UNIVERSITÄT ZU LÜBECK

Theoretische Informatik
Weitere Anwendungen der Reduktionsmethode

Maciej Liśkiewicz

Institut für Theoretische Informatik

7. Januar 2014, Vorlesung 18

IM FOCUS DAS LEBEN

Wiederholung (1)

Das Schema des Beweises, dass EmptyString $= \{w \in \{0,1\}^* : \lambda \in L(M_w)\}$ nicht rekursiv ist.

- Wir wissen, dass HP nicht rekursiv ist.
- Wir finden eine Transformation $w\#x \mapsto w'$, so dass

$$L(M_{w'}) = \begin{cases} \Sigma^* & \text{wenn } M_w \text{ auf } x \text{ hält,} \\ \emptyset & \text{wenn } M_w \text{ auf } x \text{ nicht hält.} \end{cases}$$

- Dies bedeutet $w\#x \in \text{HP} \quad \Leftrightarrow \quad w' \in \text{EmptyString}$.

  Angenommen es gibt eine totale TM für die Sprache EmptyString:

  ```
  EmptyString-Solver(w)
  Input: Codewort w einer TM M_w
  Output: 'Ja', wenn λ ∈ L(M_w), 'Nein', wenn λ ∉ L(M_w).
  ```

  Algorithmus für das Halteproblem:

  ```
  HP-Solver(w#x)
  Input: Codewort w einer TM M_w und Eingabe x
  Output: 'Ja', wenn M_w auf x hält; 'Nein', wenn nicht
  1. berechne die Transformation w#x ↦ w'
  2. Antwort := EmptyString-Solver(w')
  3. if Antwort = 'Ja' then Output('Ja')
  4. if Antwort = 'Nein' then Output('Nein')
  ```

  `HP-Solver` ist total und `HP-Solver`$(w\#x)$=`'Ja'` gdw. M_w auf x hält.
  Ein Widerspruch.

Wiederholung (2)

Das Schema des Beweises, dass NonEmpty $= \{w \in \{0,1\}^* : L(M_w) \neq \emptyset\}$ nicht rekursiv ist.

- Wir wissen, dass HP nicht rekursiv ist.
- Wir benutzen eine Transformation $w\#x \mapsto w'$, so dass

$$L(M_{w'}) = \begin{cases} \Sigma^* & \text{wenn } M_w \text{ auf } x \text{ hält,} \\ \emptyset & \text{wenn } M_w \text{ auf } x \text{ nicht hält.} \end{cases}$$

- Dies bedeutet $w\#x \in \text{HP} \quad \Leftrightarrow \quad w' \in \text{NonEmpty}$.

  Angenommen es gibt eine totale TM für die Sprache NonEmpty:

  `NonEmpty-Solver`(w)

  `Input: Codewort` w `einer TM` M_w

  `Output: 'Ja' wenn` $L(M_w) \neq \emptyset$, `'Nein' wenn` $L(M_w) = \emptyset$

  Algorithmus für das Halteproblem:

  `HP-Solver`$(w\#x)$

  `Input: Codewort` w `einer TM` M_w `und Eingabe` x

  `Output: 'Ja' wenn` M_w `auf` x `hält; 'Nein' wenn nicht`

  `1. berechne die Transformation` $w\#x \mapsto w'$

  `2. Antwort := NonEmpty-Solver`(w')

  `3. if Antwort = 'Ja' then Output('Ja')`

  `4. if Antwort = 'Nein' then Output('Nein')`

  `HP-Solver` ist total und `HP-Solver`$(w\#x)$`='Ja'` gdw. M_w auf x hält.

  Ein Widerspruch.

Wiederholung (3)

Das Schema des Beweises, dass AllStrings $= \{w \in \{0,1\}^* : L(M_w) = \Sigma^*\}$ nicht rekursiv ist.

