Karlsruher Institut für Technologie

Theoretische Grundlagen der Informatik
Vorlesung am 11. November 2014

INSTITUT FÜR THEORETISCHE INFORMATIK

Turing-Maschinen und Berechenbarkeit

Karlsruher Institut für Technologie

Beobachtung:
Endliche Automaten sind als Berechnungsmodell nicht mächtig genug.

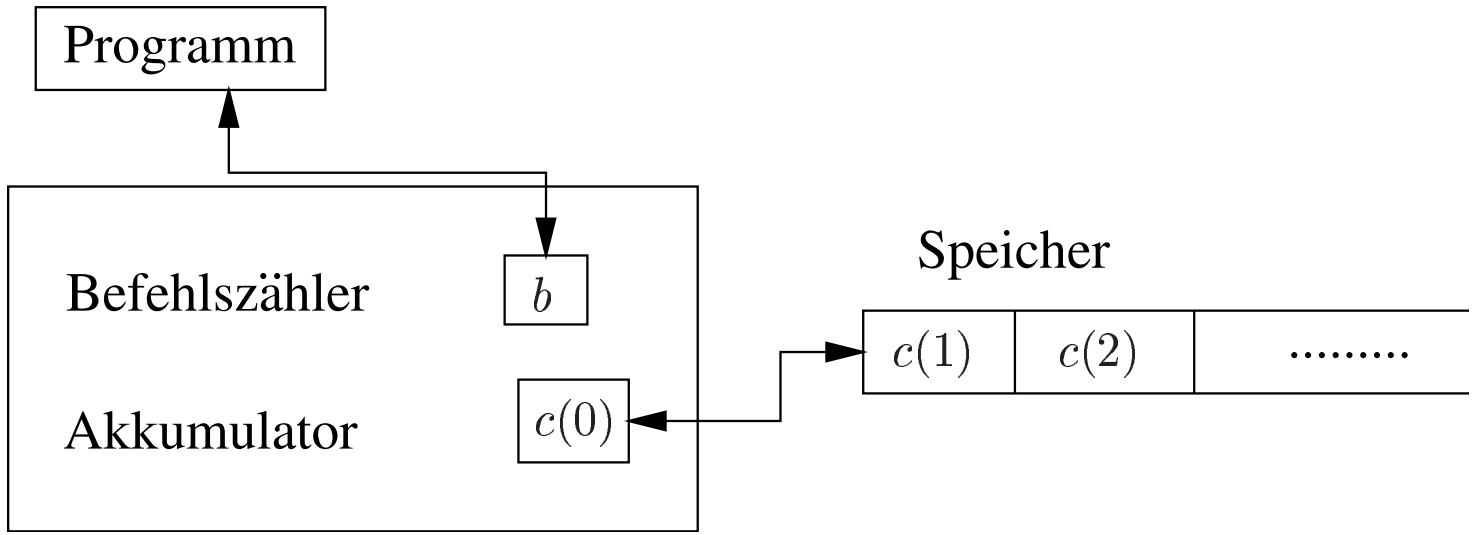
Frage:
Gibt es ein mächtigeres, realistisches Rechnermodell, das als Grundlage für allgemeine theoretische Aussagen über Berechenbarkeit, Entscheidbarkeit und Komplexität geeignet ist?

Hauptfrage in diesem Kapitel:
Welche Probleme sind berechenbar?

Die Registermaschine (RAM)

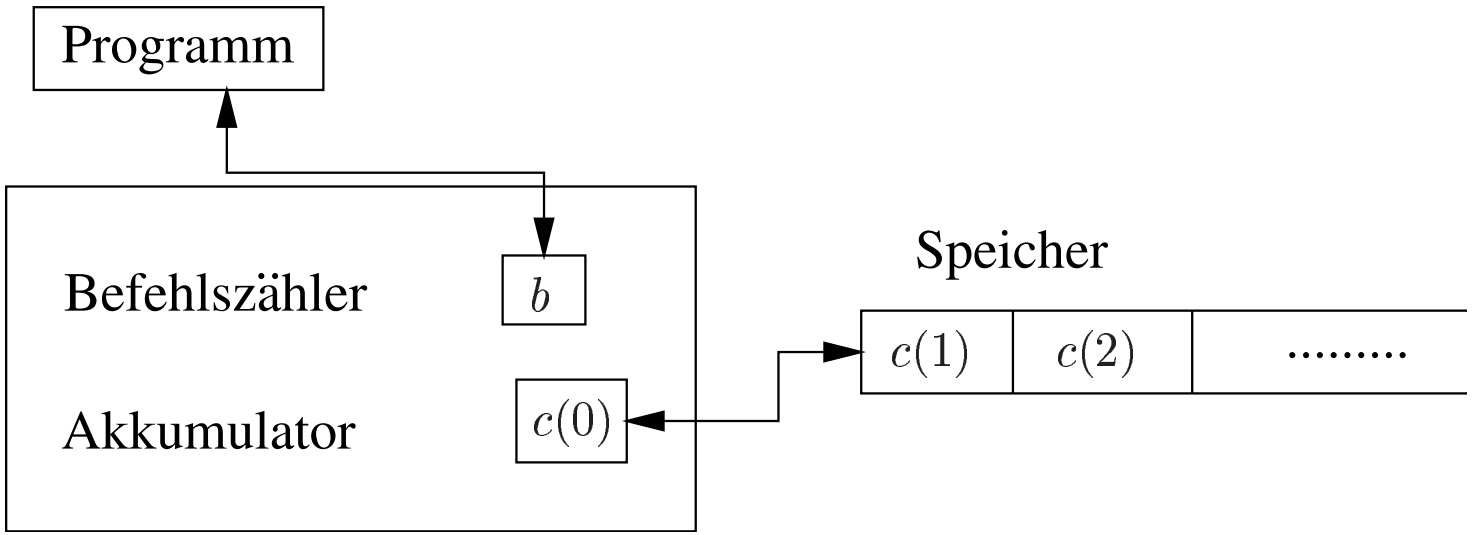
Die RAM besteht aus

- Befehlszähler (zeigt auf den nächsten Befehl im Programm),
- Akkumulatoren (endlicher Speicher zum Ausführen der Befehle),
- Registern (unendlicher Speicher), und
- Programm.



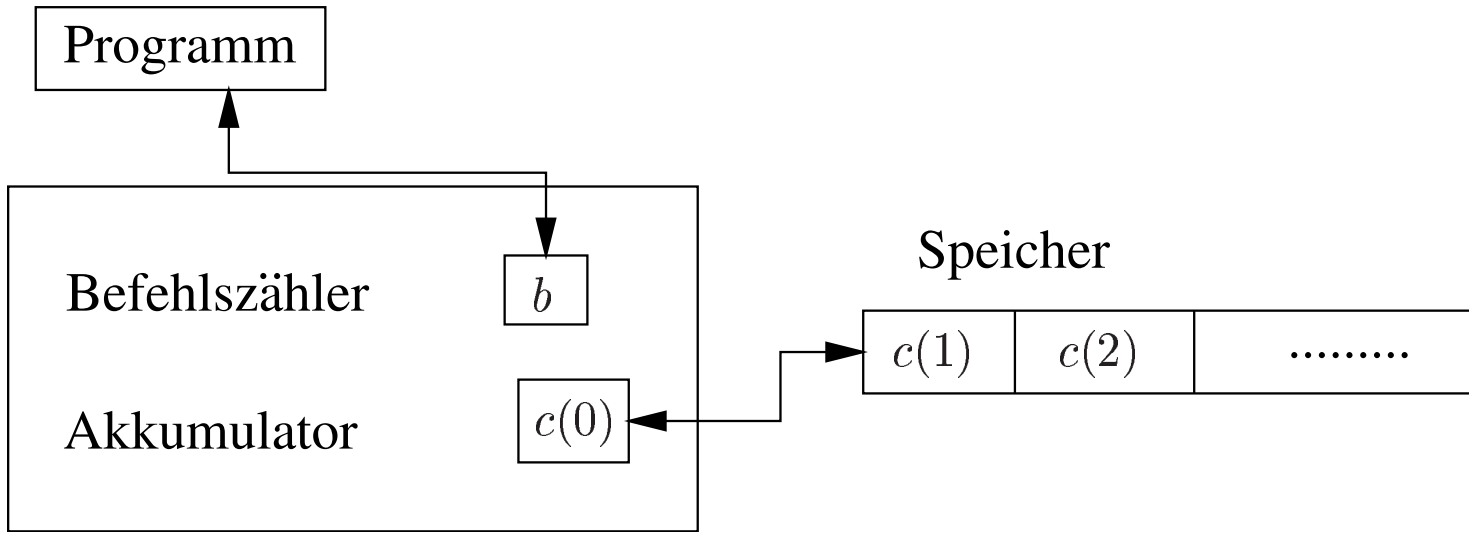
Die Registermaschine (RAM)

- Ein Programm besteht aus einer Folge von Befehlen.
- Programmzeilen sind durchnummeriert.
- Der Befehlszähler *b* startet bei 1 und enthält jeweils die Nummer des nächsten auszuführenden Befehls.



