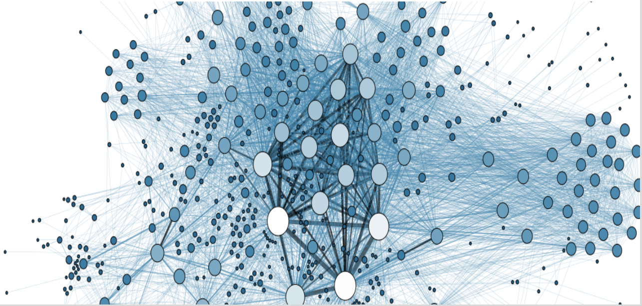
Karlsruher Institut für Technologie

Grundbegriffe der Informatik
Tutorium 38

Turingmaschinen, Reguläre Sprachen

Patrick Fetzer, uxkln@student.kit.edu | 08.02.2018



Definition von Turingmaschinen

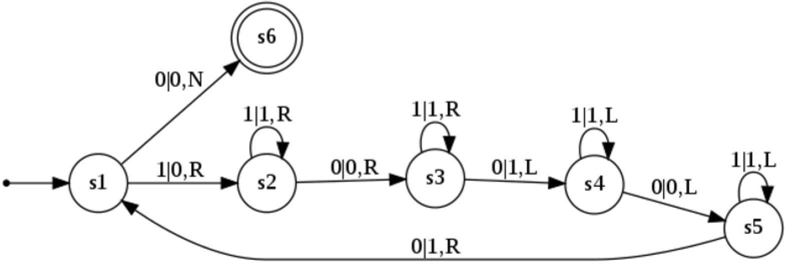
Definition von Turingmaschinen

Eine Turingmaschine $T = (Z, z_0, X, f, g, m)$ besteht aus:

- Z Zustandsmenge
- $z_0 \in Z$ Startzustand
- X Bandalphabet
- $\square$ Blanksymbol (sozusagen Markierung für Leerzeichen)
- $f : Z \times X \rightarrow Z$ partielle Zustandsübergangsfunktion
- $g : Z \times X \rightarrow X$ partielle Ausgabefunktion
- $m : Z \times X \rightarrow \{L, N, R\}$ partielle Bewegungsfunktion

Anmerkung: partielle Funktionen sind nicht linkstotal, also manche Elemente des Definitionsbereichs werden nicht abgebildet.

Beispiel einer Turingmaschine



Halten von Turingmaschinen

Halten von Turingmaschinen

Halten einer Turingmaschine

Wenn eine Turingmaschine in einem Zustand ist, für den das nächste Eingabezeichen durch die Übergangsfunktion f nicht definiert ist, hält die Maschine.

Halten von Turingmaschinen

Halten einer Turingmaschine

Wenn eine Turingmaschine in einem Zustand ist, für den das nächste Eingabezeichen durch die Übergangsfunktion f nicht definiert ist, hält die Maschine.

Wann hält also eine Turingmaschine nicht?

Halten von Turingmaschinen

Halten einer Turingmaschine

Wenn eine Turingmaschine in einem Zustand ist, für den das nächste Eingabezeichen durch die Übergangsfunktion f nicht definiert ist, hält die Maschine.

Wann hält also eine Turingmaschine nicht?

Nicht-Halten einer Turingmmaschine

Wenn eine Turingmaschine in eine endlose Schleife gerät, so hält sie nicht.

Entscheidbarkeit

Entscheidbarkeit

Durch Turingmaschine akzeptierte Sprache

Eine Turingmaschine akzeptiert eine formale Sprache L, wenn sie für jedes Wort $w \in L$ in einem akzeptierenden Zustand hält.

Entscheidbarkeit

Durch Turingmaschine akzeptierte Sprache

Eine Turingmaschine akzeptiert eine formale Sprache L, wenn sie für jedes Wort $w \in L$ in einem akzeptierenden Zustand hält.

Entscheidbare Sprache

Eine formale Sprache L heißt entscheidbar, wenn es eine Turingmaschine gibt, die immer hält und L akzeptiert.

Entscheidbarkeit

Karlsruher Institut für Technologie

Durch Turingmaschine akzeptierte Sprache

Eine Turingmaschine akzeptiert eine formale Sprache L, wenn sie für jedes Wort $w \in L$ in einem akzeptierenden Zustand hält.

Entscheidbare Sprache

Eine formale Sprache L heißt entscheidbar, wenn es eine Turingmaschine gibt, die immer hält und L akzeptiert.

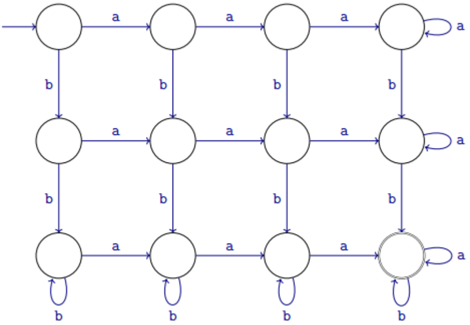
Aufzählbare Sprache

Eine formale Sprache L heißt aufzählbar, wenn es eine Turingmaschine gibt, die L akzeptiert.

Vom endlichen Akzeptor zur Turingmaschine

Akzeptieren von Turingmaschinen

Wie kann man aus dem gegebenen endlichen Akzeptor eine Turingmaschine machen, die dieselbe Sprache akzeptiert?



Lösung

Einfach gesagt: mache aus jedem Übergang a einen Turingmaschinen-Übergang der Art $a|a, R$, also bei jedem Zeichen mache den Zustandsübergang, überschreibe aber das Zeichen nicht und gehe zum nächsten Zeichen.

Lösung

Einfach gesagt: mache aus jedem Übergang a einen Turingmaschinen-Übergang der Art $a|a, R$, also bei jedem Zeichen mache den Zustandsübergang, überschreibe aber das Zeichen nicht und gehe zum nächsten Zeichen.

Formaler ausgedrückt?

Lösung

Einfach gesagt: mache aus jedem Übergang a einen Turingmaschinen-Übergang der Art $a|a, R$, also bei jedem Zeichen mache den Zustandsübergang, überschreibe aber das Zeichen nicht und gehe zum nächsten Zeichen.

Formaler ausgedrückt?

- Für allgemeinen endlichen Akzeptor (Z, z_0, X, f, Y, h), definiere eine Turingmaschine $T := (Z, z_0, X \cup Y, f, g, h)$, also Z, z_0, f gleich und mit Bandalphabet $=$ Eingabealphabet $\cup$ Ausgabealphabet
- $g(z, x) := x \quad \forall (z, x)$ in f definiert
- $m(z, x) := R \quad \forall (z, y)$ in f definiert

Lösung

Einfach gesagt: mache aus jedem Übergang a einen Turingmaschinen-Übergang der Art $a|a, R$, also bei jedem Zeichen mache den Zustandsübergang, überschreibe aber das Zeichen nicht und gehe zum nächsten Zeichen.

Formaler ausgedrückt?

