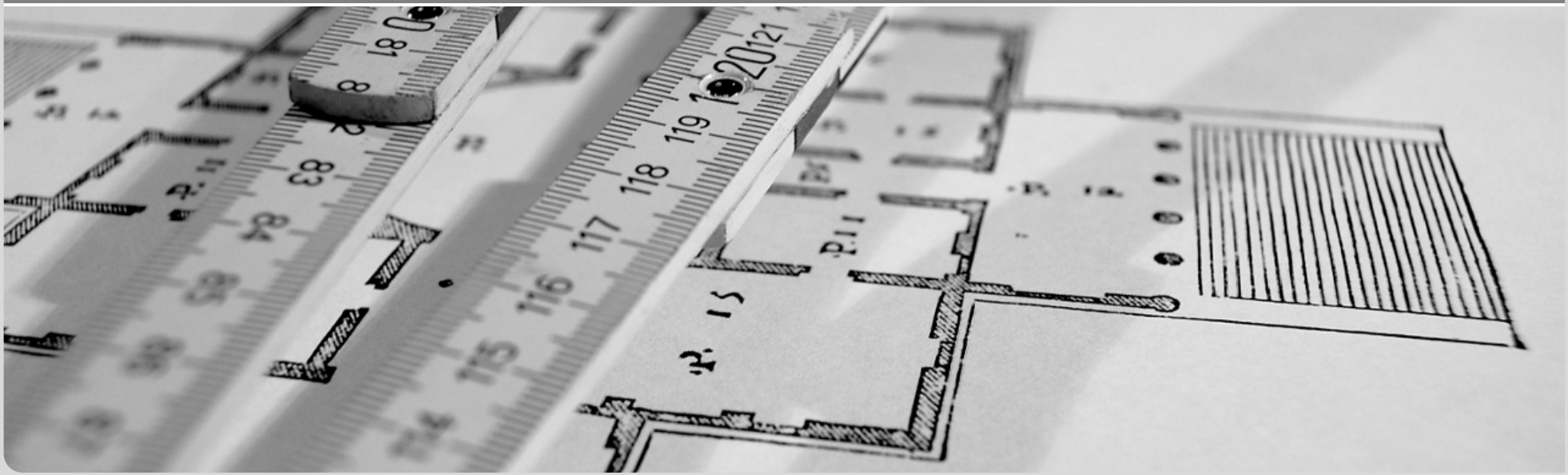
Grundbegriffe der Informatik
Tutorium 36

Termin 12 | 27.01.2017
Thassilo Helmold

KIT – Karlsruher Institut für Technologie



Inhalt

Master-Theorem

Automaten

Endliche Akzeptoren

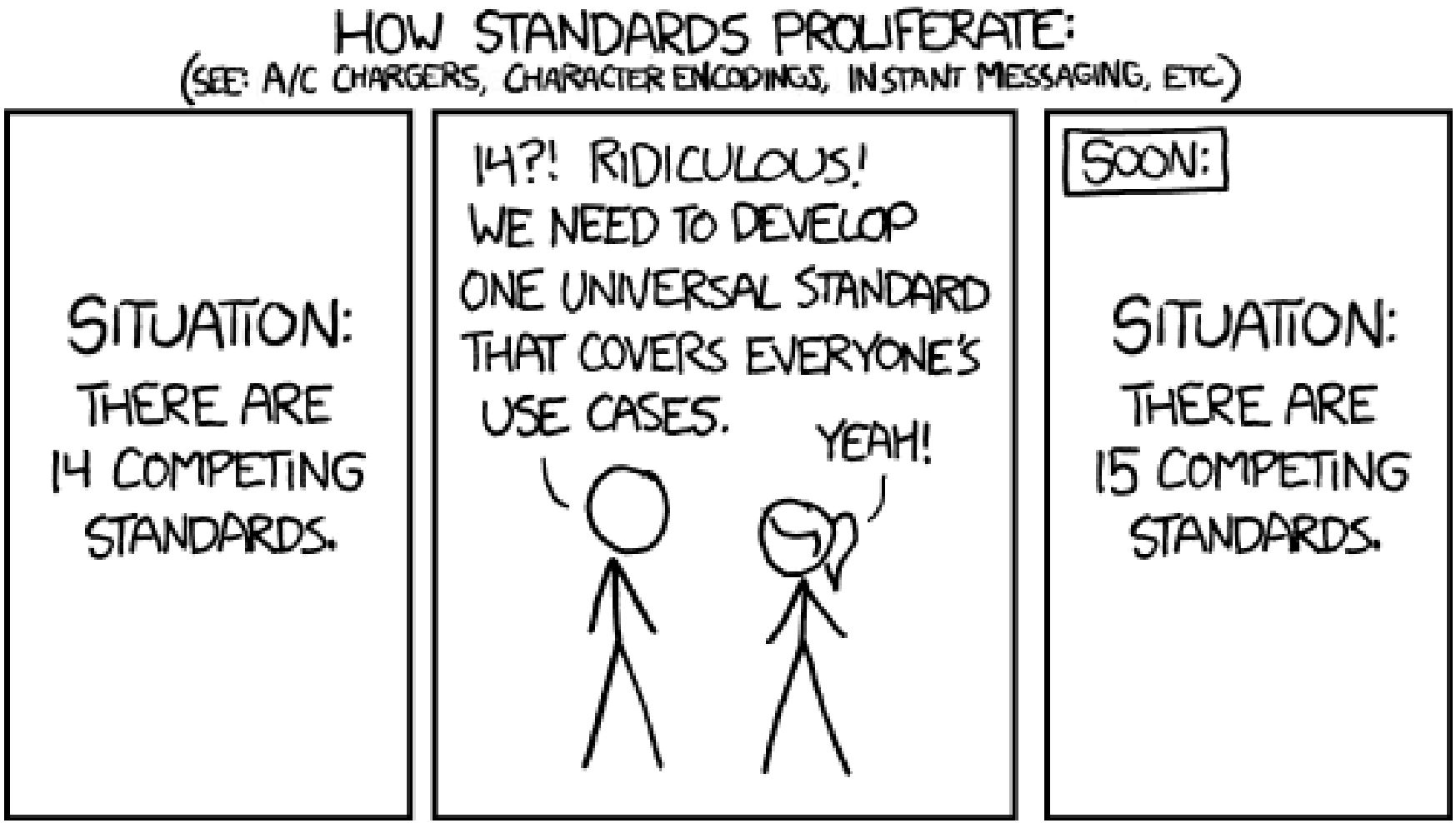


Abbildung: https://www.xkcd.com/

In the previous episode of GBI...

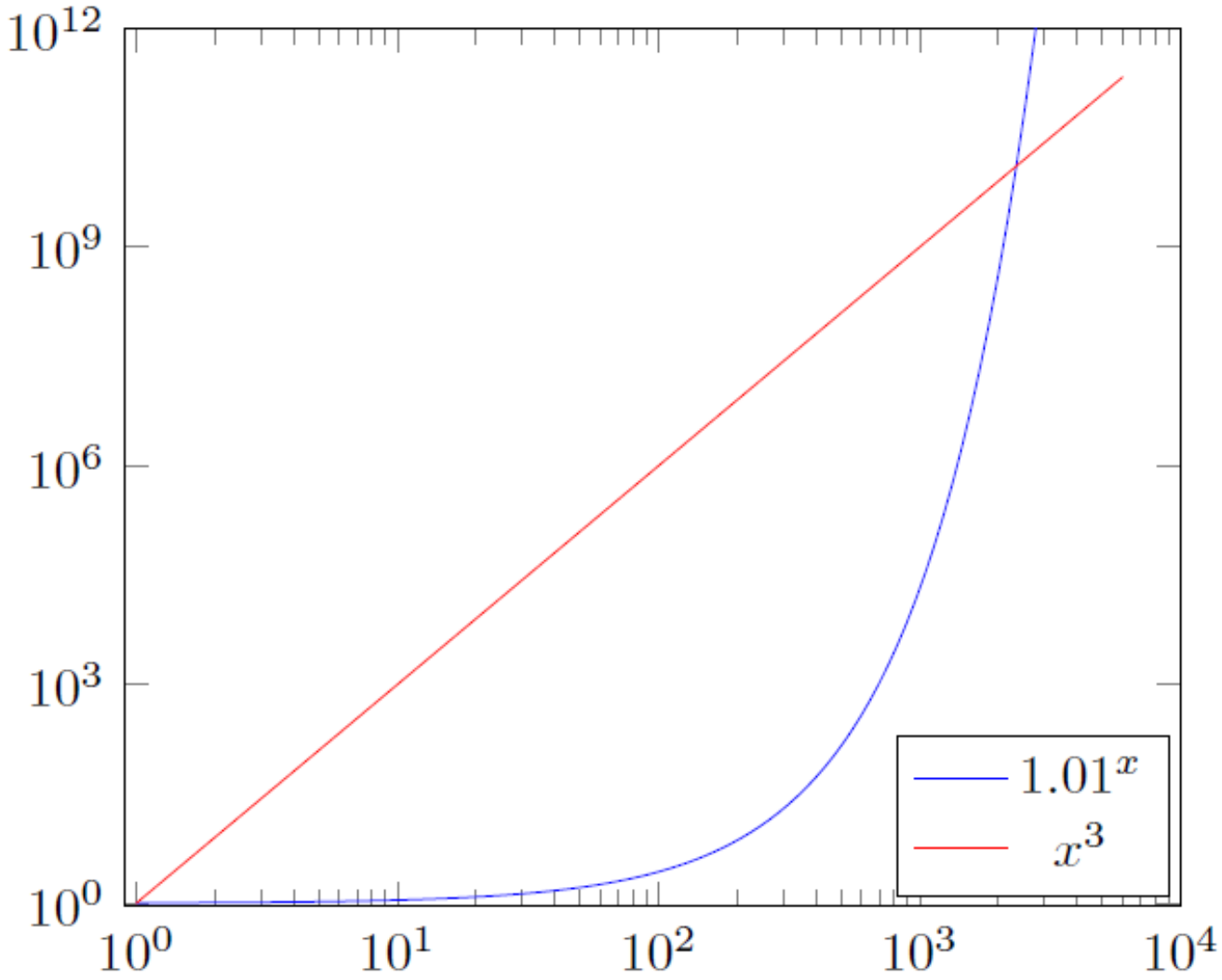
Rückblick: Wegematrix

- Adjazenzmatrizen von Graphen
- Ablesen verschiedener Eigenschaften
- Wegematrizen berechnen

Rückblick: Laufzeitabschätzungen

- Asymptotisches Wachstum
- O, Θ, Ω
- Beweisverfahren
- Rechenregeln im O-Kalkül

Rückblick



Wahr oder Falsch?