Die Registermaschine (RAM)

- In den ersten Registern steht zu Beginn der Berechnung die Eingabe.
- In den übrigen Registern steht 0.
- Am Ende der Berechnung stehen die Ausgabedaten in vorher festgelegten Registern.
- Den Inhalt des Registers i bezeichnen wir mit $c(i)$.



Befehle der Registermaschine (RAM)

Befehl	Wirkung
LOAD i	$c(0) := c(i); \quad b := b+1$
STORE i	$c(i) := c(0); \quad b := b+1$
ADD i	$c(0) := c(0) + c(i); \quad b := b+1$
SUB i	$c(0) := \max\{0, c(0) - c(i)\}; \quad b := b+1$
MULT i	$c(0) := c(0) \cdot c(i); \quad b := b+1$
DIV i	$c(0) := \left\lfloor \frac{c(0)}{c(i)} \right\rfloor; \quad b := b+1$
GOTO j	$b := j$
IF $c(0) \# \ell$ GOTO j	$\begin{cases} b := j & \text{falls } c(0) \# \ell \\ b := b+1 & \text{sonst} \end{cases}$ wobei $\# \in \{\leq, \geq, <, >, \neq, =\}$
END	$b := b$

Befehle der Registermaschine (RAM)

Befehl	Wirkung
LOAD i	$c(0) := c(i); \quad b := b+1$
STORE i	$c(i) := c(0); \quad b := b+1$
ADD i	$c(0) := c(0) + c(i); \quad b := b+1$
SUB i	$c(0) := \max\{0, c(0) - c(i)\}; \quad b := b+1$
MULT i	$c(0) := c(0) \cdot c(i); \quad b := b+1$
DIV i	$c(0) := \left\lfloor \frac{c(0)}{c(i)} \right\rfloor; \quad b := b+1$
GOTO j	$b := j$
IF $c(0) \# \ell$ GOTO j	$\begin{cases} b := j & \text{falls } c(0) \# \ell \\ b := b+1 & \text{sonst} \end{cases}$ wobei $\# \in \{\leq, \geq, <, >, \neq, =\}$
END	$b := b$

Befehle können modifiziert werden zu:

CLOAD, CSTORE, CADD, CSUB, CMULT, CDIV

ersetze hierzu immer c(i) durch die Konstante i

Befehle der Registermaschine (RAM)

Befehl	Wirkung
LOAD i	$c(0) := c(i); \quad b := b+1$
STORE i	$c(i) := c(0); \quad b := b+1$
ADD i	$c(0) := c(0) + c(i); \quad b := b+1$
SUB i	$c(0) := \max\{0, c(0) - c(i)\}; \quad b := b+1$
MULT i	$c(0) := c(0) \cdot c(i); \quad b := b+1$
DIV i	$c(0) := \left\lfloor \frac{c(0)}{c(i)} \right\rfloor; \quad b := b+1$
GOTO j	$b := j$
IF $c(0) \# \ell$ GOTO j	$\begin{cases} b := j & \text{falls } c(0) \# \ell \\ b := b+1 & \text{sonst} \end{cases}$ wobei $\# \in \{\leq, \geq, <, >, \neq, =\}$
END	$b := b$

Befehle können modifiziert werden zu:

INDLOAD, INDSTORE, INDADD, INDSUB, INDMULT, INDDIV

ersetze hierzu immer c(i) durch $c(c(i))$ (indirekte Addressierung)

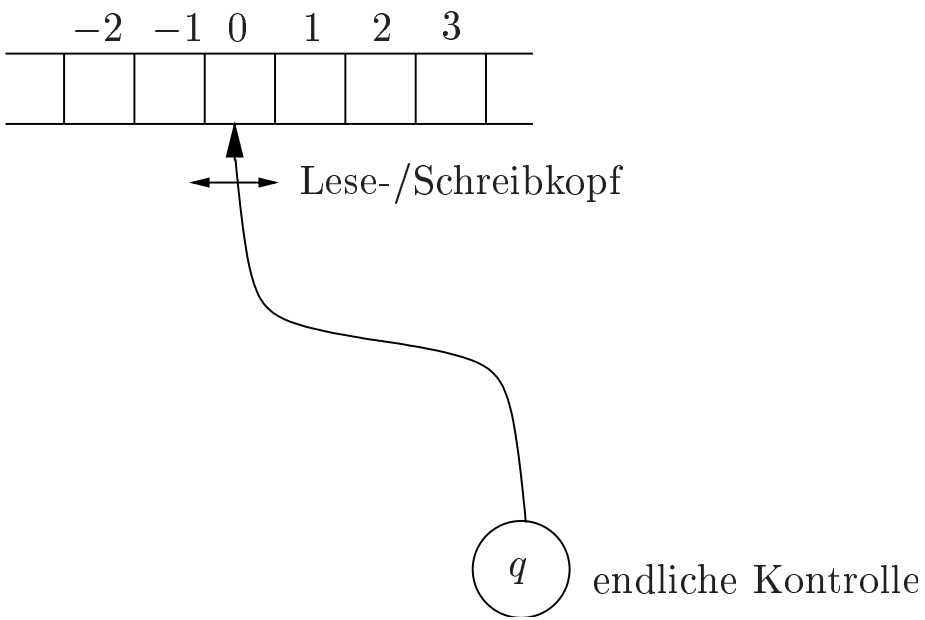
Kostenmodell der Registermaschine (RAM)

- Üblicherweise wird das **uniforme** Kostenmodell verwendet.
- Dabei kostet jede Programmzeile bis auf END eine Einheit.
- Dieses Modell ist gerechtfertigt solange keine großen Zahlen auftreten.
- Ansonsten ist das **logarithmische** Kostenmodell realistischer.
- Kosten entsprechen dann der Länge der benutzten Zahlen.

Die Turingmaschine (TM)

Eine TM besteht aus

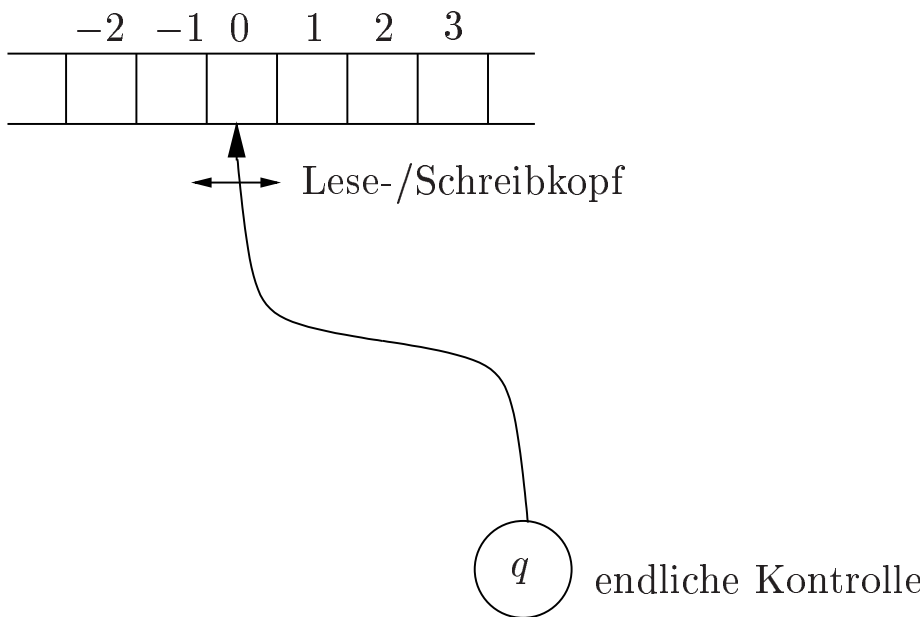
- beidseitig unendlichen Eingabe- und Rechenband,
- freibeweglichem Lese-/Schreibkopf, und
- endlicher Kontrolle.



Die Turingmaschine (TM)

Die Kontrolle

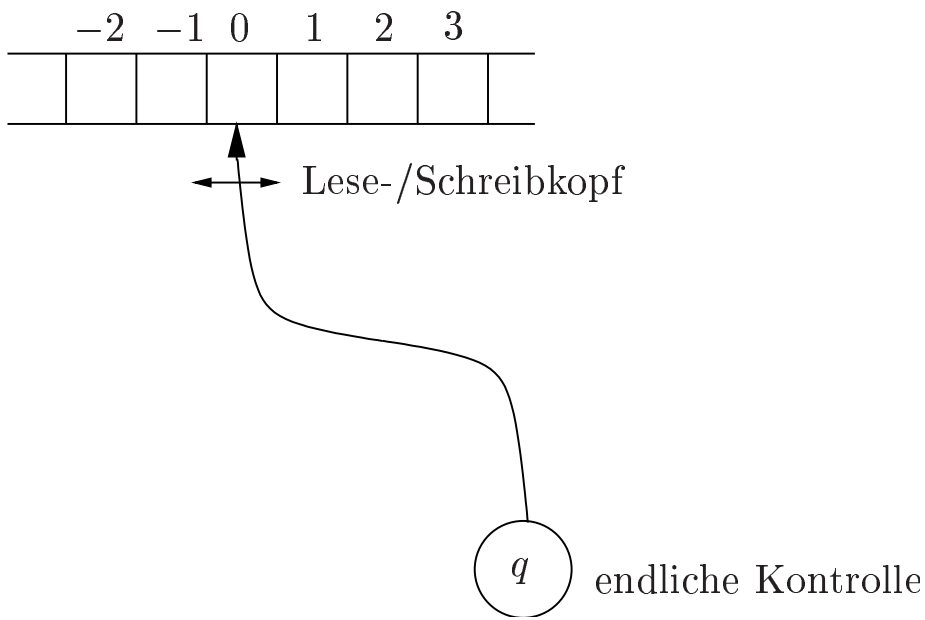
- ist immer in einem von endlich vielen Zuständen, und
- entspricht dem Befehlszähler der RAM.