- Für allgemeinen endlichen Akzeptor (Z, z_0, X, f, Y, h), definiere eine Turingmaschine $T := (Z, z_0, X \cup Y, f, g, h)$, also Z, z_0, f gleich und mit Bandalphabet $=$ Eingabealphabet $\cup$ Ausgabealphabet
- $g(z, x) := x \quad \forall (z, x)$ in f definiert
- $m(z, x) := R \quad \forall (z, y)$ in f definiert

Jeder endliche Akzeptor kann so zu einer Turingmaschine umgeformt werden, die dieselbe Sprache akzeptiert.

Über endliche Akzeptoren hinaus

Sei L die Sprache von Palindromen über $\{a, b\}$
($L = \{aabaa, bbababb, aa, \varepsilon\}$).

Über endliche Akzeptoren hinaus

Sei L die Sprache von Palindromen über $\{a, b\}$
$(L = \{aabaa, bbababb, aa, \varepsilon\})$.

- Ist die Sprache regulär, also gibt es einen endlichen Akzeptor, der diese akzeptiert?

Über endliche Akzeptoren hinaus

Sei L die Sprache von Palindromen über $\{a, b\}$
($L = \{aabaa, bbababb, aa, \varepsilon\}$).

- Ist die Sprache regulär, also gibt es einen endlichen Akzeptor, der diese akzeptiert? Nein.

Über endliche Akzeptoren hinaus

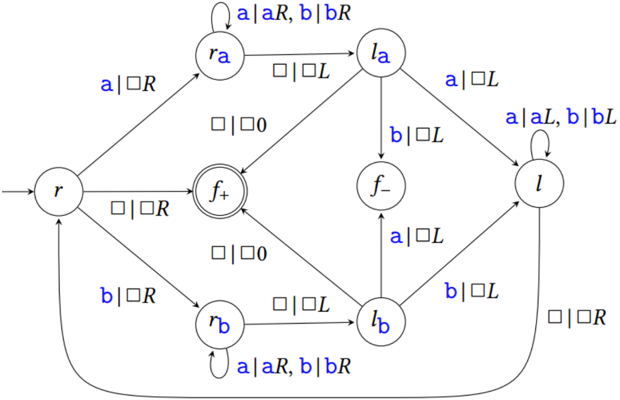
Karlsruher Institut für Technologie

Sei L die Sprache von Palindromen über $\{a, b\}$
($L = \{aabaa, bbababb, aa, \varepsilon\}$).

- Ist die Sprache regulär, also gibt es einen endlichen Akzeptor, der diese akzeptiert? Nein.
- Ist die Sprache entscheidbar, also gibt es eine stets haltende Turingmaschine, die L akzeptiert?

Palindromerkennung mit Turingmaschinen

Ja, nämlich:



Turingmaschinen Entwurfsaufgabe

Turingmaschine Entwurf

Entwerfe eine Turingmaschine, die...

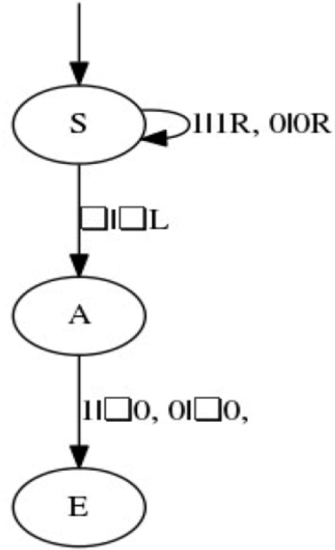
- als Eingabe eine Binärzahl auf dem Band erhält
- als Ausgabe diese Zahl restlos durch zwei teilt und auf dem Band stehen lässt

Turingmaschinen Entwurfsaufgabe

Turingmaschine Entwurf

Entwerfe eine Turingmaschine, die...

- als Eingabe eine Binärzahl auf dem Band erhält
- als Ausgabe diese Zahl restlos durch zwei teilt und auf dem Band stehen lässt



Turingmaschinen Entwurfsaufgabe

Turingmaschine Entwurf

Entwerfe eine Turingmaschine, die...

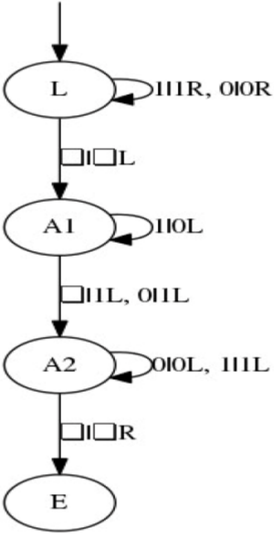
- als Eingabe eine Binärzahl auf dem Band erhält
- als Ausgabe diese Zahl um eins erhöht auf dem Band stehen lässt
- den Kopf der Turingmaschine auf dem ersten Zeichen der Ausgabe stehen hat.

Turingmaschinen Entwurfsaufgabe

Turingmaschine Entwurf

Entwerfe eine Turingmaschine, die...

- als Eingabe eine Binärzahl auf dem Band erhält
- als Ausgabe diese Zahl um eins erhöht auf dem Band stehen lässt
- den Kopf der Turingmaschine auf dem ersten Zeichen der Ausgabe stehen hat.



Turingmaschine Entwurfsaufgabe

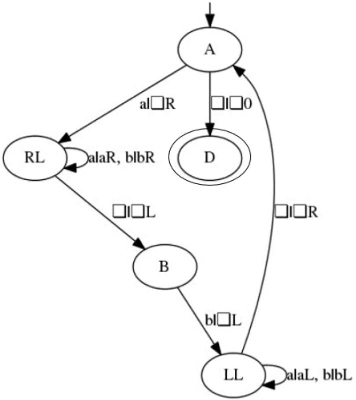
Turingmaschine Entwurf

Entwerfe eine Turingmaschine, die die Sprache $\{a^k b^k : k \in \mathbb{N}_0\}$ erkennt.

Turingmaschine Entwurfsaufgabe

Turingmaschine Entwurf

Entwerfe eine Turingmaschine, die die Sprache $\{a^k b^k : k \in \mathbb{N}_0\}$ erkennt.



Konfiguration von Turingmaschinen

Angenommen, man kennt eine Turingmaschine, hat mit der Abarbeitung eines Wortes angefangen, will aber pausieren, um später weiterzumachen...

Was muss man sich alles merken, um später weiter zu machen?

Konfiguration von Turingmaschinen

Angenommen, man kennt eine Turingmaschine, hat mit der Abarbeitung eines Wortes angefangen, will aber pausieren, um später weiterzumachen...

Was muss man sich alles merken, um später weiter zu machen?

- Derzeitiger Zustand, in dem die Turingmaschine steht

Konfiguration von Turingmaschinen

Angenommen, man kennt eine Turingmaschine, hat mit der Abarbeitung eines Wortes angefangen, will aber pausieren, um später weiterzumachen...

Was muss man sich alles merken, um später weiter zu machen?

- Derzeitiger Zustand, in dem die Turingmaschine steht
- Inhalt des Bandes

Konfiguration von Turingmaschinen

Angenommen, man kennt eine Turingmaschine, hat mit der Abarbeitung eines Wortes angefangen, will aber pausieren, um später weiterzumachen...

Was muss man sich alles merken, um später weiter zu machen?

