Reduzierbarkeit

Dr. Eva Richter

18.Mai 2012

Reduzierbarkeit

- A_{TM} ist Beispiel für ein unentscheidbares (algorithmisch unlösbares) Problem
- Methode für Unentscheidbarkeitsbeweise: Reduzierbarkeit
- Reduktion von Problem A auf Problem B bedeutet, mit Hilfe der Lösung von B kann man eine Lösung von A konstruieren
- Beispiel:
  - A Finde den Weg vom Bahnhof zum Hotel „Gute Nacht“ in Happytown.
  - B Kaufe einen Stadtplan von Happytown.
- Reduzierbarkeit sagt nichts über die Lösbarkeit von A oder B allein; es geht nur um die Lösbarkeit von A bei Vorhandensein einer Lösung von B

Reduzierbarkeit für Entscheidbarkeit

- Klassifizierung von Problemen in Bezug auf Entscheidbarkeit
- Klassifizierung von Problemen der Komplexitätstheorie in Bezug auf den Berechnungsaufwand
- Wenn A auf B reduzierbar ist, dann kann die Lösung von A nicht schwerer als die Lösung von B sein.
- Wenn A sich auf B reduzieren lässt und B entscheidbar ist, dann ist auch A entscheidbar.
- Beweismethode für Unentscheidbarkeit von B:
  Reduziere ein bekanntermaßen unentscheidbares Problem A auf B.

Das echte Halteproblem

Unentscheidbarkeit von A_{TM} wurde mit Diagonalisierungsmethode bewiesen

$$HALT_{TM} = \{< M, w >| \text{ Turingmaschine } M \text{ hält auf } w \text{ an}\}$$

Satz

$HALT_{TM}$ ist unentscheidbar.

Beweisidee:

- zeigen, dass A_{TM} sich auf $HALT_{TM}$ reduzieren lässt
- Lösung für $HALT_{TM}$ liefert dann eine Lösung für A_{TM}
- nehmen an, wir hätten Entscheider R für $HALT_{TM}$
- konstruieren Entscheider S für A_{TM}
- S soll bei Eingabe $< M, w >$ akzeptieren, falls $w \in L(M)$ akzeptiert und ablehnen, falls $w \notin M$
- Wie kann R feststellen, ob M in eine Schleife gerät?
- R weiß, ob M anhält.

Beweis für Unentscheidbarkeit des Halteproblems

Annahme: Es gibt einen Entscheider R für $HALT_{TM}$.

$S =$

„Auf Eingabe $< M, w >$, für Turingmaschine M und Wort w:

1. Starte R auf $< M, w >$.
2. Wenn R ablehnt, reject.
3. Wenn R akzeptiert, simuliere M auf w bis M anhält.
4. Falls M akzeptiert, accept. Falls M ablehnt, reject.“

- Falls R existiert, wäre S ein Entscheider für A_{TM}.
- Widerspruch zur Unentscheidbarkeit von A_{TM}
- R kann also nicht existieren: $HALT_{TM}$ ist unentscheidbar.

Leerheitsproblem für Turingmaschinen

$$E_{TM} = \{< M >| \text{ Turingmaschine } M \text{ mit } L(M) = \emptyset\}$$

Satz

E_{TM} ist unentscheidbar.

Beweisidee:

- Nehmen an, es gäbe Entscheider R für E_{TM}.
- Bauen daraus einen Entscheider S für A_{TM}.
- Idee 1: Starte R auf Eingabe $< M >$
  accept gdw. $L(M) = \emptyset$, d.h. $\forall w \in \Sigma^* w \notin L(M)$
  reject: ???
- Idee 2: Modifiziere M zu M_w mit $L(M_w) = \{w\}$ falls $w \in L(M)$ und $L(M_w) = \emptyset$ falls $w \notin L(M)$

Beweis:

$M_w =$

„Auf Eingabe x:

1. Falls $x \neq w$, reject.
2. Falls $x = w$, starte M mit Eingabe w und accept, falls M akzeptiert."

Annahme: R ist Entscheider für E_{TM}

$S =$

„Auf Eingabe $< M, w >$, für eine Turingmaschine M und Wort w:

1. Verwende Beschreibung von M und w um die Maschine M_w zu konstruieren.
2. Starte R mit Eingabe $< M_w >$.
3. Falls R akzeptiert, reject. Falls R zurückweist, accept.

