Technische Universität München
Fakultät für Informatik
C. Buckl, D. Gmach, K. Schlüter,
Dr. M. Schneider

WS 2005/2006
Lösungsvorschläge zu Blatt 1
28. Oktober 2005

Übungen zu Einführung in die Informatik I

Aufgabe 1 Funktionen in OCaml (Lösungsvorschlag)

a) **Definition von Funktionen:**

(i)
```
# let square x = x *. x;;
val square : float -> float = <fun>
# square 5.;;
- : float = 25.
```

(ii)
```
# let square_root x = sqrt x;;
val square_root : float -> float = <fun>
# square_root 25.;;
- : float = 5.
```

Alternativlösung:

```
# let square_root = sqrt;;
val square_root : float -> float = <fun>
# square_root 25.;;
- : float = 5.
```

(iii)
```
# let square_id x = square_root (square x);;
val square_id : float -> float = <fun>
# square_id 5.;;
- : float = 5.
```

(iv)
```
# let validate_id x = x = square_id(x);;
val validate_id : float -> bool = <fun>
# validate_id 5.;;
- : bool = true
```

b) **Zusammengesetzte Funktionen:**

(i)
```
# let diag_r a b = square_root(square a +. square b);;
val diag_r : float -> float -> float = <fun>
# diag_r 3. 4.;;
- : float = 5.
# diag_r 4. 5.;;
- : float = 6.4031242374328485
```

(ii)
```
# let diag_q a b c = square_root(square (diag_r a b) +. square c);;
val diag_q : float -> float -> float -> float = <fun>
# diag_q 3. 4. 5.;;
- : float = 7.0710678118654755
```

(iii)
```
# let diag_c a = diag_q a a a;;
val diag_c : float -> float = <fun>
# diag_c 5.;;
- : float = 8.6602540378443873
# diag_q 5. 5. 5.;;
- : float = 8.6602540378443873
```

c) **Partielle Funktionsanwendung:**

```
# let diag_q1 = diag_q 1.;;
val diag_q1 : float -> float -> float = <fun>
# diag_q1 3. 4.;;
- : float = 5.0990195135927845
# diag_q 1. 3. 4.;;
- : float = 5.0990195135927845
```

Alternativlösung:

```
# let diag_q1 b c = diag_q 1. b c;;
val diag_q1 : float -> float -> float = <fun>
# diag_q1 3. 4.;;
- : float = 5.0990195135927845
```

Aufgabe 2 Boolesche Algebra (Lösungsvorschlag)

a) Wahrheitstafeln für `xor`, `nand` und `nor`:

x	y	xor (x,y)
false	false	false
false	true	true
true	false	true
true	true	false

x	y	nand (x,y)
false	false	true
false	true	true
true	false	true
true	true	false

x	y	nor (x,y)
false	false	true
false	true	false
true	false	false
true	true	false

Die Funktion $xor(x,y)$ wird über $(\neg x \wedge y) \vee (x \wedge \neg y)$ definiert.

$nor(x,y)$ kann auch als $\neg(x \vee y)$ geschrieben werden.

Da $nand(x,y)$ über $\neg(x \wedge y)$ definiert werden kann, kann eine Negation durch $\neg(x \wedge x) = \neg x$ und damit $nand(x,x)$ definiert werden:

x	nand (x,x)
false	true
true	false

Diese Berechnung des Komplements kann nun verwendet werden, um $x \wedge y = \neg\neg(x \wedge y)$ zu berechnen: $\neg(x \wedge y)$ ist genau die Definition von $nand(x,y)$ und die Negation davon, wäre also nach obiger Herleitung $nand(nand(x,y), nand(x,y))$.

Damit wurde also sowohl das Komplement als auch die Und–Verknüpfung durch die Nand-Funktion ausgedrückt. Da alle booleschen Funktionen (wie in der Vorlesung erwähnt) durch Komplement und Und–Verknüpfung dargestellt werden können, genügt also alleine die Nand–Funktion ebenfalls, um alle booleschen Funktionen darzustellen. Dies hat besondere Bedeutung für Schaltfunktionen (siehe TGI), da nur ein Typ von Bausteinen ausreicht, um Schaltungen zu realisieren.

b)
- Die Aussage enthält folgende Teilaussagen:

  A Das Studium der Informatik verläuft erfolgreich.