Die Turingmaschine (TM)

Das Eingabe- und Rechenband

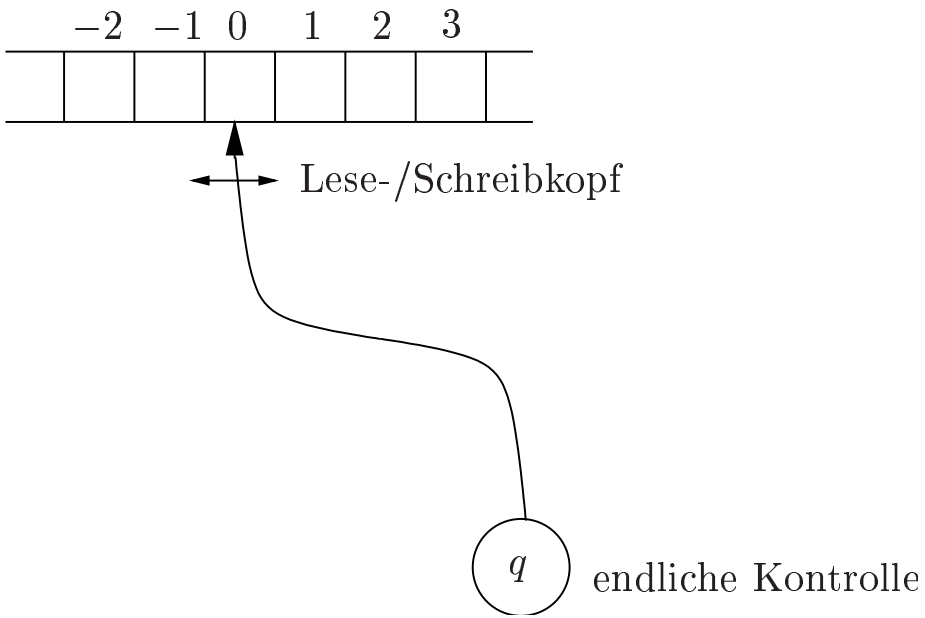
- enthält eine Folge von Symbolen (höchstens eins pro Zelle), und
- entspricht den Registern der RAM.



Die Turingmaschine (TM)

Ausgehend vom aktuellen Zustand verhält sich die TM wie folgt:

- lese das Symbol auf der aktuellen Position des Lese-/ Schreibkopfes,
- gehe in einen Folgezustand über,
- überschreibe evtl. das Symbol, und
- bewege den Lese-/ Schreibkopf nach rechts, links oder gar nicht.



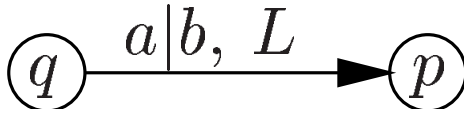
Formale Definition der Turingmaschine

Eine deterministische Turing-Maschine ((D)TM) besteht aus:

- Q, einer endlicher Zustandsmenge,
- Σ, einem endlichen Eingabealphabet,
- $\sqcup$, einem Blanksymbol mit $\sqcup \notin \Sigma$,
- Γ, einem endlichen Bandalphabet mit $\Sigma \cup \{\sqcup\} \subseteq \Gamma$,
- $s \in Q$, einem Startzustand,
- $\delta\colon Q \times \Gamma \to Q \times \Gamma \times \{L, R, N\}$, einer Übergangsfunktion.
  Dabei bedeutet L eine Bewegung des Lese-/Schreibkopfes nach links, R eine Bewegung nach rechts und N ein Stehenbleiben. Die Übergangsfunktion beschreibt, wie das aktuell eingelesene Zeichen verarbeitet werden soll.
- $F \subseteq Q$, einer Menge von Endzuständen.
  Die Menge der Endzustände kann auch entfallen.

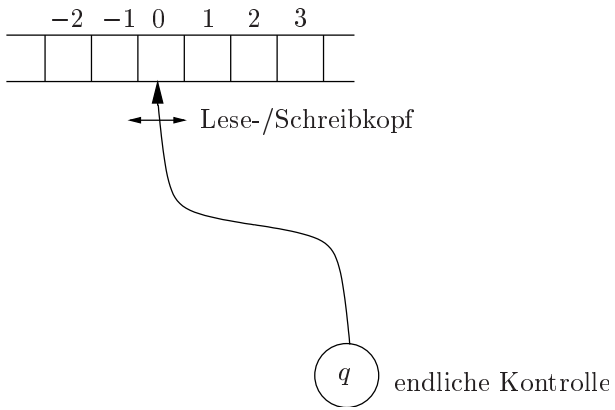
Bemerkungen zur TM

- Der Übergang $\delta(q, a) = (p, b, L)$ wird graphisch wie folgt dargestellt



Bedeutung:
Ist die Turing-Maschine im Zustand *q* und liest das Symbol *a*, so

- überschreibt sie dieses *a* mit *b*,
- geht auf dem Band eine Stelle nach links, und
- wechselt in den Zustand *p*.

