Formale Grundlagen der Informatik 1
Kapitel 9
Entscheidbarkeit und Aufzählbarkeit

Frank Heitmann
heitmann@informatik.uni-hamburg.de

5. Mai 2014

Aufzählbarkeit

Definition

1. Eine Menge $M \subseteq \Sigma^*$ heißt **(rekursiv) aufzählbar** genau dann, wenn $M = \emptyset$ ist oder eine totale, berechenbare Funktion $g : \mathbb{N} \to \Sigma^*$ existiert mit $g(\mathbb{N}) = M$.
2. Die Familie aller aufzählbaren Mengen wird mit RE (recursively enumerable) bezeichnet.

Satz

Eine Menge M ist aufzählbar genau dann, wenn

1. *eine k-Band off-line TM existiert, die jedes Wort der Menge M genau einmal auf ihr Ausgabeband schreibt.*
2. *$M = L(A)$ für eine DTM A gilt.*

Entscheidbarkeit

Definition

1. Eine Menge $M \subseteq \Sigma^*$ heißt **entscheidbar** genau dann, wenn die charakteristische Funktion $\chi_M : \Sigma^* \to \{0, 1\}$ berechenbar ist.
2. Die Familie aller entscheidbaren Mengen wird mit REC (recursive sets) bezeichnet.

Bemerkung

Die charakteristische Funktion einer Menge M tut folgendes:

$$\chi_M(x) = 1 \text{ gdw. } x \in M$$

(Ist also $x \notin M$, so ist $\chi_M(x) = 0$.)

Entscheidbarkeit

Definition

Eine Menge $M \subseteq \Sigma^*$ heißt **entscheidbar** genau dann, wenn die charakteristische Funktion $\chi_M : \Sigma^* \to \{0, 1\}$ berechenbar ist.

Satz

Eine Menge M ist genau dann entscheidbar, wenn es eine TM A gibt mit $L(A) = M$ und derart, dass A auf jeder Eingabe anhält.

Anmerkung

Man könnte dies auch als alternative Definition benutzen. Das wollen wir ab sofort auch machen!

Definitionen von RE und REC

Definition (Alternative Definition von RE)

Die von Turing-Maschinen akzeptierten Sprachen bilden die Sprachfamilie RE der **aufzählbaren Sprachen**.

Definition (Alternative Definition von REC)

Die Sprachen M, die von einer TM A akzeptiert werden können, so dass A auf jeder Eingabe hält, bilden die Sprachfamilie REC der **entscheidbaren Sprachen**.

Entscheidbarkeit vs. Aufzählbarkeit

Entscheidbarkeit von M:

- Es gibt eine TM A mit $L(A) = M$.
- **und** A hält auf jeder Eingabe (in einem Endzustand, wenn das vorgelegte Wort in M ist, sonst in einem Nicht-Endzustand).

Aufzählbarkeit von M:

- Es gibt eine TM A mit $L(A) = M$.

Insb. muss die TM für eine aufzählbare Menge nicht bei jeder Eingabe anhalten! Bei Worten in M tut sie es (da die Worte ja auch in $L(A)$ sind). Bei Worten, die nicht in M sind, tut sie es nicht zwingend. Rechnet eine TM also lange, dann kann das daran liegen, dass das Wort nicht in M ist oder dass sie noch nicht fertig ist!

Sprachen in REC

Satz

Die Sprache

$$DFA_{acc} := \{\langle A, w\rangle \mid A \text{ ist ein DFA und akzeptiert } w\}$$

ist entscheidbar.

Wichtige Anmerkung

Wir benutzen $\langle A, w\rangle$ um **eine Kodierung** von A und w zu bezeichnen. Wir notieren allgemein $\langle M\rangle$ für ein mathematisches Objekt M (DFA, PDA, TM, Wort, ...). Wichtig ist, dass man M endlich beschreiben kann.

Sprachen in REC

Satz

Die Sprache

$$DFA_{acc} := \{\langle A, w\rangle \mid A \text{ ist ein DFA und akzeptiert } w\}$$

ist entscheidbar.

