Entscheidbare und unentscheidbare Probleme

Hans U. Simon (RUB)

Email: simon@lmi.rub.de

Homepage: http://www.ruhr-uni-bochum.de/lmi

Entscheidbarkeit und Semi–Entscheidbarkeit

Eine Sprache $L \subseteq \Sigma^*$ heißt entscheidbar **gdw** die charakteristische Funktion

$$\chi_L(w) := \begin{cases} 1 & \text{falls } w \in L \\ 0 & \text{falls } w \notin L \end{cases}$$

von L berechenbar ist.

Eine Sprache $L \subseteq \Sigma^*$ heißt semi–entscheidbar **gdw** die „halbe“ charakteristische Funktion

$$\chi'_L(w) := \begin{cases} 1 & \text{falls } w \in L \\ \text{„undefiniert“} & \text{falls } w \notin L \end{cases}$$

von L berechenbar ist.

Erinnerung: Sprache und Haltebereich einer DTM

Wir erinnern an die Definition der Sprache $T(M)$ und des Haltebereiches $H(M)$ einer DTM M:

$$\begin{aligned} T(M) &= \{w \in \Sigma^* \mid \exists z_e \in E, \alpha, \beta \in \Gamma^* : z_0 w \vdash^* \alpha z_e \beta\} \\ H(M) &= \{w \in \Sigma^* \mid \\ &\quad \exists z \in Z, A \in \Gamma, \alpha, \beta \in \Gamma^* : z_0 w \vdash^* \alpha z A \beta, \delta(z, A) = \text{„undefiniert“}\} \end{aligned}$$

Hierbei setzen wir folgendes voraus:

- $\delta(z_e, A) =$„undefiniert“ für alle $z_e \in E$.
- $\delta(z, A)$ mit $z \in Z \setminus E$ darf undefiniert sein (muss aber nicht).

Die DTM stoppt also stets nach endlich vielen Schritten, wenn sie auf einer Eingabe $w \in L$ gestartet wird (und evtl. bei Eingaben $w \notin L$). Somit gilt

$$T(M) \subseteq H(M) \subseteq \Sigma^* \ .$$

Stoppzustände

Ein Zustand $z \in Z \setminus E$ heißt nicht–akzeptierender Stoppzustand bzw. einfach Stoppzustand **gdw** $\delta(z, A) =$„undefiniert“ für alle $A \in \Gamma$.

Intuition:

- In Endzuständen stoppt die DTM akzeptierend.
- In Stoppzuständen stoppt die DTM nicht-akzeptierend.

In der Folge nehmen wir o.E. an, dass DTMs nur in End– oder Stoppzuständen stoppen. (Jede DTM kann leicht so modifiziert werden, dass sie anschließend diese Eigenschaft hat.)

Entscheidbarkeit (fortgesetzt)

Satz: L ist entscheidbar **gdw** es eine DTM M gibt mit

$$T(M) = L \text{ und } H(M) = \Sigma^* \ .$$

Beweis: Eine DTM, die $\chi_L(w)$ berechnet, kann so modifiziert werden, dass sie

- sich in einen Stoppzustand begibt anstatt „0“ auszugeben.

Umgekehrt kann eine DTM M mit $T(M) = M$ und $H(M) = \Sigma^*$ so modifiziert werden, dass sie

- vor dem Wechsel in einen Endzustand die Ausgabe „1“ auf das Band schreibt,
- vor dem Wechsel in einen Stoppzustand die Ausgabe „0“ auf das Band schreibt.

Semi–Entscheidbarkeit (fortgesetzt)

Satz: L ist semi–entscheidbar **gdw** es eine DTM M gibt mit $T(M) = L$.

Beweis: Eine DTM, die $\chi'_L(w)$ berechnet, ist auch ein Akzeptor von L, da sie nur bei Ausgabe 1 in einen Endzustand gelangt.

Eine DTM M mit $T(M) = L$ kann so modifiziert werden, dass sie

- vor dem Wechsel in einen Endzustand die Ausgabe „1“ auf das Band schreibt.

Sprachen und Entscheidungsprobleme

(Binäre) Entscheidungsprobleme sind Probleme, welche nur die Antworten JA oder NEIN zulassen.

Sprachen und Entscheidungsprobleme sind zwei Seiten der gleichen Münze:

- Ein Entscheidungsproblem kann auch aufgefasst werden als die Sprache aller Eingabeinstanzen, welche zur Antwort JA führen.
- Eine Sprache kann auch als das Problem aufgefasst werden zu entscheiden, ob eine Eingabeinstanz zur Sprache gehört (Wortproblem).

Wir werden daher im Folgenden „Sprache“ und „Problem“ zuweilen synonym verwenden.

Erinnerung: Abzählbarkeit

Erinnerung: Eine Menge A ist abzählbar **gdw** wenn wir ihre Elemente durchnummerieren können, d.h., wenn die Elemente sich bijektiv (1–zu–1) auf $\mathbb{N}$ oder (falls A endlich ist) auf eine endliche Teilmenge von $\mathbb{N}$ abbilden lassen.

Äquivalent hierzu können wir fordern, dass eine Abbildung $f : \mathbb{N} \rightarrow A$ existiert, so dass

$$A = \{f(0), f(1), f(2), \ldots\} \ .$$

Beachte, dass $f(i) = f(j)$ für $i \neq j$ zulässig ist (sonst würden endliche Mengen A ausgeschlossen).

- Jede Teilmenge einer abzählbaren Menge ist ebenfalls abzählbar.
- Da die Wortmenge Σ^* über einem endlichen Alphabet Σ abzählbar ist, ist jede formale Sprache $L \subseteq \Sigma^*$ eine abzählbare Menge.

Aufzählbarkeit

intuitiv: „Aufzählbarkeit“ = „algorithmisch durchführbare Abzählbarkeit“.

Formale Definition: Eine Sprache L heißt (rekursiv) aufzählbar **gdw** $A = \emptyset$ oder es gibt eine total berechenbare (= total definierte und berechenbare) Abbildung f mit

$$L = \{f(0), f(1), f(2), \ldots\} \ .$$

Eine DTM zur Berechnung von f nennen wir im Folgenden eine „Abzählmaschine“ für L.

- Σ^* ist aufzählbar.
- Es gibt nicht aufzählbare formale Sprachen (Beispiele hierfür später).
- Die Teilmenge einer aufzählbaren Menge ist nicht notwendig aufzählbar.

