

Kapitel 6

Fehlertoleranz

Albert-Ludwigs-Universität Freiburg

Dr. Tobias Schubert, Dr. Ralf Wimmer

Professur für Rechnerarchitektur
WS 2016/17

- Bisher: Rechnerarchitektur am Beispiel von ReTI
- Fehler auf verschiedenen Ebenen entdecken und beheben
 - Fehler durch Fertigungsdefekte, Störungen: Kapitel 6
 - Konzeptuelle Fehler, Entwurfsfehler: Kapitel 7
- Allgemeine Rechnerarchitektur und Entwurfskonzepte

- Bei der Informationsverarbeitung und -Übermittlung können physikalische Störungen auftreten.
 - Elektrisches Rauschen, Radiation, Defekte.
 - Absichtliche Manipulation durch Angreifer.
- Die Störungen äußern sich darin, dass Wert 0 statt Wert 1 berechnet/übertragen wird und umgekehrt.
 - „Kippen“ des Werts $0 \rightarrow 1$, $1 \rightarrow 0$.
- Hier konzentrieren wir uns auf Fehler bei der Datenübertragung.
 - Manche Fehler betreffen direkt die Hardware oder andere Systemteile. Ihre Behandlung ist komplexer.

Sei $A = \{a_1, \dots, a_m\}$ ein endliches Alphabet der Größe m .

- Eine Abbildung $c: A \rightarrow \{0, 1\}^n$ heißt **Code fester Länge**, falls c injektiv ist.
- Die Menge $c(A) := \{w \in \{0, 1\}^n \mid \exists a \in A : c(a) = w\}$ heißt Menge der **Codewörter**.
- Minimale Codelänge: Für einen Code $c: A \rightarrow \{0, 1\}^n$ fester Länge gilt: $n \geq \lceil \log_2 m \rceil$.

$a \rightarrow 000$	$e \rightarrow 100$
$b \rightarrow 001$	$f \rightarrow 101$
$c \rightarrow 010$	$g \rightarrow 110$
$d \rightarrow 011$	$h \rightarrow 111$

Fehler bei Datenübertragung (1/2)

$$w \xrightarrow{w \neq \tilde{w}} \tilde{w}$$

Annahme:

- Sei c ein Code minimaler fester Länge n .
- Ein Datum a (z.B. ein Buchstabe, eine Zahl) wird, durch ein Codewort $w = c(a)$ repräsentiert übertragen.
- Sei $\tilde{w} \in \{0, 1\}^n$ das empfangene Wort.
- Bei der Übertragung (z.B. über Internet) können einzelne Bits von $c(a)$ kippen. Dann ist $w \neq \tilde{w}$.

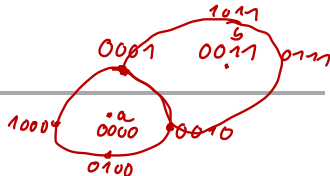
Ziel:

- Durch Verändern des Codes c in einen Code C fester Länge $\underline{n} + \underline{r}$ sollen diese Bits
 - erkannt
 - Fehlererkennende Codes
 - Bsp. für 1-fehlererkennenden Code: Parity-Code
 - korrigiert
 - Fehlerkorrigierende Codes
 - Bsp. für 1-fehlerkorrigierenden Code: Hamming-Code
- werden.

Fehlererkennende Codes

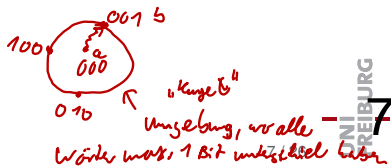
Idee:

- Wähle Codewörter $\underline{w} \in c(A)$ so, dass nach Kippen von Bits Wörter $\underline{\tilde{w}} \in \{0, 1\}^n$ entstehen, die **keine Codewörter** sind, d.h. $\underline{\tilde{w}} \notin c(A)$.
- Wird ein **Nicht-Codewort** empfangen, so muss ein **Übertragungsfehler** aufgetreten sein.
- Benutze Codes mit $\underline{n} = \underline{\lceil \log_2 m \rceil} + \underline{r}$ mit $r > 0$.
- Benutze die r zusätzlichen Bits zum **Test auf Übertragungsfehler**.
- Beispiel: Parity-Code