Beweis.

Eine TM M, die DFA_{acc} entscheidet, arbeitet wie folgt:

1. Simuliere A mit Eingabe w
2. Endet die Simulation in einem Endzustand von A, akzeptiere, sonst lehne ab.

□

Sprachen in REC

Details...

- Die Eingabe $\langle A, w \rangle$ soll als "sinnvolle Repräsentation" gegeben sein, z.B.
  - einfach als Liste der Komponenten von A und
  - dort als Liste der Elemente (alle Zustände hintereinand dann die Symbole, dann die Übergangsfunktion als Liste von Tupeln usw.)
  - Die TM überprüft zu Anfang, ob wirklich ein DFA und ein Eingabewort vorliegt.
- M merkt sich dann die aktuelle Position in w und den aktuellen Zustand von A.
- Dann wird immer für jedes Tupel aus Zustand z und gerade zu lesendes Symbol a
  - der Zustand entsprechend $\delta(z, a)$ gewechselt und
  - die Position im Wort um eins erhöht

Sprachen in REC

Mehr Details...

- Liest A das Wort zu Ende und ist dann in einem Endzustand, wird akzeptiert, sonst (blockieren oder nicht in einem Endzustand) nicht.
- Die TM hält bei jeder Eingabe und gibt auch immer die korrekte Antwort aus, damit entscheidet M die Sprache DFA_{acc}.

Entscheidbare Sprachen

Wichtige Anmerkung

In der Zukunft machen wir das schneller und überspringen Schritte wie bspw. Syntaxüberprüfungen oder genauer auszuführen, wie eine TM $\delta(z, a)$ nachgucken kann etc.
Wir beschreiben also auf einem gewissen Abstraktionslevel die Funktionsweise der TM.
Zur Übung ist es aber am Anfang gut, sich immer wieder zu überlegen, wie man eine TM programmieren müsste (oder auch in einer höheren Programmiersprache).

Sprachen in REC

Satz

Die folgende Sprache ist entscheidbar

$$NFA_{acc} := \{\langle A, w \rangle \mid A \text{ ist ein NFA und akzeptiert } w\}$$

Beweis.

Man kann das so ähnlich machen wie eben, merkt sich dann aber immer die Menge der Zustände, in denen der NFA sein kann (und aktualisiert diese Menge mit jedem gelesenen Buchstaben). □

Sprachen in REC

$$NFA_{acc} := \{\langle A, w\rangle \mid A \text{ ist ein NFA und akzeptiert } w\}$$

Beweis.

[Alternative]

1. Konstruiere zu A den Potenzautomaten B
2. B ist ein DFA. Benutze die im vorherigen Satz konstruierte TM M um $\langle B, w\rangle \in DFA_{acc}$ zu entscheiden.
3. Akzeptiere, falls M akzeptiert, sonst lehne ab.

(Dies klappt, weil wir von der Potenzautomatenkonstruktion wissen, dass sie korrekt ist und dass sie in einen stets terminierenden Algorithmus umgewandelt werden kann, der auch auf einer TM implementiert werden kann.) □

Sprachen in REC

Satz

Die Sprache

$$DFA_{\emptyset} := \{\langle A\rangle \mid A \textit{ ist ein DFA und } L(A) = \emptyset\}$$

ist entscheidbar.

Beweis.

Hausaufgabe! □

Sprachen in REC

Satz

Die Sprache

$$DFA_{=} := \{\langle A, B\rangle \mid A \textit{ und } B \textit{ sind DFAs mit } L(A) = L(B)\}$$

ist entscheidbar.

Beweis

Wir konstruieren einen DFA C mit

$$L(C) = (L(A) \cap \overline{L(B)}) \cup (\overline{L(A)} \cap L(B))$$

Es gilt

$$L(C) = \emptyset \text{ gdw. } L(A) = L(B) \quad \text{(Beweis?)}$$

Sprachen in REC

Beweis.