- Derzeitiger Zustand, in dem die Turingmaschine steht
- Inhalt des Bandes

Konfiguration von Turingmaschinen

Wenn während dem Arbeiten einer Turingmaschine auf dem Band das Wort $w_1 a w_2$ mit $w_1, w_2 \in X^*, a \in X$ steht

Konfiguration von Turingmaschinen

Angenommen, man kennt eine Turingmaschine, hat mit der Abarbeitung eines Wortes angefangen, will aber pausieren, um später weiterzumachen...

Was muss man sich alles merken, um später weiter zu machen?

- Derzeitiger Zustand, in dem die Turingmaschine steht
- Inhalt des Bandes

Konfiguration von Turingmaschinen

Wenn während dem Arbeiten einer Turingmaschine auf dem Band das Wort $w_1 a w_2$ mit $w_1, w_2 \in X^*, a \in X$ steht, der Kopf der Turingmaschine auf das Zeichen a zeigt

Konfiguration von Turingmaschinen

Angenommen, man kennt eine Turingmaschine, hat mit der Abarbeitung eines Wortes angefangen, will aber pausieren, um später weiterzumachen...

Was muss man sich alles merken, um später weiter zu machen?

- Derzeitiger Zustand, in dem die Turingmaschine steht
- Inhalt des Bandes

Konfiguration von Turingmaschinen

Wenn während dem Arbeiten einer Turingmaschine auf dem Band das Wort $w_1 a w_2$ mit $w_1, w_2 \in X^*, a \in X$ steht, der Kopf der Turingmaschine auf das Zeichen a zeigt und die Turingmaschine im Zustand z ist

Konfiguration von Turingmaschinen

Angenommen, man kennt eine Turingmaschine, hat mit der Abarbeitung eines Wortes angefangen, will aber pausieren, um später weiterzumachen...

Was muss man sich alles merken, um später weiter zu machen?

- Derzeitiger Zustand, in dem die Turingmaschine steht
- Inhalt des Bandes

Konfiguration von Turingmaschinen

Wenn während dem Arbeiten einer Turingmaschine auf dem Band das Wort $w_1 a w_2$ mit $w_1, w_2 \in X^*$, $a \in X$ steht, der Kopf der Turingmaschine auf das Zeichen a zeigt und die Turingmaschine im Zustand z ist, so schreibt man die Konfiguration der Turingmaschine als $\square w_1 (z) a w_2 \square$.

Konfiguration von Turingmaschinen

Beispiel:

$$\square\square\square\square\square\square\square\square abcbabb\,d\,aabc\square\square\square\square\square$$

$\uparrow$

KOPF

Konfiguration von Turingmaschinen

Karlsruher Institut für Technologie

Beispiel:

$$\square\square\square\square\square\square\square\square abcbabb\textcolor{blue}{d}aabc\square\square\square\square\square$$

$$\uparrow$$

KOPF

...sei das Band der Turingmaschine während Abarbeitung der Eingabe, dazu steht sie im Zustand z_4.

Konfiguration von Turingmaschinen

Beispiel:

$$\square\square\square\square\square\square\square\square abcbabb\color{blue}{d}\color{black}aabc\square\square\square\square\square$$

$$\uparrow$$

KOPF

...sei das Band der Turingmaschine während Abarbeitung der Eingabe, dazu steht sie im Zustand z_4.

Dann sieht sieht die Konfiguration der Turingmaschine so aus:

Konfiguration von Turingmaschinen

Karlsruher Institut für Technologie

Beispiel:

$$\square\square\square\square\square\square\square\square abcbabb\textcolor{blue}{d}aabc\square\square\square\square\square$$

$$\uparrow$$

KOPF

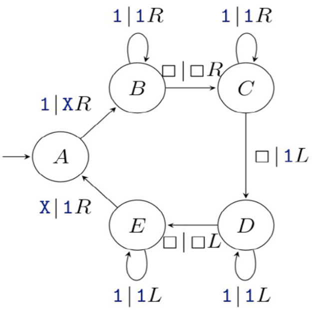
...sei das Band der Turingmaschine während Abarbeitung der Eingabe, dazu steht sie im Zustand z_4.

Dann sieht sieht die Konfiguration der Turingmaschine so aus: $\square abcbabb(z_4)daabc\square$

Dokumentation einer Abarbeitung mit Konfigurationen

Aufgabe zu Konfigurationen

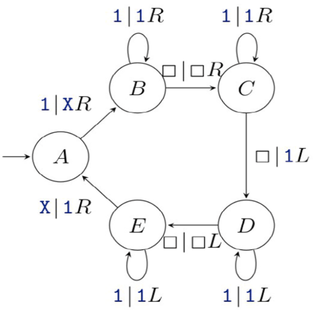
Gebe alle Konfigurationen der Turingmaschine bei Abarbeitung des Wortes 11 an.



Dokumentation einer Abarbeitung mit Konfigurationen

Aufgabe zu Konfigurationen

Gebe alle Konfigurationen der Turingmaschine bei Abarbeitung des Wortes 11 an.

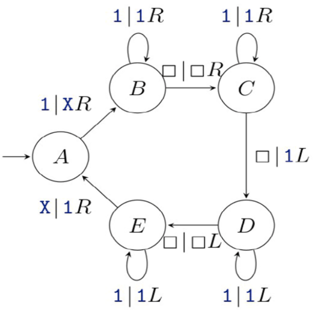


$\square(A)11\square$

Dokumentation einer Abarbeitung mit Konfigurationen

Aufgabe zu Konfigurationen

Gebe alle Konfigurationen der Turingmaschine bei Abarbeitung des Wortes 11 an.



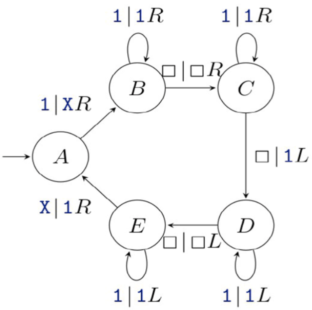
$$\square(A)11\square$$

$$\rightarrow \square X(B)1\square$$

Dokumentation einer Abarbeitung mit Konfigurationen

Aufgabe zu Konfigurationen

Gebe alle Konfigurationen der Turingmaschine bei Abarbeitung des Wortes 11 an.

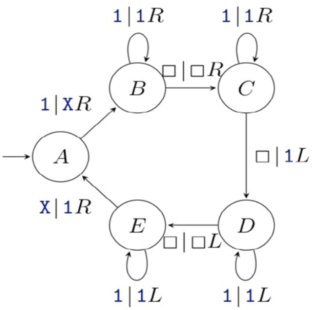


$$\Box(A)11\Box$$
$$\rightarrow \Box X(B)1\Box$$
$$\rightarrow \Box X1(B)\Box$$

Dokumentation einer Abarbeitung mit Konfigurationen

Aufgabe zu Konfigurationen

Gebe alle Konfigurationen der Turingmaschine bei Abarbeitung des Wortes 11 an.

