Karlsruhe Institute of Technology

Grundbegriffe der Informatik - Tutorium 21

Christian Jülg

Wintersemester 2012/13

8. Januar 2013

http://gbi-tutor.blogspot.com

Übersicht

Karlsruhe Institute of Technology

Aufwachen

Aufgabenblatt 9

Aufgabenblatt 10

Algorithmen-Effizienz

Master-Theorem

Endliche Automaten

Abschluss

Übersicht

Zum Warmwerden...

Algorithmen-Effizienz...

1. ... wird häufig in Abhängigkeit der Eingeabelänge angegeben.
2. ... ist unabhängig von der Struktur der eingegebenen Daten.
3. ... muss für jede Rechenmaschine einzeln ermittelt werden.

Das O-Kalkül ...

1. ... eignet sich gut um einen Mindestaufwand anzugeben.
2. ... ist unabhängig von einfachen Faktoren.
3. ... beschreibt eine Menge von Funktionen.

Das Θ-Kalkül...

1. ... gibt einen „Korridor“ an, den der Algorithmus nie verlässt.
2. ... $\Theta(f(n))$ entält alle Funktionen, die auch in $O(f(n))$ enthalten sind.
3. ... ist reflexiv (Es gilt: $f(n) \in \Theta(f(n))$).

Zum Warmwerden...

Algorithmen-Effizienz...

1. ... wird häufig in Abhängigkeit der Eingeabelänge angegeben.
2. ... ist unabhängig von der Struktur der eingegebenen Daten.
3. ... muss für jede Rechenmaschine einzeln ermittelt werden.

Das O-Kalkül ...

1. ... eignet sich gut um einen Mindestaufwand anzugeben.
2. ... ist unabhängig von einfachen Faktoren.
3. ... beschreibt eine Menge von Funktionen.

Das Θ-Kalkül...

1. ... gibt einen „Korridor“ an, den der Algorithmus nie verlässt.
2. ... $\Theta(f(n))$ entält alle Funktionen, die auch in $O(f(n))$ enthalten sind.
3. ... ist reflexiv (Es gilt: $f(n) \in \Theta(f(n))$).

Zum Warmwerden...

Algorithmen-Effizienz...

1. ... wird häufig in Abhängigkeit der Eingeabelänge angegeben.
2. ... ist unabhängig von der Struktur der eingegebenen Daten.
3. ... muss für jede Rechenmaschine einzeln ermittelt werden.

Das O-Kalkül ...

1. ... eignet sich gut um einen Mindestaufwand anzugeben.
2. ... ist unabhängig von einfachen Faktoren.
3. ... beschreibt eine Menge von Funktionen.

Das Θ-Kalkül...

1. ... gibt einen „Korridor“ an, den der Algorithmus nie verlässt.
2. ... $\Theta(f(n))$ entält alle Funktionen, die auch in $O(f(n))$ enthalten sind.
3. ... ist reflexiv (Es gilt: $f(n) \in \Theta(f(n))$).

Zum Warmwerden...

Algorithmen-Effizienz...

1. ... wird häufig in Abhängigkeit der Eingeabelänge angegeben.
2. ... ist unabhängig von der Struktur der eingegebenen Daten.
3. ... muss für jede Rechenmaschine einzeln ermittelt werden.

Das O-Kalkül ...

1. ... eignet sich gut um einen Mindestaufwand anzugeben.
2. ... ist unabhängig von einfachen Faktoren.
3. ... beschreibt eine Menge von Funktionen.

Das Θ-Kalkül...

1. ... gibt einen „Korridor“ an, den der Algorithmus nie verlässt.
2. ... $\Theta(f(n))$ entält alle Funktionen, die auch in $O(f(n))$ enthalten sind.
3. ... ist reflexiv (Es gilt: $f(n) \in \Theta(f(n))$).

Übersicht

Aufgabenblatt 9

Blatt 9

- Abgaben: 14 / 19
- Punkte: Durchschnitt 6,1 von 20

Probleme

- 9.1: Die Laufzeitabschätzungen müssen für fast alle n gelten!

Übersicht

Aufgabenblatt 10

Blatt 10

- Abgabe: 11.01.2013 um 12:30 Uhr im Untergeschoss des Infobaus
- Punkte: maximal 20

Themen

- Rekursion
- Master-Theorem
- Endliche Automaten
- Mealy, Moore, endl. Akzeptoren

Übersicht

Aufwandsklassen

Fallunterscheidung: Aufwandsklassen

O-Kalkül	Obere Schranke, die der Algorithmus erreichen, aber nicht überschreiten kann
Ω-Kalkül	Untere Schranke und ein "Mindestaufwand", den der Algorithmus hat
θ-Kalkül	Schnittmenge der Betrachtung aus $\Omega(n)$ und $O(n)$. Es entsteht eine Art „Korridor", den der Algorithmus nie verlässt.

O-Kalkül

Karlsruhe Institute of Technology

Definition

$$O(g(n)) = \{f(n) | \exists c > 0\ \exists n_0 \in \mathbb{N}\ \forall n \geq n_0 : 0 \leq f(n) \leq c \cdot g(n)\}$$

Umgangssprachlich

$O(g(n))$ enthält alle nicht-negativen Funktionen, die höchstens so schnell wie $g(n)$ wachsen.
Dabei kümmern wir uns nicht

- darum, was am Anfang passiert ($\exists n_0 \in \mathbb{N} \ldots \forall n \geq n_0$).
- um einfache Faktoren ($\exists c \in \mathbb{R} \ldots c \cdot g(n)$).

