EINFÜHRUNG IN DIE THEORETISCHE INFORMATIK - SCRIPT

https://github.com/C0d3Crush/ITH-Script
Lukas.Dzielski@stud.uni-heidelberg.de

28. Mai 2023

Inhaltsverzeichnis

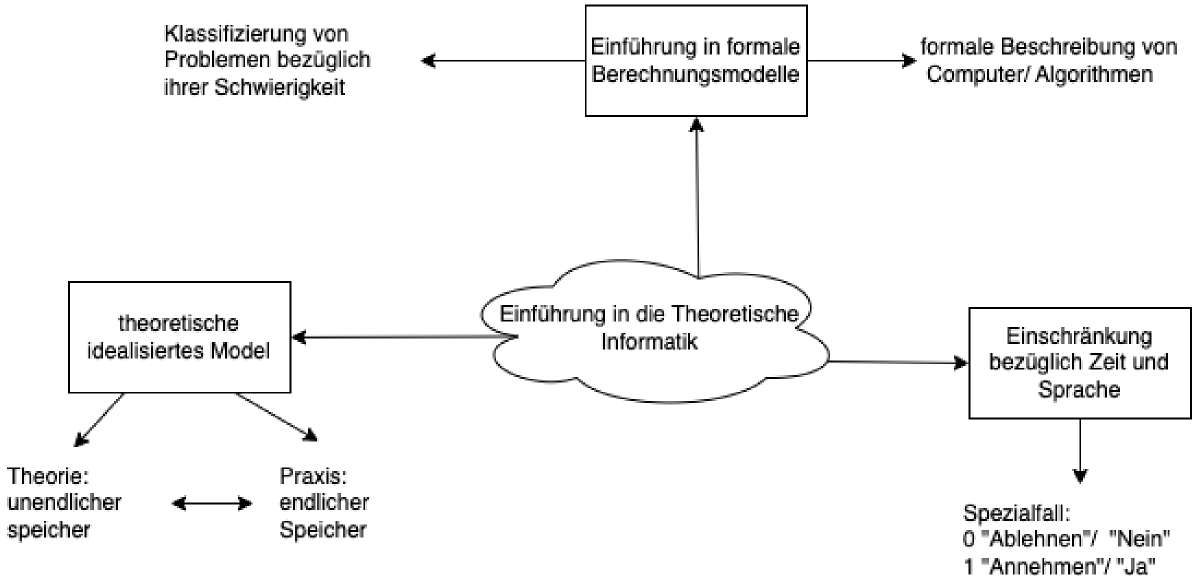


Abbildung 1: Überblick theoretische Informatik

1 Grundlagen

1.1 Notationen und begriffe

- $\mathbb{N}$ bezeichnet die $\{1, 2, 3\}$
- $\mathbb{N}_0$, sei $[n]$ = {1,…,n} und $[n]_0$ = {0, 1, …,n}
- Für eine Menge A und n$\in \mathbb{N}$ ist $A^n = \{(a_1, \dots, a_n)$: $a_1, \dots\ a_n \in$ A}
- Für n $\in \mathbb{N}$ ist eine n-äre partielle funktion $\varphi : A^n \rightsquigarrow B$ eine Funktion mit dom(φ) $\supseteq A^n$ und Im(φ) $\subseteq$ B. Für a_1, ..., $a_n \in$ A bedeuted $\varphi(a_1, ..., a_n)\downarrow$, dass ($a_1$, ..., a_n) $\in$ dom(φ) gilt und $\varphi(a_1, ..., a_n)\uparrow$ bedeutet, dass (a_1, ..., a_n) $\notin$ dom(φ). Statt $\varphi(a_1, ..., a_n)\uparrow$ schreiben wir auch $\varphi(a_1, ..., a_n) = \uparrow$. Die partielle Funktion φ ist total, wenn dom(φ) = A^n gilt.
- Eine lineare Ordnung, auch totale Ordnung, auf einer Menge A ist eine Relation $\leq \subseteq A^n$m sodass die folgende Eigenschaften erfüllt sind. (wie für Relationen üblich verwenden wir hier Infixntation, schreiben also für a,b $\in$ A den Ausdruck a $\leq$ b anstatt (a,b) $\in \leq$):
  (i) a $\leq$ a $\forall$ a $\in$ A (Reflexivität)
  (ii) a $\leq$ b $\wedge$ b $\leq$ a $\Rightarrow$ a = b $\forall$a,b $\in$ A (Antisymetrie)
  (iii) a $\leq$ b, b $\leq$ c $\Rightarrow$ a $\leq$ c $forall$a,b,c $\in$ A (Transitiität)
  (iv) a $\leq$ b $\vee$ b $\leq$ a $\forall$a,b $\in$ A (Totalität)

1.2 Alphabet, Wörter und Sprachen

Eingaben und Ausgaben in unseren Berechnungsmodellen werden wörter genannt, wobei wir beliebige Zeichenketten als Wörter zulassen.

1.2.1 Definition (Alphabet)

Ein Alphabet ist eine nichtleere endliche Menge Σ. Das Alphabet Σ wird $|\Sigma|$- är bezeichnet. Die Elemente von Σ heißen Buchstaben oder Symbole.

1.2.2 Definition (Wörter)

Ein Wort über einem Alphabet Σ ist eine endliche Folge von Symbolen aus Σ. Die Länge eines Wortes w ist |w|. Für i ∈ |w| bezeichnet w(i) das i-te Element von w und für Symbole $a_1, \cdots, a_n \in \Sigma$ bezeichnet $a_1, \cdots, a_n$ das Wort w der Länge n mit w(i) das i-te Element von w und für Symbole $a_1, \cdots, a_n \in \Sigma$ bezeichnet $a_1, \cdots, a_n$ das Wort w der Länge n mit w(i) = a_i ∀i∈ $[n]$. Das Wort der Länge 0 heißt leeres Wort und wird λ bezeichnet. Ein Wort der länge 1 wird mit dem Symbol w(1) identifiziert.

1.2.3 Definition (Binäraphabet, Binärwörter)

Das Alphabet $\{0, 1\}$ heißt Binäralphabet. Die Wörter über dem Binäralphabet heißen Binärwörter.

1.2.4 Sprache

Eine **Sprache** ist eine Menge von Wörter über einem gemeinsahmen Alphabet Σ. Einige einfache grundlegenden Sprachen sind die folgenden.

1.2.5 Definition

Die Menge Aller Wörter über Σ wird mit Σ^* bezeichnet. Für n ∈ $\mathbb{N}_0$ setzen wir:

$\Sigma^{\leq n}$:= {w ∈ Σ^* : |w| ≤n}

$\Sigma^{=n}$:= {w ∈ Σ^* : |w| =n}

$\Sigma^{\geq n}$:= {w ∈ Σ^* : |w| ≥n}

$\Sigma^+ := \Sigma^{\leq 1}$

1.2.6 Definition (Verkettung)

Für Wörter w_1, w_2 ist die verkettung $w_1 \circ w_2$, auch $w_1 w_2$, von w_1 und w_2 ist definiert durch:

$$w_1 \circ w_2 := w_1 \cdots w_1(|w_1|) w_2 \cdots w_2(|w_2|)$$

Für ein Wort w und n ∈ $\mathbb{N}_0$ ist w^k induktiv definiert durch w^n := λ falls n = 0 und w^n := $w^{n-1} \circ w^n$ falls n ≥ 1. Für eine Sprachen L_1, L_2 sei durch $L_1 \circ L_1$, auch $L_1 L_2$ definiert durch

$$L_1 \circ L_1 := \{w_1 w_2 : w_1 \in L_1, w_2 \in L_2\}$$

Für eine Sprache L und und n ∈ $\mathbb{N}_0$ ist L^n moduliert definiert durch $L^n = \{\lambda\}$ falls n = 0 und L^n := L · L^{n-1} falls n ≥ 1. Zudem sei $L^* := \bigcup_{n \in \mathbb{N}_0} L^n$. Für ein Wort w und eine Sprache L sei wL :={w}∘L und Lw := L∘ {w}.

Wir folgen der Konventrion, dass $\bullet^n$ und $\bullet^*$ stärker binden als 0;?? für Wörter u, v gilt also uv = u ∘ (v^n). Insbesondere gilt auch ab^n = a(b^n) für Symbole a, b eines Alphabets Σ.

1.2.7 Definition (Präfix, Infix, Suffix)

Seiene u, v Wörter.

(i) u ist Präfix von v, kurz u ⊑v, falls es ein Wort w gibt ,sodass $uw = v$.

(ii) u ist Infix von v falls es Wörter w_1, w_1 gibt sodass v = $w_1 u w_2$

(iii) u ist Suffic von v, falls es ein Wort w gibt, sodass $v = wu$.

1.2.8 Definition (präfixfrei)

Eine Sprache heißt **präfixfrei**, wenn $u \sqsubseteq v \Rightarrow u = v \forall u, v \in L$.

1.2.9 Definition (Homomorphismus)

Für Sprache L und M heißt eine Funktion $\varphi : L \to M$ **Homomorphismus von Sprachen**, wenn $\varphi(uv) = \varphi(u)\varphi(v) \forall u, v \in L$ gilt.

1.2.10 Definition (Längenlexikographische Ordnung)

Ist Σ ein Alphabet und $\leq$ eine lineare Ordnung auf Σ, so ist die zu $\leq$ gehörige **längenlexikographische Ordnung** $\leq_{llex}$ auf Σ^* die lineare Ordnung für die $u \leq_{llex} v$genau dann für zwei verschiedene $u, v \in \Sigma^*$ gilt, wenn eine der folgenden Bedingungen gilt:

- $|u| < |v|$
- $|u| = |v|$ und ist $i \in [|u|]$ minimal mit $u(i) \neq v(i)$m so gilt $u(i) \leq v(i)$.

Bemerkung: Oft gehen wir von einer impliziten Ordnung auf Σ aus. Ist $\Sigma = a_1, \cdots, a_n$ so gilt $a_1 \leq \cdots \leq a_n$

1.2.11 Bemerkung

Sei Σ ein Alphabet $\forall w \in \Sigma^*$ ist $v \in \Sigma^* : v \leq_{llex} w$ endlich. Dies erlaubt es uns für ein Alphabet Σ die Wörter über Σ in längenlexilographischen Reihenfolge $w_1, w_2, \cdots$ zu betrachten, wobei wir w_i für $i \in \mathbb{N}$ als kleinstes Element von $\Sigma^*/w_1, \cdots, w_{i-1}$ gewählt sei. Wir identifizieren oft $\mathbb{N}_0$ mit $0, 1^*$ indem wir $i \in \mathbb{N}_0$ mit in die längenlexilographische Reihenfolge $(i+1)$-ten Wort $w_{i+1} \in 0, 1^*$ identifiziernen.

$\mathbb{N}_0$	0	1	2	3
$\{0,1\}^*$	λ	0	1	00

1.2.12 Definition

Es bezeichnet $bin : \mathbb{N}_0 \to \{0, 1\}^*$ die Funktion, für die $bin(i)$ das in längenlexikographischer Reihenfolge (i+1)-te Binärwort ist $\forall i \in \mathbb{N}_0$

1.2.13 Bemerkung

$\forall i \in \mathbb{N}_0$ ist $1bin(i)$ die Binärdarstellung von i+1. Umgekehrt ist $\forall w \in 0, 1^*$ das $(2^{|w|} + \sum_{i \in [|w|]} w(i) 2^{|w|-i})$-te Binärwort.

Turingmachine

A Turing machine is like a wise old person, sitting at an endless table, playing a complex game. They have a magical pen that reads and writes on the game board. They follow strict rules, do not move from their spot, but the table mysteriously moves back and forth. Their concentration is deep and calm as they perform a complex ballet of reading, writing, and state-changing.
- ChatGPT

2 Turingmachine

Wir Betrachte das folgende, sehr bekannt, berechnunsmodell. Anschaulich lässt es sich wie folht beschreiben.

