Rekursions- und Lerntheorie

WiSe 2010/11; Univ. Trier

Henning Fernau
Universität Trier
fernau@uni-trier.de

Rekursions- und Lerntheorie Gesamtübersicht

1. Einführung: Grundsätzliche Betrachtungen

2. Berechenbarkeitstheorie: Die Churchsche These (3-4 VL)

3. Ausblicke auf weitere Ergebnisse der Rekursionstheorie

4. Lerntheorie: Modelle und Aussagen

Aufzählbarkeit und Entscheidbarkeit

Betrachte Teilmenge $A \subseteq E^*$:

- Die *charakteristische Funktion* $\chi_A : E^* \to \{0,1\}$ ist definiert durch

$$\chi_A(w) := \begin{cases} 1, & w \in A \\ 0, & w \notin A \end{cases}$$

- Die *eingeschränkte charakteristische Funktion* $\chi'_A : E^* \multimap\!\!\to \{0,1\}$ ist definiert durch

$$\chi'_A(w) := \begin{cases} 1, & w \in A \\ \text{undefiniert}, & w \notin A \end{cases}$$

<u>Wichtig</u>: χ_A ist total; χ'_A ist partiell mit Definitionsbereich A

Aufzählbarkeit und Entscheidbarkeit

1. Eine Teilmenge $A \subseteq E^*$ heißt *entscheidbar*, wenn ihre charakteristische Funktion $\chi_A : E^* \to \{0, 1\}$ berechenbar ist.

2. Eine Teilmenge $A \subseteq E^*$ heißt *semi-entscheidbar*, wenn ihre eingeschränkte charakteristische Funktion $\chi'_A : E^* \multimap \{0, 1\}$ berechenbar ist.

Für Mengen $A \subseteq \mathbb{N}^k$ werden die Begriffe analog definiert.

Man mache sich klar, was diese Begriffe konkret für Turingmaschinen-Berechenbarkeit bedeuten.

Der Satz von Post

Satz: Eine Sprache A ist genau dann entscheidbar, wenn sowohl A als auch ihr Komplement $E^* \setminus A$ semi-entscheidbar sind.

Beweis '$\Rightarrow$': Sei A entscheidbar, d.h. χ_A ist berechenbar (mit einer Maschine M).
Konstruiere M' wie folgt:

- Bei Eingabe w berechnet M' zunächst $\chi_A(w)$ (mittels M).
- Ist das Resultat 1, so hält M' und akzeptiert w.
- Ist das Resultat 0, so startet M' eine Endloschleife.

$\Rightarrow$ M' berechnet χ'_A, d.h. A ist semi-entscheidbar.

Vertausche Rollen von 1 und 0 bei M':
$\Rightarrow$ M' berechnet jetzt $\chi'_{E^* \setminus A}$, d.h. $E^* \setminus A$ ist semi-entscheidbar.

Der Satz von Post

Satz: Eine Sprache A ist genau dann entscheidbar, wenn sowohl A als auch ihr Komplement $E^* \setminus A$ semi-entscheidbar sind.

Beweis '$\Leftarrow$': Gegeben Maschinen M‘, M'':
M' berechne χ'_A, M'' berechne $\chi'_{E^* \setminus A}$.
Konstruiere neue Turingmaschine M wie folgt:

- w sei Eingabe für M.
- M simuliert abwechselnd einen Rechenschritt von M' auf w und einen Schritt von M'' auf w.
- Hält M' an, so hält auch M und akzeptiert w.
- Hält M'' an, so hält auch M und verwirft w.
- Bei Eingabe von w hält genau eine der Maschinen M' und M'', also M hält in jedem Fall an und berechnet χ_A.

Semi-Entscheidbarkeit und rekursive Aufzählbarkeit

Eine Sprache $A \subseteq E^*$ heißt *rekursiv aufzählbar*, falls A leer ist oder Bildbereich einer totalen berechenbaren Funktion $f : \mathbb{N} \to E^*$ ist:

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

Für Teilmengen $A \subseteq \mathbb{N}^k$ wird der Begriff analog definiert.