Aufwandsklassen

Obere asymptotische Schranke
$O(g(n)) = \{f(n) \mid$
$\exists c \in \mathbb{R}^+, n_0 \in \mathbb{N} \, \forall n > n_0 : 0 \leq f(n) \leq c \cdot g(n)\}$

Untere asymptotische Schranke
$\Omega(g(n)) = \{f(n) \mid$
$\exists c \in \mathbb{R}^+, n_0 \in \mathbb{N} \, \forall n > n_0 : 0 \leq c \cdot g(n) \leq f(n)\}$

Asymptotisch scharfe Schranke
$\theta(g(n)) = \{f(n) \mid$
$\exists c_1, c_2 \in \mathbb{R}^+, n_0 \in \mathbb{N} \, \forall n > n_0 : 0 \leq c_1 \cdot g(n) \leq f(n) \leq c_2 \cdot g(n)\}$

Beachte:
Alle Kalküle geben eine **Menge** von Funktionen an. $f(n) = O(n^2)$ bedeutet also eigentlich $f(n) \in O(n^2)$!

Rechenregeln

Reflexivität

- $f(n) \in O(f(n))$
- $g(n) \in \Omega(g(n))$
- $h(n) \in \theta(h(n))$

Symmetrie

Hier gilt nur: $f(n) \in \theta(g(n)) \Leftrightarrow g(n) \in \theta(f(n))$

asymptotisches Wachstum

- $O(n^2 + n + log(n)) = O(n^2)$
- $\Omega(n^2 + n + log(n)) = \Omega(n^2) \subset \Omega(log(n))$

Beispiel

Karlsruhe Institute of Technology

Es gilt nicht:

$$f(n) \notin \theta(g(n)) \Rightarrow g(n) \in O(f(n)) \vee f(n) \in O(g(n))$$

Sucht Gegenbeispiele:

Beispiel

Karlsruhe Institute of Technology

Es gilt nicht:

$$f(n) \notin \theta(g(n)) \Rightarrow g(n) \in O(f(n)) \vee f(n) \in O(g(n))$$

Sucht Gegenbeispiele:

- $|\cos(n)| * n^2$ und

Beispiel

Karlsruhe Institute of Technology

Es gilt nicht:

$$f(n) \notin \theta(g(n)) \Rightarrow g(n) \in O(f(n)) \vee f(n) \in O(g(n))$$

Sucht Gegenbeispiele:

- $|\cos(n)| * n^2$ und n
- n und

Beispiel

Karlsruhe Institute of Technology

Es gilt nicht:

$$f(n) \notin \theta(g(n)) \Rightarrow g(n) \in O(f(n)) \vee f(n) \in O(g(n))$$

Sucht Gegenbeispiele:

- $|\cos(n)| * n^2$ und n
- n und $f(n) = n^2$ für gerade, 0 für ungerade Werte von n

Übersicht

Master-Theorem

Karlsruhe Institute of Technology

Wozu?

Manche Rekursionen passen in einen der drei Fälle des Master-Theorems und lassen sich so mit dem θ-Kalkül abschätzen

Master-Theorem

Karlsruhe Institute of Technology

Wozu?

Manche Rekursionen passen in einen der drei Fälle des Master-Theorems und lassen sich so mit dem θ-Kalkül abschätzen

Vorsicht

nicht jede Rekursion eignet sich für das Master-Theorem!

Master-Theorem

Karlsruhe Institute of Technology

Definition

Der Algorithmus hat Laufzeit $T(n) = a * T(\frac{n}{b}) + f(n)$ wobei a, b konstant

Fall 1: wenn $f(n) \in O(n^{log_b(a)-\epsilon})$ mit $\epsilon > 0$
dann $T(n) \in \theta(n^{log_b(a)})$

Fall 2: wenn $f(n) \in \theta(n^{log_b(a)})$
dann $T(n) \in \theta(n^{log_b(a)) * log(n)}$

Fall 3: wenn $f(n) \in \Omega(n^{log_b(a)+\epsilon})$ mit $\epsilon > 0$, und
$\exists d : 0 < d < 1$ und für fast alle n gilt $a * f(\frac{n}{b}) \leq d * f(n)$
dann $T(n) \in \theta(f(n))$

Master-Theorem

Karlsruhe Institute of Technology

Beispiel

Quicksort hat die Struktur

- wähle Pivot Element und teile damit Liste in zwei Teile
- Quicksort(linker Teil)
- Quicksort(rechter Teil)

Was wären hier a, b und $f(n)$?

Master-Theorem

Karlsruhe Institute of Technology

Beispiel

Quicksort hat die Struktur

- wähle Pivot Element und teile damit Liste in zwei Teile
- Quicksort(linker Teil)
- Quicksort(rechter Teil)

Was wären hier a, b und $f(n)$?
Trifft einer der drei Fälle zu? Wenn ja welcher?

Master-Theorem

Karlsruhe Institute of Technology

Beispiel

Quicksort hat die Struktur

- wähle Pivot Element und teile damit Liste in zwei Teile
- Quicksort(linker Teil)
- Quicksort(rechter Teil)

Was wären hier a, b und $f(n)$?
Trifft einer der drei Fälle zu? Wenn ja welcher?
Fall 2 passt hier. Quicksort liegt damit in $\theta(n \cdot log(n))$

Übersicht

Endlich ein Automat!

Karlsruhe Institute of Technology

Wozu?

Ein endlicher Automat ist gerade mächtig genug, um einen regulären Ausdruck zu erkennen. Der Vorteil von endlichen Automaten ist, dass sie sehr einfach zu implementieren sind.