Eine Abzählmaschine für $\{0,1\}^*$

Betrachte die Abbildung $f_* : \mathbb{N} \to \{0,1\}^*$, die i abbildet auf den i–ten Binärstring der unendlichen Liste

$$\varepsilon, 0, 1, 00, 01, 10, 11, 000, \ldots \quad . \tag{1}$$

Also

$f_*(0) = \varepsilon, f_*(1) = 0, f_*(2) = 1, f_*(3) = 00, f_*(4) = 01, f_*(5) = 10, f_*(6) = 11, \ldots$

Die Berechnung von $f_*(i)$ (mit $bin(i)$ auf Band 1) ist nicht schwer:

- Zähle auf Band 2 einen Zähler hoch, der die Binärstrings gemäß (1) durchläuft.
- Zähle parallel dazu die Eingabe $bin(i)$ auf Band 1 runter.
- Wenn der Zähler von Band 1 auf Null steht, gib den auf Band 2 stehenden String aus.

Äquivalenz von Aufzählbarkeit und Semi–Entscheidbarkeit

Satz Eine Sprache L ist aufzählbar **gdw** L semi–entscheidbar ist.

Wir haben zwei Beweisrichtungen:

1. Transformation einer Abzählmaschine für L in eine DTM zur Berechnung von χ'_L.
2. Transformation einer DTM zur Berechnung von χ'_L in eine Abzählmaschine für L (vorausgesetzt $L \neq \emptyset$).

1. Beweisrichtung

Gegeben: eine Abzählmaschine M für L, die eine passende Funktion f berechnet

Gesucht: eine DTM M' zur Berechnung von $\chi'(w)$

Methode: Für $i = 0, 1, 2, \ldots$ mache folgendes:

1. Berechne $f(i)$ mit Hilfe von M.
2. Falls $f(i) = w$, dann gib 1 aus und stoppe.

- Ein Eingabewort $w \in L$ taucht für mindestens einen Index i als $w = f(i)$ in der Abzählung auf und führt zur Ausgabe 1.
- Für $w \notin L$ gerät M' in eine Endlosschleife (was der Berechnung der partiellen Funktion χ'_L ja nicht widerspricht).

2. Beweisrichtung

Erinnerung: berechenbare Bijektion $c : \mathbb{N}^2 \to \mathbb{N}$ mit berechenbaren Umkehrfunktionen

Gegeben: eine DTM M' zur Berechnung von χ'_L, Abzählmaschine für $\{0,1\}^*$

Gesucht: eine Abzählmaschine M für L, die eine passende Funktion $f(n)$ berechnet

Methode:
1. Zu Eingabe n berechne $(x, y) \in \mathbb{N}^2$ mit $n = c(x, y)$.
2. Simuliere y Schritte von M' auf Eingabe $x' := f_*(x)$.
3. Falls M' in dieser Zeit Ausgabe „1“ produziert, dann gib x' aus; andernfalls gib einen „Default–String“ $w \in L$ aus.

- Der Trick, Eingabe n als Paar (x, y) und y als Laufzeitschranke für die Simulation von M' aufzufassen, wird „dove tailing” genannt.
- Er ist notwendig, da M' evtl. in eine Endlosschleife gerät, aber Abzählmaschine M eine totale Funktion f berechnen soll.

Überblick zur Semi–Entscheidbarkeit

Folgende Aussagen zu einer Sprache $L \subseteq \Sigma^*$ sind äquivalent:

- L ist aufzählbar.
- L ist semi–entscheidbar.
- L hat eine DTM als Akzeptor.
- L hat eine NTM als Akzeptor.
- L ist vom Typ 0.

Abschluss–Eigenschaften

Satz 1: Die Klasse der entscheidbaren Sprachen ist abgeschlossen unter den Operationen „$\cup, \cap, \neg, \cdot, *$“.

Satz 2: Die Klasse der semi–entscheidbaren Sprachen ist abgeschlossen unter den Operationen $\cup, \cap, \cdot, *$, aber (wie wir später noch zeigen werden) nicht unter der Operation $\neg$.

- Der Nachweis der Abschluss–Eigenschaften kann (relativ leicht) geführt werden, indem zwei gegebene DTMs für L_1 und L_2 benutzt werden, um DTMs für

$$L_1 \cup L_2, L_1 \cap L_2, \bar{L}_1, L_1 \cdot L_2, L_1^*$$

  zusammenzubasteln (Syntheseprobleme).

- Bei Semi–Entscheidbarkeit könnte man alternativ auch über NTMs oder Typ 0 Grammatiken argumentieren.

Mehr Details evtl. in der Vorlesung.

Zweimal „halb“ macht „ganz“

Satz: Eine Sprache L ist entscheidbar **gdw** L und $\bar{L}$ semi–entscheidbar sind.

$\Rightarrow$:

Wenn L entscheidbar ist, dann ist auch $\bar{L}$ entscheidbar.
Da jede entscheidbare Sprache erst recht semi–entscheidbar ist, sind folgerichtig dann L und $\bar{L}$ semi–entscheidbar.

$\Leftarrow$:

Wenn L und $\bar{L}$ semi–entscheidbar sind, sagen wir $\chi'_{\bar{L}}$ und χ'_L werden durch DTMs M_0 und M_1 berechnet, dann ist χ_L nach folgendem Muster berechenbar:
Für $t = 0, 1, 2, \ldots$ mache folgendes:

1. Simuliere M_0 und M_1 jeweils auf Eingabe w für t Schritte.
2. Sowie eine der DTMs, sagen wir M_i, eine 1 ausgibt und stoppt, dann gib i aus und stoppe ebenfalls.

Eine binäre Kodierung von Turing–Maschinen

Arbeitsalphabet und Zustandsmenge können stets so gewählt werden, dass jedes Symbol und jeder Zustand eine Nummer erhält:

$$\begin{aligned} \Gamma &= \{A_0, \ldots, A_r\} \\ Z &= \{z_0, \ldots, z_s\} \end{aligned}$$

Ebenso können die Richtungsangaben nummeriert werden:

$$d_0 = L, d_1 = R, d_2 = N$$

Binäre Kodierung von Turing–Maschinen (fortgesetzt)

- Ein Eintrag
$$\delta(z_i, A_j) = (z_{i'}, A_{j'}, d_k)$$
der Turing–Tafel kann dann durch den String
$$\#\#bin(i)\#bin(j)\#bin(i')\#bin(j')\#bin(k)$$
kodiert werden.

