Theoretische Informatik

Teil 8
Entscheidbarkeit

Frühlingssemester 2024 (Version: 25. Januar 2024)

E. Bazzi
L. R. Büchi
D. Flumini
O. Stern

Überblick Berechenbarkeit

Teil 1 - Modelle der Berechenbarkeit

- Church-Turing-These und der Berechenbarkeitsbegriff
- Ansätze zur Formalisierung des Berechenbarkeitsbegriffes
  - Rekursive und primitiv rekursive Funktionen
  - LOOP und WHILE berechenbare Funktionen.
  - Turing-berechenbare Funktionen
- Die verschiedenen Ansätze im Vergleich
  - LOOP Berechenbarkeit und primitiv rekursive Funktionen
  - Turingvollständigkeit
  - Ackermannfunktion
  - LOOP-Interpreter

Überblick Berechenbarkeit

Teil 2 - Entscheidbarkeit und Semi-Entscheidbarkeit

- Entscheidbarkeit
  - Abschlusseigenschaften entscheidbarer Mengen
  - Reduktionen von entscheidbaren Mengen
- Semi-Entscheidbarkeit
  - Charakterisierung von Entscheidbarkeit durch semi-Entscheidbarkeit
  - Rekursiv aufzählbare Mengen
  - Reduktionen von semi-entscheidbaren Mengen
  - Das Halteproblem
- Satz von Rice

(Semi-) Entscheidbarkeit

Definition (Entscheidbarkeit)

Eine Sprache $A \subset \Sigma^*$ heisst **entscheidbar**, wenn eine Turingmaschine T existiert, die das Entscheidungsproblem (Σ, A) löst.

Bemerkung

- Ist eine Sprache $A \subset \Sigma^*$ entscheidbar, dann gibt es eine Turingmaschine T, die sich wie folgt verhält:
  - Wenn T mit Bandinhalt $x \in A$ gestartet wird, dann hält T nach endlich vielen Schritten mit Bandinhalt "1"(Ja) an.
  - Wenn T mit Bandinhalt $x \in \Sigma^* \setminus A$ gestartet wird, dann hält T nach endlich vielen Schritten mit Bandinhalt "0" (Nein) an.
- Insbesondere muss die Turingmaschine T bei jeder Eingabe $x \in \Sigma^*$ nach endlich vielen Schritten halten.

(Semi-) Entscheidbarkeit

Definition (Semi-Entscheidbarkeit)

Eine Sprache $A \subset \Sigma^*$ heisst **semi-entscheidbar**, wenn eine Turingmaschine T existiert, die sich wie folgt verhält:

- Wenn T mit Bandinhalt $x \in A$ gestartet wird, dann hält T nach endlich vielen Schritten mit Bandinhalt "1"(Ja) an.
- Wenn T mit Bandinhalt $x \in \Sigma^* \setminus A$ gestartet wird, dann hält T nie an.

Bemerkung

Informell kann man sagen, dass zu einer semi-entscheidbaren Sprache A eine Turingmaschine existiert, die zum Entscheidungsproblem (Σ, A) nur die positiven ("Ja") Antworten liefert und anstelle von negativen Antworten ("Nein") gar keine Antwort zurückgibt.

(Semi-) Entscheidbarkeit

Bemerkung (Konvention)

Wie bereits erwähnt werden natürliche Zahlen mit ihrer Binärdarstellung identifiziert.

Eine Teilmenge $X \subset \mathbb{N}$ betrachten wir also genau dann als (semi-) entscheidbar, wenn die Sprache

$$\{bin(x) \mid x \in X\} \subset \{0,1\}^*$$

(semi-) entscheidbar ist.

(Semi-) Entscheidbarkeit

Bemerkung (Folgerungen in Bezug auf Turing-vollständigkeit)

- Eine Sprache $A \subset \Sigma^*$ ist genau dann entscheidbar, wenn das Entscheidungsproblem (Σ, A) mit einem WHILE-Programm gelöst werden kann.

  Ein solches WHILE-Programm nennen wir ein **Entscheidungsverfahren** für A.