- Es gibt einen SSpeicher"$\rightsquigarrow$ k unendlich lange Arrays(**Bänder**)
- Es gibt einen Ärbeitsspeicher"$\rightsquigarrow$ eine endliche Menge von Zusänden, die die Machine einnehmen kann
- Für jedes Band gibt es einen Schreib- und Lesekopf
- Jeder Schritt ist wie folgt:
  Abhängig von Zustand und gelesenene Symbol, Schreiben die Küpfe genau ein Symbol, bewegen sich nun maximal eine Position und der Zustand der Machine wird geändert.
- Stellt die Machine ihhr schrittweises Arbeiten ein, so wird die Ausgabe entweder den Zustand entnommen oder von einem der Bänder in geeigneter Weise abgelesen.

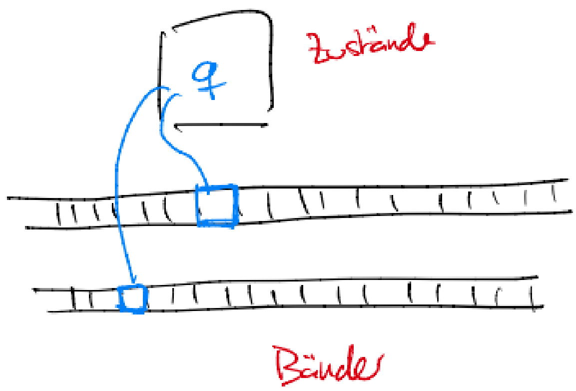


Abbildung 2: Turingmachine.

2.1 Definition (Turingmachine, Alan Tuing, 1936)

Sei $k \in \mathbb{N}$ eine **k-Band-Turingmachinem** kurz k-TM, ist ein Tupe $M = (Q, \Sigma, \Gamma, \Delta, s, F)$. Dabei ist:

- Q eine endliche Menge, **Zustandmenge**
- Σ das **Eingabealphabet**, ein Alphabet $\Box \notin \Sigma$
- Γ das **Bandaphabet**, ein Alphabet mit $\Sigma \subseteq \Gamma$ und $\Box \in \Gamma / \Sigma$
- $\Delta \subseteq Q \times \Gamma^k \rightarrow \subseteq Q \times \Gamma^k \times L, S, R^k$ die **Übergangsrelation**
- $s \in Q$ der **Startzustand**
- $F \subseteq Q$ die Menge der **akzeptierenden Zustände**

Das Symbol $\Box$ heißt **Blank**. Die Elemente von Δ heißen **instruktionen**. Für eine Instruktion $(q_1, a_1, \cdots, a_k, q', a'_1, \cdots, a'_k, B_1, \cdots, B_k)$ **Anweisungteil**. Die TM M ist eine **deterministische k-Band Turingmachine**, kurz k-DTM, wenn es $\forall b \in Q \times \Gamma^k$ höchstens eine Instruktion $i \in \Delta$ mit Bedingungsteil b.

2.2 Definition (Konfiguration)

Sei $M = (Q, \Sigma, \Gamma, \Delta, s, F)$ eine k-TM. EIne **Konfigration** von M ist ein Tupel

$$C = (q, w_1, \cdots, w_k, p_1, \cdots, p_k) \in Q \times (p^*)^k \times \mathbb{N}^k$$

Die **Startkonfiguration** von M zur Eingabe $(u_1, \cdots, u_n) \in (\Sigma^*)^n$, wobei $n \in \mathbb{N}$, ist die Konfiguration

$$Start_M(u_1, \cdots, u_n) = (s, u_1 \Box u_2 \Box \cdots \Box u_n, \Box, \cdots, 1, \cdots, 1)$$

Die Konfiguration C ist eine **Stoppkonfigration** von M, wenn es keine Instruktion $i \in \Delta$ mit Bedingungsteil $(q, w_1(p_1), \cdots, w_k(p_k))$ gibt.

2.3 Definition (Nachfolgekonfiguration)

Sei $M = (Q, \Sigma, \Gamma, \Delta, s, F)$ eine k-DTM. Für Konfiguration $C = q_1, w_1, \cdots, w_k, p_1, \cdots, p_k$ und $C' = q'_1, w'_1, \cdots, w'_k, p'_1, \cdots, p'_k$ von M ist die Konfigration C' Nachfolgekonfiguration von C, wenn es eine Instruktion

$$(q, w_1(p_1), \cdots, w_k(p_k), a'_1, a'_k, B_1, \cdots, B_k) \in \Delta$$

gibt, sodass

$$w'_i = \begin{cases} \Box a'_i w_i(2) \cdots w_i(|w_i|), & \text{falls } p_i = 1 \text{und} B_i = L \\ w_i \cdots w_i(|w_i| - 1) a'_i \Box, & \text{falls } pi = |w_i| \text{und} B_i = R \\ w_i \cdots w_i(p_i - 1) a'_i w_i(p_i + 1) \cdots w_i(|w_i|), & \text{sonst} \end{cases}$$

und

$$p'_i = \begin{cases} 1, & \text{falls } p_i = 1 \text{ und } B_i = L \\ p_i - 1, & \text{falls } p_i \geq 2 \text{ und } B_i = L \\ p_i, & \text{falls } B_i = S \\ p_i + 1, & \text{falls } B_i = R \end{cases}$$

$\forall i \in [k]$ gelten.
Es bezeichnen $\to M$ die Relation auf der Menge der Konfiguration von M, sodass $C \to_M C'$ falls C, C' Konfig von M sind wobei C' eine Nachfolgekonfiguration von C ist.

2.4 Definition (Rechnung)

Sei $M = (Q, \Sigma, \Gamma, \Delta, s, F)$ eine k-DTM. Eine **endliche partielle Rechnung** von M ist eine endliche Folge $C_1, \cdots, C_n$ von Konfig von M mit $C_i \to_M C_{i+1} \forall i \in [n-1]$. Eine **unendliche partielle Rechnung** von M ist eine unendliche Folge $C_1, C_2, \cdots$ von Konfigration von M mit $C_1 \to_M C_{1+1} \forall i \in \mathbb{N}$. Eine **Rechnung von M zur Eingabe** $(w_1, \cdots, w_n) \in (\Sigma^*)^n$ (mit $n \in \mathbb{N}$) ist eine endliche partielle Rechnung $start_M = C_1, \cdots, C_m$ bei der C_m eine Stoppkonfiguration von M oder eine unendliche partielle rechnung $start_M(w_1, \cdots, w_n) = C_1, C_2, \cdots$

2.5 Bemerkung

Ist M eine k-DTM, so gilt es $\forall n \in \mathbb{N}$ und $(w_1, \cdots, w_n) \in (\Sigma^*)^n$ genau eine Rechnung zur Eingabe $(w_1, \cdots, w_n)$.

2.6 Definition (total)

Eine k-DTM $M = (Q, \Sigma, \Gamma, \Delta, s, F)$ **terminiert** bei Eingabe $(w_1, \cdots, w_n) \in (\Sigma^*)^n$ wenn die Rechnung von M zur Eingabe $(w_1, \cdots, w_n)$ endlich ist. Eine k-TM $M = (Q, \Sigma, \Gamma, \Delta, s, F)$ ist **total**, wenn $\forall n \in \mathbb{N}$ und $(w_1, \cdots, w_n) \in (\Sigma^*)^n$ alle Rechnungen von M zur Eingabe $(w_1, \cdots, w_n)$ endlich sind.

2.7 Definition (akzeptierte Sprache)

Sei $M = (Q, \Sigma, \Gamma, \Delta, s, F)$ eine k-TM. Eine Stoppkonfiguration $(q, w_1, \cdots, w_k, p_1, \cdots, p_k)$ von M ist **akzeptierend**, wenn $q \in F$. Die **akzeptierte Sprache L(M)** von M ist die Sprache über dem Alphabet Σ so dass $w \in L(M)$ gilt, wenn es eien endliche Rechnung $C_1, \cdots, C_n$ von M zur Eingabe w gibt, bei der C_n eine akzeptierende Stoppkonfigration von M ist.
Hinweis: Für nicht deterministische TM heißt das insbesondere, dass es für die Wörter w in der akzeptierten Sprache nur mindestend **eine** im einer akzeptierten Stoppkonfigration endende endliche Rechnung zur Eingabe w geben muss. Für Wörter w, die nicht in L(M) sind, sind **alle** rechnungen von M zur Eingabe am Ende nicht in einer akzeptierten Stoppkonfigration oder unendlich.

2.8 Definition(entscheidbar)

Eine Sprache L ist genau dann **entscheidbar**, wenn es eien totale k-TM M mit L(M) = L gibt. Wir schreiben **REC** für die Klasse der entscheidbaren Sprachen. Der Begriff entscheidbar für Sprachen ergibt sich hier daraus, dass effektiv entschieden werden kann ob eine gegebene Eingabe in der Sprache liegt oder nicht. Insbesondere steht? dies voraus, dass Eingabe, die nicht in der Sprache liegen effektiv als nicht in der Sprache liegend erkannt werden.
Begriff: effektiv $\rightsquigarrow$ eine TM erlefigt dies in endicher Zeit. Da sich der durch TM formatierte Berechenbarkeitsbegriff, also die Formalisierung dessen was effektiv durchführbar ist, auch äquivalent durch rekursive Funktion definieren lässt, weden entscheidbare Sprachen auch als rekuriv bezeichnet.

2.9 Definition(rekursiv aufzählbar)

Eine Sprache L ist genau dann **rekursiv aufzähbar**, wenn es eine k-TM mit akzeptierten Sprache L gibt. Wir schreiben **RE** für die Klasse der rekursiv aufzählbaren Sprachen. Die Aufzählbarkeit leitet sich daraus ab, dass es für eine rekuriv aufzählbare Sprache L über einem Alühabet Σ möglich ist effektive Verfahren anzugeben ,die die Wörter von L aufzählen, also dass eine endlich oder unendliche Aufzählung von $A = w_1, w_2, \cdots$ mit $w_1, w_2, \cdots = L$ existiert.

"Rekursiv aufzählbarïst ein Begriff der verwendet wird um eine Menge zu beschreiben, die wir mit einem Computerprogramm oder Algorithmus äuflisten"können. Stellen Sie sich vor, Sie haben eine Box mit nummerierten Bällen, und Sie haben ein Programm, das Bälle aus der Box zieht. Wenn Sie sicherstellen können, dass Sie jeden Ball in der Box mindestens einmal ziehen, egal wie lange es dauert, dann ist die Menge der Bälle in der Box "rekursiv aufzählbar

Wenn wir sagen, dass eine Sprache "rekursiv aufzählbarïst, bedeutet das, dass es einen Algorithmus oder ein Computerprogramm gibt, das alle Wörter in dieser Sprache äuflisten"kann. Es könnte einige Wörter mehrmals auflisten und es könnte eine sehr lange Zeit dauern, aber es würde schließlich jedes Wort in der Sprache "treffen". Eine "k-TMïst eine Art von Maschine, die wir in der theoretischen Informatik verwenden, um diese Art von Aufzählung zu machen. Wenn es eine k-TM gibt, die eine Sprache akzeptiert, bedeutet das, dass die Sprache rekursiv aufzählbar ist.

2.10 Bemerkung

Jede entscheidbare Sprache ist rekursiv aufzähbar.

2.11 Bemerkung

Alle endlichen Sprachen sind entscheidbar.

2.12 Bemerkung

Eine Sprache L über einem Alphabet Σist genau dann entscheidbar, wenn L und $L^c := (\Sigma^*)/L$ rekursiv aufzählbar sind.