- Die komplette Turing–Tafel ist dann kodiert durch die Konkatenation der Kodewörter ihrer Einträge (wobei diese, sagen wir, zeilenweise durchlaufen werden).

- Schließlich erhalten wir ein binäres Kodewort durch die Substitutionen
$$0 \mapsto 00, 1 \mapsto 01, \# \mapsto 11 \ \ .$$

Binäre Kodierung von Turing–Maschinen (fortgesetzt)

Es ist nicht schwer zu zeigen, dass

$$G := \{w \in \{0,1\}^* |\ w \text{ ist Codewort einer DTM}\}$$

entscheidbar ist.

- Falls $w \in G$, dann bezeichne M_w die von w kodierte DTM.
- Falls $w \notin G$, dann bezeichne M_w eine (beliebig aber fest ausgewählte) „Default–DTM".

Universelle Turing–Maschine

Die universelle Sprache ist definiert wie folgt:

$$U := \{w\#x \mid x \in T(M_w)\}$$

Eine DTM heißt universelle Turing–Maschine **gdw** sie ein Akzeptor von U ist.

Eine universelle Turing–Maschine ist eine Art „General Purpose Computer“, der auf Eingaben der Form $w\#x$ vorgeht wie folgt:

- Simuliere M_w auf x.
- Akzeptiere $w\#x$ **gdw** M_w ihre Eingabe x akzeptiert.

Und es gibt sie wirklich ...

Das folgende Resultat ist nicht schwer zu zeigen:

Satz: Es gibt eine universelle Turing–Maschine.

Bezeichnung: UTM

Folgerung 1: U ist semi–entscheidbar.

Eine nicht semi–entscheidbare Sprache

Die sogenannte Diagonalsprache ist definiert wie folgt:

$$D := \{w \in \{0,1\}^* | \; w \notin T(M_w)\}$$

In Worten: D besteht aus allen (Kodierungen von) DTMs, die ihre eigene Beschreibung (durch ein Kodewort) nicht akzeptieren.

Satz: D ist nicht semi–entscheidbar.

Beweis durch Widerspruch: Wir machen die (heuchlerische) Annahme, es gäbe eine DTM M_0 mit $T(M_0) = D$. Betrachte das Kodewort w_0 von M_0. Die folgenden Aussagen sind äquivalent:

(1) $w_0 \in T(M_0)$ (2) $w_0 \in D$. (3) $w_0 \notin T(M_0)$.

- Zur Äquivalenz von (1) und (2) nutze aus, dass $T(M_0) = D$.
- Zur Äquivalenz von (2) und (3) nutze die Definition von D aus.
- Die Äquivalenz von (1) und (3) ist ein WIDERSPRUCH.

Reduzierbarkeit

Definition: Betrachte zwei Sprachen $L_1, L_2 \subseteq \Sigma^*$.
L_1 heißt reduzierbar auf L_2 **gdw** eine total berechenbare Abbildung

$$f : \Sigma^* \rightarrow \Sigma^*$$

existiert mit der Eigenschaft

$$\forall w \in \Sigma^* : w \in L_1 \Leftrightarrow f(w) \in L_2 \quad .$$

f nennen wir in diesem Zusammenhang eine Reduktionsabbildung.

Notation: $L_1 \leq L_2$.

Eigenschaften dieser Relation

Reflexivität: $L \leq L$.

Transitivität: Aus $L_1 \leq L_2$ und $L_2 \leq L_3$ folgt $L_1 \leq L_3$.

- Zum Nachweis der Reflexivität benutze die identische Reduktionsabbildung $f(w) = w$.
- Zum Nachweis der Transitivität setze die Reduktionsabbildungen f_1 und f_2 für die Reduktionen $L_1 \leq L_2$ und $L_2 \leq L_3$ zu einer Reduktionsabbildung $f(w) := f_2(f_1(w))$ für die Reduktion $L_1 \leq L_3$ zusammen:

$$w \in L_1 \Leftrightarrow f_1(w) \in L_2 \Leftrightarrow f_2(f_1(w)) \in L_3$$

Eigenschaften (fortgesetzt)

Voraussetzung: $L_1 \leq L_2$ (Reduktionsabbildung f).

Behauptungen: 1. Falls L_2 (semi–)entscheidbar ist, dann ist auch L_1 (semi–)entscheidbar.

2. Falls L_1 nicht (semi–)entscheidbar ist, dann ist auch L_2 nicht (semi–)entscheidbar.

Beweis: 1. Wegen $w \in L_1 \Leftrightarrow f(w) \in L_2$ gilt

$$\chi'_{L_1}(w) = \chi'_{L_2}(f(w) \ .$$

Abbildung $\chi'_{L_1}(w)$ kann also berechnet werden, indem zunächst $f(w)$ und anschließend $\chi'_{L_2}(f(w)$ berechnet wird. Eine analoge Bemerkung gilt für die Funktion $\chi_{L_1}(w)$.

2. Die zweite Behauptung ist logisch äquivalent zur ersten. (Umkehrschluss: $A \Rightarrow B$ ist logisch äquivalent zu $\neg B \Rightarrow \neg A$.)

Verbreitung „guter und schlechter Nachrichten“

Betrachte eine Reduktionskette

$$L_1 \le L_2 \le \cdots \le L_{k-1} \le L_k \ .$$

- Wenn L_k (semi–)entscheidbar ist, so auch $L_{k-1}, \ldots, L_2, L_1$.
- Wenn L_1 nicht (semi–)entscheidbar ist, so auch $L_2, \ldots, L_{k-1}, L_k$.

Salopp formuliert:

- „Gute Nachrichten“ verbreiten sich entlang von Reduktionsketten von rechts nach links.
- „Schlechte Nachrichten“ verbreiten sich entlang von Reduktionsketten von links nach rechts.

Wir werden diese Denkweise ausnutzen, um aus der Diagonalsprache D weitere nicht entscheidbare bzw. nicht semi–entscheidbare Sprachen abzuleiten.

Eine kleine Sammlung nicht entscheidbarer Sprachen

Neben dem Komplement der Diagonalsprache

$$\bar{D} = \{w \in \{0,1\}^* | \ w \in T(M_w)\}$$

und der universellen Sprache

$$U = \{w\#x | \ x \in T(M_w)\}$$

betrachten wir noch die folgenden Sprachen:

$$
\begin{array}{rcll}
H & := & \{w\#x | \ x \in H(M_w)\} & \text{(Halteproblem)} \\
H_0 & := & \{w | \ \varepsilon \in H(M_w)\} & \text{(Halteproblem auf leerem Band)} \\
K & := & \{w | \ w \in H(M_w)\} &
\end{array}
$$

Kleine Sammlung (fortgesetzt)

Bei diesen Sprachen geht es also um die folgenden Fragen:

$\bar{D}$: Akzeptiert eine DTM ihre eigene Beschreibung ?

