VL-01: Turing Maschinen I

(Berechenbarkeit und Komplexität, WS 2018)

Gerhard Woeginger

WS 2018, RWTH

Organisatorisches

Nächste Vorlesung:

- Donnerstag, Oktober 25, 12:30–14:00, Aula

Danach:

- Freitag, Oktober 26, 16:30–18:00, Audimax
- Freitag, November 2, 16:30–18:00, Audimax
- Donnerstag, November 15, 12:30–14:00, Aula
- Freitag, November 16, 16:30–18:00, Audimax

Webseite:

http://algo.rwth-aachen.de/Lehre/WS1819/BuK.php

Turing Maschinen I

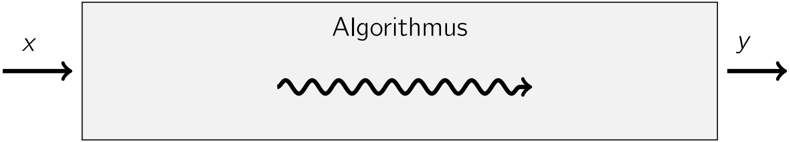
- Probleme und Berechnungen
- Turing Maschinen
- Turing Berechenbarkeit
- Programmierung von Turing Maschinen
- Techniken zur Programmierung von Turing Maschinen

Probleme und Berechnungen

Was ist ein Berechnungsproblem?

Informelle Definitionen

- In einem Berechnungsproblem sollen für bestimmte gegebene Eingaben bestimmte Ausgaben produziert werden.
- Eine Berechnung geschieht mit einem Algorithmus (Handlungsvorschrift).



Wir benötigen viel präzisere Definitionen . . .

Alphabete und Wörter

- Die Ein- und Ausgaben sind Wörter über einem Alphabet Σ.
- Beispiele für Alphabete: $\Sigma = \{0, 1, \#\}$
  $\Sigma = \{a, b, c, \ldots, z\}$
  $\Sigma = \{$あ,い,う,え,お,か,き,く,...,わ$\}$
- Σ^k ist die Menge aller Wörter der Länge k, z.B.

$$\{0,1\}^3 \; = \; \{000, 001, 010, 011, 100, 101, 110, 111\}$$

- Das leere Wort (= Wort der Länge 0), bezeichnen wir mit ϵ.
  Dann gilt: $\Sigma^0 = \{\epsilon\}$
- $\Sigma^* = \bigcup_{k \in \mathbb{N}_0} \Sigma^k$ ist der Kleenesche Abschluss von Σ und enthält alle Wörter über Σ. Diese kann man z.B. der Länge nach aufzählen:

$$\epsilon, 0, 1, 00, 01, 10, 11, 000, 001, 010, 011, 100, 101, \ldots$$

Probleme (1): Als Relation

- Im Allgemeinen entspricht ein Problem einer Relation $R \subseteq \Sigma^* \times \Sigma'^*$
- Ein Paar (x, y) liegt in R,
  wenn y eine zulässige Ausgabe zur Eingabe x ist.

Beispiel: Primfaktorbestimung

Beispiel: Primfaktorbestimmung

Zu einer natürlichen Zahl $q \geq 2$ (Eingabe) suchen wir einen Primfaktor (Ausgabe).

- Wir einigen uns darauf, Zahlen binär zu kodieren.
- Die Binärkodierung der natürlichen Zahl i bezeichnen wir mit $\text{bin}(i)$. Also zum Beispiel: $\text{bin}(0) = 0$, $\text{bin}(6) = 110$

Die entsprechende Relation zur Primfaktorbestimmung ist dann

$$R = \{(x, y) \in \{0,1\}^* \times \{0,1\}^* \mid x = \text{bin}(q),\ y = \text{bin}(p),\\ q, p \in \mathbb{N},\ q \geq 2,\ p \text{ prim},\ p \text{ teilt } q\}$$

Also zum Beispiel $(110, 11) \in R$, aber $(101, 11) \notin R$.

Probleme (2): Als Funktion

- Bei vielen Problemen gibt es zu jeder Eingabe eine eindeutige Ausgabe.
- Dann können wir das Problem durch eine Funktion $f : \Sigma^* \to \Sigma'^*$ beschreiben.
- Die zur Eingabe $x \in \Sigma^*$ gesuchte Ausgabe ist $f(x) \in \Sigma'^*$.

Beispiel: Multiplikation

Beispiel: Multiplikation

Zu zwei natürlichen Zahlen $i_1, i_2 \in \mathbb{N}$ (Eingabe) suchen wir das entsprechende Produkt $i_1 \cdot i_2$ (Ausgabe).

Um die Zahlen i_1 und i_2 in der Eingabe voneinander trennen zu können, erweitern wir das Alphabet um ein Trennsymbol: $\Sigma = \{0, 1, \#\}$.

