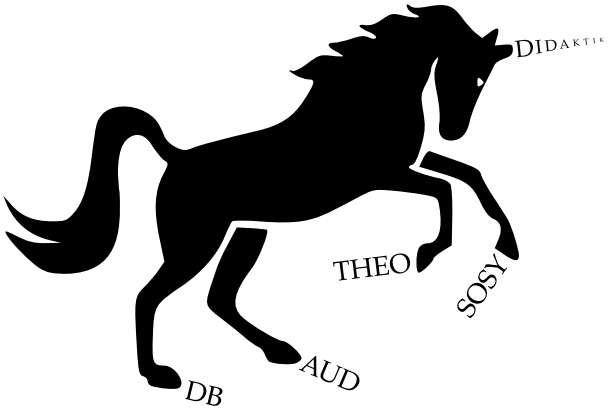
66115
Frühjahr
2015

Theoretische Informatik / Algorithmen (vertieft)

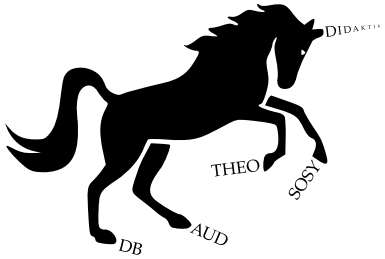
Aufgabenstellungen mit Lösungsvorschlägen



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 1 [Alphabet „01“ Anzahl Unterschied höchstes 3]

Die Sprache L über den Alphabet $\Sigma = \{0, 1\}$ enthält alle Wörter, bei denen beim Lesen von links nach rechts der Unterschied in der Zahl der 0en und 1en stets höchstens 3 ist. Also ist $w \in L$ genau dann, wenn für alle u, v mit $w = uv$ gilt $||u|_0 - |u|_1| \leq 3$. Erinnerung: $|w|_a$ bezeichnet die Zahl der a's im Wort w.

(a) Sei $A = (Q, \Sigma, \delta, q_0, E)$ ein deterministischer endlicher Automat für L. Es sei $w_1 = 111$, $w_2 = 11$, $w_3 = 1$, $w_4 = \varepsilon$, $w_5 = 0$, $w_6 = 00$, $w_7 = 000$. Machen Sie sich klar, dass der Automat jedes dieser Wörter verarbeiten können muss. Folgern Sie, dass der Automat mindestens sieben Zustände haben muss. Schreiben Sie Ihr Argumentation schlüssig und vollständig auf.

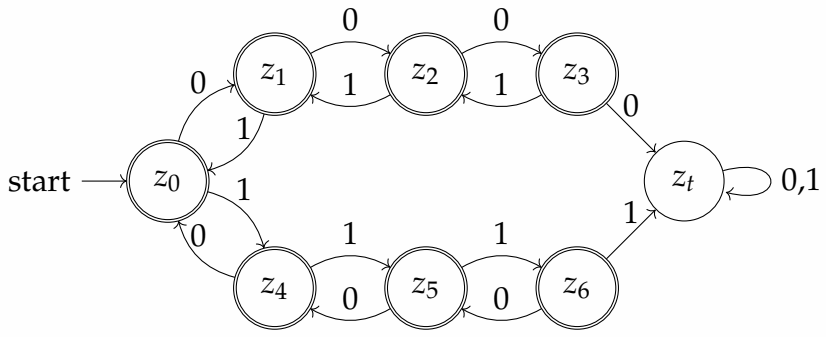
Lösungsvorschlag

Ein deterministischer endlicher Automat hat keinen zusätzlichen Speicher zur Verfügung, in dem die Anzahl der bisher vorkommenden Einsen und Nullen gespeichert werden könnte. Ein deterministischer endlicher Automat kann die von der Sprache benötigten Anzahl an Einsen und Nullen nur in Form von Zustanden speichern. Um die Anzahl von 3 Einsen bzw. 3 Nullen zu speichern, sind also 6 Zustände nötig.

Die Wörter 01 oder 0011 oder 0101 etc. haben eine Differenz von 0, wenn die Anzahl an Nullen und Einsen abzogen wird. Um auch diese Wörter darstellen zu können, ist mindestens ein weiterer Zustand nötig.

(b) Begründen Sie, dass L regulär ist.