U: Akzeptiert eine DTM ihre Eingabe ?

H: Stoppt eine DTM auf ihrer Eingabe nach endlich vielen Schritten ?

H_0: Stoppt eine auf das leere Band angesetzte DTM nach endlich vielen Schritten ?

K: Stoppt eine auf ihre eigene Beschreibung angesetzte DTM nach endlich vielen Schritten ?

Alle diese Fragen werden sich als nicht entscheidbar erweisen.

Kleine Sammlung (fortgesetzt)

Satz: $\bar{D}$ ist nicht entscheidbar.

Beweis: Wäre $\bar{D}$ entscheidbar, so wäre auch D entscheidbar.
D ist aber noch nicht einmal semi–entscheidbar.

Satz: $K = \{w|\ w \in H(M_w)\}$ ist semi–entscheidbar.

Beweis: Verwende eine (vereinfachte) Variante UTM' der UTM, die folgendes macht:

1. Simuliere (Schritt für Schritt) die DTM M_w auf Eingabe w.
2. Falls M_w irgendwann stoppt, dann produziere Ausgabe „1" und stoppe ebenfalls.

Offensichtlich berechnet UTM' die Funktion $\chi'_K(w)$.

Kleine Sammlung (fortgesetzt)

Wir werden die folgende Reduktionskette nachweisen:

$$\bar{D} \leq U \leq H \leq H_0 \leq K$$

Wegen der Art, wie sich gute und schlechte Nachrichten verbreiten, erhalten wir die

Folgerung
$\bar{D}, U, H, H_0, K$ sind zwar semi-entscheidbar aber nicht entscheidbar.

Bleibt der Nachweis der obigen Reduktionskette.

Reduktion 1

Lemma: $\bar{D} \leq U$.

Verwende Reduktionsabbildung

$$w \mapsto w\#w \quad .$$

Offensichtlich gilt:

$$w \in \bar{D} \Leftrightarrow w \in T(M_w) \Leftrightarrow w\#w \in U \quad .$$

Reduktion 2

Lemma: $U \leq H$.

Verwende Reduktionsabbildung

$$w\#x \mapsto w'\#x \quad .$$

Ziel: $w\#x \in U \Leftrightarrow x \in T(M_w) \Leftrightarrow x \in H(M_{w'}) \Leftrightarrow w'\#x \in H$.

Ändere dazu das „Programm“ w von M_w zu einem neuen „Programm“ w' einer DTM $M_{w'}$ ab:

- $M_{w'}$ simuliert M_w Schritt-für-Schritt,
- außer dass $M_{w'}$ sich in eine Endlosschleife begibt, falls M_w nicht–akzeptierend stoppt.

Reduktion 3

Lemma: $H \leq H_0$.

Verwende Reduktionsabbildung

$$w\#x \mapsto w' \ .$$

Ziel: $w\#x \in H \Leftrightarrow x \in H(M_w) \Leftrightarrow \varepsilon \in H(M_{w'}) \Leftrightarrow w' \in H_0$.

Ändere dazu das „Programm“ w von M_w zu einem neuen (auch von x abhängigen) „Programm“ w' einer TM $M_{w'}$ ab:

- $M_{w'}$, angesetzt auf das leere Band, schreibt zunächst den String x auf
- und simuliert dann Schritt-für-Schritt M_w auf Eingabe x.

Reduktion 4

Lemma: $H_0 \leq K$.

Verwende Reduktionsabbildung

$$w \mapsto w' \ .$$

Ziel: $w \in H_0 \Leftrightarrow \varepsilon \in H(M_w) \Leftrightarrow w' \in H(M_{w'}) \Leftrightarrow w' \in K$.

Ändere dabei das „Programm“ w von M_w zu einem neuen Programm w' einer TM $M_{w'}$ ab:

- $M_{w'}$ löscht zunächst ihre Eingabe
- und simuliert dann Schritt-für-Schritt M_w angesetzt auf das leere Band.

PKP und MPKP

Postsches Korrespondenzproblem (PKP)

Entscheide zu einer gegeben Folge

$$K = [(x_1, y_1), \ldots, (x_k, y_k)]$$

von Wortpaaren über einem endlichen Alphabet Σ, ob es eine Folge

$$i_1, \ldots, i_n \in [1 : k]$$

von Indizes, genannt „Lösung“, gibt, so dass

$$x_{i_1} \ldots x_{i_n} = y_{i_1} \ldots y_{i_n} \quad .$$

Modifiziertes Postsches Korrespondenzproblem (MPKP)

Wie PKP, außer dass die Indexfolge mit $i_1 = 1$ beginnen muss.

Beispiel 1

Zu

$$K = [(1, 111), (10111, 10), (10, 0)]$$

ist $(2, 1, 1, 3)$ eine passende Indexfolge:

$$\overbrace{10111}^{x_2}\overbrace{1}^{x_1}\overbrace{1}^{x_1}\overbrace{10}^{x_3} = 101111110 = \overbrace{10}^{y_2}\overbrace{111}^{y_1}\overbrace{111}^{y_1}\overbrace{0}^{y_3}$$

Beispiel 2

Zu

$$K = [(10, 101), (011, 11), (101, 011)]$$

gibt es keine passende Indexfolge (Zugzwangargument):

1. Jede potenzielle Lösung müßte beginnen mit $i_1 = 1$:

$$x_1 = 10,\ y_1 = 101$$

2. Wann immer die y-Sequenz eine 1 Vorsprung hat, ist die einzig aussichtsreiche Fortsetzung

$$\begin{array}{lll} x\text{-Sequenz}: & \ldots & \overbrace{101}^{x_3} \\ y\text{-Sequenz}: & \ldots & 1\,\underbrace{011}_{y_3} \end{array},$$

was den Vorsprung von der y-Sequenz auf ewig reproduziert.

Beispiel 3

Zu

$$K = [(001, 0), (01, 011), (01, 101), (10, 001)]$$

gibt es eine passende Indexfolge $i_1, \ldots, i_n$, aber erst ab $n = 66$.

