Grundbegriffe der Theoretischen Informatik

Sommersemester 2018 - Thomas Schwentick

Teil C: Berechenbarkeit und Entscheidbarkeit

15: Unentscheidbare Probleme 2

Version von: 14. Juni 2018 (17:29)

Einleitung

- Wie gesagt: es ist möglich, automatisch zu überprüfen, ob die Implementierung einer regulären Sprache (durch einen DFA) korrekt ist bezüglich der Spezifikation (durch einen regulären Ausdruck) ☞ Teil A

- Es ist auch möglich,
  - eine „optimale“ Implementierung aus einer Spezifikation zu konstruieren
  - zu testen, ob zwei Spezifikationen äquivalent sind
  - zu testen, ob zwei Implementierungen äquivalent sind

- Natürlich wäre es schön, wenn dies alles auch allgemeiner möglich wäre:
  - für Programme/Algorithmen statt DFAs und
  - für allgemeine semantische Spezifikationen statt reguläre Ausdrücke

- In diesem Kapitel werden wir sehen:
  - Für allgemeine Programme und Spezifikationen ist **nichts** von dem möglich
  - Semantische Eigenschaften von Programmen sind nicht automatisch überprüfbar
    - Das ist unterm Strich die Aussage des *Satzes von Rice*

- Wir werden sogar zeigen, dass die Unmöglichkeit von Äquivalenztests schon für kontextfreie Sprachen und ihre Beschreibungsformen gilt

Komplemente entscheidbarer Sprachen

- Wie verwenden in diesem Kapitel mehrfach die folgende einfache Einsicht

Lemma 15.1

(a) Ist eine Sprache L entscheidbar, dann ist auch ihr Komplement $\overline{L}$ entscheidbar

(b) Ist das Komplement $\overline{L}$ einer Sprache L unentscheidbar, dann ist auch L unentscheidbar

Beweis

(a) Dazu genügt es in einer TM, die L entscheidet bei allen Transitionen ja durch nein und nein durch ja zu ersetzen

(b) Durch Kontraposition von (a)

Inhalt

▷ **15.1 Der Satz von Rice**

15.2 Das Postsche Korrespondenzproblem

15.3 Unentscheidbare Grammatikprobleme

Satz von Rice (1/3)

- Wir zeigen jetzt, dass jede nicht-triviale semantische Eigenschaft von Algorithmen (Programmen, Turingmaschinen) unentscheidbar ist

- Wir betrachten dazu zunächst Turingmaschinen, die Funktionen berechnen
- Semantische Eigenschaften formalisieren wir duch Mengen von Funktionen

- Für jede Menge $S \subseteq \mathcal{R}$ berechenbarer Funktionen definieren wir das folgende algorithmische Problem

Definition (Definition: TM-FUNC(S))

Gegeben: Turingmaschine M

Frage: Ist $f_M \in S$?

Satz 15.2 [Rice 53]

- Sei S eine Menge berechenbarer partieller Funktionen mit $\varnothing \neq S \neq \mathcal{R}$
- Dann ist TM-FUNC(S) unentscheidbar

- Wählt man beispielsweise S als Menge aller totalen, berechenbaren Funktionen, so folgt mit dem Satz von Rice, dass es unentscheidbar ist, ob eine gegebene TM eine totale Funktion berechnet
  - Das ist natürlich nicht sehr überraschend, aber wir werden gleich noch weitere Anwendungen betrachten

Satz von Rice: Anwendungen

- Aus dem Satz von Rice lassen sich viele Unentscheidbarkeitsresultate folgern, beispielsweise:
- Für jede feste berechenbare Funktion f ist es unentscheidbar, ob ein gegebenes Programm f berechnet:
  - Wähle $S = \{f\}$
- Für beliebige feste Strings u, w ist es unentscheidbar, ob ein gegebenes Programm bei Eingabe u die Ausgabe w hat:
  - Wähle $S = \{f \mid f(u) = w\}$
- Es ist unentscheidbar, ob zwei gegebene Programme äquivalent sind:
  - Das folgt direkt aus der Unentscheidbarkeit des ersten Problems
  - Dieses lässt sich nämlich auf die Äquivalenz zweier Programme reduzieren
- Auf ähnliche Weise folgt die Unentscheidbarkeit der in der Einleitung genannten Probleme
- Dass wir hier „Programm“ statt „TM“ schreiben, ist durch die (empirische Gültigkeit der) Church-Turing-These gerechtfertigt

Satz von Rice (2/3)

Beweisskizze

- Sei $f_\perp(w) \overset{\text{def}}{=} \perp$, für alle $w \in \Sigma^*$

- 1. Fall: $f_\perp \notin S$:
  - Sei $f \in S$ beliebig, M_f TM für f

- Wir zeigen: TM-E-HALT $\leqslant$ TM-FUNC(S)
- Genauer wollen wir M so auf M' abbbilden, dass gilt:
  - $M(\epsilon)$ terminiert $\Rightarrow$ M' berechnet f
  - $M(\epsilon)$ terminiert nicht $\Rightarrow$ M' berechnet $f_\perp$

- Für jede TM M sei dazu M' die TM, die bei Eingabe x
  (1) zuerst M bei Eingabe ϵ simuliert (allerdings „hinter x“ und ohne x zu verändern)
  (2) und falls diese Berechnung terminiert, die TM M_f bei Eingabe x simuliert

✎ Ein solches M' kann z.B. mit Satz 13.3 aus einer geeigneten 2-TM gewonnen werden

Beweisskizze (Forts.)