- Eine Sprache $A \subset \Sigma^*$ ist genau dann semi-entscheidbar, wenn ein WHILE-Programm existiert, das bei Eingabe von einem zu A gehörenden Wort stets terminiert und “Ja” zurückgibt und bei Eingabe von Wörtern, die nicht zu A gehören, nicht terminiert.

  Ein solches WHILE-Programm nennen wir ein **semi-Entscheidungsverfahren** für A.

(Semi-) Entscheidbarkeit

Beispiele

- Die Menge aller geraden natürlichen Zahlen ist entscheidbar, da die Funktion, $F(x) = \text{mod}_2(x+1)$, berechenbar ist.
- Die Menge aller Primzahlen ist entscheidbar, da folgender Pseudocode das entsprechende Entscheidungsproblem löst:

```
INPUT(n)
FOR i = 2 to n-1 DO
  IF Mod(n,i) = 0 THEN return 0
END
return 1
```

Bemerkung

Der Befehl “return” beendet die Ausführung der Schleife.

(Semi-) Entscheidbarkeit

Satz

Jede entscheidbare Sprache ist auch semi-entscheidbar.

Aufgabe

Beweisen Sie den Satz:

(Semi-) Entscheidbarkeit

Satz

Eine Sprache $A \subset \Sigma^*$ ist genau dann entscheidbar, wenn sowohl A als auch $\overline{A}$ semi-entscheidbar ist.

Bemerkung

Der Ausdruck $\overline{A}$ steht für das Komplement von A in Σ^*:

$$\overline{A} = \Sigma^* \setminus A = \{w \in \Sigma^* \mid w \notin A\}$$

(Semi-) Entscheidbarkeit

Beweis (⇒).

- Wenn wir ein Entscheidungsverfahren für die Menge A haben, dann ist A gemäss dem vorherigen Satz auch semi-entscheidbar.
- Wenn wir ein Entscheidungsverfahren für die Menge A haben, dann erhalten wir durch Verneinung der Ausgabe auch ein Entscheidungsverfahren für $\overline{A}$.

  Somit ist mit jeder entscheidbaren Sprache A auch das Komplement $\overline{A}$ entscheidbar und somit auch semi-entscheidbar.

(Semi-) Entscheidbarkeit

Fortsetzung Beweis ($\Leftarrow$).

Wir müssen zeigen, dass für jede semi-entscheidbare Sprache mit semi-entscheidbarem Komplement ein Entscheidungsverfahren existiert. Dies erreichen wir durch folgenden Algorithmus (Pseudocode):

```
INPUT(w)
n = 0;
WHILE true DO
    n = n + 1;
    IF A(w,n) THEN return 1;
    IF B(w,n) THEN return 0
END
```

$\square$

Anmerkungen: A(w,n) bedeutet, dass das semi-Entscheidungsverfahren von A nach n Schritten terminiert; B(w,n): Wie A(w,n) aber mit der Komplementärmenge von A.

(Semi-) Entscheidbarkeit

Satz (Abschlusseigenschaften)

- *Ist* $A \subset \Sigma^*$ *eine entscheidbare Sprache, dann ist auch* $\overline{A}$ *eine entscheidbare Sprache.*
- *Sind* A, B *(semi-) entscheidbare Sprachen, dann sind auch* $A \cup B$ *und* $A \cap B$ *(semi-) entscheidbare Sprachen.*

Beweis.

Die erste Tatsache folgt sofort aus dem soeben bewiesenen Satz. Die zweite Behauptung ist als Übung zu beweisen. □

(Semi-) Entscheidbarkeit

Satz (Charakterisierungen)

Folgende Aussagen für $A \subset \Sigma^*$ sind äquivalent:

- A *ist rekursiv aufzählbar.*
- A *ist semi-entscheidbar*[1].
- A *ist der Wertebereich einer totalen berechenbaren Funktion.*
- A *ist der Definitionsbereich einer berechenbaren Funktion.*

[1] Hält genau für alle $w \in A$

Reduktion

Wir betrachten zwei **Entscheidungsprobleme**:

Problem P_1

Gegeben: Eine natürliche Zahl x.
Gefragt: Ist x eine Primzahl?

