4 Turingmaschinen

Nachdem wir in den letzten Kapiteln verschiedene Automatenmodelle und Grammatiken eingeführt bzw. gestreift haben, wollen wir in diesem Kapitel ein weiteres, sehr mächtiges Automatenmodell einführen: die Turingmaschine. Die Turingmaschine ähnelt zunächst sehr einem DFA. Wir erlauben der Turingmaschine aber zusätzlich auf dem Eingabeband nicht nur nach rechts, sondern auch nach links zu gehen und wir erlauben ihr nicht nur ein Symbol zu lesen, sondern auch ein Symbol auf das Band zu schreiben. Diese Änderung führen zu einem sehr mächtigen Modell. Tatsächlich gilt die Turingmaschine als abstraktes Modell dessen, was von jeder Maschine oder jedem Menschen berechnet werden kann, d.h. was immer wir oder eine von uns konstruierte Maschine berechnen kann, das kann auch die Turingmaschine berechnen. Wir werden dieser Aussage später als Church-Turing-These wieder begegnen. Diese These ist so formuliert nicht beweisbar, denn wer weiß schon mit Sicherheit, was wir in der Zukunft konstruieren? Sie hat sich aber als erstaunlich robust erwiesen, denn auch alle neueren Rechnermodelle wie bspw. Parallelrechner lassen sich von Turingmaschinen simulieren. Besonders interessant ist auch der Umkehrschluss: was von einer Turingmaschine *nicht* berechnet werden kann, das kann auch kein Mensch oder keine von ihm erbaute Maschine berechnen.

In den folgenden Abschnitten wollen wir die Turingmaschine einführen, Varianten von ihr wie z.B. eine nichtdeterministische Turingmaschine betrachten und dann die wichtigen Sprachfamilien der entscheidbaren und der aufzählbaren Sprachen einführen. Insb. die entscheidbaren Sprachen spielen bei Algorithmen eine wichtige Rolle. Wir werden dann zeigen, dass es auch unentscheidbare Probleme gibt, d.h. Probleme, für die es keinen Algorithmus geben kann, der sie löst. Ein Resultat, das von besonderer Bedeutung für die Informatik ist, da es uns unsere Grenzen mit dem Computer aufzeigt – und wir werden sehen, dass diese Grenze schon überraschend früh kommt.

4.1 Deterministische Turingmaschinen

Als erstes Modell führen wir die deterministische Turingmaschine ein. Mit dieser wird es möglich sein, alle Sprachen, denen wir bisher begegnet sind, zu akzeptieren und alle Modelle, die wir bisher hatten, zu simulieren. Insbesondere können also alle regulären und alle kontextfreien Sprachen von Turingmaschinen akzeptiert werden.

4.1.1 Definition der DTM

Die deterministische Turingmaschine (kurz DTM oder TM, wenn der Determinismus nicht relevant ist) hat zunächst wie der DFA ein Eingabeband, auf dem zu Anfang ein Eingabewort $w \in \Sigma^*$ steht. Die DTM ist zudem wie der DFA in einem Startzustand z_0. Während der DFA nun aber nur einen Lesekopf hat, der ein Symbol vom Eingabeband liest und sich dann ein Feld nach rechts bewegt (während der DFA noch einen Zustandswechsel macht), hat die Turingmaschine einen *Lese-Schreib-Kopf* (kurz LSK). Mit dem LSK kann die DTM ein Symbol wie der DFA lesen. Dann kann sie an diese Stelle aber ein neues Symbol schreiben und den LSK dann wahlweise nach rechts oder auch nach links bewegen oder die Position des LSK beibehalten.

Damit die DTM genügend Platz für Nebenrechnungen und ähnliches hat, ist das Eingabeband zudem in beide Richtungen unendlich lang. Die Idee dahinter ist, dass ein Mensch, der eine Rechnung auf einem Blatt Papier ausführt, sich auch immer wieder neues Papier holen kann. Man beachte außerdem, dass der Keller des Kellerautomaten im Grunde auch nicht beschränkt war. Auch wenn dies also zunächst nach einer recht umfrangreichen Erweiterung aussieht, ist sie im Grunde genommen ganz naheliegend.

