität**). *Dies ist nicht notwendig, aber oft erwünscht!*

M. Jantzen, FGI 1, SoSe 2009: 9

Donnerstag, 25. Juni 2009

9

Kostenmaß bei Turingmaschinen

Definition 55:

Bei der TM wird die Berechnungskomplexität über ein einfaches Kostenmaß definiert:

- Ein *Schritt* (Konfigurationsübergang) kostet eine *Zeiteinheit*;
- eine *Bandzelle* kostet eine *Platzeinheit*.

Wir werden diese Definition besonders bei der Berechnung und Bestimmung der Komplexität von Algorithmen benötigen, welche durch Turingmaschinen ausgeführt werden!

M. Jantzen, FGI-1, SoSe 2009: 10

Donnerstag, 25. Juni 2009

10

Definition 56:

- Ein **linear beschränkter Automat** (*LBA*) ist eine *NTM*, die bei beliebiger Eingabe w auf dem Arbeitsband höchstens $c \cdot |w|$ Felder bis zur Akzeptierung besucht.
- $c \in \mathbb{R}^+$ ist eine Konstante, die nicht von w abhängt.
- Arbeitet die TM bei gleicher Beschränkung ihres Arbeitsbandes deterministisch, so wird der linear beschränkte Automat mit *DLBA* abgekürzt.

Oft wird nur $c=1$ betrachtet, aber jeder *LBA* der allgemeineren Definition lässt sich durch den spezielleren simulieren, siehe 2. Teil des Beweises!

Äquivalenz von \mathcal{LBA} und \mathcal{C}_s

Satz 31:

$$\mathcal{LBA} = \mathcal{C}_s$$

Beweis:

1. $\mathcal{C}_s \subseteq \mathcal{LBA}$:

Eine NTM kann auf einem zweiten Arbeitsband (Spur) die Ableitung eines Wortes mit den Produktionen der Grammatik durchführen. Wird die Eingabelänge überschritten: Abbruch.

2. $\mathcal{LBA} \subseteq \mathcal{C}_s$: **dazu: äquivalente Typ-1 Grammatik konstruieren!**

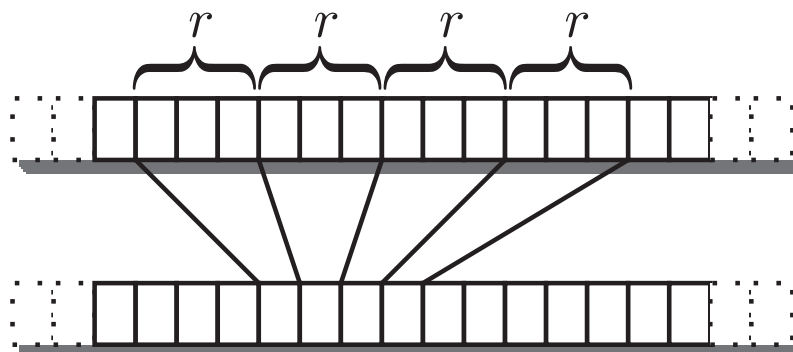
Gegeben ist ein LBA, d.h. eine TM, die mit $c \cdot |w|$ Platz auskommt. Zunächst „normieren“ wir den LBA auf maximalen Platzbedarf $|w|$.

Wir gehen etwas genauer vor, als Vossen/Witt, aber prinzipiell nach der selben Idee!

Die Familie der von *LBA*'s bzw. *DLBA*'s akzeptierten Sprachen wird mit \mathcal{LBA} bzw. \mathcal{DLBA} bezeichnet.

Bandkompression

- Für $0 \leq c \leq 1$ ist nichts zu tun.
- Ansonsten. Vergrößerung der Bandalphabets und Blockbildung.
 - Blocklänge $r \in \mathbb{N}$;
 - Bandbedarf nur noch $\frac{c}{r} \cdot |w|$;
 - Für $r := \min(n \in \mathbb{N} \mid n \geq c)$ sind das höchstens $|w|$ Felder.



