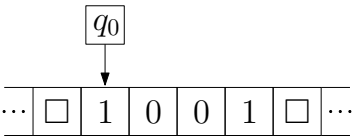
Übersicht

2.2.1 Turingmaschinen

Turingmaschine (TM) $\approx$ endlicher Automat mit Band mit unendlich vielen Speicherzellen



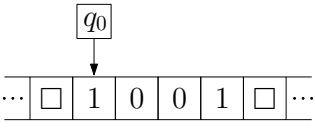
- Es gibt **Lese-/Schreibkopf**, der zu jedem Zeitpunkt auf einer Zelle steht.
- In jeder Zelle steht ein Zeichen aus endlichem **Bandalphabet** Γ.
- Zu jedem Zeitpunkt ist die TM in einem Zustand aus endlicher **Zustandsmenge** Q.
- Abhängig vom Zustand und dem gelesenen Zeichen
  (1) ändert die TM ihren Zustand,
  (2) schreibt ein Zeichen
  (3) und bewegt den Kopf.

2.2.1 Turingmaschinen

Definition 2.1

Eine **Turingmaschine (TM)** M ist ein 7-Tupel $(Q, \Sigma, \Gamma, \square, q_0, \bar{q}, \delta)$, das aus den folgenden Komponenten besteht.

- Q, die **Zustandsmenge**, ist eine endliche Menge von **Zuständen**.
- $\Sigma \supseteq \{0, 1\}$, das **Eingabealphabet**, ist eine endliche Menge von Zeichen.
- $\Gamma \supseteq \Sigma$, das **Bandalphabet**, ist eine endliche Menge von Zeichen.
- $\square \in \Gamma \setminus \Sigma$ ist das **Leerzeichen**.
- $q_0 \in Q$ ist der **Startzustand**.
- $\bar{q}$ ist der **Endzustand**.
- $\delta\colon (Q \setminus \{\bar{q}\}) \times \Gamma \to Q \times \Gamma \times \{L, N, R\}$ ist die **Zustandsüberführungsfunktion**.



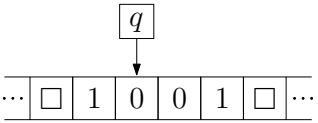
$\delta(q_0, 1) = (q, 1, R)$

2.2.1 Turingmaschinen

Definition 2.1

Eine **Turingmaschine (TM)** M ist ein 7-Tupel $(Q, \Sigma, \Gamma, \square, q_0, \bar{q}, \delta)$, das aus den folgenden Komponenten besteht.

- Q, die **Zustandsmenge**, ist eine endliche Menge von **Zuständen**.
- $\Sigma \supseteq \{0, 1\}$, das **Eingabealphabet**, ist eine endliche Menge von Zeichen.
- $\Gamma \supseteq \Sigma$, das **Bandalphabet**, ist eine endliche Menge von Zeichen.
- $\square \in \Gamma \setminus \Sigma$ ist das **Leerzeichen**.
- $q_0 \in Q$ ist der **Startzustand**.
- $\bar{q}$ ist der **Endzustand**.
- $\delta\colon (Q \setminus \{\bar{q}\}) \times \Gamma \to Q \times \Gamma \times \{L, N, R\}$ ist die **Zustandsüberführungsfunktion**.



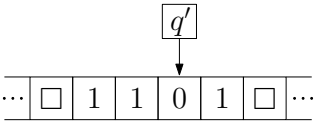
$$\delta(q, 0) = (q', 1, R)$$

2.2.1 Turingmaschinen

Definition 2.1

Eine **Turingmaschine (TM)** M ist ein 7-Tupel $(Q, \Sigma, \Gamma, \square, q_0, \bar{q}, \delta)$, das aus den folgenden Komponenten besteht.

- Q, die **Zustandsmenge**, ist eine endliche Menge von **Zuständen**.
- $\Sigma \supseteq \{0, 1\}$, das **Eingabealphabet**, ist eine endliche Menge von Zeichen.
- $\Gamma \supseteq \Sigma$, das **Bandalphabet**, ist eine endliche Menge von Zeichen.
- $\square \in \Gamma \setminus \Sigma$ ist das **Leerzeichen**.
- $q_0 \in Q$ ist der **Startzustand**.
- $\bar{q}$ ist der **Endzustand**.
- $\delta\colon (Q \setminus \{\bar{q}\}) \times \Gamma \to Q \times \Gamma \times \{L, N, R\}$ ist die **Zustandsüberführungsfunktion**.



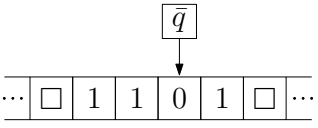
$\delta(q', 0) = (\bar{q}, 0, N)$

2.2.1 Turingmaschinen

Definition 2.1

Eine **Turingmaschine (TM)** M ist ein 7-Tupel $(Q, \Sigma, \Gamma, \square, q_0, \bar{q}, \delta)$, das aus den folgenden Komponenten besteht.

- Q, die **Zustandsmenge**, ist eine endliche Menge von **Zuständen**.
- $\Sigma \supseteq \{0, 1\}$, das **Eingabealphabet**, ist eine endliche Menge von Zeichen.
- $\Gamma \supseteq \Sigma$, das **Bandalphabet**, ist eine endliche Menge von Zeichen.
- $\square \in \Gamma \setminus \Sigma$ ist das **Leerzeichen**.
- $q_0 \in Q$ ist der **Startzustand**.
- $\bar{q}$ ist der **Endzustand**.
- $\delta\colon (Q \setminus \{\bar{q}\}) \times \Gamma \to Q \times \Gamma \times \{L, N, R\}$ ist die **Zustandsüberführungsfunktion**.