- Wir zeigen, dass $M \mapsto M'$ eine Reduktion von TM-E-HALT auf TM-FUNC(S) ist

- Sei $M \in$ TM-E-HALT

➨ $M(\epsilon)$ terminiert

➨ Phase (1) von M' terminiert

➨ Phase (2) simuliert das Verhalten von M_f bei Eingabe x und gibt also $f(x)$ aus ☞ falls das definiert ist

➨ M' berechnet f

➨ $M' \in$ TM-FUNC(S)

- Sei nun $M \notin$ TM-E-HALT

➨ $M(\epsilon)$ terminiert nicht

➨ M' berechnet $f_\perp$

➨ $M' \notin$ TM-FUNC(S)

Satz von Rice (3/3)

Beweisskizze (Forts.)

- Im zweiten Fall ($f_\perp \in S$) lässt sich analog zeigen, dass TM-E-HALT auf das Komplement $\overline{\text{TM-FUNC}(S)}$ reduzierbar ist

➨ $\overline{\text{TM-FUNC}(S)}$ ist unentscheidbar

➨ $\text{TM-FUNC}(S)$ ist unentscheidbar ☞ Lemma 15.1

Inhalt

15.1 Der Satz von Rice

▷ **15.2 Das Postsche Korrespondenzproblem**

15.3 Unentscheidbare Grammatikprobleme

PCP **ist unentscheidbar (1/7)**

- Zur Erinnerung:

Definition (PCP)

Gegeben: eine Folge $(u_1, v_1), \ldots, (u_k, v_k)$ von Paaren nicht-leerer Strings

Frage: Gibt es eine Indexfolge $i_1, \ldots, i_n$ mit $n \geqslant 1$, so dass $u_{i_1} u_{i_2} \cdots u_{i_n} = v_{i_1} v_{i_2} \cdots v_{i_n}$?

Satz 15.3

- PCP ist unentscheidbar

Beweisidee

- Wir definieren ein „Zwischen-Problem“ SPCP und zeigen:
  - TM-HALT $\leqslant$ SPCP und
  - SPCP $\leqslant$ PCP
- Daraus folgt mit Lemma 14.3, dass SPCP und dann auch PCP unentscheidbar sind

Definition (PCP-Problem mit Startpaar (SPCP))

Gegeben: eine Folge $(u_1, v_1), \ldots, (u_k, v_k)$ von Paaren von Strings

Frage: Gibt es eine Indexfolge $i_1, \ldots, i_n$ mit $n \geqslant 1$, so dass
- $u_{i_1} u_{i_2} \cdots u_{i_n} = v_{i_1} v_{i_2} \cdots v_{i_n}$
- und $i_1 = 1$?

Die erste Beispiel-TM (leicht modifiziert)

Beispiel

Turing-Maschine zum Test, ob Eingabe von der Form ww^R ist

a: 0 und 1 überlesen, nach links — falls $\rhd$ oder $\sqcup$ nach rechts in b

b: Falls 0 nach rechts in c — falls 1 nach rechts in d ($0/1$ durch $\sqcup$ überschreiben) — falls $\sqcup$ akz.

c: 0 und 1 überlesen nach rechts bis $\sqcup$, dann nach links in e

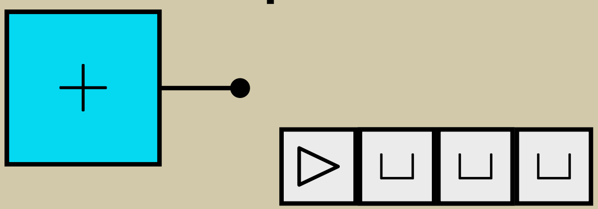
d: 0 und 1 überlesen nach rechts bis $\sqcup$, dann nach links in f

e: Falls 0 durch $\sqcup$ ersetzen nach links in a — falls 1 oder $\sqcup$ ablehnen

f: Falls 1 durch $\sqcup$ ersetzen nach links in a — falls 0 oder $\sqcup$ ablehnen

