Klausur Theorie der Programmierung

Braindump

Sommersemester 2014

Benotung:
Gesamt Punkte: 22; Bestanden ab 7 Punkte; Note 1,0 ab 19 Punkte.

Inhaltsverzeichnis

1 Konfluenz und Terminierung (4 Punkte)

Wir definieren ein Termersetzungssystem über das aus zwei binären Funktionssymbolen $\Uparrow$ und $\Downarrow$ (in Infixnotation geschrieben) bestehenden Signatur Σ durch:

$$x \Uparrow (y \Downarrow z) \to (z \Downarrow x) \Uparrow y$$
$$x \Uparrow y \to (y \Downarrow y) \Downarrow x$$

1. Zeigen Sie mittels Polynomordnungen, dass das Sytem stark normalisierend ist.

2. Ist das System konfluent? Geben Sie einen Beweis bzw. ein Gegenbeispiel an.

2 Zahlen in System F (5 Punkte)

Man erinnere sich an folgende auf Church-Kodierung definierte Funktionen:

- Church-Numerale

  $succ\ n = \lambda fa.f(n\ f\ a)$
  $pred\ n = fst(n(\lambda p.pair(snd\ p)(succ(snd\ p)))(pair\lceil 0\rceil\lceil 0\rceil))$
  $isZero\ n = n(\lambda x.false)\ true$
  $geq\ n\ m = isZero(n\ pred\ m)$

- Church-Booleans

  $true = \lambda xy.x$
  $false = \lambda xy.y$

- Church-Paare

  $pair\ a\ b = \lambda\ select.select\ a\ b$
  $fst\ p = p(\lambda xy.x)$
  $snd\ p = p(\lambda xy.y)$

Man erinnere sich ferner, dass die Zahl n durch den λ-Term $\lceil n\rceil = \lambda fa.f^n a$ dargestellt wird, also z.B.

$\lceil 0\rceil = \lambda fa.a$
$\lceil 1\rceil = \lambda fa.fa$
$\lceil 1\rceil = \lambda fa.f(fa)$

1. Geben Sie die ersten vier $\delta\beta$-Rekursionsschritte des Terms $geq\lceil 2\rceil\lceil 1\rceil$ unter a) normaler und b) applikativer Reduktion an. Markieren Sie (durch Unterstreichen) in jedem Schritt den zu reduzierenden Redex.

   Nevenbemerkung: Wir sehen (anders als für die Operationen one, two, etc. inder Probeklausur!) $\lceil n\rceil$ als Abkürzung für den entsprechenden λ-Term an, z.B. ist also $geq\lceil 2\rceil\lceil 1\rceil$ ohne Reduktion als $geq(\lambda fa.f(fa))(\lambda fa.fa)$ zu lesen.

   Hinweis: die Reduktionsstrategie bezieht sich ausdrücklich auch auf δ-Reduktion. D.h. bei normaler Reduktion werden zuerst die linken Seiten von Funktionsanwendungen (also die Funktionen $\beta\delta$-reduziert, von außen nach innen, und erst dann die Argumente von

Funktionsanwendungen, bei applikativer Reduktion dagegen zuerst die Argumente von Funktionsanwendungen.

2. Man erinnere sich, dass die Church-Numerale und die Church-Booleans in System F den Typ $\mathbb{N} := \forall a(a \to a) \to a \to a$ bzw. $\mathbb{B} := \forall a.a \to a \to a$ haben.

   Zeigen Sie unter der Annahme, dass *isZero* den Typ $\mathbb{N} \to \mathbb{B}$ hat und *pred* den Typ $\mathbb{N} \to \mathbb{N}$, dass der Term $\lambda nm.isZero(n\ pred\ m)$ den Typ $\mathbb{N} \to \mathbb{N} \to \mathbb{B}$ hat; d.h. geben Sie eine Typherleitung in System F für $\{pred : \mathbb{N} \to \mathbb{N}, isZero : \mathbb{N} \to \mathbb{B}\} \vdash \lambda nm.isZero(n\ pred\ m) : \mathbb{N} \to \mathbb{N} \to \mathbb{B}$ an.

   (Hinweis: Für diesen Teil der Aufgabe sind obige Definition von *isZero* und *pred* erkennbar irrelevant!)

3 Strukturelle Induktion (5 Punkte)

Wir erinnern an den Datentyp der Listen und einige hierauf rekursiv definierte Standadfunktionen:

```
data List a = Nil | Cons a (List a)

Nil ⊕ ys = ys
(Cons x xs) ⊕ ys = Cons x (xs ⊕ ys)

mapList f Nil = Nil
mapList f (Cons x xs) = Cons(fx)(mapList f xs)
```

Beweisen Sie mittels struktureller Induktion, dass

$\forall f\ xs\ ys.mapList\ f(xs \oplus ys) = (mapList\ f\ xs) \oplus (mapList\ f\ ys)$

Geben Sie im Induktionsschritt die Induktionsannahme explizit an, und erläutern Sie alle Schritte des Beweises.

4 Korekursion und Koinduktion (5 Punkte)

Ein digitales Signal ist ein zeitlich veränderlicher Wert zwischen $-\infty$ und $+\infty$ wobei wir Zeit als diskret behaldeln, entsprechend etwa fortgesetztem Sampling. Solche Signale lassen sich in natürlicher Weise durch einen Kodatentyp repräsentieren

```
codata Signal where
       currentSample: Signal -> Int
       discardSample: Siglal -> Signal
```

Dann konstruiert die wie folgt korekursiv definierte Funktion square aus ihren Argumenten x,y ein zw. x und y alternierendes Signal:

```
currentSample(square x y) = x
discardSample(square x y) = square y x
```

1. Definieren Sie korrekursiv eine Fuktion invert: Signal -> Signal, so dass invert s das Vorzeichen jedes Wertes s umkehrt.

   Hinweis: Sie dürfen bei der Definition die üblichen Operationen auf Basistypen (z.B. Arithmetik auf Int) als gegeben annehmen.

2. Geben Sie die Bedingungen an, die eine Relation erfüllen muss, um eine Bisimulation auf signal zu sein. (Diese ergeben sich durch Spezialisierung des allgemeinen Begriffs aus der Vorlesung auf den Kodatentyp signal)

3. Beweisen Sie die folgende Eigenschaft durch Koinduktion:

   $$\forall s \in \text{Signal}\ \forall x \in \textbf{Int}.\ \text{discardSample}\ (\text{square}\ x\ -x) = \text{invert}(\text{square}\ x\ -x)$$

   Rechtfertigen Sie alle Beweisschritte.

5 Reguläre Ausdrücke (3 Punkte)

Sei L die Sprache über $\Sigma = \{0, 1\}$, die gerade aus allen Worten über Σ besteht, die mehr 1en als 0en enthalten. Z.B. sind 1, 101 und 01000111110 in L, nicht aber ε, 01, 100, ...

Ist L regulär? Wenn ja, geben sie einen regulären Ausdruck für L an und erläutern Sie, warum dieser L definiert. Andernfalls zeigen Sie mittels des Pumping-Lemma, dass L nicht regulär ist.