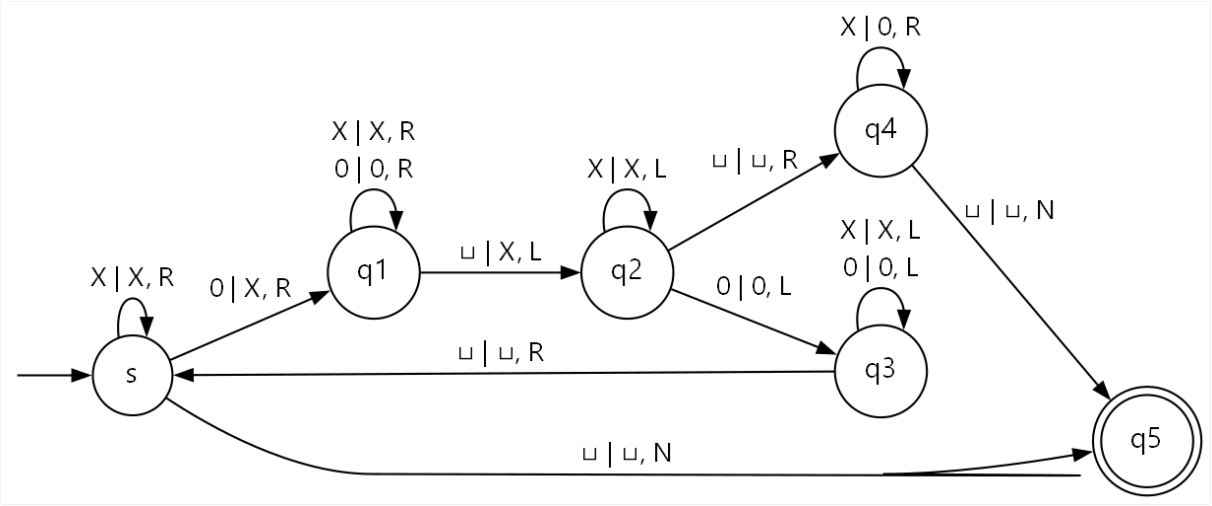
1. (5 Punkte) Gegeben ist die folgende Turingmaschine über dem Alphabet $\Sigma = \{0\}$ mit dem zusätzlichen Bandzeichen X.



(a) Gib die Konfigurationen bei der Eingabe `00` an, bis die TM in den Zustand q_4 kommt.

(b) Welche Wörter akzeptiert diese TM?

(c) Was berechnet diese TM?

Lösung:

(a) Das Zeichen unter dem Kopf ist unterstrichen.

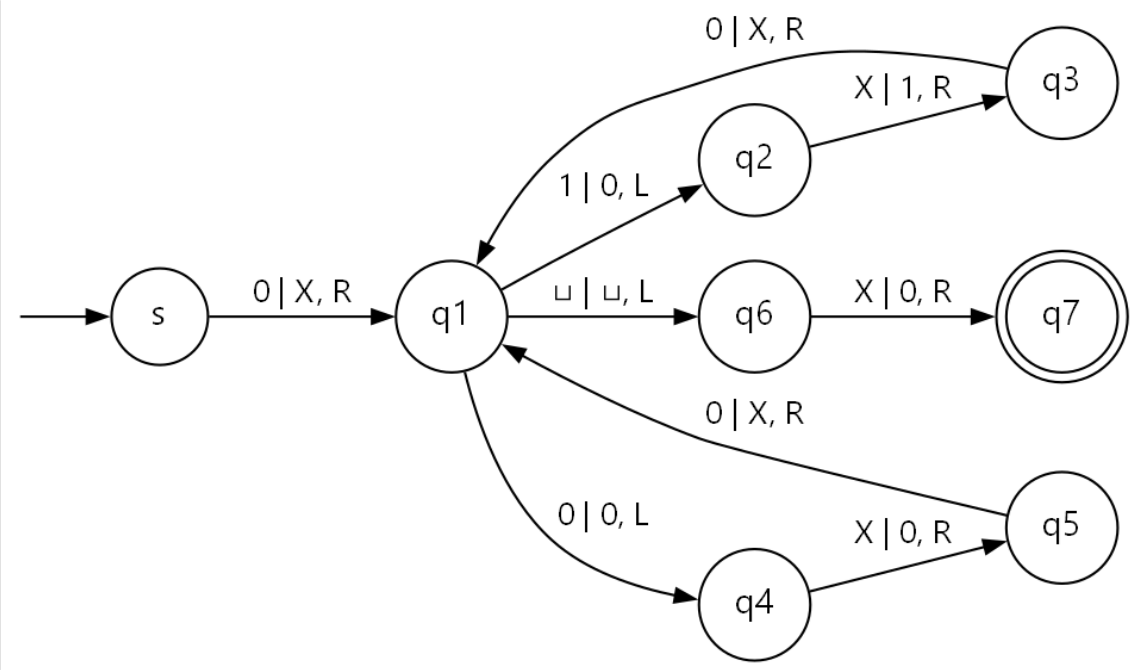
Zustand	Band
s	$\underline{0}0$
q_1	X$\underline{0}$
q_1	X0$\underline{␣}$
q_2	X$\underline{0}$X
q_3	$\underline{X}$0X
q_3	$\underline{␣}$X0X
s	$\underline{X}$0X
s	X$\underline{0}$X