$\square$

- S muss tatsächlich in der Lage sein, eine Beschreibung von M_w aus der Beschreibung von M und w zu konstruieren
- Das ist möglich durch Hinzufügen von Zuständen zu M, um den Vergleich $x = w$ auszuführen.
- Wäre R wirklich ein Entscheider für E_{TM}, dann wäre S ein Entscheider für A_{TM}.
- Es gibt aber keine Entscheider für A_{TM}!
- R kann auch nicht existieren, also ist E_{TM} unentscheidbar.

Entscheidbarkeit von Regularität

$REGULAR_{TM} = \{< M >|$ Turingmaschine M mit $L(M)$ ist regulär$\}$.

Satz

$REGULAR_{TM}$ ist unentscheidbar.

Beweisidee: indirekter Beweis durch Reduktion von A_{TM}

- Nehmen an, es gäbe Entscheider R für $REGULAR_{TM}$.
- Bauen daraus einen Entscheider S für A_{TM}.
- Verwende Eingabe $< M, w >$, um eine modifizierte Maschine M' zu konstruieren, so dass M' genau dann eine reguläre Sprache akzeptiert, wenn M die Eingabe w akzeptiert.
- M' soll $\{0^n 1^n \mid n \geq 0\}$ akzeptieren, wenn $w \notin L(M)$ und reguläre Sprache $\{0, 1\}^*$ wenn $w \in L(M)$.
- Konstruktionsvorschrift für M' aus $< M, w >$ ist erforderlich

Beweis: Annahme: R ist Entscheider für $REGULAR_{TM}$

$S =$

„Auf Eingabe $< M, w >$ für Turingmaschine M und Wort w:

1. Konstruiere $M'_w =$
   „Auf Eingabe x:
   1. Falls $x = 0^n1^n$ für ein n accept.
   2. Falls x nicht von dieser Form ist, starte M auf w und accept, wenn w von M akzeptiert wird.“
2. Starte R mit der Eingabe $< M'_w >$.
3. Falls R akzeptiert, dann accept; falls R ablehnt, reject.“

□

Satz von Rice

Satz (von Rice)

Sei P ein Menge von Turingmaschinen mit:

1. *P ist extensional:*
   Für Turingmaschinen M_1 und M_2 mit $L(M_1) = L(M_2)$ gilt:
   $< M_1 >\in P$ gdw. $< M_2 >\in P$.
2. *P ist nichttrivial:*
   Es existieren Turingmaschinen M_1 und M_2, wobei $< M_1 >\in P$ und $< M_2 >\notin P$.

Dann ist P nicht entscheidbar.

Beweis: Nachschlagen, Hausaufgabe $\square$

$$EQ_{TM} = \{< M_1, M_2 >|\ M_1, M_2 \text{ Turingmaschinen mit } L(M_1) = L(M_2)\}$$

Satz

EQ_{TM} ist unentscheidbar.

Beweisidee: indirekter Beweis durch Reduktion von E_{TM}

1. E_{TM} ist das Leerheitsproblem
2. Wäre $L(M_1)$ leer, dann müsste man nur feststellen, ob auch $L(M_2)$ leer ist.
3. Also ist E_{TM} Spezialfall von EQ_{TM} mit $L(M_1) = \emptyset$.