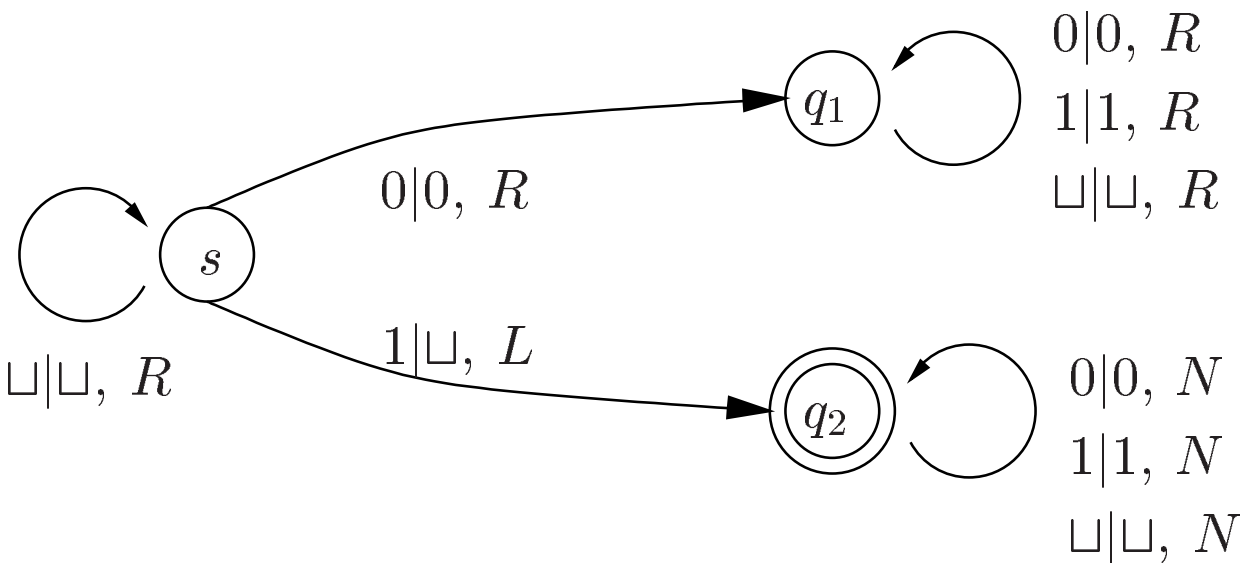


Bemerkungen zur TM

Konventionen

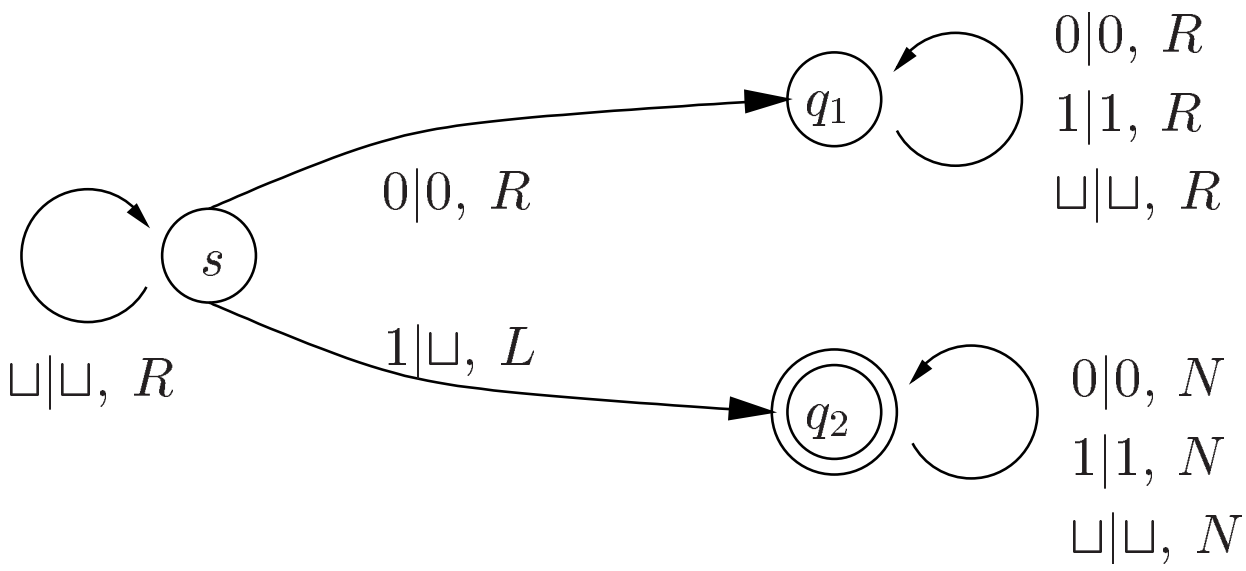
- Die Turing-Maschine startet im Zustand s.
- Der Lese-/Schreibkopf startet an der linkesten Stelle des Bandes, in der ein Eingabesymbol steht.
- Die Turing-Maschine stoppt, wenn sie
  - zum ersten Mal in einen Endzustand kommt, oder
  - in einem Zustand q ein Symbol a liest und $\delta(q, a) = (q, a, N)$ ist.
- Das bedeutet, dass Übergänge, die aus Endzuständen herausführen, sinnlos sind.

Beispiel-Turingmaschine



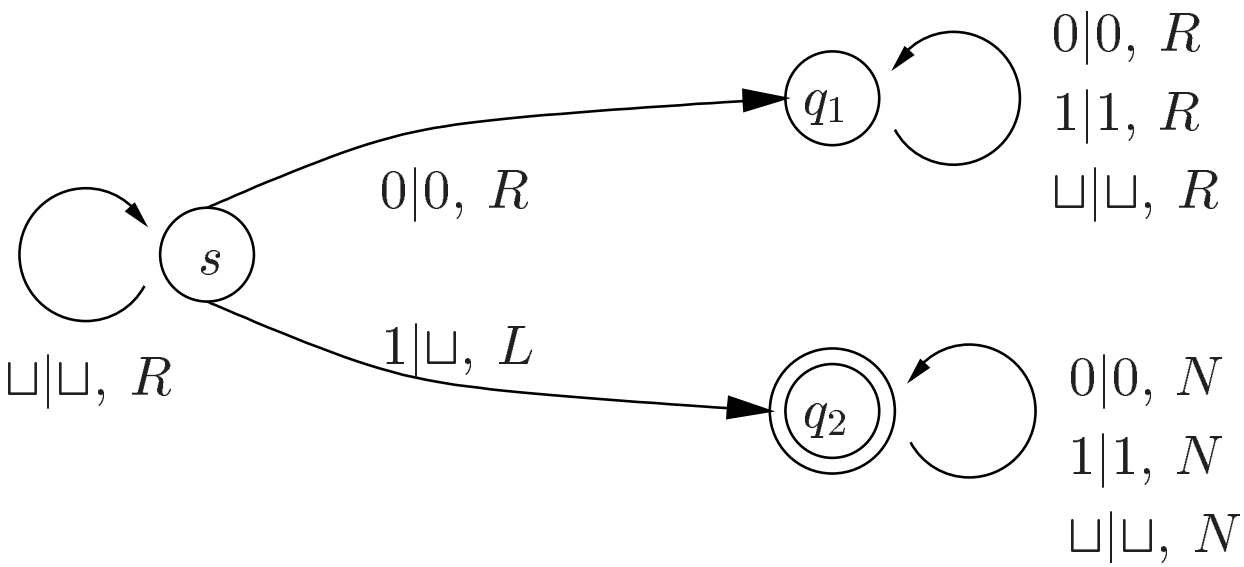
Frage: Was erkennt / berechnet diese TM ?

Beispiel-Turingmaschine



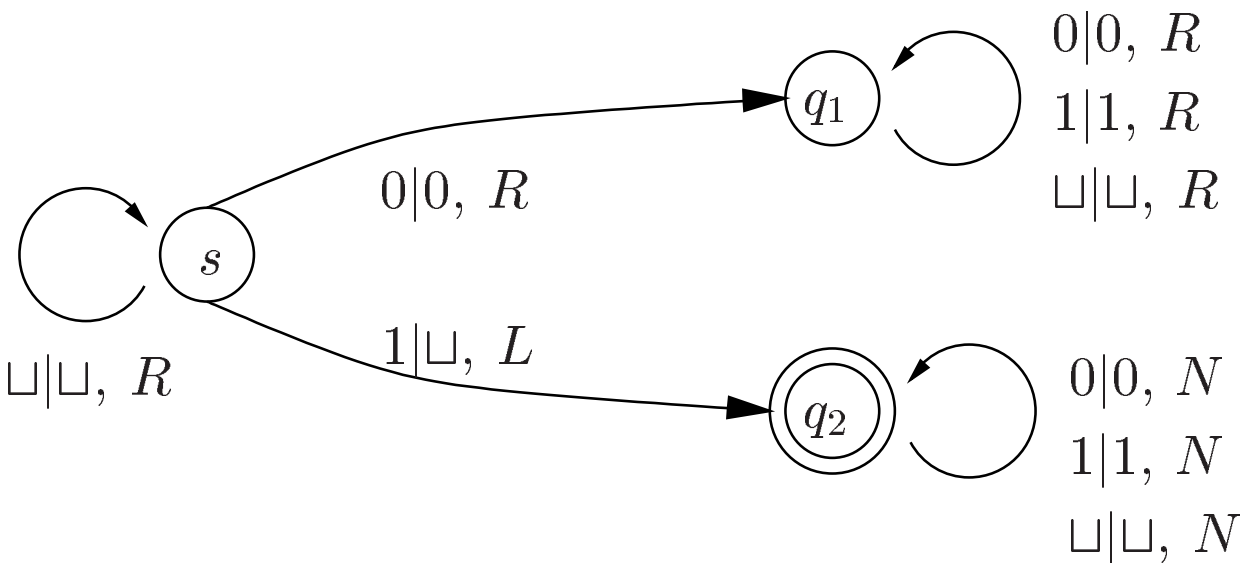
- Die TM erkennt alle Wörter aus $\{0,1\}^*$, die mit einer Eins beginnen.
- Die TM löscht die die führende Eins, falls vorhanden.
- Alles andere auf dem Band bleibt unverändert.
- Der Lese-/Schreibkopf steht nach dem Stop links neben der Stelle an der die führende Eins gelesen wurde.
- Der Zustand q_1 ist unwesentlich.

Beispiel-Turingmaschine



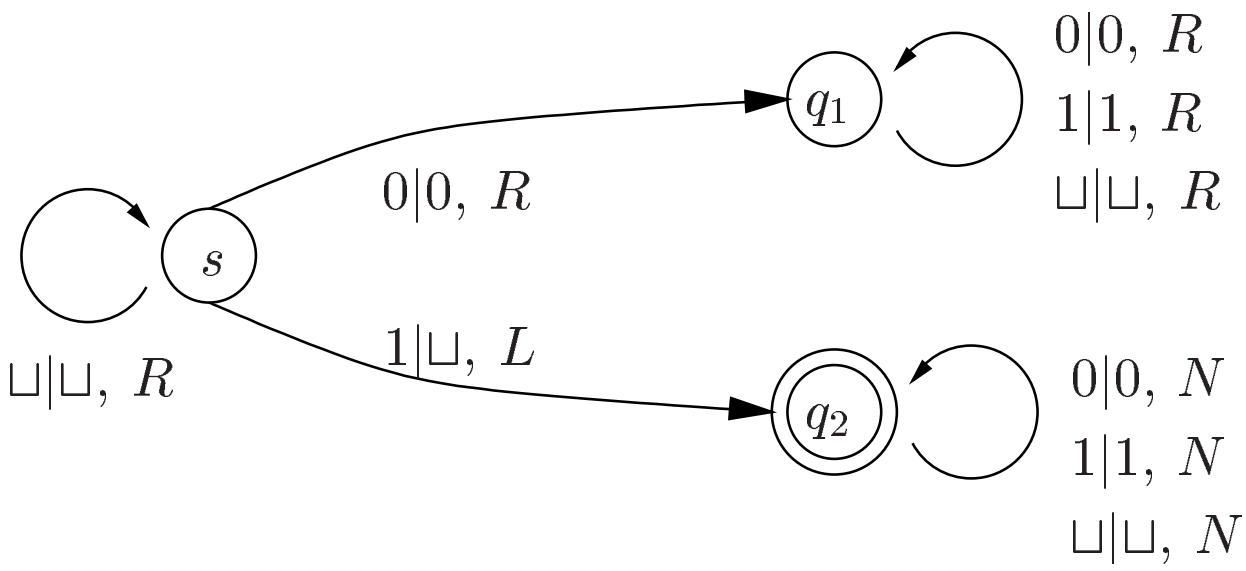
- Es gibt Eingaben, für die eine Turing-Maschine unter Umständen niemals stoppt.
- **Welche Eingaben sind dies in diesem Beispiel?**