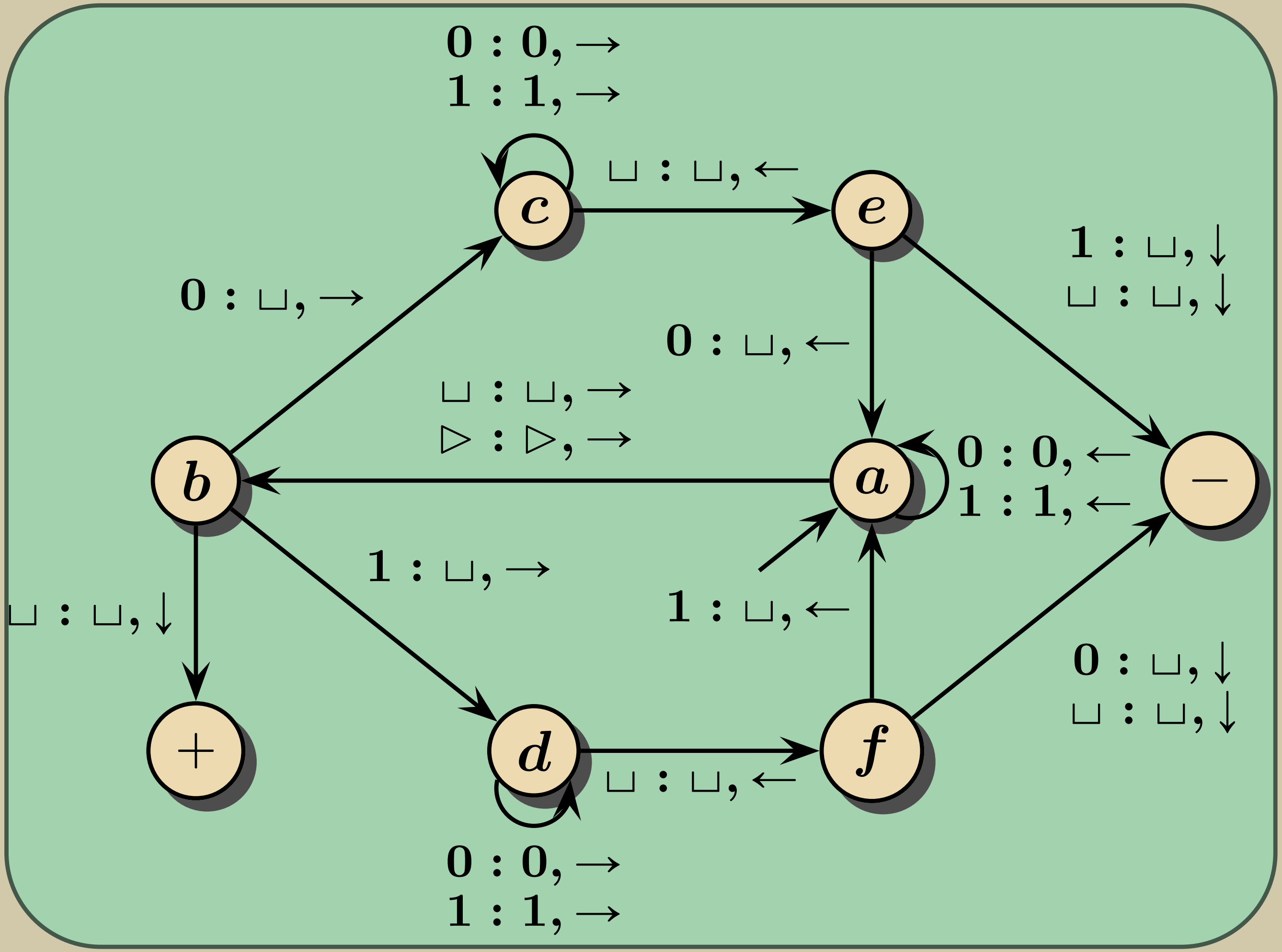
✎ Die TM nutzt $\sqcup$ statt $\#$, da wir $\#$ für die Konkatenation verwenden wollen

Beispielberechnung:



Beispiel-TM als Diagramm

Beispiel-Turingmaschine als Diagramm



Beispielberechnung als Präfix einer Lösung

Beispiel (Forts.)

- Terminierende Berechnung von M bei Eingabe 11 als Präfix einer SPCP-Lösung
- Präfix des Lösungsstrings:

$$\left[\frac{\#}{\#a\triangleright 11\#}\right]\left[\frac{a\triangleright}{\triangleright b}\right]\left[\frac{1}{1}\right]\left[\frac{1}{1}\right]\left[\frac{\#}{\#}\right]\ \left[\frac{\triangleright}{\triangleright}\right]\left[\frac{b1}{\sqcup d}\right]\left[\frac{1}{1}\right]\left[\frac{\#}{\#}\right]\ \left[\frac{\triangleright}{\triangleright}\right]\left[\frac{\sqcup}{\sqcup}\right]\left[\frac{d1}{1d}\right]\left[\frac{\#}{\#}\right]$$

$$\left[\frac{\triangleright}{\triangleright}\right]\left[\frac{\sqcup}{\sqcup}\right]\left[\frac{1d\#}{f1\sqcup\#}\right]\left[\frac{\triangleright}{\triangleright}\right]\left[\frac{\sqcup f1}{a\sqcup\sqcup}\right]\left[\frac{\sqcup}{\sqcup}\right]\left[\frac{\#}{\#}\right]$$

$$\left[\frac{\triangleright}{\triangleright}\right]\left[\frac{a\sqcup}{\sqcup b}\right]\left[\frac{\sqcup}{\sqcup}\right]\left[\frac{\sqcup}{\sqcup}\right]\left[\frac{\#}{\#}\right]\ \left[\frac{\triangleright}{\triangleright}\right]\left[\frac{\sqcup}{\sqcup}\right]\left[\frac{b\sqcup}{\text{ja}\sqcup}\right]\left[\frac{\sqcup}{\sqcup}\right]\left[\frac{\#}{\#}\right]$$

- Verwendete Steintypen:

$$\left[\frac{\#}{\#a\triangleright 11\#}\right], \left[\frac{a\triangleright}{\triangleright b}\right], \left[\frac{1}{1}\right], \left[\frac{\#}{\#}\right], \left[\frac{\triangleright}{\triangleright}\right], \left[\frac{b1}{\sqcup d}\right], \left[\frac{\sqcup}{\sqcup}\right], \left[\frac{d1}{1d}\right],$$

$$\left[\frac{1d\#}{f1\sqcup\#}\right], \left[\frac{\sqcup f1}{a\sqcup\sqcup}\right], \left[\frac{a\sqcup}{\sqcup b}\right], \left[\frac{b\sqcup}{\text{ja}\sqcup}\right]$$

PCP **ist unentscheidbar (2/7)**

Beweisskizze (Forts.)

