assignment05

May 28, 2019

1 Functional Programming SS19

2 Assignment 05 Solutions

2.0.1 Exercise 1 (Fixpoints)

Consider the function ee : $(\mathbb{Z}_\perp \to \mathbb{B}_\perp) \to (\mathbb{Z}_\perp \to \mathbb{B}_\perp)$, *which is defined as follows:*

$$(ee(g))(x) = \begin{cases} True & \text{if } x = 0 \\ \neg g(x-1) & \text{if } x > 0 \\ \neg g(x+1) & \text{if } x < 0 \\ \perp_{\mathbb{B}_\perp} & \text{if } x = \perp_{\mathbb{Z}_\perp} \end{cases}$$

The function ee is continuous. Thus, by Kleene's Fixpoint Theorem it has a least fixpoint. This least fixpoint of ee is a well-known function. What is the least fixpoint of ee? Prove your claim!

Hints: * *We have* $\neg\perp_{\mathbb{B}_\perp} = \perp_{\mathbb{B}_\perp}$. *Thus, the function* $\neg$ *is monotonic and also continuous.* * *If the cases* $x > 0$ *and* $x < 0$ *are analogous, then it sufficies to prove one case and note the analogy.*

The least fixpoint of *ee* is the function that returns *True* if x is an even number and *False* otherwise, namely the function $f : \mathbb{Z}_\perp \to \mathbb{B}_\perp$ such that

$$f(x) = \begin{cases} True & \text{if } x = 0 \\ \neg f(x-1) & \text{if } x > 0 \\ \neg f(x+1) & \text{if } x < 0 \\ \perp_{\mathbb{B}_\perp} & \text{if } x = \perp_{\mathbb{Z}_\perp} \end{cases}$$

To prove this, we can use Kleene's Fixpoint Theorem (Theorem 2.1.17 from the lecture notes), according to which the least fixpoint of a function f is the function $\operatorname{lfp} f = \sqcup\{f^i(\perp) \mid i \in \mathbb{N}\}$ provided that $\sqsubseteq$ is a cpo on D and $f : D \to D$ is continuous.

Let us show that $\sqcup\{ee^i(\perp) \mid i \in \mathbb{N}\} = f$ by induction. In the base case, we have $ee^0(\perp) = \perp$ and $ee^1(\perp) = f(\perp)$. Let's now assume that $ee^i(\perp) = f$ and show that $e^{i+1}(\perp) = f$ as well (which is equivalent to $ee(f) = f$). We have four cases to consider now depending on the value of x:

- $x = 0$: In this case, we have:

$$\begin{aligned} ee^{i+1}(\perp)(0) &= ee(ee^i(\perp))(0) \\ &= (ee(f))(0) \text{ -- due to the inductive hypothesis} \\ &= True \\ &= f(0) \end{aligned}$$

- $x > 0$: We now have

$$\begin{aligned} ee^{i+1}(\perp)(x) &= ee(ee^{i}(\perp))(x) \\ &= (ee(f))(x) \text{ – due to the inductive hypothesis} \\ &= \neg f(x-1) \end{aligned}$$

  where $f(x-1)$ is *True* if $x-1$ is even and *False* otherwise. Since odd and even numbers are alternating, we can easily see that $\neg f(x-1)$ gives the correct result for x.

- $x < 0$: The case for negative integers is analogous to the case for positive integers since negative odd and even numbers are alternating as well. Since $f(x+1)$ is *True* if $x+1$ is even and *False* otherwise, we can see that $(ee(f))(x)$ gives the correct result for negative x as well.

- $x = \perp$: $(ee(f))(\perp) = \perp = f(\perp)$

We can thus see that $ee^{i+1}(\perp)(x) = f(x)$, which means that f is indeed a fixpoint of ee. In addition, this is the only fixpoint of ee, so it is also its least fixpoint. □

2.1 Exercise 2 (Continuity and Fixpoints)

In this exercise, we will prove that for any real number $0 < r \leq 1$ we have

$$\lim_{n\to\infty} f^n(r) = \underbrace{f(f(...f(r)...))}_{n \text{ times}} = 1$$

where $f : [0, \infty) \to \mathbb{R}$, $x \mapsto \sqrt{x}$. In this exercise you can use that f is topological–continuous (cf. Exercise Sheet 4, Ex. 4), i.e., for any converging sequence $(x_n)_{n\in\mathbb{N}}$ in $[0, \infty)$ we have $\lim_{n\to\infty} f(x_n) = f(\lim_{n\to\infty} x_n)$. In each of the following parts you can use the results of Exercise Sheet 4, Ex. a)-c).