$|r|=1$

a	000	0
b	001	1
	010	1
	011	0
	100	1
	101	0
	110	0
	111	1



- Eine Bitfolge $w \in \{0, 1\}^n$ besteht den **Paritätstest** (engl. **Parity-Check**), wenn die Anzahl der auf **1** gesetzten Bitstellen gerade ist.
- Sei $c : A \rightarrow \{0, 1\}^n$ ein Code fester Länge von A . Betrachte den Code $C : A \rightarrow \{0, 1\}^{n+1}$, der aus Code c entsteht, in dem eine Bitstelle an jedes Codewort $c(a)$ hinten angefügt wird und so gesetzt wird, dass der neue Code $C(a)$ den Paritätstest besteht.

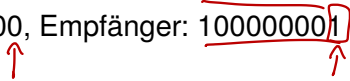
- Ein Code $c : A \rightarrow \{0, 1\}^n$ fester Länge heißt k -fehlererkennend, wenn der Empfänger in jedem Fall entscheiden kann, ob ein gesendetes Codewort durch Kippen von bis zu k Bits verfälscht wurde.
- Der Parity-Code C ist 1-fehlererkennend.
 - **Beweis:** Kippt bei der Übertragung von $C(a)$ genau eine Bitstelle, so kommt eine Bitfolge an, die den Paritätstest nicht besteht und somit kein Codewort von C darstellt. Überprüft der Empfänger die Parität der empfangenen Bitfolge, kann er auf einen Fehler schließen.



- Ein Code $c : A \rightarrow \{0, 1\}^n$ fester Länge heißt **k -fehlererkennend**, wenn der Empfänger in jedem Fall entscheiden kann, ob ein gesendetes Codewort durch Kippen von bis zu k Bits verfälscht wurde.
- Der Parity-Code C ist **1-fehlererkennend**.
 - **Beweis:** Kippt bei der Übertragung von $C(a)$ genau eine Bitstelle, so kommt eine Bitfolge an, die den Paritätstest nicht besteht und somit kein Codewort von C darstellt. Überprüft der Empfänger die Parität der empfangenen Bitfolge, kann er auf einen Fehler schließen.

SMILE - Fehlererkennung bei Parity-Code

Welche Fehler bei der Übertragung werden durch den Paritätstest beim Empfänger erkannt? Das letzte Bit bei Sender und Empfänger sei dabei das Parity-Bit.

- a. Sender: 100010101, Empfänger: 100010101 ✗
 - b. Sender: 100110100, Empfänger: 100010100 ✓
 - c. Sender: 111010101, Empfänger: 101010101 ✓
 - d. Sender: 000000000, Empfänger: 100000001 ✗
- 

Hamming-Abstand (1/2)

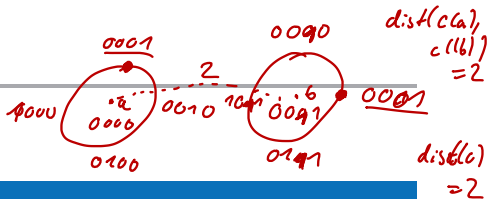
Definition

Der Hamming-Abstand $\text{dist}(v, w)$ zweier n -Bitfolgen v und w ist die Anzahl der Stellen, an denen v und w sich unterscheiden.

- $\text{dist}(\underline{00001101}, \underline{10001100}) = 2$
- $\text{dist}(00001101, 00001101) = 0$
- Ist v das übertragene und w das empfangene Codewort, so liegt ein Übertragungsfehler genau dann vor, wenn $\text{dist}(v, w) \neq 0$.
 - Ein Übertragungsfehler heißt einfach, wenn $\text{dist}(v, w) = 1$.
- Der Hamming-Abstand eines Codes $c : A \rightarrow \{0, 1\}^n$ ist der kleinste Abstand zweier Codewörter von c :
 $\text{dist}(c) := \min\{\text{dist}(c(a_i), c(a_j)); a_i, a_j \in A \text{ mit } a_i \neq a_j\}.$

Handwritten example:
 $\begin{matrix} & 0 & 0 & 0 & 1 \\ 2 \left[& 0 & 1 & 0 & 0 \\ & 1 & 0 & 1 & 0 \end{matrix} \right. \begin{matrix} 1 \\ 2 \\ 3 \end{matrix}$
 $\text{dist}(\dots) = 1$

Hamming-Abstand (2/2)



Lemma

Ein Code c fester Länge ist genau dann k -fehlererkennend, wenn $\text{dist}(c) \geq k + 1$ gilt.