Endlich ein Automat!

Karlsruhe Institute of Technology

Wozu?

Ein endlicher Automat ist gerade mächtig genug, um einen regulären Ausdruck zu erkennen. Der Vorteil von endlichen Automaten ist, dass sie sehr einfach zu implementieren sind.

Was braucht man?

- endliche Menge Z von Zuständen
- einen Anfangszustand $z_0 \in Z$
- ein Eingabealphabet X
- Zustandsübergangsfunktion $f : Z \times X \to Z$
- ein Ausgabealphabet Y
- eine Ausgabefunktion (abhängig vom Typ des Automaten)

Endlich ein Automat!

Karlsruhe Institute of Technology

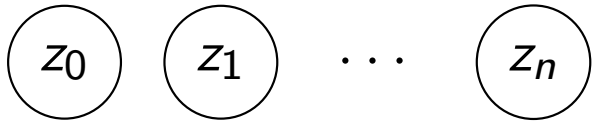
Wie arbeitet er?

Das Lesen eines Zeichens $x \in X$ führt zu einem Zustandsübergang vom aktuellen Zustand $z \in Z$ in einen neuen Zustand $z' \in Z$

- Notation: $f(z, x) = z'$
- Der Zustand läßt sich als ein **Gedächtnis** über die Vorgeschichte, also die bisher eingegebenen Zeichen, auffassen.
  Dieses ist leider nur **endlich** (endliche Menge an Zuständen!)

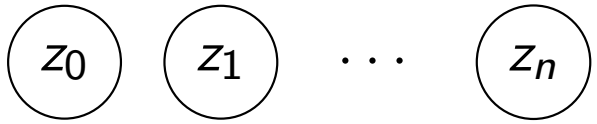
Darstellung von endlichen Automaten als Graphen

Zustandsmenge $Z = \{z_0, z_1, \ldots, z_n\}$ des endlichen Automaten lassen sich als Ecken eines Graphen auffassen

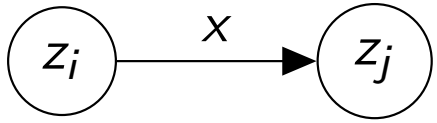


Darstellung von endlichen Automaten als Graphen

Zustandsmenge $Z = \{z_0, z_1, \ldots, z_n\}$ des endlichen Automaten lassen sich als Ecken eines Graphen auffassen

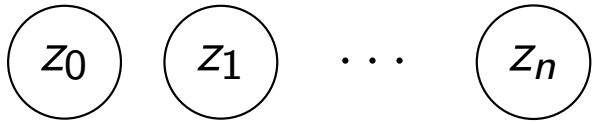


Zustandsübergänge $f(z_i, x) = z_j$ mit $x \in X$ entsprechen markierten gerichteten Kanten

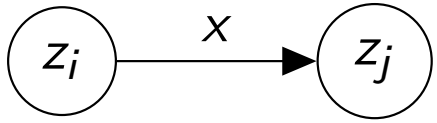


Darstellung von endlichen Automaten als Graphen

Zustandsmenge $Z = \{z_0, z_1, \ldots, z_n\}$ des endlichen Automaten lassen sich als Ecken eines Graphen auffassen



Zustandsübergänge $f(z_i, x) = z_j$ mit $x \in X$ entsprechen markierten gerichteten Kanten



Ein im endlichen Automaten erreichter Zustand z_k ist durch den Anfangszustand z_0 und die bisher eingegebene Zeichenreihe $w \in X^*$ mit $w = x_1 \ldots x_i$ bestimmt

f^* und f^{**}

f^*

Nach Eingabe des ganzen Wortes $w \in X^*$ erreichen wir den Zustand $f^* : Z \times X^* \to Z$ mit

$$f^*(z, \epsilon) = z$$
$$\forall w \in X^* : \forall x \in X : \quad f^*(z, wx) = f(f^*(z, w), x)$$

f^* und f^{**}

Karlsruhe Institute of Technology

f^*

Nach Eingabe des ganzen Wortes $w \in X^*$ erreichen wir den Zustand $f^* : Z \times X^* \to Z$ mit

