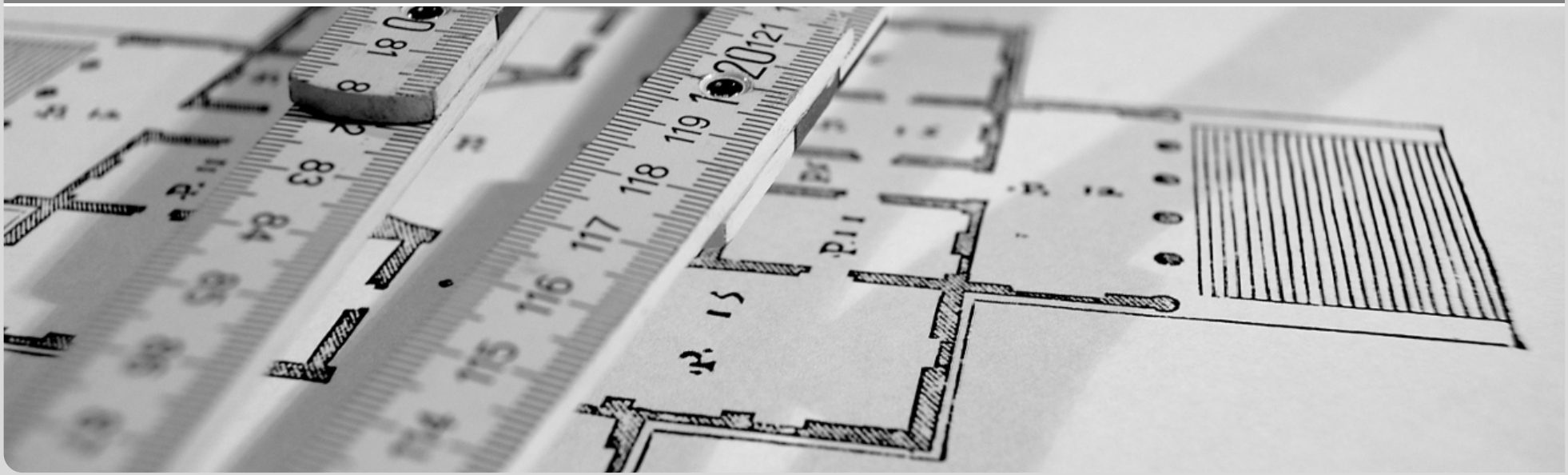
Grundbegriffe der Informatik
Tutorium 36

Termin 14 | 10.02.2017
Thassilo Helmold

KIT – Karlsruher Institut für Technologie



Inhalt

Turingmaschinen

Komplexität

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

In the previous episode of GBI...

Rückblick: Reguläre Sprachen

- Endliche Akzeptoren
- Reguläre Ausdrücke
- Rechtslineare Grammatiken
- Strukturelle Induktion

Wahr oder Falsch?

- Jede kontextfreie Grammatik lässt sich als regulärer Ausdruck darstellen

Wahr oder Falsch?

- Jede kontextfreie Grammatik lässt sich als regulärer Ausdruck darstellen
  F Die Grammatik muss rechtslinear sein, kontextfrei kann „mehr“

Wahr oder Falsch?

- Jede kontextfreie Grammatik lässt sich als regulärer Ausdruck darstellen
  F Die Grammatik muss rechtslinear sein, kontextfrei kann „mehr“
- Jeder regulären Ausdruck lässt sich durch eine kontextfreie Grammatik darstellen

Wahr oder Falsch?

- Jede kontextfreie Grammatik lässt sich als regulärer Ausdruck darstellen
  F Die Grammatik muss rechtslinear sein, kontextfrei kann „mehr“
- Jeder regulären Ausdruck lässt sich durch eine kontextfreie Grammatik darstellen W

Wahr oder Falsch?

- Jede kontextfreie Grammatik lässt sich als regulärer Ausdruck darstellen
  F Die Grammatik muss rechtslinear sein, kontextfrei kann „mehr“
- Jeder regulären Ausdruck lässt sich durch eine kontextfreie Grammatik darstellen W
- Für jede Sprache L und jedes Wort $w \in L$ gilt: Es existiert ein endlicher Automat, der w erkennt

Wahr oder Falsch?

- Jede kontextfreie Grammatik lässt sich als regulärer Ausdruck darstellen
  F Die Grammatik muss rechtslinear sein, kontextfrei kann „mehr“
- Jeder regulären Ausdruck lässt sich durch eine kontextfreie Grammatik darstellen W
- Für jede Sprache L und jedes Wort $w \in L$ gilt: Es existiert ein endlicher Automat, der w erkennt W

Wahr oder Falsch?

- Jede kontextfreie Grammatik lässt sich als regulärer Ausdruck darstellen
  F Die Grammatik muss rechtslinear sein, kontextfrei kann „mehr“
- Jeder regulären Ausdruck lässt sich durch eine kontextfreie Grammatik darstellen W
- Für jede Sprache L und jedes Wort $w \in L$ gilt: Es existiert ein endlicher Automat, der w erkennt W
- Die Sprache der gültigen Klammerausdrücke ist regulär

Wahr oder Falsch?

- Jede kontextfreie Grammatik lässt sich als regulärer Ausdruck darstellen
  F Die Grammatik muss rechtslinear sein, kontextfrei kann „mehr“
- Jeder regulären Ausdruck lässt sich durch eine kontextfreie Grammatik darstellen W
- Für jede Sprache L und jedes Wort $w \in L$ gilt: Es existiert ein endlicher Automat, der w erkennt W
- Die Sprache der gültigen Klammerausdrücke ist regulär F

Wahr oder Falsch?

- Jede kontextfreie Grammatik lässt sich als regulärer Ausdruck darstellen
  F Die Grammatik muss rechtslinear sein, kontextfrei kann „mehr“
- Jeder regulären Ausdruck lässt sich durch eine kontextfreie Grammatik darstellen W
- Für jede Sprache L und jedes Wort $w \in L$ gilt: Es existiert ein endlicher Automat, der w erkennt W
- Die Sprache der gültigen Klammerausdrücke ist regulär F
- Die Sprache der gültigen Klammerausdrücke ist kontextfrei

Wahr oder Falsch?

- Jede kontextfreie Grammatik lässt sich als regulärer Ausdruck darstellen
  F Die Grammatik muss rechtslinear sein, kontextfrei kann „mehr“
- Jeder regulären Ausdruck lässt sich durch eine kontextfreie Grammatik darstellen W
- Für jede Sprache L und jedes Wort $w \in L$ gilt: Es existiert ein endlicher Automat, der w erkennt W
- Die Sprache der gültigen Klammerausdrücke ist regulär F
- Die Sprache der gültigen Klammerausdrücke ist kontextfrei W

Dr. Meta ist zurück!

