Inhaltsverzeichnis

1 Polynomordungen [9 Punkte]

$$(\mu\ x)\ \because\ y\ =\ y\ \because\ x \tag{1.1}$$
$$(x\ \because\ y)\ \because\ z\ =\ z\ \because\ (y\ \because\ (\mu\ x)) \tag{1.2}$$
$$(x\ \because\ (\mu\ (\mu\ y)))\ =\ (x\ \because\ (\mu\ y))\ \because\ y \tag{1.3}$$

1. Dieses System ist *nicht* stark normalisierend. Geben Sie ein Gegenbeispiel an.
2. Zweite Regel wird *ersetzt* durch:

$$(x\ \because\ y)\ \because\ z\ =\ z\ \because\ (y\ \because\ x) \tag{1.2'}$$

Nehmen Sie an, dass das neue System stark normalisierend ist. Ist es auch konfluent? Bringen Sie einen Beweis oder ein Gegenbeispiel.

2 System F [9 Punkte]

1. Man erinnere sich an den Typ der Paare definiert als $a \times b = \forall r.(a \to b \to r) \to r$.
   Geben Sie in System F eine Typherleitung an, sodass gilt:

   $$\{eval : \ \forall a, b.(a \to b) \to a \to b\} \vdash (\lambda x.(x\ eval)) : \forall ab.((a \to b) \times a) \to b$$

2. Wir definieren den Term

   $$s = \text{let } (f = \lambda t.t\ succ) \text{ in } (map\ f)\ (map\ (run\ 0)\ (f\ cons\ nil))$$

   und sowie den Kontext

   $$\begin{aligned}\Gamma \ = \ \{&0 : \mathbb{N},\ run : \mathbb{N} \to (\mathbb{N} \to \mathbb{N}) \to \mathbb{N},\ map : \forall ab.\ (a \to b) \to \mathbb{L}\ a \to \mathbb{L}\ b,\\ &nil : \forall a.\ \mathbb{L}\ a,\ cons : \forall, a.\ a \to \mathbb{L}\ a \to \mathbb{L}\ a,\ succ : \mathbb{N} \to \mathbb{N}\}.\end{aligned}$$

   Bestimmen Sie die Menge $PT(\Gamma;\ s;\ \alpha)$. Hinweis: Sie müssen das daraus resultierende Unifikationsproblem nicht lösen. Das kleinere Unifikationsproblem können Sie durch alleiniges Angeben des allgemeinen Unifikators angeben.

3 Strukturelle Induktion und Folds [9 Punkte]

1. Strukturelle Induktion

   Wir definieren die Funktionen $\oplus$ und rem:

   $$Nil \oplus ys = ys$$
   $$(Cons\ x\ xs) \oplus ys = Cons\ x\ (xs \oplus ys)$$
   $$rem\ x\ Nil = Nil$$
   $$rem\ x\ (Cons\ y\ ys) = if\ (x == y)\ \text{then}\ (rem\ x\ ys)\ \text{else}\ (Cons\ y\ (rem\ x\ ys))$$

   Zeigen Sie per Induktion folgende Behauptung:

   $$\forall\ x :\ a,\ ys,\ xs :\ List\ a.rem\ x\ (\ ys \oplus zs) = (rem\ x\ ys) \oplus (rem\ x\ zs)$$

   Hinweis: Eine Fallunterscheidung könnte sich als hilfreich erweisen. Geben Sie Ihre Induktionsvoraussetzung explizit an.

2. Induktive Datentypen, Folds

```
data FizzBuzz a where
  Done: () -> FizzBuzz a
  Fizz: a -> FizzBuzz a -> FizzBuzz a
    Buzz: a -> FizzBuzz a -> FizzBuzz a
```

   a) Definieren Sie die entsprechende fold-Funktion $foldfz$ auf dem induktiven Datentypen FizzBuzz rekursiv und geben Sie ihren Typ an.

   b) Definieren Sie die Funktion fizzbuzz : $FizzBuzz\ Nat \to FizzBuzz\ Nat$ mittels der fold-Funktion aus a), sodass sie konsekutive Fizz'es und Buzz'es durch Addition zusammenfässt:

   Bsp.:

   $$\text{fizzbuzz Done} = \text{fizzbuzz Done}$$
   $$\text{fizzbuzz (Fizz 3 (Buzz 4 Done))} = \text{Fizz 3 (Buzz 4 Done)}$$
   $$\text{fizzbuzz (Fizz 3 (Buzz 4 (Fizz 5 (Fizz 6 Done))))} = \text{Fizz 3 (Buzz 4 (Fizz 11 Done))}$$
   $$\text{fizzbuzz (Buzz 8 (Buzz 9 Done))} = \text{Buzz 17 Done}$$

   Hinweis: Sie können die an foldfz übergebene Funktionen separat induktiv auf den Konstruktoren von FizzBuzz definieren.
   Hinweis 2: Sie können die üblichen Operationen auf Nat als gegeben ansehen.