- Wir konstruieren eine Reduktionsfunktion f mit:
  $(M, x) \in$ TM-HALT $\Longleftrightarrow$
  $f((M, x)) \in$ SPCP
- Lösungen von $f((M, x))$ entsprechen dabei terminierenden Berechnungen $M(x)$

- Wir kodieren dazu Konfigurationen $K = (q, (u, \sigma, v))$ von M durch Strings $\hat{K} \overset{\text{def}}{=} uq\sigma v$

- Idee der Reduktion:
- Sei $M(x) = K_0 \vdash_M \cdots \vdash_M K_t$
  ☞ $K_0 \overset{\text{def}}{=} K_0(x)$
- Lösungsstrings für $f((M, x))$ beginnen mit der Konkatenation der Konfigurationen dieser Berechnung:
  $\hat{K}_0\#\hat{K}_1\#\cdots\#\hat{K}_t$
  ☞ über das Ende reden wir später

Beweisskizze (Forts.)

- Für jedes $\ell < t$
  - ist $\hat{K}_0\#\hat{K}_1\#\cdots\#\hat{K}_\ell$ ein Präfix $u_{i_1}\cdots u_{i_k}$ des Lösungsstrings
  - und $v_{i_1}\cdots v_{i_k}$ ist $\hat{K}_0\#\hat{K}_1\#\cdots\#\hat{K}_{\ell+1}$
- Also übereinander geschrieben:

$\#$	$\hat{K}_0\#$	$\hat{K}_1\#$	$\cdots$	$\hat{K}_\ell\#$
$\#\hat{K}_0\#$	$\hat{K}_1\#$	$\hat{K}_2\#$	$\cdots$	$\hat{K}_{\ell+1}\#$

- Es gilt dann für jedes i:
  wenn u_i eine Position in einer Konfiguration K_j repräsentiert, dann repräsentiert v_i dieselbe Position in K_{j+1}
- Dieses „Übereinanderlegen“ aufeinanderfolgender Konfigurationen ermöglicht es, die Konfigurationsfolge auf Korrektheit bezüglich der Transitionsfunktion von M zu testen

PCP **ist unentscheidbar (3/7)**

Beweisskizze (Forts.)

- Sei $M = (Q, \Gamma, \delta, s)$ TM, $x \in \Sigma^*$
  - OBdA: $\# \notin \Gamma$
- Die zugehörige SPCP-Eingabe hat das Alphabet $Q \cup \Gamma \cup \{\#\}$ und die folgenden Regeln:

(1) Start: $\left[\frac{\#}{\# s \triangleright x \#}\right]$

(2) Kopierregeln: $\left[\frac{\sigma}{\sigma}\right]$, für jedes $\sigma \in \Gamma \cup \{\#\}$

(3) δ-Regeln: für alle $q, q' \in Q$ und alle $\sigma, \sigma', \tau \in \Gamma \cup \{\#\}$:

- falls $\delta(q, \sigma) = (q', \sigma', \downarrow)$: $\left[\frac{q\sigma}{q'\sigma'}\right]$,
- falls $\delta(q, \sigma) = (q', \sigma', \rightarrow)$: $\left[\frac{q\sigma}{\sigma' q'}\right]$
- falls $\delta(q, \sigma) = (q', \sigma', \leftarrow)$: $\left[\frac{\tau q \sigma}{q' \tau \sigma'}\right]$, für alle $\tau \in \Gamma$,

Beweisskizze (Forts.)

(4) δ-Regeln für den „rechten Rand“:

Für alle $q, q' \in Q$ und alle $\sigma', \tau \in \Gamma \cup \{\#\}$:

- falls $\delta(q, \sqcup) = (q', \sigma', \downarrow)$: $\left[\frac{q\#}{q'\sigma'\#}\right]$,
- falls $\delta(q, \sqcup) = (q', \sigma', \rightarrow)$: $\left[\frac{q\#}{\sigma' q'\#}\right]$
- falls $\delta(q, \sqcup) = (q', \sigma', \leftarrow)$: $\left[\frac{\tau q\#}{q'\tau\sigma'\#}\right]$, für alle $\tau \in \Gamma$,

PCP ist unentscheidbar (4/7)

- Wie gesagt: der erste Teil des Lösungsstrings ist die Konkatenation der Kodierungen aller Konfigurationen der terminierenden Berechnung:

$$\begin{array}{lllll} \# & \hat{K}_0\# & \hat{K}_1\# & \cdots & \hat{K}_{t-1}\# \\ \#\hat{K}_0\# & \hat{K}_1\# & \hat{K}_2\# & \cdots & \hat{K}_t\# \end{array}$$

