Grundbegriffe der Informatik - Tutorium

- Wintersemester 2011/12 -

Christian Jülg

http://gbi-tutor.blogspot.com

25. Januar 2012



Quellennachweis & Dank an: Martin Schadow, Susanne Putze, Tobias Dencker, Sebastian Heßlinger, Joachim Wilke

Übersicht



- Guten Morgen...
- 2 Aufgabenblatt 11
- 3 Aufgabenblatt 12
- Reguläre Ausdrücke
- **5** Rechtslineare Grammatiken
- 6 Kantorowitsch-Bäume, Regex-Bäume
- Berechenbarkeit
- 8 Turingmaschinen
- Abschluss

- Guten Morgen...
- 2 Aufgabenblatt 11
- 3 Aufgabenblatt 12
- 4 Reguläre Ausdrücke
- **5** Rechtslineare Grammatiken
- 6 Kantorowitsch-Bäume, Regex-Bäume
- Berechenbarkeit
- Turingmaschinen
- Abschluss



Der Reguläre Ausdruck ab*c*a ...

- ... beschreibt eine Typ-3 Sprache.
- ② ... beschreibt die Sprache $L_1 = \{ab^nc^na|n \in \mathbb{N}_0\}$
- ... beschreibt eine endliche Sprache.

Rechtslineare Grammatiken ...

- 1 ... beschreiben reguläre Sprachen.
- \odot ... dürfen ϵ -Produktionen enthalten.
- 3 ... lassen sich in einen endlichen Automaten überführen.



Der Reguläre Ausdruck ab*c*a ...

- ... beschreibt eine Typ-3 Sprache.
- ② ... beschreibt die Sprache $L_1 = \{ab^nc^na|n \in \mathbb{N}_0\}$
- ... beschreibt eine endliche Sprache.

Rechtslineare Grammatiken ...

- 1 ... beschreiben reguläre Sprachen.
- 2 ... dürfen ϵ -Produktionen enthalten.
- 3 ... lassen sich in einen endlichen Automaten überführen.



Der Reguläre Ausdruck ab*c*a ...

- ... beschreibt eine Typ-3 Sprache.
- ② ... beschreibt die Sprache $L_1 = \{ab^nc^na|n \in \mathbb{N}_0\}$
- ... beschreibt eine endliche Sprache.

Rechtslineare Grammatiken ...

- 1 ... beschreiben reguläre Sprachen.
- \mathbf{Q} ... dürfen ϵ -Produktionen enthalten.
- 3 ... lassen sich in einen endlichen Automaten überführen.



Für die Typen-Hierachie für Grammatiken gilt, ...

- 1 ... Typ-3 ist die mächtigste Klasse.
- 2 ... alle Grammatiken lassen sich als regulärer Ausdruck angeben
- \bullet ... wenn die Grammatik G Typ-2 ist, so ist L(G) auch Typ-2.

Ein Akzeptor, bei dem der Startzustand auch akzeptierender Zustand ist ...

- **1** ... akzeptiert alle Worte $w \in A$.
- $oldsymbol{0}$... akzeptiert immer auch das leere Wort ϵ .
- **3** ... akzeptiert alle Worte $w \in A^*$.



Für die Typen-Hierachie für Grammatiken gilt, ...

- 1 ... Typ-3 ist die mächtigste Klasse.
- 2 ... alle Grammatiken lassen sich als regulärer Ausdruck angeben
- \odot ... wenn die Grammatik G Typ-2 ist, so ist L(G) auch Typ-2.

Ein Akzeptor, bei dem der Startzustand auch akzeptierender Zustand ist ...

- **1** ... akzeptiert alle Worte $w \in A$.
- $oldsymbol{0}$... akzeptiert immer auch das leere Wort ϵ .
- **3** ... akzeptiert alle Worte $w \in A^*$.



Für die Typen-Hierachie für Grammatiken gilt, ...

- 1 ... Typ-3 ist die mächtigste Klasse.
- 2 ... alle Grammatiken lassen sich als regulärer Ausdruck angeben
- \bullet ... wenn die Grammatik G Typ-2 ist, so ist L(G) auch Typ-2.

Ein Akzeptor, bei dem der Startzustand auch akzeptierender Zustand ist ...

- **1** ... akzeptiert alle Worte $w \in A$.
- $oldsymbol{0}$... akzeptiert immer auch das leere Wort ϵ .
- **3** ... akzeptiert alle Worte $w \in A^*$.