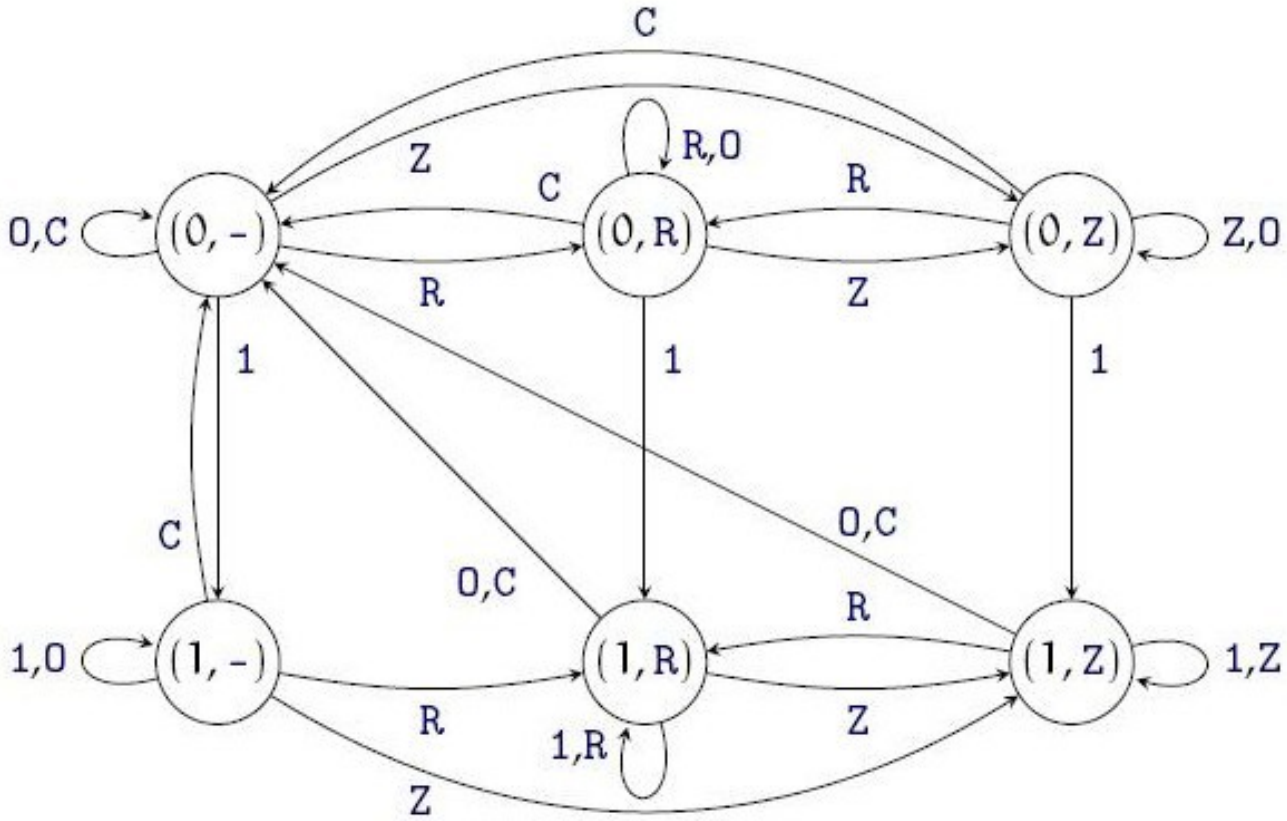
$$f^*(z, \epsilon) = z$$
$$\forall w \in X^* : \forall x \in X : \quad f^*(z, wx) = f(f^*(z, w), x)$$

f^{**}

Nach Eingabe des ganzen Wortes $w \in X^*$ haben wir die Zustände $f^{**} : Z \times X^* \to Z^*$ durchlaufen, mit

$$f^{**}(z, \epsilon) = z$$
$$\forall w \in X^* : x \in X : \quad f^{**}(z, wx) = f^{**}(z, w) f(f^*(z, w), x)$$

Ein Beispielautomat...



Was ist $f^*((0,-),R10)$? Was ist $f^{**}((0,-),R10)$?

Arten von Automaten

Es gibt zwei Arten, wie ein Automat eine Ausgabe tätigen kann. Wir unterscheiden dabei:

Arten von Automaten

Es gibt zwei Arten, wie ein Automat eine Ausgabe tätigen kann. Wir unterscheiden dabei:

Mealy-Automat

- Erzeugung einer Ausgabe bei jedem Zustandsübergang
- Ausgabefunktion $g : Z \times X \rightarrow Y^*$
- Markieren der Kanten mit $x_i | y_i$

Arten von Automaten

Es gibt zwei Arten, wie ein Automat eine Ausgabe tätigen kann. Wir unterscheiden dabei:

Mealy-Automat

- Erzeugung einer Ausgabe bei jedem Zustandsübergang
- Ausgabefunktion $g : Z \times X \to Y^*$
- Markieren der Kanten mit $x_i | y_i$

Moore-Automat

- Erzeugung einer Ausgabe bei Erreichen eines Zustands
- Ausgabefunktion $h : Z \to Y^*$

In beiden Fällen ist die Ausgabe ein Wort $y = y_0 \dots y_{n-1}$ über einem Ausgabealphabet Y.

