Abiturvorbereitung Informatik

Inhaltsverzeichnis

1 Datenbanken

2 Datenstrukturen

Eine Datenstruktur ist ein Objekt zur Speicherung und Organisation von Daten. Dabei sind Daten immer so angeordnet, dass sie effizient, in Bezug auf Zeit und Speicher, für eine bestimmte Aufgabe genutzt werden können.

2.1 Schlangen

2.2 Stapel

2.3 Listen

2.4 Bäume

2.4.1 Grundbegriffe

Baum Besteht aus Knoten und Kanten; Zyklenfreier Graph

Wurzel Einziger Knoten im Baum, der keinen Vater hat

Kinder Menge von Knoten aus Sicht von Knoten k, deren Vater k ist

Höhe Abstand zum am weitesten entfernten Blatt

Niveau Abstand zur Wurzel

Wald Eine Menge an Bäumen

```
# include <vector>

struct Baum {
    Knoten* Wurzel;
};

struct Knoten {
    int Wert;              // Wert, den der Knoten speichert; muss nicht int sein
    Knoten* Vater;         // Pointer zu Vater
    Knoten* Kinder;        // Array mit Pointern zu Kindern
    int AnzahlKinder;      // Anzahl der Kinder im Array
    bool IstWurzel() {
        return Vater == 0; // Null-Pointer
    }
    bool IstBlatt() {
        return AnzahlKinder == 0;
    }
};
```

2.4.2 Binärbäume

Binärbäume sind spezielle Bäume, in denen jeder Knoten maximal zwei Kinder hat. Jeder Baum lässt sich als Binärbaum darstellen. (s. *left most child*)

2.4.3 Suchbäume

2.5 Warteschlangen

2.6 Graphen

3 Technische Grundlagen

3.1 Grundbegriffe

Bool'sche Algebra, Schaltalgebra, Aussage, Atome, Junktoren, Zweiwertigkeit, Extensionabilität, Kontraposition, Kettenschluss, DeMorgan'sche Regeln, Wahrheitstabellen

3.2 Bool'sche Funktionen

3.2.1 Vereinfachungen

3.2.1.1 Karnough-Veitch

3.2.1.2 Quine-McCluskey

3.3 Addierer

3.3.1 Halbaddierer

3.3.2 Volladdierer

3.3.3 Paralleladdierer

3.3.4 Serienaddierwerk

3.4 Flip-Flops

3.4.1 RS-Flip-Flop

3.4.2 D-Flip-Flop

3.4.3 MS-JK-Flip-Flop

4 Sprachen & Automaten

4.1 Automaten

4.1.1 Arten

Automaten lassen sich grundsätzlich in zwei Arten unterteilen:
Akzeptor:

$$A = (Z, z_0, x, f, F) \quad \text{(Akzeptor)} \tag{1}$$

$$Z \quad \text{(endliche Zustandsmenge)} \tag{2}$$

$$z_0 \in Z \quad \text{(Startzustand)} \tag{3}$$

$$x \quad \text{(Eingabealphabet)} \tag{4}$$

$$f \subseteq Z \times X \times Z \quad \text{(Zustandsüberführungsrelation)} \tag{5}$$

$$F \subseteq Z \quad \text{(akzeptierende Zustände)} \tag{6}$$

Transduktor:

$$T = (Q, \Sigma, \Gamma, q_0, \delta, F, \omega) \quad \text{(Transduktor)} \tag{7}$$
$$Q \quad \text{(endliche Zustandsmenge)} \tag{8}$$
$$\Sigma \quad \text{(Eingabealphabet: endliche, nicht-leere Menge von Symbolen)} \tag{9}$$
$$\Gamma \quad \text{(Ausgabealphabet: endliche, nicht-leere Menge von Symbolen)} \tag{10}$$
$$q_0 \in Q \quad \text{(Anfangszustand)} \tag{11}$$
$$\delta : Q \times \Sigma \cup \{\epsilon\} \to 2^Q \quad \text{(Zustandsübergangsfunktion)} \tag{12}$$
$$F \quad \text{(endliche Menge von Endzuständen)} \tag{13}$$
$$\omega : Q \times \Sigma \cup \{\epsilon\} \times Q \to \Gamma^* \quad \text{(Ausgabefunktion)} \tag{14}$$

4.1.2 Endliche Automaten (EA)

4.1.3 Kellerautomaten (PDA)

4.1.4 Linear beschränkte Automaten (LBA)

4.1.5 Turingmaschinen (TM)

4.2 Sprachen

Grammatiken beschreiben Sprachen und es ist möglich Grammatiken nach der Chomsky-Hierarchie zu unterteilen.