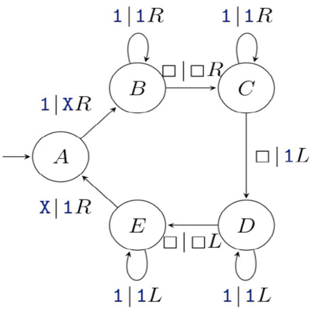


$$\square(A)11\square$$
$$\rightarrow \square X(B)1\square$$
$$\rightarrow \square X1(B)\square$$
$$\rightarrow \square X1\square(C)\square$$

Dokumentation einer Abarbeitung mit Konfigurationen

Aufgabe zu Konfigurationen

Gebe alle Konfigurationen der Turingmaschine bei Abarbeitung des Wortes 11 an.

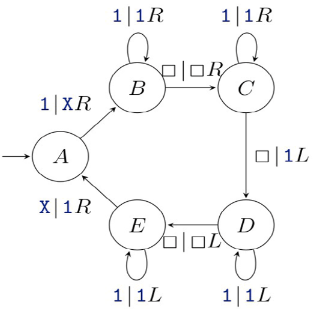


$$\square(A)11\square$$
$$\rightarrow \square X(B)1\square$$
$$\rightarrow \square X1(B)\square$$
$$\rightarrow \square X1\square(C)\square$$
$$\rightarrow \square X1(D)\square1\square$$

Dokumentation einer Abarbeitung mit Konfigurationen

Aufgabe zu Konfigurationen

Gebe alle Konfigurationen der Turingmaschine bei Abarbeitung des Wortes 11 an.

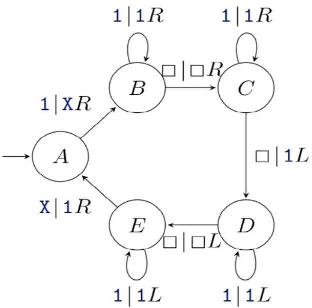


$$\square(A)11\square$$
$$\rightarrow \square X(B)1\square$$
$$\rightarrow \square X1(B)\square$$
$$\rightarrow \square X1\square(C)\square$$
$$\rightarrow \square X1(D)\square 1\square$$
$$\rightarrow \square X(E)1\square 1\square$$

Dokumentation einer Abarbeitung mit Konfigurationen

Aufgabe zu Konfigurationen

Gebe alle Konfigurationen der Turingmaschine bei Abarbeitung des Wortes 11 an.

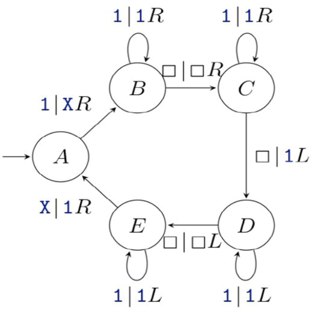


$$\square(A)11\square$$
$$\rightarrow \square X(B)1\square$$
$$\rightarrow \square X1(B)\square$$
$$\rightarrow \square X1\square(C)\square$$
$$\rightarrow \square X1(D)\square 1\square$$
$$\rightarrow \square X(E)1\square 1\square$$
$$\rightarrow \square(E)X1\square 1\square$$

Dokumentation einer Abarbeitung mit Konfigurationen

Aufgabe zu Konfigurationen

Gebe alle Konfigurationen der Turingmaschine bei Abarbeitung des Wortes 11 an.

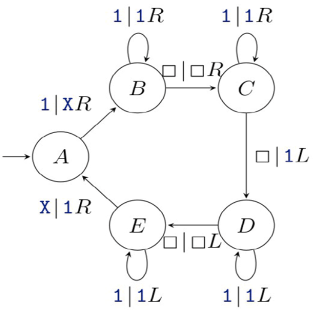


$$
\begin{aligned}
& \square(A)11\square \\
\rightarrow\ & \square X(B)1\square \\
\rightarrow\ & \square X1(B)\square \\
\rightarrow\ & \square X1\square(C)\square \\
\rightarrow\ & \square X1(D)\square 1\square \\
\rightarrow\ & \square X(E)1\square 1\square \\
\rightarrow\ & \square(E)X1\square 1\square \\
\rightarrow\ & \square 1(A)1\square 1\square
\end{aligned}
$$

Dokumentation einer Abarbeitung mit Konfigurationen

Aufgabe zu Konfigurationen

Gebe alle Konfigurationen der Turingmaschine bei Abarbeitung des Wortes 11 an.

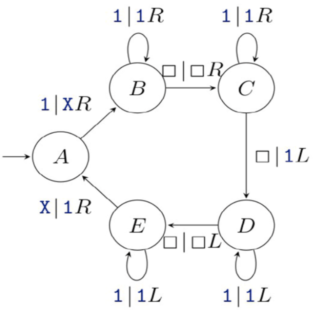


$$\square(A)11\square$$
$$\to \square X(B)1\square$$
$$\to \square X1(B)\square$$
$$\to \square X1\square(C)\square$$
$$\to \square X1(D)\square 1\square$$
$$\to \square X(E)1\square 1\square$$
$$\to \square(E)X1\square 1\square$$
$$\to \square 1(A)1\square 1\square$$
$$\to \square 1X(B)\square 1\square$$

Dokumentation einer Abarbeitung mit Konfigurationen

Aufgabe zu Konfigurationen

Gebe alle Konfigurationen der Turingmaschine bei Abarbeitung des Wortes 11 an.



$\Box(A)11\Box$

$\rightarrow \Box X(B)1\Box$

$\rightarrow \Box X1(B)\Box$

$\rightarrow \Box X1\Box(C)\Box$

$\rightarrow \Box X1(D)\Box 1\Box$

$\rightarrow \Box X(E)1\Box 1\Box$

$\rightarrow \Box(E)X1\Box 1\Box$

$\rightarrow \Box 1(A)1\Box 1\Box$

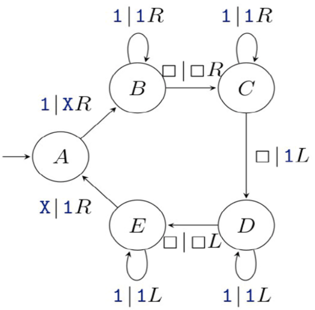
$\rightarrow \Box 1X(B)\Box 1\Box$

$\rightarrow \Box 1X\Box(C)1\Box$

Dokumentation einer Abarbeitung mit Konfigurationen

Aufgabe zu Konfigurationen

Gebe alle Konfigurationen der Turingmaschine bei Abarbeitung des Wortes 11 an.



$\square(A)11\square$

$\rightarrow \square X(B)1\square$

$\rightarrow \square X1(B)\square$

$\rightarrow \square X1\square(C)\square$

$\rightarrow \square X1(D)\square 1\square$

$\rightarrow \square X(E)1\square 1\square$

$\rightarrow \square(E)X1\square 1\square$

$\rightarrow \square 1(A)1\square 1\square$

$\rightarrow \square 1X(B)\square 1\square$

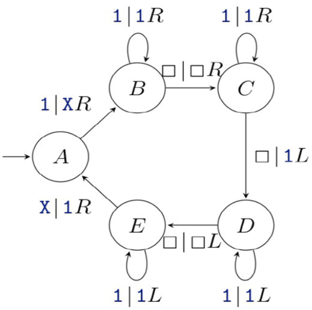
$\rightarrow \square 1X\square(C)1\square$

$\rightarrow \square 1X\square 1(C)\square$

Dokumentation einer Abarbeitung mit Konfigurationen

Aufgabe zu Konfigurationen

Gebe alle Konfigurationen der Turingmaschine bei Abarbeitung des Wortes 11 an.