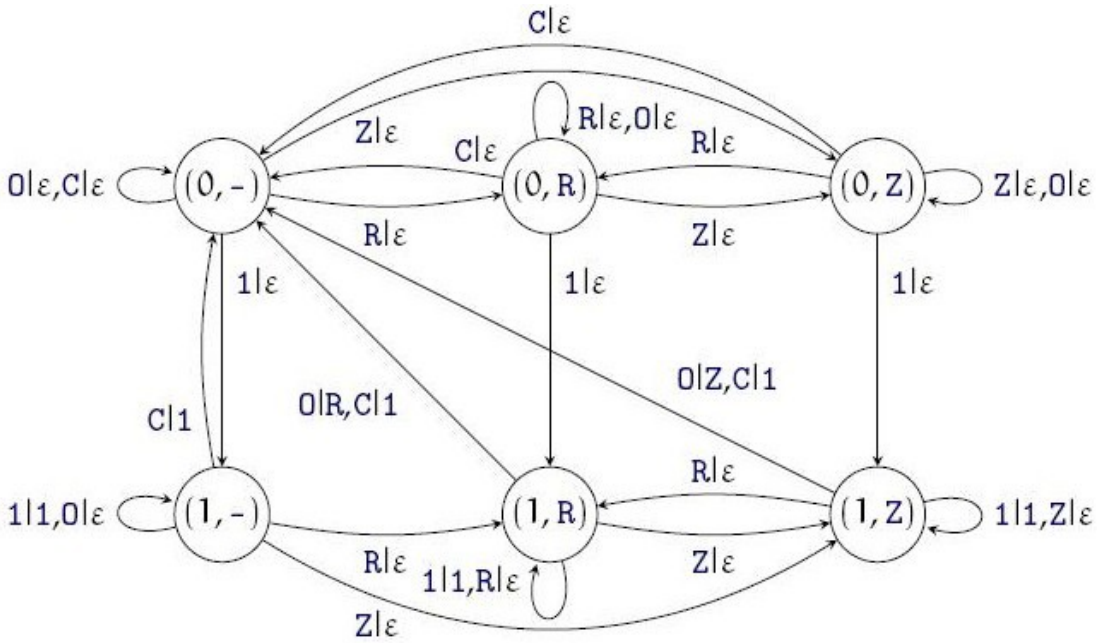
Mealy-Automat

Für die Ausgabefunktion $g : Z \times X \to Y^*$ lassen sich analog zur Zustandsübergangsfunktion $g^* : Z \times X^* \to Y^*$ und $g^{**} : Z \times X^* \to Y^*$ definieren:

$$g^*(z, \epsilon) = \epsilon$$
$$g^*(z, wx) = g(f^*(z, w), x)$$

$$g^{**}(z, \epsilon) = \epsilon$$
$$g^{**}(z, xw) = g(z, x) \cdot g^{**}(f(z, x), w)$$

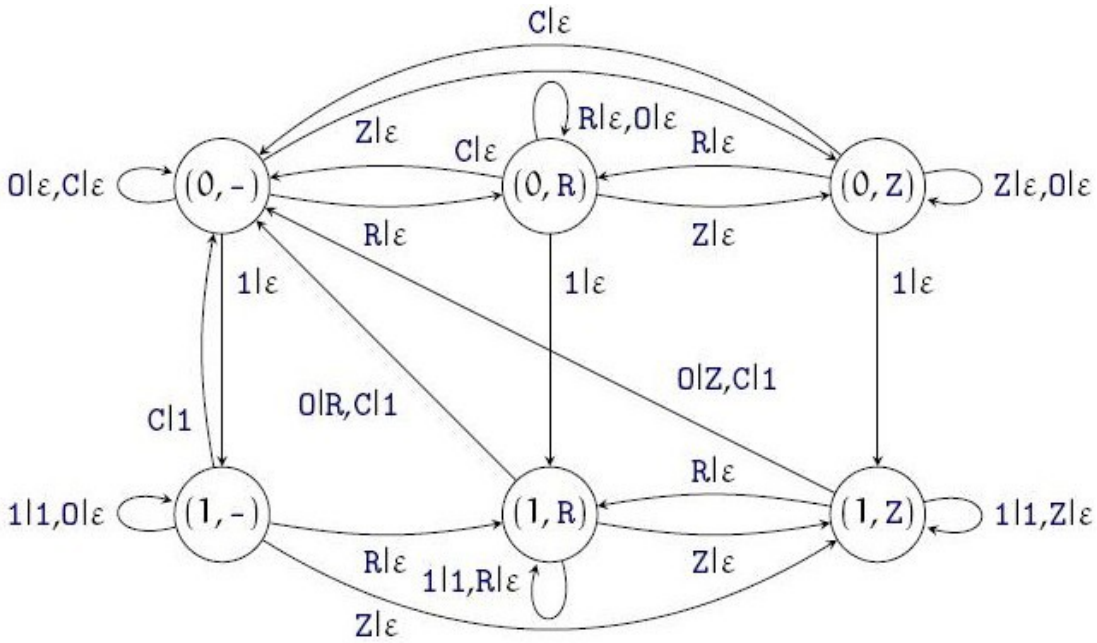
Noch ein Beispielautomat...



Was ist...

- $g^*((0,-),R1O)$?
- $g^{**}((0,-),R1O)$?
- $g^{**}((0,-),R11O)$?

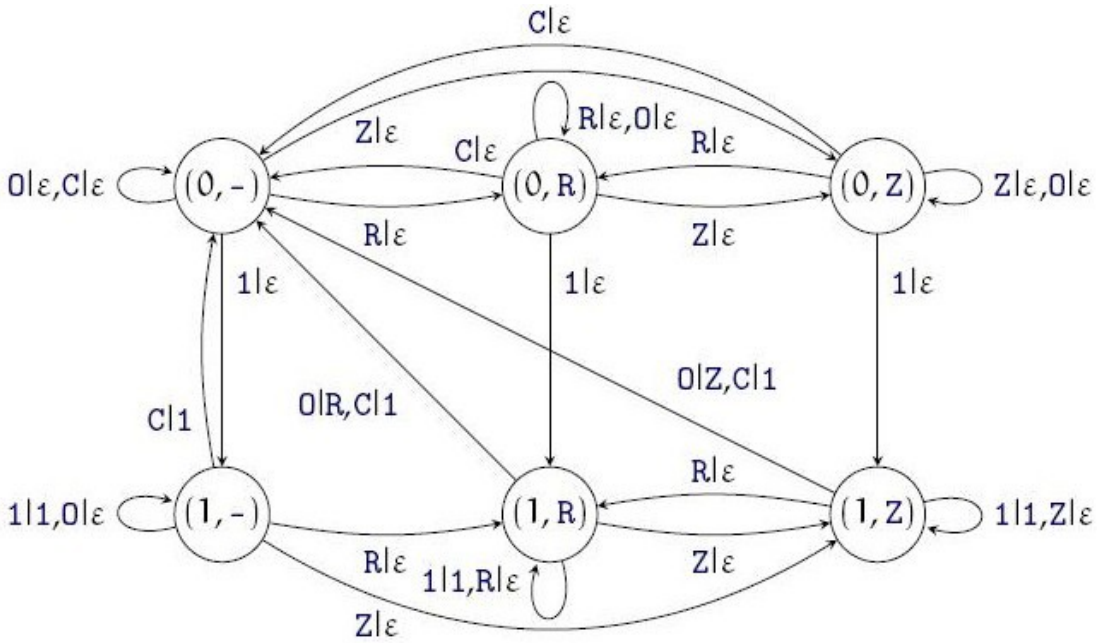
Noch ein Beispielautomat...