Wer findet die Lösung ?

Hauptresultat

Wir werden die Reduktionskette

$$H \leq MPKP \leq PKP$$

nachweisen.

Folgerung MPKP und PKP sind nicht entscheidbar.

Bemerkungen:

1. Die Unentscheidbarkeit ergibt sich bereits für binäres Alphabet (wie sich zeigen wird).
2. Bei unärem Alphabet hingegen sind MPKP und PKP entscheidbar (s. Übung).

Reduktion von MPKP auf PKP

Die Eingabeinstanz von MPKP über Alphabet Σ sei

$$K = [(x_1, y_1), \ldots, (x_k, y_k)] \ .$$

Seien $\#, \$ \notin \Sigma$ zwei neue Symbole. Dann soll

$$f(K) = [(x'_0, y'_0), (x'_1, y'_1), \ldots, (x'_k, y'_k), (x'_{k+1}, y'_{k+1})]$$

die folgende Eingabe von PKP sein:

- Für $i = 1, \ldots, k$ entsteht x'_i, indem **hinter** jedem Buchstaben von x_i Symbol $\#$ eingefügt wird; y'_i entsteht aus y_i, indem **vor** jedem Buchstaben von y_i Symbol $\#$ eingefügt wird.
- $x'_0 = \# x'_1,\ x'_{k+1} = \$,\ y'_0 = y'_1,\ y'_{k+1} = \#\$.$

Reduktionsabbildung f an einem Beispiel

$x_1 = 10111$	$x'_1 = 1\#0\#1\#1\#1\#$	$x'_0 = \#1\#0\#1\#1\#1\#$
$y_1 = 10$	$y'_1 = \#1\#0$	$y'_0 = \#1\#0$
$x_2 = 1$	$x'_2 = 1\#$	
$y_2 = 111$	$y'_2 = \#1\#1\#1$	
$x_3 = 10$	$x'_3 = 1\#0\#$	$x'_4 = \$$
$y_3 = 0$	$y'_3 = \#0$	$y'_4 = \#\$$

$(1, 2, 2, 3)$ ist eine passende Indexfolge für $K = [(x_1, y_1), (x_2, y_2), (x_3, y_3)]$:

$$\overbrace{10111}^{x_1}\overbrace{1}^{x_2}\overbrace{1}^{x_2}\overbrace{10}^{x_3} = 101111110 = \overbrace{10}^{y_1}\overbrace{111}^{y_2}\overbrace{111}^{y_2}\overbrace{0}^{y_3} .$$

$(0, 2, 2, 3, 4)$ ist eine passende Indexfolge für $f(K)$, die folgenden „gepolsterten“ Lösungsstring liefert:

$$\#1\#0\#1\#1\#1\#1\#1\#1\#0\#\$$$

Reduktion von MPKP auf PKP (fortgesetzt)

Aus dem Design von $f(K)$ ergibt sich leicht:

1. Für alle n und alle $i_2, \ldots, i_n \in [1:k]$:

$$1, i_2, \ldots, i_n \text{ Lösung für } K \Leftrightarrow 0, i_2, \ldots, i_n, k+1 \text{ Lösung für } f(K) \ .$$

2. Jede Lösung (= passende Indexfolge) kürzester Länge für $f(K)$ startet mit Index 0, endet mit Index $k+1$ und verwendet dazwischen nur Indizes aus $\{1, \ldots, k\}$.

Es folgt:
K besitzt eine passende Indexfolge **gdw** $f(K)$ besitzt eine passende Indexfolge.
Da f außerdem berechenbar ist, ergibt sich MPKP$\leq$PKP.

Reduktion von H auf MPKP

Ziel:

Entwurf einer Reduktionsabbildung f, die Eingaben von H der Form $M\#x$, so auf Eingaben von MPKP abbildet, dass gilt:

$$x \in H(M) \Leftrightarrow f(M\#x) \text{ hat eine Lösung } . \tag{2}$$

Idee:

Um (2) zu erzwingen, werden die x- und y-Sequenzen Konfigurationsfolgen von M entsprechen, wobei die y-Sequenz immer eine Konfiguration Vorsprung hat. Der x-Sequenz erlauben wir erst nach Stoppen von M diesen Vorsprung einzuholen.

Normierung der Turing-Maschine M

Indem wir (falls nötig) die TM M leicht normieren (ohne ihr Stoppverhalten auf Eingabe x zu verändern), können wir o.E. vorausetzen:

1. M stoppt genau dann, wenn sie sich in einem Zustand aus einer Menge $S \subseteq Z$ von Stoppzuständen befindet.
2. M hat ein einseitig unendliches Band.
3. M bewegt in jedem Schritt (außer in Stoppzuständen) den Kopf.
4. M druckt niemals ihr Leerzeichen B.

Diese Normierung vereinfacht die folgende Konstruktion der MPKP-Eingabeinstanz $f(M\#x)$.

Die MPKP-Eingabeinstanz K=f(M#x)

Die Stringpaare von K zerfallen in fünf Gruppen:

- das Anfangspaar $(\#, \#z_0x\#)$
- Kopierpaare (X, X) für alle $X \in \Gamma \cup \{\#\}$
- Überführungspaare

$$\begin{array}{ll} (zY, Y'z') & \text{, falls } \delta(z,Y) = (z', Y', R) \\ (XzY, z'XY') & \text{, falls } \delta(z,Y) = (z', Y', L) \\ (z\#, Y'z'\#) & \text{, falls } \delta(z,B) = (z', Y', R) \\ (Xz\#, z'XY'\#) & \text{, falls } \delta(z,B) = (z', Y', L) \end{array}$$

  für alle $z \in Z \setminus S$, $z' \in Z$, $X, Y, Y' \in \Gamma \setminus \{B\}$.

- Löschpaare (XsY, s), $(Xs\#, s\#)$, $(\#sY, \#s)$
  für alle $s \in S$, $X, Y \in \Gamma \setminus \{B\}$.
- Abschlusspaare $(s\#\#, \#)$ für alle $s \in S$.

Intuition hinter diesem Design

- Das Anfangspaar dient dazu, der y-Sequenz eine Konfiguration (hier die Anfangskonfiguration) Vorsprung zu geben.
- Die Kopier- und Überführungspaare dienen dazu, beide Sequenzen um eine Konfiguration zu verlängern.
- Die Lösch- und Abschlußpaare sollen die x-Sequenz den Vorsprung aufholen lassen, sofern ein Stoppzustand erreicht wurde.

