17125

Grundlagen der theoretischen Informatik
Sommersemester 2022

Henri Gotzeck

10. Übungsblatt

1 Merge Request

a)

$$\begin{aligned} merge(xyz, yu) = \{&yyxzu, yzxuy, yzyxu, yuyzx, xuzyy, yxzuy,\\ &xzuyy, yuyxz, yzuxy, uyxzy, xyzyu, zyuxy, zyxyu, uyzxy,\\ &zuyyx, yxyuz, yyxuz, yyuzx, yxuzy, uxyyz, yyuxz, uzxyy,\\ &uyyxz, uyzyx, yuzyx, xzyuy, zyxuy, uyyzx, yzyux, zyyux,\\ &uxzyy, yxzyu, uzyyx, xyuzy, zyuyx, xzyyu, uyxyz, yuxyz,\\ &yyzux, uzyxy, zyyxu, zxyyu, xyyzu, yxyzu, yyzxu, xuyyz,\\ &uxyzy, zxuyy, yuxzy, yzuyx, xuyzy, zuyxy, zxyuy, yxuyz,\\ &yuzxy, xyuyz, xyzuy, yzxyu, zuxyy, xyyuz\} \end{aligned}$$

Das sind leider viel zu viele ... z.B. kann das Wort das aus xyz und yu entsteht nie mit u beginnen.

f −2

b) Beide reguläre Sprachen L_1 und L_2 haben eine NFA der diese Sprache akzeptiert:
$M_1 = (Z_1, \Sigma_1, \delta_1, S_1, E_1)$,$M_2 = (Z_2, \Sigma_2, \delta_2, S_2, E_2)$.
Hier raus kann man einen Automaten M bauen für dies Sprache L die durch das mergen von L_1 und L_2 entsteht: ✓
wobei $M = (Z_1 \times Z_2, \Sigma_1 \cup \Sigma_2, \delta, S_1 \times Z_2 \cup Z_1 \times S_2, E_1 \times E_2)$ ✓
Hier ist δ definiert als:
Für alle $z_1 \in Z_1$ gilt:
$\forall z_1' \in Z_1, z_2 \in Z_2, a \in \Sigma_1[\delta_1((z_1), a) \in z_1' \rightarrow \delta((z_1, z_2), a) \in (z_1', z_2)]$

und für alle $z_2 \in Z_2$ gilt:
$\forall z_2' \in Z_1, z_1 \in Z_1, a \in \Sigma_1[\delta_2((z_2), a) \in z_2' \rightarrow \delta((z_1, z_2), a) \in (z_1, z_2')]$

Also man baut einen Kreuzprodukt-Automaten, dessen Übergänge immer die Verarbeitung einer Variable eines der beiden Sprachen / Automaten ist. Und wenn beide Wörter in beliebiger Reihenfolge gelesen wurden, dann ist man bei einem Endzustand. Folglich müssen beide anderen Automaten auch am Ende $(E_1 \times E_2)$ sein.

kein Beweis für $T(M) = merge(L_1, L_2)$ via „$\subseteq$" und „$\supseteq$" −3

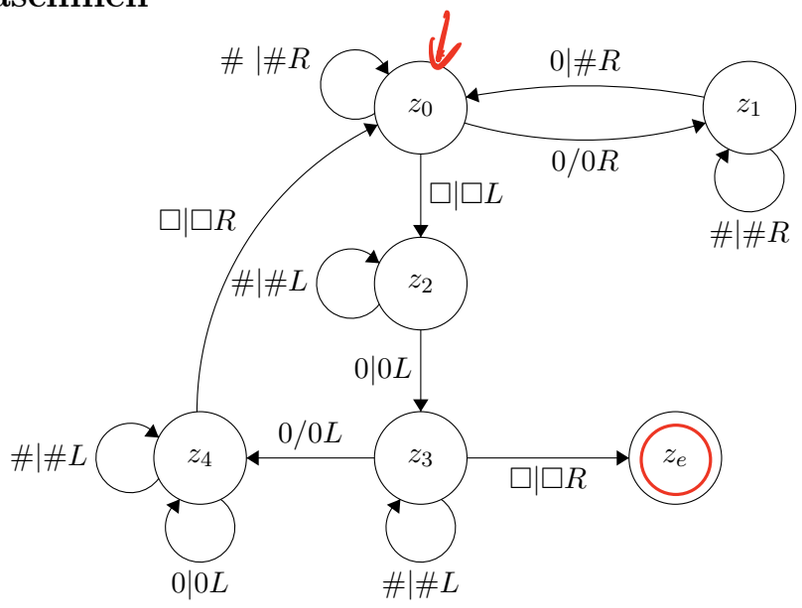
4|4

2 Turingmaschine

1. $L = \{0^n 1 1^j | n, j \in \mathbb{N}\}$ ✓
2. $L = \{01^j | j \in \mathbb{N}\}$ ✓

3,514

3 Turingmaschinen



-0,5

b) Der Automat erkennt die Sprache $L\{0^{2^n} | n \in \mathbb{N} \setminus \{0\}\}$. ✓

1. z_0 und z_1 halbieren die Anzahl an 0len
2. z_2 Wenn man am Ende des Wortes angekommen ist, dann geht man wieder zur nächst gelesenen $0 \rightarrow z_3$
3. z_3 falls es noch eine weitere 0 gelesen wird $\rightarrow z_4$, sonst wenn man zu Anfang kommt akzeptiere das Wort $\rightarrow z_e$
4. z_4 Gehe zu Anfang des Wortes und beginne wieder von vorne $\rightarrow z_0$ ✓

2,4

4 Mehrband-Turingmaschine

Es wird umgesetzt mithilfe des Brute-Force Algorithmus einfach alle Zahlen ausprobieren. Der besitzt 4 Bänder:

1. Eingabe Wort
2. Hierauf wird gezählt wie viele 0 das erste Wort besitzt
3. Rechen zahl wie viel noch übrigbleibt
4. n – Variabel die getestet werden soll

zu kompliziert, die Idee der TM sollte informell beschrieben werden!