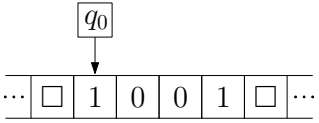


Initialisierung:

- Eingabe $w = w_1 \dots w_n \in \Sigma^*$ steht auf dem Band
- links und rechts von der Eingabe nur Leerzeichen
- Kopf steht auf erstem Zeichen der Eingabe
- Zustand q_0

Beispiel

Eingabe: 1001

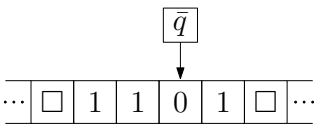


Ausgabe:

- Wenn Zustand $\bar{q}$ erreicht wird, produziert TM eine Ausgabe.
- Ausgabe beginnt an Kopfposition.
- Ausgabe endet direkt vor dem ersten Zeichen aus $\Gamma \setminus \Sigma$

Beispiel

Ausgabe: 01



Definition (partielle Funktion)

Eine Relation $R \subseteq A \times B$ zwischen den Mengen A und B ist eine Funktion $f : A \to B$, wenn folgende Eigenschaften gelten:

(i) $\forall a \in A \exists b \in B : (a, b) \in R$ (linksvollständig oder linkstotal)

(ii) $\forall a \in A \forall b, c \in B : (a, b) \in R \wedge (a, c) \in R \to b = c$ (rechtseindeutig)

Definition (partielle Funktion)

Eine Relation $R \subseteq A \times B$ zwischen den Mengen A und B ist eine Funktion $f\colon A \to B$, wenn folgende Eigenschaften gelten:

(i) $\forall a \in A \exists b \in B : (a, b) \in R$ (linksvollständig oder linkstotal)

(ii) $\forall a \in A \forall b, c \in B : (a, b) \in R \wedge (a, c) \in R \to b = c$ (rechtseindeutig)

Eine Relation für die nur (ii) aber nicht (i) gilt, wird auch als **partielle Funktion** bezeichnet. Eine partielle Funktion kann als Funktion $\bar{f}$ modelliert werden, mit

$$\bar{f}\colon A \to B \cup \{\bot\}, \quad a \mapsto \begin{cases} f(a) & \text{falls } a \in \mathrm{Def}(f) \\ \bot & \text{sonst} \end{cases}$$

wobei wir annehmen, dass $\bot \notin B$.

Definition (Funktion einer Turingmaschine)

Mit jeder TM M kann man eine Funktion $f_M \colon \Sigma^* \to \Sigma^* \cup \{\perp\}$ assoziieren, die für jede Eingabe $w \in \Sigma^*$ angibt, welche Ausgabe $f_M(w)$ die TM bei dieser Eingabe produziert.

Definition (Funktion einer Turingmaschine)

Mit jeder TM M kann man eine Funktion $f_M \colon \Sigma^* \to \Sigma^* \cup \{\perp\}$ assoziieren, die für jede Eingabe $w \in \Sigma^*$ angibt, welche Ausgabe $f_M(w)$ die TM bei dieser Eingabe produziert.

Erreicht die Turingmaschine M bei einer Eingabe w den Endzustand $\bar{q}$ nicht nach endlich vielen Schritten, so sagen wir, dass sie bei Eingabe w **nicht hält** (oder **nicht terminiert**), und wir definieren $f_M(w) = \perp$.

Definition (Funktion einer Turingmaschine)

Mit jeder TM M kann man eine Funktion $f_M \colon \Sigma^* \to \Sigma^* \cup \{\perp\}$ assoziieren, die für jede Eingabe $w \in \Sigma^*$ angibt, welche Ausgabe $f_M(w)$ die TM bei dieser Eingabe produziert.

Erreicht die Turingmaschine M bei einer Eingabe w den Endzustand $\bar{q}$ nicht nach endlich vielen Schritten, so sagen wir, dass sie bei Eingabe w **nicht hält** (oder **nicht terminiert**), und wir definieren $f_M(w) = \perp$.

Wir sagen, dass **die Turingmaschine M die Funktion f_M berechnet**.

Definition (Funktion einer Turingmaschine)

Mit jeder TM M kann man eine Funktion $f_M \colon \Sigma^* \to \Sigma^* \cup \{\bot\}$ assoziieren, die für jede Eingabe $w \in \Sigma^*$ angibt, welche Ausgabe $f_M(w)$ die TM bei dieser Eingabe produziert.

Erreicht die Turingmaschine M bei einer Eingabe w den Endzustand $\bar{q}$ nicht nach endlich vielen Schritten, so sagen wir, dass sie bei Eingabe w **nicht hält** (oder **nicht terminiert**), und wir definieren $f_M(w) = \bot$.

Wir sagen, dass **die Turingmaschine M die Funktion f_M berechnet**.

Definition 2.2

Eine (partielle) Funktion $f \colon \Sigma^* \to \Sigma^*$ heißt **berechenbar** (oder **rekursiv**), wenn es eine Turingmaschine M mit $f_M = f$ gibt. Für eine berechenbare Funktion f_M, die linkstotal ist, terminiert eine Turingmaschine M auf jeder Eingabe.

Definition 2.3

Eine Turingmaschine M **akzeptiert** eine Eingabe $w \in \Sigma^*$, wenn sie bei Eingabe w terminiert und ein Wort ausgibt, das mit 1 beginnt. Sie **verwirft** eine Eingabe $w \in \Sigma^*$, wenn sie bei Eingabe w terminiert und ein Wort ausgibt, das nicht mit 1 beginnt.

Definition 2.3

Eine Turingmaschine M **akzeptiert** eine Eingabe $w \in \Sigma^*$, wenn sie bei Eingabe w terminiert und ein Wort ausgibt, das mit 1 beginnt. Sie **verwirft** eine Eingabe $w \in \Sigma^*$, wenn sie bei Eingabe w terminiert und ein Wort ausgibt, das nicht mit 1 beginnt.

Eine Turingmaschine M **entscheidet** eine Sprache $L \subseteq \Sigma^*$, wenn sie jedes Wort $w \in L$ akzeptiert und jedes Wort $w \in \Sigma^* \setminus L$ verwirft.

Definition 2.3

Eine Turingmaschine M **akzeptiert** eine Eingabe $w \in \Sigma^*$, wenn sie bei Eingabe w terminiert und ein Wort ausgibt, das mit 1 beginnt. Sie **verwirft** eine Eingabe $w \in \Sigma^*$, wenn sie bei Eingabe w terminiert und ein Wort ausgibt, das nicht mit 1 beginnt.

Eine Turingmaschine M **entscheidet** eine Sprache $L \subseteq \Sigma^*$, wenn sie jedes Wort $w \in L$ akzeptiert und jedes Wort $w \in \Sigma^* \setminus L$ verwirft.

Eine Sprache $L \subseteq \{0,1\}^*$ heißt **entscheidbar** oder **rekursiv**, wenn es eine Turingmaschine M gibt, die L entscheidet. Wir sagen dann, dass M **eine Turingmaschine für die Sprache L ist**. Eine solche Turingmaschine terminiert insbesondere auf jeder Eingabe.