Wir definieren nun die DTM formal. Ähnlich wie beim DFA findet man den LSK nicht in der formalen Definition. Seine Bewegung wird allerdings bei der Überführungsfunktion behandelt. Auch das beidseitig unendliche Band wird in der Definition nicht erwähnt. Man hat es später bei der Definition der Konfigurationsübergänge im Kopf, denn bei einem anderen Band, müsste man hier dann anders definieren.

Definition 4.1.1 (Deterministische Turingmaschine)**.** *Eine* ***deterministische Turingmaschine*** *(kurz DTM) ist ein* 6*-Tupel* $A = (Z, \Sigma, \Gamma, \delta, z_0, Z_{end})$ *mit*

1. *Der endlichen Menge von* ***Zuständen*** Z*.*
2. *Dem endlichen Alphabet* Σ *von* ***Eingabesymbolen****.*
3. *Dem endlichen Alphabet* Γ *von* ***Bandsymbolen****, wobei* $\Gamma \supsetneq \Sigma$ *und* $\Gamma \cap Z = \emptyset$ *gilt.*
4. *Der (partiellen)* ***Überführungsfunktion*** $\delta : (Z \times \Gamma) \to (\Gamma \times \{L, R, H\} \times Z)$*.*
5. *Dem* ***Startzustand*** $z_0 \in Z$*.*
6. *Der Menge der* ***Endzustände*** $Z_{end} \subseteq Z$*.*
7. *Dem* ***Symbol für das leere Feld*** $\# \in \Gamma \setminus \Sigma$*.*

Die Turingmaschine ist nach Alan Turing benannt, der sie in den 1930er Jahren einführte. Alan Turing ist eine der bedeutendsten Persönlichkeiten in der Informatik.

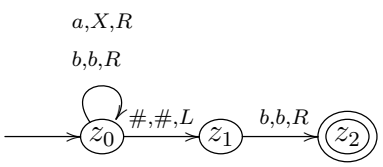
Man beachte, dass in der Definition zwischen Eingabe- und Bandsymbolen unterschieden wird. Ein Eingabewort soll nur aus Eingabesymbolen bestehen. Die Turingmaschine kann dann aber z.B. für Nebenrechnungen weitere Symbole benutzen, daher ist $\Gamma \supset \Sigma$. Zudem ist das Symbol für das leere Feld nicht im Eingabealphabet. Mit diesem will man ja gerade ausdrücken, dass auf einem Feld nichts steht. Zuletzt drückt die Überführungsfunktion oben informal beschriebenes formal aus. Abhängig vom Zustand z, in dem die DTM ist, und Bandsymbol x, das gerade gelesen wird, gibt $\delta(z, x) = (x', B, z')$ den Nachfolgezustand z', das Symbol x' was anstelle von x geschrieben wird und die Bewegung $B \in \{L, R, H\}$ an (links, rechts oder Position halten).

Wir bringen noch einmal auf den Punkt, wie die Turingmaschine arbeitet. Sei $\delta(z, x) = (x', B, z')$, dann

- ist die TM im Zustand z und
- der LSK gerade über einem Feld, das mit dem Symbol x beschriftet ist.
- Nun wird das x durch x' überschrieben und
- dann der LSK nach $B \in \{L, R, H\}$ bewegt, wobei
  - L nach links bewegen bedeutet,
  - R nach rechts bewegen und
  - H den LSK an der Stelle halten bedeutet.
- Zuletzt wird in den Zustand z' gewechselt.

Wir betonen schon einmal, dass $B = H$ nur bedeutet, dass der LSK nicht bewegt wird. Es heißt insbesondere nicht, dass die TM anhalten würde. Dies wird später, wenn es um entscheidbare Sprachen geht, noch wichtig.