$k+1 \Rightarrow k$ -fehlererkennend

- **Beweisidee:** Durch das Kippen von bis zu k Bits kann aus Codewort $a \in c(A)$ kein anderes Codewort $a^* \in c(A)$ entstehen, denn sonst wäre $\text{dist}(c(a), c(a^*)) = k$, was kleiner als der Hamming-Abstand des Codes wäre.

Idee:

- Benutze r **zusätzliche Bits**, so dass das gesendete Codewort aus dem empfangenen Wort **rekonstruiert** werden kann.
- Beispiel: **Hamming-Code**

Hamming-Code

3 2 1 6 5 4 3 2 1
↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓
0 0 0 → 0 0 _ 0 _ _
0 0 1 0 0 _ 1 _ _
0 1 0
0 0 1
1 0 0
1 0 1
1 1 0
1 1 1

- Benutze die Bitstellen $\underline{2^0}$, $\underline{2^1}$, ..., $\underline{2^{r-1}}$ als Überprüfungsbits, wobei die Bitstelle $\underline{2^j}$ die Bitstellen überprüft, deren Binärdarstellungen an der \underline{j} -ten Stelle eine 1 haben. $2^0=1$ $2^1=2$
 $2^2=4$
- Die Bitstelle $\underline{2^j}$ wird so belegt, dass gerade viele Bitstellen, deren Binärdarstellungen an der \underline{j} -ten Stelle eine 1 haben, gesetzt sind. (vgl. Paritätstest)

Hamming-Code an einem Beispiel

- Uncodiertes Wort: 0111 0101 0000 1111.

→ $m = 16$.

↳ Länge des Code Wort

- Konstruktion des Hamming-codierten Codeworts:

- Das Wort wird unter Auslassung der „Zweierpotenz“-Bitstellen aufgeschrieben:

01110 _ 1010000 _ 111 _ 1 _ _ .
 2^4 2^3 2^2 $2^1 2^0$

- Dies ergibt insgesamt 21 Bitstellen ($r = 5$).
- Die „Zweierpotenz“-Bitstellen werden als Überprüfungsbits benutzt (Nummerierung beginnt rechts mit der Stelle 1).
- Zur Erinnerung: Die Bitstelle 2^j wird so belegt, dass gerade viele Bitstellen, deren Binärdarstellungen an der j -ten Stelle eine 1 haben, gesetzt sind.

Hamming-Code-Beispiel (1/4)

011101010001111
 01110_101000_111_1_--
 ↑
 2¹ ... 111
 321

	2 ⁴	2 ³	2 ²	2 ¹	2 ⁰	zu codierende Bitfolge
3						↑
5						1
6						1
7						1
9						0
10						0
11						0
12						0
13						1
14						0
15						1
17						0
18						1
19						1
20						1
21						0

11
 101
 110
 111
 1001
 1016
 ...
 1

Das Überprüfungsbit 2^j überprüft die Bitstellen, die in ihrer Binärdarstellung an der j -ten Stelle eine 1 haben.

Hamming-Code-Beispiel (2/4)

	2^4	2^3	2^2	2^1	2^0	zu codierende Bitfolge
3				—	—	1
5				—	—	1
6			—		—	1
7			—	—		1
9		—			—	0
10		—		—		0
11		—		—	—	0
12		—	—			0
13		—	—		—	1
14		<u>0</u>	<u>0</u>	<u>0</u>		0
15		<u>1</u>	<u>1</u>	<u>1</u>	<u>1</u>	1
17	—				—	0
18	—			—		1
19	—			—	—	1
20	—		—			1
21	—		—		—	0

Das Überprüfungsbit 2^j überprüft die Bitstellen, die in ihrer Binärdarstellung an der j -ten Stelle eine 1 haben.

$$10101_2 = 21_{10}$$

$$11111_2 = 31_{10}$$

$$43210$$

Hamming-Code-Beispiel (3/4)

	2^4	2^3	2^2	2^1	2^0	zu codierende Bitfolge
3				1 0	1 0	1 ←
5			1		1	1
6			1 0	1 0		1 ← 0
7			1 0	1 0	1 0	1 ← 0
9		0			0	0
10		0 0 1		0 0 1		0 → 1
11		0		0	0	0
12		0	0			0
13		1	1		1	1
14		0	0 1	0		0
15		1	1	1	1	1
17	0				0	0
18	1			1		1
19	1			1	1	1
20	1 0		1 0			1 ←
21	0		0		0	0

Der Prüfbitwert ergibt sich aus der Summe modulo 2 der jeweiligen Spalte.