- 1 Guten Morgen...
- 2 Aufgabenblatt 11
- 3 Aufgabenblatt 12
- 4 Reguläre Ausdrücke
- **(5)** Rechtslineare Grammatiken
- 6 Kantorowitsch-Bäume, Regex-Bäume
- Berechenbarkeit
- Turingmaschinen
- Abschluss



Blatt 11

• Abgaben: 17 / 24

• Punkte: 11,4/20

Probleme

bei jedem Automaten Startzustand angeben



Blatt 11

• Abgaben: 17 / 24

• Punkte: 11,4/20

Probleme

- bei jedem Automaten Startzustand angeben
- zu jedem Akzeptor die akzeptierenden Zustände angeben

- 1 Guten Morgen...
- 2 Aufgabenblatt 11
- 3 Aufgabenblatt 12
- 4 Reguläre Ausdrücke
- 6 Rechtslineare Grammatiken
- 6 Kantorowitsch-Bäume, Regex-Bäume
- Berechenbarkeit
- 8 Turingmaschinen
- Abschluss



Blatt 12

 Abgabe: 27.01.2012 um 12:30 Uhr im Untergeschoss des Infobaus

• Punkte: maximal 20

Themen

Strukturelle Induktion



Blatt 12

- Abgabe: 27.01.2012 um 12:30 Uhr im Untergeschoss des Infobaus
- Punkte: maximal 20

- Strukturelle Induktion
- Akzeptoren, reguläre Ausdrücke und rechtslineare Grammatiken



Blatt 12

- Abgabe: 27.01.2012 um 12:30 Uhr im Untergeschoss des Infobaus
- Punkte: maximal 20

- Strukturelle Induktion
- Akzeptoren, reguläre Ausdrücke und rechtslineare Grammatiken
- Turing Maschinen



Blatt 12

- Abgabe: 27.01.2012 um 12:30 Uhr im Untergeschoss des Infobaus
- Punkte: maximal 20

- Strukturelle Induktion
- Akzeptoren, reguläre Ausdrücke und rechtslineare Grammatiken
- Turing Maschinen
- Turing Maschinen bauen



Blatt 12

- Abgabe: 27.01.2012 um 12:30 Uhr im Untergeschoss des Infobaus
- Punkte: maximal 20

- Strukturelle Induktion
- Akzeptoren, reguläre Ausdrücke und rechtslineare Grammatiken
- Turing Maschinen
- Turing Maschinen bauen
- Turing Maschinen verstehen

- 1 Guten Morgen...
- 2 Aufgabenblatt 11
- 3 Aufgabenblatt 12
- 4 Reguläre Ausdrücke
- 5 Rechtslineare Grammatiken
- 6 Kantorowitsch-Bäume, Regex-Bäume
- Berechenbarkeit
- Turingmaschinen
- 9 Abschluss

Definition



Reguläre Ausdrücke sind eine verbreitete und geeignete Notation, um reguläre Sprachen zu formalisieren.

Die Regeln (= Metazeichen)

Metazeichen	Bedeutung
()	Klammerung von Alternativen
*	n-maliges Vorkommen
	trennt Alternativen

Es gelten folgende Vorrangregeln:

- * bindet stärker als Verkettung
- Verkettung (RS) bindet stärker als "oder" (R|S)
- Überflüssige Klammern dürfen wir weglassen. So sind (RS), ((RS)), ... und RS äquivalent

Definition



Die Sprache von R

Wenn **R** ein regulärer Ausdruck ist, dann bezeichnen wir mit $\langle R \rangle$ die Sprache, die dieser erzeugt.

- $\bullet \ \langle \emptyset \rangle = \{\}$
- Für $a \in A$ ist $\langle a \rangle = \{a\}$
- $\bullet \ \langle R_1 | R_2 \rangle = \langle R_1 \rangle \cup \langle R_2 \rangle$
- $\langle R_1 R_2 \rangle = \langle R_1 \rangle \cdot \langle R_2 \rangle$

 R_1 und R_2 sind hier zwei beliebige reguläre Ausdrücke.



Beispiel 1

Welche Wörter erzeugt der folgende reguläre Ausdruck R?

•
$$R = (a|b) * abb(a|b) * ?$$



Beispiel 1

Welche Wörter erzeugt der folgende reguläre Ausdruck R?

- R = (a|b) * abb(a|b) * ?
- \(\begin{align*} R \rangle \) enthält genau die W\(\text{orter}, \) in denen das Teilwort \(abb \)
 vorkommt.

Beispiel 2

Gebe einen regulären Ausdruck für die Sprache aller Wörter die nicht ab enthalten



Beispiel 1

Welche Wörter erzeugt der folgende reguläre Ausdruck R?

- R = (a|b) * abb(a|b) * ?
- \(\begin{align*} R \rangle \) enthält genau die W\(\text{orter}, \) in denen das Teilwort \(abb \)
 vorkommt.

Beispiel 2

Gebe einen regulären Ausdruck für die Sprache aller Wörter die nicht ab enthalten

b*a*



Beispiel 3

Welcher reguläre Ausdruck R erzeugt die Sprache $\{\epsilon\}$?



Beispiel 3

Welcher reguläre Ausdruck R erzeugt die Sprache $\{\epsilon\}$?

• \emptyset *, denn $\langle \emptyset \rangle^* = \{\}^* = \{\epsilon\}$



Beispiel 3

Welcher reguläre Ausdruck R erzeugt die Sprache $\{\epsilon\}$?

• \emptyset *, denn $\langle \emptyset \rangle^* = \{\}^* = \{\epsilon\}$

Beispiel 4

Gebe einen regulären Ausdruck für die Sprache aller Wörter mit mindestens 3 b's an!



Beispiel 3

Welcher reguläre Ausdruck R erzeugt die Sprache $\{\epsilon\}$?