Turingmaschinen

Komplexität

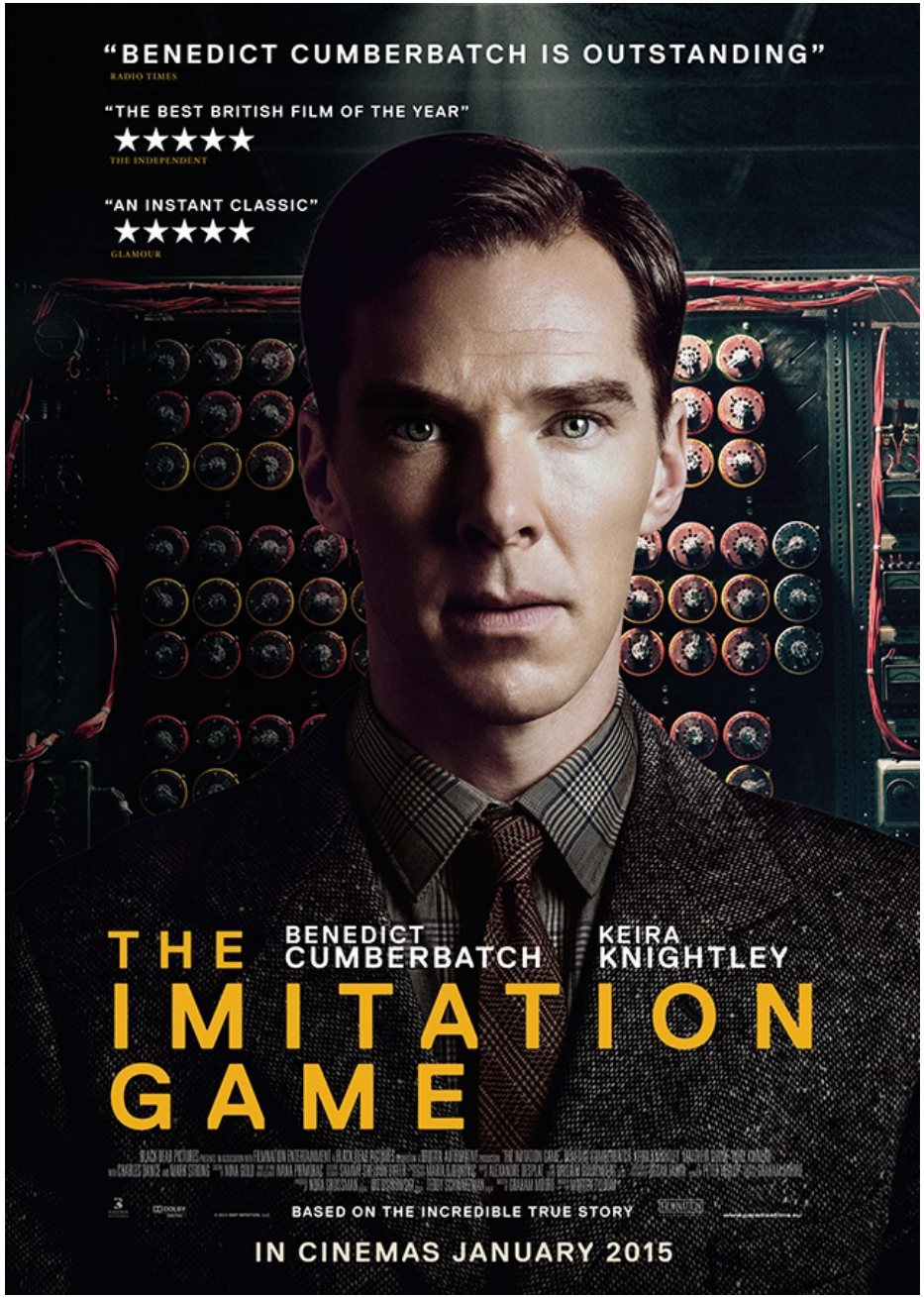
Sprachen mal wieder...

Wir haben gesehen: Kontextfreie Grammatiken können mehr als endliche Automaten (Klammerausdrücke!).
Wir wollen daher ein passendes Berechnungsmodell, das ebenfalls „mehr kann“ als endliche Automaten.

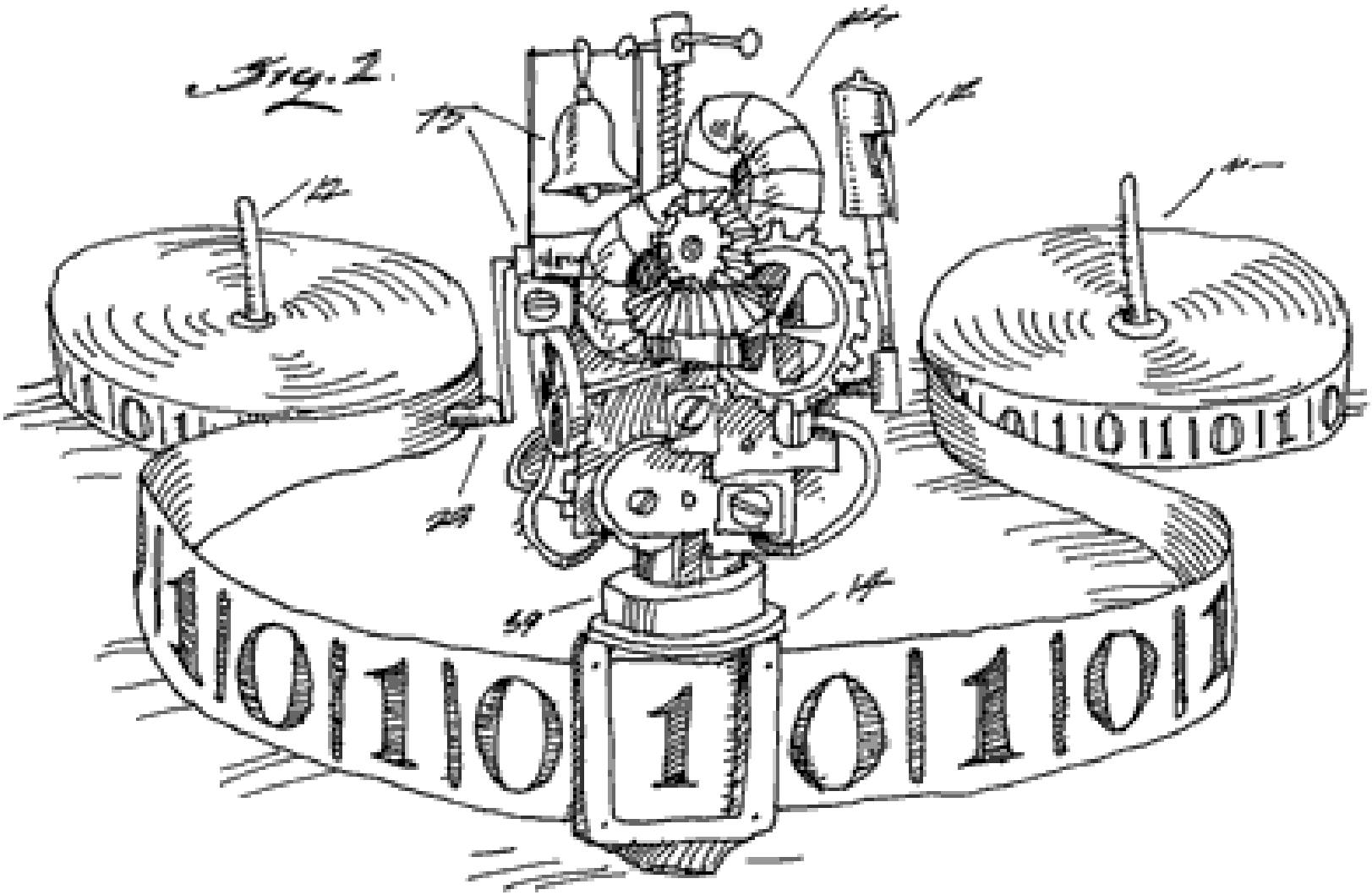
Wie wäre es mit einem Smartphone?
Das kann ja wohl offensichtlich gültige Klammerausdrücke erkennen.
Als theoretisches Modell aber ungeeignet, da viel zu komplex.

Einfacheres Modell: Die **Turingmaschine**

Alan Turing


"BENEDICT CUMBERBATCH IS OUTSTANDING"
RADIO TIMES
"THE BEST BRITISH FILM OF THE YEAR"
THE INDEPENDENT
"AN INSTANT CLASSIC"
GLAMOUR
BENEDICT CUMBERBATCH
KEIRA KNIGHTLEY
THE IMITATION GAME
BASED ON THE INCREDIBLE TRUE STORY
IN CINEMAS JANUARY 2015

Turingmaschine



Turingmaschine

Eine Turingmaschine besteht aus...