Beispiel:

Betrachte TM $M = (Q, \Sigma, \Gamma, \square, q_0, \bar{q}, \delta)$ mit $Q = \{q_0, q_1, q_2, \bar{q}\}$, $\Sigma = \{0, 1\}$ und $\Gamma = \{0, 1, \square\}$. Die Zustandsüberführungsfunktion δ sei wie folgt:

	q_0	q_1	q_2
0	$(q_1, 0, R)$	$(q_1, 0, R)$	$(q_1, 0, R)$
1	$(q_0, 1, R)$	$(q_2, 1, R)$	$(q_0, 1, R)$
$\square$	$(\bar{q}, 0, N)$	$(\bar{q}, 0, N)$	$(\bar{q}, 1, N)$

Beispiel:

Betrachte TM $M = (Q, \Sigma, \Gamma, \square, q_0, \bar{q}, \delta)$ mit $Q = \{q_0, q_1, q_2, \bar{q}\}$, $\Sigma = \{0, 1\}$ und $\Gamma = \{0, 1, \square\}$. Die Zustandsüberführungsfunktion δ sei wie folgt:

	q_0	q_1	q_2
0	$(q_1, 0, R)$	$(q_1, 0, R)$	$(q_1, 0, R)$
1	$(q_0, 1, R)$	$(q_2, 1, R)$	$(q_0, 1, R)$
$\square$	$(\bar{q}, 0, N)$	$(\bar{q}, 0, N)$	$(\bar{q}, 1, N)$

Diese TM **verhält sich wie ein endlicher Automat**, der die Eingabe Zeichen für Zeichen von links nach rechts liest und dabei Zustandsübergänge durchführt. Sie akzeptiert die Eingabe genau dann, wenn sie das erste Leerzeichen im Zustand q_2 erreicht.

Beispiel:

Betrachte TM $M = (Q, \Sigma, \Gamma, \square, q_0, \bar{q}, \delta)$ mit $Q = \{q_0, q_1, q_2, \bar{q}\}$, $\Sigma = \{0, 1\}$ und $\Gamma = \{0, 1, \square\}$. Die Zustandsüberführungsfunktion δ sei wie folgt:

	q_0	q_1	q_2
0	$(q_1, 0, R)$	$(q_1, 0, R)$	$(q_1, 0, R)$
1	$(q_0, 1, R)$	$(q_2, 1, R)$	$(q_0, 1, R)$
$\square$	$(\bar{q}, 0, N)$	$(\bar{q}, 0, N)$	$(\bar{q}, 1, N)$

Diese TM **verhält sich wie ein endlicher Automat**, der die Eingabe Zeichen für Zeichen von links nach rechts liest und dabei Zustandsübergänge durchführt. Sie akzeptiert die Eingabe genau dann, wenn sie das erste Leerzeichen im Zustand q_2 erreicht.

Zustand q_2 wird genau dann erreicht, wenn der **bisher gelesene Teil der Eingabe mit 01 endet**.

Techniken zum Entwurf von Turingmaschinen:

1. **Variablen mit endlichen Wertebereichen**

   Möchte man beispielsweise eine Variable realisieren, die für ein festes $k \in \mathbb{N}$ Werte aus der Menge $\{0, \ldots, k\}$ annehmen kann, so kann man die Zustandsmenge Q zu der Menge $Q' = Q \times \{0, \ldots, k\}$ erweitern.

Techniken zum Entwurf von Turingmaschinen:

1. **Variablen mit endlichen Wertebereichen**

   Möchte man beispielsweise eine Variable realisieren, die für ein festes $k \in \mathbb{N}$ Werte aus der Menge $\{0, \ldots, k\}$ annehmen kann, so kann man die Zustandsmenge Q zu der Menge $Q' = Q \times \{0, \ldots, k\}$ erweitern.

2. **Bänder mit mehreren Spuren**

   In jeder Zelle stehen k Zeichen aus Γ, die alle gleichzeitig in einem Schritt gelesen und geschrieben werden. Ist $k \in \mathbb{N}$ eine Konstante, so kann dies dadurch realisiert werden, dass das Bandalphabet Γ zu $\Gamma' = \Sigma \cup \Gamma^k$ erweitert wird.

Techniken zum Entwurf von Turingmaschinen:

1. **Variablen mit endlichen Wertebereichen**

   Möchte man beispielsweise eine Variable realisieren, die für ein festes $k \in \mathbb{N}$ Werte aus der Menge $\{0, \ldots, k\}$ annehmen kann, so kann man die Zustandsmenge Q zu der Menge $Q' = Q \times \{0, \ldots, k\}$ erweitern.

2. **Bänder mit mehreren Spuren**

   In jeder Zelle stehen k Zeichen aus Γ, die alle gleichzeitig in einem Schritt gelesen und geschrieben werden. Ist $k \in \mathbb{N}$ eine Konstante, so kann dies dadurch realisiert werden, dass das Bandalphabet Γ zu $\Gamma' = \Sigma \cup \Gamma^k$ erweitert wird.

3. **Variablen mit unendlichen Wertebereichen**

   Für jede Variable eine Spur, auf der ihr Wert gespeichert ist.

Techniken zum Entwurf von Turingmaschinen:

1. **Variablen mit endlichen Wertebereichen**
   Möchte man beispielsweise eine Variable realisieren, die für ein festes $k \in \mathbb{N}$ Werte aus der Menge $\{0, \ldots, k\}$ annehmen kann, so kann man die Zustandsmenge Q zu der Menge $Q' = Q \times \{0, \ldots, k\}$ erweitern.
2. **Bänder mit mehreren Spuren**
   In jeder Zelle stehen k Zeichen aus Γ, die alle gleichzeitig in einem Schritt gelesen und geschrieben werden. Ist $k \in \mathbb{N}$ eine Konstante, so kann dies dadurch realisiert werden, dass das Bandalphabet Γ zu $\Gamma' = \Sigma \cup \Gamma^k$ erweitert wird.
3. **Variablen mit unendlichen Wertebereichen**
   Für jede Variable eine Spur, auf der ihr Wert gespeichert ist.
4. **Unterprogramme**

Techniken zum Entwurf von Turingmaschinen:

1. **Variablen mit endlichen Wertebereichen**
   Möchte man beispielsweise eine Variable realisieren, die für ein festes $k \in \mathbb{N}$ Werte aus der Menge $\{0, \ldots, k\}$ annehmen kann, so kann man die Zustandsmenge Q zu der Menge $Q' = Q \times \{0, \ldots, k\}$ erweitern.
2. **Bänder mit mehreren Spuren**
   In jeder Zelle stehen k Zeichen aus Γ, die alle gleichzeitig in einem Schritt gelesen und geschrieben werden. Ist $k \in \mathbb{N}$ eine Konstante, so kann dies dadurch realisiert werden, dass das Bandalphabet Γ zu $\Gamma' = \Sigma \cup \Gamma^k$ erweitert wird.
3. **Variablen mit unendlichen Wertebereichen**
   Für jede Variable eine Spur, auf der ihr Wert gespeichert ist.
4. **Unterprogramme**
5. **for- und while-Schleifen**

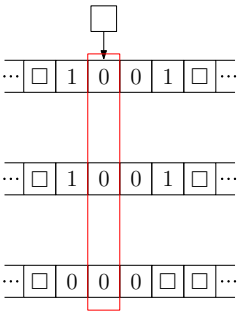
Turingmaschinen mit mehreren Bändern:

Turingmaschine mit k Bändern und k **separaten Lese-/Schreibköpfen**

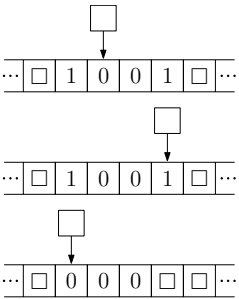
Turingmaschinen mit mehreren Bändern:

Turingmaschine mit k Bändern und k **separaten Lese-/Schreibköpfen**

k-Spur TM



k-Band TM



$$\delta\colon (Q \setminus \{\bar{q}\}) \times \Gamma^k \to Q \times \Gamma^k \times \{L, N, R\}^k$$

2.2.1 Turingmaschinen

Definition 2.4

Es sei M eine k-Band-Turingmaschine.

Die **Rechenzeit $t_M(w)$ von M auf Eingabe w** ist die Anzahl an Rechenschritten, die M bei Eingabe w bis zur Terminierung durchführt. Terminiert M auf w nicht, so ist die Rechenzeit unendlich.

Definition 2.4

Es sei M eine k-Band-Turingmaschine.

Die **Rechenzeit $t_M(w)$ von M auf Eingabe w** ist die Anzahl an Rechenschritten, die M bei Eingabe w bis zur Terminierung durchführt. Terminiert M auf w nicht, so ist die Rechenzeit unendlich.

Der **Platzbedarf $s_M(w)$ von M auf Eingabe w** ist die Anzahl (summiert über alle Bänder) an verschiedenen Zellen, auf denen sich im Laufe der Rechnung mindestens einmal ein Lese-/Schreibkopf befunden hat.

2.2.1 Turingmaschinen

Definition 2.4

Es sei M eine k-Band-Turingmaschine.

Die **Rechenzeit $t_M(w)$ von M auf Eingabe w** ist die Anzahl an Rechenschritten, die M bei Eingabe w bis zur Terminierung durchführt. Terminiert M auf w nicht, so ist die Rechenzeit unendlich.

Der **Platzbedarf $s_M(w)$ von M auf Eingabe w** ist die Anzahl (summiert über alle Bänder) an verschiedenen Zellen, auf denen sich im Laufe der Rechnung mindestens einmal ein Lese-/Schreibkopf befunden hat.

Die **Rechenzeit $t_M(n)$ von M auf Eingaben der Länge n** ist definiert als $t_M(n) = \max_{w \in \Sigma^n} t_M(w)$.

2.2.1 Turingmaschinen

Definition 2.4

Es sei M eine k-Band-Turingmaschine.

Die **Rechenzeit $t_M(w)$ von M auf Eingabe w** ist die Anzahl an Rechenschritten, die M bei Eingabe w bis zur Terminierung durchführt. Terminiert M auf w nicht, so ist die Rechenzeit unendlich.

Der **Platzbedarf $s_M(w)$ von M auf Eingabe w** ist die Anzahl (summiert über alle Bänder) an verschiedenen Zellen, auf denen sich im Laufe der Rechnung mindestens einmal ein Lese-/Schreibkopf befunden hat.

Die **Rechenzeit $t_M(n)$ von M auf Eingaben der Länge n** ist definiert als $t_M(n) = \max_{w \in \Sigma^n} t_M(w)$.

Analog ist der **Platzbedarf $s_M(n)$ von M auf Eingaben der Länge n** als $s_M(n) = \max_{w \in \Sigma^n} s_M(w)$ definiert.

Theorem 2.5

Eine k-Band Turingmaschine M mit Rechenzeit $t(n)$ und Platzbedarf $s(n)$ kann durch eine 1-Band-Turingmaschine M' mit Rechenzeit $O(t(n)^2)$ und Platzbedarf $O(s(n))$ simuliert werden.

Theorem 2.5

Eine k-Band Turingmaschine M mit Rechenzeit $t(n)$ und Platzbedarf $s(n)$ kann durch eine 1-Band-Turingmaschine M' mit Rechenzeit $O(t(n)^2)$ und Platzbedarf $O(s(n))$ simuliert werden.

Beweis: M' simuliert M Schritt für Schritt und verwendet $2k$ Spuren.

Theorem 2.5

Eine k-Band Turingmaschine M mit Rechenzeit $t(n)$ und Platzbedarf $s(n)$ kann durch eine 1-Band-Turingmaschine M' mit Rechenzeit $O(t(n)^2)$ und Platzbedarf $O(s(n))$ simuliert werden.

Beweis: M' simuliert M Schritt für Schritt und verwendet $2k$ Spuren.

Nach der Simulation des t-ten Rechenschrittes von M gilt folgende **Invariante**:

1. Die ungeraden Spuren $1, 3, \ldots, 2k-1$ enthalten den Inhalt der k Bänder von M.

Theorem 2.5

Eine k-Band Turingmaschine M mit Rechenzeit $t(n)$ und Platzbedarf $s(n)$ kann durch eine 1-Band-Turingmaschine M' mit Rechenzeit $O(t(n)^2)$ und Platzbedarf $O(s(n))$ simuliert werden.

Beweis: M' simuliert M Schritt für Schritt und verwendet $2k$ Spuren.