Was ist...

- $g^*((0,-),R1O) = R$
- $g^{**}((0,-),R1O)$
- $g^{**}((0,-),R11O)$

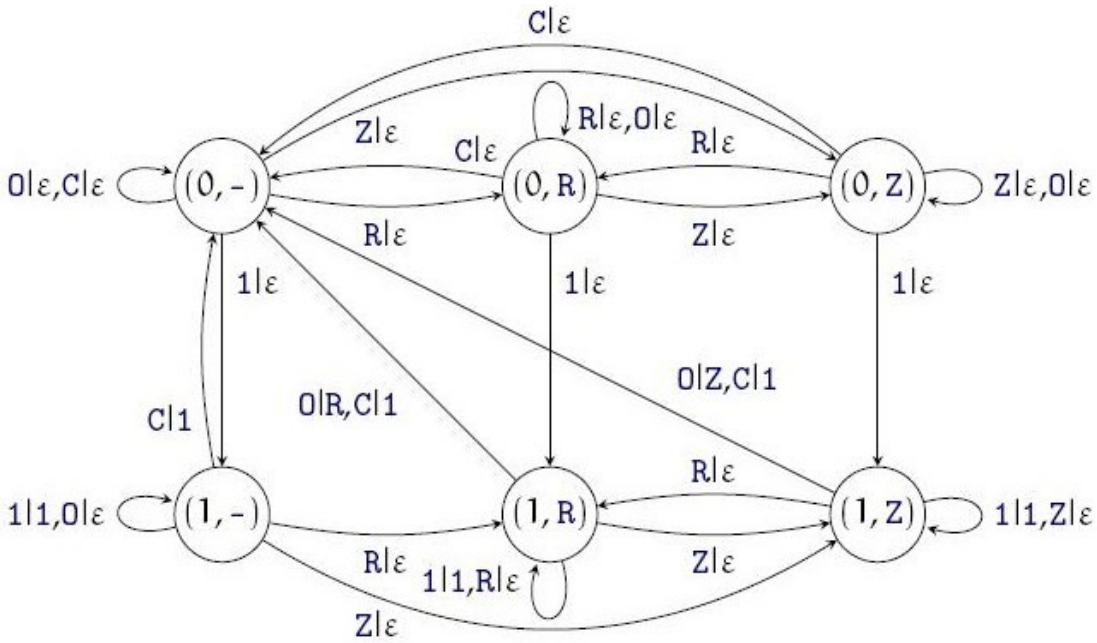
Noch ein Beispielautomat...



Was ist...

- $g^*((0,-),R1O) = R$
- $g^{**}((0,-),R1O) = R$
- $g^{**}((0,-),R11O)$

Noch ein Beispielautomat...



Was ist...

- $g^*((0,-),R1O) = R$
- $g^{**}((0,-),R1O) = R$
- $g^{**}((0,-),R11O) = 1R$

Ihr seid dran...

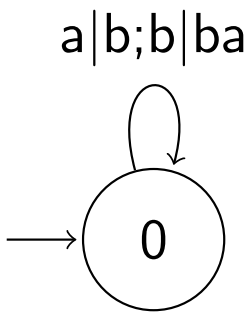
Entwickelt einen Mealy-Automaten, der...

- nur einen Zustand z hat und $X = Y = \{a, b\}$, $g(z, a) = b$ und $g(z, b) = ba$ erfüllt
  - wie sieht $w_1 = g^{**}(z, a)$ aus?
  - $w_2 = g^{**}(z, w_1)$, ... $w_{i+1} = g^{**}(z, w_i)$?
  - könnte man den Automaten mit weniger Zuständen darstellen?
- und einen weiteren Automat mit $Z = \mathbb{G}_5$, $X = \{a, b\}$, $Y = \{0, 1\}$, bei b gleicher Zustand und Ausgabe 0, bei a einen Zustand weiter und bei jedem 5.a Ausgabe 1, sonst Ausgabe 0. Was tut der Automat?

Ihr seid dran...

Entwickelt einen Mealy-Automaten, der...

- nur einen Zustand z hat und $X = Y = \{a, b\}$, $g(z, a) = b$ und $g(z, b) = ba$ erfüllt
  - wie sieht $w_1 = g^{**}(z, a)$ aus?
  - $w_2 = g^{**}(z, w_1)$, ... $w_{i+1} = g^{**}(z, w_i)$?
  - könnte man den Automaten mit weniger Zuständen darstellen?