• \emptyset *, denn $\langle \emptyset \rangle^* = \{\}^* = \{\epsilon\}$

Beispiel 4

Gebe einen regulären Ausdruck für die Sprache aller Wörter mit mindestens 3 b's an!

•
$$(a|b) * b(a|b) * b(a|b) * b(a|b) * oder$$



Beispiel 3

Welcher reguläre Ausdruck R erzeugt die Sprache $\{\epsilon\}$?

• \emptyset *, denn $\langle \emptyset \rangle^* = \{\}^* = \{\epsilon\}$

Beispiel 4

Gebe einen regulären Ausdruck für die Sprache aller Wörter mit mindestens 3 b's an!

- (a|b) * b(a|b) * b(a|b) * b(a|b) * oder
- a * ba * ba * b(a|b)*

- 1 Guten Morgen...
- 2 Aufgabenblatt 11
- 3 Aufgabenblatt 12
- 4 Reguläre Ausdrücke
- **5** Rechtslineare Grammatiken
- 6 Kantorowitsch-Bäume, Regex-Bäume
- Berechenbarkeit
- Turingmaschinen
- Abschluss



etwas genauer...

Eine rechtslineare Grammatik ist eine kontextfreie Grammatik G=(N,T,S,P) mit folgenden Einschränkungen. Jede Produktion ist entweder von der Form

- $X \rightarrow w$ oder
- $X \rightarrow wY$ mit $w \in T^*$ und $X, Y \in N$



etwas genauer...

Eine rechtslineare Grammatik ist eine kontextfreie Grammatik G=(N,T,S,P) mit folgenden Einschränkungen. Jede Produktion ist entweder von der Form

- $X \rightarrow w$ oder
- $X \rightarrow wY$ mit $w \in T^*$ und $X, Y \in N$

Regex

Zu jeder rechtslinearen Grammatik gibt es:

• ... einen entsprechenden regulären Ausdruck und

Grundbegriffe der Informatik - Tutorium



etwas genauer...

Eine rechtslineare Grammatik ist eine kontextfreie Grammatik G=(N,T,S,P) mit folgenden Einschränkungen. Jede Produktion ist entweder von der Form

- $X \rightarrow w$ oder
- $X \to wY$ mit $w \in T^*$ und $X, Y \in N$

Regex

Zu jeder rechtslinearen Grammatik gibt es:

- ... einen entsprechenden regulären Ausdruck und
- ... einen einen deterministischen endlichen Automaten

Zu jeder rechtslinearen gibt es äquivalente linkslineare Grammatiken. Diese "können" nichts anderes als rechtslineare Grammatiken, daher ignorieren wir sie in dieser Vorlesung.



Ein Beispiel

Gegeben Sei die Grammatik

$$G = (\{X, Y\}, \{a, b\}, X, \{X \rightarrow aY | \epsilon, Y \rightarrow Xb\})$$

Ist diese Grammatik rechtslinear?



Ein Beispiel - oder auch nicht...

Gegeben Sei die Grammatik

$$G = (\{X, Y\}, \{a, b\}, X, \{X \to aY | \epsilon, Y \to Xb\})$$

- Ist diese Grammatik rechtslinear?
 - G ist offensichtlich nicht rechtslinear, denn die Produktion
 - $Y \rightarrow Xb$ hat das Nichtterminalsymbol links vom

Terminalsymbol

(Die Produktion ist linkslinear)!



Ein Beispiel - oder auch nicht...

Gegeben Sei die Grammatik

$$G = (\{X, Y\}, \{a, b\}, X, \{X \rightarrow aY | \epsilon, Y \rightarrow Xb\})$$

- Ist diese Grammatik rechtslinear?
 G ist offensichtlich nicht rechtslinear, denn die Produktion
 Y → Xb hat das Nichtterminalsymbol links vom
 Terminalsymbol
 (Die Produktion ist linkslinear)!
- ullet Die Grammatik erzeugt die Sprache $L(G)=\{a^kb^k|k\in\mathbb{N}_0\}$

rechtslineare Grammatiken



Ein Beispiel - oder auch nicht...

Gegeben Sei die Grammatik

$$G = (\{X, Y\}, \{a, b\}, X, \{X \rightarrow aY | \epsilon, Y \rightarrow Xb\})$$

- Ist diese Grammatik rechtslinear? G ist offensichtlich nicht rechtslinear, denn die Produktion $Y \to Xb$ hat das Nichtterminalsymbol links vom Terminalsymbol (Die Produktion ist linkslinear)!
- Die Grammatik erzeugt die Sprache $L(G) = \{a^k b^k | k \in \mathbb{N}_0\}$
- Kann es eine rechtslineare Grammatik für diese Sprache geben?
 Ist diese Sprache regulär?

rechtslineare Grammatiken



Ein Beispiel - oder auch nicht...