Um zu verifizieren, dass dieser Plan aufgeht, benötigen wir das folgende Konzept der *partiellen Lösung* für K.

Partielle Lösung für K=f(M#x)

Stringpaar $(x, y) \in \Sigma^* \times \Sigma^*$ heißt eine *partielle Lösung* für K ist, wenn gilt:

1. x ist Anfangswort von y.
2. Es existiert ein $n \geq 1$ und $i_1, \ldots, i_n$ mit $i_1 = 1$ (das Anfangspaar), so dass x die x-Sequenz und y die y-Sequenz zu $i_1, \ldots, i_n$ ist.

Zentrale Beobachtung:

Falls M aus Startkonfiguration z_0x die Folgekonfigurationen

$$\alpha_1 z_1 \beta_1, \alpha_2 z_2 \beta_2, \ldots, \alpha_k z_k \beta_k \text{ mit } z_0, \ldots, z_{k-1} \notin S$$

produziert, dann besitzt K eine partielle Lösung der Form

$$\begin{aligned}(x, y) \quad = \quad & (\#z_0w\#\alpha_1z_1\beta_1\#\ldots\#\alpha_{k-1}z_{k-1}\beta_{k-1}\# \ , \\ & \#z_0w\#\alpha_1z_1\beta_1\#\ldots\#\alpha_{k-1}z_{k-1}\beta_{k-1}\#\alpha_kz_k\beta_k\#)\end{aligned} \tag{3}$$

Induktiver Beweis der zentralen Beobachtung

Der Beweis erfolgt durch Induktion nach k.

k = 0: $(x, y) = (\#, \#z_0w\#)$ realisiert durch das Anfangspaar.

Schritt von k auf k+1: Sei per Induktionsvoraussetzung eine partielle Lösung der Form (3) gegeben und $z_k \notin S$. Wir können die x-Sequenz um $\alpha_k z_k \beta_k \#$ und die y-Sequenz um $\alpha_{k+1} z_{k+1} \beta_{k+1} \#$ verlängern, indem wir

- die identischen Teile von Konfigurationen k und $k+1$ mit den Kopierpaaren aufbauen,
- die verschiedenen Teile (lokale Umgebung der Zustandssymbole z_k, z_{k+1}, weil dort jeweils der Kopf von M positioniert ist) mit dem eindeutig bestimmten Überführungspaar aufbauen.

Auf diese Weise erhalten wir die partielle Lösung

$$(x', y') = (y, y\alpha_{k+1} z_{k+1} \beta_{k+1} \#) \enspace .$$

Damit ist der induktive Beweis abgeschlossen.

Illustration des Induktionsschrittes

Es sei $ABCzDEF\#$ die Konfiguration, welche den Vorsprung der y-Sequenz ausmacht. Wir nehmen an, dass die Turing-Tafel von M die Aktion

$$\delta(z, D) = (z', D', L) \tag{4}$$

vorschreibt. Wir machen drei „Schnappschüsse“ der x- und y-Sequenz:

x-Sequenz (Einsatz Kopierpaare):	...	AB
y-Sequenz (Einsatz Kopierpaare):	...	$ABCzDEF\#AB$
x-Sequenz (Einsatz Überführungspaar):	...	$ABCzD$
y-Sequenz (Einsatz Überführungspaar:	...	$ABCzDEF\#ABz'CD'$
x-Sequenz (Einsatz Kopierpaare):	...	$ABCzDEF\#$
y-Sequenz (Einsatz Kopierpaare):	...	$ABCzDEF\#ABz'CD'EF\#$

Nachweis der Reduktionseigenschaften von f

Behauptung 1 Falls $x \in H(M)$, dann besitzt K eine Lösung.

Falls $x \in H(M)$, erhalten wir irgendwann eine partielle Lösung der Form

$$(y, y\alpha s\beta\#) \text{ mit } s \in S, \alpha, \beta \in \Gamma^* \quad .$$

Nun können wir die Kopierpaare und die Löschpaare einsetzen, um den Vorsprung $\alpha s\beta\#$ zu vermindern:

- Jede Anwendung eines Löschpaares vermindert den Vorsprung um ein α- oder β-Symbol.
- Irgendwann ist der Vorsprung auf $s\#$ zusammengeschmolzen und die Sequenzen haben die Form
$$(y', y's\#) \quad .$$
Anwendung des Abschlußpaares für s egalisiert die Sequenzen:
$$(y's\#\#, y's\#\#)$$

Illustration der „Aufholjagd“

x-Sequenz:	$\ldots$	
y-Sequenz (Endkonfiguration als Vorsprung):	$\ldots$	$ABsD\#$
x-Sequenz (Einsatz Kopierpaar):	$\ldots$	A
y-Sequenz (Einsatz Kopierpaar):	$\ldots$	$ABsD\#A$
x-Sequenz (Einsatz Löschpaar):	$\ldots$	$ABsD$
y-Sequenz (Einsatz Löschpaar):	$\ldots$	$ABsD\#As$
x-Sequenz (Einsatz Kopierpaar):	$\ldots$	$ABsD\#$
y-Sequenz (Einsatz Kopierpaar):	$\ldots$	$ABsD\#As\#$
x-Sequenz (Einsatz Löschpaar):	$\ldots$	$ABsD\#As\#$
x-Sequenz (Einsatz Löschpaar):	$\ldots$	$ABsD\#As\#s\#$
x-Sequenz (Einsatz Abschlusspaar):	$\ldots$	$ABsD\#As\#s\#\#$
x-Sequenz (Einsatz Abschlusspaar):	$\ldots$	$ABsD\#As\#s\#\#$

Nachweis der Reduktionseigenschaften von f (fortgesetzt)

Behauptung 2 Falls K eine Lösung besitzt, dann gilt $x \in H(M)$.

Indirekter Beweis: Wir zeigen, dass K keine Lösung besitzt, falls $x \notin H(M)$.

Da wir bei der Produktion von partiellen Lösungen keine Freiheiten hatten (Anwendung von anderen Paaren als die beschriebenen führt immer direkt in eine Sackgasse), besteht der einzige Lösungsversuch im Produzieren partieller Lösungen der Form (3), wobei (solange kein Stoppzustand erreicht wird), die Löschpaare nicht zum Einsatz kommen und die y-Sequenz daher stets eine Konfiguration Vorsprung hat. Eine Egalisierung der Sequenzen kann nicht stattfinden.