- Dies ist aber noch kein Lösungsstring, da der „u-String“ nur ein Präfix des v-Strings ist
  - Der v-String ist genau um $\hat{K}_t\#$ länger
  - Da die Konfiguration K_t (im Gegensatz zu K_0) nicht bekannt ist, kann diese Lücke nicht durch eine einzelne Regel der SPCP-Eingabe überbrückt werden

- Stattdessen verwenden wir zusätzliche Löschregeln der Arten $\begin{array}{|c|}\hline \sigma\text{ja} \\ \hline \text{ja} \\ \hline \end{array}$ und $\begin{array}{|c|}\hline \text{ja}\sigma \\ \hline \text{ja} \\ \hline \end{array}$, durch deren Anwendung immer kürzere Teilstrings von $\hat{K}_t$ entstehen, bis der v-String nur noch zwei Zeichen länger ist als der u-String
- Sei dazu $C_0 \stackrel{\text{def}}{=} \hat{K}_t$ und entstehe C_{i+1} aus C_i jeweils durch Löschen *eines* Nachbarzeichen des Zustandssymboles (ja, nein, oder h), und sei $C_c \in \{\text{ja}, \text{nein}, h\}$
- Insgesamt ergibt sich dann folgende Korrespondenz:

$$\begin{array}{llllllllll} \# & \hat{K}_0\# & \hat{K}_1\# & \cdots & \hat{K}_{t-1}\# & \hat{K}_t\# & C_1\# & \cdots & C_{c-1}\# & C_c\#\# & = \\ \#\hat{K}_0\# & \hat{K}_1\# & \hat{K}_2\# & \cdots & \hat{K}_t\# & C_1\# & C_2\# & \cdots & C_c\# & \# & \end{array}$$

PCP-Lösungsstring für die Beispielberechnung

Beispiel (Forts.)

- Terminierende Berechnung von M bei Eingabe 11 als SPCP-Lösung
- Lösungsstring:

$$\begin{bmatrix}\#\\ \#a\triangleright 11\#\end{bmatrix}\begin{bmatrix}a\triangleright\\ \triangleright b\end{bmatrix}\begin{bmatrix}1\\ 1\end{bmatrix}\begin{bmatrix}1\\ 1\end{bmatrix}\begin{bmatrix}\#\\ \#\end{bmatrix}\ \begin{bmatrix}\triangleright\\ \triangleright\end{bmatrix}\begin{bmatrix}b1\\ \sqcup d\end{bmatrix}\begin{bmatrix}1\\ 1\end{bmatrix}\begin{bmatrix}\#\\ \#\end{bmatrix}\ \begin{bmatrix}\triangleright\\ \triangleright\end{bmatrix}\begin{bmatrix}\sqcup\\ \sqcup\end{bmatrix}\begin{bmatrix}d1\\ 1d\end{bmatrix}\begin{bmatrix}\#\\ \#\end{bmatrix}$$

$$\begin{bmatrix}\triangleright\\ \triangleright\end{bmatrix}\begin{bmatrix}\sqcup\\ \sqcup\end{bmatrix}\begin{bmatrix}1d\#\\ f1\sqcup\#\end{bmatrix}\begin{bmatrix}\triangleright\\ \triangleright\end{bmatrix}\begin{bmatrix}\sqcup f1\\ a\sqcup\sqcup\end{bmatrix}\begin{bmatrix}\sqcup\\ \sqcup\end{bmatrix}\begin{bmatrix}\#\\ \#\end{bmatrix}$$

$$\begin{bmatrix}\triangleright\\ \triangleright\end{bmatrix}\begin{bmatrix}a\sqcup\\ \sqcup b\end{bmatrix}\begin{bmatrix}\sqcup\\ \sqcup\end{bmatrix}\begin{bmatrix}\sqcup\\ \sqcup\end{bmatrix}\begin{bmatrix}\#\\ \#\end{bmatrix}\ \begin{bmatrix}\triangleright\\ \triangleright\end{bmatrix}\begin{bmatrix}\sqcup\\ \sqcup\end{bmatrix}\begin{bmatrix}b\sqcup\\ \text{ja}\sqcup\end{bmatrix}\begin{bmatrix}\sqcup\\ \sqcup\end{bmatrix}\begin{bmatrix}\#\\ \#\end{bmatrix}$$

$$\begin{bmatrix}\triangleright\\ \triangleright\end{bmatrix}\begin{bmatrix}\sqcup\text{ja}\\ \text{ja}\end{bmatrix}\begin{bmatrix}\sqcup\\ \sqcup\end{bmatrix}\begin{bmatrix}\sqcup\\ \sqcup\end{bmatrix}\begin{bmatrix}\#\\ \#\end{bmatrix}\ \begin{bmatrix}\triangleright\text{ja}\\ \text{ja}\end{bmatrix}\begin{bmatrix}\sqcup\\ \sqcup\end{bmatrix}\begin{bmatrix}\sqcup\\ \sqcup\end{bmatrix}\begin{bmatrix}\#\\ \#\end{bmatrix}$$

$$\begin{bmatrix}\text{ja}\sqcup\\ \text{ja}\end{bmatrix}\begin{bmatrix}\sqcup\\ \sqcup\end{bmatrix}\begin{bmatrix}\#\\ \#\end{bmatrix}\ \begin{bmatrix}\text{ja}\sqcup\\ \text{ja}\end{bmatrix}\begin{bmatrix}\#\\ \#\end{bmatrix}\ \begin{bmatrix}\text{ja}\#\#\\ \#\end{bmatrix}$$