- einem unendlichen Band mit einzelnen Zellen, in denen jeweils genau ein Symbol aus dem Bandalphabet steht
- einem Schreib-/Lesekopf, der auf genau einer Bandzelle steht
- einer Steuerungseinheit (Mealy-Automat), die sich in genau einem Zustand befindet

Funktionsweise der TM

In jedem Ausführungsschritt:

- TM liest das Symbol, auf dem der Kopf steht
- Steuerung entscheidet über die Ausgabe, den Folgezustand und die Bewegung
- Ausgabe wird an der aktuellen Position auf das Band geschrieben
- Steuerung wechselt in den Folgezustand
- Kopf bewegt sich einen Schritt nach links oder rechts oder bleibt stehen

Die TM hält, wenn für einen Zustand und das gelesene Symbol keine Übergänge in der Steuerung definiert sind. **Eine TM muss nicht halten!**

Ein-/Ausgabe

Eingabe

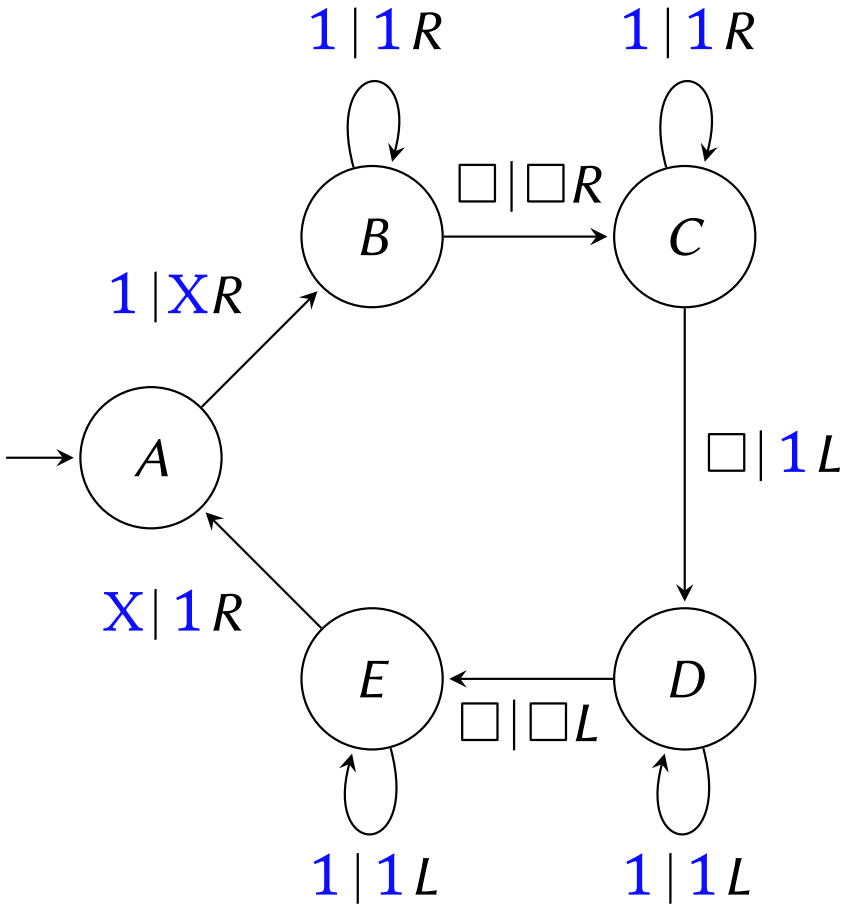
Am Anfang steht das Eingabewort umgeben von Blanksymbolen auf dem Band, der Kopf steht auf dem ersten Zeichen des Eingabeworts.

Ausgabe

Zwei Möglichkeiten:

- Berechnung von Funktionen: Ausgabewort steht am Ende auf dem Band
- Erkennen von Sprachen: Halten in akzeptierendem Zustand
  Wir bezeichnen dann wieder mit $L(T)$ die akzeptierte Sprache

