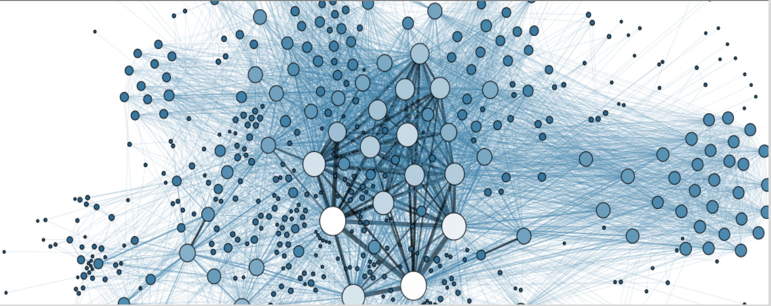
Grundbegriffe der Informatik
Tutorium 33

Lukas Bach, lukas.bach@student.kit.edu | 24.11.2016



Lukas Bach, lukas.bach@student.kit.edu

Gliederung

Lukas Bach, lukas.bach@student.kit.edu

Übersetzungen

Definition der Semantikabbildung

Sei *Sem* die Menge der Bedeutungen. Ferner seien A und B Alphabete und $L_A \subseteq A^*$ und $L_B \subseteq B^*$.
Weiter sei $sem_A : L_A \longrightarrow Sem$ und $sem_B : L_B \longrightarrow Sem$
Dann heißt $f : L_A \longrightarrow L_B$ Übersetzung , wenn gilt: für jedes $w \in L_A$ gilt $sem_A(w) = sem_B(f(w))$.

- Bedeutungserhaltende Abbildungen von Wörtern auf Wörter

Beispiel

Betrachte $Trans_{2,16} : \mathbb{Z}_{16}^* \longrightarrow \mathbb{Z}_2^*$ mit $Trans_{2,16}(w) = Repr_2(Num_{16}(w))$

- $Trans_{2,16}(A3) = Repr_2(Num_{16}(A3)) = Repr_2(163) = 10100011$

Lukas Bach, lukas.bach@student.kit.edu

Wozu Übersetzungen

- Lesbarkeit (vergleiche DF_{16} mit 11011111_2)
- Verschlüsselung
- Kompression (Informationen platzsparend aufschreiben)
- Kontextabhängige Semantiken (Deutsch $\longrightarrow$ Englisch)
- Fehlererkennung

Lukas Bach, lukas.bach@student.kit.edu

Codierungen

Definitionen

- Codewort $f(w)$ einer Codierung $f : L_A \to L_B$
- Code: $\{f(w)|w \in L_A\} = f(L_A)$
- Codierung: **Injektive** Übersetzung
  - Ich komme immer eindeutig von einem Codewort f(w) zu w zurück

Bemerkung

- Was ist, wenn L_A unendlich ist (man kann nicht alle Möglichkeiten aufzählen)
- Auswege: Homomorphismen, Block-Codierungen

Lukas Bach, lukas.bach@student.kit.edu

Homomorphismen

Definition von Homomorphismen

Seien A, B Alphabete. Dann ist $h : A^* \to B^*$ ein Homomorphismus, falls für alle $w_1, w_2 \in A^*$ gilt:

$$h(w_1 w_2) = h(w_1)h(w_2)$$

- Ein Homomorphismus ist Abbildung, die mit Konkatenation verträglich ist
- Homomorphismus ist ε-frei, wenn für jedes $x \in A : h(x) \neq \varepsilon$
- Homomorphismen lassen das leere Wort unverändert, also $h(\varepsilon) = \varepsilon$

Lukas Bach, lukas.bach@student.kit.edu

Sei h ein Homomorphismus.

Übung zu Homomorphismen

1. $h(a) = 001$ und $h(b) = 1101$. Was ist dann $h(bba)$?

