Fakultät IV

Klausur Informatik 1
26. Februar 2002

Jähnichen / Flik
Lechtchinsky / Neunast

Name: ....

Vorname: ....

Matr.-Nr.: ....

Bearbeitungszeit: 120 Minuten

- Benutzen Sie für die Lösung der Aufgaben nur das mit diesem Deckblatt ausgeteilte Papier. **Lösungen, die auf anderem Papier geschrieben werden, können nicht gewertet werden!**
- Schreiben Sie Ihre Lösungen auf das Aufgabenblatt der jeweiligen Aufgabe. Verwenden Sie auch die Rückseiten. **Schreiben Sie keine Lösungen einer Aufgabe auf ein Blatt, das nicht zu dieser Aufgabe gehört!** Wenn Sie zusätzliche, von uns ausgegebene Blätter verwenden, geben Sie unbedingt an, zu welcher Aufgabe die Lösung gehört!
- Schreiben Sie deutlich! Doppelte, unleserliche oder mehrdeutige Lösungen können nicht gewertet werden!
- Schreiben Sie nur in **blau** oder **schwarz**. Lösungen, die mit Bleistift geschrieben sind, können nicht gewertet werden!
- Bei der Implementation von OPAL-Funktionen brauchen *keine* Importe mit angegeben zu werden, wenn Funktionen aus dem gegebenen Teil der Bibliotheca Opalica (oder aus der Struktur `Nat`, `Real`, `Seq` oder den Strukturen für die Listenfunktionale) oder sonstigen (in den Aufgaben) gegebenen Strukturen benutzt werden!
- Die OPAL-Implementierungen müssen nicht dokumentiert werden.
- Erscheint Ihnen eine Aufgabe mehrdeutig, wenden Sie sich an die Betreuer.
- Tragen Sie jetzt (vor Beginn der Bearbeitungszeit) auf *allen* Blättern Ihren Namen und Ihre Matrikelnummer ein.

Aufgabe	Punkte	erreicht
1	18	
2	15	
3	2	
4	6	
5	5	
6	3	

1. Aufgabe (18 Punkte): Sequenzen

In dieser Aufgabe sollen einige Funktionen zur Verwaltung von Spielergebnissen der Fußball-Bundesliga entwickelt werden. Eine Mannschaft wird durch ihren Namen (Typ `denotation`) repräsentiert. An einem Spiel nehmen eine Heim- und eine Gastmannschaft teil, wobei jede Mannschaft eine bestimmte Anzahl von Toren erzielt. Somit läßt sich das Ergebnis eines Bundesligaspiels durch den folgenden Datentyp beschreiben:

```
DATA match == match (homeTeam   : denotation,  -- Heimmannschaft
                     guestTeam  : denotation,  -- Gastmannschaft
                     homeGoals  : nat,         -- Tore der Heimmannschaft
                     guestGoals : nat)         -- Tore der Gastmannschaft
```

Eine komplette Saison wird durch eine Liste von solchen Spielergebnissen dargestellt.

1.1. Mannschaftsnamen (5 Punkte) Deklarieren und definieren Sie die Funktion `teams`, die die Namen aller Mannschaften bestimmt, die an einer Saison teilgenommen haben. Die Namen dürfen in der Ergebnisliste nicht doppelt vorkommen. Sie können davon ausgehen, daß jede Mannschaft mindestens ein Heimspiel und ein Auswärtsspiel bestritten hat.

Hinweis: Sie können, falls notwending, die Funktion `exist? : (α -> bool) ** seq[α] -> bool` aus der Struktur `Seq` benutzen, die überprüft, ob ein Prädikat für mindestens ein Element einer Liste erfüllt ist.

Lösung:

```
FUN teams : seq[match] -> seq[denotation]
DEF teams(<>) == <>
DEF teams(m :: ms) == LET team == homeTeam(m)
                          ts == teams(ms)
                      IN IF exist?((team = _), ts) THEN ts
                                                   ELSE team :: ts
                         FI
```

1.2. Heimspiele (2 Punkte) Definieren Sie die Funktion `homeMatches`, die die Liste aller Heimspiele einer Mannschaft in einer Saison liefert. Die Signatur der Funktion ist

```
FUN homeMatches : denotation ** seq[match] -> seq[match]
```

Lösung:

```
DEF homeMatches(t, ms) == filter (\\m. homeTeam(m) = t)(ms)
```

1.3. Punkte (6 Punkte) Deklarieren und definieren Sie die Funktion `homePoints`, die die Gesamtanzahl der Punkte berechnet, die eine Mannschaft in ihren Heimspielen in einer Saison erzielt hat. Dabei zählt ein Sieg als 3 Punkte, ein Unentschieden als 1 Punkt und eine Niederlage als 0 Punkte.

