



Lösung zu Aufgabe 1

a) Zunächst bin:

bin :: Integer → Integer → Integer

bin k 0 = 1

$$T(n, 0) = 0$$

bin 0 n = 0

$$T(0, n) = 0$$

bin n k = (bin (n-1) k) + (bin (n-1) (k-1)) $T(n, k) = 1 + T(n-1, k) + T(n-1, k-1)$

Die Rekurrenz ist somit:

$$T(n, k) = \begin{cases} 0, & n = 0 \\ 0, & k = 0 \\ 1 + T(n-1, k) + T(n-1, k-1), & \text{sonst} \end{cases}$$

Die geschlossene Form ist:

$$T(n, k) = \sum_{j=1}^k \binom{n}{j}.$$

. Falls man die Negationen als Additionen zählen möchte ergibt sich einfach ein zusätzlicher Faktor 4.

Nun zu bin2:

Der Ausdruck `product [k+1..n]` berechnet das Produkt von $n - k$ Zahlen und benötigt dazu $n - k - 1$ Multiplikationen. Der Ausdruck `product [1..n-k]` benötigt analog $n - k - 1$ Multiplikationen. Insgesamt ist also $T(n, k) = 2 * (n - k - 1)$ linear in n und k .

b) Bei jedem rekursiven Aufruf verringert sich das größere der beiden Argumente (oBdA. *a*) modulo des Kleineren (oBdA. *b*). Im schlimmsten Fall findet dabei keine echte Division statt, sondern $a \bmod b$ ist einfach $a - b$. Die beiden Argumente für den nächsten Aufruf sind dann b und $a - b$. Bezeichnet man nun die Folge der Argumente mit (f_n) , so gilt also im schlimmsten Fall: $f_{n-2} = f_n - f_{n-1}$ oder $f_n = f_{n-1} + f_{n-2}$. Der schlimmste Fall sind also zwei aufeinanderfolgende Fibonacci-Zahlen! Dann wird in jedem Rekursionsschritt nur subtrahiert statt dividiert. Da die Fibonacci-Zahlen exponentiell wachsen, ist der Aufwand im schlimmsten Fall also logarithmisch.

Lösung zu Aufgabe 2

a) Seien F, G und H Formeln und \mathcal{A} eine passende Struktur:

- reflexiv: Klar, $\mathcal{A}(F) = \mathcal{A}(F)$
- symmetrisch: Einfach, denn “=” ist symmetrisch.
- transitiv: läßt sich auf die Transitivität von “=” zurückführen.
- $F \equiv G$ gdw. $(F \wedge H) \equiv (G \wedge H)$:
Ist \mathcal{A} eine passende Struktur für F, G und H , so gilt
 $F \equiv G$ gdw. $\mathcal{A}(F) = \mathcal{A}(G)$ gdw.
 $\mathcal{A}(F)$ und $\mathcal{A}(H) = \mathcal{A}(G)$ und $\mathcal{A}(H)$ gdw.
 $(F \wedge H) \equiv (G \wedge H)$.
- $F \equiv G$ gdw. $(F \vee H) \equiv (G \vee H)$: analog.
- $F \equiv G$ gdw. $\neg F \equiv \neg G$: Einfach!
- $F \equiv G$ gdw. $\forall x F \equiv \forall x G$:
Ist \mathcal{A} eine passende Struktur für $\forall x F$ und $\forall x G$, so gilt:
 $F \equiv G$ gdw. $\mathcal{A}(F) = \mathcal{A}(G)$ gdw.
für alle $d \in U_{\mathcal{A}}$ $\mathcal{A}(F[x/d]) = \mathcal{A}(G[x/d])$, wobei $[x/d]$ hier alle freien Vorkommen von x ersetzen soll. Wäre dieser Schritt hier nicht erlaubt, so gäbe es oBdA. in F ein freies x und in G nicht, dann würde es aber auch eine Struktur geben (mit x), die G erfüllt, aber F nicht! Die letzte Formel ist dann gleichbedeutend mit: $\forall x F \equiv \forall x G$
- $F \equiv G$ gdw. $\exists x F \equiv \exists x G$: analog

transitiv: Ist $f \in \Theta(g)$ mit den Konstanten (c_1, c_2, n_0) und $g \in \Theta(h)$ mit (d_1, d_2, m_0) , so ist $f \in \Theta(h)$ mit den Konstanten $(\text{maximum}(c_1, d_1), \text{maximum}(c_2, d_2), \text{maximum}(n_0, m_0))$

Lösung zu Aufgabe 3

a) Der Beweis erfolgt mit vollständiger Induktion über n

Induktionsanfang: Für $m = 1$ gilt:

$$F_{n+1} = F_1 F_{n+1} + F_0 F_n = F_{n+1}. \text{ Analog gilt für } m = 2: F_{n+2} = F_2 F_{n+2} = F_2 F_n + F_1 F_n = F_{n+2} + F_n.$$

Induktionsannahme: Die Behauptung gelte für m und $m + 1$ und ein n .