Die entsprechende Funktion $f : \Sigma^* \to \Sigma^*$ ist dann gegeben durch

$$f(\mathrm{bin}(i_1)\#\,\mathrm{bin}(i_2)) \;=\; \mathrm{bin}(i_1 \cdot i_2)$$

Probleme (3): Entscheidungsprobleme als Sprachen

- Viele Probleme lassen sich als Ja-Nein-Fragen formulieren.
- Derartige Entscheidungsprobleme sind von der Form $f : \Sigma^* \to \{0, 1\}$, wobei wir 0 als "Nein" und 1 als "Ja" interpretieren.
- Es sei $L = f^{-1}(1) \subseteq \Sigma^*$ die Menge derjenigen Eingaben, die mit „Ja" beantwortet werden.
- L ist eine Teilmenge der Wörter über dem Alphabet Σ.
- Eine Teilmenge von Σ^* wird allgemein als Sprache bezeichnet.
- Die Sprache L ist die zu dem durch f definierten Entscheidungsproblem gehörende Sprache.

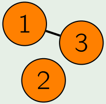
Beispiel: Entscheidungsproblem als Sprache

Beispiel: Graphzusammenhang

Problemstellung: Für einen gegebenen ungerichteten Graphen $G = (V, E)$ soll bestimmt werden, ob G zusammenhängend ist.

Der Graph G liegt dabei in einer geeigneten Kodierung $\text{code}(G) \in \Sigma^*$ vor, zum Beispiel als binär kodierte Adjazenzmatrix.

Beispiel:



$$\rightsquigarrow \begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix} \rightsquigarrow 001000100$$

Die zu diesem Entscheidungsproblem gehörende Sprache ist

$$L = \{\, w \in \Sigma^* \mid \exists \text{ Graph } G\colon w = \text{code}(G) \text{ und } G \text{ ist zusammenhängend} \,\}$$

Turing Maschinen

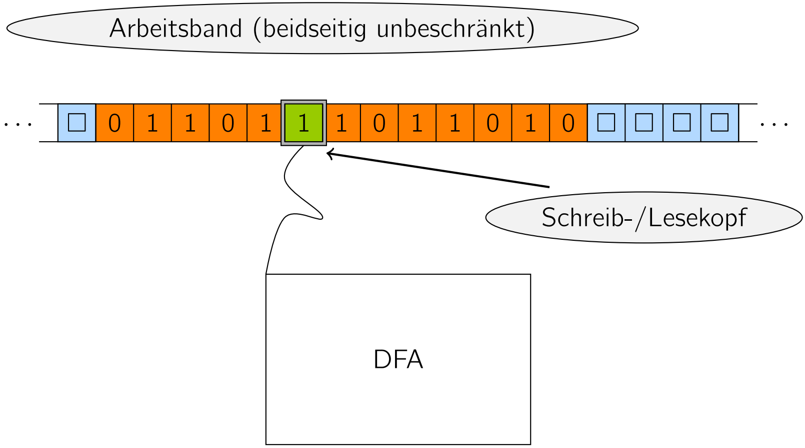
Zentrale Fragestellung

Frage

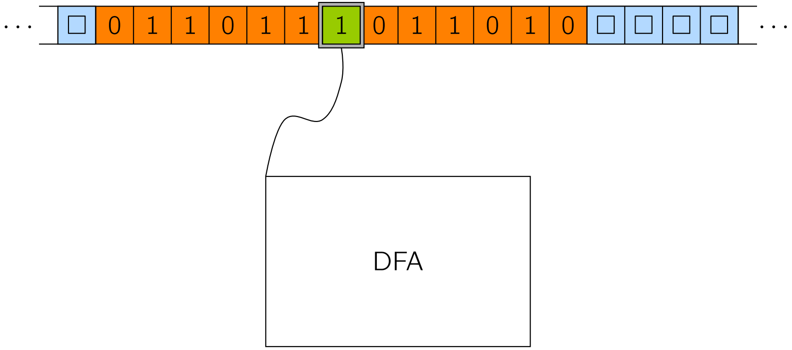
Welche Funktionen sind durch einen Algorithmus berechenbar?
Welche Sprachen können von einem Algorithmus entschieden werden?

- Um diese Fragen in einem mathematisch exakten Sinne klären zu können, müssen wir festlegen, was eigentlich ein Algorithmus ist.
- Zu diesem Zweck definieren wir ein sehr einfaches Computer-Modell: Die Turingmaschine (TM).