Zustand	Band
q_1	XX$\underline{X}$
q_1	XX$\underline{X}␣$
q_2	X$\underline{X}$XX
q_2	$\underline{X}$XXX
q_2	$\underline{X}$XXX
q_2	$\underline{␣}$XXXX
q_4	$\underline{X}$XXX

(b) Jedes Wort aus Nullen, auch das leere Wort.

(c) Sie verdoppelt die Anzahl der Nullen: Für jede Null, die sie mit X abhakt, hängt sie hinten ein X an. Zum Schluss (q_4) werden alle X zu Nullen.

2. (6 Punkte) Gegeben ist die folgende Turingmaschine über $\Sigma = \{0, 1\}$ mit dem zusätzlichen Bandzeichen X.



(a) Gib die Konfigurationen bei der Eingabe `01101` an.

(b) Was macht diese Turingmaschine?

(c) Gib eine Turingmaschine mit höchstens 3 Zuständen an, die dasselbe macht.

Lösung:

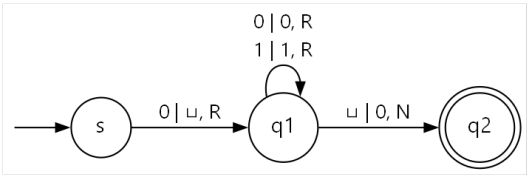
(a)

Zustand	Band
s	$\underline{0}1101$
q_1	X$\underline{1}$101
q_2	$\underline{X}$0101
q_3	1$\underline{0}$101
q_1	1X$\underline{1}$01
q_2	1$\underline{X}$001
q_3	11$\underline{0}$01
q_1	11X$\underline{0}$1

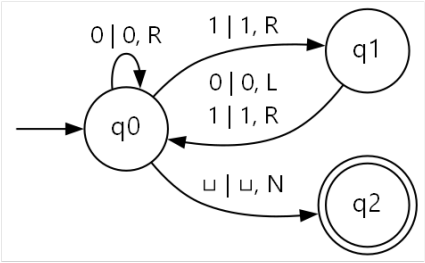
Zustand	Band
q_4	11$\underline{X}$01
q_5	110$\underline{0}$1
q_1	110X$\underline{1}$
q_2	110$\underline{X}$0
q_3	1101$\underline{0}$
q_1	1101X$_{\underline{␣}}$
q_6	1101$\underline{X}$
q_7	11010$_{\underline{␣}}$

(b) Sie verschiebt das Wort um eine Stelle nach links und hängt die erste Ziffer hinten an: aus `01101` wird `11010`. Das klappt nur, wenn das Wort mit 0 beginnt, sonst lehnt sie ab. X markiert die Lücke, in die die nächste Ziffer geschrieben wird; ob eine 0 oder 1 nachrückt, merkt sie sich in den Zuständen q_2, q_3 bzw. q_4, q_5.

(c) Man muss nichts verschieben: Die erste 0 löschen und hinten eine 0 anhängen.



3. (4 Punkte) Gegeben ist die folgende Turingmaschine über $\Sigma = \{0, 1\}$:



   (a) Gib für die Eingaben `00`, `1`, `11`, `10`, `0110`, `0100` an, ob die TM akzeptiert, hält und ablehnt oder nie hält.

   (b) Begründe für die Eingabe `10`, warum die TM nie hält.