- Wir wissen, dass HP nicht rekursiv ist.
- Wir benutzen eine Transformation $w\#x \mapsto w'$, so dass

$$L(M_{w'}) = \begin{cases} \Sigma^* & \text{wenn } M_w \text{ auf } x \text{ hält,} \\ \emptyset & \text{wenn } M_w \text{ auf } x \text{ nicht hält.} \end{cases}$$

- Dies bedeutet $w\#x \in \mathrm{HP} \quad \Leftrightarrow \quad w' \in$ AllStrings.

Angenommen es gibt eine totale TM für die Sprache AllStrings:

```
AllStrings-Solver(w)
Input: Codewort w einer TM M_w
Output: 'Ja' wenn L(M_w) = Σ*, 'Nein' wenn L(M_w) ≠ Σ*
```

Algorithmus für das Halteproblem:

```
HP-Solver(w#x)
Input: Codewort w einer TM M_w und Eingabe x
Output: 'Ja' wenn M_w auf x hält; 'Nein' wenn nicht
1. berechne die Transformation w#x ↦ w'
2. Antwort := AllStrings-Solver(w')
3. if Antwort = 'Ja' then Output('Ja')
4. if Antwort = 'Nein' then Output('Nein')
```

`HP-Solver` ist total und `HP-Solver`$(w\#x)$=`'Ja'` gdw. M_w auf x hält.
Ein Widerspruch.

Wiederholung (4)

Das Schema des Beweises, dass für jede nicht triviale Eigenschaft $\mathcal{P}$ die Sprache $\{w \in \{0,1\}^* : L(M_w) \in \mathcal{P}\}$ nicht rekursiv ist (der Satz von Rice).

- Wir wissen, dass HP nicht rekursiv ist.
- Wir finden eine Transformation $w\#x \mapsto w'$, so dass

$$M_w \text{ hält auf } x \quad \Leftrightarrow \quad L(M_{w'}) \in \mathcal{P}.$$

- Dies bedeutet $w\#x \in \mathrm{HP} \quad \Leftrightarrow \quad w' \in \{w \in \{0,1\}^* : L(M_w) \in \mathcal{P}\}$.

  Angenommen es gibt eine totale TM für diese Sprache:

  $\mathcal{P}$-Property-Tester(w)
  Input: Codewort w einer TM M_w
  Output: 'Ja' wenn $L(M_w) \in \mathcal{P}$, 'Nein' wenn $L(M_w) \notin \mathcal{P}$

  Algorithmus für das Halteproblem:

  HP-Solver($w\#x$)
  Input: Codewort w einer TM M_w und Eingabe x
  Output: 'Ja' wenn M_w auf x hält; 'Nein' wenn nicht
  1. berechne die Transformation $w\#x \mapsto w'$
  2. Antwort := $\mathcal{P}$-Property-Tester(w')
  3. if Antwort = 'Ja' then Output('Ja')
  4. if Antwort = 'Nein' then Output('Nein')

  HP-Solver ist total und HP-Solver($w\#x$)='Ja' gdw. M_w auf x hält.
  Ein Widerspruch.

Reduktionsmethode: Reduktion des Halteproblems auf B

Es ist zu beweisen, dass eine Sprache $B \subseteq \Gamma^*$ nicht rekursiv ist.

- Wir wissen, dass HP nicht rekursiv ist.
- Beweismethode: Reduktion des Halteproblems auf B (HP $\leq B$):
  d.h. wir finden eine Transformation $f : w\#x \mapsto y$, so dass
  - $w\#x \in \text{HP} \quad \Leftrightarrow \quad y \in B$ und
  - die Transformation f „berechenbar" ist.

  Angenommen es gibt eine totale TM für die Sprache B:

  ```
  B-Solver(y)
  Input: Eine Instanz y ∈ Γ*
  Output: 'Ja' wenn y ∈ B, 'Nein' wenn y ∉ B
  ```

  Algorithmus für das Halteproblem:

  ```
  HP-Solver(w#x)
  Input: Codewort w einer TM M_w und Eingabe x
  Output: 'Ja' wenn M_w auf x hält; 'Nein' wenn nicht
  1. berechne y := f(w#x)
  2. Antwort := B-Solver(y)
  3. if Antwort = 'Ja' then Output('Ja')
  4. if Antwort = 'Nein' then Output('Nein')
  ```

  `HP-Solver` ist total und `HP-Solver`$(w\#x)$=`'Ja'` gdw. M_w auf x hält.
  Ein Widerspruch.