Beispiel-Turingmaschine



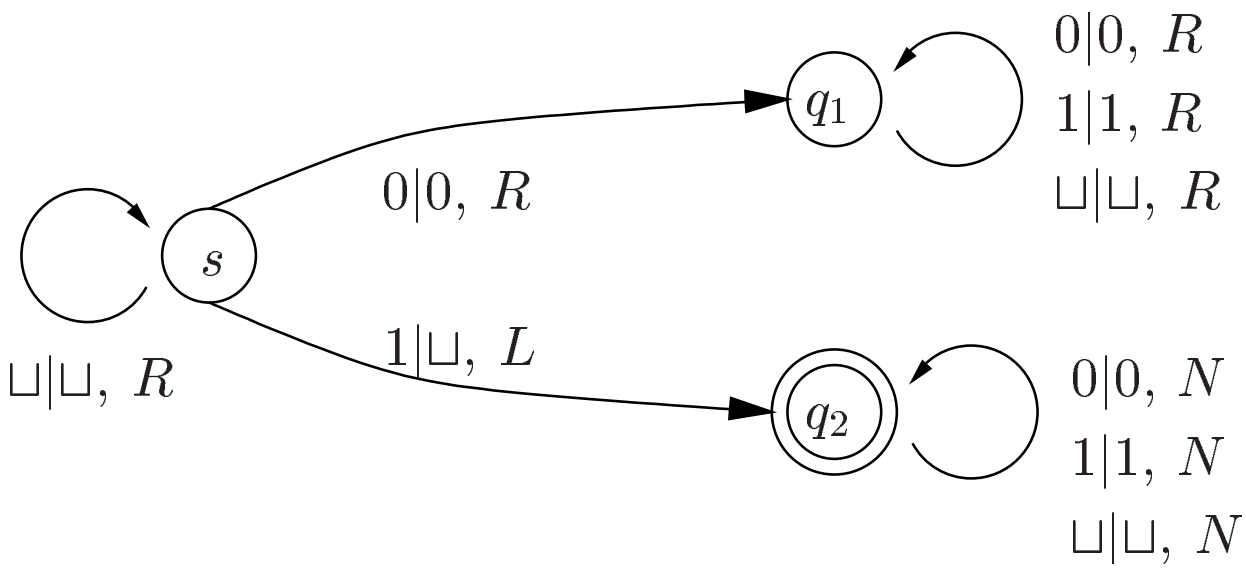
- Die TM stoppt nicht, falls die Eingaben nicht mit Eins beginnt.

Beispiel-Turingmaschine



- Eine Turing-Maschine erkennt nicht nur eine Sprache,
- sondern sie verändert auch die Eingabe, und
- hat insofern auch eine Ausgabe
  (= Inhalt des Bandes nach der Bearbeitung).
- Die Turing-Maschine realisiert also eine partielle Funktion $f\colon \Sigma^* \to \Gamma^*$.

Beispiel-Turingmaschine



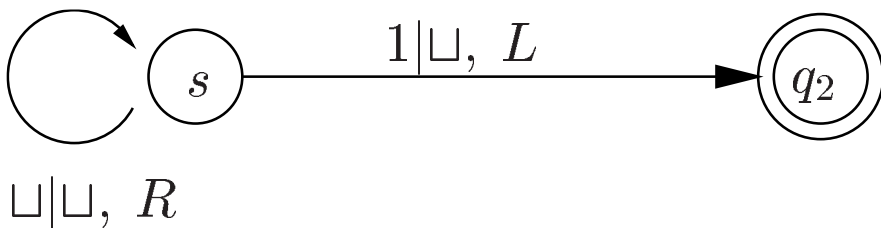
- Die Turing-Maschine realisiert also eine partielle Funktion $f\colon \Sigma^* \to \Gamma^*$.
- Im Beispiel ist

$$f(w) = \begin{cases} v & \text{falls } w = 1v \\ \text{undefiniert} & \text{sonst} \end{cases}$$

Bemerkungen zur TM

- Oft werden wir die Turing-Maschine beziehungsweise deren Übergangsfunktion nur unvollständig beschreiben.

- Beispiel:



- Eine Vervollständigung ist immer möglich.

- Wenn für eine bestimmte Kombination q, a kein Übergang $\delta(q, a)$ definiert ist, dann stoppt die Turing-Maschine im Zustand q.

Definitionen zur TM

- Eine Turing-Maschine **akzeptiert** eine Eingabe $w \in \Sigma^*$, wenn sie nach Lesen von w in einem Zustand aus F stoppt.
- Sie **akzeptiert** eine Sprache L genau dann, wenn sie ausschließlich Wörter aus $w \in L$ als Eingabe akzeptiert.
- Eine Sprache $L \subseteq \Sigma^*$ heißt **rekursiv** oder **entscheidbar**, wenn es eine Turing-Maschine gibt, die auf allen Eingaben stoppt und eine Eingabe w genau dann akzeptiert, wenn $w \in L$ gilt.
- Eine Sprache $L \subseteq \Sigma^*$ heißt **rekursiv-aufzählbar** oder **semi-entscheidbar**, wenn es eine Turing-Maschine gibt, die genau die Eingaben w akzeptiert für die $w \in L$.

  Das Verhalten der Turing-Maschine für Eingaben $w \notin L$ ist damit nicht definiert. D.h., die Turing-Maschine stoppt entweder nicht in einem Endzustand oder aber stoppt gar nicht.

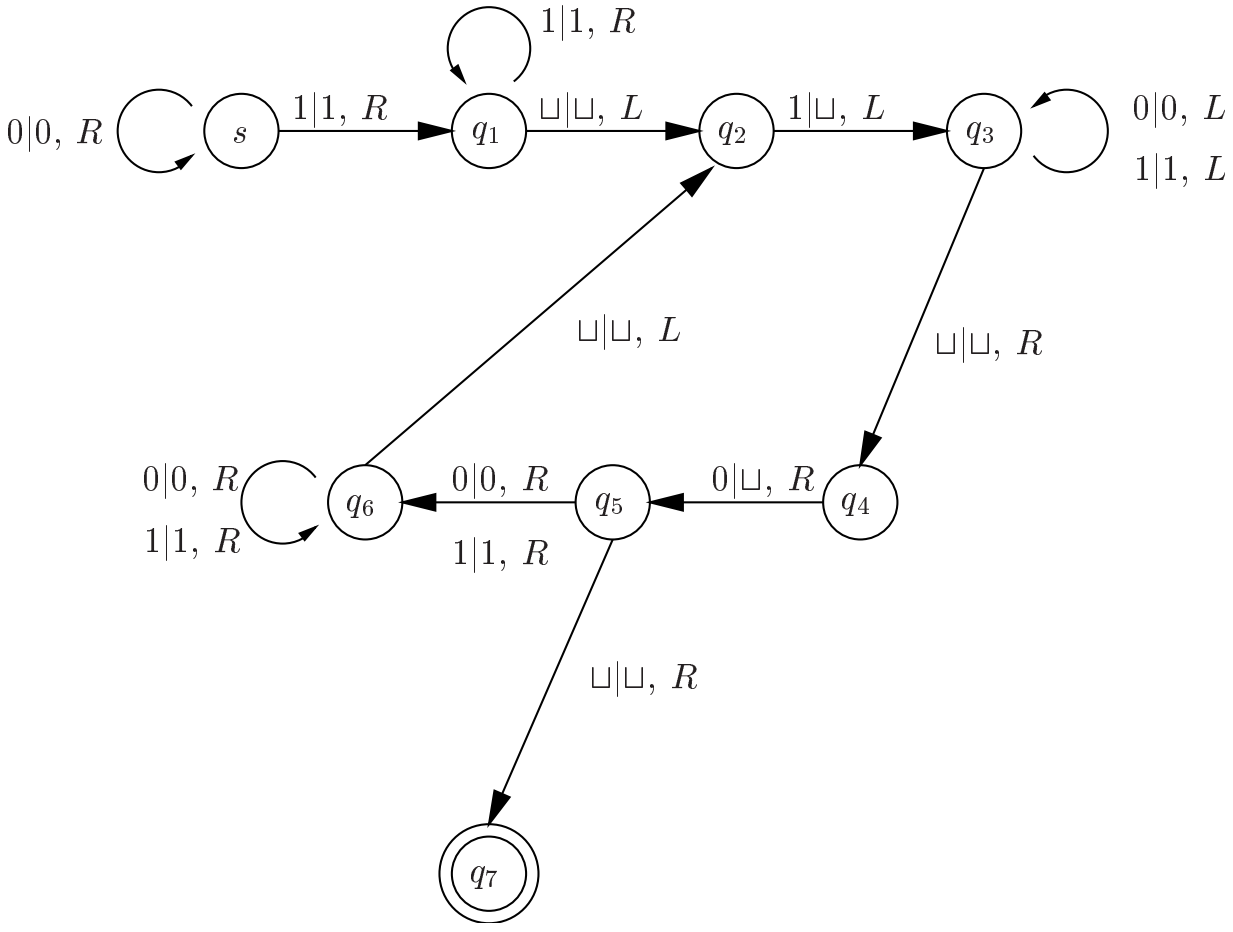
Notation: Konfiguration

- Situation in der sich eine TM $\mathcal{M} := (Q, \Sigma, \Gamma, \delta, s, F)$ befindet wird durch die Angabe der **Konfiguration** codiert.

