Informatik A, WS 2016/17 — Nachklausur

Dienstag, 28. März 2017 – Musterlösungen

1. Aussagenlogik, 10 Punkte

   (a) Umformung, 6 Punkte

   Beweisen Sie folgende Äquivalenzen, entweder durch Anwendung von Umformungsregeln oder durch Vergleichen der Wahrheitstafeln:

$$(p \wedge s) \Rightarrow q \;\equiv\; (p \Rightarrow q) \vee (s \Rightarrow q) \tag{1}$$

Beweis durch Umformen

$$\begin{aligned}
\text{linke Seite} &\equiv (p \wedge s) \Rightarrow q \\
&\equiv (\neg(p \wedge s)) \vee q && \text{(Elimination von } \Rightarrow) \\
&\equiv (\neg p \vee \neg s) \vee q && \text{(de Morgan)} \\
&\equiv \neg p \vee \neg s \vee q && \text{(Assoziativität von } \vee) \\
\text{rechte Seite} &\equiv (p \Rightarrow q) \vee (s \Rightarrow q) \\
&\equiv ((\neg p) \vee q) \vee ((\neg s) \vee q) && \text{(Elimination von } \Rightarrow) \\
&\equiv \neg p \vee \neg s \vee (q \vee q) && \text{(Assoziativität, Kommutativität von } \vee) \\
&\equiv \neg p \vee \neg s \vee q && \text{(Itempotenz von } \vee)
\end{aligned}$$

$$x \Leftrightarrow y \;\equiv\; (\neg x) \Leftrightarrow (\neg y) \tag{2}$$

Beweis durch Umformen:

$$\begin{aligned}
\text{linke Seite} &\equiv x \Leftrightarrow y \\
&\equiv (x \Rightarrow y) \wedge (y \Rightarrow x) && \text{(Elimination von } \Leftrightarrow) \\
&\equiv (\neg x \vee y) \wedge (\neg y \vee x) && \text{(Elimination von } \Rightarrow) \\
\text{rechte Seite} &\equiv (\neg x) \Leftrightarrow (\neg y) \\
&\equiv (\neg x \Rightarrow \neg y) \wedge (\neg y \Rightarrow \neg x) && \text{(Elimination von } \Leftrightarrow) \\
&\equiv (\neg(\neg x) \vee \neg y) \wedge (\neg(\neg y) \vee \neg x) && \text{(Elimination von } \Rightarrow) \\
&\equiv (x \vee \neg y) \wedge (y \vee \neg x) && \text{(doppelte Negation)} \\
&\equiv (y \vee \neg x) \wedge (x \vee \neg y) && \text{(Kommutativität von } \wedge) \\
&\equiv (\neg x \vee y) \wedge (\neg y \vee x) && \text{(Kommutativität von } \vee)
\end{aligned}$$

Einfacher geht es mit der Wahrheitstafel.

   (b) NOR, 4 Punkte

   Der Junktor $\bar{\vee}$ (weder-noch, engl. „nor“) ist definiert als

$$a \mathbin{\bar{\vee}} b \;\equiv\; \neg(a \vee b). \tag{3}$$

Ist der Junktor $\bar{\vee}$ kommutativ?

JA. Beweis von $a \mathbin{\bar{\vee}} b \;\equiv\; b \mathbin{\bar{\vee}} a$ durch Umformung mit Hilfe der Definition (3) und Ausnützen der Kommutativität von $\vee$: (oder mit der Wahrheitstafel)

$$a \mathbin{\bar{\vee}} b \overset{def}{\equiv} \neg(a \vee b) \equiv \neg(b \vee a) \overset{def}{\equiv} b \mathbin{\bar{\vee}} a$$

Ist er assoziativ?

NEIN. $(0 \mathbin{\bar{\vee}} 0) \mathbin{\bar{\vee}} 1 = 1 \mathbin{\bar{\vee}} 1 = 0$, aber $0 \mathbin{\bar{\vee}} (0 \mathbin{\bar{\vee}} 1) = 0 \mathbin{\bar{\vee}} 0 = 1$.

Anderes Gegenbeispiel: $(0 \mathbin{\bar{\vee}} 1) \mathbin{\bar{\vee}} 1 = 0 \mathbin{\bar{\vee}} 1 = 0$, aber $0 \mathbin{\bar{\vee}} (1 \mathbin{\bar{\vee}} 1) = 0 \mathbin{\bar{\vee}} 0 = 1$.

Geben Sie jeweils einen Beweis oder ein Gegenbeispiel.