$\rightarrow$ $h(bba) = h(b)h(b)h(a) = 1101 \cdot 1101 \cdot 001 = 11011101001$

2. Sei $h(a) = 01$, $h(b) = 11$ und $h(c) = \varepsilon$. Nun sei $h(w) = 011101$. Was war w?

$\rightarrow$ *aba* oder *cabccac*, ... Allgemein: $w \in \{c\}^* \cdot \{a\} \cdot \{c\}^* \cdot \{b\} \cdot \{c\}^* \cdot \{a\} \cdot \{c\}^*$ ε-Freiheit hat also die Eindeutigkeit zerstört!

3. Kann h aus 2 eine Codierung sein?

$\rightarrow$ Nein, da nicht injektiv!

4. Warum will man ε-freie Homomorphismen?

$\rightarrow$ Information geht sonst verloren!

5. Was heißt hier Ïnformation geht verloren"?

$\rightarrow$ Es gibt $w_1 \neq w_2$ mit $h(w_1) = h(w_2)$

Grundbegriffe der Informatik

Lukas Bach, lukas.bach@student.kit.edu

- Information kann auch anders "verloren"gehen

$\rightarrow$ z.B. $h(a) = 0, h(b) = 1, h(c) = 10$ – Wie das?

Präfixfreiheit

Gegeben ist ein Homomorphismus $h : A^* \rightarrow B^*$.
Wenn für keine zwei verschiedenen $x_1, x_2 \in A$ gilt, dass $h(x_1)$ Präfix von $h(x_2)$ ist, dann ist h präfixfrei.

Satz

Präfixfreie Codes sind injektiv.

Beispiele

- $h(a) = 01$ und $h(b) = 1101$ ist präfixfrei
- $g(a) = 01$ und $g(b) = 011$ ist nicht präfixfrei

Lukas Bach, lukas.bach@student.kit.edu

Huffman-Codierung

- Komprimiert eine Zeichenkette
- Kodiert häufiger vorkommende Zeichen zu kürzeren Codewörter als Zeichen die seltener vorkommen.
- Vorgehensweise:
  1. Zähle Häufigkeiten aller Zeichen der Zeichenkette
  2. Schreibe alle vorkommenden Zeichen und ihre Häufigkeiten nebeneinander
  3. Wiederhole, bis der Baum fertig ist:
     - Verbinde die zwei Zeichen mit niedrigsten Häufigkeiten zu neuem Knoten über diesen
     - Dieser hat als Zahl die aufsummierte Häufigkeiten
  4. Danach: Alle linken Kanten werden mit 0 kodiert, alle rechten Kanten mit 1

Das Ergebnis ist eine Zeichenkette aus $\{0, 1\}$, die kürzer ist als die ursprüngliche Zeichenkette in binär.

Lukas Bach, lukas.bach@student.kit.edu

Huffman-Codierung

Gegeben

- $w \in A^*$

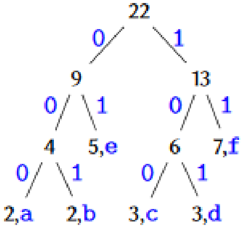
  w = `afebfecaffdeddccefbeff`
- Anzahl der Vorkommen aller Zeichen in w ($N_x(w)$)

Häufigkeiten:

x	a	b	c	d	e	f
$N_x(w)$	2	2	3	3	5	7

Zwei Phasen zur Bestimmung eines Huffman-Codes

1. Konstruieren eines "Baumes"
   - Blätter entsprechen den Zeichen
   - Kanten mit 0 und 1 beschriften



Häufigkeiten:

x	a	b	c	d	e	f
$N_x(w)$	2	2	3	3	5	7

Übung zu Huffman Codierung

Lukas Bach, lukas.bach@student.kit.edu

Übung

Sei $A = \{\mathrm{a, b, c, d, e, f, g, h}\}$

- Codiere das Wort `badcfehg` mit Hilfe der Huffman-Codierung

$\rightarrow$ Mögliche Lösung: 001 100 010 011 101 000 111 110

- Wie lauten die Codewörter, wenn für das Wort w gilt: $N_a(w) = 1, N_b(w) = 2, N_c(w) = 2, N_d(w) = 8, N_e(w) = 16, N_f(w) = 32, N_g(w) = 64, N_h(w) = 128$