- $x^4 \in O\left(\left(x^3\right)^3\right)$

Wahr oder Falsch?

- $x^4 \in O\left((x^3)^3\right)$ W $(x^3)^3 = x^9$

Wahr oder Falsch?

- $x^4 \in O\left(\left(x^3\right)^3\right)$ W $\left(x^3\right)^3 = x^9$
- $\sqrt{n} \in \Omega\left(2^n\right)$

Wahr oder Falsch?

- $x^4 \in O\left(\left(x^3\right)^3\right)$ W $\left(x^3\right)^3 = x^9$
- $\sqrt{n} \in \Omega\left(2^n\right)$ F

Wahr oder Falsch?

- $x^4 \in O\left(\left(x^3\right)^3\right)$ W $\left(x^3\right)^3 = x^9$
- $\sqrt{n} \in \Omega\left(2^n\right)$ F
- $log_{5000} n \in \Theta\left(\log_2 n^4\right)$

Wahr oder Falsch?

- $x^4 \in O\left(\left(x^3\right)^3\right)$ W $\left(x^3\right)^3 = x^9$
- $\sqrt{n} \in \Omega\left(2^n\right)$ F
- $log_{5000} n \in \Theta\left(\log_2 n^4\right)$ W $\log_2 n^4 = 4 \cdot \log_2 n$

Wahr oder Falsch?

- $x^4 \in O\left(\left(x^3\right)^3\right)$ W $\left(x^3\right)^3 = x^9$
- $\sqrt{n} \in \Omega\left(2^n\right)$ F
- $log_{5000} n \in \Theta\left(\log_2 n^4\right)$ W $\log_2 n^4 = 4 \cdot \log_2 n$
- $O(f_1) + f_2 = O(f_1 + f_2)$

Wahr oder Falsch?

- $x^4 \in O\left(\left(x^3\right)^3\right)$ W $\left(x^3\right)^3 = x^9$
- $\sqrt{n} \in \Omega\left(2^n\right)$ F
- $log_{5000} n \in \Theta\left(\log_2 n^4\right)$ W $\log_2 n^4 = 4 \cdot \log_2 n$
- $O(f_1) + f_2 = O(f_1 + f_2)$ F
  Aber: $O(f_1) + O(f_2) = O(f_1 + f_2)$

Wahr oder Falsch?

- Für zwei Funktionen f, g gilt immer $f \preceq g$ oder $f \succeq g$

Es gibt unvergleichbare Funktionen.

$$f = \begin{cases} 1 & \text{falls } n \text{ gerade} \\ n & \text{falls } n \text{ ungerade} \end{cases}$$

$$g = \begin{cases} n & \text{falls } n \text{ gerade} \\ 1 & \text{falls } n \text{ ungerade} \end{cases}$$

Es gilt *nicht* $g \preceq f$, es gilt *nicht* $f \preceq g$ und es gilt erst recht *nicht* $f \asymp g$.

Wahr oder Falsch?

- Für zwei Funktionen f, g gilt immer $f \preceq g$ oder $f \succeq g$ F

Es gibt unvergleichbare Funktionen.

$$f = \begin{cases} 1 & \text{falls } n \text{ gerade} \\ n & \text{falls } n \text{ ungerade} \end{cases}$$

$$g = \begin{cases} n & \text{falls } n \text{ gerade} \\ 1 & \text{falls } n \text{ ungerade} \end{cases}$$

Es gilt *nicht* $g \preceq f$, es gilt *nicht* $f \preceq g$ und es gilt erst recht *nicht* $f \asymp g$.

Logaithmen encore

Berechnet:

$$\log_5(25 \cdot 125) = \log_5(25) + \log_5(125) = 2 + 3 = 5$$
$$16^{\log_4(5)} = 5^{\log_4(16)} = 5^2 = 25$$

Laufzeiten

Lineare Suche: $\Theta(n)$
Binäre Suche: $\Theta(\log n)$
Potenzmenge berechnen: $\Theta(2^n)$

Master-Theorem

Automaten

Endliche Akzeptoren

Teile und herrsche – divide and conquer

- Probleminstanz in kleinere Teile zerlegen
- Teile rekursiv nach gleichen Verfahren bearbeiten
- Aus Teilergebnissen Resultat rekonstruieren

Das Master-Theorem (einfache Form)

Situation aus dem Leben:

- Wir haben einen rekursiven Algorithmus $A(n)$ für die Problemgröße n
- Bei jedem Schritt wird das Problem um b geteilt und es ergeben sich a neue Probleminstanzen der Größe n/b
- Wie können die Laufzeit der jeweiligen „Auseinandernehmungs-“ und „Zusammenführungsschritte“ **linear** abschätzen.

Das Master-Theorem (einfache Form)

Für positive Konstanten a, b, c, d, sei $n = b^k$ für ein $k \in \mathbb{N}$.

$$r(n) = \begin{cases} a & \text{falls } n = 1 \text{ Basisfall} \\ cn + dr(n/b) & \text{falls } n > 1 \text{ teile und herrsche.} \end{cases}$$

Es gilt

$$r(n) \in \begin{cases} \Theta(n) & \text{falls } d < b \\ \Theta(n \log n) & \text{falls } d = b \\ \Theta\left(n^{\log_b d}\right) & \text{falls } d > b. \end{cases}$$

MT: Beispiele

Wir betrachten verschiedene Sortierverfahren:

Mergesort
msort(L: Liste mit |L| = n):
teile L in der Mitte auf in L_1 und L_2
sortiere L_1 und L_2
füge L_1 und L_2 in O(n) zusammen

$$r(n) = \begin{cases} 1 & n = 1 \\ c \cdot n + 2 \cdot r(n/2) & n > 1 \end{cases}$$

Nach MT, Fall 2 also: $r(n) \in \Theta(n \log n)$

MT: Beispiele

Wir betrachten verschiedene Sortierverfahren:

DoubleMergesort
dmsort(L: Liste mit |L| = n):
teile L in der Mitte auf in L_1 und L_2
sortiere L_1 und L_2 **jeweils zwei mal**
Ja, das hat natürlich keinerlei Sinn und ist rein konstruiert
füge L_1 und L_2 in O(n) zusammen

$$r(n) = \begin{cases} 1 & n = 1 \\ c \cdot n + 4 \cdot r(n/2) & n > 1 \end{cases}$$

Nach MT, Fall 3 also: $r(n) \in \Theta\left(n^2\right)$

MT: Beispiele

Wir betrachten verschiedene Sortierverfahren:

Magicsort
msort(L: Liste mit |L| = n):
teile L in der Mitte auf in L_1 und L_2
sortiere L_1
L_2 ist an dieser Stelle durch Magie bereits sortiert
füge L_1 und L_2 in O(n) zusammen

$$r(n) = \begin{cases} 1 & n = 1 \\ c \cdot n + 1 \cdot r(n/2) & n > 1 \end{cases}$$

Nach MT, Fall 1 also: $r(n) \in \Theta(n)$

ACHTUNG: Magicsort kann es nicht geben! In Algorithmen I:
Untere Schranke für vergleichsbasiertes Sortieren ist $\Omega(n \log n)$

Limitierungen des MT

Das Master-Theorem ist mächtig, aber leider längst nicht so mächtig, wie der Name es vielleicht vermuten lässt.
Dieses (einfache) MT funktioniert nur bei sehr speziellen Rekkurenzformen (auch wenn diese häufig vorkommen).
Ein erweitertes MT wurde in der VL vorgestellt, ist aber auch nicht vollständig!

Bei ungleichen Aufteilungen hilft einem das MT überhaupt nicht weiter: Berechnet man $(n+1)! = n! \cdot (n+1)$, so kann die Laufzeit der Berechnung nicht mit dem MT abgeschätzt werden.

Master-Theorem

Automaten

Endliche Akzeptoren

Endliche Automaten

Ein deterministischer endlicher Automat...

- ... besteht aus endlich vielen Zuständen
- ... ist zu jedem Zeitpunkt in *genau einem* dieser Zustände
- ... wechselt bei Eingabe *genau eines* Zeichens den Zustand in *genau einen, definierten* Folgezustand
- ... und gibt dabei ein Wort als Ausgabe aus

Die gültigen „Zustandswechsel“ sind als Zustandsübergänge definiert.

Anwendungen

- Getränkeautomaten
- Textsuche
- Textersetzung

Für Beispiele zur Textsuche und Textersetzung: Übung 12, WS 15/16

Mealy-Automat

Bei einem Mealy-Automaten findet die Ausgabe **bei den Zustandsübergängen** statt.