Hinweis: Sie können die Funktion `homeMatches` aus Aufgabe 1.2. verwenden, auch wenn Sie diese nicht implementiert haben.

Lösung:

```
FUN homePoints : denotation ** seq[match] -> nat
DEF homePoints(t, ms) ==
  LET hs == homeMatches(t, ms)
      hps == map (\\m . IF homeGoals(m) > guestGoals(m) THEN 3
                        IF homeGoals(m) = guestGoals(m) THEN 1
                        ELSE 0
                        FI)(hs)
  IN
    reduce(+,0)(hps)
```

1.4. Statistiken (5 Punkte) Deklarieren und definieren Sie die Funktion `average`, die das durchschnittliche Ergebnis der Anwendung einer Funktion auf alle Spiele einer Saison berechnet. So soll z.B. `average(homeGoals)` die durchschnittliche Anzahl der von den Heimmannschaften erzielten Tore pro Spiel berechnen. Die übergebene Funktion liefert dabei Ergebnisse vom Typ `nat`, allerdings muß das Ergebnis von `average` natürlich nicht ganzzahlig sein. Sie können die Funktion `asReal : nat -> real` aus der Struktur `NatConv` benutzen, um einen Wert vom Typ `nat` in einen Wert vom Typ `real` umzuwandeln.

Lösung:

```
FUN average : (match -> nat) -> seq[match] -> real
DEF average(f)(ms) == reduce(+,0)(map(asReal o f)(ms)) / asReal(#(ms))
```

2. Aufgabe (15 Punkte): Rekursive Datentypen

In dieser Aufgabe sollen einige Funktionen zur Verwaltung von Turnieren nach dem KO-System erstellt werden. In einem solchen Turnier scheidet der Verlierer eines Spiels aus, der Sieger zieht in die nächste Runde ein (Unentschieden kommen hier nicht vor). Der Gewinner eines Turniers ist die Mannschaft, die kein Spiel verloren hat. Somit kann das Gesamtergebnis eines solchen Turniers durch den folgenden baumartigen Datentyp repräsentiert werden:

```
DATA tourney == team (name       : denotation)
                match(home       : tourney,
                      guest      : tourney,
                      homeGoals  : nat,
                      guestGoals : nat)
```

Ein einfacher Turnierbaum besteht also aus genau einer Mannschaft (`team`), die durch ihren Namen (`name`) repräsentiert wird. Bei einem Turnier mit mehreren Mannschaften wird der Sieger in einem Endspiel (`match`) ermittelt, wobei es auch hier eine Heim- und eine Gastmannschaft gibt, die jeweils eine bestimmte Anzahl von Toren erzielen. Die beiden Mannschaften werden hier jedoch nicht direkt angegeben, sondern wiederum in zwei „Teilturnieren“ (`home` und `guest`) ermittelt - die Sieger nehmen an dem Endspiel teil.

2.1. Induzierte Signatur (2 Punkte) Geben Sie zu dem oben definierten Datentyp `tourney` die induzierte Signatur an und benennen Sie die einzelnen Elemente. Um welche Art von Datentyp handelt es sich bei `tourney`?

Lösung:

```
-- Sorte
SORT tourney

-- Konstruktoren
FUN match : tourney ** tourney ** nat ** nat -> tourney
FUN team  : denotation -> tourney

-- Diskriminatoren
FUN match? : tourney -> bool
FUN team? : tourney -> bool

-- Selektoren
FUN home  : tourney -> tourney
FUN guest : tourney -> tourney
FUN homeGoals  : tourney -> nat
FUN guestGoals : tourney -> nat
FUN name : tourney -> denotation
```

2.2. Gewinner (6 Punkte) Definieren Sie die Funktion `winner`, die den Namen der Siegermannschaft eines Turniers ermittelt. Die Signatur der Funktion ist

```
FUN winner : tourney -> denotation
```

Lösung:

```
DEF winner(team(name)) == name
DEF winner(match(h, g, hg, gg)) == IF hg > gg THEN winner(h)
                                              ELSE winner(g)
                                   FI
```

2.3. Gegner (7 Punkte) Deklarieren und definieren Sie die Funktion `opponents`, die die Namen aller Gegner einer Mannschaft (also die Mannschaften, gegen die diese Mannschaft gespielt hat) in einem Turnier ermittelt.