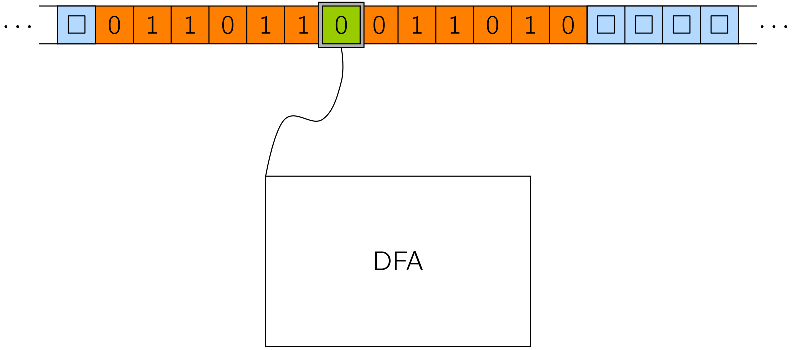
Deterministische Turingmaschine (TM bzw. DTM)



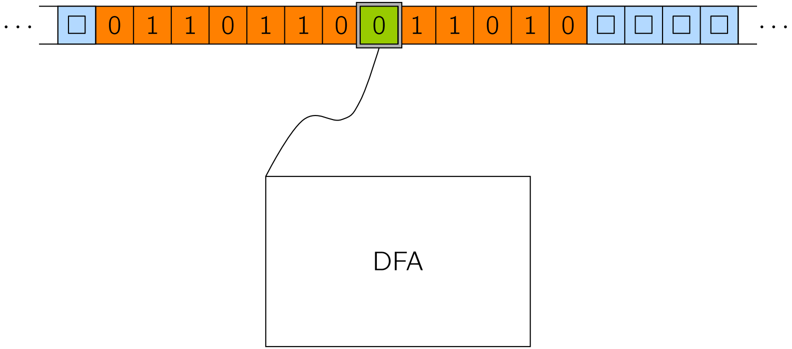
Deterministische Turingmaschine (TM bzw. DTM)



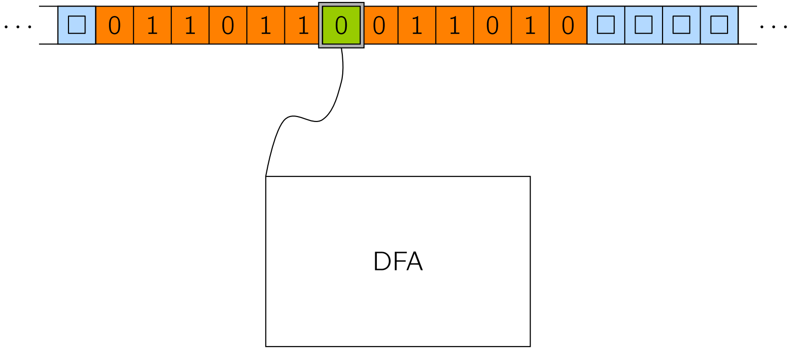
Deterministische Turingmaschine (TM bzw. DTM)



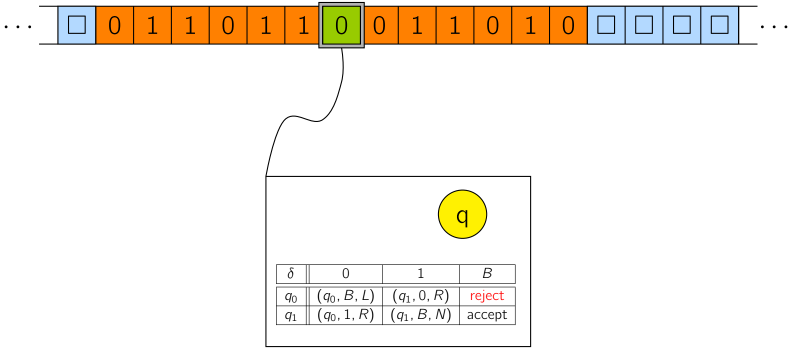
Deterministische Turingmaschine (TM bzw. DTM)



Deterministische Turingmaschine (TM bzw. DTM)



Deterministische Turingmaschine (TM bzw. DTM)



δ	0	1	B
q_0	(q_0, B, L)	$(q_1, 0, R)$	reject
q_1	$(q_0, 1, R)$	(q_1, B, N)	accept

Komponenten der TM

- Q die endliche Zustandsmenge
- Σ das endliche Eingabealphabet
- $\Gamma \supset \Sigma$ das endliche Bandalphabet
- $B \in \Gamma \setminus \Sigma$ das Leerzeichen (Blank, in Bildern $\square$)
- $q_0 \in Q$ der Anfangszustand
- $\bar{q} \in Q$ der Endzustand
- $\delta : (Q \setminus \{\bar{q}\}) \times \Gamma \to Q \times \Gamma \times \{R, L, N\}$
  die Zustandsüberführungsfunktion