- Verwendete Steintypen:

$$\begin{bmatrix}\#\\ \#a\triangleright 11\#\end{bmatrix},\begin{bmatrix}a\triangleright\\ \triangleright b\end{bmatrix},\begin{bmatrix}1\\ 1\end{bmatrix},\begin{bmatrix}\#\\ \#\end{bmatrix},\begin{bmatrix}\triangleright\\ \triangleright\end{bmatrix},\begin{bmatrix}b1\\ \sqcup d\end{bmatrix},\begin{bmatrix}\sqcup\\ \sqcup\end{bmatrix},\begin{bmatrix}d1\\ 1d\end{bmatrix},$$

$$\begin{bmatrix}1d\#\\ f1\sqcup\#\end{bmatrix},\begin{bmatrix}\sqcup f1\\ a\sqcup\sqcup\end{bmatrix},\begin{bmatrix}a\sqcup\\ \sqcup b\end{bmatrix},\begin{bmatrix}b\sqcup\\ \text{ja}\sqcup\end{bmatrix},\begin{bmatrix}\sqcup\text{ja}\\ \text{ja}\end{bmatrix},\begin{bmatrix}\triangleright\text{ja}\\ \text{ja}\end{bmatrix},\begin{bmatrix}\text{ja}\sqcup\\ \text{ja}\end{bmatrix},\begin{bmatrix}\text{ja}\#\#\\ \#\end{bmatrix}$$

PCP **ist unentscheidbar (5/7)**

Beweisskizze (Forts.)

- Sei $M = (Q, \Gamma, \delta, s)$ TM, $x \in \Sigma^*$
  - OBdA: $\# \notin \Gamma$
- Die zugehörige SPCP-Eingabe hat das Alphabet $Q \cup \Gamma \cup \{\#\}$ und die folgenden Regeln:

(1) Start: $\left[\frac{\#}{\#s \triangleright x\#}\right]$

(2) Kopierregeln: $\left[\frac{\sigma}{\sigma}\right]$, für jedes $\sigma \in \Gamma \cup \{\#\}$

(3) δ-Regeln: für alle $q, q' \in Q$ und alle $\sigma, \sigma', \tau \in \Gamma \cup \{\#\}$:

- falls $\delta(q, \sigma) = (q', \sigma', \downarrow)$: $\left[\frac{q\sigma}{q'\sigma'}\right]$,
- falls $\delta(q, \sigma) = (q', \sigma', \rightarrow)$: $\left[\frac{q\sigma}{\sigma' q'}\right]$
- falls $\delta(q, \sigma) = (q', \sigma', \leftarrow)$: $\left[\frac{\tau q\sigma}{q'\tau\sigma'}\right]$, für alle $\tau \in \Gamma$,

Beweisskizze (Forts.)

(4) δ-Regeln für den „rechten Rand“:

Für alle $q, q' \in Q$ und alle $\sigma', \tau \in \Gamma \cup \{\#\}$:

- falls $\delta(q, \sqcup) = (q', \sigma', \downarrow)$: $\left[\frac{q\#}{q'\sigma'\#}\right]$,
- falls $\delta(q, \sqcup) = (q', \sigma', \rightarrow)$: $\left[\frac{q\#}{\sigma' q'\#}\right]$
- falls $\delta(q, \sqcup) = (q', \sigma', \leftarrow)$: $\left[\frac{\tau q\#}{q'\tau\sigma'\#}\right]$, für alle $\tau \in \Gamma$,

(5) Löschregeln:

$\left[\frac{\sigma\text{ja}}{\text{ja}}\right]$, $\left[\frac{\text{ja}\sigma}{\text{ja}}\right]$, $\left[\frac{\sigma\text{nein}}{\text{nein}}\right]$, $\left[\frac{\text{nein}\sigma}{\text{nein}}\right]$, $\left[\frac{\sigma h}{h}\right]$, $\left[\frac{h\sigma}{h}\right]$,

für alle $\sigma \in \Gamma \cup \{\#\}$

(6) Abschlussregeln:

$\left[\frac{\text{ja}\#\#}{\#}\right]$, $\left[\frac{\text{nein}\#\#}{\#}\right]$, $\left[\frac{h\#\#}{\#}\right]$

PCP ist unentscheidbar (6/7)

Beweisskizze (Forts.)

- Falls $M(x)$ anhält, gibt es $\hat{K}_0, \ldots, \hat{K}_t, C_0, \ldots, C_c$, so dass für alle i gilt:
  - $\hat{K}_0$ kodiert $K_0(x) = (s, (x, 0))$,
  - $K_i \vdash_M K_{i+1}$,
  - $C_0 = \hat{K}_t$,
  - C_{i+1} entsteht aus C_i durch Löschen eines Nachbarzeichens von ja, nein, h, und
  - $C_c =$ ja (oder nein oder h)

- Lösungswort:
  $\#\hat{K}_0\#\hat{K}_1\# \cdots \hat{K}_t\#C_1\# \cdots$
  $\cdots C_{c-1}\#C_c\#\#$

Beweisskizze (Forts.)