Satz: Eine Sprache A aus E^* ist genau dann semi-entscheidbar, wenn sie rekursiv aufzählbar ist.

Semi-Entscheidbarkeit und rekursive Aufzählbarkeit; Beweis: $\Leftarrow$

Sei A rekursiv aufzählbar.

Fall (1), $A = \emptyset$: Dann $\chi_A(x) = 0$ für alle x, also A entscheidbar.

Fall (2), $A = \{f(0), f(1), \ldots\}$ mit totalem berechenbaren $f : \mathbb{N} \to E^*$:

Betrachte (berechenbares!) g definiert über

```
INPUT(w);
FOR n = 0,1,2,3,... DO
    IF f(n) = w THEN OUTPUT(1) END;
END.
```

$w \in A \Rightarrow g(w) = 1$

$w \notin A \Rightarrow g(w)$ undefiniert

$\Longrightarrow A$ semi-entscheidbar

Semi-Entscheidbarkeit und rekursive Aufzählbarkeit; Beweis: $\Rightarrow$

Sei nun A semi-entscheidbar, $A \neq \emptyset$.
M sei eine Turingmaschine, die χ'_A berechnet.
a sei beliebig gewähltes Element von A.
Betrachte (berechenbares!) g definiert über

```
INPUT(n);
k := p1(n); l := p2(n); w := ν(k);
IF  Angesetzt auf w stoppt M
          nach höchstens l Schritten mit Ausgabe von 1
THEN OUTPUT(w) ELSE OUTPUT(a)
END.
```

- p_1, p_2: LOOP-berechenbare Umkehrfunktionen der Cantorschen Bijektion $\langle\cdot,\cdot\rangle$ zwischen $\mathbb{N}^2$ und $\mathbb{N}$
- n codiert (bijektiv!) zwei Zahlen k und l
- k codiert (bijektiv!) Worte w

Schwalbenschwanzkonstruktion (dove-tailing)

$\Longrightarrow$ Algorithmus testet

- für alle möglichen Wörter $w \in E^*$ und
- für alle möglichen Laufzeiten l,
- ob M bei Eingabe von w nach l Schritten anhält.

1. Die berechnete Funktion g ist total.
2. Stets gilt $g(n) \in A$.
3. Für jedes $w \in A$ hält M nach l_w Schritten für ein $l_w \in \mathbb{N}$, d.h. $g(n_w) = w$ für n_w mit $p_2(n_w) = l_w$ und $\nu p_1(n_w) = w$.

Damit $A = \{g(n) \mid n \in \mathbb{N}\}$.

Äquivalenzen auf einen Blick

Für $A \subseteq \Sigma^*$ sind folgende Aussagen äquivalent:

- A ist Sprache vom Typ 0.
- A wird von einer nichtdeterministischen Turingmaschine erkannt.
- A wird von einer deterministischen Turingmaschine erkannt.
- Es gibt eine deterministische Turingmaschine, die bei Eingabe eines Wortes $w \in \Sigma^*$ genau dann nach endlich vielen Schritten anhält, wenn $w \in A$ ist.
- A ist semi-entscheidbar, d.h. die eingeschränkte charakteristische Funktion χ'_A ist berechenbar.
- A ist Definitionsbereich einer berechenbaren Funktion.
- A ist rekursiv aufzählbar.
- A ist leer oder Wertebereich einer totalen berechenbaren Funktion.

Äquivalent dazu ist auch

- A ist Wertebereich einer (partiellen) berechenbaren Funktion.

Das spezielle Halteproblem

Sei h wieder die Notation der berechenbaren Wortfunktionen.
M_w sei die durch $w \in \{0,1\}^*$ bezeichnete Turingmaschine, die h_w berechnet.