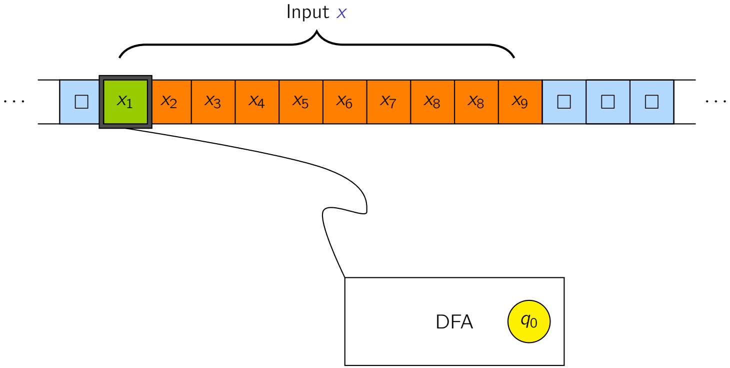
Definition

Eine Turingmaschine ist definiert durch das 7-Tupel $(Q, \Sigma, \Gamma, B, q_0, \bar{q}, \delta)$.

Funktionsweise der TM (1)

Ausgangssituation

- Auf dem Band steht die Eingabe $x \in \Sigma^*$ eingerahmt von Blanks
- Der initiale Zustand ist q_0
- Der Kopf steht über dem ersten Symbol von x



Funktionsweise der TM (2)

Nummerierung der Zellen des Bandes

- Die initiale Kopfposition wird als Position 0 bezeichnet
- Bewegt sich der Kopf einen Schritt "nach rechts", so erhöht sich die Position um 1
- Bewegt sich der Kopf um einen Schritt "nach links", so verringert sich die Position um 1

Funktionsweise der TM (3)

Durchführung eines einzelnen Rechenschrittes

- $a \in \Gamma$ bezeichne das gelesene Symbol (unterm Kopf)
- $q \in Q \setminus \{\bar{q}\}$ bezeichne den aktuellen Zustand
- Angenommen $\delta(q, a) = (q', a', d)$, für $q' \in Q, a' \in \Gamma, d \in \{R, L, N\}$

- Dann wird der Zustand auf q' gesetzt
- an der Kopfposition wird das Symbol a' geschrieben
- der Kopf bewegt sich

$$\begin{cases} \text{um eine Position nach rechts} & \text{falls } d = R \\ \text{um eine Position nach links} & \text{falls } d = L \\ \text{nicht} & \text{falls } d = N \end{cases}$$

Funktionsweise der TM (4)

Ende der Berechnung

- Die TM stoppt, wenn sie den Endzustand $\bar{q}$ erreicht
- Das Ausgabewort $y \in \Sigma^*$ kann dann vom Band abgelesen werden: y beginnt an der Kopfposition und endet unmittelbar vor dem ersten Symbol aus $\Gamma \setminus \Sigma$

Spezialfall

Bei Entscheidungsproblemen wird die Antwort wie folgt als JA oder NEIN interpretiert:

- Die TM akzeptiert das Eingabewort, wenn sie terminiert und das Ausgabewort mit einer 1 beginnt
- Die TM verwirft das Eingabewort, wenn sie terminiert und das Ausgabewort nicht mit einer 1 beginnt

Funktionsweise der TM (5)

Anmerkungen

- Es besteht die Möglichkeit, dass die TM den Endzustand niemals erreicht. Wir sagen dann, dass die Berechnung nicht terminiert.
- Laufzeit = Anzahl von Zustandsübergängen bis zur Terminierung
- Speicherbedarf = Anzahl von Bandzellen, die während der Berechnung besucht werden

Funktionsweise der TM: Beispiel

Beispiel (1)

Es sei $L = \{w1 \mid w \in \{0,1\}^*\}$ die Sprache der Wörter, die auf 1 enden.

L wird entschieden durch die TM $M = (Q, \Sigma, \Gamma, B, q_0, \bar{q}, \delta)$ mit

- $Q = \{q_0, q_1, \bar{q}\}$
- $\Sigma = \{0, 1\}$
- $\Gamma = \{0, 1, B\}$
- δ gemäß Tabelle

δ	0	1	B
q_0	(q_0, B, R)	(q_1, B, R)	reject
q_1	(q_0, B, R)	(q_1, B, R)	accept

„accept“ steht hier als Abkürzung für $(\bar{q}, 1, N)$.
„reject“ steht hier als Abkürzung für $(\bar{q}, 0, N)$.

Allgemein:

- M entscheidet L, wenn M alle Wörter in L akzeptiert und alle Wörter verwirft, die nicht in L sind
- Wenn TM eine Sprache entscheidet, so muss sie immer halten

Beispiel (2)

Die Übergangsfunktion ist zentraler Bestandteil der Turingmaschine.