2.13 Definition (Ausgabe)

Sei $M = (Q, \Sigma, \Gamma, \Delta, s, F)$ eine k-TM und $C = (q, w_1, \cdots, w_k, p_1, \cdots, p_k)$ eine Konfigration von M. Die Ausgabe $out_M(C)$ von M bei Konfiguration C ist das Präfix $w \sqsubseteq w_1(p_1), \cdots, w_1(|w_1|)$ maximale Länge mit $w \in (\Gamma/\Box)^*$.

2.14 Definition (berechnete Funktion)

Sei $M = (Q, \Sigma, \Gamma, \Delta, s, F)$ eine k-DTM und $n \in \mathbb{N}$. Die von M berechnete **n-äre partielle Funktion Φ_M** ist die partielle Funktion $\Phi_M : (\Sigma^*)^n \rightsquigarrow (\Gamma/\Box)^*$, so dass $\forall (w_1, \cdots, w_n) \in (\Sigma^*)^n$ folgendes gilt:

1. Ist die rechnung von M zur Eingabe $(w_1, \cdots, w_n)$ die endliche Rechnung $C_1, \cdots, C_M$, so gilt $\Phi_M(w_1, \cdots, w_n) = out_M(C_M)$.
2. Ist die Rechnung von M zur Eingabe $(w_1, \cdots, w_n)$ unendlich, so gilt $\Phi_M(w_1, \cdots, w_n) \uparrow$

Für $w_1, \cdots, w_n \in \Sigma^*$ schreiben wir statt $\Phi_M(w_1, \cdots, w_n)$ auch $M(w_1, \cdots, w_n)$.

2.15 Definition (partiell berechenbar)

Für Alphabet Σ, Γ und eine partielle Funktion $\Phi : \Sigma^* \rightsquigarrow \Gamma^*$ ist Φ **partiell berechenbar**, wenn es eine $k \in \mathbb{N}$ gibt und eine k-DTM M mit $\Phi_M = \Phi$ gibt. Ist Φtotal und partiell berechenbar, so ist Φ berechenbar. Wir schreiben **RF** für die Klasse der partiellen Funktionen.

Mittels der Induktivität von $\mathbb{N}_0$ und $0, 1^*$ können so auch partielle Funktionen, die von oder nach $\mathbb{N}_0$ abbilden als (partielle) berechenbare Funktion bezeichnent werden. Beispielsweise ist eine partielle Funktion $\Phi : \mathbb{N}_0 \rightsquigarrow \mathbb{N}_0$ dennoch genau dann partiell berechenbar, wenn die partielle Funktion $bin \circ \Phi \circ bin^-1$ partiell berechenbar ist. Gewissermaßen verfügen die hier definierten TM über zewi Ausgabemechanismen. Die Ausgabeim engeren Sinne in Definition 2.13 und das Ablesen vn Akzeptanz anhand des schließlich erreichten Zustands in Definition 2.7. Im Sinne der folgenden Bemerkung wäre der zweiten Fall nicht notwendig, allerdings ist dies ein wichtiger spezialfall.

2.16 Definition (charackteristische Funktion, partielle charachteristische Funktion)

Sei L eine Sprache über dem Alphabet Σ

(i) Die **charackteristische Funktion** von L als Sprache über Σ ist die Funktion $\mathbb{1}_L : \Sigma \to \{0, 1\}$ mit $\mathbb{1}_L = 1\ \forall w \in L$ und $\mathbb{1}_L(u) = 0\ \forall w \in \Sigma^*/L$.

(ii) Die **partielle charackteristische Funktion** von L als Sprache über Σ ist die partielle Funktion $x_L : \Sigma^* \rightsquigarrow \{1\}$ mit $x_L(w) = 1\ \forall w \in L$ und $x_L(w) \uparrow \forall w \in \Sigma^*/L$.

2.17 Bemerkung

Sei L eine Sprache über einem Alphabet Σ.

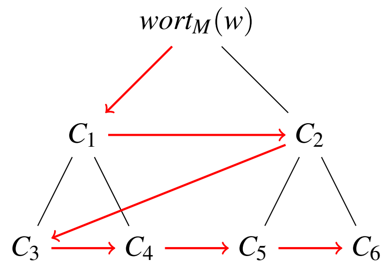
(i) L ist genau dann entscheidbar, wenn $\mathbb{1}_L$ berechenbar ist.

(ii) L ist genau dann rekursiv aufzählbar, wenn x_L partiell berechenbar ist.

2.18 Bemerkung (normiert)

Eine 1-DTM $M = (Q,\Sigma,\Gamma,\Delta,s,F)$ heißt **normiert**, wenn $Q = 0,\cdots,n$ für eine $n \in \mathbb{N}_0$, $\Sigma = 0,1$, $\Gamma = \Box,0,1$, $s = 0$, $F = s$. Alle TMs mit Eingabealphabet $0,1$ lassen sich mit folgenden Schritten in eine normierte TM mit gleicher erkannter Sprache und gleicher berechneter Funktion umwandeln.

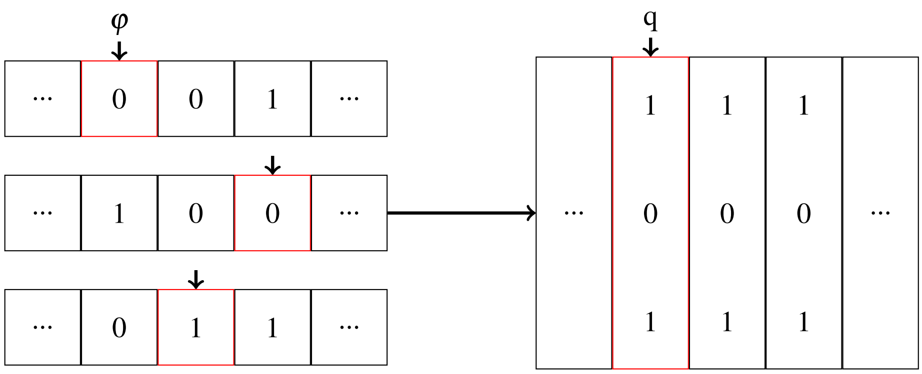
- Von Nichtdeterminismus zu Determinismus: Eine DTM kann die Rechnungen einer nichtdeterministischen TM parallel im Sinne von abwechend schrittweise durchführen um schließlich das Verhalten der simulierten TM zu ??. Dies entspricht einer **Breitensuche im Rechnungsbaum**.



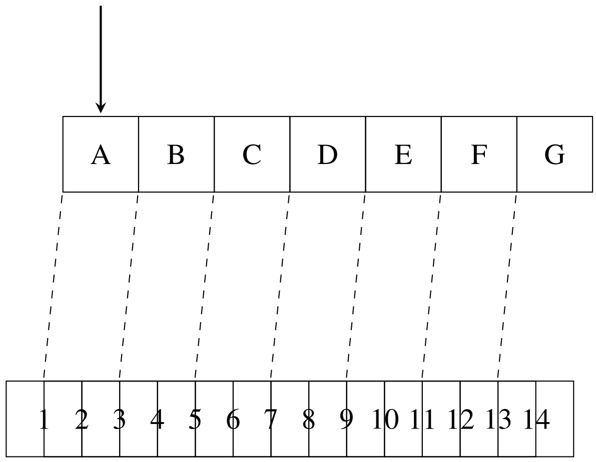
- Von mehreren Bändern zu einem Band: Intuitiv können k Bänder auf ein Band simuliert werden, indem die Felder des einen Bandes in k-teilfelder unterteilt werden, die jeweils die gleiche Bandalphabetbuchstaben wie zufor als Beschreibung zulassen und es zudem erlaubt zu markieren, dass der simulierte Kopf des simulierten Bandes dort steht. Eine dieser Idee folgende Konstruktion wird als **Spurentechnik** bezeichnet. Formal: Übergang vom Bandalphabet Γ zu

$$((\Gamma \cup \underline{a} : a \in \Gamma)^k / \Box^k) \cup \Box$$

wobei $\underline{a} \notin \Gamma$ für $a \in \Gamma$. Hierbei bedeutet a̲, dass das simulierte Feld mit a beschriftet ist und dass dort der simulierte Kopf steht. Weiter spielt $\Box$ die Rolle des k-Tupels $(\Box,\cdots,\Box)$ um der Tatsache gerecht zu werden, dan alle Felderzu Begin mit $\Box$ beschriftet sind.



- Von beliebigen bandalphabet zu $\{\Box,0,1\}$: Andere bandalphabete können bei einem **Alphabetwechel** zum Bandalphabet $\{\Box,0,1\}$ simuliert werden um ein Symbol des vorherigen Bandlaphabets durch ein Binärwort zu beschreiben. Die TM liest stets nur ein Feld, es wird dabei also nötig sein die Zustandsmenge so zu erweitern, dass angrenzende Felder im Zustand gespeichert weden können.



2.19 Bemerkung

Sei $L \subseteq \{0,1\}^*$ eine Sprache und sei $\Phi : \{0,1\}^* \rightsquigarrow \{0,1\}^*$ eine partielle Funktion.

(i) L ist genau dann entscheidbar, wenn L akzeptierte Sprache einer totalen normierten TM ist.

(ii) L ist genau dann rekursiv aufzähbar, wenn L akzeptierte Sprache einer normierten TM ist.

(iii) Φ ist genau dann partiell berechenbar, wenn Φ berechnete Funktion einer normierten TM ist.

2.20 Church- Turing- These

Berechenbarkeit auf eienr Turingmachine entspricht intuitiver Berechenbarkeit.

Berechenbarkeit

Predictability is like knowing the path a river takes. The river starts at its source and flows down to the sea. Along the way, it may turn, twist, and divide, but it always follows the path of least resistance due to gravity. Knowing the terrain allows us to predict where the river will go.
- ChatGPT

3 Berechenbarkeit

Konvention: Sprechen wir von einer $e \in \mathbb{N}_0$ oder $(e_1, \cdots, e_n) \in \mathbb{N}_0^n$ wobei $n \in \mathbb{N}$ als Eingabe für eine TM oder Ausgabe einer TM, so bedetet dies, dass die Eingabe bzw. Ausgabe $bin(e)$ bzw $(bin(e_1), \cdots, bin(e_n))$ ist. Dies erlaubt es über partiell berechnenbare Funktionene $\Phi : \mathbb{N}_0^n \rightsquigarrow \mathbb{N}_0$ wobei $n \in \mathbb{N}$ zu sprechen und $L \subseteq \mathbb{N}_0$ als Sprache über $\{0,1\}$ aufzufassen.

3.1 Definition (Code)

Wir betrachten die Funktion code (mit geeignetem Definitionsbereich) und Zielmenge $\{0,1\}^*$, für die folgendes gilt. Zunächst gelte

$$code(L) = 10 \qquad code(S) = 00 \qquad code(R) = 01$$

Für eine Instruktion $I = (q, a, q', a', B) \in \mathbb{N}_0 \times \{0,1\} \to \mathbb{N}_0 \times \{0,1\} \times \{L, S, R\}$ einer normierten TM sei

$$code(I) = 0^{|bin(q)|} 1 bin(q) a 0^{|bin(q')|} 1 bin(q') a' code(B)$$

Für eine endliche Menge $\Delta \subseteq \mathbb{N}_0 \times \{0,1\} \to \mathbb{N}_0 \times \{0,1\} \times \{L, S, R\}$ von Instruktionen einer normierten TM und $i \in [|\Delta|]$ sein $code_i(\Delta)$ dann ein längenlexikographische Ordnung i-te Wort in $\{code(I) : I \in \Delta\}$ und sei

$$code(\Delta) = code_1(\Delta), \cdots, code_{|\Delta|}(\Delta)$$

Für eine normierte TM $M = (\{0, \cdots, n\}, \{0,1\}, \{\square, 0, 1\}, \Delta, 0, \{0\})$ sei

$$code(M) = 0^{|bin(n)|} 1 bin(n) code(\Delta)$$

der **Code** von M. Relevant ist hierbei dass es eine geeignete effektive Codierung von Turingmachinen durch Binärwörter gibt, so dass folgendes gilt

- Jede normierte TM hat einen Code
- Keine zwei verschiedene normierten TMs haben den gleichen Code.
- Die Sprache der Codes von Turingmachinen ist entscheidbar
- Codes können eine geeignete Repräsentation der durch sie codierten TMs umgewandelt werden, die es insbesondere erlauben die codierten TMs effekiv zu simulieren.
- geignete Repräsentationen von TMs können effektiv in ihre Codes umgewandet werden.