Das *spezielle Halteproblem* oder *Selbstanwendbarkeitsproblem* ist die Menge

$$\begin{aligned} K &:= \{w \in \{0,1\}^* \mid M_w \text{ angesetzt auf } w \\ &\qquad\qquad \text{hält nach endlich vielen Schritten an}\} \\ &= \{w \in \{0,1\}^* \mid h_w(w) \text{ ist definiert }\} \end{aligned}$$

Satz: Das spezielle Halteproblem K ist rekursiv aufzählbar,
aber nicht entscheidbar.

Das spezielle Halteproblem; der Beweis

K ist rekursiv aufzählbar:

- $g(w\#v) := h_w(v)$ ist berechenbar (utm-Eigenschaft)
- also auch f mit $f(w) := g(w\#w) = h_w(w)$
- $w \in K \Leftrightarrow f(w)$ definiert

Annahme: K sei entscheidbar.

- Dann insbesondere $E^* \setminus K$ semi-entscheidbar
- Sei TM eine Turingmaschine, die genau dann auf w hält, wenn $w \in E^* \setminus K$
- Sei x ein Codewort für TM, d.h.

$$h_x(w) \text{ ist definiert } \Leftrightarrow w \in E^* \setminus K$$

Betrachte $h_x(x)$:

$$\begin{aligned} x \in K \quad &\Leftrightarrow \quad TM \text{ hält nicht auf } x \\ &\Leftrightarrow \quad h_x(x) \text{ nicht definiert} \\ &\Leftrightarrow \quad x \notin K \text{ (Widerspruch...)} \end{aligned}$$

Also war die Annahme falsch, und K ist *nicht* entscheidbar.

Das spezielle Halteproblem

Der Beweis als *Diagonalisierung* jetzt für Zahlfunktionen:

K jetzt Zahlmenge; $i \in K$ gdw. $h_i(i)$ ist definiert.

Übliche Schreibweise: $\uparrow$ für undef. und $\downarrow$ für def..

Betrachte unendliche Tabelle für $h_i(j)$, die z.B. wie folgt aussieht:

$h_i(j)$	$j = 0$	$j = 1$	$j = 2$	$j = 3$	$\dots$
$i = 0$	$\uparrow$	$\downarrow$	$\downarrow$	$\uparrow$	$\dots$
$i = 1$	$\downarrow$	$\downarrow$	$\uparrow$	$\uparrow$	$\dots$
$i = 2$	$\uparrow$	$\downarrow$	$\uparrow$	$\uparrow$	$\dots$
$i = 3$	$\uparrow$	$\uparrow$	$\downarrow$	$\uparrow$	$\dots$

Invertiert man alle Pfeile der Diagonalen, so beschreibt dann der j-te (invertierte) Diagonaleintrag $\chi'_{\mathbb{N}\setminus K}$ (lies $\uparrow = 1$), diese partielle Zahlfunktion hat keine Notation, da sie sich von jedem h_i bei Eingabe von i unterscheidet.

Gibt es andere (zu) schwierige Problem für Rechner?

Seien $A, B \subseteq E^*$ Sprachen.

Dann heißt *A auf B reduzierbar (i.Z.: $A \leq B$)*, wenn es eine totale berechenbare Funktion $f : E^* \to E^*$ gibt, so dass für alle $x \in E^*$ gilt

$$x \in A \iff f(x) \in B$$

- $A \leq B$: Lösung von Problem A durch Lösung von Problem B (daher: A auf B 'reduziert')
- B ist in der Regel 'schwerer' lösbar als A, aber evtl. 'Lösung' für B schon bekannt.
- Beispiel: B Bibliotheksfunktionen, A zu lösende (Programmier)-Aufgabe...

Vom Nutzen des Reduktionsbegriffs

Satz: Sei die Sprache A auf B mittels der Funktion f reduzierbar. Dann gilt:

- Ist B entscheidbar, so ist auch A entscheidbar.
- Ist A nicht entscheidbar, so ist auch B nicht entscheidbar.
- Ist B rekursiv-aufzählbar, so ist auch A rekursiv-aufzählbar.
- Ist A nicht rek.-aufzählbar, so ist auch B nicht rek.-aufzählbar.