Nach der Simulation des t-ten Rechenschrittes von M gilt folgende **Invariante**:

1. Die ungeraden Spuren $1, 3, \ldots, 2k-1$ enthalten den Inhalt der k Bänder von M.
2. Auf den geraden Spuren $2, 4, \ldots, 2k$ sind die Kopfpositionen von M mit dem Zeichen $\#$ markiert.

Theorem 2.5

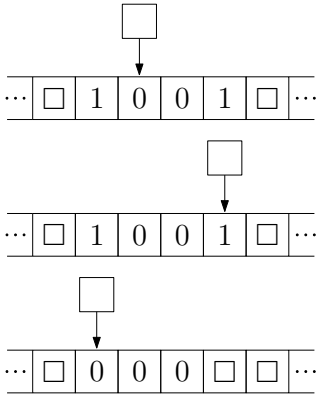
Eine k-Band Turingmaschine M mit Rechenzeit $t(n)$ und Platzbedarf $s(n)$ kann durch eine 1-Band-Turingmaschine M' mit Rechenzeit $O(t(n)^2)$ und Platzbedarf $O(s(n))$ simuliert werden.

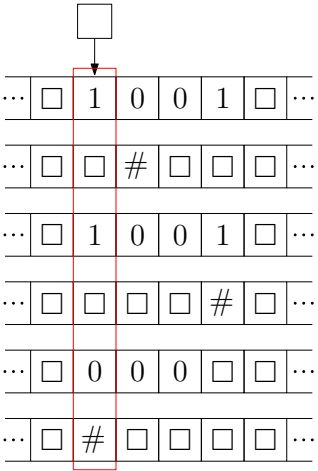
Beweis: M' simuliert M Schritt für Schritt und verwendet $2k$ Spuren.

Nach der Simulation des t-ten Rechenschrittes von M gilt folgende **Invariante**:

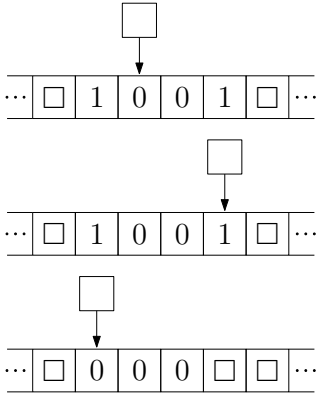
1. Die ungeraden Spuren $1, 3, \ldots, 2k-1$ enthalten den Inhalt der k Bänder von M.
2. Auf den geraden Spuren $2, 4, \ldots, 2k$ sind die Kopfpositionen von M mit dem Zeichen $\#$ markiert.
3. Der Kopf von M' steht an der Position am weitesten links, die auf einem der geraden Bänder mit $\#$ markiert ist.

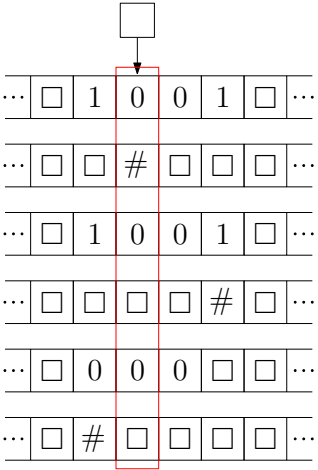
2.2.1 Turingmaschinen



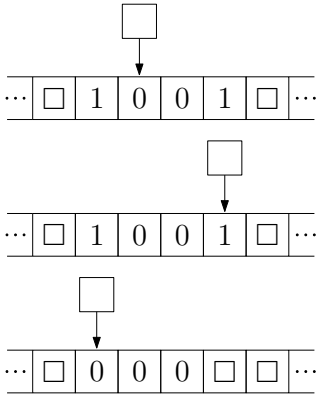


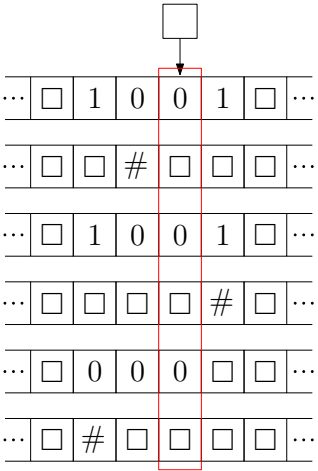
2.2.1 Turingmaschinen



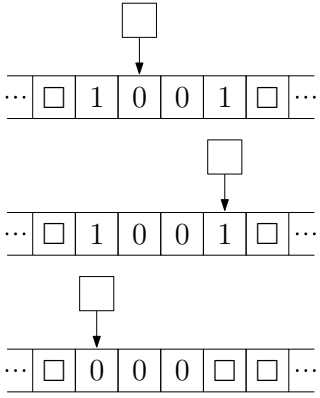


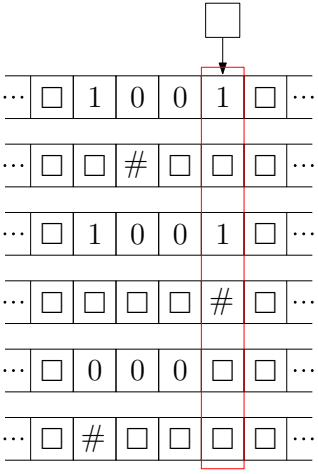
2.2.1 Turingmaschinen



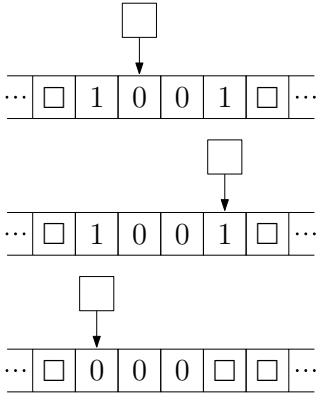


2.2.1 Turingmaschinen

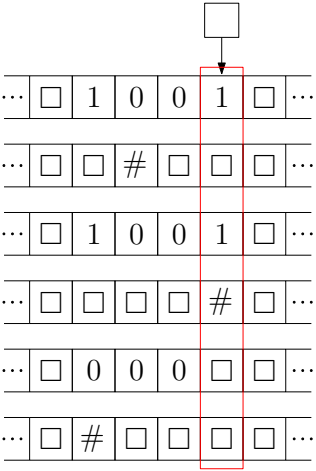




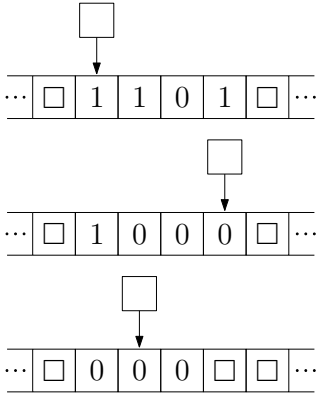
2.2.1 Turingmaschinen



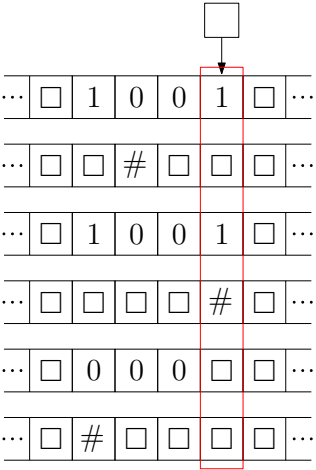
$$\delta(q, (0,1,0)) = (q', (1,0,0), (L, N, R))$$



