Formale Grundlagen der Informatik 1
Kapitel 10
Aufzählbarkeit und (Un-)Entscheidbarkeit

Frank Heitmann
heitmann@informatik.uni-hamburg.de

11. Mai 2015

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.*

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.*

Aufzählbarkeit

Insbesondere als Satz oder alternative Definition:

Satz/Definition

eine Menge M ist **(rekursiv) aufzählbar** genau dann, wenn eine DTM A existiert mit $L(A) = M$.

Anmerkung

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

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!

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!

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!

Fragen

Sind alle regulären Sprachen entscheidbar?

1. Ja!
2. Nein!
3. Keine Teilmengenbeziehung!
4. Weiß ich nicht...

Fragen

Sind alle kontextfreien Sprachen entscheidbar?

1. Ja!
2. Nein!
3. Keine Teilmengenbeziehung!
4. Weiß ich nicht...

Fragen

Sind alle aufzählbaren Sprachen entscheidbar?

1. Ja!
2. Nein!
3. Keine Teilmengenbeziehung!
4. Weiß ich nicht...

Fragen

Wenn eine Sprache entscheidbar ist, ist dann auch das Komplement entscheidbar?

1. Ja!
2. Nein!
3. Weiß ich nicht...

Fragen

Wenn zwei Sprachen L_1 und L_2 entscheidbar sind, ist dann auch $L_1 \cap L_2$ entscheidbar?

1. Ja!
2. Nein!
3. Weiß ich nicht...

Zur Nachbereitung

Zur Nachbereitung

1. Ja!
2. Ja! (Sehen wir noch genauer...)
3. Nein! (Sehen wir noch genauer...)
4. Ja!
5. Ja!

Fragen

Wenn eine Sprache aufzählbar ist, ist dann auch das Komplement aufzählbar?

1. Ja!
2. Nein!

Fragen

Wenn eine Sprache aufzählbar ist, ist dann auch das Komplement aufzählbar?

1. Ja!
2. Nein!

Zur Nachbereitung

Nein, nicht zwingend, wie der folgende Satz zeigt würde dies nämlich bereits implizieren, dass die Sprache dann sogar entscheidbar ist. Und da wir noch genauer sehen werden, dass die entscheidbaren Sprachen eine echte Teilmenge der aufzählbaren Sprachen sind, kann obiges dann nicht allgemein gelten.

Entscheidbar und Aufzählbar

Satz

Eine Sprache L ist entscheidbar genau dann, wenn L und $\overline{L}$ (das Komplement von L) aufzählbar sind.

Beweis.

($\Rightarrow$) Ist L entscheidbar, dann ist L auch aufzählbar. Dreht man zudem die Ergebnisse einer TM, die L entscheidet, um, hat man eine TM, die $\overline{L}$ entscheidet. Damit ist $\overline{L}$ dann auch aufzählbar. $\square$

Entscheidbar und Aufzählbar

Satz

Eine Sprache L ist entscheidbar genau dann, wenn L und $\overline{L}$ (das Komplement von L) aufzählbar sind.

Beweis.

($\Rightarrow$) Ist L entscheidbar, dann ist L auch aufzählbar. Dreht man zudem die Ergebnisse einer TM, die L entscheidet, um, hat man eine TM, die $\overline{L}$ entscheidet. Damit ist $\overline{L}$ dann auch aufzählbar. $\square$

Entscheidbar und Aufzählbar

Satz

Eine Sprache L ist entscheidbar genau dann, wenn L und $\overline{L}$ (das Komplement von L) aufzählbar sind.

Beweis.

($\Leftarrow$) Seien A und $\overline{A}$ TMs, die L bzw. $\overline{L}$ akzeptieren. Eine TM B, die L entscheidet arbeitet wie folgt:

1. Bei Eingabe von w
2. Lasse A einen Schritt arbeiten
3. Dann $\overline{A}$ einen Schritt
4. Wiederhole Schritte 2 und 3 bis eine akzeptiert.
5. Hat A akzeptiert, akzeptiere. Hat $\overline{A}$ akzeptiert, lehne ab.

□

Entscheidbar und Aufzählbar

Satz

Eine Sprache L ist entscheidbar genau dann, wenn L und $\overline{L}$ (das Komplement von L) aufzählbar sind.

Beweis.