Beweis: f sei berechenbar und reduziere A auf B

(a) B sei entscheidbar.
Dann χ_B berechenbar, Komposition $\chi_B \circ f$ berechenbar und

$$\begin{aligned} \chi_A(x) = 1 &\iff x \in A \iff f(x) \in B \iff \chi_B(f(x)) = 1 \\ \chi_A(x) = 0 &\iff x \notin A \iff f(x) \notin B \iff \chi_B(f(x)) = 0 \end{aligned}$$

Also $\chi_A = \chi_B \circ f$ berechenbar, und A entscheidbar

(b) B sei rekursiv-aufzählbar.
Dann: χ'_B berechenbar, Komposition $\chi'_B \circ f$ berechenbar und

$$\begin{aligned} \chi'_A(x) = 1 &\iff x \in A \iff f(x) \in B \iff \chi'_B(f(x)) = 1 \\ \chi'_A(x) = \text{undef.} &\iff x \notin A \iff f(x) \notin B \iff \chi'_B(f(x)) = \text{undef.} \end{aligned}$$

Also $\chi'_A = \chi'_B \circ f$ berechenbar, und A rekursiv-aufzählbar

Das allgemeine Halteproblem ist die Sprache

$$H = \{\ w\$x \in \{0,1\}^*\{\$\}E^* \quad | M_w \text{ angesetzt auf } x \text{ hält an}\}$$

Dabei sei $\$$ irgendein Symbol, das in E nicht vorkommt.

Satz: Das allgemeine Halteproblem ist rek. aufzählbar, aber nicht entscheidbar.

Beweis: H semi-entscheidbar / rekursiv aufzählbar: sofort mit utm-Eigenschaft.
H unentscheidbar: K ist auf H reduzierbar mit folgendem f

$$f(w) := w\$w$$

f ist sicherlich total und berechenbar.

Der Satz von Rice

Satz: Sei R die Klasse aller Turing-berechenbaren Wortfunktionen über dem Alphabet E.
Sei S eine echte, nichttriviale Teilmenge von R, d.h. $\emptyset \neq S \neq R$.
Dann ist die folgende Sprache unentscheidbar:

$$C(S) = \{w \in \{0,1\}^* | \text{ die von } M_w \text{ berechnete Funktion } h_w \text{ liegt in } S\}$$

<u>Beweis</u>: Sei S mit $\emptyset \neq S \neq R$ gegeben,
sei ud die überall undefinierte (berechenbare!) Funktion.
Also $ud \in S$ oder $ud \notin S$.
Wir behandeln diese beiden Fälle getrennt.

Der Satz von Rice

Beweis 1. Fall: Sei $\mathrm{ud} \in S$.

Wegen $S \neq R$ gibt es $q \in R \setminus S$
Sei Q eine Turingmaschine für q.
Betrachte folgende Turingmaschine TM:

> Bei Eingabe von $w\#x$ simuliert TM erst die Maschine M_w angesetzt auf w.
> Kommt diese Rechnung zu einem Ende, so soll TM anschließend Q, angesetzt auf x, simulieren.

g sei die von TM berechnete Funktion.
Mit der smn-Eigenschaft von h gibt es ein totales berechenbares $r : \{0,1\}^* \to \{0,1\}^*$ mit $g(w\#x) = h_{r(w)}(x)$.
Damit gilt:

- Hält M_w auf w, so ist $h_{r(w)} = q$.
- Hält M_w nicht auf w, so ist $h_{r(w)} = \mathrm{ud}$.

Der Satz von Rice

Beweis 1. Fall Forts.

Dann gilt:

$$
\begin{array}{lll}
w \in K & \Rightarrow & \text{Angesetzt auf } w \text{ stoppt } M_w \\
 & \Rightarrow & h_{r(w)} \text{ ist die Funktion } q \\
 & \Rightarrow & h_{r(w)} \text{ liegt nicht in } S \\
 & \Rightarrow & r(w) \notin C(S).
\end{array}
$$

$$
\begin{array}{lll}
w \notin K & \Rightarrow & \text{Angesetzt auf } w \text{ stoppt } M_w \text{ nicht} \\
 & \Rightarrow & h_{r(w)} \text{ ist die Funktion } ud \\
 & \Rightarrow & h_{r(w)} \text{ liegt in } S \\
 & \Rightarrow & r(w) \in C(S).
\end{array}
$$

Also: r reduziert $E^* \setminus K$ auf $C(S)$, d.h $C(S)$ nicht entscheidbar.