Lösungsvorschlag



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/Ait6va31c

(c) Jemand behauptet, diese Sprache sei nicht regulär und gibt folgenden „Beweis“ dafür an: Wäre L regulär, so sei n eine entsprechende Pumping-Zahl. Nun ist $w = (01)^n \in L$. Zerlegt man nun $w = uxv$, wobei $u = 0$, $x = 1$, $v = (01)^{n-1}$, so ist zum Beispiel $ux^5v \notin L$, denn es ist $ux^5v = 01111101010101....$ Legen Sie genau dar, an welcher Stelle dieser „Beweis“ fehlerhaft ist.

Exkurs: Pumping-Lemma für Reguläre Sprachen

Es sei L eine reguläre Sprache. Dann gibt es eine Zahl j, sodass für alle Wörter $\omega \in L$ mit $|\omega| \geq j$ (jedes Wort ω in L mit Mindestlänge j) jeweils eine Zerlegung $\omega = uvw$ existiert, sodass die folgenden Eigenschaften erfüllt sind:

(i) $|v| \geq 1$ (Das Wort v ist nicht leer.)

(ii) $|uv| \leq j$ (Die beiden Wörter u und v haben zusammen höchstens die Länge j.)

(iii) Für alle $i = 0, 1, 2, \ldots$ gilt $uv^i w \in L$ (Für jede natürliche Zahl (mit 0) i ist das Wort $uv^i w$ in der Sprache L)

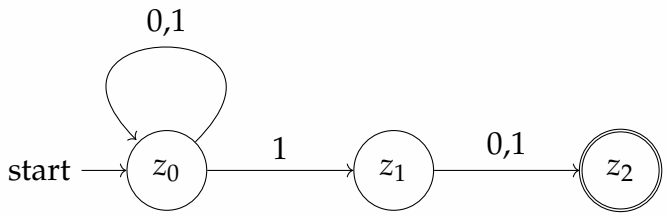
Die kleinste Zahl j, die diese Eigenschaften erfüllt, wird Pumping-Zahl der Sprache L genannt.

Lösungsvorschlag

Das Wort $(01)^n$ wurde falsch zerlegt. Für die Pumping-Zahl $n = 3$ gibt es sehr wohl eine Zerlegung, die beim Aufpumpen regulär ist, also: $\omega = 010101$ ($u = 01$, $x = 01$ und $v = 01$). $ux^5v = 01010101010101 \in L$. Es gibt also eine Zerlegung, die beim Aufpumpen die 3 Pumping-Lemma-Eigenschaften erfüllt. Daher kann man das Pumping-Lemma so nicht widerlegt werden, indem man ein einziges Gegenbeispiel gibt.

(d) In anderen Fällen können nichtdeterministische endliche Automaten echt kleiner sein als die besten deterministischen Automaten. Ein Beispiel ist die Sprache $L_2 = \Sigma^* 1 \Sigma$ aller Wörter, deren vorletztes Symbol 1 ist. Geben Sie einen nicht-deterministischen Automaten mit nur drei Zuständen an, L_2 erkennt.

Lösungsvorschlag



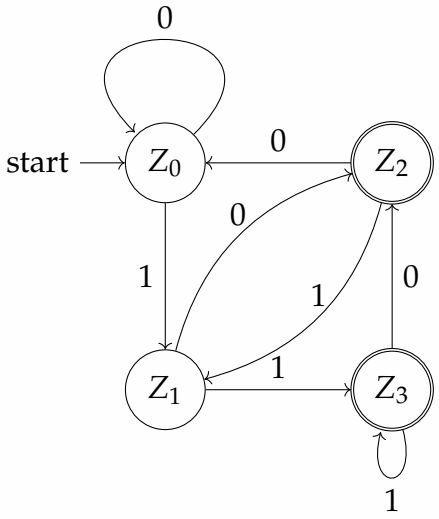
Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/Apwzjufbg

(e) Führen Sie auf Ihrem Automaten die Potenzmengenkonstruktion und anschließend den Minimierungsalgorithmus durch. Wie viele Zustände muss ein deterministischer Automat für L_2 also mindestens haben?