Definition

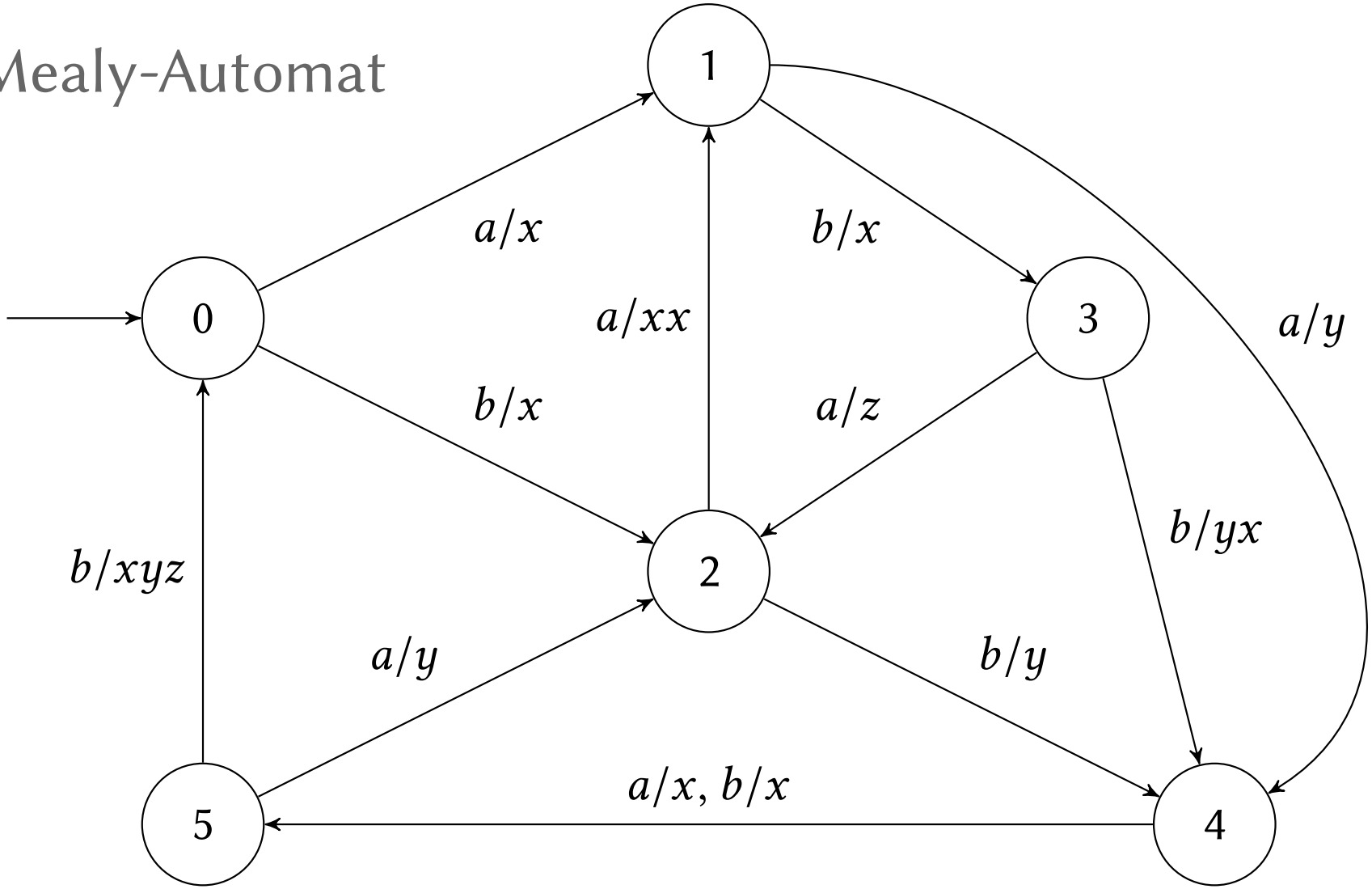
Ein Mealy-Automat $A = (Z, z_0, X, f, Y, g)$ besteht aus ...

- ... einer endlichen Zustandsmenge Z
- ... einem Startzustand z_0.
- ... einem Eingabealphabet X
- ... einer Zustandsübergangsfunktion $f : Z \times X \to Z$
- ... einem Ausgabealphabet Y
- ... einer Ausgabefunktion $g : Z \times X \to Y^*$

Graphische Darstellung

Meistens werden kleine Automaten graphisch dargestellt. (Achtung: Die formale Schreibweise wird trotzdem auch im Studium verwendet! Also auswendig lernen!)
Zustände sind Knoten und Übergänge gerichtete Kanten in einem Graphen!
Der Startzustand wird mit einem Pfeil „aus dem Nichts“ gekennzeichnet!
NICHT VERGESSEN! Das kostet Punkte! Immer!

Mealy-Automat



Eingabe: *abbab*

Ausgabe:

Eingabe eines Zeichens

Was ist die Zustandsübergangsfunktion f?
Bei Eingabe eines Symbols $x \in X$ und des aktuellen Zustands $z \in Z$ gibt uns die Zustandsübergangsfunktion an, in welchen Zustand z' der Automat dann sein wird.
Formell:

$$f(z, x) = z'$$

Eingabe eines Wortes

Was machen wir mit ganzen Wörtern?
Wir definieren uns ganz analog

Zustandsübergangsfunktion extended

$$f_*(z, \varepsilon) = z$$
$$\forall w \in X^* \; \forall x \in X : f_*(z, wx) = f(f_*(z, w), x)$$

Was macht diese Funktion?
Bei Eingabe eines Wortes w und Anfangszustand z gibt sie den Zustand $z' = f_*(z, w)$ aus, in dem der Automat enden wird.

Eingabe eines Wortes

Definieren wir nun weiter

Zustandsübergangsfunktion extended extended

$$
\begin{aligned}
f_{**}(z, \varepsilon) &= z \\
\text{und für alle } x \in X \text{ und } w \in X^* \text{ ist } f_{**}(z, xw) &= z \cdot f_{**}(f(z, x), w)
\end{aligned}
$$

Was macht diese Funktion?
Diese Funktion gibt die Reihe aller Zustände aus, die der Automat bei Eingabe des Wortes w im Startzustand z durchläuft.

Eingabe eines Wortes

Alternative Definition:

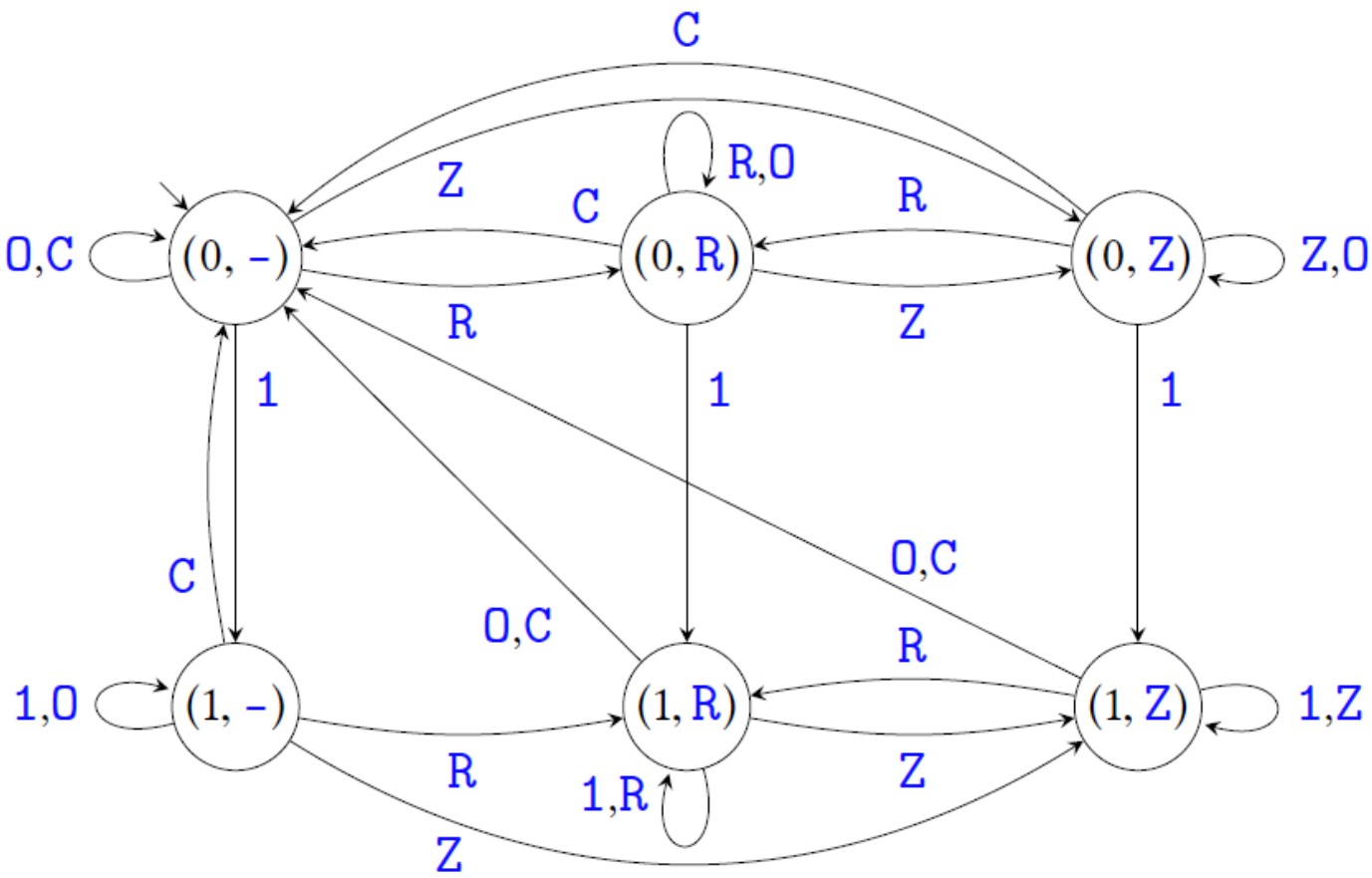
Zustandsübergangsfunktion extended extended

$$f_{**}(z, \varepsilon) = z$$
$$\forall w \in X^* \; \forall x \in X : f_{**}(z, wx) = f_{**}(z, w) \cdot f(f_*(z, w), x)$$

Beispiel: Ein Getränkeautomat

R sei reines Wasser, *Z* ist Zitronenlimonade, *O* ist OK (Getränkeausgabe), *C* ist Clear und 1 ist die Eingabe von einer Münze mit Wertigkeit 1
Wir wollen:

- Getränk wählen können
- Geld rein- und rausholen können
- und immer zurück kommen können!



Was macht $f((0,-),Z), f_*((0,-),R10), f_{**}((0,-),R10)$? Rechnerisch und graphisch.

Rechnerisch nur der 3. Fall:

$$
\begin{aligned}
f_{**}((0,-),R10) &= f_{**}((0,-),R1)) \cdot f(f_{*}((0,-),R1),0) \\
&= f_{**}((0,-),R) \cdot f(f_{*}((0,-),R),1) \\
&\quad \cdot f(f_{*}((0,-),R1),0) \\
&= f_{**}((0,-),\varepsilon) \cdot f(f_{*}((0,-),\varepsilon),R) \\
&\quad \cdot f(f_{*}((0,-),R),1) \cdot f(f_{*}((0,-),R1),0) \\
&= (0,-) \cdot f((0,-),R) \cdot f(f(f_{*}((0,-),\varepsilon),R),1) \\
&\quad \cdot f(f(f_{*}((0,-),R),1),0) \\
&= (0,-) \cdot f((0,-),R) \cdot f(f((0,-),R),1) \\
&\quad \cdot f(f(f(f_{*}(0,-),\varepsilon),R),1),0) \\
&= (0,-) \cdot f((0,-),R) \cdot f(f((0,-),R),1) \\
&\quad \cdot f(f(f((0,-),R),1),0) \\
&= (0,-) \cdot (0,R) \cdot (1,R) \cdot (0,-)
\end{aligned}
$$