3.2 Definition (standardaufzählung)

Sei $\hat{w}_0, \hat{w}_1, \cdots$ die Aufzählung aller Codes normierter TMs in längenlexikographischer Ordnung. Für $e \in \mathbb{N}_0$ sei M_e die durch $\hat{w}_e$ codierte TM und für $n \in \mathbb{N}$ sei $\Phi_e^n : \mathbb{N}_0^n \to \mathbb{N}_0$ die von M_e berechnete n-äre partielle Funktion. Für $n \in \mathbb{N}$ heißt die Folge (Φ_e^n) mit $e \in \mathbb{N}$ **standardaufzählung** der n-ären partiell berechenbaren Funktion. Für $n \in \mathbb{N}$ und eine partiell berechenbare n-äre Funktion $\varphi : \mathbb{N}_0^n \to \mathbb{N}_0$ heißt jede zahl $e \in \mathbb{N}_0$ mit $\Phi_e^n = \varphi$ **Index** von φ.

Konvention:

Ergibt sich n aus dem Kontext, so schreiben wir auch Φ_e statt Φ_e^n

3.3 Bemerkung

Für $n \in \mathbb{N}$ und eine partielle berechnbare n-äre partielle Funktion $\Phi : \mathbb{N}_0^n \to \mathbb{N}_0$ gibt es unendlich viele Indizes von φ.

3.4 Definition (U)

Es bezeichnet U die normierte TM, die bei Eingabe $(e, x_1, \cdots, x_n) \in \mathbb{N}_0^{n+1}$ wobei $n \in \mathbb{N}$ die normierte TM $\mathscr{M}_e$ bei Eingabe $(x_1, \cdots, x_n)$ simuliert und falls diese terminiert die Ausgabe der Simulierten ausgibt.

3.5 Definition (Universell)

Eine DTM U heißt **Universell**, wenn es für alle $n \in \mathbb{N}$ und alle partiell berechenbaren Funktionen $\varphi : \mathbb{N}_0^n \rightsquigarrow \mathbb{N}_0$ eine $e \in \mathbb{N}$, so dass

$$U(e, x_1, \cdots, x_n) = \varphi(x_1, \cdots, x_n)$$

$\forall x_1, \cdots, x_n \in \mathbb{N}_0$ gilt.

3.6 Bemerkung

Die TM U ist universell, denn für $e \in \mathbb{N}_0$, $n \in \mathbb{N}$ und $x_1, \cdots, x_n \in \mathbb{N}_0$ gilt

$$U(e, x_1, \cdots, x_n) = \Phi_e(x_1, \cdots, x_n)$$

$$(x,y) \mapsto x^y$$

$$y \mapsto 2^y$$

$$(x_1, \cdots, x_m, y_1, \cdots, y_n) \mapsto \varphi(x_1, \cdots, x_m, y_1, \cdots, y_n) partiellberechenbar$$

$$\rightsquigarrow (y_1, \cdots, y_m) \mapsto \varphi(x_1, \cdots, x_m, y_1, \cdots, y_n) partiellberechenbar$$

3.7 Satz (s_n^m - Theorem)

$\forall m, n \in \mathbb{N}$ existiert eine berechenbare Funktion $s_n^m : \mathbb{N}_0^{m+1} \to \mathbb{N}_0$ mit

$$\Phi_e^{m+1}(x_1 \cdots, x_m, y_1, \cdots, y_n) = \Phi^n_{s_n^m(e, x_1, \cdots, x_m)}(y_1, \cdots, y_n)$$

$\forall e, x_1, \cdots, x_m, y_1, \cdots, y_n \in \mathbb{N}_0$

Beweis. Fixiere $m \in \mathbb{N}$. Betrachte die DTM S , die bei Eingabe $(e, x_1, \cdots, x_m) \in \mathbb{N}_0^{m+1}$ wie folgt vorfährt.

- Zunächst bestimmt S den Code von $\mathscr{M}_e$
- der Code von $\mathscr{M}_\rceil$ wird dann in einen Code einer normierten TM $\mathscr{M}$ umgewandet, die zunächst $x_1 \square \cdots \square x_m \square$ neben die Eingabe schreibt, dan den Kopf auf das erste Feld des beschriebenen Bandteilsbewegt und dann wie $\mathscr{M}_\rceil$ arbeitet.
- Es wird bestimmt an welcher Stelle der Standardaufzählung der Code von auftaucht und diese Stelle wird ausgegeben.

Sei s_n^m die von S berechnete $(m+1)$-äre partielle Funktion. Dann ist s_n^m eine Funktion wie gewünscht. Es gibt überabzählbar viele Binärsprachen, denn: Betrachte Aufzählung von Binärsprachen $L_1, L_2, \cdots$

L mit $\mathbb{1}_L(i) = \begin{cases} 0, & \text{wenn } \mathbb{1}_{L_i}(i) = 1 \\ 1, & \text{wenn } \mathbb{1}_{L_i}(i) = 0 \end{cases}$ □

3.8 Definition (diagonales Halteproblem)

Die Menge $H_{diag} := \{e \in \mathbb{N}_0 : \Phi_e(e) \downarrow\}$ heißt **diagonales Halteproblem**.

Proposition 1. *Das diagonale Halteproblem ist rekursiv aufzählbar.*

$$\begin{array}{cccc}
\mathbb{1}_{L_0}(0) & \mathbb{1}_{L_0}(1) & \mathbb{1}_{L_0}(2) & \mathbb{1}_{L_0}(3) \\
\mathbb{1}_{L_1}(0) & \mathbb{1}_{L_1}(1) & \mathbb{1}_{L_1}(2) & \mathbb{1}_{L_1}(3) \\
\mathbb{1}_{L_2}(0) & \mathbb{1}_{L_2}(1) & \mathbb{1}_{L_2}(2) & \mathbb{1}_{L_2}(3)
\end{array}$$

Standardaufzählung

Beweis. Die DTM, die bei Eingabe $e \in \mathbb{N}_0$ wie U bei Eingabe (e,e) arbeitet, aber bei terminieren 1 statt der Ausgabe von U ausgibt berechnet die partielle charachteristische Funktion von H_{diag}. Die partielle Funktion $x_{H_{diag}}$ ist also partiell berechenbar. Die partielle Funktion $x_{H_{diag}^c}$ ist nicht partiell berechenbar, dann: Betrachte Standardaufzählung □

$$\begin{array}{cccc}
\Phi_{L_0}(0) & \Phi_{L_0}(1) & \Phi_{L_0}(2) & \Phi_{L_0}(3) \\
\Phi_{L_1}(0) & \Phi_{L_1}(1) & \Phi_{L_1}(2) & \Phi_{L_1}(3) \\
\Phi_{L_2}(0) & \Phi_{L_2}(1) & \Phi_{L_2}(2) & \Phi_{L_2}(3)
\end{array}$$

Standardaufzählung

$$\varphi \text{ mit } \varphi(i) = \begin{cases} \uparrow, & \text{wenn } \Phi_i(i) \downarrow \\ \downarrow, & \text{wenn } \Phi_i(i) \uparrow \end{cases} \text{Wird nicht aufgezählt.}$$

3.9 Satz

Das diagonale Halteproblem ist nicht entscheidbar.

Beweis. Angenommen H_{diag} wäre entscheidbar. Dann wäre die partielle charakteristische Funktion φ von $H_{diag}^c = \mathbb{N}_0 / H_{diag}$ partiell berechenbar, es gäbe also ein Index $e \in \mathbb{N}_0$ von φ. Es folge

$$e \in H_{diag}^c \Leftrightarrow \varphi(e) \downarrow \Leftrightarrow \Phi_e(e) \downarrow \Leftrightarrow e \in H_{diag} \Leftrightarrow e \notin H_{diag}^c$$

Die ist ein Wiederspruch. □

3.10 m-Reduktion

Für eine Sprache A über einem Alphabet Σ und eine Sprache B über einem Alphabet Γ ist A genau dann **many-one-reduzierbar**, auch **m-reduzierbar**, auf B, kurz $A \leq_m B$, wenn es eine berechebare Funktion. $f : \Sigma^* \to \Gamma^*$ gibt so dass

$$w \in A \Leftrightarrow f(w) \in B$$

$\forall w \in \Sigma^*$ gilt. Gelten $A \leq_m B$ und $B \leq_m A$, so sind A und B **many-one-äquivalent** auch **m-äquivalent**, kurz $A =_m B$.

3.11 Bemerkung

(i) $\leq_m$ ist transitiv.

(ii) Gilt $A \leq_m B$ für Sprachen A und B und ist B entscheidbar, so ist auch A entscheidbar.

(iii) Alle entscheidbaren Sprachen L mit $\varnothing \neq L \neq \mathbb{N}_0$ und m-äquivalent.

3.12 Satz

Das **initiale Halteproblem** $H_{init} = e \in \mathbb{N}_0 = \Phi_e(0) \downarrow$ ist nicht entscheidbar.

Idee:

suche $f : \mathbb{N}_0 \to \mathbb{N}_0$ mit $\Phi_e(e) \downarrow \Leftrightarrow \Phi_{f(e)}(0) \downarrow$ Wähle f so dass $\Phi_{f(e)}(x) = \Phi_e(e)\ \forall x \in \mathbb{N}_0$

Beweis. Sei $\psi : \mathbb{N}_0^2 \rightsquigarrow \mathbb{N}_0$ mit $\psi(e,x) = \Phi_e(e) \forall e,x \in \mathbb{N}_0$. Dann ist ψ partiell berechenbar. Sei e_0 ein Index von ψ und $s : \mathbb{N}_0^2 \to \mathbb{N}_0$ gilt. Sei $f : \mathbb{N}_0 \to \mathbb{N}_0$ mit $f(e) = s(e_0, e) \forall e / in \mathbb{N}_0$. Dann ist f berechenbar. $\forall e \in \mathbb{N}_0$ gilt.

$$e \in H_{diag} \Leftrightarrow \Phi_e(e) \downarrow \Leftrightarrow \psi(e,0) \downarrow \Leftrightarrow \Phi_{e_0}(e,0) \downarrow \Leftrightarrow \Phi_s(e_0,e)(0) \downarrow \Leftrightarrow \Phi_{f(e)}(0) \downarrow \Leftrightarrow f(e) \in H_{init}$$

Es gilt also $H_{diag} \leq_m H_{init}$, da H_{diag} nicht entscheidbar ist, ist damit H_{init} nicht entscheidbar. $\square$

Dominosteinspiel!

Gegeben:

Endlich viele typen von Spielsteinen mit jeweils zwei beschrifteten Feldern: öberes Feld, unteres Feld". Beschritungen sind nichtleere Wörter über einem Alphabet. Spielsteine sind vom gleichen Typ, wenn die beiden oberen Felder gleich beschriftet sind und die beiden unteren Felder gleich beschriftet sind. Es gibt von jedem Typ beliebig viele steine.