$\square(A)11\square$

$\to \square X(B)1\square$

$\to \square X1(B)\square$

$\to \square X1\square(C)\square$

$\to \square X1(D)\square 1\square$

$\to \square X(E)1\square 1\square$

$\to \square(E)X1\square 1\square$

$\to \square 1(A)1\square 1\square$

$\to \square 1X(B)\square 1\square$

$\to \square 1X\square(C)1\square$

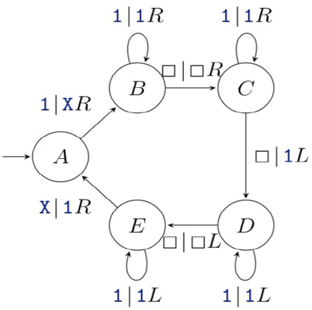
$\to \square 1X\square 1(C)\square$

$\to \square 1X\square(D)11\square$

Dokumentation einer Abarbeitung mit Konfigurationen

Aufgabe zu Konfigurationen

Gebe alle Konfigurationen der Turingmaschine bei Abarbeitung des Wortes 11 an.



$\square(A)11\square$

$\rightarrow \square X(B)1\square$

$\rightarrow \square X1(B)\square$

$\rightarrow \square X1\square(C)\square$

$\rightarrow \square X1(D)\square 1\square$

$\rightarrow \square X(E)1\square 1\square$

$\rightarrow \square(E)X1\square 1\square$

$\rightarrow \square 1(A)1\square 1\square$

$\rightarrow \square 1X(B)\square 1\square$

$\rightarrow \square 1X\square(C)1\square$

$\rightarrow \square 1X\square 1(C)\square$

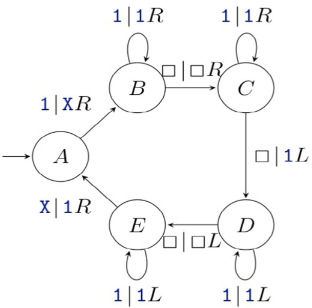
$\rightarrow \square 1X\square(D)11\square$

$\rightarrow \square 1X(D)\square 11\square$

Dokumentation einer Abarbeitung mit Konfigurationen

Aufgabe zu Konfigurationen

Gebe alle Konfigurationen der Turingmaschine bei Abarbeitung des Wortes 11 an.



$\square(A)11\square$

$\rightarrow \square X(B)1\square$

$\rightarrow \square X1(B)\square$

$\rightarrow \square X1\square(C)\square$

$\rightarrow \square X1(D)\square 1\square$

$\rightarrow \square X(E)1\square 1\square$

$\rightarrow \square(E)X1\square 1\square$

$\rightarrow \square 1(A)1\square 1\square$

$\rightarrow \square 1X(B)\square 1\square$

$\rightarrow \square 1X\square(C)1\square$

$\rightarrow \square 1X\square 1(C)\square$

$\rightarrow \square 1X\square(D)11\square$

$\rightarrow \square 1X(D)\square 11\square$

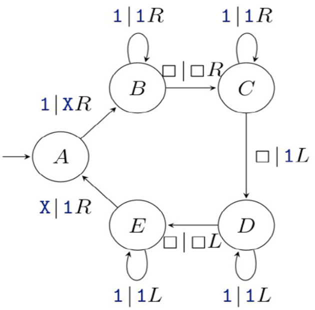
$\rightarrow \square 1(E)X\square 11\square$

Dokumentation einer Abarbeitung mit Konfigurationen

Karlsruher Institut für Technologie

Aufgabe zu Konfigurationen

Gebe alle Konfigurationen der Turingmaschine bei Abarbeitung des Wortes 11 an.



$\square(A)11\square$

$\rightarrow \square X(B)1\square$

$\rightarrow \square X1(B)\square$

$\rightarrow \square X1\square(C)\square$

$\rightarrow \square X1(D)\square 1\square$

$\rightarrow \square X(E)1\square 1\square$

$\rightarrow \square(E)X1\square 1\square$

$\rightarrow \square 1(A)1\square 1\square$

$\rightarrow \square 1X(B)\square 1\square$

$\rightarrow \square 1X\square(C)1\square$

$\rightarrow \square 1X\square 1(C)\square$

$\rightarrow \square 1X\square(D)11\square$

$\rightarrow \square 1X(D)\square 11\square$

$\rightarrow \square 1(E)X\square 11\square$

$\rightarrow \square 11(A)\square 11\square$

Halteproblem

Halteproblem: Für einen gegebenen Algorithmus, gelangt dieser bei seiner Abarbeitung zu einem Ende und hält?

Halteproblem

Halteproblem: Für einen gegebenen Algorithmus, gelangt dieser bei seiner Abarbeitung zu einem Ende und hält?

- Algorithmen können durch Turingmaschinen durchgeführt werden

Halteproblem

Halteproblem: Für einen gegebenen Algorithmus, gelangt dieser bei seiner Abarbeitung zu einem Ende und hält?

- Algorithmen können durch Turingmaschinen durchgeführt werden
- Turingmaschinen können durch sogenannte universelle Turingmaschinen simuliert werden

Halteproblem

Halteproblem: Für einen gegebenen Algorithmus, gelangt dieser bei seiner Abarbeitung zu einem Ende und hält?

- Algorithmen können durch Turingmaschinen durchgeführt werden
- Turingmaschinen können durch sogenannte universelle Turingmaschinen simuliert werden
    - Wenn eine Turingmaschine T kodiert ist mit dem Wort w, dann ist $T_w : X \longrightarrow X$ eine Funktion, die Eingaben auf die Ausgabe der Turingmaschine T mappt.

Halteproblem

Halteproblem: Für einen gegebenen Algorithmus, gelangt dieser bei seiner Abarbeitung zu einem Ende und hält?

- Algorithmen können durch Turingmaschinen durchgeführt werden
- Turingmaschinen können durch sogenannte universelle Turingmaschinen simuliert werden
  - Wenn eine Turingmaschine T kodiert ist mit dem Wort w, dann ist $T_w : X \longrightarrow X$ eine Funktion, die Eingaben auf die Ausgabe der Turingmaschine T mappt.
  - Also mit $X = \{1, 0\}$ gibt z.B. $T_w(100101) = 001$ genau dann zurück, wenn, sofern man 100101 als Eingabe an die Turingmaschine mit der Kodierung w gibt, diese hält und als Ausgabe 001 erzeugt.

Halteproblem

Halteproblem: Für einen gegebenen Algorithmus, gelangt dieser bei seiner Abarbeitung zu einem Ende und hält?