Als Beispiel zeigt die Abbildung das Zustandsübergangsdiagramm einer Turingmaschine. Start- und Endzustände werden wie bei DFAs notiert. Die Kantenbeschriftungen sind entsprechend der Überführungsfunktion anders. Die Schleife an z_0 entspricht z.B. $\delta(z_0, a) = (X, R, z_0)$ und $\delta(z_0, b) = (b, R, z_0)$.



Wir können nun die Konfiguration einer Turingmaschine definieren und darauf aufbauend dann die Konfigurationsübergänge. Mit letzterem können wir

dann, basierend auf der Überführungsfunktion, die Dynamik der Turingmaschine beschreiben. Mit der Konfiguration wollen wir erfassen, in welchem Zustand die DTM ist und wie die aktuelle Bandinschrift aussieht. In der Definition der DTM wurde $\Gamma \cap Z = \emptyset$ gefordert, damit der Zustand nun benutzt werden kann, um die Position des LSK anzugeben. Eine Konfiguration wird ein Wort uzv sein, wobei $uv \in \Gamma^*$ die Bandinschrift angibt und z der Zustand ist, in dem die DTM sich gerade befindet. Unter dem LSK ist dann das erste Symbol von v. Zusätzlich wird gefordert, dass u nicht mit # beginnt und v nicht mit # endet. uv ist dann also die relevante Bandinschrift und links (von u) und rechts (von v) stehen nur #.

Definition 4.1.2 (Konfiguration)**.** *Ein Wort $w \in \Gamma^* \cdot Z \cdot \Gamma^*$ heißt **Konfiguration** der DTM $A := (Z, \Sigma, \Gamma, \delta, z_0, Z_{end})$. Ist $w = uzv$ mit $z \in Z$ und $u, v \in \Gamma^*$, dann ist*

1. *A im Zustand z,*
2. *die Bandinschrift uv (links/rechts davon nur #) und*
3. *das erste Symbol von v unter dem LSK. (Ist $v = \lambda$, so ist # unter dem LSK. Ist $v \neq \lambda$, so ist $v \in \Gamma^*(\Gamma \setminus \{\#\})$. Ebenso ist im Fall $u \neq \lambda$ dann $u \in (\Gamma \setminus \{\#\})\Gamma^*$.)*

Die Menge aller Konfigurationen der TM M ist $KONF_M$.

Auch wenn dies nicht nötig ist, wird bei $v = \lambda$ oder $u = \lambda$ manchmal links bzw. rechts von z noch ein # notiert. Diese Konfigurationen wollen wir dann auch erlauben. Allgemein wollen wir links und rechts von u bzw. v beliebig viele # erlauben. In manchen Konfigurationen bzw. Konfigurationsübergängen erleichtert dies die Arbeitsweise der Turingmaschine zu verstehen, wenn man dies notiert.

Mögliche Konfigurationen sind also $zaabb$, $aazbb$ und $aabbz$, aber auch $z\#\#\#aXXb$, $aa\#\#zbbb$, $aa\#\#bbz\#\#cc$ und $aabb\#\#z$. Ferner wollen wir wie gesagt manchmal links oder rechts noch ein # notieren. So beschreiben z.B. die Konfigurationen $aabbz$ und $aabbz\#$ das gleiche. Bei letzterer erkennt man besser, dass unter dem LSK gerade ein # ist.

Will man die Konfigurationsübergänge definieren, so wird dies durch die vielen Fälle, in denen links oder rechts etwas gelöscht wird und nicht mehr in der Konfiguration auftritt und durch die Bewegungen erschwert. Eine formale Definition folgt gleich. Das arbeiten mit Konfigurationsübergängen ist aber im Allgemeinen deutlich einfacher. Ist man in einer Konfiguration $uzxv$ mit $u, v \in \Gamma^*$, $x \in \Gamma$ und $z \in Z$, dann betrachtet man $\delta(z, x)$ und ändert die Konfiguration entsprechend (d.h. ersetzt x, wechselt den Zustand und positioniert den Zustand in der Konfiguration neu). Dabei achtet man dann noch darauf, dass links und rechts nichts unnötiges stehen bleibt und schon erreicht man informal genau das, was formal ausgedrückt recht kompliziert ist.