2. Induktion, 10 Punkte

(a) (4 Punkte) Bestimmen Sie alle Paare (k, l) von ganzen Zahlen, für die die folgende Gleichung gültig ist:

```
(drop k) . (drop l) = drop (k+l)
```

(Die Möglichkeit des Überlaufs bei der Berechnung von `k+l` können Sie außer Acht lassen.) *Geben Sie eine Begründung* für die Fälle, wo die Gleichung nicht gilt.

Die Gleichung gilt genau dann, wenn $kl \geq 0$ ist. Wenn zum Beispiel $k > 0$ und $l < 0$ ist, dann hat `drop l` keine Wirkung. Auf der linken Seite werden also insgesamt die ersten k Elemente entfernt, auf der rechten Seite werden $k + l$ Elemente entfernt, aber $k + l$ ist kleiner als k (möglicherweise sogar negativ). Für genügend lange Listen ist das Ergebnis also verschieden. (Zum Beispiel kommt für `[1..k]` auf der linken Seite die leere Liste heraus, auf der rechten Seite nicht.)

Der Fall $k < 0$ und $l > 0$ ist genauso.

(b) (6 Punkte) Für die Fälle, wo die Gleichung gilt, beweisen Sie sie unter Verwendung folgender Definitionen:

```
drop :: Int -> [a] -> [a]
drop n xs  | n <= 0 =  xs                  -- drop.0
drop _ []           =  []                  -- drop.1
drop n (x:xs)       =  drop (n-1) xs       -- drop.2

(.) :: (b -> c) -> (a -> b) -> (a -> c)
(f . g) x = f (g x)                        -- punkt.0
```

Begründen Sie jeden Schritt, indem Sie angeben, welche Definitionsgleichung (drop.0 oder drop.1 oder drop.2 oder punkt.0) oder welche anderen Eigenschaften sie verwenden.

Zu zeigen ist

```
((drop k) . (drop l)) xs = drop (k+l) xs
```

für alle Listen `xs`. Wegen (punkt.0) ist das äquivalent zu

```
(drop k) (drop l xs) = drop (k+l) xs
```

oder ohne Klammern:

```
drop k (drop l xs) = drop (k+l) xs,
```
(4)

für alle Listen `xs`.

Lemma 1. `drop n [] = []` für alle $n \in \mathbb{Z}$.

Beweis. (drop.1) im Fall $n > 0$, (drop.0) für $n \leq 0$.

Wir unterscheiden die beiden Fälle $k, l \leq 0$ und $k, l \geq 0$, die sich teilweise überlappen.
Fall 1: $k \leq 0$ und $l \leq 0$. Dann ist auch $k + l \leq 0$, und gemäß (drop.0) sind alle drei beteiligten Funktionen die Identität:

$$
\begin{aligned}
\text{linke Seite} &= \texttt{drop k (drop l xs)} \\
&= \texttt{drop k xs} && \text{(drop.0)} \\
&= \texttt{xs} && \text{(drop.0)} \\
\text{rechte Seite} &= \texttt{drop (k+l) xs} \\
&= \texttt{xs} && \text{(drop.0)}
\end{aligned}
$$

Fall 2: $k \geq 0$ und $l \geq 0$. Dann ist auch $k + l \geq 0$.
Möglichkeit I: Beweis durch vollständige Induktion nach l.
Induktionsbasis ($l = 0$):

$$\texttt{(drop k) (drop 0 xs) = drop (k+0) xs} \tag{5}$$

für alle Listen `xs`.

$$
\begin{aligned}
\text{linke Seite} &= \texttt{drop k (drop 0 xs)} \\
&= \texttt{drop k xs} && \text{(drop.0)} \\
&= \texttt{drop (k+0) xs} = \text{rechte Seite} && \text{(Arithmetik)}
\end{aligned}
$$

Induktionsschritt von l auf $l + 1$: Wir nehmen (4) für alle Listen als Induktionannahme an und wollen folgende Induktionsbehauptung zeigen:

$$\texttt{drop k (drop (l+1) xs) = drop (k+(l+1)) xs}$$

Fall 2a: `xs = []`. Ergibt sich direkt ohne Benützung der Induktionsannahme:

$$
\begin{aligned}
\text{linke Seite} &= \texttt{drop k (drop (l+1) xs)} \\
&= \texttt{drop k (drop (l+1) [])} \\
&= \texttt{drop k []} && \text{(Lemma 1)} \\
&= \texttt{[]} && \text{(Lemma 1)} \\
\text{rechte Seite} &= \texttt{drop (k+l+1) xs} \\
&= \texttt{drop (k+l+1) []} \\
&= \texttt{[]} && \text{(Lemma 1)}
\end{aligned}
$$