Induktionsschritt:

$$F_{n+m+1} = F_{n+m} + F_{n+(m-1)} \text{ nach Definition} \quad (1)$$

$$= (F_m F_{n+1} + F_{m-1} F_n) + (F_{m-1} F_{n+1} + F_{m-2} F_n) \text{ nach Induktionsannahme} \quad (2)$$

$$= (F_m + F_{m-1}) F_{n+1} + (F_{m-1} + F_{m-2}) F_n \quad (3)$$

$$= F_{m+1} F_{n+1} + F_m F_n \quad (4)$$

b) *Induktionsanfang:*

$$n = 1: \quad F_2 F_0 - F_1^2 = -1 = (-1)^1$$

$$n = 2: \quad F_3 F_1 - F_2^2 = 1 = (-1)^2$$

$$n = 3: \quad F_4 F_2 - F_3^2 = -1 = (-1)^3$$

Induktionsannahme: Die Behauptung gelte für $n, n - 1$ und $n - 2$.

Induktionsschritt:

$$\begin{aligned} F_{n+2} F_n - F_{n-1}^2 &= (F_{n+1} + F_n) \cdot (F_{n-1} + F_{n-2}) - (F_n + F_{n-1})^2 \\ &= (F_{n+1} F_{n-1}) + (F_{n+1} F_{n-2} + F_n F_{n-2} - (F_n^2 + 2F_n F_{n-1} + F_{n-1}^2)) \\ &= (F_n F_{n-1} + F_{n-1}^2 + F_n F_{n-1} + (F_{n+1} F_{n-2} + F_n F_{n-2} - F_n^2 - 2F_n F_{n-1} - F_{n-1}^2)) \\ &= F_{n+1} F_{n-2} + F_n F_{n-2} - 2 - F_n^2 \\ &= F_{n+1} F_{n-2} + F_n F_{n-2} - 2 - (F_n F_{n-1} + F_n F_{n-2}) \\ &= F_{n+1} F_{n-2} - F_n F_{n-1} \\ &= F_n F_{n-2} + F_{n-1} F_{n-2} - F_n F_{n-1} \\ &= F_n F_{n-2} + F_{n-1} F_{n-2} - (F_{n-1}^2 + F_{n-2} F_{n-1}) \\ &= F_n F_{n-2} - F_n^2 \\ &= (-1)^{n-1} \text{ nach Induktionsannahme} \\ &= (-1)^{n+1}. \end{aligned}$$

c) Ein gemeinsamer Teiler von F_{n+1} und F_n teilt nach Teilaufgabe b) auch $(-1)^n$. Damit ist der ggT von F_{n+1} und F_n Eins.

d) Aus Teilaufgabe b) ergibt sich, daß jeder gemeinsame Teiler von F_m und F_n auch ein Teiler von F_{m+n} ist.

Umgekehrt ist jeder gemeinsame Teiler von F_{m+n} und F_m ein Teiler von $F_m F_m + n$. Da wegen Teilaufgabe c) F_n und F_{n+1} relativ prim sind, ist jeder gemeinsame Teiler von F_{m+n} und F_n ein Teiler von F_m .

Es gilt also für jedes $d \in \mathbb{N}$

$$d \text{ teilt } F_m \text{ und } F_n \text{ gdw. } d \text{ teilt } F_{m+n} \text{ und } F_n$$

e) In Teilaufgabe d) wurde bewiesen, daß für jede Zahl d gilt:

$$d \text{ teilt } F_m \text{ und } F_n \text{ gdw. } d \text{ teilt } F_{m+n} \text{ und } F_n$$

Durch Induktion ist leicht zu zeigen, daß auch

$$d \text{ teilt } F_m \text{ und } F_n \text{ gdw. } d \text{ teilt } F_{m+kn} \text{ und } F_n$$

Das ist genau die Bedingung für den Euklidischen Algorithmus. Wendet man diese also wie im Euklidischen Algorithmus solange an bis eines der Argumente Null ist, so hat man die Aussage für den ggT.