Beweis der Unentscheidbarkeit des Äquivalenzproblems

Beweis: Annahme: R ist Entscheider für EQ_{TM}

$S =$

„Auf Eingabe von $< M >$, für Turingmaschine M:

1. Starte R auf der Eingabe $< M, M_1 >$, wobei M_1 die Turingmaschine ist, die alle Eingaben zurückweist.
2. Falls R akzeptiert, accept; sonst reject."

- Würde R tatsächlich EQ_{TM} entscheiden, dann wäre S ein Entscheider für E_{TM}.
- Nach Satz 2 ist E_{TM} aber unentscheidbar.
- Die Annahme, dass es einen Entscheider für EQ_{TM} gibt ist daher falsch.

$\square$

Reduktion über Berechnungshistorie

- wichtige Methode, um die Reduzierbarkeit von A_{TM} auf bestimmte Sprachen zu beweisen
- sinnvoll, bei Unentscheidbarkeitsproblemen, die mit der Existenz von Zeugen zu tun haben
- Beispiel: Hilberts zehntes Problem – die Feststellung, ob ein Polynom ganzzahlige Wurzeln hat

Berechnungshistorien

Definition

Sei M eine Turingmaschine und w ein Eingabewort.

1. *Eine akzeptierende Berechnungshistorie für M auf w ist eine Folge von Konfigurationen $C_1, C_2, \dots C_l$, wobei C_1 die Startkonfiguration von M auf w ist und C_l eine akzeptierende Konfiguration von M und jedes C_i eine Nachfolgefonfiguration von C_{i-1} ist.*
2. *Eine ablehnende Berechnungshistorie ist Folge von Konfigurationen*
   $C_1, C_2, \dots C_l$ wie oben, bis auf dass C_l eine ablehnende Konfiguration ist.

Berechnungshistorien sind praktikabel, da sie

- endliche Folgen und
- für deterministische Maschinen eindeutig bestimmt sind.

Definition

Linear beschränkte Automaten(LBA) sind Turingmaschinen, bei denen der Kopf sich nur auf dem Teil des Bandes bewegen kann, auf dem die Eingabe steht.

Definition

Eine kontextsensitive Grammatik ist eine formale Grammatik $G = (N, T, P, S)$ mit Nichtterminalen N, Terminalen T, Startsymbol $S \in N$ und Produktionen P der Form

(i) $aXb \to acb$ *oder*

(ii) $S \to \varepsilon$,

wobei $a, b \in (N \cup T)^*, X \in N, c \in (N \cup T)^+$ und S kommt nur links vor.

„Rechte Produktionsseiten sind nicht kürzer als die linken."

Satz

Die Sprachklasse, die von kontextsensitiven Grammatiken erzeugt wird ist dieselbe wie die, die von LBAs erkannt wird.

- Bei der Programmierung behandelt man das rechte Bandende genauso wie bisher schon das linke Bandende.
- Linear beschränkte Automaten haben also nur beschränkten Speicherplatz zur Verfügung.
- Indem man ein Bandalphabet verwendet, das größer ist als das Eingabealphabet kann man den zur Verfügung stehenden Platz um einen konstanten Faktor vervielfachen, daher der Name linear beschränkt
- LBAs sind ziemlich mächtig, so sind z.B. die Entscheider von A_{DEA}, A_{GG}, E_{DEA} und E_{CFG} alle linear beschränkte Automaten.

$$A_{LBA} = \{< M, w >| \ M \text{ ist LBA, der } w \text{ akzeptiert}\}$$

ist entscheidbar!

Lemma

Sei M ein linear beschränkte Automat mit q Zuständen und g Symbolen im Bandalphabet. Dann gibt es bei Bandlänge n genau $g^n qn$ verschiedene Konfigurationen von M .

Beweis:

- Konfiguration besteht aus dem aktuellen Zustand, der Position und dem Bandinhalt.
- Es gibt g^n mögliche Bandinhalte.
- M hat q verschieden Zustände.
- Es gibt n verschiedenen Stellen für den Kopf.

