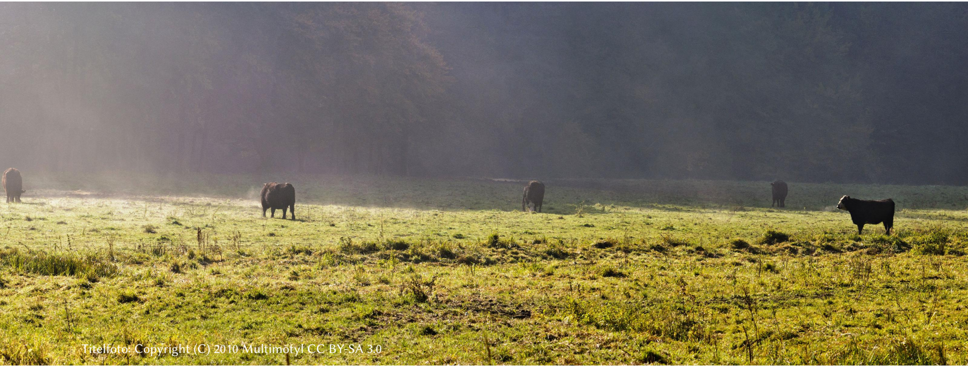
Theoretische Grundlagen der Informatik

Tutorium 7 | Wintersemester 2014/15

Moritz Klammler | 10. Februar 2015



Organisatorisches

- Ihr könnt das korrigierte letzte Übungsblatt vermutlich ab Freitag bei den Übungsleitern im Büro abholen.
- Schreibt mir ein E-Mail `moritz.klammler@student.kit.edu`, wenn Ihr Eure Punkte wissen wollt.

Tagesthemen

- Informationstheorie
- Zusammenfassung & Wiederholung
- Klausur-Übungsaufgaben
- ...

Quelle

Sei Σ ein Alphabet und p eine Wahrscheinlichkeitsverteilung auf Σ, also

$$\forall \sigma \in \Sigma : 0 \leq p(\sigma) \leq 1 \qquad \text{und} \qquad \sum_{\sigma \in \Sigma} p(\sigma) = 1 \ .$$

Eine **gedächtnislose Quelle** ist ein Gerät, das einen (potentiell unendlichen) Strom von Zeichen aus Σ erzeugt, wobei die unabhängige Wahrscheinlichkeit dafür, dass als nächstes Zeichen $\sigma \in \Sigma$ produziert wird, $p(\sigma)$ ist.

Ist das eine gedächtnislose Quelle?

- Werfen einer fairen Münze.
- Werfen einer gezinkten Münze.
- Ziehen eines „Lotto-Balls“. Wenn der Behälter leer ist, werden alle Bälle wieder eingefüllt und von vorne begonnen.
- Die Konstante 4.
- Messen der aktuellen Temperatur.
- Aufrufen der Funktion `rand` (aus der C-Standardbibliothek).

Ist das eine gedächtnislose Quelle?

- ✓ Würfeln (mit einem fairen Würfel).
  - ✓ $\Sigma = \{$⚀, ⚁, ⚂, ⚃, ⚄, ⚅$\}$
  - ✓ $p($⚀$) = p($⚁$) = p($⚂$) = p($⚃$) = p($⚄$) = p($⚅$) = \frac{1}{6}$
- ✓ Werfen einer gezinkten Münze.
  - ✓ $\Sigma = \{$€, $\}$
  - ✓ $p($€$) = \frac{1}{2} + \delta$ und $p($$) = \frac{1}{2} - \delta$ für ein festes $\delta \in [-\frac{1}{2}, \frac{1}{2}]$
- ✗ Ziehen eines „Lotto-Balls“. Wenn der Behälter leer ist, werden alle Bälle wieder eingefüllt und von vorne begonnen.
  - ✓ $\Sigma = \{1, \ldots, 49\}$
  - ✗ Die Wahrscheinlichkeiten sind nicht unabhängig