a) *Prove that f is monotonic w.r.t. $\leq$, i.e. if $0 \leq x \leq y$, then $f(x) \leq f(y)$.*

Hints: ** You can use that if $0 \leq x \leq y$ then $x^2 \leq y^2$ * You can also use that for every $0 \leq z$ we have $(\sqrt{z})^2 = z$ and $\left(\sqrt{z^2}\right) = z$.*

Let us take two arbitrary values x and y, such that $x, y \in [0, \infty), x \leq y$ and suppose that f is not monotonic, i.e. $f(x) > f(y)$.

We now have

$$\begin{aligned} 0 \leq x \leq y &\implies x^2 \leq y^2 \\ &\implies \sqrt{x^2} > \sqrt{y^2} \text{ (due to the non-monotonicity assumption)} \\ &\implies x > y \text{ (due to the fact that } \sqrt{x^2} = x \text{ and } \sqrt{y^2} = y) \end{aligned}$$

This however contradicts our initial assumption that $x \leq y$; thus, f must be monotonic. □

b) *Let $0 < r \leq 1$. Prove that $f_r : [r, 1] \to [r, 1]$, $x \mapsto \sqrt{x}$ is well-defined, i.e. if $r \leq x \leq 1$, then $r \leq \sqrt{x} \leq 1$*

Let us take an x such that $r \leq x \leq 1$:

$$\begin{aligned}
&\implies \sqrt{r} \leq \sqrt{x} \leq \sqrt{1} \text{ (due to monotonicity)} \\
&\implies \sqrt{r} \leq \sqrt{x} \leq 1
\end{aligned}$$

Let us now show that $r \leq \sqrt{r}$. In particular, let us assume that $\sqrt{r} < r$:

$$\begin{aligned}
&\implies 1 < \frac{r}{\sqrt{r}} \\
&\implies \sqrt{r} \leq \sqrt{x} \leq 1 < \frac{r}{\sqrt{r}} \\
&\implies \left(\sqrt{r}\right)^2 \leq \sqrt{x}\sqrt{r} \leq \sqrt{r} < r \text{ (multiplying through by} \sqrt{r}) \\
&\implies r \leq \sqrt{x}\sqrt{r} \leq \sqrt{r} < r
\end{aligned}$$

This implies that $r < r$, which clearly cannot hold. As a result, it must hold that $r \leq \sqrt{r}$ and consequently

$$\begin{aligned}
r \leq \sqrt{r} \leq \sqrt{x} \leq 1 \\
\implies r \leq \sqrt{x} \leq 1
\end{aligned}$$

which is what we wanted to show. □

c) *Let $0 < r \leq 1$. Use a), b) and Ex. 4 a)-c) from Exercise Sheet 4 to conclude that f_r is Scott-continuous for any $0 < r \leq 1$. You can use that if $(x_n)_{n\in\mathbb{N}}$ is a converging sequence in $[r, 1]$, then $\lim\limits_{n\to\infty} x_n \in [r, 1]$, too.*

From part b), we know that f_r is a monotonic function and, furthermore, since f is topologically-continuous, f_r is topologically-continuous as well. From Ex. 4c) in Exercise Sheet 4, we know that every monotonic and topologically-continuous function is also Scott-continuous; thus, it follows that f_r is Scott-continuous.

We can also prove this directly by taking an arbitrary subset $[i, j] \subseteq [r, 1]$ and creating a sequence $S = (x_n)_{n\in\mathbb{N}}$, $x_n \in [i, j]$. S is bounded above by r, so we have that

$$\lim_{n\to\infty} x_n = \sup S = j \implies f_r(\sup S) = f_r(j)$$

Let us now also construct a sequence $S' = f_r(x_n)_{n\in\mathbb{N}}$ from S. Due to the monotonicity of f_r, we know that $f(x_n) \leq f(x_{n+1})$ for every $x_n \in S$; thus

$$\lim_{n\to\infty} f(x_n) = \sup S' = f_r(j)$$

where the last equality follows from the fact that f_r is continuous. Thus, $f_r(\sqcup S) = \sqcup S'$ for any S, which means that f_r is Scott-continuous. □

d) *Use Kleene's Fixpoint theorem and Ex. 4 a) from Exercise Sheet 4 to conclude that for each $0 < r \leq 1$, $\lim\limits_{n\to\infty} f^n(r) = 1$.*

From part c), we know that f_r is Scott-continuous on $[r, 1]$, and thus f is Scott-continuous on $[r, 1]$. From Ex. 4a in Exercise Sheet 4, we also know that $\leq$ is complete for any closed interval $[a, b]$; thus, $\leq$ is complete on $[r, 1]$.