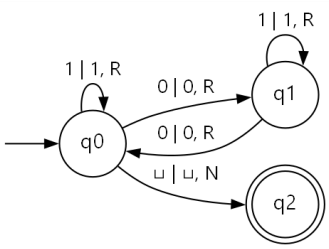
Lösung:

(a) `00`: akzeptiert, `1`: hält und lehnt ab, `11`: akzeptiert, `10`: hält nie, `0110`: akzeptiert, `0100`: hält nie.

(b) In q_0 liest sie die 1 und geht nach rechts in q_1. Dort liest sie die 0 und geht zurück nach links in q_0, wieder auf die 1. Das wiederholt sich ohne Ende, das Band ändert sich dabei nicht.

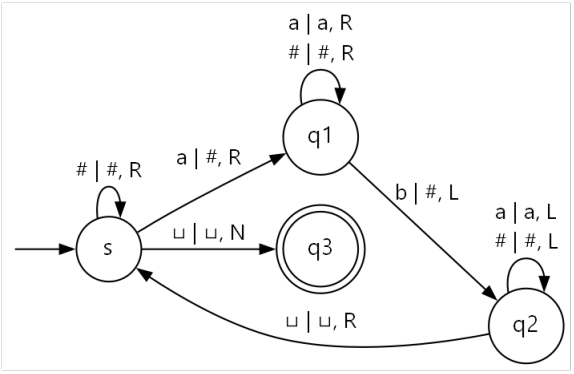
4. (3 Punkte) Gib eine Turingmaschine über dem Alphabet $\Sigma = \{0, 1\}$ an, die alle Wörter mit einer geraden Anzahl von Nullen akzeptiert.

Lösung: q_0: bisher gerade Anzahl Nullen, q_1: ungerade. Die TM liest das Wort einmal von links nach rechts, wie ein DEA.



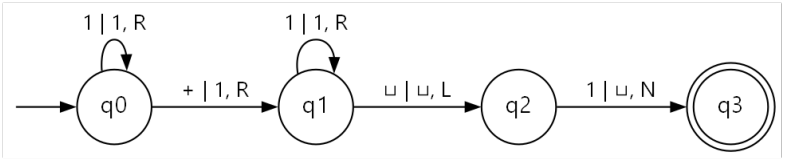
5. (5 Punkte) Die Sprache L über $\Sigma = \{a, b\}$ besteht aus allen Wörtern mit gleich vielen a und b, bei denen in jedem Anfangsstück höchstens so viele b wie a vorkommen. Zum Beispiel liegen *aabb* und *abab* in L, *abba* nicht. Gib eine Turingmaschine an, die L akzeptiert. Du darfst das Bandzeichen # zusätzlich verwenden.

Lösung: Idee: Das erste noch nicht abgehakte Zeichen muss ein a sein. Es wird mit # abgehakt, zusammen mit dem nächsten b dahinter. Dann zurück an den Anfang. Ist alles abgehakt, wird akzeptiert. Findet sich zu einem a kein b oder ist das erste Zeichen ein b, gibt es keinen Übergang, die TM lehnt ab.



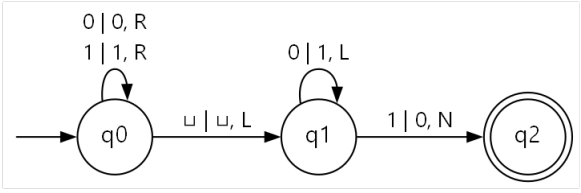
6. (4 Punkte) Zahlen kann man auch unär schreiben: Die Zahl n wird durch n Einsen dargestellt. Gib eine Turingmaschine über $\Sigma = \{1, +\}$ an, die zwei unär geschriebene Zahlen addiert. Aus der Eingabe `11+111` soll das Band `11111` werden.

Lösung: Das + wird durch eine 1 ersetzt, am Ende wird eine 1 gelöscht. Das klappt auch, wenn eine der Zahlen 0 ist.



7. (4 Punkte) Gib eine Turingmaschine an, die von einer Binärzahl $n > 0$ eins abzieht. Aus `1100` soll `1011` werden. Führende Nullen im Ergebnis dürfen stehen bleiben (aus `100` wird `011`).

Lösung: Wie beim schriftlichen Subtrahieren von rechts: Jede 0 wird zu 1 (man muss sich etwas leihen), die erste 1 wird zu 0, dann ist sie fertig. Weil $n > 0$ ist, kommt immer irgendwann eine 1.