Der Satz von Rice

Beweis 2. Fall: Sei $\mathrm{ud} \notin S$.

Analog: Reduktion von K auf $C(S)$, d.h. $C(S)$ nicht entscheidbar.

Einzelheiten zur Übung.

Der Satz von Rice Anwendungsbeispiele:

- $S_1 = \{\text{f berechenbar und total}\}$
  Man kann bei (realen) Programmen i.d.R. nicht entscheiden, ob sie immer halten.
- $S_2 = \{g\}$ für ein gegebenes g
  Man kann bei (realen) Programmen i.d.R. nicht entscheiden,
  ob sie eine exakt vorgegebene Funktion berechnen.
- $S_3 = \{g \mid g(\lambda) = \lambda\}$
  Man kann bei (realen) Programmen i.d.R. nicht entscheiden,
  ob sie eine vorgegebene Spezifikation erfüllen (hier: $g(\lambda) = \lambda$).

Die Beispiele zeigen, dass zum Beispiel Verifikation von Software schwierig ist, sobald die Programmiersprache 'vernünftig' ist (d.h. alle berechenbaren Funktionen umfasst und utm/smn-Eigenschaften hat).

Erinnerung an Grundvorlesung: *Das Postsche Korrespondenzproblem*

- Eingabe: eine endliche Folge von Wortpaaren
$$(x_1, y_1),\ \ (x_2, y_2), \ldots,\ \ (x_k, y_k)$$
wobei $k \geq 1$ und $x_i, y_i \in E^+$ seien, (also $x_i, y_i \neq \lambda$).

- Frage: gibt es eine Folge $\mathcal{I}$ von Indizes i_1,..., i_n aus $\{1, 2, \ldots, k\}$ mit $n \geq 1$ und mit
$$x_{i_1} \ldots x_{i_n} = y_{i_1} \ldots y_{i_n}$$

Eine derartige Folge $\mathcal{I}$ nennt man *Lösung* des Korrespondenzproblems (x_1, y_1), (x_2, y_2),..., (x_k, y_k).

Satz: Das Postsche Korrespondenzproblem PCP ist unentscheidbar.

$01-\mathrm{PCP}$ (PCP, beschränkt auf das Alphabet $\{0, 1\}$)

Satz: $01-\mathrm{PCP}$ ist unentscheidbar.

Zum Beweis zeige, dass PCP auf $01-\mathrm{PCP}$ reduzierbar ist:

Sei $E = \{a_1, \ldots, a_m\}$ das Alphabet des gegebenen PCPs.
Jedem Symbol $a_j \in E$ ordne das Wort $a_j' = 01^j \in \{0, 1\}^*$ zu;
verallgemeinere dies auf beliebige Wörter

$$w = a_1 \ldots a_n \in E^+ \quad \mapsto \quad w' = a_1' \ldots a_n' \in \{0, 1\}^*$$

Dann gilt offensichtlich:

$$\begin{aligned} & (x_1, y_1), \ldots, (x_k, y_k) \text{ hat eine Lösung} \\ \iff \quad & (x_1', y_1'), \ldots, (x_k', y_k') \text{ hat eine Lösung} \end{aligned}$$

Eine formalsprachliche Anwendung

Unter dem *Schnittleerheitsproblem* einer Grammatikfamilie $\mathcal{G}$ versteht man:
Gegeben $G_1, G_2 \in \mathcal{G}$, gilt $L(G_1) \cap L(G_2) = \emptyset$?

Satz: Das Schnittleerheitsproblem für kontextfreie Grammatiken ist unentscheidbar.