Ausgaben

Nun betrachten wir einen Automaten mit Ausgabemöglichkeit. Die Kanten sind nun beschriftet mit $x \mid y$, wobei $x \in X$, $y \in Y^*$. Also wird bei Eingabe von x das Wort y ausgegeben.
Formal:

$$g(z, x) = y$$

Ausgabefunktion extended

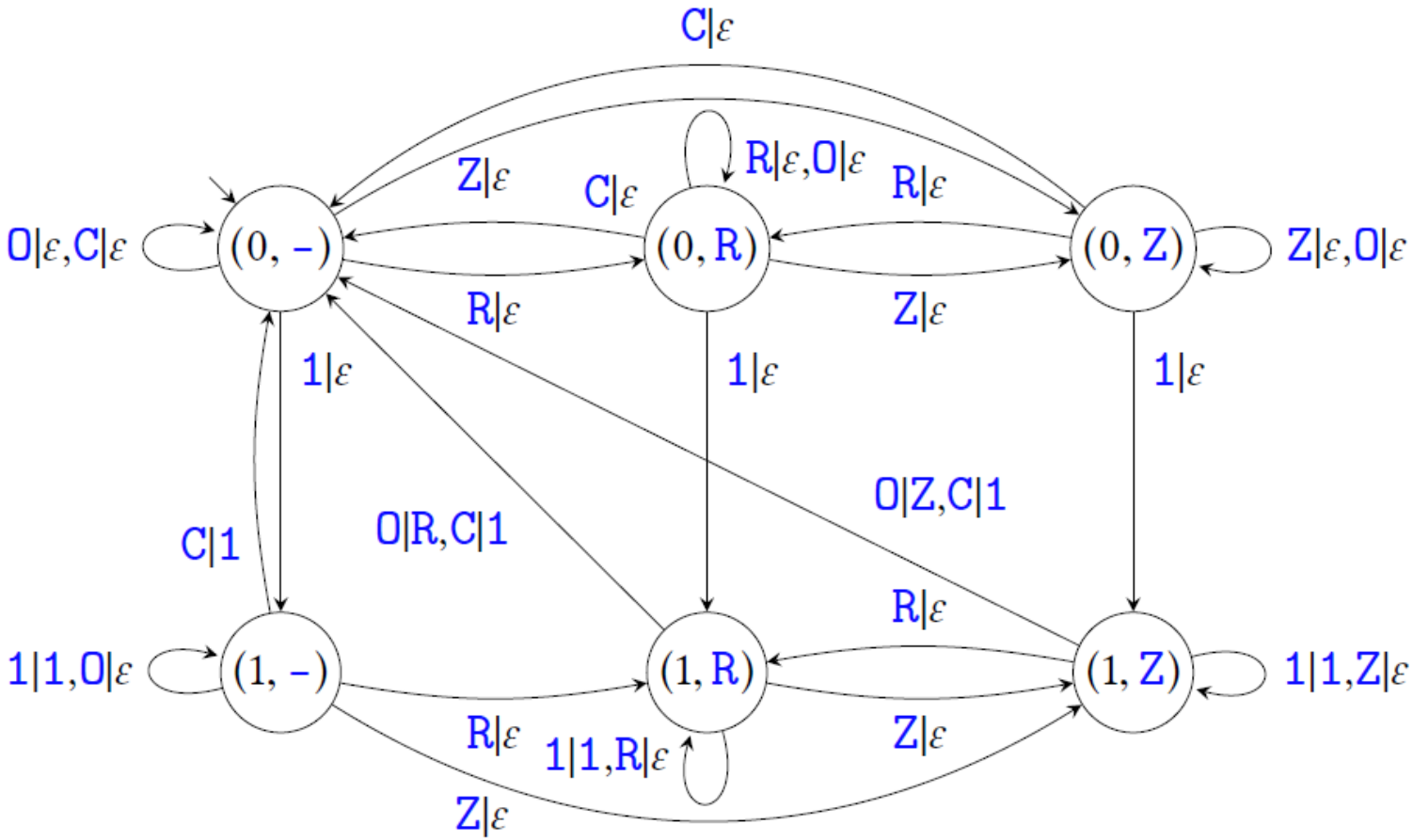
Wir können also analog zu f_* und f_{**} definieren

$$g_* : Z \times X^* \to Y^*$$
$$g_*(z, \varepsilon) = \varepsilon$$
$$\forall w \in X^* \; \forall x \in X : g_*(z, wx) = g(f_*(z, w), x)$$

$$g_{**} : Z \times X^* \to Y^*$$
$$g_{**}(z, \varepsilon) = \varepsilon$$
$$\forall w \in X^* \; \forall x \in X : g_{**}(z, wx) = g_{**}(z, w) \cdot g_*(z, wx)$$

Dies gibt nun nicht die durchlaufenen Zustände bzw. den Abschlusszustand an, sondern die letzte Ausgabe bzw. alle durchlaufenen Ausgaben.

Beispiel



Was macht $g_*((0,-),R10), g_{**}((0,-),R10), g_{**}((0,-),R110)$?

$g_*((0,-),R10) = g_{**}((0,-),R10) = R, \qquad g_{**}((0,-),R110) = 1R$

Moore-Automat

Moore-Automaten sind ähnlich aufgebaut wie Mealy-Automaten.
Unterschied: Ausgabe erfolgt im Zustand, nicht beim Zustandsübergang.
Ausgabefunktion: $h : Z \to Y$
Dann ist

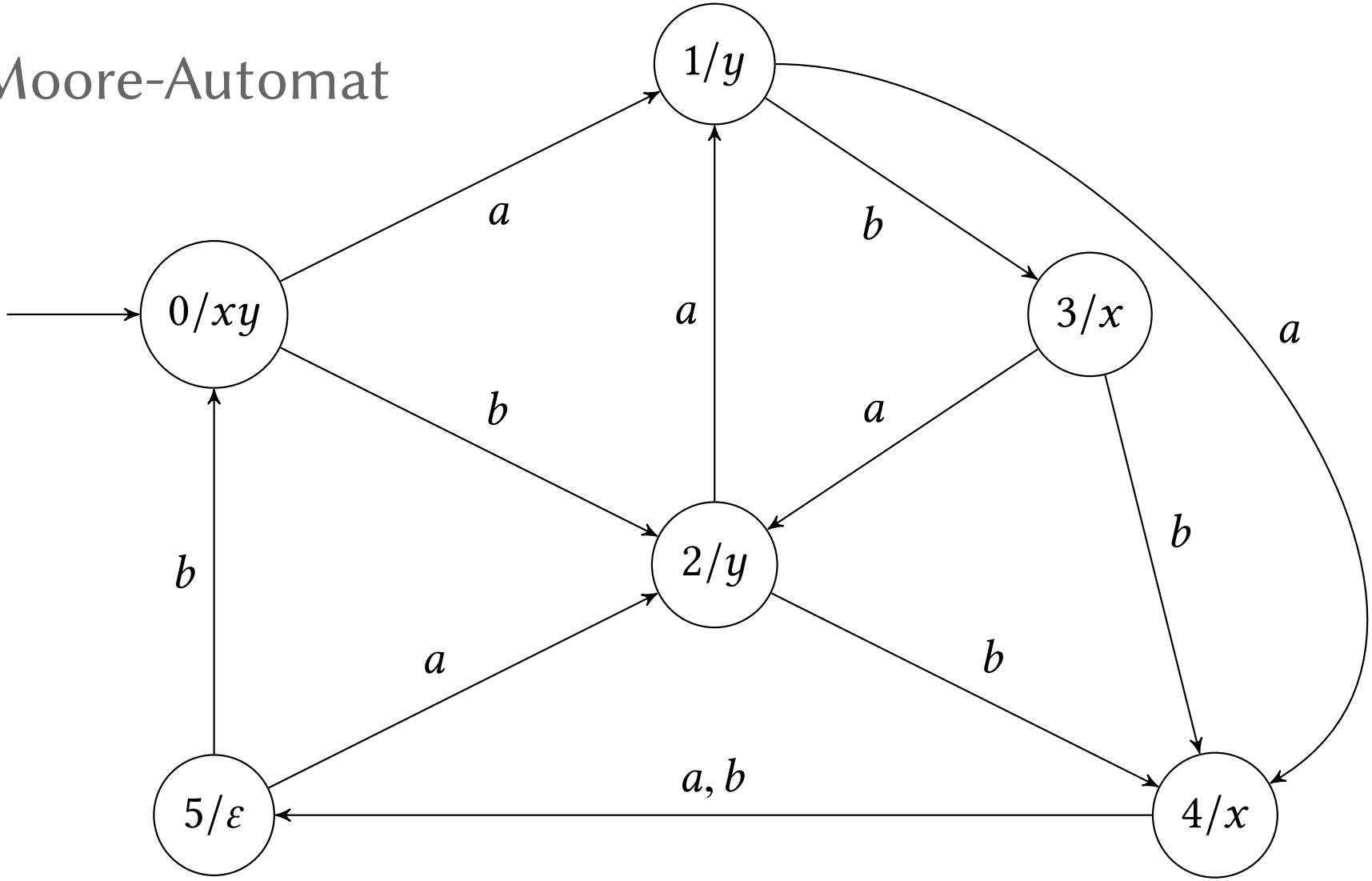
$$g_*(z, w) = h(f_*(z, w))$$

Für g_{**} gilt dann mit $h^{**} : Z^* \to Y$

$$g_{**}(z, w) = h^{**}(f_{**}(z, w))$$

Wir wenden also h auf jeden durchlaufenen Zustand an.

Moore-Automat



Eingabe: *aabab*

Ausgabe:

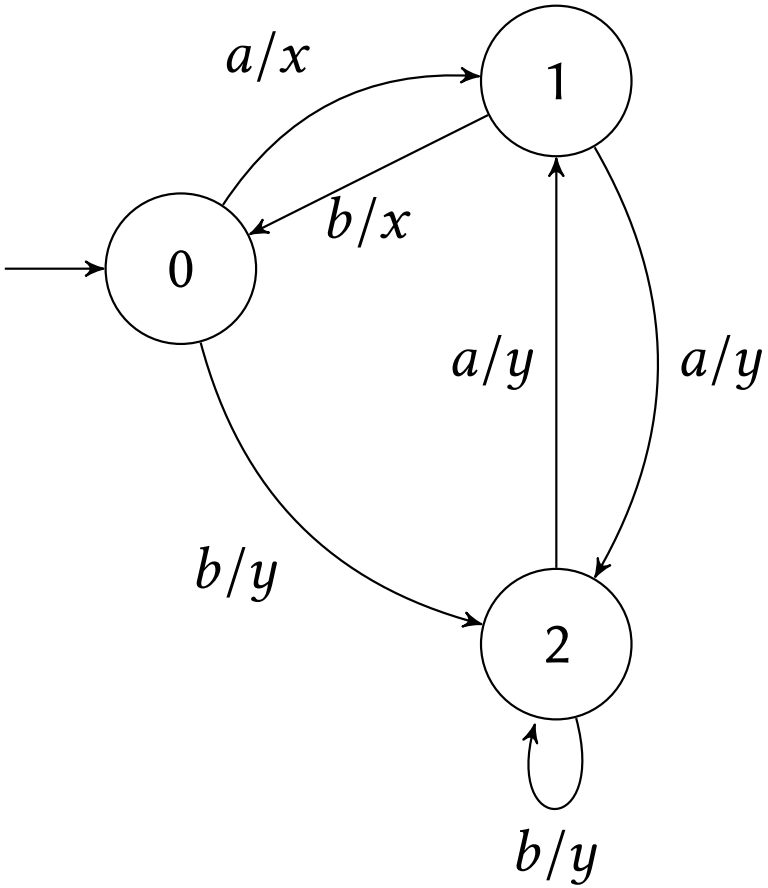
Von Moore zu Mealy

Relativ straightforward.
Ausgaben „aus den Knoten auf die Übergänge ziehen“
Beachte: Für die Eingabe ε kann ein Moore-Automat eine Ausgabe $g_{**}(s, \varepsilon) =\neq \varepsilon$ erzeugen, ein Mealy-Automat *jedoch nicht*.

