Klausur Braindump*

Theorie der Programmierung

Diverse Teilnehmer

Sommersemester 19

1 Polynomordnungen und TES

9 Punkte

Wir definieren ein Termersetzungssystem mit der Signatur Σ bestehend aus einem unärem Operator $*(x)$ (geschrieben als x^*) und einem binärem, infix Operator $x \vdash y$:

$$(x \vdash y^*) \ \to_0 \ (y \vdash x) \tag{1.1}$$
$$(x \vdash y) \vdash z \ \to_0 \ z \vdash (y \vdash x^*) \tag{1.2}$$
$$x \vdash (y^*)^* \ \to_0 \ (x \vdash y^*) \vdash y \tag{1.3}$$

1. Zeigen Sie, dass das TES nicht stark normalisierend ist. 3 Punkte
2. Gehen Sie im Folgenden davon aus, dass Regel (1.2) entfernt wird. Zeigen Sie mithilfe von geeigneten Polynomordnungen, dass das TES jetzt stark normalisierend ist. 3 Punkte
3. Es gibt nur ein einziges kritisches Paar. Bestimmen Sie dieses und prüfen Sie, ob das TES konfluent ist. 3 Punkte

2 System F

8 Punkte

In funktionalen Programmiersprachen werden *Maybe* Datentypen benutzt um eine Berechnung mit Fehlermöglichkeit zu beschreiben. Dieses kann in System F mit dem Typen

$$\mathbb{M}a := \forall r \,.\, (a \to r) \to r \to r$$

kodiert werden. Nehmen Sie zudem an, dass Listen den Typ

$$\mathbb{L}a := \forall r \,.\, r \to (a \to r \to r) \to r$$

besitzen.

1. Definieren Sie die Konstruktoren für „some : $\forall a \,.\, a \to \mathbb{M}a$“ und „none : $\forall a \,.\, \mathbb{M}a$“ (d.h. geben sie die Lambda-Terme an). 2 Punkte
2. Sei $t : \forall a \,.\, \mathbb{L}a \to \mathbb{M}a$ definiert als 5 Punkte

   $$t = \lambda l \,.\, l \text{ none } (\lambda x q \,.\, \text{some } x)$$

   Geben Sie die Typeherleitung in System F für

   $$\{ \text{ some} : \forall a \,.\, a \to \mathbb{M}a, \text{ none} : \forall a \,.\, \mathbb{M}a \ \} \vdash t : \forall a \,.\, \mathbb{L}a \to \mathbb{M}a$$

3. Welche bekannte Funktion auf Listen stellt t dar? 1 Punkt

* *Wie Immer:* Keine Garantie auf Richtigkeit. Angaben werden zum meisten Teil vereinfacht wiedergegeben. Fehler und Verbesserungen via Gitlab melden: `https://gitlab.cs.fau.de/oj14ozun/thprog-ss19`.

3 Strukturelle Induktion und Folds **9 Punkte**

In dieser Aufgabe wird der Grunddatentyp `char` verwendet. Es wird die gewöhnliche Schreibweise `'_'` für `char`- und `"_"` `String`-Literale benutzt. Definiert Seien

```
data Nat = Z | S Nat

data String where
    Empty: String
    Cons: char -> String -> String
```

Auf diesen Typen sind die Operationen „$\oplus : \texttt{String} \to \texttt{String} \to \texttt{String}$“ und „$+ : \texttt{Nat} \to \texttt{Nat} \to \texttt{Nat}$“ definiert:

$$\texttt{Empty} \oplus t = t \qquad (\texttt{Cons}\ x\ s) \oplus t = \texttt{Cons}\ x\ (s \oplus t)$$

$$\texttt{Z} + s = s \qquad (\texttt{S}\ n) + m = \texttt{S}\ (n + m)$$

Mitetls der *fold*-Funktion für natürliche Zahlen

```
foldN : a -> (a -> a) -> Nat -> a
foldN x f Z = x
foldn x f (S n) = f (foldN x f n)
```

ist die Funktion `unary`, welche ein `Nat` in die unäre Repräsentation dieser Zahl in Form eines Strings umwandelt, ist definiert als:

```
unary = foldN Empty (Cons '|')
```

Per Definition, besitzt diese Funktion die Eigenschaft:

```
unary (S n) = Cons '|' (unary n)
```

1. Beweisen Sie mittels struktureller Induktion folgende weitere Eigenschaft: **6 Punkte**

$$\forall n, m \in \texttt{Nat}\,.\,(\texttt{unary}\ n) \oplus (\texttt{unary}\ m) = \texttt{unary}\ (n + m)$$

2. Betrachten Sie diese Variante eines Binärbaums, bei dem alle Knoten durch Natürliche Zahlen gelabelt' sind: **3 Punkte**

```
data NatTree where
  leaf:   Nat -> NatTree
  branch: NatTree -> NatTree -> NatTree
```

Geben Sie Typ und Definition einer *fold* Funktion, für `NatTree` an.

4 Korekursion und Koinduktion **8 Punkte**

```
codata S a where
    hd: S a -> a
    tl: S a -> S a


hd (genl t) = node t
tl (genl t) = genl (left t)

hd (genr t) = node t
tl (genr t) = genl (right t)
```

```
codata IT a where
    node: IT a -> a
    left: IT a -> IT a
    right: IT a -> IT a

hd (const x) = x
tl (const x) = const x

node (slant s) = hd s
left (slant s) = slant (tl s)
right (slant s) = slant s
```

1. Zeigen Sie, dass folgende Aussagen gelten: 5 Punkte

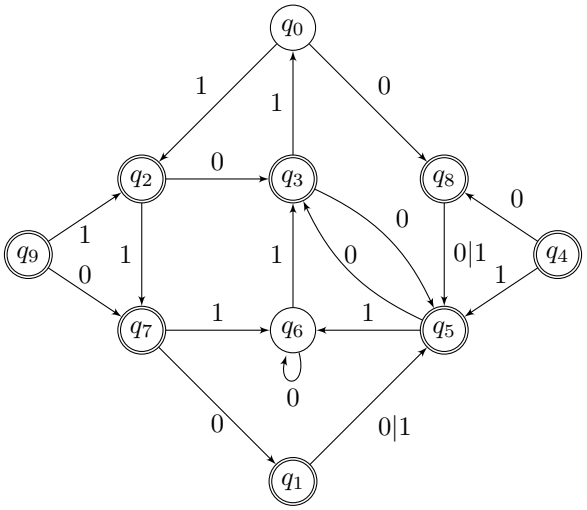
```
genl (slant s) = s
genr (slant s) = const (hd s)
```

2. Definieren Sie korekursiv eine Funktion layers : S$a \to$ ITa, die einen Baum erzeugt, dessen n'tes Element alle Knoten der n'ten Ebene beschriftet. 3 Punkte

   Beispiel: Wenn s der Stream $1, 2, 3, \ldots$ der natürlichen Zahlen ist, ist layers$s$ der Baum, dessen Wurzel mit 0 beschriftet ist, deren Kinder mit 1, die Kinder der Kinder mit 2, etc.

5 Automatenminimierung **6 Punkte**

Gegeben sei der abgebildete deterministische Automat, über dem Alphabet $\Sigma = \{0, 1\}$:



Minimieren sie den Automaten mittels des tabellarischen Algorithmus aus der Vorlesung und zeichnen sie den resultierenden Minimalautomaten.