M. Jantzen, FGI-1, SoSe 2009: 13

Donnerstag, 25. Juni 2009

13

Die Simulation der Rechnung mit Grammatik

Erklärung der verwendeten Nonterminale:

1. Spur	$\begin{bmatrix} x \end{bmatrix}$	Eingabesymbol
2. Spur	y	Bandsymbol
3. Spur	$\$$	Anfang-/Ende-Marker
4. Spur	$\begin{bmatrix} q \end{bmatrix}$	Zustand/Kopfposition

Die *Anfangs-Konfiguration* q_0w für ein beliebiges Wort $w \in \Sigma^*$ wird aus S mit folgenden Regeln *erzeugt*:

$$S \longrightarrow A \begin{bmatrix} x \\ x \\ \phi \\ \# \end{bmatrix}, \quad A \longrightarrow A \begin{bmatrix} x \\ x \\ \# \\ \# \end{bmatrix} \quad \text{und} \quad A \longrightarrow \begin{bmatrix} x \\ x \\ \in \\ q_0 \end{bmatrix},$$

für alle $x \in \Sigma$.

M. Jantzen, FGI-1, SoSe 2009: 14

Donnerstag, 25. Juni 2009

14

Die Simulation der Rechnung mit Grammatik

Erklärung der verwendeten Nonterminale:

1. Spur	$\begin{bmatrix} x \\ y \\ \$ \\ q \end{bmatrix}$	Eingabesymbol
2. Spur		Bandsymbol
3. Spur		Anfang-/Ende-Marker
4. Spur		Zustand/Kopfposition

Die *Anfangs-Konfiguration* q_0w für ein beliebiges Wort $w \in \Sigma^*$ wird aus S mit folgenden Regeln *erzeugt*:

$$S \longrightarrow A \begin{bmatrix} x \\ x \\ \phi \\ \# \end{bmatrix}, \quad A \longrightarrow A \begin{bmatrix} x \\ x \\ \# \\ \# \end{bmatrix} \quad \text{und} \quad A \longrightarrow \begin{bmatrix} x \\ x \\ \epsilon \\ q_0 \end{bmatrix},$$

für alle $x \in \Sigma$.

Die Simulation der Rechnung mit Grammatik

Erklärung der verwendeten Nonterminale:

1. Spur	$\begin{bmatrix} x \\ y \\ \$ \\ q \end{bmatrix}$	Eingabesymbol
2. Spur		Bandsymbol
3. Spur		Anfang-/Ende-Marker
4. Spur		Zustand/Kopfposition

Die *Anfangs-Konfiguration* q_0w für ein beliebiges Wort $w \in \Sigma^*$ wird aus S mit folgenden Regeln *erzeugt*:

$$S \longrightarrow A \begin{bmatrix} x \\ x \\ \phi \\ \# \end{bmatrix}, \quad A \longrightarrow A \begin{bmatrix} x \\ x \\ \# \\ \# \end{bmatrix} \quad \text{und} \quad A \longrightarrow \begin{bmatrix} x \\ x \\ \epsilon \\ q_0 \end{bmatrix},$$

für alle $x \in \Sigma$.

Simulation eines Linksschritts

Linksschritt: Für $x_1, x_2, x_3 \in \Sigma$, $y_1, y_2 \in \Gamma$ und für $(q, z, l) \in \delta(p, y)$, folgende Regeln in G :

