Theoretische Informatik II

Einheit 4.6

Elementare Berechenbarkeitstheorie II:

Unlösbare Probleme

1. Beweistechniken für Unlösbarkeit
2. Wichtige unlösbare Probleme
3. Der Satz von Rice
4. Unentscheidbare Probleme für kontextfreie Grammatiken

Gibt es Probleme, die prinzipiell unlösbar sind?

- **Was bedeutet Unlösbarkeit?**
  - Funktionen, die von keinem Computer je berechnet werden können
  - Mengen ($\equiv$ Sprachen $\equiv$ Probleme), die unentscheidbar sind
  - Mengen, die nicht einmal aufzählbar ($\equiv$ semi-entscheidbar) sind

  Also prinzipielle Grenzen für die Fähigkeiten von Computern
  unabhängig davon, welche Fortschritte die Informatik je erzielen wird

- **Warum muß es unlösbare Probleme geben?**

  Kardinalitätsargument: Es gibt mehr Funktionen als Programme
  - Die Menge $\mathbb{N} \rightarrow \mathbb{N}$ der Funktionen auf $\mathbb{N}$ ist nicht abzählbar (Beweis folgt)
  - Die Menge der Sprachen über Σ^* (oder Funktionen) ist nicht abzählbar
  - Programme sind durch Textketten beschreibbar – also abzählbar

$$\Downarrow$$

Nicht jede Funktion kann berechenbar sein

Cantors Beweis für Überabzählbarkeit von $\mathbb{N}\to\mathbb{N}$

Die Menge $\mathbb{N}\to\mathbb{N}$ ist "überabzählbar" unendlich

Es ist nicht möglich, alle Funktionen über $\mathbb{N}$ durchzuzählen

Beweis: Wir nehmen an $\mathbb{N}\to\mathbb{N}$ sei abzählbar

– Dann kann man alle Funktionen über $\mathbb{N}$ in eine Tabelle eintragen

	0	1	2	3	4	...
f_0	$f_0(0)+1$	$f_0(1)$	$f_0(2)$	$f_0(3)$	$f_0(4)$	...
f_1	$f_1(0)$	$f_1(1)+1$	$f_1(2)$	$f_1(3)$	$f_1(4)$	...
f_2	$f_2(0)$	$f_2(1)$	$f_2(2)+1$	$f_2(3)$	$f_2(4)$	...
f_3	$f_3(0)$	$f_3(1)$	$f_3(2)$	$f_3(3)+1$	$f_3(4)$	...
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	...

– Definiere eine neue Funktion $f{:}\mathbb{N}\to\mathbb{N}$ durch $f(x) = f_x(x)+1$

– f ist offensichtlich total, kann aber in der Tabelle nicht vorkommen

– Ansonsten wäre $f{=}f_i$ für ein i und $f_i(i) = f(i) = f_i(i)+1$ ✓

Wie beweist man Unlösbarkeit konkret?

- **Ein Pumping-Lemma für $\mathcal{L}_0$ kann es nicht geben**
  - Jede endlich beschreibbare syntaktische Struktur ist entscheidbar
  - Unlösbare Probleme müssen komplizierte syntaktische Struktur haben
- **Beginne mit konstruierten Gegenbeispielen**
  - Beispiele, in welche die Unlösbarkeit "hineincodiert" ist
    - $S = \{i \mid i \in \mathbf{domain}(\varphi_i)\}$ ist unentscheidbar (Selbstanwendbarkeit)
  - Führe Unlösbarkeit "echter" Probleme auf diese Beispiele zurück
- **Welche Probleme werden unentscheidbar sein?**
  - Halteproblem: Terminiert ein Programm bei einer bestimmten Eingabe?
    Intuitives Argument: Programme können beliebig kompliziert sein
    - Sie können versteckte Endlosschleifen enthalten
    - Man kann nicht prüfen, ob diese Programme überhaupt anhalten
  - Korrektheitsproblem: Erfüllt ein Programm eine gegebene Spezifikation?
  - Äquivalenzproblem: Berechnen zwei Programme die gleiche Funktion ?
    Argument: Man kann ja nicht einmal die Terminierung überprüfen

Techniken zum Nachweis von Unlösbarkeit

- **Diagonalisierung** (Cantors Beweis)
  - Gegenbeispielkonstruktion für Klassen unendlicher Objekte
    z.B. berechenbare Funktionen, entscheidbare & aufzählbare Mengen
  - Konstruiere ein Objekt, das sich von jedem anderen Objekt der Klasse an einer Stelle unterscheidet, und somit nicht zur Klasse gehören kann
- **Wachstums- und Monotoniearguments**
  - Funktion wächst stärker als jede berechenbare Funktion
    und kann daher nicht selbst berechenbar sein
- **Rückführung auf bekanntes unlösbares Problem**
  - Lösung des Problems würde Lösung des unlösbaren Problems liefern
- **Anwendung allgemeiner theoretischer Resultate**
  - Unlösbarkeit folgt direkt aus bekannten Sätzen

Selbstanwendbarkeit ist unentscheidbar

- $\overline{S} = \{i \mid i \notin \mathbf{domain}(\varphi_i)\}$ **ist nicht aufzählbar**
  - Annahme: $\overline{S}$ ist aufzählbar
  - Dann ist $\overline{S} = \mathrm{domain}(f)$ für ein berechenbares $f : \mathbb{N} \to \mathbb{N}$
  - Dann gibt es ein $j \in \mathbb{N}$ mit $f = \varphi_j$ und es folgt

$$j \in \overline{S} \;\Leftrightarrow\; j \notin \mathrm{domain}(\varphi_j) \;\Leftrightarrow\; j \notin \mathrm{domain}(f) \;\Leftrightarrow\; j \notin \overline{S}$$

