UNIVERSITÄT ZU LÜBECK

Theoretische Informatik

Reduktionsmethode und der Satz von Rice

Maciej Liśkiewicz

Institut für Theoretische Informatik

17. Dezember 2013, Vorlesung 17

IM FOCUS DAS LEBEN

Plan für Heute

- Weitere Beispiele für *entscheidbare* und *unentscheidbare* Probleme.
- *Reduktion*: eine Beweismethode für Sätze über Unentscheidbarkeit.
- *Der Satz von Rice*: *Alle* nicht-trivialen Eigenschaften der rekursiv aufzählbaren Sprachen sind unentscheidbar!

1. Entscheidbare und Unentscheidbare Probleme

Turing-Maschine als Eingabe

Entscheidbare und Unentscheidbare Probleme

Turingmaschine als Eingabe

Einige Beispiele für Entscheidungsprobleme mit Turing-Maschinen als Eingaben. Ist es entscheidbar ob eine gegebene Turing-Maschine M

(1) 531 Zustände hat?

(2) auf dem leeren Wort λ mindestens 532 Schritte macht?

(3) auf jeder Eingabe mindestens 532 Schritte macht?

(4) das leere Wort λ akzeptiert?

(5) mindestens ein Wort akzeptiert?

(6) alle Wörter akzeptiert?

(7) unendlich viele Wörter akzeptiert?

(8) eine reguläre Sprache akzeptiert?

(9) eine kontextfreie Sprache akzeptiert?

(10) eine rekursive Sprache akzeptiert?

Entscheidbare und Unentscheidbare Probleme

Turingmaschine als Eingabe

- Es ist nicht schwer zu zeigen, dass Probleme 1–3 entscheidbar sind.
- Probleme 4–10 sind unentscheidbar.
- Die Beweismethode: Man zeigt, dass die Entscheidbarkeit jedes der Probleme 4–10 würde die Entscheidbarkeit des Halteproblems impliziert.
- Wir zeigen auch, dass die Unentscheidbarkeit der Probleme eigentlich Spezialfall eines weit allgemeineren Satzes von Rice ist.

Wir starten mit dem Problem 4. Sei $\text{EmptyString} \stackrel{def}{=} \{w \in \{0,1\}^* : \lambda \in L(M_w)\}$.

Satz

Das Problem, ob eine gegebene Turing-Maschine das leere Wort akzeptiert ist unentscheidbar, d.h. die Sprache EmptyString ist nicht rekursiv.

Beweis

- Nehmen wir an es existiert eine totale Turing-Maschine für die Sprache EmptyString.
- Wir zeigen, dass man mit Hilfe dieser Maschine das Halteproblem entscheiden kann.
- Wir bekommen einen Widerspruch, weil das Halteproblem unentscheidbar ist.

Beweis (Fortsetzung)

- Sei $w\#x$ das Eingabewort für das Halteproblem.
- Zur Erinnerung: w ist ein Codewort einer TM M_w, x ist das Eingabewort und $w\#x \in \text{HP}$ gdw. M_w bei x anhält.
- Wir modifizieren die Eingabe $w\#x \mapsto w'$, wobei w' das Codewort einer TM $M_{w'}$ ist.
- Unser Ziel ist eine Modifikation $w\#x \mapsto w'$ zu finden, so dass

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

äquivalent: $w\#x \in \text{HP} \quad \Leftrightarrow \quad w' \in \text{EmptyString}$.

Beweis (Fortsetzung)

- Wir modifizieren $w\#x \mapsto w'$, so dass $M_{w'}$ die TM M_w als Unterprogramm verwendet. Zusätzlich speichert $M_{w'}$ im endlichen Gedächtnis das Wort x.
- Die Arbeitsweise von $M_{w'}$ auf Eingabe y :
  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, so akzeptiere.
- Merke: Bei jeder Eingabe y macht $M_{w'}$ das gleiche
  - wenn M_w bei x anhält, hält auch $M_{w'}$ und akzeptiert das Wort y.
- Unsere Modifikation $w\#x \mapsto w'$ hat also folgende Eigenschaft:

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

Beweis (Fortsetzung)

- Nun verwenden wir unsere Annahme, dass eine totale Turing-Maschine M für die Sprache EmptyString existiert.
- Eine TM für das Halteproblem führt bei der Eingabe $w\#x$ den folgenden Algorithmus aus:
  1. Berechne die Modifikation: $w\#x \mapsto w'$.
  2. Führe M mit Eingabe w' aus.
  3. Akzeptiere wenn M bei w' akzeptiert.
- Merke: (1) Die TM ist total; (2) Es existiert so eine TM, weil es möglich ist eine TM zu spezifizieren, die die Modifikation $w\#x \mapsto w'$ berechnet.
- Wäre also das Problem EmptyString entscheidbar, dann wäre auch das Halteproblem entscheidbar.
- Ein Widerspruch.