- ✓ Die Konstante 4.
  - ✓ $\Sigma = \{4\}$
  - ✓ $p(4) = 1$
- ✗ Messen der aktuellen Temperatur.
  - ☯ $\mathbb{R}$ ist kein Alphabet (der Wertebereich eines `double` allerdings schon…)
  - ✗ Temperaturmesungen sind keine stochasisch unabhängigen Ereignisse
- ☯ Aufrufen der Funktion `rand` (aus der C-Standardbibliothek).
  - ✓ $\Sigma = \{0, 1, \ldots, \texttt{RAND\_MAX}\}$
  - ☯ p ist „kompliziert“…

Information

Sei Q eine gedächtnislose Quelle, die eine Folge von Zeichen aus einem Alphabet Σ ausgibt, wobei die unabhängige Wahrscheinlichkeit dafür, dass ein Zeichen $\sigma \in \Sigma$ ausgegeben wird, $0 \leq p(\sigma) \leq 1$ sei.

Die **Information**, die ein von Q emittiertes Zeichen $\sigma \in \Sigma$ trägt, ist

$$I_\sigma(Q) = -\log(p(\sigma)) \quad .$$

Sofern der Logarithmus zur Basis 2 verwendet wird, ist die Einheit der Information Bit.

Entropie

Sei Q eine gedächtnislose Quelle, die eine Folge von Zeichen aus einem Alphabet Σ ausgibt, wobei die unabhängige Wahrscheinlichkeit dafür, dass ein Zeichen $\sigma \in \Sigma$ ausgegeben wird, $0 \leq p(\sigma) \leq 1$ sei.

Die **Entropie** der Quelle ist

$$H(Q) = -\sum_{\sigma \in \Sigma} p(\sigma) \log(p(\sigma))$$

wobei die Konvention „$0 \log(0) = 0$“ gilt.

Die Entropie ist also der Erwartungswert der Information. Sofern der Logarithmus zur Basis 2 verwendet wird, ist die Einheit der Entropie Bit.

Entropie

Sei Q eine gedächtnislose Quelle, die Zeichen aus einen Alphabet Σ mit Wahrscheinlichkeit p emittiert, und $|\Sigma| = n$.

- $0 \leq H(Q) \leq \log(n)$
- $H(Q)$ wird minimal für

  $$p(\sigma) \to \begin{cases} 1, & \sigma = \sigma_0 \\ 0, & \text{sonst} \end{cases}$$

  für ein $\sigma_0 \in \Sigma$.
- $H(Q)$ wird maximal für $p(\sigma) \to \frac{1}{n}$ für alle $\sigma \in \Sigma$

Hamming-Distanz

Sei Σ ein Alphabet und $x, y \in \Sigma^n$ für $n \in \mathbb{N}$.

Die **Hamming-Distanz** zwischen x und y ist definiert als

$$d(x,y) = \sum_{i=1}^{n} \left(1 - \delta(x_i, y_i)\right)$$

wobei $w_i \in \Sigma$ das i-te Symbol in w für $w \in \Sigma^*$ bezeichne.

- $d(w,w) = 0 \quad \forall w \in \Sigma^*$
- $d(\texttt{1},\texttt{1}) = 0$
- $d(\texttt{0},\texttt{1}) = 1$
- $d(\texttt{TGI},\texttt{GBI}) = 2$
- $d(\texttt{H2SO4},\texttt{H3PO4}) = 2$
- $d(\texttt{Alf},\texttt{red}) = 3$

Quellcodierung

Sei Q eine gedächtnislose Quelle, die Zeichen aus einem Alpgabet Σ emittiert. Eine **Quellcodierung** ist eine berechenbare bijektive Funktion $f_{\mathrm{s}} : \Sigma^+ \to \Gamma^+$ für ein Alphabet Γ.

$f_{\mathrm{s}}(Q)$ ergibt eine neue Quelle, die den von Q erzeugten Zeichenstrom codiert mit f_{s} ausgibt.