Typ	Grammatik	benötigter Automat
Typ-3-Grammatik	reguläre Grammatik	EA (Endlicher Automat)
Typ-2-Grammatik	kontextfreie Grammatik	PDA (Kellerautomat)
Typ-1-Grammatik	kontextsensitive Grammatik	LBA (linear beschränkter Automat)
Typ-0-Grammatik	allgemeine Grammatik	Turingmaschine

4.3 Grenzen

5 Berechenbarkeitstheorie

5.1 Entscheidbarkeit

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

$$\chi_L(x) = \begin{cases} 1, & \text{wenn } x \in L \\ 0, & \text{wenn } x \notin L \end{cases} \tag{15}$$

5.2 Semi-Entscheidbarkeit

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

$$\chi_L(x) = \begin{cases} 1, & \text{wenn } x \in L \\ \text{undefiniert}, & \text{sonst} \end{cases} \tag{16}$$

bzw.

$$\chi_L(x) = \begin{cases} 0, & \text{wenn } x \notin L \\ \text{undefiniert}, & \text{sonst} \end{cases} \tag{17}$$

5.3 Berechenbarkeit

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

5.4 Rekursive Aufzählbarkeit

$\Rightarrow NICHT Abzhlbarkeit!$

$Menge$MausA*$heitrekursivaufzhlbar, genaudannwenn|M| = 0 \vee f : \mathbb{N} \to A* \wedge M = \{f(0), f(1), ...\}$(18)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$istsemi - entscheidbar$:
   $Solange$f(n) $\neq w$, inkrementiere n
2. M ist semi-entscheidbar $\Rightarrow$M$istrekursivaufzhlbar$:
   $Zhlealle$waus A*$aufundgibtdiejenigenzurck, frdie\chi_M(x) = 1$ ist. □

5.5 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. □

5.6 Probleme

5.6.1 Allgemein

Eine n-stellige Wortfunktion:

$$f : (A*)^n \to A*;\ n \in \mathbb{N} \tag{19}$$

Für eine n-stellige Eingabe gibt es eine Ausgabe.

5.6.2 Entscheidungsproblem

Eine n-stellige Wortfunktion:

$$f : (A*)^n \to \{0, 1\} \tag{20}$$

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

5.6.3 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. □

5.7 Turingberechenbarkeit

s. Turingmaschine

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

$$f(w_1) = w_2 \Leftrightarrow [TM - z_0, w_1] \to [TM - z_1, w_2] \tag{21}$$

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

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

5.8.1 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...

5.8.2 Unberechenbarkeit der Radó-Funktion

Satz. *Die Funktion Σ ist nicht berechenbar.*

Beweis. Wir zeigen, dass Σ schneller wächst als jede berechenbare Funktion f.
Dazu brauchen wir drei Turingmaschinen:

- M_N druckt n Striche auf ein leeres Band, weniger als n zustände.
- M_D verdoppelt die Anzahl der Striche auf dem Band, hat c Zustände.
- M_F berechnet f mit p Zuständen.

Wir verknüpfen: $M_N|M_D|M_F$
$\Rightarrow$f(2n)$Strichemit$p+n+c$Zustnden.$
$lt.Definitionist\Sigma(p+n+c)$ mindestens so groß wie $f(2n)$, weil mit $p+n+c$ Zuständen nun $f(2n)$ Striche auf das Band gemalt werden können.
$\Rightarrow\Sigma(n+p+c) \geq f(2n)$
mit $n \to \infty$: $n > p+c$
$\Rightarrow \Sigma(2n) > \Sigma(p+n+c) \geq f(2n)$
$\Rightarrow\Sigma(2n) > f(2n)$ □

5.9 Satz von Rice

Satz von Rice. *Es gibt keinen Algorithmus, der in der Lage ist zu entscheiden, ob eine beliebige Turingmaschine (bzw. Funktion) eine nicht triviale Eigneschaft hat, oder nicht.*

Dabei ist eine triviale Eigenschaft solch eine, die entweder Teil aller Turingmaschinen ist oder gar keiner.

Somit ist es unmöglich die simpelsten Eigenschaften allgemein für alle Programme zu bestimmen.