Gesucht:

Können ein oder mehrere (aber endlich viele) Spielsteine so nebeneinander gelegt werden, dass sich oben und unten von links nach rechts gelesen das gleiche Wort ergibt?

0111	1	0	0	1
0	01	1	000	011

$\longrightarrow$

0111	0	0	0	0	1
0	1	1	1	000	01

3.13 Definition (Postsches Korrespondenzproblem, Emil Port, 1946)

Für ein Alphabet Σ sei eine Instanz des Postschen Korrespondenzproblems über Σ eine endliche Teilmenge $I \subseteq (\Sigma^+)^2$. Eine Lösung für eine solche Instanz ist eine endliche Folge $(u_1, v_1), \cdots, (u_n, v_n)$ von Paaren in I mit $n \geq 1$,so dass

$$u_1 \cdots u_n = v_1 \cdots v_n$$

Gibt es eine Lösung für eine instanz des Postschen Korrespondenzproblems, so heißt diese Instanz lösbar. Das **Postsche Korrespondenzproblem** über einem Alphabet Σ, kurz PCP_Σ ist die Menge aller lösbaren Instanzen des Postschen Korrespondenzproblems über Σ.

Für ein Alphabet Σ sei eine Instanz des modifizierten Postschen Korrespondenzproblems über Σ ein Paar (p,I), wobei $I \subseteq (\Sigma^+)^2$ eine endliche Teilmenge und $p \in I$ ein Paar von Wörtern ist. Eine Lösung für eine solche Instanz ist eine endlcihe Folge $(u_1,v_1),\cdots,(u_n,v_n)$ von Paaren ist I, so dass

$$p = (u_1,v_1) und u_1 \cdots u_n = v_1 \cdots v_n$$

Gibt es eine Lösung für eine Instanz des modifizierten Postschen Korrespondenzproblems so heißt diese Instanz lösbar. Das **modifizierte Postsche Korrespondenzproblem** über einem Alphabet Σ, kurz $MPCP_\Sigma$ ist die Menge aller lösbaren Instanzen des modifizierten Postschen Korrespondenzproblems über Σ.

Plan:

Für Alphabet mit $|\Sigma| \geq 2$:

$$H_{init} \overset{(3)}{\leq_m} MPCP_\Gamma \overset{(2)}{\leq_m} PCP_\Gamma \overset{(1)}{\leq_m} PCP_\Sigma$$

3.14 Lemma

Für ein Alphabet Σ und Γ mit $|\Sigma| \geq w$ gilt $PCP_\Gamma \leq_m PCP_\Sigma$

Beweis. Wir suchen eine effektive Transformation, die jede Instanz I des Postschen Korrespondenzproblems über Γ in eine Instanz I' des postschen Korrespondenzproblems über Σ transformiert, so dass I genau dann lösbar ist, wenn I' lösbar ist. Seien $a_1,a_2 \in \Sigma$ verschieden und sein $b_1,\cdots,b_{|\Gamma|}$ die Elemente von Γ. Es bezeichne $\varphi : \Gamma^* \to \Gamma^*$ den eindeutigen Homomorphismus von Sprachen mit $\varphi(b_i) = a_1^i a_2$ $\forall i \in [|\Gamma|]$. Gegeben eine solche Instanz I wie oben sei $I' := \{(\varphi(u),\varphi(v)) : (u,v) \in I\}$. Die Funktion, die geeignete Codes von Instanzen I auf geeignete Codes von Instanzen I' abbildet ist berechenbar. Ist $(u_1,v_1),\cdots,(u_n,v_n)$ eine lösung I, so gilt

$$\varphi(u_1)\cdots\varphi(v_1) = \varphi(u_1,\cdots,\varphi(v_n)) = \varphi(v_1,\cdots,v_n) = \varphi(v_1)\cdots\varphi(v_n)$$

und somit ist $(\varphi(u_1),\varphi(v_1),\cdots,(\varphi(u_n)),\varphi(v_n))$ eine Lösung von I'. Die Instanz I' ist also lösbar wenn I lösbar ist. Ist $(u_1',v_1'),\cdots,(u_n',v_n')$ eine Lösung von I', so gibt es eine Folge $(u_1,v_1)\cdots(u_n,v_n)$ von Paaren in I mit $\varphi(u_i')$ und $\varphi(v_i) = v_i'$ $\forall i \in [n]$, also mit

$$\varphi(u_1,\cdots,u_n) = u_1',\cdots,u_n' = u_1',\cdots,u_n' = \varphi(u_1,\cdots,u_n)$$

Da $\varphi|_\Sigma$ injektiv und $\varphi(\Sigma)$ präfixfrei ist, ist φ injektiv (siehe Übung), folglich gilt $u_1,\cdots,u_n = v_1,\cdots,v_n$ und somit ist $(u_1,v_1),\cdots,(u_n,v_n)$ eine Lösung von I. Die Instanz I ist also lösbar wenn I' Lösbar ist. □

3.15 Lemma

Für Jedes alphabet Σ mit $|\Sigma| \leq w$ gitl $MPCP_\Sigma \leq_m PCP_\Sigma$.

Beweis. Sei Σ ein Alphabet mit $|\Sigma| \geq 2$. Nach lemma 3.14 genügt es ein Alphabet Γ zu finden, so das $MPCP_\Sigma \leq_m PCP_\Gamma$ gilt.
Wir suchen eine effektive Transformation , die jede instanz (p,I) des modifizierten Postschen Korrespondenzproblems über Σ in eine Instanz I' des Postschen Korrespondenzproblems über einem geeignetem AlphabetΓ transformiert, so dass (p,I) genau dann lösbar ist, wenn I' lösbar ist. □

Idee:

0	1	0	0	1	0	1	1	1	0	1
0	1	0	0	1	0	1	1	1	0	1

... Betrachte die Homomorphismus von Sprachen $\delta_{\rightarrow}$, $\delta_{\leftarrow} : \Sigma^* \rightarrow (\Sigma \cup *)^*$ mit $\delta_a = a*$ und $\delta_{\leftarrow}(a) = *a$ $\forall a \in \Sigma$. Für jede Instanz $(p,I) = ((u_1,v_1),I)$ wie oben sei

$$I' = \{(\delta_{\leftarrow}(u_1), *\delta_{\rightarrow}(v_1))\} \cup \{\delta_{leftarrow}(u), \delta_{rightarrow}(v) : (u,v) \in I\} \cup \{\delta_{\leftarrow}(u)*, \delta_{\rightarrow}(v) : (u,v) \in I\}$$

Die Funktion die geeignete Codes von Instanzen (p,I) auf geeignete Codes der zugehörigen Instanzen I' abbildet ist berechenbar. Gibt es eine Lösung $(u_1,v_1),\cdots,(u_n,u_n)$ von (p,I) dann ist

$$\delta_{\leftarrow}(u_1)\cdots\delta_{\leftarrow}(u_n)* = \delta_{\leftarrow}(u_1\cdots u_n)*$$

$$= \delta_{\leftarrow}(v_1\cdots v_n)*$$

$$= *\delta_{\rightarrow}(v_1\cdots v_n)$$

$$= *\delta_{\rightarrow}(v_1)\cdots\delta_{\rightarrow}(v_n)$$

und folglich ist

$$(\delta_{\leftarrow}(u_1), *\delta_{\rightarrow}(v_1)), (\delta_{\leftarrow}(u_2), \delta_{\rightarrow}(v_2)), \cdots, (\delta_{\leftarrow}(u_{n-1}), \delta_{\rightarrow}(v_{n-1})), (\delta_{\leftarrow}(u_n), \delta_{\rightarrow}(v_n))$$

eine Lösung von I'. Es bleibt zu zeigen das (p,I) lösbar ist, wenn I' lösbar ist. Sei $\tau : (\Sigma \cup \{*\})^* \rightarrow \Sigma^*$ der Homomorphismus von Sprachen mit $\tau|_{\Sigma} = id_{\Sigma}$ und $\tau(*) = \lambda$. Für $(u',v') \in I'$ gilt $(\tau(u'),\tau(v')) \in I$. Sei $(u'_1,v'_1),\cdots,(u'_n,v'_n)$ eine Lösung von I' und $(u'_i,v'_i) = (\tau(u'_i),\tau(v'_i))$ für $i \in [n]$. Es gilt

$$\tau(u'_1)\cdots\tau(u'_n) = \tau(u'_1\cdots u'_n) = \tau(v'_1\cdots v'_n) = \tau(v'_1)\cdots\tau(v'_n)$$

und somit ist $(u_1,v_1),\cdots,(u_n,u_v)$ eine Lösung von I als Instanz des Postschen Korrespondenzproblems über Σ. Es genügt aber zu zeigen, dass $(u_1,v_1) = p$ gilt. Sei $p' = (\delta_{\leftarrow}(u_1), \not\tau(?wirklichnichttau?)\delta_{\rightarrow}(v_1))$. Für $(u',v') \in I'/\{p'\}$ gilt $u'(1) \neq v'(1)$, da $(u'_1,v'_1),\cdots,(u'_n,v'_n)$ eine Lösung von I' ist gilt also $(u'_1,v'_1) = p'$ und damit $(u_1,v_1) = (\tau(u'_1),\tau(v'_1)) = p$.

3.16 Lemma

Für jedes Alphabet Σ mit $|\Sigma| \geq 2$ gilt $H_{init} \leq_m MPCP_{\Box,0,1,*,6,+}$

Beweis. Wir suchen eine effektive Transformation, die jede natürliche Zahl e auf eine Instanz (p_e,I_e) des modifizierten Portschen Korrespondenzproblems über $\{\Box,0,1,*,,+\}$ abbildet, so dass $\mathscr{M}_e(\lambda)\downarrow$ genau dann gilt, wenn (p_e,I_e) lösbar ist. Sei $e \in \mathbb{N}_0$. Sei Q Die Zustandsmenge und Δ die Übergangsrelation von $\mathscr{M}_e$. Es gelte also $\mathscr{M}_e = (Q,\Sigma,\Gamma,\Delta,s,F)$ für $\Sigma = \{0,1\}$, $\Gamma = \{\Box,0,1\}$, $S = 0$, $F = \{0\}$
Für eine Instanz (p,I) des modifizierten Postschen Korrespondenzproblems über einem Alphabet bezeichnen wir eine Folge $p = (u_1,v_1),\cdots,(u_n,v_n)$ für die $u_1\cdots u_n \sqsubseteq v_1\cdots v_n$ oder $v_1\cdots v_n \sqsubset u_1\cdots u_n$ gilt als **partielle Lösung** von (p,I). Wir wollen (p_e,I_e) so wählen, dass partielle Lösungen von (p_e,I_e) partielle Rechungen von $\mathscr{M}_e$ entsprechen. Dabei codieren wie eine Konfiguration $(p,w,p) \in Q \times (\Gamma^*) * \mathbb{N}_0$ von $\mathscr{M}_e$ durch das Wort

$$code(q,w,p) := \#w(1)\cdots w(p-q) * bin(q) * w(p)\cdots w(|w|)\#$$

Im wesentlichen wollen wir erreichen, dass es genau dann für ein Wort w eine partielle lösung $(u_1,v_1),\cdots,(u_n,v_n)$ von (p_e,I_e) mit $w = v_1\cdots v_n$ gibt, wenn w Präfix der Konkation $code(C_1)\cdots code(C_n)$

der Code der Konfiguration einer partiellen Rechnung $C_1, \cdots, C_n$ von $\mathscr{M}_e$ bei Eingabe λ ist. Eine solche partielle Lösung soll genau dann zu einer Lösung von (p_e, I_e) vervollständigt werden können, wenn die durch w beschriebene partielle Rechung mit einer Stoppkonfiguraion endet, alsp eine Rechung ist. Dann ist (p_e, I_e) genau dann lösbar, wenn die Rechung von $\mathscr{M}_e$ zur Eingabe λ endlich ist.