Hinweis: Sie können die Funktion `winner` aus Aufgabe 2.2. verwenden, auch wenn Sie diese nicht implementiert haben.

Lösung:

```
FUN opponents : denotation ** tourney -> seq[denotation]
DEF opponents(_, team(_)) == <>
DEF opponents(t, match(h, g, _, _)) == IF winner(h) = t
                                         THEN winner(g) :: opponents(t, h)
                                       IF winner(g) = t
                                         THEN winner(h) :: opponents(t, g)
                                       ELSE
                                         opponents(t, h) ++ opponents(t, g)
                                       FI
```

3. Aufgabe (2 Punkte): Codesicherung

Erzeugen Sie für folgenden Datenstrom die CRC-Prüfbits (Generatorpolynom $x^8 + x^2 + x + 1$):

0 0 0 0 0 0 1 0 0 0 0 1 0 0 1 0

Lösung:

$$
\begin{array}{lllllllll}
(x^{17} & +x^{12} & & & +x^{9}) & \multicolumn{4}{l}{:(x^8+x^2+x+1) = x^9+x^4+x^3+x^2} \\
-(x^{17} & & +x^{11} & +x^{10} & +x^{9}) & & & & \\
\cline{1-5}
 & x^{12} & +x^{11} & +x^{10} & & & & & \\
 & -(x^{12} & & & +x^{6} & +x^{5} & +x^{4}) & & \\
\cline{2-7}
 & & x^{11} & +x^{10} & +x^{6} & +x^{5} & +x^{4} & & \\
 & & -(x^{11} & & & +x^{5} & +x^{4} & +x^{3}) & \\
\cline{3-8}
 & & & x^{10} & +x^{6} & & & +x^{3} & \\
 & & & -(x^{10} & & & +x^{4} & +x^{3} & +x^{2}) \\
\cline{4-9}
 & & & & +x^{6} & & +x^{4} & & +x^{2} \\
\cline{5-9}
\end{array}
$$

Die Prüfbits lauten: 01010100.

4. Aufgabe (6 Punkte): Zahlendarstellungen

4.1. (1 Punkte) Stellen Sie CAFE_{16} zur Basis 2 und 8 dar.

Lösung:

$CAFE_{16} = 1100101011111110_2 = 145376_8$

4.2. (1 Punkte) Wandeln Sie -125_{10} in eine 8-Bit-2-Komplementzahl um. Die Rechenschritte müssen erkennbar sein!

Lösung:

$$
\begin{aligned}
125/2 = &\ 62 && \mathit{Rest}\ 1\\
62/2 = &\ 31 && \mathit{Rest}\ 0\\
31/2 = &\ 15 && \mathit{Rest}\ 1\\
15/2 = &\ 7 && \mathit{Rest}\ 1\\
7/2 = &\ 3 && \mathit{Rest}\ 1\\
3/2 = &\ 1 && \mathit{Rest}\ 1\\
1/2 = &\ 0 && \mathit{Rest}\ 1\\
+125_{10} = & && 01111101_2\\
-125_{10} = & && 10000010_2 + 1 = 10000011_2
\end{aligned}
$$

4.3. (2 Punkte) Multiplizieren Sie als vorzeichenlose Zahlen im Dualsystem 0100.1111_2 und 0100.101_2.

Lösung:

0	1	0	0.	1	1	1	1	·	0	1	0	0.	1	0	1
								0	1	0	0	1	1	1	1
						0	1	0	0	1	1	1	1		
			0	1	0	0	1	1	1	1					
			0	1	0	1	1	0.	1	1	0	1	0	1	1

Für die nächsten beiden Teilaufgaben ist das aus der Vorlesung bekannte Gleitpunktformat gegeben: 1-Bit-Vorzeichen, 4-Bit-Exponent, 11-Bit-Mantisse, Bias=7.

4.4. (1 Punkte) Geben Sie für die normalisierte Darstellung den kleinsten und den größten Exponenten als Dezimalzahlen sowie in codierter Form an.