Damit:

1. Konstruiere C.
2. Benutze eine TM M, die $DFA_{\emptyset}$ entscheidet, auf C.
3. Falls M akzeptiert, akzeptiere, sonst nicht.

Wichtig hier ist, dass alle Schritte zur Konstruktion von C von einer TM ausführbar sind! □

Sprachen in REC

Satz

Die Sprache

$$CFG_{acc} := \{\langle G, w\rangle \mid G \text{ ist eine CFG und generiert } w\}$$

ist entscheidbar.

Beweis

1. Gehe durch alle Ableitungen
2. Teste, ob eine davon w generiert
3. Falls ja, akzeptiere, sonst nicht

Funktioniert nicht, da es unendlich viele Ableitungen geben könnte!

Sprachen in REC

$$CFG_{acc} := \{\langle G, w\rangle \mid G \text{ ist eine CFG und generiert } w\}$$

Wir nutzen aus, dass jede Ableitung eines Wortes w der Länge $n \geq 1$ in einer Grammatik in *Chomsky-Normalform* genau $2n-1$ Schritte hat. (Warum?)

1. Wandle G in eine äquivalente Grammatik in (erweiterter) Chomsky-Normalform.
2. Sei $n := |w|$. Ist $n > 0$?
   - Ja! Liste alle Ableitungen der Länge $2n-1$ auf.
   - Nein! Liste alle Ableitungen der Länge 1 auf.
3. Generiert eine davon w, akzeptiere, sonst nicht.

□

Sprachen in REC

Hinweis

Das Verfahren eben ist recht ineffizient. Ein viel effizienteres Verfahren geht auf Cocke, Younger und Kasami zurück. Ihr findet dieses Verfahren im Skript und in [HMU].

CF und REC

Satz

Jede kontextfreie Sprache ist entscheidbar!

Beweis.

Sei A eine kontextfreie Sprache und G eine Grammatik mit $L(G) = A$. Wir wollen eine TM M angeben, die A entscheidet. M nutzt die vorherige TM M', die CFG_{acc} entscheidet. Bei Eingabe w

1. startet M die TM M' auf der Eingabe $\langle G, w\rangle$.
2. Akzeptiert M', so akzeptiert M; sonst nicht.

□

Wichtige Anmerkung

Eine PDA Simulation geht hier nicht. Da könntet ihr in Endlosschleifen geraten und dann nicht anhalten!

Sprachen in REC

Satz

Sprachen wie

- $L = \{\langle x, y\rangle \mid y = x + 1\}$
- $L = \{\langle x, y, z\rangle \mid x + y = z\}$
- $L = \{\langle G, v\rangle \mid G$ *ist ein Graph und* v *ein erreichbarer Knoten*$\}$
- ...

sind entscheidbar.

Bemerkung

Die ersten zwei hatten wir in vergangenen Vorlesungen. Die dritte kennt ihr aus DM. Wichtig ist, dass dort ein Algorithmus vorgestellt wurde, der nach endlich vielen Schritten immer mit 'Ja' oder 'Nein' terminiert.

Sprachen in RE

Satz

Die Sprache

$$TM_{acc} := \{\langle M, w\rangle \mid M \text{ ist eine TM und akzeptiert } w\}$$

ist aufzählbar.

Beweis.

Sei $\langle M, w\rangle$ die Eingabe, wobei M eine TM und w ein Wort ist.

1. Simuliere M auf w.
2. Wenn M jemals einen Endzustand erreicht, akzeptiere. Lehnt M ab, so lehne ab.

Sollte M bei w aber in eine Endlosschleife geraten, so auch die TM, die M simuliert. D.h. wir akzeptieren die Sprache TM_{acc} zwar, aber wir entscheiden sie (noch?) nicht! □

Zur Bedeutung von TM_{acc}

Sollten wir eine Möglichkeit haben algorithmisch zu entscheiden (d.h. insb. ohne Simulation), dass M auf w nicht anhält, könnten wir anhalten. Deswegen heißt obiges Problem manchmal auch **Halteproblem** (hier eigentlich: *Akzeptanzproblem*).