Fall 2b: $\texttt{xs} \neq \texttt{[]}$, also `xs = y:ys`

$$
\begin{aligned}
\text{linke Seite} &= \texttt{drop k (drop (l+1) (y:ys)} \\
&= \texttt{drop k (drop l ys)} && (\text{drop.2, } l + 1 > 0) \\
&= \texttt{drop (k+l) ys} && \text{(Induktionsannahme)} \\
\text{rechte Seite} &= \texttt{drop (k+l+1) (y:ys)} \\
&= \texttt{drop (k+l) ys} && (\text{drop.2, } k + l + 1 > 0)
\end{aligned}
$$

Möglichkeit II: Beweis durch strukturelle Induktion nach `xs`. (Der Beweis erstreckt sich dann allerdings nur auf die endlichen Listen.)
Induktionsbasis `xs = []`. Ähnlich wie Fall 2a oben:

$$\begin{aligned}
\text{linke Seite} &= \texttt{drop k (drop l xs)} \\
&= \texttt{drop k (drop l [])} \\
&= \texttt{drop k []} && \text{(Lemma 1)} \\
&= \texttt{[]} && \text{(Lemma 1)} \\
\text{rechte Seite} &= \texttt{drop (k+l) xs} \\
&= \texttt{drop (k+l) []} \\
&= \texttt{[]} && \text{(Lemma 1)}
\end{aligned}$$

Induktionsschritt von `xs` auf `x:xs`. Wir nehmen als Induktionannahme an, dass (4) für alle $k, l \geq 0$ und die Liste `xs` gilt, und wollen folgende Induktionsbehauptung zeigen:

`drop k (drop l (x:xs)) = drop (k+l) (x:xs),`

für alle $k, l \geq 0$.
Fall 2.A. $l = 0$. Siehe oben die Induktionsbasis (5) von Möglichkeit I.
Fall 2.B. $l > 0$. Dann ist auch $k + l > 0$. Der Beweis ist ähnlich wie im Fall 2b:

$$\begin{aligned}
\text{linke Seite} &= \texttt{drop k (drop l (x:xs))} \\
&= \texttt{drop k (drop (l-1) xs)} && (\text{drop.2}, l > 0) \\
&= \texttt{drop (k+l-1) xs} && \text{(Induktionsannahme)} \\
\text{rechte Seite} &= \texttt{drop (k+l) (x:xs)} \\
&= \texttt{drop (k+l-1) xs} && (\text{drop.2}, k + 1 > 0)
\end{aligned}$$

---

3. Programmieren, 10 Punkte.

   Siehe `http://www.inf.fu-berlin.de/lehre/WS16/INFA/wieVieleGleiche.hs`

4. Entwurf eines Schaltnetzes, 10 Punkte

   Gesucht ist ein Schaltnetz mit drei Eingängen $a, b, c \in \{0, 1\}$ und zwei Ausgängen $u, v \in \{0, 1\}$.

   Der Ausgang u soll gleich a sein, falls $b = 0$ ist; andernfalls soll er gleich c sein.

   Wenn alle drei Eingänge den gleichen Wert haben, soll der Ausgang v gleich 0 sein; andernfalls soll v gleich b sein.

   (a) (4 Punkte) Beschreiben Sie die Ausgänge u und v durch logische Ausdrücke (nur mit Junktoren, nicht mit if-then-else-Ausdrücken).

$u = (\neg b \wedge a) \vee (b \wedge c)$ (Multiplexer mit b als Steuersignal)
$v = \neg\big((a \Leftrightarrow b) \wedge (b \Leftrightarrow c)\big) \wedge b = \neg\big((a \wedge b \wedge c) \vee (\bar{a} \wedge \bar{b} \wedge \bar{c})\big) \wedge b$
$v = b \wedge \neg(a \wedge c)$
Es gibt auch viele andere Möglichkeiten, die man zum Beispiel aus der konjunktiven oder der disjunktiven Normalform gewinnen kann.

   (b) (6 Punkte) Konstruieren ein Schaltnetz und zeichnen Sie es.
   Verwenden Sie dabei die DIN-Symbole für die Gatter. Gatter mit mehr als zwei Eingängen sind nicht erlaubt.
   Schreiben Sie zum Ausgang jedes Gatters (außer zu u und v) *einen Booleschen Ausdruck* in den Eingabevariablen, der den Wert charakterisiert.

Siehe `http://www.inf.fu-berlin.de/lehre/WS16/INFA/Schaltnetz-4b-N.jpg`