Lösungsvorschlag

Potenzmengenkonstruktion

Name	Zustandsmenge	Eingabe 0	Eingabe 1
Z_0	Z_0 $\{z_0\}$	Z_0 $\{z_0\}$	Z_1 $\{z_0, z_1\}$
Z_1	Z_1 $\{z_0, z_1\}$	Z_2 $\{z_0, z_2\}$	Z_3 $\{z_0, z_1, z_2\}$
Z_2	Z_2 $\{z_0, z_2\}$	Z_0 $\{z_0\}$	Z_1 $\{z_0, z_1\}$
Z_3	Z_3 $\{z_0, z_1, z_2\}$	Z_2 $\{z_0, z_2\}$	Z_3 $\{z_0, z_1, z_2\}$



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/Ajfcofpb9

Minimierungsalgorithmus

Z_0	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$
Z_1	x_2	$\varnothing$	$\varnothing$	$\varnothing$
Z_2	x_1	x_1	$\varnothing$	$\varnothing$
Z_3	x_1	x_1	x_2	$\varnothing$
	Z_0	Z_1	Z_2	Z_3

x_1 Paar aus End-/ Nicht-Endzustand kann nicht äquivalent sein.

x_2 Test, ob man mit der Eingabe zu einem bereits markiertem Paar kommt.

x_3 In weiteren Iterationen markierte Zustände.

x_4 ...

Übergangstabelle

Zustandspaar	**0**	**1**
(Z_0, Z_1)	(Z_0, Z_2) x_2	(Z_1, Z_3) x_2
(Z_2, Z_3)	(Z_0, Z_1)	(Z_1, Z_3) x_2

Wie aus der oben stehenden Tabelle abzulesen ist, gibt es keine äquivalenten Zustände. Der Automat kann nicht minimiert werden. Er ist bereits minimal.

Thema Nr. 2

Aufgabe 4 [Gödelisierung aller Registermaschinen (RAMs)]

Sei M 0 , M 1 , . . . eine Gödelisierung aller Registermaschinen (RAMs). Geben Sie für die folgenden Mengen D 1 , D 2 , D 3 an, ob sie entscheidbar oder aufzählbar sind. Begründen Sie Ihre Behauptungen, wobei Sie die Aufzählbarkeit und Un- entscheidbarkeit des speziellen Halteproblems K 0 = x ∈ N|M x haelt bei Eingabe x verwenden dürfen. D 1 = x ∈ N|x < 9973und M x haelt bei Eingabe x D 2 = x ∈ N|x ≥ 9973und M x haelt bei Eingabe x D 3 = x ∈ N|M x haelt nicht bei Eingabe x

Lösungsvorschlag

D 1 ist eine endliche Menge und damit entscheidbar. Auch eine endliche Teilmenge des Halteproblems. Anschaulich kann man sich dies so verstellen: Man stellt dem Rechner eine Liste zur Verfügung, die alle haltenden Maschinen M x mit x < 9973 enthält. Diese Liste kann zum Beispiel vorab von einem Menschen erstellt worden sein, denn die Menge der zu prüfenden Programme ist endlich.

D 2 x ≥ 9973 entscheidbar, L halt semi-entscheidbar → semi-entscheidbar (Hier wäre auch eine Argumentation über die Cantorsche Paarungsfunktion möglich). Es ist weiterhin nicht entscheidbar. Dazu betrachten wir dei Reduktin des speziel- len Halteproblems H 0 : H 0 ≤ D 2 Für alle x < 9973 lassen wir M x durch eine Turingmaschine M y simulieren, die eine höhere Nummer hat.

D 3 ist unentscheidbar, denn angenommen D 3 wäre semi-entscheidbar, dann würde sofort folgen, dass L halt entscheidbar ist, da aus der Semientscheidbarkeit von L halt und L halt die Entscheidbarkeit von L halt folgen würde

Aufgabe 4 [Sortieren mit Stapel]

Gegeben seien die Standardstrukturen Stapel (Stack) und Schlange (Queue) mit folgenden Standardoperationen:

Stapel	Schlange
`boolean isEmpty()`	`boolean isEmpty()`
`void push(int e)`	`enqueue(int e)`
`int pop()`	`int dequeue()`
`int top()`	`int head()`

Beim Stapel gibt die Operation `top()` das gleiche Element wie `pop()` zurück, bei der Schlange gibt `head()` das gleiche Element wie `dequeue()` zurück. Im Unterschied zu `pop()`, beziehungsweise `dequeue()`, wird das Element bei `top()` und `head()` nicht aus der Datenstruktur entfernt.