$$\begin{array}{ccc}
 \begin{bmatrix} x_1 \\ y_1 \\ \# \\ \# \end{bmatrix} \begin{bmatrix} x_2 \\ y \\ \# \\ p \end{bmatrix} & \longrightarrow & \begin{bmatrix} x_1 \\ y_1 \\ \# \\ q \end{bmatrix} \begin{bmatrix} x_2 \\ z \\ \# \\ \# \end{bmatrix}, \quad \begin{bmatrix} x_1 \\ y_1 \\ \epsilon \\ \# \end{bmatrix} \begin{bmatrix} x_2 \\ y \\ \# \\ p \end{bmatrix} \longrightarrow \begin{bmatrix} x_1 \\ y_1 \\ \epsilon \\ q \end{bmatrix} \begin{bmatrix} x_2 \\ z \\ \# \\ \# \end{bmatrix} \\
 \begin{bmatrix} x_1 \\ y_1 \\ \# \\ \# \end{bmatrix} \begin{bmatrix} x_2 \\ y \\ \phi \\ p \end{bmatrix} & \longrightarrow & \begin{bmatrix} x_1 \\ y_1 \\ \# \\ q \end{bmatrix} \begin{bmatrix} x_2 \\ z \\ \phi \\ \# \end{bmatrix}, \quad \begin{bmatrix} x_1 \\ y_1 \\ \epsilon \\ \# \end{bmatrix} \begin{bmatrix} x_2 \\ y \\ \phi \\ p \end{bmatrix} \longrightarrow \begin{bmatrix} x_1 \\ y_1 \\ \epsilon \\ q \end{bmatrix} \begin{bmatrix} x_2 \\ z \\ \phi \\ \# \end{bmatrix}
 \end{array}$$

Simulation eines Rechtsschritts

Rechtsschritt: Für $x_1, x_2, x_3 \in \Sigma$, $y_1, y_2 \in \Gamma$ und für $(q, z, r) \in \delta(p, y)$, folgende Regeln in G :

$$\begin{array}{ccc}
 \begin{bmatrix} x_2 \\ y \\ \# \\ p \end{bmatrix} \begin{bmatrix} x_3 \\ y_2 \\ \# \\ \# \end{bmatrix} & \longrightarrow & \begin{bmatrix} x_2 \\ z \\ \# \\ \# \end{bmatrix} \begin{bmatrix} x_3 \\ y_2 \\ \# \\ q \end{bmatrix}, \quad \begin{bmatrix} x_2 \\ y \\ \# \\ p \end{bmatrix} \begin{bmatrix} x_3 \\ y_2 \\ \phi \\ \# \end{bmatrix} \longrightarrow \begin{bmatrix} x_2 \\ z \\ \# \\ \# \end{bmatrix} \begin{bmatrix} x_3 \\ y_2 \\ \phi \\ q \end{bmatrix} \\
 \begin{bmatrix} x_2 \\ y \\ \epsilon \\ p \end{bmatrix} \begin{bmatrix} x_3 \\ y_2 \\ \# \\ \# \end{bmatrix} & \longrightarrow & \begin{bmatrix} x_2 \\ z \\ \epsilon \\ \# \end{bmatrix} \begin{bmatrix} x_3 \\ y_2 \\ \# \\ q \end{bmatrix}, \quad \begin{bmatrix} x_2 \\ y \\ \epsilon \\ p \end{bmatrix} \begin{bmatrix} x_3 \\ y_2 \\ \phi \\ \# \end{bmatrix} \longrightarrow \begin{bmatrix} x_2 \\ z \\ \epsilon \\ \# \end{bmatrix} \begin{bmatrix} x_3 \\ y_2 \\ \phi \\ q \end{bmatrix}
 \end{array}$$

Ende der Simulation

Bei Erreichen eines Endzustands des LBA müssen alle Nonterminale in entsprechende Terminalsymbol überführt werden.

Für alle $x, z \in \Sigma$, $y \in \Gamma$ und $\& \in \{\#, \epsilon, \phi\}$ folgende Regeln:

$$\begin{bmatrix} x \\ y \\ \& \\ q \end{bmatrix} \longrightarrow x \text{ falls } q \in F, \quad \begin{bmatrix} x \\ y \\ \& \\ \# \end{bmatrix} z \longrightarrow xz, \quad z \begin{bmatrix} x \\ y \\ \& \\ \# \end{bmatrix} \longrightarrow zx$$

Problem/Fehler dieser Simulation

... leider steckt ein Fehler in diesem Satz von Regeln!

- Es können nur Wörter w mit $|w| \geq 2$ generiert werden!
- Warum? ϕ und ϵ sind keine echten Begrenzer!
- Abhilfe: alle Wörter bis zur Länge 2 daraufhin untersuchen, ob sie vom LBA akzeptiert werden!