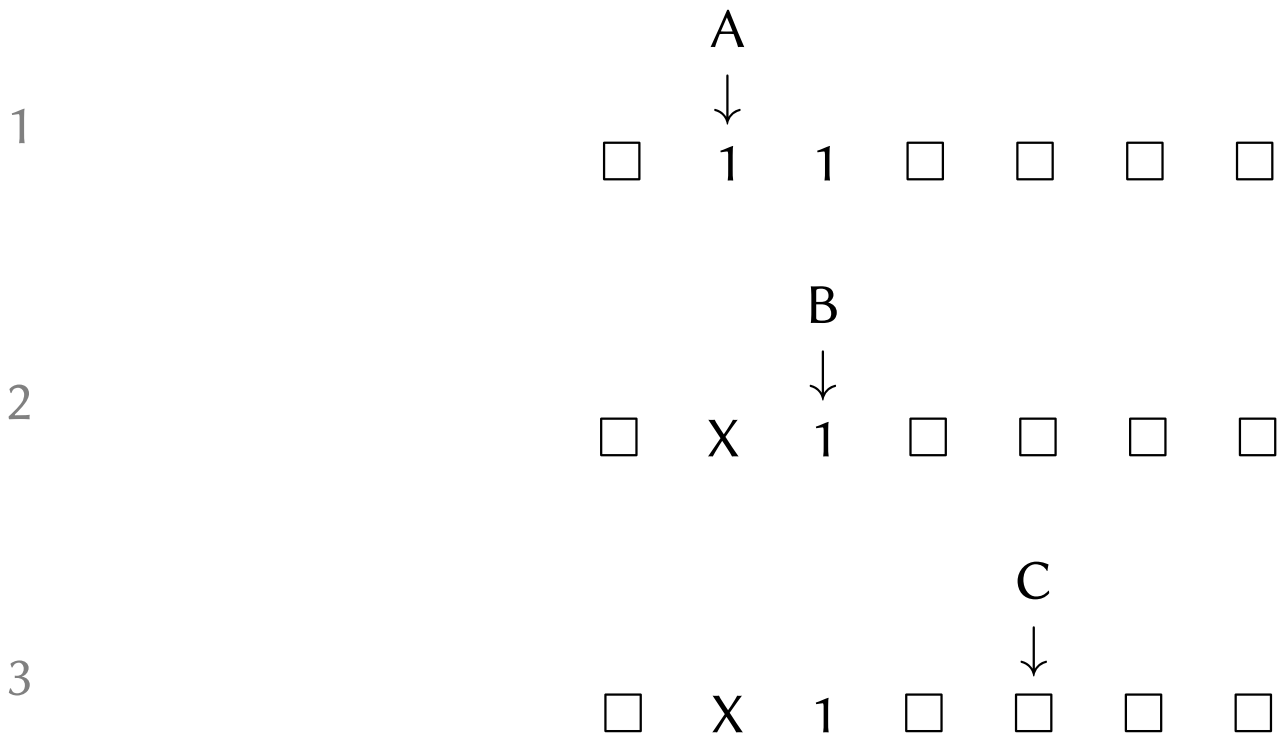
Beispiel



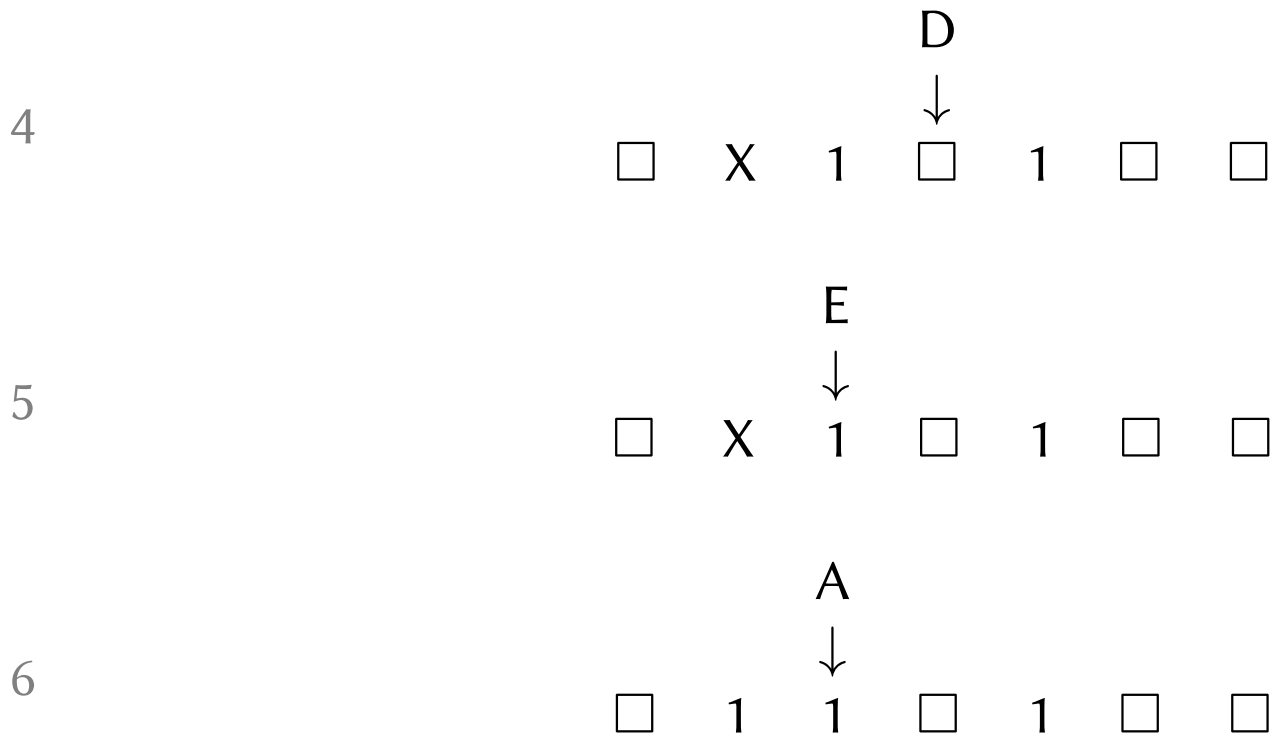
	A	B	C	D	E
□		C,□,R	D, 1, L	E,□, L	
1	B,X, R	B,1, R	C,1, R	D,1, L	E,1, L
X					A,1,R