(a) Geben Sie in Pseudocode einen Algorithmus `sort(Stapel s)` an, der als Eingabe einen Stapel `s` mit `n` Zahlen erhält und die Zahlen in `s` sortiert. (Sie dürfen die Zahlen

wahlweise entweder aufsteigend oder absteigend sortieren.) Verwenden Sie als Hilfsdatenstruktur ausschließlich eine Schlange `q`. Sie erhalten volle Punktzahl, wenn Sie außer `s` und `q` keine weiteren Variablen benutzen. Sie dürfen annehmen, dass alle Zahlen in `s` verschieden sind.

Lösungsvorschlag

```
q := neue Schlange
while s not empty:
    q.enqueue(S.pop())
while q not empty:
    while s not empty and s.top() < q.head():
        q.enqueue(s.pop())
    s.push(q.dequeue)
```

Als Java-Code

```java
  /**
   * So ähnlich wie der <a href=
   * "https://www.geeksforgeeks.org/sort-stack-using-temporary-stack/">Stapel-
↪   Sortiert-Algorithmus
   * der nur Stapel verwendet</a>, nur mit einer Warteschlange.
   *
   * @param s Der Stapel, der sortiert wird.
   */
  public static void sort(Stapel s) {
    Schlange q = new Schlange();
    while (!s.isEmpty()) {
      q.enqueue(s.pop());
    }
    while (!q.isEmpty()) {
      // Sortiert aufsteigend. Für absteigend einfach das „kleiner"
      // Zeichen umdrehen.
      while (!s.isEmpty() && s.top() < q.head()) {
        q.enqueue(s.pop());
      }
      s.push(q.dequeue());
    }
  }
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_66115/jahr_2015/fruehjahr/schlange/Sort.java`

Klasse Sort

```java
public class Sort {

  /**
   * So ähnlich wie der <a href=
   * "https://www.geeksforgeeks.org/sort-stack-using-temporary-stack/">Stapel-
↪   Sortiert-Algorithmus
   * der nur Stapel verwendet</a>, nur mit einer Warteschlange.
   *
```

```
   * @param s Der Stapel, der sortiert wird.
   */
  public static void sort(Stapel s) {
    Schlange q = new Schlange();
    while (!s.isEmpty()) {
      q.enqueue(s.pop());
    }
    while (!q.isEmpty()) {
      // Sortiert aufsteigend. Für absteigend einfach das „kleiner"
      // Zeichen umdrehen.
      while (!s.isEmpty() && s.top() < q.head()) {
        q.enqueue(s.pop());
      }
      s.push(q.dequeue());
    }
  }

  public static Stapel stapelBefüllen(int[] zahlen) {
    Stapel s = new Stapel();
    for (int i : zahlen) {
      s.push(i);
    }
    return s;
  }

  public static void zeigeStapel(Stapel s) {
    while (!s.isEmpty()) {
      System.out.print(s.pop() + " ");
    }
    System.out.println();
  }

  public static void main(String[] args) {
    Stapel s1 = stapelBefüllen(new int[] { 4, 2, 1, 5, 3 });
    sort(s1);
    zeigeStapel(s1);

    Stapel s2 = stapelBefüllen(new int[] { 1, 2, 6, 3, 9, 11, 4 });
    sort(s2);
    zeigeStapel(s2);
  }
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_66115/jahr_2015/fruehjahr/schlange/Sort.java`