Zusätzliche Regeln $S \longrightarrow x$ für diejenigen $x \in \{\epsilon\} \cup \Sigma$, die dazugehören müssen! Warum ist $x \in L(LBA)$ entscheidbar?

- Es gibt nur endlich viele Rechnungen *ohne Schleifen* für ein Wort!

Das ist der Grund!

Vorteile der *LBA*-Darstellung

Mit den Produktionen kann ein terminales Wort erzeugt werden, gdw. es eine Erfolgsrechnung für w im *LBA* mit $|w|$ Speicherplatz gibt.

- Mit Hilfe der Charakterisierungen von Sprachfamilien durch Automaten, lassen sich häufig Abschlusseigenschaften leichter beweisen als mit Grammatiken.
- Die Familie $\mathcal{C}s = \mathcal{LBA}$ ist gegen Durchschnittsbildung abgeschlossen.
- Beweisidee: Spurenbildung und Kombination der beiden *LBA*'s

Typ-0 Sprachen und Turingmaschinen

Satz 32:

$$\mathcal{R}e = \mathcal{L}_0$$

Beweis:

1. $\mathcal{L}_0 \subseteq \mathcal{R}e$:

Eine NTM kann auf einem zweien Arbeitsband (Spur) die Ableitung eines Wortes mit den Produktionen der Grammatik durchführen.

2. $\mathcal{R}e \subseteq \mathcal{L}_0$:

Die Simulation eines *LBA* durch eine kontextsensitive Grammatik kann abgewandelt werden für beliebige Einband *NTM*'s und erfordert Typ-0 Grammatik.

charakteristische Funktion

Diese Funktion, gibt Auskunft über das Vorhandensein von Elementen in einer Menge:

Definition 57:

Sei $M \subseteq C$, dann ist die **charakteristische Funktion** von M die auf ganz C definierte Funktion $\chi_M : C \longrightarrow \{0, 1\}$ mit

$$\chi_M(x) := \begin{cases} 1 & \text{falls } x \in M \\ 0 & \text{sonst.} \end{cases}$$

Unter Benutzung des Iverson'schen Prädikates könnte man auch dies schreiben:

$$\chi_M(x) := [x \in M].$$

$$|2^M| = 2^{|M|}$$

Beweis: Es gibt genau $2^{|M|}$ viele Binärzahlen mit $|M|$ vielen Stellen. Jeder Stelle ist ein Element m aus M zugeordnet und enthält 1 gdw. $m \in M$ gilt.

M. Jantzen, FGI-1, SoSe 2009: 21

Donnerstag, 25. Juni 2009

21

charakteristische Funktion

Diese Funktion, gibt Auskunft über das Vorhandensein von Elementen in einer Menge:

Definition 57:

Sei $M \subseteq C$, dann ist die **charakteristische Funktion** von M die auf ganz C definierte Funktion $\chi_M : C \longrightarrow \{0, 1\}$ mit

$$\chi_M(x) := \begin{cases} 1 & \text{falls } x \in M \\ 0 & \text{sonst.} \end{cases}$$

Unter Benutzung des Iverson'schen Prädikates könnte man auch dies schreiben:

$$\chi_M(x) := [x \in M].$$

Was ist das denn?

$$|2^M| = 2^{|M|}$$

Beweis: Es gibt genau $2^{|M|}$ viele Binärzahlen mit $|M|$ vielen Stellen. Jeder Stelle ist ein Element m aus M zugeordnet und enthält 1 gdw. $m \in M$ gilt.