Beschreibung der Übergangsfunktion als Tabelle:

δ	0	1	B
q_0	(q_0, B, R)	(q_1, B, R)	reject
q_1	(q_0, B, R)	(q_1, B, R)	accept

Verbale Beschreibung des Algorithmus der TM

Solange ein Symbol aus $\{0, 1\}$ gelesen wird:

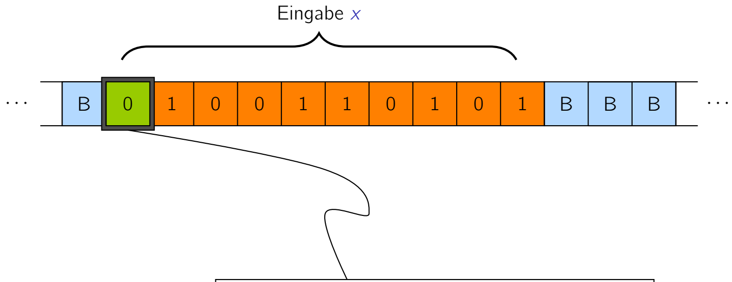
- Überschreibe das Symbol mit B,
- bewege den Kopf nach rechts, und
- gehe in Zustand q_0, wenn Symbol=0, sonst in Zustand q_1

Sobald ein Blank gelesen wird,

- akzeptiere die Eingabe, falls der aktuelle Zustand q_1 ist, und
- andernfalls verwirf die Eingabe.

Veranschaulichung des Algorithmus

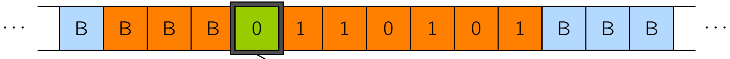
Eingabe $x = 0100110101$



δ	0	1	B
q_0	(q_0, B, R)	(q_1, B, R)	reject
q_1	(q_0, B, R)	(q_1, B, R)	accept

Veranschaulichung des Algorithmus

Eingabe $x = 0100110101$



δ	0	1	B
q_0	(q_0, B, R)	(q_1, B, R)	reject
q_1	(q_0, B, R)	(q_1, B, R)	accept

Turing Berechenbarkeit

Zum Sinn und Zweck des TM-Modells

Die TM dient als unser formales (bzw. mathematisches) Modell zur Beschreibung von Algorithmen.

Die Frage, *ob es für ein Problem einen Algorithmus gibt*,
setzen wir ab jetzt gleich mit
der Frage, *ob es eine TM gibt, die dieses Problem löst*.

- Frage: Ist das TM-Modell sinnvoll? Ist es allgemein genug?
- Frage: Kann das TM-Modell alle denkbaren Algorithmen und alle Fähigkeiten abdecken, die ein moderner Computer hat und die ein zukünftiger Computer haben könnte?
- Auf diese Problematik kommen wir später noch zurück.

Turing Berechenbarkeit

Bzgl. der Berechnungsprobleme beschränken wir uns in dieser Vorlesung auf Funktionen und Entscheidungsprobleme (Sprachen).

Definition

Eine Funktion $f : \Sigma^* \to \Sigma^*$ heisst rekursiv (T-berechenbar),
wenn es eine TM gibt, die aus der Eingabe x
den Funktionswert $f(x)$ berechnet.

Definition

Eine Sprache $L \subseteq \Sigma^*$ heisst rekursiv (T-entscheidbar),
wenn es eine TM gibt, die für alle Eingaben terminiert und
die Eingabe w genau dann akzeptiert, wenn $w \in L$ ist.

Programmierung von Turing Maschinen

Programmierung der TM am Beispiel

Wir entwickeln nun eine TM für die Sprache $L = \{0^n 1^n \mid n \geq 1\}$

Es sei $\Sigma = \{0, 1\}$, $\Gamma = \{0, 1, B\}$, $Q = \{q_0, \ldots, q_6, \bar{q}\}$

Unsere TM arbeitet in zwei Phasen:

Phase 1: Teste, ob Eingabe von der Form $0^i 1^j$ mit $i \geq 0$ und $j \geq 1$

Phase 2: Teste, ob $i = j$ gilt.

Phase 1 verwendet $\{q_0, q_1\}$ und wechselt bei Erfolg zu q_2.
Phase 2 verwendet $\{q_2, \ldots, q_6\}$ und akzeptiert bei Erfolg.

Programmierung der TM am Beispiel: Phase 1

δ	0	1	B
q_0	$(q_0, 0, R)$	$(q_1, 1, R)$	reject
q_1	reject	$(q_1, 1, R)$	(q_2, B, L)

q_0 : Laufe von links nach rechts über die Eingabe, bis ein Zeichen ungleich 0 gefunden wird.