The conditions of Kleene's Fixpoint theorem are fulfilled and, as a result, $\text{lfp}\, f = \sqcup\{f^i(\perp) \mid i \in \mathbb{N}\}$. In our case, r is the smallest element, so we have that $\text{lfp}\, f = \sqcup\{f^i(r) \mid i \in \mathbb{N}\}$.

But since f_r maps values from $[r, 1]$ to $[r, 1]$ and due to the completeness of $[r, 1]$, we have that for the chain $S = \{f^i(r) \mid i \in \mathbb{N}\}$, $\sqcup S = 1$.

Since f_r is monotonic, we also have that $f_r^i, i \in \mathbb{N}$ is monotonic (short proof: based on the definition of monotonicity, we know that if $x \leq y$, then $f_r(x) \leq f_r(y)$, but then it also follows that $f_r(f_r(x)) \leq f_r(f_r(y))$).

But based on part c), due to the monotonicity, we have $\sup S = \lim_{n\to\infty} f^n(r) = 1$, which is what we wanted to show. $\square$

2.2 Exercise 3 (Fixpoints and Higher Order Functions)

Consider the following Haskell functions:

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

true :: Bool -> Bool
true = \x -> True

neg_inf :: Int -> Int
neg_inf = \x -> neg_inf (x-3)

fib :: Int -> Int
fib = \n -> if n <= 1 then 1 else fib (n - 1) + fib (n - 2)
```

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:

$$(\varphi_{f\_fact}(g))(x) = \begin{cases} 1 & \text{if } x \leq 0 \\ g(x-1) \cdot x & \text{otherwise} \end{cases}$$

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

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

a) *Give the Haskell definitions for the higher-order functions `f_true`, `f_neg_inf`, and `f_fib` corresponding to `true`, `neg_inf`, and `fib`.*

The definitions of the functions are given below:

```
f_true = \g -> \x -> True

f_neg_inf = \g -> \x -> g (x - 3)