Reduktionsmethode: Reduktion des Halteproblems auf B

- Das Halteproblem ist *das* grundlegende Beispiel für ein unentscheidbares Problem in der theoretischen Informatik.
- Um die Reduktionsmethode zu verwenden, muss man nicht unbedingt das Halteproblem benutzen.
- Man kann jede beliebige Sprache A benötigen, für die wir festgestellt haben, dass sie nicht rekursiv ist.

Reduktionsmethode: Reduktion des Problems A auf B

Es ist zu beweisen, dass eine Sprache $B \subseteq \Gamma^*$ nicht rekursiv ist.

- Wir benötigen eine Sprache $A \subseteq \Sigma^*$, für die wir festgestellt haben, dass sie nicht rekursiv ist.
- Beweismethode: Reduktion $A \leq B$, d.h. wir finden eine totale Abbildung $f : \Sigma^* \to \Gamma^*$, so dass
  - $\forall x \in \Sigma^* \quad (x \in A \quad \Leftrightarrow \quad f(x) \in B)$ und
  - f „berechenbar" ist.

  Angenommen es gibt eine totale TM für die Sprache B:

  ```
  B-Solver(y)
  Input: Eine Instanz y ∈ Γ*
  Output: 'Ja' wenn y ∈ B, 'Nein' wenn y ∉ B
  ```

  Algorithmus für die Sprache A:

  ```
  A-Solver(x)
  Input: Instanz x ∈ Σ*
  Output: 'Ja' wenn x ∈ A; 'Nein' wenn x ∉ A
  1. berechne y := f(x)
  2. Antwort := B-Solver(y)
  3. if Antwort = 'Ja' then Output('Ja')
  4. if Antwort = 'Nein' then Output('Nein')
  ```

  A-Solver ist total und A-Solver(x)='Ja' gdw. $x \in A$. Ein Widerspruch.

Reduktionsmethode: Reduktion des Problems A auf B

Zur Wiederholung (Vorlesung 17):

Definition

Seien A und B zwei Sprachen $A \subseteq \Sigma^*$ und $B \subseteq \Gamma^*$. A heißt reduzierbar auf B gdw. eine totale Abbildung $f : \Sigma^* \to \Gamma^*$ existiert mit der Eigenschaften:

- $\forall x \in \Sigma^* \; (x \in A \Leftrightarrow f(x) \in B)$,
- es existiert eine totale TM, die auf der Eingabe $x \in \Sigma^*$ nach endlich vielen Schritten mit dem Ergebnis $f(x)$ auf dem Band hält.

Notation: $A \leq B$.

Zur Überlegung

Sind alle Annahmen notwendig um die Reduktionsmethode zu verwenden? D.h. warum muss

- f total sein?
- $\forall x \in \Sigma^* \; (x \in A \Leftrightarrow f(x) \in B)$?
- f „berechenbar“ sein?

Einige technische Eigenschaften der Reduktion

1. Reflexivität: für jede Sprache A gilt $A \leq A$.

   Zum Beweis benutze die identische Abbildung $f(x) = x$.

2. Transitivität: Seien A, B, C Sprachen. Dann gilt:

$$\text{aus } A \leq B \text{ und } B \leq C \text{ folgt } A \leq C.$$

Beweis. Sei $A \leq B$ mittels der Abbildung $f_1 : \Sigma^* \to \Gamma^*$ und $B \leq C$ mittels der Abbildung $f_2 : \Gamma^* \to \Phi^*$. Für die Reduktion $A \leq C$ verwende die Abbildung $f : \Sigma^* \to \Phi^*$, mit $f(x) = f_2(f_1(x))$ für alle $x \in \Sigma^*$. Es gilt, für alle x

$$x \in A \quad \Leftrightarrow \quad f_1(x) \in B \quad \Leftrightarrow \quad f_2(f_1(x)) \in C.$$

Es ist einfach zu sehen, dass f total ist und es eine totale TM existiert, die auf der Eingabe x nach endlich vielen Schritten mit dem Ergebnis $f(x)$ auf dem Band hält. $\square$

Einige technische Eigenschaften der Reduktion

3. $A \leq B$ mittels $f \quad \Leftrightarrow \quad \overline{A} \leq \overline{B}$ mittels f .