Beispiel

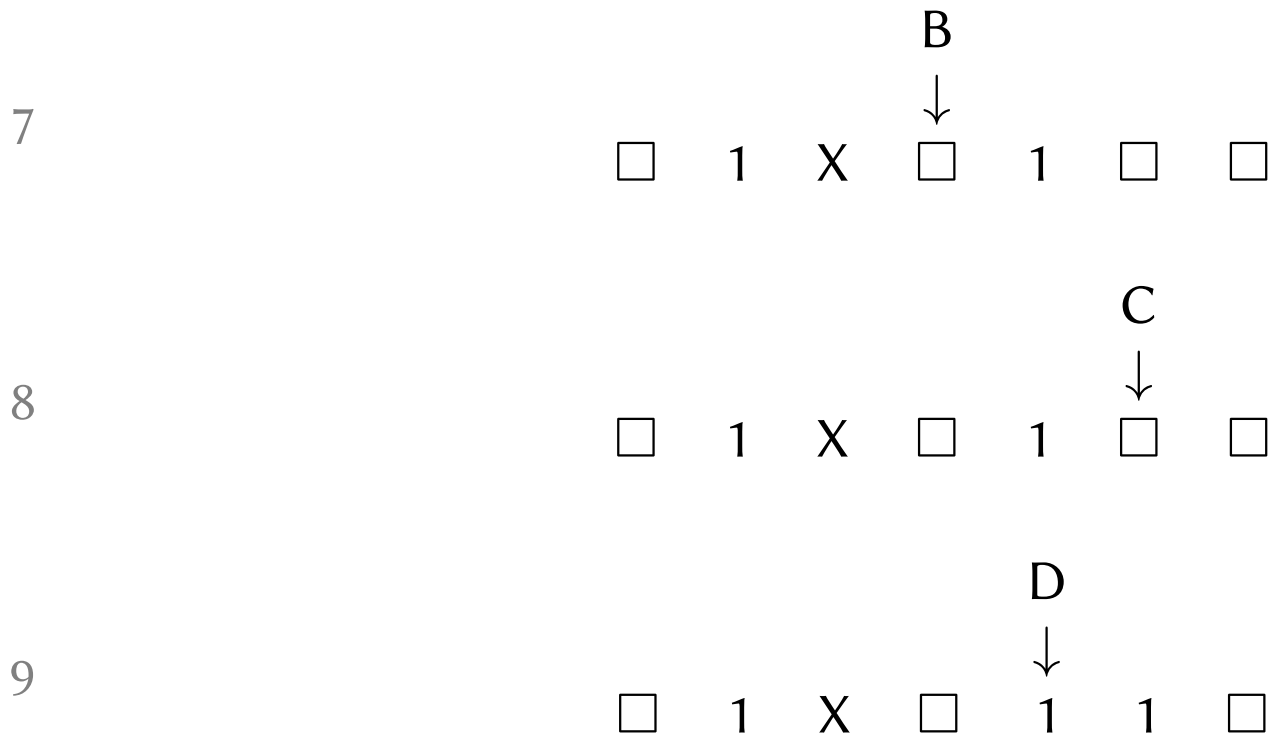
Eingabe: 11



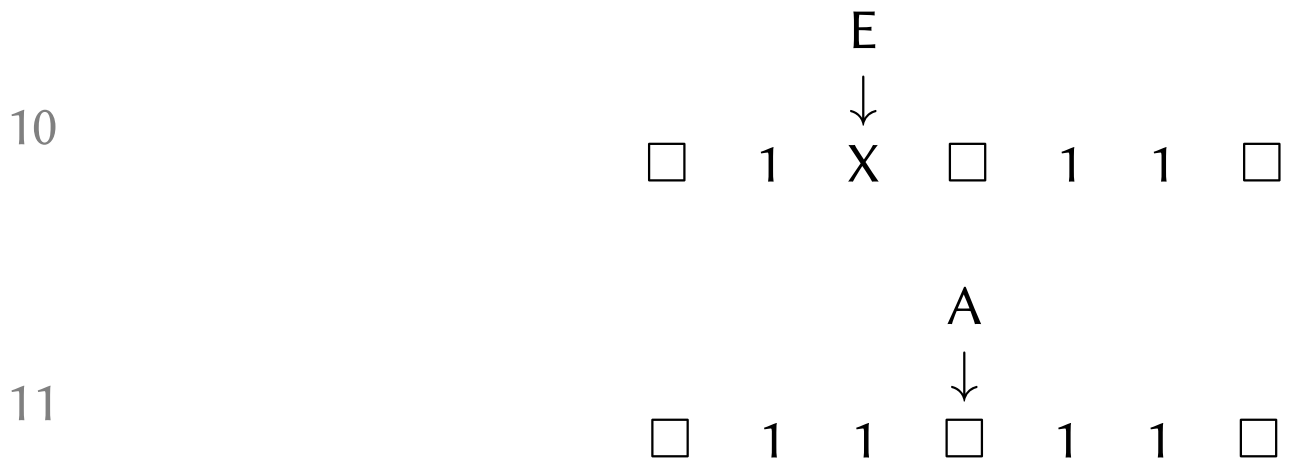
Beispiel



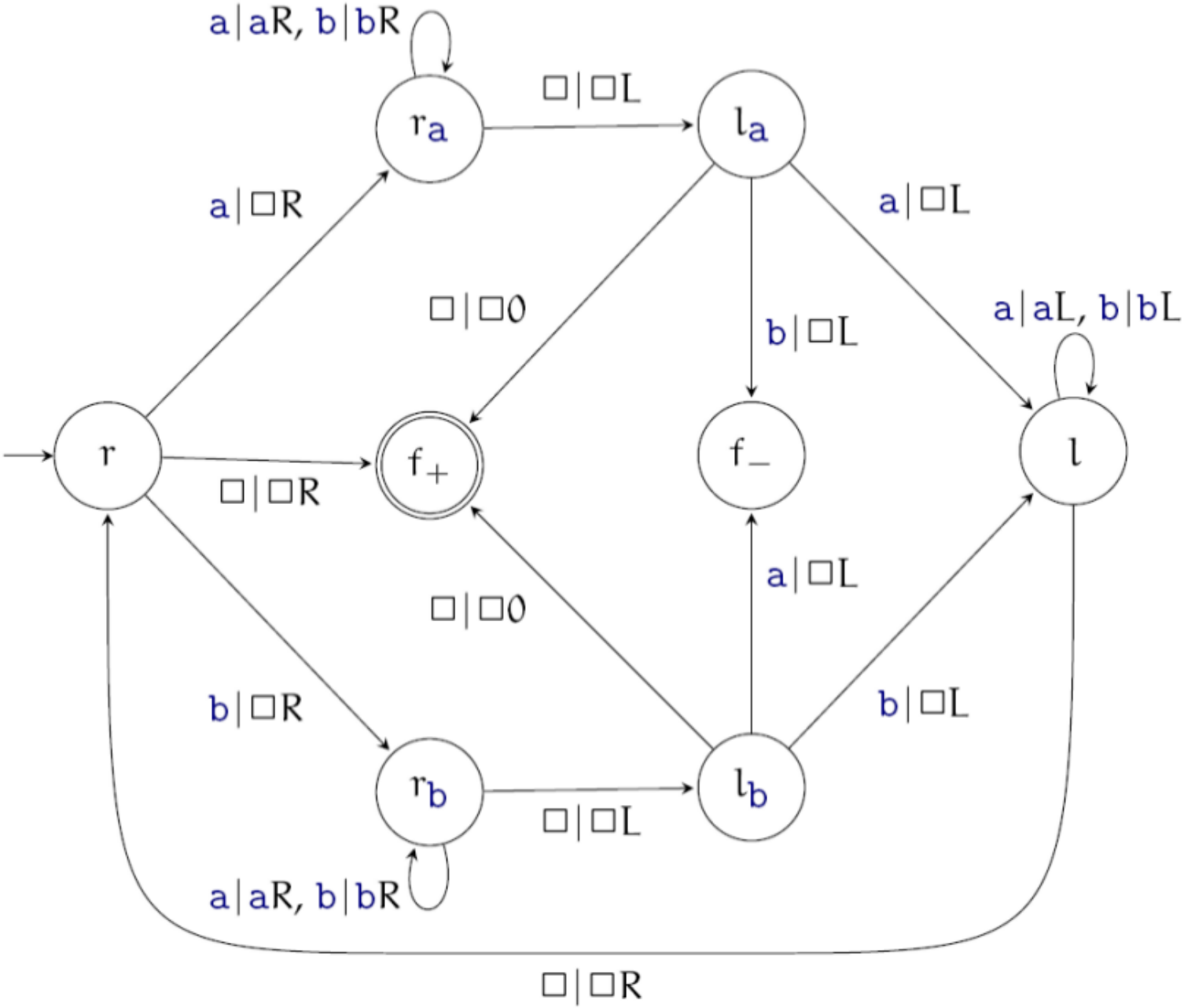
Beispiel



Beispiel



Also allgemein : Eingabe von 1^k wird zu $\square\, 1^k \square\, 1^k \square$


a|aR, b|bR
□|□L
r_a
l_a
a|□R
a|□L
□|□0
a|aL, b|bL
b|□L
r
f_+
f_-
l
□|□R
a|□L
□|□0
b|□R
b|□L
r_b
l_b
□|□L
a|aR, b|bR
□|□R

Turingmaschine

Definition

Eine Turingmaschine T ist definiert als

$$T = (Z, z_0, X, f, g, m)$$

- Z Zustandsmenge
- $z_0 \in Z$ Startzustand
- X Bandalphabet mit $\square \in X$
- $f : Z \times X \dashrightarrow Z$ Übergangsfunktion
- $g : Z \times X \dashrightarrow X$ Ausgabefunktion
- $m : Z \times X \dashrightarrow \{\mathrm{L}, \mathrm{N}, \mathrm{R}\}$ Bewegungsfunktion

Alle Funktionen können auch nur partiell definiert sein.

Konfiguration

Definition

Als **Konfiguration** $c = (z, b, p)$ bezeichnen wir den Zustand einer Turing-Maschine zu einem Zeitpunkt. Dabei ist

- $z \in Z$ der Zustand
- $b : \mathbb{Z} \to X$ die Bandbeschriftung
- $p \in \mathbb{Z}$ die Position des Zeigers.

Berechnungsschritte

Definition

In einem Berechnungsschritt geht eine TM aus einer Konfiguration c in die Konfiguration

$$\Delta_1(c) := c' = (z', b', p')$$

über, wobei gilt:

- $z' = f(z, b(p))$
- $\forall i \in \mathbb{Z} : b'(i) = \begin{cases} b(i) & \text{falls } i \neq p \\ g(z, b(p)) & \text{falls } i = p \end{cases}$
- $p' = p + m(z, b(p))$

Wenn $\Delta_1(c)$ nicht definiert ist, bezeichnet man c als **Endkonfiguration** und die TM **hält**.

Berechnungen

Definition

Eine Berechnung ist eine Folge von Konfigurationen $(c_0, c_1, ..., c_t)$ bei der gilt

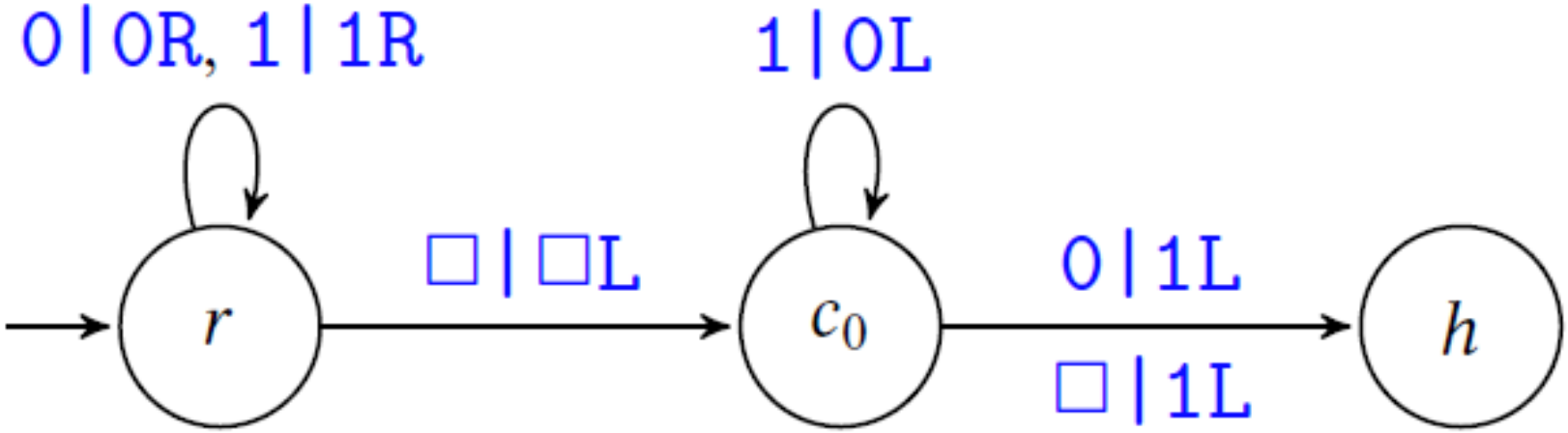
$$c_{i+1} = \Delta_1(c_i)$$

Es gibt endliche, haltende (endet in einer Endkonfiguration) und unendliche Berechnungen.

Induktiv definiert man $\Delta_t(c), t \in \mathbb{N}_0$ als die Konfiguration, welche nach t Berechnungsschritten erreicht werden kann und $\Delta_*(c)$ als Endkonfiguration, die von c aus erreicht wird (falls die Berechnung endet).