Alle diese Anzahlen miteinander multipliziert ergeben die angegebene obere Schranke.

□

Das Wortproblem für LBA ist entscheidbar

Satz

A_{LBA} ist entscheidbar.

Beweisidee:

- Um zu entscheiden, ob der LBA M auf der Eingabe w anhält, simulieren wir M auf w.
- Um festzustellen, ob M in eine Schleife kommt, „beobachtet“ man die Berechnung: wiederholt sich eine Konfiguration?
- Das muss nach mindestens nach qng^n Schritten passieren.

Beweis:
Konstruiere einen Entscheider L für A_{LBA}.

$L =$

„Bei Eingabe von $< M, w >$ für LBA M und Wort w:

1. Simuliere M auf w für qng^n Schritte oder bis M anhält.
2. Falls M anhält und akzeptiert, dann accept; wenn M anhält und ablehnt, dann reject. Falls M nicht angehalten hatte, muss es eine Schleife geben. Daher wird w von M abgelehnt–reject."

□

Leerheitsproblem von LBA

$E_{LBA} = \{< M >|\ M$ ist ein linear beschränkter Automat und $L(M) = \emptyset\}$

Satz

E_{LBA} ist unentscheidbar.

Beweisidee:

- indirekt, durch Reduktion von A_{TM} auf E_{LBA}
- aus Eingabe für A_{TM} bauen wir Eingabe für E_{LBA}:
  Aus $< M, w >$ bauen wir LBA $B_{M,w}$, der als Sprache die akzeptierenden Berechnungshistorien von M bei Eingabe w hat.
- Falls $w \in L(M)$, dann enthält $L(B_{M,w})$ genau ein Wort, wenn nicht, dann ist $L(B_{M,w})$ leer.
- Konstruktion von $B_{M,w}$ muss von einer TM mit Eingabe $< M, w >$ ausgeführt werden können.

Was der LBA tun soll

- Eingabe x wird von $B_{M,w}$ genau dann akzeptiert, wenn sie eine akzeptierende Berechnungshistorie von M auf w ist
- akzeptierende Berechnungshistorie ist Folge von Konfigurationen $C_1, C_2 \dots C_l$
- gültige Eingabe x hat die Form

$$\sharp \underbrace{\qquad}_{C_1} \sharp \underbrace{\qquad}_{C_2} \sharp \underbrace{\qquad}_{C_3} \sharp \dots \sharp \underbrace{\qquad}_{C_l} \sharp$$

- $B_{M,w}$ zerlegt x mit Hilfe der Trennzeichen in Wörter $C_1, \dots C_l$ und prüft die drei Bedingungen einer Berechnungshistorie:
    1. C_1 ist Startkonfiguration von M auf w.
    2. Jedes C_{i+1} ist eine Nachfolgekonfiguration von C_i.
    3. C_l ist eine akzeptierende Konfiguration von M.

LBA lässt sich automatisch konstruieren

- C_1 ist Startkonfiguration für M auf w, wenn $C_1 = q_0 w_1 w_2 \dots w_n$, wobei q_0 der Startzustand von M ist. Sowohl q_0 als auch w sind direkt in $B_{M,w}$ eingebaut, also kann $B_{M,w}$ diese Bedingung einfach testen.
- C_l ist akzeptierende Konfiguration, wenn q_{accept} in C_l vorkommt. q_{accept} ist direkt aus $< M >$ ablesbar...
- für alle i muss man testen, ob C_{i+1} und C_i bis auf die Zeichen unter und neben dem Kopf bei C_i übereinstimmen; beide Stellen müssen entsprechened δ_M angepasst worden sein. $B_{M,w}$ überprüft das, indem er im Zickzack über beide Zeichenfolgen läuft und diese vergleicht

Beweis: (von Satz 8)
Annahme: E_{LBA} wäre entscheidbar und R wäre der Entscheider.

