11. Aufgabenblatt vom Freitag, den 24. Januar 2020 zur Vorlesung

ALP I: Funktionale Programmierung

Bearbeiter: A. Rudolph
Tutor: Stephanie Hoffmann
Tutorium 06

Abgabe: bis Montag, den 03. Februar 2020, 10:10 Uhr

1. **Aufgabe** (2 Punkte) *S(KK)I = K*
   Wir zeigen die Äquivalenz durch das einsetzen von 2 beliebigen Ausdrücken a und b:
   K a b = a
   S(KK)I a b ⇒ ((KK) a (I a)) b ⇒ ((KK) a a) b ⇒ K a b ⇒ a
   Die Ausdrücke sind äquivalent

2. **Aufgabe** (6 Punkte) *SS(SI(K(KI)))(KK(S(KK)I))(KI)*

⇒(SS(S(K(KI)))(KK(S(KK)I))(KI))	(Identität)
⇒(SS(S(K)))(KK(S(KK)I))(KI))	(Kanzellator)
⇒(SK(SK))(KK(S(KK)I))(KI))	(Funktionsapplikation)
⇒(KK(KS))(KK(S(KK)I))(KI))	(Funktionsapplikation)
⇒(K)(KK(S(KK)I))(KI))	(Kanzellator)
⇒(KK(S(KK)I))	(Kanzellator)
⇒K	(Kanzellator)

S(SI(K(II)))(S(S(KK)I))IS(KKI)

⇒(SI(K(II)))I((S(S(KK)I))I)S(KKI)	(Funktionsapplikation)
⇒II((K(II))I)((S(S(KK)I))I)S(KKI)	(Funktionsapplikation)
⇒((K(II))I)((S(S(KK)I))I)S(KKI)	(Identität)
⇒(II)((S(S(KK)I))I)S(KKI)	(Kanzellator)
⇒((S(S(KK)I))I)S(KKI)	(Identität)
⇒(S(KK)I)S(IS)(KKI)	(Funktionsapplikation)
⇒((KK)S(IS))(IS)(KKI)	(Funktionsapplikation)
⇒((K)(IS))(IS)(KKI)	(Kanzellator)
⇒(IS)(KKI)	(Kanzellator)
⇒(S)(KKI)	(Identität)
⇒(KI(KI))	(Funktionsapplikation)
⇒I	(Kanzellator)

3. **Aufgabe** (6 Punkte) *λx.y(xy) ≡ S(K y)(SI(K y))*
   λx.y(xy) $\overset{6}{\Rightarrow}$ (S T[λx.y] T[λx.(xy)]) $\overset{4}{\Rightarrow}$ (S (K T[y]) T[λx.(xy)]) $\overset{1}{\Rightarrow}$ (S (K y) T[λx.(xy)])
   $\overset{6}{\Rightarrow}$ (S (K y) (S T[λx.x] T[λx.y])) $\overset{3}{\Rightarrow}$ (S (K y) (S I T[λx.y])) $\overset{4}{\Rightarrow}$ (S (K y) (S I (K T[y])))
   $\overset{1}{\Rightarrow}$ (S (K y) (S I (K y)))
   ⇒ *S(K y)(SI(K y)*

4. **Aufgabe** (6 Punkte) *λs.λx.s(s(s(x))) ≡ (S(S(KS)K)(S(S(KS)K)I))*
   Seien a und b zwei beliebige Ausdrücke:
   λs.λx.s(s(s(x))) a b
   ⇔ λx.a(a(a(x))) b
   ⇔ a(a(a(b)))

   (S(S(KS)K)(S(S(KS)K)I)) a b
   ⇒(S(KS)K)a((S(S(KS)K)I))a) b (Funktionsapplikation))
   ⇒((KS)a(Ka))((S(S(KS)K)I))a) b (Funktionsappliktaon)
   ⇒(S(Ka))((S(S(KS)K)I))a) b (Kanzellator)
   ⇒((Ka)b(((S(S(KS)K)I))a)b) (Funktionsapplikation)
   ⇒(a(((S(S(KS)K)I))a)b) (Kanzellator)
   ⇒(a((S(KS)K)a)I(Ia)b) (Funktionsapplikation)
   ⇒(a((KS)a(Ka)I(Ia)b) (Funktionsapplikation)
   ⇒(a(S(Ka)I(Ia)b) (Kanzellator)
   ⇒(a((Ka)(Ia)(I(Ia)))b) (Funktionsapplikation)
   ⇒(a(a(I(Ia))b) (Kanzellator)
   ⇒(a(a(Ia)b)) (Identität)
   ⇒(a(a(a(b))) (Identität)
   Die Ausdrücke sind äquivalent

5. **Aufgabe** (8 Punkte)

```
ski_parser :: String -> Expr
ski_parser str = parse Nil str


parse :: Expr -> String -> Expr
parse Nil [] = emptyExpr
parse expr [] = expr

parse Nil ('(':rest) = parse (parse Nil inside) out
                          where (inside, out) = extract [] rest 0

parse expr (')':rest) = parse expr rest

parse Nil (a:rest) | letter a = parse (Var [a]) rest
                   | ((length rest) == 0) = (char2Exp a)


parse Nil (a:b:rest)
| ((expression a) && (expression b))
= parse (App (char2Exp a) (char2Exp b)) rest

| ((expression a) && (letter b)) = parse (App (char2Exp a) (Var [b])) rest
```

```
| ((letter a) && (expression b)) = parse (App (Var [a]) (char2Exp b)) rest

| ((letter a) && (letter b)) = parse (App (Var [a]) (Var [b])) rest
| otherwise = parse (Var [a]) (b:rest)

parse expr ('(':rest) = parse (App expr (parse Nil inside)) out
                          where (inside, out) = extract [] rest 0

parse expr (a:rest) | (expression a) = parse (App expr (char2Exp a)) rest

parse expr rest = illegalExpr rest

char2Exp :: Char -> Expr
char2Exp 'S' = S
char2Exp 'K' = K
char2Exp 'I' = I

expression :: Char -> Bool
expression x = (x == 'S') || (x == 'I') || (x == 'K')

emptyExpr = error "the empty expression is not a valid SKI-Expression"
notAnumber = error "an empty string is not a number"
illegalExpr str = error ("there is a syntax error in the expression
" ++ str)
```