Grundlagen Theoretischer Informatik 2

WiSe 2011/12 in Trier

Henning Fernau
Universität Trier
fernau@uni-trier.de

Grundlagen Theoretischer Informatik 2 Gesamtübersicht

- Organisatorisches; Einführung
- Ersetzungsverfahren: Grammatiken und Automaten
- Rekursionstheorie-Einführung
- Komplexitätstheorie-Einführung

Vergleichbarkeit von Notationen

Eine Notation h'' ist in eine Notation h' *übersetzbar*, wenn es eine totale(!) berechenbare Wortfunktion (*Übersetzungsfunktion*) $u : \{0,1\}^* \to \{0,1\}^*$ gibt mit $h''_w(x) = h'_{u(w)}(x)$ für alle $w \in \{0,1\}^*$ und $x \in E^*$, d.h. $h''_w = h'_{u(w)}$.
Jedes 'Programm' w der 'Programmiersprache' h'' kann also dann automatisch in ein 'Programm' der 'Sprache' h' übersetzt werden.
Notationen h' und h'' heißen *äquivalent*, wenn man jede in die andere übersetzen kann.

Satz: (Äquivalenzsatz von Rogers) Eine Notation h' für die berechenbaren Wortfunktionen ist genau dann zu der von uns definierten Notation h äquivalent, wenn sie die utm-Eigenschaft und die smn-Eigenschaft hat.

Beweisrichtung $\Rightarrow$:

Sei h wie oben und sei h' weitere äquivalente Notation für berechenbaren Wortfunktionen.

(a) utm-Eigenschaft für h: f_h mit $f_h(w\#x) := h_w(x)$ ist berechenbar.
Definiere f' durch $f'(w\#x) := f_h(u'(w)\#x)$
$\rightsquigarrow f'$ berechenbar und

$$h'_w(x) = h_{u'(w)}(x) = f_h(u'(w)\#x) = f'(w\#x)$$

D.h., f' zeigt utm-Eigenschaft für h'.
(b) Sei $g : E^* \multimap E^*$ irgendeine berechenbare Wortfunktion.
smn-Eigenschaft für h: r mit $h_{r(x)}(y) := g(x\#y)$ ist berechenbar
Dann ist $u \circ r$ berechenbar (und total) mit

$$g(x\#y) = h_{r(x)}(y) = h'_{u\circ r(x)}(y)$$

D.h.: h' hat auch die smn-Eigenschaft.

Beweisrichtung $\Leftarrow$:

h' besitze ebenfalls die utm- und die smn-Eigenschaft:

utm-Eigenschaft für h: f_h mit $f_h(w\#x) = h_w(x)$ berechenbar

Wende smn-Eigenschaft für h' auf f_h an, mit entsprechendem r':

$$h_w(x) = f_h(w\#x) = h'_{r'(w)}(x)$$

Damit Übersetzung von h nach h' mit r'

Analog: utm-E. für h' und smn-E. für $h \Rightarrow h'$ in h übersetzbar.

Damit: h und h' äquivalent!

Ein Ersatz für smn

Satz: Eine Notation h' für die berechenbaren Wortfunktionen ist genau dann zu der von uns definierten Notation h äquivalent, wenn sie die utm-Eigenschaft und die folgende Eigenschaft (*effektive Komposition*) hat:

> Es gibt eine totale berechenbare Wortfunktion $r : \{0,1\}^*\#\{0,1\}^* \to \{0,1\}^*$ derart, dass für alle v, w aus $\{0,1\}^*$ und alle x aus E^* gilt:
>
> $$h'_v(h'_w(x)) = h'_{r(v\#w)}(x)$$

r bestimmt also aus zwei 'Programmen' v, w ein 'Programm' $r(v\#w)$ für die Komposition der Funktionen h'_v und h'_w.

(Ohne Beweis.)

- Alle 'vernünftigen' Notationen der berechenbaren Wortfunktionen sind äquivalent (mit 'vernünftig' = 'utm+smn' = 'utm+effKomp')
- Jede zu h äquivalente Notation heißt *Standardnotation*.

