



Lösung zu Aufgabe 1

- a) O.B.d.A. kann angenommen werden, daß F in bereinigter Pränexform vorliegt (zur Erinnerung $\text{BPF}(F) \equiv F$). Der Beweis erfolgt durch Induktion nach der Anzahl n der Quantoren. Dies entspricht der Elimination der Quantoren von außen nach innen.

I.A.: $n = 0$

Enthält F keine Quantoren, so sind $\mathcal{A}(F)$ und $\mathcal{A}'(F)$ identisch.

I.S.: 1. Fall: Der äußerste Quantor ist ein \forall

$F = \forall x F'$, wobei die Behauptung für F' schon gezeigt sei. Dann gilt nach der Definition der Semantik von \forall :

$$\mathcal{A}(F) = \begin{cases} W, & \text{falls für alle } u \in U_{\mathcal{A}} : \mathcal{A}(F'_{[x/u]}) = W \\ F, & \text{sonst} \end{cases}$$
$$\mathcal{A}'(F) = \begin{cases} W, & \text{falls für alle } u \in U_{\mathcal{A}'} : \mathcal{A}'(F'_{[x/u]}) = W \\ F, & \text{sonst} \end{cases}$$

Dann kann die Gültigkeit von $\forall x F'$ auf: für alle $u \in U_{\mathcal{A}}$ gilt

$$\mathcal{A}(F'_{[x/u]}) = W \text{ bzw.}$$

$$\mathcal{A}'(F'_{[x/u]}) = W$$

zurückgeführt werden. Bezgl. F' aber sind $\mathcal{A}(F)$ und $\mathcal{A}'(F)$ auf allen frei vorkommenden Variablen identisch und erfüllen somit die Voraussetzung für die Anwendung der Induktionshypothese.

2. Fall: Der äußerste Quantor ist ein \exists :

analog zum 1. Fall.

- b) Die Kongruenz gilt nicht. Beweis durch Angabe einer Struktur, die Modell von F aber kein Modell von G ist: $\mathcal{A} = (U, I)$, wobei $U = \{\text{Kopf}, \text{Zahl}\}$ und $I(P) = \{\text{Kopf}\}$, $I(Q) = \{\text{Zahl}\}$.

Es gilt zwar, daß bei Münzwürfen immer entweder Kopf oder Zahl fällt, aber keinesfalls, daß immer Kopf oder immer Zahl fällt.

Lösung zu Aufgabe 2

- a) Angenommen, Clemens besucht die Party:

1. Clemens besucht die Party \Rightarrow Albert besucht die Party (Bed. 4)
2. Da Albert zur Party geht \Rightarrow Bob besucht die Party (Bed. 1)
3. *Widerspruch*: Wer beaufsichtigt das Baby des Nachbarn? (Bed. 2)

Angenommen, Clemens geht nicht zur Party:

1. Clemens geht nicht zur Party \Rightarrow Bob muß zur Party (Bed. 2)
2. Clemens geht nicht zur Party \Rightarrow Albert muß zur Party (Bed. 3)
3. Albert geht zur Party \Rightarrow Bob geht auch hin (Bed. 1)
4. Auch Bed. 4 gilt, da Albert die Party besucht.

Es besuchen also Bob und Albert die Party.

- b) Identifiziere die Tage Montag mit 1, Dienstag mit 2, ..., Freitag mit 5 und führe eine vollständig Induktion über den Tag n , an dem der Test stattfinden könnte, durch.

I.A.: $n = 5$

Wurde der Test bis inkl. Donnerstag nicht geschrieben, so muß er am Freitag (Tag 5) stattfinden. Dies ist aber keine Überraschung mehr. Also kann er am Freitag nicht stattfinden.

I.H.: Es gibt ein $n \leq 5$, so daß der Test an allen Tagen $\geq n$ kein Überraschungstest wäre, also dort nicht stattfinden kann.

I.S.: $n \rightarrow n - 1$

Nach I.H. kann der Test an allen Tagen $\geq n$ nicht geschrieben werden. Wurde er aber bis inkl. zum Tag $n - 2$ noch nicht geschrieben, so kann er nur noch am Tag $n - 1$ stattfinden. Da dies keine Überraschung mehr ist, kann dies nicht sein. Er muß also früher stattfinden.

⇒ Der Test findet nie statt, da über das Wochenende alle Schüler ausreichend viel Zeit haben, die obigen Schlußfolgerungen zu ziehen. So wäre nicht einmal der Montag eine Überraschung.

Hinweis: Geht man davon aus, daß alle Schüler diesen Schluß ziehen, wäre es dennoch eine Überraschung, wenn der Test dann stattfindet. Allerdings kann diese „Denktiefe“ sukzessive erhöht werden, so daß letztlich keine eindeutige Aussage möglich ist.

Lösung zu Aufgabe 3

Eine Möglichkeit ist folgende:

1. Lege den ersten Schalter um und warte eine Weile
2. Lege den ersten und den zweiten Schalter um
3. Gehe sofort in den Keller und berühre die Lampe

Ist das Licht an, so ist es der zweite Schalter. Ist das Licht aus, aber die Lampe warm, so muß es der erste Schalter sein. Ist das Licht aus und die Lampe kalt, ist es offensichtlich der dritte Schalter.

Lösung zu Aufgabe 4

a) Folgende Lösung fiel mir ein:

```
funktion1 :: Integer → Integer → Integer
funktion1 = (+)
```

```
funktion2 :: (Integer → Integer) → Integer
funktion2 = (+)
```

```
funktion3 :: Integer → (Integer → Integer)
funktion3 x = (\y → x+y)
```

```
funktion4 :: (Integer,Integer) → Integer
funktion4 (x,y) = x+y
```

b) Eine von vielen möglichen Lösungen:

```
pascal :: Integer → [Integer]
pascal n = [bin n m | m ← [0..n]]
```

```
bin :: Integer → Integer → Integer
bin n k = ((product [k+1..n]) 'div' (product [1..n-k]))
```

Diese Variante die Binomialkoeffizienten zu berechnen (wie im Pascalschen Dreieck) ist natürlich auch richtig, aber sehr ineffizient!

```
bin2 :: Integer → Integer → Integer
bin2 k 0 = 1
bin2 0 n = 0
bin2 n k = (bin2 (n-1) k) + (bin2 (n-1) (k-1))
```

c) Die Funktion berechnet eine Matrix-Multiplikation, wobei eine Matrix als Liste von Listen dargestellt wird.

d) Die Fibonacci Funktion:

```
fibonacci :: Integer → Integer → Integer → Integer
fibonacci n0 _ 0 = n0
fibonacci _ n1 1 = n1
fibonacci n0 n1 n = fn + fm
  where fn, fm :: Integer
        fn = fibonacci n0 n1 (n-2)
        fm = fibonacci n0 n1 (n-1)
```