M. Jantzen, FGI-1, SoSe 2009: 21

Donnerstag, 25. Juni 2009

21

Iverson'sches Prädikat

Definition 59:

Ist p ein Prädikat, so ist $[p]$ (also das Prädikat p in eckigen Klammern), folgendermaßen definiert:

$$[p] := \begin{cases} 1 & \text{falls } p \text{ wahr ist} \\ 0 & \text{falls } p \text{ nicht wahr ist.} \end{cases}$$

Beispiel:

Es sind „ $3 \cdot 2 = 6$ “ und „ $\pi = 3,14$ “ zwei Prädikate für die gemäß Definition gilt:

$$1 = [3 \cdot 2 = 6] \quad \text{und} \quad 0 = [\pi = 3,14]$$

Entscheidbarkeit

„Entscheidbarkeit“ von Mengen kann auf unterschiedliche Weisen definiert werden; dieses ist oft die Basisdefinition:

Definition 58:

Eine Menge $L \subseteq \Sigma^*$ heißt (relativ zu Σ^*) **entscheidbar** oder **rekursiv** gdw. ihre charakteristische Funktion $\chi_L : \Sigma^* \rightarrow \{0, 1\}$ berechenbar ist.

Die Klasse aller entscheidbaren Mengen wird mit $\mathcal{R}ec$ (*recursive sets*) bezeichnet.

Daraus folgt, dass jede endliche Menge entscheidbar ist:

Die endliche Kontrolle der DTM ist im Wesentlichen ein DFA der die endliche Menge akzeptiert!

Die DTM prüft bei jeder Eingabe, ob diese in dieser endlichen Menge vorkommt. Wenn ja, so wird 1 ausgegeben, sonst 0.

Rekursivität von $\mathcal{LBA} = \mathcal{C}_S$

Satz 33:

$$\mathcal{C}_S \subseteq \mathcal{R}ec$$

Der Beweis ist am leichtesten mit Grammatiken zu führen:

- Für jedes $L \in \mathcal{C}_S$ und jedes Wort $w \in \Sigma^*$ kann entschieden werden, ob $w \in L$ gilt.
 - Es gibt endlich viele Wörter v mit $|v| \leq |w|$.
 - Es gibt eine kontextsensitive Grammatik $G = (\Sigma, N, P, S)$ mit $L(G) = L$.
 - Keine Regel verkürzt die Satzform!
 - Ohne Satzformwiederholungen gibt es nur endlich viele Ableitungsschritte, bis die Satzform zu lang ist!!

$\mathcal{C}_S \neq \mathcal{R}ec$

Satz 34:

$$\mathcal{C}_S \not\subseteq \mathcal{R}ec \not\subseteq \mathcal{R}e$$

Diagonalbeweis:

- Die Menge der kontextsensitiven Grammatiken ist abzählbar: G_1, G_2, \dots
- Sei $f : \{0, 1\}^* \rightarrow \mathbb{N}$ eine berechenbare Bijektion, mit: $f(w) = i$ gdw. w ist i -tes Wort in der lexikalischen Ordnung auf $\{0, 1\}^*$.
- $L_e := \{w \mid w \notin L(G_{f(w)})\}$ ist entscheidbar!
- ABER: L_e ist nicht kontextsensitiv!

Beweisfortsetzung

Annahme: $\exists n$ mit $L(G_n) = L_e$.

Für das n -te Wort u der lexikalischen Aufzählung von $\{0,1\}^*$ mit $f(u) = n$ ergibt sich ein Widerspruch für beide möglichen Fälle $u \in L_e$ und $u \notin L_e$:

$$u \in L_e \xrightarrow{\text{(Def. von } L_e)}} u \notin L(G_n) \xrightarrow{\text{(Annahme)}} u \notin L_e.$$

Andererseits ergibt sich auch:

$$u \notin L_e \xrightarrow{\text{(Def. von } L_e)}} u \in L(G_n) \xrightarrow{\text{(Annahme)}} u \in L_e.$$

Existenz nicht entscheidbarer Mengen

Definition 60:

Sei $H := \left\{ \langle A \rangle \langle w \rangle \in \{0,1\}^* \mid \right.$
die TMA hält bei Eingabe von $w \in \Sigma^*$ an $\left. \right\}$.

Hierbei ist $\langle A \rangle$ eine Binärzahlcodierung (genauer: Godelisierung) der TM A .

**Diese Menge H ist nicht entscheidbar,
was wir im nächsten Vorlesungstermin
zeigen werden!**