   Beweis. Offensichtlich ist die Aussage

   $$\forall x \in \Sigma^* \; (x \in A \;\Leftrightarrow\; f(x) \in B)$$

   äquivalent mit

   $$\forall x \in \Sigma^* \; (x \notin A \;\Leftrightarrow\; f(x) \notin B).$$

   Daraus folgt $\overline{A} \leq \overline{B}$ mittels f. $\square$

Nicht-Semientscheidbarkeit

Es ist zu beweisen, dass $B \subseteq \Gamma^*$ nicht rekursiv aufzählbar ist.

- Wir benötigen eine Sprache $A \subseteq \Sigma^*$, für die wir festgestellt haben, dass sie nicht rekursiv aufzählbar ist (z.B. $\overline{\text{HP}}$).
- Beweismethode: Reduktion $A \leq B$ mittels der Abbildung f.

Angenommen es gibt eine TM (nicht unbedingt total) für B:

```
B-SemiSolver(y)
Input: Eine Instanz y ∈ Γ*
Output: 'Ja' genau dann wenn y ∈ B
```

Algorithmus für die Sprache A:

```
A-SemiSolver(x)
Input: Instanz x ∈ Σ*
Output: 'Ja' genau dann wenn x ∈ A
1. berechne y := f(x)
2. Antwort := B-SemiSolver(y)
3. if Antwort = 'Ja' then Output('Ja')
```

A-SemiSolver(x)='Ja' gdw. $x \in A$. Ein Widerspruch.

Reduktionsmethode: Reduktion des Problems A auf B

Aus unseren Überlegungen zur Reduktionsmethode folgt der Satz (Vorlesung 17):

- Betrachte die Reduktion $A \leq B$.
  - Wenn B rekursiv ist, so auch A.
    Wenn A nicht rekursiv ist, so auch B.
  - Wenn B rekursiv aufzählbar ist, so auch A.
    Wenn A nicht rekursiv aufzählbar ist, so auch B.

Rezept

- Um zu zeigen, dass B nicht rekursiv ist, wähle eine geeignete als nicht rekursiv bekannte Sprache A und definiere eine Abbildung f, die A auf B reduziert: $A \leq B$.
- Um zu zeigen, dass B nicht rekursiv aufzählbar ist, wähle eine geeignete als nicht rekursiv aufzählbar bekannte Sprache A und definiere eine Abbildung f, die A auf B reduziert: $A \leq B$.

Sowohl FIN als auch $\overline{\text{FIN}}$ sind nicht rekursiv aufzählbar

Beispiel

Sei $\text{FIN} \stackrel{def}{=} \{w \in \{0,1\}^* : L(M_w) \text{ ist endlich}\}$. Die Probleme

- ob eine gegebene Turing-Maschine eine endliche Sprache akzeptiert,
- ob eine gegebene Turing-Maschine eine unendliche Sprache akzeptiert

sind nicht semientscheidbar, d.h. weder die Sprache FIN noch $\overline{\text{FIN}}$ ist rekursiv aufzählbar.

Merke

Aus dem Satz von Rice folgt, dass weder FIN noch $\overline{\text{FIN}}$ rekursiv sind. Wir zeigen mehr, nämlich dass die beiden Sprachen sogar nicht rekursiv aufzählbar sind.

Beweis durch Reduktion von $\overline{\text{HP}}$:

(1) $\overline{\text{HP}} \leq \text{FIN}$ und

(2) $\overline{\text{HP}} \leq \overline{\text{FIN}}$.