Ausblick

Obiges Problem ist tatsächlich **unentscheidbar**! D.h. es gibt keinen Algorithmus, der TM_{acc} akzeptiert und auf jeder Eingabe anhält.

Zur Bedeutung

Die Bedeutung des obigen Problems kann nicht genug betont werden. Mit obigem Problem sind etliche wichtige Verifikationsfragen außerhalb unserer Möglichkeiten! Zum Beispiel, ob ein Algorithmus terminiert, ob eine bestimmte Ausgabe produziert wird, ob eine bestimmte Methode jemals aufgerufen wird usw. - einige davon, werden wir noch sehen.

Das Halteproblem

Satz

Die folgende Sprache ist nicht entscheibar

$$TM_{acc} := \{\langle M, w\rangle \mid M \text{ ist eine TM und akzeptiert } w\}$$

Beweis

Wir nehmen an, wir hätten eine TM H, die TM_{acc} entscheidet und führen dies zu einem Widerspruch. H ist also eine TM mit

- H akzeptiert $\langle M, w\rangle$, wenn M das Wort w akzeptiert
- H lehnt $\langle M, w\rangle$ ab, wenn M das Wort w nicht akzeptiert

oder kürzer:

- $H(\langle M, w\rangle) = 1$, wenn $M(w) = 1$
- $H(\langle M, w\rangle) = 0$, wenn $M(w) \neq 1$

Das Halteproblem

Beweis (Fortsetzung)

Wir konstruieren nun eine TM D, die H als Subroutine benutzt. D nimmt als Eingabe eine TM M (die Eingabe ist also $\langle M \rangle$) und arbeitet wie folgt:

1. Starte H mit Eingabe $\langle M, \langle M \rangle \rangle$
2. Drehe das Ergebnis von H um und gebe es aus

D.h. es ist

- $D(\langle M \rangle) = 1$, wenn M die Eingabe $\langle M \rangle$ nicht akzeptiert
- $D(\langle M \rangle) = 0$, wenn M die Eingabe $\langle M \rangle$ akzeptiert

Das Halteproblem

Beweis (Fortsetzung)

Insgesamt haben wir jetzt:

$$H(\langle M, w \rangle) = \begin{cases} 1, & \text{wenn } M(w) = 1 \\ 0, & \text{wenn } M(w) \neq 1 \end{cases}$$

$$D(\langle M \rangle) = \begin{cases} 1, & \text{wenn } M(\langle M \rangle) \neq 1 \\ 0, & \text{wenn } M(\langle M \rangle) = 1 \end{cases}$$

Was passiert nun, wenn man D mit $\langle D \rangle$ ausführt?

$$D(\langle D \rangle) = \begin{cases} 1, & \text{wenn } D(\langle D \rangle) \neq 1 \\ 0, & \text{wenn } D(\langle D \rangle) = 1 \end{cases}$$

Widerspruch! *Daher kann H nicht existieren!* □

Intermezzo

Als Programmcode ...

Die Sprache L_d

Man kann die Wörter eines Alphabets aufzählen, z.B. indem man sie erst nach Länge und dann lexikalisch sortiert:

$$a, b, aa, ab, ba, bb, aaa, aab, aba, abb, baa, \ldots$$

So kann man sowohl die Kodierungen aller Turing-Maschinen aufzählen als auch die Wörter, die sie lesen. Wir können dann von den Turing-Maschinen $M_1, M_2, M_3, \ldots$ und den Wörtern $w_1, w_2, w_3, \ldots$ sprechen.

Anmerkung

Genauer kann man hier die lexikalische Ordnung und Gödelisierungen einführen. Siehe Skript.

Die Sprache L_d

Satz

$L_d := \{w_i \mid w_i \notin L(A_i)\}$ *ist nicht aufzählbar.*

Beweis

Man kann die Wörter und die TMs in einer Matrix anordnen:

	M_1	M_2	M_3	M_4	$\ldots$	L_d
w_1	1	1	0	0		–
w_2	1	0	0	0	$\ldots$	*Ja*
w_3	0	1	1	1	$\ldots$	–
w_4	0	0	1	0		*Ja*
.	.	.	.	.	.	.