  B Der Student ist fähig zu einer mathematisch formalen Arbeitsweise.

  C Der Student ist fähig zu einer anwendungsbezogenen praktischen Arbeitsweise.

  Es ergibt sich der Term: $A \Rightarrow B \wedge C$

- Die Aussage enthält folgende Teilaussagen:

  A Die Fachprüfungen der Diplom-Hauptprüfung können alle vor der Anfertigung der Diplomarbeit absolviert werden.

  B Die Fachprüfungen der Diplom-Hauptprüfung können alle nach der Anfertigung der Diplomarbeit absolviert werden.

  C Es sind die im § 27 Abs. 7 und 8 der Diplomprüfung genannten Fristen einzuhalten

  Es ergibt sich der Term: $((\neg A \wedge B) \vee (A \wedge \neg B)) \wedge C$

c) (i) Für die angegebene Implikation gilt:

$$\begin{array}{ll} & (\neg A \wedge \neg B) \Rightarrow \neg C \\ \Leftrightarrow & \neg(\neg A \wedge \neg B) \vee \neg C \\ \Leftrightarrow & \neg\neg A \vee \neg\neg B \vee \neg C \\ \Leftrightarrow & A \vee B \vee \neg C \end{array}$$

Somit ergibt sich die Wahrheitstabelle:

A	B	C	$(\neg A \wedge \neg B) \Rightarrow \neg C$
false	false	false	true
false	false	true	false
false	true	false	true
false	true	true	true
true	false	false	true
true	false	true	true
true	true	false	true
true	true	true	true

(ii) Der angegebene Satz besteht aus 7 verschiedenen Teilaussagen:

A Der Informatiker wird eingestellt.

B Er hat ein Universitäts-Diplom.

C Er hat ein Fachhochschul-Diplom.

D Er hat sein Studium in weniger als 10 Semestern beendet.

E Er ist älter als 30 Jahre.

F Er ist promoviert.

G Er kann einen Auslandsaufenthalt von mindestens 5 Jahren nachweisen.

Als logische Formel ergibt sich:
$(((B \wedge \neg C) \vee (\neg B \wedge C)) \wedge ((D \wedge \neg E) \vee F \vee G)) \Rightarrow A$

Bemerkung: Die XOR–Verknüpfung zwischen **B** und **C** entspricht der Teilaussage „*Er hat genau ein Diplom*“.

(iii) Die Anweisung zur Behandlung des Unterschleifs besteht aus folgenden Teilaussagen:

A Ein Student bedient sich bei der Anfertigung einer zu benotenden schriftlichen Arbeit unerlaubter Hilfsmittel.

B Ein Student bedient sich bei der Anfertigung einer zu benotenden praktischen Arbeit unerlaubter Hilfsmittel.

C Ein Student stellt derartige Hilfsmittel zur Verfügung.

D Die Arbeit wird abgenommen.

E Die Arbeit gilt als nicht bestanden.

Unter Verwendung des Aussagenkalküls ergibt sich die Formel: $(A \vee B \vee C) \Rightarrow (D \wedge E)$

d) Für den Ausdruck $(a \wedge (a \Rightarrow b)) \Rightarrow b$ soll geprüft werden, ob er allgemeingültig ist:

(i) Erstellen einer Wahrheitstafel: Zu prüfen sind $2^2 = 4$ Belegungen:

a	b	$a \Rightarrow b$	$a \wedge (a \Rightarrow b)$	$(a \wedge (a \Rightarrow b)) \Rightarrow b$
false	false	true	false	true
false	true	true	false	true
true	false	false	false	true
true	true	true	true	true

Der Ausdruck ist also für jede Belegung wahr und damit allgemeingültig bzw. eine Tautologie.

