# 64-041 Übung Rechnerstrukturen WS 15/16

## Aufgabenblatt 7 Ausgabe: 25.11., Abgabe: 02.12. 24:00

| Gruppe | |
|---|---|
| Name(n) | Matrikelnummer(n) |
| | |

### Aufgabe 7.1 (Punkte 5+5+5+10)

*Entropie und Redundanz*: Für einige Anwendungen (z.B. Finanzmathematik) werden Berechnungen im Dezimalsystem bevorzugt. Wir betrachten die Entropie $H$ und Redundanz $R$ verschiedener Codierungen der Dezimalziffern $(0 \ldots 9)$:

(a) Im ersten Ansatz analyisieren Sie die einfache BCD-Codierung (binary coded decimal), die den Ziffern von 0 bis 9 die zugehörigen 4-bit Bitmuster aus dem Dualcode zuordnet, also $0000, 0001, \ldots, 1001$.

Wir nehmen zunächst an, dass die einzelnen Ziffern mit gleicher Häufigkeit auftreten. Geben Sie den möglichen Informationsgehalt $H_0$, die Entropie $H$ und die Redundanz $R$ dieser Codierung an.

(b) Versuchen Sie, die Redundanz zu verkleinern, indem Sie jeweils zwei Dezimalziffern zu einem Codewort zusammenfassen, also $\{00, 01, 02, \ldots\ 10, 11, \ldots\ 97, 98, 99\}$.

Wie viele Bits werden für die Codewörter benötigt? Geben Sie Ihren Code, den möglichen Informationsgehalt $H_0$, die Entropie $H$ und die Redundanz $R$ an. Wie groß ist jetzt die Redundanz bezogen auf eine einzelne Dezimalziffer?

(c) Wie groß ist der der mögliche Informationsgehalt $H_0$ (jeweils bezogen auf eine einzelne Ziffer) für Gruppen von drei (000... 999) Ziffern?

(d) In vielen realen Datensätzen und Meßdaten treten die Dezimalziffern nicht mit gleicher Wahrscheinlichkeit auf, sondern lassen sich durch das Benford'sche Gesetz sehr gut beschreiben. Lesen Sie dazu den Hintergrund nach, zum Beispiel unter https://de.wikipedia.org/wiki/Benfordsches_Gesetz.

Reduzieren Sie jetzt die Redundanz, indem Sie die Dezimalziffern 0 bis 9 auf einen Code mit variabler Länge (Fano) abbilden. Nehmen Sie dazu die Wahrscheinlichkeiten der einzelnen Ziffern entsprechend des Benford'schen Gesetzes (für eine einzelne Ziffer) an.

Geben Sie Ihren Code, den möglichen Informationsgehalt $H_0$, die Entropie $H$ und die Redundanz $R$ an.

## Aufgabe 7.2 (Punkte 15+5)

*Hamming-Code*: Entsprechend dem in der Vorlesung vorgestellten Schema wird ein 7-Bit Hamming- Code gebildet, um Einzelbitfehler korrigieren zu können. Wie in der Tabelle dargestellt, besitzt er vier Informationsbits ($d_i$) und drei Prüfbits ($p_j$). Insgesamt sind $2^4 = 16$ Informationen codierbar; die Codewörter sind in der linken Tabelle aufgelistet:

| | $c_1$ | $c_2$ | $c_3$ | $c_4$ | $c_5$ | $c_6$ | $c_7$ |
|---|---|---|---|---|---|---|---|
| Nr. | $p_1$ | $p_2$ | $d_1$ | $p_3$ | $d_2$ | $d_3$ | $d_4$ |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 1 | 0 | 1 | 0 | 0 | 1 |
| 2 | 0 | 1 | 0 | 1 | 0 | 1 | 0 |
| 3 | 1 | 0 | 0 | 0 | 0 | 1 | 1 |
| 4 | 1 | 0 | 0 | 1 | 1 | 0 | 0 |
| 5 | 0 | 1 | 0 | 0 | 1 | 0 | 1 |
| 6 | 1 | 1 | 0 | 0 | 1 | 1 | 0 |
| 7 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| 8 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| 9 | 0 | 0 | 1 | 1 | 0 | 0 | 1 |
| 10 | 1 | 0 | 1 | 1 | 0 | 1 | 0 |
| 11 | 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| 12 | 0 | 1 | 1 | 1 | 1 | 0 | 0 |
| 13 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |
| 14 | 0 | 0 | 1 | 0 | 1 | 1 | 0 |
| 15 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

| Codewortstelle | $c_1$ | $c_2$ | $c_3$ | $c_4$ | $c_5$ | $c_6$ | $c_7$ |
|---|---|---|---|---|---|---|---|
| Bedeutung | $p_1$ | $p_2$ | $d_1$ | $p_3$ | $d_2$ | $d_3$ | $d_4$ |
| Prüfgruppe A | * | | * | | * | | * |
| Prüfgruppe B | | * | * | | | * | * |
| Prüfgruppe C | | | | * | * | * | * |

Für die Prüfstellen gilt:

$c_1 = c_3 \oplus c_5 \oplus c_7$
$c_2 = c_3 \oplus c_6 \oplus c_7$
$c_4 = c_5 \oplus c_6 \oplus c_7$

Um (einen) Einzelbitfehler zu lokalisieren, bildet man ein Prüfwort $(x_a, x_b, x_c)$, wobei gilt:

$x_a = c_1 \oplus c_3 \oplus c_5 \oplus c_7$
$x_b = c_2 \oplus c_3 \oplus c_6 \oplus c_7$
$x_c = c_4 \oplus c_5 \oplus c_6 \oplus c_7$

(a) Zeigen Sie anhand einiger Beispiele, wie ein auftretender Einzelbitfehler lokalisiert und damit korrigiert werden kann. Verfälschen Sie dazu die erst die Codewortstelle $c_1$ und dann die Stelle $c_7$ im 9. Codewort (bzw. Nr. 8 in der Tabelle) sowie $c_2$ im 2. Codewort und bilden Sie die Prüfbits.

(b) Wie kann man dann aus dem Prüfwort die fehlerhafte Codewortstelle bestimmen?

## Aufgabe 7.3 (Punkte 15+15)

*Kanonische Formen*: Die beiden folgenden Funktionen einer 3-bit Variablen $x$ sind in der kanonischen DNF, der kanonischen KNF und der Reed-Muller-Form zu notieren.

(a) $f(x) = (x_3 \vee \overline{x_2}) \wedge (x_2 \vee \overline{x_1})$

(b) $g(x) = \overline{x_2} \oplus \overline{x_1}$

## Aufgabe 7.4 (Punkte 5+5+5+5+5)

*NAND als vollständige Basis*: Geben Sie an, wie die folgenden boolschen Funktionen durch geeignete Schaltungen nur aus (einem oder mehreren) NAND-Gattern gebildet werden können. Für diese NAND-Verknüpfung können Sie der Übersichtlichkeit halber das Symbol $\circ$ verwenden. Ebenso ist die Einführung von Hilfsvariablen erlaubt, wenn das der Klarheit dient.

(a) $f_1(a, b) = \overline{a}$

(b) $f_2(a, b) = a \vee b$

(c) $f_3(a, b) = a \wedge b$

(d) $f_4(a, b) = \mathbf{0}$ (die Funktion, die immer eine 0 liefert)

(e) Zeigen Sie jetzt noch anhand eines Beispiels, dass die NAND-Verknüpfung, anders als AND und OR, nicht assoziativ ist, d.h., es gilt nicht unbedingt $a \circ (b \circ c) = (a \circ b) \circ c$.