Problem P_2

Gegeben: Ein Paar (x, y) von natürlichen Zahlen.
Gefragt: Ist x der kleinste Primfaktor von y?

Reduktion

Fragestellung: Können wir ein Lösungsverfahren vom Problem P_2 auch dazu verwenden das Problem P_1 zu lösen?

Ansatz: Für jede natürliche Zahl x gilt:

$$x \text{ erfüllt } P_1 \Leftrightarrow (x, x) \text{ erfüllt } P_2.$$

Die Frage ob x zu P_1 gehört, lässt sich also auf die Frage **reduzieren** ob das Paar (x, x) zu P_2 gehört.

Bemerkung

Offenbar können wir jede Instanz des Problems P_1 zu einer (gleichwertigen) Instanz des Problems P_2 umformulieren. Solch eine Umformulierung nennt man eine **Reduktion** von P_1 auf P_2.

Reduktion

Definition

Eine Sprache $A \subset \Sigma^*$ heisst auf eine Sprache $B \subset \Gamma^*$ **reduzierbar**, wenn es eine totale, Turing-berechenbare Funktion $F : \Sigma^* \to \Gamma^*$ gibt, so dass für alle $w \in \Sigma^*$

$$w \in A \Longleftrightarrow F(w) \in B$$

gilt. Ist die Sprache A auf die Sprache B reduzierbar, dann schreiben wir $A \preceq B$.

Reduktion

Aufgabe (Weitere einfache Beispiele zur Reduktion)

a) P_1: Gegeben ist eine Zahl n. Frage: Ist n durch 3 teilbar?
   P_2: Gegeben ist eine Zahl n. Frage: Ist n durch 6 teilbar?

b) P_1: Gegeben sind die Zahlen n und x. Ist x die Quadratwurzel von n?
   P_2: Gegeben sind die Zahlen n, x, y. Ist n das Produkt von x und y?

Geben Sie $F(w)$ an, so dass P_1 auf P_2 reduziert werden kann.

Reduktion

Satz (Transitivität)

Für beliebige Sprachen A, B und C gilt

$$A \preceq B \text{ und } B \preceq C \;\Rightarrow\; A \preceq C.$$

Beweis.

Dies folgt aus der Tatsache, dass die Komposition (Einsetzung) von totalen, Turing-berechenbaren Funktionen total, Turing-berechenbar ist. $\square$

Reduktion

Satz

Für beliebige Sprachen $A \subset \Sigma^*$ und $B \subset \Gamma^*$ gilt:

- *Ist B entscheidbar und $A \preceq B$, dann ist auch A entscheidbar.*
- *Ist B semi-entscheidbar und $A \preceq B$, dann ist auch A semi-entscheidbar.*

Beweis.

Wir gehen von einem Entscheidungsverfahren `P` für B und einer totalen berechenbaren Funktion $F : \Sigma^* \to \Gamma^*$ mit

$$w \in A \Longleftrightarrow F(w) \in B$$

aus. Wir müssen auf dieser Grundlage ein Entscheidungsverfahren für A angeben.

Reduktion

Fortsetzung Beweis.

Da die Funktion F berechenbar ist, können wir sie in unserem Entscheidungsverfahren aufrufen (Pseudocode).

```
INPUT(w)
u = F(w)
RETURN P(u)
```

Aus der Totalität von F folgt, dass dieses Programm das Entscheidungsproblem (Σ, A) löst. Somit ist A entscheidbar.

Der Beweis von der zweiten Behauptung geht analog. $\square$

Bemerkung (Erinnerung)

- Wir ordnen jeder Turingmaschine einen Code aus $w \in \{0,1\}^*$ zu.
- Für jeden Code $w \in \{0,1\}^*$ sei T_w die Turing-Maschine mit Code w.
- Es sei M eine beliebige[2] aber feste Turing-Maschine. Für alle Wörter $w \in \{0,1\}^*$, die nicht Code einer Turing-Maschine sind, setzen wir $T_w = M$. Somit ist jedes Binärwort der Code einer Turing-Maschine.

Bemerkung (Konvention)

Ist T eine Turing-Maschine mit Code w, dann schreiben wir für die von T berechnete Funktion auch F_w.