Eine gute Quellcodierung komprimiert die Daten. Das wird erreicht, wenn für möglichst alle $w \in \Gamma^+$ gilt, dass $w \in \mathrm{img}(f_{\mathrm{s}})$. Man ist daher bestrebt, f_{s} so zu wählen, dass

$$\frac{H(f_{\mathrm{s}}(Q))}{\log(|\Gamma|)}$$

maximal wird.

Kanalcodierung

Sei Q eine gedächtnislose Quelle, die Zeichen aus einem Alpgabet Σ emittiert. Eine **Kanalcodierung** ist eine berechenbare bijektive Funktion $f_{\mathrm{c}} : \Sigma^+ \to \Gamma^+$ für ein Alphabet Γ.

$f_{\mathrm{c}}(Q)$ ergibt eine neue Quelle, die den von Q erzeugten Zeichenstrom codiert mit f_{c} ausgibt.

Eine gute Kanalcodierung spreizt den Code-Raum gleichmäßig (fügt Redundanz ein). Man ist daher bestrebt, f_{c} so zu wählen, dass

$$d_{\mathsf{min}} = \min \{d(w_1, w_2) : w_1, w_2 \in \operatorname{img}(f_{\mathrm{c}}) \wedge w_1 \neq w_2\}$$

maximal und gleichzeitig die Entropie nicht unnötig weit reduziert wird.

Kanal<u>de</u>codierung

Seien Σ und Γ Alphabete, $n, m \in \mathbb{N}$ und $f_{\mathsf{c}} : \Sigma^n \to \Gamma^m$ ein Kanalcode mit $\operatorname{img}(f_{\mathsf{c}}) = C \subseteq \Gamma^m$ und Minimaldistanz d_{min}. Sei $c \in \Gamma^m$ ein empfangenes Wort.

Fall 1 $c \in C$: decodiere c zu $f_{\mathsf{c}}^{-1}(c) = w \in \Sigma^n$

Fall 2 $c \notin C$:

Fehlererkennung Breche Decodierung mit Fehler ab. Dieses Vorgehen erlaubt es, Übertragungsfehler, die bis zu $d_{\mathsf{min}} - 1$ Zeichen „kippen", zu erkennen.

Fehlerkorrektur Decodiere zu jenem $w \in \Sigma^n$, das $d(f_{\mathsf{c}}(c), w)$ minimiert (*Maximum-Likelyhood Decoding*). Dieses Vorgehen erlaubt es, Übertragungsfehler, die bis zu $\lfloor (d_{\mathsf{min}} - 1)/2 \rfloor$ Zeichen „kippen", zu korrigieren.

Quellcodierung

- erhöht Entropie
- Setzt Wissen über die Quelle voraus
- Decodierung: idR einfach (LUTs, DFAs, ...)
- Beispiele:
  - UTF-8
  - Gzip
  - Huffman-Codes

Kanalcodierung

- erhöht Redundanz...
- ...aber ohne zu viel Entropie zu opfern
- Setzt Wissen über den Kanal voraus
- Decodierung (mit Fehlerkorrektur): iA $\mathcal{NP}$-schwer (Maximum-Likelyhood)
- Beispiele:
  - Vervielfachung
  - Hamming-Codes

THIS SLIDE INTENTIONALLY LEFT BLANK

Wie zeigen wir, dass ...

... eine Sprache L regulär ist?

Wie zeigen wir, dass ...

... eine Sprache L regulär ist?

- ✓ Endlichen Automaten (Akzeptor) $\mathcal{A}$ angeben, sodass $L(\mathcal{A}) = L$
- ✓ Regulären Ausdruck für L angeben
- ✓ Rechtslineare Grammatik G angeben, sodass $L(G) = L$
- ✗ Mittels Pumping-Lemma

Wie zeigen wir, dass ...

... eine Sprache L nicht regulär ist?

Wie zeigen wir, dass ...

... eine Sprache L nicht regulär ist?

✓ Annehmen, dass L regulär ist, und mittels Pumping-Lemma für reguläre Sprachen zum Widerspruch führen

Wie zeigen wir, dass ...