PKP mit binärem Alphabet

01-PKP sei das PKP über dem Alphabet $\Sigma = \{0, 1\}$.

Satz: PKP$\leq$01-PKP

Beweis: Sei K eine PKP-Eingabeinstanz über einem beliebigen Alphabet der Form $\Sigma = \{a_1, \ldots, a_m\}$. Wähle als $f(K)$ die 01-PKP-Eingabeinstanz, die aus K durch die Substitutionsregel

$$a_j \mapsto 10^j$$

hervorgeht. Die Lösungen (sofern vorhanden) für K und $f(K)$ entsprechen sich (in der offensichtlichen Weise) 1-zu-1. Insbesondere besitzt K eine Lösung **gdw** $f(K)$ eine Lösung besitzt.

Folgerung: 01-PKP ist nicht entscheidbar.

Zu 01-PKP assoziierte kontextfreie Sprachen

Zu einem Wort $w = w_1 \cdots w_n$ bezeichne $\tilde{w} = w_n \cdots w_1$ sein „Spiegelbild“. Zu der Eingabeinstanz

$$K = [(x_1, y_1), \ldots, (x_k, y_k)]$$

von 01-PKP assoziieren wir die folgenden Sprachen über dem Alphabet $\{0, 1, \$, a_1, \ldots, a_k,\}$:

$$\begin{aligned} L_1[K] &:= \{a_{i_m} \cdots a_{i_1} x_{i_1} \cdots x_{i_m} \$ \tilde{y}_{j_n} \cdots \tilde{y}_{j_1} a_{j_1} \cdots a_{j_n} \mid m, n \geq 1\} \\ L_2[K] &:= \left\{uv\$\tilde{v}\tilde{u} \mid u \in \{a_1, \ldots, a_k\}^+, v \in \{0,1\}^+\right\} \end{aligned}$$

Erinnerung: $i_1, \ldots, i_n$ ist eine Lösung für K **gdw** $x_{i_1} \cdots x_{i_n} = y_{i_1} \cdots y_{i_n}$.

Zentrale Beobachtung: $L_1[K] \cap L_2[K]$ ist identisch zu

$$\{a_{i_n} \cdots a_{i_1} x_{i_1} \cdots x_{i_n} \$ \tilde{y}_{i_n} \cdots \tilde{y}_{i_1} a_{i_1} \cdots a_{i_n} \mid i_1, \ldots, i_n \text{ ist eine Lösung für } K\} \ .$$

(Deterministische) Kontextfreiheit dieser Sprachen

Wir geben hier kfG's $G_1[K], G_2[K]$ für $L_1[K], L_2[K]$ an:

1. $G_1[K]$ enthält die Regeln

$$S_1 \rightarrow A\$B \ , \ A \rightarrow a_i A x_i \mid a_i x_i \ , \ B \rightarrow \tilde{y}_i B a_i \mid \tilde{y}_i a_i$$

   für $i = 1, \ldots, k$.

2. $G_2[K]$ enthält die Regeln

$$S_2 \rightarrow a_i S_2 a_i \mid a_i R a_i \ , \ R \rightarrow 0R0 \mid 1R1 \mid 00 \mid 11$$

   für $i = 1, \ldots, k$.

Übung: Grammatiken G_1 und G_2 sind eindeutig und es lassen sich auch DPDA's $M_1[K]$ und $M_2[K]$ für $L_1[K]$ und $L_2[K]$ angeben.

Folgerung: $L_1[K], L_2[K]$ sind eindeutige deterministisch kontextfreie Sprachen.

Beispiel

Zu $K = [(1,111),(10111,10),(10,0)]$ erhalten wir bei G_1 die Regeln

$$\begin{array}{rcl} S_1 & \rightarrow & A\$B \\ A & \rightarrow & a_1A1|a_2A10111|a_3A10|a_11|a_210111|a_310 \\ B & \rightarrow & 111Ba_1|01Ba_2|0B|a_3|111a_1|01a_2|0 \end{array}$$

und bei G_2 die Regeln

$$\begin{array}{rcl} S_2 & \rightarrow & a_1S_2a_1|a_2S_2a_2|a_3S_2a_3|a_1Ra_1|a_2Ra_2|a_3Ra_3 \\ R & \rightarrow & 0R0|1R1|00|11 \ . \end{array}$$

Zwei sehr unterschiedliche Szenarios

K hat eine Lösung	K hat keine Lösung
1. $L_1[K] \cap L_2[K] \neq \emptyset$	1'. $L_1[K] \cap L_2[K] = \emptyset$
2. $\|L_1[K] \cap L_2[K]\| = \infty$	2'. $\|L_1[K] \cap L_2[K]\| < \infty$
3. $L_1[K] \cap L_2[K]$ ist nicht kontextfrei	3'. $L_1[K] \cap L_2[K]$ ist kontextfrei
4. $\overline{L_1[K]} \cup \overline{L_2[K]}$ ist nicht regulär	4'. $\overline{L_1[K]} \cup \overline{L_2[K]}$ ist regulär
5. $\overline{L_1[K]} \cup \overline{L_2[K]}$ ist nicht deterministisch kontextfrei	5'. $\overline{L_1[K]} \cup \overline{L_2[K]}$ ist deterministisch kontextfrei

Begründungen

- Unsere „zentrale Beobachtung“ liefert 1. und 1’.
- Aus $L_1[K] \cap L_2[K] = \emptyset$ (und somit $\overline{L_1[K]} \cup \overline{L_2[K]} = \overline{L_1[K] \cap L_2[K]} = \Sigma^*$) ergeben sich unmittelbar die Aussagen 2’. bis 5’.
- Zu Aussage 2. nutze aus, dass K unendlich viele Lösungen hat, sofern es mindestens eine Lösung hat.
- Aussage 3. ergibt sich leicht mit dem Pumping–Lemma (Übung).
- Aussage 5. ergibt sich mit einem Beweis durch Widerspruch:
  Wäre $\overline{L_1[K]} \cup \overline{L_2[K]}$ deterministisch kontextfrei, dann müsste auch $L_1[K] \cap L_2[K]$ deterministisch kontextfrei sein im Widerspruch zu 3.
- Aussage 4. ergibt sich direkt aus 5.