- Falls dieses Zeichen eine 1 ist, gehe über in Zustand q_1.
- Sonst ist dieses Zeichen ein Blank. Verwirf die Eingabe.

q_1 : Gehe weiter nach rechts bis zum ersten Zeichen ungleich 1.

- Falls dieses Zeichen eine 0 ist, verwirf die Eingabe.
- Sonst ist das gefundene Zeichen ein Blank. Bewege den Kopf um eine Position nach links auf die letzte gelesene 1. Wechsle in den Zustand q_2. Phase 2 beginnt.

Programmierung der TM am Beispiel: Phase 2

δ	0	1	B
q_2	reject	(q_3, B, L)	reject
q_3	$(q_3, 0, L)$	$(q_3, 1, L)$	(q_4, B, R)
q_4	(q_5, B, R)	reject	reject
q_5	$(q_6, 0, R)$	$(q_6, 1, R)$	accept
q_6	$(q_6, 0, R)$	$(q_6, 1, R)$	(q_2, B, L)

q_2 : Kopf steht auf dem letzten Nichtblank. Falls dieses Zeichen eine 1 ist, so lösche es, gehe nach links, und wechsele in Zustand q_3. Andernfalls verwirf die Eingabe.

q_3 : Bewege den Kopf auf das erste Nichtblank. Dann q_4.

q_4 : Falls das gelesene Zeichen eine 0 ist, ersetze es durch ein Blank und gehe nach q_5, sonst verwirf die Eingabe.

q_5 : Wir haben jetzt die linkeste 0 und die rechteste 1 gelöscht. Falls Restwort leer, dann akzeptiere, sonst q_6.

q_6 : Laufe wieder zum letzten Nichtblank und starte erneut in q_2.

Veranschaulichung der TM

Eingabe $x = 0000011$

...	B	0	0	0	0	0	1	1	B	B	B	...

δ	0	1	B
q_0	$(q_0, 0, R)$	$(q_1, 1, R)$	reject
q_1	reject	$(q_1, 1, R)$	(q_2, B, L)
q_2	reject	(q_3, B, L)	reject
q_3	$(q_3, 0, L)$	$(q_3, 1, L)$	(q_4, B, R)
q_4	(q_5, B, R)	reject	reject
q_5	$(q_6, 0, R)$	$(q_6, 1, R)$	accept
q_6	$(q_6, 0, R)$	$(q_6, 1, R)$	(q_2, B, L)

Veranschaulichung der TM

Eingabe $x = 0000011$

$\cdots$	B	0	0	0	0	0	1	1	B	B	B	$\cdots$

δ	0	1	B
q_0	$(q_0, 0, R)$	$(q_1, 1, R)$	reject
q_1	reject	$(q_1, 1, R)$	(q_2, B, L)
q_2	reject	(q_3, B, L)	reject
q_3	$(q_3, 0, L)$	$(q_3, 1, L)$	(q_4, B, R)
q_4	(q_5, B, R)	reject	reject
q_5	$(q_6, 0, R)$	$(q_6, 1, R)$	accept
q_6	$(q_6, 0, R)$	$(q_6, 1, R)$	(q_2, B, L)

Konfigurationen: Definitionen

Definition

(i) Eine Konfiguration einer TM ist ein String $\alpha q\beta$, mit $q \in Q$ und $\alpha, \beta \in \Gamma^*$.
Bedeutung: auf dem Band steht $\alpha\beta$ eingerahmt von Blanks, der Zustand ist q, und der Kopf steht über dem ersten Zeichen von β.

(ii) $\alpha' q' \beta'$ ist direkte Nachfolgekonfiguration von $\alpha q\beta$, falls $\alpha' q' \beta'$ in einem Rechenschritt aus $\alpha q\beta$ entsteht. Wir schreiben

$$\alpha q\beta \vdash \alpha' q' \beta'$$

(iii) $\alpha'' q'' \beta''$ ist Nachfolgekonfiguration von $\alpha q\beta$, falls $\alpha'' q'' \beta''$ in endlich vielen Rechenschritten aus $\alpha q\beta$ entsteht. Wir schreiben

$$\alpha q\beta \vdash^* \alpha'' q'' \beta''.$$

Anmerkung: insbesondere gilt $\alpha q\beta \vdash^* \alpha q\beta$

Konfigurationen: Beispiel

Die für die Sprache $L = \{0^n 1^n | n \geq 1\}$ beschriebene TM liefert in Phase 1 auf der Eingabe 0011 die folgende Konfigurationsfolge.

Phase 1:

$$q_0 0011 \ \vdash \ 0 q_0 011 \ \vdash \ 00 q_0 11 \ \vdash \ 001 q_1 1 \ \vdash \ 0011 q_1 B \ \vdash \ 001 q_2 1$$

Anmerkung: Abgesehen von Blanks am Anfang und Ende des Strings sind die Konfigurationen eindeutig.