  - Dies ist ein Widerspruch. Also kann $\overline{S}$ nicht aufzählbar sein ✓

Kurzer Beweis, da Diagonalisierung bereits in Definition von $\overline{S}$ enthalten

  - $\overline{S}$ wird zuweilen auch **Diagonalisierungssprache L_d** genannt

- $S = \{i \mid i \in \mathbf{domain}(\varphi_i)\}$ **ist nicht entscheidbar**
  - Wenn S entscheidbar wäre, müsste $\overline{S}$ aufzählbar sein ✓

Das Halteproblem ist unentscheidbar (vgl. TI-1)

Annahme: $H = \{(i,n) \mid n \in \text{domain}(\varphi_i)\}$ ist entscheidbar

– Dann ist $\chi_H : \mathbb{N}^2 \to \mathbb{N}$ mit $\chi_H(i,n) = \begin{cases} 1 & n \in \text{domain}(\varphi_i) \\ 0 & \text{sonst} \end{cases}$ berechenbar

– Definiere eine neue Funktion $f : \mathbb{N} \to \mathbb{N}$ durch

$$f(i) := \begin{cases} 0 & \text{falls } i \notin \text{domain}(\varphi_i) \\ \perp & \text{sonst} \end{cases}$$

$\varphi_i(n)$	0	1	2	3	4	...
φ_0	$\perp$	$\times$	$\times$	$\perp$	$\times$	...
φ_1	$\perp$	$\times$	$\times$	$\times$	$\times$	...
φ_2	$\times$	$\times$	$\times$	$\times$	$\times$	...
φ_3	$\perp$	$\times$	$\perp$	$\perp$	$\perp$	...
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	...

– Dann ist f berechenbar,
da $f(i) = \mu_z[\chi_H(i,i) = 0]$

– Also gibt es ein j mit $f = \varphi_j$

– Aber für j gilt: $j \in \text{domain}(\varphi_j) \Leftrightarrow j \in \text{domain}(f) \Leftrightarrow j \notin \text{domain}(\varphi_j)$

– Dies ist ein Widerspruch, also kann H nicht entscheidbar sein ✓

Terminierung von Programmen ist nicht testbar

Diagonalisierung – was ist die Methode?

- **Ziel: Konstruktion eines Gegenbeispiels**
  - Zeige, daß eine unendliche Menge M eine Eigenschaft P nicht besitzt
  - z.B. Entscheidbarkeit, Aufzählbarkeit, Abzählbarkeit
- **Methodik**
  - Wir nehmen an, M habe die Eigenschaft P
  - Konstruiere ein x, das von allen Elementen in M verschieden ist
  - Zeige, daß $x \in M$ aufgrund der Annahme gelten muß
  - Aus dem Widerspruch folgt, daß die Annahme nicht gelten kann
- **Konstruktion: Cantorsches Diagonalverfahren**
  - Trage alle Elemente von M als Zeilen einer Tabelle auf
  - Konstruiere x als Diagonale mit Abweichung an jedem Punkt
  - Also kann x nicht als Zeile vorkommen

Mächtige, aber komplizierte Beweistechnik

Total rekursive Funktionen sind nicht aufzählbar

Annahme: $TR = \{i \mid \varphi_i \text{ total}\}$ ist aufzählbar

– Dann gibt es eine total rekursive Funktion f mit $\text{range}(f) = TR$

– Somit kann man alle Funktionen aus $\mathcal{TR}$ in eine Tabelle eintragen

	0	1	2	3	4	...
$\varphi_{f(0)}$	$\varphi_{f(0)}(0)+1$	$\varphi_{f(0)}(1)$	$\varphi_{f(0)}(2)$	$\varphi_{f(0)}(3)$	$\varphi_{f(0)}(4)$	...
$\varphi_{f(1)}$	$\varphi_{f(1)}(0)$	$\varphi_{f(1)}(1)+1$	$\varphi_{f(1)}(2)$	$\varphi_{f(1)}(3)$	$\varphi_{f(1)}(4)$	...
$\varphi_{f(2)}$	$\varphi_{f(2)}(0)$	$\varphi_{f(2)}(1)$	$\varphi_{f(2)}(2)+1$	$\varphi_{f(2)}(3)$	$\varphi_{f(2)}(4)$	...
$\varphi_{f(3)}$	$\varphi_{f(3)}(0)$	$\varphi_{f(3)}(1)$	$\varphi_{f(3)}(2)$	$\varphi_{f(3)}(3)+1$	$\varphi_{f(3)}(4)$	...
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	...

– Definiere eine neue Funktion $h{:}\mathbb{N}\to\mathbb{N}$ durch $h(n) = \varphi_{f(n)}(n)+1$

– h ist offensichtlich total und berechenbar, denn $h(n) = u(f(n), n)+1$

– Also gibt es ein $i \in TR$ mit $h = \varphi_i$ und damit ein $j \in \mathbb{N}$ mit $i = f(j)$

– Für dieses j gilt: $\varphi_{f(j)}(j) = h(j) = \varphi_{f(j)}(j)+1$

– Dies ist ein Widerspruch, also kann TR nicht aufzählbar sein ✓

Monotoniearguments

- **Ziel**
  - Zeige, daß eine Funktion f eine Eigenschaft P nicht besitzt
  - z.B. primitiv rekursiv, berechenbar, maximale Komplexität
- **Methodik**
  - Zeige daß f stärker wächst als jede Funktion mit Eigenschaft P
    - Induktive Analyse des Wachstumsverhaltens von f
    - Analysiere maximales Wachstum von Funktionen mit Eigenschaft P
  - f kann also nicht selbst Eigenschaft P besitzen
- **Beispiele**
  - Die Ackermann Funktion ist nicht primitiv-rekursiv
  - Die Busy-Beaver Funktion ist nicht berechenbar (Beweis folgt)
  - Komplexitätsanalysen

Das Busy Beaver Problem

Biber stauen Bäche, indem sie Holzstücke in den Bach tragen. Fleißige Biber tragen mehr Holzstücke zusammen als faule. Größere Biber können mehr leisten als kleine.