- Eine Konfiguration hat die Form $w(q)av$, wobei
  - $w, v \in \Gamma^*$
  - $a \in \Gamma$
  - $q \in Q$

- Bedeutung:
  - $\mathcal{M}$ befindet sich gerade im Zustand q.
  - Der Lesekopf steht auf dem Zeichen a.
  - Links vom Lesekopf steht das Wort w auf dem Rechenband.
  - Rechts vom Lesekopf steht das Wort v auf dem Rechenband.

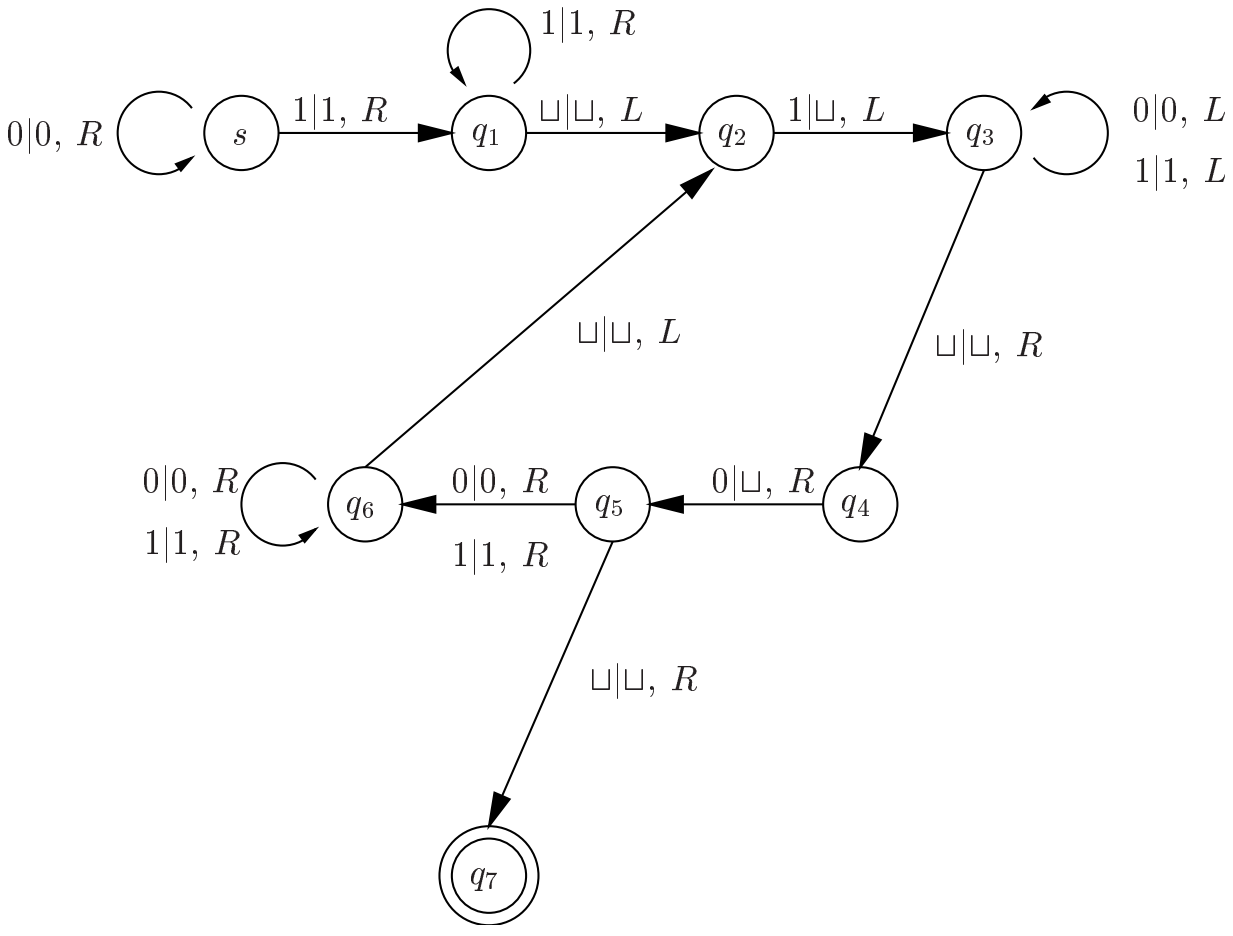
Beispiel: Konfiguration



TM akzeptiert $\{0^n1^n : n \geq 1\}$.

Eingabe: 0011

Beispiel: Konfiguration



$\sqcup(s)0011$
$0(s)011$
$00(s)11$
$001(q_1)1$
$0011(q_1)\sqcup$
$001(q_2)1$
$00(q_3)1$
$0(q_3)01$
$\sqcup(q_3)001$
$\sqcup(q_3)\sqcup001$
$\sqcup(q_4)001$
$\sqcup(q_5)01$
$0(q_6)1$
$01(q_6)\sqcup$
$0(q_2)1$
$\sqcup(q_3)0$
$\sqcup(q_3)\sqcup0$
$\sqcup(q_4)0$
$\sqcup(q_5)\sqcup$
$\sqcup(q_7)\sqcup$

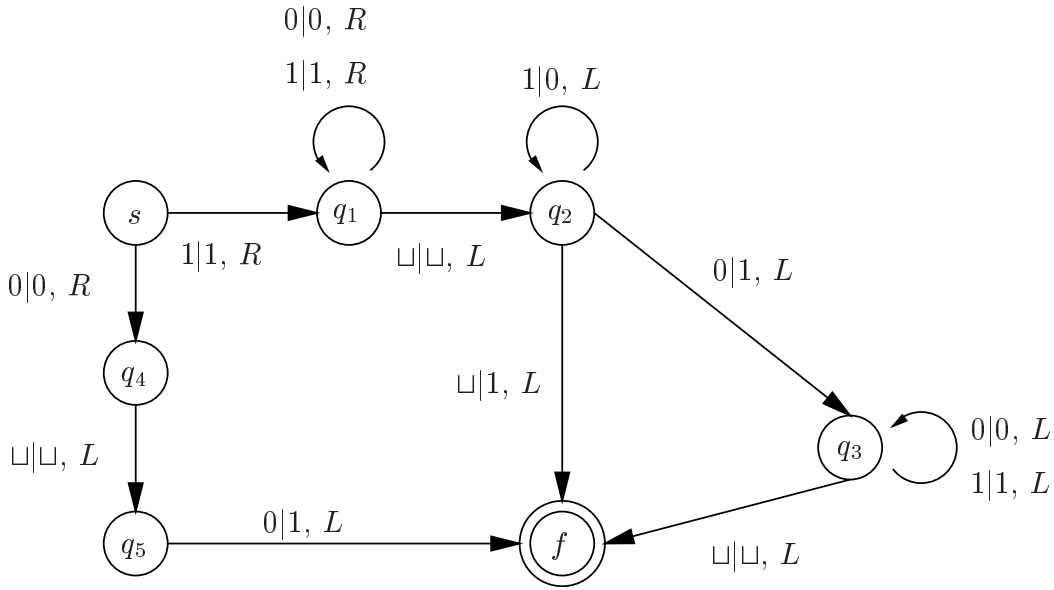
Definition: berechenbar / totalrekursiv

- Eine Funktion $f\colon \Sigma^* \to \Gamma^*$ heißt **(Turing-)berechenbar** oder **totalrekursiv**, wenn es eine Turing-Maschine gibt, die bei Eingabe von $w \in \Sigma^*$ den Funktionswert $f(w) \in \Gamma^*$ ausgibt.

