Praktikum IV – TMs

GTI SoSe 2020 Prof. A. Siebert, A. Wallis

Aufgabe 1. PCPs

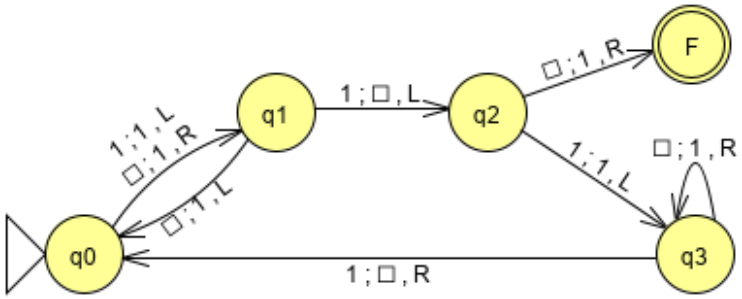
Untersuchen Sie bei den folgenden PCP-Instanzen, ob eine Lösung existiert:

(i) PCP-Instanz = `(#*, #*#), (*##, ##), (#*#, *##)`

(ii) PCP-Instanz = `(aba, a), (bbb, aaa), (aab, abab), (bb, babba)`

Aufgabe 2. TMs: Fleißige Biber

a. Handelt es sich bei der folgenden Turingmaschine um einen Fleißigen Biber (BB-4)?



b. Welches sind die ersten vier Konfigurationen, die von dieser TM durchlaufen werden?

c. Geben Sie eine Gödelnummer für diese TM an (oder überzeugen Sie sich zumindest davon, dass dies eine mühsame, aber prinzipiell einfache Angelegenheit ist).

Aufgabe 3. TMs: Berechnungen (Klausuraufgabe Feb. 2020)

Entwerfen Sie eine deterministische Turingmaschine über dem Eingabealphabet $\Sigma = \{1\}$, welche das gegebene Eingabewort, interpretiert als unäre Zahl, in eine äquivalente Binärzahl umwandelt.
Beispiele: $1 \to 1$, $11 \to 10$, $111 \to 11$, $1111 \to 100$, $11111 \to 101$.
Am Ende der Berechnung muss der Schreib-/Lesekopf am Wortanfang stehen.
Sie dürfen davon ausgehen, dass zu Beginn mindestens eine 1 auf dem Band steht.

Hinweis: Im Skript 02 befindet sich auf Folie 103 eine TM, welche Binärzahlen in äquivalente Unärzahlen umwandelt.

Die Musterlösung zur Klausur Feb. 2020 ist bereits veröffentlicht. Versuchen Sie dennoch zunächst, die TM eigenständig zu entwerfen und zu testen.

Da eine TM alles Berechenbare berechnen kann, gibt es auch genauso viele TMs wie z.B. Java-Programme (abzählbar unendlich viele, wenn Sie es genau wissen wollen). Wenn 100 Programmierer dasselbe Problem unabhängig voneinander lösen, dann resultiert das in 100 nicht-identischen Programmen. Und genauso wird man dann auch 100 nicht-identische TMs erhalten. Ihre eigene TM wird also von meiner TM in der Musterlösung allerhöchstwahrscheinlich abweichen – nur ein wenig, wenn Sie dieselbe Lösungsidee haben, oder sehr stark, wenn Sie einen anderen Lösungsansatz verfolgen.

Und welche TM ist dann die beste?

Man könnte die TM mit den wenigsten Zuständen favorisieren. Sinnvoller ist es aber, diejenige TM zum Sieger zu krönen, die ihr Ziel mit den wenigsten Schritten erreicht. Aber Sie dürfen auch die TMs bevorzugen, die am leichtesten zu entwerfen oder zu verstehen sind. Auch hier ist es wie in der "normalen" Programmierung: Sie können unterschiedliche Kriterien zur Beurteilung der Qualität anlegen.

Aufgabe 4. TMs: Sprachen akzeptieren

Entwerfen Sie eine Turingmaschine über dem Eingabealphabet $\Sigma = \{s, t, x\}$, welche alle Worte akzeptiert, deren Anzahl an x gleich der Differenz aus der Anzahl der s und der Anzahl der t ist.
Anders gesagt: Ist die Anzahl der s gleich n und die Anzahl der t gleich p, so muss die Anzahl der x gleich n-p sein, mit $n \geq p \geq 0$.
Zu Λ gehören z.B. die Worte ε, st, ts, sstt, sstx, xsts, txss, xsxs, txxsss, stxsst, aber nicht tx, xxst, ssttttx.

In JFLAP haben Sie die Option, mit `Input` → `Multiple Run` zahlreiche Eingaben gleichzeitig zu testen (und diese Testeingaben bleiben auch nach Änderungen an der TM erhalten). Ruhen Sie nicht, ehe nicht alle oben aufgeführten Beispielworte korrekt verarbeitet werden.

Aufgabe 5. CSLs/CSGs

Die Sprache $\Lambda = \{ww \mid w \in (a+b)^+\}$ ist kontextsensitiv.
[Zur Erinnerung: $r^+ = rr^*$, d.h. r muss mindestens einmal vorkommen.]
Zu Λ gehören also z.B. aa, bb, abab, baba, aabaab, baabaa, abbbabbb, aber nicht ε, a, b, ab, aaa, aaab, baab.

Schnelltest: Sie glauben nicht, dass Λ kontextsensitiv ist? Falls Λ kontextfrei wäre, müsste es Ihnen gelingen, für Λ eine kontextfreie Grammatik oder einen Kellerautomaten zu basteln...

Eine kontextsensitive Grammatik für Λ ist gegeben durch `G = (V, Σ, P, S)` mit
`V = {S, T, A, B, E, F, Y, Z}`, `Σ = {a, b}` und
`P = { S→TZE, S→TYF, S→ZE, S→YF, T→TZA, T→TYB, T→ZA, T→YB, AZ→ZA, AY→YA, BZ→ZB, BY→YB, AE→Ea, AF→Eb, BE→Fa, BF→Fb, Z→a, Y→b, E→a, F→b}`.

Lösen Sie das Wortproblem: Prüfen Sie mit der im Skript 02, Folien 118-123, beschriebenen (Holzhammer-) Methode, ob die Worte
w_1=`aaab`, w_2=`baba`
von G erzeugt werden. Schreiben Sie hierzu für jede Generation i die Menge T_i der erzeugten Sentenzen auf.

Wenn Sie sich nicht ausgelastet fühlen, dann können Sie jetzt eine TM entwerfen, die Λ akzeptiert.