Gegeben Sei die Grammatik

$$G = (\{X, Y\}, \{a, b\}, X, \{X \to aY | \epsilon, Y \to Xb\})$$

- Ist diese Grammatik rechtslinear?
 G ist offensichtlich nicht rechtslinear, denn die Produktion
 Y → Xb hat das Nichtterminalsymbol links vom
 Terminalsymbol
 (Die Produktion ist linkslinear)!
- Die Grammatik erzeugt die Sprache $L(G) = \{a^k b^k | k \in \mathbb{N}_0\}$
- Kann es eine rechtslineare Grammatik für diese Sprache geben?
 Ist diese Sprache regulär?
 Nein, ist sie nicht!

von G zu L(G)



Aufgabe

Betrachte
$$G = (\{X, Y, Z\}, \{a, b\}, X, P)$$
 mit $P = \{X \rightarrow aX|bY|\epsilon, Y \rightarrow aX|bZ|\epsilon, Z \rightarrow aZ|bZ\}$

- Was ist *L*(*G*)?
- Geben Sie einen endlichen Automaten an, der L(G) akzeptiert.
- Lässt sich diese Grammatik noch vereinfachen?

von G zu L(G)

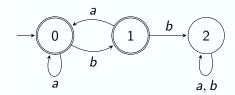


Aufgabe

Betrachte
$$G = (\{X, Y, Z\}, \{a, b\}, X, P)$$
 mit $P = \{X \rightarrow aX|bY|\epsilon, Y \rightarrow aX|bZ|\epsilon, Z \rightarrow aZ|bZ\}$

- Was ist *L*(*G*)?
- Geben Sie einen endlichen Automaten an, der L(G) akzeptiert.
- Lässt sich diese Grammatik noch vereinfachen?

Lösung



Ist doch alles das Gleiche, oder?



Gleiche Sprache - andere Grammatik

Folgende Grammatiken erzeugen die gleiche Sprache

•
$$G = (\{X, Y, Z\}, \{a, b\}, X, P)$$
 mit $P = \{X \rightarrow aX|bY|\epsilon, Y \rightarrow aX|bZ|\epsilon, Z \rightarrow aZ|bZ\}$

Ist doch alles das Gleiche, oder?



Gleiche Sprache - andere Grammatik

Folgende Grammatiken erzeugen die gleiche Sprache

- $G = (\{X, Y, Z\}, \{a, b\}, X, P)$ mit $P = \{X \rightarrow aX|bY|\epsilon, Y \rightarrow aX|bZ|\epsilon, Z \rightarrow aZ|bZ\}$
- $G = (\{X, Y\}, \{a, b\}, X, P) \text{ mit } P = \{X \rightarrow aX | bY | \epsilon, Y \rightarrow aX | \epsilon\}$

Ist doch alles das Gleiche, oder?



Gleiche Sprache - andere Grammatik

Folgende Grammatiken erzeugen die gleiche Sprache

- $G = (\{X, Y, Z\}, \{a, b\}, X, P)$ mit $P = \{X \rightarrow aX|bY|\epsilon, Y \rightarrow aX|bZ|\epsilon, Z \rightarrow aZ|bZ\}$
- $G = (\{X, Y\}, \{a, b\}, X, P) \text{ mit } P = \{X \rightarrow aX | bY | \epsilon, Y \rightarrow aX | \epsilon\}$
- $G = (\{X\}, \{a, b\}, X, P) \text{ mit } P = \{X \rightarrow aX | baX | b | \epsilon\}$

- 1 Guten Morgen...
- 2 Aufgabenblatt 11
- 3 Aufgabenblatt 12
- 4 Reguläre Ausdrücke
- **6** Rechtslineare Grammatiken
- 6 Kantorowitsch-Bäume, Regex-Bäume
- Berechenbarkeit
- 8 Turingmaschinen
- Abschluss



Kantorowitsch

Beispiel:

• wie sieht der Baum zu 3 + (a + b) * (-c) aus?



Kantorowitsch

Beispiel:

- wie sieht der Baum zu 3 + (a + b) * (-c) aus?
- wie hoch ist der Baum?



Kantorowitsch

Beispiel:

- wie sieht der Baum zu 3 + (a + b) * (-c) aus?
- wie hoch ist der Baum? Antwort: 3



Kantorowitsch

Beispiel:

- wie sieht der Baum zu 3 + (a + b) * (-c) aus?
- wie hoch ist der Baum? Antwort: 3

Die Höhe eines Baumes entspricht auch dem längsten wiederholungsfreien Weg von der Wurzel zu den Blättern.

Kantorowitsch-Bäume zu Reg. Ausdrücken



Kantorowitsch

Beispiel:

- wie sieht der Baum zu 3 + (a + b) * (-c) aus?
- wie hoch ist der Baum? Antwort: 3

Die Höhe eines Baumes entspricht auch dem längsten wiederholungsfreien Weg von der Wurzel zu den Blättern.

Kantorowitsch-Bäume zu Reg. Ausdrücken

 Wird ein regulärer Ausdruck als Baum dargestellt, sind Klammern auch unnötig



Kantorowitsch

Beispiel:

- wie sieht der Baum zu 3 + (a + b) * (-c) aus?
- wie hoch ist der Baum? Antwort: 3

Die Höhe eines Baumes entspricht auch dem längsten wiederholungsfreien Weg von der Wurzel zu den Blättern.