Die Busy Beaver Funktion bestimmt die Länge der längsten ununterbrochenen Staumauer, die ein Biber ohne eine bereits vorhandene Teilmauer zusammentragen kann.

- **Beschreibe Biber durch Turingmaschinen**
  - Holzstücke werden durch das Symbol | beschrieben
  - $M = (\{1..n\}, \{|\}, \{|, B\}, \delta, 1, B, \emptyset)$ heißt Busy-Beaver TM der Größe n
  - **BBT(n)** sei die Menge aller Busy-Beaver Turingmaschinen der Größe n
- **Beschreibe Produktivität von Bibern**
  - $\textbf{Produktivität}(M) = \begin{cases} n & \text{wenn } f_M(\epsilon) = |^n \\ 0 & \text{wenn } M \text{ bei Eingabe } \epsilon \text{ nicht hält} \end{cases}$
- **Beschreibe maximal mögliche Leistung von Bibern**
  - $\mathbf{BB}(n) = \max\{\text{Produktivität}(M) \mid M \in \text{BBT}(n)\}$

Ist die Busy-Beaver Funktion berechenbar?

Busy Beaver Problem: Intuitive Analyse

- **Beispiel einer BBT(2) Maschine**

δ	$\vert$	B
$\rightarrow$ 1	$(2,\vert,L)$	$(2,\vert,R)$
2	—	$(1,\vert,L)$

Arbeitsweise:

$(\epsilon,1,B) \vdash (\vert,2,B)$
$\vdash (\epsilon,1,\vert\vert) \vdash (\epsilon,2,B\vert\vert)$
$\vdash (\epsilon,1,B\vert\vert\vert) \vdash (\vert,2,\vert\vert\vert)$ stop

– Produktivität ist 3 (4, wenn man alle Holzstücke zählt)

- **BB(n) bekannt für kleine n:** (alle Holzstücke auf dem Band gezählt)

n	1	2	3	4	5	6	...
	1	4	6	13	$\geq$4098	$\geq 1.2*10^{865}$	

http://www.drb.insel.de/~heiner/BB

- **Vollständige Analyse nicht möglich**
  - $|\mathrm{BBT}(n)|$ ist etwa $(|Q|*|\Gamma|*|\{R,L\}|)^{(|Q|*|\Gamma|)} = (4n)^{2n}$

    $|\mathrm{BBT}(1)|$=16, $|\mathrm{BBT}(2)|$=4096, $|\mathrm{BBT}(3)|$=2985984, ...
  - Anzahl möglicher Bandkonfigurationen einer TM ist unbegrenzt

Das Busy Beaver Problem ist unlösbar

- **BB ist streng monoton:** $\boldsymbol{i>j \;\Leftrightarrow\; \mathrm{BB}(i)>\mathrm{BB}(j)}$
  - Für alle n gilt $\mathrm{BB}(n+1)>\mathrm{BB}(n)$
    - Beginne mit der $\mathrm{BB}(n)$-Maschine; wechsele am Ende in Zustand $n+1$
    - Laufe an das rechte Ende bis zum ersten Blank und schreibe ein |
  - $i>j \;\Leftrightarrow\; \mathrm{BB}(i)>\mathrm{BB}(j)$ folgt nun durch Induktion über $i-j$
- **Für alle n gilt:** $\boldsymbol{\mathrm{BB}(n+5)\geq 2n}$
  - Mit n Zuständen kann man n Striche generieren
  - Mit 5 Zuständen kann man Striche verdoppeln (vgl M_4 aus Einheit 4.2)
- **BB berechenbar** $\boldsymbol{\Rightarrow\; \exists k\in\mathbb{N}.\; \mathrm{BB}(n+2k)\geq \mathrm{BB}(\mathrm{BB}(n))}$
  - Wähle k = Zahl der Zustände der TM über $\Gamma=\{|, B\}$, die BB berechnet
  - Mit n Zuständen generiere n Striche
  - Mit k Zuständen berechne jetzt $\mathrm{BB}(n)$
  - Mit weiteren k Zuständen berechne $\mathrm{BB}(\mathrm{BB}(n))$
- **BB kann nicht berechenbar sein**
  - Sonst $\mathrm{BB}(n+5+2k) \geq \mathrm{BB}(\mathrm{BB}(n+5)) \geq \mathrm{BB}(2n)$ für ein k und alle n
  - Für $n=2k+6$ widerspräche $\mathrm{BB}(4k+11) \geq \mathrm{BB}(4k+12)$ der Monotonie ✓

Beweisführung durch Reduktion

- **Das Halteproblem braucht keinen Diagonalbeweis**

  Die Unentscheidbarkeit von $\boldsymbol{H = \{(i,n) \mid n \in \mathbf{domain}(\varphi_i)\}}$ läßt sich auf die Unentscheidbarkeit von $S = \{i \mid i \in \mathrm{domain}(\varphi_i)\}$ zurückführen

  – Annahme: H ist entscheidbar, d.h. χ_H ist berechenbar

  – Wir zeigen, daß die charakteristische Funktion von S berechenbar ist

  · Für alle $i \in \mathbb{N}$ gilt $i \in S \Leftrightarrow (i,i) \in H$

  · Damit ist $\chi_S(i) = \chi_H(i,i)$

  · Da χ_H berechenbar ist, gilt dies auch für χ_S

  – Da S aber unentscheidbar ist, ergibt sich ein Widerspruch ✓

- **Beweis "reduziert" S auf H**

  – Eine Funktion f transformiert Eingaben für S in Eingaben für H

  – Die Transformation f ist berechenbar und es gilt $x \in S \;\Leftrightarrow\; f(x) \in H$