Klasse Schlange

```
public class Schlange {

  public Element head;

  public Schlange() {
```

```
    head = null;
  }

  public int head() {
    if (head.getNext() == null) {
      return head.getValue();
    }
    Element element = head;
    Element previous = head;
    while (element.getNext() != null) {
      previous = element;
      element = element.getNext();
    }
    element = previous.getNext();
    return element.getValue();
  }

  /**
   * @param value Eine Zahl, die zur Schlange hinzugefügt werden soll.
   */
  public void enqueue(int value) {
    Element element = new Element(value);
    element.setNext(head);
    head = element;
  }

  /**
   * @return Das Element oder null, wenn der Schlange leer ist.
   */
  public int dequeue() {
    if (head.getNext() == null) {
      int result = head.getValue();
      head = null;
      return result;
    }
    Element element = head;
    Element previous = null;
    while (element.getNext() != null) {
      previous = element;
      element = element.getNext();
    }
    element = previous.getNext();
    previous.setNext(null);
    return element.getValue();
  }

  /**
   * @return Wahr wenn der Schlange leer ist.
   */
  public boolean isEmpty() {
    return head == null;
  }

  public static void main(String[] args) {
```

```
    Schlange s = new Schlange();
    s.enqueue(1);
    s.enqueue(2);
    s.enqueue(3);
    System.out.println(s.head());
    System.out.println(s.dequeue());
    System.out.println(s.head());
    System.out.println(s.dequeue());
    System.out.println(s.head());
    System.out.println(s.dequeue());
  }

}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_66115/jahr_2015/fruehjahr/schlange/Schlange.java`

Klasse Element

```
public class Element {
  public int value;

  public Element next;

  public Element() {
    this.next = null;
  }

  public Element(int value, Element element) {
    this.value = value;
    this.next = element;
  }

  public Element(int value) {
    this.value = value;
    this.next = null;
  }

  public int getValue() {
    return value;
  }

  public Element getNext() {
    return next;
  }

  public void setNext(Element element) {
    next = element;
  }

}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_66115/jahr_2015/fruehjahr/schlange/Element.java`

Test-Klasse

```java
import static org.junit.Assert.*;

import org.junit.Test;

public class TestCase {

  @Test
  public void testeStapel() {
    Stapel s = new Stapel();
    s.push(1);
    s.push(2);
    s.push(3);

    assertEquals(false, s.isEmpty());

    assertEquals(3, s.top());
    assertEquals(3, s.pop());

    assertEquals(2, s.top());
    assertEquals(2, s.pop());

    assertEquals(1, s.top());
    assertEquals(1, s.pop());
    assertEquals(true, s.isEmpty());
  }

  @Test
  public void testeSchlange() {
    Schlange s = new Schlange();
    s.enqueue(1);
    s.enqueue(2);
    s.enqueue(3);

    assertEquals(false, s.isEmpty());

    assertEquals(1, s.head());
    assertEquals(1, s.dequeue());

    assertEquals(2, s.head());
    assertEquals(2, s.dequeue());

    assertEquals(3, s.head());
    assertEquals(3, s.dequeue());
    assertEquals(true, s.isEmpty());
  }

}
```

Code-Beispiel auf Github ansehen: `src/test/java/org/bschlangaul/examen/examen_66115/jahr_2015/fruehjahr/schlange/TestCase.java`

(b) Analysieren Sie die Laufzeit Ihrer Methode in Abhängigkeit von n.

Zeitkomplexität: $\mathcal{O}(n^2)$, da es zwei ineinander verschachtelte `while` -Schleifen gibt, die von der Anzahl der Elemente im Stapel abhängen.

Aufgabe 7

Auf folgendem ungerichteten, gewichteten Graphen wurde der Dijkstra-Algorithmus (wie auf der nächsten Seite beschrieben) ausgeführt, doch wir wissen lediglich, welcher Knoten als letztes schwarz (black) wurde (Nr. 8) und was seine Distanz zum Startknoten (Nr. 1) ist. Die Gewichte der Kanten sind angegeben.

Finden Sie zunächst den Startknoten, nummerieren Sie anschließend die Knoten in der Reihenfolge, in der sie schwarz wurden, und geben Sie in jedem Knoten die Distanz zum Startknoten an.

Hinweis: Der Startknoten ist eindeutig.

Dijkstra(WeightedGraph G, Vertex s)

```
Initialize(G, s);
S=ß;
Q = new PriorityQueue(V, d) ;
while not Q.Empty() do
  u = Q.ExtractMin() ;
  S = S U {u};
  foreach v € Adj[u| do
    Relax(u, v; w);

  u.color = black;
```

Initialize(Graph G, Vertex s)

```
foreach u € V do
  u.color = white;
  u.d = 00;
s.color = gray;
s.d = 0;
```

Relax(u, v; w)

```
if v.d> u.d+ w(u,v) then
  v.color = gray;
  v.d = u.d + w(u,v);
  Q.DecreaseKey(v, v.d);
```