Für $q \in Q$ sei $\hat{q} : *bin(q)$
Als Startpaar sehen wir

$$p_e = (0, 0\# * \; *\Box\#)$$

(die 0en sind nur dafür da da, damit ïm?"komplment nicht leer ist.) Wir beschreiben nun die Konstruktion von I_e. Für jede Instruktion $(q, a, q', a', L) \in \Delta$ fügen wir folgende Paare ein

$$(\#\hat{q}a, \#\hat{q}'\Box a'), (\Box\hat{q}a, \hat{q}'\Box a'), (0\hat{q}a, \hat{q}'0a')(1\hat{q}a, \hat{q}'1a')$$

ein. Weiter, um unveränderte Infixe kopieren zu können fügen wir die Paare

$$(\#, \#), (0, 0), (1, 1), (\Box, \Box)$$

ein. Nun brauchen wir noch Paare, die bei Terminierung der TM zu einer validen Instanz der *MPCP* -Instanz führen.
$\rightsquigarrow \forall q \in Q \forall a \in \{\Box, 0, 1\}$ für die es keine Instruktion (q, a, q', a', B) fürgen wir das Paar $(\hat{q}a, \dagger a)$ hinzu und auch

$$(\dagger\Box, \dagger), (\dagger 0, \dagger), (\dagger 1, \dagger)$$

$$(\Box\dagger, \dagger), (0\dagger, \dagger), (1\dagger, \dagger)$$

$$(\# \dagger \#0, 0)$$

Dies beschreibt die Konstruktion von (p_e, I_e). Wir verzichten auf die einfache aber aufwändige Verfifikation, dass $\mathscr{M}_e$ genau dann bei Eingabe λ terminiert, wenn (p_e, I_e) lösbar ist. □

3.17 Beispiel

Sei $e \in \mathbb{N}_0$ mit $\mathscr{M}_e = (\{0,1\}, \{0,1\}, \{\Box, 0, 1\}, \Delta, 0, \{0\})$, wobei $\Delta = \{(0, \Box, 1, 1, R), (1, \Box, 1, 1, L)\}$. Mitder Notation aus dem Beweis aus Lemma 3.17 gilt dann [hier bild einfügen!]

3.18 Satz

Für jedes Alphabet Σ mit $|\Sigma| \geq 2$ ist PCP_Σ nicht entscheidbar.

Beweis. Mit Lemma 3.15, 3.16 und 3.17 folgt

$$H_{init} \leq_m MPCP_{\Box,0,1,*,\#,\dagger} \leq_m PCP_{\Box,0,1,*,\#,\dagger} \leq_m PCP_\Sigma$$

und damit $H_{init} \leq_m PCP_\Sigma$. Folglich ist PCP_Σ nicht entscheidbar, da H_{init} nicht entscheidbar ist. □

asd

3.19 Fixpunktsatz, Rekusionstheorem und Satz von Rice

Wir beschäftigen uns nun mit weiteren Konsequenzen der Standardaufzählung von TM.

$$\Phi_0, \Phi_1, \Phi_2, \cdots$$

Standardaufzählung

$$\Phi_{\Phi_e(0)}, \Phi_{\Phi_e(1)}, \Phi_{\Phi_e(2)}, \cdots$$

andere Aufzählung $\Rightarrow$

$$\Phi_{f(0)}, \Phi_{f(1)}, \Phi_{f(2)}$$

3.20 Definition (Fixpunktsatz)

Ein **Fixpunkt** eine berechenbaren Funktion $f : \mathbb{N}_0 \to \mathbb{N}_0$ ist ein $e \in \mathbb{N}_0$ mit $\Phi_{f(e)} = \Phi_e$.

3.21 Satz (Fixpunktsatz, Hartley Rogers jr., 1967)

Alle berechenbaren Funktionen $f : \mathbb{N}_0 \to \mathbb{N}_0$ haben einen Fixpunkt.

Beweis. $\forall e, x \in \mathbb{N}_0$ mit $\Phi_e(x) \uparrow$ sei $\Phi_{\Phi_e(x)} : \mathbb{N}_0 \rightsquigarrow \mathbb{N}_0$ die aprtiell berechenabre partielle Funktion mit $dom(\Phi_{\Phi_e(x)}) = \varnothing$. Sei e_ψ ein Index von ψ. Gemäß S_n^m-Theorem (Satz 3.7) existiert eine berechenbare Funktion $s_1^1 : \mathbb{N}_0^2 \to \mathbb{N}_0$ mit $\Phi_{s_1^1(e_\psi, e)}(x) = \psi(e, x)$. $\forall e, x \in \mathbb{N}_0$.Sei $\eta : \mathbb{N}_0 \to \mathbb{N}_0$ die berechenbare Funktion mit $\eta(e) := s_1^1(e_\psi, e)$. Dann gilt

$$\psi_{\eta(e)}(x) = \psi_{s_1^1(e_\psi, e)}(x) = \psi(e, x) = \Phi_{\Phi_e(e)}(x) \forall x \in \mathbb{N}_0$$

also gilt

$$\Phi_{\eta(e)} = \Phi_{\Phi_e(e)} (*)$$

Sei $e_{f \circ h}$ ein Index der berechneten Funktion $f \circ h$ und $e_{fix} := \eta(e_{f \circ h})$.

$$\Phi_{f(e_{fix})} =$$

$\square$

Automaten und Grammatiken

Imagine you're in a city with a limited number of locations (like a park, library, cafe, etc.). You can move from one place to another following specific paths (like roads). The paths you take depend on some rules, like the time of the day, or the type of ticket you have. The places you can reach with these rules represent different states in a finite automaton, and the rules themselves act like the transition function.

- ChatGPT

4 Endliche Automaten

Wir wollen Turingmahinen un stark einschränken. wir betrahten ein Modell, das im wesentlichen ohne speicher zurechtkommt (=Tm ohne band $\longrightarrow$ brauchen es nur für die Eingabe). Der Ausgabemechanismus kennt nur Akzeptanz und Nichtakzeptanz.

Als TM kann der wie folgt realisiert werden:

- Es ist nur ein Band erlaubt.
- Bei jedem Rechenschritt bewegt sich der Kopf nach rechts. Ob und wie die Felder des Bandes dabei überschreiben werden spielt dann keine Rolle, denn der Kopf kann nie zurück bewegt werden; wir lehen aber fest, dass Symbole nicht überschrieben werden. Die Symbole die des Bandalphabet Γ neben denen des Eingabealphabets Σ und des □ Symbols ?? hat ?? spielen keine Rolle. Wir legen hier Γ = Σ∪ {□} fest.
- Beim Einlesen des ersten □ Symbols muss die Rechnung der Machine enden. Wir soll die Rechnung nicht vor dem Einlesen des ersten □ Symbols enden.

??Die?? bedeutet, dass wir DM M = (Q, Σm Σ∪{□}), δ, s, F) die nur Instruktionen der Form (q, a, q', a, R) mit q ∈ Q und a ∈ σ hat. Dies sind nun stark eingeschränkte TM. Wir wählen eine äquivalente Form, die als endliche Automaten bezeichnet werden.

4.1 Definition (Endliche Automaten)

Ein endicher Automat, kurz EA, ist ein Tupel A = (Q, Σ, δ, s, F). Dabei ist

- Q eine endliche Menge, der Zustandsmenge;
- Σ das Eingabealphabet;
- Δ ⊆ Q x σ x Q die Übergangsrelation, eine relation, so dass es für alle q ∈ Q und a∈ σ ein q' $/in$ Q mit (q, a, q');
- s ∈ Q der Startzustand;
- F ⊆ die Menge der akzeptierten Zustände.

er endliche Automat A ist ein deterministischer endlicher Automa, kurz DEA, wenn es ∀ (q,a) ∈ Q x σ genau ein q' gibt mit (q,a,q') ∈ δ. Im Sinne der obigen Betrachtung entspricht ein EA A = (Q, Σm Δ, s, F) der 1-TM M_a = (Q, σ, ...) ⇝ Band spielt keine wesentliche Rolle, Zustände mir gerade gelesenen Symbol bilden die Konfiurationen.

4.2 Definition (Übergangsfunktion eines EA)

Sei A = (Q, σ, Δ,s,F) ein EA. Die Übergangsfunktion von A ist die Funktion
S_A : Q x Σ → 2^Q mit S_A(q, a) = {q' ∈ Q: (q, a, q')∈ δ} ∀q ∈ Q a ∈ σ.
Die erweiterte Übergangsfunktion von A ist die Funktion
δ_A^* (q,a): Q x Σ^* → 2^Q mit S_A^*(q,λ) = {q} und $\delta_A^*(q,aw) \bigcup\limits_{q'\in\delta_A^*(q,a)} \delta_A^*$(q', w) ∀q ∈ Q a∈ Σ und w∈ Σ^*.
Für Q_0 ⊆ Q und w ∈ Σ^* schreiben wir $\delta_A^*(Q_0,w)$ statt $\bigcup\limits_{q\in Q_0} \delta_A^*$ (q, w).

4.3 Definition (Übergangsfunktion eines EA)

Sei A = (Q, Σ, Δ, s, F) ein EA. Die **Übergangsfunktion** von A ist die Funktion $\delta_A : Q \times \Sigma \to 2^Q$ (=Potenzmenge von Q) mit δ_A(q,a) = {q'∈ Q : (q, a, q')∈ Δ} ∀q ∈ Q, a∈ Σ **erweiterte Übergangsfunktion** von A ist die Funktion $\delta_A^* : Q \times \Sigma^* \to 2^Q$ δ_A^*(q, λ) = {q} und δ_A^*(q, aw) = $\bigcup_{q' \in \delta_A(q,a)} \delta_A^*(q', w)$ ∀q∈Q a∈ Σ und w∈ Σ*. Für $Q_0 \subseteq Q$ und w ∈ Σ* schreiben wir δ_A^* (Q_0, w) statt $\bigcup_{q \in Q_0} \delta_A^*(q,w)$. Für einen EA A = (Q, Σ, Δ, s, F) ,mit entsprechnder TM M_A = (Q, Σ, Γ, Δ', s, F), q ∈ Q und w ∈ Σ* ist δ_A^*(s,w) die Menge der zustände, die sich als erst Komp.?? der ubtem?? Konfig einer Rechnung von M_A zur Eingabe zu ergeben.

4.4 Bemerkung

Sei A = (Q, Σ, Δ, s, F) ein EA

(i) ∀ q ∈ Q und a∈ Σ gilt δ_A^*(q,a) = δ_A(q, a).

(ii) Ist A ein DEA, q ∈ Q, a ∈ Σ und w ∈ Σ*, und $|\delta_A^*(q,w)| = 1$??.

(iii) Seien u,v ∈ Σ* ∀ q ∈ Q gilt δ_A^*(q, uv) = $\delta_A^*(\delta_A^*(Q_0,u),v)$.

4.5 Definition (Übergangsfunktion eines DEA)

Sei A = (Q, Σ, Δ, s, F) eine DEA. Auch die Funktion $\delta_{det,A}$: Q ×Σ → Q mit $\delta_A(q,a) = \{\delta_{det,A}(q,a)\}$ ∀ q ∈ Q und a ∈ Σ wird auch **Übergangsfunktion** von A gennant. Analoges gilt für $\delta_{det,A}^*$(Q_0, w) statt $\bigcup_{q \in Q_0} \{\delta_{det,A}^*(q,w)\}$.

4.6 Bemerkung

Ist A = (Q, Σ, Δ, s, F) ein DEA, so gelten Berkung 4.3 (i) und (iii) auch wenn δ_A durch $\delta_{det,A}$ und δ_A^* durch $\delta_{det,A}^*$ ersetzt wird.

