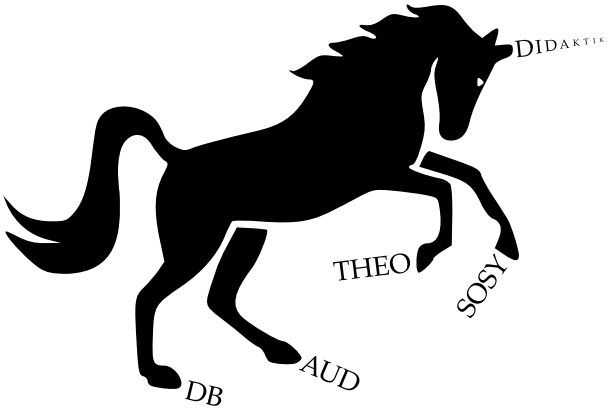
66115
Frühjahr
2013

Theoretische Informatik / Algorithmen (vertieft)

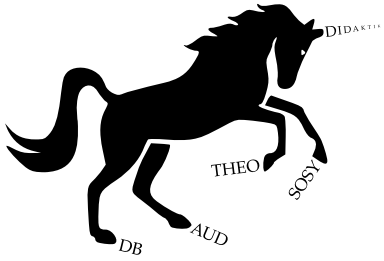
Aufgabenstellungen mit Lösungsvorschlägen



Die Bschlangaul-Sammlung

Hermine Bschlangaul and Friends

Aufgabenübersicht



Die Bschlangaul-Sammlung
Hermine Bschlangaul and Friends

Eine freie Aufgabensammlung mit Lösungen von Studierenden für Studierende zur Vorbereitung auf die 1. Staatsexamensprüfungen des Lehramts Informatik in Bayern.

Thema Nr. 1

Aufgabe 2 [Kontextfreie Grammatiken]

Gegeben sei die Grammatik $G = (\{S, A, B, C\}, \{a, b\}, P, S)$ und

$$
\mathrm{P} = \left\{ \begin{array}{l}
S \to AB \\
S \to CS \\
A \to BC \\
A \to BB \\
A \to a \\
B \to AC \\
B \to b \\
C \to AA \\
C \to BA
\end{array} \right\}
$$

Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/Gr46a6j0a

$L = L(G)$ ist die von G erzeugte Sprache.

(a) Zeigen Sie, dass G mehrdeutig ist.

Lösungsvorschlag

Das Wort *baab* kann in zwei verschiedenen Ableitungen hergeleitet werden:

(i) $S \vdash AB \vdash BCB \vdash bCB \vdash bAAB \vdash baAB \vdash baaB \vdash baab$

(ii) $S \vdash CS \vdash BAS \vdash bAS \vdash baS \vdash baAB \vdash baaB \vdash baab$

(b) Entscheiden Sie mithilfe des Algorithmus von Cocke, Younger und Kasami (CYK), ob das Wort $w = babaaa$ zur Sprache L gehört. Begründen Sie Ihre Entscheidung.

Lösungsvorschlag

b	a	b	a	a	a
B	A	B	A	A	A
C	S	C	C	C	
-	B	A	B		
A	C	A,C			
A,C	B,C,A				
A,C,B					

$\Rightarrow babaaa \notin L(G)$

Das Startsymbol S ist nicht in der Zelle $V(1,5) = \{A, C, B\}$ enthalten.

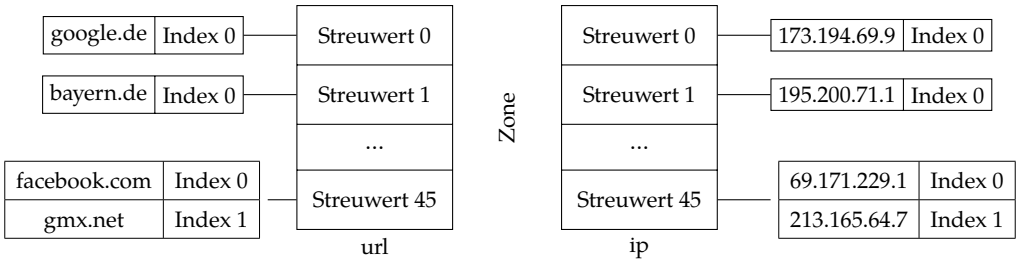
(c) Geben Sie eine Ableitung für $w = babaaa$ an.

Lösungsvorschlag

$A \vdash BB \vdash bB \vdash bAC \vdash baC \vdash baAA \vdash baBCA \vdash babCA \vdash babAAA \vdash babaAA \vdash babaaA \vdash babaaa$

Aufgabe 6 [IP und ULR mit Hashes]

Um die URL (zum Beispiel google.de) und die zugehörige IP des Servers (hier 173.194.69.9) zu verwalten, werden Streutabellen verwendet, die eine bestimmte Zone von Adressen abbilden. Die Streutabellen werden als zwei dynamische Arrays (in Java: ArrayLists) realisiert. Kollisionen innerhalb einer Zone werden ebenfalls in dynamischen Arrays verwaltet.



Um zu einer URL die IP zu finden, berechnet man zunächst mittels der Funktion `hash()` den entsprechenden Streuwert, entnimmt dann den Index der Tabelle URL und sucht schließlich an entsprechender Stelle in der Tabelle IP die IP-Adresse.

(a) Erläutern Sie am vorgestellten Beispiel, wie ein Hash-Verfahren zum Speichern großer Datenmengen prinzipiell funktioniert und welche Voraussetzungen und Bedingungen daran geknüpft sind.