$\square$

Entscheidbare und Unentscheidbare Probleme

Man kann den gleichen Beweis benutzen um zu zeigen, dass die Probleme, ob eine gegebene Turing-Maschine

(5) mindestens ein Wort akzeptiert?

(6) alle Wörter akzeptiert?

(7) unendlich viele Wörter akzeptiert?

unentscheidbar sind. Zur Erinnerung, die Modifikation $w\#x \mapsto w'$ hat folgende Eigenschaft:

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

Wäre das Problem 5, 6 oder 7 entscheidbar, dann benutzen wir eine totale TM M_i für das Problem $i = 5, 6$ oder 7 und verwenden den folgenden Algorithmus um zu entscheiden, ob $w\#x \in \text{HP}$:

1. Berechne die Modifikation: $w\#x \mapsto w'$.
2. Führe M_i mit Eingabe w' aus.
3. Akzeptiere wenn M bei w' akzeptiert.

2. Reduktion

Reduktion

- Wir haben zwei Methoden benutzt, um zu zeigen, dass ein Problem unentscheidbar ist: Diagonalisierung und Reduktion.
- Die Idee der Reduzierbarkeit: ein Verfahren zur Entscheidung des Problems B in das neue Verfahren zur Entscheidung des Problems A effektiv zu konvertieren.

Reduktion

- Nachdem wir festgestellt haben, dass HP unentscheidbar ist, kann man Instanzen $w\#x$ des Problems HP modifizieren um Instanzen $f(w\#x)$ des Problems A zu erzeugen, so dass

$$w\#x \in \text{HP} \quad \Leftrightarrow \quad f(w\#x) \in A.$$

- Falls es möglich ist mit Hilfe einer TM die Transformation f zu berechnen, kann man den folgenden Algorithmus benutzen um das HP zu entscheiden.
  1. Berechne die Modifikation: $f(w\#x)$.
  2. Führe TM M für die Sprache A mit Eingabe $f(w\#x)$ aus.
  3. Akzeptiere wenn M bei $f(w\#x)$ akzeptiert.
- Dies impliziert, dass A auch unentscheidbar sein muss.
- Wir verwenden dann die Notation: $\text{HP} \leq A$.

Reduktion

Formal definiert man die Reduktion wie folgt.

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 den 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: $A \leq B$.

Reduktion

Beispiel

Im Beweis des Satzes, dass es unentscheidbar ist, ob eine gegebene Turing-Maschine das leere Wort akzeptiert, haben wir das Codewort w einer TM M_w und die Eingabe x modifiziert, so dass die resultierende TM $M_{w'}$ das leere Wort akzeptiert gdw. $w\#x \in \text{HP}$. Wir haben

$$
\begin{aligned}
A &= \{w\#x : w, x \in \{0,1\}^* \text{ und } M_w \text{ hält auf } x\} \\
B &= \{w \in \{0,1\}^* : \lambda \in M_w\}.
\end{aligned}
$$

Reduktion

„Gute und schlechte Neuigkeiten“

Satz

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.

Reduktion

„Gute und schlechte Neuigkeiten“

Beispiel

- Um zu beweisen, dass es unentscheidbar ist, ob eine gegebene Turing-Maschine das leere Wort akzeptiert, haben wir gezeigt:

$$\text{HP} \leq \text{EmptyString} = \{w \in \{0,1\}^* : \lambda \in L(M_w)\}.$$

- Man kann die gleiche Modifikation $w\#x \mapsto w'$ benutzen um zu beweisen, dass

$$\overline{\text{HP}} \leq \{w \in \{0,1\}^* : L(M_w) = \emptyset\}.$$

Damit ist die Sprache $\{w : L(M_w) = \emptyset\}$ nicht rekursiv aufzählbar.

3. Der Satz von Rice

Unentscheidbarkeit ist die Regel, nicht die Ausnahme.

Der Satz von Rice

- Es ist unentscheidbar, ob eine gegebene Turing-Maschine die leere Sprache akzeptiert.
- Dies ist ein Spezialfall des allgemeinen Prinzips, dass alle nicht-trivialen Eigenschaften der rekursiv aufzählbaren Sprachen unentscheidbar sind.