Der Algorithmus selber:

1. Schritt: Band 2 als Binär-Zahl das Wort 0 schreiben
2. Schritt: Band 4 als Binär-Zahl das Wort 0 schreiben
3. Schritt: Wenn man eine Eins auf Band 1 (0?) liest dann auf Band 2 eine 1 Binär-Addieren und den Lese-Kopf nach rechts bewegen(= Zählen wie viele Nullen es gibt)
4. Schritt: danach Band 2 auf Band 3 Kopieren
5. Schritt: Band 4 mit 1 addieren
6. Schritt: Falls ob Band 4 größer ist als Band 2 wird das Wort nicht akzeptiert

-2

$\Sigma = \{0\}$?

7. Schritt: Testen ob Band 2, Band 3 und Band 4 die identische Zahl/das Wort besitzen. Falls dies der Fall ist, dann geht der Automat in einen Endzustand (= Es wurde das richtige N gefunden, das $n-$mal in n passt)

8. Schritt: Sonst subtrahieren von Band 3 das Band 4

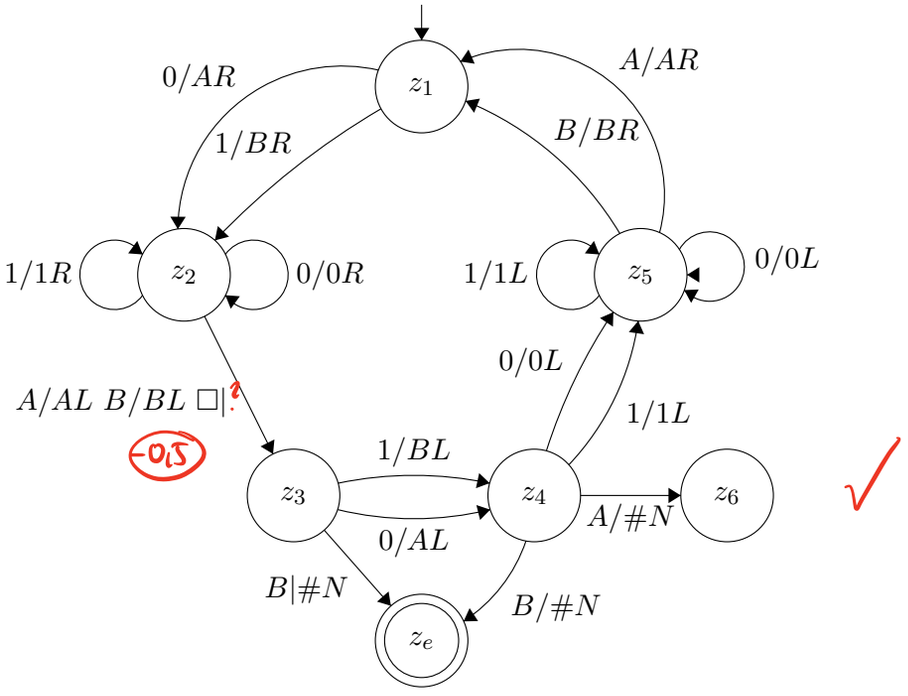
9. Schritt: Falls die Zahl noch nicht negativ ist gehen zu Schritt 7. sonst gehe zu Schritt 4.

Idee in Pseudocode (Schritt = line of code):

```
band[2] = '0'
band[4] = '0'
band[2] += num_of_zeros(band[0])
do{ band[3] = band[2]
    band[4] += 1
    if(band[4] > band[2]) exit('not accetped')
    do{ if(band[2] == band[3] == band[4]) exit('accetped')
        band[3] -= band[4]
    }while(band[3] >= 0)
}while(true)
```

3,514

5 Turingmaschinen



-0,5

Idee Auf der Linken Seite ein Zeichen Lesen und ersetzen mit einem geschriebenen Zeichen (= A für 0, B für 1. Dann an das Ende des gesamten Wortes gehen und hier letzte noch nicht geschriebene Zeichen mit einem geschriebenen Zeichen ersetzen. Dann Schreib-Kopf nach links bewegen, falls hier schon eine B ist, ist man fertig. Sonst wieder zum erste noch nicht geschrieben Zeichen...

313

6 Eine einsteige Frage

b) ist, war da die Turing einfach, wenn sie links mehr Platz benötigen würde einfach das Wort nach rechts verschieben könnte. Und somit auch links im theoretischen Sinne unendlich wäre. Diese Argumentation ist ähnlich wie das $\mathbb{N}$(= Rechts unendliches Band) und $\mathbb{Z}$ (= unendliches Band) gleich abzählbar sind.