Lösung:

$0001_2 - 7 \ldots 1110_2 - 7 = -6 \ldots + 7$

4.5. (1 Punkte) Geben Sie für die unnormalisierte Darstellung den kleinsten und den größten positiven Zahlenwert in der Form $a \cdot 2^b$ an, a als duale Festpunktzahl und b als Dezimalzahl.

Lösung:

$+0.00000000001_2 \cdot 2^{-6} \ldots + 0.11111111111_2 \cdot 2^{-6}$

5. Aufgabe (5 Punkte): Minimierung boolescher Funktionen

Gegeben sei die unvollständig definierte Funktion f_1 durch die unten abgebildete Wertetabelle. Eine weitere Funktion f_2 sei definiert durch

$$f_2(x_0, x_1, x_2, x_3) = \varnothing f_1(x_0, x_1, x_2, x_3) + \varnothing x_0 \cdot \varnothing x_1 \cdot \varnothing x_2 \cdot \varnothing x_3.$$

Zur Erinnerung:

$$\begin{aligned} a + 0 &= a & (8) \\ a + 1 &= 1 & (14) \end{aligned}$$

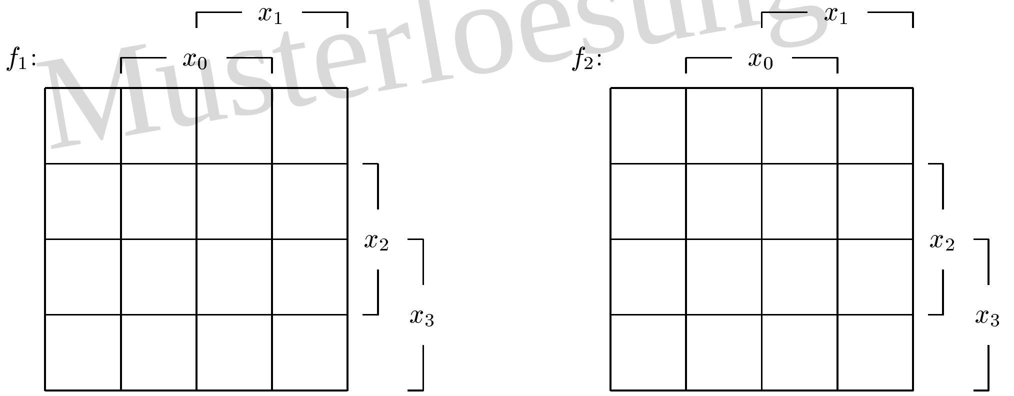
5.1. (1 Punkte) Ergänzen Sie die Wertetabelle um die Funktionswerte von f_2.

x_3	x_2	x_1	x_0	f_1	f_2
0	0	0	0	*	
0	0	0	1	1	
0	0	1	0	1	
0	0	1	1	*	
0	1	0	0	*	
0	1	0	1	0	
0	1	1	0	*	
0	1	1	1	0	
1	0	0	0	*	
1	0	0	1	*	
1	0	1	0	*	
1	0	1	1	1	
1	1	0	0	*	
1	1	0	1	*	
1	1	1	0	0	
1	1	1	1	1	

Lösung:

x_3	x_2	x_1	x_0	f_1	f_2
0	0	0	0	*	1
0	0	0	1	1	0
0	0	1	0	1	0
0	0	1	1	*	*
0	1	0	0	*	*
0	1	0	1	0	1
0	1	1	0	*	*
0	1	1	1	0	1
1	0	0	0	*	*
1	0	0	1	*	*
1	0	1	0	*	*
1	0	1	1	1	0
1	1	0	0	*	*
1	1	0	1	*	*
1	1	1	0	0	1
1	1	1	1	1	0

5.2. (4 Punkte) Bestimmen Sie mit Hilfe der KV-Tafeln eine minimale *konjunktive* Normalform für f_1 und eine minimale *disjunktive* Normalform für f_2. Markieren Sie die zusammengefaßten Bereiche.