Definition

Eine Eigenschaft $\mathcal{P}$ der rekursiv aufzählbaren Sprachen ist eine Teilmenge von RE:

$$\mathcal{P} \subseteq \text{RE}.$$

Beispiel

- Eigenschaft $\mathcal{P}$: „Sprache ist leer“

$$\mathcal{P} = \{\emptyset\}$$

- Eigenschaft $\mathcal{P}$: „Sprache ist regulär“

$$\mathcal{P} = \text{REG}.$$

Der Satz von Rice

Eigenschaften der rekursiv aufzählbaren Sprachen

- Unser Ziel ist zu testen ob eine Eigenschaft $\mathcal{P}$ entscheidbar oder unentscheidbar ist.
- Das heißt, wir untersuchen das folgende Entscheidungsproblem:
  - Gegeben eine rekursiv aufzählbare Sprache A;
  - Bestimme, ob $A \in \mathcal{P}$, d.h. ob die Eigenschaft $\mathcal{P}$ auf die Sprache A zutrifft oder nicht.
- Die Eingabesprache A wird als Codewort w einer TM spezifiziert, die diese Sprache charakterisiert, d.h. $L(M_w) = A$.
- Merke
  - $\mathcal{P}$ ist eine Eigenschaft von $L(M_w) = A$ und nicht von M_w.
  - Dies bedeutet, dass $A \in \mathcal{P}$ unabhängig davon ist welche TM wir wählen um A zu spezifizieren.

Wir untersuchen also, ob die Sprache

$$\{w \in \{0,1\}^* : L(M_w) \in \mathcal{P}\}$$

rekursiv ist.

Zur Erinnerung

- Eine Eigenschaft P von Strings (bzw. ein Entscheidungsproblem P) heißt entscheidbar, wenn die Sprache $\{x \in \{0,1\}^* : P(x)\}$ rekursiv ist.

Wir untersuchen also, ob die Sprache

$$\{w \in \{0,1\}^* : L(M_w) \in \mathcal{P}\}$$

rekursiv ist.

Beispiel

- Ist die Sprache $\{w \in \{0,1\}^* : L(M_w) = \textsc{prime}\}$ rekursiv?
- Anders gesagt: Ist das folgende Problem entscheidbar?
  - Gegeben ein C++ Programm Π;
  - Bestimme, ob Π einen Primzahltest *korrekt* implementiert, d.h. ob Π für jede natürliche Zahl n die Antwort „Ja" gibt wenn n eine Primzahl und „Nein", wenn n keine Primzahl ist.

Wir untersuchen also, ob die Sprache

$$\{w \in \{0,1\}^* : L(M_w) \in \mathcal{P}\}$$

rekursiv ist.

Weitere Beispiele

Sind die folgenden Sprachen rekursiv?

- $\{w \in \{0,1\}^* : L(M_w) = \{x : x \text{ ist ein Palindrom}\}\}$.
- $\{w \in \{0,1\}^* : L(M_w) = \emptyset\}$.
- $\{w \in \{0,1\}^* : L(M_w) = \Sigma^*\}$.
- $\{w \in \{0,1\}^* : L(M_w) \in \textsc{reg}\}$.
- $\{w \in \{0,1\}^* : L(M_w) \in \textsc{cfl}\}$.

Wir untersuchen also, ob die Sprache

$$\{w \in \{0,1\}^* : L(M_w) \in \mathcal{P}\}$$

rekursiv ist.

Merke

Beispiele für Eigenschaften von Turing-Maschinen, die jedoch keine Eigenschaften von rekursiv aufzählbaren Sprachen sind:

- M_w hat 531 Zustände.
- M_w hält auf dem leeren Wort λ.
- M_w hält auf dem Wort w.
- M_w hält auf allen Eingaben $x \in \Sigma^*$.
- Es existiert eine TM $M_{w'}$, mit $|w'| < |w|$, die äquivalent zu M_w ist, d.h. $L(M_{w'}) = L(M_w)$.

Der Satz von Rice sagt:

- Jede nicht triviale Eigenschaft der rekursiv aufzählbaren Sprachen ist unentscheidbar.

Definition

Eine Eigenschaft der rekursiv aufzählbaren Sprachen $\mathcal{P}$ ist nicht trivial wenn es eine Sprache A gibt mit $A \in \mathcal{P}$ und eine andere A' mit $A' \notin \mathcal{P}$.

Bemerkung

- Es gibt nur zwei Eigenschaften, die trivial sind, und diese Eigenschaften sind offensichtlich entscheidbar.

Der Satz von Rice

Der Satz von Rice

Jede nicht triviale Eigenschaft $\mathcal{P}$ der rekursiv aufzählbaren Sprachen ist unentscheidbar, d.h. die Sprache

$$\{w \in \{0,1\}^* : \; L(M_w) \in \mathcal{P}\}$$

ist nicht rekursiv.

Der Satz von Rice

und sein Beweis

Die Beweismethode: Reduktion.