L_d entspricht gerade der Diagonalen, von der nur die Einträge mit 0 in L_d aufgenommen werden.

Die Sprache L_d

$$L_d := \{w_i \mid w_i \notin L(A_i)\} \notin RE$$

Das Problem ist, dass eine TM, die L_d akzeptiert, ein M_j in der Matrix sein müsste und dann wird es einen Widerspruch mit w_j geben (akzeptieren/nicht akzeptieren).
Genauer: Angenommen L_d wäre aufzählbar. Dann gibt es eine TM A_j aus der Aufzählung mit $L(A_j) = L_d$. Wir betrachten w_j. Es muss entweder $w_j \in L_d$ oder $w_j \notin L_d$ sein. Beide Fälle führen zu einem Widerspruch:

- $w_j \in L_d$, dann $w_j \notin L(A_j)$ (nach Def. von L_d), dann $w_j \notin L_d$ (nach Annahme $L(A_j) = L_d$).
- $w_j \notin L_d$, dann $w_j \notin L(A_j)$ (nach Annahme), dann $w_j \in L_d$ (Def. von L_d).

Also ist L_d nicht aufzählbar! □

Die Sprache L_d und darüber hinaus...

- L_d ist also nicht einmal aufzählbar!
- Sie ist aber *ab*zählbar!
- Eine Menge, die dann nicht einmal mehr abzählbar ist, ist z.B. die Menge der reellen Zahlen.

Nebenbemerkung

Eine Menge ist abzählbar, wenn sie endlich ist oder eine Bijektion von den natürlichen Zahlen angegeben werden kann.

Reduktion

Das Ziel

Wir wollen nun weitere Probleme als unentscheidbar nachweisen

Will man eine Sprache L als unentscheidbar nachweisen, so ist das übliche Vorgehen:

1. Wir nehmen an L wäre entscheidbar. Sei M eine TM, die L entscheidet.
2. Unter Nutzung von M nun eine TM konstruieren, die ein schon als unentscheidbar nachgewiesenes Problem entscheidet.
3. Das ist dann ein Widerspruch, also kann M nicht existieren.

Dies nennt man eine *Reduktion*, und sagt, dass das unentscheidbare Problem auf das neue Problem reduziert wurde.

Weitere unentscheidbare Probleme

Satz

$$TM_{halt} := \{\langle M, w\rangle \mid M \text{ ist eine TM und hält auf } w\}$$

ist nicht entscheidbar.

Beweis

Angenommen TM_{halt} wäre entscheidbar. Sei H eine TM, die TM_{halt} entscheidet. Wir konstruieren nun eine TM S, die TM_{acc} (unter Nutzung von H) entscheidet. Da aber TM_{acc} nicht entscheidbar ist, muss die Annahme falsch sein und daher ist auch TM_{halt} nicht entscheidbar.

Anmerkung

TM_{halt} ist das **Halteproblem**. Dies wird oft in Unentscheidbarkeitsbeweisen benutzt.

Weitere unentscheidbare Probleme

Satz

$$TM_{halt} := \{\langle M, w\rangle \mid M \text{ ist eine TM und hält auf } w\}$$

ist nicht entscheidbar.

Beweis.

H entscheidet nach Annahme TM_{halt}. S arbeitet nun bei Eingabe $\langle M, w\rangle$ wie folgt:

1. Starte H mit Eingabe $\langle M, w\rangle$
2. Lehnt H ab, lehne ab.
3. Akzeptiert H, simuliere M auf w (M terminiert, da H akzeptiert hat!)
4. Wenn M akzeptiert, akzeptiere; sonst lehne ab.

