

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:

$$K \ a \ b = a$$

$$S(KK)I \ a \ b \Rightarrow ((KK) \ a \ (I \ a)) \ b \Rightarrow ((KK) \ a \ a) \ b \Rightarrow K \ a \ b \Rightarrow a$$

Die Ausdrücke sind äquivalent

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

$$\begin{aligned} &\Rightarrow (SS(S(K(KI)))(KK(S(KK)I))(KI)) && \text{(Identität)} \\ &\Rightarrow (SS(S(K)))(KK(S(KK)I))(KI) && \text{(Kanzellator)} \\ &\Rightarrow (SK(SK))(KK(S(KK)I))(KI) && \text{(Funktionsapplikation)} \\ &\Rightarrow (KK(KS))(KK(S(KK)I))(KI) && \text{(Funktionsapplikation)} \\ &\Rightarrow (K)(KK(S(KK)I))(KI) && \text{(Kanzellator)} \\ &\Rightarrow (KK(S(KK)I)) && \text{(Kanzellator)} \\ &\Rightarrow K && \text{(Kanzellator)} \end{aligned}$$

$$\begin{aligned} &S(SI(K(II)))(S(S(KK)I))IS(KKI) \\ &\Rightarrow (SI(K(II)))I((S(S(KK)I))I)S(KKI) && \text{(Funktionsapplikation)} \\ &\Rightarrow II((K(II))I)((S(S(KK)I))I)S(KKI) && \text{(Funktionsapplikation)} \\ &\Rightarrow ((K(II))I)((S(S(KK)I))I)S(KKI) && \text{(Identität)} \\ &\Rightarrow (II)((S(S(KK)I))I)S(KKI) && \text{(Kanzellator)} \\ &\Rightarrow ((S(S(KK)I))I)S(KKI) && \text{(Identität)} \\ &\Rightarrow (S(KK)I)S(IS)(KKI) && \text{(Funktionsapplikation)} \\ &\Rightarrow ((KK)S(IS))(IS)(KKI) && \text{(Funktionsapplikation)} \\ &\Rightarrow ((K)(IS))(IS)(KKI) && \text{(Kanzellator)} \\ &\Rightarrow (IS)(KKI) && \text{(Kanzellator)} \\ &\Rightarrow (S)(KKI) && \text{(Identität)} \\ &\Rightarrow (KI(KI)) && \text{(Funktionsapplikation)} \\ &\Rightarrow I && \text{(Kanzellator)} \end{aligned}$$

3. **Aufgabe** (6 Punkte) $\lambda x.y(xy) \equiv S(K \ y)(SI(K \ y))$

$$\begin{aligned} &\lambda x.y(xy) \xrightarrow{6} (S \ T[\lambda x.y] \ T[\lambda x.(xy)]) \xrightarrow{4} (S \ (K \ T[y]) \ T[\lambda x.(xy)]) \xrightarrow{1} (S \ (K \ y) \ T[\lambda x.(xy)]) \\ &\xrightarrow{6} (S \ (K \ y) \ (S \ T[\lambda x.x] \ T[\lambda x.y])) \xrightarrow{3} (S \ (K \ y) \ (S \ I \ T[\lambda x.y])) \xrightarrow{4} (S \ (K \ y) \ (S \ I \ (K \ T[y]))) \\ &\xrightarrow{1} (S \ (K \ y) \ (S \ I \ (K \ y))) \\ &\Rightarrow S(K \ y)(SI(K \ y)) \end{aligned}$$

4. **Aufgabe** (6 Punkte) $\lambda s. \lambda x. s(s(x)) \equiv (S(S(KS)K)(S(S(KS)K)I))$

Seien a und b zwei beliebige Ausdrücke:

$\lambda s. \lambda x. s(s(x)) \ a \ b$

$\Leftrightarrow \lambda x. a(a(x)) \ b$

$\Leftrightarrow a(a(b))$

$(S(S(KS)K)(S(S(KS)K)I)) \ a \ b$

$\Rightarrow (S(KS)K)a((S(S(KS)K)I))a \ b$

(Funktionsapplikation))

$\Rightarrow ((KS)a(Ka))((S(S(KS)K)I))a \ b$

(Funktionsapplikation)

$\Rightarrow (S(Ka))((S(S(KS)K)I))a \ b$

(Kanzellator)

$\Rightarrow ((Ka)b((S(S(KS)K)I))a)b$

(Funktionsapplikation)

$\Rightarrow (a(((S(S(KS)K)I))a)b)$

(Kanzellator)

$\Rightarrow (a((S(KS)K)a)I(Ia)b)$

(Funktionsapplikation)

$\Rightarrow (a((KS)a(Ka)I(Ia)b)$

(Funktionsapplikation)

$\Rightarrow (a(S(Ka)I(Ia)b)$

(Kanzellator)

$\Rightarrow (a((Ka)(Ia)(I(Ia)))b)$

(Funktionsapplikation)

$\Rightarrow (a(a(I(Ia)))b)$

(Kanzellator)

$\Rightarrow (a(a(Ia))b)$

(Identität)

$\Rightarrow (a(a(a(b)))$

(Identität)

Die Ausdrücke sind äquivalent

5. **Aufgabe** (8 Punkte)

```
----- parser -----
{--
Everything I added or changed is to be found in this section.
Also: Please note, that miss Esponda created a show instance
that removes "Var" and "App" from our expressions so to test everything
you have to remove that completely
--}

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)
```

Abbildung 1: Code. Auch nochmal in der angehängten Datei zu finden