10 ÜBERSETZUNGEN UND CODIERUNGEN

Hinweise für die Tutorien

10.1 VON WÖRTERN ZU ZAHLEN UND ZURÜCK

10.1.1 Dezimaldarstellung von Zahlen

zur Definition

$$\mathrm{Num}_{10}(\varepsilon) = 0$$
$$\forall w \in Z_{10}^* \; \forall x \in Z_{10} : \mathrm{Num}_{10}(wx) = 10 \cdot \mathrm{Num}_{10}(w) + \mathrm{num}_{10}(x)$$

noch ein Beispiel rechnen?

10.1.2 Andere Zahldarstellungen

- beachte Analogie der Definitionen von Num_{10} und Num_2

$$\mathrm{Num}_2(\varepsilon) = 0$$
sowie $$\forall w \in Z_2^* \; \forall x \in Z_2 : \mathrm{Num}_2(wx) = 2 \cdot \mathrm{Num}_2(w) + \mathrm{num}_2(x)$$

- Algorithmus für Umwandlung von Wort nach Zahl erarbeiten:

```
// Eingabe: w ∈ Z_2^*
x ← 0
for i ← 0 to |w| − 1 do
    x ← 2x + num_2(w(i))
od
// am Ende: x = Num_2(w)
```

- Die Schleifeninvariante sieht man besser bei

```
// Eingabe: w ∈ Z_2^*
x ← 0
v ← ε
for i ← 0 to |w| − 1 do
    v ← v · w(i)
    x ← 2x + num_2(w(i))
od
// am Ende: v = w ∧ x = Num_2(w)
```

- Invariante suchen und finden lassen: $x = \mathrm{Num}_2(v)$
  Dass das eine Schleifeninvariante ist, nicht in allen Details beweisen. Aber den Kern erkennen: laut Definition von Num_2 ist nämlich
  $$\mathrm{Num}_2(v \cdot w(i)) = 2\mathrm{Num}_2(v) + \mathrm{num}_2(w(i))$$
- klar machen, wie allgemein bei Basis k die Umwandlung funktioniert: $\mathrm{Num}_k(wx) = k \cdot \mathrm{Num}_k(w) + \mathrm{num}_k(x)$.
- Beispiel rechnen, z.B. $\mathrm{Num}_3(\mathtt{111}) = \cdots = 13$.
- $\mathrm{Num}_2(\mathtt{1}) = 1$, $\mathrm{Num}_2(\mathtt{11}) = 3$, $\mathrm{Num}_2(\mathtt{111}) = 7$, $\mathrm{Num}_2(\mathtt{1111}) = 15$,
  Wer sieht allgemein: $\forall m \in \mathbb{N}_0 : \mathrm{Num}_2(\mathtt{1}^m) = 2^m - 1$?
  Wie überträgt sich das auf den Fall $k = 3$? $\forall m \in \mathbb{N}_0 : \mathrm{Num}_3(\mathtt{2}^m) = 3^m - 1$.
- Und dann vielleicht noch die folgende Spielerei:
  - $Z_3' = \{\mathtt{1}, \mathtt{0}, \mathtt{\bar{1}}\}$ mit
    $\mathrm{num}_3'(\mathtt{\bar{1}}) = -1$, $\mathrm{num}_3'(\mathtt{0}) = 0$, und $\mathrm{num}_3'(\mathtt{1}) = 1$ sowie
    $$\mathrm{Num}_3'(\varepsilon) = 0$$
    $$\forall w \in {Z_3'}^* \; \forall x \in Z_3' : \mathrm{Num}_3'(wx) = 3 \cdot \mathrm{Num}_3'(w) + \mathrm{num}_3'(x)$$
  - Man berechne erst mal z. B. $\mathrm{Num}_3'(\mathtt{\bar{1}10})$ (gibt -6) und $\mathrm{Num}_3'(\mathtt{1\bar{1}1})$ (gibt 7)
  - Was passiert, wenn man in einer Zahldarstellung aus allen $\mathtt{1}$ ein $\mathtt{\bar{1}}$ macht und umgekehrt?: Darstellung der entsprechenden negierten Zahl.
    Z.B. $\mathrm{Num}_3'(\mathtt{\bar{1}1\bar{1}}) = -\mathrm{Num}_3'(\mathtt{1\bar{1}1}) = 6$

10.1.3 Von Zahlen zu ihren Darstellungen

- Eventuell noch mal die Spielerei mit Num_3':
  - Welche positiven Zahlen haben eine Repräsentation? Welche negativen? (Antwort: alle alle) Und die Null geht natürlich auch.
  - Wie sieht man einem Wort an, ob es eine positive oder eine negative Zahl repräsentiert? (betrachte vorderste Nicht-Null: $\mathtt{1}$ oder $\mathtt{\bar{1}}$?)
  - (Hausaufgabe für Tüftler: Wie addiert man zwei solche Zahlen?)
  - Hinweis: Im Rechner benutzt man aber nur $Z_2 = \{\mathtt{0}, \mathtt{1}\}$. Da muss man sich was anderes überlegen für negative Zahlen.

10.2 VON EINEM ALPHABET ZUM ANDEREN

10.2.1 Ein Beispiel: Übersetzung von Zahldarstellungen

Warum macht man Übersetzungen? Zumindest die folgenden vier Möglichkeiten fallen einem ein:

- *Lesbarkeit:*
- *Kompression:*
- *Verschlüsselung:*
- *Fehlererkennung* und *Fehlerkorrektur:*

Fällt Ihnen noch was ein?