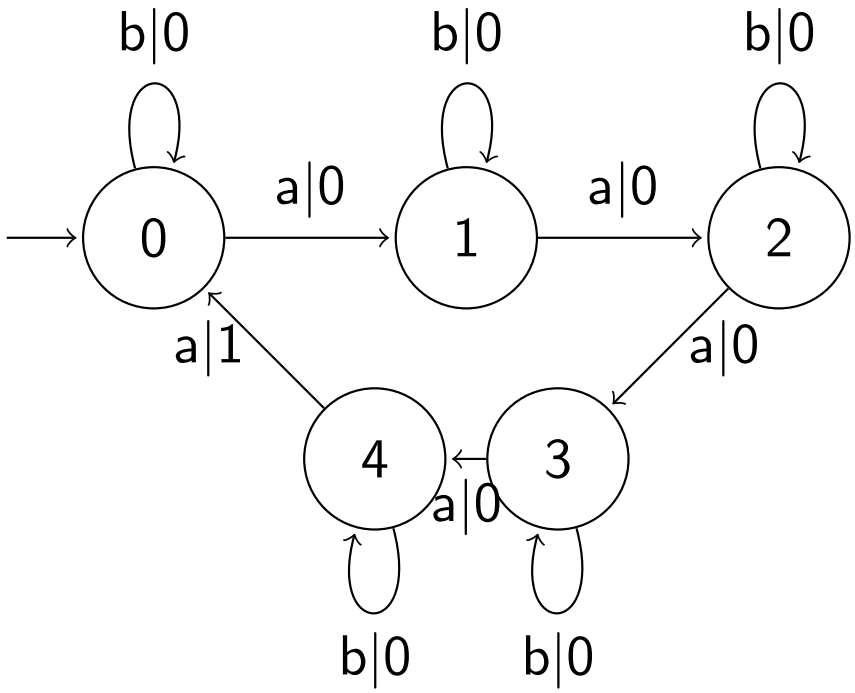


- und einen weiteren Automat mit $Z = \mathbb{G}_5$, $X = \{a, b\}$, $Y = \{0, 1\}$, bei b gleicher Zustand und Ausgabe 0, bei a einen Zustand weiter und bei jedem 5.a Ausgabe 1, sonst Ausgabe 0. Was tut der Automat?

Ihr seid dran...

Entwickelt einen Mealy-Automaten, der...

- nur einen Zustand z hat und $X = Y = \{a, b\}$, $g(z, a) = b$ und $g(z, b) = ba$ erfüllt
- und einen weiteren Automat mit $Z = \mathbb{G}_5$, $X = \{a, b\}$, $Y = \{0, 1\}$, bei b gleicher Zustand und Ausgabe 0, bei a einen Zustand weiter und bei jedem 5.a Ausgabe 1, sonst Ausgabe 0. Was tut der Automat?



Der Akzeptor

- Ist der häufigste **Spezialfall** eines Moore-Automaten
- Eine Ausgabe findet nicht bei allen Zuständen statt

Der Akzeptor

- Ist der häufigste **Spezialfall** eines Moore-Automaten
- Eine Ausgabe findet nicht bei allen Zuständen statt
- Die Zustände $F \subseteq Z$, bei denen eine Ausgabe (immer ein Bit lang) erfolgt, heißen `akzeptierende Zustände`
  Es gilt $F = \{z | h(z) = 1\}$

Der Akzeptor

- Ist der häufigste **Spezialfall** eines Moore-Automaten
- Eine Ausgabe findet nicht bei allen Zuständen statt
- Die Zustände $F \subseteq Z$, bei denen eine Ausgabe (immer ein Bit lang) erfolgt, heißen `akzeptierende Zustände`
  Es gilt $F = \{z | h(z) = 1\}$
- graphisch werden diese durch Doppelkreise angegeben

Der Akzeptor

- Ist der häufigste **Spezialfall** eines Moore-Automaten
- Eine Ausgabe findet nicht bei allen Zuständen statt
- Die Zustände $F \subseteq Z$, bei denen eine Ausgabe (immer ein Bit lang) erfolgt, heißen `akzeptierende Zustände`
  Es gilt $F = \{z | h(z) = 1\}$
- graphisch werden diese durch Doppelkreise angegeben

- Ein Wort $w \in X^*$ wird `akzeptiert`, wenn gilt $f^*(z_0, w) \in F$
- Die von einem Akzeptor A `akzeptierte formale Sprache` ist
  $L(A) = \{w \in X^* | f^*(z_0, w) \in F\}$

Ihr seid dran...

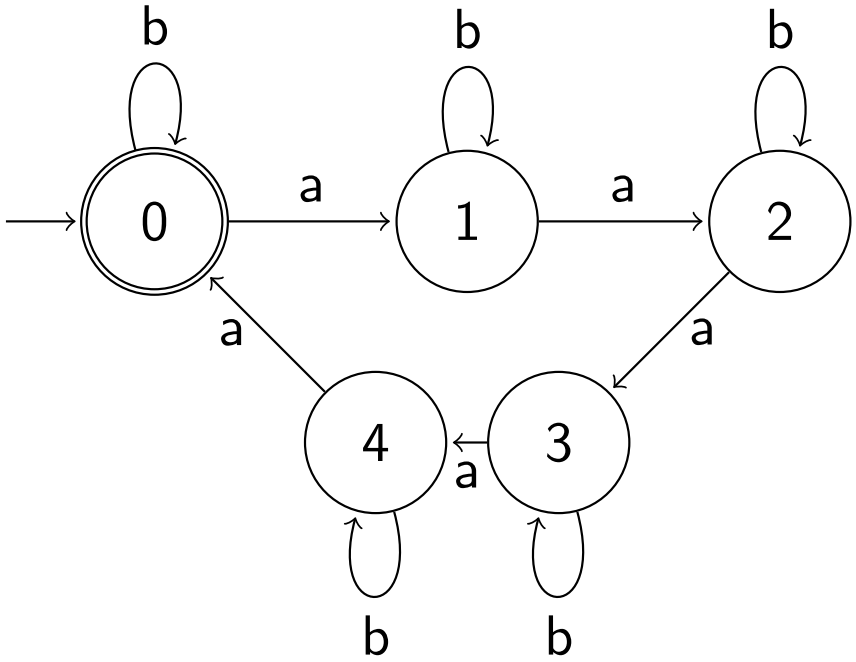
Entwickelt einen Akzeptor mit

- $X = \{a, b\}$, der alle Wörter akzeptiert, bei denen die Anzahl der a durch 5 teilbar ist. (Anzahl der b ist egal).
- $X = \{a, b\}$, der alle Wörter akzeptiert, in denen nirgends hintereinander zwei b vorkommen.