- Umgekehrt gibt es eine Lösung (mit $i_1 = 1$) nur, falls $M(x)$ anhält

- Denn:
  - Die Regeln der Typen (2), (3) und (4) erzwingen, dass der Anfang des Lösungsstrings das Anfangsstück einer Berechnung von $M(x)$ kodiert
  - Die Regeln (1)-(4) bewirken, dass der v-String zunächst immer länger als der u-String ist
  - Ein Längenausgleich zwischen dem v-String und dem u-String ist nur durch Anwendung der Regeln aus (5) und (6) möglich
  - Diese können jedoch nur angewendet werden, wenn die Berechnungsfolge eine Konfiguration mit einem Endzustand erreicht

  ➨ $M(x)$ terminiert

- Insgesamt haben wir also: TM-HALT $\leqslant$ SPCP

✎ Der vollständige, formale Beweis, dass f eine Reduktion ist, ist natürlich etwas komplizierter

SPCP $\leqslant$ PCP: Beispiel

Beispiel

- Die PCP-Eingabe $z = \begin{bmatrix}0\\01\end{bmatrix}, \begin{bmatrix}1\\11\end{bmatrix}, \begin{bmatrix}10\\0\end{bmatrix}$ hat die PCP-Lösungsstrings
  - $\begin{bmatrix}0\\01\end{bmatrix}\begin{bmatrix}10\\0\end{bmatrix}$ mit Indexfolge $1, 3$ und
  - $\begin{bmatrix}1\\11\end{bmatrix}\begin{bmatrix}10\\0\end{bmatrix}$ mit Indexfolge $2, 3$

- Wir bilden diese PCP-Eingabe ab auf

$$f(z) = \begin{bmatrix}0@\\@0@1\end{bmatrix}, \begin{bmatrix}1@\\@1@1\end{bmatrix}, \begin{bmatrix}1@0@\\@0\end{bmatrix}, \begin{bmatrix}@0@\\@0@1\end{bmatrix}, \begin{bmatrix}\$\\@\$\end{bmatrix}$$

- Die Indexfolge $2, 3$ führt dann nicht mehr zu einer Lösung
- Die Indexfolge $1, 3$ führt jetzt zur Lösung $4, 3, 5$ mit dem Lösungsstring

$$\begin{bmatrix}@0@\\@0@1\end{bmatrix}\begin{bmatrix}1@0@\\@0\end{bmatrix}\begin{bmatrix}\$\\@\$\end{bmatrix}$$

- Allgemein gilt: $f(z)$ hat genau dann eine Lösung, wenn z eine Lösung mit $i_1 = 1$ hat

PCP ist unentscheidbar (7/7)

Beweisskizze (Forts.)

- Es bleibt zu zeigen: SPCP $\leqslant$ PCP
- Wir nehmen dazu an, dass die Zeichen $\$$ und $@$ nicht in den Eingaben für SPCP vorkommen
- Für jeden String $w = a_1 \cdots a_m$, mit $a_i \in \Sigma$ sei $\tilde{w}$ der String $a_1 @ a_2 @ \cdots @ a_m$
- Wir definieren die Reduktion f wie folgt:
  - Eine SPCP-Eingabe $\begin{bmatrix} u_1 \\ v_1 \end{bmatrix}, \ldots \begin{bmatrix} u_k \\ v_k \end{bmatrix}$ wird abgebildet auf

$$\begin{bmatrix} \tilde{u}_1 @ \\ @\tilde{v}_1 \end{bmatrix}, \ldots, \begin{bmatrix} \tilde{u}_k @ \\ @\tilde{v}_k \end{bmatrix}, \begin{bmatrix} @\tilde{u}_1 @ \\ @\tilde{v}_1 \end{bmatrix}, \begin{bmatrix} \$ \\ @\$ \end{bmatrix}$$

Beweisskizze (Forts.)

- Zu zeigen: f ist total und berechenbar und es gilt für alle SPCP-Eingaben z:
  - z hat genau dann eine Lösung mit $i_1 = 1$, wenn $f(z)$ überhaupt eine Lösung hat

- Denn:
- Sei $(1, i_2, \ldots, i_n)$ eine Lösung für z

➨ $(k+1, i_2, \ldots, i_n, k+2)$ ist eine Lösung für $f(z)$

- Sei umgekehrt $(i_1, \ldots, i_n)$ eine Lösung minimaler Länge für $f(z)$

➨ – $i_1 = k+1$
  – $i_n = k+2$ sowie
  – $i_j \in \{1, \ldots, k\}$, für $j \in \{2, \ldots, n-1\}$

➨ $(1, i_2, \ldots, i_{n-1})$ ist Lösung für z

Inhalt

15.1 Der Satz von Rice

15.2 Das Postsche Korrespondenzproblem

▷ **15.3 Unentscheidbare Grammatikprobleme**

Unentscheidbare Grammatik-Probleme (1/3)

Satz 15.4

- CFG-SCHNITT ist unentscheidbar

Beweis

- Dies folgt sofort aus der Unentscheidbarkeit von PCP und PCP $\leqslant$ CFG-SCHNITT
  ☞ Satz 14.2, Lemma 14.3

Definition (CFG-UNAMB)

Gegeben: Kontextfreie Grammatik G

Frage: Ist G eindeutig?