Definition 4.1.3 (Schrittrelation einer DTM)**.** *Sei A eine DTM. Die* ***Schrittrelation*** $\vdash_A \subseteq KONF_A \times KONF_A$ *ist definiert durch: $w \vdash_A w'$ gilt gdw. 1, 2, 3 oder 4 unten gilt.*

Es ist $u, v, w \in \Gamma^*$, $x, y, z \in \Gamma$, $p, q \in Z$.

1. $w = uypxv$ *und*

$$w' = \begin{cases} uqyzv, \text{ falls } (v \neq \lambda \text{ oder } z \neq \#) \text{ und } \delta(p,x) = (z,L,q) \\ uqy, \text{ falls } v = \lambda, y \neq \# \text{ und } \delta(p,x) = (\#,L,q) \\ uq, \text{ falls } v = \lambda, y = \# \text{ und } \delta(p,x) = (\#,L,q) \\ uyzqv, \text{ falls } \delta(p,x) = (z,R,q) \\ uyqzv \text{ falls } (v \neq \lambda \text{ oder } z \neq \#) \text{ und } \delta(p,x) = (z,H,q) \\ uyq \text{ falls } v = \lambda \text{ und } \delta(p,x) = (\#,H,q) \end{cases}$$

2. $w = uyp$ *und*

$$w' = \begin{cases} uqyz, \text{ falls } z \neq \# \text{ und } \delta(p,\#) = (z,L,q) \\ uqy, \text{ falls } y \neq \# \text{ und } d(p,\#) = (\#,L,q) \\ uq, \text{ falls } y = \# \text{ und } \delta(p,\#) = (\#,L,q) \\ uyzq, \text{ falls } \delta(p,\#) = (z,R,q) \\ uyqz, \text{ falls } z \neq \# \text{ und } \delta(p,\#) = (z,H,q) \\ uyq, \text{ falls } \delta(p,\#) = (\#,H,q) \end{cases}$$

3. $w = pxv$ *und*

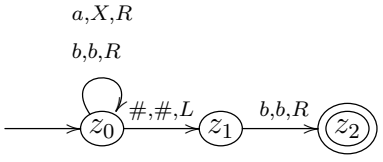
$$w' = \begin{cases} q\#zv, \text{ falls } \delta(p,x) = (z,L,q) \\ zqv, \text{ falls } \delta(p,x) = (z,R,q) \\ qzv, \text{ falls } \delta(p,x) = (z,H,q) \end{cases}$$

4. $w = p$ *und*

$$w' = \begin{cases} q\#z, \text{ falls } \delta(p,\#) = (z,L,q) \\ zq, \text{ falls } \delta(p,\#) = (z,R,q) \\ qz, \text{ falls } z \neq \# \text{ und } \delta(p,\#) = (z,H,q) \\ q, \text{ falls } \delta(p,\#) = (\#,H,q) \end{cases}$$

Die Definition ist wie gesagt recht technisch, um etwas eigentlich ganz einfaches vollständig auszudrücken. Ein kleines Beispiel verdeutlicht wie einfach das Arbeiten mit der Schrittrelation ist.

Betrachten wir die abgebildete DTM. Sei das Eingabewort $w = aab$. Die DTM startet also in der Konfiguration z_0aab.



Als Konfigurationsübergänge haben wir dann durch die Schrittrelation

$$z_0aab \vdash Xz_0ab \vdash XXz_0b \vdash XXbz_0\# \vdash XXz_1b \vdash XXbz_2\#$$

An beiden Stelle, wo dies auftritt, wäre das # ganz rechts nicht nötig (und nach Definition auch nicht vorhanden). Es erleichtert aber insbesondere im ersten Fall zu erkennen, was die DTM als nächstes tun kann.