$S =$

„Auf Eingabe $< M, w >$ für Turingmaschine M und Wort w:

1. Konstruiere einen linear beschränkten Automaten $B_{M,w}$, sd.

   $$L(B_{M,w}) = \{\text{akzeptierende Berechnungshistorien von } M \text{ auf } w\}$$

2. Starte R mit der Eingabe $< B_{M,w} >$.
3. Falls R ablehnt, accept; falls R akzeptiert, reject"

S ist Entscheider für A_{TM}, das ist Widerspruch zu Unentscheidbarkeit von A_{TM}, also war die Annahme falsch, d.h. E_{LBA} ist unentscheidbar. □

Erzeugt eine Grammatik alle Wörter?

- Durch Reduktion mit Hilfe von Berechnungshistorien kann man auch die Unentscheidbarkeit von Problemen kontextfreier Grammatiken beweisen.
- Haben bereits gezeigt, dass E_{CFG} entscheidbar ist.
- ALL_{CFG} hingegen ist unentscheidbar.

$$ALL_{CFG} = \{< G >| \ G \text{ ist eine kontextfreie Grammatik und} L(G) = \Sigma^*\}$$

Satz

ALL_{CFG} ist unentscheidbar.

Beweis mit Berechnungshistorie

Satz

ALL_{CFG} ist unentscheidbar.

- Beweis durch Widerspruch und Reduktion von A_{TM}
- müssen aus Eingabe $< M, w >$ für A_{TM} kontextfreie Grammatik G konstruieren, die
  1. ganz Σ^* erzeugt, wenn w nicht von M akzeptiert wird, aber
  2. ein bestimmtes Wort **nicht** erzeugt, falls $w \in L(M)$, nämlich die akzeptierende Berechnungshistorie von M bei Eingabe w
- G erzeugt alle Wörter, die keine akzeptierende Berechnungshistorie von M für w sind.

Strategie zur Erzeugung von G

G soll alle Wörter erzeugen, die

1. **nicht** mit C_1 beginnen, oder
2. **nicht** mit C_l enden, oder
3. es ein C_{i+1}in der Folge gibt, das **keine Nachfolgekonfiguration** von C_i gemäß der Überführungsfunktion von M ist.

Konstruiere einen PDA mit dieser Sprache, der in CFG G „umgewandelt"werden kann

Konstruktion des Kellerautomaten

PDA D muss im Startzustand nichtdeterministisch verzweigen, um zu entscheiden, welche Bedingung geprüft werden soll.

Zweig 1	beginnt x mit C_1?	wenn nein, accept
Zweig 2	endet x mit akzeptierender Konf.?	wenn nein, accept
Zweig 3	gibt es einen ungültigen Übergang	wenn ja, accept

- für dritten Zweig wählt D nichtdeterministisch ein C_i aus und legt es auf den Stack
- Stack wird mit C_{i+1} verglichen, Bedingungen für Vergleich stammen von $< M >$
- Eingabe für D in der Form:

$$\sharp \underbrace{\rightarrow}_{C_1} \sharp \underbrace{\leftarrow}_{C_2^R} \sharp \underbrace{\rightarrow}_{C_3} \sharp \underbrace{\leftarrow}_{C_4^R} \sharp \dots \sharp \underbrace{\qquad\qquad}_{C_l} \sharp$$

Könnte man ALL_{CFG} entscheiden, dann wüsste man, ob akzeptierende Berechnungshistorie existiert, d.h. ob w zu $L(M)$ gehört.

Entscheidbarkeit von Sprachproblemen

Sprache	Typ 3	Typ 2	Typ 1	Typ 0
Modell	DEA	PDA	LBA	TM
Grammatik	regulär	kontextfrei	kontextsensitiv	allgemein
A	ja	ja	ja	nein
E	ja	ja	nein	nein
EQ	ja	nein	nein	nein
Schnitt	ja	nein	nein	nein