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 **B**ibliotheksfunktionen, A zu lösende (Programmier)-**A**ufgabe...

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

$$\chi_A(x) = 1 \iff x \in A \iff f(x) \in B \iff \chi_B(f(x)) = 1$$
$$\chi_A(x) = 0 \iff x \notin A \iff f(x) \notin B \iff \chi_B(f(x)) = 0$$

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

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

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:

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

Beweis: 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 $ud \in S$.

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 $h_{r(w)} = ud$.

Der Satz von Rice

Beweis 1. Fall Forts.

Dann gilt:

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

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

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:

- $S_1 = \{f \text{ berechenbar und total } \}$
  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).

Das Postsche Korrespondenzproblem

- Eingabe: eine endliche Folge von Wortpaaren

$$(x_1, y_1), \quad (x_2, y_2), ..., \quad (x_k, y_k)$$

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 $\{1, 2, ..., k\}$ mit $n \geq 1$ und mit

$$x_{i_1} ... x_{i_n} = y_{i_1} ... y_{i_n} \ ?$$

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

Das Postsche Korrespondenzproblem

Ein einfaches Beispiel

i	x_i	y_i
1	bab	a
2	ab	abb
3	a	ba

Bei der Indexfolge $\mathcal{I} = (2, 1, 3)$ ergibt sich $ab\ bab\ a = abb\ a\ ba$.

Das Postsche Korrespondenzproblem

Ein nicht so einfaches Beispiel. . .

i	x_i	y_i
1	001	0
2	01	011
3	01	101
4	10	001

Kürzeste Lösung:

01|1 0|01|10|1 0|01|001|01|1 0|01|1 0| ...001|001|01
01 1|0 01|10 1|0 01|001|01 1|0|01 1|0 01|101|001|...0|0|101

mit 66 Indizes:

$$2, 4, 3, 4, 4, 2, 1, 2, 4, 3, 4, 3, 4, 4, 3, 4, 4, 2, 1, 4, 4, 2, 1, 3, 4, 1, 1, 3,$$
$$4, 4, 4, 2, 1, 2, 1, 1, 1, 3, 4, 3, 4, 1, 2, 1, 4, 4, 2, 1, 4, 1, 1, 3, 4, 1, 1, 3,$$
$$1, 1, 3, 1, 2, 1, 4, 1, 1, 3.$$

Das Postsche Korrespondenzproblem

Eine leichte Hilfsaussage

PCP ist semi-entscheidbar (durch einfache Suche nach einer Lösung):

- Eingabe sei (x_1, y_1), (x_2, y_2),..., (x_k, y_k)

- Für alle Längen $n = 1, 2, 3, ...$
  und alle Indexfolgen $i_1, i_2, ..., i_n$ der Länge n:
  teste, ob $x_{i_1}...x_{i_n} \stackrel{?}{=} y_{i_1}...y_{i_n}$
  Wenn ja, akzeptiere die Eingabe!

Das Postsche Korrespondenzproblem PCP ist unentscheidbar!

Genauer: Die Menge

$$\begin{aligned} \mathrm{PCP} \;=\; \{((x_1,y_1),(x_2,y_2),\ldots,(x_k,y_k)) \mid \\ k \in \mathbb{N}, x_i, y_i \in E^+ \text{ für } i = 1,\ldots,k, \text{ und} \\ ((x_1,y_1),(x_2,y_2),\ldots,(x_k,y_k)) \text{ hat eine Lösung}\} \end{aligned}$$

ist nicht entscheidbar, mit Beweisweg:

(1) Selbstanwendbarkeitsproblem K ist reduzierbar auf MPCP ('modifiziertes PCP')

(2) MPCP ist reduzierbar auf PCP.

Das Hilfsproblem MPCP

- Eingabe: eine endliche Folge von Wortpaaren

$$(x_1, y_1),\ \ (x_2, y_2), ...,\ \ (x_k, y_k)$$

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 $\{1, 2, ..., k\}$ mit $n \geq 0$ und mit

$$x_1 x_{i_1} ... x_{i_n} = y_1 y_{i_1} ... y_{i_n} \ ?$$

Im Gegensatz zum PCP wird also der erste Index vorgegeben.

