

Prof. Dr. Jürgen Giesl
Peter Schneider-Kamp

Exercises *Functional Programming* – Sheet 7

Solutions will be collected until Wednesday, June 15, 2005 in the exercise course.

Exercises can be solved both in English and in German.

Exercise 1 (1 + 1 + 2 + 3 points)

Consider the following Haskell functions:

```
fact  :: Int -> Int
fact  = \x -> if x <= 0 then 1 else fact (x-1) * x

two   :: Int -> Int
two   = \x -> 2

inf   :: Int -> Int
inf   = \x -> inf (x+2)

times :: (Int,Int) -> Int
times = \(x,y) -> if x <= 0 then 0 else y + times (x-1,y)
```

The higher-order Haskell function `f_fact` corresponding to `fact` is:

```
f_fact = \g -> \x -> if x <= 0 then 1 else g (x-1) * x
```

The semantics ϕ_{f_fact} of `f_fact` is:

$$(\phi_{f_fact}(g))(x) = \begin{cases} 1, & \text{if } x \leq 0 \\ g(x-1) \cdot x, & \text{otherwise} \end{cases}$$

The semantics ϕ_{fact} of `fact` is the least fixpoint of ϕ_{f_fact} (where for all $x < 0$ we define $x! = 1$):

$$\phi_{fact}(x) = \begin{cases} x!, & \text{if } x \in \mathbb{Z} \\ \perp & \text{otherwise} \end{cases}$$

- Give the Haskell definitions for the higher-order functions `f_two`, `f_inf`, and `f_times` corresponding to `two`, `inf`, and `times`.
- Give the semantics ϕ_{f_two} , ϕ_{f_inf} , and ϕ_{f_times} of the functions `f_two`, `f_inf`, and `f_times`.

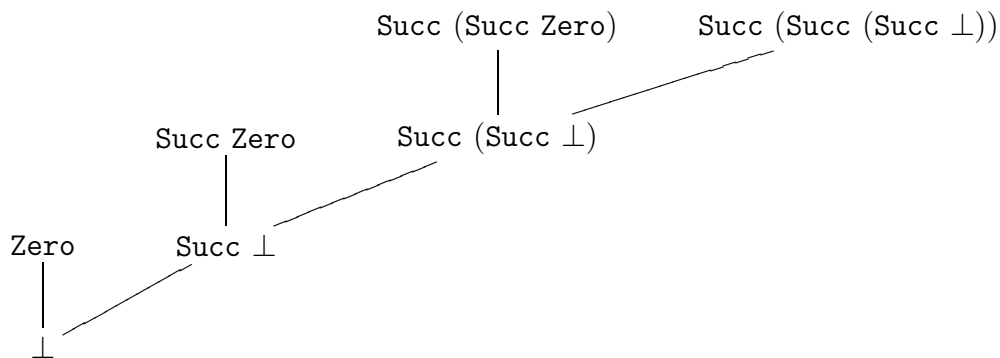
- (c) What does the function $\phi_f^n(\perp)$ compute for $n \in \mathbb{N}$, $f \in \{\mathbf{f_two}, \mathbf{f_inf}, \mathbf{f_times}\}$? Here, $\phi_f^n(\perp)$ denotes n applications of ϕ_f to the undefined function \perp .
- (d) Give all fixpoints of the semantic functions $\phi_{\mathbf{f_two}}$, $\phi_{\mathbf{f_inf}}$, and $\phi_{\mathbf{f_times}}$ from (a). Which ones are the least fixpoints? Label them $\phi_{\mathbf{two}}$, $\phi_{\mathbf{inf}}$, and $\phi_{\mathbf{times}}$ respectively.

Exercise 2 (2 + 2 + 2 + 1 points)

Consider the following data type declaration for natural numbers:

```
data Nats = Zero | Succ Nats
```

A graphical representation of the first four levels of the domain for `Nats` could look like this:



Now consider the following data type declarations

- (i) `data BoolPair = P (Bool, Bool)`
 (ii) `data NatsList = N | C Nats NatsList`

- (a) Give a graphical representation of the whole domain for the type `BoolPair` from (i).
- (b) Give a graphical representation of the first four levels of the domain for the type `NatsList` from (ii). The fourth level contains the element `C Zero N`, for example.
- (c) How many elements does the fifth level of the domain for the type `NatsList` from (ii) contain?
- (d) Give Haskell expressions that correspond to the following elements of the domain for the type `NatsList` from (ii), i.e., for each of these elements, give a Haskell expression that has this element as its semantics:

- `C ⊥ N`
- `C ⊥ (C ⊥ N)`
- `C Zero (C (Succ ⊥) ⊥)`