

66115 Frühjahr 2013

Theoretische Informatik / Algorithmen (vertieft)

Aufgabenstellungen mit Lösungsvorschlägen



Die Bschlangaul-Sammlung

Hermine Bschlangaul and Friends

Aufgabenübersicht

Thema Nr. 1	3
Aufgabe 2 [Kontextfreie Grammatiken]	3
Aufgabe 6 [IP und ULR mit Hashes]	4
 Thema Nr. 2	 6



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.



Diese Materialsammlung unterliegt den Bestimmungen der Creative Commons Namensnennung-Nicht kommerziell-Share Alike 4.0 International-Lizenz.

Thema Nr. 1

Aufgabe 2 [Kontextfreie Grammatiken]

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

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

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.

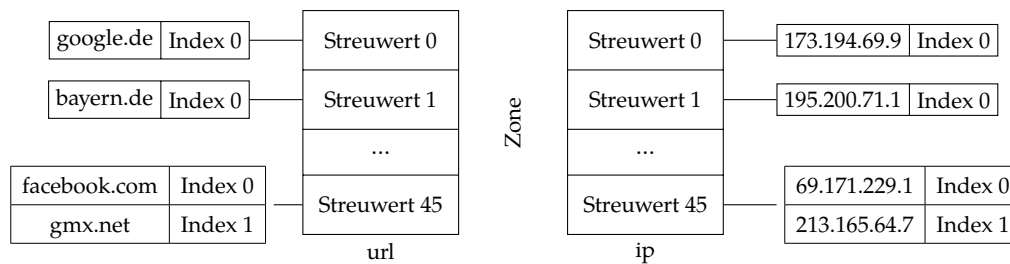
(c) Geben Sie eine Ableitung für $w = babaaa$ an.

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 URL 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.

- Erläutern Sie am vorgestellten Beispiel, wie ein Hash-Verfahren zum Speichern großer Datenmengen prinzipiell funktioniert und welche Voraussetzungen und Bedingungen daran geknüpft sind.
- 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 */
    }
```

- 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.

```

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](https://github.com/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.

```

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](https://github.com/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.

```

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](https://github.com/bschlangaul/examen/examen_66115/jahr_2013/fruehjahr/Zone.java)

Thema Nr. 2