Wie bei unseren bisherigen Automatenmodellen führen wir wieder den Begriff der Rechnung und der Erfolgsrechnung ein und definieren dann, was die von einer DTM akzeptierte Sprache ist.

Definition 4.1.4 (Rechnung einer DTM)**.** *Sei $A = (Z, \Sigma, \Gamma, \delta, z_0, Z_{end})$ eine DTM.*

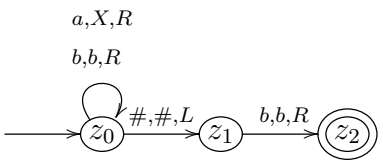
1. *Mit $\vdash_A^*$ wird die reflexive, transitive Hülle von $\vdash_A$ bezeichnet. Der Index A kann auch weggelassen werden, wenn klar ist, welche DTM gemeint ist.*
2. *Eine Folge $k_1 \vdash k_2 \vdash \ldots \vdash k_i \vdash \ldots$ heißt **Rechnung** der DTM A.*
3. *Eine endliche Rechnung $k_1 \vdash k_2 \vdash \ldots \vdash k_n$ heißt **Erfolgsrechnung**, wenn $k_1 \in \{z_0\} \cdot \Sigma^*$ und $k_n \in \Gamma^* \cdot Z_{end} \cdot \Gamma^*$ gilt.*
4. *Die von A akzeptierte Sprache ist*

$$L(A) := \{w \in \Sigma^* \mid \exists u, v \in \Gamma^* \; \exists z_e \in Z_{end} : \; z_0w \vdash^* uz_ev\}.$$

Die akzeptierte Sprache ist anders als wir dies bisher gewohnt sind. Man beachte, dass Worte aus $w \in \Sigma^*$ akzeptiert werden (das waren wir noch gewohnt) und dass dies geschieht, sobald die DTM bei der Rechnung einen Endzustand besucht. Es ist also anders als sonst nicht nötig, dass die Rechnung im Endzustand *endet*! Man könnte zwar alle Kanten aus einem Endzustand entfernen, so dass die Rechnung hier stets endet, aber dies ist nicht gefordert. Außerdem muss die DTM sich nicht das ganze Eingabewort angesehen haben, um zu akzeptieren. Auch dies ist anders als bei z.B. DFAs. Das Eingabewort muss nicht einmal mehr auf dem Band sein. Der Bandinhalt kann im Gegenteil nun ganz anders aussehen. Wichtig ist nur, dass die DTM einen Endzustand erreicht. Warum diese Definition? Die Turingmaschine wurde als Modell des Berechenbaren eingeführt und dient also als Modell für einen Algorithmus. Wenn man z.B. in einem Algorithmus einen kürzesten Pfad in einem Graphen bestimmt, dann muss man sich dazu nicht zwingend den ganzen Graphen ansehen. Auch wenn man wissen will, ob in einem Wort über $\{a, b\}$ mindestens zwei b vorkommen, kann man aufhören, sobald man das zweite gesehen hat. Daher ist es sinnvoll bei der DTM die Definition so zu machen, dass sie nur in einen Endzustand gelangen muss. Das nun ein ganz anderes Wort auf dem Band stehen kann, liegt daran, dass hier u.U. auch viele Zwischenrechnungen und ähnliches notiert sein können.

Wir betonen noch einmal: die DTM startet in der Konfiguration z_0w mit $w \in \Sigma^*$ ganz so wie auch bei einem DFA. Dann macht sie Konfigurationsübergänge entsprechend der Übergangsfunktion. Gelangt sie dabei in eine Konfiguration uz_ev in der z_e ein Endzustand ist, so wird das *ursprüngliche* Eingabewort w akzeptiert (nicht etwas uv). Was aus w wurde ist egal, wie die aktuelle Bandinschrift aussieht ist egal und ob sie noch weiterrechnen könnte ist auch egal. Erreicht sie eine Konfiguration uz_ev, in der z_e ein Endzustand ist, wird das ursprüngliche Eingabewort w akzeptiert.