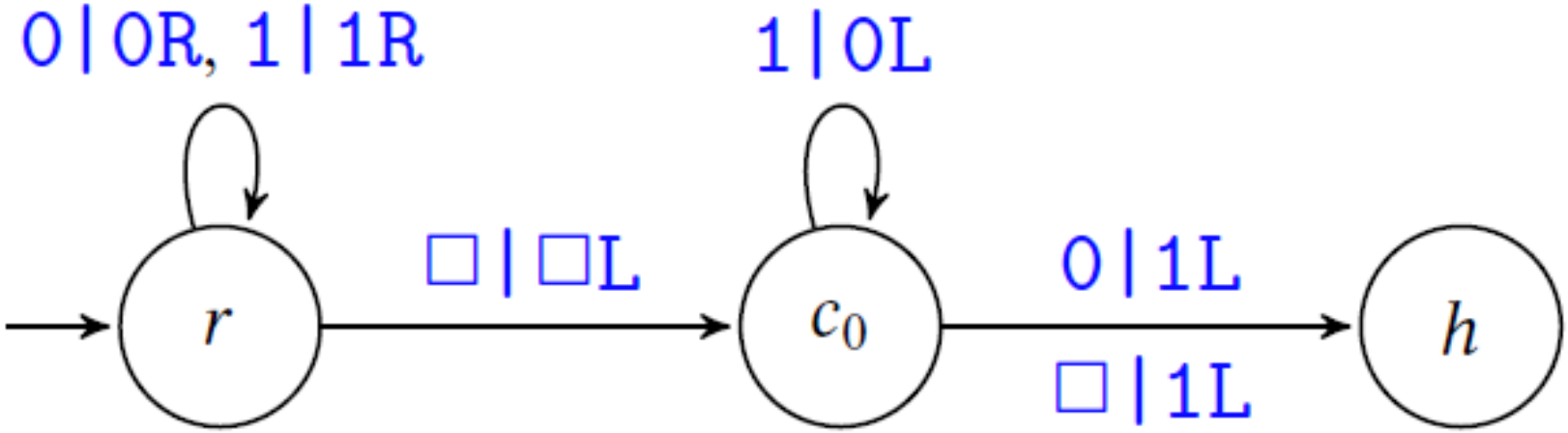
Aufgabe

Was macht die folgende Turingmaschine für Eingaben aus $\{0, 1\}^*$?



Aufgabe

Was macht die folgende Turingmaschine für Eingaben aus $\{0, 1\}^*$?



Inkrementieren der dargestellten Zahl um 1.

Aufgabe

Entwerfen Sie eine Turingmaschine, die für eine Eingabe $w \in \{0, 1\}^*$

$$Repr_2(Num_2(w) - 1)$$

berechnet. Sie dürfen davon ausgehen, dass $Num_2(w) > 0$ gilt.
Achtung: Was passiert mit führenden Nullen?

Wie würde man eine (einfache, nicht effiziente) TM zum Addieren von zwei Zahlen in Binärdarstellung aufbauen?

Entscheidbarkeit

Definition

Eine Sprache L ist eine

- **aufzählbare Sprache**, wenn es eine Turingmaschine gibt, die L akzeptiert.
- **entscheidbare Sprache**, wenn es eine Turingmasche gibt, die L akzeptiert und *für jede Eingabe* hält.

Bei aufzählbaren Sprachen ist nicht definiert, wie sich die TM für Wörter $w \notin L$ verhält. Sie kann diese ablehnen oder nicht halten. Ob eine TM für eine Eingabe nicht hält, können wir „von Außen“ nicht einfach feststellen.

Aufgabe

Entwerfen Sie eine Turingmaschine, die $w \in \{0^k 1^k | k \in \mathbb{N}_0\}$ akzeptiert.

Aufgabe

Entwirf eine TM, die alle gültigen Klammerausdrücke entscheidet.
Gültige Klammerausdrücke werden dabei von der folgenden Grammatik produziert: $G = (\{S\}, \{(,)\}, S, P)$ mit den Produktionen $S \to \varepsilon \mid (S)S$.

Turingmaschinen

Komplexität

Wir betrachten zuerst nur Turingmaschinen, die bei jedem Eingabewort halten!

Zeitkomplexität

Definition

Die Zeitkomplexität Time(n) einer Turingmaschine ist die maximale Anzahl an Schritten, die eine Turingmaschine bei Eingabe eines Worts der Länge n benötigen kann (worst-case).

Beispiel: Überprüfung auf Palindrom:

- Erstes Symbol -> letztes Symbol (n)
- Zurück zum ersten (n)
- Mit kürzerem Wort wiederholen ($|n-2|$)

Also insgesamt

$$T(n) \leqslant 2n + 1 + T(n-2)$$

Daraus folgern wir

$$T(n) \in O(n^2)$$

Platzkomplexität

Definition

Die Platzkomplexität Space(n) einer Turingmaschine ist die maximale Anzahl an Feldern, die eine Turingmaschine bei Eingabe eines Worts der Länge n benötigen kann (worst-case). Benötigt wird ein Feld, wenn es vom Schreibkopf besucht wird oder von der Eingabe belegt wurde.

Beispiel: Überprüfung auf Palindrom:

- Erstes Symbol -> letztes Symbol $(n+1)$
- Zurück zum ersten (0)
- Mit kürzerem Wort wiederholen (0)

Also insgesamt

$$n+1 \in \Theta(n)$$

Platzkomplexität

Wie hängen Zeit- und Platzkomplexität zusammen?

- Wenn eine TM nur n Schritte macht, kann sie auch nur n Felder besuchen
- Aber anders herum: Auch mit n Feldern können exponentiell viele Schritte durchgeführt werden.
  Beispiel: Band mit 0^n. Solange inkrementieren (siehe Aufgabe), bis 1^n auf dem Band steht. Also $2^n - 1$ mal inkrementieren, somit mindestens $2^n - 1$ Schritte.

Definition

- **P** ist die Menge aller Entscheidungsprobleme, die von Turingmaschinen entschieden werden können, deren Zeitkomplexität polynomiell ist.
- **PSPACE** ist die Menge aller Entscheidungsprobleme, die von Turingmaschinen entschieden werden können, deren Raumkomplexität polynomiell ist.

Es ist

$$\mathbf{P} \subset \mathbf{PSPACE}$$

Die andere Richtung ist **ein großes offenes Problem**
Wir haben zwar ein Beispiel für eine TM mit polynomiellen Platz und exponentieller Zeit gesehen, aber das **Problem** (alle 0 zu 1 umwandeln) hätte man deutlich effizienter lösen können.

Aufgabe

Die Turingmaschine T mit Anfangszustand S sei durch folgende Überführungsfunktion gegeben

	S	S_a	S_b	R
a	(X, S_a, L)	(a, S_a, L)	(a, S_b, L)	(a, R, R)
b	(X, S_b, L)	(b, S_a, L)	(b, S_b, L)	(b, R, R)
X	(X, S, R)	(X, S_a, L)	(X, S_b, L)	(X, S, R)
$\square$	-	(a, R, R)	(b, R, R)	-

Tipp: Manchmal hilft es, die TM zu zeichnen.

- Was steht bei der Eingabe des Wortes $w \in \{a, b\}^*$ am Ende der Berechnung auf dem Band?
- Welche Platzkomplexität hat T?
- Geben Sie eine einfache Funktion $f : \mathbb{N}_0 \to \mathbb{N}_0$ an, so dass die Zeitkomplexität von T in $\Theta(f(n))$ liegt.

Aufgabe

- Was steht bei der Eingabe des Wortes $w \in \{a, b\}^*$ am Ende der Berechnung auf dem Band?
  Lösung: Am Ende steht das Wort $R(w)\mathrm{X}^{|w|}$ auf dem Band, wobei $R(w)$ das Spiegelbild von w ist.
- Welche Platzkomplexität hat T?
  Lösung: Eingabe der Länge n: Platzbedarf ist $2n + 1$
- Geben Sie eine einfache Funktion $f : \mathbb{N}_0 \to \mathbb{N}_0$ an, so dass die Zeitkomplexität von T in $\Theta(f(n))$ liegt.
  Lösung: Eingabe der Länge n: Zeitbedarf in $\Theta(n^2)$

Unentscheidbare Probleme

Es existieren Probleme, die von keiner Turingmaschine entschieden werden können.
Erinnerung: Entscheidbar heißt, dass es eine Turingmaschine gibt, die für jede Eingabe hält und entscheiden kann, ob das Wort in der Sprache liegt oder nicht. Statt Sprachen spricht man auch gerne von Problemen.