f_fib = \g -> \n -> if n <= 1 then 1 else g(n - 1) + g(n - 2)
```

b) *Give the semantics ϕ_{f_true}, $\phi_{f\_neg\_inf}$, and ϕ_{f_fib} of the functions `f_true`, `f_neg_inf`, and `f_fib`.*

The functions have the following semantics:

$$(\phi_{f\_true}(g))(x) = True$$

$$(\phi_{f\_neg\_inf}(g))(x) = \begin{cases} g(x-3) & \text{if } x \in \mathbb{Z} \\ \bot & \text{otherwise} \end{cases}$$

$$(\phi_{f\_fib}(g))(n) = \begin{cases} 1 & \text{if } x \leq 1 \\ g(n-1) + g(n-2) & \text{if } x > 1 \\ \bot & \text{otherwise} \end{cases}$$

c) *What does the function $\phi^n_f(\bot)$ compute for $n \in \mathbb{N}$, $f \in \{$ `f_true`, `f_neg_inf`, `f_fib` $\}$? Here, $\phi^n_f(\bot)$ denotes n applications of ϕ_f to the undefined function $\bot$.*

Since $(\phi_{f\_true}(g))(x)$ returns *True* for every argument, the result of $\phi^n_{f\_true}(\bot)$ is *True* for any $n \in \mathbb{N}$.

For $\phi^n_{f\_neg\_inf}(\bot)$, we have

$$\begin{aligned} \phi^0_{f\_neg\_inf}(\bot)(x) &= \bot \\ \phi^1_{f\_neg\_inf}(\bot)(x) &= \begin{cases} \bot(x-3) & \text{if } x \in \mathbb{Z} \\ \bot & \text{otherwise} \end{cases} = \begin{cases} \bot & \text{if } x \in \mathbb{Z} \\ \bot & \text{otherwise} \end{cases} = \bot \end{aligned}$$

namely $\phi^n_{f\_neg\_inf}(\bot) = \bot$ for any $n \in \mathbb{N}$.

For $\phi^n_{f\_fib}(\bot)$, we have

$$\begin{aligned} g_0(n) &= \phi^0_{f\_fib}(\bot)(x) = \bot \\ g_1(n) &= \phi^1_{f\_fib}(\bot)(x) = \begin{cases} 1 & \text{if } x \leq 1 \\ \bot(n-1) + \bot(n-2) & \text{if } x > 1 \\ \bot & \text{otherwise} \end{cases} = \begin{cases} 1 & \text{if } x \leq 1 \\ \bot & \text{if } x > 1 \\ \bot & \text{otherwise} \end{cases} \\ g_2(n) &= \phi^2_{f\_fib}(g_1) = \begin{cases} 1 & \text{if } x \leq 1 \\ g_1(n-1) + g_1(n-2) & \text{if } x > 1 \\ \bot & \text{otherwise} \end{cases} = \begin{cases} 1 & \text{if } x \leq 1 \\ 2 & \text{if } x = 2 \\ \bot & \text{if } x > 2 \\ \bot & \text{otherwise} \end{cases} \\ g_3(n) &= \phi^3_{f\_fib}(g_2) = \begin{cases} 1 & \text{if } x \leq 1 \\ g_2(n-1) + g_2(n-2) & \text{if } x > 1 \\ \bot & \text{otherwise} \end{cases} = \begin{cases} 1 & \text{if } x \leq 1 \\ 2 & \text{if } x = 2 \\ 3 & \text{if } x = 3 \\ \bot & \text{if } x > 3 \\ \bot & \text{otherwise} \end{cases} \end{aligned}$$

namely $\phi^n_{f\_fib}(\perp)$ produces a sequence of increasingly better approximations to the Fibonacci sequence.

d) *Give all fixpoints of the semantic functions ϕ_{f_true}, $\phi_{f\_neg\_inf}$, and ϕ_{f_fib} from b). Which ones are the least fixpoints?*

ϕ_{f_true} has a single fixpoint, namely the constant function that always returns *True*; this is thus also the least fixpoint of ϕ_{f_true}.

$\phi_{f\_neg\_inf}$ has as fixpoints all constant functions $f : \mathbb{Z}_\perp \to \mathbb{Z}_\perp$; the least fixpoint is the constant function $\perp_{\mathbb{Z}\to\mathbb{Z}}$.

ϕ_{f_fib} has a single fixpoint, namely the function that computes the Fibonacci sequence for any $n \in \mathbb{N}$ and returns $\perp$ otherwise; this function is therefore the least fixpoint of ϕ_{f_fib}.

2.3 Exercise 4 (Domain Lifts)

Let $D_1, ..., D_n$ be domains with complete partial orders $\sqsubseteq_{D_1}, ..., \sqsubseteq_{D_n}$.