Phase 2:

$$001 q_2 1 \ \vdash \ 00 q_3 1 \ \vdash \ 0 q_3 01 \ \vdash \ q_3 001 \ \vdash \ q_3 B001 \ \vdash \ q_4 001$$

$$\vdash \ q_5 01 \ \vdash \ 0 q_6 1 \ \vdash \ 01 q_6 \ \vdash \ 0 q_2 1 \ \vdash \ \ldots$$

Techniken zur Programmierung von Turing Maschinen

Techniken zur Programmierung von TMs (1)

Technik 1: Speicher im Zustandsraum

Für beliebiges festes $k \in \mathbb{N}$ können wir k Zeichen unseres Bandalphabets im Zustand abspeichern, indem wir den Zustandsraum um den Faktor $|\Gamma|^k$ vergrössern, d.h. wir setzen

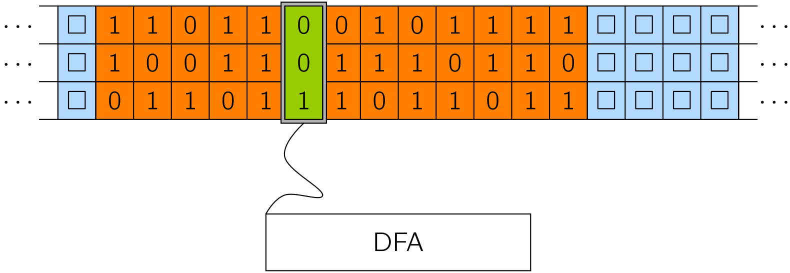
$$Q_{\text{neu}} \;:=\; Q \times \Gamma^k$$

Neue Zustände für $k = 2$ sind dann zum Beispiel (q_0, BB), $(q_1, 10)$.

Techniken zur Programmierung von TMs (2)

Technik 2: Mehrspurmaschinen

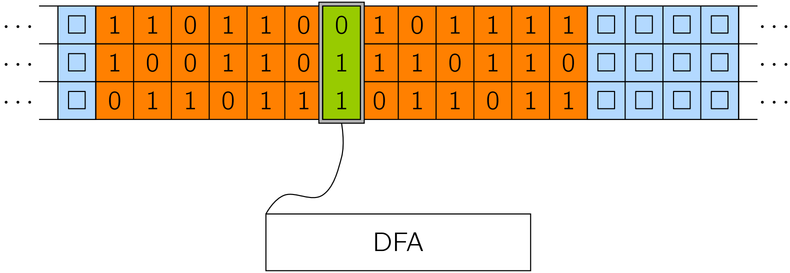
- *k*-spurige TM: eine TM, bei der das Band in k sogenannte Spuren eingeteilt ist. D.h. in jeder Bandzelle stehen k Zeichen, die der Kopf gleichzeitig einlesen kann.
- Das können wir erreichen, indem wir das Bandalphabet um k-dimensionale Vektoren erweitern, z.B. $\Gamma_{\mathrm{neu}} := \Gamma \cup \Gamma^k$



Techniken zur Programmierung von TMs (2)

Technik 2: Mehrspurmaschinen

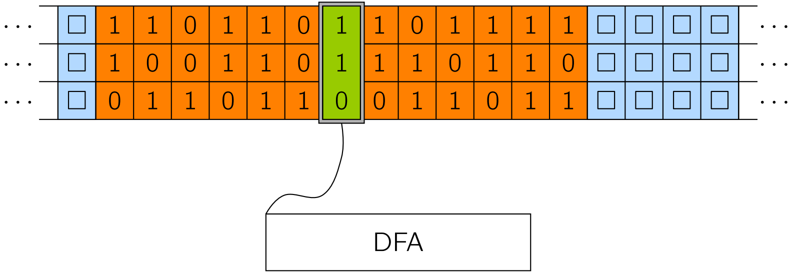
- *k*-spurige TM: eine TM, bei der das Band in k sogenannte Spuren eingeteilt ist. D.h. in jeder Bandzelle stehen k Zeichen, die der Kopf gleichzeitig einlesen kann.
- Das können wir erreichen, indem wir das Bandalphabet um k-dimensionale Vektoren erweitern, z.B. $\Gamma_{\text{neu}} := \Gamma \cup \Gamma^k$



Techniken zur Programmierung von TMs (2)