10.2.2 Homomorphismen

- Definitionen: Es seien A und B zwei Alphabete und $h : A \to B^*$ eine Abbildung. Zu h kann man in der Ihnen inzwischen vertrauten Art eine Funktion $h^{**} : A^* \to B^*$ definieren vermöge

$$h^{**}(\varepsilon) = \varepsilon$$
$$\forall w \in A^* : \forall x \in A : h^{**}(wx) = h^{**}(w)h(x)$$

  Homomorphismus ε-frei, wenn für alle $x \in A$ gilt: $h(x) \neq \varepsilon$.
- Beispiel:
  - $h(\texttt{a}) = \texttt{001}$ und $h(\texttt{b}) = \texttt{1101}$
  - dann ist $h(\texttt{bba}) = h(\texttt{b})h(\texttt{b})h(\texttt{b}) = \texttt{1101} \cdot \texttt{1101} \cdot \texttt{001} = \texttt{11011101001}$
- ε-freier Homomorphismus: Warum will man das? Sonst geht „Information verloren“. keine Codierung mehr; Betrachte
  - $h(\texttt{a}) = \texttt{001}$ und $h(\texttt{b}) = \varepsilon$
  - angenommen $h(w) = \texttt{001}$ Was war dann w? Man weiß nur: es kam genau ein `a` vor, aber wieviele `b` und wo ist nicht klar.
- präfixfreier Code: für *keine* zwei verschiedenen Symbole $x_1, x_2 \in A$ gilt: $h(x_1)$ ist ein Präfix von $h(x_2)$.

  Beispiel
  - $h(\texttt{a}) = \texttt{001}$ und $h(\texttt{b}) = \texttt{1101}$ ist präfixfrei
  - $h(\texttt{a}) = \texttt{01}$ und $h(\texttt{b}) = \texttt{011}$ ist *nicht* präfixfrei
  - Präfixfreiheit leicht zu sehen, wenn alle $h(x)$ gleich lang sind: präfixfrei $\iff$ injektiv; Beispiel: ASCII

10.2.3 Beispiel Unicode: UTF-8 Codierung

Man könnte, wenn die Zeit reicht, ja mal für ein paar Zeichen die UTF-8 Codierung bestimmen. Zum Beispiel gibt es für π in Unicode ein Zeichen, nämlich das mit der Nummer `0x03C0`. Und das Integralzeichen $\int$ hat Nummer `0x222B`.

Wenn ich den Algorithmus richtig gemacht habe, ergibt sich

- für π: Code Point `0x03C0`, in Bits `0000 0011 1100 0000` = `00000 01111 000000` :

  man benutzt die Zeile

Char. number range (hexadecimal)	UTF-8 octet sequence (binary)
`0000 0080` - `0000 07FF`	`110`*xxxxx* `10`*xxxxxx*

  also UTF-8 Codierung `11001111 10000000`
- für $\int$: Code Point `0x222B`, in Bits `0010 0010 0010 1011` = `0010 001000 101011`
  man benutzt die Zeile

Char. number range (hexadecimal)	UTF-8 octet sequence (binary)
0000 0800 - 0000 FFFF	1110*xxxx* 10*xxxxxx* 10*xxxxxx*

also UTF-8 Codierung 11100010 10001000 10101011

- Man überlege sich: UTF-8 ist präfixfrei

10.3 HUFFMAN-CODIERUNG

Nehmen Sie acht Symbole: a, b, c, d, e, f, g, h

- 1. Fall: Jedes Zeichen kommt genau einmal vor.
  Huffman-Code-Baum erstellen, Wort badcfehg codieren, wie lang wird die Codierung?
- 2. Fall: a kommt einmal vor, b zweimal, c 4-mal, d 8-mal, e 16-mal, f 32-mal, g 64-mal, h 128-mal.
  Huffman-Code-Baum erstellen, Wort badcfehg codieren, wie lang wird die Codierung?
- Wie lang wird ein Wort mit zweiter Zeichenverteilung, wenn man es mit dem ersten Code codiert?
- Wie lang wird ein Wort mit erster Zeichenverteilung, wenn man es mit dem zweiten Code codiert?
- Ziel: Sehen, dass Huffman-Codierung irgendwie zu funktionieren scheint, aber eben h auf das zu codierende Wort w zugeschnitten wird.

10.3.1 Weiteres zu Huffman-Codes

Verallgemeinerung: nicht von den Häufigkeiten einzelner Symbole ausgehen, sondern für Teilwörter einer festen Länge $b > 1$ die Häufigkeiten berechnen.

Beispiel: Man hat einen Text über dem Alphabet $\{a, b, c, d\}$, der nur aus Teilwörtern der Länge 10 zusammengesetzt ist, die denen in jedem immer nur ein Symbol vorkommt, also aaaaaaaaaabbbbbbbbbbddddddddddccccccccccc usw. Angenommen $a^{10}, \ldots, d^{10}$ kommen alle gleich häufig vor. Wie lang ist dann die Huffman-Codierung?

Ein Fünftel, weil jeder Zehnerblock durch zwei Bits codiert wird.

