Praktikum III – CFLs, KAs, Graphisomorphie

GTI SoSe 2016 Prof. A. Siebert

Aufgabe 1. Kellerautomat.
Entwerfen Sie einen deterministischen Kellerautomaten, der die Sprache $\Lambda = \{w \mid w = x^n p^m s^{n-3m}, m \geq 0, n > 3m\}$ akzeptiert.

Testen Sie Ihre Lösung mit JFLAP.

Aufgabe 2. CFG, CNF, CYK.
Gegeben sei die kontextfreie Grammatik G = (V, Σ, P, S) mit V = {S, A, B, C, D}, Σ = {(,)} und P = { S → SS | A | D; A → (S) | B; B → (); C → ()S }.

a. Wandeln Sie G in CNF um.

b. Wandeln Sie die CNF-Grammatik aus (a) mit dem im Skript (Teil 02, Folien 19-21) beschriebenen Verfahren in einen Kellerautomaten um.

c. Lösen Sie für die CNF-Grammatik aus (a) und das Wort w = `(())()` das Wortproblem mit Hilfe des CYK.
Stellen Sie hierzu insbesondere die Variablentabelle V[i, j] auf.

d. Zeichnen Sie für die CNF-Grammatik aus (a) den Ableitungsbaum für w = `(())()`.

Aufgabe 3. Java: Graphisomorphie.
Implementieren Sie in Java ein Programm, das ausgibt, ob zwei gegebene minimale DEAs A, B isomorph sind.
Bei beiden DEAs sind die Zustände von 0 bis n-1 durchnummeriert.

Sind die DEAs isomorph, so soll die 1:1 Abbildung der Zustände von A auf B ausgegeben werden.

Ansonsten soll ausgegeben werden, an welcher Stelle Ihr Programm die DEAs als nicht isomorph erkennt.

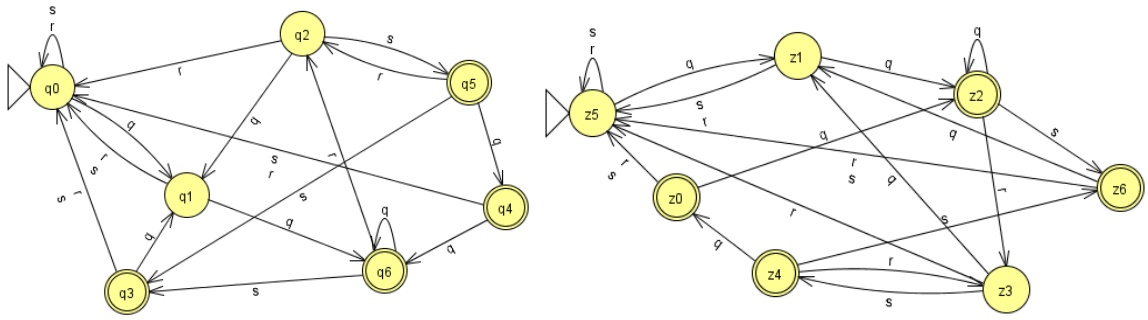
Führen Sie zunächst die offensichtlichen Checks durch:

- Die Anzahl der Zustände muss gleich sein.
- Die Anzahl der Endzustände muss gleich sein.
- Die Alphabete müssen gleich sein.

Danach können Sie die Isomorphie mit der folgenden Strategie überprüfen:
Bauen Sie die 1:1 Abbildung zwischen den Zuständen von A und B auf. Beginnen Sie mit den Startzuständen. Führen Sie den Erreichbarkeits-Algorithmus vom vorhergehenden Praktikum auf A durch und ziehen Sie exakt denselben Algorithmus parallel in B nach. Protokollieren Sie hierbei die jeweils neu erreichten Zustände.
Sind A und B isomorph, so erreichen Sie in A und B in den gleichen Schritten die jeweils korrespondierenden Zustände, welche dann beide Endzustände oder beide Nichtendezustände sein müssen.

Damit der Erreichbarkeits-Algorithmus auf A und B gleich abläuft, dürfen Sie für die Listen RL und WL nicht *die Klasse* `TreeSet` *verwenden, weil diese im Hintergrund die Listen (um-) sortiert. Es bietet sich an, auf die Klasse* `ArrayDeque` *zurück zu greifen.*

Testen Sie Ihr Programm mit den folgenden beiden DEAs. Diese sind bereits in der gegebenen Datei `DEA_iso.java` definiert.



Diese beiden DEAs sind isomorph, und die 1:1 Abbildung der Zustände ist wie folgt:

```
0 =>  5
1 =>  1
2 =>  3
3 =>  6
4 =>  0
5 =>  4
6 =>  2
```

Modifizieren Sie einen der DEAs und stellen Sie sicher, dass Ihr Programm die DEAs dann als nicht mehr isomorph einstuft.