Mögliche Lösung:

x	a	b	c	d	e	f	g	h
h(x)	0000000	0000001	000001	00001	0001	001	01	1

Lukas Bach, lukas.bach@student.kit.edu

- Wie lang wäre das zweite Wort ($\mathtt{abbcccc}\ \mathtt{d}^8 ... \mathtt{g}^{64}\mathtt{h}^{128}$) mit dem ersten Code codiert?

$\longrightarrow$ 741 Symbole. Also dreimal so lang wie das Original.

- Wie lang wäre das zweite Wort mit dem zweiten Code codiert?

$\longrightarrow$ 501 Symbole. Also nur zweimal so lang wie das Original.

- Was fällt euch auf?

Lukas Bach, lukas.bach@student.kit.edu

Wahr oder falsch?

Sei $h : A^* \rightarrow \mathbb{Z}_2$ eine Huffman-Codierung

- h ist ein ε-freier Homomorphismus **Wahr!**
- Häufigere Symbole werden mit langen Worten codiert, seltene mit kürzeren **Falsch!**
- Die Kompression ist am stärksten, wenn die Häufigkeiten aller Zeichen ungefähr gleich sind. **Falsch!**
- h ist präfixfrei **Wahr!**
- Es kann noch kürzere Codierungen geben **Falsch!**

Lukas Bach, lukas.bach@student.kit.edu

Huffman-Codierung

Eigenschaften

Sei A ein Alphabet und $w \in A$. Dann gilt für die Huffman-Codierung h:

- $h : A^* \to \mathbb{Z}_2$
- h ist ε-freier Homomorphismus
- h ist präfixfreier Homomorphismus
- Häufigere Symbole werden mit kurzen Worten codiert, seltene mit längeren
- Produziert kürzestmögliche Codierungen

Lukas Bach, lukas.bach@student.kit.edu

Block-Codierung mit Huffman

- Wir betrachten nicht mehr einzelne Symbole, sondern Blöcke von fester Länge $b > 1$
- Blätter des Huffman-Baums sind jetzt *Wörter der Länge b*

Beispiel an der Tafel: Codierung von
$aab \cdot deg \cdot deg \cdot aab \cdot ole \cdot aab \cdot deg \cdot aab$.

- Alphabet $A = \{\mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{d}\}$
- Text über A, der nur aus Teilwörtern der Länge 10 zusammengesetzt ist, in denen jeweils immer nur ein Symbol vorkommt
- Angenommen $\mathtt{a}^{10}$, ..., $\mathtt{d}^{10}$ kommen alle gleich häufig vor. Wie lang ist dann die Huffman-Codierung?

$\rightarrow$ Ein Fünftel, weil jeder Zehnerblock durch zwei Bits codiert wird

Lukas Bach, lukas.bach@student.kit.edu

Speicher

- Ein **Bit** ist Zeichen aus $A = \{0, 1\}$
- Ein **Byte** ist ein Wort aus acht Bits
- Abkürzungen
  - Für Bit: `bit`
  - Für Byte: `B`

Präfixe

Lukas Bach, lukas.bach@student.kit.edu

Dezimal

10^{-3}	10^{-6}	10^{-9}	10^{-12}	10^{-15}	10^{-18}
1000^{-1}	1000^{-2}	1000^{-3}	1000^{-4}	1000^{-5}	1000^{-6}
milli	mikro	nano	pico	femto	atto
m	μ	n	p	f	a
10^{3}	10^{6}	10^{9}	10^{12}	10^{15}	10^{18}
1000^{1}	1000^{2}	1000^{3}	1000^{4}	1000^{5}	1000^{6}
kilo	mega	giga	tera	peta	exa
k	M	G	T	P	E