- Eine Turing-Maschine **realisiert** eine Funktion $f\colon \Sigma^* \to \Gamma^*$, falls gilt:

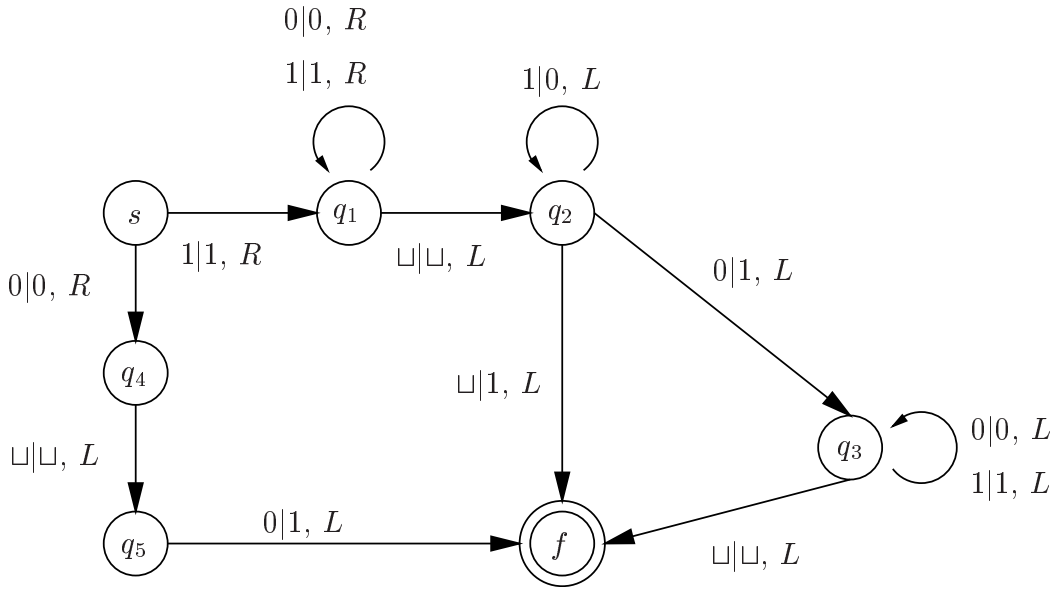
$$f(w) = \begin{cases} \text{Ausgabe der Turing-Maschine, wenn sie bei Eingabe } w \text{ stoppt} \\ \text{undefiniert sonst} \end{cases}$$

Beispiel



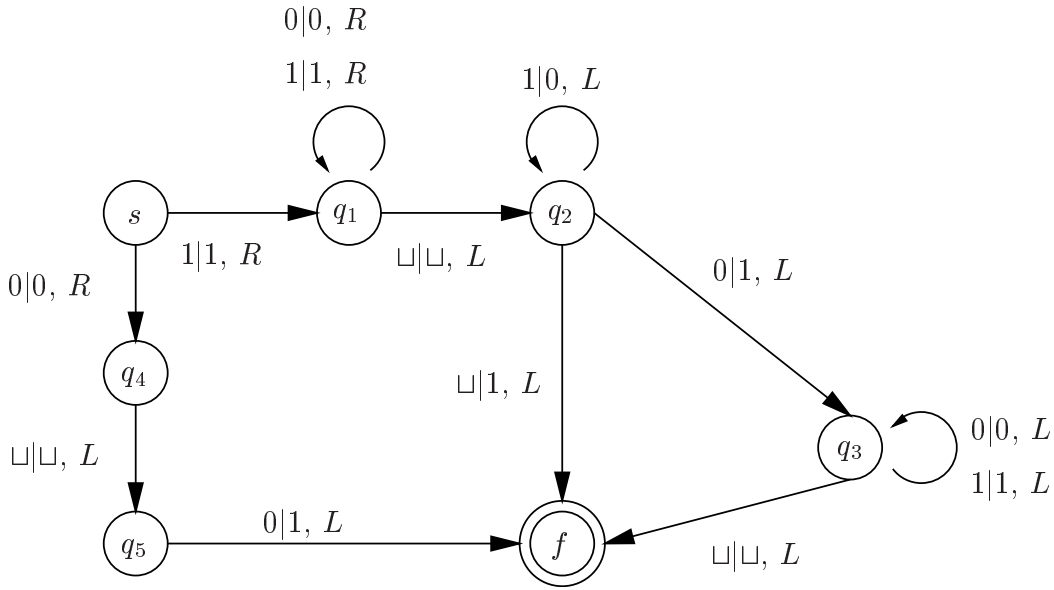
- Fasse die Eingabe *w* als binäre Zahl auf.
- Es sollen nur Eingaben ohne führende Nullen und die Null selbst akzeptiert werden.
- Addiere zur Eingabe $w \in (0 \cup 1)^*$ eine Eins.

Beispiel



Es gilt: $f(w) = \begin{cases} w+1 & \text{falls } w \in 0 \cup 1(0 \cup 1)^*, \\ & \quad w \text{ interpretiert als Binärzahl} \\ \text{undefiniert} & \text{sonst} \end{cases}$

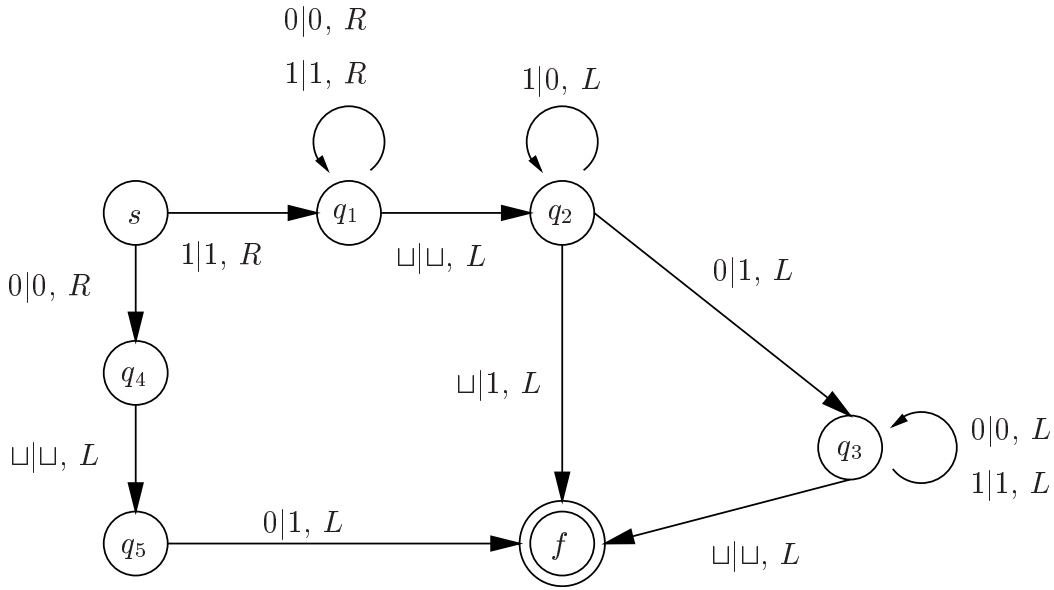
Beispiel



Dabei sind die Zustände jeweils für die folgenden Aufgaben verantwortlich:

- q_1 Bewegung des Lese-/Schreibkopfes nach rechts bis zum Eingabeende,
- q_2 Bildung des Übertrages, der durch die Addition von Eins zu einer bereits vorhandenen Eins entsteht,

Beispiel



Dabei sind die Zustände jeweils für die folgenden Aufgaben verantwortlich:

- q_3 Bewegung des Lese-/Schreibkopfes nach links, nachdem die Aufsummierung abgeschlossen ist (kein Übertrag mehr),
- q_4, q_5 Sonderbehandlung für den Fall der Eingabe 0, und
- f Endzustand.

Entscheidbarkeit und Berechenbarkeit

Entscheidbarkeit von Sprachen und Berechenbarkeit von Funktionen sind verwandt:

- Eine Turing-Maschine akzeptiert eine Sprache L, wenn sie genau auf den Eingaben $w \in L$ in einem ausgezeichneten Endzustand stoppt.
- L ist entscheidbar, wenn es eine Turing-Maschine gibt, die auf allen Eingaben stoppt und L akzeptiert.
- Die Funktion f heißt berechenbar, wenn eine Turing-Maschine existiert, die f realisiert.

Entscheidbarkeit und Berechenbarkeit

Entscheidbarkeit von Sprachen und Berechenbarkeit von Funktionen sind verwandt:

- Man kann eine Turing-Maschine $\mathcal{M}$, die auf allen Eingaben stoppt, so modifizieren, dass es zwei ausgezeichnete Zustände q_J und q_N gibt und dass $\mathcal{M}$ stets in einem der Zustände q_J oder q_N hält.
- Dabei stoppt sie bei der Eingabe w genau dann in q_J, wenn sie w akzeptiert, ansonsten in q_N.
- Damit ist die Sprache L genau dann entscheidbar, wenn es eine Turing-Maschine gibt, die immer in einem der Zustände $\{q_J, q_N\}$ stoppt, wobei sie gerade für $w \in L$ in q_J hält.

Korollar

- Eine Sprache $L \subseteq \Sigma^*$ ist **entscheidbar** genau dann, wenn ihre **charakteristische Funktion** χ_L berechenbar ist, wobei gilt:

$$\chi_L : \Sigma^* \to \{0,1\} \quad \text{mit} \qquad \chi_L(w) = \begin{cases} 1 & \text{falls } w \in L \\ 0 & \text{sonst} \end{cases}$$

- Eine Sprache L ist **semi-entscheidbar** genau dann, wenn die Funktion χ_L^* berechenbar ist, wobei gilt:

$$\chi_L^*(w) = \begin{cases} 1 & \text{falls } w \in L \\ \text{undefiniert} & \text{sonst} \end{cases}$$

Die Church'sche These

Church'sche These
Die Menge der (Turing-)berechenbaren Funktionen ist genau die Menge der im intuitiven Sinne überhaupt berechenbaren Funktionen.

