Abiturvorbereitung Informatik

Inhaltsverzeichnis

1 Sprachen & Automaten

1.1 Automaten

1.2 Sprachen

1.3 Grenzen

2 Berechenbarkeitstheorie

2.1 Entscheidbarkeit

Eine Sprache L ist entscheidbar genau dann, wenn die charakteristische Funktion χ berechenbar ist.

$$\chi_L(x) = \begin{cases} 1, & \text{wenn } x \in L \\ 0, & \text{wenn } x \notin L \end{cases} \tag{1}$$

2.2 Semi-Entscheidbarkeit

Eine Sprache L ist entscheidbar genau dann, wenn die partielle charakteristische Funktion χ berechenbar ist.

$$\chi_L(x) = \begin{cases} 1, & \text{wenn } x \in L \\ \text{undefiniert}, & \text{sonst} \end{cases} \tag{2}$$

bzw.

$$\chi_L(x) = \begin{cases} 0, & \text{wenn } x \notin L \\ \text{undefiniert}, & \text{sonst} \end{cases} \tag{3}$$

2.3 Berechenbarkeit

Ein Problem ist berechenbar, genau dann wenn ein Algorithmus zur Lösung des Problems existiert.

2.4 Rekursive Aufzählbarkeit

⇒NICHT Abzählbarkeit!

Menge M aus $A*$ heißt rekursiv aufzählbar, genau dann wenn

$$|M| = 0 \vee f : \mathbb{N} \rightarrow A * \wedge M = \{f(0), f(1), ...\} \tag{4}$$

Dabei muss f total und berechenbar sein.

Satz. *M ist semi-entscheidbar genau dann, wenn M rekursiv aufzählbar ist.*

Beweis. Zweiteilig:

1. M ist rekursiv aufzählbar $\Rightarrow M$ ist semi-entscheidbar:
   Solange $f(n) \neq w$, inkrementiere n
2. M ist semi-entscheidbar $\Rightarrow M$ ist rekursiv aufzählbar:
   Zähle alle w aus $A*$ auf und gibt diejenigen zurück, für die $\chi_M(x) = 1$ ist.

□

2.5 Das Wortproblem

$\Rightarrow W \in L(G)$?

Zu erkennende Sprache	Erkennender Automat
endliche Sprache	Zyklenfreier endlicher Automat
Typ-3-Sprache	EA (Endlicher Automat)
Typ-2-Sprache	PDA (Kellerautomat)
Typ-1-Sprache	LBA (linear beschränkter Automat)
Turingentscheidbare Sprache	TM (Turingmaschine)
Typ-0-Sprache	nicht allg. entscheidbar

Satz. *Typ-1-Sprachen sind entscheidbar.*

Beweis. Ansatz: Längenmonotonie
Verfolge alle Pfade der Bildungsvorschriften solange, bis der Ausdurck länger ist als das Wort. Entweder das Wort wird dabei gefunden oder es ist nicht in der Sprache enthalten. □

2.6 Probleme

2.6.1 Allgemein

Eine n-stellige Wortfunktion:

$$f : (A*)^n \to A*;\ n \in \mathbb{N} \tag{5}$$

Für eine n-stellige Eingabe gibt es eine Ausgabe.

2.6.2 Entscheidungsproblem

Eine n-stellige Wortfunktion:

$$f : (A*)^n \rightarrow \{0, 1\} \tag{6}$$

Für eine n-stellige Eingabe gibt es eine Ausgabe, wahr oder falsch.