  – Damit transformiert f jede Lösung für H in eine für S

Problemreduktion

- **Ziel: Wiederverwendung bekannter Ergebnisse**
  - Zur Lösung eines Problems P bzw. zum Nachweis seiner Unlösbarkeit
  - Lösbar $\hat{=}$ entscheidbar, aufzählbar, berechenbar, in Zeit t lösbar, ...
- **Methodik zum Nachweis der Unlösbarkeit**
  - Transformiere ein bekannt unlösbares Problem P_1 in Spezialfälle von P
  - Zeige, daß jede Lösung für P in eine Lösung für P_1 transformiert würde
  - P kann also nicht lösbar sein
- **Methodik zur Konstruktion einer Lösung**
  - Transformiere P in Spezialfälle eines als lösbar bekannten Problems P_2
  - Zeige, wie eine Lösung für P_2 in eine Lösung für P transformiert wird
- **Theoretisches Konzept: Reduzierbarkeit**
  - $\boldsymbol{P' \leq P}$, falls $P' = f^{-1}(P) = \{x \mid f(x) \in P\}$ für ein total berechenbares f

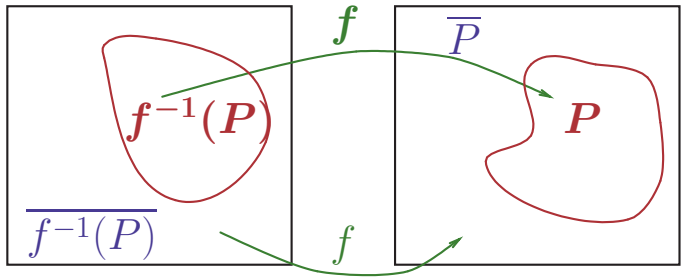
    "**P' ist reduzierbar auf P**"

Achtung: umgangssprachlich wird zuweilen die Problemstellung P auf P' "reduziert"

Beweisführung durch Reduktion

- **Ähnlich zu inversen Homomorphismen in $\mathcal{L}_3$**
  - Es gilt $x \in P' \Leftrightarrow f(x) \in P$
  - Aber: keine syntaktische Restriktion an f

    f muß nur total berechenbar sein



- **$P' \leq P$ bedeutet "P' ist nicht schwerer als P"**
  - Ist P lösbar, dann kann P' wie folgt gelöst werden
    - Bei Eingabe x bestimme $f(x)$ (f ist die Reduktionsfunktion)
    - Löse $f(x)$ mit der Lösungsmethode für P
    - Wegen $x \in P' \Leftrightarrow f(x) \in P$ überträgt sich das Ergebnis
- **Konsequenzen von Reduzierbarkeit $P' \leq P$**
  - Aus P entscheidbar folgt P' entscheidbar: $\chi_{P'}(x) = \chi_{f^{-1}(P)}(x) = \chi_P(f(x))$
  - **Aus P' unentscheidbar folgt P unentscheidbar**
  - Aus P aufzählbar folgt P' aufzählbar: $\psi_{P'}(x) = \psi_{f^{-1}(P)}(x) = \psi_P(f(x))$
  - **Aus P' nicht aufzählbar folgt P nicht aufzählbar**

Beispiele von Problemreduktion

- $\boldsymbol{S}=\{i \mid i \in \text{domain}(\varphi_i)\} \quad \leq \quad \boldsymbol{H}=\{(i,n) \mid n \in \text{domain}(\varphi_i)\}$
  - Es gilt $i \in S \Leftrightarrow (i,i) \in H$.
  - Wähle $f(i) := (i,i)$.
  - Dann ist f total-berechenbar und $S = f^{-1}(H)$
- $\overline{\boldsymbol{H}}=\{(i,n) \mid n \notin \text{domain}(\varphi_i)\} \quad \leq \quad \boldsymbol{PROG_z}=\{i \mid \varphi_i = z\}$ $(z = c_0^1)$
  - Es gilt $(i,n) \in \overline{H} \;\Leftrightarrow\; \forall t \in \mathbb{N}.\Phi_i(n) \neq t$.
  - Da $\Phi = \{(i,n,t) \mid \Phi_i(n) = t\}$ entscheidbar ist, gibt es ein j mit
    $$\varphi_j(i,n,t) = \chi_\Phi(i,n,t) = \begin{cases} 1 & \text{falls } (i,n,t) \in \Phi \\ 0 & \text{sonst} \end{cases}$$
  - Nach dem SMN Theorem gibt es eine total-berechenbare Funktion f mit $\varphi_{f(i,n)}(t) = \varphi_j(i,n,t)$
  - Es folgt $(i,n) \in \overline{H} \;\Leftrightarrow\; \forall t \in \mathbb{N}.\Phi_i(n) \neq t \Leftrightarrow\; \forall t \in \mathbb{N}.\varphi_{f(i,n)}(t) = 0$
    $\Leftrightarrow\; \varphi_{f(i,n)} = z \Leftrightarrow\; f(i,n) \in PROG_z$

    also $\overline{H} = f^{-1}(PROG_z)$

Problemreduktion mit Abschlusseigenschaften

- **Reduktion ist Spezialfall der Abschlußeigenschaften**
  - P, P' entscheidbar, dann auch $P \cup P'$, $P \cap P'$, $P \setminus P'$, $\overline{P}$, $f^{-1}(P)$
  - P, P' aufzählbar, dann auch $P \cup P'$, $P \cap P'$, $g(P)$, $g^{-1}(P)$
  - P entscheidbar $\Leftrightarrow$ P und $\overline{P}$ aufzählbar
- **Abschlußeigenschaften lassen sich umkehren**
  - P nicht entscheidbar $\Rightarrow$ $\overline{P}$ nicht entscheidbar
  - P aufzählbar, nicht entscheidbar $\Rightarrow$ $\overline{P}$ weder aufzählbar noch entscheidbar
  - P entscheidbar, $P \cup P'$ nicht entscheidbar $\Rightarrow$ P' nicht entscheidbar
  - P entscheidbar, $P \setminus P'$ nicht entscheidbar $\Rightarrow$ P' nicht entscheidbar