(ii) Prüfung nach dem Verfahren, das auf dem Übungsblatt beschrieben ist: Die äußerste Operation ist eine Implikation. Um nun für den gesamten Term den Wahrheitswert $false$ zu erhalten, müsste der erste Operand $a \wedge (a \Rightarrow b)$ den Wert $true$ haben und b die Belegung $\beta(b) = false$ haben:

$$\overbrace{\underbrace{(a \wedge (a \Rightarrow b))}_{true}}^{false} \Rightarrow \underbrace{b}_{false}$$

Betrachtet man nun den Teilausdruck $(a \wedge (a \Rightarrow b))$ weiter, so kann dieser nur dann $true$ sein, wenn beide Teilausdrücke der Und-Operation $true$ als Wert haben:

$$(\underbrace{a}_{true} \wedge \overbrace{\underbrace{(a \Rightarrow b)}_{true}}^{true}) \Rightarrow b$$

Damit ergibt sich also für den linken Teilausdruck a, dass $\beta(a) = true$, was wir in den anderen Teilausdruck $(a \Rightarrow b)$ einsetzen können. Unsere Belegung $\beta(a) = true, \beta(b) = false$ ergibt für diesen Teilausdruck allerdings nicht den Wert $true$, den wir erzielen müssen (dazu bräuchten wir mit $\beta(a) = true$ auch $\beta(b) = true$), um für den gesamten Ausdruck $(a \wedge (a \Rightarrow b)) \Rightarrow b$ den Wert $false$ zu erzielen. Wir finden keine Belegung, also ist der Ausdruck allgemeingültig und eine Tautologie.

(iii) Für den Ausdruck $(a \wedge (a \Rightarrow b)) \Rightarrow b$ soll geprüft werden, ob er allgemeingültig ist:

i. Erstellen einer Wahrheitstafel: Zu prüfen sind $2^2 = 4$ Belegungen:

a	b	$a \Rightarrow b$	$a \wedge (a \Rightarrow b)$	$(a \wedge (a \Rightarrow b)) \Rightarrow b$
false	false	true	false	true
false	true	true	false	true
true	false	false	false	true
true	true	true	true	true

Der Ausdruck ist also für jede Belegung wahr und damit allgemeingültig bzw. eine Tautologie.

ii. Prüfung nach dem Verfahren, das auf dem Übungsblatt beschrieben ist: Die äußerste Operation ist eine Implikation. Um nun für den gesamten Term den Wahrheitswert $false$ zu erhalten, müsste der erste Operand $a \wedge (a \Rightarrow b)$ den Wert $true$ haben und b die Belegung $\beta(b) = false$ haben:

$$\overbrace{\underbrace{(a \wedge (a \Rightarrow b))}_{true} \Rightarrow \underbrace{b}_{false}}^{false}$$

Betrachtet man nun den Teilausdruck $(a \wedge (a \Rightarrow b))$ weiter, so kann dieser nur dann *true* sein, wenn beide Teilausdrücke der Und-Operation *true* als Wert haben:

$$(\overbrace{\underbrace{a}_{true} \wedge \underbrace{(a \Rightarrow b)}_{true}}^{true}) \Rightarrow b$$

Damit ergibt sich also für den linken Teilausdruck a, dass $\beta(a) = true$, was wir in den anderen Teilausdruck $(a \Rightarrow b)$ einsetzen können. Unsere Belegung $\beta(a) = true, \beta(b) = false$ ergibt für diesen Teilausdruck allerdings nicht den Wert *true*, den wir erzielen müssen (dazu bräuchten wir mit $\beta(a) = true$ auch $\beta(b) = true$), um für den gesamten Ausdruck $(a \wedge (a \Rightarrow b)) \Rightarrow b$ den Wert $false$ zu erzielen. Wir finden keine Belegung, also ist der Ausdruck allgemeingültig und eine Tautologie.

(iv) Für den Ausdruck müssten wir für 6 boolesche Variablen a, w, p, i, b, e $2^6 = 64$ Belegungen mit einer Wahrheitstabelle überprüfen. Nach dem anderen Verfahren gehen wir folgendermaßen vor:

Um für den gesamten Ausdruck $false$ zu erhalten, müssen wir wierderum den linken Teilausdruck der Implikation zu *true* machen, den rechten zu $false$:

$$\overbrace{\underbrace{((a \wedge w) \Rightarrow p) \wedge (\neg a \Rightarrow i) \wedge (\neg w \Rightarrow b) \wedge (\neg p) \wedge (e \Rightarrow (\neg i \wedge \neg b))}_{true} \Rightarrow \underbrace{\neg e}_{false}}^{false}$$