8. (4 Punkte) (a) Was besagt die Church-Turing-These? Warum ist sie eine These und kein Satz?
   (b) Was bedeutet es, dass Python Turing-vollständig ist?
   (c) Was ist der Unterschied zwischen einer TM, die eine Sprache entscheidet, und einer, die sie nur erkennt?

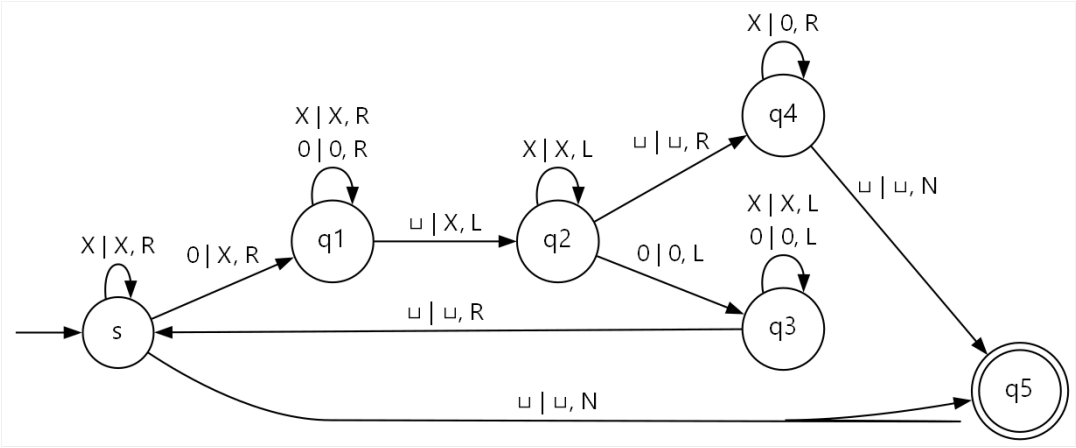
Lösung:

(a) Alle Funktionen, die im intuitiven Sinn berechenbar sind (für die es irgendein Rechenverfahren gibt), kann eine Turingmaschine berechnen. „Im intuitiven Sinn berechenbar“ ist kein mathematisch definierter Begriff, deshalb kann man es nicht beweisen. Gestützt wird die These dadurch, dass alle bekannten Präzisierungen (Turingmaschinen, Registermaschinen, Lambda-Kalkül, Programmiersprachen) dieselben Funktionen ergeben.

(b) Man kann in Python jede Turingmaschine simulieren, also alles berechnen, was eine TM berechnen kann (bis auf den begrenzten Speicher eines echten Computers).

(c) Eine TM, die entscheidet, hält bei jeder Eingabe und akzeptiert genau die Wörter der Sprache. Eine TM, die nur erkennt, akzeptiert genau die Wörter der Sprache, darf bei anderen Wörtern aber auch nie halten. Die Sprache der haltenden Programme kann man erkennen, aber nicht entscheiden (Halteproblem).

9. (4 Punkte) Die folgende Turingmaschine verdoppelt die Anzahl der Nullen auf dem Band:



Ändere die Funktion `lauf` aus dem Notebook so, dass sie die Anzahl der Schritte zurückgibt. Bestimme die Schrittzahl für die Eingaben 0^n mit $n = 0, 1, \ldots, 5$. Wie wächst sie mit n?

Lösung:

```
def schritte(tm, wort, max_schritte=100000):
    band = dict(enumerate(wort))
    zustand, pos = tm['start'], 0
    for schritt in range(max_schritte):
        if zustand in tm['ende']:
            return schritt
        zeichen = band.get(pos, '_')
        if (zustand, zeichen) not in tm['delta']:
            return None
        zustand, neu, richtung = tm['delta'][(zustand, zeichen)]
        band[pos] = neu
        pos = pos + {'L': -1, 'R': 1, 'N': 0}[richtung]
    return None
```

Mit der TM als Dictionary `verdoppeln` ergibt `[schritte(verdoppeln, '0' * n) for n in range(6)]` die Werte 1, 7, 19, 37, 61, 91. Die Differenzen 6, 12, 18, 24, 30 wachsen jedes Mal um 6, die Schrittzahl wächst also quadratisch ($3n^2 + 3n + 1$): Für jede der n Nullen fährt die TM einmal über das ganze Band.