[2] Z.B. $M = (\{q_0\}, \{0\}, \{0, \sqcup\}, \varnothing, q_0, \sqcup, \varnothing)$

Das Halteproblem

Im Rahmen des **allgemeinen Halteproblems** "wird gefragt", ob eine gegebene Turingmaschine auf einem gegebenen Input anhält. Das **allgemeine Halteproblem** kann man wie folgt als Entscheidungsproblem formulieren:

Allgemeines Halteproblem H

Gegeben: Der Code $w \in \{0,1\}^*$ einer Turing-Maschine T_w und ein Input x.

Gefragt: Hält die Turing-Maschine T_w an, wenn man sie auf x ansetzt?

Das Halteproblem

Das **allgemeine Halteproblem** als Sprache formuliert.

Definition (Das allgemeine Halteproblem)

Das **allgemeine Halteproblem** ist die Sprache

$$H := \{w\#x \in \{0,1,\#\}^* \mid T_w \text{ angesetzt auf } x \text{ hält}\}.$$

Bemerkung

Die Funktion des Zeichens $\#$ ist das Trennen des Inputstrings in zwei Inputs.

Das Halteproblem

Beim **leeren Halteproblem** ist man bloss darin interessiert, ob eine gegebene Turingmaschine auf dem leeren Band anhält. Das **Halteproblem auf leerem Band** kann man wie folgt als Entscheidungsproblem formulieren:

Halteproblem auf leerem Band H_0

Gegeben: Der Code $w \in \{0,1\}^*$ einer Turing-Maschine T_w.

Gefragt: Hält die Turing-Maschine T_w an, wenn man sie auf das leere Band ansetzt?

Das Halteproblem

Das **Halteproblem auf leerem Band** als Sprache formuliert.

Definition (Das leere Halteproblem)

Das **leere Halteproblem** ist die Sprache

$$H_0 := \{w \in \{0,1\}^* \mid T_w \text{ angesetzt auf das leere Band hält}\}.$$

Das Halteproblem

Das **spezielle Halteproblem**, auch Selbstanwendungsproblem genannt, ist der Spezialfall des allgmeinen Halteproblems bei dem der Inputstring gerade dem Code der gegebenen Turingmaschine entspricht. Das **spezielle Halteproblem** kann man wie folgt als Entscheidungsproblem formulieren:

Spezielles Halteproblem H_S

Gegeben: Der Code $w \in \{0, 1\}^*$ einer Turing-Maschine T_w.

Gefragt: Hält die Turing-Maschine T_w an, wenn man sie auf ihren eigenen Code w (als Input) ansetzt?

Das Halteproblem

Das **spezielle Halteproblem** als Sprache formuliert.

Definition (Das spezielle Halteproblem)

Das **spezielle Halteproblem** ist die Sprache

$$H_S := \{w \in \{0,1\}^* \mid T_w \text{ angesetzt auf } w \text{ hält}\}.$$

Bemerkung

Das spezielle Halteproblem H_S wird auch als **Selbstanwendungsproblem** bezeichnet.

Das Halteproblem

Im folgenden werden wir die Unentscheidbarkeit der eingeführten Halteprobleme beweisen. Wir gehen dabei wie folgt vor:

1. Wir zeigen, dass H_S nicht entscheidbar ist.
2. Wir reduzieren H_S auf H und folgern daraus, dass H nicht entscheidbar ist (siehe Bemerkung unten).
3. Wir reduzieren H auf H_0 und folgern daraus, dass H_0 nicht entscheidbar ist.

Bemerkung

Lässt sich ein unentscheidbares Problem A auf ein Problem B reduzieren, i.e. gilt $A \preceq B$, dann ist auch das Problem B unentscheidbar.

Satz (Unentscheidbarkeit von H_S)

Das spezielle Halteproblem ist nicht entscheidbar.

Widerspruchsbeweis.

Wir nehmen an, dass es eine Turing-Maschine T das Halteproblem H_S entscheidet. Wir konstruieren, ausgehend von T, eine neue Turing-Maschine P.