Beweis:

Gegeben Postsches Korrespondenzproblem P über $\{0, 1\}$ mit

$$P = \{(x_1, y_1), \ldots, (x_k, y_k)\}$$

Konstruiere zwei kontextfreie Grammatiken $G_1 = (N_1, T, P_1, S)$ und $G_2 = (N_2, T, P_2, S)$ wie folgt:

$$N_1 = \{S, A, B\}, \quad N_2 = \{S, R\}, \quad T = \{0, 1, \$, a_1, \ldots, a_k\}$$

Die Grammatik G_1 besitzt die Ableitungsregeln P_1

$$\begin{array}{rcl} S & \rightarrow & A\$B \\ A & \rightarrow & a_1Ax_1 \mid \dots \mid a_kAx_k \mid a_1x_1 \mid \dots \mid a_kx_k \\ B & \rightarrow & y_1^rBa_1 \mid \dots \mid y_k^rBa_k \mid y_1^ra_1 \mid \dots \mid y_k^ra_k \end{array}$$

wobei mit w^r das gespiegelte Wort zu w bezeichnet wird.

Diese Grammatik G_1 erzeugt die Sprache

$$L_1 = \{a_{i_1} \dots a_{i_n} x_{i_n} \dots x_{i_1} \$ y_{j_1}^r \dots y_{j_m}^r a_{j_m} \dots a_{j_1} \mid n, m \geq 1, i_\nu, j_\mu \in \{1, \dots, k\}\}$$

Wir führen eine zweite Grammatik G_2 mit den folgenden Regeln P_2 ein:

$$\begin{array}{rcl} S & \rightarrow & a_1Sa_1 \mid \dots \mid a_kSa_k \mid R \\ R & \rightarrow & 0R0 \mid 1R1 \mid \$ \end{array}$$

Sie erzeugt die Sprache

$$L_2 = \{uv\$v^ru^r \mid u \in \{a_1, \dots, a_k\}^*, v \in \{0, 1\}^*\}$$

Beide Sprachen sind übrigens sogar deterministisch kontextfrei.

Idee der Konstruktion:

- Jedes Wort w aus L_1 bzw. L_2 hat vier Komponenten: $w = ab\$c^r d^r$ mit $a, d \in \{a_1, ..., a_k\}^*$ und $b, c \in \{0, 1\}^*$
- L_1: Indizes der x_i bei b in a bzw. der y_i bei c in d beim PCP P, aber kein Zusammenhang zwischen a und d oder b und c.
- L_2: Übereinstimmung $a = d$ bzw. $b = c$, aber keinerlei Verbindung mit P
- Schnittbildung: b und c enstehen aus P (wg. L_1), sind gleich ($b = c$) und nutzen die gleichen Indizes ($a = d$)

Damit: P besitzt Lösung $i_1, ..., i_n$ genau dann, wenn $w \in L_1 \cap L_2$ für

$$w = a_{i_n} ... a_{i_1} x_{i_1} ... x_{i_n} \$ y^r_{i_n} ... y^r_{i_1} a_{i_1} ... a_{i_n}$$

Also: Abbildung $f : P \mapsto (G_1, G_2)$ reduziert PCP auf das Komplement des Schnittproblems.

Insgesamt: PCP unentscheidbar
$\Rightarrow$ Komplement des Schnittleerheitsproblems unentscheidbar
$\Rightarrow$ Schnittleerheitsproblem unentscheidbar.

Weitere unentscheidbare Probleme 1

Folgerung: Das Schnittunendlichkeitsproblem für zwei kontextfreie Sprachen (genauer: Grammatiken) ist unentscheidbar.