Binär

2^{10}	2^{20}	2^{30}	2^{40}	2^{50}	2^{60}
1024^{1}	1024^{2}	1024^{3}	1024^{4}	1024^{5}	1024^{6}
kibi	mebi	gibi	tebi	pebi	exbi
Ki	Mi	Gi	Ti	Pi	Ei

Lukas Bach, lukas.bach@student.kit.edu

Gesamtzustand eines Speichers

Zu jedem Zeitpunkt ist

- für jede Adresse festgelegt, welcher Wert dort ist
- beides meist Bitfolgen

Vorstellung: Tabelle mit zwei Spalten

Adresse	Wert
Adresse 1	Wert 1
Adresse 2	Wert 2
Adresse 3	Wert 3
...	...
Adresse n	Wert n

Lukas Bach, lukas.bach@student.kit.edu

Zustand eines Speichers – formal

Definition des Speicherzustandes

Sei *Adr* die Menge aller Adressen und *Val* die Menge aller Werte. Dann ist

$$m : Adr \longrightarrow Val$$

der aktuelle Zustand des Speichers. Dabei ist $m(a)$ der aktuelle Wert an der Adresse a.

Lukas Bach, lukas.bach@student.kit.edu

Lesen und Speichern

Mem

Menge aller möglichen Speicherzustände, also Menge aller Abbildungen von *Adr* nach *Val*

$$Mem := Val^{Adr}$$

Anmerkung: Für zwei Mengen A, B gilt: $A^B := \{f : B \to A\}$.

memread

$$memread : Mem \times Adr \to Val \text{ mit } (m, a) \mapsto m(a)$$

memwrite

$memwrite : Mem \times Adr \times Val \to Mem$ mit $(m, a, v) \mapsto m'$

Für m' wird folgendes gefordert:

$$m(a') := \begin{cases} v & \text{falls } a' = a \\ m(a') & \text{falls } a' \neq a \end{cases}$$

Lukas Bach, lukas.bach@student.kit.edu

Eigenschaften von *memread* und *memwrite*

Karlsruher Institut für Technologie

Eigenschaften ("Invarianten")

- $memread(memwrite(m, a, v), a) = v$ (Also: An a einen Wert v zu schreiben und danach bei a zu lesen gibt den Wert v zurück $\Rightarrow$ Konsistente Datenhaltung)
- $memread(memwrite(m, a', v'), a) = memread(m, a)$ (Also: Auslesen einer Speicherstelle ist unabhängig davon, was vorher an eine andere Adresse geschrieben wurde $\Rightarrow$ Unabhängige Datenhaltung)

Lukas Bach, lukas.bach@student.kit.edu

Aufgaben

Aktueller Speicherzustand:

Adresse	Wert
00000	01110
00001	00100
00010	00111
00011	00000
...	...

Was ist?

- $\mathit{memread}(\mathit{memwrite}(m, \mathit{memread}(m, 00011), 01010), 00000)$

$\rightarrow$ 01010

Lukas Bach, lukas.bach@student.kit.edu

Informationen

Karlsruher Institut für Technologie

Zum Tutorium

- Lukas Bach
- Tutorienfolien auf:
  - `http://gbi.lukasbach.com`
- Tutorium findet statt:
  - Donnerstags, 14:00 - 15:30
  - 50.34 Informatikbau, -107

Mehr Material

- Ehemalige GBI Webseite:
  - `http://gbi.ira.uka.de`
  - Altklausuren!

Zur Veranstaltung

- Grundbegriffe der Informatik
- Klausurtermin:
  - 06.03.2017, 11:00
  - Zwei Stunden Bearbeitungszeit
  - 6 ECTS für Informatiker und Informationswirte, 4 ECTS für Mathematiker und Physiker

Zum Übungsschein

- Übungsblatt jede Woche
- Ab 50% insgesamt hat man den Übungsschein
- Keine Voraussetzung für die Klausur, aber für das Modul