Ihr seid dran...

Entwickelt einen Akzeptor mit

- $X = \{a, b\}$, der alle Wörter akzeptiert, bei denen die Anzahl der a durch 5 teilbar ist. (Anzahl der b ist egal).

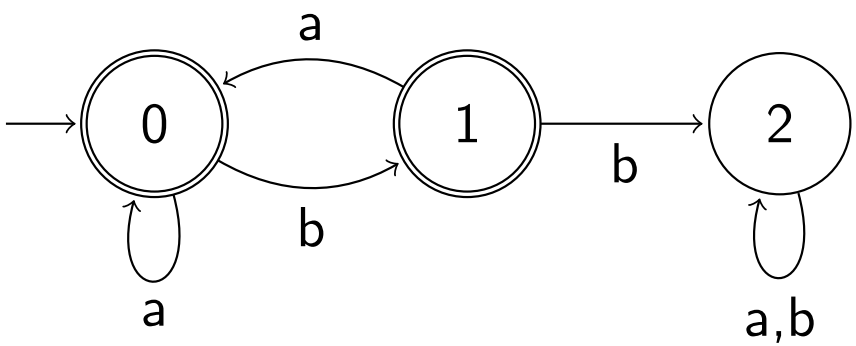


- $X = \{a, b\}$, der alle Wörter akzeptiert, in denen nirgends hintereinander zwei b vorkommen.

Ihr seid dran...

Entwickelt einen Akzeptor mit

- $X = \{a, b\}$, der alle Wörter akzeptiert, bei denen die Anzahl der a durch 5 teilbar ist. (Anzahl der b ist egal).
- $X = \{a, b\}$, der alle Wörter akzeptiert, in denen nirgends hintereinander zwei b vorkommen.



Ihr seid dran...

Entwickelt einen Akzeptor...

- der alle hexadezimalen IP-Adressen der Form $1A.BF.43.0F$ akzeptiert
- was ändert sich, wenn man auch Adressen ohne führende 0 akzeptieren möchte?
- bei Langeweile: versucht alle IP-Adressen bei denen die Blöcke aus dezimalen Zahlen zwischen 000 und 255 bestehen zu akzeptieren

Übersicht

Zum Schluss...

Was ihr nun wissen solltet!

Zum Schluss...

Karlsruhe Institute of Technology

Was ihr nun wissen solltet!

- Was ist ein (endlicher) Automat?
  Aus welchen Teilen besteht er?

Zum Schluss...

Karlsruhe Institute of Technology

Was ihr nun wissen solltet!

- Was ist ein (endlicher) Automat?
  Aus welchen Teilen besteht er?
- Worin unterscheiden sich Mealy-, Moore-Automaten und endl. Akzeptoren?

Zum Schluss...

Karlsruhe Institute of Technology

Was ihr nun wissen solltet!

- Was ist ein (endlicher) Automat?
  Aus welchen Teilen besteht er?
- Worin unterscheiden sich Mealy-, Moore-Automaten und endl. Akzeptoren?
- Wie sind $f^*, f^{**}, g^*, g^{**}, h^*, h^{**}$ definiert?

Zum Schluss...

Karlsruhe Institute of Technology

Was ihr nun wissen solltet!

- Was ist ein (endlicher) Automat?
  Aus welchen Teilen besteht er?
- Worin unterscheiden sich Mealy-, Moore-Automaten und endl. Akzeptoren?
- Wie sind $f^*, f^{**}, g^*, g^{**}, h^*, h^{**}$ definiert?
- Wie könnte man sie auch noch anders definieren?

Zum Schluss...

Karlsruhe Institute of Technology

Was ihr nun wissen solltet!

- Was ist ein (endlicher) Automat?
  Aus welchen Teilen besteht er?
- Worin unterscheiden sich Mealy-, Moore-Automaten und endl. Akzeptoren?
- Wie sind $f^*, f^{**}, g^*, g^{**}, h^*, h^{**}$ definiert?
- Wie könnte man sie auch noch anders definieren?
- Was haben Automaten mit Sprachen zu tun? Warum sind Automaten relevant?

Zum Schluss...

Karlsruhe Institute of Technology

Was ihr nun wissen solltet!

- Was ist ein (endlicher) Automat?
  Aus welchen Teilen besteht er?
- Worin unterscheiden sich Mealy-, Moore-Automaten und endl. Akzeptoren?
- Wie sind $f^*, f^{**}, g^*, g^{**}, h^*, h^{**}$ definiert?
- Wie könnte man sie auch noch anders definieren?
- Was haben Automaten mit Sprachen zu tun? Warum sind Automaten relevant?

Ihr wisst was nicht?

Stellt **jetzt** Fragen!

Ende


UND ICH HAB
MAL EINEN GEFANGEN,
DER WAR SO GROSS.
UND SIE?
WWW.NICHTLUSTIG.DE