2.6.3 Wieviele Probleme gibt es?

Gödel'scher Unvollständigkeitssatz. *In jedem widerspruchsfreien Axiomensystem gibt es Sätze, die nicht mit den Mitteln dieses Systems bewiesen werden können. (Kurt Gödel)*

Satz. *Es gibt mehr Probleme als Algorithmen.*

Beweis. Was ist ein Algorithmus? Eine Turingmaschine. Über Gödelisierung wird klar, dass sich jede Turingmaschine als natürliche Zahl darstellen lässt. Somit ist die Menge der Turingmaschinene abzählbar.
Was ist ein Problem? Eine n-stellige Wortfunktion. Jene Worte können aus einer beliebigen Menge bzw. Sprache stammen. Wir wählen die Menge der reellen Zahlen $\mathbb{R}$. Mit Hilfe der Cantor-Diagonalisierung können wir zeigen, dass $\mathbb{R}$ überabzählbar ist. Somit gibt es mehr Worte und damit auch mehr Wortfunktionen als Algorithmen. □

2.7 Turingberechenbarkeit

s. Turingmaschine

$f : A* \rightarrow A*$ heißt turingberechenbar genau dann, wenn eine Turingmaschine TM existiert, sodass für alle w_1, w_2 aus $A*$ gilt:

$$f(w_1) = w_2 \Leftrightarrow [TM - z_0, w_1] \rightarrow [TM - z_1, w_2] \tag{7}$$

Churche These. *Jede im intuitiven Sinn berechenbare Funktion ist auch turingberechenbar.*

2.8 Die Radó-Funktion

Die Rado-Funktion $\Sigma(n)$ gibt zurück, wie viele Zeichen eine terminierende Turingmaschine mit n Zuständen auf ein anfangs leeres Band schreiben kann. Dabei verfügt die Maschine nur über ein binäres Alphabet $\{0, 1\}$ und das leere Zeichen ist 0. Die Turingmaschine, die mit n Zuständen $\Sigma(n)$ Zeichen schreibt heißt fleißiger Bieber (engl. *busy beaver*).

Dabei ist $\Sigma(6) > 3.5 \cdot 10^{18267}$.

2.9 Ermittlung des Fleißigen Biebers

Alle Kandidaten können aufgezählt werden.

- beim binären Alphabet gibt es $2n$ mögliche Konstellationen, da in n Zuständen zwei Zeichen gelesen werden können.
- Außerdem gibt es $2 \cdot 2 \cdot n$ mögliche Reaktionen (Zwei Zeichen · Zwei Bewegungen $\cdot n$ Reaktionen)
- somit gibt es für n Zustände $(4n)^{2n}$ Kandidaten. Die probiert man aus...

3 Praktisch Unberechenbares

> *Berechenbarkeit ist nur praktikabel, wenn weniger Bits als Atome im Universum und weniger Rechenzeit als die Lebensdauer der Sonne erforderlich sind.*

⇒Exponentiell wächst zu schnell!

Rechenzeit:

Entspricht der Anzahl der benötigten Rechenschritte (Operationen mit konstanter Dauer)

- Worst-Case: Maximum aller Rechenzeiten für denkbare Eingaben
- Average-Case: Erwartungswert der Rechenzeit

3.1 Die Klassen P und NP

Klasse:

Menge mit Zugehörigkeitskriterium.

3.1.1 [P]olynomiell Berechenbares

Existiert ein Algorithmus *zur Lösung des Problems*, dessen Worst-Case-Laufzeit durch ein Polynom über der Problemgröße abschätzbar ist?

Robustheit von P:

Unabhängig von Rechnermodell variantenstabil:
$\Rightarrow$ Zahlprobleme (ZP), Entscheidungsprobleme (EP), Optimierungsprobleme (OP)

Umwandlung	Rechenzeit
ZP $\rightarrow$ EP	$O(1)$
EP $\rightarrow$ ZP	$O(log(N))$
ZP $\rightarrow$ OP	$O(N)$

Beispiele

- $\in P$:
  - Sortierprobleme
  - Kürzeste Wege
- $\notin P$:
  - Ackermann-Péter-Funktion

3.1.2 [N]ichtdeterministisch [P]olynomiell Berechenbares

Existiert ein Algorithmus *zur Überprüfung einer potentiellen Lösung*, dessen Worst-Case-Laufzeit durch ein Polynom über der Problemgröße abschätzbar ist?

Beispiele

- Hamilton-Pfad

MERKE:

$P \subseteq NP \Rightarrow P = NP \not\equiv P \subset NP$