4.7 Bemerkung

Sei Q eine endliche Menge, Σ ein Alphabet, s∈ Q, und F⊆ Q.

(i) ∀ Funktionen $\delta : Q \times \Sigma \to 2^Q$ gibt es genau einen EA A = (Q, Σ, Δ, s, F) mit $\delta_A = \delta$.

(ii) ∀ Funktionen $\delta : Q \times \Sigma \to Q$ gibt es genau einen $\delta_{det,A} = \delta$.

4.8 Definition (akzeptierte Sprache)

Sei A = (Q, Σ, Δ, s, F) ein EA. Die Sprache L(A) := {w ∈ Σ* : δ_A^*(s, w)∩ F ≠ ∅} ist die **akzeptierte Sprache** von A.

4.9 Definition (regulär)

Eine Sprache L heißt **regulär** wenn es einen EA A mit L(A) = L gibt. Wir schreiben REG für die Klasse der regulären Sprachen. Zu jedem Zeitpunkt während der Verbindung der Eingabe durch einen endlichen Automaten höngt der restliche Bearbeitung immer nur vom gegewärtigen Zustand und dem noch einzulesenden Teil der Eingabe ab, nicht aber wie bei TM im allgemeinen von vergangenen Bandmanipulation. Interpretiert man die Eingabe als von einer äußeren Quelle kommend, so ist der Zustand des Automaten also allein durch seinen Zustand gegeben und der nächste Zustand hängt nur vom Zugeführten Symbol ab. Daher bietet sich eine Darstellung eines EA durch ein Übergangsdiagramm oder eine sogenannte Übergangstabelle an.

4.10 Beispiel

Sei A := ({q_0, q_1}, {0, 1}, Δ, q_0, {q_1}) mit Δ = {(q_0)} Übergangsdiagramm und übergangstabelle von sehen wie folgt aus:

Zustand/Symbol	0	1
q_0	q_0	q_1
q_1, *	q_1	q_0

[Hier muss noch ein Übergangsdiagramm hin!]

Übergangsdiagramm:

Für jeden Zustand gibt es einen Kreis. Zustände in F bekommen einen Doppelkreis. Für (q, a, q') ∈ Δ für einen Pfeil von dem Kreis von q zu dem Kreis von q' mit der Beschreibung a. Zusätzlich gibt es einen Pfeil (ohne Beschriftung) aus dem "Nichtsßus deom Kreis des Starzustandes.

Ähnlich wie bei allgemeinen und normierten TM bleibt die Klasse der akzeptierten Sprachen glich wenn man nur deterministisch endliche Automaten zulässt. Um dies zu beweisen führen wir den Potentautomaten ein.

4.11 Definition (Potenzautomaten)

Sei A = (Q, Σ, Δ, s, F) ein EA. der Potenzautomat von A ist der DEA P_A = (2^Q, Σ, Δ', {s}, {P⊆Q : P∪F ≠ ∅ }) mit $\delta_{det}, P_A(Q_0, a) = \bigcup \ldots$

4.12 Satz

Eine Sprache L ist genau dann regulär, wenn es eine DEA A mit L(A) = L gibt.

Beweis:

Sei A = (Q, Σ, Δ, s, F) ein EA mit Potenzautomat P_A. Es genügt zu zeigen, dass L(A) = L(P_A). Hierfür genügt es zu zeigen, dass:

$$\delta^*_{det,P} = \delta^*_A(s,w) \forall w \in \Sigma^*(*)$$

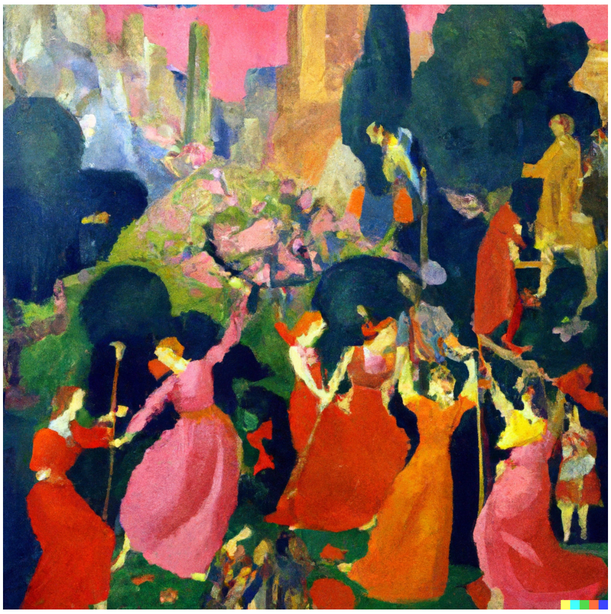
Denn damit folgt

$$\begin{aligned} w \in L(P_A) &\Leftrightarrow \delta^*_{P_A}(\{s\},w) \cap \{P \subseteq Q : P \cap F \neq \varnothing\} \neq \varnothing \\ &\Leftrightarrow \delta^*_{det,P_A}(\{s\},w) \cap F \neq \varnothing \\ &\underset{(*)}{\Leftrightarrow} \delta^*_A(\{s\},w) \cap F \neq \varnothing \\ &\Leftrightarrow w \in L(A) \end{aligned}$$

Wir zeigen (*) mittels vollständiger Induktion über |w|. Es gilt δ^*_{det,P_A}({s}, λ) = δ^*_A(s, λ). Sei w ∈ Σ^+ mit $\delta^*_{det,A}$({s}, v) = δ^*_A(s, v) ∀ v ∈ $\Sigma^{\leq |w|-1}$. Nun zeigen wir (*) Sei va := w mit a ∈ Σ und |v| = |w| - 1.

$$\begin{aligned} \delta^*_{det,P_A}(\{s\},w) &\underset{\text{Bem 4.5}}{=} \delta^*_{det,P_A}(\delta^*_{det,P_A}(\{s\},v),a) \\ &\underset{\text{Ind. hyp}}{=} \delta^*_{det,P_A}(\delta^*_{det,P_A}(\{s\},v),a) \\ &= \bigcup_{q \in \delta^*_{det,A}} \delta_A(q,a) \\ &= \delta^*_A(\delta^*_A(s,v),a) \end{aligned}$$

$$= \delta_A^*(s, va)$$
$$= \delta_A^*(s, w)$$



Reguläre Sprachen

A regular language can be thought of as a collection of sentences in a secret code. This secret code has a set of rules that determine which sentences are valid. You can think of it like a secret handshake, where only certain movements are allowed to be performed in a particular order.
- ChatGPT

5 Reguläre Sprachen

5.1 Definition (Äquivalenzrelation)

Sei A eine Menge. Eine Äquivalenzrelation auf A ist eine Relation $\leq A^2$, so dass die folgende Eigenschaft erfüllt sind. (wie bei Relationen üblich verwenden wir Infixnotation)

(i) $a \sim a \forall a \in A$ (Reflexivität)

(ii) $a \sim b \Rightarrow b \sim a \forall a, b, c \in A$ (Symetrie)

(iii) $a \sim b, b \sim c \rightarrow a \sim c$ (Transitivität)

Die **Äquivalenzklasse** eines Elements $a \in A$ bezüglich $\sim$ ist die Menge $[a] \ := a' \in A : a' \ a$. Der **Index** von $\sim$ ist die Kardinalität der Menge $A_{/\sim} := [a]_{\sim} : a \in A$ falls diese endlich ist und ∞ andernfalls.

5.2 Definition (A-Äquivalenz)

Sei $A = (Q, \Sigma, \Delta, s, F)$ ein DEA mit erweiterter Übergangsfunktion $\delta^* : Q \times \Sigma \rightarrow Q$. Die A-Äquivalenz ist die Relation $_A$ auf $\Sigma^* \cdots$

5.3 Bemerkung

Sei $A = (Q, \Sigma, \Delta, s, F)$ eine DEA.

(i) Die A-Äquivalenz ist eine Äquivalenzrelation.

(ii) Der Index von $\sim_A$ ist höchstens $|Q|$.

(iii) Es gilt $L(A) = \bigcup\limits_{w \in L(A)} [w]_{\sim A}$.

5.4 Definition (Rechtskongruenz)

Sei Σ ein Alpha. Eine Rechtskongruenz auf Σ^* ist eine Äquivalenzrelation $\sim ? \leq ?(\Sigma^*)^2$ mit $u \sim v \Rightarrow uw \sim vw \forall u, v, w \in Sigma^*$.

Proposition 2. *Sei $A = (Q, \Sigma, \Delta, s, F)$ ein DEA. Die A-Äquivalenz $\sim_A$ ist eine Rechtskonqruenz auf Σ^*.*

Beweis. Seien $u, v, w \in \Sigma^*$ mit $u \sim_A v$. Dann gilt

$$\begin{aligned} \delta^*_{det,A}(s, uw) = \delta^*_{det,A}(\delta^*_{det,A}(s, u), w) &= \delta^*_{det,A}(\delta^*_{det,A}(s, v), w) \\ &= \delta^*_{det,A}(s, vw). \end{aligned}$$

(hier benutzen wir Bem 4.3 und 4.5) □

Dann gilt $uw \sim_A vw$. □ Zu jedem DEA A gibt es also eine dazugehärige Rechtskonguenz $\sim$ auf Σ^* mit endlichem Index so dass L(A) die Vereinigung von Äquivalenzklasse von $\sim_A$ ist. Tatsächlich gilt auch die Umkehrung: Ist L die Vereinigung von Äquivalenzklasse einer Rechtskongruenz $\sim$ mit endlichem Index, so gibt es einen DEA A mit $L(A) = L$

5.5 Definition

Sei Σ eine Alphabet und L Vereinigung von Äquivalenzklasse einer Rechtskongruenz $\sim$ auf Σ^* mit endlichem Index. Es bezeichne

$$A_{\sim,L} := (\Sigma^*_{/\sim}, \Sigma, \Delta, [\lambda]_\sim, [w]_\sim : w \in L$$

den DEA mit $\delta_{det,A_{\sim},L}([w]_\sim, a) = [wa]_\sim \forall w \in \Sigma^*$ und $a \in \Sigma$. Die Wohldefiniertheit von $\delta_{det,A_{\sim},L}$ ergibt sich daraus, dass $\sim$ eine Rechtskongruenz ist. Um uns davon zu überzeugen, dass $L(A_{\sim,L}) = L$ gilt betrachten wr zunächst die Arbeitsweise von $A_{\sim,L}$.

5.6 Lemma

Sei Σ ein Alphabet, L Vereinigung von Äquivalenzklassem einer Rechtskongruenz $\sim$ auf Σ^* mit endlichem Index und sei $\delta^* : \Sigma^*_{/\sim} \times \Sigma^* \to \Sigma^*_{/\sim}$ die erweiterte Übergangsfunktion von $A_{\sim,L}$. Dann gilt $\delta^*([\lambda]_\sim, w) = [w]_\sim \forall w \in \Sigma^*$.

Beweis. Wir verwenden vollständige Induktion über $|w|$. Es gilt $\delta^*([\lambda]_\sim, \lambda) = [\lambda_\sim]$. Sei nun $w \in \Sigma^+$ … □

5.7 Satz

Sei L die vereinigung von Äquivalenzklasse einer Rechtskongruenz $\sim$ mit endlichem Index Es gibt $L(A_{\sim,L}) = L$

Beweis. Sei Σ das Alphabet, so dass $\sim$ eine Rechtskongruenz auf Σ^* ist. Sei $\delta^* : \Sigma^*_{/\sim} \times \Sigma^* \to \Sigma^*_{/\sim}$ die erweiterte Übergangsfunktion von $A_{\sim,L}$ und sei $w \in \Sigma^*$. Aus Lemma 5.7 folgt

$$w \in L(A_{\sim,L}) \Leftrightarrow \delta^*([\lambda]_\sim, w) \in [v]_\sim : v \in L$$

$$\Leftrightarrow [w]_\sim \in [v]_\sim : v \in L$$

$$\Leftrightarrow \exists v \in L : [w]_\sim = [v]_\sim$$

$$\Leftrightarrow \exists v \in L : w \sim v$$

$$\Leftrightarrow w \in L$$

… □

5.8 Korollar

Eine Sprache L ist genau dann regulär, wenn sie die Verienigung von Äquivalenzklasse einer Rechtskongruenz mit endlichem Index ist.