a) *Prove that $\sqsubseteq_{D_1 \oplus ... \oplus D_n}$ is a complete partial order on $D_1 \oplus ... \oplus D_n$.*

According to definition 2.1.12, $\sqsubseteq_{D_1 \oplus ... \oplus D_n}$ is complete if $D = D_1 \oplus ... \oplus D_n$ has a smallest element $\perp_{D_1 \oplus ... \oplus D_n}$ and there is a least upper bound $\sqcup S \in D_1 \oplus ... \oplus D_n$ for every chain in $D_1 \oplus ... \oplus D_n$. The existence of a $\perp_{D_1 \oplus ... \oplus D_n}$ is guaranteed by the definition of a coalesced sum (definition 2.2.2). Furthermore, as a consequence of definition 2.2.2, a chain $S \in D$ can contain $\perp_D$ and elements of only one domain D_i, $1 \le i \le n$; hence, provided that every D_i is a complete partial order, D will also be a complete partial order. $\Box$

b) *Prove that for any $1 \le k \le n$ the embedding*

$$l_k : D_k \to D_1 \bigoplus ... \bigoplus D_n,\ x \mapsto \begin{cases} \perp_{D_1 \oplus ... \oplus D_n}, & x = \perp_{D_k} \\ x^{D_k}, & \text{otherwise} \end{cases}$$

is continuous.

Based on definition 2.1.14, $l_k : D_k \to D_1 \oplus ... \oplus D_n$ is continuous if and only if, for every chain S of D_k, we have $l_k(\sqcup S) = \sqcup\{l_k(d) \mid d \in S\}$, provided that D_k and $D_1 \oplus ... \oplus D_n$ is complete, such that the completeness of $D_1 \oplus ... \oplus D_n$ was established in part a).

Since D_k is complete, we know that it has a smallest element $\perp_{D_k}$ and every chain S has a least upper bound $\sqcup S \in D_k$ (definition 2.1.12). The given l_k acts as an identity operator based on which each element $x \in D_k$ is mapped to itself, except that the result lies in the domain of the coalesced sum instead of in D_k; in particular, for every chain $S = d_{k,1}, d_{k,2}...$, we have

$$l_k(S) = \{l_k(d_{k,1}), l_k(d_{k,2}), ...\}_{D_k} = \{d_{k,1}, d_{k_2}, ...\}_{D_1 \oplus ... \oplus D_n}$$

Since the identity operator is continuous, it follows that l_k is also continuous for every $1 \le k \le n$. $\Box$

c) *Let $f : D_1 \oplus ... \oplus D_n \to D$ be a monotonic function, where D is some domain with cpo $\sqsubseteq_D$. Prove that f is continuous if $f \circ l_k : D_k \to D$ is continuous for every $1 \le k \le n$.*

For any chain $S \in D_k$, we have

$$\begin{aligned} l_k(S) &= \{l_k(d_{k,1}), l_k(d_{k,2}), ...\}_{D_1 \oplus ... \oplus D_n} \\ &\implies (f(l_k))(S) = \{f(l_k(d_{k,1})), f(l_k(d_{k,2})), ...\}_D \end{aligned}$$

Since $f \circ l_k : D_k \to D = f(l_k)$ is continuous for every $1 \leq k \leq n$, we have $(f(l_k))(\sqcup S) = \sqcup\{(f(l_k))(d) \mid d \in S\}$ for every chain S in D_k. But since $l_k(d) = d^{D_k}$, we can define a chain $S' = \{d_{k,1}, d_{k,2}, ...\}_{D_1 \oplus ... \oplus D_n}$ and

$$f(S') = \{f(d_{k,1}), f(d_{k,2}), ...\}$$

As f is monotonic, we have that $f(d_{k,1}) \sqsubseteq f(d_{k,2}) \sqsubseteq ....$ But l_k is continuous (and also monotonic since it does not change the definedness of the arguments) and we have that $\sqcup S_{D_k} = \sqcup S'_{D_1 \oplus ... \oplus D_n}$, so

$$f(\sqcup S) = \sqcup\{(f(l_k))(d) \mid d \in S\} = \sqcup\{f(d) \mid d \in S'\} = f(\sqcup S')$$

which is what we wanted to show. $\square$

2.4 Exercise 5 (Domain Construction)

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:

```
data Unit = U ()
data Foo = A Unit Unit | B Bool
```

a) *Give a graphical representation of the whole domain for the type `Foo`. The graphical representation must be a directed graph, as in the example, and must contain all elements of `Foo`.*

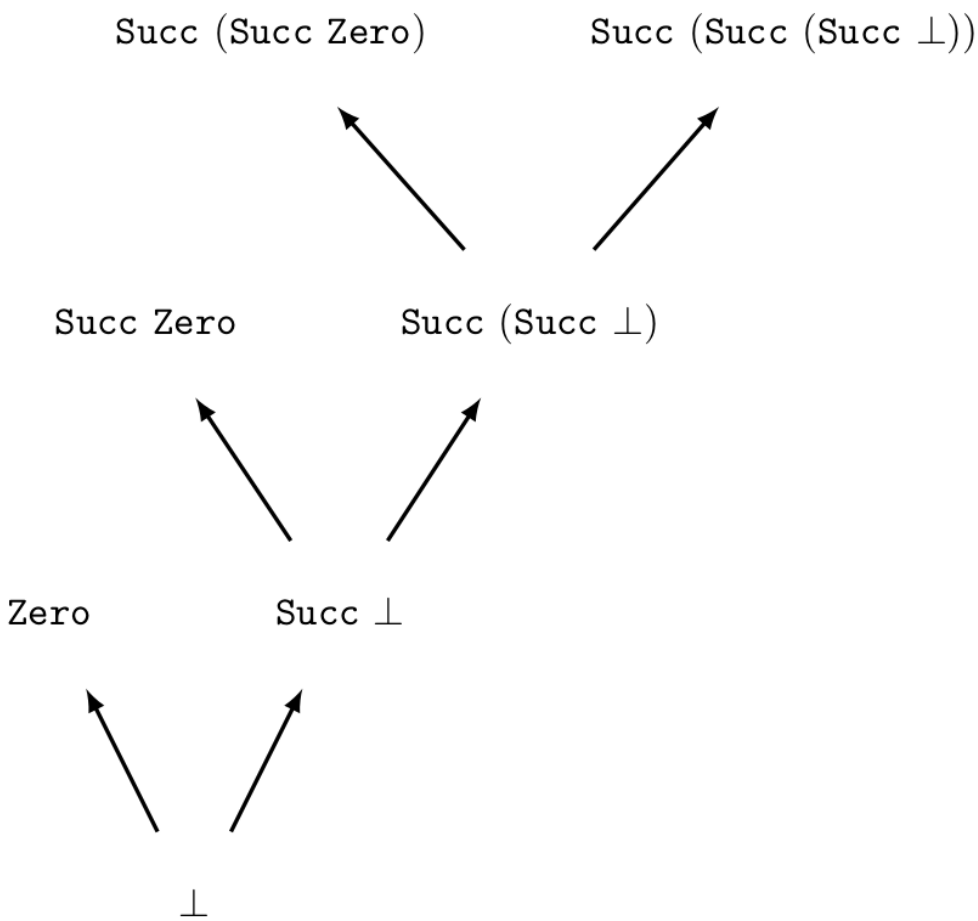
The domain of the type `Foo` is depicted below:

b) *Give Haskell expressions that correspond to the following elements of the domains for the type `Foo` and the type `Unit` respectively, i.e., for each of these elements, give a Haskell expression that has this element as its semantics:*

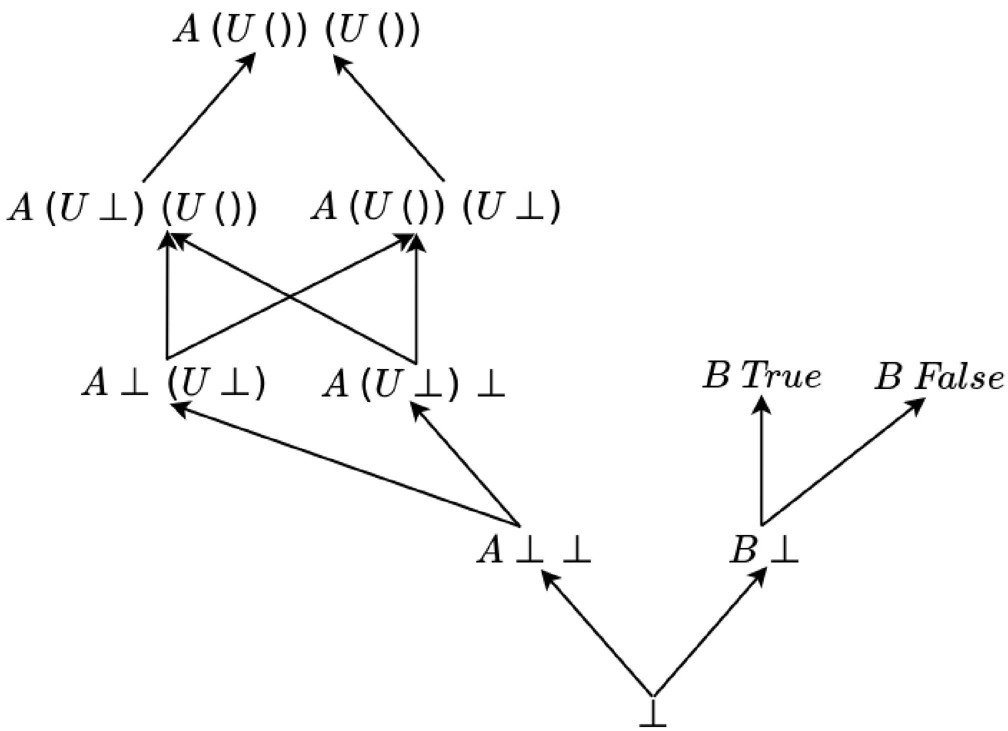
- $A \perp \perp$
- $U \perp$

Let us define a bottom element as `bot = bot` as in the lecture. Then, the following two expressions have the semantics of the above two elements of the domains of `Foo` and `Unit` respectively:

- `a = A bot bot` has the semantics $A \perp \perp$
- `b = U bot` has the semantics $U \perp$



nats_domain



foo_domain

c) *Consider the following Haskell functions:*