Lösung:

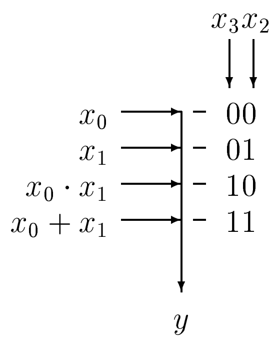
*	1	*	1
*	0	0	*
*	*	1	0
*	*	1	*

1	0	*	0
*	1	1	*
*	*	0	1
*	*	0	*

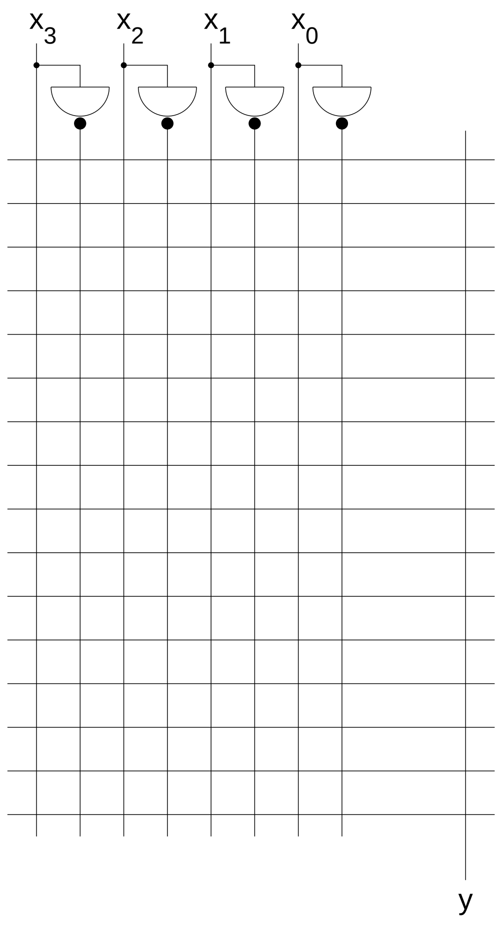
$$
\begin{aligned}
ø f_1 &= x_2 \cdot ø x_3 + ø x_0 \cdot x_3 \\
f_1 &= (ø x_2 + x_3) \cdot (x_0 + ø x_3) \\
f_2 &= x_2 \cdot ø x_3 + ø x_0 \cdot x_3 + ø x_0 \cdot ø x_1
\end{aligned}
$$

6. Aufgabe (3 Punkte): Schaltnetzentwurf

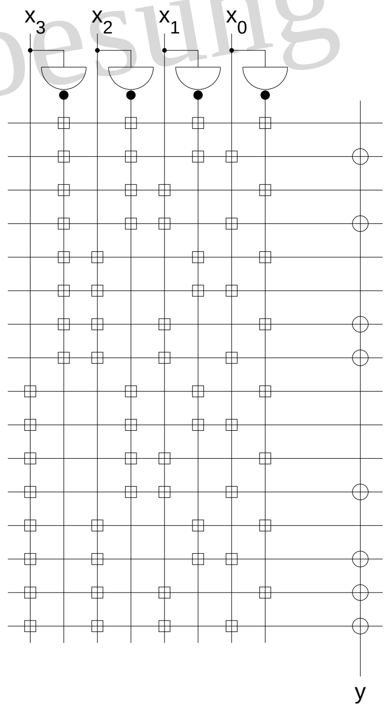
Gegeben ist folgende Multiplexerschaltung:



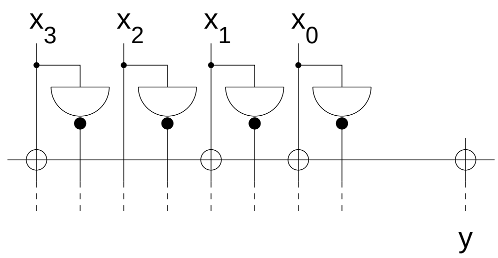
6.1. ROM (2 Punkte) Die Funktion y soll durch einen ROM-Baustein realisiert werden. Tragen Sie dazu in der Und-Matrix und in der Oder-Matrix die erforderlichen Verbindungen ein: Quadrate für vorgegebene, Kreise für programmierbare Verbindungen.



Lösung:



6.2. PLA (1 Punkte) Gegeben ist folgender Ausschnitt einer PLA-Realisierung der Funktion y. Markieren Sie in der obigen ROM-Darstellung jene Zeilen, die den im PLA dargestellten Funktionsterm realisieren.



Lösung:

$x_3 x_1 x_0 = x_3 \o x_2 x_1 x_0 + x_3 x_2 x_1 x_0 = 1011 + 1111$ *d.h. Zeilen 12 und 16.*