Beweis. Folgt aus Bemerkung 5.3, Proposition 5.5 und Satz 5.8 □ □

Betrachten man nur deterministische endliche Automaten ohne unerreichbare Zustände, so entsprechen diese bis auf Unbenutzung von Zuständen sogar den Rechtskongruenz mit endlichem Index zusammen mit Vereinigung von Äquivalenzklassn dieser.

5.9 Definition(erreichbar)

Sei Σ ein Alphabet. Sei $A = (Q, \Sigma, \Delta, s, F)$ ein EA mit erweiterter Übergangsfunktion δ^*. Ein zustand $q \in Q$ heißt erreichbar in A wenn es ein Wort $w \in \Sigma^*$ mit $q \in \delta^*(s, w)$ gilt.

5.10 Definition(isomorph)

Sei $A_i = (Q_i, \Sigma, \Delta_i, s_i, F_i)$ für $i \in 1,2$ ein EA mit Übergangsfunktion δ_i. Die endliche Automaten A_1 und A_2 sind **isomorph**, kurz $A_1 ? \cong A_2$, wenn es eine Projektion $f : Q_1 \to Q_2$ gibt, sodass folgendes gilt:

(i) $f(s_1) = s_2$

(ii) $\delta_2(f(q_1), a) = f(\delta_1(q_1), a)$

(iii) $f(F_1) = F_2$

5.11 Satz

(i) Ist A eine DEA ohne unereichbare Zustände, so gilt $A \cong A_{\sim A, L(A)}$

(ii) Ist L die Vereinuíngung von Äquivalenzklasse einer Rechtskongruenz $\sim$ mit endlichem Index, so gilt $(\sim, L) = (\sim_{A_{\sim,L}, L(A_{\sim,L})})$.

Beweis. (i) Sei $A = (Q, \Sigma, \Delta, s, F)$ eine DEA mit erweiterte Übergangsfunktion $\delta^* : Q \times \Sigma^* \to Q$ ohne unereichbare Zustände , $\sim := \sim_A$, $A' := A_{\sim, L(A)}$ und sei $\delta' : \Sigma^* / \sim \times \Sigma^* \to \Sigma / \sim$ die erweiterte Übergangsfunktion von A'. Sei $f : Q \to \Sigma^* / \sim$ die Bijektive mit $f(q) := \{w \in \Sigma^* : \delta^*(s, w) = q\}$. Es gelte $f(s) = [\lambda]_\sim$ und $f(F) = \{[w]_\sim : w \in L(A)\}$. Es genügt somit zu zeigen , dass $\delta'(f(q), a) = f(\delta(q, a)) \forall q \in Q, a \in \Sigma$. Sei $q \in Q, a \in \Sigma^*$. Es genügt $w \in \delta'(f(q), a) \Leftrightarrow \delta^*(s, w) = \delta^*(q, a)$ zu zeigen. Sei $v \in \Sigma^*$ mit $\delta^*(s, v) = q$. Nun gilt $w \in \delta'(f(q), a) \leftrightarrow w \in \delta'([v]_\sim, a) \leftrightarrow w \sim va \leftrightarrow \delta^*(s, w) = \delta^*(s, va) \leftrightarrow \delta^*(s, w) = \delta^*(q, a)$

(ii) Sei Σ ein Alphabet, $\sim$ eine Rechtskongruenz aud Σ^*, L Vereinigung von Äquivalenzklassen von $\sim$, $A' := A_{\sim, L} = (\Sigma^* / \sim, \Sigma, A', [\lambda]_\sim, \uparrow)$, $\delta'^* : \Sigma^* / \sim \times \Sigma^* \to \Sigma^* / \sim$ die erweiterterte Übergangsfunktion von A' $\{w \in \Sigma^* : w \in L\}$ und $\sim' := \sim_{A'}$. Nach Satz 5.8 gilt $L = L(A')$, es genügt also $\sim = \sim'$ zu zeigen. Sei $u, v \in \Sigma^*$. Aus lemma 5.7 folgt $u \sim v \leftrightarrow [u]_\sim = [v]_\sim \leftrightarrow \delta'(...)\ldots$

□

5.12 Satz

Bedeutet insbesondere folgendes: Ist A_i, $i \in \{1,2\}$ ein DEA ohne unereichbare Zustände, so gilt $A_1 \cong A_2 \leftrightarrow (\sim_{A_1}, L(A_1)) = (\sim_{A_2}, L(A_2))$ und ist L_i für $i \in \{1,2\}$. Vereinigung von Äquivalenzklassen einer Rechtskongruenz $\sim_i$ mit endlichem Index, so gilt $(\sim_1, L_1) = (\sim_2, L_2) \leftrightarrow A_{\sim_1, L_1} \cong A_{\sim_2, L_2}$.

Ist L eine reguläre Sprache, so gibt es verschiedene endliche Automaten (ohne unereichbare Zustände) mit $L(A) = L$. Äquivalenzklassen verschiedener Rechtskongruenz mit endlichem Index. Für alle solche Rechtskongruenz $\sim$ und $\forall u, v, w \Sigma^*$ mit $u \sim v$ gilt aber

$$uw \in L \leftrightarrow \delta^*_{det,A}(s, uw) \in F \leftrightarrow \delta^*_{det,A}(s, vw) \in F \leftrightarrow vw \in L$$

Dies führt zum Begriff der L-Äquivalenz und zeigt, dass die Parition in die Äquivalenzklassen von $\sim$ Vereinfacht der Parition in die Äquivalenzklasse der L-Äquivalenz ist.

5.13 Definition (L-Äquivalenz)

Sei L eine Sprache über einem Alphabet Σ. Die **L-Äquivalenz** von L als Sprache ist die Relation $\sim_L$ auf Σ^* mit

$$u \sim_L v \leftrightarrow (uw \in L \leftrightarrow vw \in L \forall w \in \Sigma^*)$$

5.14 Bemerkung

Sei L eine Sprache über Σ.

(i) Die L-Äquivalenz ist eine Rechtskongruenz.

(ii) Es gilt $L = \bigcup_{w \in L} [w]_{\sim L}$.

5.15 Definition (Parition)

Sei A eine Menge. Eine Parition von A ist eine Menge $\mathscr{A} = A_1, \cdots, A_n$ paarweise disjunkt nichtleere Teilmengen von A mit $\bigcup_{i \in [n]} A_i = A$.

5.16 Definition (Verefeinerung)

Seien $\mathscr{A}_1$ und $\mathscr{A}_2$ Paritionen einer Menge A. Die Parition $\mathscr{A}_2$ **Verefeinert** $\mathscr{A}_1$ (heißt Verefeinerung von $\mathscr{A}_1$), wenn es $\forall A_2 \in \mathscr{A}_2$ ein $A_1 \in \mathscr{A}_1$, mit $A_2 \subseteq A_1$ gibt.

5.17 Bemerkung

Seien $\mathscr{A}_1$ und $\mathscr{A}_2$ Paritionen einer Menge A_1, so dass A_2 die Parition $\mathscr{A}_1$ verefeinert.

(i) $\forall A' \in \mathscr{A}_1$, gibt es eine Teilmengen $\mathscr{A}_2' \subseteq \mathscr{A}_2$, die eine Parition von A' ist.

-

5.18 Proposition

Sei Σ eine Alphabet und L eine Sprache über Σ und $\sim$ eine Rechtskongruenz auf Σ^* mit $L = \bigcup_{w \in L} [w]_\sim$. Die Parition $\Sigma^* / \sim$ ist eine Verefeinerung der partition $\Sigma^* / \sim L$.

Beweis. Seien $u, v \in \Sigma^*$ mit $u \sim v$. Es genügt zu zeigen, dass $u \sim_L v$. Sei $w \in \Sigma^*$. Es genügt $uw \in L \leftrightarrow vw \in L$ zu zeigen. Da $\sim$ eine Rechtskongruenz ist dolgt $uw \sim vw$. Ist $u, w \in L$, so folgt aus $L = \bigcup_{w' \in L} [w']_\sim$ auch $vw \in L$(analog folgt auch auch $vw \in L \Rightarrow uw \in L$). □

$\Rightarrow u \sim_L v$.

Das heißt $\sim_L$ ist die größte Parition, die L darstellen kann.

5.19 Definition (Minimalautomat)

Sei L eine reguläre Sprache über Σ. Der Minimalautomat von L als Sprache über Σ ist der DEA $A_{\sim L, L}$.

5.20 Satz

Sei L eine reguläre Sprache über Σ und sei $M = (Q, \Sigma, \Delta, s, F)$ der Minimalautomat von L. Dann gilt:

(i) $L(M) = L$

(ii) Ist A ein DEA mit Zustandsmenge Q_A und $L(A) = L$, so gilt $|Q_A| \leq |Q|$.

(iii) Ist A ein DEA mit $|Q|$ Zuständen und $L(A) = L$, so gilt $A \cong M$.

...

Für eine Sprache L über einem Alphabet Σ sind die folgenden Aussagen äquivalent:

(i) L ist regulär.

(ii) Der Index von $\sim_L$ ist endlich.

(iii) L ist die Vereinigung von Äquivalenzklasse einer Rechtskongruenz mit endlichem Index.

Beweis. (i) $\Leftrightarrow$ (iii) ist die Aussage von Korollar 5.9. Die Relation $\sim_L$ ist nach Bemerkung 5.14 eine Rechtskongruenzund es gilt $L = \bigcup_{w\in L} [w]_{\sim L}$. Somit folgt folgt (i) $\Rightarrow$ (iii). Die Implikation (iii) $\Rightarrow$ (ii) folgt aus Bemerkung 5.17(ii) und Proposition 5.18. □

Wie wollen nun ein Kriterium beschreiben das hilft nicht reguläre Sprachen zu erkennen.

5.21 Satz (Pumping-Lemma)

Sei Σ ein Alphabet. Für jede reguläre Sprache $L \subseteq \Sigma^*$ gibt es eine Konstante $k \in \mathbb{N}$, so dass folgendes gilt:
Ist $z \in L$ mit $|z| \geq k$, so gilt es Wörter $umv \in \Sigma^*$ mit $z = uvw$, so dass folgendes gilt:

(i) $v \neq \lambda$

(ii) $|uv| \leq k$

(iii) $uv^iw \in L \forall i \in \mathbb{N}_0$ (?ist das w noch in dem wort oder ist es ausserhalt aber in l?)

Beweis. □

5.22 Beispiel

Die Sprache $L = \{0^n i^n : u \in \mathbb{N}_0\}$ ist nicht regulär. Dies lässt sich mit den Pumping-Lemma wie folgt zeigen.

Beweis. Angenommen L wäre regulär.
$\Rightarrow \exists k \in \mathbb{N}_0 : \forall z \in L$ mit $|z| \geq k$ gilt, $\exists u, v, w \in \Sigma^*$ mit $z = uvw$ und (i) $v \neq \lambda$ (ii) $|uv| \leq k$ (iii) $uv^iw \in L$ $\forall i \in \mathbb{N}_0$.
Sei $z := 0^k 1^k$.
Aus (i) und (ii) folgt, dass $v = 0^l$ für $l > 0$ und damit folgt $uw = 0^{k \cdot l} 1^k$ □