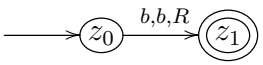
Wir wollen noch zwei Beispiele betrachten. Sei zunächst abermals die folgende DTM gegeben, die wir mit A bezeichnen wollen.



Welche Sprache $L \subseteq \{a, b\}^*$ akzeptiert diese TM? In z_0 wird zunächst über die a und b des Wortes gelesen. Dabei wird der LSK immer weiter nach rechts bewegt. Aus einem a wird ein X, die b werden belassen. Man gelangt so von einer Konfiguration z_0w in eine Konfiguration $w'z_0$, wobei w' aus w hervorgeht, indem jedes a durch X ersetzt wird. Dann wird der Übergang von z_0 nach z_1 gemacht und der LSK nach links bewegt. Er befindet sich nun auf dem letzten Symbol von w'. In z_1 ist nun nur eine Kante. Diese kann genutzt werden, wenn das letzte Symbol von w' gerade ein b ist. Ist dies der Fall, so wird in den einzigen Endzustand z_2 gewechselt und das Eingabewort akzeptiert. w wird also genau dann akzeptiert, wenn w' auf b endet, da w' nur auf b endet, wenn auch w dies tut (die b wurden ja nicht ersetzt), akzeptiert die DTM also genau die Worte, die auf b enden. Damit haben wir

$$L(A) = \{w \in \{a, b\}^* \mid w \text{ endet auf } b\}.$$

Betrachten wir als nächstes die folgende abgebildete sehr kleine DTM B.



Welche Sprache $L \subseteq \{a, b\}^*$ akzeptiert diese TM? Offensichtlich kann sie keine a lesen. Ein b überführt die DTM aber nach z_1. Beginnt ein Eingabewort also mit a, so blockiert B, beginnt ein Eingabewort hingegen mit b, so wird nach z_1 gewechselt und dort akzeptiert. Man beachte noch einmal, dass es bei Turingmaschinen nicht wichtig ist, das Eingabewort zu Ende zu lesen. Hier haben wir also

$$L(B) = \{w \in \{a, b\}^* \mid w \text{ beginnt mit } b\}.$$

Wir fassen das bisherige zusammen. Wir haben die deterministische Turingmaschine (DTM) eingeführt, die recht ähnlich zum DFA ist, aber auf dem

Eingabeband nicht nur lesen, sondern auch schreiben kann und die zudem mit ihrem Lese-Schreib-Kopf (LSK) nach links und rechts gehen kann. Das Eingabeband ist zudem in beide Richtungen unendlich lang.

Um die Arbeitsweise exakt zu definieren, haben wir wieder Konfigurationen und die Schrittrelation eingeführt. Darauf aufbauend haben wir dann die Begriffe Rechnung und Erfolgsrechnung eingeführt und die akzeptierte Sprache definiert. Die DTM akzeptiert ein Wort dann, wenn sie einen Endzustand erreicht. Wie die aktuelle Konfiguration dann genau aussieht und ob sie das Eingabewort zu Ende gelesen hat, ist anders als bei DFAs nicht relevant.

Fragen

1. Muss eine DTM ein Wort zu Ende gelesen haben, um es zu akzeptieren?
   a) Ja!
   b) Nein!
2. Kann eine DTM das leere Wort λ akzeptieren?
   a) Ja!
   b) Nein!
3. Kann eine DTM das Wort # akzeptieren?
   a) Ja!
   b) Nein!
4. Kann eine DTM in eine Endlosschleife geraten, d.h. in eine unendlich lange Rechnung?
   a) Ja!
   b) Nein!