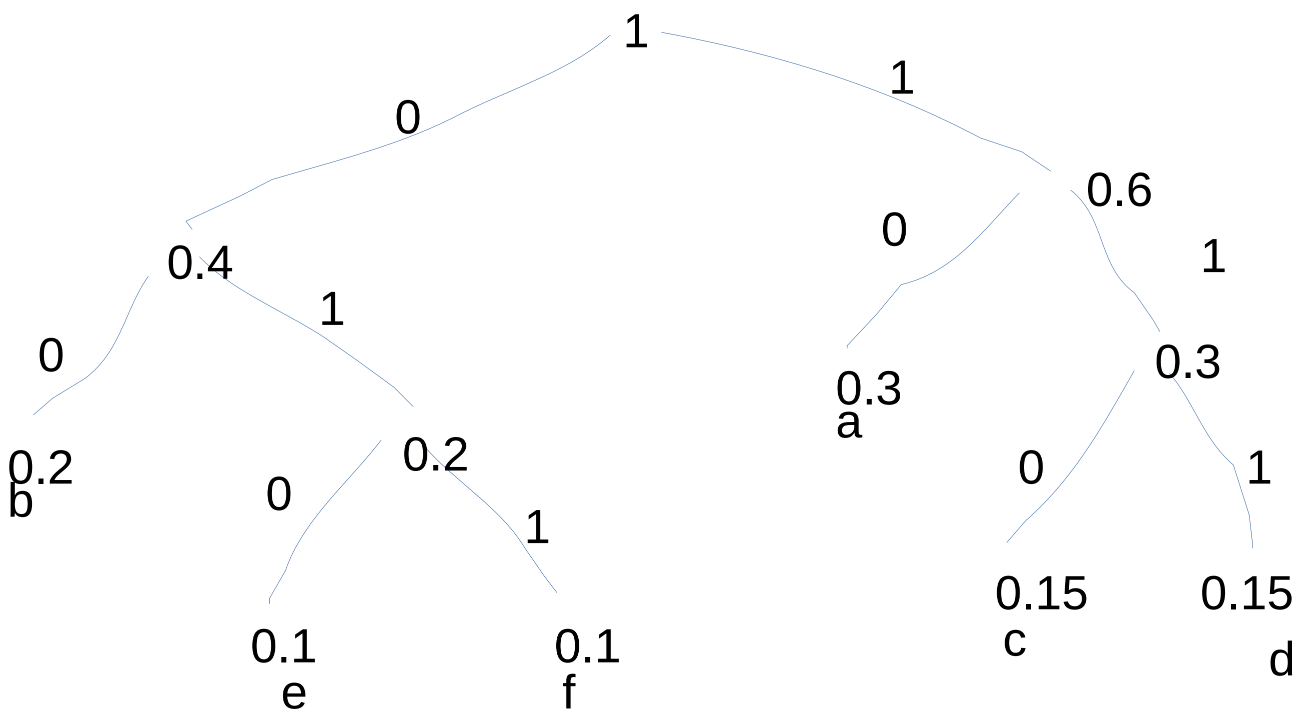
Übung 11
Tim Rumrich
Dienstag 10Uhr

11.1
a

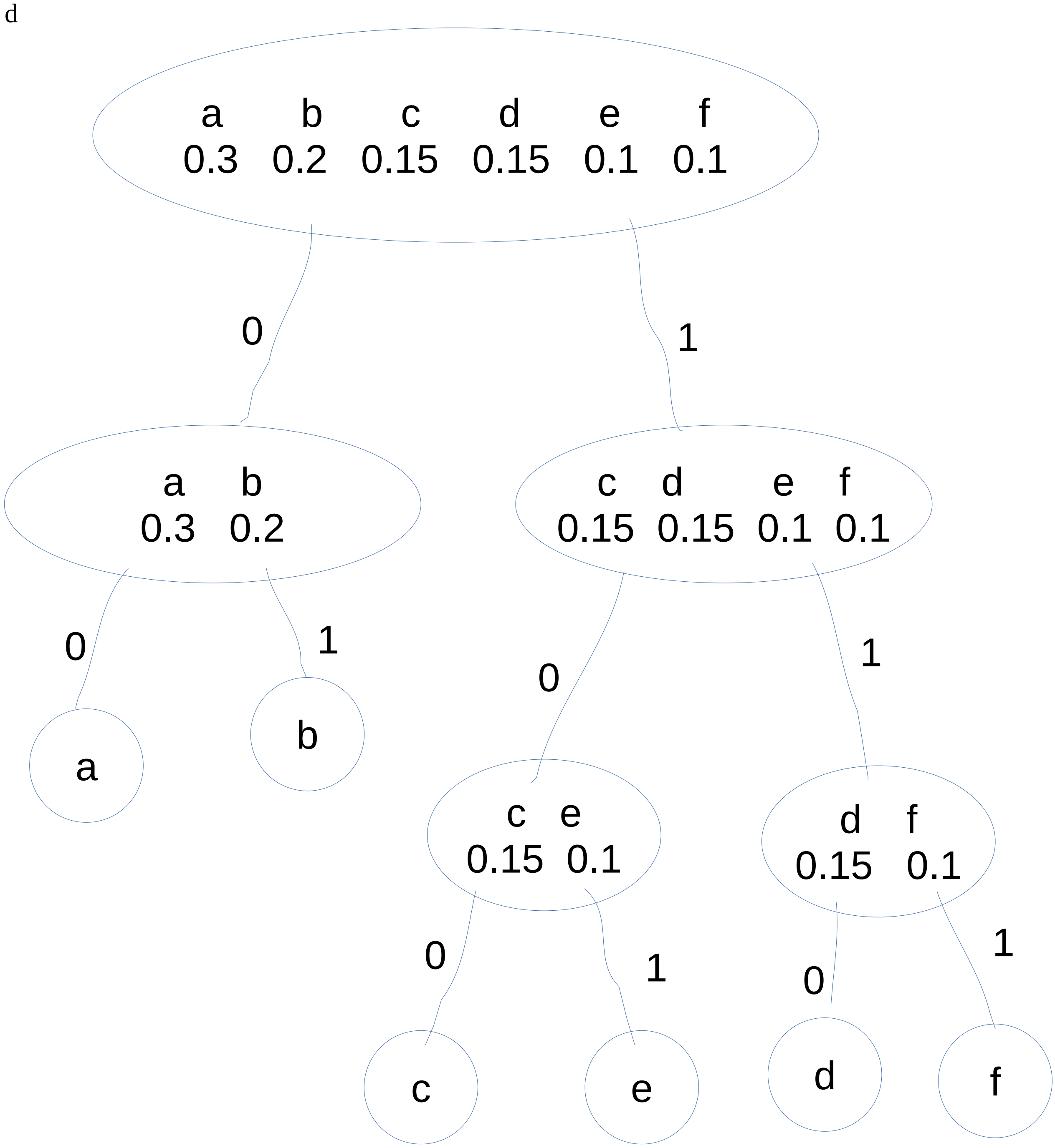


b
3*0.1+3*0.1+2*0.2+2*0.3+3*0.15+3*0.15= 2.5

c

b	a	d	e	f	c	f	a
00	10	111	010	011	110	011	10

d



mittlere wortlänge:
0.3*2+0.2*2+0.15*3*2+0.1*3*2=2.5

b	a	d	e	f	c	f	a
01	00	110	101	111	100	111	00

11.2

a
Code Redundanz ist der Zusatzaufwand im Code der über die reine Darstellung der Daten hinausgeht. Der Hamming Abstand sind die Anzahl stellen in denen sich zwei Codewörter unterscheiden.

B
Um n-Bit Fehler erkennen zu können braucht man einen Hamming Abstand von n+1
=> 1Bit = Hamming Abstand 2
=> 3Bit = Hamming Abstand 4

c
Um n-Bit Fehler korrigieren zu können braucht man einen Hamming Abstand von 2n+1
=> 1Bit = Hamming Abstand von 3
=> 3Bit = Hamming Abstand von 7

11.3
a
weil der Hamming Abstand von 1(Dichter Blockcode) auf 2 gehoben wurde sind jetzt 1-Bit Fehler erkennbar. Für 2 Bit Fehler braucht man aber mindestens einen Abstand von 3, daher sind sie nicht erkennbar.