6 Praktisch Unberechenbares

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

$\Rightarrow Exponentiellwchstzuschnell!$

6.0.0.1 Rechenzeit:

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

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

6.1 Die Klassen P und NP

Klasse: Menge mit Zugehörigkeitskriterium.

$$x \in K \Leftrightarrow x \text{ hat Kriterium für } K \tag{22}$$

6.1.1 [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 (i.d.R. $O(n\ log(n))$)
  - Kürzeste Wege
- $\notin P$:
  - Ackermann-Péter-Funktion

6.1.2 [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:

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

Hamilton-Pfad/Kreis (HAM-P/K)

Existiert in einem gegebenen Graphen einen Pfad/Kreis, der jeden Knoten genau ein mal enthält (und dabei keine Kante doppelt verwendet)?

Satz. *Das Finden eines Hamilton-Kreises bzw. eines Hamilton-Pfades ist ein Problem aus NP.*

Beweis. Lösungen lassen sich in polynomialzeit Überprüfen indem nacheinander jeder Knoten des Pfades besucht wird. Sollte zwischen zwei in der Lösung aufeinanderfolgenden Knoten keine Kante existieren, so ist die Lösung ungültig. Im Falle des Hamilton-Kreises muss natürlich überprüft werden, ob eine Kante vom letzten zum ersten Knoten existiert.
Das besuchen aller Knoten wächst linear mit der Anzahl der Knoten, somit erfolgt die Überprüfung in $O(N)$.
□

6.2 Die Klasse NPC

Frage: $P = NP$ oder $P \subset NP$?

6.2.1 NP-Vollständigkeitstheorie

Man nehme das schwerste Problem aus NP und versuche es in P-Zeit zu lösen.
Entweder das funktioniert, dann ist $P = NP$, oder das funktioniert nicht, dann ist $P \subset NP$.

6.2.2 Was sind die schwersten NP-Probleme?

- Lösung beinhaltet Lösung aller Probleme in NP.
- Lösung aller anderer Probleme in NP kann in P-Zeit auf die Lösung der NPC-Probleme zurückgeführt werden.

6.2.3 Definition

Menge aller Probleme aus NP, auf die alle Probleme aus NP in P-Zeit zurückführbar sind.

Merke. *Alle Probleme aus NPC sind in P-Zeit aufeinander rückführbar und somit gleich schwer.*

6.2.4 Erfüllbarkeitsproblem (SAT)

Für einen bool'schen Term T ist eine Variablenbelegung B zu finden, die dazu führt, dass T wahr wird. Dabei sind Terme in disjunktiver Normalform trivial und meistens nicht interessant.

6.2.5 Travelling Salesman Problem (TSP)

Es ist ein möglichst kurzer Hamilton-Kreis in einem gegebenen Graphen zu finden. Dabei kann entweder das Entscheidungsproblem gemeint sein, wobei gefragt wird ob ein Hamilton-Kreis existiert, der die Länge x nicht überschreitet, oder es geht um das Optimierungsproblem, wobei es den kürzesten Hamilton-Kreis zu finden gilt.

Lösungsansätze

- Durchlaufen aller Permuationen
- Dynamisch
- Branch and Bound

Anwendungen

- Routenplanung
- Finden von Superpermutationen
- Lochkartenautomaten

6.2.6 Färbungsproblem (COL)

Es ist eine Belegung der Knoten eines Graphen mit sogenannten Farben, d.h. Werten, zu finden, die sicherstellt, dass keine zwei benachbarten Knoten die gleiche Färbung haben.

Dabei ist beim Entscheidungsproblem die Frage zu beantworten, ob eine Färbung mit x Farben existiert. Beim Optimierungsproblem hingegen ist eine der Färbungen zu finden, die die wenigsten Farben enthalten.

Anwendung

- Mobilfunknetze
- Landkarten
- Zeitplanung

6.2.7 Cliquenproblem (CLIQUE)

Es ist eine möglichst große Menge an Knoten zu finden in der alle Elemente paarweise durch direkte Kanten miteinander verbunden sind. Dabei kann entweder das Entscheidungsproblem gemeint sein, wobei gefragt wird ob eine Clique der Größe x existiert oder das Optimierungsproblem, wobei es die größte Clique zu finden gilt.

Dabei sind nicht zwingend alle Variationen in NPC, da sich beispielsweise alle potentiellen Cliquen der Größe x in $O(\frac{N!}{(N-x)!}) \approx O(N^x)$, also in P-Zeit überprüfen lassen.