Wie eben beschrieben, kann TM_{halt} nicht entscheidbar sein. □

Weitere unentscheidbare Probleme

Satz

$$TM_{print0} := \{\langle M\rangle \mid \text{gestartet auf } \lambda \text{ gibt } M \text{ irgendwann 0 aus}\}$$

ist nicht entscheidbar.

Beweis.

In den Präsenzaufgaben! □

Weitere unentscheidbare Probleme

Definition

Ein *nutzloser Zustand* in einer TM ist ein Zustand, der bei keinem Eingabewort jemals betreten wird. Sei

$$\texttt{UselessState} = \{\langle A, q\rangle \mid A \text{ ist eine TM und } q \text{ ein nutzloser Zustand von } A\}$$

Bemerkung

Dieses Problem ist eng verwandt mit der Frage, ob eine bestimmte Stelle bspw. in einem Java-Programm jemals erreicht wird.

Weitere unentscheidbare Probleme

Satz

$$\texttt{UselessState} = \{\langle A, q\rangle \mid A \text{ TM, } q \text{ nutzloser Zustand von } A\}$$

ist unentscheidbar.

Beweis

Angenommen `UselessState` wäre entscheidbar und A_{US} eine TM, die das Problem entscheidet. Wir wollen damit nun das Halteproblem TM_{halt} entscheiden.

Die Idee

- Zu einer gegebenen TM M mit Zustandsmenge Z und Bandalphabet Γ konstruieren wir eine TM M' mit:
  - einem neuen Zustand z_{neu}
  - für jeden Zustand $z \in Z$ von M und jedes Symbol $x \in \Gamma$, für das es keinen Übergang aus z in M gab (d.h. es gab keine Kante (z, x, X, Y, z') in M mit X, Y, z' beliebig) wird eine neue Kante (z, x, x, R, z_{neu}) hinzugefügt

  Hält M, so kann M' noch einen Übergang nach z_{neu} machen. Dort hält M'. *M hält also genau dann in irgendeinem Zustand, wenn M' in z_{neu} hält.*
- Zu $\langle M, w\rangle$ kann außerdem eine TM M'' konstruiert werden, die bei jeder Eingabe das Band löscht und dann w auf M' startet, wobei M' wie eben beschreiben aus M hervorgeht.

Die Idee

Zwischenstand

- Wenn M auf w hält, dann hält M'' bei jeder Eingabe in z_{neu}.
- Wenn M nicht auf w hält, dann besucht M'' z_{neu} niemals.

Abschluss

$$\texttt{UselessState} = \{\langle A, q\rangle \mid A \text{ TM, } q \text{ nutzloser Zustand von } A\}$$

Bei Eingabe $\langle M, w\rangle$

1. Konstruiere aus $\langle M, w\rangle$ die TM M''.
2. Starte A_{US} auf $\langle M'', z_{neu}\rangle$.
3. Akzeptiert A_{US}, dann ist z_{neu} ein nutzloser Zustand und nach obigem hält dann M nicht auf w und wir lehnen ab.
4. Lehnt A_{US} ab, so ist z_{neu} nicht nutzlos, wird also besucht. Dann aber hält M auf w an und wir akzeptieren.

Dies entscheidet das Halteproblem, was nicht sein kann! A_{US} existiert also nicht! □

Ausblick

Wir wissen nun, dass es viele entscheidbare Probleme und leider auch wichtige unentscheidbare Probleme gibt.

Für die Praxis genügt es aber nicht, dass ein Problem entscheidbar ist! Die "Kosten" sind auch wichtig!

- Wie lange brauchen wir um das Problem zu lösen?
- Wie viel Platz/Speicher benötigen wir?

Damit beschäftigen wir uns morgen...

Zusammenfassung

Wir haben heute

- entscheidbare und
- aufzählbare

Sprachen kennengelernt und gesehen, dass

- $REG \subsetneq CF \subsetneq REC \subsetneq RE$

gilt!

Ferner haben wir das Halteproblem und weitere **nicht entscheidbare** Probleme kennengelernt und eine Technik (Reduktion), wie man neue Probleme als unentscheidbar nachweisen kann.