(b) Nun implementieren Sie Teile dieser IP- und URL-Verwaltung in einer objektorientierten Sprache Ihrer Wahl. Verwenden Sie dabei die folgende Klasse (die Vorgaben sind in der Sprache Java gehalten):

```
class Zone {
  private ArrayList<ArrayList<String>> urlList =
    new ArrayList<ArrayList<String>>();
  private ArrayList<ArrayList<String>> ipList =
    new ArrayList<ArrayList<String>>();
  public int hash(String url) {/* calculates hash-value h, >=0 */}
}
```

(i) Prüfen Sie in einer Methode `boolean exists(int h)` der Klasse `Zone`, ob bereits mindestens ein Eintrag für einen gegebenen Streuwert vorhanden ist. Falls `h` größer ist als die derzeitige Größe der Streutabelle, existiert der Eintrag nicht.

Lösungsvorschlag

```
boolean exists(int h) {
  if (urlList.size() - 1 < h || ipList.size() - 1 < h)
    return false;

  ArrayList<String> urlCollisionList = urlList.get(h);
  ArrayList<String> ipCollisionList = ipList.get(h);
  if (urlCollisionList.size() == 0 || ipCollisionList.size() == 0)
    return false;

  return true;
}
```

Code-Beispiel auf Github ansehen: src/main/java/org/bschlangaul/examen/examen_66115/jahr_2013/fruehjahr/Zone.java

(ii) Die Methode `int getIndex (string url, ArrayList<String> urlList)` soll den Index einer URL in der Kollisionsliste berechnen. Ist die URL in der Kollisionsliste nicht vorhanden, soll `-1` zurückgeliefert werden.

Lösungsvorschlag

```
int getIndex(String url, ArrayList<String> urlList) {
  for (int i = 0; i < urlList.size(); i++) {
    if (urlList.get(i).equals(url))
      return i;
  }
  return -1;
}
```

Code-Beispiel auf Github ansehen: src/main/java/org/bschlangaul/examen/examen_66115/jahr_2013/fruehjahr/Zone.java

(iii) Ergänzen Sie die Klasse Zone um eine Methode `String lookup (String url)`, die in der Streutabelle die IP-Adresse zur `url` zurückgibt. Wird eine nicht vorhandene Adresse abgerufen, wird eine Fehlermeldung zurückgegeben.

Lösungsvorschlag

```
String lookup(String url) {
  int h = hash(url);
  int collisionIndex = getIndex(url, urlList.get(h));
  if (collisionIndex == -1)
    return "Die URL kannte nicht in der Tabelle gefunden werden";
  return ipList.get(h).get(collisionIndex);
}
```

Code-Beispiel auf Github ansehen: src/main/java/org/bschlangaul/examen/examen_66115/jahr_2013/fruehjahr/Zone.java

Thema Nr. 2

Aufgabe 5 [Drei Missionare und drei Kannibalen]

Drei Missionare und drei Kannibalen befinden sich am Ufer eines Flusses und möchten diesen überqueren. Dazu steht ihnen ein Boot zur Verfügung, das ein oder zwei Personen befördern kann. Verbleiben an einem Ufer mehr Kannibalen als Missionare, so werden die Missionare verspeist. Die Frage besteht nun darin, wie alle Personen unversehrt auf die andere Seite des Flusses gelangen.

Im Anfangszustand befinden sich alle Personen und das Boot auf einer Seite des Flusses. Nehmen Sie an, es sei die linke Seite des Flusses. Im Zielzustand befinden sich alle Personen und das Boot auf der anderen Seite des Flusses. Jeden Zustand kann man durch folgendes Fünftupel beschreiben:

Missionareyngs X Kannibalen;;„xs X Missionareyeens X Kannibalenyechıs X Boolposition

Dabei werden die Elemente für Missionare und Kannibalen durch natürliche Zahlen und die Bootsposition durch einen der Strings „links“ oder „rechts“ modelliert. Der Anfangszustand wäre somit also (3, 3, 0, 0, » links”) und der Zielzustand (0, 0, 3, 3, „rechts‘“).

Schreiben Sie Algorithmen zur Lösung dieses Suchproblems und retten Sie die Missionare! Es ist empfehlenswert (aber nicht zwingend), hierzu eine funktionale Programmiersprache zu verwenden. Gehen Sie im Einzelnen vor, wie folgt:

(a) a.) Schreiben Sie eine Funktion, die alle möglichen Überfahrten von links nach rechts oder umgekehrt modelliert, d. h. die zu einem gegebenen Zustand die Liste der möglichen Folgezustände im Sinne der o. g. Regeln berechnet. Gehen Sie dazu von allen möglichen Überfahrten aus und überprüfen Sie für jede konkrete Überfahrt mittels einer geeigneter Funktion, ob diese zu einem zulässigen Zustand führt. Zustände, die die Missionare nicht überleben, gelten im Sinne des Rettungsvorhabens ebenfalls als nicht zulässig. Nicht zulässige Zustände werden nicht in die Liste der möglichen Folgezustände eingefügt.

(b) b.) Geben Sie eine Funktion an, die feststellt, ob ein Zyklus vorliegt, d. h. ob ein Zustand in einer Liste bereits besuchter Zustände schon enthalten ist.

(c) c.) Verwenden Sie Ihre Ergebnisse aus a.) und b.), um eine Funktion anzugeben, die dieses Suchproblem mittels Breitensuche löst. (Sie können die Funktionen aus a.) und b.) hier auch dann verwenden, wenn Sie diese Teilaufgaben nicht vollständig gelöst haben.) Die Funktion erhält als Eingabe einen Start- und einen Zielzustand und liefert als Ergebnis die erste gefundene Liste von Zuständen, die das Problem löst.