Kantorowitsch-Bäume zu Reg. Ausdrücken

- Wird ein regulärer Ausdruck als Baum dargestellt, sind Klammern auch unnötig
- durch die "Eltern-Kind"- bzw. "ist-Teilausdruck"-Beziehung im Baum ist die Klammerung eindeutig festgelegt

- 1 Guten Morgen...
- 2 Aufgabenblatt 11
- 3 Aufgabenblatt 12
- 4 Reguläre Ausdrücke
- **5** Rechtslineare Grammatiken
- 6 Kantorowitsch-Bäume, Regex-Bäume
- Berechenbarkeit
- Turingmaschinen
- Abschluss

Typische Fragestellung

- Gibt es Aufgaben, die von einem Rechner unabhängig
 - von der Art der Programmierung
 - von physikalischen und elektronischen Beschränkungen
- nicht gelöst werden können?
- Hauptfrage: Welche Probleme sind dann berechenbar?

Typische Fragestellung

- Gibt es Aufgaben, die von einem Rechner unabhängig
 - von der Art der Programmierung
 - von physikalischen und elektronischen Beschränkungen
- nicht gelöst werden können?
- Hauptfrage: Welche Probleme sind dann berechenbar?

Lösungsansätze

Entwicklung neuer Konzepte wie

- eine grundlegende Problemformulierung
- ein grundlegendes Rechnermodell

Dafür muss geklärt werden wie ein Rechner (der nur *Nullen* und *Einsen* kennt) ein Problem überhaupt löst.





Vorschlag

endliche Automaten als Grundlage für "allgemeine" theoretische Aussagen über "Berechenbarkeit"



Vorschlag

endliche Automaten als Grundlage für "allgemeine" theoretische Aussagen über "Berechenbarkeit"

Problem

- nicht mächtig genug!
- es ist zwar erfassbar, ob ein Getränkeautomat (wie aus der Vorlesung) eine Eingabe verabeiten kann, ABER das Modell wäre spätestens bei kontextsensitiven Sprachen überfordert



Vorschlag

endliche Automaten als Grundlage für "allgemeine" theoretische Aussagen über "Berechenbarkeit"

Problem

- nicht mächtig genug!
- es ist zwar erfassbar, ob ein Getränkeautomat (wie aus der Vorlesung) eine Eingabe verabeiten kann, ABER das Modell wäre spätestens bei kontextsensitiven Sprachen überfordert

Frage: Gibt es ein mächtigeres, realistisches Rechnermodell, das geeignet ist?

- 1 Guten Morgen...
- 2 Aufgabenblatt 11
- 3 Aufgabenblatt 12
- 4 Reguläre Ausdrücke
- 6 Rechtslineare Grammatiken
- 6 Kantorowitsch-Bäume, Regex-Bäume
- Berechenbarkeit
- Turingmaschinen
- Abschluss

Aufbau der Turing-Maschine



Erfinder: Alan Turing (1936)

Aufbau der Turing-Maschine



Erfinder: Alan Turing (1936)

Bestandteile

- beidseitig unendliches Eingabe- und Rechenband
- freibeweglicher Lese-/Schreibkopf
- gesteuert von einer endlichen Kontrolle

Aufbau der Turing-Maschine



Erfinder: Alan Turing (1936)

Bestandteile

- beidseitig unendliches Eingabe- und Rechenband
- freibeweglicher Lese-/Schreibkopf
- gesteuert von einer endlichen Kontrolle

Eigenschaften

- Eingabe- und Rechenband enthält eine Folge von Symbolen
- Kontrolle ist in einem von endlich vielen Zuständen
- Zellen des Bandes enthalten jeweils höchstens ein Symbol aus dem Bandalphabet

Funktionsweise



So arbeitet die TM

Ist die Turing-Maschine (TM) in einem bestimmten Zustand und liest ein Symbol...

- so geht sie in einen Folgezustand über
- überschreibt eventuell das Symbol und
- bewegt den Lese-/Schreibkopf eine Stelle nach rechts, nach links oder überhaupt nicht



Ganz genau

Eine Turingmaschine $T = (Z, z_0, X, f, g, m)$ ist festgelegt durch

- eine endliche **Zustandsmenge** Z
- einen Anfangszustand $z_0 \in Z$
- ein endliches Bandalphabet X



Ganz genau

Eine Turingmaschine $T = (Z, z_0, X, f, g, m)$ ist festgelegt durch

- eine endliche **Zustandsmenge** Z
- einen Anfangszustand $z_0 \in Z$
- ein endliches Bandalphabet X
- eine partielle **Zustandsüberführung**sfunktion $f: Z \times X \longrightarrow Z$
- eine partielle **Ausgabe**funktion $g: Z \times X \longrightarrow X$ und
- eine partielle **Bewegung**sfunktion $m: Z \times X \longrightarrow \{-1, 0, 1\}$



Anmerkungen

 Die Funktionen f, g und m beschreiben, wie das eingelesene Zeichen verarbeitet werden soll (gemeinsamer Definitionsbereich)