Die TM P:

$$\begin{tikzcd}
\boxed{\text{TM } T} \arrow[r] & \boxed{\text{Bandinhalt?}} \arrow[r, "0"] \arrow[loop below, "1"] & \boxed{\text{Band} := 1} \arrow[r] & \boxed{\text{Halt}}
\end{tikzcd}$$

Das Halteproblem

Fortsetzung Beweis.

Nun sei w der Code der Turing-Maschine P, i.e. $T_w = P$. Aus der Konstruktion von P erhalten wir

$$P \text{ angesetzt auf } w \text{ hält } \Leftrightarrow T(w) = 0.$$

Weil T das spezielle Halteproblem entscheidet, erhalten wir auch

$$\begin{aligned} T(w) = 0 &\Leftrightarrow T_w \text{ angesetzt auf } w \text{ hält nicht} \\ &\Leftrightarrow P \text{ angesetzt auf } w \text{ hält nicht} \end{aligned}$$

und damit den gesuchten Widerspruch. $\square$

Das Halteproblem

- Wir zeigen, dass H_S nicht entscheidbar ist.$\checkmark$
- Wir reduzieren H_S auf H und folgern daraus, dass H nicht entscheidbar ist.
- Wir reduzieren H auf H_0 und folgern daraus, dass H_0 nicht entscheidbar ist.

Das Halteproblem

Theorem (Unentscheidbarkeit von H)

Das allgemeine Halteproblem ist nicht entscheidbar.

Beweis.

Offensichtlich ist die Funktion $F : \{0,1\}^* \to \{0,1,\#\}^*$ die durch die Zuordnung

$$F(x) = x\#x$$

gegeben ist, eine Reduktion von H_S auf H. Die Unentscheidbarkeit von H folgt damit aus der Unentscheidbarkeit von H_S. $\square$

Bemerkung (Unmittelbare Konsequenz)

Man kann allgemein nicht algorithisch überprüfen (d. h. per Programm), ob ein gegebenes Programm für eine konkrete Eingabe terminiert.

Das Halteproblem

- Wir zeigen, dass H_S nicht entscheidbar ist. $\checkmark$
- Wir reduzieren H_S auf H und folgern daraus, dass H nicht entscheidbar ist. $\checkmark$
- Wir reduzieren H auf H_0 und folgern daraus, dass H_0 nicht entscheidbar ist.

Das Halteproblem

Theorem (Unentscheidbarkeit von H_0)

Das Halteproblem auf leerem Band ist nicht entscheidbar.

Beweisidee.

Das allgemeine Halteproblem H wird auf das Halteproblem auf leerem Band reduziert. Anschaulich funktioniert die Reduktion wie folgt.

Die Entscheidung, ob eine Turing-Maschine T auf dem Input x anhält, ist äquivalent zur Entscheidung, ob die Turing-Maschine T' auf dem leeren Band anhält.

Die TM T': $\boxed{\text{Band} := \text{x}} \longrightarrow \boxed{\text{T}}$

(Die TM T' schreibt zunächst die Eingabe x auf das leere Band und verhält sich dann wie die TM T.) □

Das Halteproblem

- Wir zeigen, dass H_S nicht entscheidbar ist.$\checkmark$
- Wir reduzieren H_S auf H und folgern daraus, dass H nicht entscheidbar ist.$\checkmark$
- Wir reduzieren H auf H_0 und folgern daraus, dass H_0 nicht entscheidbar ist. $\checkmark$

Das Halteproblem

Satz

Die Probleme H_0, H_S und H sind semi-entscheidbar.

Beweisidee.