- Algorithmen können durch Turingmaschinen durchgeführt werden
- Turingmaschinen können durch sogenannte universelle Turingmaschinen simuliert werden
  - Wenn eine Turingmaschine T kodiert ist mit dem Wort w, dann ist $T_w : X \to X$ eine Funktion, die Eingaben auf die Ausgabe der Turingmaschine T mappt.
  - Also mit $X = \{1, 0\}$ gibt z.B. $T_w(100101) = 001$ genau dann zurück, wenn, sofern man 100101 als Eingabe an die Turingmaschine mit der Kodierung w gibt, diese hält und als Ausgabe 001 erzeugt.

Dann lässt sich das Halteproblem auch als Sprache formulieren:

$$H = \{w \in A^* : w \text{ ist eine TM-Codierung und } T_w(w) \text{ hält.}\}$$

bzw. als allgemeinerer Fall:

$$\hat{H} = \{(w, x) \in A^* \times A^* : w \text{ ist eine TM-Codierung und } T_w(x) \text{ hält.}\}$$

Halteproblem

Das Halteproblem ist unentscheidbar

Halteproblem

Das Halteproblem ist unentscheidbar, dh. es gibt keine Turingmaschine, die H entscheidet.

Busy Beaver

Busy Beaver TM ist eine Turingmaschine mit n Zuständen, die möglichst viele Einsen auf das Band schreibt und hält.

Busy Beaver

Busy Beaver TM ist eine Turingmaschine mit n Zuständen, die möglichst viele Einsen auf das Band schreibt und hält.

- Also nicht einfach ewig Einsen aufschreibt und nie aufhört.

Busy Beaver

Karlsruher Institut für Technologie

Busy Beaver TM ist eine Turingmaschine mit n Zuständen, die möglichst viele Einsen auf das Band schreibt und hält.

- Also nicht einfach ewig Einsen aufschreibt und nie aufhört.

Busy Beaver Problem: Für eine gegebene Turingmaschine mit n Zuständen, die Einsen aufschreibt und hält: Schreibt sie auch maximal viele Einsen auf?

Busy Beaver

Karlsruher Institut für Technologie

Busy Beaver TM ist eine Turingmaschine mit n Zuständen, die möglichst viele Einsen auf das Band schreibt und hält.

- Also nicht einfach ewig Einsen aufschreibt und nie aufhört.

Busy Beaver Problem: Für eine gegebene Turingmaschine mit n Zuständen, die Einsen aufschreibt und hält: Schreibt sie auch maximal viele Einsen auf?

Das Busy Beaver Problem ist nicht entscheidbar, bzw. die Busy Beaver Funktion $bb(n)$, die definiert, wieviele einsen von einer Busy Beaver TM maximal geschrieben werden können, ist nicht berechenbar.

Busy Beaver

Busy Beaver TM ist eine Turingmaschine mit n Zuständen, die möglichst viele Einsen auf das Band schreibt und hält.

- Also nicht einfach ewig Einsen aufschreibt und nie aufhört.

Busy Beaver Problem: Für eine gegebene Turingmaschine mit n Zuständen, die Einsen aufschreibt und hält: Schreibt sie auch maximal viele Einsen auf?

Das Busy Beaver Problem ist nicht entscheidbar, bzw. die Busy Beaver Funktion $bb(n)$, die definiert, wieviele einsen von einer Busy Beaver TM maximal geschrieben werden können, ist nicht berechenbar.

Beispielwerte von bb:
$bb(1) = 1$

Busy Beaver

Busy Beaver TM ist eine Turingmaschine mit n Zuständen, die möglichst viele Einsen auf das Band schreibt und hält.

- Also nicht einfach ewig Einsen aufschreibt und nie aufhört.

Busy Beaver Problem: Für eine gegebene Turingmaschine mit n Zuständen, die Einsen aufschreibt und hält: Schreibt sie auch maximal viele Einsen auf?

Das Busy Beaver Problem ist nicht entscheidbar, bzw. die Busy Beaver Funktion $bb(n)$, die definiert, wieviele einsen von einer Busy Beaver TM maximal geschrieben werden können, ist nicht berechenbar.

Beispielwerte von bb:
$bb(1) = 1, bb(2) = 4$

Busy Beaver

Busy Beaver TM ist eine Turingmaschine mit n Zuständen, die möglichst viele Einsen auf das Band schreibt und hält.

- Also nicht einfach ewig Einsen aufschreibt und nie aufhört.

Busy Beaver Problem: Für eine gegebene Turingmaschine mit n Zuständen, die Einsen aufschreibt und hält: Schreibt sie auch maximal viele Einsen auf?

Das Busy Beaver Problem ist nicht entscheidbar, bzw. die Busy Beaver Funktion $bb(n)$, die definiert, wieviele einsen von einer Busy Beaver TM maximal geschrieben werden können, ist nicht berechenbar.

Beispielwerte von bb:
$bb(1) = 1, bb(2) = 4, bb(5) \geq 4098$

Busy Beaver

Busy Beaver TM ist eine Turingmaschine mit n Zuständen, die möglichst viele Einsen auf das Band schreibt und hält.

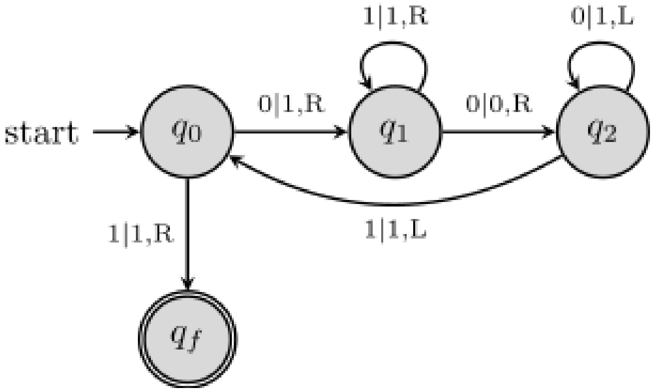
- Also nicht einfach ewig Einsen aufschreibt und nie aufhört.

Busy Beaver Problem: Für eine gegebene Turingmaschine mit n Zuständen, die Einsen aufschreibt und hält: Schreibt sie auch maximal viele Einsen auf?

Das Busy Beaver Problem ist nicht entscheidbar, bzw. die Busy Beaver Funktion $bb(n)$, die definiert, wieviele einsen von einer Busy Beaver TM maximal geschrieben werden können, ist nicht berechenbar.

Beispielwerte von bb:
$bb(1) = 1, bb(2) = 4, bb(5) \geq 4098, bb(6) > 10^{18276}$.

Busy Beaver für $n = 3$



Regulärer Ausdruck

Regulärer Ausdruck

Regulärer Ausdruck

Regulärer Ausdruck

- $Z = \{|, (,), *, \emptyset\}$ Alphabet von “Hilfssymbolen”

Regulärer Ausdruck

Regulärer Ausdruck

- $Z = \{|, (,), *, \emptyset\}$ Alphabet von “Hilfssymbolen”
- Alphabet $A \setminus Z$

Regulärer Ausdruck

Regulärer Ausdruck

- $Z = \{|, (,), *, \emptyset\}$ Alphabet von “Hilfssymbolen”
- Alphabet $A \setminus Z$
- Ein **regulärer Ausdruck** (RA) über A ist eine Zeichenfolge über dem Alphabet $A \cup Z$, die gewissen Vorschriften genügt.