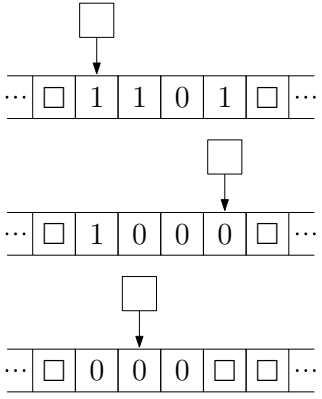
2.2.1 Turingmaschinen



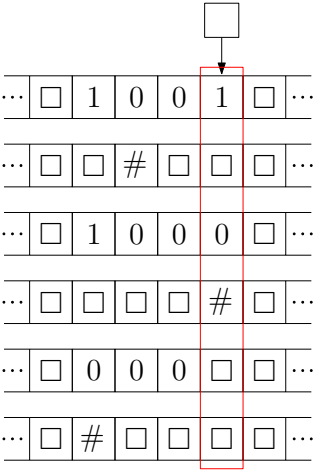
$$\delta(q, (0,1,0)) = (q', (1,0,0), (L, N, R))$$



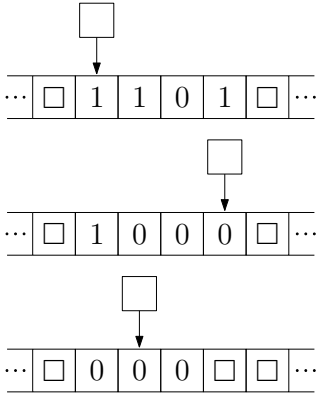
2.2.1 Turingmaschinen



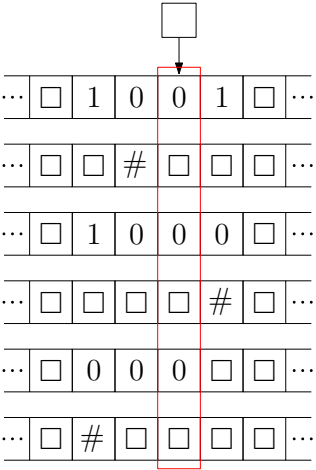
$$\delta(q, (0,1,0)) = (q', (1,0,0), (L, N, R))$$



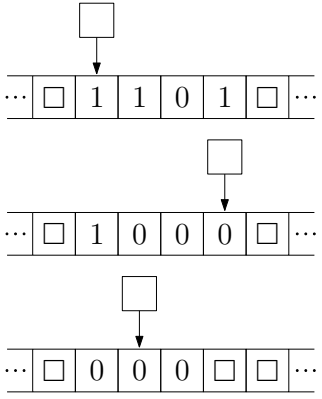
2.2.1 Turingmaschinen



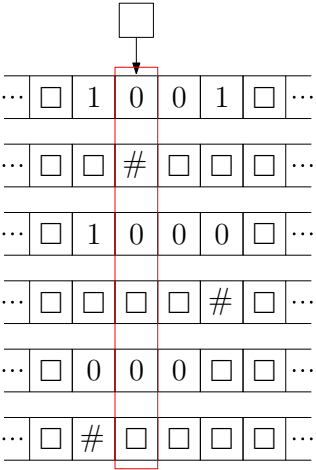
$$\delta(q, (0,1,0)) = (q', (1,0,0), (L, N, R))$$



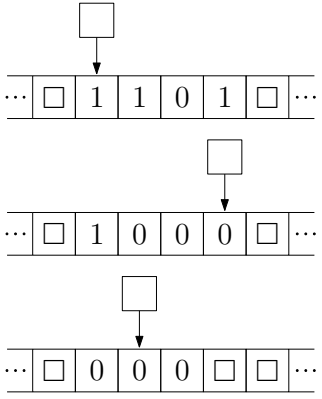
2.2.1 Turingmaschinen



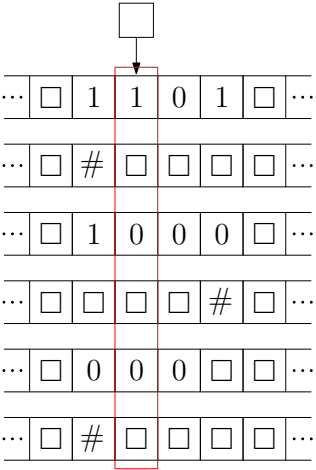
$$\delta(q, (0,1,0)) = (q', (1,0,0), (L, N, R))$$



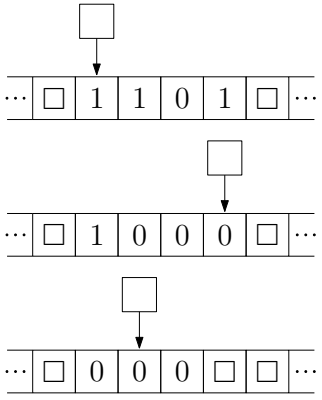
2.2.1 Turingmaschinen



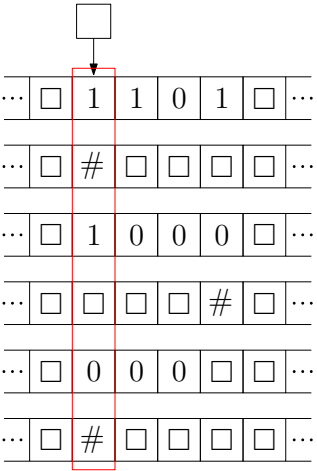
$$\delta(q, (0,1,0)) = (q', (1,0,0), (L, N, R))$$