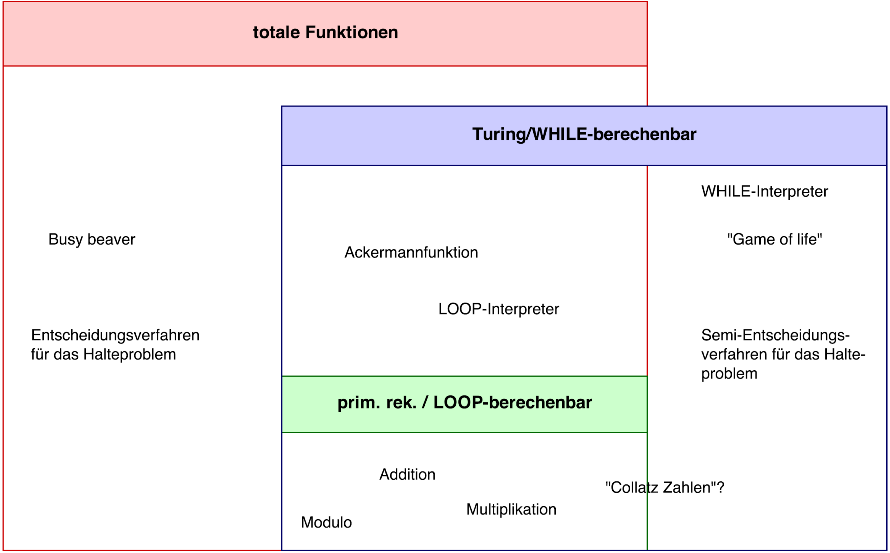
Wegen $H_s \preceq H \preceq H_0$ genügt es nachzuweisen, dass H_0 semi-entscheidbar ist.

Ein semi-Entscheidungsverfahren für H_0 kann gemäss dem folgenden Schema mit Hilfe einer universellen Turing-Maschine U angegeben werden:

$$\boxed{\text{U}} \longrightarrow \boxed{\text{Band} := 1}$$

$\square$

Überblick


totale Funktionen
Turing/WHILE-berechenbar
prim. rek. / LOOP-berechenbar
Busy beaver
Entscheidungsverfahren für das Halteproblem
Ackermannfunktion
LOOP-Interpreter
WHILE-Interpreter
"Game of life"
Semi-Entscheidungs-verfahren für das Halte-problem
Addition
Multiplikation
Modulo
"Collatz Zahlen"?

Satz von Rice

Satz

Ist R die Menge aller berechenbaren Funktionen und $S \subset R$ eine echte, nichtleere Teilmenge, dann ist die Sprache

$$C(S) = \{w \in \{0,1\}^* \mid F_w \in S\}$$

unentscheidbar.

Beweis.

Für einen Beweis sei auf das Buch «Theoretische Informatik – kurz gefasst» (Seiten 122 und 123) von Uwe Schöning verwiesen. □

Satz von Rice

Bemerkung (Konsequenzen)

- Es ist (im Allgemeinen) unmöglich mechanisch zu überprüfen, ob ein gegebenes Programm eine bestimmte Spezifikation erfüllt.
- Es ist (im Allgemeinen) unmöglich mechanisch zu überprüfen, ob ein gegebenes Programm frei von "bugs" ist.
- Es ist (im Allgemeinen) unmöglich mechanisch zu überprüfen, ob ein gegebenes Programm bei jeder Eingabe terminiert.
- Es ist (im Allgemeinen) unmöglich mechanisch zu überprüfen, ob zwei gegebene Programme dieselbe Funktionalität haben.

(Semi-) Entscheidbarkeit

Beispiel (Collatz-Zahlen)

Gegeben: Eine natürliche Zahl $n > 0$

Bildungsvorschrift: Ist n gerade, setze $n = n/2$
Ist n ungerade: setze $n = 3n + 1$

Gefragt: Mündet die Folge mit Startwert n in den Zyklus $4, 2, 1$?

Für $n = 8 :\ 8 \rightarrow 4 \rightarrow 2 \rightarrow 1$

Für $n = 9 :\ 9 \rightarrow 28 \rightarrow 14 \rightarrow 7 \rightarrow 22 \rightarrow 11 \rightarrow 34 \rightarrow 17 \rightarrow 52 \rightarrow 26$
$\rightarrow 13 \rightarrow 40 \rightarrow 20 \rightarrow 10 \rightarrow 5 \rightarrow 16 \rightarrow 8 \rightarrow 4 \rightarrow 2 \rightarrow 1$

(Semi-) Entscheidbarkeit

Aufgabe (Collatz-Zahlen)

Frage: Sind 27, $6‘171$ und $837‘799$ Collatz-Zahlen?