Anmerkungen

- Die Funktionen f, g und m beschreiben, wie das eingelesene Zeichen verarbeitet werden soll (gemeinsamer Definitionsbereich)
- Bei der Bewegungsfunktion bedeutet -1 oder L eine Bewegung des Lese-/Schreibkopfes nach links, 1 oder R eine Bewegung nach rechts und 0 oder N ein Stehenbleiben



Anmerkungen

- Die Funktionen f, g und m beschreiben, wie das eingelesene Zeichen verarbeitet werden soll (gemeinsamer Definitionsbereich)
- Bei der Bewegungsfunktion bedeutet -1 oder L eine Bewegung des Lese-/Schreibkopfes nach links, 1 oder R eine Bewegung nach rechts und 0 oder N ein Stehenbleiben
- Im Bandalphabet gibt es ein sogenanntes Blanksymbol $\square \in X$ zur Beschreibung einer leeren Zelle auf dem Band
- ein endliches Eingabealphabet $A \subset X \setminus \{\Box\}$
- ullet eine endliche Menge von akzpetierenden Zuständen $F\subseteq Z$



Eine Turingmaschine befindet sich zu jedem Zeitpunkt in einem Gesamtzustand, der als Konfiguration $(z, b, p) \in Z \times X^{\mathbb{Z}} \times \mathbb{Z}$ bezeichnet wird

Vollständig beschrieben durch...



Eine Turingmaschine befindet sich zu jedem Zeitpunkt in einem Gesamtzustand, der als **Konfiguration** $(z, b, p) \in Z \times X^{\mathbb{Z}} \times \mathbb{Z}$ bezeichnet wird

Vollständig beschrieben durch...

• den aktuellen **Zustand** $z \in Z$ der Steuereinheit,



Eine Turingmaschine befindet sich zu jedem Zeitpunkt in einem Gesamtzustand, der als **Konfiguration** $(z, b, p) \in Z \times X^{\mathbb{Z}} \times \mathbb{Z}$ bezeichnet wird

Vollständig beschrieben durch...

- den aktuellen **Zustand** $z \in Z$ der Steuereinheit,
- die aktuelle **Beschriftung des gesamten Bandes**, die man als Abbildung $b: \mathbb{Z} \to X$ formalisieren kann, und



Eine Turingmaschine befindet sich zu jedem Zeitpunkt in einem Gesamtzustand, der als Konfiguration $(z, b, p) \in Z \times X^{\mathbb{Z}} \times \mathbb{Z}$ bezeichnet wird

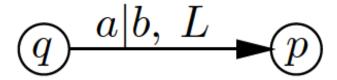
Vollständig beschrieben durch...

- den aktuellen **Zustand** $z \in Z$ der Steuereinheit,
- die aktuelle **Beschriftung des gesamten Bandes**, die man als Abbildung $b: \mathbb{Z} \to X$ formalisieren kann, und
- die aktuelle **Position** $p \in Z$ **des Kopfes**.

Beispiele



Einfach I

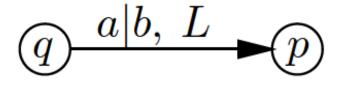


(der Startzustand muss natürlich auch durch einen Pfeil markiert werden, die Markierung fehlt hier)

Beispiele



Einfach I

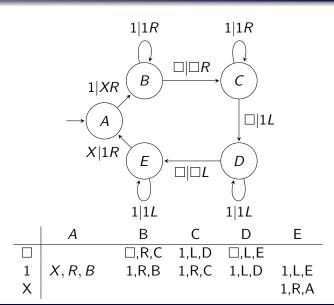


Bedeutung des Übergangs

Ist die TM im Zustand q und liest das Sysmbol a, so überschreit sie dieses a mit b, geht auf dem Bande eine Stelle nach links und wechselt in den Zustand p

Beispiel





Verhalten von TM bei Entscheidungsproblemen



akzeptierende Zustände

Eine Turing-Maschine akzeptiert eine Eingabe w, wenn sie nach Lesen von w in einem akzeptierenden Zustand $F \subset Z$ stoppt. Es gibt Eingaben, für die eine Turing-Maschine unter Umständen niemals stoppt, bzw. nur einem nicht akzeptierenden Zustand. Wann terminiert eine TM?

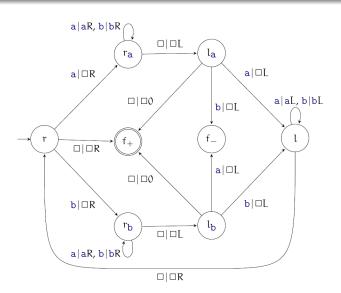


Aufgabe

Gebt die Berechnung der folgenden Turingmaschine für die Eingaben

- aba
- a baba







Aufgabe

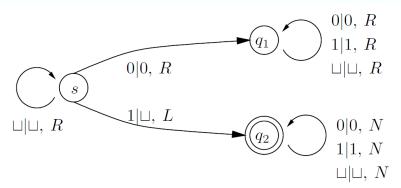
Gebt eine Turingmaschine an die alle Wörter aus $\{0,1\}^*$ erkennt, die mit einer Eins beginnen



Aufgabe

Gebt eine Turingmaschine an die alle Wörter aus $\{0,1\}^*$ erkennt, die mit einer Eins beginnen

Lösung (die Markierung des Startzustands fehlt)



partielle Funktion von TM



Eine TM kann mehr...