Regulärer Ausdruck

Regulärer Ausdruck

- $Z = \{|, (,), *, \emptyset\}$ Alphabet von “Hilfssymbolen”
- Alphabet $A \setminus Z$
- Ein **regulärer Ausdruck** (RA) über A ist eine Zeichenfolge über dem Alphabet $A \cup Z$, die gewissen Vorschriften genügt.
- Vorschriften

Regulärer Ausdruck

Regulärer Ausdruck

- $Z = \{|, (,), *, \emptyset\}$ Alphabet von "Hilfssymbolen"
- Alphabet $A \setminus Z$
- Ein **regulärer Ausdruck** (RA) über A ist eine Zeichenfolge über dem Alphabet $A \cup Z$, die gewissen Vorschriften genügt.
- Vorschriften
  - $\emptyset$ ist ein RA

Regulärer Ausdruck

Regulärer Ausdruck

- $Z = \{|, (,), *, \emptyset\}$ Alphabet von "Hilfssymbolen"
- Alphabet $A \setminus Z$
- Ein **regulärer Ausdruck** (RA) über A ist eine Zeichenfolge über dem Alphabet $A \cup Z$, die gewissen Vorschriften genügt.
- Vorschriften
  - $\emptyset$ ist ein RA
  - Für jedes $x \in A$ ist x ein RA

Regulärer Ausdruck

Regulärer Ausdruck

- $Z = \{|, (,), *, \emptyset\}$ Alphabet von "Hilfssymbolen"
- Alphabet $A \setminus Z$
- Ein **regulärer Ausdruck** (RA) über A ist eine Zeichenfolge über dem Alphabet $A \cup Z$, die gewissen Vorschriften genügt.
- Vorschriften
  - $\emptyset$ ist ein RA
  - Für jedes $x \in A$ ist x ein RA
  - Wenn R_1 und R_2 RA sind, dann auch $(R_1|R_2)$ und $(R_1 R_2)$

Regulärer Ausdruck

Regulärer Ausdruck

- $Z = \{|, (,), *, \emptyset\}$ Alphabet von "Hilfssymbolen"
- Alphabet $A \setminus Z$
- Ein **regulärer Ausdruck** (RA) über A ist eine Zeichenfolge über dem Alphabet $A \cup Z$, die gewissen Vorschriften genügt.
- Vorschriften
  - $\emptyset$ ist ein RA
  - Für jedes $x \in A$ ist x ein RA
  - Wenn R_1 und R_2 RA sind, dann auch $(R_1|R_2)$ und (R_1R_2)
  - Wenn R ein RA ist, dann auch $(R*)$

Klammerregeln

Klammerregeln

Karlsruher Institut für Technologie

- "Stern- vor Punktrechnung"

Klammerregeln

- “Stern- vor Punktrechnung”
- “Punkt- vor Strichrechnung”

Klammerregeln

Karlsruher Institut für Technologie

- “Stern- vor Punktrechnung”
- “Punkt- vor Strichrechnung”

$\rightarrow$ $R_1 | R_2 R_3 *$ Kurzform für $(R_1 | (R_2 (R_3 *)))$

Klammerregeln

- "Stern- vor Punktrechnung"
- "Punkt- vor Strichrechnung"

$\rightarrow$ $R_1 | R_2 R_3 *$ Kurzform für $(R_1 | (R_2 (R_3 *)))$

- Bei mehreren gleichen Operatoren ohne Klammern links geklammert

Klammerregeln

- "Stern- vor Punktrechnung"
- "Punkt- vor Strichrechnung"

$\rightarrow$ $R_1|R_2R_3*$ Kurzform für $(R_1|(R_2(R_3*)))$

- Bei mehreren gleichen Operatoren ohne Klammern links geklammert

$\rightarrow$ $R_1|R_2|R_3$ Kurzform für $((R_1|R_2)|R_3)$

Klammerregeln

- "Stern- vor Punktrechnung"
- "Punkt- vor Strichrechnung"

$\rightarrow$ $R_1 | R_2 R_3 *$ Kurzform für $(R_1 | (R_2 (R_3 *)))$

- Bei mehreren gleichen Operatoren ohne Klammern links geklammert

$\rightarrow$ $R_1 | R_2 | R_3$ Kurzform für $((R_1 | R_2) | R_3)$

Aufgabe

Entferne so viele Klammern wie möglich, ohne die Bedeutung des RA zu verändern.

- $(((((ab)b)*)*)|(\emptyset*))$

Klammerregeln

- "Stern- vor Punktrechnung"
- "Punkt- vor Strichrechnung"

$\rightarrow$ $R_1|R_2R_3*$ Kurzform für $(R_1|(R_2(R_3*)))$

- Bei mehreren gleichen Operatoren ohne Klammern links geklammert

$\rightarrow$ $R_1|R_2|R_3$ Kurzform für $((R_1|R_2)|R_3)$

Aufgabe

Entferne so viele Klammern wie möglich, ohne die Bedeutung des RA zu verändern.

- $(((((ab)b)*)*)|(\emptyset*)) \longrightarrow (abb)**|\emptyset*$

Klammerregeln

- "Stern- vor Punktrechnung"
- "Punkt- vor Strichrechnung"

$\rightarrow$ $R_1|R_2R_3*$ Kurzform für $(R_1|(R_2(R_3*)))$

- Bei mehreren gleichen Operatoren ohne Klammern links geklammert

$\rightarrow$ $R_1|R_2|R_3$ Kurzform für $((R_1|R_2)|R_3)$

Aufgabe

Entferne so viele Klammern wie möglich, ohne die Bedeutung des RA zu verändern.

- $(((((ab)b)*)*)|(\emptyset*)) \longrightarrow (abb)**|\emptyset*$
- $((a(a|b))|b)$

Klammerregeln

- “Stern- vor Punktrechnung”
- “Punkt- vor Strichrechnung”

$\rightarrow$ $R_1|R_2R_3*$ Kurzform für $(R_1|(R_2(R_3*)))$

- Bei mehreren gleichen Operatoren ohne Klammern links geklammert

$\rightarrow$ $R_1|R_2|R_3$ Kurzform für $((R_1|R_2)|R_3)$

Aufgabe

Entferne so viele Klammern wie möglich, ohne die Bedeutung des RA zu verändern.

- $(((((ab)b)*)*)|(\emptyset*)) \longrightarrow (abb)**|\emptyset*$
- $((a(a|b))|b) \longrightarrow a(a|b)|b$

Alternative Definition

Wir können die Syntax von regulären Ausdrücken auch über eine kontextfreie Grammatik definieren.

Aufgabe

Vervollständigt die folgende Grammatik.

$G = (\{R\}, \{|, (,), *, \emptyset\} \cup A, R, P)$
mit $P = \{R \to \emptyset, R \to$

Alternative Definition

Wir können die Syntax von regulären Ausdrücken auch über eine kontextfreie Grammatik definieren.