MPCP ist reduzierbar auf PCP

Verwende neue Symbole $\$$ und $\#$, die im Alphabet E des MPCPs nicht vorkommen.

Für ein Wort $w = a_1 a_2 ... a_m$ aus E^+ sei

$$\begin{aligned} w^a &:= \#a_1\#a_2\#...\#a_m\# \\ w^b &:= \quad a_1\#a_2\#...\#a_m\# \\ w^c &:= \#a_1\#a_2\#...\#a_m \end{aligned}$$

Definiere Reduktion f von MPCP auf PCP wie folgt:

$$\begin{array}{lrcl} \text{zu} & P & = & ((x_1,y_1),(x_2,y_2),...,(x_k,y_k)) \\ \text{setze} & f(P) & := & ((x_1^a,y_1^c),(x_1^b,y_1^c),(x_2^b,y_2^c),...,(x_k^b,y_k^c),(\$,\#\$)) \end{array}$$

MPCP ist reduzierbar auf PCP (Details)

- f ist offensichtlich berechenbar

- Zeige:
$$P \text{ besitzt eine Lösung mit } i_1 = 1, \text{ d.h. } P \in MPCP$$
$$\iff \quad f(P) \text{ besitzt (irgend)eine Lösung, d.h. } f(P) \in PCP$$

Beweis von '$\Longrightarrow$':
P besitze eine Lösung $i_1, i_2, ..., i_n$ mit $i_1 = 1$ (MPCP!).

Dann ist $(1, i_2+1, ..., i_n+1, k+2)$ eine Lösung für $f(P)$.

Beweis von '$\Longleftarrow$':
$f(P)$ besitze eine Lösung $i_1, i_2, ..., i_n$ aus $\{1, ..., k+2\}$.

- Aufbau der Wortpaare in $f(P)$: nur $i_1 = 1$ und $i_n = k+2$ möglich
- Falls $i_m = k+2$ für ein $m < n$: $i_1, i_2, ..., i_m$ ist ebenfalls Lösung...
- Daher o.B.d.A.: $i_m \neq k+2$ für $m < n$,
  dann auch $i_m \neq 1$ für $m < n$
- Dann ist $(1, i_2-1, ..., i_{n-1}-1)$ eine Lösung für P (d.h. bei MPCP)

K ist reduzierbar auf $MPCP$.

Gesucht: totale berechenbare Funktion f mit $f(w) = P$, wobei
w eine Turingmaschine M_w und ein Eingabewort w für M_w darstellt, und
$P = ((x_1, y_1), (x_2, y_2), ..., (x_k, y_k))$ eine Probleminstanz passend zum $MPCP$ ist,
so dass gilt:

$(*)$ M_w angesetzt auf w hält nach endlich vielen Schritten an,
$\Longleftrightarrow$
$P = ((x_1, y_1), (x_2, y_2), ..., (x_k, y_k))$ besitzt Lösung (mit $i_1 = 1$).

Sei w gegeben und $M_w = (S, E, A, \delta, s_0, \square, F)$.

- Nutze $A \cup S \cup \{\#\}$ als Alphabet für das $MPCP$
- Erstes Wortpaar sei $(\#, \#\square s_0 w \square \#)$

$\Rightarrow$ jede Lösung muss mit diesem Paar beginnen!

K ist reduzierbar auf $MPCP$. (Forts.) Die weiteren Paare werden wie folgt konstruiert:

1. *Kopierregeln*:

(a, a) für alle $a \in A \cup \{\#\}$

2. *Überführungsregeln*:

Für alle Übergänge der Turingmaschine M_w:

$(sa, s'c)$	falls $\delta(s, a) = (s', c, N)$
(sa, cs')	falls $\delta(s, a) = (s', c, R)$
$(bsa, s'bc)$	falls $\delta(s, a) = (s', c, L)$, für alle b aus A
$(\#sa, \#s'\Box c)$	falls $\delta(s, a) = (s', c, L)$
$(s\#, s'c\#)$	falls $\delta(s, \Box) = (s', c, N)$
$(s\#, cs'\#)$	falls $\delta(s, \Box) = (s', c, R)$
$(bs\#, s'bc\#)$	falls $\delta(s, \Box) = (s', c, L)$, für alle b aus A

3. *Löschregeln*:

(as_f, s_f) und $(s_f a, s_f)$ für alle $a \in A$ und $s_f \in F$.

