Turing Maschinen

Ein viel mächtigeres Berechnungsmodell als endliche Automaten sind Turing Maschinen. Man kann diese auch als Zustandsmaschinen beschreiben, und wir werden das auch tun. Im Gegensatz zu endlichen Automaten haben Turing Maschinen aber potentiell unendlichen Speicher. Wir geben jetzt eine präzise (wenn auch informelle) Definition von Turing Maschinen.

Eine *Turing Maschine* besteht aus einem unendlichen langen Speicherband das in diskrete Zellen unterteilt ist, auf die mit einem Lese/Schreib-Kopf zugegriffen wird, der in einem Schritt ein Symbol lesen und dann ein Symbol schreiben kann. Die Maschine wird durch Vorgabe einer endlichen Anzahl von Übergangsregeln der Form

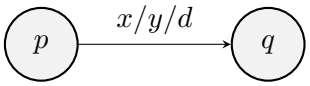
$$R(p, x, y, d, q)$$

programmiert. Eine solche Regel wird interpretiert als: Wenn die Maschine im Zustand p ist und der Lese/Schreib-Kopf sich auf einer Zelle befindet die das Symbol x enthält, dann üeberschreibt der Kopf x mit dem Symbol y, fährt zur in Richtung d benachbarten Zelle, und versetzt die Maschine in den Zustand q.

- Es gibt endlich viele Zustände der Maschine, die vom Programmierer vorgegeben werden. Diese beinhalten einen Anfangszustand "0", und einen Haltezustand "HALT" (bei Entscheidungsproblemen hat man auch zwei Haltezustände "ACCEPT" und "REJECT").
- Die Symbole am Band sind "0" oder "1" (man könnte auch größere Alphabete verwenden), oder das Symbol "#" welches eine leere Zelle symbolisiert.
- Die Bewegungsrichtung d ist entweder "Links" oder "Rechts".

Zu Beginn des Prozesses beinhaltet das Band einen Eingabestring, und der Lese/Schreib-Kopf ist auf der ersten Zelle des Eingabestrings plaziert. Wenn die Maschine einen Haltezustand erreicht, wird der Ausgabestring vom Band abgelesen.

Genauso wie endliche Automaten können Turing Maschinen als Zustandsmaschinen skizziert werden: wir machen einen Kreis für jeden Zustand, bezeichnen den Anfangszustand mit einem (aus dem Nichts) hereinkommenden Pfeil, und zeichnen für jede Regel $R(p, x, y, d, q)$ einen Pfeil von p nach q der mit dem Label $x/y/d$ versehen ist.



Turing Maschinen sind ein theoretisches Berechnungsmodell. Ein realistischeres Modell ist durch die von Neumann Architektur gegeben, die die Basis für moderne Computer ist. Turing Maschinen sind wesentliche Objekte, als einer der ersten Versuche das Konzept von "Berechnung" zu formalisieren. Sie sind einerseits einfach spezifizierbar und können mit mathematischer Präzision behandelt werden, andererseits sind sie in einer recht intuitiven Weise zu programmieren. Man bemerke an dieser Stelle, dass es viele Varianten der oben gegebenen Definition existieren, die aber alle im wesentlichen äquivalent sind. Die Church-Turing These behauptet dass Turing Maschinen berechnungsvollständig sind: Alles was berechnet werden kann, kann schon mit einer Turing Maschine berechnet werden. Daher werden Berechnungsmodelle die Turing Machinen simulieren knnen berechnungsvollständig genannt. Normale Algorithmen sind auch berechnungsvollständig.

Um Turing Maschinen zu spezifizieren und zu simulieren, werden wir den folgenden online Simulator benützen: `http:/turingmachine.vassar.edu/`.

Aufgabe 1 Bestimme eine Turing Maschine die einen binären String in dem Sinne invertiert dass jede "0" durch eine "1" ersetzt wird, und jede "1" durch eine "0". Zum Beispiel soll also der Eingabestring "11100" zum Ausgabestring "00011" führen.

Aufgabe 2 Modifiziere, falls nötig, Deine Maschine aus Aufgabe 1 sodass der Lese/schreib-Kopf vor dem anhalten der Maschine zum Anfang des Ausgabestrings fährt.

Aufgabe 3 Bestimme eine Turing Maschine die einen binären String in umgekehrter Reihenfolge ausgibt. Zum Beispiel wäre die Umkehrung von "11110" der String "01111".
Hinweis: Eine Möglichkeit diese Aufgabe zu lösen ist den folgenden Prozess zu iterieren: Lies das letzte Symbol des Eingabestrings, merke es Dir, entferne es aus der Zelle, gehe zum Ende des Ausgabestrings (der anfänglich leer ist), schreibe das gemerkte Symbol, und gehe zurück zum Ende der Eingabestrings. Um dieses Schema zu realisieren, muss man sich das Symbol das in jeder Iteration gelesen wird, als speziellen Zustand kodieren (zum Beispiel "READ0" bzw. "READ1"). Man muss auch sicherstellen, dass man Anfang und Ende des Eingabestrings verlässlich bestimmen kann, unabhängig davon wie lange der String ist. Man erinnere ich hier, dass der Eingabestring von leeren Zellen (sprich, mit dem speziellen Symbol "#" gefüllten) umgeben ist. Es kann auch hilfreich sein, den gerade verbliebenen Rest des

Eingabestrings vom bereits konstruierten Teil des Ausgabestrings mit einer leeren Zelle zu trennen.

Aufgabe 4 Bestimme eine Turing Maschine die zwei, in unärer Notation gegebene (die Zahl n ist durch n viele Nullen dargestellt), natürliche Zahlen addiert. Im Eingabestring sind die beiden gegebenen Zahlen durch eine leere Zelle getrennt.