Technik 2: Mehrspurmaschinen

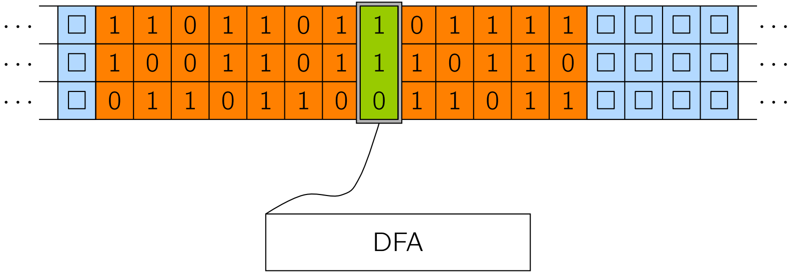
- *k*-spurige TM: eine TM, bei der das Band in k sogenannte Spuren eingeteilt ist. D.h. in jeder Bandzelle stehen k Zeichen, die der Kopf gleichzeitig einlesen kann.
- Das können wir erreichen, indem wir das Bandalphabet um k-dimensionale Vektoren erweitern, z.B. $\Gamma_{\mathrm{neu}} := \Gamma \cup \Gamma^k$



Techniken zur Programmierung von TMs (2)

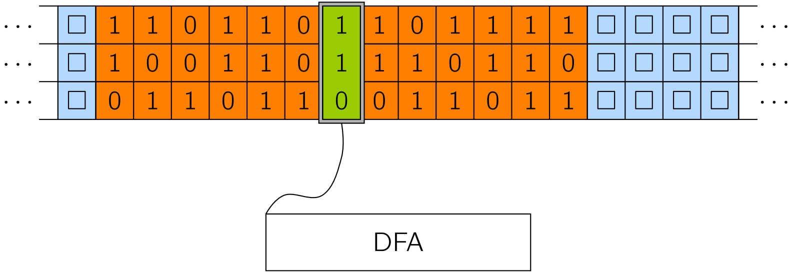
Technik 2: Mehrspurmaschinen

- *k*-spurige TM: eine TM, bei der das Band in k sogenannte Spuren eingeteilt ist. D.h. in jeder Bandzelle stehen k Zeichen, die der Kopf gleichzeitig einlesen kann.
- Das können wir erreichen, indem wir das Bandalphabet um k-dimensionale Vektoren erweitern, z.B. $\Gamma_{\text{neu}} := \Gamma \cup \Gamma^k$



Technik 2: Mehrspurmaschinen

- *k*-spurige TM: eine TM, bei der das Band in k sogenannte Spuren eingeteilt ist. D.h. in jeder Bandzelle stehen k Zeichen, die der Kopf gleichzeitig einlesen kann.
- Das können wir erreichen, indem wir das Bandalphabet um k-dimensionale Vektoren erweitern, z.B. $\Gamma_{\mathrm{neu}} := \Gamma \cup \Gamma^k$



Beispiel: Addition mittels 3-spuriger TM (1)

Die Verwendung einer mehrspurigen TM erlaubt es, Algorithmen einfacher zu beschreiben.

Wir verdeutlichen dies am Beispiel der Addition: Aus der Eingabe $\text{bin}(i_1)\#\,\text{bin}(i_2)$ für $i_1, i_2 \in \mathbb{N}$ soll $\text{bin}(i_1 + i_2)$ berechnet werden.

Wir verwenden eine 3-spurige TM mit den Alphabeten $\Sigma = \{0, 1, \#\}$ und

$$\Gamma = \left\{0, 1, \#, \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}, \ldots, \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}, B\right\}$$

Beispiel: Addition mittels 3-spuriger TM (2)

- *Schritt 1:* Transformation in Spurendarstellung. Schiebe die Eingabe so zusammen, dass die Binärkodierungen von i_1 und i_2 in der ersten und zweiten Spur rechtsbündig übereinander stehen.
  Aus der Eingabe 0011#0110 wird beispielsweise

$$B^* \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} B^*$$

- *Schritt 2:* Addition nach der Schulmethode, indem der Kopf das Band von rechts nach links abläuft. Überträge werden im Zustand gespeichert. Als Ergebnis auf Spur 3 ergibt sich

$$B^* \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix} B^*$$

- *Schritt 3:* Rücktransformation von Spur 3 ins Einspur-Format: Ausgabe 1001.

Techniken zur Programmierung von TMen (continued)

Standardtechniken aus der Programmierung können auch auf TMen implementiert werden.

- *Schleifen* haben wir bereits an Beispielen gesehen.
- *Variablen* können realisiert werden, indem wir pro Variable eine Spur reservieren.
- *Felder (Arrays)* können ebenfalls auf einer Spur abgespeichert werden.
- *Unterprogramme* können implementiert werden, indem wir eine Spur des Bandes als Prozedurstack verwenden.
- *Rekursive Unterprogramme* können ebenfalls implementiert werden.

Basierend auf diesen Techniken können wir uns klar machen, dass bekannte Algorithmen (wie z.B. MergeSort) ohne Weiteres auf einer TM ausgeführt werden können.