... ein(e) X mit $X \in$ {endlicher Automat, Kellerautomat, Turingmaschine} [nicht] deterministisch ist?

Wie zeigen wir, dass …

… ein(e) X mit $X \in$ {endlicher Automat, Kellerautomat, Turingmaschine} [nicht] deterministisch ist?

✓ Nachschauen, ob es Zustandsübergänge mit Entscheidungsspielraum gibt

Wie zeigen wir, dass ...

... ein endlicher Automat $\mathcal{A} = (Q, \Sigma, q_0, F)$ [nicht] minimal ist?

Wie zeigen wir, dass …

… ein endlicher Automat $\mathcal{A} = (Q, \Sigma, q_0, F)$ [nicht] minimal ist?

✓ Minimierungsverfahren anwenden, um Äquivalenzklassenautomat $\mathcal{A}' = (Q', \Sigma, q_0', F')$ zu erhalten, und prüfen, ob $|Q'| < |Q|$

Wie zeigen wir, dass ...

... eine Sprache L kontextfrei ist?

Wie zeigen wir, dass ...

... eine Sprache L kontextfrei ist?

- ✓ PDA $\mathcal{K}$ angeben, sodass $L(\mathcal{K}) = L$
- ✓ Kontextfreie Grammatik G angeben, sodass $L(G) = L$
- ✗ Mittels Pumping-Lemma

Wie zeigen wir, dass ...

... eine Sprache L nicht kontextfrei ist?

Wie zeigen wir, dass ...

... eine Sprache L nicht kontextfrei ist?

- ✓ Annehmen, dass L kontextfrei ist, und mittels Pumping-Lemma für kontextfreie Sprachen zum Widerspruch führen
- ✓ Falls das nicht funktioniert, Ogdens Lemma versuchen

Wie zeigen wir, dass ...

... eine Grammatik $G = (T, V, S, P)$ ein Wort $w \in T^*$ erzeugen kann?

Wie zeigen wir, dass ...

... eine Grammatik $G = (T, V, S, P)$ ein Wort $w \in T^*$ erzeugen kann?

- ✓ Explizite Ableitung $S \Rightarrow^* w$ angeben
- ✓ Für allgemeines w mittels Induktionsbeweis
- ✓ Für kontextfreie Grammatik: CYK-Algorithmus (ggf G zunächst in CNF überführen)

Wie zeigen wir, dass ...

... eine Sprache L entscheidbar ist?

Wie zeigen wir, dass ...

... eine Sprache L entscheidbar ist?

- ✓ TM (bzw Algorithmus) angeben, die L entscheidet (akzeptiert und immer hält)
- ☯ Reduktion (muss nicht poly-many-one sein) auf bekanntermaßen entscheidbare Sprache L' angeben: $L \leq L'$ (vermutlich unnötig kompliziert)

Wie zeigen wir, dass ...

... eine Sprache L semi-entscheidbar ist?

Wie zeigen wir, dass ...

... eine Sprache L semi-entscheidbar ist?

- ✓ TM angeben, die L erkennt (akzeptiert und für jedes Wort aus L hält)
- ☯ Grammatik G angeben, sodass $L(G) = L$ (vermutlich unnötig kompliziert)
- ☯ Reduktion (muss nicht poly-many-one sein) auf bekanntermaßen semi-entscheidbare Sprache L' (zB HALT) angeben: $L \leq L'$ (vermutlich unnötig kompliziert)

Wie zeigen wir, dass ...

... eine Sprache L nicht entscheidbar (unentscheidbar) ist?

Wie zeigen wir, dass ...

... eine Sprache L nicht entscheidbar (unentscheidbar) ist?

- ✓ Annehmen, dass L entscheidbar ist ($\Rightarrow$ Entscheider und Enumerator existieren) und zum Widerspruch führen
- ✓ Zeigen, dass ein Entscheider für L eine bekanntermaßen unentscheidbare Sprache (zB HALT) entscheiden könnte
- ✓ Falls bereits bekannt ist, dass L^{C} nicht entscheidbar ist, kann L auch nicht entscheidbar sein
- ✗ Entscheider für L angeben, der nicht funktioniert

Wie zeigen wir, dass ...