Achtung: Unentscheidbar meint wirklich „hält für manche Eingaben nie“ und nicht „braucht $2^{2^{2^n}}$ Zeit“.

Codierung von Turingmaschinen

In der VL: Codierung einer Turingmaschine als Wort über dem Alphabet $\{0, 1, [,]\}$ (Gödelisierung)

Satz

Es existiert eine universelle Turingmaschine U, die für zwei Eingaben $[w_1][w_2]$

- *überprüft ob* w_1 *eine Turingmaschine T codiert*
- *falls ja, die Eingabe* w_2 *auf dieser Turingmaschine simuliert*
- *Das Ergebnis davon berechnet (falls T hält)*

Halteproblem

Satz

Es ist nicht möglich eine Turingmaschine H zu bauen, die für jede Turingmaschine T und jede Eingabe w entscheidet, ob T bei der Eingabe von w hält.

Beweis

Sei eine Tabelle x_i, f_j gegeben, wobei die x_i alle Codierungen einer Turingmaschine sind und die f_j die berechneten Funktionen der Turingmaschine T_j sind. Sei jetzt H eine TM, die das Halteproblem löst und G eine Maschine, die

- Wenn H mitteilt, dass $T_{x_i}(x_i)$ hält, dann geht G in eine Endlosschleife.
- Wenn H mitteilt, dass $T_{x_i}(x_i)$ nicht hält, dann hält G

Jede mögliche Turingmaschine T_{x_i} verhält sich also für die Eingabe x_i genau anders wie G. Also ist G eine Turingmaschine, die nicht in der Tabelle liegt, aber in der Tabelle sind alle TMs enthalten, da diese ja abzählbar sind. Widerspruch!

Beweis 2

Sei wieder H eine TM, die das Halteproblem löst und G eine Maschine, die

- Wenn H mitteilt, dass $T_{x_i}(x_i)$ hält, dann geht G in eine Endlosschleife.
- Wenn H mitteilt, dass $T_{x_i}(x_i)$ nicht hält, dann hält G

Also gilt:

$$G \text{ hält für Eingabe } w \iff T_w \text{ hält nicht für Eingabe } w$$

Setzen wir nun für w die Codierung von G ein, so erhalten wir:

$$G \text{ hält für Eingabe } w \iff G \text{ hält nicht für Eingabe } w$$

Widerspruch!

Fleißige Biber

Definition

Ein fleißiger Biber ist eine Turingmaschine, die $n+1$ Zustände hat, wobei ein Anfangszustand und ein Haltezustand darunter sind und die nur Einsen produzieren kann.

Definition

Als Busy-Beaver-Funktion $bb(n)$ wird die maximale Anzahl an Einsen bezeichnet, die ein fleißiger Bieber mit $n+1$ Zuständen auf dem Band hinterlassen kann.

Satz

Die Busy-Beaver-Funktion ist nicht berechenbar (es gibt keine Turingmaschine, die die Funktionswerte als Ausgabe liefert).

Turingmaschinen: Klausur

Noch ein Hinweis zum Schluss:
Bisher kam in **jeder**, wirklich jeder Klausur eine Aufgabe zu Turingmaschinen dran.
Diese gibt meist relativ viele Punkte.

Zu 99 % wird auch dieses Mal wieder eine TM-Aufgabe drankommen.
Also übt das! Hier zählt vor allem Geschwindigkeit (und Präzision).

Was ihr nun wissen solltet

- Turingmaschinen
- Komplexität
- Entscheidbarkeit – Wir können nicht alles lösen!

Und so geht es weiter...

- Algorithmen I – Mehr zu Algorithmen, Laufzeiten, Datenstrukturen, Graphen
- TI – Realisierung von Schaltungen, Prozessoren (MIMA, ...)
- TGI – Mehr zu Komplexität, Entscheidbarkeit, Turingmaschinen

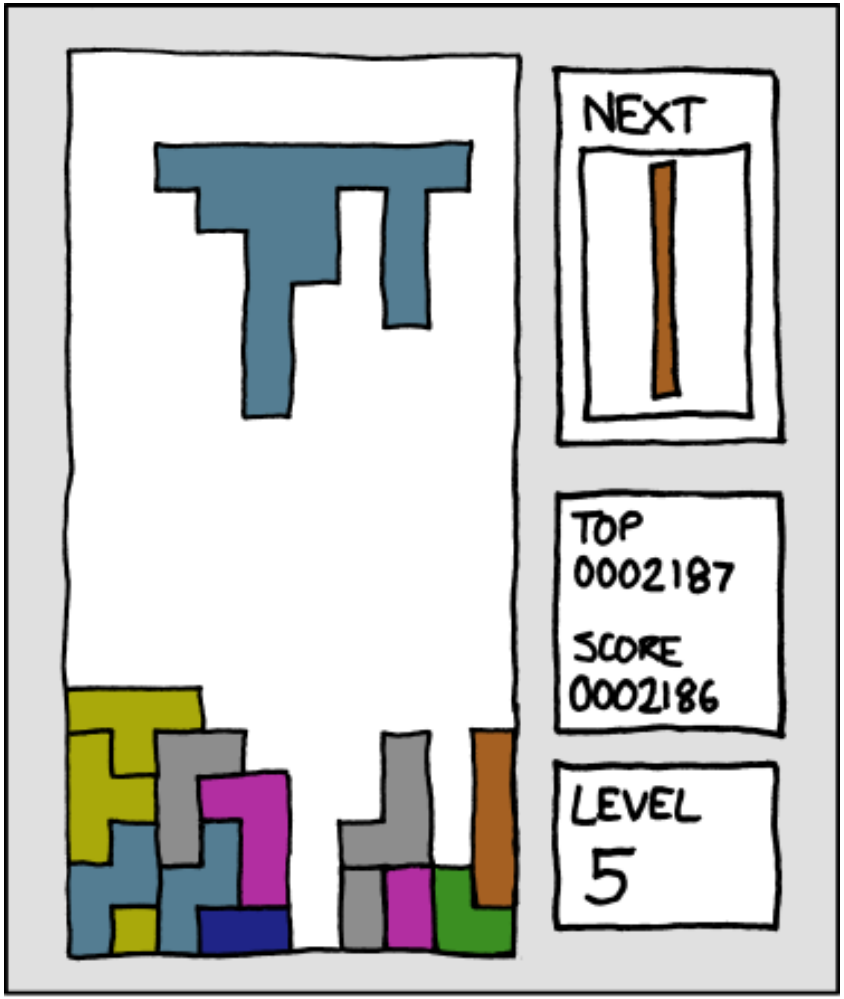
Falls ihr mehr wollt...