Eine Turing-Maschine erkennt nicht nur Mengen von Wörtern (Sprachen), sondern sie verändert auch die Eingabe und hat insofern auch eine Ausgabe (= Inhalt des Bandes nach der Bearbeitung).

Eine TM kann so zur Berechnung von Funktionen genutzt werden, wie der Addition oder dem Vergleich zweier Binärzahlen

Wir konstruieren eine solche TM...



Ihr seid dran...

Gebt eine Turing-Maschine an, die eine in Binärcodierung gegebene natürliche Zahl $n \geq 1$ mit der Zahl 2 multipliziert. Der Schreib-/Lesekopf soll sich zu Anfang und Ende der Berechnung jeweils auf dem ersten Bit (der größten Zweierpotenz zugeordnet) der Darstellung von n befinden...

Wir konstruieren eine solche TM...



Ihr seid dran...

Gebt eine Turing-Maschine an, die eine in Binärcodierung gegebene natürliche Zahl $n \geq 1$ mit der Zahl 2 multipliziert. Der Schreib-/Lesekopf soll sich zu Anfang und Ende der Berechnung jeweils auf dem ersten Bit (der größten Zweierpotenz zugeordnet) der Darstellung von n befinden...

Lösung

Entscheidbarkeit und Berechenbarkeit



Die "verschiedenen Arten" von TMs

Verknüpfung von Entscheidbarkeit von Sprachen und der Berechenbarkeit von Funktionen:

- Eine Turing-Maschine akzeptiert eine Sprache L, wenn sie genau für die Eingaben $w \in L$ 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.





Die Menge aller Konfigurationen bezeichnen wir als \mathbb{C}_t



Die Menge aller Konfigurationen bezeichnen wir als \mathbb{C}_t

Schritt einer TM

- $\Delta_{\mathsf{x}}:\mathbb{C}_{t} \dashrightarrow \mathbb{C}_{t}$
- $\Delta_1(c)$ liefert direkte Nachfolgekonfiguration zu c



Die Menge aller Konfigurationen bezeichnen wir als \mathbb{C}_t

Schritt einer TM

- $\bullet \ \Delta_{\mathsf{X}} : \mathbb{C}_t \dashrightarrow \mathbb{C}_t$
- ullet $\Delta_1(c)$ liefert direkte Nachfolgekonfiguration zu c

Endkonfigurationen einer TM

ist erreicht, falls $\Delta_1(c)$ nicht definiert ist





endliche Berechnung

- endliche Folge von Konfigurationen $(c_0, c_1, c_2, ..., c_t)$,
- wobei $0 < i \le t$ gilt $c_i = \Delta_1(c_{i-1})$



endliche Berechnung

- endliche Folge von Konfigurationen $(c_0, c_1, c_2, ..., c_t)$,
- wobei $0 < i \le t$ gilt $c_i = \Delta_1(c_{i-1})$

haltende Berechnung

- endliche Berechnung
- deren letzte Konfiguration eine Endkonfiguration ist



endliche Berechnung

- endliche Folge von Konfigurationen $(c_0, c_1, c_2, ..., c_t)$,
- wobei $0 < i \le t$ gilt $c_i = \Delta_1(c_{i-1})$

haltende Berechnung

- endliche Berechnung
- deren letzte Konfiguration eine Endkonfiguration ist

unendliche Berechnung

- unendliche Folge von Konfigurationen $(c_0, c_1, c_2, ...)$
- wobei 0 < i gilt $c_i = \Delta_1(c_{i-1})$
- nicht haltend





analog zu endlichen Automaten

• Erkennung formaler Sprachen: ein Bit akzeptiert/abgelehnt



- Erkennung formaler Sprachen: ein Bit akzeptiert/abgelehnt
- Teilmenge F ⊂ Z akzeptierender Zustände
- TM akzeptiert Eingabewort w, wenn



- Erkennung formaler Sprachen: ein Bit akzeptiert/abgelehnt
- Teilmenge F ⊂ Z akzeptierender Zustände
- TM akzeptiert Eingabewort w, wenn
 - TM für Eingabe w hält und



- Erkennung formaler Sprachen: ein Bit akzeptiert/abgelehnt
- Teilmenge F ⊂ Z akzeptierender Zustände
- TM akzeptiert Eingabewort w, wenn
 - TM für Eingabe w hält und
 - ullet der Zustand der Endkonfiguration $\Delta_*(c_0(w))$ akzepierend ist



- Erkennung formaler Sprachen: ein Bit akzeptiert/abgelehnt
- ullet Teilmenge $F \subset Z$ akzeptierender Zustände
- TM akzeptiert Eingabewort w, wenn
 - TM für Eingabe w hält und
 - ullet der Zustand der Endkonfiguration $\Delta_*(c_0(w))$ akzepierend ist
- L(T): Menge der akzeptierten Wörter