Sowohl $\textsc{fin}$ als auch $\overline{\textsc{fin}}$ sind nicht rekursiv aufzählbar

(1) $\overline{\textsc{hp}} \leq \textsc{fin}$

- Wir finden eine totale Abbildung $f : \{0, 1, \#\}^* \to \{0, 1\}^*$, so dass für alle u

$$u \in \overline{\textsc{hp}} \iff f(u) \in \textsc{fin}$$

  gilt und es eine totale TM existiert, die die Abbildung f berechnet.
- Formal, ist

$$\overline{\textsc{hp}} = \{w\#x : M_w \text{ hält nicht auf } x\} \cup \textsc{trash},$$

  wobei $\textsc{trash} = L((0+1)^* + (0+1)^*\#(0+1)^*\#(0+1+\#)^*)$ keine Wörter der Form $w\#x$ enthält, mit $w, x \in \{0, 1\}^*$.
- Für alle $u \in \textsc{trash}$, sei $f(u) = w_0$, wobei w_0 ein festes Codewort einer TM M_{w_0} ist, mit $L(M_{w_0}) = \emptyset$.

Sowohl FIN als auch $\overline{\text{FIN}}$ sind nicht rekursiv aufzählbar

(1) $\overline{\text{HP}} \leq$ FIN

- Der wesentliche Teil der Abbildung ist

$$f : w\#x \mapsto w',$$

für alle $w, x \in \{0,1\}^*$, so dass

$$M_w \text{ hält nicht auf } x \quad \Leftrightarrow \quad L(M_{w'}) \text{ endlich ist.}$$

- Wir verwenden die gleiche Konstruktion wie in der Vorlesung 16: Wir konstruieren ein Codewort $w' = f(w\#x)$, so dass $M_{w'}$ im endlichen Gedächtnis das Wort x speichert und die TM M_w als Unterprogramm verwendet und
- die Arbeitsweise von $M_{w'}$ auf Eingabe y ist wie folgt:
  1. Lösche Eingabe y.
  2. Schreibe auf dem Band das Wort x.
  3. Lasse M_w mit Eingabe x laufen.
  4. Wenn M_w hält, dann akzeptiere.

Sowohl FIN als auch $\overline{\text{FIN}}$ sind nicht rekursiv aufzählbar

(1) $\overline{\text{HP}} \leq \text{FIN}$

- Für die Abbildung
$$f : w\#x \mapsto w'$$
gilt:
- M_w hält auf $x \quad \Rightarrow \quad L(M_{w'}) = \Sigma^* \quad \Rightarrow \quad L(M_{w'})$ ist unendlich,
- M_w hält nicht auf $x \quad \Rightarrow \quad L(M_{w'}) = \emptyset \quad \Rightarrow \quad L(M_{w'})$ ist endlich.
- Es ist einfach zu sehen, dass f total ist und es eine totale TM existiert, die auf der Eingabe $w\#x$ nach endlich vielen Schritten mit dem Ergebnis $f(w\#x)$ auf dem Band hält.

$\square$

Bemerkung: die Funktion f dient als Reduktionsabbildung für viele Sprachen, die auf TM-Codewörter basieren. Das liegt daran, dass

$$L(M_{w'}) = \begin{cases} \Sigma^* & \text{wenn } M_w \text{ auf } x \text{ hält} \\ \emptyset & \text{wenn } M_w \text{ auf } x \text{ nicht hält.} \end{cases}$$

Sowohl FIN als auch $\overline{\text{FIN}}$ sind nicht rekursiv aufzählbar

(2) $\overline{\text{HP}} \leq \overline{\text{FIN}}$

- Nach der Eigenschaft 3 (Folie 12), gilt $\overline{\text{HP}} \leq \overline{\text{FIN}}$ mittels der Abbildung g dann ist auch $\text{HP} \leq \text{FIN}$ mittels g.
- So, zu finden ist eine totale Abbildung $g : \{0, 1, \#\}^* \to \{0, 1\}^*$, so dass für alle u
$$u \in \text{HP} \Leftrightarrow g(u) \in \text{FIN}$$
und g berechenbar ist.
- Für alle $u \in \text{TRASH}$, sei jetzt $g(u) = w_0$, wobei w_0 ein festes Codewort einer TM M_{w_0} ist, mit $L(M_{w_0}) = \Sigma^*$.