- Nehmen wir an die Sprache $\{w \in \{0,1\}^* : L(M_w) \in \mathcal{P}\}$ ist rekursiv.
- Wir werden das Halteproblem HP auf die Sprache $\{w \in \{0,1\}^* : L(M_w) \in \mathcal{P}\}$ reduzieren.
- Wir betrachten zwei Fälle:
  - $\emptyset \notin \mathcal{P}$ und
  - $\emptyset \in \mathcal{P}$.

Der Satz von Rice

und sein Beweis

Fall: $\emptyset \notin \mathcal{P}$.

- Unser Ziel ist einen Algorithmus für die Reduktion $w\#x \mapsto w'$ zu konstruieren, so dass

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

äquivalent: $w\#x \in \text{HP} \quad \Leftrightarrow \quad w' \in \{w \in \{0,1\}^* : L(M_w) \in \mathcal{P}\}$

- Weil $\mathcal{P}$ nicht-trivial ist, existiert ein $A \in \text{RE}$ mit $A \in \mathcal{P}$.
- Sei N eine Turing-Maschine für A, d.h. $L(N) = A$.
- Wir verwenden $w\#x$ und N um w' zu konstruieren.

Der Satz von Rice
und sein Beweis

Fall: $\emptyset \notin \mathcal{P}$.

- Der Reduktionsalgorithmus mit Eingabe $w\#x$ generiert Codewort w' der Turing-Maschine $M_{w'}$.
- $M_{w'}$ verwendet die TMs M_w und N als Unterprogramme.
- Zusätzlich speichert $M_{w'}$ im endlichen Gedächtnis das Wort x.
- $M_{w'}$ verwendet ein 2-spuriges Band.
- Die Arbeitsweise von $M_{w'}$ auf Eingabe y :
  1. Sie speichert auf der Spur 2 das Eingabewort y.
  2. Schreibt auf der Spur 1 das Wort x.
  3. Führt M_w mit Eingabe x aus.
  4. Wenn M_w bei x anhält, $M_{w'}$ führt N mit Eingabe y und akzeptiert, wenn N bei y akzeptiert.

Der Satz von Rice

und sein Beweis

Fall: $\emptyset \notin \mathcal{P}$.

- M_w hält auf x oder hält nicht.
- Es gilt:

$$\begin{aligned} M_w \text{ hält auf } x \quad &\Rightarrow \quad L(M_{w'}) = A \quad \Rightarrow \quad L(M_{w'}) \in \mathcal{P} \\ M_w \text{ hält auf } x \text{ nicht} \quad &\Rightarrow \quad L(M_{w'}) = \emptyset \quad \Rightarrow \quad L(M_{w'}) \notin \mathcal{P} \end{aligned}$$

Das heißt M_w hält auf $x \Leftrightarrow L(M_{w'}) \in \mathcal{P}$ und wäre die Eigenschaft $\mathcal{P}$ entscheidbar, dann wäre auch das Halteproblem entscheidbar. Ein Widerspruch.

Fall: $\emptyset \in \mathcal{P}$.

- In diesem Fall sehen wir uns die Komplementeigenschaft $\overline{\mathcal{P}}$ an:

$$\{w \in \{0,1\}^* : L(M_w) \notin \mathcal{P}\}.$$

- Nach der vorstehenden Aussage ist $\overline{\mathcal{P}}$ unentscheidbar, das heißt die Sprache $\{w \in \{0,1\}^* : L(M_w) \notin \mathcal{P}\}$ ist nicht rekursiv.
- Wäre die Sprache $\{w \in \{0,1\}^* : L(M_w) \in \mathcal{P}\}$ rekursiv, dann wäre auch das Komplement

$$\{0,1\}^* - \{w \in \{0,1\}^* : L(M_w) \in \mathcal{P}\} \quad = \quad \{w \in \{0,1\}^* : L(M_w) \notin \mathcal{P}\}$$

  rekursiv (warum?).
- Ein Widerspruch.

$\square$

Literaturangaben

Der Satz von Rice stammt aus [1] und [2].

1. Henry Gordon Rice: *Classes of Recursively Enumerable Sets and Their Decision Problems*, Transactions of the American Mathematical Society (American Mathematical Society) 74(2): 358–366, 1953
2. Henry Gordon Rice: *On Completely Recursively Enumerable Classes and Their Key Arrays*, J. Symb. Log. 21(3): 304-308, 1956

Empfohlene Literatur zum Weiterlesen

3. D. Kozen, *Automata and Computability*, Springer, 1997.
   Kap. 32, 33, 34
4. J. Hopcroft, R. Motwani, J. Ullman, *Einführung in die Automatentheorie, Formale Sprachen und Komplexitätstheorie*, Addison-Wesley, 2002.
   Kap. 9.3