  $\vdots$ $\vdots$
- **Gleiche Beweismethodik**
  - Zeige, wie Problem mit bekannten Problemen zusammenhängt

Resultate aus Abschlusseigenschaften

- $\{(i,n) \mid n \notin \text{domain}(\varphi_i)\}$ **ist nicht aufzählbar**
  - $\{(i,n) \mid n \notin \text{domain}(\varphi_i)\}$ ist das Komplement des Halteproblems H
  - H ist aufzählbar aber unentscheidbar
  - Also kann $\overline{H}$ nicht aufzählbar sein
- $PROG_z = \{i \mid \varphi_i = z\}$ **ist nicht aufzählbar**
  - Es gilt $\overline{H} \leq PROG_z$ und $\overline{H}$ ist nicht aufzählbar
  - Damit kann $PROG_z$ nicht aufzählbar sein

  Die Menge der Turingmaschinen M mit $L(M)=\emptyset$ ist nicht aufzählbar
  - Beweis ist minimale Modifikation von $\overline{H} \leq PROG_z$
- $PF = \{i \mid \varphi_i \text{ partiell}\}$ **ist unentscheidbar**
  - PF ist das Komplement von $TR = \{i \mid \varphi_i \text{ total}\}$
  - TR ist nicht aufzählbar,
  - Also kann $PF = \overline{TR}$ nicht entscheidbar sein

Welche Verifikationsprobleme sind entscheidbar?

- **Halteproblem** unentscheidbar
  - Kann man von einem beliebigen Programm entscheiden, ob es bei bestimmten Eingaben hält oder nicht?
- **Korrektheitsproblem** unentscheidbar für Nullfunktion
  - Kann man von einem beliebigen Programm entscheiden, ob es eine bestimmte Funktion berechnet oder nicht?
- **Spezifikationsproblem**
  - Kann man von einem beliebigen Programm entscheiden, ob es eine gegebene Spezfikation erfüllt oder nicht?
- **Äquivalenzproblem**
  - Kann man entscheiden, ob zwei beliebige Programme die gleiche Funktion berechnen oder nicht?

Gibt es eine allgemeine Antwort?

Der Satz von Rice

Für $\emptyset \neq P \subset \mathcal{R}$ ist $L_P = \{i \mid \varphi_i \in P\}$ unentscheidbar

Beweis durch Reduktion von $S = \{i \mid i \in \text{domain}(\varphi_i)\}$ auf L_P

– Sei g die nirgends definierte Funktion ($g(x) = \perp$ für alle x).

– Falls $g \notin P$, so wähle $f \in P$ beliebig und definiere

$$f'(i,x) = \begin{cases} f(x) & \text{falls } \varphi_i(i) \neq \perp \\ \perp & \text{sonst} \end{cases}$$

– Dann ist f' berechenbar und nach dem SMN Theorem gibt es ein total-berechenbares h mit $f'(i,x) = \varphi_{h(i)}(x)$

– Damit $i \in S \Rightarrow \forall x.\ f'(i,x) = \varphi_{h(i)}(x) = f(x) \Rightarrow \varphi_{h(i)} = f \in P \Rightarrow h(i) \in L_P$

$i \notin S \Rightarrow \forall x.\ f'(i,x) = \varphi_{h(i)}(x) = \perp \Rightarrow \varphi_{h(i)} = g \notin P \Rightarrow h(i) \notin L_P$

– Es folgt: $i \in S \Leftrightarrow h(i) \in L_P$, d.h. $S \leq L_P$. Also ist L_P unentscheidbar. ✓

– Falls $g \in P$ wähle ein beliebiges $f \notin P$ und zeige so $S \leq \overline{L_P}$ ✓

Keine nichttriviale (extensionale) Eigenschaft berechenbarer Funktionen ist entscheidbar

Anwendungen des Satzes von Rice

- $MON = \{i \mid \forall k.\ \varphi_i(k) < \varphi_i(k+1)\}$
  - Monotone Funktionen sind unentscheidbar
- $EF = \{i \mid \forall j.\ \varphi_i(j) \in \{0,1\}\}$
  - Entscheidungsfunktion zu sein ist unentscheidbar
- $PROG_{spec} = \{i \mid \varphi_i \text{ erfüllt Spezifikation } spec\}$
  - Allgemeines Spezifikationsproblem ist unentscheidbar
- $PROG_f = \{i \mid \varphi_i = f\}$
  - Korrektheitsproblem ist unentscheidbar
- $EQ = \{(i,j) \mid \varphi_i = \varphi_j\}$
  - Äquivalenzproblem ist unentscheidbar
- $RG = \{(i,j) \mid j \in \text{range}(\varphi_i)\}$
  - Bildbereiche sind unentscheidbar

Keine Programmeigenschaft kann getestet werden

Beweise müssen von Hand geführt werden

Rechnerunterstützung nur in Spezialfällen möglich

Das Post'sche Korrespondenzproblem

Codiere Unentscheidbarkeiten auf Grammatiken

- **Informale Beschreibung**
  - Gegeben eine Menge von Wortpaaren $\{(u_1, v_1), .., (u_k, v_k)\}$ in Σ^+
  - Eine **Korrespondenz** ist eine Indexfolge $i_1, .., i_n$ mit $u_{i_1}..u_{i_n} = v_{i_1}..v_{i_n}$
  - $\{(u_1, v_1), .., (u_k, v_k)\}$ heißt **lösbar**, wenn es eine Korrespondenz gibt
- **Beispiele**
  - $K_1 = \{(1, 101),\ (10, 00),\ (011, 11)\}$