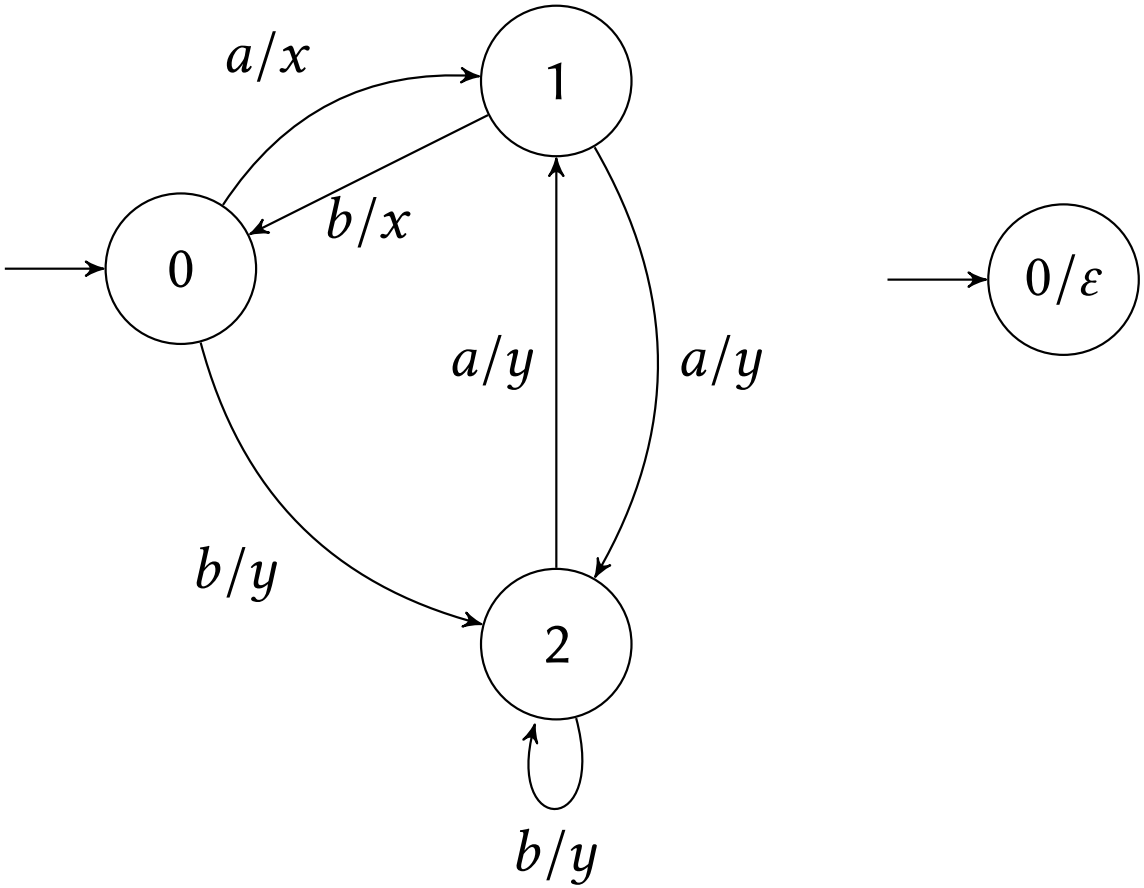
Von Mealy zu Moore

Komplizierter

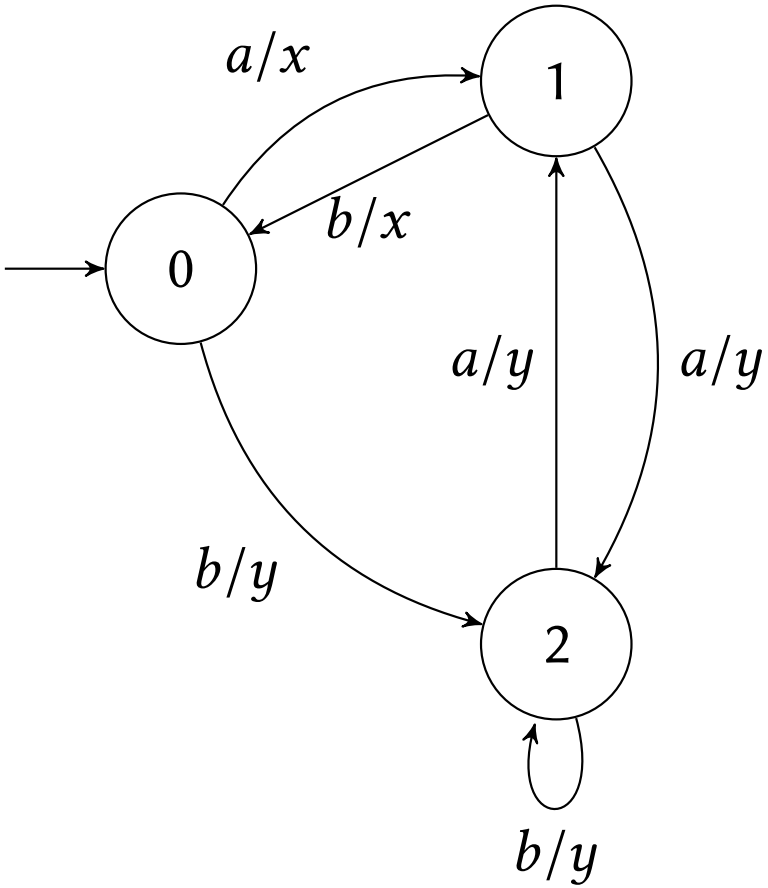
Umwandlung von Mealy- in Moore-Automat

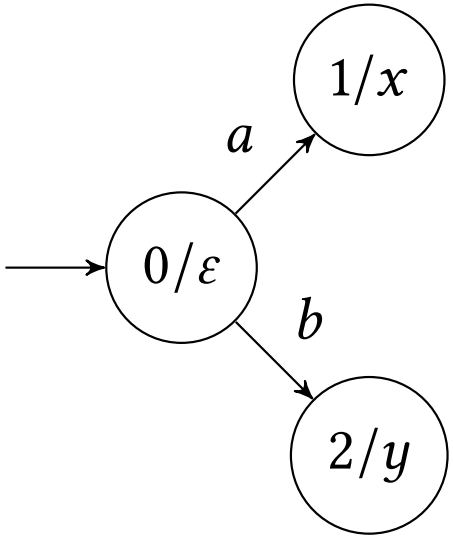


Umwandlung von Mealy- in Moore-Automat

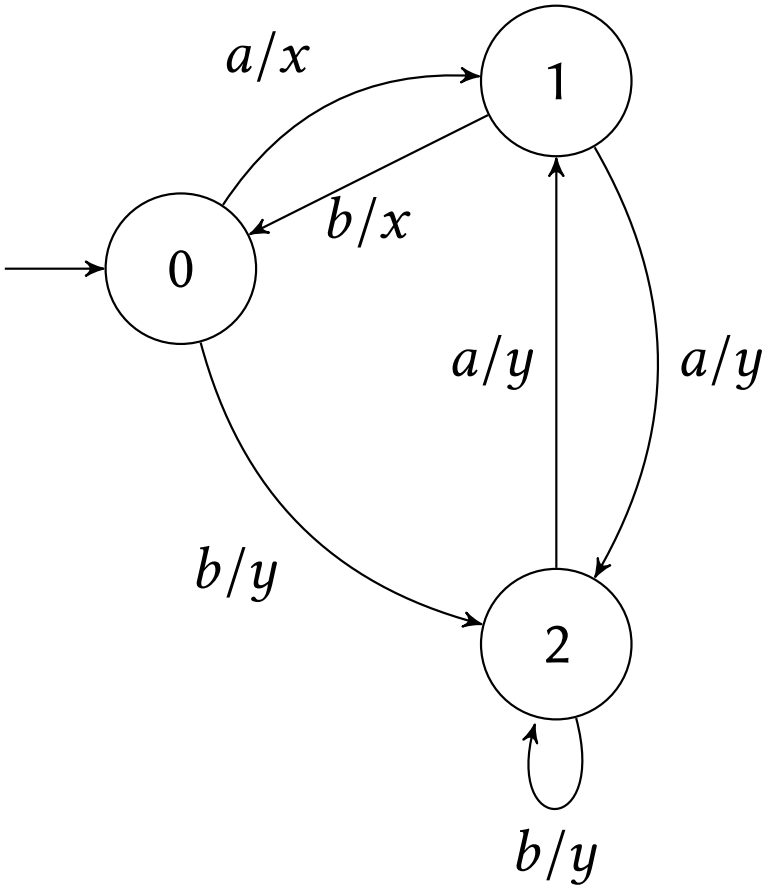


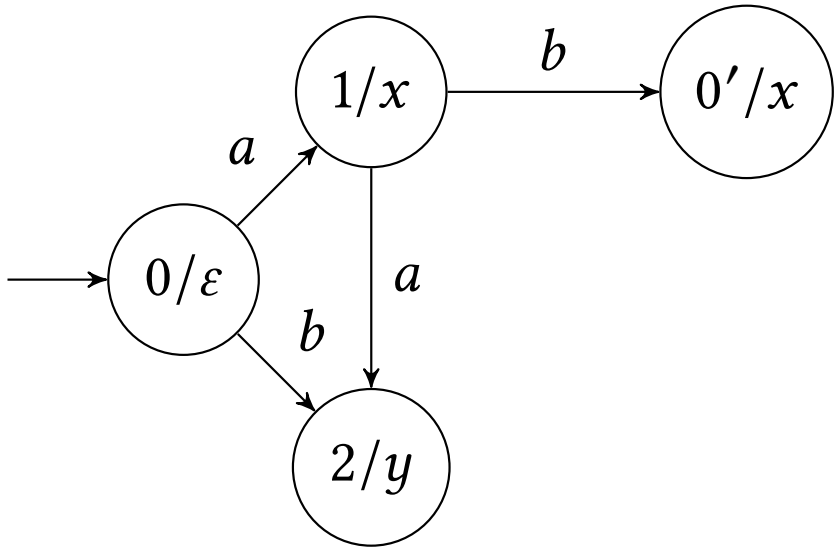
Umwandlung von Mealy- in Moore-Automat



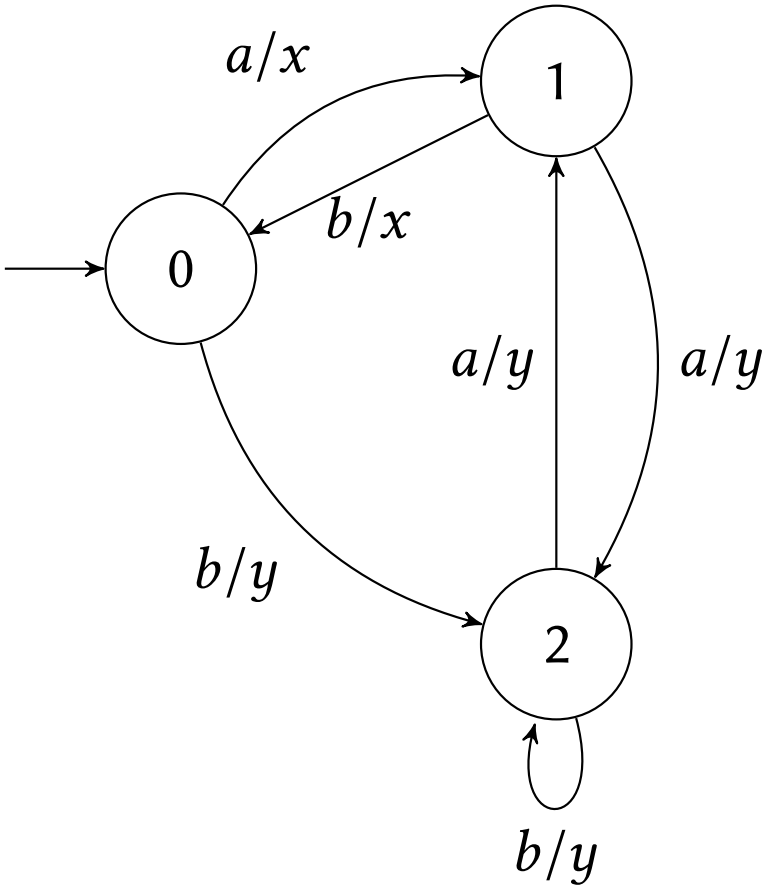


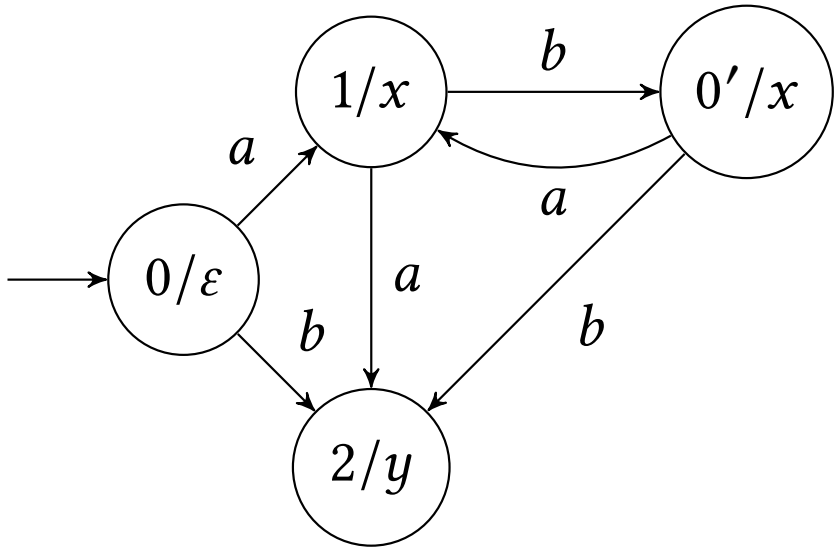
Umwandlung von Mealy- in Moore-Automat



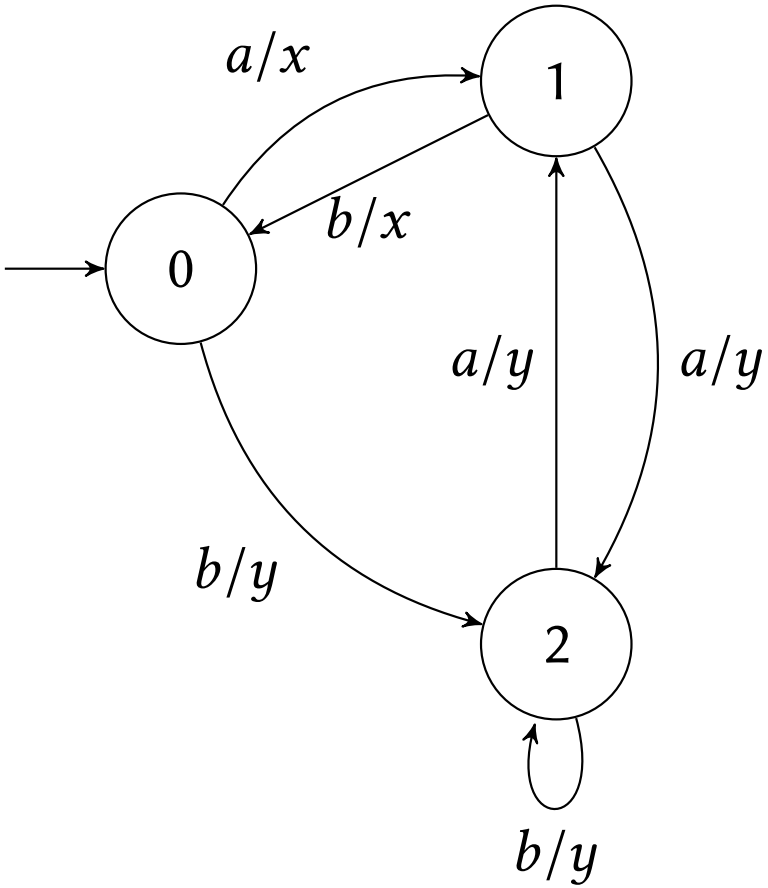


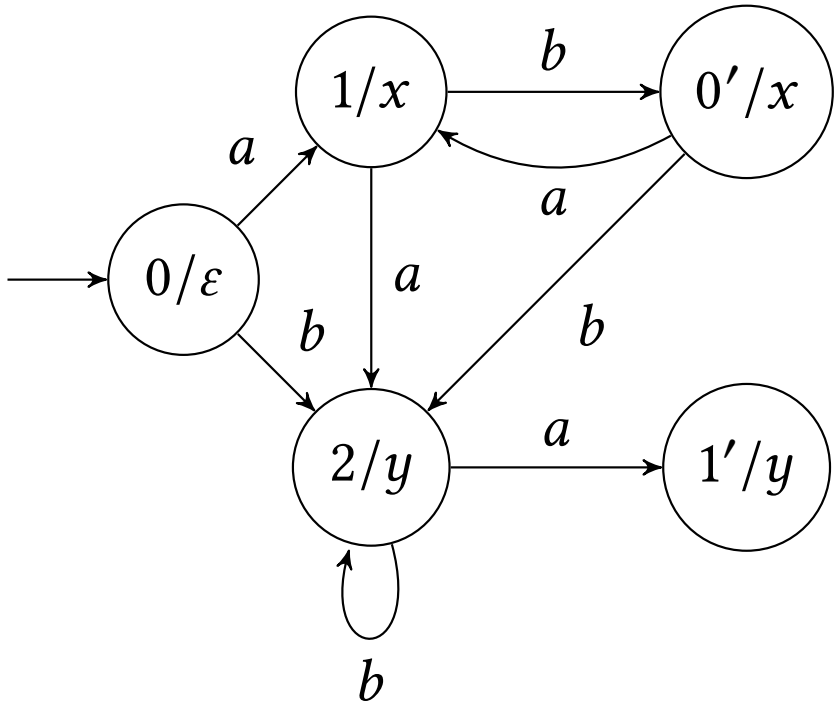
Umwandlung von Mealy- in Moore-Automat



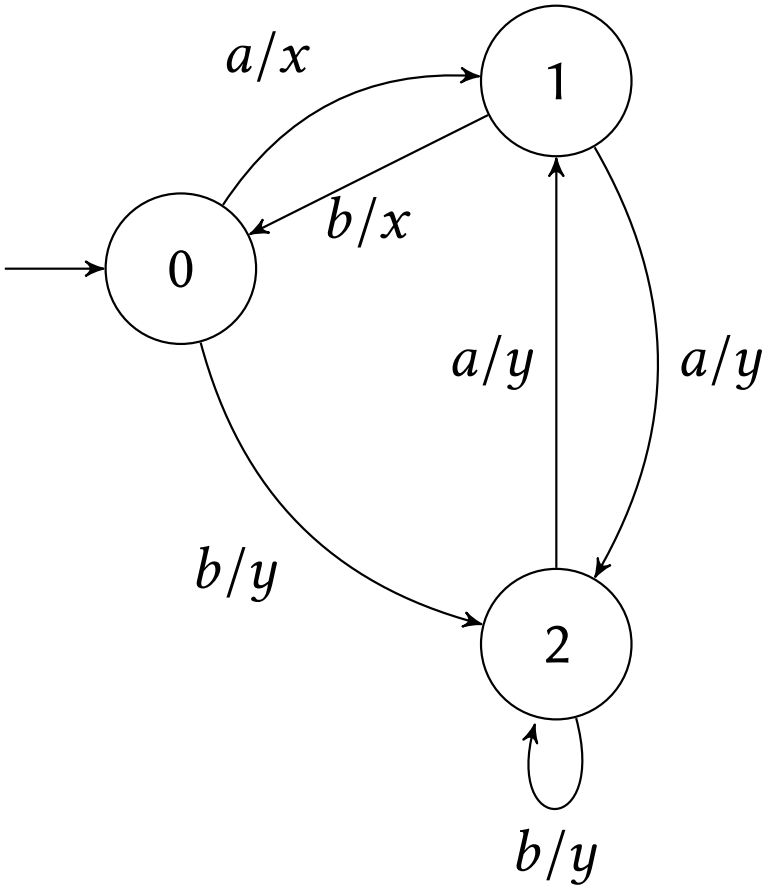


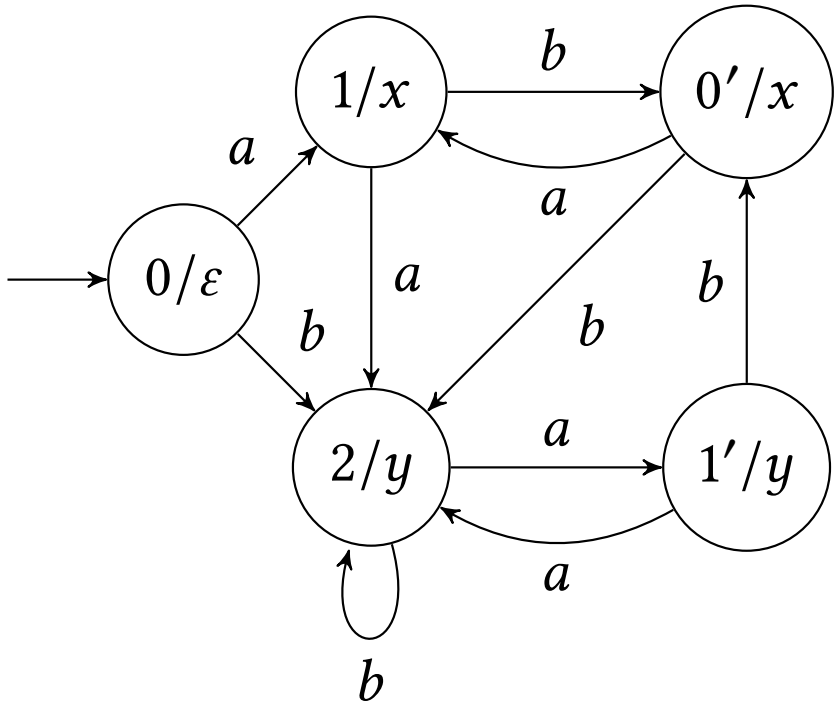
Umwandlung von Mealy- in Moore-Automat





Umwandlung von Mealy- in Moore-Automat





Master-Theorem

Automaten

Endliche Akzeptoren

Endliche Akzeptoren

Definition

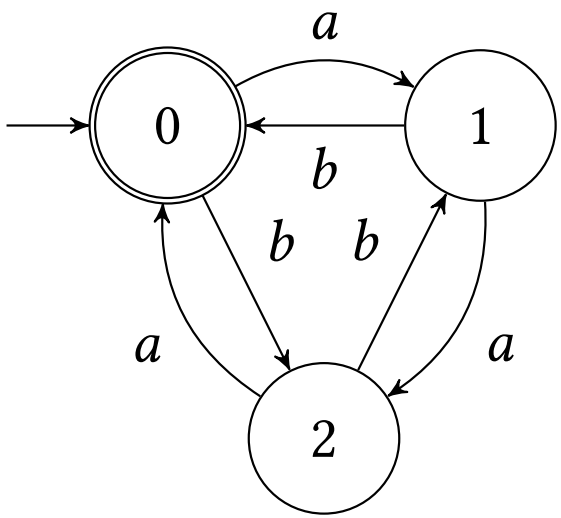