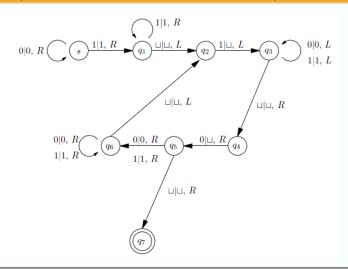


Aufgabe

Gebt ein TM-Akzeptor an, der die Sprache $L^==\{0^n1^n:n\geq 1\}$ akzeptiert



Lösung (die Markierung des Startzustands fehlt)





zwei Möglichkeiten, wenn w von TM nicht akzeptiert wird



zwei Möglichkeiten, wenn w von TM nicht akzeptiert wird

● TM hält für Eingabe w, aber Endzustand nicht akzeptierend



zwei Möglichkeiten, wenn w von TM nicht akzeptiert wird

- TM hält für Eingabe w, aber Endzustand nicht akzeptierend
- TM hält für Eingabe w nicht



zwei Möglichkeiten, wenn w von TM nicht akzeptiert wird

- 1 TM hält für Eingabe w, aber Endzustand nicht akzeptierend
- 2 TM hält für Eingabe w nicht

Was wissen wir über die Berechnung?



zwei Möglichkeiten, wenn w von TM nicht akzeptiert wird

- 1 TM hält für Eingabe w, aber Endzustand nicht akzeptierend
- 2 TM hält für Eingabe w nicht

Was wissen wir über die Berechnung?

• TM ist fertig und lehnt die Eingabe ab



zwei Möglichkeiten, wenn w von TM nicht akzeptiert wird

- TM hält für Eingabe w, aber Endzustand nicht akzeptierend
- TM hält für Eingabe w nicht

Was wissen wir über die Berechnung?

- TM ist fertig und lehnt die Eingabe ab
- 2 TM ist noch nicht fertig (Ob TM irgendwann w noch akzeptiert oder ablehnt, ist unklar!)



zwei Möglichkeiten, wenn w von TM nicht akzeptiert wird

- TM hält für Eingabe w, aber Endzustand nicht akzeptierend
- 2 TM hält für Eingabe w nicht

Was wissen wir über die Berechnung?

- TM ist fertig und lehnt die Eingabe ab
- TM ist noch nicht fertig (Ob TM irgendwann w noch akzeptiert oder ablehnt, ist unklar!)

Wir halten in zwei Definitionen fest

• L heißt entscheidbare Sprache, wenn es eine TM gibt, die immer hält und L akzeptiert.



zwei Möglichkeiten, wenn w von TM nicht akzeptiert wird

- TM hält für Eingabe w, aber Endzustand nicht akzeptierend
- 2 TM hält für Eingabe w nicht

Was wissen wir über die Berechnung?

- TM ist fertig und lehnt die Eingabe ab
- TM ist noch nicht fertig (Ob TM irgendwann w noch akzeptiert oder ablehnt, ist unklar!)

Wir halten in zwei Definitionen fest

- L heißt entscheidbare Sprache, wenn es eine TM gibt, die immer hält und L akzeptiert.
- L heißt aufzählbare(semi-entscheidbar) Sprache, wenn es eine TM gibt, die L akzeptiert

- 1 Guten Morgen...
- 2 Aufgabenblatt 11
- 3 Aufgabenblatt 12
- 4 Reguläre Ausdrücke
- 5 Rechtslineare Grammatiken
- 6 Kantorowitsch-Bäume, Regex-Bäume
- Berechenbarkeit
- Turingmaschinen
- Abschluss



Was ihr nun wissen solltet!		



Was ihr nun wissen solltet!

• Was verstehen wir unter Berechenbarkeit?



- Was verstehen wir unter Berechenbarkeit?
- Welches Rechenmodell ist in der Informatik von zentrale Bedeutung?



- Was verstehen wir unter Berechenbarkeit?
- Welches Rechenmodell ist in der Informatik von zentrale Bedeutung?
- Was sind die Bestandteile dieses Rechenmodells? Und wie arbeitet es?



- Was verstehen wir unter Berechenbarkeit?
- Welches Rechenmodell ist in der Informatik von zentrale Bedeutung?
- Was sind die Bestandteile dieses Rechenmodells? Und wie arbeitet es?
- Was lässt sich aus diesem Rechenmodell ableiten?



- Was verstehen wir unter Berechenbarkeit?
- Welches Rechenmodell ist in der Informatik von zentrale Bedeutung?
- Was sind die Bestandteile dieses Rechenmodells? Und wie arbeitet es?
- Was lässt sich aus diesem Rechenmodell ableiten?
- Wann ist eine Sprache entscheidbar?



Was ihr nun wissen solltet!

- Was verstehen wir unter Berechenbarkeit?
- Welches Rechenmodell ist in der Informatik von zentrale Bedeutung?
- Was sind die Bestandteile dieses Rechenmodells? Und wie arbeitet es?
- Was lässt sich aus diesem Rechenmodell ableiten?
- Wann ist eine Sprache entscheidbar?

Ihr wisst was nicht?

Stellt jetzt Fragen!