    Lösbar mit Korrespondenz 1323, denn $u_1u_3u_2u_3 = v_1v_3v_2v_3 = 101110011$
  - $K_2 = \{(1, 10),\ (101, 01)\}$

    Unlösbar: alle u_i haben mehr Einsen als Nullen, die v_i sind ausgewogen
  - $K_3 = \{(001, 0),\ (01, 011),\ (01, 101),\ (10, 001)\}$

    Lösbar mit 2434421243434434421442134113444212111343412144214113411311312141 13
- **Post'sches Korrespondenzproblem, präzisiert**
  - $\boldsymbol{PKP} = \{(u_1, v_1), .., (u_k, v_k) \mid u_i, v_i \in \Sigma^+ \wedge \exists i_1, .., i_n.\ u_{i_1}..u_{i_n} = v_{i_1}..v_{i_n}\}$

Post'sches Korrespondenzproblem: Aufzählbarkeit

- **Aufzählungsalgorithmus:**
  - **Eingabe**: Wort $w = \{(u_1, v_1), .., (u_k, v_k)\} \in (X \cup \{\texttt{"("}, \texttt{")"}, \texttt{","}\})^*$
  - Durch Klammerzählung bestimme alle u_i und v_i und die Anzahl k
  - Zähle alle möglichen Indexfolgen $i_1, .., i_n$ mit $i_j \leq k$ auf
    (Verwende Umkehrung der Standardtupelfunktion für Listen $\langle i_1, .., i_n \rangle^*$)
    - Falls $u_{i_1}..u_{i_n} = v_{i_1}..v_{i_n}$, so akzeptiere w (Ausgabe 1)
    - Ansonsten generiere die nächste Indexfolge
- **Algorithmus berechnet ψ_{PKP}**
  - $w \in PKP \Rightarrow$ Es gibt $i_1, .., i_n$ mit $u_{i_1}..u_{i_n} = v_{i_1}..v_{i_n}$
    $\Rightarrow$ Aufzählung endet bei $\langle i_1, .., i_n \rangle^*$ mit Ausgabe 1
  - $w \notin PKP \Rightarrow$ Es gibt keine Korrespondenz
    $\Rightarrow$ Aufzählung terminiert nicht, da Test niemals erfolgreich

Unentscheidbarkeit von PKP

Beweis durch doppelte Reduktion

- **Verwende eingeschränkte Version des Problems**
  - $\boldsymbol{MPKP} = \{(u_1, v_1), .., (u_k, v_k) \mid u_i, v_i \in \Sigma^+ \wedge \exists i_2, .., i_n.\ u_1 u_{i_2} .. u_{i_n} = v_1 v_{i_2} .. v_{i_n}\}$
- **Zeige:** $\boldsymbol{MPKP \leq PKP}$
- **Zeige:** $\boldsymbol{H \leq MPKP}$ (also $MPKP$ unentscheidbar)
  - Zeige, daß jede Turingmaschine M als MPKP beschrieben werden kann
    - (u_1, v_1) beschreibt die Erzeugung der Anfangskonfiguration in v_1
    - Weitere Wortpaare entsprechen Konfigurationsübergängen
      (vgl Simulation von Turingmaschinen durch Typ-0 Grammatiken)
  - M hält, wenn das MPKP eine terminierende Berechnung beschreiben kann
  - Korrespondenz bedeutet, daß jedes Ergebnis eines Konfigurationsübergangs Anfangspunkt des nächsten Übergangs ist

Beweise $MPKP \leq PKP$

- **Reduktion erzwingt erstes Wortpaar als Anfang**
  - Erweitere Alphabet Σ zu $\Sigma' = \Sigma \cup \{\#, \$\}$
  - Modifiziere Wörter $w = a_1 ... a_n$ zu $\hat{w} = a_1 \# a_2 \# ... \# a_n$
  - Definiere Abbildung f durch

$$f\{(u_1, v_1), ..(u_k, v_k)\} = \{\underbrace{(\#\hat{u}_1\#, \#\hat{v}_1)}_{(u'_1, v'_1)}, \underbrace{(\hat{u}_1\#, \#\hat{v}_1)}_{(u'_2, v'_2)}, .. \underbrace{(\hat{u}_k\#, \#\hat{v}_k)}_{(u'_{k+1}, v'_{k+1})}, \underbrace{(\$, \#\$)}_{(u'_{k+2}, v'_{k+2})} \}$$

- **Zeige $K \in MPKP \Leftrightarrow f(K) \in PKP$**
  - $K \in MPKP \Rightarrow$ Es gibt $i_2, .., i_n$ mit $u_1 u_{i_2} .. u_{i_n} = v_1 v_{i_2} .. v_{i_n}$

$$\Rightarrow \underbrace{\#\hat{u}_1\#}_{u'_1} \underbrace{\hat{u}_{i_2}\#}_{u'_{i_2+1}} .. \# \underbrace{\hat{u}_{i_n}\#}_{u'_{i_n+1}} \underbrace{\$}_{u'_{k+2}} = \underbrace{\#\hat{v}_1}_{v'_1} \underbrace{\#\hat{v}_{i_2}}_{v'_{i_2+1}} \#.. \underbrace{\#\hat{v}_{i_n}}_{v'_{i_n+1}} \underbrace{\#\$}_{v'_{k+2}}$$

$\Rightarrow 1, i_2{+}1, .., i_n{+}1, k{+}2$ löst $f(K)$ also $f(K) \in PKP$

  - $f(K) \in PKP \Rightarrow$ Es gibt $i_1, .., i_n$ mit $u'_{i_1} .. u'_{i_n} = v'_{i_1} .. v'_{i_n}$

$\Rightarrow i_1{=}1$, da nur u'_1, v'_1 dasselbe Anfangssymbol haben

$i_n{=}k{+}2$, da nur u'_{k+2}, v'_{k+2} dasselbe Endsymbol haben

$\Rightarrow 1, i_2{-}1, .., i_{n-1}{-}1$ löst K also $K \in MPKP$ ✓

$H \leq MPKP$: Transformation

Transformiere M,w in eine Korrespondenz K

(Eingaben (i, n) des Halteproblems werden zunächst in M_i, w_n transformiert)