Lösung zu Aufgabe 4

- a) Hier ist eine recht einfach Version von Merge-Sort:

```
mergeSort :: Ord a => [a] -> [a]
mergeSort [x] = [x]
mergeSort xs = let (zs,ys) = divideList xs
                in merge (mergeSort zs) (mergeSort ys)
```

```
merge :: (Ord a) => [a] -> [a] -> [a]
merge xs [] = xs
merge [] ys = ys
merge (x:xs) (y:ys)
  | x <= y = x:merge xs (y:ys)
  | otherwise = y:merge (x:xs) ys
```

```
divideList :: [a] -> ([a],[a])
divideList xs = (take h xs, drop h xs)
  where h = (length xs) `div` 2
```

- b) Insertion-Sort ist einfach:

```
insertionSort :: (Ord a) => [a] -> [a]
insertionSort xs = foldl einf [] xs
  where einf :: Ord a => [a] -> a -> [a]
        einf [] x = [x]
        einf (y:ys) x | x <= y = x:y:ys
                      | otherwise = y: einf ys x
```

- c) Die fehlenden Funktionen:

```
instance Num F2 where
  0 + x = x
  1 + x = if x == 0 then 1 else 0
  (-) = (+)
  0 * x = 0
  1 * x = x
  negate = id
  signum _ = 0
  abs = id
  fromInteger i = if (even i) then 0 else 1
```

Lösung zu Aufgabe 5

- a) Die Kugeln werden mit $1, 2, \dots, 12$ bezeichnet.

Erstes Wiegen:

Wiege die Kugeln $1 - 4$ und $5 - 8$: Ist das Gewicht gleich, so muß die abweichende Kugel in $9 - 12$ sein. Das ist *Fall 1*. Unterscheidet sich das Gewicht, so sind wir in *Fall 2*.

Fall 1: Klar ist nun Kugeln $1 - 8$ haben alle das gleiche Gewicht.

zweites Wiegen: $1 - 3$ und $9 - 11$. Ist das Gewicht gleich, so muß 12 ein abweichendes Gewicht haben. Durch Vergleichen von 12 mit 4 läßt sich im *dritten Wiegen* bestimmen, ob die Kugel leichter oder schwerer ist.

Ist das Gewicht aber unterschiedlich, so steht fest ob die abweichende Kugel in $9 - 11$ ein größeres oder kleineres Gewicht hat. Im *dritten Wiegen* 9 und 10 entscheidet sich welche der drei Kugeln ein abweichendes Gewicht hat: Ist das Gewicht gleich, so ist es 11 sonst je nach Abweichung 9 oder 10 .

Fall 2: Hier steht schon fest, daß die Kugeln $9 - 12$ alle das Gleiche wiegen und oBdA. sind $1 - 4$ leichter als $5 - 8$ (sonst umbenennen!).

Zweites Wiegen: $1, 2, 5$ und $3, 6, 9$. Ist das Gewicht gleich, so muß 4 leichter sein oder $7, 8$ sind schwerer. Durch ein *drittes Wiegen* von 3 und 4 läßt sich herausfinden welche Kugel welches abweichende Gewicht hat.

Der verbleibende Fall 1, 2, 5 schwerer als 3, 6, 9 geht analog.

- b) Konstruktionsbedingt kann eine Balkenwaage nur jeweils gleichviele Kugeln gegeneinander aufwiegen. Im ersten Schritt erzielt man offensichtlich den größten Gewinn an Information, wenn man die Kugeln in drei Teile aufteilt, wobei zwei Teile gleichviele Kugeln enthalten müssen. Somit reduziert sich die Frage darauf, ob man das Problem für fünf Kugeln und zwei Wägungen lösen kann.

Hier ist jedoch nur eine Aufteilung $5 = 2 + 2 + 1$ sinnvoll und im Falle von Ungleichheit ist es nur möglich festzustellen, welche Kugel ein abweichendes Gewicht hat, nicht jedoch ob es kleiner oder größer ist.

Interessanterweise ist das Problem für sechs Kugeln hingegen lösbar und damit auch das für 14.

Fazit: Es fehlt eine Kugel oder es ist eine zuviel!