Verwende gleiche Konstruktion $P \mapsto (G_1, G_2)$ wie soeben:

$$\begin{aligned} P \in PCP \quad &\Leftrightarrow \quad \text{es gibt eine Lösung } \mathcal{I} = (i_1, \ldots, i_n) \text{ für } P \\ &\Leftrightarrow \quad \text{es gibt } \infty\text{-viele Lösungen } \mathcal{I}, \mathcal{II}, \mathcal{III}, \ldots \text{ für } P \\ &\Rightarrow \quad |L_1 \cap L_2| = \infty \\ &\Rightarrow \quad L_1 \cap L_2 \neq \emptyset \quad \Leftrightarrow \quad P \in PCP \end{aligned}$$

Weitere unentscheidbare Probleme 2

Folgerung: Das Schnittkontextfreiheitsproblem für zwei kontextfreie Sprachen (genauer: Grammatiken) ist unentscheidbar.

Wieder gleiche Konstruktion $P \mapsto (G_1, G_2)$,
setze $L := L_1 \cap L_2$

Jedes Wort in L hat Form $ab\$c^r d^r$, mit $a = d$, $b = c$ und 'Index-Übereinstimmung' zwischen a und b bzw. c und d

Also: a ober d bekannt $\Rightarrow ab\$c^r d^r$ eindeutig festgelegt!

Behauptung: Falls $|L| = \infty$, dann L ist nicht kontextfrei.

Annahme: L sei kontextfrei (und unendlich), d.h. Pumping-Lemma für kontextfreie Sprachen sei anwendbar, n sei die Pumpingzahl.

Wähle $ab\$c^r d^r \in L$ mit $n < |a| = |d| \leq |b| = |c|$

Betrachte beliebige Zerlegung $ab\$c^r d^r = uvwxy$ mit
$uwy = uv^0wx^0y \in L$, $|vx| \neq 0$ und $|vwx| \leq n$

Fall (1): $|u| \geq |a|$: damit a Präfix von uwy,
aber b, c, d durch a festgelegt
$\Rightarrow$ Widerspruch zu $|vx| \geq 0$ und $uwy \in L$.

Fall (2): $|u| < |a|$: damit $|uvwx| < |ab|$ und d Suffix von uwy,
aber a, b, c durch d festgelegt
$\Rightarrow$ wieder Widerspruch zu $|vx| \geq 0$ und $uwy \in L$.

Damit also: $|L| = \infty \Rightarrow L$ nicht kontextfrei

Zusammen:

$$\begin{array}{lclcl} P \in PCP & \Rightarrow & |L| = \infty & \Rightarrow & L \text{ nicht kontextfrei} \\ P \notin PCP & \Rightarrow & L = \emptyset & \Rightarrow & L \text{ kontextfrei} \end{array}$$

Weitere unentscheidbare Probleme 3

Folgerung: Das Inklusionsproblem für zwei kontextfreie Sprachen (genauer: Grammatiken) ist unentscheidbar.

Nutze, dass L_1, L_2 deterministisch kontextfrei sind.
Deterministisch kontextfreie Sprachen sind effektiv unter Komplementbildung abgeschlossen, d.h.:
Zu G_1 und G_2 kann man effektiv Grammatiken G_1' und G_2' konstruieren mit $L(G_1') = \text{Komplement}(L_1)$ und $L(G_2') = \text{Komplement}(L_2)$.
Dann folgt

$$L_1 \cap L_2 = \emptyset \quad \Leftrightarrow \quad L_1 \subseteq L(G_2')$$

D.h. $P \mapsto (G_1, G_2')$ reduziert PCP auf das Inklusionsproblem kontextfreier Sprachen.

Weitere unentscheidbare Probleme 4

Folgerung: Das Äquivalenzproblem für zwei kontextfreie Sprachen (genauer: Grammatiken) ist unentscheidbar.

Aus G_1 und G_2' konstruiere G_3 mit $L(G_3) = L_1 \cup L(G_2')$, dann

$$\begin{aligned} L_1 \cap L_2 = \emptyset \quad &\Leftrightarrow \quad L_1 \cup L(G_2') = L(G_2') \\ &\Leftrightarrow \quad L(G_3) = L(G_2') \end{aligned}$$

D.h.: $P \mapsto (G_3, G_2')$ reduziert PCP auf das Äquivalenzproblem kontextfreier Sprachen.