- Eindeutige Grammatiken heißen auf Englisch *unambiguous*

Definition (CFGEQUI)

Gegeben: Kontextfreie Grammatiken G_1, G_2

Frage: Ist $L(G_1) = L(G_2)$?

Definition (CFGREGEQUI)

Gegeben: Kontextfreie Grammatik G, RE α

Frage: Ist $L(G) = L(\alpha)$?

Definition (CFGALL)

Gegeben: Kontextfreie Grammatik G

Frage: Ist $L(G) = \Sigma^*$?

Definition (CFGCONT)

Gegeben: Kontextfreie Grammatiken G_1, G_2

Frage: Ist $L(G_1) \subseteq L(G_2)$?

Satz 15.5

- Die folgenden Entscheidungsprobleme sind unentscheidbar:
  - CFG-UNAMB
  - CFGEQUI
  - CFGREGEQUI
  - CFGALL
  - CFGCONT

Weitere unentscheidbare Grammatik-Probleme (2/3)

Beweisskizze: CFG-UNAMB

- Beweis durch Reduktion:
  $$\text{PCP} \leqslant \overline{\text{CFG-UNAMB}}$$
- Sei $z = (u_1, v_1), \ldots, (u_k, v_k)$ eine Eingabe für PCP

- Wir definieren wieder:
  - G_1: $S_1 \to u_1 S_1 1 \mid \cdots \mid u_k S_1 k \mid u_1\$1 \mid \cdots \mid u_k\$k$
  - G_2: $S_2 \to v_1 S_2 1 \mid \cdots \mid v_k S_2 k \mid v_1\$1 \mid \cdots \mid v_k\$k$

- Klar: G_1 und G_2 sind jeweils eindeutig
  - Die Zahlenfolge am Ende des Strings bestimmt eindeutig den Ableitungsbaum ☞ sogar die Ableitung

- Sei G nun die durch Vereinigung der Regeln aus G_1 und G_2 entstehende Grammatik, ergänzt um $S \to S_1 \mid S_2$

Beweisskizze (Forts.)

- Es gilt:
  - Ist $\vec{i}$ eine Lösung für z mit Lösungswort w, so hat der String $w\$\overleftarrow{i}$ zwei Ableitungsbäume

    ✎ Dabei bezeichnet $\overleftarrow{i}$ die Umkehrung der Zahlenfolge $\vec{i}$
  - Also: z hat Lösung $\Rightarrow$ G ist nicht eindeutig

- Andererseits:
  - Hat ein Wort $w\$\overleftarrow{i}$ zwei Ableitungsbäume, so verwendet der eine S_1, und der andere S_2, also:
    $$w\$\overleftarrow{i} \in L(G_1) \cap L(G_2)$$

  ➡ $\vec{i}$ ist Lösung von z
  - Also: G ist nicht eindeutig $\Rightarrow$ z hat Lösung
- Insgesamt:
  $$z \in \text{PCP} \iff G \text{ nicht eindeutig}$$

Weitere unentscheidbare Grammatik-Probleme (3/3)

Beweisskizze (Forts.)

- Die Unentscheidbarkeit von CFGALL folgt ebenfalls aus dem Beweis der Unentscheidbarkeit von CFG-SCHNITT

- Zu den Grammatiken G_1, G_2 aus diesem Beweis lassen sich kontextfreie Grammatiken H_1, H_2 für die Komplemente von $L(G_1)$ und $L(G_2)$ finden
  - Denn: $L(G_1)$ und $L(G_2)$ sind deterministisch kontextfrei
  - ➨ Die Komplemente von $L(G_1)$ und $L(G_2)$ sind kontextfrei!

➨ $L(G_1) \cap L(G_2) = \varnothing \iff L(H_1) \cup L(H_2) = \Sigma^*$

- Sei nun H die durch Vereinigung der Regeln von H_1 und H_2 und Hinzufügung einer neuen Startvariablen S und zusätzlichen Regeln $S \to S_1 \mid S_2$ entstehende Grammatik

➨ $L(G_1) \cap L(G_2) = \varnothing \iff L(H) = \Sigma^*$

- Wir erhalten insgesamt: PCP $\leqslant \overline{\text{CFGALL}}$
- Daraus folgt die Unentscheidbarkeit von CFGALL

- Die Unentscheidbarkeit der anderen Probleme folgt leicht durch Reduktion von $\overline{\text{CFGALL}}$ und damit auch von CFGALL

Zusammenfassung

- Der Satz von Rice sagt aus, dass semantische Aussagen über Programme nicht algorithmisch getestet werden können
- Das **Postsche Korrespondenzproblem** ist unentscheidbar
- Aus diesem Resultat lässt sich die Unentscheidbarkeit einiger Probleme nachweisen, die mit kontextfreien Sprachen zu tun haben

Literatur

- PCP:
  - Emil L. Post. A variant of a recursively unsolvable problem. *Bulletin of the American Mathematical Society*, 52:264–269, 1946