Ein endlicher Akzeptor ist ein Moore-Automat mit $Y = \{0, 1\}$, der 1 ausgibt, falls das eingegebene Wort der vorliegenden Syntax entspricht und 0 sonst.

Notation : Wir kennzeichnen akzeptierende Zustände mit einem Doppelkreis. Wir sprechen von einer akzeptierten Sprache über einem Alphabet. Sie ist definiert als

$$L = \{w \mid g_*(z_0, w) = 1\}$$

Also sind in einer akzeptierten Sprache alle Wörter, die akzeptiert werden.

Von Akzeptor erkannte Sprache



0 ist Anfangszustand und einziger akzeptierender Zustand
L^*, wobei $L \sim \{(0, v_1, \ldots, v_n, 0) \mid v_i \neq 0\}$

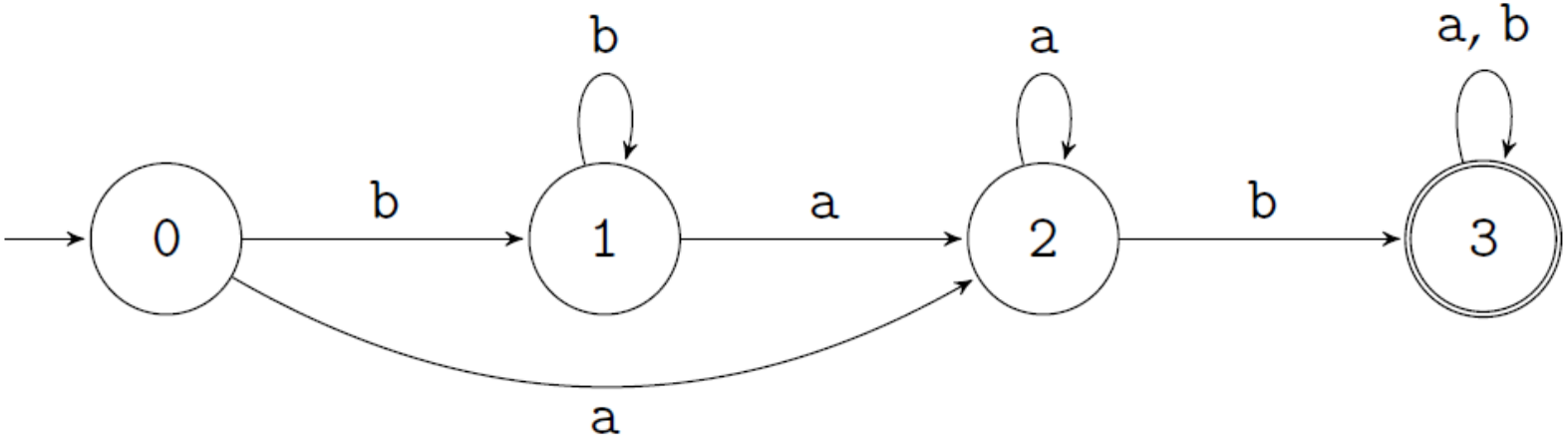
Fall $v_1 = 1$: Erst mit a von 0 nach 1
Dann mit ab beliebig oft nach 2 und zurück
Schließlich mit b oder aa nach 0

Fall $v_1 = 2$: Erst mit b von 0 nach 2
Dann mit ba beliebig oft nach 1 und zurück
Schließlich mit a oder bb nach

$$\Big(\big(\{a\} \cdot \{ab\}^* \cdot \big(\{b\} \cup \{aa\}\big)\big) \cup \big(\{b\} \cdot \{ba\}^* \cdot \big(\{a\} \cup \{bb\}\big)\big)\Big)^*$$