B

Codewort:	0010010	1111111	1010101	0001000
Paritätsbit:	0	1	0	1

c
00100101 → es könnte ein 1, 3, 5 oder 7 Bit Fehler aufgetreten sein, da die Parität gerade sein sollte aber ungerade ist

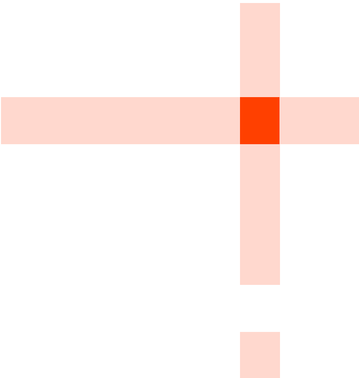
11111111 → es könnte ein 2, 4, 6 oder 8Bit Fehler aufgetreten sein, da die Parität wie erwartet positiv ist

d
wenn nur ein 1-Bit Fehler aufgetreten ist muss das erste Codewort das fehlerhafte sein, da das zweite keinen Fehler aufweist (gerade Parität wird erwartet & wurde empfangen)

11.4
a
3-528-05783-6 3+10+6+32+0+30+49+64+24 mod 11 = 9 → ungültig
3-528-05738-6 3+10+6+32+0+30+49+24+72 mod 11 = 6 → gültig

b
281234554321x
2*1+8*3+1*1+2*3+3*1+4*3+5*1+5*3+4*1+3*3+2*1+1*3 + x=
2 + 24 + 1 + 6 + 3 + 12 + 5 + 15 + 4 + 9 + 2 + 3 + x = 86 + x
86 + x mod 10 = 0 => x = 4

11.5



Geheim

11.6
a)
01001011
P1= 0+1+0+1+1 mod2 = 1
P2=0+0+0+0+1 mod 2 = 1
P3=1+0+0+1 mod2 = 0
P4=1+0+1+1 mod2 = 1

=> Codewort 110010011011

b)
1)
p1= 1+1+0+1+1 mod2=0
p2=1+0+0+0+1 mod2 = 0
p3=1+0+0+1 mod2 = 0
p4= 1+0+1+1 mod2 = 1
=> alles richtig

2)
p1= 0+0+1+0+1 mod2 = 0 ← FEHLER
p2= 0+1+1+1+1 mod2 =0 ← FEHLER
p3= 0+1+1+0 mod2 = 0
p4 = 0+1+1+0 mod2 =0 ← FEHLER
=> Fehlerhaftes Bit ist an stelle 1011 (11), ist korrigierbar.

Fehlerfreies CodeWort : 110001110100

11.7
a)
$2^r \geq m + r + 1$
r = anzahl paritäts Bits
m = Wortlänge

=> für Wortlänge 128
$2^r \geq 128 + r + 1$
r=8 => 8 zusätzliche Bits
+1 Bit zusätzlich für 2-Bit Fehler
=> 9 zusätzliche Bits

b)
128Bit:
9/128 = 7% Zuwachs

64Bit:
8/64 = 12.5% Zuwachs

32Bit:
7/32 = 21% Zuwachs

16Bit:
6/16 = 37.5% Zuwachs

8Bit:
5/8 = 62.5% Zuwachs

11.8
a)
01110101000:1011 = 01100001 CRC = 011

b)
10011010010101:1011=10100111011 → geht auf, nicht fehlerhaft
10010010:1011 = 1010 REST 101 → geht nicht auf, fehlerhaft

c)
Man muss auf das Wort ein Vielfaches von 1011 so addieren, dass nur 3 Stellen geflippt werden, damit man mit der CRC Methode keine Fehler erkennen kann.
Z.B.:
01110101 → echtes Wort
11111001 → fehlerhaftes Wort (3*G(x) addiert)

Für empfangene Codewörter gilt das gleiche, wenn man das empfangen Codewort aus a) nimmt, 01100001011 kann man es mit -1*G(x) addieren und erhält ein Codewort neues Codewort 01100000000. Dieses Codewort ist an 3 Stellen fehlerhaft und trotzdem durch G(x) teilbar.