Lösung: $foldfz : b \to (a \to b \to b) \to (a \to b \to b) \to FizzBuzza \to b$

$$foldfz\ f\ g\ h\ \text{Done} = f$$
$$foldfz\ f\ g\ h\ (\text{Fizz}\ x\ y) = g\ x\ (foldfz\ f\ g\ h\ y)$$
$$foldfz\ f\ g\ h\ (\text{Buzz}\ x\ y) = h\ x\ (foldfz\ f\ g\ h\ y)$$

$fizzbuzz = foldfz\ f\ g\ h$ // f ist für Done, g ist für Fizz, h für Buzz (Erinnerung)
$f = Done$

$$g\ x\ Done = \text{Fizz}\ x\ Done$$
$$g\ x\ (\text{Fizz}\ y\ z) = \text{Fizz}\ (x + y)\ z$$
$$g\ x\ (\text{Buzz}\ y\ z) = \text{Fizz}\ \ x\ (\text{Buzz}\ y\ z)$$

$$h\ x\ \text{Done} = BuzzxDone$$
$$h\ x\ (\text{Fizz}\ y\ z) = \text{Buzz}\ x\ (Fizzyz)$$
$$h\ x\ (\text{Buzz}\ y\ z) = \text{Buzz}\ (y + x)\ z$$

4 Korekursion und Koinduktion [9 Punkte]

```
codata Bitstream where
    cur: Bitstream -> Bool
    next: Bitstream -> Bitstream
```

Es ist die Funktion alt : $Bool \to Bool \to Bitstream$ gegeben, die zwei Bits in einem Bitstream alterniert:

$$\text{cur}\ (\text{alt}\ a\ b) = a$$
$$\text{next}\ (\text{alt}\ a\ b) = \text{alt}\ b\ a$$

1. Definieren Sie die Funktion xor : $Bitstream \to Bitstream \to Bitstream$, die zwei Streams XOR-verknüpft. Sie können die üblichen Operationen auf den Grunddatentypen (z. B. $Bool$) als gegeben annehmen.
2. Definieren Sie den Begriff einer Bisimulation für diese Kodatentyp. Er ergibt sich durch Spezialisierung des in der vorgestellten Konzepts auf diesen Fall.
3. Wir schreiben nunmehr $\top$ und $\bot$ für $true$ und $false$. Beweisen Sie mittels Koinduktion dass gilt:

$$\text{xor}\ (\text{alt}\ \bot\ \top)\ (\text{alt}\ \bot\ \bot) = \text{alt}\ \bot\ \top$$

Lösung:

1.

$$\text{cur}\ (\text{xor}\ a\ b) = \text{xor'}\ (\text{cur}\ a)\ (\text{cur}\ b)$$
$$\text{next}\ (\text{xor}\ a\ b) = \text{xor}\ (\text{next}\ a)\ (\text{next}\ b)$$

Dabei ist xor' die xor-Funktion auf Bools, laut Angabe erlaubt.

2. $R \subseteq Bitstream \times Bitstream$ heißt Bisimulation $\Leftrightarrow$ für alle (s, t) $\in R$ gilt:

$$\text{cur}\ s = \text{cur}\ t \quad \text{und} \quad (\text{next}\ s)\ R\ (\text{next}\ t).$$

3. Zeige, dass

$$R = \{(\text{xor}\ (\text{alt}\ \bot\ \top)\ (\text{alt}\ \bot\ \top),\ \text{alt}\ \bot\ \top),$$
$$(\text{xor}\ (\text{alt}\ \top\ \bot)\ (\text{alt}\ \bot\ \bot), \text{alt}\ \top\ \bot)\}$$

eine Bisimulation ist. Beachte, dass das zweite Element in R erst beim Beweisen für das erste Paar 'herausgefunden' wird :)

5 Automatenminimierung [5 Punkte]

Minimieren Sie folgenden Automaten nach dem tabellarischen Algorithmus aus der Vorlesung. Geben Sie auch den resultierenden Automaten an.