```
a :: (Integer, Integer) -> Integer
a (k, n) = if k <= 0 then n+1 else if n <= 0 then a(k-1,1) else a(k-1, a(k, n-1))

u :: Bool -> Unit
u b = if (not b) then U () else u b

f :: Bool -> Bool
f b = f b
```

What is the semantics of the following Haskell expressions?

- `B (f False)`

Let $exp = B(f\ False) = B\ exp'$. We then have $Val[\![exp]\!]_\rho = Val[\![B\ exp']\!]_\rho = g$ *in* Functions *in* Dom, where $g(exp') = (B, exp')$ *in* Constructions$_1$ *in* Dom.

Since exp' is an expression, we need to find the semantics d of exp' to fully determine the semantics of exp. In this case, we have $d = Val[\![exp']\!]_\rho = Val[\![f\ False]\!]_\rho = h(Val[\![False]\!]_\rho)$ for h *in* Functions *in* Dom, where $h(x) = \bot, x \in \mathbb{B}_\bot$.

As *False* is a constructor of zero arity, we know that $Val[\![False]\!]_\rho = False$ *in* Constructions$_0$ *in* Dom.

The semantics of *exp* is thus finally $Val[\![exp]\!]_\rho = g(h(False)) = (B, h(False)) = (B, \bot)$.

- `A (u (f False)) (u False)`

As above, let $exp = A\ (u\ (f\ False))\ (u\ False) = A\ exp'\ exp''$. Thus, $Val[\![exp]\!]_\rho = g\ in$ Functions *in* Dom such that $g(exp', exp'') = (A, exp', exp'')$ *in* Constructions$_2$ *in* Dom.

We can now find the semantics of exp' and exp'' as follows:

$Val[\![exp']\!]_\rho = Val[\![u\ (f\ False)]\!]_\rho = l(Val[\![f\ False]\!]_\rho)$ for l *in* Functions *in* Dom (where $d = Val[\![f\ False]\!]_\rho = \perp$ as in part a)), such that

$$l(x) = \begin{cases} U\ () & \text{if } x\ =\ False \\ \perp & \text{otherwise} \end{cases}$$

Using the semantics d from part a), we have that $Val[\![exp']\!]_\rho = l(\perp) = \perp$.

On the other hand, $Val[\![exp'']\!]_\rho = \mathcal{V}\dashv\updownarrow[\![u\ False]\!]_\rho = l(Val[\![False]\!]_\rho) = l(False) = U\ ()$, where l is as defined above.

The semantics of exp is thus $Val[\![exp]\!]_\rho = g(\perp, U\ ()) = (A, \perp, U\ ())$.

- `let g = \n -> a (3, n) < 3 in A (U ()) (u (g 2))`

In this case, we have

$$\begin{aligned} & Val[\![let\ g\ =\ \backslash n -> \ a\ (3,\ n)\ <\ 3\ in\ A\ (U\ ())\ (u\ (g\ 2))]\!]_\rho \\ & = Val[\![A\ (U\ ())\ (u\ (g\ 2))]\!]_\rho\ (\rho + \{g/\operatorname{lfp} f\}) \\ & = (A, exp', exp'')\ in \text{ Constructions}_1\ in \text{ Dom} \end{aligned}$$

where

$$\begin{aligned} f(d) &= Val[\![\backslash n -> a\ (3,\ n)\ <\ 3]\!]\ (\rho + \{g/d\}) = Val[\![\backslash n -> a\ (3,\ n)\ <\ 3]\!]_\rho \\ Val[\![exp']\!] &= Val[\![U\ ()]\!] = (U, ())\ in \text{ Constructions}_1\ in \text{ Dom} \\ Val[\![exp'']\!] &= Val[\![u\ (g\ 2)]\!] = l(Val[\![(g\ 2)]\!])\ in \text{ Functions}_1\ in \text{ Dom} \end{aligned}$$

We can subsequently find that $a(3,2) = 29$; thus $Val[\![g\ 2]\!] = False$ and $Val[\![u\ (g\ 2)]\!] = U\ ()$, so

$$Val[\![A\ (U\ ())\ (u\ (g\ 2))]\!]_\rho\ (\rho + \{g/\operatorname{lfp} f\}) = (A, U\ (), U\ ())$$