Aufgabe

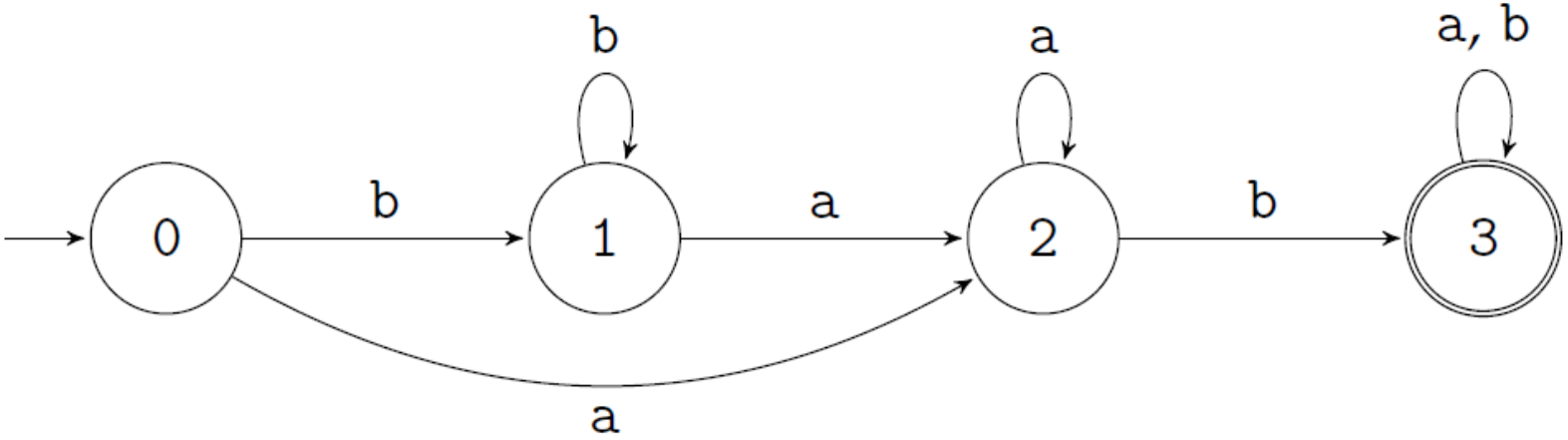
Der endliche Akzeptor $A = (Z, z_0, X, f, F)$ sei gegeben durch



Geben Sie die von A akzeptierte Sprache $L(A)$, unter ausschließlicher Benutzung der formalen Sprachen $\{\mathtt{a}\}$, $\{\mathtt{b}\}$, sowie $\{\mathtt{a}, \mathtt{b}\}$, des Konkatenationsabschlusses und des Produkts formaler Sprachen, an.
Beispiel: $\{\mathtt{a}, \mathtt{b}\}^* \cdot \{\mathtt{a}\} \cdot \{\mathtt{b}\}$

Aufgabe

Der endliche Akzeptor $A = (Z, z_0, X, f, F)$ sei gegeben durch



Geben Sie die von A akzeptierte Sprache $L(A)$, unter ausschließlicher Benutzung der formalen Sprachen $\{\mathrm{a}\}$, $\{\mathrm{b}\}$, sowie $\{\mathrm{a}, \mathrm{b}\}$, des Konkatenationsabschlusses und des Produkts formaler Sprachen, an.
Beispiel: $\{\mathrm{a}, \mathrm{b}\}^* \cdot \{\mathrm{a}\} \cdot \{\mathrm{b}\}$