Damit erhalten wir also sofort $\beta(e) = true$, was wir in den restlichen Ausdruck einsetzen können. Der linke Teilausdruck der Implikation ist nur dann wahr, wenn jeder mit der Und–Operation verknüpfte Teilausdruck wahr ist:

$$\overbrace{\underbrace{((a \wedge w) \Rightarrow p)}_{true} \wedge \underbrace{(\neg a \Rightarrow i)}_{true} \wedge \underbrace{(\neg w \Rightarrow b)}_{true} \wedge \underbrace{(\neg p)}_{true} \wedge \underbrace{(e \Rightarrow (\neg i \wedge \neg b))}_{true}}^{true} \Rightarrow \neg e$$

Es ergibt sich $\beta(p) = false$ und wegen $\beta(e) = true$ muss auch $(\neg i \wedge \neg b)$ wahr sein, damit $(e \Rightarrow (\neg i \wedge \neg b))$ wahr ist:

$$((a \wedge w) \Rightarrow p) \wedge (\neg a \Rightarrow i) \wedge (\neg w \Rightarrow b) \wedge (\neg p) \wedge \overbrace{(e \Rightarrow \underbrace{(\neg i \wedge \neg b)}_{true})}^{true} \Rightarrow \neg e$$

Es folgt also $\beta(i) = false, \beta(b) = false$:

$$((a \wedge w) \Rightarrow p) \wedge (\neg a \Rightarrow i) \wedge (\neg w \Rightarrow b) \wedge (\neg p) \wedge (e \Rightarrow$$
$$\overbrace{(\underbrace{\neg i}_{true} \wedge \underbrace{\neg b}_{true})}^{true}) \Rightarrow \neg e$$

Mit $\beta(p) = false, \beta(i) = false, \beta(b) = false$ ergibt sich sofort folgendes:

$$\overbrace{(\underbrace{(a \wedge w)}_{false} \Rightarrow p)}^{true} \wedge \overbrace{(\underbrace{\neg a}_{false} \Rightarrow i)}^{true} \wedge \overbrace{(\underbrace{\neg w}_{false} \Rightarrow b)}^{true} \wedge (\neg p) \wedge (e \Rightarrow (\neg i \wedge \neg b)) \Rightarrow \neg e$$

Es soll also im ersten Ausdruck $a \wedge w = false$ gelten, im zweiten $a = true$ und im dritten $w = true$, wofür wir keine Belegung finden können. Insgesamt finden wir also keine Belegung, so dass der gesamte Ausdruck den Wert $false$ erhält, also ist er allgemeingültig.

Aufgabe 3 Osterberechnung (Lösungsvorschlag)

Eine mögliche Implementierung könnte folgendermaßen aussehen:

```
type monat = Januar | Februar | März | April | Mai
             | Juni | Juli | August | September
             | Oktober | November | Dezember;;

type datum = Datum of (int * monat * int);;

let tabelle_m j = match j with
| j when (j > 1582 && j < 1700) -> 22
| j when (j > 1699 && j < 1900) -> 23
| j when (j > 1899 && j < 2200) -> 24
| j when (j > 2199 && j < 2300) -> 25
| _ -> failwith "Falsche Eingabe";;

let tabelle_n j = match j with
| j when (j > 1582 && j < 1700) -> 2
| j when (j > 1699 && j < 1800) -> 3
| j when (j > 1799 && j < 1900) -> 4
| j when (j > 1899 && j < 2100) -> 5
| j when (j > 2099 && j < 2200) -> 6
| j when (j > 2199 && j < 2300) -> 0
| _ -> failwith "Falsche Eingabe";;

let d j = (19 *(j mod 19) + tabelle_m j) mod 30;;

let e j = (2 * (j mod 4) + 4 * ( j mod 7) + 6 * (d j) + tabelle_n j) mod 7;;

let ostersonntag j = let d_plus_e = d j + e j
in if ((22 + d_plus_e)<32) then Datum(22 + d_plus_e, März, j)
else Datum (d_plus_e - 9, April, j);;
```