Die Church'sche These

Church'sche These
Die Menge der (Turing-)berechenbaren Funktionen ist genau die Menge der im intuitiven Sinne überhaupt berechenbaren Funktionen.

Interpretation

- Turing-Maschinen sind formale Modelle für Algorithmen.
- Kein Berechnungsverfahren kann algorithmisch genannt werden, wenn es nicht von einer Turing-Maschine ausführbar ist.

Bemerkung

- Die Church'sche These ist nur eine These, kann also nicht bewiesen werden.
- Sie ist aber in der Informatik allgemein akzeptiert.

Die Church'sche These

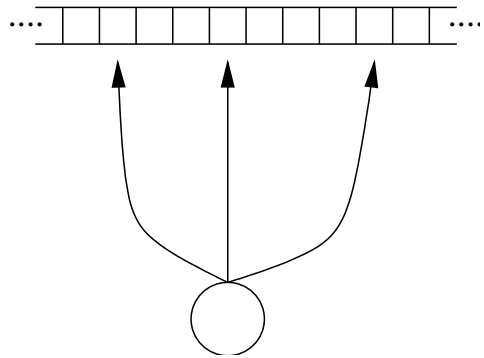
Church'sche These
Die Menge der (Turing-)berechenbaren Funktionen ist genau die Menge der im intuitiven Sinne überhaupt berechenbaren Funktionen.

Begründung

- Es existieren keine Beispiele von Funktionen, die als intuitiv berechenbar angesehen werden, aber nicht Turing-berechenbar sind.
- Alle Versuche, realistische Modelle aufzustellen, die mächtiger sind als Turing-Maschinen, schlugen fehl.
- Eine Reihe von völlig andersartigen Ansätzen, den Begriff der Berechenbarkeit formal zu fassen, wie zum Beispiel die Registermaschine, haben sich als äquivalent erwiesen.

Erweiterungen der Turing-Maschine

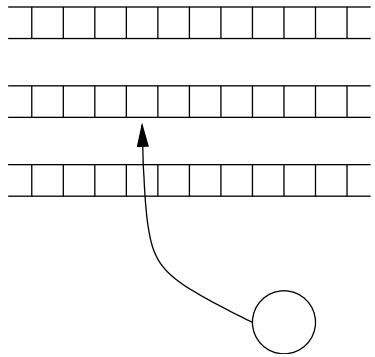
Mehrere Lese-/Schreibköpfe



- Mehrere Lese-/Schreibköpfe $(n \in \mathbb{N})$ greifen auf das eine Eingabeband zu und werden von der endlichen Kontrolle gesteuert.
- Die Übergangsfunktion ist dann vom Typ $\delta\colon Q \times \Gamma^n \to Q \times \Gamma^n \times \{L, N, R\}^n$.
- Die Zustände $q \in Q$ kann man als n-Tupel auffassen.
- Es ist nötig eine Prioritätenregel für die einzelnen Köpfe anzugeben, falls mehrere auf einem Feld des Eingabebandes stehen.

Erweiterungen der Turing-Maschine

Mehrere Bänder



- Ein Lese-/Schreibkopf kann auf mehrere Eingabebänder ($n \in \mathbb{N}$) zugreifen.
- Die Übergangsfunktion ist dann vom Typ

$$\delta\colon Q \times \Gamma \times \{1, \ldots, n\} \to Q \times \Gamma \times \{L, N, R\} \times \{1, \ldots, n\}.$$

Erweiterungen der Turing-Maschine

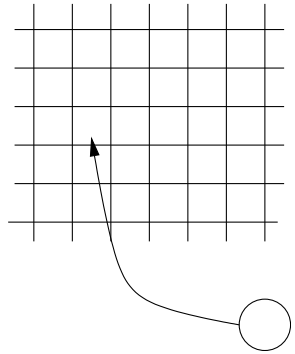
Mehrere Lese-/Schreibköpfe für mehrere Bänder

- Wir haben jetzt m Bänder und n Lese-/Schreibköpfe.
- Die Übergangsfunktion ist dann vom Typ

$$\delta\colon Q \times \Gamma^n \times \{1,\ldots,m\}^n \to Q \times \Gamma^n \times \{L,N,R\}^n \times \{1,\ldots,m\}^n.$$

Erweiterungen der Turing-Maschine

Mehrdimensionale Bänder



- Das Eingabeband ist nun mehrdimensional und hat hier im Beispiel die Dimension zwei.
- Wir sprechen dann von einem Arbeitsfeld.
- Dabei ist

$$\delta\colon Q \times \Gamma \to Q \times \Gamma \times \{L(eft), U(p), R(ight), D(own), N(othing)\}$$

Erweiterungen der Turing-Maschine

Bemerkungen

- Fragestellungen der Art:
  - Wann stoppt eine Mehrkopf-Maschine?
  - Welcher Kopf ‚gewinnt', wenn mehrere Köpfe (verschiedene) Symbole an dieselbe Stelle schreiben wollen?

  müssen bei solchen Modifikationen noch geklärt werden.

- Es hat sich allerdings gezeigt, dass keine dieser Erweiterungen mehr leistet, als eine normale Turing-Maschine.

- **Alle angegebenen Modifikationen können durch eine normale 1–Band Turing-Maschine simuliert werden.**

Die universelle Turing-Maschine

Ziel

- Bisher: Nur Turing-Maschinen, die eine bestimmte Aufgabe erfüllen.
- Jetzt: Konstruktion einer Turing-Maschine, die als Eingabe
  - ein Programm und
  - eine spezielle Eingabe

  erhält.
- Die Aufgabe besteht darin, das gegebene Programm auf der gegebenen speziellen Eingabe auszuführen.

Die universelle Turing-Maschine

Wir betrachten dazu eine normierte Turing-Maschine, d.h.

- $Q := \{q_1, \ldots, q_n\}$
- $\Sigma := \{a_1, \ldots, a_k\}$
- $\Gamma := \{a_1, \ldots, a_k, a_{k+1}, \ldots, a_l\}$
- $s := q_1$
- $F := \{q_2\}$

- Dies bedeutet keine Einschränkung in der Mächtigkeit der Turing-Maschinen:
  - Jede beliebige Turing-Maschine kann durch eine derart normiert Turing-Maschine der obigen Form simuliert werden.
  - Jede normierte Turing-Maschine $\mathcal{M}$ lässt sich eindeutig als Wort aus $(0 \cup 1)^*$ kodieren.

Die Gödelnummer

Sei $\mathcal{M} := (Q, \Sigma, \Gamma, \delta, s, F)$ eine Turing-Maschine.

Die Gödelnummer von $\mathcal{M}$, bezeichnet als $\langle \mathcal{M} \rangle$, ist definiert durch folgende Kodierungsvorschrift:

- Kodiere

$$\delta(q_i, a_j) = (q_r, a_s, d_t) \text{ durch } 0^i\,1\,0^j\,1\,0^r\,1\,0^s\,1\,0^t,$$

  wobei $d_t \in \{d_1, d_2, d_3\}$ und
  - d_1 für L,
  - d_2 für R und
  - d_3 für N steht.
- Die Turing-Maschine wird dann kodiert durch:

$$111\text{code}_1 11\text{code}_2 11 \ldots 11\text{code}_z 111,$$

  wobei code_i für $i = 1, \ldots, z$ alle Funktionswerte von δ in beliebiger Reihenfolge beschreibt.

Die Gödelnummer - Bemerkungen

- Die eigentlichen Werte der Turing-Maschine werden also (unär) durch Nullen beschrieben und die Einsen dienen als Begrenzung der Eingabewerte.
- Jede Turing-Maschine kann also durch ein Wort aus $(0 \cup 1)^*$ kodiert werden.
- Umgekehrt beschreibt jedes Wort aus $(0 \cup 1)^*$ (höchstens) eine Turing-Maschine.
- Wir vereinbaren, dass ein Wort, das keine Turing-Maschine in diesem Sinne beschreibt, (zum Beispiel ε, 0, 000) eine Turing-Maschine kodiert, die $\emptyset$ akzeptiert.
- Eine *universelle Turing-Maschine* erhält als Eingabe ein Paar $(\langle \mathcal{M} \rangle, w)$, wobei $w \in \{0,1\}^*$ ist, und sie simuliert $\mathcal{M}$ auf w.

Die Gödelnummer - Beispiel

Sei $\mathcal{M} = (\{q_1, q_2, q_3\}, \{0, 1\}, \sqcup, \{0, 1, \sqcup\}, \delta, q_1, \{q_2\})$, mit

$$\delta(q_1, 1) = (q_3, 0, R)$$
$$\delta(q_3, 0) = (q_1, 1, R)$$
$$\delta(q_3, 1) = (q_2, 0, R)$$
$$\delta(q_3, \sqcup) = (q_3, 1, L)$$

$\mathcal{M}$ zusammen mit der Eingabe 1011 ist dann:

111010010001010011000101010010011000
100100101001100010001000100101111011