Persönliche Empfehlungen

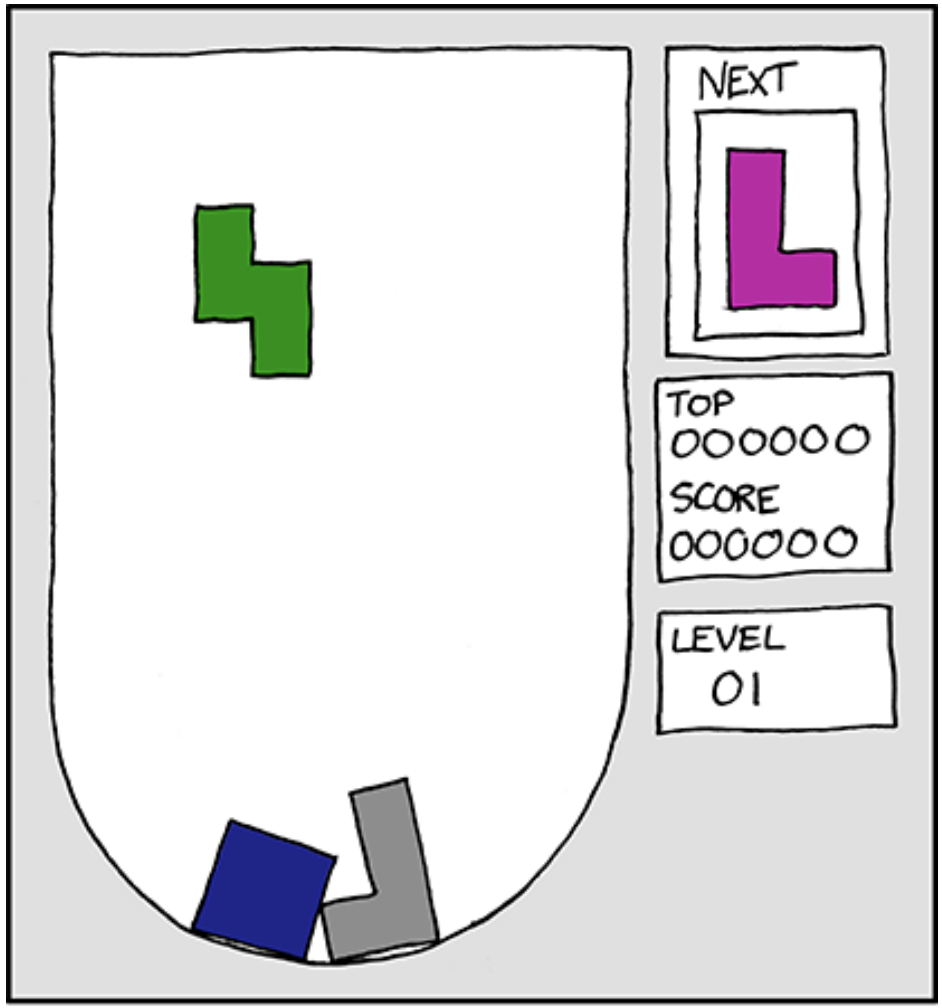
- Design and Analysis of Algorithms (für Algorithmen I)
- CS50x
- From Nand to Tetris

- EDX (edx.org)
- Coursera (coursera.org)

Das war GBI



HEAVEN

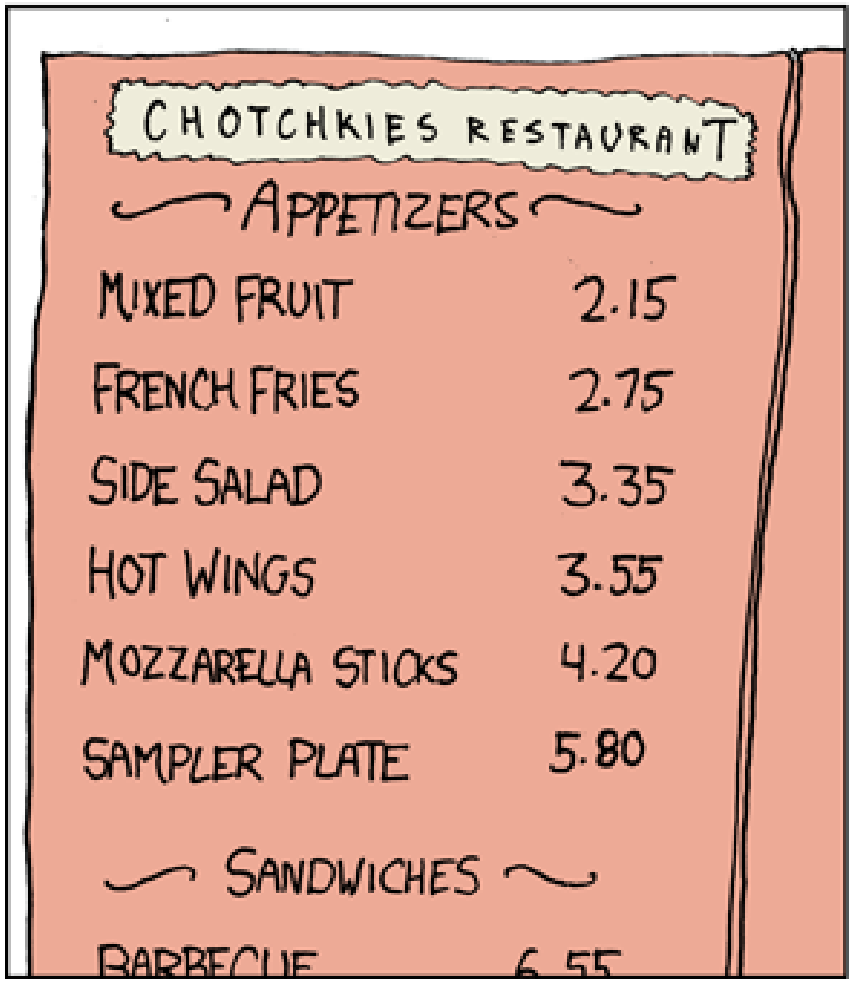


HELL

Viel Erfolg bei der Klausur!

My Hobby:

Embedding NP-Complete Problems in Restaurant Orders



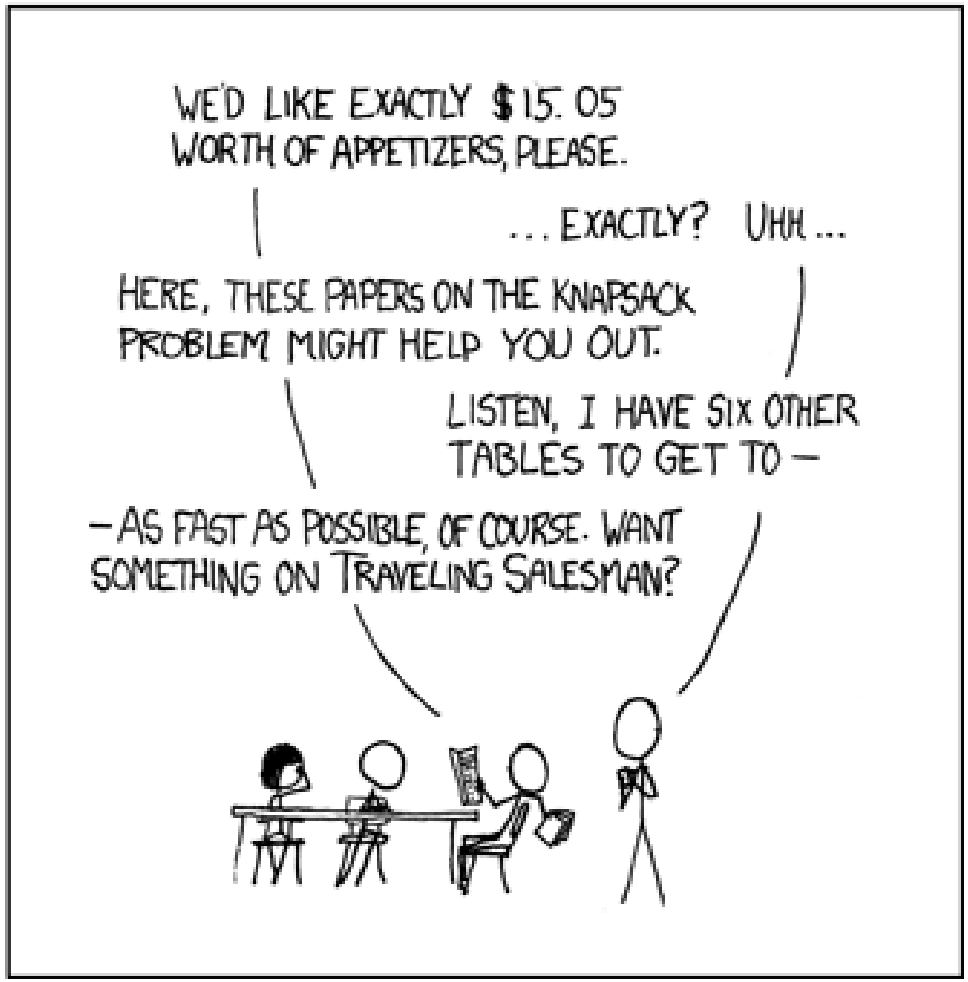


Abbildung: http://www.xkcd.com

Credits

Vorgänger dieses Foliensatzes wurden erstellt von:

Thassilo Helmold
Philipp Basler
Nils Braun
Dominik Doerner
Ou Yue