... eine Sprache L nicht semi-entscheidbar ist?

Wie zeigen wir, dass ...

... eine Sprache L nicht semi-entscheidbar ist?

- ✓ Annehmen, dass L semi-entscheidbar ist ($\Rightarrow$ Akzeptor und Enumerator existieren) und zum Widerspruch führen
- ✓ Zeigen, dass ein Akzeptor für L eine bekanntermaßen nicht semi-entscheidbar Sprache (zB L_d) erkennen könnte
- ✓ Falls bereits bekannt ist, dass L nicht entscheidbar und L^C semi-entscheidbar ist, kann L nicht semi-entscheidbar sein (sonst wäre L entscheidbar)
- ✗ Akzeptor für L angeben, der nicht funktioniert

Wie zeigen wir, dass ...

... eine Sprache $L \in \mathcal{P}$ ist?

Wie zeigen wir, dass ...

... eine Sprache $L \in \mathcal{P}$ ist?

- ✓ Deterministische Turingmaschine (bzw Algorithmus) angeben, die L in Polynomialzeit entscheidet
- ☯ Poly-many-one Reduktion auf Sprache $L' \in \mathcal{P}$ angeben, also zeigen dass $L \leq_{\mathrm{p}} L'$ (vermutlich unnötig kompliziert)
- ☯ Kontextfreie (bzw reguläre) Grammatik G angeben, sodass $L(G) = L$ (vermutlich unnötig kompliziert, es sei denn für triviales L)

Wie zeigen wir, dass …

… unter der Annahme, dass $\mathcal{P} \neq \mathcal{NP}$, eine Sprache $L \notin \mathcal{P}$ ist?

Wie zeigen wir, dass ...

... unter der Annahme, dass $\mathcal{P} \neq \mathcal{NP}$, eine Sprache $L \notin \mathcal{P}$ ist?

✓ Zeigen, dass $L \in \mathcal{NPC}$ (bzw $\mathcal{NP}$-hart)

Wie zeigen wir, dass ...

... eine Sprache $L \in \mathcal{NP}$ ist?

Wie zeigen wir, dass ...

... eine Sprache $L \in \mathcal{NP}$ ist?

- ✓ Deterministische Turingmaschine (bzw Algorithmus) angeben, die einen Zeugen für L in Polynomialzeit verifiziert (Verifizierer).

  Keine Prüfung vergessen (auch nicht die trivialen)!

- ✓ Nichtdeterministische Turingmaschine angeben, die L in Polynomialzeit entscheidet (Entscheider).

- ☯ Deterministische Turingmaschine (bzw Algorithmus) angeben, die L in Polynomialzeit entscheidet.

  Nicht falsch, aber iA nur möglich, falls nebenbei auch $\mathcal{P} = \mathcal{NP}$ bewiesen wird, was nicht realistisch ist. In jedem Fall unnötig kompliziert.

- ☯ Poly-many-one Reduktion auf Sprache $L' \in \mathcal{NP}$ angeben, also zeigen dass $L \leq_p L'$ (vermutlich unnötig kompliziert)

Wie zeigen wir, dass ...

... eine Sprache $L \in \mathcal{NPC}$ ist?

Wie zeigen wir, dass ...

... eine Sprache $L \in \mathcal{NPC}$ ist?

✓ 1. Zeigen, dass $L \in \mathcal{NP}$

2. Reduktion $L \geq_{\mathrm{p}} \cdots \geq_{\mathrm{p}} \mathrm{SAT}$

Wie zeigen wir, dass ...

... es unter der Annahme, dass $\mathcal{P} \neq \mathcal{NP}$, für ein Problem Π keinen ϵ-approximativen Algorithmus mit polynomieller Laufzeit geben kann?