Sei $M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$, $w \in \Sigma^*$. Bestimme $K := f(M, w)$ wie folgt

– Wähle Alphabet $\Delta := \Gamma \cup S \cup \{\#\}$ für das MPKP

– Erzeuge Anfangskonfiguration in $(u_1, v_1) := (\#, \#q_0 w \#)$

– Beschreibe Konfigurationsübergänge durch Wortpaare

- $(q\,X,\ Y\,p)$ für $\delta(q, X) = (p, Y, R)$
- $(q\,\#,\ Y\,p\#)$ für $\delta(q, B) = (p, Y, R)$
- $(Z\,q\,X,\ p\,Z\,Y)$ für $Z \in \Gamma$ und $\delta(q, X) = (p, Y, L)$
- $(Z\,q\,\#,\ p\,Z\,Y\,\#)$ für $Z \in \Gamma$ und $\delta(q, B) = (p, Y, L)$

– Ergänze "Kopierregeln" (X, X) für alle $X \in \Gamma \cup \{\#\}$

– Ergänze "Löschregeln" $(X\,q_f, q_f)$, $(q_f\,Y, q_f)$ und $(X\,q_f\,Y, q_f)$

für alle $X, Y \in \Gamma$, $q_f \in F$

– Ergänze "Abschlußregel" $(q_f\,\#\,\#,\ \#)$ für alle $q_f \in F$

$H \leq MPKP$: Korrektheit der Transformation f

(1) Zeige: M hält bei Eingabe $w \Rightarrow f(M,w) \in MPKP$

- Da M bei Eingabe w anhält, gibt es o.B.d.A. eine Konfigurationsfolge $\kappa_0 = (\epsilon, q_0, w) \vdash \kappa_1 \ldots \vdash \kappa_t = (x_0..x_m, q_f, y_0..y_n)$ für ein $q_f \in F$
- Mit Überführungs- und Kopierregeln bauen wir folgendes Wortpaar auf
  - $u = \#q_0 w \# \kappa_1 \# \ldots \# \kappa_t \# x_0..x_{m-1}$
  - $v = \#q_0 w \# \kappa_1 \# \ldots \# \kappa_t \# x_0..x_{m-1} x_m\, q_f\, y_0..y_n \# x_0..x_{m-1}$
- Mit der Löschregel $(x_m\, q_f, q_f)$ erzeugen wir daraus
  - $\#q_0 w \# \kappa_1 \# \ldots \# \kappa_t \# x_0..x_{m-1} x_m\, q_f$
  - $\#q_0 w \# \kappa_1 \# \ldots \# \kappa_t \# x_0..x_{m-1} x_m\, q_f\, y_0..y_n \# x_0..x_{m-1} q_f$
- Mit den Kopierregeln bekommen wir
  - $\#q_0 w \# \kappa_1 \# \ldots \# \kappa_t \# x_0..x_{m-1} x_m\, q_f\, y_0..y_n \#$
  - $\#q_0 w \# \kappa_1 \# \ldots \# \kappa_t \# x_0..x_{m-1} x_m\, q_f\, y_0..y_n \# x_0..x_{m-1}\, q_f\, y_0..y_n \#$
- Mit den Lösch- und Kopierregeln ergibt sich
  - $\#q_0 w \# \ldots \# x_0..x_{m-1}\, q_f\, y_0..y_n \# x_0..x_{m-2}\, q_f\, y_0..y_n \# \ldots \# q_f\, y_n \#$
  - $\#q_0 w \# \ldots \# x_0..x_{m-1}\, q_f\, y_0..y_n \# x_0..x_{m-2}\, q_f\, y_0..y_n \# \ldots \# q_f\, y_n \# q_f \#$
- Mit der Abschlußregel ergibt sich schließlich
  - $\#q_0 w \# \ldots \# x_0..x_{m-1}\, q_f\, y_0..y_n \# x_0..x_{m-2}\, q_f\, y_0..y_n \# \ldots \# q_f\, y_n \# q_f \# \#$
  - $\#q_0 w \# \ldots \# x_0..x_{m-1}\, q_f\, y_0..y_n \# x_0..x_{m-2}\, q_f\, y_0..y_n \# \ldots \# q_f\, y_n \# q_f \# \#$ ✓

$H \leq MPKP$: Korrektheit der Transformation f

(2) Zeige: $f(M,w) \in MPKP \Rightarrow M$ hält auf w

- Es gelte $f(M,w) \in MPKP$
- Also gibt es $i_2, .., i_n$ mit $u_1 u_{i_2} .. u_{i_n} = v_1 v_{i_2} .. v_{i_n}$
- Wegen der Struktur der Korrespondenzen muß $u_1 u_{i_2} .. u_{i_n}$ mit $\#q_0 w\#$ beginnen und mit $q_f \# \#$ enden
- Wegen der Überführungsregeln gilt $\#u_{i_2} \hat{=} v_1$, $\#u_{i_3} \hat{=} v_{i_2}$, ...
- Aus der Korrespondenz können wir daher eine Konfigurationsfolge $\kappa_0 = (\epsilon, q_0, w) \vdash \kappa_1 \ldots \vdash \kappa_t = (x_0..x_m, q_f, y_0..y_n)$ für ein $q_f \in F$ konstruieren
- Also hält M bei Eingabe w an ✓

$$\Downarrow$$

$H \leq MPKP$, also ist $MPKP$ unentscheidbar

Unentscheidbarkeiten auf Grammatiken (HMU §9.5)