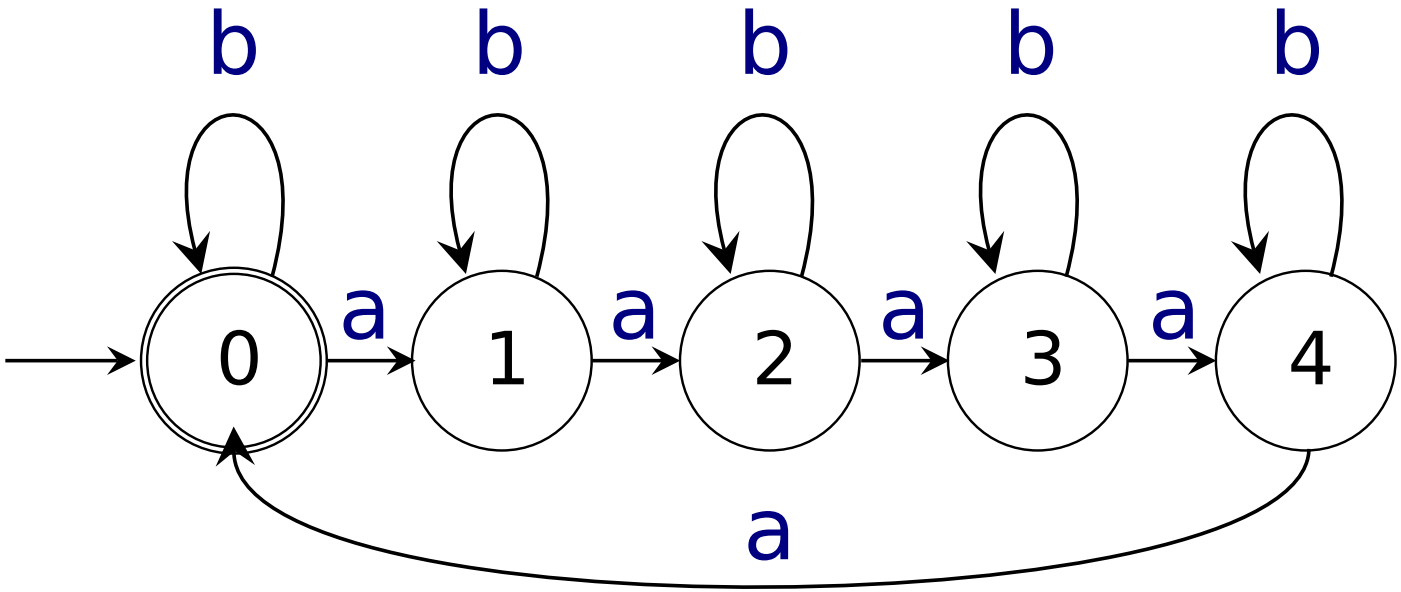
Lösung

$L(A) = \{\mathrm{b}\}^* \cdot \{\mathrm{a}\} \cdot \{\mathrm{a}\}^* \cdot \{\mathrm{b}\} \cdot \{\mathrm{a}, \mathrm{b}\}^*$

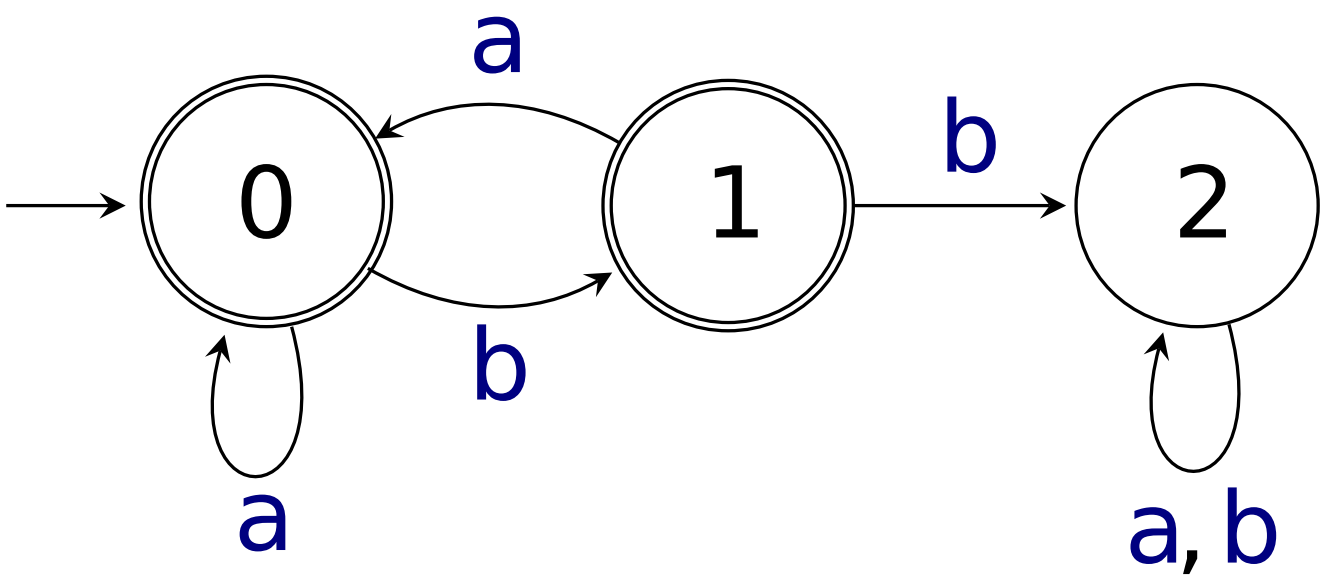
Aufgabe

- Zeichnen Sie einen möglichst kleinen endlichen Akzeptor mit $X = \{a, b\}$, der alle Wörter akzeptiert, bei denen die Anzahl der a durch 5 teilbar ist.
- Zeichnen Sie einen Akzeptor mit $X = \{a, b\}$, der alle Wörter akzeptiert, in denen nirgends hintereinander zwei b vorkommen.

Lösung



Lösung



Was ihr nun wissen solltet

- Einfache Laufzeitabschätzungen mit dem Master-Theorem
- Endliche Automaten: Mealy und Moore
- Endliche Akzeptoren

Was nächstes Mal kommt

- Grenzen endlicher Automaten – Wer kann zählen?
- Ein Regelwerk für einen Ausdruck – Reguläre Ausdrücke

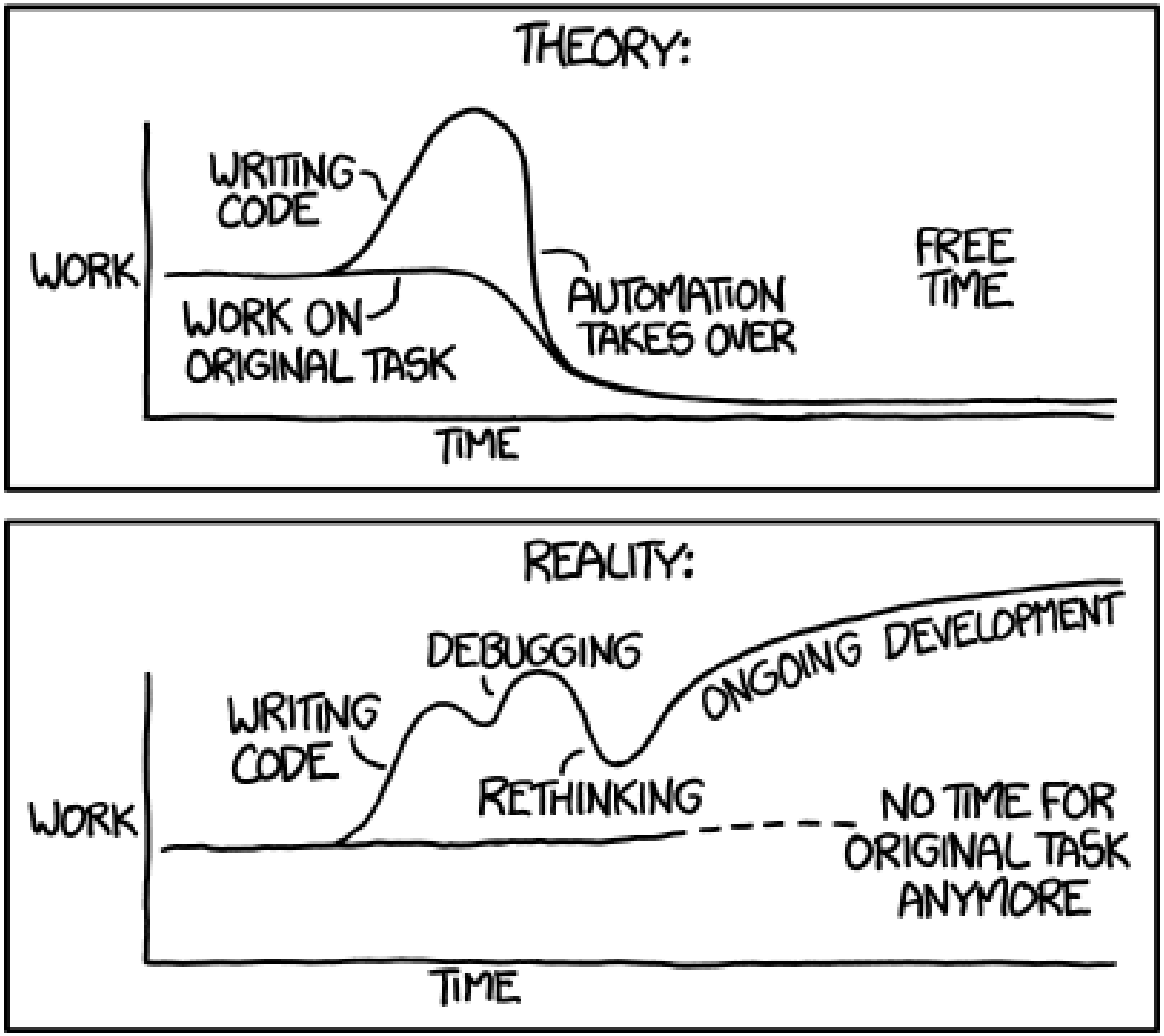


Abbildung: http://www.xkcd.com/1319

Credits

Vorgänger dieses Foliensatzes wurden erstellt von:

Thassilo Helmold
Philipp Basler
Nils Braun
Dominik Doerner
Ou Yue