Anwendung

- Zimmerbelegung

6.2.8 Rucksackproblem (KNAPSACK)

Gegeben sei eine endliche Menge an Objekten U, denen die Funktionen w und v ein festes Gewicht sowie einen festen Wert zuordnen. Mit einer gegebenen Gewichtsschranke W ist nun jene Teilmenge von U zu bestimmen, die einerseits in Summe W nicht überschreitet, aber andererseits die Summe der Werte maximiert.

6.2.9 Teilsummenproblem (SUBSET-SUM)

Gegeben sei eine endliche Menge an Zahlen Z, die Summanden, sowie eine Zielzahl x, die Summe. Nun ist eine Menge $z \subseteq Z$ zu finden, für die gilt: $x = \sum_{i=1}^{|z|} z_i$.

6.2.10 Reduktionen

6.2.10.1 $NCOL \leq_p SAT$

Für jeden Knoten führen wir Variablen für alle möglichen Farben ein ($N \cdot |V|$). Nun benötigen wir für jeden Knoten Klauseln, die besagen, dass genau eine Farben wahr sein muss. Des Weiteren wird nun für jede Kante eine Klausel aufgeschrieben, die besagt, dass die beiden Knoten nicht die gleiche Farbe haben dürfen.

Somit können Knoten nur genau eine Farbe haben und benachbarte Knoten können nicht die gleiche Farbe haben. Sollte es eine Variablenbelegung geben, die das erfüllt, so gibt es eine Färbung für den Graphen mit N Farben.

6.2.10.2 $HAM \leq_p TSP$

Gesucht ist ein Hamilton-Kreis auf $G = (V, E)$. Wir erstellen einen Graphen G' mit der gleichen Anzahl an Knoten. Nun erstellen wir alle Kanten in G', die in G exisitieren mit dem Gewicht 1. Alle weiteren möglichen Kanten erhalten ein beliebiges Gewicht, welches Größer als 1 sein muss. Auf diesem Graphen lassen wir das TSP laufen, welches uns die Länge der kürzesten Rundreise zurückgibt.

Sollte diese Rundreise haben, die gleich $|V|$ ist, so wissen wir, dass es in G einen Hamilton-Kreis gibt. Sollte diese Rundreise länger sein, wissen wir, dass dieser Hamilton-Kreis nicht exisitiert, weil das TSP mindestens eine weitere Kante benötigt hat, obwohl eine Hamiltonkreis mit den Kantenlängen 1 definitiv kürzer gewesen wäre.

7 Nebenläufige Prozesse

7.1 Petri-Netze

7.1.1 Definition

$$N = (P, T, F, W, C, m_0) \quad \text{(Petri-Netz)} \tag{23}$$

$$P \cap T = \emptyset \quad \text{(Disjunktheit)} \tag{24}$$

$$|P| > 0 \quad \text{(endl. Menge an Stellen)} \tag{25}$$

$$|T| > 0 \quad \text{(endl. Menge an Transitionen)} \tag{26}$$

$$F \subseteq (P \times T) \cup (T \times P) \quad \text{(Flussrelation)} \tag{27}$$

$$W : F \to \mathbb{N} \quad \text{(Gewichte)} \tag{28}$$

$$C : P \to \mathbb{N} \quad \text{(Kapazitäten)} \tag{29}$$

$$m_0 : P \to \mathbb{N}_0 \quad \text{(Anfangsmarkierung)} \tag{30}$$

7.1.2 Schaltregel

Eine Transition t kann genau dann schalten, wenn alle Stellen im Vorbereich mindestens so viele Marken haben, wie durch die Kantengewichte beschreiben, und alle Stellen im Nachbereich nur so viele Marken haben, dass das Schalten der Transition diese nicht zum überlaufen bringt. Wenn t schaltet, werden aus jeder Stelle des Vorbereiches so viele Marken entfernt, wie durch die Kantengewichte beschrieben. Des Weiteren werden jeder Stelle des Nachbereiches so viele Stellen hinzugefügt, wie durch die Kantengewichte beschrieben.

7.1.3 Begriffe

Vorbereich Der Vorbereich eines Knotens $x \in P \cup T$ ist die Menge aller Knoten, die ausgehende Kanten zu x haben.

Nachbereich Der Nachbereich eines Knotens $x \in P \cup T$ ist die Menge aller Knoten, die eingehende Kanten von x haben.