Für kontextfreie Grammatiken G, G' sind die folgende Probleme unentscheidbar

1. $L(G) \cap L(G') = \emptyset$
2. $L(G) \cap L(G')$ **unendlich**
3. $L(G) \cap L(G')$ **kontextfrei**
4. $L(G) \subseteq L(G')$
5. $L(G) = L(G')$
6. $L(G) = \Sigma^*$
7. G **mehrdeutig**
8. $\overline{L(G)}$ **kontextfrei**
9. $L(G)$ **regulär**
10. $L(G) \in$ **DPDA**

Beweise Unentscheidbarkeiten durch Reduktion

- **Transformiere ein PKP in Grammatiken G und G'**
  - Gegeben $K = \{(u_1, v_1), .., (u_k, v_k)\}$ über $\Sigma = \{0, 1\}$
  - Wähle Terminalalphabet $:= \{0, 1, \$, a_1, .., a_k\}$
  - Konstruiere $G := S \to A\$B$,
    $A \to a_1 A u_1, .., A \to a_k A u_k, \; A \to a_1 u_1, .., A \to a_k u_k$
    $B \to v_1^R B a_1, .., B \to v_k^R B a_k, \; B \to v_1^R a_1, .., B \to v_k^R a_k$
  - Dann gilt $L(G) = \{a_{i_n}..a_{i_1} u_{i_1}..u_{i_n} \,\$\, v_{j_m}^R..v_{j_1}^R a_{j_1}..a_{j_m} \mid i_\nu, j_\mu \leq k\}$
  - Konstruiere $G' := S \to a_1 S a_1, .., S \to a_k S a_k, \; S \to T$,
    $T \to 0T0, \; T \to 1T1, \; T \to \$$,
  - Dann gilt $L(G') = \{uv \,\$\, v^R u^R \mid u \in \{a_1, .., a_k\}^*, v \in \{0, 1\}^*\}$
- **$L(G) \cap L(G') = \emptyset$ ist unentscheidbar**
  - Folgt direkt aus $K \in PKP \;\Leftrightarrow\; i_1..i_n = j_1..j_m$ und $u_{i_1}..u_{i_n} = v_{j_1}..v_{j_m}$
    $\Leftrightarrow\; L(G) \cap L(G') \neq \emptyset$ ✓
- **$L(G) \cap L(G')$ unendlich ist unentscheidbar**
  - Es gilt $u_{i_1}..u_{i_n} = v_{i_1}..v_{i_n} \;\Rightarrow\; u_{i_1}..u_{i_n} u_{i_1}..u_{i_n} = v_{i_1}..v_{i_n} v_{i_1}..v_{i_n} \;\Rightarrow\; \ldots$
  - Es folgt $K \in PKP \;\Leftrightarrow\; L(G) \cap L(G')$ unendlich ✓

Mehrdeutigkeit ist unentscheidbar

- **G mehrdeutig ist unentscheidbar**
  - Gegeben $K = \{(u_1, v_1), .., (u_k, v_k)\}$ über $\Sigma = \{0, 1\}$
  - Wähle Terminalalphabet $:= \{0, 1, \$, a_1, .., a_k\}$
  - Konstruiere $G := S \to A\$B$,

    $A \to a_1 A u_1, .., A \to a_k A u_k, \; A \to a_1 u_1, .., A \to a_k u_k$

    $B \to v_1^R B a_1, .., B \to v_k^R B a_k, \; B \to v_1^R a_1, .., B \to v_k^R a_k$
  - Wir zeigen $\boldsymbol{K \in PKP \Leftrightarrow G}$ **mehrdeutig**

    $\Rightarrow$: Es gelte $u_{i_1}..u_{i_n} = v_{i_1}..v_{i_n}$. Dann sind

    $S \longrightarrow A \longrightarrow a_{i_1} A u_{i_1} \longrightarrow a_{i_2} a_{i_1} A u_{i_1} u_{i_2} \ldots \longrightarrow a_{i_n}..a_{i_2} a_{i_1} u_{i_1} u_{i_2}..u_{i_n}$

    $S \longrightarrow B \longrightarrow a_{i_1} B v_{i_1} \longrightarrow a_{i_2} a_{i_1} B v_{i_1} v_{i_2} \ldots \longrightarrow a_{i_n}..a_{i_2} a_{i_1} v_{i_1} v_{i_2}..v_{i_n}$

    verschiedene (Links-)Ableitungen desselben Wortes in G ✓

    $\Leftarrow$: Jedes $w \in L(G)$ besitzt nur eine Ableitung aus A bzw. aus B.

    Hat w zwei (Links-)Ableitungen in G, so muß die erste mit

    $S \longrightarrow A$ und die zweite mit $S \longrightarrow B$ beginnen.

    Es folgt $a_{i_n}..a_{i_2} a_{i_1} u_{i_1} u_{i_2}..u_{i_n} = w = a_{i_n}..a_{i_2} a_{i_1} v_{i_1} v_{i_2}..v_{i_n}$

    Also $K \in PKP$ ✓

Rückblick: Beweismethoden für Unlösbarkeit

- **Diagonalisierung**
  - Nur für maschinennah formulierte Probleme
- **Wachstums- und Monotoniegargumente**
  - Aufwendig: Busy Beaver Problem / Ackermannfunktion
- **Eigenschaften des Komplements**
  - Ist $\overline{P}$ nicht entscheidbar, dann ist auch P nicht entscheidbar
  - Ist $\overline{P}$ nicht entscheidbar aber aufzählbar, dann ist P nicht aufzählbar
- **Reduktionsbeweise**
  - Einbettung eines anderen unlösbaren Problem P_{alt} in P

    $P_{alt} = f^{-1}(P) \qquad (\text{“}P_{alt} \leq P\text{”})$
  - Ähnliches Vorgehen mit anderen Abschlußeigenschaften
- **Anwendung des Satzes von Rice**
  - Keine nichttriviale extensionale Programmeigenschaft ist entscheidbar