Sowohl FIN als auch $\overline{\text{FIN}}$ sind nicht rekursiv aufzählbar

(2) $\overline{\text{HP}} \leq \overline{\text{FIN}}$

- Der wesentliche Teil der Abbildung ist $g : w\#x \mapsto w''$, für alle $w, x \in \{0,1\}^*$, so dass

$$M_w \text{ hält auf } x \quad \Leftrightarrow \quad L(M_{w''}) \text{ endlich ist.}$$

- Für gegebenes Wort $w\#x$ konstruieren wir $w'' = g(w\#x)$, so dass die Arbeitsweise von $M_{w''}$ auf Eingabe y wie folgt ist:
  1. Speichere auf der Spur 2 das Eingabewort y.
  2. Schreibe auf dem Band das Wort x.
  3. Lasse M_w auf x für $|y|$ Schritte laufen. (simuliere die Schritte von M_w in der Standardweise und nach jedem Schritt, lösche ein Zeichen von y).
  4. Akzeptiere, wenn M_w nicht nach $|y|$ Schritten hält und verwerfe sonst.

Sowohl FIN als auch $\overline{\text{FIN}}$ sind nicht rekursiv aufzählbar

(2) $\overline{\text{HP}} \leq \overline{\text{FIN}}$

- Wenn M_w nicht hält mit Eingabe x, dann akzeptiert die TM $M_{w''}$ jede Eingabe y.
- D.h. M_w hält nicht mit $x \quad \Rightarrow \quad L(M_{w''}) = \Sigma^*$.
- Wenn M_w hält mit Eingabe x, dann hält M_w mit x nach endlich vielen Schritten (sagen wir n Schritten).
- D.h.
  - $M_{w''}$ akzeptiert jede Eingabe y mit $|y| < n$ und
  - $M_{w''}$ verwirft jede Eingabe y mit $|y| \geq n$.
- Daraus folgt:

$$\begin{array}{rcl} M_w \text{ hält mit } x & \Rightarrow & L(M_{w''}) = \{y : |y| < \text{Anzahl der Schritte von } M_w \text{ mit } x\} \\ & \Rightarrow & L(M_{w''}) \text{ ist endlich} \\ \\ M_w \text{ hält nicht mit } x & \Rightarrow & L(M_{w''}) = \Sigma^* \\ & \Rightarrow & L(M_{w''}) \text{ ist unendlich} \end{array}$$

- Es ist einfach zu sehen, dass g total ist und es eine totale TM existiert, die auf $w\#x$ nach endlich vielen Schritten mit dem Ergebnis $g(w\#x)$ auf dem Band hält.

$\square$

Weitere nicht rekursiv aufzählbare Sprachen

Beispiel

Sei $\textsc{total} \stackrel{def}{=} \{w \in \{0,1\}^* : M_w \text{ hält bei allen Eingaben}\}$. Das Problem

- ob eine gegebene Turing-Maschine auf allen Eingaben hält

ist nicht semientscheidbar. Auch das Komplement des Problems ist nicht semientscheidbar. Das heißt, weder $\textsc{total}$ noch $\overline{\textsc{total}}$ ist rekursiv aufzählbar.

Beweis durch Reduktion von $\overline{\textsc{hp}}$ (Übung).

Literaturangaben

Empfohlene Literatur zum Weiterlesen

1. D. Kozen, *Automata and Computability*, Springer, 1997.
   Kap. 33, 34
2. J. Hopcroft, R. Motwani, J. Ullman, *Einführung in die Automatentheorie, Formale Sprachen und Komplexitätstheorie*, Addison-Wesley, 2002.
   Kap. 9.3