Wie zeigen wir, dass …

… es unter der Annahme, dass $\mathcal{P} \neq \mathcal{NP}$, für ein Problem Π keinen ϵ-approximativen Algorithmus mit polynomieller Laufzeit geben kann?

✓ Zeigen, dass ein solcher Algorithmus verwendet werden könnte, um ein $L \in \mathcal{NPC}$ in Polynomialzeit zu entscheiden. Häufig bietet es sich an, die Entscheidungssprache zu verwenden, durch deren Reduktion bereits gezeigt wurde, dass die Entscheidungssprache zu Π $\mathcal{NP}$-vollständig ist.

Das funktioniert natürlich analog auch für α-Approximierbarkeit (mit gegebenem festem $\alpha \in \mathbb{R}^+$) und für absolute Gütegarantien. Siehe die Beispiele, die wir im Laufe des Semesters zu BinPacking, TSP und Clique betrachtet haben.

Wie zeigen wir, dass ...

... eine Sprachklasse $\mathcal{L}$ abgeschlossen ist unter einer Verknüpfung $\circ$?

Wie zeigen wir, dass ...

... eine Sprachklasse $\mathcal{L}$ abgeschlossen ist unter einer Verknüpfung $\circ$?

- ✓ Konstruktiver Beweis:
  1. Seien $L_1, L_2 \in \mathcal{L}$ beliebig aber fest.
  2. Dann existieren Maschinen X_1 und X_2 mit diesen und jenen Eigenschaften (je nach $\mathcal{L}$) für L_1 und L_2, dh $L(X_1) = L_1$ und $L(X_2) = L_2$.
  3. Konstruiere aus X_1 und X_2 eine neue Maschine X mit diesen und jenen Eigenschaften für $L_1 \circ L_2$, also $L(X) = L_1 \circ L_2$.
- ✓ Argumentation über Mengenarithmetik (wenn die Abschlusseigenschaften anderer Verküpfungen bereits bekennt sind).

Wie zeigen wir, dass ...

... eine Sprachklasse $\mathcal{L}$ nicht abgeschlossen ist unter einer Verknüpfung $\circ$?

Wie zeigen wir, dass ...

... eine Sprachklasse $\mathcal{L}$ nicht abgeschlossen ist unter einer Verknüpfung $\circ$?

✓ Widerspruchsbeweis:

1. Wähle $L_1, L_2 \in \mathcal{L}$
2. Zeige, dass $L_1 \circ L_2 \notin \mathcal{L}$

✓ Argumentation über Mengenarithmetik (wenn die Abschlusseigenschaften anderer Verküpfungen bereits bekennt sind).

Wie zeigen wir, dass ...

... $\exists x \in M : p(x)$ für eine Menge M und ein unäres Prädikat p?

Quintessenz

Wie zeigen wir, dass ...

... $\exists x \in M : p(x)$ für eine Menge M und ein unäres Prädikat p?

- ✓ Ein Beispiel für ein solches x angeben (konstruktiver Beweis)
- ☯ ...

Wie zeigen wir, dass ...

... $\forall x \in M : p(x)$ für eine Menge M und ein unäres Prädikat p?

Wie zeigen wir, dass ...

Quintessenz

... $\forall x \in M : p(x)$ für eine Menge M und ein unäres Prädikat p?

- ✓ Annehmen, $\exists x \in M : \neg p(x)$ und zum Widerspruch führen (Widerspruchsbeweis).
- ✓ Falls M abzählbar und $<$ eine totale Ordnung auf M ist: Zeige $p(x_0)$ für ein $x_0 \in M$ und alle $\{x' \in M : x' < x_0\}$ sowie, dass für alle $x_1, x_2 \in M$ mit $x_0 \leq x_1 \leq x_2$ gilt: $p(x_1) \Rightarrow p(x_2)$ (Induktionsbweis).
- ✓ Zeigen, dass $M = \emptyset$ (eher ein schlechter Witz)
- ✗ Ein Beispiel für ein solches x angeben