Das Überprüfungsbit 2^j überprüft die Bitstellen, die in ihrer Binärdarstellung an der j -ten Stelle eine 1 haben.

Parity Code

1 0 0 0 0

↑ 2^3 + 2^1 = 10

Hamming-Code Beispiel (4/4)

- Die Bitfolge 0111 0101 0000 1111
- wird mit dem Hamming-Code zum Codewort 0 1110 1101 0000 0111 0100.

Und wie findet man einen Fehler? (1/2)

- Nehme einen Übertragungsfehler an Position 13 des Codeworts an.
- Fehlerhaft empfangenes Wort:
0 1110 11000000 0111 0100.

Und wie findet man einen Fehler? (2/2)

	2^4	2^3	2^2	2^1	2^0	empfangene Bitfolge
3				1	1	1
5			1		1	1
6			1	1		1
7			1	1	1	1
9		0			0	0
10		0		0		0
11		0		0	0	0
12		0	0			0
13		<u>0</u>	<u>0</u>		<u>0</u>	0
14		0	0	0		0
15		1	1	1	1	1
17	0				0	0
18	1			1		1
19	1			1	1	1
20	1		1			1
21	0		0		0	0

⇒ Fehler muss sich in
Zeile $8 + 4 + 1 = 13$
befinden!

$$1101_2 = 13_{10}$$

"5 Einsen"

1	0	0	0	0
---	---	---	---	---

Die Spalten 8,4 und 1 bestehen
den Paritätstest nicht!

Definition

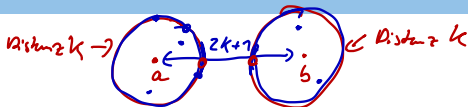
Ein Code $c : A \rightarrow \{0, 1\}^n$ fester Länge heißt k -**fehlerkorrigierend**, wenn der Empfänger in jedem Fall entscheiden kann, ob ein gesendetes Codewort w durch Kippen von bis zu k Bits verfälscht wurde und daraufhin w aus dem empfangenen Wort \tilde{w} rekonstruiert werden kann.

- Der Hamming-Code ist 1-fehlerkorrigierend. Die Anzahl der Zusatzbits $r = 1 + \lfloor \log_2 m \rfloor$ ist minimal (Korrektheit folgt aus noch folgendem Satz).

Lemma

Ein Code c fester Länge ist genau dann k -fehlerkorrigierend, wenn $\text{dist}(c) \geq 2k + 1$ gilt.

$$2 \cdot 1 + 1 = 3$$



Beweis:

- Sei $\underline{M(c(a_i), k)} := \{ \underline{w} \in \{0, 1\}^n \mid \underline{\text{dist}(c(a_i), w)} \leq k \}$ die Kugel um $c(a_i)$ mit Radius k .
- Dann gilt:
 c ist k -fehlerkorrigierend $\Leftrightarrow \forall a_i, a_j, i \neq j$ gilt: $M(c(a_i), k) \cap M(c(a_j), k) = \emptyset$.
- Für den Beweis ist also zu zeigen:
 $[\forall a_i, a_j, i \neq j : M(c(a_i), k) \cap M(c(a_j), k) = \emptyset] \Leftrightarrow \underline{\underline{\text{dist}(c) \geq 2k + 1}}$.