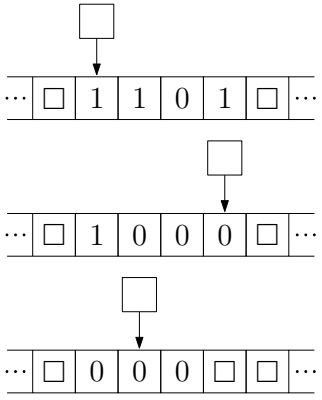
2.2.1 Turingmaschinen



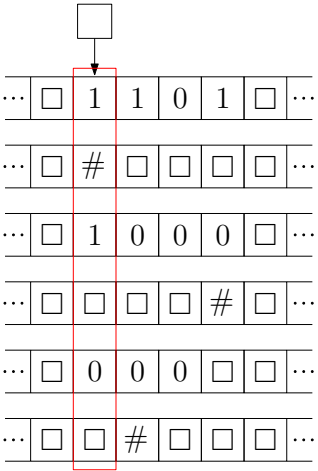
$$\delta(q, (0,1,0)) = (q', (1,0,0), (L, N, R))$$



2.2.1 Turingmaschinen



$$\delta(q, (0,1,0)) = (q', (1,0,0), (L, N, R))$$



Simulation eines Schrittes von M

1. M' läuft von linkem $\#$ zu rechtem $\#$ und liest dabei die Zeichen an den Kopfpositionen von M.
2. M' läuft von rechtem $\#$ zu linkem $\#$ und ändert dabei den Bandinhalt sowie die Kopfpositionen.

Simulation eines Schrittes von M

1. M' läuft von linkem $\#$ zu rechtem $\#$ und liest dabei die Zeichen an den Kopfpositionen von M.
2. M' läuft von rechtem $\#$ zu linkem $\#$ und ändert dabei den Bandinhalt sowie die Kopfpositionen.

Laufzeit pro Schritt von M: $O(D)$, wobei D Abstand von rechtem zu linkem $\#$ bezeichnet.

Simulation eines Schrittes von M

1. M' läuft von linkem $\#$ zu rechtem $\#$ und liest dabei die Zeichen an den Kopfpositionen von M.
2. M' läuft von rechtem $\#$ zu linkem $\#$ und ändert dabei den Bandinhalt sowie die Kopfpositionen.

Laufzeit pro Schritt von M: $O(D)$, wobei D Abstand von rechtem zu linkem $\#$ bezeichnet.
Es gilt $D \leq 2t(n)$.

Simulation eines Schrittes von M

1. M' läuft von linkem $\#$ zu rechtem $\#$ und liest dabei die Zeichen an den Kopfpositionen von M.
2. M' läuft von rechtem $\#$ zu linkem $\#$ und ändert dabei den Bandinhalt sowie die Kopfpositionen.

Laufzeit pro Schritt von M: $O(D)$, wobei D Abstand von rechtem zu linkem $\#$ bezeichnet.
Es gilt $D \leq 2t(n)$.
$\Rightarrow$ Gesamtlaufzeit der Simulation $O(t(n)^2)$

2.2.2 Registermaschinen

Registermaschine (RAM) $\approx$ rudimentäre Assemblersprache

Syntax	Zustandsänderung	Änderung von b
LOAD i	$c(0) := c(i)$	$b := b + 1$
CLOAD i	$c(0) := i$	$b := b + 1$
INDLOAD i	$c(0) := c(c(i))$	$b := b + 1$
STORE i	$c(i) := c(0)$	$b := b + 1$
INDSTORE i	$c(c(i)) := c(0)$	$b := b + 1$
ADD i	$c(0) := c(0) + c(i)$	$b := b + 1$
CADD i	$c(0) := c(0) + i$	$b := b + 1$
INDADD i	$c(0) := c(0) + c(c(i))$	$b := b + 1$
SUB i	$c(0) := c(0) - c(i)$	$b := b + 1$
CSUB i	$c(0) := c(0) - i$	$b := b + 1$
INDSUB i	$c(0) := c(0) - c(c(i))$	$b := b + 1$
MULT i	$c(0) := c(0) \cdot c(i)$	$b := b + 1$
CMULT i	$c(0) := c(0) \cdot i$	$b := b + 1$
INDMULT i	$c(0) := c(0) \cdot c(c(i))$	$b := b + 1$
DIV i	$c(0) := \lfloor c(0)/c(i) \rfloor$	$b := b + 1$
CDIV i	$c(0) := \lfloor c(0)/i \rfloor$	$b := b + 1$
INDDIV i	$c(0) := \lfloor c(0)/c(c(i)) \rfloor$	$b := b + 1$

2.2.2 Registermaschinen

Syntax	Zustandsänderung	Änderung von b
GOTO j	-	$b := j$
IF $c(0) = x$ GOTO j	-	$b := \begin{cases} j \text{ falls } c(0) = x \\ b+1 \text{ sonst} \end{cases}$
IF $c(0) < x$ GOTO j	-	$b := \begin{cases} j \text{ falls } c(0) < x \\ b+1 \text{ sonst} \end{cases}$
IF $c(0) \leq x$ GOTO j	-	$b := \begin{cases} j \text{ falls } c(0) \leq x \\ b+1 \text{ sonst} \end{cases}$
END	Ende der Rechnung	-

Laufzeit einer Registermaschine

uniformes Kostenmaß: Ausführung jedes Befehls benötigt eine Zeiteinheit

Vorteil: einfach

Nachteil: nicht realistisch bei Algorithmen, die mit sehr großen Zahlen arbeiten

Laufzeit einer Registermaschine

uniformes Kostenmaß: Ausführung jedes Befehls benötigt eine Zeiteinheit

Vorteil: einfach

Nachteil: nicht realistisch bei Algorithmen, die mit sehr großen Zahlen arbeiten

logarithmisches Kostenmaß: Laufzeit eines Befehls proportional zu der Länge der Zahlen in den angesprochenen Registern in Binärdarstellung (also zum Logarithmus der Zahlen).

2.2.2 Registermaschinen

Beispiel: Registermaschine zur Berechnung von $\sum_{i=0}^{n} i$

Eingabe $n \in \mathbb{N}$ zu Beginn in Register $c(1)$.

2.2.2 Registermaschinen

Beispiel: Registermaschine zur Berechnung von $\sum_{i=0}^{n} i$

Eingabe $n \in \mathbb{N}$ zu Beginn in Register $c(1)$.