($\Leftarrow$) Seien A und $\overline{A}$ TMs, die L bzw. $\overline{L}$ akzeptieren. Eine TM B, die L entscheidet arbeitet wie folgt:

1. Bei Eingabe von w
2. Lasse A einen Schritt arbeiten
3. Dann $\overline{A}$ einen Schritt
4. Wiederhole Schritte 2 und 3 bis eine akzeptiert.
5. Hat A akzeptiert, akzeptiere. Hat $\overline{A}$ akzeptiert, lehne ab.

□

Entscheidbar und Aufzählbar

Satz

Eine Sprache L ist entscheidbar genau dann, wenn L und $\overline{L}$ (das Komplement von L) aufzählbar sind.

Beweis.

($\Leftarrow$) Seien A und $\overline{A}$ TMs, die L bzw. $\overline{L}$ akzeptieren. Eine TM B, die L entscheidet arbeitet wie folgt:

1. Bei Eingabe von w
2. Lasse A einen Schritt arbeiten
3. Dann $\overline{A}$ einen Schritt
4. Wiederhole Schritte 2 und 3 bis eine akzeptiert.
5. Hat A akzeptiert, akzeptiere. Hat $\overline{A}$ akzeptiert, lehne ab.

□

Entscheidbar und Aufzählbar

Satz

Eine Sprache L ist entscheidbar genau dann, wenn L und $\overline{L}$ (das Komplement von L) aufzählbar sind.

Beweis.

($\Leftarrow$) Seien A und $\overline{A}$ TMs, die L bzw. $\overline{L}$ akzeptieren. Eine TM B, die L entscheidet arbeitet wie folgt:

1. Bei Eingabe von w
2. Lasse A einen Schritt arbeiten
3. Dann $\overline{A}$ einen Schritt
4. Wiederhole Schritte 2 und 3 bis eine akzeptiert.
5. Hat A akzeptiert, akzeptiere. Hat $\overline{A}$ akzeptiert, lehne ab.

□

Entscheidbar und Aufzählbar

Satz

Eine Sprache L ist entscheidbar genau dann, wenn L und $\overline{L}$ (das Komplement von L) aufzählbar sind.

Beweis.

($\Leftarrow$) Seien A und $\overline{A}$ TMs, die L bzw. $\overline{L}$ akzeptieren. Eine TM B, die L entscheidet arbeitet wie folgt:

1. Bei Eingabe von w
2. Lasse A einen Schritt arbeiten
3. Dann $\overline{A}$ einen Schritt
4. Wiederhole Schritte 2 und 3 bis eine akzeptiert.
5. Hat A akzeptiert, akzeptiere. Hat $\overline{A}$ akzeptiert, lehne ab.

□

Der weitere Ablauf...

Was heute kommt ...

- Sprachen in REC (entscheidbare Sprachen)
- Sprachen in RE (aufzählbare Sprachen)
- Unentscheidbare Sprachen
- Sprachen, die nicht einmal mehr aufzählbar sind

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.

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.

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

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

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}.

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

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

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

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

ist entscheidbar.

Sprachen in REC

Satz

Die Sprache

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

ist entscheidbar.

Beweis.

Präsenzaufgabe! □

Sprachen in REC

Satz

Die Sprache

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

ist entscheidbar.

Sprachen in REC

Satz

Die Sprache

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

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.

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

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

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

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!

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!

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.

Zur Bedeutung von REC

Anmerkung

Diese Beispiele sollen verdeutlichen welche zentrale Rolle die Familie der entscheidbaren Sprachen spielt. Insbesondere sind all diese Probleme durch Algorithmen lösbar - und umgekehrt ist alles, was algorithmisch lösbar ist, auch auf einer TM lösbar. Die TM ist ein abstraktes Modell für einen Algorithmus bzw. das Berechenbare!

Nebenbemerkung

Mit 'Algorithmus' ist oben ein deterministischer Algorithmus gemeint, der eine Eingabe erhält und eine Ausgabe berechnet. Andere Algorithmen wie z.B. Algorithmen, die mit Zufall arbeiten, müssen erst umgewandelt werden.

Sprachen in RE

Satz

Die Sprache

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

ist aufzählbar.

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! □

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.

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.

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 Akzeptanz-/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 Akzeptanz-/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 Akzeptanz-/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 Akzeptanz-/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 Akzeptanz-/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 Akzeptanz-/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 Akzeptanz-/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!* □

Das Akzeptanz-/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!* □

Das Akzeptanz-/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 zur Bedeutung

Im Programmcode ...

Zusammenfassung

Wir haben heute

- entscheidbare und
- aufzählbare

Sprachen kennengelernt und gesehen, dass

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

gilt!