Beweis der Hilfs-Behauptung

$$A \Rightarrow B$$

$$\neg B \Rightarrow \neg A$$

$$\rightarrow \blacksquare [\forall a_i, a_j \ i \neq j : \underbrace{M(c(a_i), k)}_{A} \cap \underbrace{M(c(a_j), k)}_{B} = \emptyset] \Leftrightarrow \text{dist}(c) \geq 2k + 1.$$

"B"

$$c(a_i) = b_0$$

$$c(a_j) = b_{2k}$$

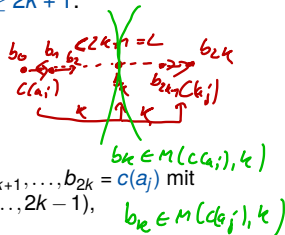
■ Beweis „ \Rightarrow “:

■ Annahme: $\text{dist}(c) < 2k + 1$

■ D.h. $\exists a_i, a_j$ mit $\text{dist}(c(a_i), c(a_j)) = l$ und $l < 2k + 1$;

■ also gibt es eine Folge: $c(a_i) = b_0, b_1, \dots, b_{k-1}, b_k, b_{k+1}, \dots, b_{2k} = c(a_j)$ mit
 $\text{dist}(b_i, b_{i+1}) = 0$ oder $\text{dist}(b_i, b_{i+1}) = 1$ (für alle $i = 0, \dots, 2k - 1$),

■ also $b_k \in M(c(a_i), k) \cap M(c(a_j), k)$. $\hookrightarrow \Rightarrow \neg A$



■ Beweis „ \Leftarrow “:

\rightarrow ■ Annahme: $M(c(a_i), k) \cap M(c(a_j), k) \neq \emptyset$.

■ Es gibt also b im Durchschnitt mit:

$$\text{dist}(c) \leq \text{dist}(c(a_i), c(a_j)) \leq \text{dist}(c(a_i), b) + \text{dist}(b, c(a_j)) \leq k + k.$$

$$\underbrace{\quad \quad \quad}_{\leq k} \quad \underbrace{\quad \quad \quad}_{\leq k} \quad \rightarrow \quad \quad \quad \hookrightarrow$$

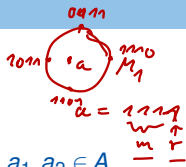
Anzahl Zusatzbits für Fehlerkorrigierende Codes

Satz

Für einen 1-fehlerkorrigierenden Code $c : A \rightarrow \{0, 1\}^{m+r}$ fester Länge über A mit $|A| = 2^m$ gilt: $r \geq 1 + \lceil \log_2 m \rceil$.

Beweis:

- $M_1(a) := \{b \in \{0,1\}^{m+r} : b \text{ entsteht aus } c(a) \text{ durch Kippen von bis zu 1 Bit}\}.$
- Nach Lemma muss gelten: $M_1(a_1) \cap M_1(a_2) = \emptyset$ für alle $a_1, a_2 \in A$,
- es gilt $|M_1(a)| = m+r+1$ für alle $a \in A$.
- Also müssen 2^m überschneidungsfreie Kugeln, jede mit $(m+r+1)$ Elementen, im Raum \mathbb{B}^{m+r} enthalten sein: $2^m(m+r+1) \leq 2^{m+r}$.
- Behauptung: Aus $2^m(m+r+1) \leq 2^{m+r}$ folgt $r \geq 1 + \lfloor \log_2 m \rfloor$.
- Sei hierzu $k := \lfloor \log_2 m \rfloor \Rightarrow 2^k \leq m$.
- $2^m(m+r+1) \leq 2^{m+r} \Leftrightarrow m+r+1 \leq 2^r \Rightarrow 2^k + 1 \leq 2^r \Rightarrow 2^k + 1 \leq 2^r \Rightarrow k < r \Leftrightarrow k+1 \leq r \Leftrightarrow \lfloor \log_2 m \rfloor + 1 \leq r.$



- Wir haben bisher angenommen, dass Fehler auf Kommunikationskanälen auftreten.
- Es gibt auch Fehler in der Hardware selbst.
 - Permanente Fehler (Fertigungsdefekte)
→ Testmethoden (Rechnerarchitektur, Spezialvorlesung „Testen“)
 - Latente Fehler („Beinahe-Defekte“)
→ Stresstest (Spezialvorlesung „Testen“)
 - Transiente Fehler (Störungen während des Betriebs)
→ Fehlertoleranz, Redundanz (Spezialvorlesung „Testen“)
 - Absichtlich herbeigeführte Fehler (Angriffe)
 - Aus Vergleich des Systemverhaltens mit und ohne Fehler auf geschützte Daten schließen (Fault-Based Cryptanalysis). (Seminar)
- Vor allem bei sicherheitskritischen Systemen in neuesten Fertigungstechnologien sind Fehler problematisch.