Aufgabe

Vervollständigt die folgende Grammatik.

$G = (\{R\}, \{|, (,), *, \emptyset\} \cup A, R, P)$
mit $P = \{R \to \emptyset, R \to x$ (mit $x \in A$),
$R \to (R|R), R \to (RR)$,
$R \to (R*)$
$R \to \varepsilon\}$

Alternative Definition

Wir können die Syntax von regulären Ausdrücken auch über eine kontextfreie Grammatik definieren.

Aufgabe

Vervollständigt die folgende Grammatik.

$G = (\{R\}, \{|, (,), *, \emptyset\} \cup A, R, P)$
mit $P = \{R \to \emptyset, R \to x$ (mit $x \in A$),
$R \to (R|R), R \to (RR)$,
$R \to (R*)$
$R \to \varepsilon\}$

Wieso brauchen wir ε?

Durch R beschriebene Sprache

Notation

- Spitze Klammern $\langle , \rangle$

Durch R beschriebene Sprache

Notation

- Spitze Klammern $\langle , \rangle$

Regeln

- $\langle \emptyset \rangle =$

Durch R beschriebene Sprache

Karlsruher Institut für Technologie

Notation

- Spitze Klammern $\langle, \rangle$

Regeln

- $\langle \emptyset \rangle = \{\}$

Durch R beschriebene Sprache

Notation

- Spitze Klammern $\langle , \rangle$

Regeln

- $\langle \emptyset \rangle = \{\}$
- $\langle x \rangle =$

Durch R beschriebene Sprache

Notation

- Spitze Klammern $\langle , \rangle$

Regeln

- $\langle \emptyset \rangle = \{\}$
- $\langle x \rangle = \{x\}$ für jedes $x \in A$

Durch R beschriebene Sprache

Karlsruher Institut für Technologie

Notation

- Spitze Klammern $\langle , \rangle$

Regeln

- $\langle \emptyset \rangle = \{\}$
- $\langle x \rangle = \{x\}$ für jedes $x \in A$
- $\langle R_1 | R_2 \rangle =$

Durch R beschriebene Sprache

Karlsruher Institut für Technologie

Notation

- Spitze Klammern $\langle , \rangle$

Regeln

- $\langle \emptyset \rangle = \{\}$
- $\langle x \rangle = \{x\}$ für jedes $x \in A$
- $\langle R_1 | R_2 \rangle = \langle R_1 \rangle \cup \langle R_2 \rangle$

Durch R beschriebene Sprache

Notation

- Spitze Klammern $\langle , \rangle$

Regeln

- $\langle \emptyset \rangle = \{\}$
- $\langle x \rangle = \{x\}$ für jedes $x \in A$
- $\langle R_1 | R_2 \rangle = \langle R_1 \rangle \cup \langle R_2 \rangle$
- $\langle R_1 R_1 \rangle =$

Durch R beschriebene Sprache

Notation

- Spitze Klammern $\langle , \rangle$

Regeln

- $\langle \emptyset \rangle = \{\}$
- $\langle x \rangle = \{x\}$ für jedes $x \in A$
- $\langle R_1 | R_2 \rangle = \langle R_1 \rangle \cup \langle R_2 \rangle$
- $\langle R_1 R_1 \rangle = \langle R_1 \rangle \cdot \langle R_2 \rangle$

Durch R beschriebene Sprache

Notation

- Spitze Klammern $\langle , \rangle$

Regeln

- $\langle \emptyset \rangle = \{\}$
- $\langle x \rangle = \{x\}$ für jedes $x \in A$
- $\langle R_1 | R_2 \rangle = \langle R_1 \rangle \cup \langle R_2 \rangle$
- $\langle R_1 R_1 \rangle = \langle R_1 \rangle \cdot \langle R_2 \rangle$
- $\langle R* \rangle =$

Durch R beschriebene Sprache

Notation

- Spitze Klammern $\langle , \rangle$

Regeln

- $\langle \emptyset \rangle = \{\}$
- $\langle x \rangle = \{x\}$ für jedes $x \in A$
- $\langle R_1 | R_2 \rangle = \langle R_1 \rangle \cup \langle R_2 \rangle$
- $\langle R_1 R_1 \rangle = \langle R_1 \rangle \cdot \langle R_2 \rangle$
- $\langle R* \rangle = \langle R \rangle *$

Charakterisierung regulärer Sprachen

Satz

Für jede formale Sprache L sind äquivalent:

1. L kann von einem endlichen Akzeptor erkannt werden.
2. L kann durch einen regulären Ausdruck beschrieben werden
3. L kann von einer rechtslinearen Grammatik erzeugt werden.

Solche Sprachen heißten regulär.

Anwendung von regulären Ausdrücken

Zum selbst probieren:
http://regexr.com/

Achtung: Reguläre Ausdrücke in praktischer Programmierung funktionieren zwar ähnlich, haben aber eine andere Syntax und können teils mehr!

Rechtslineare Grammatiken

Definition

Eine rechtslineare Grammatik ist eine reguläre Grammatik $G = (N, T, S, P)$ mit der Einschränkung, dass alle Produktionen die folgende Form haben:

- $X \longrightarrow w$ mit $w \in T^*$ oder
- $x \longrightarrow wY$ mit $w \in T^*$, $Y \in N$

Aufgabe zu rechtslinearen Grammatiken

Gebe zu $L = \{w \in \{0,1\}^* | \exists k \in \mathbb{N}_0 : Num_2(w) = 2^k + 1\}$ jeweils einen regulären Ausdruck R und eine rechtslineare Grammatik G an, sodass $L = \langle R \rangle = L(G)$ gilt.

Aufgabe zu rechtslinearen Grammatiken

Gebe zu $L = \{w \in \{0,1\}^* | \exists k \in \mathbb{N}_0 : Num_2(w) = 2^k + 1\}$ jeweils einen regulären Ausdruck R und eine rechtslineare Grammatik G an, sodass $L = \langle R \rangle = L(G)$ gilt.

Lösung

- $R = (0 * 10)|(0 * 1(0) * 1) =$

Aufgabe zu rechtslinearen Grammatiken

Gebe zu $L = \{w \in \{0,1\}^* | \exists k \in \mathbb{N}_0 : Num_2(w) = 2^k + 1\}$ jeweils einen regulären Ausdruck R und eine rechtslineare Grammatik G an, sodass $L = \langle R \rangle = L(G)$ gilt.

Lösung

- $R = (0 * 10)|(0 * 1(0) * 1) = 0 * 10|0 * 10 * 1$

Aufgabe zu rechtslinearen Grammatiken

Gebe zu $L = \{w \in \{0,1\}^* | \exists k \in \mathbb{N}_0 : Num_2(w) = 2^k + 1\}$ jeweils einen regulären Ausdruck R und eine rechtslineare Grammatik G an, sodass $L = \langle R \rangle = L(G)$ gilt.

Lösung

- $R = (0*10)|(0*1(0)*1) = 0*10|0*10*1$
- $G = (\{S, A\}, \{0, 1\}, S, \{S \rightarrow 0S|10|1A, A \rightarrow 0A|1\})$


That's all Folks!