Folgerungen

Die folgenden Probleme sind nicht entscheidbar (Reduktion jeweils von PKP bzw. dem Komplement von PKP):

1. Ist der Durchschnitt zweier (durch DPDAs oder kfGs gegebener) deterministisch kontextfreier Sprachen leer (Schnittproblem für deterministisch kontextfreie Sprachen) ?
2. Ist der Durchschnitt zweier (durch DPDAs oder kfGs gegebener) deterministisch kontextfreier Sprachen endlich ?
3. Ist der Durchschnitt zweier (durch DPDAs oder kfGs gegebener) deterministisch kontextfreier Sprachen kontextfrei (Kontextfreiheit des Durchschnittes) ?
4. Ist eine (durch eine kfG gegebene) kontextfreie Sprache regulär ?
5. Ist eine (durch eine kfG gegebene) kontextfreie Sprache deterministisch kontextfrei ?

Zugehörige Reduktionsabbildungen

- Für die ersten drei Reduktionen verwende die Reduktionsabbildung

$$K \mapsto (M_1[K], M_2[K]) \text{ bzw. } K \mapsto (G_1[K], G_2[K]) \enspace .$$

- Für die letzten zwei Reduktionen verwende die Reduktionsabbildung $K \mapsto G'[K]$, wobei $G'[K]$ eine (aus K berechenbare !) kfG für die (kontextfreie !) Sprache $\overline{L_1[K]} \cup \overline{L_2[K]}$ ist.

Da die kfGs $G_1[K], G_2[K]$ eindeutig sind, bleiben die ersten drei Probleme der obigen Liste unentscheidbar, selbst wenn die beteiligten kontextfreien Sprachen durch eindeutige kfGs gegeben sind.

Weitere Folgerungen

Die folgenden Probleme sind nicht entscheidbar:

Inklusionsproblem: Gilt für zwei (durch DPDAs oder kfGs gegebene) deterministisch kontextfreie Sprachen L_1, L_2 die Beziehung $L_1 \subseteq L_2$?

Eindeutigkeitsproblem: Ist eine gegebene kfG eindeutig ?

Kontextfreiheit des Komplementes: Ist das Komplement einer (durch eine kfG gegebenen) kontextfreien Sprache ebenfalls kontextfrei ?

Leerheit des Komplementes: Stimmt eine (durch eine kfG gegebene) kontextfreie Sprache mit Σ^* überein (d.h., sind alle Terminalstrings ableitbar) ?

Äquivalenzproblem: Sind zwei gegebene kfGs äquivalent (d.h., sind die von ihnen erzeugten Sprachen identisch) ?

Zugehörige Reduktionen

Das Schnittproblem für deterministisch kontextfreie Sprachen

$$N(M_1) \cap N(M_2) = \emptyset \ ?$$

ist wegen

$$N(M_1) \cap N(M_2) = \emptyset \Leftrightarrow N(M_1) \cap \overline{N(M_2)}$$

auf das Inklusionsproblem reduzierbar.

Als Reduktionsabbildung verwende

$$(M_1, M_2) \mapsto (M_1, M_2') \ ,$$

wobei M_2' ein (aus M_2 berechenbarer) DPDA für $\overline{N(M_2)}$ ist.

Zugehörige Reduktionen (fortgesetzt)

Das Schnittproblem für eindeutige kontextfreie Sprachen

$$L(G_1) \cap L(G_2) = \emptyset \ (G_1, G_2 \text{ eindeutige kfGs}) \ ?$$

ist auf das Eindeutigkeitsproblem reduzierbar.

Als Reduktionsabbildung verwende

$$(G_1, G_2) \mapsto G_3 \ \ ,$$

wobei G_3 die (aus G_1, G_2 mit der Standardtechnik berechenbare) kfG für die Sprache $L(G_1) \cup L(G_2)$ bezeichnet. Da G_1 und G_2 eindeutige kfGs sind, gibt es ein Wort mit zwei verschiedenen Syntaxbäumen über G_3 **gdw** $L(G_1) \cap L(G_2) \neq \emptyset$.

Zugehörige Reduktionen (fortgesetzt)

Das Problem der Kontextfreiheit des Durchschnittes

$$N(M_1) \cap N(M_2) \text{ kontextfrei } (M_1, M_2 \text{ DPDAs}) \ ?$$

ist auf das Problem der Kontextfreiheit des Komplementes reduzierbar.

Verwende Reduktionsabbildung

$$(M_1, M_2) \mapsto G_3' \ ,$$

wobei G_3' die (aus M_1, M_2 berechenbare) kfG für

$$\overline{N(M_1) \cap N(M_2)} = \overline{N(M_1)} \cup \overline{N(M_2)}$$

bezeichne.

Dieselbe Reduktionsabbildung reduziert das Schnittproblem für deterministisch kontextfreie Sprachen

$$N(M_1) \cap N(M_2) = \emptyset \ ?$$

auf das Problem der Leerheit des Komplementes.

Zugehörige Reduktionen (fortgesetzt)

Schließlich ist das Problem der Leerheit des Komplementes

$$L(G) = \Sigma^* \ ?$$

auf das Äquivalenzproblem reduzierbar.

Wenn G_* eine (leicht zu berechnende) kfG für Σ^* bezeichnet, dann kann als Reduktionsabbildung

$$G \mapsto (G, G_*)$$

verwendet werden.

Unentscheidbare Probleme für kontextsensitive Sprachen

„Meta–Satz“: Jedes Problem, das für den Durchschnitt zweier kontextfreier Sprachen nicht entscheidbar ist, ist auch für eine einzelne kontextsensitive Sprache nicht entscheidbar.

Beweis:

- Die gegebenen kontextfreien Grammatiken G_1, G_2 sind (erst recht) kontextsensitive Grammatiken.
- Daher sind sie (mit der Methode der Vorlesung) in äquivalente LBAs M_1, M_2 transformierbar.
- Aus M_1, M_2 lässt sich leicht ein LBA M_3 für

  $$T(M_3) = T(M_1) \cap T(M_2) = L(G_1) \cap L(G_2)$$

  zusammenbasteln.
- Daher hat die zu $L(G_1) \cap L(G_2)$ gestellte Frage (wie immer sie lautet) dieselbe Antwort wie die zu $T(M)$ gestellte Frage (Reduktionsabbildung $(G_1, G_2 \mapsto M_3)$.

Folgerungen

Die folgenden Probleme zu einer (durch einen LBA gegebenen) kontextsensitiven Sprache sind nicht entscheidbar:

- Ist die Sprache leer ?
- Ist die Sprache endlich ?
- Ist die Sprache kontextfrei ?