4. *Abschlussregeln*:

$(s_f\#\#, \#)$ für alle $s_f \in F$.

K ist reduzierbar auf $MPCP$. (Forts.)

- $f : w \mapsto P$ mit o.a. Paaren P ist berechenbar
- Zeige noch: Bedingung $(*)$ ist erfüllt.

'$\Longrightarrow$', M_w stoppt bei Eingabe w:
Dann gibt es eine Folge von Konfigurationen $(k_0, k_1, ..., k_t)$ mit

- $k_0 = \square s_0 w \square$ (Zusätzliche $\square$ stören nicht...)
- k_t ist Endkonfiguration (also $k_t = u s_f v$ mit $u, v \in A^*, s_f \in F$)
- $k_i \vdash k_{i+1}$ für $i = 0, ..., t-1$.

P besitzt dann eine Lösung für das $MPCP$ der Form:

$$\begin{array}{lllllllll} x: & \# & k_0\# & k_1\# & k_2\# & ... & k_t\# & k_t'\# & k_t''\# & ... & s_f\#\# \\ y: & \#k_0\# & k_1\# & k_2\# & ... & k_t\# & k_t'\# & k_t''\# & ... & s_f\# & \# \end{array}$$

- Folge $x_{i_1}...x_{i_j}$ liegt eine Konfiguration hinter den $y_{i_1}...y_{i_j}$ zurück.
- $k_t', k_t'', ...$ entstehen aus $k_t = u s_f v$ durch Löschen um s_f herum

K ist reduzierbar auf $MPCP$. (Forts.)

'$\Longleftarrow$', P hat eine Lösung (mit $i_1 = 1$):

- Lösungswort ist durch # in Konfigurationen $k_0, k_1, ...k_t$ unterteilt
- Wegen Start mit $i_1 = 1$: $k_0 = \square s_0 w \square$
- Nach Konstruktion: $k_i \vdash k_{i+1}$ oder k_i ist Endkonfiguration

Also hält M_w auf Eingabe w.

Insgesamt:

- K ist auf $MPCP$ reduzierbar
- K ist nicht entscheidbar, also auch $MPCP$ nicht entscheidbar
- $MPCP$ ist auf PCP reduzierbar, also auch PCP nicht entscheidbar

Satz: Das Postsche Korrespondenzproblem PCP ist unentscheidbar.

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

Satz: $01-PCP$ ist unentscheidbar.

Zum Beweis zeige, dass PCP auf $01-PCP$ reduzierbar ist:

Sei $E = \{a_1, ..., a_m\}$ 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

$$w = a_1...a_n \in E^+ \quad \mapsto \quad w' = a'_1...a'_n \in \{0,1\}^*$$

Dann gilt offensichtlich:

$$\begin{array}{ll} & (x_1, y_1), ..., (x_k, y_k) \text{ hat eine Lösung} \\ \iff & (x'_1, y'_1), ..., (x'_k, y'_k) \text{ hat eine Lösung} \end{array}$$

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

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

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

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

Die Grammatik G_1 besitzt die Ableitungsregeln P_1

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

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

Diese Grammatik G_1 erzeugt die Sprache

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

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

$$\begin{aligned} S &\rightarrow a_1Sa_1 \mid ... \mid a_kSa_k \mid R \\ R &\rightarrow 0R0 \mid 1R1 \mid \$ \end{aligned}$$

Sie erzeugt die Sprache

$$L_2 = \{u\nu\$\nu^ru^r \mid u \in \{a_1, ..., a_k\}^*, \nu \in \{0, 1\}^*\}$$

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

$$w = a_{i_n} ... a_{i_1} x_{i_1} ... x_{i_n} \$ y_{i_n}^r ... y_{i_1}^r a_{i_1} ... a_{i_n}$$

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.