Höhere Programmiersprache

```
s = n
i = n
while(i != 0)
 i = i - 1
 s = s + i
return s
```

2.2.2 Registermaschinen

Beispiel: Registermaschine zur Berechnung von $\sum_{i=0}^{n} i$

Eingabe $n \in \mathbb{N}$ zu Beginn in Register $c(1)$.

Höhere Programmiersprache

```
s = n
i = n
while(i != 0)
 i = i - 1
 s = s + i
return s
```

Registermaschine

// $c(1) = i \quad c(2) = s$

```
 1. LOAD(1)
 2. STORE(2)
 3. CSUB(1)
 4. STORE(1)
 5. ADD(2)
 6. STORE(2)
 7. LOAD(1)
 8. IF c(0)=0 GOTO 10
 9. GOTO 3
10. LOAD(2)
11. END
```

2.2.2 Registermaschinen

Beispiel: Registermaschine zur Berechnung von $\sum_{i=0}^{n} i$

Eingabe $n \in \mathbb{N}$ zu Beginn in Register $c(1)$.

Höhere Programmiersprache

```
s = n
i = n
while(i != 0)
 i = i - 1
 s = s + i
return s
```

Registermaschine

// $c(1) = i \quad c(2) = s$

```
 1. LOAD(1)
 2. STORE(2)
 3. CSUB(1)
 4. STORE(1)
 5. ADD(2)
 6. STORE(2)
 7. LOAD(1)
 8. IF c(0)=0 GOTO 10
 9. GOTO 3
10. LOAD(2)
11. END
```

Laufzeit der Registermaschine

Anzahl Operationen $O(n)$

Laufzeit im logarithmischen Kostenmaß: $\Theta(n \log n)$

2.2.2 Registermaschinen

Theorem 2.6

Jede im logarithmischen Kostenmaß $t(n)$-zeitbeschränkte Registermaschine kann durch eine Turingmaschine simuliert werden, deren Rechenzeit $O(q(n + t(n)))$ für ein Polynom q beträgt.

2.2.2 Registermaschinen

Theorem 2.6

Jede im logarithmischen Kostenmaß $t(n)$-zeitbeschränkte Registermaschine kann durch eine Turingmaschine simuliert werden, deren Rechenzeit $O(q(n + t(n)))$ für ein Polynom q beträgt.

Ist die Laufzeit der Registermaschine polynomiell, so auch die der TM:

2.2.2 Registermaschinen

Theorem 2.6

Jede im logarithmischen Kostenmaß $t(n)$-zeitbeschränkte Registermaschine kann durch eine Turingmaschine simuliert werden, deren Rechenzeit $O(q(n + t(n)))$ für ein Polynom q beträgt.

Ist die Laufzeit der Registermaschine polynomiell, so auch die der TM:

Laufzeit der Registermaschine sei $t(n) = O(n^d)$ für $d \geq 1$.

2.2.2 Registermaschinen

Theorem 2.6

Jede im logarithmischen Kostenmaß $t(n)$-zeitbeschränkte Registermaschine kann durch eine Turingmaschine simuliert werden, deren Rechenzeit $O(q(n + t(n)))$ für ein Polynom q beträgt.

Ist die Laufzeit der Registermaschine polynomiell, so auch die der TM:

Laufzeit der Registermaschine sei $t(n) = O(n^d)$ für $d \geq 1$.

Gemäß Theorem 2.6 existiert ein Polynom q mit Grad $d^\star$, für das die Rechenzeit der TM wie folgt beschränkt ist:

$$O(q(n + t(n))) = O(q(t(n))) = O(q(n^d)) = O(n^{dd^\star}).$$

Theorem 2.6

Jede im logarithmischen Kostenmaß $t(n)$-zeitbeschränkte Registermaschine kann durch eine Turingmaschine simuliert werden, deren Rechenzeit $O(q(n + t(n)))$ für ein Polynom q beträgt.

Ist die Laufzeit der Registermaschine polynomiell, so auch die der TM:

Laufzeit der Registermaschine sei $t(n) = O(n^d)$ für $d \geq 1$.
Gemäß Theorem 2.6 existiert ein Polynom q mit Grad $d^\star$, für das die Rechenzeit der TM wie folgt beschränkt ist:

$$O(q(n + t(n))) = O(q(t(n))) = O(q(n^d)) = O(n^{dd^\star}).$$

Damit ist auch die Laufzeit der TM durch ein Polynom beschränkt.

Theorem 2.7

Jede Turingmaschine, deren Rechenzeit durch $t(n)$ beschränkt ist, kann durch eine im logarithmischen Kostenmaß $O((t(n) + n) \log(t(n) + n))$-zeitbeschränkte Registermaschine simuliert werden.

2.2.2 Registermaschinen

Theorem 2.7

Jede Turingmaschine, deren Rechenzeit durch $t(n)$ beschränkt ist, kann durch eine im logarithmischen Kostenmaß $O((t(n) + n) \log(t(n) + n))$-zeitbeschränkte Registermaschine simuliert werden.

Theorem 2.6 und 2.7 implizieren Folgendes:

Klasse der von Turingmaschinen in polynomieller Zeit berechenbaren Funktionen
$=$ Klasse der von Registermaschinen in polynomieller Zeit berechenbaren Funktionen

2.2.3 Die Church-Turing-These

These 2.8 (Church-Turing-These)

Alle „intuitiv berechenbaren“ Funktionen können von Turingmaschinen berechnet werden.

2.2.3 Die Church-Turing-These

These 2.8 (Church-Turing-These)

Alle „intuitiv berechenbaren“ Funktionen können von Turingmaschinen berechnet werden.

These 2.9 (Physikalische Church-Turing-These)

Die Gesetze der Physik erlauben es nicht, eine Maschine zu konstruieren, die eine Funktion berechnet, die nicht auch von einer Turingmaschine berechnet werden kann.

2.2.3 Die Church-Turing-These

These 2.8 (Church-Turing-These)

Alle „intuitiv berechenbaren“ Funktionen können von Turingmaschinen berechnet werden.

These 2.9 (Physikalische Church-Turing-These)

Die Gesetze der Physik erlauben es nicht, eine Maschine zu konstruieren, die eine Funktion berechnet, die nicht auch von einer Turingmaschine berechnet werden kann.

These (Erweiterte Church-Turing-These)

Die Klasse der in polynomieller Zeit berechenbaren Funktionen ist für jedes realistische Maschinenmodell dieselbe.